



Norwegian University of  
Science and Technology

# The Amazing City Game

Sondre Wigmostad Bjerkaug  
Runar Os Mathisen  
Lawrence Alexander Valtola

Master of Science in Computer Science  
Submission date: June 2011  
Supervisor: Alf Inge Wang, IDI  
Co-supervisor: Bian Wu, IDI

Norwegian University of Science and Technology  
Department of Computer and Information Science



## Assignment Text

In this project, the students should design and develop a prototype application for the mobile platform Android where users can run knowledge competitions within the context of the tech-city Trondheim. The prototype should allow players to compete in puzzle races that support different types of tasks, such as solving riddles, finding specific locations, and other interesting tasks related to the city life. The main inspiration for the game is the American television show "The Amazing Race". The prototype should take advantage of the technical capabilities of the mobile phones running Android, such as the camera, wireless Internet and GPS.

The students should use this prototype to run a race where users are organized in groups and compete with each other in the city using Android mobile phones. Finally, the students should evaluate the effectiveness of the prototype, and propose a platform for lifelong learning where users can both create and run their own knowledge competitions.



# Abstract

Smartphones with capabilities for wireless Internet and GPS have become increasingly common in recent years, and a consequence of this is that pervasive games have become more interesting from both an academical and a commercial point of view. Another area of interest is lifelong learning, which offers a more modern take on education compared to the traditional learning model. In this thesis we aim to discover whether or not pervasive gaming can help achieve lifelong learning. This is done by creating a prototype of a pervasive game in a lifelong learning context for Android, analyzing the effectiveness of the prototype, and using the experiences drawn from it to design a platform to run knowledge competitions.

We achieved this by conducting a prestudy on the Android mobile phone operating system (including extension applications), the history of Trondheim, lifelong learning, pervasive games, and the use of pervasive games in a lifelong learning context. During the prestudy we found out that there are several external applications and features of Android that can be utilized to expand the social, spatial, and temporal expansions of pervasive games. We also found that, in theory, pervasive games proved to be a suitable platform to support lifelong learning.

We then designed and developed a prototype on Android to run a puzzle race called "The Amazing City Game". The race consisted of completing different tasks related to the history of Trondheim, while traveling between many of the historical sites in the city. A demonstration race was conducted in early May with four groups of two students each, using the authors and supervisors of this thesis as group observers. At the end of the race, the participants filled out a survey. Using the observations from the race and the results from the survey we found that the prototype was perceived as fun and educational. However, construction of the race was challenging with many pitfalls concerning ambiguous tasks, use of language, and game balance.

Finally, we have provided a possible design for a platform for running knowledge competitions. We used the experiences from the development of the prototype, and the results from the demonstration race to design a cleaner and more complete framework. This includes a refinement of the existing functionality and user interface, adding requirements, and providing an extended discussion on topics such as having an online community, possible server solutions, and security measures against cheating. We believe that the concept of puzzle races in a lifelong learning context is an interesting concept that could have positive effects if utilized in the real world.



# Preface

This document is the master thesis of Runar Os Mathisen, Lawrence Alexander Valtola and Sondre Wigmostad Bjerkhaug, and concludes our master's degree in computer science. The project group is composed of three students from the Norwegian University of Science and Technology, Department of Computer and Information Science. In this thesis we create a prototype puzzle race application for the mobile phone operating system Android in the context of achieving life-long learning. We use the experiences from this prototype to suggest a platform for pervasive games in the context of lifelong learning.

We would like to thank Lene Kristin Solvik for her advice on the pedagogical aspect of the prototype, Rita Løkhaug for helping out with the demonstration race, Trygve Bragstad for providing valuable information about Trondheim, the students who participated in the demonstration race, and last but not least, supervisor Alf Inge Wang and co-advisor Bian Wu for their guidance on the thesis.

Trondheim, June 15, 2011.

---

Runar Os Mathisen

---

Lawrence A. Valtola

---

Sondre W. Bjerkhaug





---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Project Goal . . . . .	2
1.3	Research Questions . . . . .	2
1.3.1	Lifelong Learning and Pervasive Games . . . . .	2
1.3.2	Applying Pervasive Games Under the Context of Lifelong Learning . . . . .	2
1.3.3	Pervasive Game Platform for Lifelong Learning . . . . .	3
1.4	Research Method . . . . .	3
1.5	Thesis Structure . . . . .	4
<b>I</b>	<b>Project Plan</b>	<b>7</b>
<b>2</b>	<b>Introduction</b>	<b>9</b>
<b>3</b>	<b>Project Context</b>	<b>11</b>
3.1	Milestones . . . . .	11
3.1.1	Prestudy . . . . .	11
3.1.2	Design . . . . .	12
3.1.3	Implementation . . . . .	12
3.1.4	Content and Demonstration Preparation . . . . .	12
3.1.5	Demonstration . . . . .	12
3.1.6	Evaluation . . . . .	13
3.1.7	Platform Design . . . . .	13
3.1.8	Finalization . . . . .	13
3.2	Development Strategy . . . . .	13
3.3	Risk Analysis . . . . .	13
3.3.1	General Risks . . . . .	13
3.3.2	Demonstration Specific Risks . . . . .	14
3.4	Templates and Standards . . . . .	16
3.5	Tools . . . . .	16
3.6	Version Control Procedures . . . . .	19

<b>4</b>	<b>Organization</b>	<b>21</b>
4.1	Group Structure . . . . .	21
4.2	Group Work Hours . . . . .	21
4.3	Meetings . . . . .	22
4.3.1	Supervisor Meetings . . . . .	22
4.3.2	Internal Meetings . . . . .	22
<b>II</b>	<b>Prestudy</b>	<b>23</b>
<b>5</b>	<b>Introduction</b>	<b>25</b>
<b>6</b>	<b>Research Context</b>	<b>27</b>
6.1	Lifelong Learning and Pervasive Games . . . . .	27
6.1.1	Lifelong Learning . . . . .	27
6.1.2	Pervasive Games . . . . .	28
6.1.3	The Relationship between Lifelong Learning and Pervasive Games . . . . .	33
6.2	Applying Pervasive Games Under the Context of Lifelong Learning	35
6.2.1	How Can We Apply Pervasive Games Under the Context of Lifelong Learning? . . . . .	35
6.2.2	How Can We Use an Experiment to Describe the Idea? . . . . .	37
<b>7</b>	<b>Android</b>	<b>39</b>
7.1	About Android . . . . .	39
7.2	Android Market Share . . . . .	39
7.3	Android Mobile Phone Features . . . . .	40
7.3.1	GPS . . . . .	40
7.3.2	Wi-Fi . . . . .	42
7.3.3	Camera . . . . .	42
7.3.4	Near Field Communication . . . . .	42
7.3.5	Gyroscope . . . . .	43
7.3.6	Accelerometer . . . . .	43
7.3.7	Ambient Light Sensor . . . . .	44
7.3.8	Proximity Sensor . . . . .	44
7.3.9	Digital Compass . . . . .	44
<b>8</b>	<b>Applications from the Android Market</b>	<b>45</b>
8.1	The Quick Response Code . . . . .	45
8.2	Google Goggles . . . . .	46
8.3	Smart Tools . . . . .	47
8.4	Layar . . . . .	47
8.5	ShopSavvy . . . . .	49

8.6	Shazam . . . . .	49
8.7	Flashlight . . . . .	49
8.8	Compass . . . . .	50
<b>9</b>	<b>Related Work</b>	<b>53</b>
9.1	The Amazing Race . . . . .	53
9.2	Shelby Logan's Run . . . . .	54
9.3	Heroes of Koskenniska . . . . .	54
9.4	Game-Based Learning in Universities and Lifelong Learning . . . . .	54
9.5	To Construct the Outdoor Experience Game-Based Learning System by Integrating Ubiquitous Technology . . . . .	55
<b>10</b>	<b>Trondheim</b>	<b>57</b>
10.1	Torvet . . . . .	57
10.2	Munkegata . . . . .	57
10.3	Vår Frue Kirke . . . . .	58
10.4	Nidarosdomen . . . . .	58
10.5	Erkebispegården . . . . .	59
10.6	Stiftsgården . . . . .	59
10.7	Bakklandet . . . . .	59
10.8	Munkholmen . . . . .	60
10.9	Ravnkloa . . . . .	61
10.10	Kristiansten Festning . . . . .	61
10.11	Gamle Bybro . . . . .	61
10.12	Britannia Hotel . . . . .	62
<b>III</b>	<b>Design</b>	<b>63</b>
<b>11</b>	<b>Introduction</b>	<b>65</b>
<b>12</b>	<b>Experiment</b>	<b>67</b>
12.1	Motivation . . . . .	67
12.2	Inspiration . . . . .	67
12.3	Game Plot . . . . .	68
12.4	Ethical Concerns . . . . .	68
<b>13</b>	<b>Tasks</b>	<b>69</b>
13.1	Location Task . . . . .	69
13.2	Barcode Task . . . . .	70
13.3	Open Task . . . . .	70
13.4	Multiple Choice Task . . . . .	71
13.5	Checkbox Task . . . . .	71

13.6	Shazam Challenge . . . . .	71
13.7	Shopsavvy Challenge . . . . .	72
<b>14</b>	<b>Requirements</b>	<b>75</b>
14.1	Functional Requirements . . . . .	75
14.1.1	Task . . . . .	75
14.1.2	Hint . . . . .	76
14.1.3	Answer Validation . . . . .	76
14.1.4	GPS . . . . .	77
14.1.5	Barcode Scanning . . . . .	77
14.1.6	Game State . . . . .	77
14.1.7	Misc . . . . .	78
14.2	Non-functional Requirements . . . . .	78
<b>15</b>	<b>User Interface</b>	<b>81</b>
<b>16</b>	<b>Software Architecture</b>	<b>83</b>
16.1	Architectural Choices . . . . .	83
16.1.1	Server Solution . . . . .	83
16.1.2	Storage Solution . . . . .	84
16.1.3	Activity Structure . . . . .	84
16.2	Data Integration Architecture Diagram . . . . .	84
16.3	State Diagram . . . . .	85
16.4	Data Flow Diagram . . . . .	85
16.5	Package Diagram . . . . .	88
16.6	Class Diagram . . . . .	89
16.6.1	Root Package . . . . .	89
16.6.2	DB Package . . . . .	89
16.6.3	Content Package . . . . .	89
16.6.4	UI Package . . . . .	90
16.7	Entity-Relationship Diagram . . . . .	92
<b>IV</b>	<b>Implementation</b>	<b>93</b>
<b>17</b>	<b>Introduction</b>	<b>95</b>
<b>18</b>	<b>Sprint 1</b>	<b>97</b>
18.1	Requirements . . . . .	97
18.2	Design . . . . .	98
18.2.1	Game Start . . . . .	98
18.2.2	Task Flow . . . . .	98
18.2.3	Task User Interface . . . . .	98

18.3	Implementation . . . . .	99
18.3.1	Game Start . . . . .	99
18.3.2	Task Flow . . . . .	99
18.3.3	Task User Interface and Answer Validation . . . . .	100
18.3.4	DbAdapter and DbHelper . . . . .	103
18.4	Testing . . . . .	104
18.4.1	Task Flow . . . . .	104
18.4.2	Game State . . . . .	105
18.4.3	User Interface . . . . .	105
18.4.4	Answer Validation . . . . .	105
18.5	Evaluation . . . . .	106
<b>19</b>	<b>Sprint 2</b>	<b>107</b>
19.1	Requirements . . . . .	107
19.2	Design . . . . .	108
19.2.1	Images . . . . .	108
19.2.2	GPS . . . . .	108
19.2.3	Shazam . . . . .	108
19.2.4	Hints . . . . .	109
19.3	Implementation . . . . .	109
19.3.1	Images . . . . .	109
19.3.2	GPS . . . . .	109
19.3.3	Shazam . . . . .	113
19.3.4	Hints . . . . .	114
19.3.5	Revised Interface and Database . . . . .	115
19.4	Testing . . . . .	117
19.4.1	Images . . . . .	117
19.4.2	GPS . . . . .	117
19.4.3	Shazam . . . . .	118
19.4.4	Hints . . . . .	118
19.5	Evaluation . . . . .	119
<b>20</b>	<b>Sprint 3</b>	<b>121</b>
20.1	Requirements . . . . .	121
20.2	Design . . . . .	122
20.2.1	Checkbox . . . . .	122
20.2.2	Barcode Scanning . . . . .	122
20.2.3	Finishing Screen . . . . .	122
20.2.4	Reset Database . . . . .	122
20.2.5	Task Failsafe . . . . .	122
20.3	Implementation . . . . .	123
20.3.1	Checkbox . . . . .	123

20.3.2	Barcode Scanning . . . . .	123
20.3.3	Finishing Screen . . . . .	124
20.3.4	Reset Database . . . . .	124
20.3.5	Task Failsafe . . . . .	125
20.4	Testing . . . . .	126
20.4.1	Checkbox . . . . .	126
20.4.2	Barcode Scanning . . . . .	126
20.4.3	Finishing Screen . . . . .	126
20.4.4	Reset Database . . . . .	127
20.4.5	Task Failsafe . . . . .	127
20.5	Evaluation . . . . .	127
<b>21</b>	<b>Prototype Finalization</b>	<b>129</b>
21.1	GPS . . . . .	129
21.2	Task Failsafe Codes . . . . .	130
21.3	Interface Changes . . . . .	131
<b>V</b>	<b>Results</b>	<b>133</b>
<b>22</b>	<b>Introduction</b>	<b>135</b>
<b>23</b>	<b>Demonstration</b>	<b>137</b>
23.1	General Information . . . . .	137
23.2	Execution . . . . .	138
<b>24</b>	<b>Survey</b>	<b>141</b>
24.1	About Surveys . . . . .	141
24.1.1	System Usability Scale . . . . .	141
24.1.2	EGameFlow . . . . .	142
24.2	The Survey . . . . .	143
24.3	Survey Results . . . . .	143
<b>25</b>	<b>Evaluation</b>	<b>147</b>
25.1	Demonstration . . . . .	147
25.2	Survey . . . . .	148
25.2.1	Respondents . . . . .	148
25.2.2	System Usability Scale . . . . .	148
25.2.3	EGameFlow . . . . .	149
25.3	Prototype . . . . .	152
25.3.1	Survey Comments . . . . .	152
25.3.2	Observations . . . . .	153
25.4	Risk Analysis . . . . .	154

25.4.1	GPS Inaccuracy . . . . .	154
25.4.2	Foreigners Struggling With the Language . . . . .	155
25.4.3	Sickness . . . . .	155
25.4.4	Wi-Fi Coverage . . . . .	155
25.4.5	Battery . . . . .	155
25.4.6	Android Difficulty . . . . .	155
25.4.7	Task Unclear/Ambiguous . . . . .	156
25.4.8	Participants Unfamiliar With Android . . . . .	156
25.4.9	Prototype Error . . . . .	156
25.5	Conclusion . . . . .	156
<b>VI Platform</b>		<b>159</b>
<b>26</b>	<b>Introduction</b>	<b>161</b>
26.1	Pervasive Game Platform for Lifelong Learning . . . . .	162
26.1.1	How can We Extend This Experiment to be a Platform to Motivate and Support Lifelong Learning? . . . . .	162
<b>27</b>	<b>Discussion</b>	<b>165</b>
27.1	Possible Server Solutions . . . . .	165
27.1.1	Multitier Architecture . . . . .	165
27.1.2	Client-Server Architecture . . . . .	166
27.1.3	Distributed Architecture . . . . .	166
27.1.4	Peer-to-Peer . . . . .	166
27.2	Financing . . . . .	167
27.3	Security . . . . .	167
27.3.1	Encryption of Race Files . . . . .	168
27.3.2	Encryption of Single Tasks . . . . .	168
27.3.3	Save Race Information in the Application's Database . . . . .	168
27.3.4	Game Modes . . . . .	169
27.4	Online Community . . . . .	169
27.4.1	Web Forum . . . . .	169
27.4.2	Puzzle Race Guide . . . . .	170
<b>28</b>	<b>Requirements</b>	<b>173</b>
28.1	Functional Requirements . . . . .	173
28.1.1	General . . . . .	173
28.1.2	Online Community . . . . .	174
28.1.3	Editor . . . . .	174
28.1.4	Application . . . . .	174
28.2	Non-functional Requirements . . . . .	176
28.2.1	Lifelong Learning . . . . .	176

28.2.2	Modifiability . . . . .	176
28.2.3	Durability . . . . .	177
28.2.4	Security . . . . .	177
<b>29</b>	<b>Software Architecture</b>	<b>179</b>
29.1	Deployment Overview . . . . .	179
29.1.1	Game Server . . . . .	180
29.1.2	Smartphones . . . . .	181
29.2	Editor . . . . .	181
29.3	Amazing Puzzle Race . . . . .	184
<b>30</b>	<b>User Interface</b>	<b>185</b>
30.1	Game Screens . . . . .	185
30.1.1	Opening Screen . . . . .	185
30.1.2	New Race Selection Screen . . . . .	185
30.1.3	Start Race Screen . . . . .	186
30.1.4	Open Race Screens . . . . .	186
30.1.5	Task Screen . . . . .	189
30.1.6	Race Information Screen . . . . .	190
30.1.7	Finished Race Screen . . . . .	191
30.2	Editor Pages . . . . .	192
30.2.1	Start Page . . . . .	192
30.2.2	Details Page . . . . .	192
30.2.3	Tasks Page . . . . .	193
30.2.4	Routes Page . . . . .	195
<b>VII</b>	<b>Conclusions</b>	<b>199</b>
<b>31</b>	<b>Summary</b>	<b>201</b>
<b>32</b>	<b>Future Work</b>	<b>203</b>
	<b>Bibliography</b>	<b>203</b>
<b>VIII</b>	<b>Appendices</b>	<b>213</b>
<b>A</b>	<b>Tasks</b>	<b>215</b>
<b>B</b>	<b>Use Cases</b>	<b>239</b>
<b>C</b>	<b>Class Diagram</b>	<b>247</b>



<b>D</b>	<b>The Survey</b>	<b>251</b>
<b>E</b>	<b>Survey Data</b>	<b>259</b>
	E.1 Data . . . . .	259
	E.2 Comments . . . . .	261
<b>F</b>	<b>Excursions</b>	<b>263</b>
	F.1 Excursion 1 . . . . .	263
	F.2 Excursion 2 . . . . .	264
	F.3 Excursion 3 . . . . .	265
<b>G</b>	<b>Demonstration</b>	<b>267</b>
	G.1 Contestant Information . . . . .	267
	G.2 Handouts . . . . .	269
	G.3 Images from the Demonstration Day . . . . .	276



---

## List of Tables

---

1.1	Research Questions . . . . .	3
3.1	Milestones . . . . .	11
3.2	Sprints . . . . .	12
3.3	General Risks . . . . .	17
3.4	Demonstration Specific Risks . . . . .	18
5.1	Prestudy Research Questions . . . . .	25
7.1	Worldwide Smartphone Sales to End Users by Operating System in 3Q10 (units in thousands) . . . . .	40
7.2	Most common Android smartphones used in the U.S. in December 2010 . . . . .	41
14.1	Functional Task Requirements . . . . .	76
14.2	Functional Hint Requirements . . . . .	76
14.3	Functional Answer Validation Requirements . . . . .	77
14.4	Functional GPS Requirements . . . . .	77
14.5	Functional Barcode Scanning Requirements . . . . .	77
14.6	Functional Game State Requirements . . . . .	78
14.7	Functional Misc Requirements . . . . .	78
14.8	Non-functional Requirements . . . . .	79
18.1	Sprint 1 Requirements . . . . .	98
19.1	Sprint 2 Requirements . . . . .	108
20.1	Sprint 3 Requirements . . . . .	121
24.1	SUS Results . . . . .	143
24.2	EGameFlow Results - Concentration . . . . .	144
24.3	EGameFlow Results - Goal Clarity . . . . .	144
24.4	EGameFlow Results - Feedback . . . . .	144
24.5	EGameFlow Results - Challenge . . . . .	145
24.6	EGameFlow Results - Autonomy . . . . .	145
24.7	EGameFlow Results - Immersion . . . . .	145

24.8	EGameFlow Results - Social Interaction . . . . .	146
24.9	EGameFlow Results - Knowledge Improvement . . . . .	146
25.1	EGameFlow Games [33] Compared With Amazing City Game (ACG) . . . . .	150
26.1	Platform Research Questions . . . . .	161
A.1	Task - Welcome . . . . .	216
A.2	Task - Kristiansten Fortress . . . . .	217
A.3	Task - The Landmark . . . . .	218
A.4	Task - Locating Nidarosdomen . . . . .	219
A.5	Task - The War Memorial . . . . .	220
A.6	Task - The Main Facade . . . . .	221
A.7	Task - Locating Stiftsgården . . . . .	222
A.8	Task - The Iron Gate . . . . .	223
A.9	Task - Locating Gamle Bybro . . . . .	224
A.10	Task - Bridge Building . . . . .	225
A.11	Task - Lykkens Portal . . . . .	226
A.12	Task - Locating Britannia . . . . .	227
A.13	Task - The Production Company . . . . .	228
A.14	Task - The Flag . . . . .	229
A.15	Task - Locating Vår Frue Kirke . . . . .	230
A.16	Task - Vår Frue Kirke Facts . . . . .	231
A.17	Task - Locating Torvet . . . . .	232
A.18	Task - Torvet Shopsavvy . . . . .	233
A.19	Task - Locating Ravnkloa . . . . .	235
A.20	Task - The Writer . . . . .	236
A.21	Task - The Avenue . . . . .	237
A.22	Task - The End . . . . .	238
D.1	System Usability Scale Survey . . . . .	252
D.2	EGameFlow Survey - Concentration . . . . .	253
D.3	EGameFlow Survey - Goal Clarity . . . . .	253
D.4	EGameFlow Survey - Feedback . . . . .	254
D.5	EGameFlow Survey - Challenge . . . . .	255
D.6	EGameFlow Survey - Autonomy . . . . .	255
D.7	EGameFlow Survey - Immersion . . . . .	256
D.8	EGameFlow Survey - Social Interaction . . . . .	257
D.9	EGameFlow Survey - Knowledge Improvement . . . . .	257

---

## List of Figures

---

8.1	QR Code and Barcode . . . . .	45
8.2	QR Code Structure . . . . .	46
8.3	Google Goggles Screenshots . . . . .	47
8.4	Smart Tools Screenshot . . . . .	48
8.5	Layar Screenshots . . . . .	48
8.6	Shopsavvy Screenshots . . . . .	49
8.7	Shazam Screenshots . . . . .	50
8.8	Flashlight and Compass Screenshots . . . . .	51
10.1	Torvet and Munkegata . . . . .	58
10.2	Vår Frue Kirke and Nidarosdomen . . . . .	58
10.3	Erkebispegården and Stiftsgården . . . . .	59
10.4	Bakklandet and Munkholmen . . . . .	60
10.5	Ravnkloa and Kristiansten Festning . . . . .	62
10.6	Gamle Bybro and Britannia Hotel . . . . .	62
13.1	Shopsavvy Barcodes . . . . .	73
15.1	User Interface: Multiple Choice and Checkbox Design . . . . .	81
15.2	User Interface: Open and Shazam Design . . . . .	82
15.3	User Interface: Location Description and Hint Confirmation Design . . . . .	82
16.1	Data Integration Architecture . . . . .	84
16.2	State Diagram . . . . .	86
16.3	Data Flow Diagram . . . . .	87
16.4	Package Diagram . . . . .	88
16.5	Simplified Class Diagram . . . . .	91
16.6	Entity-Relationship Diagram . . . . .	92
18.1	User Interface: Welcome Screen Screenshot . . . . .	100
18.2	Sequence Diagram: Task Flow . . . . .	101
18.3	User Interface: Open and Multiple Choice Screenshots . . . . .	102
18.4	User Interface: Task Layout . . . . .	102
18.5	User Interface: Confirmation Box and Wrong Answer Screenshots . . . . .	103
19.1	User Interface: GPS Screenshot . . . . .	111

19.2	User Interface: Shazam Screenshot . . . . .	114
19.3	User Interface: Task with Hints Screenshot . . . . .	115
19.4	User Interface: Updated Task Layout . . . . .	116
19.5	User Interface: Before and after UI revision . . . . .	117
20.1	User Interface: Checkbox Screenshot . . . . .	123
20.2	User Interface: Reset Database Options Menu and Custom Dialog Screenshots . . . . .	125
21.1	User Interface: GPS Progress and Completed Screenshot . . . . .	130
23.1	Using the Smartphone . . . . .	139
23.2	Solving a Task . . . . .	139
23.3	Filling Out the Survey at Burger King . . . . .	140
29.1	Platform Data Integration Architecture . . . . .	179
29.2	Platform Deployment Diagram . . . . .	180
29.3	Workflow For Editing Races . . . . .	182
29.4	Workflow for Editing Tasks . . . . .	183
29.5	Amazing Puzzle Race Screen Flow . . . . .	184
30.1	Opening Screen . . . . .	186
30.2	New Race Screen and Start Race Screen . . . . .	187
30.3	Open Race from SD Card and Barcode Screens . . . . .	188
30.4	Open Race From Web Screen . . . . .	188
30.5	Task Screens . . . . .	190
30.6	Race Information Screen . . . . .	191
30.7	Finished Race Screen . . . . .	192
30.8	Editor Start Page . . . . .	193
30.9	Editor Details Page . . . . .	194
30.10	Editor Tasks Page . . . . .	196
30.11	Editor Routes Page . . . . .	197
A.1	Torvet Shopsavvy Barcodes . . . . .	234
B.1	Use Case: Location Task . . . . .	239
B.2	Use Case: Barcode Task . . . . .	240
B.3	Use Case: Open Task - The Iron Gate . . . . .	241
B.4	Use Case: Multiple Choice Task - The Main Facade . . . . .	242
B.5	Use Case: Checkbox Task - Kristiansten Fortress . . . . .	243
B.6	Use Case: Shopsavvy Challenge - Torvet Shopsavvy . . . . .	244
B.7	Use Case: Shazam Challenge - The Production Company . . . . .	245
C.1	Class Diagram . . . . .	248

C.2	Content Class Diagram . . . . .	249
C.3	UI Class Diagram . . . . .	250
G.1	The Groups and Tutors at Hovedbygget . . . . .	276
G.2	Some Groups and Tutors at Kristiansten Fortress . . . . .	277
G.3	A Group with Tutor solving a Task at Kristiansten Fortress . . . . .	277
G.4	A Group with Tutor traveling to a new Location . . . . .	278
G.5	A Group solving a Task at Stiftsgården . . . . .	278
G.6	A Group asking the Tourist Information about a Task . . . . .	279
G.7	A Group with Tutor solving a Task at Ravnkloa . . . . .	279





# CHAPTER 1

---

## Introduction

---

This chapter is an introduction outlining the Motivation, Project Goal, Research Questions, Research Method, and Thesis Structure for the thesis.

### 1.1 Motivation

Today, the game industry is a multi-billion dollar industry. In 2010 alone, American consumers spent 25.1 billion dollars on video games, hardware, and accessories. In addition, 72 percent of American households play computer or video games, and 55 percent of gamers play games on their phones or other handheld devices [71]. This is not surprising when one considers the technological developments made in the area of mobile devices in recent years. Mobile games have been around since the 1990's, but did not become a big market until the release of smartphones like iPhone and Android-based phones. These devices also have a number of technical capabilities that enable for many different kinds of gameplay.

It goes without saying that the mobile platform is extremely interesting from a commercial point of view. Less obvious is the fact that it is also very interesting from an academic point of view. Several studies have shown that retention of learning is better when using game-based learning rather than traditional learning [65]. Educational games are a well-established genre, and these games have proven to be useful in teaching content in a different and fun way. Educators are starting to realize the potential in entertainment software and are starting to utilize it in learning environments. With the availability of mobile devices with technical capabilities like wireless Internet and GPS, pervasive games have become a lot more viable.

Pervasive games can be combined with educational games, and can prove interesting as they encourage participation and activity from the users, as well as

being engaging and ubiquitous. All of these are vital areas in lifelong learning, a modern approach to education. These educational pervasive games could be used, for instance, in schools and universities as part of the curriculum, in tourist information centers to provide guidance to tourists, or in companies to train staff.

## 1.2 Project Goal

The goal of this project is to design and develop a prototype, an educational pervasive game, for the mobile platform Android where users can run knowledge competitions within the context of the tech-city Trondheim. The prototype should include elements of lifelong learning, and the research should therefore be focused on pervasive games in the context of lifelong learning. The prototype should then be used to run a race where users are organized in groups and compete with each other in the city using Android mobile phones. After the race, the prototype should be evaluated in terms of educational benefit and usability. Based on the results of the evaluation, a design for a general platform for lifelong learning where users can both create and run their own knowledge competitions should be proposed.

## 1.3 Research Questions

To structure our research, we have formulated the research questions found in Table 1.1.

### 1.3.1 Lifelong Learning and Pervasive Games

In this thesis we intend to create a pervasive game under the context of lifelong learning. To prepare for this we needed to find the relationship between pervasive games and lifelong learning, which gives us Research Question 1.3. In order to find the similarities we first needed to study these concepts separately, which gives us Research Question 1.1 and 1.2.

### 1.3.2 Applying Pervasive Games Under the Context of Lifelong Learning

In order for us to create a prototype of a game that supports lifelong learning, we need to perform a study on how we can apply pervasive games under the

RQ1	Lifelong learning and pervasive games
RQ1.1	What is lifelong learning?
RQ1.2	What is a pervasive game?
RQ1.3	What is the relationship between lifelong learning and pervasive games?
RQ2	Applying pervasive games under the context of lifelong learning
RQ2.1	How can we apply pervasive games under the context of lifelong learning?
RQ2.2	How can we use an experiment to describe the idea?
RQ3	Pervasive game platform for lifelong learning
RQ3.1	How can we extend this experiment to be a platform to motivate and support lifelong learning?

Table 1.1: Research Questions

context of lifelong learning, hence Research Question 2.1. We also need to have a plan for how to extract information from our prototype. Research Question 2.2 concerns how an experiment can be used to describe this idea.

### 1.3.3 Pervasive Game Platform for Lifelong Learning

Finally, the goal of this thesis is to use the experiences from our prototype and experiment to create an extended platform for creating knowledge games. This gives us Research Question 3.1.

## 1.4 Research Method

Zelkowitz and Wallace [85] describe four different research methods:

- **Scientific method:** Scientists develop a theory in order to explain a phenomenon and then propose a hypothesis. The hypothesis is then tested, and data is collected to verify or refute the claims of the hypothesis.
- **Engineering method:** Engineers develop and test a prototype to a hypothesis. Based upon the results of the testing, they improve the prototype until no further improvements are required.

- **Empirical method:** A statistical method is proposed in order to validate a hypothesis. Data is collected to verify the hypothesis.
- **Analytical method:** A formal theory is developed, and the results gathered from the theory is compared with empirical observations.

In this master thesis we have used the engineering method as our main research method. In addition, we have used a survey to collect data, and we have also studied theoretical papers in order to answer our research questions.

We have researched pervasive games, lifelong learning, and the operating system Android. We then created a prototype of an educational pervasive game on Android based on our research. We used this prototype for an experiment where students competed against each other in a race. The race consisted of completing different tasks related to the history of Trondheim, while traveling between many of the historical sites in the city. To evaluate this experiment we used a survey based on System Usability Scale [27] and EGameFlow [33], as well as observations made by the game officials during the demonstration. The results gathered from the evaluation was then used to design a general platform for creating puzzle race games similar to the prototype.

## 1.5 Thesis Structure

This thesis consists of eight parts.

**Part I** contains the project plan. This includes areas such as milestones, risk analysis, and group organization.

**Part II** contains the prestudy. The prestudy includes lifelong learning, pervasive games, Android with extension applications, Trondheim, as well as related work.

**Part III** contains the design chosen for the prototype. This part includes information about the experiment, as well as task design, requirements, user interface, and software architecture of the prototype used in the experiment.

**Part IV** contains the implementation of the prototype.

**Part V** contains the results of the experiment. This includes information about the demonstration day, the survey with results, as well as an evaluation of the demonstration, survey, prototype, and risk analysis.

**Part VI** contains the platform design based on the experiment. This includes a discussion on topics such as server solutions, financing, security, and an online community. It also includes requirements for the platform, system architecture, and an user interface proposal.

**Part VII** contains the conclusion of the thesis. The conclusion includes a summary and future work.

**Part VIII** contains all the appendices for the thesis.



**Part I**  
**Project Plan**





## CHAPTER 2

---

### Introduction

---

In this part, the project plan will be presented. This includes the chapters Project Context and Organization. The Project Context chapter contains general project information like milestones, development strategy, and risk analysis. It also contains specific information about templates and standards, tools, and version control procedures. The Organization chapter contains information about the group. This includes group structure, group work hours, and meetings.



# CHAPTER 3

---

## Project Context

---

This chapter contains information related to the project plan. This includes Milestones, Development Strategy, Risk Analysis, Templates and Standards, Tools, and Version Control Procedures.

### 3.1 Milestones

The different milestones of the project are outlined in Table 3.1, and the different phases of the project are detailed in the sections below.

Milestone	Start	Finish
Prestudy	19th of January	1st of February
Design	24th of January	11th of March
Implementation	14th of March	1st of April
Content and Demonstration Preparation	4th of April	2nd of May
Demonstration	3rd of May	3rd of May
Evaluation	4th of May	11th of May
Platform Design	12th of May	27th of May
Finalization	30th of May	15th of June

Table 3.1: Milestones

#### 3.1.1 Prestudy

This phase consists of gaining an insight into the different subjects that concern this thesis. This preliminary study will also include creating the research part of

the written report.

### 3.1.2 Design

This phase consists of gathering requirements and creating a software architecture for the game. The phase will be revisited several times during the implementation phase.

### 3.1.3 Implementation

This phase consists of the iterative process of creating the game. The overall architecture should be implemented, and then the actual code should be written, along with tests and a thorough documentation. The implementation phase is divided into three sprints.

<b>Milestone</b>	<b>Start</b>	<b>Finish</b>
Sprint 1	14th of March	18th of March
Sprint 2	21st of March	25th of March
Sprint 3	28th of March	1st of April

Table 3.2: Sprints

### 3.1.4 Content and Demonstration Preparation

This phase consists of preparing for the demonstration on 3rd of May. The content of the game must also be prepared. This includes the creation and finalizing of tasks.

### 3.1.5 Demonstration

This is a one day test consisting of letting several groups compete against each other. The experiences and results from this demonstration will form the basis for the evaluation.

### 3.1.6 Evaluation

This phase consists of evaluating the experiences and results from the demonstration, and will form part of the basis for the platform design.

### 3.1.7 Platform Design

This phase consists of designing the platform, based on the evaluation of the results and experiences from our game, as well as any conclusions made on the background of our research questions.

### 3.1.8 Finalization

This phase consists of finalizing the thesis document.

## 3.2 Development Strategy

In order to implement the prototype, we are going to use a combination of the waterfall and the iterative software development strategy. We will make an overall design of the application using a waterfall approach, which means we will have a good idea about what the system should look like; and what we are going to implement. The implementation will be based on an iterative software development strategy. This means that the implementation process will be divided into smaller segments of design, implementation, and testing; thus allowing for a better planning of the time used on implementation.

## 3.3 Risk Analysis

This section is about the risks in this project. They are separated into two parts, general risks and demonstration specific risks.

### 3.3.1 General Risks

The following risks are the general risks in this project and they are summarized in Table 3.3.

### **Internal Conflicts**

Conflicts between team members can occur, and is even more likely since we will be working closely together over a long period of time. It is therefore important to communicate openly, and handle issues internally before they become too much to handle. Should this be insufficient one could always seek help from the supervisors, or in the worst case scenario - dissolve the group.

### **Android Difficulty**

Development with Android is something none of us are familiar with. If we spend too much time getting an understanding of the framework, development will take more time than expected and other parts of the project will suffer. We can avoid this by planning accordingly and allocate sufficient time to understand Android, as well as start development early.

### **Sickness**

With teams of any size, sickness is always a potential problem. For a small team like ours, sickness is even more dangerous since each of us make up a large part of the work force. There is no viable way to reduce the probability of this occurring, but we can reduce the impact by redelegating tasks when someone is ill, and working more to make up for the hours lost.

### **Bad Prototype Design**

The survey and evaluation tells us that the concept for the prototype is bad. This is a risk that can make us unable to design a satisfactory platform, which would have a huge impact on the project. One way to reduce the probability of this occurring is to interview teachers to verify the learning aspect of the concept. The impact can be reduced by making a survey that will give us useful information even if the design of the prototype is bad.

## **3.3.2 Demonstration Specific Risks**

The following are the risks specific to the demonstration of the prototype and they are summarized in Table 3.4.

### **Battery**

The low battery time for smartphones is a well-known issue, especially when GPS is enabled or the touch screen is used extensively. This can disrupt the demonstration of our prototype, should any of the phones run out of battery. We need to program with battery usage in mind, i.e. use GPS as little as possible.

### **Wi-Fi Coverage**

The Wi-Fi coverage of Trondheim is limited to the town center. Wi-Fi will therefore not be available on certain locations. With no Internet connections, task solving will be difficult. To avoid this problem we need to provide SIM cards so that 3G can be used if there is no Wi-Fi coverage on a certain location.

### **Prototype Errors**

A number of problems may occur during the demonstration of the prototype. As mentioned previously, battery is an issue. Also, we might run into problems with prototype bugs, like GPS not working, or other unforeseen environmental factors. We can reduce the impact of this by implementing bypasses in our prototype, so that it is possible to progress even if something is not working correctly. We can also implement resets just in case.

### **Changes at Task Locations**

Should the nature or the availability of a task location change for the demonstration date, the task would need to be changed or excluded from the prototype. Since the task locations are out of our control, there is no way we can reduce the chance of this happening. We can however mitigate the effect by having enough tasks to fill the gap left by an excluded task, or keep an eye on task locations so that we can change tasks in time in case of any changes.

### **Unclear Tasks**

When creating task descriptions, one runs the risk of making tasks that are unclear/ambiguous. This can lead to frustration from participants and/or they may get stuck. This can be avoided by scrutinizing tasks and making sure that they are clear and have no ambiguity. Should that not work, the participants could always contact a demonstration supervisor and receive clarification.

### Participants Unfamiliar with Android

Demonstration participants may be unfamiliar with using an Android device. This can lead to frustration and problems during the demonstration. The problem can be avoided by giving them a brief introduction before the demonstration starts, and prepare the phones ahead of time for ease of use.

### Unfavourable Weather Conditions

Weather conditions in Trondheim are somewhat unstable, and there is always the risk of bad weather during the demonstration. This can be a problem because the participants need to interact with the phone quite often, and mostly outside. This is especially the case with GPS location tasks. In case of bad weather, groups should have an umbrella. If that is not viable, the demonstration should be postponed if possible.

## 3.4 Templates and Standards

- **L<sup>A</sup>T<sub>E</sub>X Standard:** We are following the standard L<sup>A</sup>T<sub>E</sub>X-hierarchy: part, chapter, section, subsection, subsubsection, and paragraph. Labels are properly named so it is easy to find out if we are referring to a figure, table, or a specific section. Implementation elements, such as class names, methods, fields, and database tables, are presented using `verbatim`.
- **Naming Convention For L<sup>A</sup>T<sub>E</sub>X Files:** \*.tex filenames should consist of lower case letters, and have no white spaces. Names consisting of multiple words should be separated by hyphens.
- **Diagram Standard:** Diagrams are all be stored in the PNG format, and the name describes what kind of diagram it is.
- **Meeting Minutes Template:** Meeting minutes contains the following elements: date, participants, what happened, and things to do.

## 3.5 Tools

- **SVN:** TortoiseSVN
- **Project Storage:** Group project space at `\\sambaad.stud.ntnu.no\groups`



ID	Activity	Risk factor	Impact	Consequence	Probability	Strategy and actions	Conclusion
1	All	Internal Conflicts	H	Inefficient work, unable to make decisions.	L	Communicate openly and handle problems early. Take to supervisor if all else fails.	M
2	Development	Android Difficulty	H	Delayed development. Rest of the project suffers.	L	Start development early. Allocate some time to understanding Android.	M
3	All	Sickness	H	Reduced workforce	L	Redelegate tasks. Work more to catch up.	M
4	All	Bad prototype design	H	Unable to design satisfactory platform.	L	Interview teacher. Make good survey questions.	M

Table 3.3: General Risks (Annotations - H: High, M: Medium, L: low)

ID	Activity	Risk factor	Impact	Consequence	Probability	Strategy and actions	Conclusion
5	Demonstration	Battery	H	Disruption of demonstration	H	Program with conservation of battery in mind. Use GPS as little as possible. Test GPS battery usage early.	H
6	Demonstration	Wi-Fi Coverage	L	Increased difficulty on task solving.	M	Provide SIM cards so that 3G is available when needed.	M-L
7	Demonstration	Prototype Errors	M	Disruption of demonstration.	H	Implement bypasses and resets.	H-M
8	Demonstration	Task Location Changes	L	Task needs to be changed or dropped	L	Have enough tasks to fill gaps. Keep an eye on task locations.	L
9	Demonstration	Task Unclear/Ambiguous	L	Frustrated/stuck participants	L	Check tasks for ambiguity and clarity. Help participants if needed.	L
10	Demonstration	Participants Unfamiliar With Android	L	Frustrated participants. Problems during demonstration.	M	Give introduction to Android. Prepare phones ahead of time.	M-L
11	Demonstration	Unfavourable Weather Conditions	M	Difficult to use device.	H	Use umbrellas or postpone demonstration	H-M

Table 3.4: Demonstration Specific Risks (Annotations - H: High, M: Medium, L: low)

- **Programming Language:** Java. Android SDK. Developing for Android 2.1.
- **Programming IDE:** Eclipse
- **Diagram Editor:** Microsoft Visio 2007/2010
- **Report Writing:** L<sup>A</sup>T<sub>E</sub>X
- **Meeting Minutes Writing:** Open Office
- **Mock-up Editor:** Balsamiq Mockups

## 3.6 Version Control Procedures

The following guidelines should be followed when using SVN:

1. Perform updates before committing any changes.
2. Only commit working code.
3. Several people should not work in the same file at once.



# CHAPTER 4

---

## Organization

---

This chapter contains information about the group of the project. This includes Group Structure, Group Work Hours, and Meetings.

### 4.1 Group Structure

In this project we are three people working together. In a group this small we do not feel it is necessary to have a group leader. All decisions will be made by majority decision. In addition, there will be no single secretary, which means that everyone in the group will have this role in turn. The responsibilities of the secretary is to takes notes during meetings, and write meeting minutes.

### 4.2 Group Work Hours

Group work hours is a period where the team participants sit together and work on the project. These hours are between 10:15 and 15:00 Monday to Friday. Work hours are held at building P15 wherever there is space, preferably the 3rd floor.

The intent of these work hours are that the participants can get to know each other better, work and communicate effectively, as well as get help if needed. It is also during these hours that discussions are held, and decisions are made. It is intended that most of the project work should be done during these work periods, but people are definitely encouraged to work separately at home as well.

## 4.3 Meetings

### 4.3.1 Supervisor Meetings

Meetings with the supervisor in the start-up phase are scheduled for every Tuesday at 12:00. In the implementation phase, meetings are scheduled as needed.

### 4.3.2 Internal Meetings

Since there only are 3 participants on this team, and we work closely together every day, most if not all communication will be handled in the group work hours. Should there however be need for an official meeting, any participant may schedule one.

# Part II

## Prestudy





# CHAPTER 5

---

## Introduction

---

In this part of the report we will go through the prestudy for this project. The prestudy will cover the research questions in Table 5.1.

RQ1	Lifelong learning and pervasive games
RQ1.1	What is lifelong learning?
RQ1.2	What is a pervasive game?
RQ1.3	What is the relationship between lifelong learning and pervasive games?
RQ2	Applying pervasive games under the context of lifelong learning
RQ2.1	How can we apply pervasive games under the context of lifelong learning?
RQ2.2	How can we use an experiment to describe the idea?

Table 5.1: Prestudy Research Questions

The prestudy part contains the following chapters: Research Context, Android, Applications from the Android Market, Related Work, and Trondheim. Research Context contains the information related to the research questions for this part. This includes subjects like lifelong learning, pervasive games, and the relationship between them. The chapter also contains information about how pervasive games can be applied under the context of lifelong learning, and how an experiment can be used to describe the idea. The Android chapter contains some general information about Android, Android market share, and the features of Android phones. Applications from the Android Market is a chapter about applications for Android phones that can be utilized in this project. This includes applications like Barcode Scanner and Google Goggles. The Related Work chapter includes information about three pervasive games: the *The Amazing Race*, which is the

tv-show that this experiment is based on, *Shelby Logan's Run*, and *Heroes of Koskenniska*. In addition, two relevant papers are presented. The chapter about Trondheim contains historical information about popular sights in Trondheim. Some of these sights are used in the game.

## Research Context

---

This chapter contains the research related to the research questions in Table 5.1. This includes sections about lifelong learning, pervasive games, and the relationship between them. The chapter also contains information about how pervasive games can be applied under the context of lifelong learning, and how an experiment can be used to describe the idea.

## 6.1 Lifelong Learning and Pervasive Games

### 6.1.1 Lifelong Learning

The Lifelong Learning definition encompasses *all purposeful learning activity, undertaken on an ongoing basis with the aim of improving knowledge, skills and competence* [28].

There are three basic types of purposeful learning activities [28]:

- Formal Learning
- Non-formal Learning
- Informal Learning

Formal learning takes place in education and training institutions in the form of structured programs that are approved by the formal education system. Formal learning leads to recognized diplomas and qualifications. Non-formal learning takes place alongside the education and training systems, and usually does not lead to formalized certificates. This type of learning often takes place in the workplace in the form of structured on-the-job training, through activities of civil society organizations, and through groups such as youth organizations,

trades unions, political parties, and apprenticeships. Other types of complementary formal systems in the areas of art, music and sports, or private tutoring also provide this type of learning. Informal learning is ongoing during everyday life. Unlike the formal and non-formal learning, this type of learning is not necessarily intentional, and may therefore not be recognized even by the individuals themselves. This includes unstructured on-the-job training [28]. Informal learning encompasses a lifelong process where individuals acquire attitudes, values, skills, and knowledge from daily experiences and the surrounding environment [7].

### Traditional Learning vs Lifelong Learning

The traditional view on learning and the more modern view on learning known as lifelong learning differ on a number of points.[21]

- In traditional learning, teachers are the only source of knowledge, and knowledge transfer is primarily from teacher to student. In lifelong learning, teachers are simply guides to sources of knowledge, and students learn by doing themselves.
- Lifelong learning encourages students to learn in groups and from each other, as opposed to traditional learning where students usually work by themselves.
- Tests are used in traditional learning to assess if students have mastered the material and ration out access to further material. Lifelong learning takes a different approach and uses tests to adapt learning strategies and identify new learning opportunities.
- Another important difference is that the educational plan in lifelong learning is highly individualized for each person, as opposed to the traditional method where all learners undergo the same educational plan.
- Furthermore, the traditional learning model identifies strong learners and allows them to continue their education. In lifelong learning, larger segments of the population have access to education over a whole lifetime.

#### 6.1.2 Pervasive Games

When reviewing existing literature it is clear that there has never been a clear definition of what pervasive games are exactly. Most papers, however, seem to focus on three distinct properties that distinguish pervasive games from other games, namely the three properties mentioned in Markus Montola's definition [58].

*A pervasive game is a game that has one or more salient features that expand the contractual magic circle of play spatially, temporally or socially.*

**Spatial Expansion** Spatial expansion indicates that the socially constructed location of the game is unclear or unlimited. The games can be played anywhere and everywhere, which makes it unclear where they are actually played [54]. Pervasive gamers inhabit a game world that is present within the ordinary world. Unlike non-pervasive games, which are usually isolated from their surroundings, pervasive games embrace their environments and contexts. In addition to physical architecture, pervasive games can use objects, vehicles, and properties of the physical world in the game [58].

**Temporal Expansion** Pervasive games expand temporally from the explicit play sessions, which means that the game session is interlaced and mixed with ordinary life [54]. A play session can include sleeping, working, and talking with non-participants. This means that the game moves from the center of attention to the periphery and back again [58]. Temporal expansion also ties in with social expansion, since the temporal span of the game is often so unclear that the players might be unaware whether they are playing at a given moment [54].

**Social Expansion** A consequence of the temporal and spatial configurations of pervasive games is that outsiders usually end up participating in the game. Outsider involvement ranges from full spectatorship to active participation. This is called the social expansion.[58]

The social expansion can be done in a number of different ways. Outsiders may for example be used as game elements, as done in *Prosopopeia* where organizers left messages and artefacts with non-players, and the players were tasked with retrieving them during the game. Outsiders may also constitute an audience to the game, and maybe even influence and/or participate. An example of this is *Whirling Dervishes*, where many bystanders voluntarily joined the game [54] as active players.

## Ethics

The spatial expansion allows the game to be played in many locations simultaneously, taking the games to places where they are not usually supposed to be. The challenges of the game might therefore include the problems of causing

unwanted public disturbance, creating hazardous situations in traffic, and risking the expansion of games to places where they should not be played (such as hospitals or airports). [54]

The temporal expansion may be problematic. The game might require attention at the worst possible times, or generally require too much attention. For instance, the game could require attention when the participant is sleeping or working. In addition, if the game is played constantly, the privacy considerations also become an issue; for instance, the search function of *Botfighters* to find friends. [54]

The social expansion may also cause some problems. The challenges of the game include the risks of drawing unwilling persons into the game. The creation of new social relationships is not without hazards; problems similar to ones encountered in dating services might emerge. In addition borderline players might not realize that they are participating in a game.[54]

## Game Types

There are several types of pervasive games. To name a few: treasure hunts, assassination games, pervasive larps, alternate reality games, smart street sports, playful public performances, urban adventure games, and reality games.[58]

**Treasure Hunts** Treasure hunts are games where the players try to find certain objects (treasures) in an unlimited game space. Often these objects are not valuable, but the discovery is a reward in itself. Treasure hunts can be either competitions between individuals or teams, or solo missions where the hunter challenges him- or herself. The challenges can either be physical, mental, or social. [58]

One notable example of a treasure hunt game is *Geocaching* [58], where users utilize GPS devices in order to hide and discover containers that are called "caches". Another example is *Insectopia* [58], where players add insects to their collections by getting close to bluetooth devices. Each active bluetooth device has a unique identification, which is translated by the game into a specific insect, and the purpose is to get the largest and rarest collection possible. Two very influential treasure hunt games are mentioned later in the report. These are *The Amazing Race* in section 9.1, and *Shelby Hogan's Run* in section 9.2.

**Assassination Games** Assassination games come in many different forms, but the most common theme is that each player has a target that they have to

”kill”. These games are played out in ordinary environments and may take up to weeks to complete. Very often, the assassin takes over their victim’s mission, and the game continues until only one assassin remains.[57]

This game form emerged in the mid 1960’s, when students in the United States starting playing games based on the movie *La Decima Vitima* (1965). There are some concerns about these forms of games, especially in regards to the current state of the political climate in the U.S.. In fact, the assassination-centered game *Killers* has been banned from university campuses.[58]

**Pervasive Larps** Larp stands for live-action role-playing, and involves physically acting out as a character in an environment. Pervasive larp is a type of pervasive gaming that uses live-action role-playing techniques. The requirement for this game type is that the user has to physically act as the character of the game, and pretend to be said character.[58]

One example is *Prosopopeia Bardo 2: Momentum*, a pervasive larp that lasted for five weeks in Stockholm in 2006; where the participants took on roles as dead revolutionaries in a game world that merged the ordinary and lucid world into a coherent role-playing experience. Also, assassination games and pervasive larps can overlap. *Killers* is an example of a game that can be played simultaneously as an assassination game and a pervasive larp, if the participants pretend to be and act like the characters in the game [58].

**Alternate Reality Games** Alternate Reality Games (short: ARG) tell interactive stories in the real world, but add additional layers of meaning, depth and interactivity by utilizing technology. Typically, ARGs favour collaboration over competition, and feature large self-organized player communities, Internet-based gameplay, and secretive production styles. A popular form of gameplay featured in ARGs are puzzles.[58]

*The Beast* is an example of an alternate reality game. It has more than 3 million active participants and is considered one of the most successful pervasive game [2]. It was created by *Microsoft* as a marketing campaign for the movie *A.I.: Artificial Intelligence*, and ran for three months in 2001 [5]. The game was set up with a great amount of game content such as websites, emails and even real telephone conversations from actors playing game characters. The entry into the game was by solving hidden codes in the movie’s promotional material such as trailers and posters, and ultimately the goal of the game was to solve a mystery. During the game an external group called ”The Cloudmakers” was formed whose members collaborated to solve this mystery.

**Smart Street Sports** Smart street sports require both physical exercise and tactical thinking. They are usually played outside in urban areas. In some games the players all move in the physical space using GPS devices, cellular phones, handheld computers; while other games combine physical and virtual gameplay. Smart street sports usually depend on technology, and are characterized by that they are quick to pick up, require very little preparation, and have a high degree of replayability [58]. A typical game for this game type is Pac-Man. Several versions of Pac-Man have been created; *PacManhattan*, *Human Pacman*, and *Pac-Lan* to name a few.

**Playful Public Performance** Playful Public Performances are games usually played in public spaces, and aim towards having fun by performing, playing, and creating something enjoyable for bystanders to watch. The participants move in the physical space, and are supported by technological devices like cell phones and laptops.[58]

One example of a playful public performance is *Big Urban Game*, where three huge inflated game pieces were moved in a route voted on by the public. Other examples involve games previously mentioned in this section, such as *Shelby Logan's Run*, *Momentum* and *The Amazing Race*, which all contain public performances to some extent. [58]

**Urban Adventure Games** Urban adventure games combine stories and puzzles with city spaces. These games take the user to historical and cultural areas of significance to solve puzzles and follow stories, as well as learning the history of the area [58].

**Reality Games** Reality games (although debatable whether or not they are games at all) are pervasive events that intentionally play around with the concepts of real and reality. They are usually more oriented towards spontaneous play rather than more formal rule-focused play. The aim is to have participants and bystanders experience the playing environment differently by changing the perception of participants and bystanders, or actively changing the environment. This form of game is played for the benefit of bystanders and unaware participants, and the goal is usually to make them question their surroundings.[58]

## Technology

Technological progress has increased the extent of pervasive games. Montola [58] points to three specific technologies as important. The first is the use of



mobile phones with large screens to be able to present information. The second is wireless Internet (both by being in an area with Wi-Fi hotspots and mobile Internet like 3G), and the third is location services such as GPS.

GPS is interesting in that it is not always accurate everywhere. Specifically, indoors and urban areas with many tall buildings. This leads to usage of other technologies such as depending on information from base stations. GPS inaccuracy has also interestingly enough been utilized as a feature in some pervasive games [32].

### Pervasive Learning

*Pervasive learning refers to a type of learning where the real and virtual worlds are bridged for learning activities in a specific context [70].* Any system that implements the pervasive learning paradigm is referred to as Pervasive Learning Spaces.

### 6.1.3 The Relationship between Lifelong Learning and Pervasive Games

In order to clearly see the relationship between lifelong learning and pervasive games, one would have to study the two subjects. The four main characteristics of lifelong learning includes informal learning, self-motivated learning, self-funded learning, and universal participation [7]. The first characteristic of lifelong learning encompasses both formal and non-formal/informal types of education and training [7]. Thus the pervasive game should provide not only formal learning, but non-formal or informal as well. Since a pervasive game in fact is a game, and therefore should be fun to use, it makes an excellent platform for including informal learning. Whether or not it contains formal and non-formal learning depends on the motivation behind the creation of the game. Games played as a part of the curriculum in a school or university would include formal learning. Games played at non-educational institutions like museums and tourist information centers would include non-formal learning.

The second characteristic of lifelong learning is self-motivated learning. It emphasizes on the need for individuals to take responsibility for their own learning [7]. Again, a pervasive game is an excellent platform for providing self-motivated learning. If the learning is fun, it is easier for people to motivate themselves to learn. A pervasive game could be motivation to learn in itself, but through the game, players might also be motivated to learn more about the subjects taught in the game.

The third characteristic of lifelong learning is self-funded learning. This characteristic is not something that can be included in pervasive games.

The fourth and final characteristic of lifelong learning is universal participation. Universal participation includes informal and formal learning for all purposes - social, economic, and personal. This includes [7]:

- **Learning to do:** acquiring and applying skills.
- **Learning to be:** promoting creativity and personal fulfillment.
- **Learning to know:** an approach to learning that is flexible, critical, and capable.
- **Learning to live together:** exercising tolerance, understanding, and mutual respect.

The social aspect of universal learning, "learning to live together", corresponds well with the social expansion of pervasive games in the way that the players may include outsiders in the game. In order for the outsiders to be included in the game, the players must exercise tolerance and understanding towards the outsiders. They must also respect the outsiders choice of being a part of the game, as well as their choices within the game. The same goes for other players. Additionally, all educational pervasive games include "learning to do". A key element to progress in educational pervasive game is to acquire new skills, and apply those skills within the game.

The aspect "learning to be" corresponds well with both the spatial and social expansion of pervasive games. In the spatial expansions, players can use physical architecture, objects, vehicles, and properties of the physical world in the game. The social expansion implies that the game can involve bystanders. Both these elements motivates the players to be more creative in the way they play the game. In addition, personal fulfillment could be achieved by solving difficult tasks in the game. In order to solve difficult tasks, the players may have to find other sources of information. It is therefore important to be flexible in retrieval of knowledge. In addition, not all sources of knowledge are good sources, which means that players must be skeptical to the information's validity. Both of these elements corresponds well with the "learning to know" characteristic of universal participation.

## 6.2 Applying Pervasive Games Under the Context of Lifelong Learning

### 6.2.1 How Can We Apply Pervasive Games Under the Context of Lifelong Learning?

The following subsections detail how pervasive games can be applied under the context of lifelong learning by considering the pervasive games and lifelong learning from different viewpoints. The viewpoints used are OECD's strategies for implementing lifelong learning, the differences between traditional learning and lifelong learning, and the properties that make a game pervasive.

#### OECD's Strategies for Implementing Lifelong Learning

OECD (Organisation for Economic Co-operation and Development) has put forward 5 strategies for countries to use in implementing lifelong learning, three of which can be used within games.[34]

The first strategy is to recognize all forms of learning, and not just formal courses of study. As previously discussed in Section 6.1.1, lifelong learning consists of three types of learning: formal, non-formal, and informal learning. A game that wishes to achieve lifelong learning will have to incorporate at least one of these forms of learning. The type of learning experience provided by the game will depend on what kind of game it is, and what one wishes to achieve with the game. For instance, a game organized by an official educational institution in combination with a subject will be considered as formal learning. An informal game would ideally be entertaining to the player and yield a subtle learning effect. These two could be combined into a game that could be used as part of a course, where the learning effect would be perceived as unintentional. *Lecture Quiz*, *World of Wisdom*, and *Age of Computers* are games that are based on formal as well as informal learning. *Heroes of Koskenniska* is an educational pervasive game that takes place in the area around a museum, and is based on both non-formal and informal learning.

The second strategy emphasizes on the importance of developing foundation skills, in particular motivation and the capacity for self-directed learning. International evidence shows that people without an upper secondary qualification and without strong literacy skills are among the least likely to participate in further education and training as adults [7]. An educational game could provide the user with a general understanding of a certain learning field, and thus promote further learning. For example, a game providing an overview of the history,

architecture, or society of a city in such a way that the players would become motivated to learn more about the field.

The third strategy concerns the reformulation of access and equity priorities in a lifelong context, by looking into the different settings in life where learning can occur. A game that wishes to implement this strategy should be made available to as large an audience as possible, and contain a large and varied content base so that the learning experience for each individual is tailored to their specific educational needs and skill levels.

### **Differences between Traditional Learning and Lifelong Learning**

If we look at the differences between the traditional learning and the lifelong learning models (see Section 6.1.1), we can make a number of observations.

In order to achieve lifelong learning, the game should be a guide to sources of knowledge, meaning that the players will learn by finding information themselves. The players should also learn by doing. Preferably the game should provide some sort of cooperative play so that the students can learn in groups and from each other.

Individualized educational plans specific to each student is a key point in the lifelong learning model. To achieve this, the game can track how the student performs, and adjust the difficulty or teaching strategy accordingly. By doing this, the student will always receive an educational benefit from playing the game. Also, traditional learning has a tendency to pick out the "good" learners and allow them to continue their education, leaving "bad" learners behind. By adapting the difficulty to the student instead of picking students suitable to the difficulty, larger parts of the population can be included.

A suggestion for a game structure would be to have the game start with an assessment of the student's knowledge level, continue with a task of the appropriate difficulty, and then increase the difficulty as the student progresses through tasks. It would be beneficial if the student used knowledge learnt in previous tasks to complete newer tasks to make sure that the knowledge is mastered.

### **Properties of Pervasive Games**

Finally we can consider the key elements that make a game pervasive. As stated in section 6.1.2, pervasive games expand on the following three dimensions of play: social, spatial and temporal.

The social expansion usually means that outsiders are included into the game, be it as an aware or unaware participant. One way to include outsiders could be to use the outsiders as sources of information, which would encourage cooperation, and as previously mentioned cooperation is one of the elements the lifelong learning model emphasizes.

The spatial expansion indicates that the location of the game is unclear or unlimited, which means that the pervasive games could be played on multiple locations. The spatial expansion also includes incorporating objects from the real world into the game. One example would be to require players to use a terminal and have them solve a number of tasks on it. By including many different locations and real world objects, it is easier to incorporate the lifelong learning aspect of "learning by doing".

The temporal expansion indicates that the game session is interlaced in everyday life. The players will often not know when they will have to play the game. Pervasive learning games that include the temporal expansion will exist over time, and therefore lifelong learning will take place purely due to the fact that lifelong learning happens over time.

### 6.2.2 How Can We Use an Experiment to Describe the Idea?

As previously discussed in Section 6.2.1, pervasive games are an excellent way to implement several aspects of lifelong learning. In order to use an experiment to describe the idea one would have to create a lifelong learning scenario. For this experiment, we have opted for an educational pervasive game that is designed to teach players about the history of Trondheim. The experiment will include both non-formal and informal learning, as well as many other aspects of lifelong learning. The experiment will also draw inspiration from the "The Amazing Race" show.

Lifelong learning aspects included in the experiment:

- **Learn by doing:** The experiment consists of a number of tasks that has to be completed. In order to complete the tasks the players must search for information.
- **Learn in groups and from each other:** The game should be played in groups to encourage this element of lifelong learning. In addition, outsiders may be included while playing the game.
- **Educators are guides to sources of knowledge:** The tasks of the experiment is created by the educators. This makes the educators guides

to sources of knowledge. Group tutors can also be used as to support this.

- **Learning to know:** The players must be flexible in how they find information to solve the tasks, and critical to what they find.
- **Motivation and self-directed learning:** The content in the game will serve as a sneak peak at Trondheim's history. By playing the game the players could become motivated to investigate the field further on their own.
- **Learning to live together:** The players will need to show tolerance, understanding, and mutual respect while interacting with other people. This includes team-mates, opposing teams, and outsiders.

## Android

---

This chapter contains general information about the operating system Android. It also includes information about Android Market Share and Android Mobile Phone Features.

### 7.1 About Android

Android is an open source mobile operating system built on the open Linux Kernel. Android was built to enable developers to create compelling mobile applications that take full advantage of all that a mobile phone has to offer. Android also does not differentiate between the phone's core applications and third-party applications. They can all be built to have equal access to a phone's capabilities. An application can call upon any of the phone's core functionality such as making calls, sending text messages, or using the camera. It also utilizes a custom virtual machine that was designed to optimize memory and hardware resources in a mobile environment [64]. Developers develop in Java. There is no developers license needed, and it is therefore very cheap to develop for Android (in contrast to for instance iPhone) [3].

### 7.2 Android Market Share

According to Gartner [35], worldwide mobile phone sales saw an increase of 35% in the 3rd quarter of 2010 (3Q10). Smartphone sales saw an increase of 96% in the same period. Smartphone sales accounted for 19.3% of the total mobile sales in 3Q10.

Of worldwide smartphone sales made in 3Q10, Android accounted for 25.5% (see Table 7.1), making it the 2nd most popular OS for that period [35]. In the U.S. Android-based smartphone sales accounted for 44% of all smartphone sales in the 3rd quarter of 2010. Up 11% from the previous quarter.[72]

Company	3Q10 units	3Q10 Market Share (%)	3Q09 units	3Q09 Market Share (%)
Symbian	29480.1	36.6	18314.8	44.6
Android	20500.0	25.5	1424.5	3.5
iOS	13484.4	16.7	7040.4	17.1
Research In Motion	11908.3	14.8	8522.7	20.7
MS Windows Mobile	2247.9	2.8	3259.9	7.9
Linux	1697.1	2.1	1918.5	4.7
Other	1214.8	1.5	612.5	1.5
<b>Total</b>	<b>80532.6</b>	<b>100.04</b>	<b>1093.3</b>	<b>100.0</b>

Table 7.1: Worldwide Smartphone Sales to End Users by Operating System in 3Q10 (units in thousands)

According to the IDC, 19 million smartphones were sold in Western Europe in the 3rd quarter of 2010, an increase of 109% when compared to the 3rd quarter of 2009. 4.3 million Android devices were sold in 3Q10, with HTC, Sony Ericsson and Samsung accounting for 84% of those sales.[45]

## 7.3 Android Mobile Phone Features

Android phones have a number of capabilities that can be useful in a pervasive game setting. Table 7.2 is an overview of the most common Android smartphones used in December 2010 extracted from Millennial Media's Mobile Mix Mobile Index report for December 2010 [56]. Also provided is a checklist on each of the phones' features.

### 7.3.1 GPS

Most Android phones have a GPS device included. GPS is a global positioning system consisting of 24 Earth-orbiting satellites that transmit radio signals. These radio signals contain the satellites' position and the time it transmitted



Phone	Touch	GPS	Camera	Wi-Fi	Accelerometer	Gyroscope	Digital compass	Proximity Sensor	Ambient Light Sensor
HTC Nexus One (Passion)[44]	x	x	x	x	x		x	x	x
Motorola Droid[60]	x	x	x	x	x		x	x	x
HTC G2 Touch Hero[42]	x	x	x	x	x		x		
Samsung Vibrant Galaxy S[47]	x	x	x	x	x	x	x	x	x
HTC Evo[66]	x	x	x	x	x		x	x	x
Motorola Droid 2[59]	x	x	x	x	x		x	x	x
HTC Aria[39]	x	x	x	x	x		x	x	x
HTC Droid Incredible[41]	x	x	x	x	x		x	x	x
Motorola Droid X[61]	x	x	x	x	x		x	x	x
HTC MyTouch 2 (Espresso)[23]	x	x	x	x	x			x	x
HTC MyTouch Magic[43]	x	x	x	x	x		x		
LG Ally[62]	x	x	x	x	x			x	
HTC Desire[40]	x	x	x	x	x		x	x	x
Samsung Moment[50]	x	x	x	x	x			x	
HTC Droid Eris[24]	x		x	x	x				x
<b>Ratio of availability</b>	<b>100%</b>	<b>93.75%</b>	<b>100%</b>	<b>100%</b>	<b>100%</b>	<b>6.25%</b>	<b>75%</b>	<b>81.25%</b>	<b>75%</b>

Table 7.2: Most common Android smartphones used in the U.S. in December 2010

the signal. The distance between a satellite and the receiver can be calculated by subtracting the time that the signal left the satellite from the time that it arrives at the receiver. To determine the three-dimensional position on Earth, the distance of four or more satellites must be measured [30]. This enables the usage of applications that are based on the location of the user. Additionally, one can easily integrate the free Google Maps service into the applications.

### 7.3.2 Wi-Fi

All of the most used Android smartphones are Wi-Fi-capable (see Table 7.2). Wi-Fi networks use radio technologies called IEEE 802.11a, 802.11b or 802.11g to provide secure, reliable, fast wireless connectivity. A Wi-Fi network can be used to connect devices to each other, to the Internet, and to wired networks (which use IEEE 802.3 or Ethernet). Wi-Fi networks operate in the unlicensed 2.4 and 5 GHz radio bands, with an 11 Mbps (802.11b) or 54 Mbps (802.11a) data rate. They can provide real-world performance similar to the basic 10BaseT wired Ethernet networks [82]. This provides the smartphone user with an alternative and much cheaper method of connecting to the Internet than 3G or EDGE.

### 7.3.3 Camera

Virtually every mobile phone has a mobile camera, with increasingly better resolution and quality. This gives interesting options when it comes to taking pictures to use in presentations, and for using barcodes in games (see Section 8.1).

### 7.3.4 Near Field Communication

Near Field Communication or NFC is a short-range, high frequency wireless communication technology that is used for remote identification and data transfer at reading ranges up to a few centimeters [67]. NFC allows for high-speed data transfer, and operates at 13.56 MHz. This means that data can be transferred at speeds up to 424 kbit/s [17]. NFC can be thought of as a subset of the Radio Frequency Identification (RFID) technology. However, NFC enables more diverse communication modes compared to RFID. It is possible for two NFC devices to communicate with each other, and NFC devices can also emulate certain RFID smart cards, which enables NFC devices to use existing reader infrastructure [67]. There are three different operation modes for NFC [67]:

- **Reader/writer mode** can be compared to the traditional RFID operation. NFC devices reads or writes data to or from an NFC compliant tag with data transfer rates up to 106 kbit/s.
- **Peer-to-peer mode** allows two NFC devices to talk to each other with a maximum data transfer rate of 424 kbit/s.
- **Emulation mode** enables the NFC device to emulate a smart card, which operates so that a build-in smart card chip is integrated in the device connected to the NFC antenna.

There are currently just a few mobile phones that support NFC. However, the newest version of Android (2.3), codename *Gingerbread*, supports NFC [16]. The new Nexus S from Google is the first Android mobile that supports NFC, and comes with Android 2.3 [9]. Due to the fact that there are very few mobile phones that support NFC at the time of this project, we are not able to use it. We suggest that future work of this project should include NFC.

### 7.3.5 Gyroscope

Gyroscopes are inertial measurement devices, that are used to detect angular rotation rate of objects. Gyroscopes are widely used for guidance, navigation, airplanes, spacecrafts, missiles, automobiles, and electronics to maintain orientation, control, and stabilization [22]. There are currently very few phones that have a gyroscope. We found only one Android mobile phone that has a gyroscope, and that is Samsung Vibrant Galaxy S [47]. The new Google mobile phone, Nexus S, will also have a gyroscope [9], and more Android phones will probably follow. We therefore suggest to use the gyroscope in the future work of this project.

### 7.3.6 Accelerometer

An accelerometer is an electromechanical device that measures acceleration forces. These forces may be static, like gravity, or dynamic caused by moving or vibrating the accelerometer [31]. An accelerometer can do the following: inertial measurement of velocity and position, vibration and shock measurement, and measurement of gravity to determine orientation [46]. Accelerometer is standard on many modern mobile phones [55], and all of the most used Android mobile phones got this feature integrated (see Table 7.2).

### 7.3.7 Ambient Light Sensor

An ambient light sensor is a photodetector that are used to detect light or brightness in a manner similar to the human eye. They are most commonly found in industrial lighting, consumer electronics, and automotive systems, where they allow light settings to be adjusted automatically in response to changing ambient light conditions [81]. By turning on and off, or adjusting features, the ambient light sensor can conserve battery power, while eliminating the need for manual adjustments.

### 7.3.8 Proximity Sensor

A proximity sensor is able to detect objects without physical contact. One way to do this is by emitting an electromagnetic field or beam and look for changes in the field, but also other sensors might be used to target different objects. For instance a capacitive or photoelectric sensor for a plastic target, or an inductive proximity sensor for a metal target [52].

### 7.3.9 Digital Compass

The compass sensor (magnetometer) provides measurement of magnetic field strength along x, y, and z axis [38]. The compass sensor can be affected by all the magnetic fields around the device, most importantly earth's magnetic field. With the use of the strength of earth's magnetic field along the three directions, the compass sensor can calculate the longitude, which is equivalent to the direction in our daily life; east, south, west, and north [48].

---

## Applications from the Android Market

---

The following applications are third-party applications, found on the Android Market, that could be used by our game to extend its functionality. This includes: Barcode Scanner, Google Goggles, Smart Tools, Layar, Shopsyvvy, Shazam, Flashlight, and Compass.

### 8.1 The Quick Response Code

QR Code, which stands for "Quick Response Code", is a two-dimensional symbology developed by Denso Wave, and was released in 1994 [1]. The fact that the QR Code (see Figure 8.1(a)) is a 2D code means that it contains information in both vertical and horizontal directions, whereas a traditional barcode (see Figure 8.1(b)) only contains information in one direction [63]. As a result, the QR Code can contain a greater volume of information compared to a single-dimension barcode. This also means that the QR Code can be much smaller than a regular barcode while containing the same amount of information. Denso-Wave [10] states that it can encode the same amount of data in approximately one-tenth the space of a traditional barcode.



Figure 8.1: QR Code and Barcode

The QR Codes can hold various types of content. For instance, URLs, e-mail

addresses, telephone numbers, SMSs, MMSs, or contact information in the form of vCard (format for encoding contact information as text) [4]. QR Codes can be created by using one of the many free online generators.

In order to read these QR Codes one would need a QR Code scanner [11]. On Android phones there are several available barcode scanner applications. *Barcode Scanner* includes the ability to display QR Codes on the screen for any application on the device, in addition to using the camera to scan the barcodes. "App Referrer" also display the codes, but in addition it can send the links via e-mail, SMS, or Twitter [12].

QR Code consists of several parts [63] (see Figure 8.2):

- Version Information - information about the QR Code and its version and size.
- Format Information - information of error correcting level and mask pattern.
- Data and Error Correction Keys - the encoded data, error correction patterns are also embedded in the data.
- Position Pattern - used to detect the code's position when decoding.
- Alignment Pattern - used to find the correct angle when decoding.
- Timing Pattern - used to determine a symbol's coordinate in the decoder application.

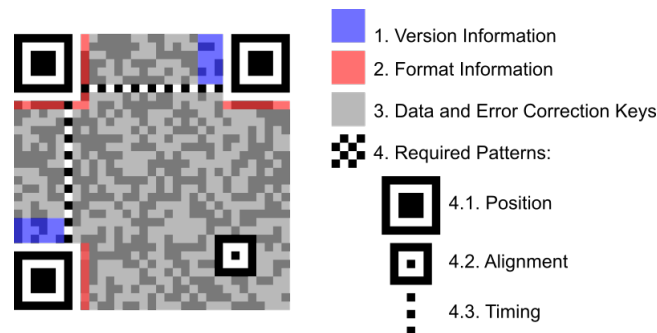
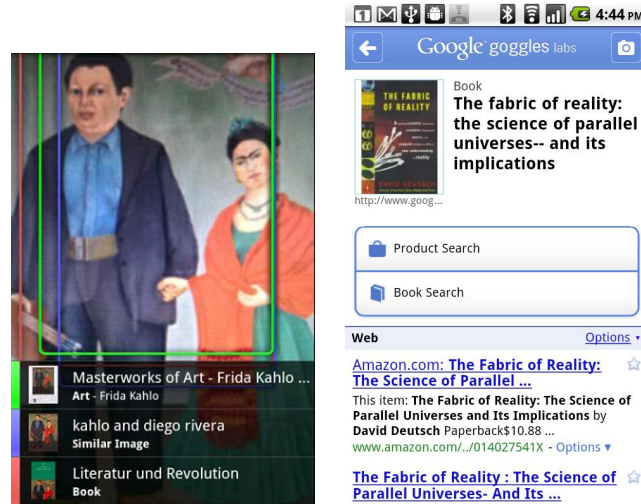


Figure 8.2: QR Code Structure

## 8.2 Google Goggles

Google Goggles is a free image recognition application created by Google. Google Goggles enables the user to use pictures taken with the mobile phone to search

the web, and is therefore ideal for things that are not easy to describe with words [37]. This can be things like text, landmarks, books, contact information, artwork, wine, and logos [36]. Google Goggles currently does not work well with pictures taken of animals, plants, cars, furniture, or apparel [37]. See Figure 8.3 for screenshots of Google Goggles.



(a) Google Goggles Screen-shot 1 (b) Google Goggles Screen-shot 2

Figure 8.3: Google Goggles Screenshots

## 8.3 Smart Tools

The Smart Tools application is a collection of tools that can measure length, angle, distance, height, width, and area of objects using the mobile phone's camera. The collection also includes tools to measure direction, sound and vibration, and can even be used as a metal detector [18]. See Figure 8.4 for a screenshot of Smart Tools.

## 8.4 Layar

Layar is a mobile platform for discovering information about the world around us by using Augmented Reality technology. Layer displays digital information, so called *layers*, by using the camera of the mobile phone. Layar supports a high level of interactivity, which includes audio and visual elements, 3D models, and social sharing capabilities [51]. See Figure 8.5 for screenshots of Layar.

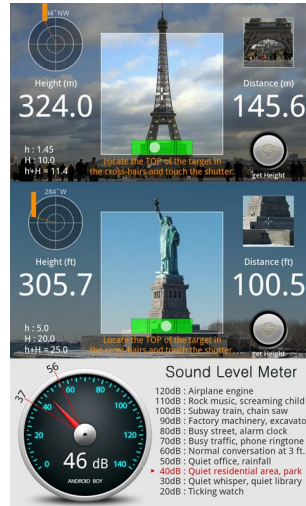


Figure 8.4: Smart Tools Screenshot



(a) Layar Screenshot 1

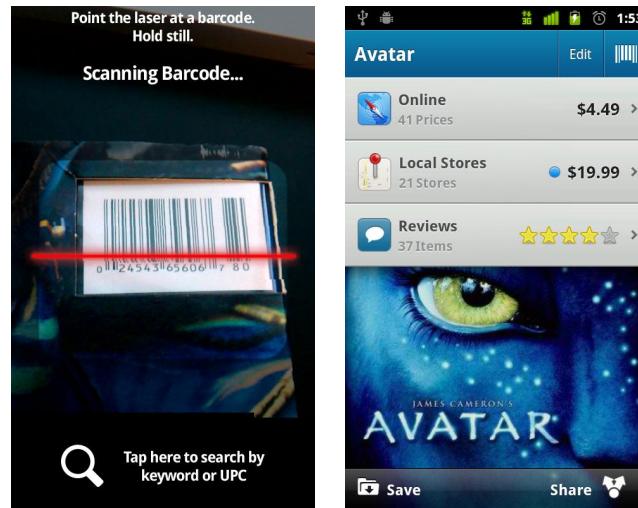
(b) Layar Screenshot 2

Figure 8.5: Layar Screenshots



## 8.5 ShopSavvy

ShopSavvy is an application for reading barcodes of products using the camera of the mobile phone. After reading the barcode, the application will identify the product and provide a list of online and local prices for it [68]. See Figure 8.6 for screenshots of Shopsavvy.



(a) Shopsavvy Screenshot 1 (b) Shopsavvy Screenshot 2

Figure 8.6: Shopsavvy Screenshots

## 8.6 Shazam

Shazam is an application for recognizing songs that are playing, for instance on the radio. The application listens to music snippets through the microphone on the phone, and creates a fingerprint of the snippet based on a spectrogram which it matches against fingerprints stored in a central database of music [6]. See Figure 8.7 for screenshots of Shazam.

## 8.7 Flashlight

There are several flashlight applications on the Android market. The core feature of the flashlight is to provide the user with bright light, like a flashlight. It basically fills the screen with a color of choice and manipulates the brightness

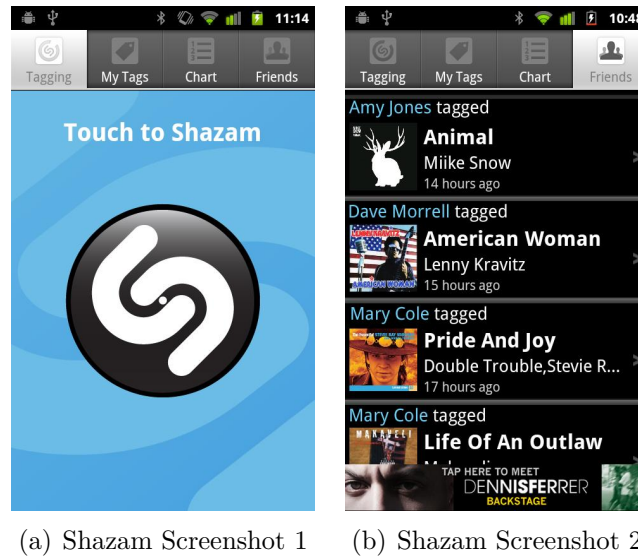


Figure 8.7: Shazam Screenshots

to get the screen as bright as possible, which enables the user to use the mobile device as a flashlight. See Figure 8.8(a) for a screenshot of a flashlight application.

## 8.8 Compass

There are several different compass applications on the Android market. The compass use the magnetometer (see Section 7.3.9) within the phone to measure the magnetic fields in order to find the direction. Some compasses also uses GPS to locate the user, but this feature is not required. See Figure 8.8(b) for a screenshot of a compass application.



Figure 8.8: Flashlight and Compass Screenshots



### Related Work

---

In this chapter we will go through the related work of this project. Three pervasive games and two relevant papers are presented. The first game is *The Amazing Race*, used as a source of inspiration on game format. The second game is *Shelby Logan's Run*, used as a reminder that ethical concerns have to be taken into account. The last game is *Heroes of Koskenniska*, which is very similar to the game we want to create. The first paper, *Game-Based Learning in Universities and Lifelong Learning*, is a paper about game-based learning that presents a framework for teachers to apply game-based learning to their classes. The second paper, *To Construct the Outdoor Experience Game-Based Learning System by Integrating Ubiquitous Technology*, is a paper describing a software architecture for a platform for creating mobile games.

#### 9.1 The Amazing Race

*The Amazing Race* is a reality television game show where a dozen teams of two people, that are in some form of personal relationship, are competing for a prize of one million dollars. The contestants are traveling to and within multiple countries by using various transportation means such as planes, bicycles, cabs, rental cars, trains, buses, boats, or by foot. Their final destination is the "pit stop". On their way to the "pit stop" they get cryptic clues on where to go next. They have to figure out where the clues lead to and how to get there. Finally, after a series of clues, they will arrive at the "pit stop". The last team to get to the "pit stop" is out of the race. Teams are eliminated until there are three left, and at that point, the first team to arrive at the final "pit stop" wins the race as well as the one million dollar prize.

## 9.2 Shelby Logan's Run

*Shelby Logan's Run* is the 2002 edition of *The Game* [14], a Seattle-based yearly puzzle hunt. It was one of the most advanced pervasive games of its time, with puzzles such as having to find a clue inside a living rat, and breaking into what looked like an abandoned prison. What sets this game apart from many other pervasive games is that it ended tragically, when a contestant misinterpreted a warning to not enter any other mineshafts than a specific one as a clue to do just that. This in turn led to him entering an unsecured mine and getting severely injured after falling 30 feet head first, and consequently suing the organizers of the game [13].

## 9.3 Heroes of Koskenniska

*Heroes of Koskenniska* [70] is a pervasive learning space (see Section 6.1.2) combining mobile and sensor technologies with environmental education. The game is based on the battle between Ukko and Hiisi, where the players are guided by Ukko through a variety of challenges or tasks. The game has three levels with several tasks on each level. At the end of each level Ukko takes the player to battle against Hiisi, where the player must solve tasks related to information acquired previously in the game. The game is implemented using a screen-based architecture, where each screen has a separate data structure (model), rendering mechanism (view), and controller; which means that it is easy to add new tasks to the game.

## 9.4 Game-Based Learning in Universities and Lifelong Learning

In [65] the authors explore the concept of game-based learning. They present a framework, called *UniGame*, which is used by teachers to apply game-based learning to their classes. *UniGame* is a framework that can be classified as a role-playing game that fosters participation in problem solving and the strengthening of social skills through, for instance, teamwork and effective communication. It consists of three modules. The first module is an information module that provides information about the game. The second module is the game module that handles user credentials and game choices. The third module is a community module which is where active players can exchange ideas and experiences.

---

## 9.5 To Construct the Outdoor Experience Game-Based Learning System by Integrating Ubiquitous Technology

In [84] the authors proposed a web-based outdoor experience game-based learning system. This system tries to integrate Geography Information System (GIS), Global Position System (GPS), and wireless connection technologies in a game inspired by treasure hunts and geocaching. The system consists of a backend server with a database that serves information to three modules. These modules consist of a course authoring tool accessed through a web browser, a mobile client on Ultra Mobile PCs, and a review module accessible by both teachers and students through a web browser.





# CHAPTER 10

---

## Trondheim

---

Trondheim was founded in 997 [15], and is the third largest city in Norway. The city is a regional capital as well as a cultural and educational center with a rich history and many historical locations. This chapter contains a brief description of some of these locations.

### 10.1 Torvet

Trondheim has had a town square and marketplace since the early 12th century. After the great fire in 1681, the marketplace is situated in the heart of Trondheim, at the intersection between Munkegata and Kongens gate. Both Munkholmen and the Nidaros Cathedral can be seen from the market square, which corresponds to the city planner Johan Caspar de Cicignon's city plan. In 1921, a statue of the city's founder, Olav Tryggvason, was raised in the middle of the square [77]. See Figure 10.1(a).

### 10.2 Munkegata

After the destructive city fire in 1681, the city planner Major General Johan Caspar de Cicignon made a new city plan. In the new city plan Munkegata was one of the main streets, and was placed at an angle to the rectangular downtown area so that it was aligned to highlight Munkholmen and Nidaros Cathedral [74]. See Figure 10.1(b).



(a) Torvet



(b) Munkegata

Figure 10.1: Torvet and Munkegata

### 10.3 Vår Frue Kirke

The oldest part of Vår Frue Kirke was erected in the 12th century. The first tower of the church was built around 1640, but the tower that stands there today was built in 1742. The spire was erected in 1779. On the old church wall, 800 year old Runes can be seen [78]. See Figure 10.2(a).

### 10.4 Nidarosdomen

Nidraos Cathedral (Nidarosdomen) was built over the grave of St. Olav, and construction started in 1070. Due to all the city fires in Trondheim, the oldest parts of the cathedral still in existence are from the 12th century. The cathedral has been rebuilt several times, but it is mainly in Gothic style [75]. To the east, Nidarosdomen points directly towards Kristiansten Festning. See Figure 10.2(b).



(a) Vår Frue Kirke



(b) Nidarosdomen

Figure 10.2: Vår Frue Kirke and Nidarosdomen

## 10.5 Erkebispegården

The Archbishop's Residence (Erkebispegården) is the oldest secular building in Scandinavia. The construction started in the second half of the 12th century, and it served as the Archbishop's residence until the Reformation in 1537. After the Reformation, the building served of the Danish governors. Later, around 1700, it was left to the military [?]. See Figure 10.1(a).

## 10.6 Stiftsgården

Stiftsgården was completed in 1778, and is one of the largest wooden buildings in the Nordic countries. It was originally built as a private residence for Cecilia Christine Schøller, but it was sold to the state in 1800. When Karl Johan was crowned in 1818, Stiftsgården served as the point of departure for the coronation procession to Nidaros Cathedral. This tradition continued with the crowning of the Swedish-Norwegian regents Karl XV and Oscar II. After the union was dissolved in 1906, and King Håkon VII was crowned, Stiftsgården became the official Royal Residence in Trondheim. The building itself got 140 rooms, and stretches 58 meters along Munkegata [76]. See Figure 10.3(b).



(a) Erkebispegården



(b) Stiftsgården

Figure 10.3: Erkebispegården and Stiftsgården

## 10.7 Bakklandet

Bakklandet is Trondheim's oldest suburb which dates back to the middle of the 17th century. It was partly destroyed when the Swedish beleaguered the city in

1658. After the Old Town Bridge was built in 1681, Bakklandet experienced an increase in traffic. In addition, after the fire regulation in 1689, ships that had open fires as well as flammable cargo could not make port on the city side of the river. This led to an increase in commerce at Bakklandet. In 1718, Bakklandet was burnt down again, but this time by Norwegian soldiers in order to defend the city of Trondheim. After the attack of Carl Gustaf Armfelt, Bakklandet was quickly rebuilt, and consisted of 77 farmhouses. In 1846, Bakklandet officially became a part of Trondheim [73]. See Figure 10.1(a).

## 10.8 Munkholmen

According to Icelandic historical record, a cloister was founded on Munkholmen around 1105, or Nidarholmen which it was called at the time. However, the monastery was laid in ruins before the Reformation. Before the war in 1658, a fortress was planned here, but the building process did not start until the war ended. The construction on the Munkholmen fortifications continued up to 1661. When the fort was finished, the deep vaults and cold cell floors were deemed suitable to hold rejects of society. In 1671 Chancellor of the Realm Peder Schumacher Griffenfeld was put in one of the cells, and was not released until 18 years later. This inspired Victor Hugo to write "The Prisoner of Munkholmen", and perhaps "Les Miserables". In 1893 the fortress was officially closed [74]. See Figure 10.4(b).



(a) Bakklandet



(b) Munkholmen

Figure 10.4: Bakklandet and Munkholmen

## 10.9 Ravnkloa

The name, Ravnkloa, is an old name and sources date it back to 1619 when they mention Peder Raffnklau. However, the name was not used about Ravnkloa until the start of the 18th century. Ravnkloa was only used as a fish market up until 1841, when it was transformed to a more general market. As a result of this, the market trade flourished. In 1896, Trondhjem Fiskeriselskab built a fish market at Munkegata 70 in the end of Ravnkloa. It was moved indoors in a temporary fish hall 1945. In 1960, the fish hall was replaced with a more permanent solution. Ravnkloklokka, which is a big clock, was erected at Ravnkloa in 1904. It was built by Endre Røe Syrstad, and it also included a barometer and a thermometer [25]. See Figure 10.5(a).

## 10.10 Kristiansten Festning

Kristiansten Fortress (Kristiansten Festning) lies just above Bakklandet, and is built on the strategically important height, Erlendshaug. The reasoning behind this is that it was possible to fire at the city from this position towards the end of the 17th century. Just below the fortress lies the most important (at that time) road into the city from both the east and the north. There were initially plans made for a fortress at Erlendshaug already in 1676, but only after the great city fire in 1681 were the plans set in motion. The first stone was set in 1682. The fortress was constructed as a part of the reconstruction effort after the fire in 1681, and had an important role to play in the city's fortification plan designed by Cignon and Coucheron. In 1816 Trondheim lost its status as a fortress city and the fortress was decommissioned. It was decided that the fortress should be abandoned completely and left to decay. However, due to the cannon battery still having a role to play (as a salute battery and a warning signal to the city's inhabitants that fire had broken out), the decision was reversed and the fortress was maintained [25]. See Figure 10.5(b).

## 10.11 Gamle Bybro

The Old Town Bridge was originally built around 1683-1685 as a result of Cignon's city plan. It was planned to cross the river further south, but to increase the defensive capabilities of the city it was built where it stands today. This location would give the city adequate defense if it was attacked from Vollan and Øvre Bakklandet. A new town bridge was built in 1861 with the design of Carl



(a) Ravnkloa



(b) Kristiansten Festning

Figure 10.5: Ravnkloa and Kristiansten Festning

Adolf Dahl. The bridge is 82 meters long and 6 meters wide. The bridge also got two portals which are called "Lykkens portal" [25]. See Figure 10.6(a).

## 10.12 Britannia Hotel

Britannia Hotel is a five star hotel that was built in 1870, making it one of the oldest hotels in Trondheim. Amongst its facilities are Palmehaven, restaurant and banquet with an Arabic garden and a fountain [83]. The fountain is the location of an annual ceremony where graduating students from NTNU release a living pike into the fountain to celebrate the memory of Jonathan the Pike who according to legend was executed during the second world war [26]. See Figure 10.6(b).



(a) Gamle Bybro



(b) Britannia Hotel

Figure 10.6: Gamle Bybro and Britannia Hotel

# Part III

## Design





# CHAPTER 11

---

## Introduction

---

This part contains design information about the prototype. This includes the following chapters: Experiment, Tasks, Requirements, User Interface, and Software Architecture. The Experiment chapter describes the motivation, inspiration, game plot, and ethical concerns of the game. The Tasks chapter contains information about the task types of the game, which also includes general concrete steps of how the task is solved. The Requirements chapter presents the requirements of the game. The User Interface chapter contains some general design ideas of the game, which includes a number of mock-up figures. In the last chapter, Software Architecture, the software architecture of the game is presented. This includes a data integration architecture diagram, a state diagram of the task progression, a data flow diagram, a package diagram of the packages of the prototype, a class diagram, and an entity-relationship diagram.



# CHAPTER 12

---

## Experiment

---

As our experiment we have chosen an amazing race through the historical parts of Trondheim, a game we call *The Amazing City Game*. The race will start at NTNU Gløshaugen, proceed through Kristiansten Fortress and into Midtbyen, before we round things up at Ravnkloa. The following sections describes the Motivation, Inspiration, Game Plot, and Ethical Concerns of the experiment.

### 12.1 Motivation

The motivation behind choosing this exact experiment is that it gives room for testing a lot of different technology demanding scenarios. By getting contestants to travel to several different locations we can thoroughly test the GPS-unit, and how Internet connectivity through Wi-Fi hotspots and 3G work. By having a huge environment we also have the ability to test a multitude of other technologies through different tasks, such as for instance barcode scanning.

### 12.2 Inspiration

There are two main inspirations for the game. The first is the American television series *The Amazing Race*, a reality show where contestants compete to be the first to reach different checkpoints all over the world. The other is the treasure hunt called *The Game*, specifically the interesting and failed version of *The Game* called *Shelby Logan's Run*.

### 12.3 Game Plot

The game is an adventure game where the contestants will have to solve different tasks at different locations. The group that reaches the final destination in the least amount of time is the winner.

The game begins at a specific location. The contestants will carry with them fully charged Android mobile phones with the game prototype containing all game content preinstalled. The game starts with the game officials revealing the destination of the first task, where they will be handed a unique code which determine in which order the tasks should be solved (with the routes predefined by the game officials). The tasks will be of different character, but they are split into groups, where the first task of a group involves finding a specific location, and the subsequent tasks are solved at that location. Should the groups struggle with finding either the locations or solutions of the tasks, hints may be given. However, using hints will incur time penalties, as will incorrect attempts at solutions. When the last task is solved (all routes ends at the same place) the total time spent including penalties is calculated and the winner is determined.

### 12.4 Ethical Concerns

There are two noticeable ethical concerns in this prototype game. The first is at Kristiansten Festning, where the contestant have to traverse walkways on the fortress walls. These walkways are unsecured, and at some points high altitude. When groups are competing at something that require quickness they might lose concentration. This may additionally be intensified if two or more groups are at the same place in the course.

The second is that the game is taking place in a trafficated area. Again, having to move quickly might lead to less concentration, and in addition one might lose focus if staring at the mobile phone screen while moving.

We do not consider these risks to be major, and we have mitigated these risks by giving clear warnings in the contestants pre-game information document, as well as warnings in the task descriptions at Kristiansten Festning, and also by stating in the rules that one has to follow the traffic rules.

# CHAPTER 13

---

## Tasks

---

The game has three main goals:

1. To test different technologies on the Android mobile operating system.
2. To give the contestant knowledge about the history of Trondheim.
3. To let the contestants have fun while playing the game.

With these goals in mind we have constructed the following types of tasks for *The Amazing City Game*. The tasks themselves can be found in Appendix A.

### 13.1 Location Task

The user will have to find a specific location and confirm it with the use of GPS. The concrete steps are as follows:

1. Read the description text and find out which location one has to travel to.
2. Travel to the location as quick as possible.
3. If needed, press the "Hint" button in order to receive further hints about the location the user has to find. Using these hints will incur a time penalty.
4. When the user is at the location he or she thinks is correct, press the "GPS" button. This will make the mobile phone's GPS unit determine which location the user is at.
5. If the location is sufficiently near the correct location (determined by if the user is within a defined range) the task is solved and the next task is presented. If the user is at the wrong location he will get a time penalty and has to try again.

See Figure B.1 in Appendix B for a use case of Location Task.

## 13.2 Barcode Task

The user will have to scan a barcode printed on a sheet of paper in order to get assigned a route. The concrete steps are as follows.

1. Read the description text and find as quickly as possible a barcode printed on a sheet of paper.
2. Press the 'Scan'-button in order to invoke the external application 'Barcode scanner'.
3. Scan the barcode.
4. If the scanned keyword corresponds to a route, the task is solved and the next task is presented. If the scanned keyword does not correspond to a route the user will have to try again.

See Figure B.2 in Appendix B for a use case of Barcode Task.

## 13.3 Open Task

The user will be given a question and will have to type answer into the answer text box. The concrete steps are as follows:

1. Read the description text and find out the solution to the given question.
2. The user might use external applications, such as a compass application or an web browser in order to find the answer.
3. If needed, press the 'Hint' button in order to receive further hints. Using these hints will incur a time penalty.
4. When the user has found the answer he or she must insert it into a text box and click the "OK" button.
5. If the answer is correct, the task is solved and the next task is presented. If the answer is wrong, the user will get a time penalty and has to try again.

See Figure B.3 in Appendix B for a use case of Open Task.

## 13.4 Multiple Choice Task

The user will be given a question and will have to select the right answer out of the possible solutions. The concrete steps are as follows:

1. Read the description text and find out the solution to the given question.
2. If needed, press the "Hint" button in order to receive further hints. Using these hints will incur a time penalty.
3. When the user has found the answer he or she must select the correct alternative (each alternative has its own radio button) and click the "OK" button.
4. If the answer is correct, the task is solved and the next task is presented. If the answer is wrong, the user will get a time penalty and has to try again.

See Figure B.4 in Appendix B for a use case of Multiple Choice Task.

## 13.5 Checkbox Task

The user will be given a question and will have to select the right answer out of the possible solutions, where multiple answers might be correct. The concrete steps are as follows:

1. Read the description text and find out the solution to the given question.
2. If needed, press the "Hint" button in order to receive further hints. Using these hints will incur a time penalty.
3. When the user has found the answer he or she must select the correct alternatives (each alternative has its own checkbox) and click the "OK" button.
4. If the answer is correct, the task is solved and the next task is presented. If the answer is wrong, the user will get a time penalty and has to try again.

See Figure B.5 in Appendix B for a use case of Checkbox Task.

## 13.6 Shazam Challenge

The user will be given a question and will have to type answer into the answer text box. The difference between this task and the Open Task is that the user is

given and audio clue in the task description, and additional audio clues for each hint he requests. The concrete steps are as follows:

1. Read the description text and find out the solution to the given question.
2. The user might use the external application Shazam in order to find out which song the audio clue contains.
3. If needed, press the "Hint" button in order to receive further hints with an additional audio clue. Using these hints will incur a time penalty.
4. When the user has found the answer he or she must insert it into a text box and click the "OK" button.
5. If the answer is correct, the task is solved and the next task is presented. If the answer is wrong, the user will get a time penalty and has to try again.

See Figure B.6 in Appendix B for a use case of Shazam Challenge.

## 13.7 Shopsavvy Challenge

The user is given a series of multiple choice questions. The difference from other tasks is that the alternatives for each question is written on a sheet of paper at the location, where each answer is next to a barcode from a commercial product. By taking the first letter of the corresponding product of the barcode paired with the correct answers the users get a handful of letters that, when put in correct order, will give a codeword that is typed into a text box. The concrete steps are as follows:

1. Read the description text and for each question:
  - (a) Find the correct answer amongst the given alternatives on the barcode sheet.
  - (b) If needed, press the "Hint" button in order to receive further hints. Using these hints will incur a time penalty.
  - (c) Scan the corresponding barcode by using the application ShopSavvy to determine which product the barcode belongs to.
  - (d) Take the first letter of the product name and store it until all questions are answered.
2. Using all the letters, interchange them so that they form a codeword.
3. When the user has found the codeword he or she must insert it into a text box and click the "OK" button.



4. If the answer is correct, the task is solved and the next task is presented. If the answer is wrong, the user will get a time penalty and has to try again.

See Figure B.7 in Appendix B for a use case of Shopsavvy Challenge. For an example of a barcode sheet used in this task type see Figure 13.1.

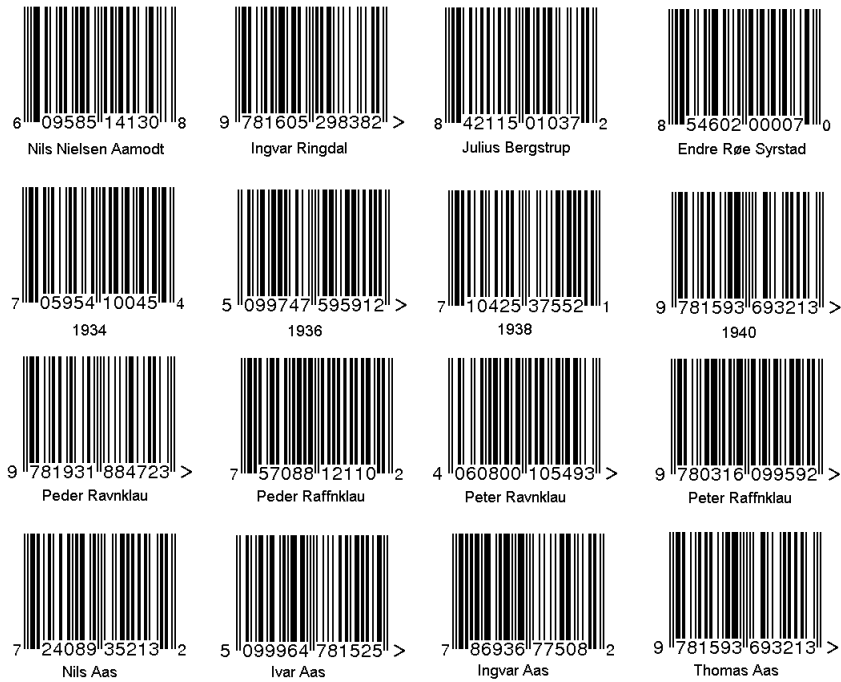


Figure 13.1: Shopsavvy Barcodes



# CHAPTER 14

---

## Requirements

---

This chapter contains the requirements of the prototype. This includes both functional and non-functional requirements.

### 14.1 Functional Requirements

This sections contains the functional requirements of the prototype. The requirements are divided into seven categories: task, hint, answer validation, GPS, barcode scanning, game state, and misc.

#### 14.1.1 Task

ID	Description	Priority
FR-TSK-01	Multiple choice: The user selects an answer to a question from several alternatives. See Figure 15.1(a).	High
FR-TSK-02	Checkbox: The user selects one or more alternatives to answer a question. See Figure 15.1(b).	High
FR-TSK-03	Open: The user types in an answer to a question. See Figure 15.2(a).	High
FR-TSK-04	Shazam: The user will be given a question and will have to answer it based on the songs provided. The answer will be given in a text box. See Figure 15.2(b)	High

*Table 14.1 – continued on next page*

*Table 14.1 – continued from previous page*

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-TSK-05	GPS: The user must get within range of a set of GPS-coordinates, and ask for a location update. See Figure 15.3(a).	High
FR-TSK-06	Barcode: The user must scan a QR-code of a poster. See Figure 15.3(a).	High
FR-TSK-07	Task descriptions shall be able to contain an image.	Low

Table 14.1: Functional Task Requirements

### 14.1.2 Hint

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-HNT-01	The user shall be able to get hints for all tasks.	Medium
FR-HNT-02	A confirmation box shall be displayed when the user asks for a hint. See Figure 15.3(b).	Low
FR-HNT-03	The user shall get a time penalty when asking for a hint.	Medium
FR-HNT-04	Hints shall be able to contain an image.	Low
FR-HNT-05	Hints shall be able to contain a song.	Medium

Table 14.2: Functional Hint Requirements

### 14.1.3 Answer Validation

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-VAL-01	A confirmation box shall be displayed when the user wants to submit an answer. Similar to the confirmation box for hints, see Figure 15.3(b).	Low
FR-VAL-02	Correct answers shall lead to the next task as defined in the route.	High
FR-VAL-03	Wrong answer shall result in a toast that will inform the user that the answer is wrong or incomplete. Will result in a time penalty.	Low

*Table 14.3 – continued on next page*

*Table 14.3 – continued from previous page*

ID	Description	Priority
FR-VAL-04	Wrong answer shall result in a time penalty.	Medium

Table 14.3: Functional Answer Validation Requirements

#### 14.1.4 GPS

ID	Description	Priority
FR-GPS-01	The application shall be able to use GPS so that tasks where the contestant will have to move within the range of a set of GPS-coordinates is possible.	High
FR-GPS-02	The application shall be able to check the GPS location when the user ask for it.	High

Table 14.4: Functional GPS Requirements

#### 14.1.5 Barcode Scanning

ID	Description	Priority
FR-BCS-01	Application shall integrate barcode scanning so that barcode scanning tasks are possible.	High

Table 14.5: Functional Barcode Scanning Requirements

#### 14.1.6 Game State

ID	Description	Priority
FR-GST-01	The current route shall be retrieved from the database upon game start.	High
FR-GST-02	The current task shall be retrieved from the database upon game start.	High
FR-GST-03	Used hints shall be displayed when the current task is loaded.	Medium

*Table 14.6 – continued on next page*

*Table 14.6 – continued from previous page*

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-GST-04	The game state shall be updated in the database during game play.	High

Table 14.6: Functional Game State Requirements

### 14.1.7 Misc

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-MSC-01	The application shall display a screen when the game finishes that shows total penalty accumulated.	Medium
FR-MSC-02	The user shall be able to display a screen to check time used and penalties.	Low
FR-MSC-03	The application shall display a welcome screen on startup.	Low
FR-MSC-04	The welcome screen shall contain a button for starting the next task.	Low
FR-MSC-05	The application shall be able to reset the database.	Low
FR-MSC-06	The application shall have a failsafe mechanism so that unsolvable task can be solved.	Medium
FR-MSC-07	The failsafe mechanism shall reset any time penalty the user has been given for the unsolvable task.	Medium

Table 14.7: Functional Misc Requirements

## 14.2 Non-functional Requirements

This section contains the non-functional requirements of the prototype.

<b>ID</b>	<b>Description</b>	<b>Priority</b>
NFR-01	The application shall use Wi-Fi whenever possible to save cost.	Medium

*Table 14.8 – continued on next page*

*Table 14.8 – continued from previous page*

<b>ID</b>	<b>Description</b>	<b>Priority</b>
NFR-02	GPS shall do as few location updates as possible to save battery.	Medium
NFR-03	Wi-Fi shall be used as little as possible to save battery.	Medium
NFR-04	Application shall encourage lifelong-learning.	High
NFR-05	Application shall have high performance and drain little battery.	Medium

Table 14.8: Non-functional Requirements



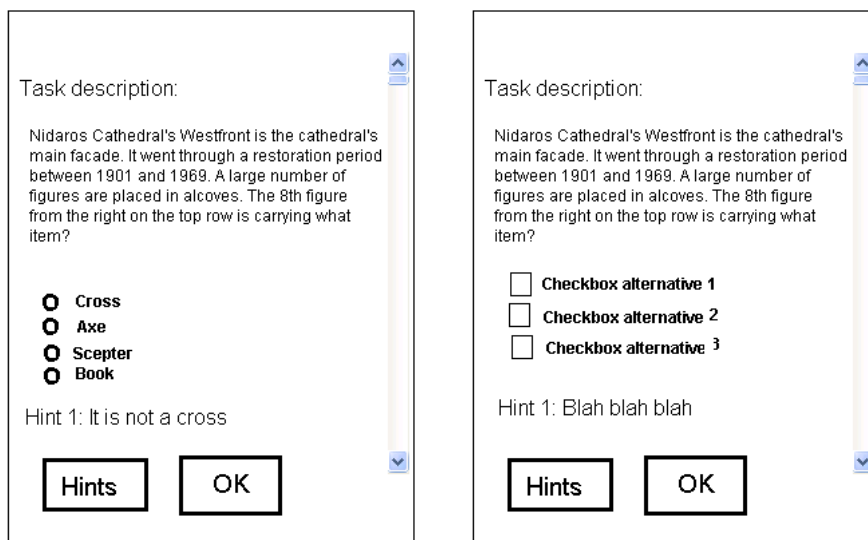


---

## User Interface

---

This chapter contains the user interface design of the game. The user interface will be clean and simple, hence all unnecessary elements will be removed. For examples of the user interface design see Figure 15.1, 15.2, and 15.3.



(a) UI: Multiple Choice

(b) UI: Checkbox

Figure 15.1: User Interface: Multiple Choice and Checkbox Design

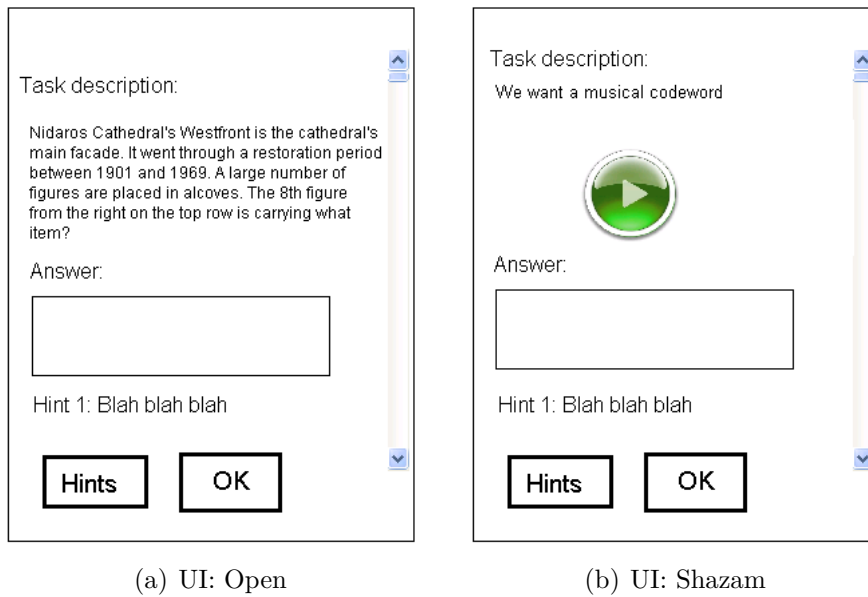


Figure 15.2: User Interface: Open and Shazam Design

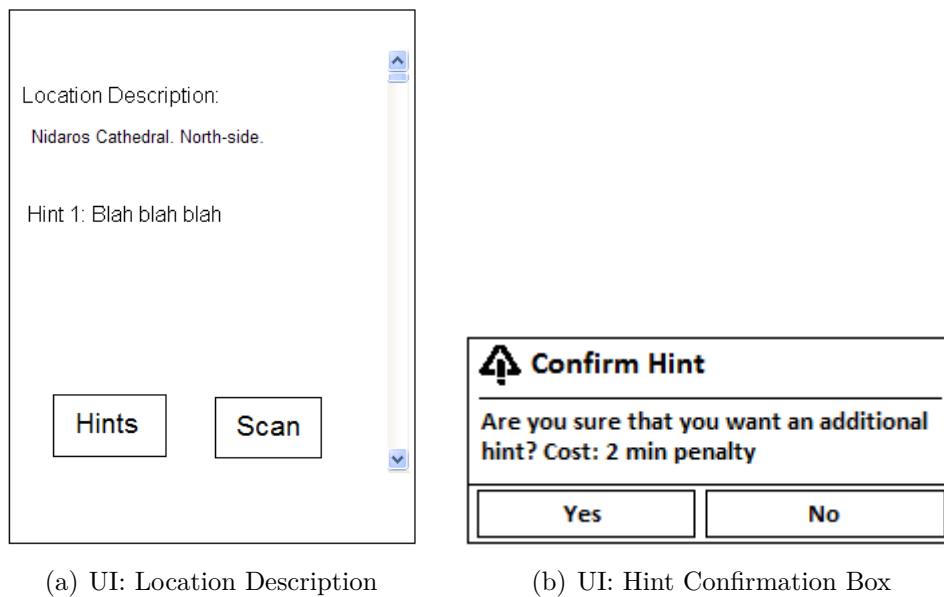


Figure 15.3: User Interface: Location Description and Hint Confirmation Design

# CHAPTER 16

---

## Software Architecture

---

This chapter describes the software architecture used in the prototype. We will first go through Architectural Choices before proceeding with presenting the architecture. The software architecture is represented using data integration architecture diagram, state diagram, data flow diagram, package diagram, class diagram, and entity-relationship diagram.

### 16.1 Architectural Choices

This section discusses the reasoning behind our major architectural choices.

#### 16.1.1 Server Solution

Initially we considered a client-server architecture for the prototype where content was located at the server and sent to clients on request. This idea was ultimately scrapped in favour of an architecture that does not include any server, meaning the whole system is on the user's device. There are several reasons for this. Most important of these, the purpose of the prototype is to test the game concept. This needs to be done relatively fast so that we have the necessary time to evaluate the prototype and create a platform design based on the results. Implementing a server would unnecessarily complicate the design process and require more time spent in implementation on an architecture that does not really yield any substantial benefits at this stage in the project. Another important reason is that the wireless Internet coverage in the Trondheim center is not as extensive and reliable as we would wish. By placing all the content in the application we can save time, money, battery, and avoid problems like connection issues.

### 16.1.2 Storage Solution

We evaluated two possible solutions for storage of game data. The alternatives were using a file or using an integrated database. We decided on using a database, because the support for databases in Android is very good, with simple interfaces for retrieving and storing data. Android offers an easy way to create, edit and manage a `SQLiteDatabase` with the class `SQLiteDatabase`. Also, each database created and its content is visible only to its parent application. A database combined with frequent saving of game data will ensure that data will not be lost even if the application were to terminate unexpectedly.

### 16.1.3 Activity Structure

Initially we planned on using one Android Activity for the entire application, and switching the user interface depending on the context. The idea was scrapped when we faced challenges with adding and removing user interface components, and additionally we reviewed the Android guidelines, which recommended usage of different activities for different screens.

## 16.2 Data Integration Architecture Diagram

For a data integration architecture diagram, see Figure 16.1.

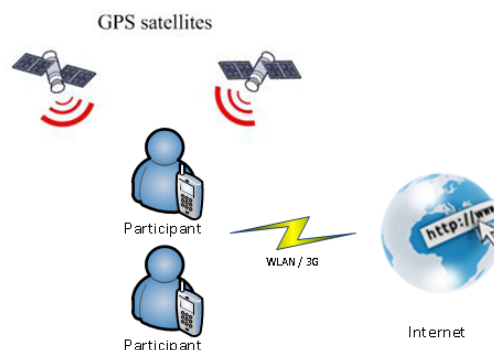


Figure 16.1: Data Integration Architecture

## 16.3 State Diagram

For a state diagram illustrating task progression see Figure 16.2.

## 16.4 Data Flow Diagram

For a data flow diagram (DFD) see Figure 16.3.

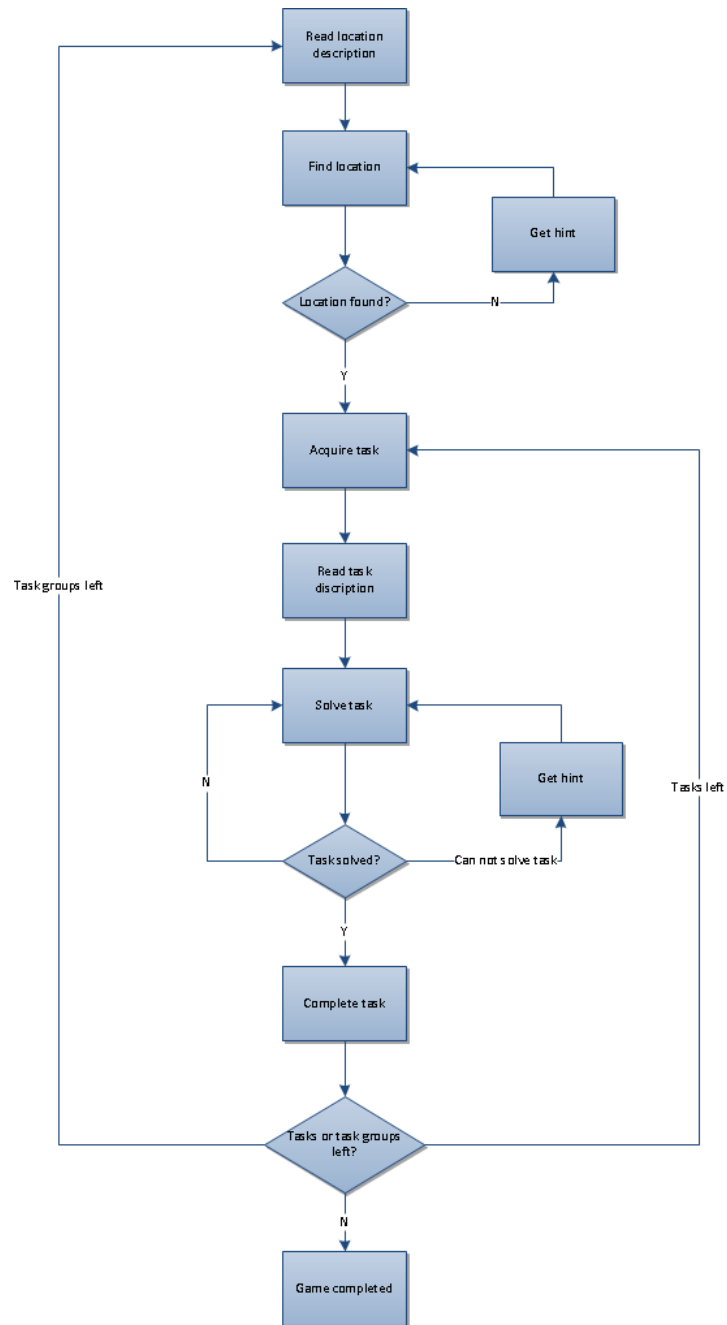


Figure 16.2: State Diagram

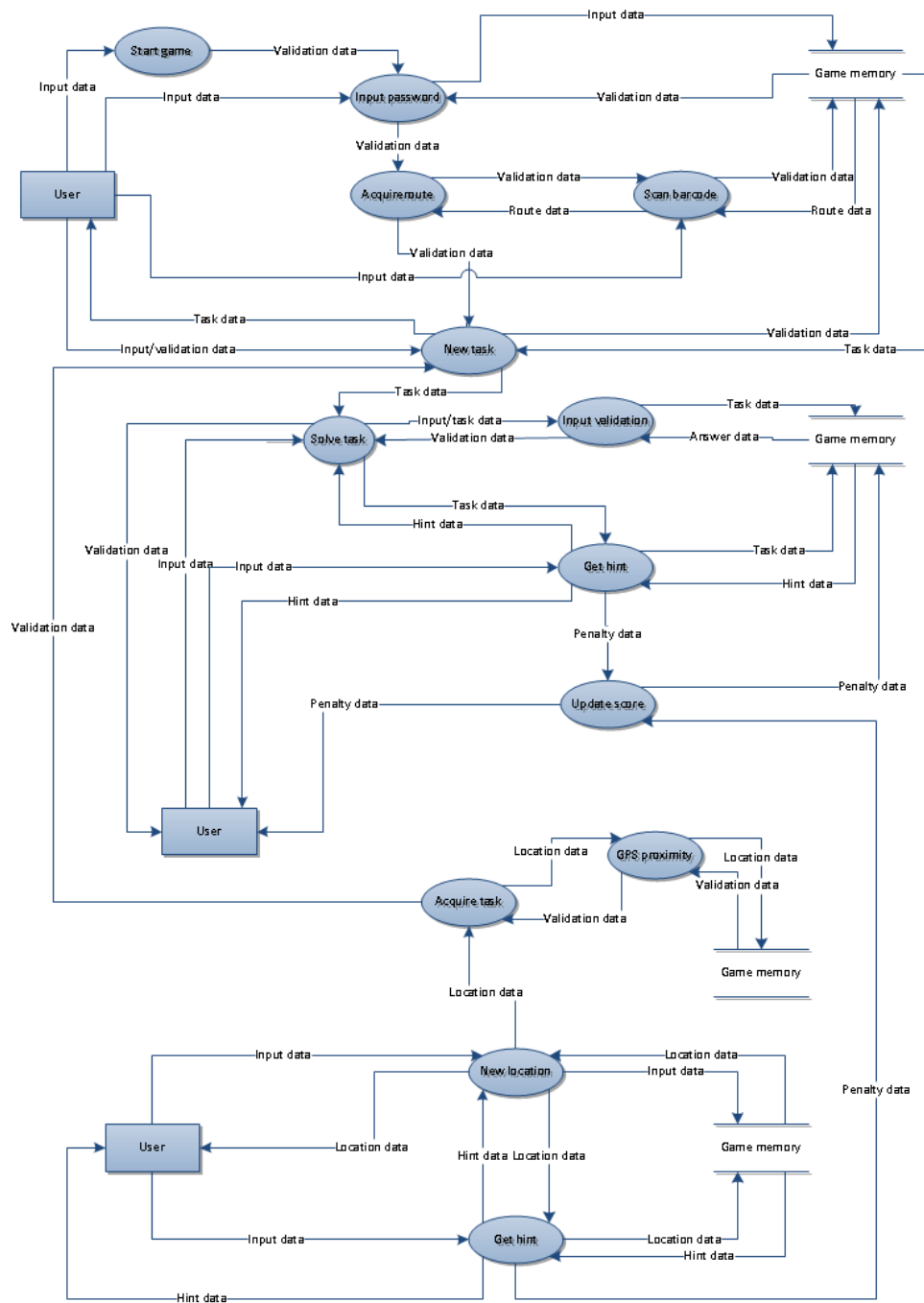


Figure 16.3: Data Flow Diagram

## 16.5 Package Diagram

The prototype has five different packages, where four of the packages are sub-packages of the root package `acg.prototype`. The root package contains the "viewable" classes of the prototype. By this we mean that the classes here are the classes that can show content to the user. However, note that these classes are not user interfaces, rather they can have user interfaces assigned to them.

The `acg.prototype.db` package contains the classes that handle the database and all interaction with it.

The `acg.prototype.content` package contains the classes that have to do with representing and manipulating content elements.

The `acg.prototype.ui` package contains the classes that are needed to represent the game elements to the player.

The `acg.prototype.util` package contains classes that provide GPS support and general support methods to the rest of the system.

For a package diagram showing the packages of the prototype see Figure 16.4.

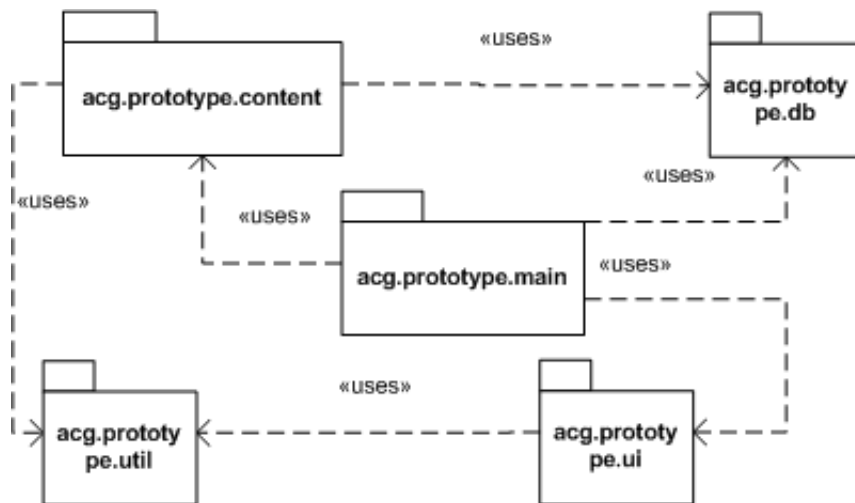


Figure 16.4: Package Diagram



## 16.6 Class Diagram

For a simplified class diagram containing only the most important fields and methods of the classes see Figure 16.5. For a detailed class diagram see Appendix C.

The following sections explain a little more about the most notable classes and some of their relationships with other classes, sorted by their parent package.

### 16.6.1 Root Package

**Main** is the starting point of the application. It initializes the application, sets the game variables and decides what comes next. It also has a user interface representation in the form of a welcome screen. This usually consists of deciding what task is next and starting **TaskActivity** to handle it.

**TaskActivity** is another of the "viewable" classes. It initializes the task and the appropriate user interface, and shows it to the player.

### 16.6.2 DB Package

**DbAdapter** is the class that other classes may use to interact with the database. This includes retrieval and editing of it's content.

**DbHelper** constructs the database and fills it with content if it does not already exist.

### 16.6.3 Content Package

**Task** is an abstract class that all task types, each with their own class, have to extend. It provides some behaviour that is common to all task types, as well as task-specific behaviour that has to be implemented by the subclasses. The task-specific behaviour concerns how answer are to be validated.

**Hint** contains the implementation of hints that the players may utilize.

**Route** handles the implementation of routes (lists of all tasks in varying order), as well as some methods to find the next task.

#### 16.6.4 UI Package

**TaskUI** is another abstract class. This one is extended by all task user interface classes. Each task type class has a corresponding user interface class that handles its representation. **TaskUI** defines a common structure for all of these task interfaces. It also provides a number of methods to create interface elements that the subclasses may use if so required by their corresponding task type.

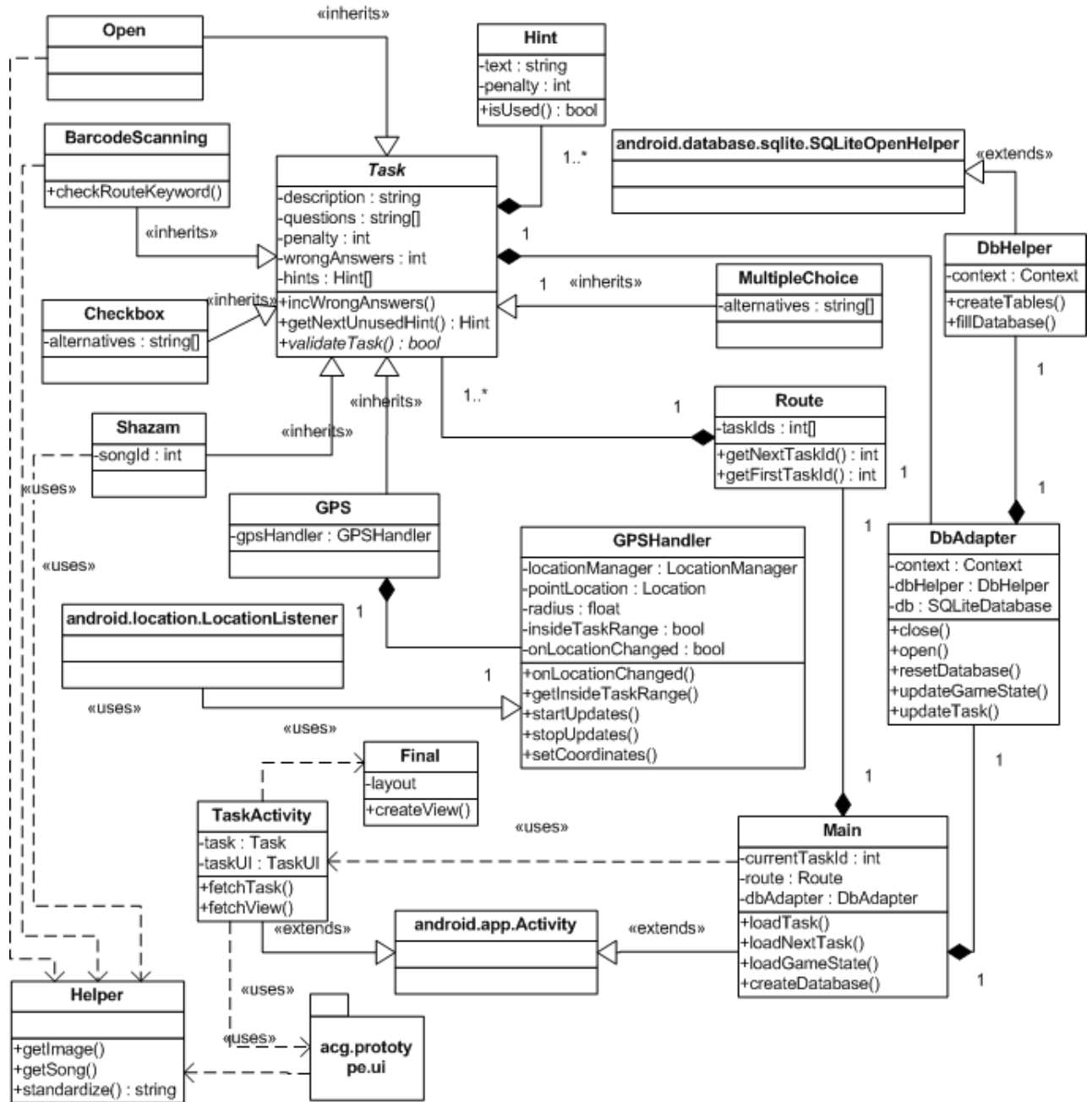


Figure 16.5: Simplified Class Diagram

## 16.7 Entity-Relationship Diagram

For a ER-diagram see Figure 16.6. In this prototype we use a simplified ER-diagram with just a few tables. The `Task` table contains the information to the general tasks. The `Alternative` table contains the alternatives for the Open and Checkbox tasks. The `Question` table holds the questions for the tasks. There can be any number of questions for each task. The `GPS` table contains the `latitude`, `longitude`, and `radius` for the Location task (called GPS task in the prototype). The `Shazam` table contains an identifier for a song which is to be used in the Shazam task. The `Hint` table contains all the information related to hints. The `GameState` table holds the information about the current game state, that is the `current_task_id` and `route_id`. The `Route` table contain all the routes, and each route has a `keyword` for it to be unlocked. The `RouteElement` table contains all the tasks of the route.

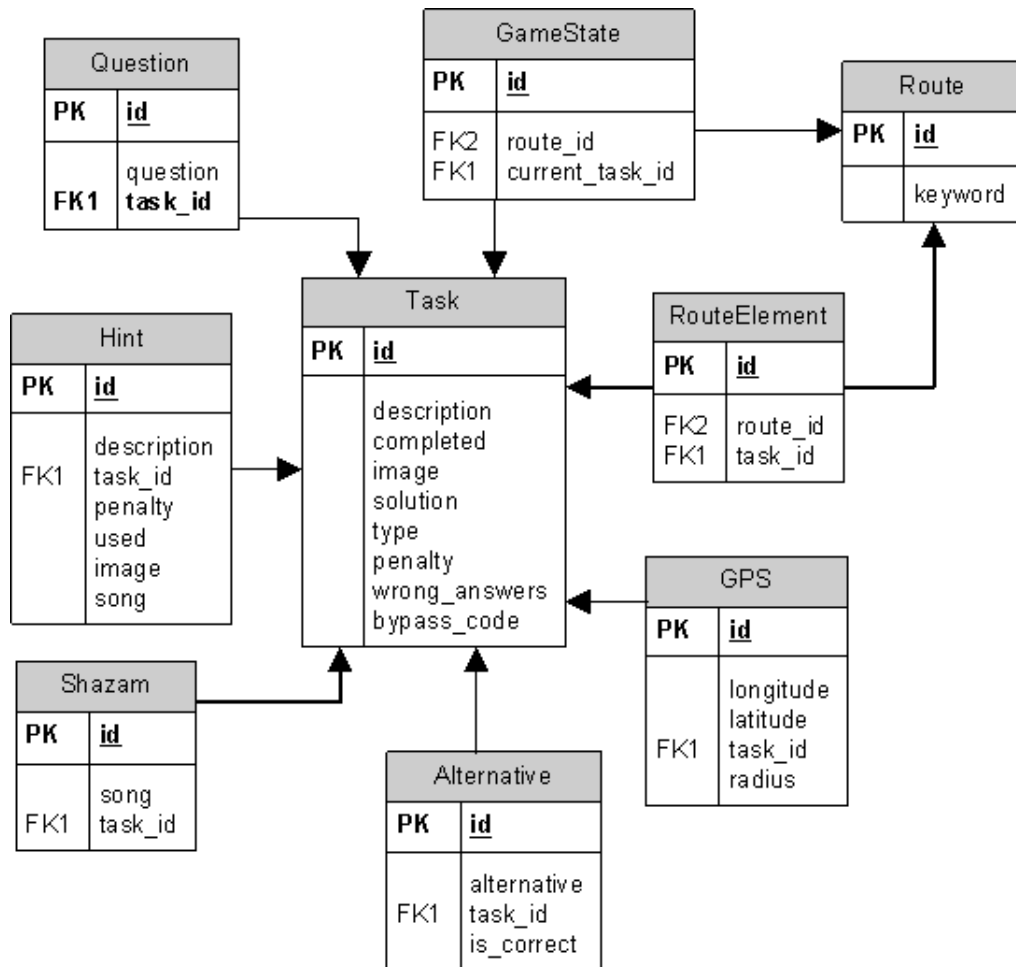


Figure 16.6: Entity-Relationship Diagram

**Part IV**

**Implementation**



# CHAPTER 17

---

## Introduction

---

This part contains the implementation phase of the prototype. The implementation was done by using an iterative approach. We separated the implementation into three sprints consisting of different parts of the prototype, hence the chapters for this part are: Sprint 1, Sprint 2, Sprint 3, and Prototype Finalization. Each sprint is divided into 5 sections: requirements, design, implementation, testing, and evaluation. The Prototype Finalization chapter contains final adjustments to the prototype.





# CHAPTER 18

---

## Sprint 1

---

### 18.1 Requirements

The requirements for this sprint are:

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-TSK-01	Multiple choice: The user selects an answer to a question from several alternatives.	High
FR-TSK-03	Open: The user types in an answer to a question.	High
FR-VAL-01	A confirmation box shall be displayed when the user wants to submit an answer.	Low
FR-VAL-02	Correct answers shall lead to the next task as defined in the route.	High
FR-VAL-03	Wrong answer shall result in a toast that will inform the user that the answer is wrong or incomplete. Will result in a time penalty.	Low
FR-VAL-04	Wrong answer shall result in a time penalty.	Medium
FR-GST-01	The current route shall be retrieved from the database upon game start.	High
FR-GST-02	The current task shall be retrieved from the database upon game start.	High
FR-GST-04	The game state shall be updated in the database during game play.	High
FR-MS-03	The application shall display a welcome screen on startup.	Low

*Table 18.1 – continued on next page*

*Table 18.1 – continued from previous page*

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-MSC-04	The welcome screen shall contain a button for starting the next task.	Low

Table 18.1: Sprint 1 Requirements

## 18.2 Design

### 18.2.1 Game Start

When the game is installed, the database is created and the game content is filled in. Upon game start the game retrieves the current game state, which is the current task as well as the route. If there is no current game state, a new route is retrieved from the database, and the current task is set to the first task in the route.

### 18.2.2 Task Flow

The first user interface screen the user encounter a welcome screen containing a welcome text and a button for getting to the next task. When the user pushes the next task button, the next task in the route is retrieved from the database. Upon completing the task, the application automatically retrieves the next task. If the user exits the game, the user will again encounter the welcome screen, and the route and task is retrieved from the database. If the user pushes the next task button, the application will load the task that the user was on when he or she killed the application.

### 18.2.3 Task User Interface

In this sprint we are implementing the Open task (FR-TSK-03) and the Multiple Choice task (FR-TSK-01). The interface for these two task types will require:

- A task description, where information about the task is presented along with the question to be answered.
- A button the user can press to validate his or her answer. When pressed a confirmation dialogue will appear where the user can choose to either proceed or revise the answer.

- A custom interface component according to task type.
- The possibility to add an image to the task description. (Sprint 2)
- A hint interface that will give the user the possibility to request hints. (Sprint 2)

Task specific components:

- Open: A field where the user can type in an answer.
- Multiple choice: A number of radio buttons the user can select an answer from.

## 18.3 Implementation

### 18.3.1 Game Start

The game starts in the class `Main`. First off the database creation is forced by creating a new `DbAdapter` then using its methods `open()` and `close()` (see Section 18.3.4). The database is created only at first installation of the game.

When the user starts the game, the game state is loaded by calling the the `loadGameState()` method. This method uses the `DbAdapter` methods `getGameStateRoute()` and `getGameStateTaskId()` to retrieve the current game state. If there is no current game state, which means that `routeId` and `currentTaskId` are 0, then a new game state is created. The new game state contains a new route and first task id of the route. If a new game state is created, the game state is stored in the database using `DbAdapter.updateGameState()`.

Finally the welcome screen is created. The welcome screen is very simple and contains a `LinearLayout`, `TextView` and a `Button` (see Figure 18.1).

### 18.3.2 Task Flow

When the user pushes the button on the welcome screen, the next task is loaded through the `Main.loadTask()` method. The `loadTask()` method creates a new `Intent` with the class `TaskActivity` and `currentTaskId` parameters. It then starts the activity.

When `TaskActivity` is loaded, `TaskActivity` first calls the the method `fetchTask()`. The `fetchTask()` method retrieves the `currentTaskId` from the `Intent` and then calls the method `DbAdapter.getTask()` to retrieve the task

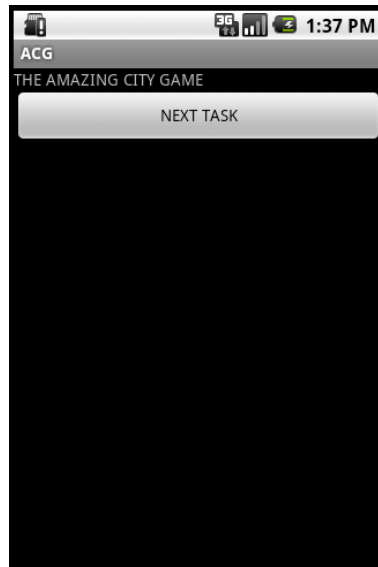


Figure 18.1: User Interface: Welcome Screen Screenshot

from the database. `TaskActivity` then uses `fetchView()` to retrieve the UI according to the task type of the current task. It then creates the view by calling `TaskUI.createView()`.

The user completes a task by finding the right answer and then pressing "OK". The `TaskUI.solveTask()` method is then called which set the task as complete in the database and finishing the activity. The application then returns to `Main`. `Main` then calls the method `loadNextTask()`. The `loadNextTask()` method first calls `getNextTaskId()` on route to retrieve the `taskId` for the next task, which is the new value of `currentTaskId`. It then calls `DbAdapter.updateGameState()` to update the game state. Finally it calls `loadTask()`, and completes the cycle.

For a state diagram of the task flow see Section 16.3. For a sequence diagram of the task flow see Figure 18.2

### 18.3.3 Task User Interface and Answer Validation

When the activity `TaskActivity` calls its `fetchView()` method, it initializes the appropriate type of task UI according to what type the active task is.

The task UI classes for this sprint are:

- `OpenTaskUI`
- `MultipleChoiceTaskUI`

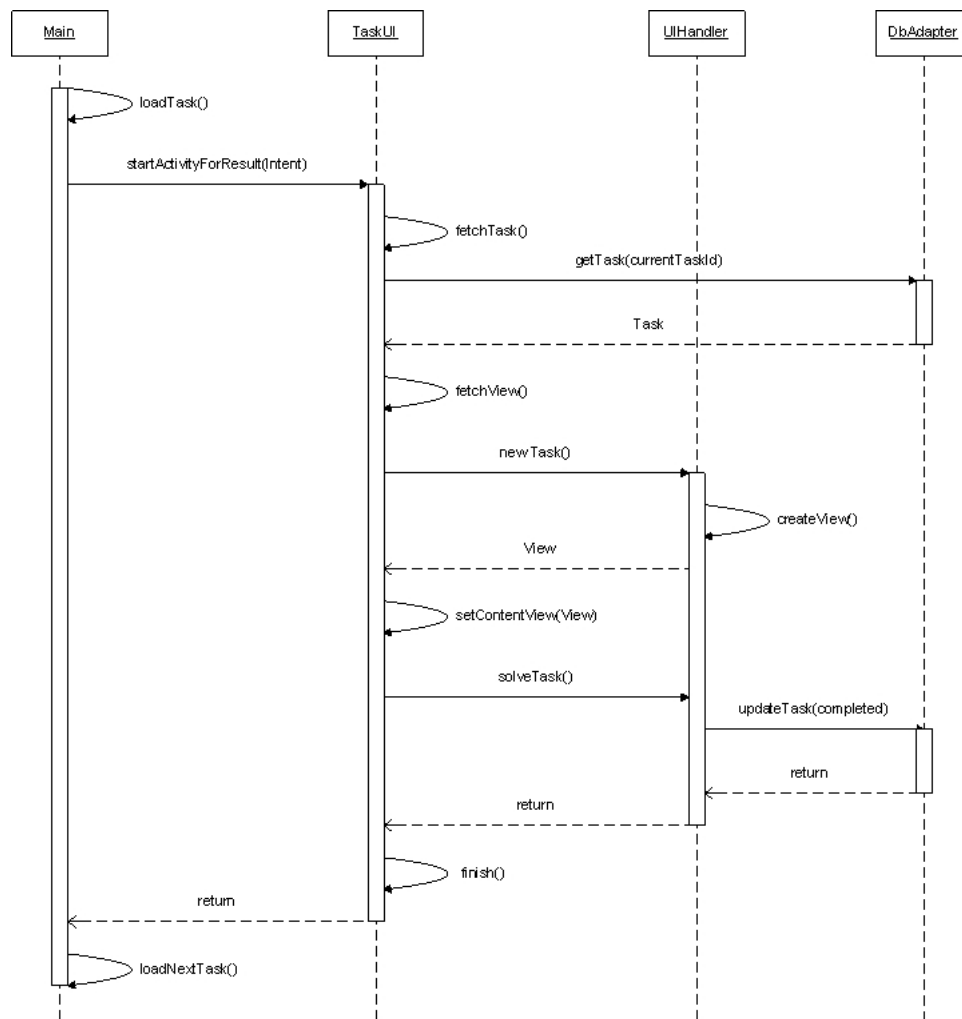
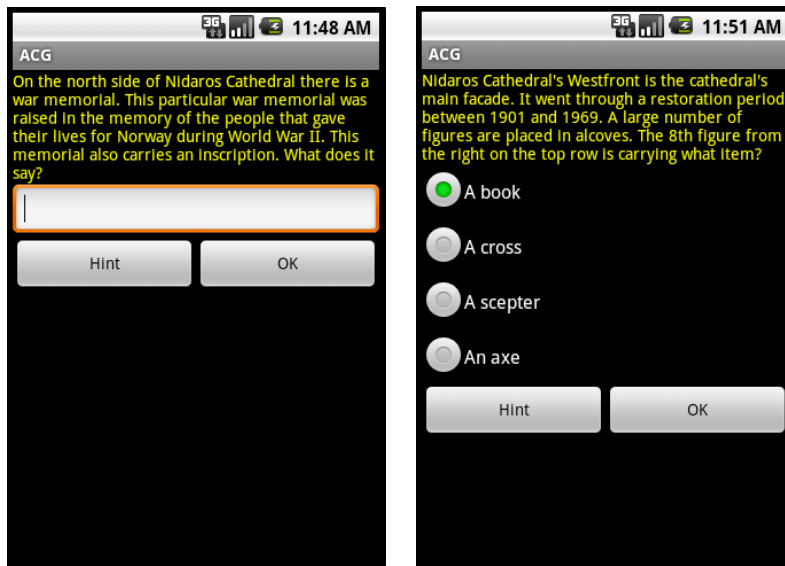


Figure 18.2: Sequence Diagram: Task Flow

Both of these inherit from the class `TaskUI` and the latter handles the common elements for all task types.

Within `fetchView()`, the `createView()` method in `TaskUI` is called. This method is declared abstract and implemented differently in each type of task UI. As explained in Section 18.2.3 Task User Interface, the only part that differs between the `Open` and the `Multiple Choice` task UI is that the first utilizes a text field for user input, and the latter uses a set of radio buttons. See Figure 18.3(a) and Figure 18.3(b) for screenshots of the `Open` task UI and the `Multiple Coice` task UI, respectively. Figure 18.4 shows the overall design of the task UI as well as the relationships between different layout containers and elements. The reasoning behind making the top-most layout a `ScrollView` is so that the view is scrollable if the content is larger than the screen size.



(a) UI: Open

(b) UI: Multiple Choice

Figure 18.3: User Interface: Open and Multiple Choice Screenshots

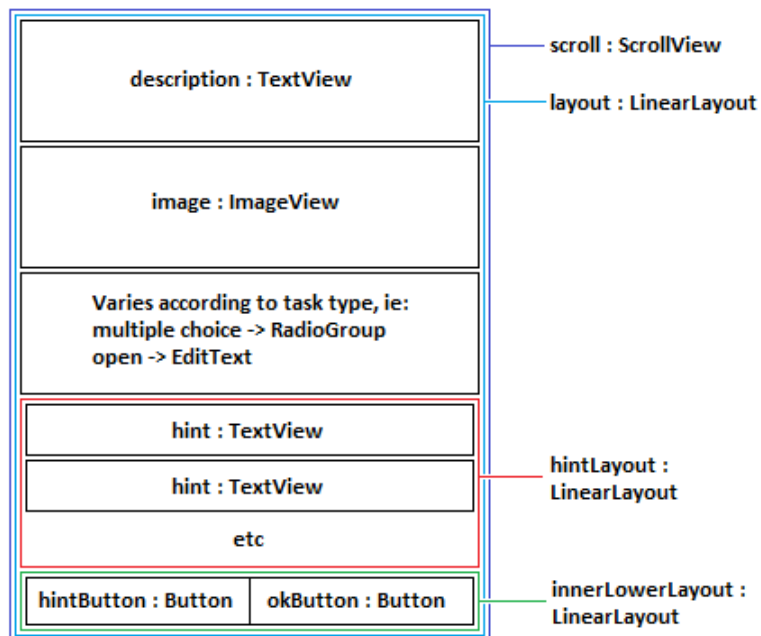


Figure 18.4: User Interface: Task Layout

The confirmation panel that appears when the user proceeds with his or her answer by pressing "OK", is implemented as an `android.app.AlertDialog`. This allows us to place 1-3 buttons as well as a customized text. When the user confirms the answer by pressing "Yes", answer validation is performed by calling `Task.validate()`, which is implemented differently for each task type. See Figure 18.5(b). A correct answer will result in the call of `solveTask()`, as explained in Section 18.3.2.

A wrong answer will result in a popup (`android.widget.Toast`) informing the user that the answer is wrong, and a penalty is given by incrementing the counter for wrong answers tracked within each task. See Figure 18.5(a).

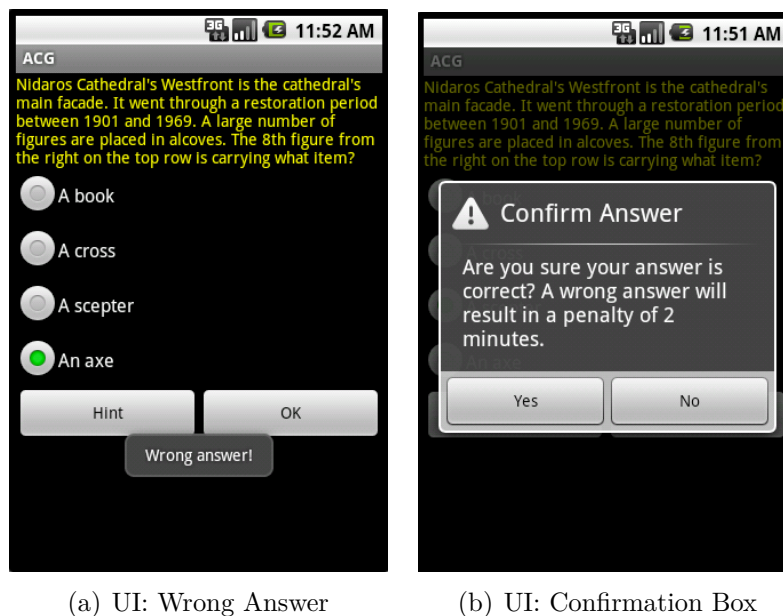


Figure 18.5: User Interface: Confirmation Box and Wrong Answer Screenshots

### 18.3.4 DbAdapter and DbHelper

The creation of a new `DbAdapter` also creates a new `DbHelper`. `DbHelper` extends `SQLiteOpenHelper` and is responsible for creating and retrieving the database. By calling the `open()` method of `DbAdapter` the method `getWritableDatabase()` is called on `DbHelper`. The `getWritableDatabase()` method retrieves the database or creates the database. `DbAdapter` is also responsible for doing all queries on the database.

## 18.4 Testing

### 18.4.1 Task Flow

Testing of the task flow was done incrementally. The overall task flow was created and tested first. This includes `Main.loadTask()`, `TaskActivity.fetchTask()`, and `TaskActivity.fetchView()`. In order to test these methods we added a dummy task and a view. `Main.loadTask()` called `startActivityForResult()` with `TaskActivity` as a parameter to start the activity. `TaskActivity` then used `fetchTask()` to create the dummy task. The method `fetchView()` was then called to load the view. When the user completed the dummy Task, `finish()` was called on `TaskActivity` to finish the activity. The application then returned to `Main`, and `Main` called `loadTask()` and the cycle was complete. We had no issues with testing the overall task flow.

We then implemented routes to make the application able to follow a preset route. In order to test this we created a dummy route containing a fixed number of dummy tasks. We added `route` and `currentTaskId` fields in `Main` to hold the game state. `Main` first created a new route and set the `currentTaskId` to zero. Then `loadTask()` was called. The method `loadTask()` first used `getNextTaskId()` on route to get the next task, which was the first task, and then created a new `Intent` containing `currentTaskId`. It then called `startActivityForResult()` with `TaskActivity` and the `Intent` as parameters. `TaskActivity` did what it did in the previous testing and returned to `Main`. The task id was displayed on the screen each time the task view was created. The task id corresponded to what we inputted in the dummy route. We had no issues with testing the route functionality.

In order to test a fully functional task flow, we implemented the database structure and added real tasks. When the application started, `Main` used the `DbAdapter` to create the database and fill in the game content. On creation of a route, it now retrieved a random route from the database with corresponding task ids. `TaskActivity.fetchTask()` now retrieved the task from the database with `currentTaskId` from the database, and `TaskActivity.fetchView()` now loaded the UI corresponding to the task type of the task retrieved. During testing we discovered that the database was only created upon installation of the application and not on start. This made the overall structure of Google's `SQLiteOpenHelper` (used by `DbHelper`) a bit clearer. Other than that we had no issues integrating the database in the task flow.



### 18.4.2 Game State

We tested the game state by displaying the `currentTaskId` and `routeId` in the log each time it was stored and retrieved. We started the game and saw that a `routeId` and `currentTaskId` was retrieved from the database. These were both 0, since this was the first time the game was started. The application then created a new route and stored the `routeId` in the database. We then solved the task and saw that the `currentTaskId` was properly stored in the database. To test if the game state was properly stored, we killed the application and restarted it. The application retrieved the `currentTaskId` and `routeId` correctly. We had no issues testing game state.

### 18.4.3 User Interface

At the end of the implementation phase, testing of the user interface was done by running the application a number of times and completing the tasks. Wrong answers were also given to make sure that the Wrong Answer alert was functioning correctly. Right answers were obviously given as there is no other way to progress through the application. Both options ("Yes" and "No") were tested in the answer confirmation dialogue.

None of the UI elements deviated from expected behaviour.

### 18.4.4 Answer Validation

Testing was done by selecting wrong and right answers multiple times for multiple choice tasks, and typing in wrong and right answers for open tasks. Answer validation did not deviate from expected behaviour.

### Penalty

The penalty system was tested by intentionally answering wrong to certain questions and making sure that the total accumulated penalty was equal to the expected total penalty. The penalty system performed according to expectations.

## 18.5 Evaluation

We encountered a few problems during the implementation of this sprint, due to our inexperience with and our limited understanding of the Android framework. For instance, we struggled a while with passing objects between Activities, and in the end decided to implement a database to bypass the problem. Most of the other problems originated in the usage of a database, like correctly loading game state variables, objects, etc.

Still, we were able to finish early, so in retrospect we might have added more elements to this sprint.

# CHAPTER 19

---

## Sprint 2

---

### 19.1 Requirements

The requirements for this sprint are:

ID	Description	Priority
FR-TSK-04	Shazam: The user will be given a question and will have to answer it based on the songs provided. The answer will be given in a text box.	High
FR-TSK-05	GPS: The user must get within range of a set of GPS-coordinates, and ask for a location update.	High
FR-TSK-07	Task descriptions shall be able to contain an image.	Low
FR-GPS-01	The application shall be able to use GPS so that tasks where the contestant will have to move within the range of a set of GPS-coordinates is possible.	High
FR-GPS-02	The application shall be able to check the GPS location when the user ask for it.	High
FR-HNT-01	The user shall be able to get hints for all tasks.	Medium
FR-HNT-02	A confirmation box shall be displayed when the user asks for a hint.	Low
FR-HNT-03	The user shall get a time penalty when asking for a hint.	Medium
FR-HNT-04	Hints shall be able to contain an image.	Low
FR-HNT-05	Hints shall be able to contain a song.	Medium

*Table 19.1 – continued on next page*

*Table 19.1 – continued from previous page*

<b>ID</b>	<b>Description</b>	<b>Priority</b>
FR-GST-03	Used hints shall be displayed when the current task is loaded.	Medium

Table 19.1: Sprint 2 Requirements

## 19.2 Design

### 19.2.1 Images

Task descriptions and hints should be able to contain images. The way we want to design this is to have tasks and hints store an `imageId` in the database. This id should also be retrieved when the application retrieves tasks and hints from the database. In order to link these ids to actual images we want to create a static helper class which gets the `R.java`-references for the images by inputting the `imageId` as a parameter. The images should be stored within the application in `res/drawable`-folder.

### 19.2.2 GPS

The application should be able to use GPS to solve GPS tasks. The way we want to design this is to have the user push a button in the GPS task in order to retrieve GPS coordinates. When the user pushes the button, the application should locate the user, and check if the user is at the right location. The right location is determined by a set of GPS coordinates and a radius around the point. To check if the user is within the range, the application should measure the distance between the two points and compare it with the radius. If the user is within the radius, the GPS task is solved and the user should get the next task. If not, a penalty should be given as well as a `Toast` saying that the user is at the wrong location. If no GPS coordinates could be retrieved, it should time out after 1 minute and give the user a `Toast` saying that the GPS coordinates could not be retrieved.

### 19.2.3 Shazam

The Shazam task should consist of all the user interface elements of the Open task, but it should also include a play button for playing the song given in the

task. Songs are designed similar to images in the way they are stored. The database `songId` is stored in the task and hints. These ids should be retrieved when the application retrieves tasks and hints from the database. Similar to the image design, we want to create a static helper class which gets the `R.java`-references for the songs by inputting the `songId` as parameter. The songs will be stored within the application in the `res/raw`-folder.

### 19.2.4 Hints

The hint interface will initially be presented to the user as a button labeled "Hint". This button will, when pressed, give the user a hint. However, it must first present the penalty cost for the hint and ask the user for confirmation.

When unlocked, a time penalty will be given to the user's final time calculation, and a hint will be shown. This will be presented as a text, image, or a button that will play a song.

See Figure 18.4 for an overview of how the hint interface relates to other interface components.

## 19.3 Implementation

### 19.3.1 Images

We implemented the images according to design. We created a new class named `Helper` with the static method `getImage(int imageId)`. This method returns the `R.java`-reference of the `imageId` used as parameter. To add an image to the user interface, we simply just create a new `ImageView` with `ImageResource` set to `Helper.getImage(int imageId)`, where `imageId` is the database id of the image of the task or hint. In order for this to work, the image must be stored in the `res/drawable`-folder of the application, and the `Helper` class must be updated to support the new image.

### 19.3.2 GPS

In order to implement GPS functionality we created a new class called `GPSHandler` which is responsible for GPS updates. The class contains a `LocationManager` to provide GPS service, which includes the methods `requestLocationUpdates()` and `removeUpdates()`. These methods are called when the user wants to start

or stop the updates. They are called within the methods `startUpdates()` and `stopUpdates()` of the `GPSTHandler`. The class also holds the current location that the user shall find in the field `pointLocation`, as well as a boolean `insideTaskRange` that denotes if the user is inside the radius of the `pointLocation`. The class also contains a boolean `onLocationChanged` that denotes if the `onLocationChanged()` method has been fired.

The `GPSTHandler` extends `LocationListener`, which means that the class listens for changes in GPS location. If changes were found, the method `onLocationChanged()` will be called. This method measures the distance in meters from the current location to the `pointLocation` that the user shall find by using the `Location.distanceTo(Location location)`. If the distance is less than the radius, the user is inside the range and `insideTaskRange` is set to true, else `insideTaskRange` is set to false. The application then stops GPS updates by calling `stopUpdates()`.

The GPS task holds an instance of the `GPSTHandler`, and sets the coordinates and radius for the `pointLocation` when constructed by using `GPSTHandler.setCoordinates()`. It also provides methods for accessing `startUpdates()`, `stopUpdates()`, `insideTaskRange`, and `onLocationChanged` of the `GPSTHandler`.

The user interface of the GPS task contains a description, a "Hint" button, and a "Check GPS" button (see Figure 19.1). When the "Check GPS" button is clicked, the application will open a confirm location box which informs the user about penalties involved with failing the task. If the user accepts, the application does the following: starts GPS updates by using the `GPS.startGPS()` method, disables the "Check GPS" button, and creates a new `Thread`. In order to have this thread have an effect on any application functionality, we needed to add a `messageHandler` for handling messages from the thread.

The thread runs for a maximum of one minute. First the application checks if the `GPSTHandler.onLocationChanged` has been called. If it has not been called, the thread continues to run until the one minute has passed or `GPSTHandler.onLocationChanged` has been called. If the time is up, the thread stops and a message is sent to the `messageHandler`. When the message has been retrieved by the `messageHandler`, the application stops GPS updates by calling `GPS.stopGPS()`, enables the "Check GPS" button, and creates a new `Toast` saying "Unable to retrieve GPS coordinates!".

If `GPSTHandler.onLocationChanged` is called the application checks if the user is inside the task range by checking `GPSTHandler.insideTaskRange`. If `insideTaskRange` is true, the application stops the thread and sends a true-

message to the `messageHandler`. If `insideTaskRange` is false, the application stops the thread and sends a false-message to the `messageHandler`.

If the `messageHandler` receive a true-message, the application stops GPS updates with the `GPS.stopGPS()` method, and calls `TaskUI.solveTask()`. If the `messageHandler` receive a false-message, the application stops GPS updates, enables the "Check GPS" button, increases number of wrong answers by calling `Task.incWrongAnswers()` method, and creates a new `Toast` saying "Wrong answer!". See the code for the `onClick()` method of the "Check GPS" button as well as the code for the `messageHandler` see Listings 19.1 and 19.2.

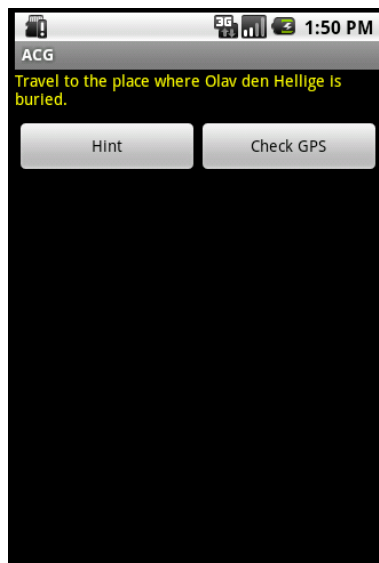


Figure 19.1: User Interface: GPS Screenshot

Listing 19.1: `onClick()` method of the "Check GPS" button

---

```
public void onClick(DialogInterface dialog, int which) {

    startTime = SystemClock.elapsedRealtime();
    gpsButton.setEnabled(false);
    ((GPS)task).startGPS();

    t = new Thread(){
        public void run() {

            boolean stop = false;

            while(!stop){
                while ((SystemClock.elapsedRealtime() - startTime)
                    < 60000) {
```

---

```

        if (((GPS) task).getOnLocationChanged()) {
            ((GPS) task).setOnLocationChanged(false);

            if (((GPS) task).getInsideTaskRange()) {
                mHandler.sendMessage(Message.obtain(
                    mHandler, 1));
                stop = true;
            } else {
                mHandler.sendMessage(Message.obtain(
                    mHandler, 0));
                stop = true;
            }
            break;
        }
    }
    if (!stop) {
        mHandler.sendMessage(Message.obtain(
            mHandler, 2));
        stop = true;
    }
}
};
if (!t.isAlive()) t.start();
}

```

---

Listing 19.2: handleMessage() method of the messageHandler

---

```

private Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {

            // If the user is not inside the radius.
            case 0:
                ((GPS) task).stopGPS();
                gpsButton.setEnabled(true);
                task.incWrongAnswers();
                Toast.makeText(activity, R.string.wrongAnswer,
                    Toast.LENGTH_LONG).show();
                break;

```



```
// If the user is inside the radius.
case 1:
    ((GPS)task).stopGPS();
    solveTask();
    break;

// If no GPS updates could be retrieved.
case 2:
    ((GPS)task).stopGPS();
    gpsButton.setEnabled(true);
    Toast.makeText(activity, R.string.unableToGPS,
    Toast.LENGTH_LONG).show();
    break;

default:
    break;
}
}
};
```

---

### 19.3.3 Shazam

We implemented the shazam task and songs according to the design. We added the static method `getSong(int songId)` in the `Helper` class. This method returns the `R.java`-reference of the `songId` used as parameter. In order for this method to work properly, new songs will be added in the `res/raw`-folder of the application. The `Helper` class must also be updated to support the new song. To use these songs in the application, all we have to do is adding a play button with the code according to Listing 19.3. For a screenshot of the shazam task user interface see Figure 19.2

Listing 19.3: `onClick()` method of the play button

---

```
public void onClick(View v) {
    MediaPlayer mp = MediaPlayer.create(activity,
    Helper.getSong(songId));
    mp.start();
}
```

---

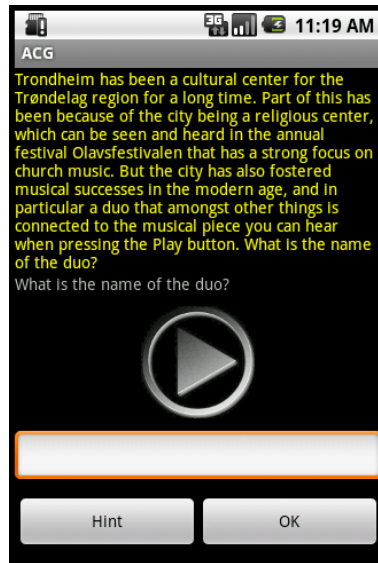


Figure 19.2: User Interface: Shazam Screenshot

### 19.3.4 Hints

Hints are loaded from the database when the application is started, and stored in the task they belong to. As the user progresses through the games and encounters different tasks, the user will have access to a hint button that is shown if and only if the active task has any hints that have not been used already. When the hint button is clicked, a confirmation dialog will appear of the type `android.app.AlertDialog`. If the user confirms his or her choice to unlock a hint, the application will retrieve the next available hint, build a interface component for the hint and update it's used state for both the application and the database. Should remaining unlocked hints ever reach 0, the hint button is removed. Listing 19.4 shows the code that runs when the user confirms his or her choice to request a hint.

Also, when the task interface is loaded, the application checks the task for any used hints and adds any it finds to the interface.

See Figure 19.3 for a screenshot of a task with unlocked hints.

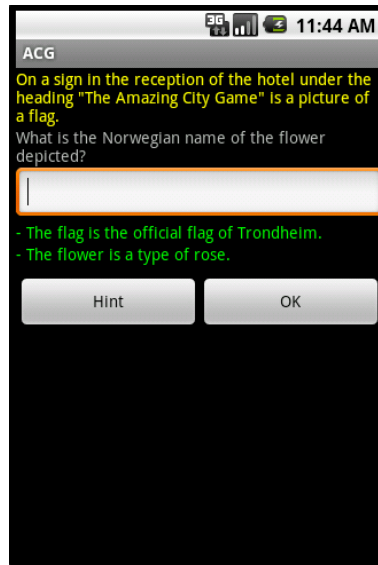


Figure 19.3: User Interface: Task with Hints Screenshot

Listing 19.4: onClick() method of the "Hint" button

---

```

public void onClick(DialogInterface dialog, int which) {
    Hint hint = task.getNextUnusedHint();
    if (hint != null) {
        // builds and adds UI element for the hint
        buildHint(hint);
        // set hint to used in the application and the db
        task.setHintUsed(hint);
    }
    if (task.getRemainingHints() == 0)
        innerLowerLayout.removeView(hintButton);
}
}

```

---

### 19.3.5 Revised Interface and Database

During this sprint, we decided to change the interface slightly. We wanted the question(s) to load after the task image and song, if any such image or song was present. As such, the interface was reorganized a little, and the database also underwent some changes to make this possible. See Figure 19.4 for an updated overview of the interface components and how they relate to each other.

Prior to these changes, only the tasks of type `Open` had questions stored in a separate table. Other task types had their questions stored in the `Task` table added

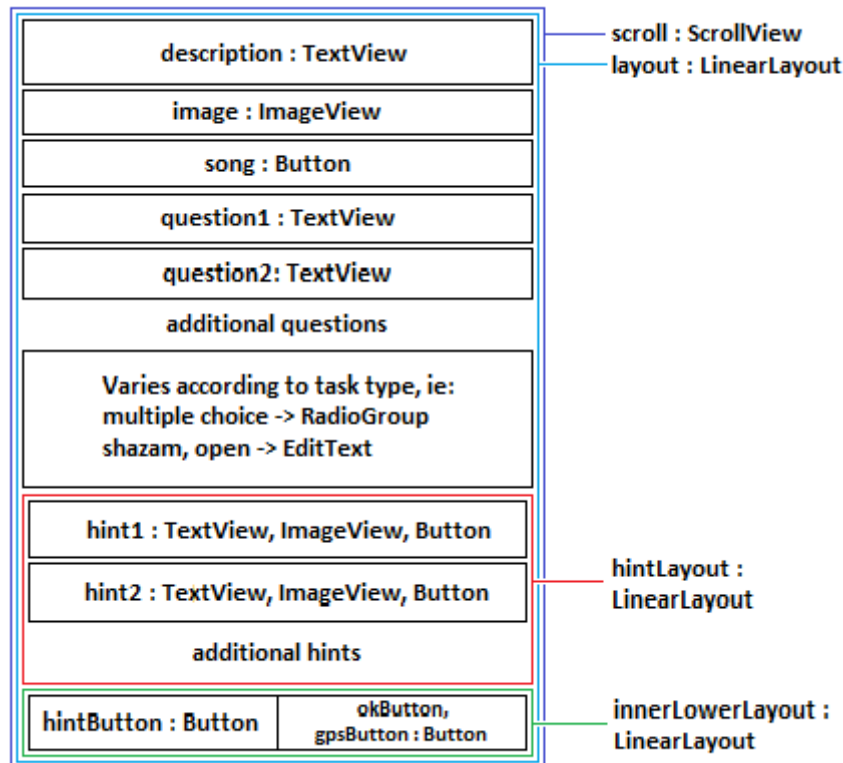
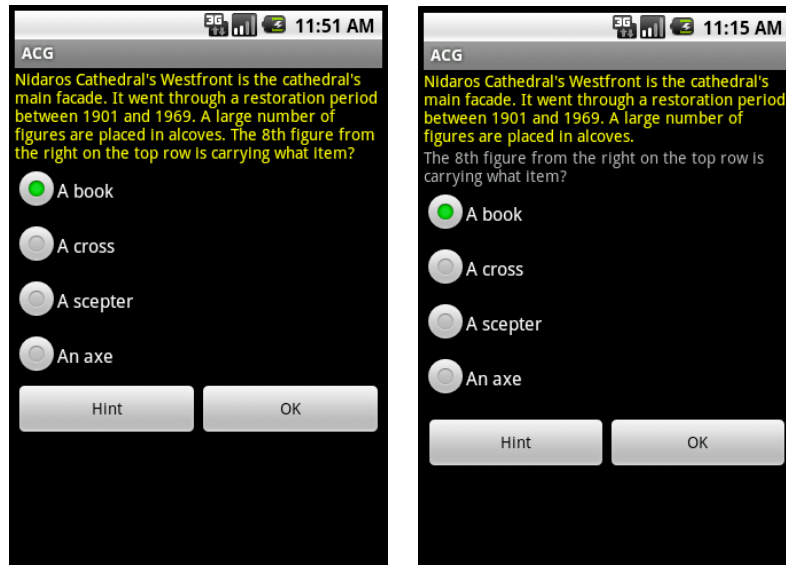


Figure 19.4: User Interface: Updated Task Layout

into the `description` field. The `OpenQuestion` table has now been extended to include the questions for all tasks. As a result, it has been renamed to `Question`.

See Figure 20.2(a) and 19.5 to see the difference introduced in these changes. Any songs or pictures will appear between the description text and the question text.



(a) UI: Multiple choice before

(b) UI: Multiple Choice after

Figure 19.5: User Interface: Before and after UI revision

## 19.4 Testing

### 19.4.1 Images

To test images we simply just stored a test image in the application and saw if the image was correctly displayed in the task and hints. We had no problems getting this functionality to work. We had an issue with the automatic downscaling of images. Even though the image was scaled down to fit the screen in width, the actual height of the image was like the original. But instead of filling it with the image, Android filled it with blank space. This meant that we would have a huge gap between the image and the other user interface elements. The solution was to `setAdjustViewBounds()` of the `ImageView` to true.

### 19.4.2 GPS

In order to test the GPS functionality we first used the emulator. We inputted dummy coordinates and tested if `GPSHandler.onLocationChanged()` was called. If the method was called we updated the text of a `TextView` with the coordinates found by the `GPSHandler`. We had some problems with getting this to work, but after adding the proper permissions in `AndroidManifest.xml` we got it to work properly.

Secondly we wanted to test if the `Location.distanceTo(Location)` of Android worked properly. We also tested this by updating the text of a `TextView` with the distance found. This worked perfectly. Now it was time to test if the entire `insideTaskRange` functionality was working properly, so we went outside and tested extensively. We tested this by updating a `TextView` with the distance for each time the "Check GPS" button was pushed. We found some issues with small radiuses because the GPS was not as accurate as we had hoped for. To fix the accuracy issue we made the radius dynamic, which meant that the radius now added the accuracy to the radius.

Lastly we had to test the `onClick()` method of the "Check GPS" button. The button started the new thread properly. We had some issues with the logic inside the method, but nothing major.

### 19.4.3 Shazam

We tested the songs in the same way we tested images. We added a test song and saw that the play button of the task and hints properly played the song. We encountered no issues while testing songs and the Shazam task.

### 19.4.4 Hints

#### Interface

We tested the interface by running the application a number of times and trying all the different tasks. Our expectations were the following:

1. The hint button would show if there were any hints available.
2. A hint interface element would be added if a hint was asked for, be it a text, image or play song button.
3. The hint button would be removed once all hints had been used.
4. The confirmation dialogue informed the user with the correct penalty.
5. The task would show previously used hints for the task, should the application ever be stopped and resumed.

We tried combinations of having no hints to unlock, unlocking some hints, unlocking all hints, and restarting the application.

The interface lived up to our expectations except for pt. 5. The problem was that the list of used hints for a task was not being built correctly. This has since been fixed.

### Penalty Calculation

Penalty calculation was tested by making the `getPenalties()` method print data to console during it's calculation. The calculation in itself consists of 2 parts.

1. Checking all tasks and summing up the product of the number of wrong answers given and the set penalty for a wrong answer.
2. Finding all used hints, and summing up the penalties for usage of those penalties.

To see that the application calculated the penalties correctly, we made notes during execution of the application on how many wrong answers we gave, as well as which hints we used. We then compared our notes with the data printed to console to check that the calculation for each individual task and hint was made correctly. Finally we compared the final sum. We encountered no deviations or problems during these tests.

## 19.5 Evaluation

We encountered a few problems during the implementation of this sprint. These problems mostly involved the GPS, but we also had some problems concerning hints. The main issue with GPS was that the GPS coordinates retrieved was not as accurate as we had hoped for. The issue with hints was that they were not being built properly in the task.

Even though we encountered some problems in this sprint we were able to finish on time.





# CHAPTER 20

---

## Sprint 3

---

### 20.1 Requirements

The requirements for this sprint are:

ID	Description	Priority
FR-TSK-02	Checkbox: The user selects one or more alternatives to answer a question.	High
FR-TSK-06	Barcode: The user must scan a QR-code of a poster.	High
FR-BCS-01	Application shall integrate barcode scanning so that barcode scanning tasks are possible.	High
FR-MS-01	The application shall display a screen when the game finishes that shows total penalty accumulated.	Medium
FR-MS-02	The user shall be able to display a screen to check time used and penalties.	Low
FR-MS-05	The application shall be able to reset the database.	Low
FR-MS-06	The application shall have a failsafe mechanism so that unsolvable task can be solved.	Medium
FR-MS-07	The failsafe mechanism shall reset any time penalty the user has been given for the unsolvable task.	Medium

Table 20.1: Sprint 3 Requirements

## 20.2 Design

### 20.2.1 Checkbox

The checkbox task interface is similar to most other tasks, but differs from other tasks in regards to selecting answers. This will be done by using the checkbox method, where a number of alternatives may be checked and submitted. One or more of the alternatives may be correct, and the final answer is only validated as correct if all the correct alternatives are selected. See Figure 15.1(b).

### 20.2.2 Barcode Scanning

The barcode scanner interface contains only a button named "Scan" which invokes the external barcode scanner application. When the barcode is scanned, a short confirmation should be shown if the barcode corresponds to a route, before proceeding to the next task. If the barcode does not correspond to a route a short error message should be shown before returning to the initial interface.

### 20.2.3 Finishing Screen

The finishing screen needs to contain one thing only, and that is the sum of all penalties.

### 20.2.4 Reset Database

The way we want to design this is to have the "Welcome Screen" have an options menu which will be displayed when the user clicks the menu button on the Android phone. The menu will contain a button to reset the database. When the button is pushed, a confirm dialog will appear where the user will have to provide a password in order to reset the database.

### 20.2.5 Task Failsafe

The application should provide a task failsafe mechanism so that unsolvable tasks could be solved. For instance, if the GPS is inaccurate or we are unable to get GPS coordinates there are currently no way of solving the task and continue progressing in the game. The task failsafe mechanism will be designed similar to

”Reset Database”. The only difference is that the options menu will be shown on the ”Task Screen” instead of the ”Welcome Screen”.

## 20.3 Implementation

### 20.3.1 Checkbox

The `Checkbox` class keeps a string array for the correct answers, and another one for alternatives. In the database these are stored in the `Alternative` table, previously known as `MultipleChoiceAlternative`. This table was previously only used by the Multiple Choice task, but has been changed to accommodate for `Checkbox` task alternatives. When the user has selected his or her choice of alternatives and confirms, the `validateTask()` method in `Task` will compare the given answer against the stored answer. These are both arrays of strings, and will be sorted with `java.util.Arrays.sort()`, followed by a comparison between the two with `java.util.Arrays.equals()`. See Figure 20.1.

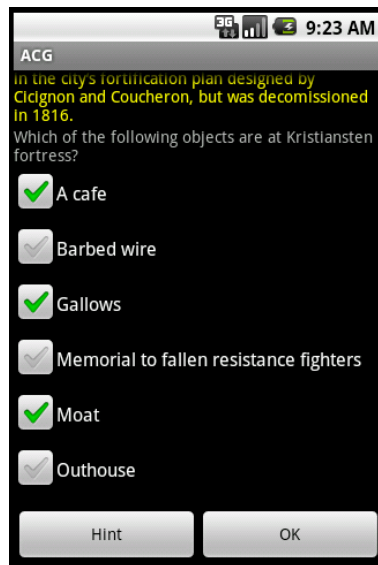


Figure 20.1: User Interface: Checkbox Screenshot

### 20.3.2 Barcode Scanning

The barcode scanning screen is a simple screen containing a reference to the `Barcode` object containing the task and a button. Pressing this button invokes

the external application *Barcode Scanner*. When this application has successfully scanned a barcode the resulting keyword is returned to `TaskActivity` object (the active `Activity`) in our ACG application, which then invokes the `provideRoute()` method in `BarcodeScanningTaskUI`. This method checks the database to see if the scanned keyword corresponds to a keyword of any of the routes. If this is not the case, an error message is displayed and the user is returned to the initial interface. If that it is the case, the database table `GameState` is updated with the new route, and the user is sent to the next task.

### 20.3.3 Finishing Screen

The finishing screen is implemented in the `Final` class. It is a very simple screen that contains only one `TextView` showing the total sum of penalties the user has been given. It does this by executing the `getPenalties` method in `DbAdapter`. This method checks all tasks in the database for number of wrong answers given, which hints were used, and calculates the penalty accordingly.

When a `TaskActivity` is created, its `onCreate()` method sets the next task with `fetchTask()`. Should the next task be `null` (indicating that there are no more tasks), an instance of `Final` will be instantiated and the current view will be set to the `Final` view. See Listing 20.1

Listing 20.1: Excerpt from `fetchView()` in `TaskActivity`

---

```
if(task == null){
    Final finish = new Final(this);
    View v = finish.createView();
    return v;
}
```

---

### 20.3.4 Reset Database

The reset database menu according to design will be shown on the "Welcome Screen". In this case the options menu implementation would have to be in the class `Main`. In order to implement the options menu we overrode `onCreateOptionsMenu()` method where we added the menu items. That is the button for "Reset database". When the button is clicked the `onOptionsItemSelected()` method is called. In this method we call the `showDialog()` method to show the dialog for inputting the password. Then the

method `showDialog()` calls the `onCreateDialog()` method. This method creates the a dialog with a custom interface consisting of a `TextView`, `EditView`, and two `Buttons`. When the "Yes" button is clicked, what the user inputed in the `EditText` is compared to the password. If the password is correct, the `dbAdapter.resetDatabase()` and `resetGameState()` methods are called, and the dialog is dismissed through the method `dismissDialog()`. The `dbAdapter.resetDatabase()` calls `dbHelper.resetDatabase()` which drops the tables in the database and recreates them, thus resets the database. The `resetGameState()` method just resets the routes so that the first task id of the route is the `currentTaskId`. For screenshots of the options menu and the custom dialog see Figure 20.2.

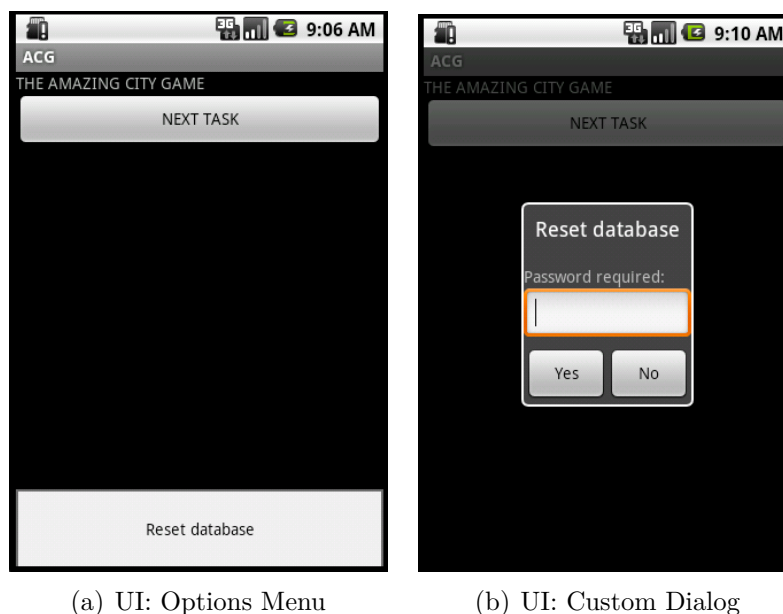


Figure 20.2: User Interface: Reset Database Options Menu and Custom Dialog Screenshots

### 20.3.5 Task Failsafe

The task failsafe mechanism is implemented similar to "Reset Database". The menu for this mechanism will be shown on the "Task Screen". This means that the options menu is implemented in the `TaskActivity` class. Everything is similar to the implementation of "Reset Database" except the code that is run when the "Yes" button is clicked. When this button is clicked the `dbAdapter.completeTask()` is called, which completes the current task and resets all penalties given to the user on the task. Then the task is completed by

calling the method `finish()`.

## 20.4 Testing

### 20.4.1 Checkbox

Testing of the interface was performed by loading the task Kristiansten Fortress and checking that all the parameters loaded correctly. Testing of the answer validation was done by submitting answers multiple times. Submissions consisted of selecting the following combinations of alternatives:

- None
- All
- A few, but wrong
- A few, but correct

The answer validation's behavior performed to our expectations.

### 20.4.2 Barcode Scanning

Testing was performed by creating several different routes with different keywords, and creating barcodes for all these keywords plus some other strings, and printing them all on paper. We then ran the application several times in succession, verifying that scanning the barcodes for the working keywords gave the user the correct next task (i.e. that the user was provided the correct route), and also verifying that scanning keywords that does not exist in the database would give an error notice and return the user to the initial interface.

### 20.4.3 Finishing Screen

This feature was tested by having a few tasks in the game that we completed. When no more task remained, the finishing screen appeared and showed the penalty. The penalty calculation was implemented and tested back in sprint 1 (Task penalties) and sprint 2 (Hint penalties).

### 20.4.4 Reset Database

In order to test this we created the options menu and saw that it was displayed correctly. Next we created the custom dialog and saw that the password matching worked correctly, which it did. We had no problems while implementing the user interface. Next we wanted to see if the database got reset. We killed the application and saw that a new route got loaded, which basically means that the game state was reset in the database. This led us to conclude that the tables were dropped and recreated as intended. After the implementation of "Barcode Scanning" we ran into an issue with reset database. When the database was reset there were no next task in the route, so the application displayed the finishing screen. The problem started in `Main.refreshRoute()` method that got added with the "Barcode Scanning". This method loaded the `Main.route` and `Main.currentTaskId` from the `GameState` table in the database, which was not set before loading the next task. This means that `Main.refreshRoute()` loaded a reset game state, which made the application not working as intended. The solution to this problem was to store the game state when the database was reset. We added `new Route(DbAdapter, routeId)` and `DbAdapter.updateGameState()` in the `Main.resetGameState()` method. This solved the problem and the reset database feature was now working properly.

### 20.4.5 Task Failsafe

We reused the user interface from "Reset Database". To test if the task was completed, we used a couple of hints and answered wrong a few times to get some time penalties. Then we completed the task and saw that finishing screen displayed no penalties. We had no problems with implementing task failsafe.

## 20.5 Evaluation

We had only some minor problems while implementing the features in this sprint. The problem was mainly related to the reset database feature, but it was the barcode scanning that provoked it. We fixed the problem in a matter of minutes. Everything else in the sprint went very well, which made us able to finish early.





## Prototype Finalization

---

This chapter was created after the final implementation was done. After testing the application and routes during the second excursion (see Appendix F), we found a few issues that we wanted to sort out.

### 21.1 GPS

In the location tasks (GPS tasks), if no GPS coordinates could be retrieved, the user had to wait for a maximum of one minute in order to receive any feedback. We felt that this was too long so we changed it to 30 seconds. In addition, we added a progress bar so that the user can see that the application is actually doing something.

We implemented the progress bar by using the standard Android `ProgressDialog`. To update the progress we use the `ProgressDialog.setProgress()` method with `((SystemClock.elapsedRealtime() - startTime)/1000)` as a parameter. For a screenshot of the user interface for this feature see Figure 21.1(a).

We also changed the task completion of location tasks so that the new task will not be acquired automatically. We felt that the current implementation of location task completion did not provide enough feedback. Now, upon completion, the location task removes the "Hint" and "Check GPS" buttons and provides a new button named "Next Task". When the clicks the "Next Task" button, the task is completed, and the next task is loaded. In addition, when the user finds a right location, the application will display a `Toast` message saying "Correct location found!". For a screenshot of the completed location task including the `Toast` see Figure 21.1(b).

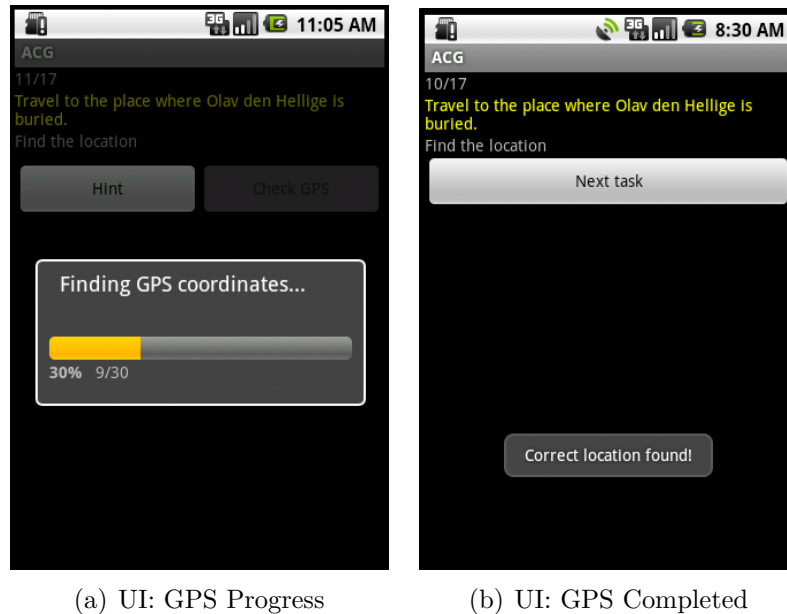


Figure 21.1: User Interface: GPS Progress and Completed Screenshot

## 21.2 Task Failsafe Codes

In the previous implementation of task failsafe, all tasks shared a common failsafe code that could be used to complete the task and reset the number of hints used and wrong answers for that task. During the finalization, we decided to implement separate failsafe codes for each task. The reason behind this was to enable us to field a larger number of groups during the demonstration if necessary. With the previous implementation, we were limited to a number of groups equal to or less than three (three being the number of members on this project). We initially planned to have group tutors on every group, and the group tutors could input the bypass code to enable the group to continue in case of any problems.

With the new solution, not every group needs a group tutor, and they may simply contact one of us for a task-specific bypass code if a problem should arise.

To enable this feature, the database was changed to accommodate task-specific failsafe codes. A field named `bypass_code` has been added to the `Task` table. The `Task` parent class has received a field called `bypassCode`. Corresponding changes have been made to the methods for task creation and task retrieval. Also, the `TaskActivity` now instead checks the failsafe code given by the user against the task's internally stored failsafe code, instead of against a common failsafe.

## 21.3 Interface Changes

The hints are now shown between the task description part and the task input method part. They were previously located between the task specific input method section and the buttons for hints and confirming answers. This was done because it was discovered during testing that for tasks with many hints, there was a large distance between the input section and the button to confirm the answer.

Some information was added to the starting and finishing screens to make them more unique and less confusing.



# Part V

## Results



## CHAPTER 22

---

### Introduction

---

This part contains the results of the experiment. The part consists of three chapters: Demonstration, Survey, and Evaluation. The demonstration chapter contains general information about the demonstration executed on the 3rd of May. The survey chapter contains information about surveys with a focus on System Usability Scale and EGameFlow. It also includes the final version of the survey we used on the demonstration day as well as the results gathered. The last chapter, Evaluation, contains the evaluation of the demonstration, survey, prototype, and risk analysis. It also includes a conclusion of the experiment.





# CHAPTER 23

---

## Demonstration

---

This chapter contains the information related to the demonstration. This includes some general information about the demonstration as well as the execution.

### 23.1 General Information

#### Purpose

The purpose of this demonstration is to have outsiders test the prototype. The results from this evaluation will be used to evaluate the game concept and ultimately lead to designing a platform.

#### Description

Groups of contestants will borrow Android phones with the application installed and play the game on the demonstration day.

#### Date and Location

The demonstration is set for the 3rd of May, and will take place in Trondheim city.

### Contestant Information

For the information given to the contestant prior to the demonstration see Appendix G.

### Handouts

For the handouts given to contestants at the demonstration, see Appendix G.

## 23.2 Execution

For the demonstration day we had 4 groups with 2 students each, for a total of 8 students. These students were recruited from Alf Inge Wang's course in Software Architecture, and 4 of them were Norwegian, 2 were Spanish, 1 was Chinese and the last was Lithuanian. We had 5 group tutors/observers, resulting in 3 of the groups having 1 each, and the last group having 2. The weather was excellent throughout the demonstration.

All groups met up in time at the main entrance of Hovedbygget at Gløshaugen. Contestants brought along their SIM cards to use in our phones as agreed upon. However, one of the phones would not accept the SIM card provided by one of the groups, so we had to replace the phone with a different phone, a Samsung Galaxy. When it was established that all groups were ready, a brief introduction was given, and the first location was disclosed. Immediately after, everyone raced off to Kristiansten Fortress.

Upon arriving at the fortress, each group received different routes according to their arrival time. The groups started solving the tasks at the fortress, observed closely by the tutors. When finished at Kristiansten fortress, the groups continued on their unique routes. From this point, each of the groups were alone with their tutor(s) for the rest of the demonstration. Figure 23.1 is a picture of a group using the smartphone. Figure 23.2 is a picture of a group solving a task in the game.

After completing the game, all the groups except for one had a meetup at Burger King in Munkegata. Here they received a meal and filled out surveys, while the tutors shared their observations with each other. The last group completed their surveys and received a meal at a different location due to finishing very early. Figure 23.3 is a picture of some contestants filling out the survey.

For more pictures taken during the demonstration see Appendix G



Figure 23.1: Using the Smartphone

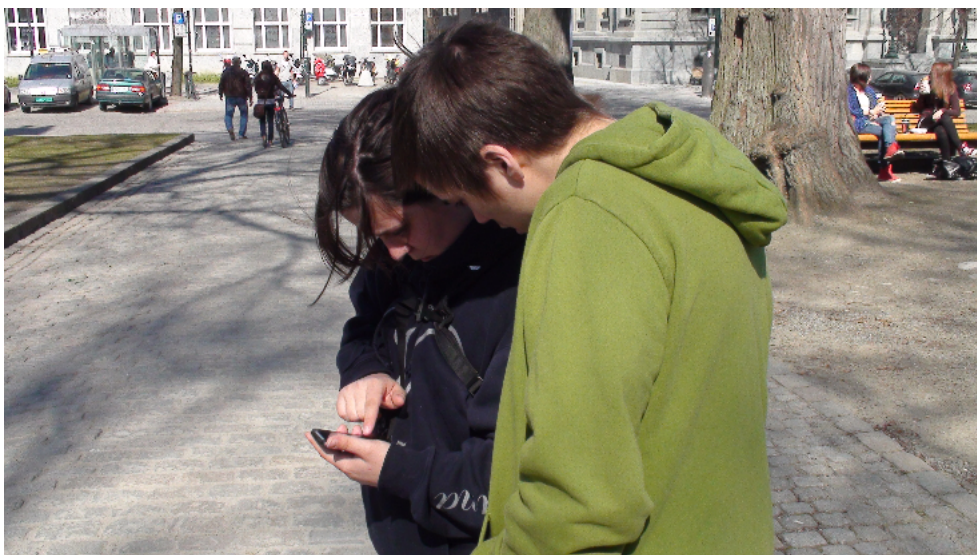


Figure 23.2: Solving a Task

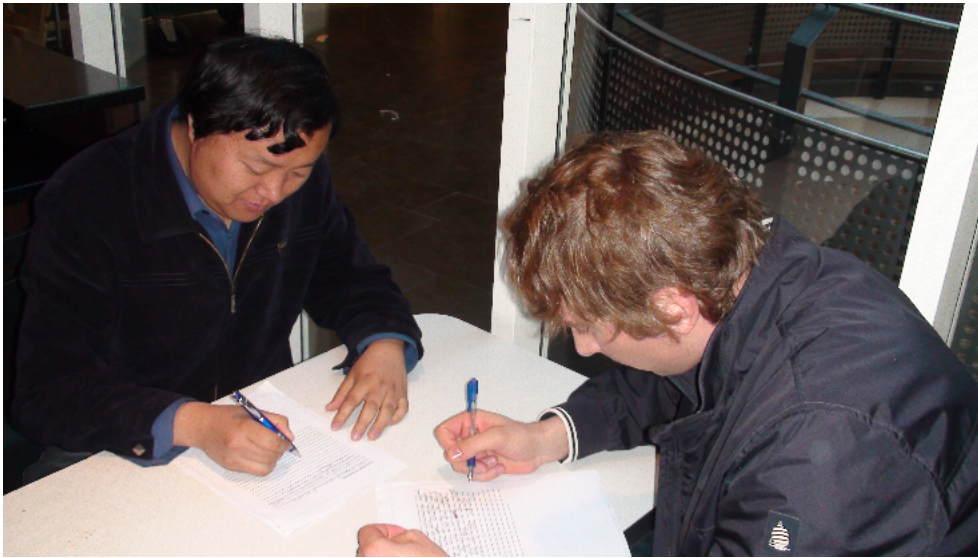


Figure 23.3: Filling Out the Survey at Burger King

## Survey

---

This chapter contains information about the survey used to evaluate the prototype. We start with presenting some general information about surveys before proceeding with presenting the survey used and the results of the survey.

### 24.1 About Surveys

Surveys are an easy way to gather information about a system. They are easy to prepare and can be quickly answered by the respondents. There are several ways to design a survey. For instance, there are four different scale types, which would obtain different kinds of information from the respondents. The four scale types [53]:

- **Nominal:** Classification without need for ranking.
- **Ordinal:** Classification with ranking but without need for equal intervals between ranks.
- **Interval:** Degree of presence of phenomena using equal intervals without need for absolute zero.
- **Ratio:** Degree of presence or absence of phenomena.

System Usability Scale and EGameFlow are two types of surveys that can be used to evaluate a system. Both use the interval scale type.

#### 24.1.1 System Usability Scale

The system usability scale was created by James Brooke as a quick and dirty questionnaire in order to evaluate the usability of a system [27]. The scale was

created from a desire to have a way to measure usability both quickly and simply, and at the same time retrieve a reliable measurement. It consists of ten statements (alternating positive and negative statements) in which respondents have to express their degree of agreement or disagreement on a scale from 1 to 5, where 1 indicates strong disagreement and 5 indicates strong agreement. SUS also includes an algorithm to get a score (0-100) from these responses. SUS has been widely used, and has proven very reliable [20] [8] [79].

### 24.1.2 EGameFlow

To summarize, the scale contains 56 items presented in Likert-type scales, with 1 and 7 respectively representing the lowest and highest degree to which respondents agree with the item

The EGameFlow scale [33] is a scale that measures the experience offered by E-learning games, and helps the game designer to understand the strengths and weaknesses of the game efficiently from the learner's point of view. EGameFlow consists of a number of questions in eight areas, presented in Likert-type scales. To answer the questions or statements, the respondents have to express their degree of agreement or disagreement on a scale of 1 to 7 where the higher number indicates agreement and the lower indicates disagreement. The eight areas of EGameFlow:

- **Concentration:** Games must provide activities that encourage the player's concentration while minimizing stress.
- **Goal Clarity:** Tasks should be clearly explained from the beginning.
- **Feedback:** Feedback allows a player to determine the gap between the current stage of knowledge and the knowledge required for completion of the task.
- **Challenge:** The game should offer challenges that fit the player's skill level, the difficulty of these challenges should change in accordance with the increase in the player's skill level.
- **Autonomy:** The learner should enjoy taking the initiative in game-playing and asserting total control over his or her choices in the game.
- **Immersion:** The game should lead the player into a state of immersion.
- **Social Interaction:** Tasks in the game should become a mean for players to interact socially.

- **Knowledge Improvement:** The game should increase the player's level of knowledge and skills while meeting the goals of the curriculum.

## 24.2 The Survey

For an overview of the questions used in the survey see Appendix D.

## 24.3 Survey Results

The results of the SUS survey can be found in Table 24.1. The results of the EGameFlow survey can be found in Table 24.2, 24.3, 24.4, 24.5, 24.6, 24.7, 24.8, and 24.9. For the raw data collected during the survey see appendix E. For the evaluation of the results, see Chapter 25.

#	Question	Avr	Score
1	I think that I would like to use this system frequently	2.63	1.63
2	I found the system unnecessarily complex	2.13	2.88
3	I thought the system was easy to use	3.88	2.88
4	I think that I would need the support of a technical person to be able to use this system	2.13	2.88
5	I found the various functions in this system were well integrated	3.38	2.38
6	I thought there was too much inconsistency in this system	2.13	2.88
7	I would imagine that most people would learn to use this system very quickly	3.88	2.88
8	I found the system very cumbersome to use	2	3
9	I felt very confident using the system	3.75	2.75
10	I needed to learn a lot of things before I could get going with this system	1.75	3.25
<b>SUS Score</b>		<b>68.44</b>	

Table 24.1: SUS Results

<b>Concentration:</b>	<b>Mean</b>	<b>Deviation</b>
Most of the gaming activities are related to the learning task	5.13	1.13
Generally speaking, I can remain concentrated in the game	5	1.85
I am not distracted from tasks that the player should concentrate on	5.13	1.64
Workload in the game is adequate	5.63	1.3
<i>Mean:</i>	<i>5.22</i>	<i>1.18</i>

Table 24.2: EGameFlow Results - Concentration

<b>Goal Clarity:</b>	<b>Mean</b>	<b>Deviation</b>
Overall game goals were presented in the beginning of the game	4.13	1.96
Overall game goals were presented clearly	4.63	1.6
Intermediate goals were presented in the beginning of each scene	5.75	0.89
Intermediate goals were presented clearly	5.63	1.19
<i>Mean:</i>	<i>5.03</i>	<i>1.1</i>

Table 24.3: EGameFlow Results - Goal Clarity

<b>Feedback:</b>	<b>Mean</b>	<b>Deviation</b>
I receive feedback on my progress in the game	5.75	1.16
I receive immediate feedback on my actions	6	0.93
I am notified of new tasks immediately	6.63	0.52
I receive information on my success (or failure) of intermediate goals immediately	6.5	0.53
<i>Mean:</i>	<i>6.22</i>	<i>0.6</i>

Table 24.4: EGameFlow Results - Feedback



<b>Challenge:</b>	<b>Mean</b>	<b>Deviation</b>
The game provides "hints" in text that help me overcome the challenges	5.38	1.06
The game provides video or audio auxiliaries that help me overcome the challenges	4.13	2.23
The game provides new challenge with an appropriate pacing	5	1.51
The game provides different levels of challenges that is tailored to different players	2.38	1.3
<i>Mean:</i>	<i>4.22</i>	<i>1.04</i>

Table 24.5: EGameFlow Results - Challenge

<b>Autonomy:</b>	<b>Mean</b>	<b>Deviation</b>
I feel a sense of control and impact over the game	4.25	1.28
I know the next step in the game	4.5	2.07
<i>Mean:</i>	<i>4.38</i>	<i>1.53</i>

Table 24.6: EGameFlow Results - Autonomy

<b>Immersion:</b>	<b>Mean</b>	<b>Deviation</b>
I forget about time passing while playing the game	5.88	1.25
I become unaware of my surroundings while playing the game	4.63	2.33
I temporarily forget worries about everyday life while playing the game	5.13	2.23
I experience an altered sense of time	5.25	1.98
I can become involved in the game	5.88	1.36
I feel emotionally involved in the game	5.88	1.64
<i>Mean:</i>	<i>5.44</i>	<i>1.53</i>

Table 24.7: EGameFlow Results - Immersion

<b>Social Interaction:</b>	<b>Mean</b>	<b>Deviation</b>
I feel cooperative toward other classmates	5.38	2.07
I strongly collaborate with other classmates	5.13	2.17
The cooperation in the game is helpful to the learning	5.63	1.6
<i>Mean:</i>	<i>5.38</i>	<i>1.62</i>

Table 24.8: EGameFlow Results - Social Interaction

<b>Knowledge Improvement:</b>	<b>Mean</b>	<b>Deviation</b>
The game increases my knowledge	5.5	0.76
I catch the basic ideas of the knowledge taught	5.38	1.19
I want to know more about the knowledge taught	4.75	1.49
<i>Mean:</i>	<i>5.21</i>	<i>0.99</i>

Table 24.9: EGameFlow Results - Knowledge Improvement

## Evaluation

---

This chapter contains the evaluation of the project. This includes an evaluation of the demonstration, survey, prototype, and risk analysis.

### 25.1 Demonstration

The following is an evaluation of elements that went well or did not go so well specific to the execution of the demonstration, as well as what we could have done differently.

The weather was excellent, meaning sunny and a clear sky. This meant that we had no trouble with using the phone outdoors in terms of rain. The weather was in fact too good, since the sun sometimes made it hard to see the content on the Android phone screens. Still, we are pretty satisfied with how this worked out, since weather conditions in Trondheim are notoriously unstable. However, it was a little chilly at times, and not always pleasant for the participants to hold the phone and solve tasks while the wind was blowing. It would have been better to run the demonstration in the summer, since the weather generally is warmer then. Doing it in the summer would also unlock Munkholmen as a possible location since the site is open during that period.

It would have been a good idea if we agreed upon a site where all the groups could meet up at and fill out surveys and eat. What happened was that one group finished way ahead of the other groups and filled out the survey at a different time and place.

The introduction given to the participants at the start of the demonstration should have been more detailed. This would have helped to make the goal and purpose of the game clearer.

It would be a good idea to give all participants an introduction on how to use Android phones. This would make the game more fair, as the more experienced users had an advantage. On the other hand, it was useful to observe how Android beginners fared with the system.

The backgrounds and skill levels of the participants were very varied. This was very useful in terms of evaluating how different users experienced the system.

The idea of having tutors/observers for each group was good. The tutors made many useful observations and were able to help groups when they were struggling too much. Especially in the case of the groups with foreign students where language was a substantial barrier for understanding and solving the tasks. There are a few downsides to having tutors. Tutors are there to observe and help the participants if deemed necessary. This interaction has an impact on the participants' experience and therefore also the data gathered from the survey. Also, tutors can potentially slow down the group if they are not able to keep the same pace.

## 25.2 Survey

### 25.2.1 Respondents

The number of survey respondents was eight. This gives us a small sample size, and thus the results are seen as useful indications rather than definite results.

Six of the respondents were students from a software architecture course, while two of the respondents were postdoctoral researchers in computer science. All of the respondents therefore have a high technical competence. There were six men and two women. Four of the respondents were foreign; one Lithuanian, one Chinese and two Spanish. However, there were large disparities in initial knowledge of the history of Trondheim. One of the respondents was a local inhabitant. In contrast, another respondents had lived in Norway for less than six months.

### 25.2.2 System Usability Scale

The SUS score for The Amazing City Game is 68.44. In order to evaluate our SUS results, we compared it with the results found by Bangor, Kortum, and Miller [20] in their empirical study of the System Usability Scale. Bangor, Kortum, and Miller [20] compared 2324 surveys over the course of 206 studies and got a

mean SUS score of 70.14 for all the studies. This leads us to conclude that our SUS score is just below average, which indicates that our system's usability is adequate.

This score could probably have been higher if the participants had been given a more extensive introduction and more time to familiarize themselves with the smartphone and the Android operating system. However, prior to the demonstration we decided not to do this, because we wished for a more varied test group. As it is now, several of the participants had no prior experience with Android, and experienced challenges with the many support applications utilized, as well as simpler things such as using the built-in virtual keyboard on android.

### 25.2.3 EGameFlow

In order to evaluate the EGameFlow questions of the survey we compared our results with results found in [33]. Table 25.1 provides an overview of the results from the survey on our game compared to the four games found in [33].

The four unnamed games in [33] are:

- **Game 1** is a motherboard-assembly pairing game where students have to remember computer components by looking at realistic photographs of equipment, and to assemble a motherboard with those components. The game teaches the player by testing the player's memory in combination with player actions and time-limits.
- **Game 2** describes computer parts, and the goal is to have the player understand how to purchase computer parts based on his or her needs. It teaches the player how to make decisions on what computer components to select, and does this by utilizing role-playing, detecting and explaining.
- **Game 3** is an adventure game with the purpose of familiarizing the player with common problems associated with the computer's operating system.
- **Game 4** is basically an introduction to a wide range of computer software. The game simulates different scenarios that are common to typical use of the software. It also includes a number of flash games in the teaching material.

### Concentration

The mean for each concentration question in our result corresponds well with the mean that was found in [33]. This indicates that the concentration aspect of

Category	Game 1	Game 2	Game 3	Game 4	ACG
Concentration	5.118	5.225	5.214	5.153	5.22
Goal Clarity	4.180	5.360	5.048	5.306	5.03
Feedback	4.890	4.950	5.230	5.149	6.22
Challenge	4.654	4.880	5.019	4.7638	4.22
Autonomy	4.686	4.880	5.019	4.764	4.38
Immersion	4.686	4.378	4.651	4.265	5.44
Social Interaction	3.163	3.250	3.365	2.826	5.38
Knowledge Improvement	4.985	5.420	5.171	5.055	5.21

Table 25.1: EGameFlow Games [33] Compared With Amazing City Game (ACG)

our game is adequate.

### Goal Clarity

The mean for questions about the overall goals in the game is lower than the results found in [33]. This indicates that the goals were not presented clearly enough in the beginning of the game. The questions about intermediate goals however got above average results in our survey, which indicates that the intermediate goals were presented clearly. The overall mean for the question group, compared to the ones in [33], was about average. A bit lower than two of the games, about the same amount as one, and alot higher than the last game. This leads us to conclude that the overall goal clarity is adequate, but has the potential to be quite good if the goals were presented clearer at the beginning of the game.

The standard deviation for "Overall game goals were presented in the beginning of the game" was high, which is rather peculiar since all participants received the same information at the start of the demonstration. This leads us to believe that not all of the participants read the information provided or did not pay attention during the introduction.

### Feedback

The mean for each question in this category was alot higher then that of [33]. The overall mean was also alot higher. This indicates that the feedback given to the user by our system is quite good.

### **Challenge**

The results of this category varied alot. The first question about hints got an above average mean compared to [33], which indicates that the usage of "hints" was good. The question about "video or audio auxiliaries" got less than average score. The reason for this might be that the sound did not work for one of the groups, while another group initially had some trouble. The fourth question, which is about challenges, got an above average mean. This indicates that the challenges were provided at an appropriate pace. The last question about "different levels of challenges that is tailored to different players" got a very low result, which indicates that one should put more resources in creating challenges for different levels. Even though creating challenges specific to the skill level of the player was not a part of the prototype work, we felt that this question would give us some feedback when it came to task difficulty. The overall mean of this category was less than that of [33].

### **Autonomy**

The mean of the specific questions as well as the mean for the overall task category was less than that of [33]. This indicates that the autonomy of the game could be improved. Also, a lower score at the question "I know the next step in the game" is not necessarily a bad thing. The game is a knowledge competition and a race. The participants are not supposed to know the next step of the game until they have arrived at it.

### **Immersion**

The means of the specific questions as well as the mean for the overall task category was alot higher than the means found in [33]. This leads us to conclude that the immersion of the game was very good. This in itself might be an ethical concern, since the game takes place in areas with traffic, and the participants need to pay attention to their surroundings.

### **Social Interaction**

The mean of social interaction was also alot higher than that of [33]. This indicates that the social part of the game is also quite good. This is particularly good in relation to lifelong learning, since working with others to learn material is important in that teaching model.

## Knowledge Improvement

When it comes to knowledge improvement, we got an above average result from the questions "the game increases my knowledge" and "I catch the basic ideas of the knowledge taught". This indicates that the contestants have learned something from playing the game, which was one of the goals of the prototype. On the question "I want to know more about the knowledge taught" we got a less than average result compared to [33], but the mean was still high (4.75). This indicates that the knowledge taught in the game is adequate.

## 25.3 Prototype

This section contains possible prototype improvements based on comments received in the survey, and observations made by the tutors on the demonstration day.

### 25.3.1 Survey Comments

The participants wished there was a way to view their own progress in relation to the other groups, and maybe even have a high-score list. This was actually something we discussed early in the prototype phase, but decided to drop since we simply wanted to test the game concept. Still, this is definitely something we consider for the platform. Some participants wished for a function to view the total time used, status per task, and penalty breakdown, which should also be considered for the platform.

Another suggestion was to integrate Google, clickable URLs, and Shopsyavvy into the application. Switching between applications repeatedly became a little distracting. Also, the application and its support applications needed better introductions.

Some participants found the tasks too hard, while others found them too easy. We believe future runs might be better off by picking participants with a similar level of knowledge and designing tasks specifically around them.

A suggestion made by some of the contestants was to have a cancel button for the GPS search. This is easy to implement. Additionally, one of the phones was very low on battery at the end of the demonstration (5-10%), which indicates that battery usage should be reconsidered for future work.



A couple of the groups had problems playing the sound clip for the Shazam task. This feature therefore needs a little more work.

### 25.3.2 Observations

Throughout the demonstration, we observed that many of the open answer tasks would have been better if multiple choice was used instead. This would be a good step towards simplifying the program, and making it less likely that players provide answers that are in fact correct, but not recognized as such by the system. Still, text input remains a useful answer form, but one should take care to define a sufficient set of acceptable answers for any tasks that utilize it.

The Shopsavvy task form was a good idea in theory, but was overly complicated in practice. Groups struggled a bit with this task. The idea might be worth revisiting in the future if the Shopsavvy application could be integrated into the application.

Some of the groups used the public actively by asking for help, and in some cases even had outsiders look for answers with the participants. This was a surprising development and shows that the prototype enables the social expansion of pervasive games (see Section 6.1.2). Also, one group involved another unexpected party, namely the people at the Tourist Information Office.

A number of interface changes could be made to improve the prototype. For example, the button "Check GPS" is somewhat ambiguous. Also, groups were reluctant to use hints. If possible, one could make it a little more inviting to use hints. A tutorial explaining the different task types as well as the importance of using hints could be useful, and should be considered.

Regarding the scanning of the route barcode at Kristiansten Fortress, the process could be explained a little better, as one group initially misunderstood how this was to be done. The reason for this might be that too many sheets of paper were handed out at Kristiansten Fortress, which might confuse the contestants. To prevent this, the barcode should be handed out separately.

In the future, the use of language should be considered more carefully. Certain words used were unfamiliar to many participants and needed explaining, while other words were badly chosen. The system also needs to be consistent in language used in the questions and answers. It is recommended that any further demonstration runs in the city of Trondheim are held in Norwegian, with Norwegian students as participants. The reasoning behind this recommendation is that it will reduce the number of misunderstandings and problems due to language. Some of these problems arise from the fact that many of the location names

only have Norwegian names. Also, the groups did not always read the details of the tasks properly and struggled unnecessarily because of it. This is probably because of the race element of the game, which might make the participants stressful and rush through descriptions. One way of alleviating this issue is to have as little text as possible in the task description so that they read and retain as much of it as possible.

Task difficulty could be balanced a little, but this is a hard task to accomplish. The backgrounds and skill levels of the participants were very varied, and while some participants found certain tasks easy and quick to complete, others found the same tasks too hard and frustrating. Defining a set of tasks that are balanced for the average participant is complicated because it is difficult to predict what he or she might know or not know. Also, this game is a race and the goal is to finish first. Knowledge and skill level should aid a participant to do well. On the other hand, the purpose of the game is to educate the participants, and while participants might do well in the race because of knowledge they already have, they will gain little in the form of newly acquired knowledge. This is clearly a balancing issue between the goal and purpose of the game.

One group used the system in an unexpected way, as they managed to play the playable songs in the game and run the Shazam song identifier simultaneously on the same phone. This was something we did not think was possible, since we assumed Android paused any playing media when switching active application. This was a welcome surprise, and makes Shazam tasks a little more viable. The application does however need some way to facilitate the switch so that the process is faster.

The GPS-functionality worked well overall, but one group got the wrong answer even though they were on the right location. This means that the GPS-functionality needs a little more work in order to work perfectly.

## 25.4 Risk Analysis

This section describes the risks we experienced during this project. We predicted most of them, with exception of two: GPS Inaccuracy, and Foreigners Struggling With the Language.

### 25.4.1 GPS Inaccuracy

The GPS was not as accurate as we had hoped, and this forced us to implement the completion of GPS tasks a little different. The main difference in the design

is that we would take into account the accuracy when calculating the area size that identifies a location.

### 25.4.2 Foreigners Struggling With the Language

Several of the foreigners had language problems during the demonstration. This had an effect on their completion time, but there was always a group tutor/observer available to explain if necessary.

### 25.4.3 Sickness

Team members suffered from illnesses quite often during the beginning of the project. We solved this by re delegating tasks to other group members. All in all, sickness did not have a big impact on the project.

### 25.4.4 Wi-Fi Coverage

The Wi-Fi coverage in Midtbyen was not as extensive as was indicated by coverage charts. This made us design the application to be a little less reliant on Wi-Fi, with all the game content integrated in the Android phone. Additionally, we utilized 3G whenever Wi-Fi was unavailable.

### 25.4.5 Battery

One of the phones almost ran out of battery during the demonstration. We are unsure of exactly why, as initial testing indicated that battery life should not be a problem, and the remaining phones had plenty of battery left. One possibility is the difference in usage patterns.

### 25.4.6 Android Difficulty

We struggled a little in the beginning with understanding how best to design our application for Android in the best possible way. This was expected however, and we had allocated sufficient time to the development phase in order to be able to gain the necessary understanding.

### 25.4.7 Task Unclear/Ambiguous

Despite our best efforts to make tasks clear and unambiguous, some tasks were still problematic. This did not turn out to be a big issue, since we had group tutors/observers available to explain unclarities.

### 25.4.8 Participants Unfamiliar With Android

A few of the participants were unfamiliar with the use of Android. This was indicated to us in the comments. In particular, one participant struggled with the Internet browser for Android. We predicted that this might happen beforehand. Still, we chose not to give a brief introduction to the Android system before the puzzle race demonstration, because we felt that it would give us a more varied test group.

### 25.4.9 Prototype Error

A few groups had trouble playing the audio clip for the Shazam task. One group managed to get it working after rebooting the phone, but another group had to skip the task completely. Luckily, we had implemented task bypass methods to handle these kinds of issues, and the group was able to continue easily.

## 25.5 Conclusion

The survey indicates that the prototype is both usable and educational. Still, it has room for improvement. We have established that good tasks are difficult to make, and that some of the task types we suggested are not appropriate for this type of game. Also, deciding on task difficulty is a balancing act between the goal and the purpose of the game. That is to say, between competing in a race and learning. Furthermore, use of language deserves careful consideration to avoid confusion and misunderstandings.

The game also needs a better introduction so that the participants can focus on playing the game rather than trying to figure out how to play the game. It would also be helpful if as many as possible of the support functions were integrated into the game application rather than be accessed as external applications.

All in all, the prototype and demonstration were a success. The prototype was well received by the contestants, who found it both fun and educational. Also, the

prototype showed that the concept of puzzle races on mobile phones in a lifelong learning context is an interesting concept that should be researched further.



**Part VI**  
**Platform**





# CHAPTER 26

---

## Introduction

---

This part contains the work on designing a platform for creating games like *Amazing City Game*, and will cover the research question in Table 26.1.

RQ3	Pervasive game platform for lifelong learning
RQ3.1	How can we extend this experiment to be a platform to motivate and support lifelong learning?

Table 26.1: Platform Research Questions

In this part of the report we will first go through possible solutions on how we can extend the experiment to be a platform to motivate and support lifelong learning. The part is then divided into four chapters: Discussion, Requirements, Software Architecture, and User Interface. The Discussion chapter is focused on the topics of server solutions, financing, security, and online community. The Requirements chapter contains the requirements of the platform. The next chapter, Software Architecture, is a chapter that contains the overall software architecture of the platform. The last chapter, called User Interface, contains screens and information related to the user interface of the game and the editor.

## 26.1 Pervasive Game Platform for Lifelong Learning

### 26.1.1 How can We Extend This Experiment to be a Platform to Motivate and Support Lifelong Learning?

The pervasive game platform we are to propose should, at a minimum, support the lifelong learning elements included in our experiment to ensure support for lifelong learning. Furthermore, the platform should be extended to include other elements to ensure a strong foundation for lifelong learning.

Our experiment consists a one prototype for an educational pervasive game, and includes only non-formal and informal learning. This is due to the fact that it is difficult to include all three learning types in the same game. However, in the platform, one may create games for all purposes. Those games may utilize other combinations of learning types, and may therefore also include formal learning.

We used the following aspects of lifelong learning in our experiment, and they will also be included in the platform:

- Learn by doing.
- Learn in groups and from each other.
- Educators are guides to sources of knowledge.
- Learn to know.

In addition, the lifelong learning element *assessment is used to guide learning strategies and to identify pathways for future learning* [7] can be utilized. This can be implemented by keeping logs over any wrong answers that have been submitted, time used, and hint usage. The information provided by these logs can be helpful in the future when creating or modifying races, or even giving feedback to participants. They may also be useful for the educators in terms of analyzing the difficulty of tasks.

To allow for the creation of puzzle races, the platform needs to provide a puzzle race editor. Also, to be able to collect feedback and game ideas from the community, a community website should be provided. The website could also serve as a place for participants to discuss the content of the game, and maybe even go further and discuss topics related to the game content. In this way the participants may benefit even more from the lifelong learning aspect of "learn in groups and from each other".

Another important element of lifelong learning is the aspect of individualized education plans. But since the game concept is based on a competition TV show, there are no viable ways to fully incorporate individualized learning plans without compromising the fairness of the game. However, the platform will make it possible for puzzle race administrators, the educators, to create puzzle races of varying difficulty. Provided that students with the same skill level are grouped together, and puzzle races are held for each of these groups, the education plans will be somewhat individualized. This is not a full incorporation of individualized learning plans, but it is a step in the right direction.

One could also further utilize the expansions of pervasive games. A suggestion would be to further expand the social and spatial expansion of pervasive games by designating key persons to hold information and objects that the players are required to use when solving tasks.



## Discussion

---

This chapter contains a discussion on the following topics: server solutions, financing, security, and online community.

### 27.1 Possible Server Solutions

There are several possible server solutions and architectures to consider. Some of the most common architectures will be discussed in the following sections.

#### 27.1.1 Multitier Architecture

In a multitier architecture, each application tier provides specific functionality to its preceding tier and uses functionality provided by its successor to carry out its part of the request processing [80]. The usage of multitier architecture is very common in Internet applications. The most common multitier architecture is three-tier. It is a client-server architecture with three layers: presentation, logic, and data. The presentation tier displays the information to the user. The logic tier controls the application's functionality by performing detailed processing. The data tier consists of database servers, and is responsible for storing and retrieving data.

We suggest that the platform should utilize some form of multitier architecture. In all likelihood it will be a multi-layered architecture, meaning that the modules will be placed at the same physical node. This will offer a logical way of separating functionality, and make the system easier to scale out if performance needs to be improved. The disadvantage is that there will be some communication overhead between different layers, but this is far outweighed by the flexibility and scalability of the system.

### 27.1.2 Client-Server Architecture

*In the Client-Server computing paradigm, one or more Clients and one or more Servers, along with the underlying operating system and interprocess communication systems, form a composite system allowing distributed computation, analysis, and presentation [69].* In the client-server architecture, the client is the process that interacts with the user, and the server provides services to the client. In this definition of client-server architecture it is assumed that the location of the client and server as well as the connection between them does not change. In a mobile environment, the distinction between clients and servers may have to be temporarily blurred, resulting in an extended client-server model [49]. In this client-server model the resource limitations of the client may require operations normally run on the client to be performed on the server. In addition, the need to cope with uncertain connectivity may require the client to emulate the functions of a server [49]. This means that the client-server architecture is more complex with mobile than stationary clients. However, the overall implementation would not be very complex compared to other architectures proposed in this chapter, even with the use of mobile clients. Client-server is therefore a good choice when it comes to platform architecture.

### 27.1.3 Distributed Architecture

The most used architectural model in distributed systems is the client-server architecture. In the distributed client-server architecture, clients may act as servers, and servers may be clients of other servers [29]. This means that the workload can be spread out over multiple servers, and that the system can keep going if one server stops responding. However, such a solution will result in more work done on each mobile phone which in turn will affect the battery usage. The number of clients that are connected to a server will probably be very limited as well, so a centralized server should have no problem handling the amount of requests. Due to the factors mentioned above, we have decided that a distributed architecture is unsuitable.

### 27.1.4 Peer-to-Peer

*Peer-to-peer systems are distributed systems consisting of interconnected nodes able to selforganize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or*

*support of a global centralized server or authority [19].* A peer-to-peer solution has the advantage of being able to treat variable connectivity and automatically adapt to failures in both network connections and peers. However, this solution will be unnecessarily complex for the platform, and it is therefore not a viable option.

## 27.2 Financing

A solution using a central server can be costly, both in terms of hardware costs and in terms of needing personnel to maintain/support the server. One of the ways such a platform can be financed is by government funding, but we would also like to suggest two alternative approaches, that very well might be combined with official funding. The first method is to have ads in the game client. This is an often used method of generating revenue on the Android platform. The second method is to take a fee for hosting a race on the central server.

## 27.3 Security

This section addresses concerns about the possibility of cheating. By cheating, we mean that contestants can find the solutions to tasks by using means other than those intended, or by manipulating race information. An example of cheating is changing the number of wrong answers given in order to decrease the amount of penalties.

Generally speaking, there are two cases. The first is where a game server administers a race and processes actions made by participants. The disadvantage here is that the game client needs to contact the server for every action the user takes (actions such as starting a race, requesting a hint, and providing an answer). If we reduce the amount of data that has to be transmitted between the server and the client, we can reduce the extent of this problem. Large parts of the content, such as task data (description texts, images, audio, and video), can be provided in a single race-file. To ensure that this file can not be read easily, it needs to be encrypted, and preferably with a randomly generated key. A desirable implementation of this is to have the race editor automatically generate these encryption keys in such a way that it is completely transparent to race creators and contestants. Additionally, the game server only returns the decryption keys when needed, for instance a decryption key to a hint requested by the user. However, not all locations have a reliable Internet coverage. Still, it is a viable solution.

The second case is where some or all of the race administration is handled by the game client. Theoretically, it is impossible to make this 100 % secure. In practice though, it is possible to make parts of the system immune to cheating, and there are techniques for making cheating attempts impractical. In most cases it is sufficient to make cheating so impractical that it is unlikely to occur. The following subsections explain a few security measures that can be used to prevent cheating.

### **27.3.1 Encryption of Race Files**

Since the race files contain both task descriptions, questions, and answers, it would be too easy if the race files were saved as plain text. This would enable contestants to get all answers simply by importing the race file into a text editor. At the very least, the game editor should encrypt the race files using an encryption scheme that can be decrypted by the game client. This is not at all safe, as smart contestants can decompile the game client binaries in order to find out the encryption scheme, and decrypt the races using the same techniques. But it will still substantially reduce the probability of cheating.

Another and more viable method is that the race creator can password protect the race. This is essentially encryption of the race file using the password as an encryption key, making the users enter the password before they can start the race. Another possibility is to make the entering of the password the action that starts the race. This way the race creator can control when the users can start the race, reducing the chance that users will hack the race files in advance of the race. In addition, during the race it might be a better strategy to just solve the tasks than trying to obtain answers by hacking the race files.

### **27.3.2 Encryption of Single Tasks**

One way to securely encrypt content without relying on server side validation is to encrypt tasks based on the answer of the previous task. This can work well with text input tasks, but not as well with multiple choice tasks.

### **27.3.3 Save Race Information in the Application's Database**

Race information, such as number of penalties and tasks solved, must be saved somewhere on the mobile phone where it cannot easily be manipulated. One place to do this is in a SQLite database on the application. This is safe on phones that are not rooted (Rooting: the process of modifying the phone operating system



in such a way that the user gains complete control over it), but unsafe on phones that are rooted (or by using Google's official developer phone, such as the Google Nexus One).

### 27.3.4 Game Modes

The security implications give us two separate game modes. The first mode is the secure mode. In this mode all the different content in the game is encrypted using an unique key, and for each game action the mobile client has to communicate with the game server to get the decryption key for the content. This mode is dependent on continuous Internet access. Without Internet access one cannot progress in the game.

The second mode is the insecure mode. In this mode all the different content in the game is encrypted, but all decryption information is stored within the race file, which means that hackers can obtain task information and manipulate race data. However, this can be implemented in such a way that cheating is too impractical to be a major concern. Also, the client may still send updates to a central game server in order to keep track of the race progression. This would allow for high score tables and logging of other useful race data. Updates could be sent as often as possible, which is the same as secure mode, except that Internet access is not required to make progress in the game. Updates could also be sent at periodic intervals. Finally, updates could be sent only when Wi-Fi is available in order to avoid using expensive mobile Internet such as 3G. This would be done after the race has ended.

## 27.4 Online Community

As a supplement to the technical part of the platform we suggest that one also attempts to make an online community around the platform, comprised primarily of a web forum.

### 27.4.1 Web Forum

The community should include a web forum for interested parties where they can discuss knowledge competitions. In order for the online community to thrive, it is important that the platform administrators maintain a continued presence on these forums. This will ensure that the forums are well moderated, as well as

making the platform administrators available to receive bug reports and suggestions for platform improvements.

### 27.4.2 Puzzle Race Guide

Additionally, we suggest that the platform administrators create a puzzle race guide containing tips on how to create good puzzle races and tips on how to avoid common mistakes. The guide could be in the form of a wiki site that certain trusted members of the community are allowed to access and edit. The information in the guide could be referenced to in the editor. For example: in the task editor section, there could be a link to a page in the wiki that provides helpful insight into creating good tasks.

The following subsections contain some of the elements we think should be part of such a guide.

#### Language

Inappropriate use of language when creating tasks can make the tasks ambiguous. In order to prevent this, the tasks should be easy to read and only a single language should be used. A combination of English and some local language could confuse the players. Additionally, words that are not commonly used should be avoided. If such words must be used in order for the tasks to work, the words should be explained. The language of the tasks should also be tailored to the intended players of the game.

#### Multiple Choice vs. Open Input

Task types where the user is required to type in an answer, create challenges in terms of validating the answer. The challenges consist of validating answers that are almost correct, or correct but merely in the wrong form. For instance, words that have the same meaning should be validated as correct. For this reason, it is better to utilize task types where the user selects an answer from a set of predefined alternatives whenever possible, and only use open input tasks if necessary. However, if open input tasks are to be used, penalties should be minimized to improve game balance.

### **Short Task Descriptions**

Long task descriptions occupy a large percentage of the screen and can make the task descriptions difficult to follow. This can make the players unable to clearly see the question of the task, which will lead to more time used on solving the task. The players could then be discouraged from progressing further in the game. Long task descriptions should therefore be avoided.

### **Instant Gratification**

Puzzles should be constructed so that they apply to the concept of Instant Gratification. There should be a short way from question to answer.

### **Penalty Usage**

Fine-tuning penalties could be an important factor to achieve good game balance. Difficult tasks require a lot of time to solve. If difficult tasks are combined with high penalties one would get an unbalanced task. For instance, using high penalties for wrong answers on a GPS task is not a good idea. To be at the wrong location is penalty enough in itself. On the other hand, easy tasks could include high penalties due to the fact that they are easy to solve. In this way, very difficult or easy tasks could be balanced.



## Requirements

---

This chapter contains the high-level requirements for the platform. It includes both functional and non-functional requirements. For low-level requirements, the requirements of the prototype (see Chapter 14) should be used in combination with the user interface proposal for the platform (see Chapter 30) as a guideline.

### 28.1 Functional Requirements

This section contains the functional requirements for the platform.

#### 28.1.1 General

##### Logging

Several pieces of information should be logged in order to subsequently be reviewed by the race editor. This information can help improve future races. Important pieces of information that should be logged are:

- Hint usage. Which hint was used at which time.
- Answer attempts. All answer attempts (both correct and wrong attempts) should be logged, including what answer the user gave as well as the time it was given. Together with hint usage this piece of information will denote the time used at the specific task.
- Battery usage per task.
- Location. If battery capacity allows it, the location of the user should be sampled at a specified interval.

## 28.1.2 Online Community

### Web Forum

The platform should include a website for the community to discuss content, deliver feedback, and provide ideas. See Section 27.4.1.

### Puzzle Race Guide

The platform should include a puzzle race guide in the form of a wiki containing tips on how to create good puzzle races and tips on how to avoid common mistakes. See Section 27.4.2.

## 28.1.3 Editor

The platform should include a puzzle race editor. It will be part of the community website, and provide race administrators with the ability to create, edit, and publish puzzle races with their Internet browsers.

## 28.1.4 Application

### Routes

A puzzle race should be able to support different routes. A route consists of many task groups in a specific order. The ordering of these task groups is what differentiates routes from each other. The server should evenly distribute routes between the contestants to avoid congestion of players during the race.

### Task Groups

Task groups are a number of tasks that are grouped together because they are meant to be solved in sequence. Generally this means that they take place at the same location.

### Tasks

The application should support a multitude of different task types. Additionally, it is important that the application is easily modifiable so that it can be extended

with additional task types (see Section 28.2.2). Giving the wrong answer to a task should incur a penalty. The different task types proposed are shown in the sections below.

**Text Input Task** The application should support a task type where the user has to enter his or her answer into a text input field. The given string is then matched for equality with the correct answer. This task type can be extended into accepting small errors in the text string. This can be achieved by, for instance, using a function that determines if a string is 95% similar to the correct answer. It can also be extended into accepting multiple answers as correct. For instance, allowing the answer to be written in both a local language and the English language. In addition, it might be an idea to allow the input field to be automatically filled with the string from a scanned barcode, delivered by using the external application Barcode Scanner.

**Number Input Task** This task is the same as text input, but will only accept numbers for input, and will also use different validation methods. For example, it could accept an answer within a range or accept an answer that is represented in a set of predefined acceptable answers. The UI control for this task could be a spinner or a slider.

**Multiple Choice Task** The application should support a task type where the user has to choose one answer from a set of predefined choices. The UI controls for this will typically be a radio button group.

**Checkbox Task** This task is the same as multiple choice, except that several answers may be correct. The UI controls for this task will typically be a checkbox group.

**Location Task** The application should support a task type where the user has to be at a specified area. The area can be defined in a number of different ways. Examples are:

- A circle defined as a coordinates with a radius.
- A pair of coordinates defining a rectangle.

This task is solved by the user entering the specified area and invoking the GPS unit to see if he or she is in the correct place. The implementation has to take into account GPS inaccuracy.

**Distance Task** The application should support a task type where the user has to move a specified distance (for example "move 200 meters north"). The user must invoke the GPS unit at a starting point of his choice, and then move the specified distance before invoking the GPS unit again. This task will be great for making a generic GPS task in a tutorial race.

### Hints

Tasks should be provided with hints that are helpful to solving the task. Usage of said hints should incur a penalty.

### Integration with External Applications

The application should be able to easily integrate with external applications. This includes starting external applications, as well as requesting the user to download said applications if not installed on the users mobile phone. The integration also includes using information from external applications, for instance receiving the result of a successful barcode scan by the Barcode Scanner.

## 28.2 Non-functional Requirements

This section contains the non-functional requirements for the platform.

### 28.2.1 Lifelong Learning

The platform should include elements of lifelong learning discussed in Section 26.1.

### 28.2.2 Modifiability

The software architecture of the application should allow for it to easily be extended and modified in the future. This concerns in particular the ability to easily add new task types, and the ability to integrate with external applications - in other words receive information from external applications through Android Intents.



**28.2.3 Durability**

In the event that something unexpected should happen and the application terminates as a result, key values regarding the game state should be saved to ensure durability.

**28.2.4 Security**

The platform should offer security measures of different levels to prevent cheating. See Section 27.3 for a discussion of several possibilities.



---

## Software Architecture

---

This chapter contains the overall software architecture of the platform and how the separate parts interact with each other. Figure 29.1 illustrates the platform data integration architecture. The **Game Server** is the key entity in the figure, since it will handle all the interaction with participants and race administrators. Participants request game content from the game server over 3G/WLAN through the game application running on their phones. The race administrator submits puzzle races to the server and makes them available to participants. Both participants and race administrators may access the community website hosted on the server over the Internet.

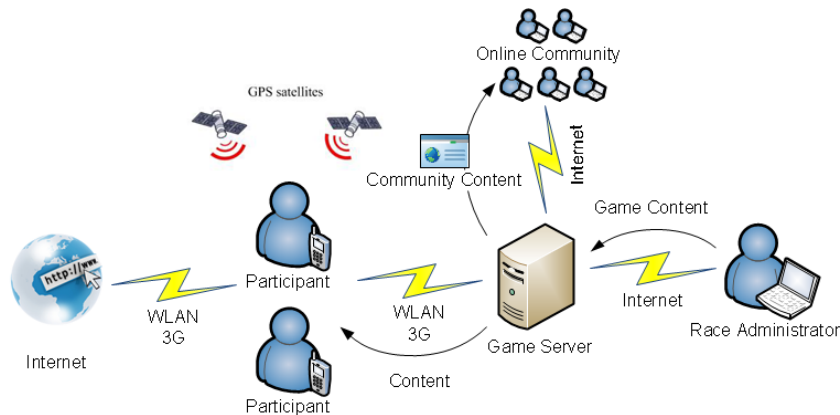


Figure 29.1: Platform Data Integration Architecture

### 29.1 Deployment Overview

Figure 29.2 summarizes how the platform will be deployed across servers and other devices. The diagram also shows that the deployment of the platform fol-

allows a multi-tiered architecture (see Section 27.1.1). On smartphone devices, the Amazing Puzzle Race application will present the content to the players, making it a presentation tier. On the game server, we will have a multi-layered architecture. The database will form the data layer, the components on the application server will form the logic and processing layer, and the web community will form the presentation layer. In more general terms, the architecture may be considered as a client-server architecture where the game server forms the server, and the game application and Internet browser forms the client.

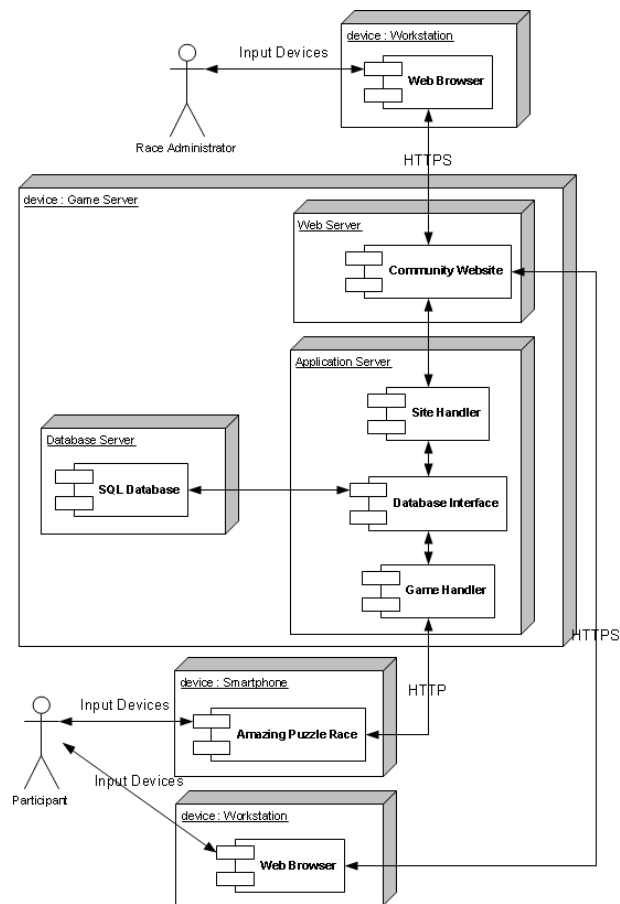


Figure 29.2: Platform Deployment Diagram

### 29.1.1 Game Server

This server consists of a database server, a web server, and an application server.

### Database Server

This server contains a data repository in the form of a database. This is where all information regarding the game content and community website content is stored and retrieved from.

### Web Server

This server handles the presentation of the community website. It will host a web forum for feedback and ideas from the community, high-score lists, and a section that will allow puzzle race administrators to add, edit, and publish puzzle races.

### Application Server

This server hosts the logic units of the system, as well as the interface between said units and the database. There are two logic components. The first is the **Site Handler**, which processes requests that originate from the race administrator and participants through the community website. The second logic component is the **Game Handler**, which processes requests from participants through the Amazing Puzzle Race application running on their smartphones.

#### 29.1.2 Smartphones

The **Amazing Puzzle Race** application will run on the participants' smartphones. It serves as the presentation layer of the game for participants, and communicates with the game server via the HTTP protocol. Processing of this interaction is handled by the **Game Handler** component on the game server.

## 29.2 Editor

The Editor is the part of the community website that will allow the race administrators to add, edit, and publish races. The functionality available in the Editor will be provided by the **Site Handler** component. Figure 29.3 shows the steps for creating and editing puzzle races. The edit tasks process in said figure is expanded in its own workflow diagram, see Figure 29.4.

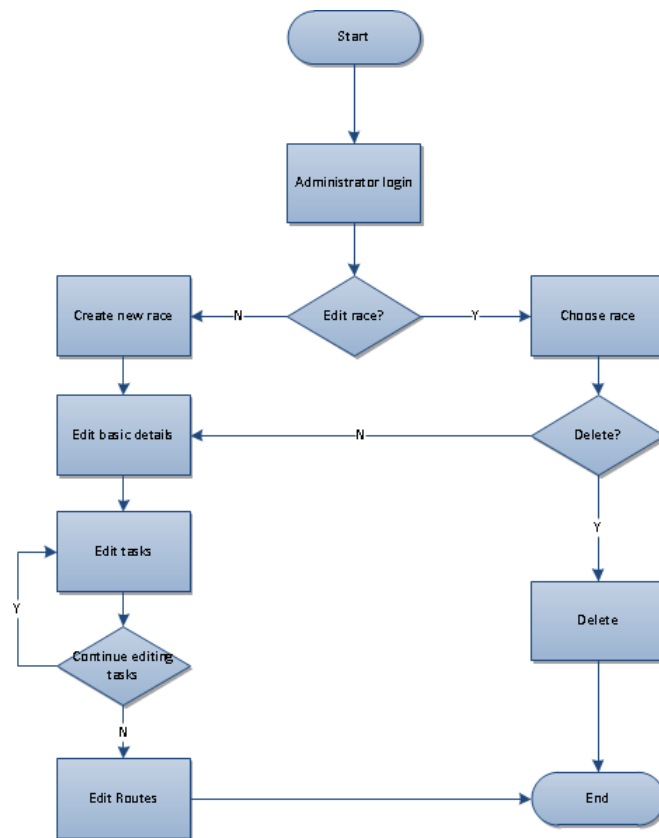


Figure 29.3: Workflow For Editing Races

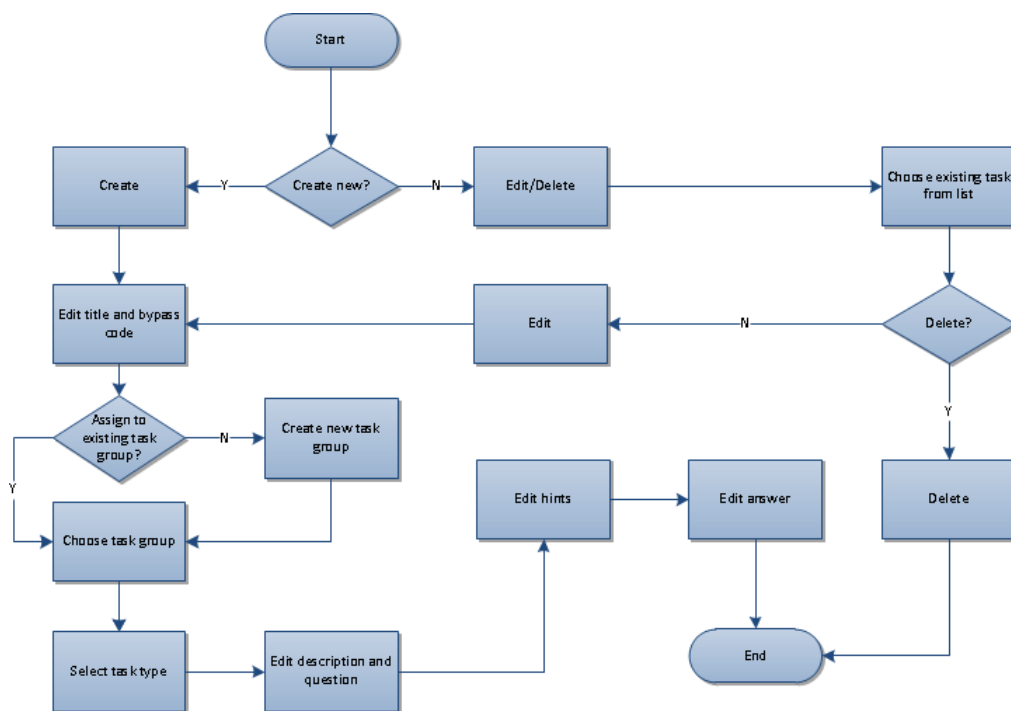


Figure 29.4: Workflow for Editing Tasks

### 29.3 Amazing Puzzle Race

This component of the platform is the game application that will run on the participants' smartphones. It will be based on the prototype designed in the experiment, with a few changes to respond to user feedback and further extend on support for lifelong learning (see Section 26.1). The user will interact with the game application, and the results of this interaction will be processed by the **Game Handler** component on the server. Figure 29.5 shows the screen flow of the application.

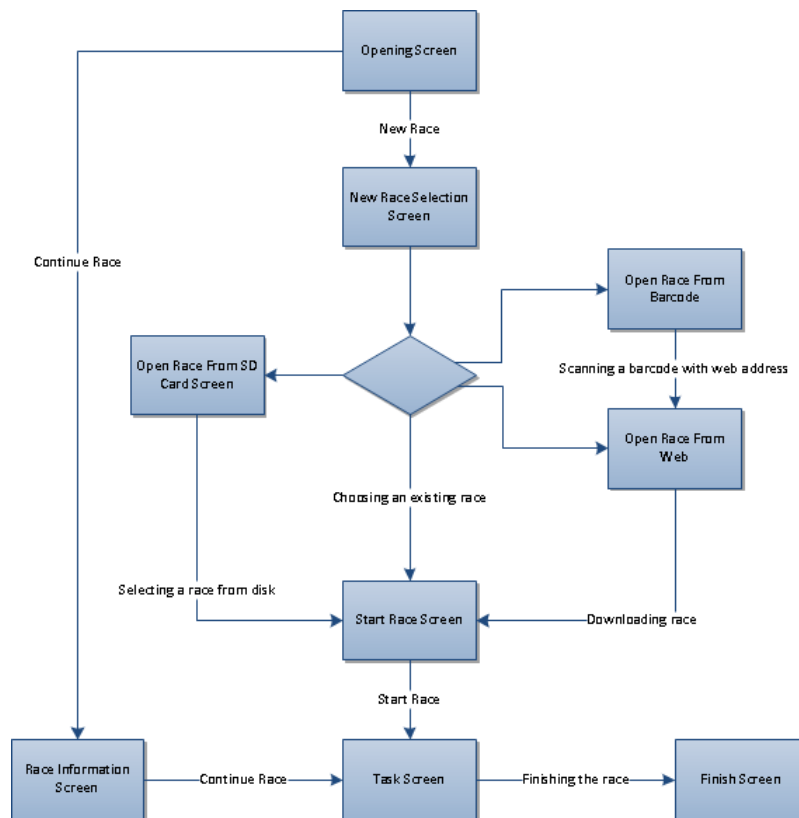


Figure 29.5: Amazing Puzzle Race Screen Flow



# CHAPTER 30

---

## User Interface

---

This chapter contains an user interface proposal for the platform. Including client game screens and editor web pages.

### 30.1 Game Screens

The application can roughly be divided into a few distinct screens. The flow between screens is shown in Figure 29.5. Navigating between the screens is achieved by pressing the navigational controls, and using the BACK-button on the phone. We assume that the users are familiar with the general BACK-button behaviour on Android phones, and that there are no back or exit buttons in the user interface of the application.

#### 30.1.1 Opening Screen

This is the screen the user is presented with when he or she first starts the game. This is a very simple screen that contains a button to start a new race, and a button to continue an existing race should there be any active races. Pressing BACK at this screen will exit the application. The screen is shown in Figure 30.1.

#### 30.1.2 New Race Selection Screen

This screen provides the user with several different possibilities in how to select a new race to start. The screen contains a list of races imported into the game, and three buttons to import a race into the game. The races can be imported from

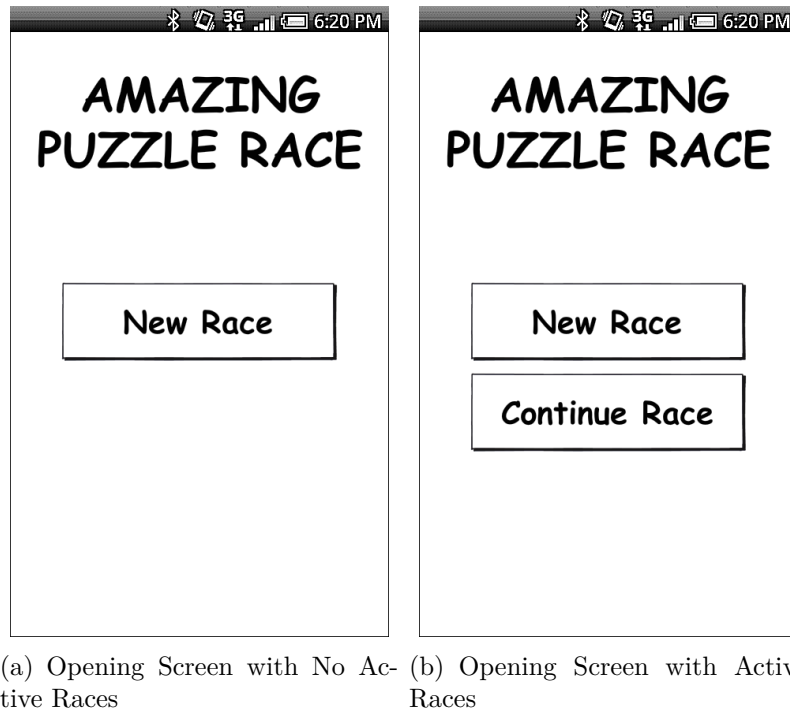


Figure 30.1: Opening Screen

a file on the SD-card, directly from the Internet, or from an Internet address obtained from a barcode using the barcode scanner. Pressing BACK at this screen will make the application go back to the Opening Screen. The screen is shown in Figure 30.2(a).

### 30.1.3 Start Race Screen

This screen contains a description of the selected race with an icon of the race and a button to start the race. The button will send the user to the Task Screen for solving the first task. Pressing BACK at the Start Race Screen will make the application go back to the New Race Selection Screen. The screen is shown in Figure 30.2(b).

### 30.1.4 Open Race Screens

These three screens give the user the possibility to open a race from a SD-card or download a race from the Internet. Pressing BACK at any of these screens will make the application go back to the New Race Selection Screen, and cancel

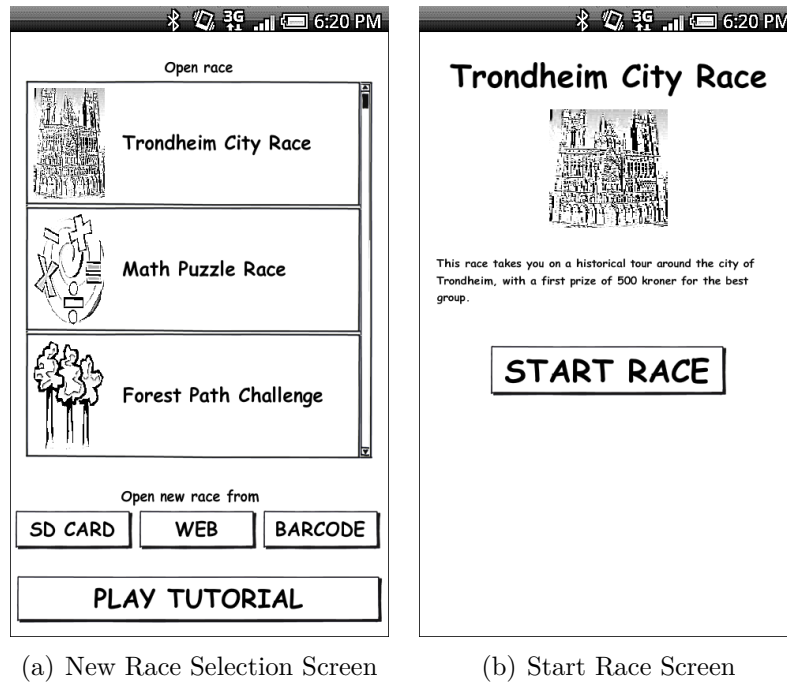


Figure 30.2: New Race Screen and Start Race Screen

any downloads in action.

The screen for opening a race from the SD-card is used when the user has saved a race on the SD-card, for instance from an attachment to an email. The screen contains a file explorer, and clicking on a race will import it into the application. The application will then go to the Start Race Screen. The screen is shown in Figure 30.3(a).

The Open Race from Barcode Screen is actually not a screen in the race application, but an external application, "Barcode Scanner", that has functionality for scanning barcodes. When the external application has successfully scanned a barcode and obtained a link, it will return to the race application and open the Open Race From Web Screen with the URL field auto-filled with the scanned link. The screen is shown in Figure 30.3(b).

The screen, Open Race From Web, contains a text field for entering a web address, and a button to start the download. While downloading, a progress bar will be shown. When the race is finished downloading, the application will automatically import the race into the application and then go to the Start Race Screen. The Open Race From Web Screen is shown in Figure 30.4.

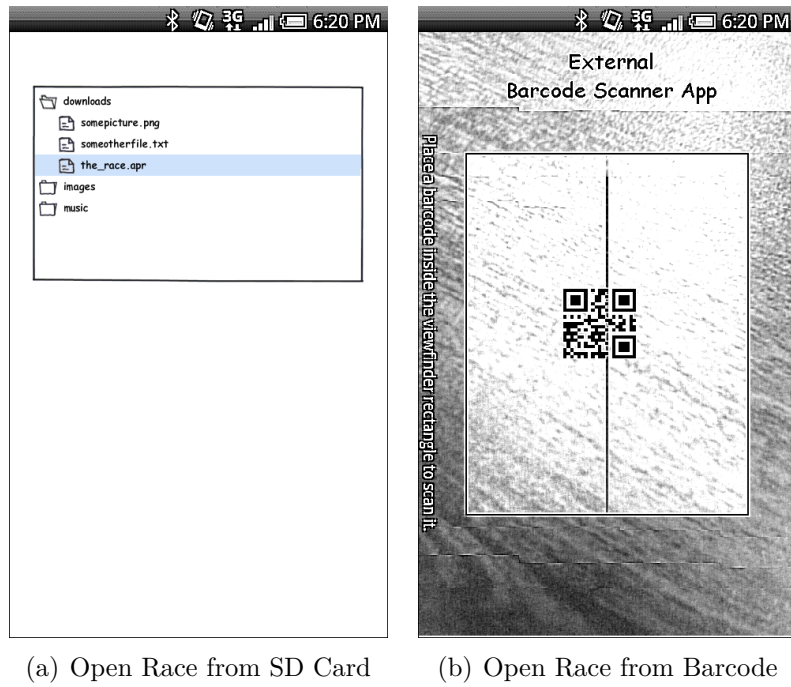


Figure 30.3: Open Race from SD Card and Barcode Screens

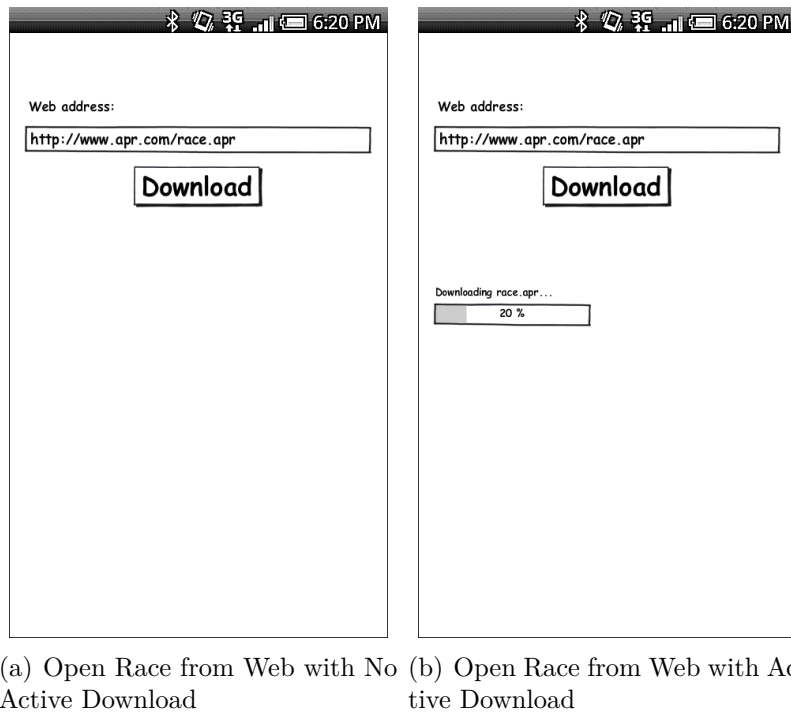


Figure 30.4: Open Race From Web Screen

### 30.1.5 Task Screen

This screen shows the tasks the user has to solve in the race, and it is the screen the user will spend the majority of his or her time at. Pressing BACK in this screen will make the application go back to the Race Information Screen. Two examples of a task screen is shown in Figure 30.5.

The top part of the screen contains a running timer displaying the race time, which is real time spent including penalties. The top part also includes a counter of how many tasks that have been solved. After that follows a few *sections*. The first section is the task description. Following the task description come the hint sections. Initially there are no hints, but as the user asks for hints they will be placed below the task description and the previous hints. Following the hints, or directly under the task description if there are no hints, comes a section showing the question or instructions on how to solve the task. The question could have been a part of the task description, but if the user wants to use hints, the question might get too far away from the task solving control. Testing showed that this can lead the user to forget and subsequently misinterpret the question.

These three sections each consist of visual elements such as text, images, audio, and video, and it is up to the race creator to determine the number, content, and order of these elements. This gives the race creator creative control of how the tasks will look. Conceptually, this works much like giving the race creator the ability to make a HTML-page. The text elements include basic formatting, such as bold, italic, and underlined text, with the ability to change font type, font size, and font color. Additionally, sections of the text can be designated links to open specific web pages or external applications. Images are also able to function as links to web pages/external applications. Audio typically consist of an image with overlay controls for playing/pausing/stopping the audio clip. Likewise, video will be presented with a still image from the video, with overlay controls for playing/pausing/stopping the video clip. An example of a task, with a description with an image, and hints with audio/video is shown in Figure 30.5(b).

Below the description, hints, and question, follows the task solving control. The task solving interface looks different depending on the task type. For example the screen shown in Figure 30.5(a) is a multiple choice task (one possible correct answer), while the screen shown in Figure 30.5(b) is a task with a text input answer.

Finally, at the bottom of the screen follows the task control panel section. This section contains two buttons. The left button is titled "New hint", and pressing it will give the user a popup warning him or her about the size of the penalty, and asks if he or she really wants to receive a hint. Should he or she accept a new

hint, a hint is added under the task description, and a penalty is added to the total race time. The button will be shown as inactive if there are no more unused hints left. The button to the right is titled "Solve task", and is initially inactive until the user has provided an answer suitable to the task type. When the user presses this button, he or she will get a popup warning about the penalty that will be given should he or she submit a wrong answer, and asks if the user really wants to submit his or her answer. Should the user accept, and the answer be wrong, the user will be notified that the answer is incorrect, and a penalty will be added to the total race time. Should he or she accept, and the answer is correct, the next task will be loaded.

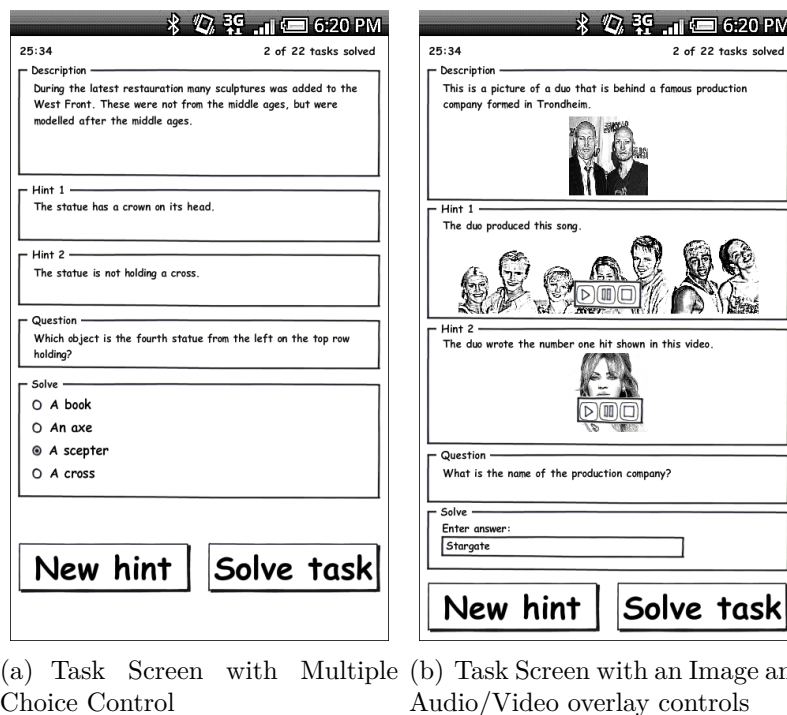
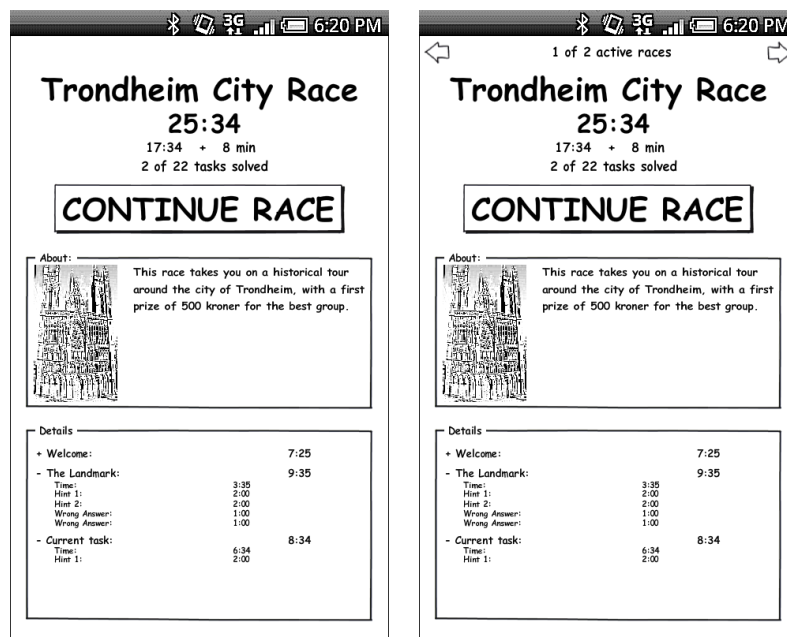


Figure 30.5: Task Screens

### 30.1.6 Race Information Screen

This screen gives the user information about the ongoing race, or races if there happens to be several active simultaneously. This includes the race title and description, an image and text, and a button to continue the race and go back to the current task. The screen also includes a detailed breakdown of time usage. There is an ongoing timer that displays the race time, broken down into real time spent and total penalty. There is also a breakdown of time used per task,

which again can be broken down into real time spent and amount of hints used, as well as wrong answers given and how large a penalty that gave. A detailed breakdown for each task can be toggled on and off by clicking the +/-symbol. The tasks are shown by their task title, except for the current task which is shown as "Current task". The reason for this is that the title of the current task might give information away that should not be given away. All ongoing timers are updated every second. This includes the total race time, the total real time spent, race time for current task, and real time spent for current task. Pressing BACK at this screen will make the application go back to the Opening Screen. The screen is shown in Figure 30.6.



(a) Race Information with One Active Race (b) Race Information with Several Active Races

Figure 30.6: Race Information Screen

### 30.1.7 Finished Race Screen

This screen is presented to the user immediately after the last task of the race has been solved. The screen is quite similar to the race information screen, and gives the user the ability to see how he or she performed. If online high-score tables are implemented, this is the most natural screen to place those. Pressing BACK at this screen will make the application go back to the Opening Screen. The screen is shown in Figure 30.7.

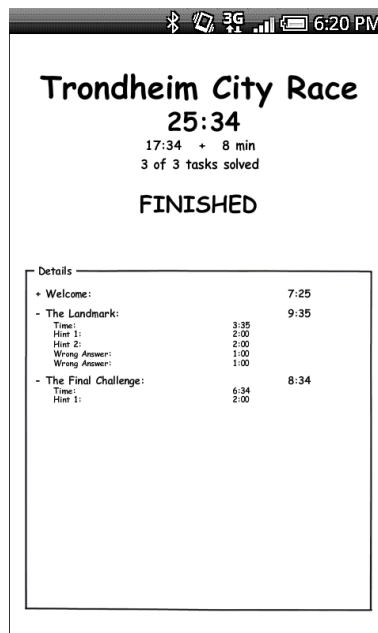


Figure 30.7: Finished Race Screen

## 30.2 Editor Pages

The web editor has the following distinct pages.

### 30.2.1 Start Page

This page (shown in Figure 30.8) gives the user the option to create a new race, edit an existing race, or to view the puzzle race guide. When choosing the option to create a new race or edit an existing one, the Race Details Page will be opened.

### 30.2.2 Details Page

This page (shown in Figure 30.9) gives the users the possibility to enter general information about the race. This includes adding a description, adding an image as a race icon, designating whether or not the race has multiple different routes (which will enable or disable the Routes-tab), or deciding if the race is password protected. A password-protected race requires contestants to enter a password in order to be allowed to start the race and read the first task on their smartphone.

The race administrator can then choose to publish the race. If the editor is



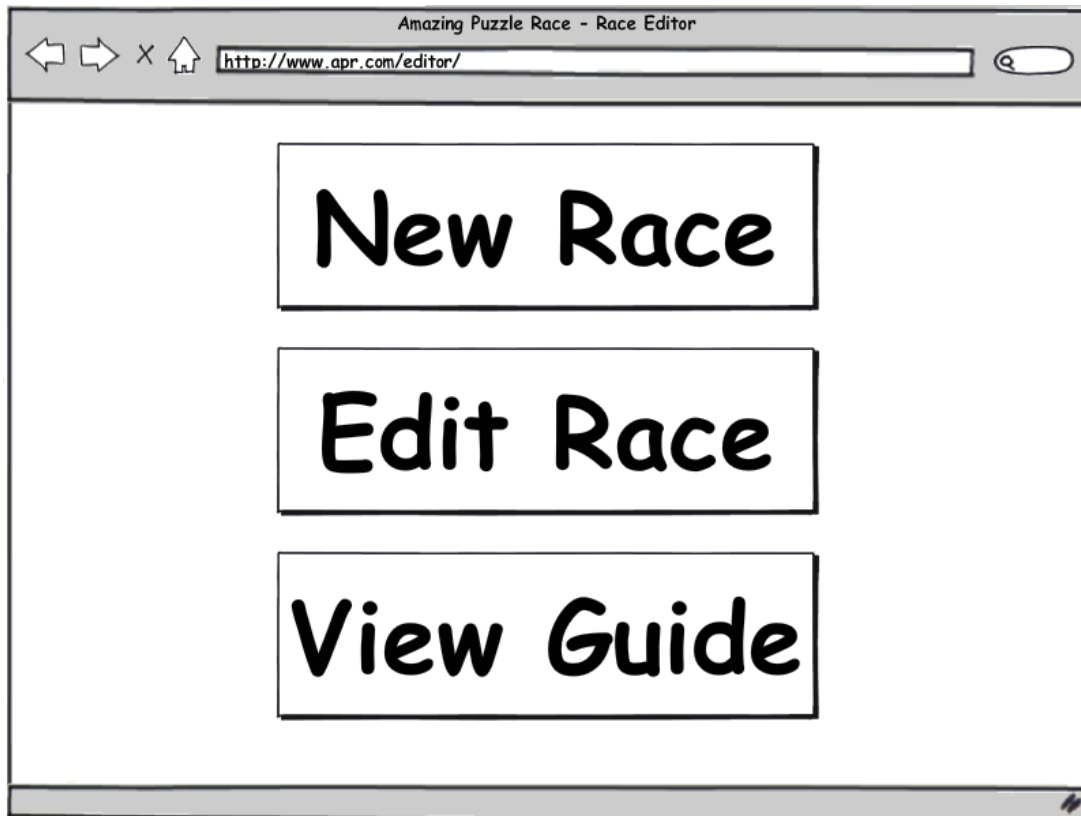


Figure 30.8: Editor Start Page

integrated in the game server the race can be automatically transferred to the server, or if there is no such solution one can publish the race by pressing "Publish Race". "Publish Race" typically means that one can download the race in the form of a race file. The race administrator can also create a PDF of the race, which is just a simple document with all of the race content.

### 30.2.3 Tasks Page

This is the editor's most complex page (shown in Figure 30.10), and it is divided into two sections. The upper section contains a list of all tasks, and gives the race creator the ability to create new tasks, change the order of task groups, and select tasks for editing.

The lower section gives the race creator the ability to edit tasks. Firstly, tasks can be given a title. The title will not be shown to contestants until after they have solved the task. The creator can also provide a bypass code that can be used

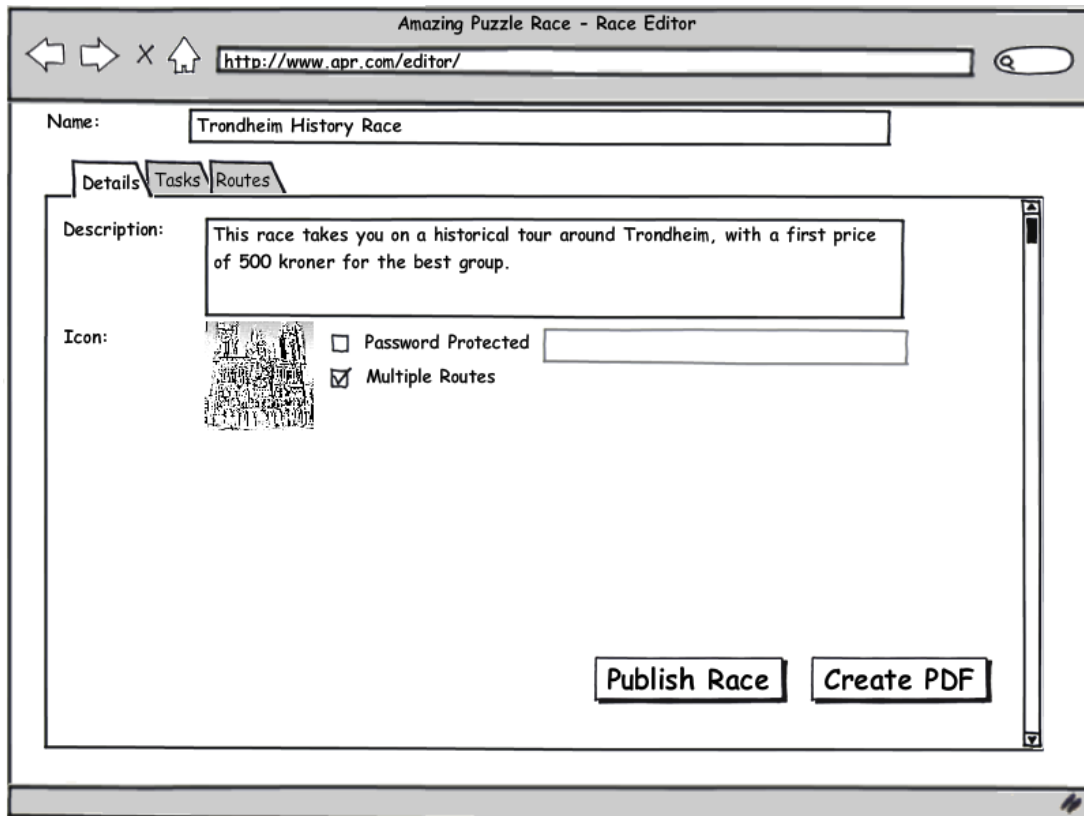


Figure 30.9: Editor Details Page

to bypass the task entirely. The bypass code is typically used as an emergency solution in case something unexpected should happen. Next, a task is assigned to a task group. Task groups group together tasks that should be solved at the same location so that routes consist of a list of task groups instead of tasks. The creator then has to designate which type of task it is.

Below these areas, there are at least two identical sections where the user can provide the task description and the task question/instructions. These sections are essentially "What you see is what you get"-editors, where the five icons on the right side can be pressed in order to add, respectively, a text snippet, an image, a hyperlink, an audio clip, or a video clip. The "Add Hint"-button will provide the creator with additional hint sections where hint content can be added. The creators can add as many hints as they would like.

The task solving control is located at the bottom of the page. This control will differ depending on the task type. Shown in Figure 30.10 is the task solving control for the location task type, where the creator can define an area on a

map. This can be implemented by for instance using Google Maps. The area designated is the area the contestant has to enter in order to solve the task.

### **30.2.4 Routes Page**

This page (shown in Figure 30.11) gives the race creator the possibility to define multiple routes. Pressing the "Add Route" button will add a new box with a list of all the different task groups created, and the race administrator may then change the order of these.

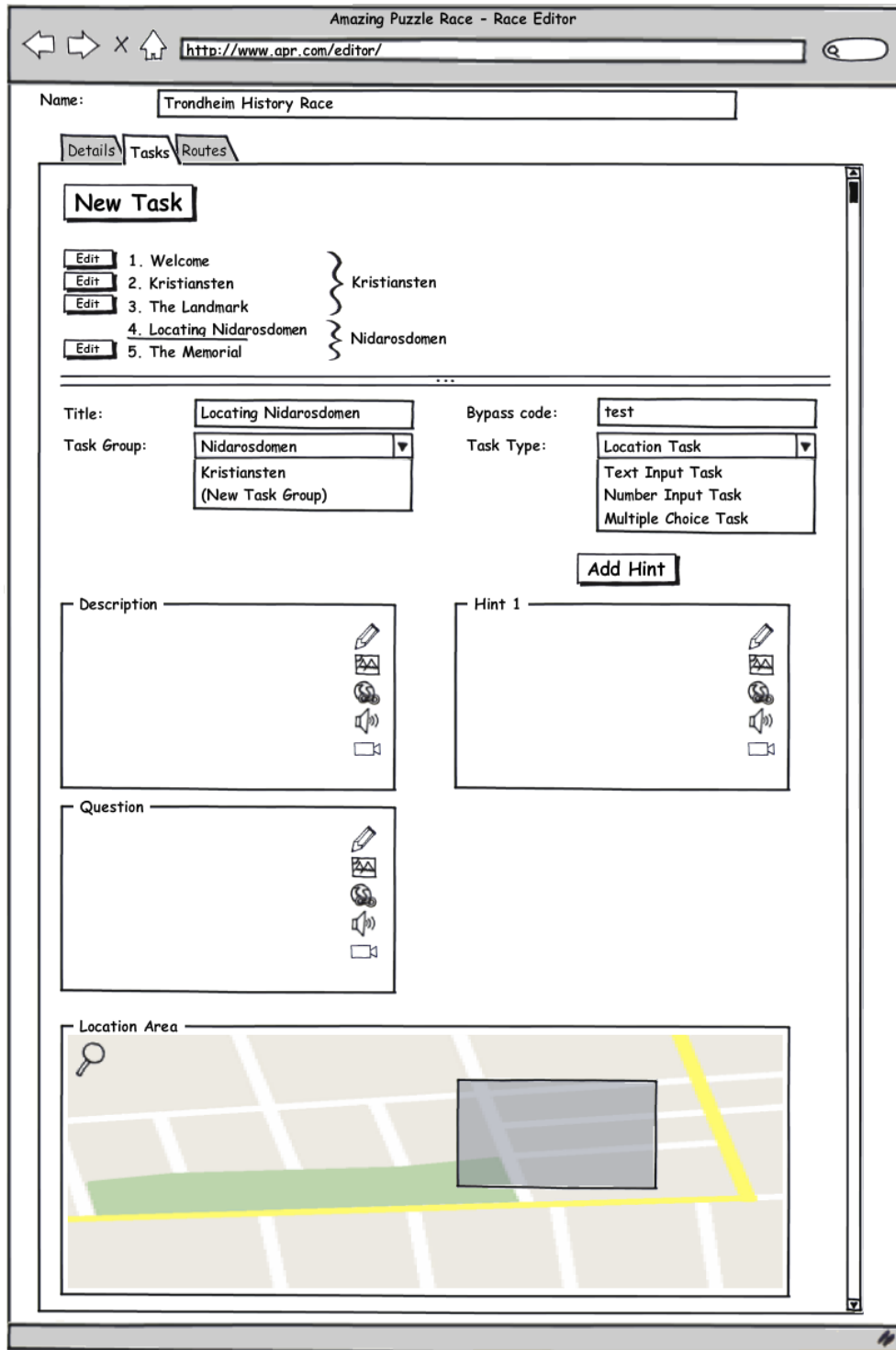


Figure 30.10: Editor Tasks Page

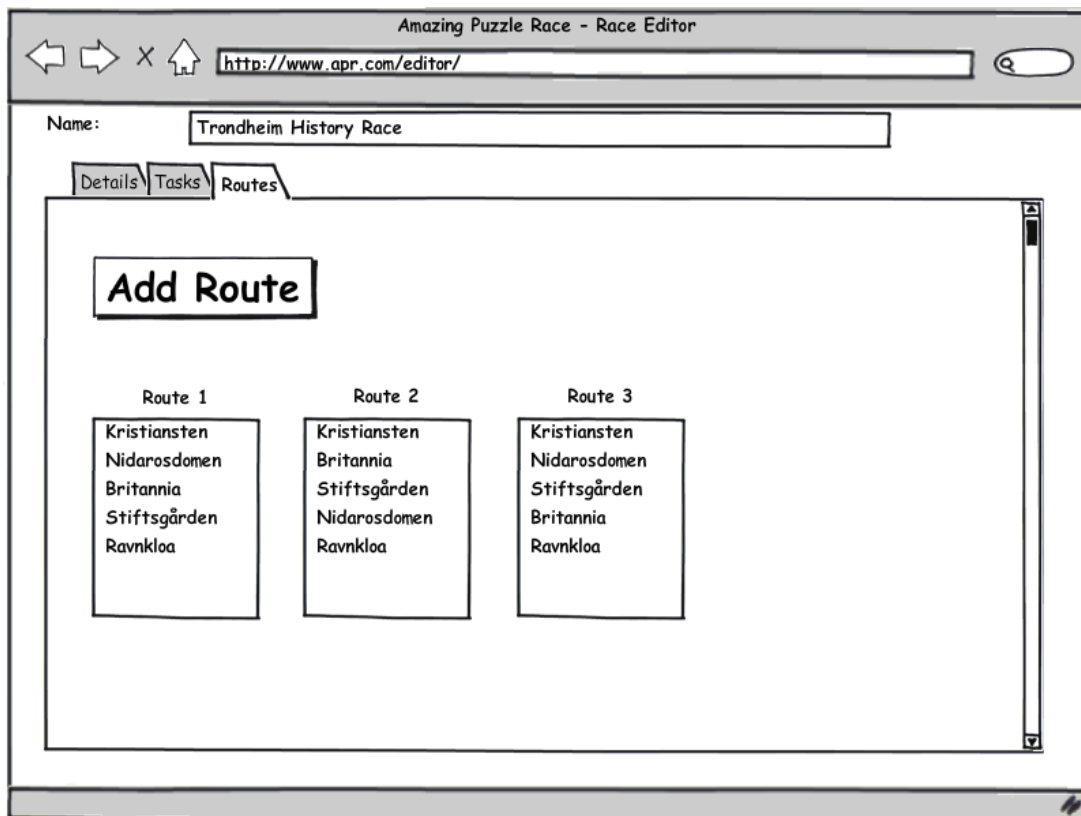


Figure 30.11: Editor Routes Page



# Part VII

## Conclusions





# CHAPTER 31

---

## Summary

---

In this thesis we created a prototype of a pervasive game in a lifelong learning context on Android. We also analyzed the effectiveness of the prototype and used the experiences drawn from it to design a platform to run knowledge competitions.

We achieved this by conducting a prestudy on the Android mobile phone operating system (including extension applications), the history of Trondheim, lifelong learning, pervasive games, and the use of pervasive games in a lifelong learning context. Lifelong learning (RQ1.1) is all learning activities undertaken throughout life to develop knowledge, skills, and competence. It emphasizes collaborative learning, individualized learning plans, educators as guides to sources of knowledge, while including everyone and offering education over a lifetime. Pervasive games (RQ1.2) are games that expand on the magic circle of play, by extending the spatial, social, or temporal dimension. The relationship between these two (RQ1.3) is complicated, but to name a few elements, both emphasize the social aspect, are motivating, and can occur over an extended period of time. To apply pervasive games to the context of lifelong learning (RQ2.1), one can use a subset of OECD's strategies for applying lifelong learning, cooperative play, spatial expansion with real-world objects, and both games and outsiders as guides to sources of knowledge. In addition, we discovered that there were several external applications and features of Android that could be utilized to make the game more pervasive. Also, we found that pervasive games, in theory, proved to be a suitable platform to support lifelong learning. For the design of our experiment (RQ2.2), we utilized a subset of our findings for RQ2.1.

Based on our results from RQ2.2, we developed a prototype on Android to run a puzzle race called "The Amazing City Game". The race consisted of completing different tasks related to the history of Trondheim, while traveling between many of the historical sites in the city. A demonstration race was conducted in early May with four groups of two students each, using the authors and supervisors of

this thesis as group observers. At the end of the race, the participants filled out a survey. Using the observations from the race and the results from the survey we found that the prototype was perceived as fun and educational, though not as usable as it could have been due to the lack of proper introductions. Also, construction of the race was challenging with many pitfalls concerning ambiguous tasks, use of language, and game balance.

Finally, we looked at how we could expand the experiment to be a platform to support lifelong learning (RQ3.1) and end up with a possible design for a platform for running knowledge competitions. To do this, we used the experiences from the development of the prototype, the results from the demonstration race, and some additional elements from RQ2.1 to design a cleaner and more complete framework. We expanded on the prototype by adding a puzzle race editor and an online community to encourage collaborative learning. The provided design also includes a refinement of the existing functionality and user interface, added requirements, and an extended discussion on a number of topics related to design choices. We believe that the concept of puzzle races in a lifelong learning context is an interesting concept that could have positive effects if utilized in the real world.

### Future Work

---

This thesis indicates that puzzle races on mobile phones are a fun and engaging way to achieve lifelong learning. We therefore feel that the concept has great potential, and suggest that any future projects are focused on the platform.

The next project could start with a refinement of the design outlined in the Platform part, leading to a more detailed requirement specification. An implementation of said requirements would follow, and finally the result of the implementation should be evaluated. The evaluation of the result should include a larger test population than was used in our project. This is necessary to gain more trust in the concept and the indications made in our evaluation.

Further down along the road, we suggest that the platform is extended by creating client applications for other mobile platforms such as iOS. The platform can also be extended with additional functionality by implementing support for new task types, knowledge competitions, and technology. For instance, when technological features such as near field communication (see Section 7.3.4) and gyroscopes (see Section 7.3.5) become more common in Android phones, new task types can be created based on these technologies. Other examples are image recognition or sound recognition.



---

## Bibliography

---

- [1] About 2D Code. <http://www.denso-wave.com/qrcode/aboutqr-e.html>. Accessed January 20th, 2011.
- [2] Alternate reality game. [http://en.wikipedia.org/wiki/Alternate\\_reality\\_game](http://en.wikipedia.org/wiki/Alternate_reality_game). Accessed June 15th, 2011.
- [3] App Store vs. Android Market. <http://www.brighthub.com/mobile/google-android/articles/63772.aspx>. Accessed Mars 20th, 2011.
- [4] Barcode Contents. <http://code.google.com/p/zxing/wiki/BarcodeContents>. Accessed January 20th, 2011.
- [5] The beast (game). [http://en.wikipedia.org/wiki/The\\_Beast\\_%28game%29](http://en.wikipedia.org/wiki/The_Beast_%28game%29). Accessed June 15th, 2011.
- [6] How Shazam Works. <http://everythingelsematterstoo.blogspot.com/2010/11/how-shazam-works.html>. Accessed Mars 20th, 2011.
- [7] Lifelong learning. [http://en.wikipedia.org/wiki/Lifelong\\_learning#Characteristics\\_of\\_Lifelong\\_Learning](http://en.wikipedia.org/wiki/Lifelong_learning#Characteristics_of_Lifelong_Learning). Accessed January 25th, 2011.
- [8] Measuring Usability. <http://www.measuringusability.com/sus.php>. Accessed Mars 20th, 2011.
- [9] Nexus s. <http://www.google.com/phone/detail/nexus-s>. Accessed January 27th, 2011.
- [10] QR Code Features. <http://www.denso-wave.com/qrcode/qrfeature-e.html>. Accessed January 20th, 2011.
- [11] QR Code Introduction. <http://www.denso-wave.com/qrcode/qrgene1-e.html>. Accessed January 20th, 2011.
- [12] QR Codes For Android Applications. [http://www.mediabistro.com/thinkmobile/qr-codes-for-android-applications\\_b5618](http://www.mediabistro.com/thinkmobile/qr-codes-for-android-applications_b5618). Accessed January 20th, 2011.
- [13] The Game. [http://seattletimes.nwsourc.com/html/pacificnw/2008177548\\_pacificpendgame14.html](http://seattletimes.nwsourc.com/html/pacificnw/2008177548_pacificpendgame14.html). Accessed Mars 20th, 2011.

- [14] The Game (treasure hunt). [http://en.wikipedia.org/wiki/The\\_Game\\_\(treasure\\_hunt\)](http://en.wikipedia.org/wiki/The_Game_(treasure_hunt)). Accessed Mars 20th, 2011.
- [15] Trondheim Today. <http://www.trondheim.com/content/1117609752/Trondheim-today>. Accessed Mars 20th, 2011.
- [16] Android 2.3 Platform and Updated SDK Tools, December 2010. <http://android-developers.blogspot.com/2010/12/android-23-platform-and-updated-sdk.html>. Accessed January 27th, 2011.
- [17] NFC-enabled mobile phones - the future of the check-in process?, August 2010. <http://www.check-in.aero/2010/08/nfc-enabled-mobile-phones-the-future-of-the-check-in-process/>. Accessed January 27th, 2011.
- [18] Android Boy. Smart Tools. <https://market.android.com/details?id=kr.aboy.tools>. Accessed February 7th, 2011.
- [19] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. 2004.
- [20] Aaron Bangor, Philip Kortum, and James Miller. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, 2008.
- [21] The World Bank. *Lifelong Learning in the Global Knowledge Economy: Challenges for Developing Countries*. The World Bank, 2003.
- [22] Bo Zhang Bohua Sun and Mohamed Toriq Khan. Modeling and Formulation of a Novel Microoptoelectromechanical Gyroscope. *Journal of Nanomaterials*, page 9, May 2008.
- [23] Bonnie Cha. T-Mobile MyTouch 3G Slide (black), May 2010. [http://reviews.cnet.com/smartphones/t-mobile-mytouch-3g/4505-6452\\_7-34083290.html#reviewPage1](http://reviews.cnet.com/smartphones/t-mobile-mytouch-3g/4505-6452_7-34083290.html#reviewPage1). Accessed January 27th, 2011.
- [24] Bonnie Cha and Kent German. HTC Droid Eris (Verizon Wireless), June 2009. [http://reviews.cnet.com/4505-6452\\_7-33799594.html#reviewPage1](http://reviews.cnet.com/4505-6452_7-33799594.html#reviewPage1). Accessed January 27th, 2011.
- [25] Terje T. V. Bratberg. *TRONDHEIM byleksikon*. Kunnskapsforlaget, 2008.
- [26] Britannia Hotel. Gjedden jonathan. <http://www.britannia.no/content.ap?thisId=1013179893>, 2011.

- [27] J. Brooke. SUS: a "quick and dirty" usability scale. *Usability evaluation in industry*, pages 189–194, 1996.
- [28] European Commission. A Memorandum on Lifelong Learning, 2000. Commission Staff Working Paper.
- [29] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems Concepts and Design*. Addison-Wesley, 2005.
- [30] National Research Council. *The Global Positioning System A Shared National Asset*. National Academy of Sciences, 1995.
- [31] Dimension Engineering LLC. A beginners guide to accelerometers. <http://www.dimensionengineering.com/accelerometers.htm>. Accessed January 28th, 2011.
- [32] Martin Flintham et. al. Where On-Line Meets On-The-Streets - Experiences With Mobile Mixed Reality Games.
- [33] Fong-Ling Fu, Rong-Chang Su and Sheng-Chin Yu. EGameFlow: A scale to measure learners enjoyment of e-learning games. *Elsevier Computers and Education*, 2008.
- [34] Organisation for Economic Co-operation and Development (OECD). Lifelong Learning. *OECD Observer*, 2004.
- [35] Gartner, Inc. Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent. <http://www.gartner.com/it/page.jsp?id=1466313>. Accessed January 26th, 2011.
- [36] Google. Google Goggles. <http://www.google.com/mobile/goggles/#text>. Accessed February 7th, 2011.
- [37] Google. Google Goggles: Overview. <http://www.google.com/support/mobile/bin/answer.py?hl=en&answer=166331>. Accessed February 7th, 2011.
- [38] Kamer Ali Yüksel Hamed Kitabdar and Mehran Roshandel. MagiTact: Interaction with Mobile Devices Based on Compass (Magnetic) Sensor. February 2010.
- [39] HTC. HTC - Products - HTC Aria - Specification. <http://www.htc.com/sea/product/aria/specification.html>. Accessed January 27th, 2011.
- [40] HTC. HTC - Products - HTC Desire - Specification. <http://www.htc.com/www/product/desire/specification.html>. Accessed January 27th, 2011.

- [41] HTC. HTC - Products - HTC Droid Incredible - Tech Specs. <http://www.htc.com/us/products/droid-incredible-verizon#tech-specs>. Accessed January 27th, 2011.
- [42] HTC. HTC - Products - HTC Hero - Specification. <http://www.htc.com/www/product/hero/specification.html>. Accessed January 27th, 2011.
- [43] HTC. HTC - Products - HTC Magic - Specification. <http://www.htc.com/www/product/magic/specification.html>. Accessed January 27th, 2011.
- [44] HTC. HTC - Products - Nexus One - Specification. <http://www.htc.com/www/product/nexusone/specification.html>. Accessed January 27th, 2011.
- [45] International Data Corporation (IDC). Android Drives Smartphone Growth in Western Europe and Ramps Up to Overtake Symbian and Become Biggest Smartphone OS in 2011, Says IDC. <http://www.idc.com/about/viewpressrelease.jsp?containerId=prUK22637010>. Accessed January 26th, 2011.
- [46] James Doscher. Accelerometer Design and Applications. <http://www.meche.net/MAE%20334/Lab%205/Lab%205%20Report/sensor%20comparisons.pdf>. Accessed January 28th, 2011.
- [47] Jason Hiner. Review: Samsung Vibrant is the flagship of the Galaxy S fleet | TechRepublic. <http://www.techrepublic.com/blog/hiner/review-samsung-vibrant-is-the-flagship-of-the-galaxy-s-fleet/6133>. Accessed January 27th, 2011.
- [48] Gang Pan Jiahui Wu, Shijian Li Daqing Zhang, and Zhaohui Wu. Magic-Phone: Pointing and Interacting. September 2010.
- [49] Jin Jing, Abdelsalam Helal, and Ahmed Elmagarmid. Client-Server Computing in Mobile Environments. 1999.
- [50] Kent German. Samsung Moment (Sprint), October 2009. [http://reviews.cnet.com/4505-6452\\_7-33775546.html#reviewPage1](http://reviews.cnet.com/4505-6452_7-33775546.html#reviewPage1). Accessed January 27th, 2011.
- [51] Layar. What is Layar and what can I do with it. <http://site.layar.com/create>. Accessed February 8th, 2011.
- [52] Machine Design. Proximity Sensor. [http://www.sensors-transducers.machinedesign.com/guiEdits/Content/bdeeee4/bdeeee4\\_7.aspx](http://www.sensors-transducers.machinedesign.com/guiEdits/Content/bdeeee4/bdeeee4_7.aspx). Accessed January 28th, 2011.
- [53] George M. Marakas. *System Analysis and Design An Active Approach*. McGraw-Hill Irwin, 2006.



- [54] Markus Montola. Exploring the Edge of the Magic Circle: Defining Pervasive Games. *DAC 2005 conference*, 2005.
- [55] Tracey J. Mehigan. Harnessing Accelerometer Technology for Inclusive Mobile Learning. 2009.
- [56] Millennial Media's Mobile Mix. The Mobile Device Index, December 2010. <http://www.millennialmedia.com/wp-content/images/mobilemix/MM-MobileMix-Dec2010.pdf>. Accessed January 31th, 2011.
- [57] Markus Montola. Tangible Pleasures of Pervasive Role-Playing. 2007.
- [58] Markus Montola, Jaakko Stenros, and Annika Waern. *Pervasive Games: Theory and Design*. Morgan Kaufmann, 2009.
- [59] Motorola. DROID 2 by Motorola - Android phone - Tech Specs - Motorola Mobility, Inc. USA. <http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/ci.Motorola-DROID-2-US-EN.alt>. Accessed January 27th, 2011.
- [60] Motorola. DROID by Motorola - Android phone - Tech Specs - Motorola Mobility, Inc. USA. <http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/ci.Motorola-DROID-US-EN.alt>. Accessed January 27th, 2011.
- [61] Motorola. DROID X by Motorola - Android phone - Tech Specs - Motorola Mobility, Inc. USA. <http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/ci.Motorola-DROID-X-US-EN.alt>. Accessed January 27th, 2011.
- [62] Nicole Lee. LG Ally (Verizon Wireless), May 2010. [http://reviews.cnet.com/smartphones/lg-ally-verizon-wireless/4505-6452\\_7-34093840.html#reviewPage1](http://reviews.cnet.com/smartphones/lg-ally-verizon-wireless/4505-6452_7-34093840.html#reviewPage1). Accessed January 27th, 2011.
- [63] Kris Antoni Hadiputra Nurwono and Raymondus Kosala. Color Quick Response Code for Mobile Content Distribution. *Proceedings of MoMM2009*, 2009.
- [64] Open Handset Alliance. Android Overview. [http://www.openhandsetalliance.com/android\\_overview.html](http://www.openhandsetalliance.com/android_overview.html). Accessed February 3rd, 2011.
- [65] Maja Pivec and Olga Dziabenko. Game-based learning in universities and lifelong learning: Unigame. *Social Skills and Knowledge Training. Game Concept. Vol, 10:4-16*, 2004.

- [66] Priya Ganapati. Showdown: iPhone 4 vs. HTC Evo 4G, June 2010. <http://www.wired.com/gadgetlab/2010/06/iphone-4-vs-htc-evo-4g/>. Accessed January 27th, 2011.
- [67] Tommi Kallonen Raine Kelkka and Jouni Ikonen. Remote Identification and Information Processing with a Near field Communication Compatible Mobile Phone. In *International Conference on Computer Systems and Technologies*, 2009.
- [68] ShopSavvy, Inc. ShopSavvy 4. <https://market.android.com/details?id=com.biggu.shopsavvy>. Accessed February 8th, 2011.
- [69] Alok Sinha. Client-Server Computing. *Current Technology Review*, 1992.
- [70] Erkki Sutinen Teemi H. Laine, Carolina Islas Sedano and Mike Joy. Viable and Portable Architecture for Pervasive Learning Spaces. 2009.
- [71] The Entertainment Software Association. Industry facts. <http://www.theesa.com/facts/index.asp>, 2011.
- [72] The NPD Group, Inc. The NPD Group: Android Extends its Smartphone Market Share in the Third Quarter of 2010. [http://www.npd.com/press/releases/press\\_101101.html](http://www.npd.com/press/releases/press_101101.html). Accessed January 26th, 2011.
- [73] Trondheim Tourist Information. Bakklundet. <http://www.trondheim.com/bakklundet/>. Accessed February 17th, 2011.
- [74] Trondheim Tourist Information. Munkholmen - prison island, fort and former monastery. [http://www.trondheim.com/monks\\_island/](http://www.trondheim.com/monks_island/). Accessed February 17th, 2011.
- [75] Trondheim Tourist Information. Nidaros Cathedral. [http://www.trondheim.com/nidaros\\_cathedral/](http://www.trondheim.com/nidaros_cathedral/). Accessed February 17th, 2011.
- [76] Trondheim Tourist Information. Stiftsgården - the Royal Residence in Trondheim. <http://www.trondheim.com/royalresidence/>. Accessed February 17th, 2011.
- [77] Trondheim Tourist Information. Torvet - Market Square. <http://www.trondheim.com/content/1117629989/Torvet---Market-Square>. Accessed February 17th, 2011.
- [78] Trondheim Tourist Information. Vår Frue Kirke. <http://www.trondheim.com/content/1117629991/Var-Frue-Kirke>. Accessed February 17th, 2011.

- [79] T. Tullis and J. N. Stetson. A comparison of questionnaires for assessing website usability. proceedings of the usability professionals association (upa). *Usability Professionals Association Conference*, 2004.
- [80] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. Analytic Modeling of Multitier Internet Applications. 2007.
- [81] Vishay Intertechnology, Inc. Ambient Light Sensors. <http://www.vishay.com/docs/49670/p10366.pdf>. Accessed January 28th, 2011.
- [82] Wi-Fi Alliance. The How and Why of Wi-Fi. [http://www.wi-fi.org/knowledge\\_center/kc-howandwhyofwi-fi](http://www.wi-fi.org/knowledge_center/kc-howandwhyofwi-fi). Accessed February 3rd, 2011.
- [83] Wikipedia. Britannia hotel. [http://no.wikipedia.org/wiki/Britannia\\_Hotel](http://no.wikipedia.org/wiki/Britannia_Hotel), 2011.
- [84] Ming-Jhe Yang, Jui-Hung Chen, Te-Hua Wang, Louis R. Chao, and Timothy K. Shih. To construct the outdoor experience game-based learning system by integrating ubiquitous technologies. In *Proceedings of the 2009 workshop on Ambient media computing, AMC '09*, pages 77–82, New York, NY, USA, 2009. ACM.
- [85] Marvin V. Zerlkowitz and Dolores R. Wallace. Experimental Models for Validating Technology. *IEEE Computer*, 31(5):23–31, 1998.



**Part VIII**  
**Appendices**



# APPENDIX A

---

## Tasks

---

This appendix contains tables that lists all of the tasks in the game.

<b>Name</b>	<b>Welcome</b>
Id	1
Group Id	Welcome
Type	Barcode
Description	Welcome to The Amazing City Game. Please scan a barcode in order to retrieve a game route.
Hints	No hints
Solution	Several solutions (random keywords), one for each route.
Routes	<ol style="list-style-type: none"> <li>1. Route 1: Welcome - Kristiansten - Nidarosdomen - Britannia - Gamle Bybro - Vår Frue Kirke - Stiftsgården - Torvet - Ravnkloa</li> <li>2. Route 2: Welcome - Kristiansten - Britannia - Vår Frue Kirke - Gamle Bybro - Nidarosdomen - Stiftsgården - Torvet - Ravnkloa</li> <li>3. Route 3: Welcome - Kristiansten - Gamle Bybro - Britannia - Nidarosdomen - Stiftsgården - Torvet - Vår Frue Kirke - Ravnkloa</li> <li>4. Route 4: Welcome - Kristiansten - Vår Frue Kirke - Stiftsgården - Nidarosdomen - Gamle Bybro - Britannia - Torvet - Ravnkloa</li> </ol>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 0 minutes</li> <li>• Wrong answer: 0 minutes</li> </ul>
Difficulty	Easy
Technologies	<ul style="list-style-type: none"> <li>• Barcode scanner: In order to find the keyword to get a route.</li> </ul>

Table A.1: Task - Welcome



<b>Name</b>	<b>Kristiansten Fortress</b>
Id	2
Group Id	Kristiansten 1
Type	Checkbox Task
Description	Kristiansten Fortress is built on the strategically important height, Erlendshaug. There were initially plans made for a fortress at Erlendshaug already in 1676, but only after the great city fire in 1681 were the plans set in motion. The fortress had an important role to play in the city's fortification plan designed by Cignon and Coucheron, but was decommissioned in 1816. <b>WARNING!</b> There is a long way down, watch your step! Which of the following objects are at Kristiansten fortress?
Alternatives	<ol style="list-style-type: none"> <li>1. Cannons</li> <li>2. Gallows</li> <li>3. Barbed wire</li> <li>4. Memorial to fallen resistance fighters</li> <li>5. Moat</li> </ol>
Hints	<ol style="list-style-type: none"> <li>1. There are no gallows.</li> <li>2. There is no moat.</li> </ol>
Solution	<ul style="list-style-type: none"> <li>• Cannons</li> <li>• Barbed wire</li> <li>• Memorial to fallen resistance fighters</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	None in particular.

Table A.2: Task - Kristiansten Fortress

<b>Name</b>	<b>The Landmark</b>
Id	3
Group Id	Kristiansten 2
Type	Open Task
Description	In this task we want you to find a famous building in Trondheim. This task will require some mathematics: $X$ is the number of cannons standing on line next to the donjon. $Y$ is the number of arched openings in the donjon. $Z = (X + Y) * 8,4375$ . Stand on the brick wall above the western entrance (between the donjon and the cannons, right in front of some barbed wire) and face north. Then turn $Z$ degrees. Which building are you looking directly at?
Hints	<ol style="list-style-type: none"> <li>1. Turn west.</li> </ol>
Solution	Nidarosdomen
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Medium
Technologies	<ul style="list-style-type: none"> <li>• Compass: To determine which direction the user should look in based on the calculated angle.</li> </ul>

Table A.3: Task - The Landmark


<b>Name</b>	<b>Locating Nidarosdomen</b>
Id	4
Group Id	Nidarosdomen 1
Type	Location Task
Description	Travel to the place the following image from 1857 depicts. 
Hints	<ol style="list-style-type: none"> <li>1. Olav den Hellige is buried at the location.</li> <li>2. The location is Nidarosdomen.</li> </ol>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.42701614318537</li> <li>• Longitude: 10.396628379821777</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.4: Task - Locating Nidarosdomen

<b>Name</b>	<b>The War Memorial</b>
Id	5
Group Id	Nidarosdomen 2
Type	Open Task
Description	On the north side of Nidaros Cathedral there is a war memorial. This particular war memorial was raised in the memory of the people that gave their lives for Norway during World War II. This memorial also carries an inscription. What does it say?
Hints	<ol style="list-style-type: none"> <li>1. The statue is close to a path.</li> <li>2. It is a statue of a woman holding a child.</li> </ol>
Solution	Til minne om frihetskampens ofre.
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	<ul style="list-style-type: none"> <li>• Compass: One way to figure out which side of Nidaros Cathedral they are.</li> </ul>

Table A.5: Task - The War Memorial

<b>Name</b>	<b>The Main Facade</b>
Id	6
Group Id	Nidarosdomen 3
Type	Multiple Choice Task
Description	Nidaros Cathedral's Westfront is the cathedral's main facade. It went through a restoration period between 1901 and 1969. A large number of figures are placed in alcoves. The 8th figure from the right on the top row is carrying what item?
Alternatives	<ol style="list-style-type: none"> <li>1. A cross.</li> <li>2. An axe.</li> <li>3. A book.</li> <li>4. A scepter.</li> </ol>
Hints	<ol style="list-style-type: none"> <li>1. The item is not a cross.</li> <li>2. The item is not an axe.</li> <li>3. The item is not a book.</li> <li>4. The correct answer is a scepter!</li> </ol>
Solution	A scepter
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Medium
Technologies	<ul style="list-style-type: none"> <li>• Camera: The user can take a picture of the statue and zoom in to see which item it is holding.</li> <li>• Compass: One way to figure out which side of Nidaros Cathedral they are.</li> </ul>

Table A.6: Task - The Main Facade


<b>Name</b>	<b>Locating Stiftsgården</b>
Id	7
Group Id	Stiftsgården 1
Type	Location Task
Description	Travel to the royal residence of the city.
Hints	<ol style="list-style-type: none"> <li>1. It is the largest wooden building in the city.</li> <li>2. An image of the location.</li> </ol>  <ol style="list-style-type: none"> <li>3. The location is Stiftsgården in Munkegata.</li> </ol>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.43151283743752</li> <li>• Longitude: 10.394783020019531</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Hard
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.7: Task - Locating Stiftsgården

<b>Name</b>	<b>The Iron Gate</b>
Id	8
Group Id	Stiftsgården 2
Type	Open Task
Description	Stiftsgården is possibly the largest wooden building in the Nordic countries. It is over 4000 square meters and has 140 rooms. Construction started in 1774 and it was originally designed as a private residence, but in 1906 it became the royal residence in Trondheim. Behind the building there is a park that is open to the public. In what year was the gate erected?
Hints	1. The answer is written on the Iron Gate of the park.
Solution	1906
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Medium
Technologies	None in particular.

Table A.8: Task - The Iron Gate


<b>Name</b>	<b>Locating Gamle Bybro</b>
Id	9
Group Id	Gamle Bybro 1
Type	Location Task
Description	Travel to the bridge that has portals.
Hints	<p>1. An image of the location.</p>  <p>2. The destination is Gamle Bybro between Midtbyen and Bakklandet.</p>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.42823036816747</li> <li>• Longitude: 10.401585102081299</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.9: Task - Locating Gamle Bybro



<b>Name</b>	<b>Bridge Building</b>
Id	10
Group Id	Gamle Bybro 2
Type	Open Task
Description	The Old Town Bridge was originally built around 1681 as a result of Cicignon's city plan. It was planned to cross the river further south, but to increase the defensive capabilities of the city it was built where it stands today. This location would give the city adequate defense if it was attacked from Vollan and Øvre Bakklundet. Later the bridge was rebuilt. In what year was the the bridge rebuilt?
Hints	1. One possible source of information could be Wikipedia.
Solution	1861
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	None in particular.

Table A.10: Task - Bridge Building

<b>Name</b>	<b>Lykkens Portal</b>
Id	11
Group Id	Gamle Bybro 3
Type	Open Task
Description	What is the name of the bridge portal?
Hints	<ol style="list-style-type: none"><li>1. The answer contains two words.</li><li>2. "Nidelven stille og vakker du er".</li></ol>
Solution	Lykkens Portal
Penalties	<ul style="list-style-type: none"><li>• Hint usage: 2 minutes</li><li>• Wrong answer: 4 minutes</li></ul>
Difficulty	Easy
Technologies	None in particular.

Table A.11: Task - Lykkens Portal


<b>Name</b>	<b>Locating Britannia</b>
Id	12
Group Id	Britannia 1
Type	Location Task
Description	Travel to the place where the pike Jonathan was executed during World War 2.
Hints	<p>1. An old image of the location.</p>  <p>2. The location is Britannia Hotel.</p>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.43179115818105</li> <li>• Longitude: 10.398699045181274</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Hard
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.12: Task - Locating Britannia

<b>Name</b>	<b>The Production Company</b>
Id	13
Group Id	Britannia 2
Type	Shazam Challenge
Description	Trondheim has been a cultural center for the Trøndelag region for a long time. Part of this has been because of the city being a religious center, which can be seen and heard in the annual festival Olavsfestivalen that has a strong focus on church music. But the city has also fostered musical successes in the modern age, and in particular a duo that amongst other things is connected to the musical piece you can hear when pressing the Play button. What is the name of the duo?
Hints	<ol style="list-style-type: none"> <li>1. The song you just heard was Curious by Noora Noor (which can be checked by using the Android app Shazam), a Norwegian soul artist. But the duo has also worked with international artists, such as those behind this song.</li> <li>2. The song you heard was S Club 7 and their song S Club. The name of the duo is actually the name of their production company, as the duo works with writing and producing songs for other artists. S Club was one of their first hits abroad, but they have been even more successful than this, and have actually produced numerous number 1 hits in the USA. One of those hits are the following song.</li> </ol>
Songs	<ul style="list-style-type: none"> <li>• Song for task description: Noora Noor - Curious</li> <li>• Song for Hint 1: S Club 7 - S Club</li> <li>• Song for Hint 2: Beyonce - Irresistible</li> </ul>
Solution	Stargate
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Medium
Technologies	<ul style="list-style-type: none"> <li>• Shazam: Look up songs by listening to short segments of music.</li> </ul>

Table A.13: Task - The Production Company


<b>Name</b>	<b>The Flag</b>
Id	14
Group Id	Britannia 3
Type	Open Task
Description	<p>The following image depicts a specific flower. A large version of the image is on one of the sheets of paper you have brought</p>  <p>with you.</p>
Hints	<ol style="list-style-type: none"> <li>1. You can use Google Goggles to determine which flag it is.</li> <li>2. The flower is a type of rose.</li> <li>3. The fruit of the flower has a powder inside that is very itchy.</li> </ol>
Solution	Nyperose
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 2 minutes</li> </ul>
Difficulty	Medium
Technologies	<ul style="list-style-type: none"> <li>• Google Goggles: Used on the picture to search the Internet.</li> </ul>

Table A.14: Task - The Flag


<b>Name</b>	<b>Locating Vår Frue Kirke</b>
Id	15
Group Id	Vår Frue Kirke 1
Type	Location Task
Description	Travel to the building that Bjørn Sigvardsson is the architect of.
Hints	<ol style="list-style-type: none"> <li>1. It is a stone church.</li> <li>2. It was built in the middle ages.</li> <li>3. St. Mary.</li> <li>4. An image of the location.</li> </ol>  <ol style="list-style-type: none"> <li>5. The location is Vår Frue Kirke.</li> </ol>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.43018357866649</li> <li>• Longitude: 10.39829134941101</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Hard
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.15: Task - Locating Vår Frue Kirke

<b>Name</b>	<b>Vår Frue Kirke Facts</b>
Id	16
Group Id	Vår Frue Kirke 2
Type	Open Task
Description	Vår Frue Kirke is a stone church from the middle ages that has been rebuilt several times after burning down in the many city fires. In which European capital were two of todays church clocks made?
Hints	<ol style="list-style-type: none"> <li>1. One of the church clocks is located just inside the main entrance.</li> <li>2. The clocks was forged by Jan Albert de Grave.</li> <li>3. It is the capital of the Netherlands.</li> </ol>
Solution	<ul style="list-style-type: none"> <li>• Amsterdam</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	None in particular.

Table A.16: Task - Vår Frue Kirke Facts


<b>Name</b>	<b>Locating Torvet</b>
Id	17
Group Id	Torvet 1
Type	Location Task
Description	Travel to the statue that Wilhelm Robert Rasmussen created.
Hints	<ol style="list-style-type: none"> <li>1. It is a statue of the founder of Trondheim.</li> <li>2. The statue is very tall.</li> <li>3. An image of the location.</li> </ol>  <ol style="list-style-type: none"> <li>4. The location is torvet.</li> </ol>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.43048110764693</li> <li>• Longitude: 10.39505660533905</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Medium
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.17: Task - Locating Torvet



<b>Name</b>	<b>Torvet Shopsavvy</b>
Id	18
Group Id	Torvet 2
Type	ShopSavvy Challenge
Description	Trondheim has had a town square and marketplace since the early 12th century. After the great fire in 1681, the marketplace is situated in the heart of Trondheim, which corresponds to Cicignon's city plan. We want you to find the name of a prominent person in the city's history, by answering the following questions and following the instructions on paper for the ShopSavvy Challenge.
Questions	<ol style="list-style-type: none"> <li>1. How many points does the star on the ground have?</li> <li>2. When was the statue erected?</li> <li>3. When was Trondheim established?</li> <li>4. When did Olav Tryggvason die?</li> </ol>
Hints	<ol style="list-style-type: none"> <li>1. For Q2: <a href="http://www.trondheim.com">www.trondheim.com</a></li> <li>2. For Q3: The founder is Olav Tryggvason.</li> <li>3. For Q4: It is a round number.</li> </ol>
Answers	<ol style="list-style-type: none"> <li>1. 16 - Letter: V</li> <li>2. 1921 - Letter: L</li> <li>3. 997 - Letter: O</li> <li>4. 1000 - Letter: A</li> </ol>
Solution	OLAV
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Hard
Technologies	<ul style="list-style-type: none"> <li>• ShopSavvy: Look up one dimensional barcodes for commercial products.</li> </ul>

Table A.18: Task - Torvet Shopsavvy

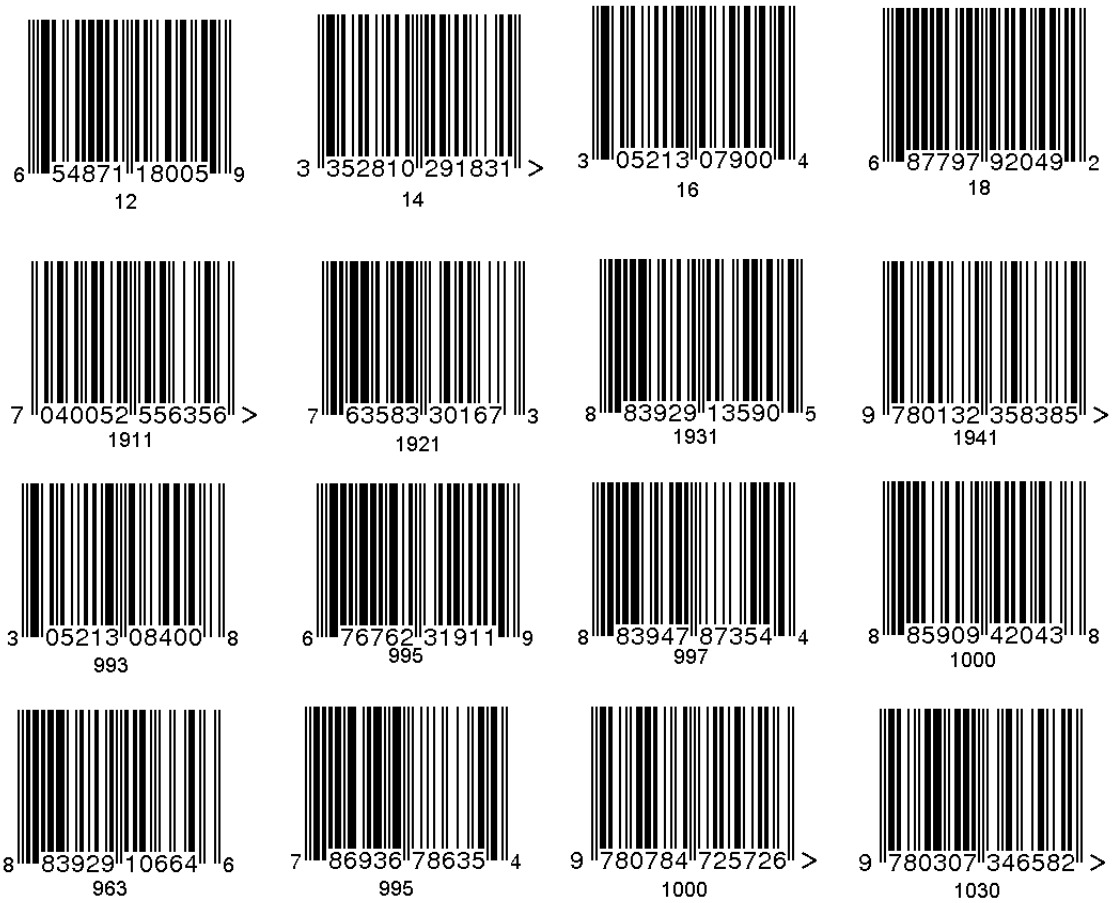


Figure A.1: Torvet Shopsavvy Barcodes


<b>Name</b>	<b>Locating Ravnkloa</b>
Id	19
Group Id	Ravnkloa 1
Type	Location Task
Description	Travel to the fish market.
Hints	<p>1. An image of the location.</p>  <p>2. The destination is Ravnkloa in Munkegata.</p>
Coordinates	<ul style="list-style-type: none"> <li>• Latitude: 63.43384968800512</li> <li>• Longitude: 10.39306640625</li> </ul>
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	<ul style="list-style-type: none"> <li>• GPS: The task is solved when the user get to the correct location.</li> </ul>

Table A.19: Task - Locating Ravnkloa

<b>Name</b>	<b>The Writer</b>
Id	20
Group Id	Ravnkloa 2
Type	Multiple Choice Task
Description	The statue at Ravnkloa depicts a man from the book "The Last of the Vikings" about local fishermen. Who wrote that book?
Alternatives	<ol style="list-style-type: none"> <li>1. Olav Duun</li> <li>2. Henrik Ibsen</li> <li>3. Magnhild Haalke</li> <li>4. Johan Bojer</li> </ol>
Hints	<ol style="list-style-type: none"> <li>1. The book was written in 1921.</li> </ol>
Solution	Johan Bojer
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	None in particular.

Table A.20: Task - The Writer

<b>Name</b>	<b>The Avenue</b>
Id	21
Group Id	Ravnkloa 3
Type	Multiple Choice Task
Description	Munkegata is one of the very few Avenues in Norway. How wide is Munkegata?
Alternatives	<ol style="list-style-type: none"> <li>1. 30,00 meters</li> <li>2. 33,33 meters</li> <li>3. 37,65 meters</li> <li>4. 40,00 meters</li> </ol>
Hints	<ol style="list-style-type: none"> <li>1. When Cicignon made the city plan he made Munkegata '60 ell' wide.</li> <li>2. 1 Norwegian ell in 1681 = 62,75 centimeters.</li> </ol>
Solution	37,65 meters
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Medium
Technologies	None in particular.

Table A.21: Task - The Avenue

<b>Name</b>	<b>The End</b>
Id	22
Group Id	Ravnkloa 4
Type	Open Task
Description	The last question of the race concerns Cicignons town plan. He made it so one of the streets, Munkegata, lies in the axis between Nidarosdomen and another famous landmark. Which landmark?
Hints	1. The landmark is an island.
Solution	Munkholmen
Penalties	<ul style="list-style-type: none"> <li>• Hint usage: 2 minutes</li> <li>• Wrong answer: 4 minutes</li> </ul>
Difficulty	Easy
Technologies	None in particular.

Table A.22: Task - The End

# APPENDIX B

---

## Use Cases

---

This appendix contains examples of use cases for the task types in the game.

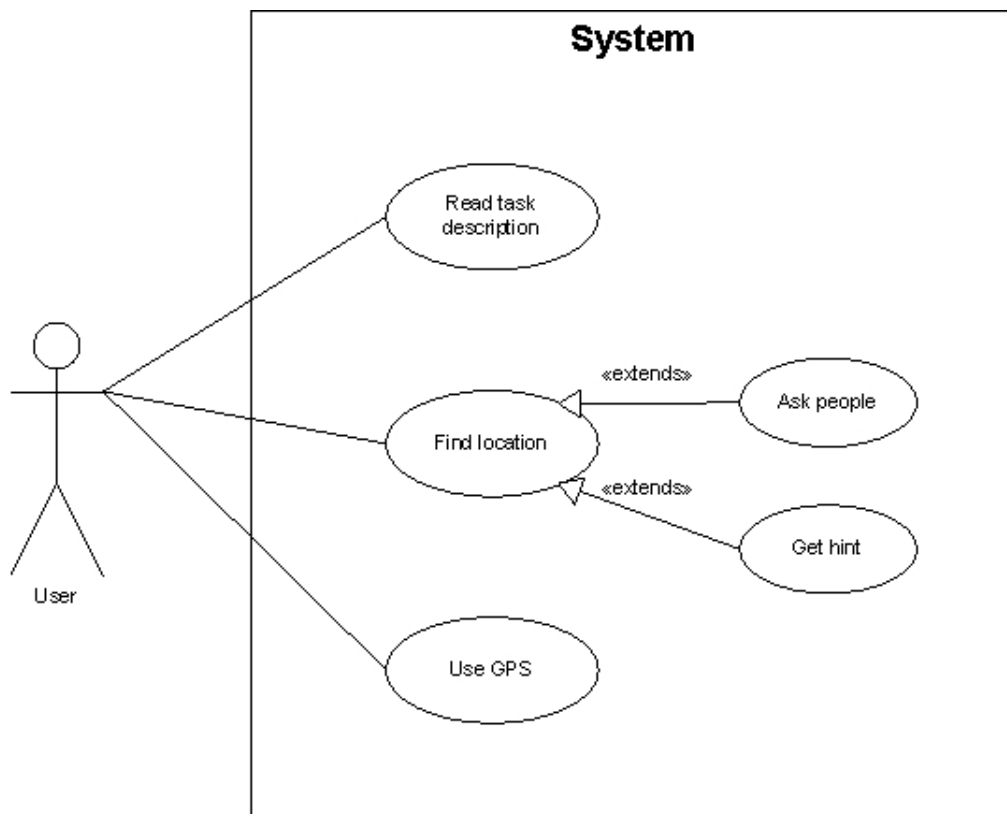


Figure B.1: Use Case: Location Task

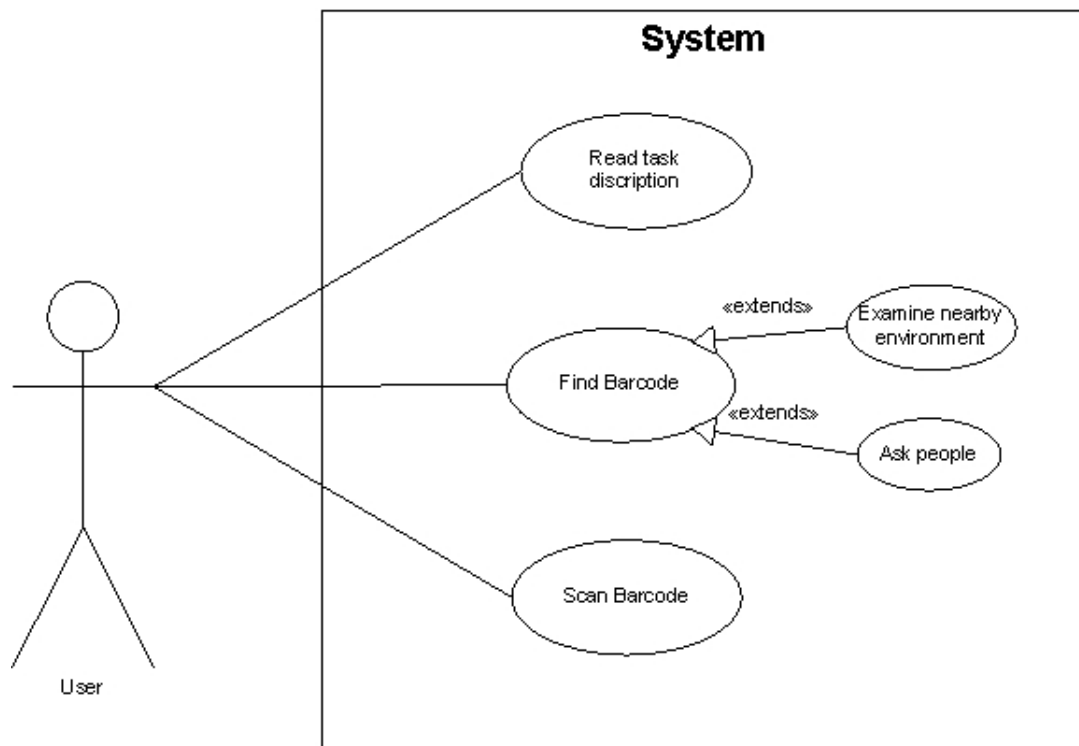


Figure B.2: Use Case: Barcode Task



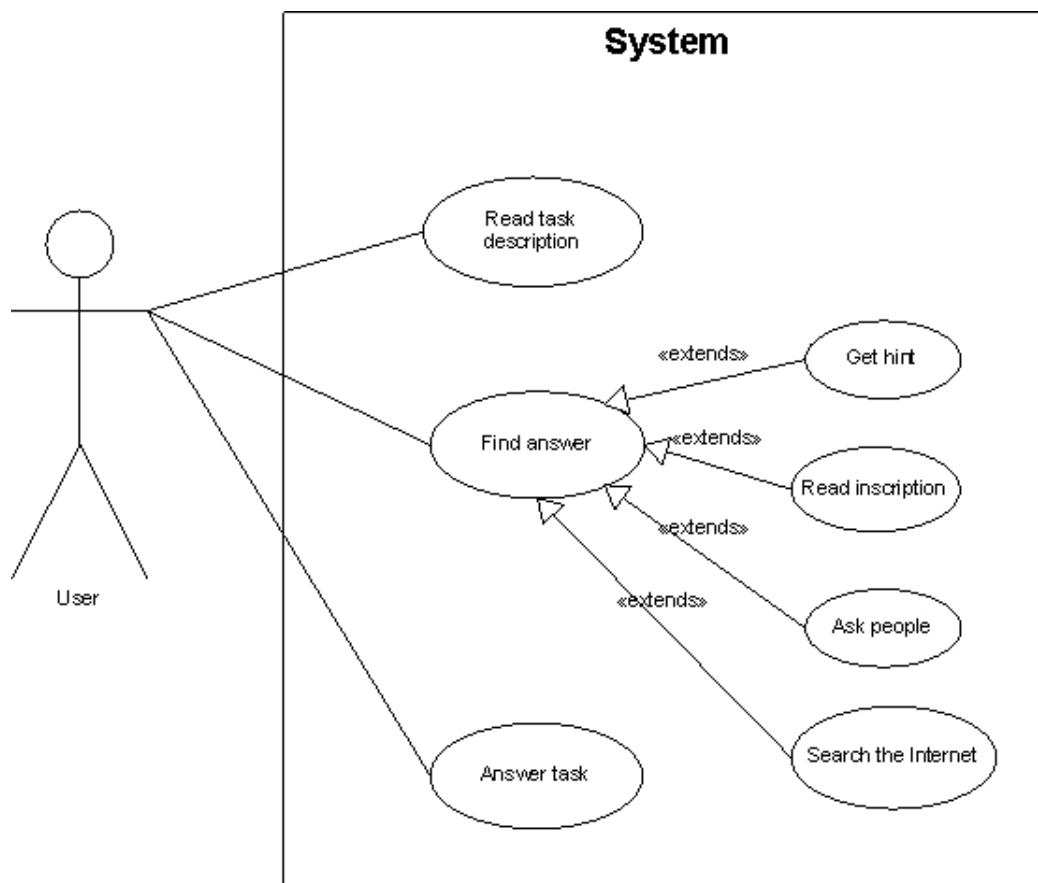


Figure B.3: Use Case: Open Task - The Iron Gate

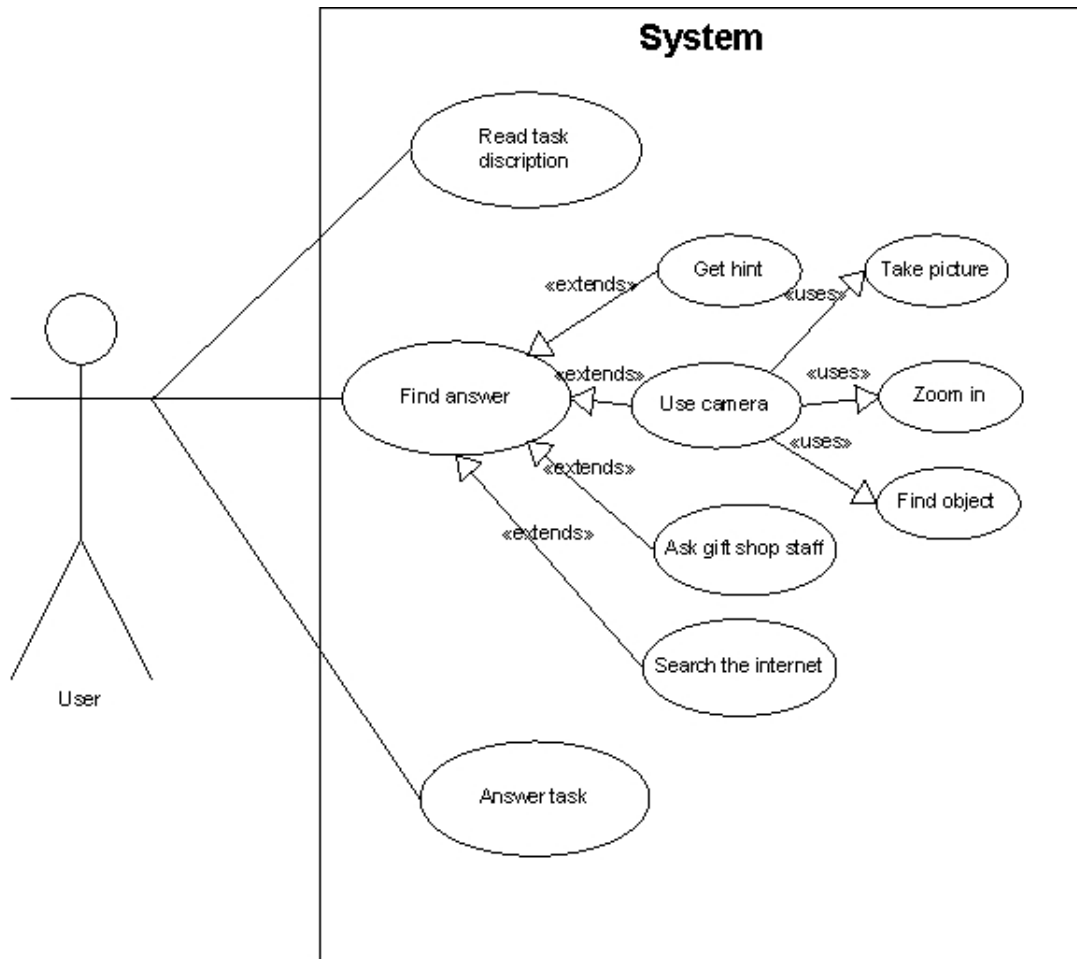


Figure B.4: Use Case: Multiple Choice Task - The Main Facade

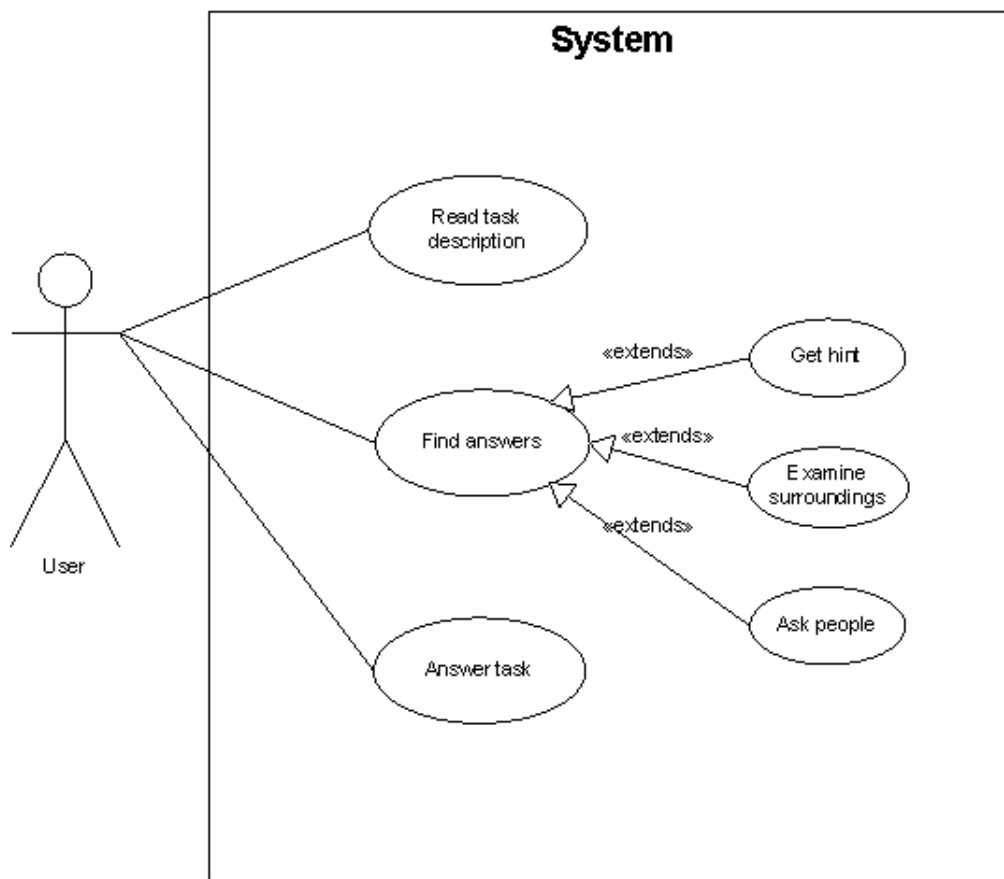


Figure B.5: Use Case: Checkbox Task - Kristiansten Fortress

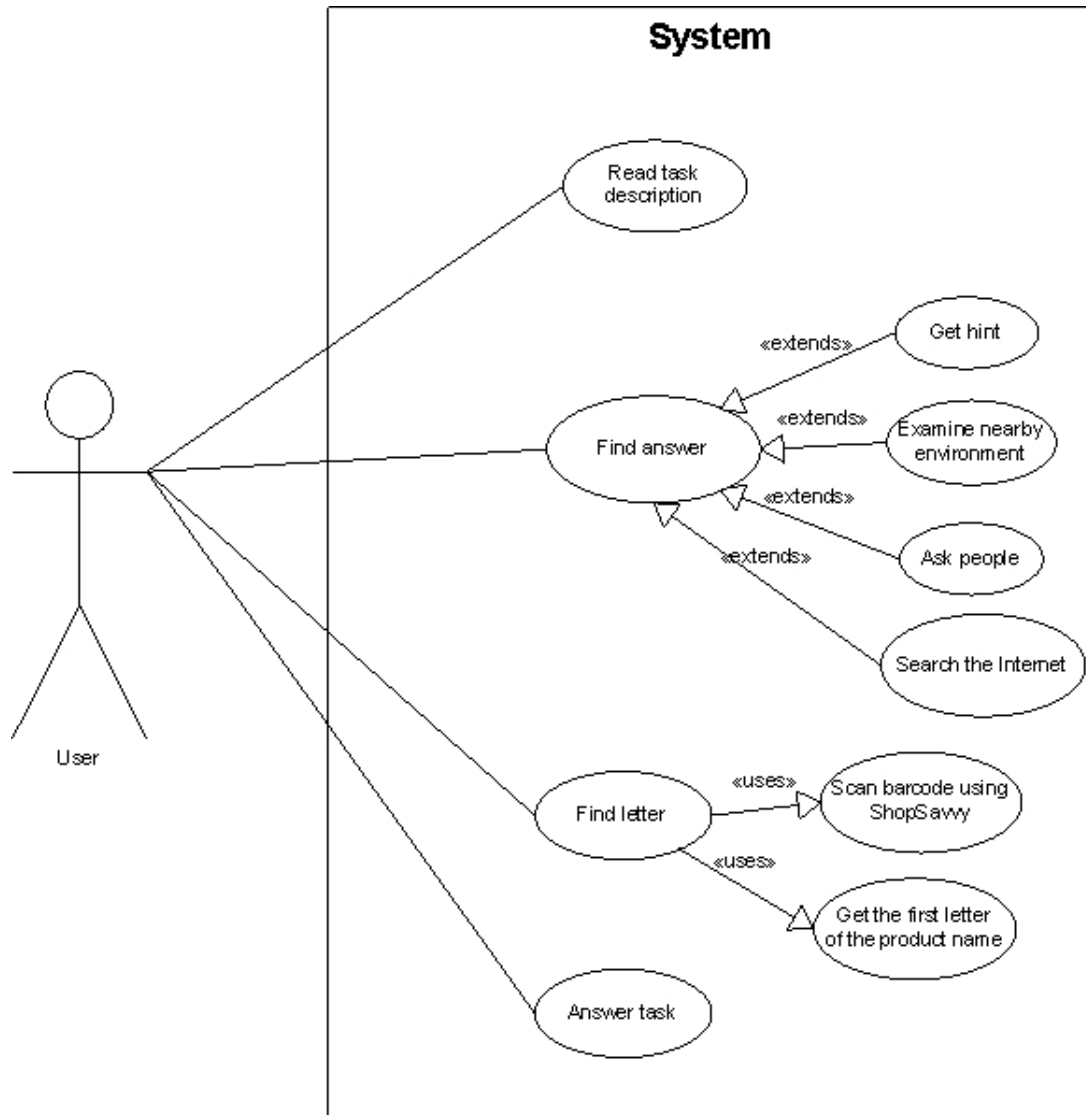


Figure B.6: Use Case: Shopsavvy Challenge - Torvet Shopsavvy

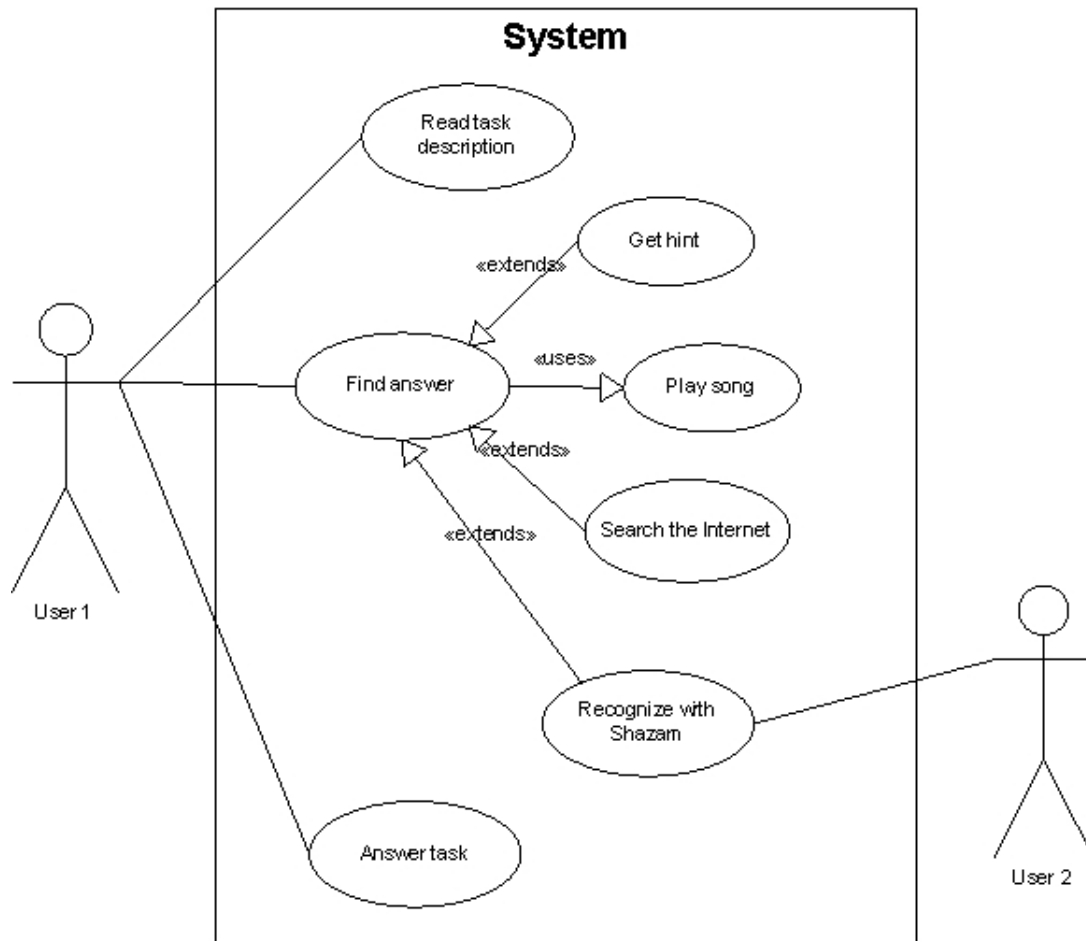


Figure B.7: Use Case: Shazam Challenge - The Production Company



## APPENDIX C

---

### Class Diagram

---

This appendix contains a detailed class diagram of the prototype.

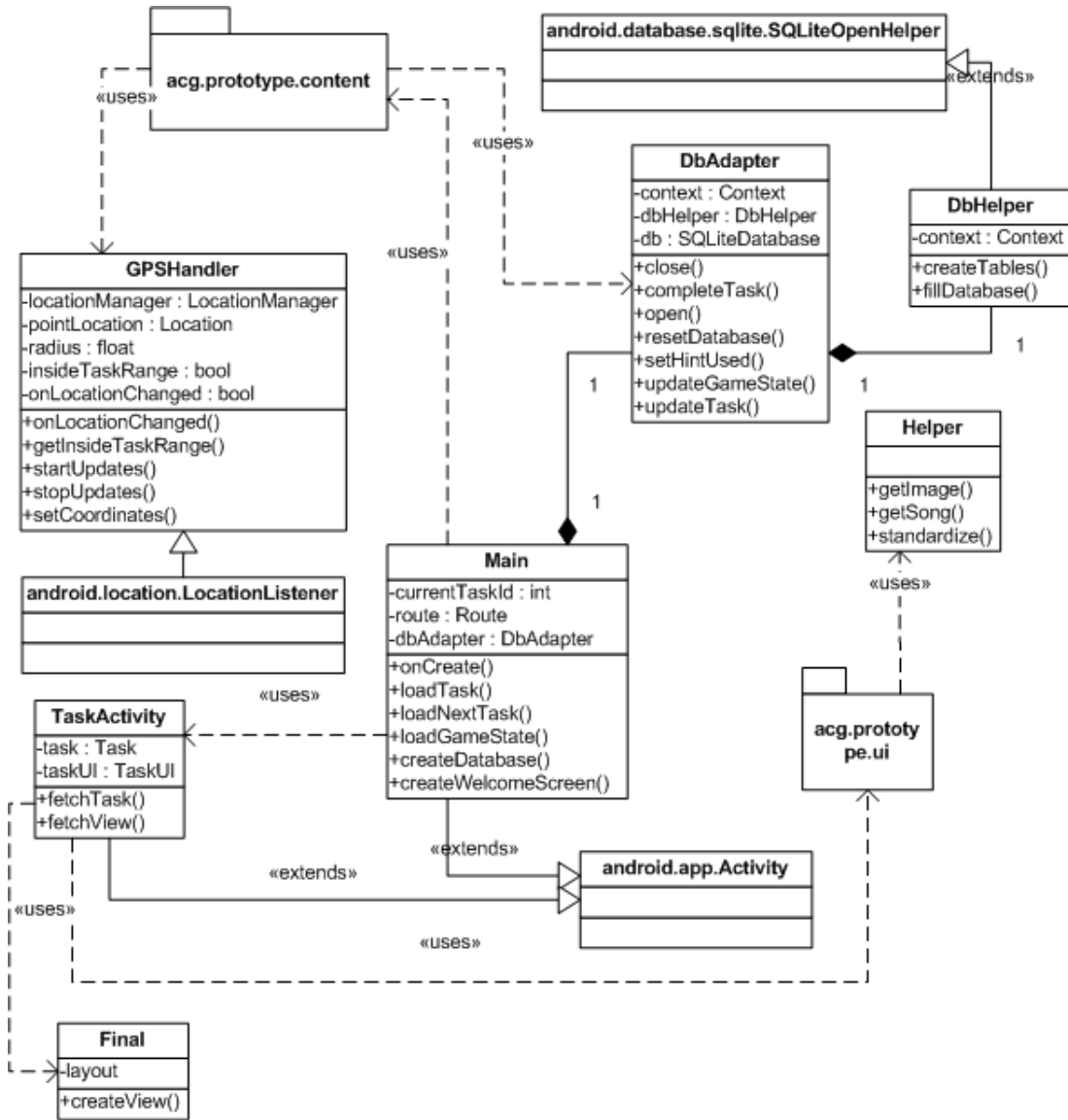


Figure C.1: Class Diagram



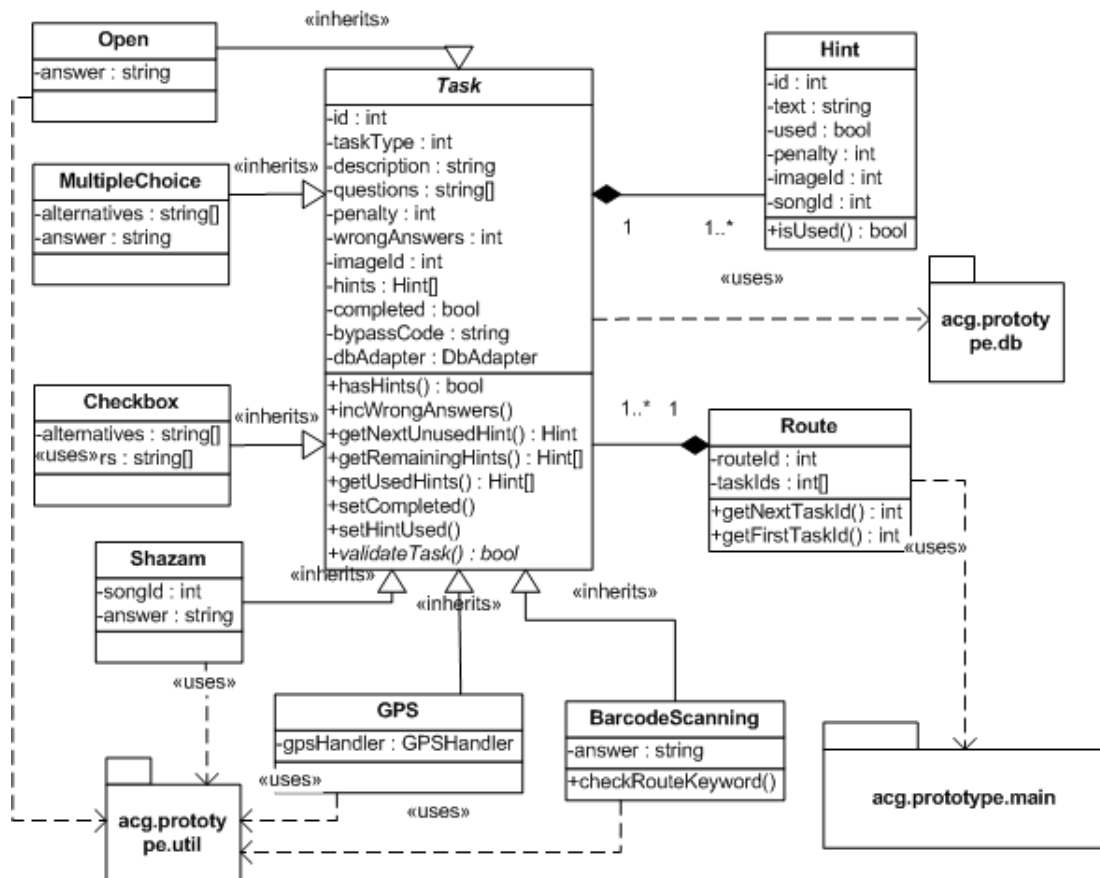


Figure C.2: Content Class Diagram

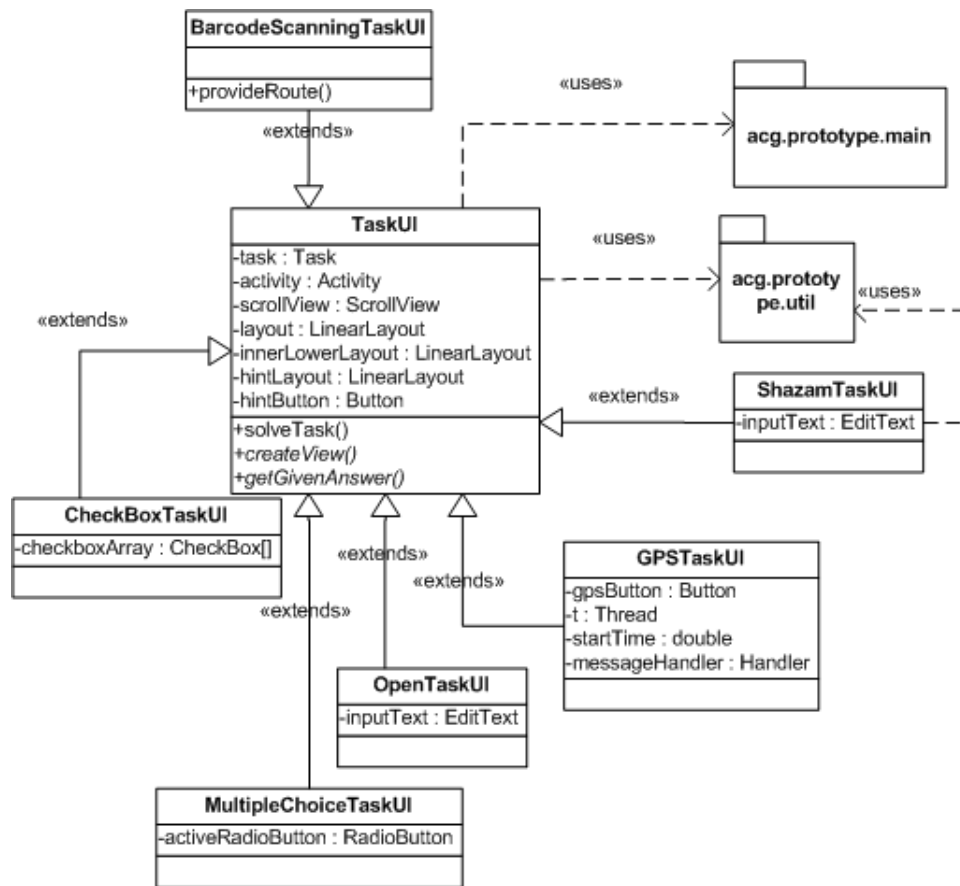


Figure C.3: UI Class Diagram

## APPENDIX D

---

### The Survey

---

This appendix contains questions used in the survey.

	Strongly dis- agree 1	2	3	4	Strongly agree 5
1. I think that I would like to use this system frequently					
2. I found the system unnecessarily complex					
3. I thought the system was easy to use					
4. I think that I would need the support of a technical person to be able to use this system					
5. I found the various functions in this system were well integrated					
6. I thought there was too much inconsistency in this system					
7. I would imagine that most people would learn to use this system very quickly					
8. I found the system very cumbersome to use					
9. I felt very confident using the system					
10. I needed to learn a lot of things before I could get going with this system					

Table D.1: System Usability Scale Survey

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. Most of the gaming activities are related to the learning task							
2. Generally speaking, I can remain concentrated in the game							
3. I am not distracted from tasks that the player should concentrate on							
4. Workload in the game is adequate							

Table D.2: EGameFlow Survey - Concentration

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. Overall game goals were presented in the beginning of the game							
2. Overall game goals were presented clearly							
3. Intermediate goals were presented in the beginning of each scene							
4. Intermediate goals were presented clearly							

Table D.3: EGameFlow Survey - Goal Clarity

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. I receive feedback on my progress in the game							
2. I receive immediate feedback on my actions							
3. I am notified of new tasks immediately							
4. I receive information on my success (or failure) of intermediate goals immediately							

Table D.4: EGameFlow Survey - Feedback

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. The game provides "hints" in text that help me overcome the challenges							
2. The game provides video or audio auxiliaries that help me overcome the challenges							
3. The game provides new challenges with an appropriate pacing							
4. The game provides different levels of challenges that is tailored to different players							

Table D.5: EGameFlow Survey - Challenge

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. I feel a sense of control and impact over the game							
2. I know the next step in the game							

Table D.6: EGameFlow Survey - Autonomy

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. I forget about time passing while playing the game							
2. I become unaware of my surroundings while playing the game							
3. I temporarily forget worries about everyday life while playing the game							
4. I experience an altered sense of time							
5. I can become involved in the game							
6. I feel emotionally involved in the game							

Table D.7: EGameFlow Survey - Immersion



	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. I feel cooperative toward other classmates							
2. I strongly collaborate with other classmates							
3. The cooperation in the game is helpful to the learning							

Table D.8: EGameFlow Survey - Social Interaction

	Strongly dis- agree 1	2	3	4	5	6	Strongly agree 7
1. The game increases my knowledge							
2. I catch the basic ideas of the knowledge taught							
3. I want to know more about the knowledge taught							

Table D.9: EGameFlow Survey - Knowledge Improvement



# APPENDIX E

---

## Survey Data

---

This appendix contains the raw data and comments obtained from the conducted survey. The abbreviations associated with each respondent mean the following: the letter (A-D) marks which group the respondent belonged to, while the number (1-2) is used to differentiate between the origin of answers within a group.

### E.1 Data

Respondent: A1 A2 B1 B2 C1 C2 D1 D2

**SUS:**

I think that I would like to use this system frequently	1	1	3	1	3	4	4	4
I found the system unnecessarily complex	2	2	4	2	1	2	2	2
I thought the system was easy to use	3	3	2	5	5	4	4	5
I think that I would need the support of a technical person to be able to use this system	2	2	3	4	2	1	2	1
I think the various functions in the system were well integrated	2	2	4	5	3	3	5	3
I thought there was too much inconsistency in the system	3	2	4	1	2	2	1	2
I would imagine that most people would learn to use the system very quickly	2	4	2	5	5	4	5	4
I found the system cumbersome/awkard to use	3	3	3	1	2	2	1	1
I felt very confident using the system	2	2	3	5	5	4	5	4
I needed to learn a lot of things before I could get going with this system	1	1	2	3	1	2	1	3

**Concentration:**

Most of the gaming activities are related to the learning task	4	5	4	7	6	5	6	4
Generally speaking, I can remain concentrated in the game	5	4	2	7	3	6	7	6
I am not distracted from tasks that the player should concentrate on	5	4	3	7	3	6	7	6
Workload in the game is adequate	6	6	5	7	5	3	7	6

**Goal Clarity:**

Overall game goals were presented in the beginning of the game	2	2	2	7	4	5	6	5
Overall game goals were presented clearly	3	5	2	7	5	4	5	6
Intermediate goals were presented in the beginning of each scene	5	6	6	7	7	5	5	5
Intermediate goals were presented clearly	4	6	6	7	7	4	5	6

**Feedback:**

I receive feedback on my progress in the game	4	5	7	7	7	5	6	5
I receive immediate feedback on my actions	5	6	7	7	6	7	5	5
I am notified of new tasks immediately	6	6	7	7	7	6	7	7
I receive information on my success (or failure) of intermediate goals immediately	6	6	7	7	6	6	7	7

**Challenge:**

The game provides "hints" in text that help me overcome the challenges	5	6	4	7	4	6	5	6
The game provides video or audio auxiliaries that help me overcome the challenges	5	5	3	7	1	1	6	5
The game provides new challenge swith an appropriate pacing	5	5	2	7	4	5	6	6
The game provides different levels of challenges that is tailored to different players	2	3	1	1	3	4	1	4

**Autonomy:**

I feel a sense of control and impact over the game	2	4	3	4	5	5	6	5
I know the next step in the game	1	5	2	7	5	6	4	6

**Immersion:**

I forget about time passing while playing the game	5	4	5	5	7	7	7	7
I become unaware of my surroundings while playing the game	2	4	4	1	5	7	7	7
I temporarily forget worries about everyday life while playing the game	2	2	4	7	5	7	7	7
I experience an altered sense of time	2	4	3	7	7	6	7	6
I can become involved in the game	5	4	4	7	7	6	7	7
I feel emotionally involved in the game	4	5	3	7	7	7	7	7

**Social Interaction:**

I feel cooperative toward other classmates	4	6	6	7	7	5	1	7
I strongly collaborate with other classmates	3	6	5	7	7	5	1	7
The cooperation in the game is helpful to the learning	3	5	5	7	7	4	7	7

**Knowledge Improvement:**

The game increases my knowledge	5	6	5	7	5	5	5	6
I catch the basic ideas of the knowledge taught	5	5	3	7	6	6	5	6
I want to know more about the knowledge taught	3	4	3	7	6	4	6	5

## E.2 Comments

The following are from the comments section of the survey filled out by demonstration participants. Note that they have been edited for the most blatant grammatical errors and in some cases translated from Norwegian.

Respondent A1:

- A bit chilly holding phone in the wind.
- Battery life was a concern.
- Needs time used status per task and total time used.
- Needs specific penalty breakdown, so software errors can be removed. Also needs cancel for GPS search.
- Also needs introduction to game app and other apps.

Respondent A2:

- Almost ran out of battery.
- Sound clip didn't work at first, not sure why, might be the media volume was at zero.

Respondent B1:

- In my view a good introduction to the game would make it easier to play. Thus, rather than struggling to find out how to use the game and other tools related to it, the player would concentrate on the learning process.
- Some of the tasks were too difficult and very frustrating. In general they should be better balanced.

Respondent B2:

- Require more background information for foreigners.
- In order to help me learn about the city I need more time to look around and get more information. The game generated interest for this city.

Respondent C1: (translated)

- Could not play the audio clip (Shazam task).
- The Shopsyavvy application should be integrated into the game. (In relation to the the task at Torget)
- Would be nice if it were integrated with Google or something.

- Would like the possibility to view the progress/status of other groups.

Respondent C2: (translated)

- Want a way to open an URL by clicking the hint, as we had to remember the URL address.
- Knew too much of the content beforehand.

Respondent D1:

- Sometimes it is difficult to manage the cellphone because the screen is too sensitive and my fingers are larger than the keys.
- It is difficult to manage some applications like the browser, because I am used to my computer browser and I cannot do things like search for something in a web page. By this, I mean look for text.

Respondent D2:

- We have some difficulties, because we are not from Trondheim, and we don't know the landmarks of the city very well.
- We also have some issues when trying to type in some landmark names in Norwegian.
- Another significant issue was that it is the first time I even used a HTC android. Especially with the Google application.

# APPENDIX F

---

## Excursions

---

Throughout the project we had three excursions in the city. In the first excursion we primarily visited prospective task locations in order to get inspiration for, and create tasks. We also tested several extension applications to see if they could be utilized in the prototype. The second excursion tested an early version of the prototype, as well as checking the duration of the race and gathering information and inspiration to refine the tasks. In the third excursion we used an almost finished version of the prototype to simulate a race. This was used to test the prototype and the different routes, as well as checking the duration of the race.

The sections below contain brief summaries of these excursions.

### F.1 Excursion 1

A general excursion with all authors inspecting sites in Trondheim.

#### Date

Friday the 11th of February.

#### Purpose

1. Test Google Goggles.
2. Test Smart Tools.
3. Test Wi-Fi coverage.
4. Find task locations.

5. Get task content inspiration.

### Results

1. Google Goggles did not work properly on less known statues and buildings.
2. We could not get Smart Tools to work at all.
3. The Wi-Fi coverage was limited.
4. We found a few locations.
5. We made some tasks based on the information gathered at the locations.

## F.2 Excursion 2

An excursion performed by one of the authors performing an instance of the game to test the application and check duration, while simultaneously performing some minor other tasks.

### Date

Tuesday the 5th of April.

### Purpose

1. Test application
2. Test duration of one of the routes.
3. Test GPS-tasks.
4. Test battery consumption.
5. Check for locations at Kristiansten Festning to hang up barcodes.
6. Create puzzle task at Kristiansten Festning.

### Results

1. The application worked generally very well, but most suggestions for minor improvements were raised.



2. The duration was around two hours, and the race course thus seems a bit short.
3. Worked generally very well.
4. Application did not consume more than 30 percent of battery.
5. No really suitable locations, have changed start of race to Hovedbygget at Gløshaugen, and groups getting route barcodes and other information delivered at a single spot near the entrance to Kristiansten Festning.
6. Puzzle task created.

## **F.3 Excursion 3**

A general excursion with all authors testing the prototype in Trondheim.

### **Date**

Thursday the 14th of April.

### **Purpose**

1. Test the prototype.
2. Test the different tasks.
3. Test different routes.

### **Results**

1. The prototype works well.
2. Most of the tasks work well, but the checkbox task was exhausting, the same with the two ShopSavvy tasks. Might reduce them a bit.
3. The routes are working as planned.



# APPENDIX G

---

## Demonstration

---

This chapter contains the contestant information and handouts of the demonstration.

### G.1 Contestant Information

**Date:** 3rd of May

**Time:** 13:15

**Start location:** Main entrance of Hovedbygget, NTNU

**Resources provided:** Three Android mobile phones for game play, 500 NOK for each group.

**Group requirements:** Each group has two persons, with a total of three groups in the competition.

**About:** The Amazing City Game is a puzzle race where you compete by solving tasks about the city of Trondheim at various locations in Trondheim using an Android mobile phone. The winner is the group that finishes the course the quickest. The race starts at Gløshaugen, and then takes a detour, before eventually ending up in Midtbyen.

**Routes:** All of the groups will start and end the race at the same destination, and each group will have to solve all of the same tasks. But the groups will get different routes, routes that are distributed when the first task is solved.

**Task 1:** In the first task the groups will compete to get to a specific destination, a destination that will be revealed by one of the game officials at the start of the

game outside Hovedbygget. At this destination each group will get some sheets of paper. The first paper will contain a clue that you have to find yourself. Do not show/give this paper to the other groups as each route is different. Some routes are slightly better than the others, and the fastest groups will get the "best" routes. The other papers are needed in order to solve some of the tasks, so bring them with you throughout the race.

**Time penalties:** If the group struggles, they can get hints about how to solve the task. Receiving a hint will incur a time penalty. Also, giving wrong answers will also give time penalties. The number of minutes added to your total time that you receive is individual for each hint/wrong answer, and will be informed in advance of receiving a hint/solving a task on the mobile phone.

**Prototype:** If the application contains bugs or has game balance issues. Do not hesitate to contact us.

**Rules:** Do not use other means of transport than walking/running. No cars/buses/bicycles. Follow the traffic laws.

**Schedule:** The game starts at 13:15, and will take 2-3 hours. After the race, a survey will take about 15 minutes.

**Prize:** The winning group will get a gift card for 500NOK.

**WARNING:** Parts of the game will take place in areas with traffic. Please keep an eye on the traffic while playing the game. It is easy to get distracted while watching a mobile phone screen and trying to solve a task simultaneously as navigating the streets as well as being under time pressure. Also, one of the tasks will be on walls that are high. Please be careful not to fall down.

**Game officials:**

Runar Os Mathisen 41 93 05 37

Lawrence Alexander Valtola 41 14 40 18

Sondre Wigmostad Bjerkhaug 41 62 07 74

Bian Wu 40 28 27 93

You can contact us for any inquiries, for instance if you are having any particularly big problems with one of the tasks.

Tips for the widgets in Android phone that you may use during the game:

- **Google Goggles:** Google Goggles is a free image recognition application created by Google. Google Goggles enables the user to use pictures taken with the mobile phone to search the web, and is therefore ideal for things that are not easy to describe with words
- **Shopsavvy:** ShopSavvy is an application for reading barcodes of products using the camera of the mobile phone. After reading the barcode, the application will identify the product.
- **Compass:** Standard compass that can be used to find the direction you are heading.

**Notice:**

The game requires network connection at certain points. Most of the areas in Trondheim are covered by Wi-Fi for free use, but there are still some blind spots. Therefore, bring your Sim card as backup and we will provide an adequate 500NOK to cover your GSM and other expenses.

## G.2 Handouts

Each group will get two pages in total. This includes one of the route QR-codes and flag, and a barcode sheet. The bypass codes are handed out to the game officials and group tutors.

# Route 1



## ShopSavvy Challenge:

In one of the tasks the you are asked four multiple-choice questions. The alternatives are written on the back of this paper, together with a barcode for a commercial product. For each questions you have to:

1. Find the correct alternative.
2. Scan the barcode using the ShopSavvy application.
3. Take the first letter of the product name showing in ShopSavvy. With first letter we mean the first letter of wht is written in ShopSavvy. If the product name is «Loreal Shampoo», the letter is L. If the product name is «Bag Of Pringles», the letter is B.

When all of the four questions are answered you will have four letters. Mixing these in the correct order will give the codeword that answers this task.

# Route 2



## ShopSavvy Challenge:

In one of the tasks the you are asked four multiple-choice questions. The alternatives are written on the back of this paper, together with a barcode for a commercial product. For each questions you have to:

1. Find the correct alternative.
2. Scan the barcode using the ShopSavvy application.
3. Take the first letter of the product name showing in ShopSavvy. With first letter we mean the first letter of wht is written in ShopSavvy. If the product name is «Loreal Shampoo», the letter is L. If the product name is «Bag Of Pringles», the letter is B.

When all of the four questions are answered you will have four letters. Mixing these in the correct order will give the codeword that answers this task.

# Route 3



## ShopSavvy Challenge:

In one of the tasks the you are asked four multiple-choice questions. The alternatives are written on the back of this paper, together with a barcode for a commercial product. For each questions you have to:

1. Find the correct alternative.
2. Scan the barcode using the ShopSavvy application.
3. Take the first letter of the product name showing in ShopSavvy. With first letter we mean the first letter of wht is written in ShopSavvy. If the product name is «Loreal Shampoo», the letter is L. If the product name is «Bag Of Pringles», the letter is B.

When all of the four questions are answered you will have four letters. Mixing these in the correct order will give the codeword that answers this task.



# Route 4



## ShopSavvy Challenge:

In one of the tasks the you are asked four multiple-choice questions. The alternatives are written on the back of this paper, together with a barcode for a commercial product. For each questions you have to:

1. Find the correct alternative.
2. Scan the barcode using the ShopSavvy application.
3. Take the first letter of the product name showing in ShopSavvy. With first letter we mean the first letter of wht is written in ShopSavvy. If the product name is «Loreal Shampoo», the letter is L. If the product name is «Bag Of Pringles», the letter is B.

When all of the four questions are answered you will have four letters. Mixing these in the correct order will give the codeword that answers this task.

# Bypass Codes

<b>Task Name</b>	<b>Bypass Code</b>
Welcome	hjerne
Kristiansten Fortress	sverige
Kristiansten Landmark	vinkel
Locating Nidarosdomen	sjokolade
The war memorial	kjakan
The main facade	paven
Locating stiftsgården	rik
The Iron Gate	gruve
Locating Gamle Bybro	niding
Bridge Building	civ
Lykkens Portal	valve
Locating Britannia	india
The production company	rihanna
The flag	rosenborg
Locating Vår Frue kirke	foss
Vår Frue kirke facts	slitsom
Locating Torvet	midten
Torvet ShopSavvy	turbo
Locating Ravnkloa	lofoten
The writer	gris
The Avenue	fem
The End	slutt

### QUESTION 1



### QUESTION 2



### QUESTION 3



### QUESTION 4



### G.3 Images from the Demonstration Day



Figure G.1: The Groups and Tutors at Hovedbygget



Figure G.2: Some Groups and Tutors at Kristiansten Fortress



Figure G.3: A Group with Tutor solving a Task at Kristiansten Fortress



Figure G.4: A Group with Tutor traveling to a new Location



Figure G.5: A Group solving a Task at Stiftsgården



Figure G.6: A Group asking the Tourist Information about a Task



Figure G.7: A Group with Tutor solving a Task at Ravnkloa