



NTNU – Trondheim
Norwegian University of
Science and Technology

Mobile Applications Mobility

Amir Hossein Ghoreshi
Neberd Salimi

Master of Science in Computer Science

Submission date: June 2014

Supervisor: Jon Atle Gulla, IDI

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



NTNU – Trondheim
Norwegian University of
Science and Technology

Mobile Apps Mobility

Amir Ghoreshi & Neberd Salimi

Master of Science in Computer Science

Submission date: June 2014

Supervisor: Jon Atle Gulla, IDI

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

Abstract

The use of mobile devices and mobile applications has soared in recent years. People are using mobile devices and applications for everyday tasks. Compared with desktop computers, smart phones have some limitations such as screen size, limited bandwidth and storage capacity that can be a big challenge. Another challenge is to ensure consistency across platforms and devices. In addition to this, a mobile application is short-lived if it is too advanced to be used. In this paper we present the user interface that is adopted in the Smart Media news recommendation project. We are building a mobile news application that retrieves news from all major Norwegian newspapers. The result of this study will be a news system where the main focus will be on the user interface part of the system and how this part can be created. The application can be either downloaded as an app or run on a browser on any mobile devices. It differs from other applications by offering users the ability to change the way news stories are presented. It also offers functionalities such as offline reading/browsing and sharing of news stories through social media. The application has been evaluated both empirically and heuristically during the development process.

Sammendrag

Bruken av mobile enheter og mobile applikasjoner har økt kraftig de siste årene. Folk bruker mobile enheter og applikasjoner for dagligdagse oppgaver. Sammenlignet med stasjonære/bærbare datamaskiner, har smarttelefoner noen begrensninger som skjermstørrelse, begrenset båndbredde og lagringskapasitet som kan være en stor utfordring. En annen utfordring er å sikre konsistens på tvers av plattformer og enheter. I tillegg til dette, er en mobilapplikasjons levetid kortvarig hvis det er for avansert til å bli brukt. I denne artikkelen presenterer vi brukergrensesnittet som er implementert i Smart Media prosjektet. Vi bygger en mobil nyhetsapplikasjon som henter nyheter fra alle store norske aviser. Resultatet av denne studien vil være et system hvor hovedfokuset vil være på brukergrensesnittdelen av systemet, og hvordan denne delen kan utvikles. Applikasjonen kan enten lastes ned som en app eller kjøres i en nettleser på alle mobile enheter. Den skiller seg fra andre applikasjoner ved at den tilbyr brukere muligheten til å endre måten nyheter blir presentert på. Den tilbyr også funksjonaliteter som offline lesing og deling av nyhetsartikler via sosiale medier. Applikasjonen har blitt evaluert både empirisk og heuristisk under utviklingsprosessen.

Preface

This report is a documentation of the project work performed as part of the Master's thesis in Software spring 2014 by Amir Ghoreshi and Neberd Salimi. The project is carried out in the last semester of two years Master Program in Computer Science at The Norwegian University Of Science and Technology (NTNU). The scope of the project corresponds to 30 units.

The project work has been defined in consultation with our supervisor, Professor Jon Atle at the department of Computer and Information Science (IDI).

We would like to thank our supervisor Jon Atle Gulla and Arne Dag Fidjestøl for guidance and invaluable feedback for the thesis work. We would also like to thank our family who have encouraged and supported us throughout our lives.

Trondheim, June 23, 2014

Amir Ghoreshi

Neberd Salimi

Contents

- Part I 1
- 1 Introduction 2
 - 1.1 Problem 3
 - 1.2 Research questions 3
 - 1.3 Approach 4
 - 1.4 Results 4
 - 1.5 Report outline 5
- 2 Smart Media (SM) 7
- Part II 11
- 3 Background 12
 - 3.1 Mobile applications 12
 - 3.1.1 Native development 13
 - 3.1.2 Web development 14
 - 3.1.3 Hybrid development 15
 - 3.2 Mobile User Interface 16
 - 3.3 Personalization & Recommendation systems 17
 - 3.4 Usability 18
 - 3.5 Usability rules and principles 19
 - 3.5.1 7±2 Principle 19
 - 3.5.2 2-Second Rule 19
 - 3.5.3 3-Click Rule 19
 - 3.5.4 Inverted Pyramid 20
 - 3.6 Psychology of Web Design 20
 - 3.7 Acceptance Models 22
 - 3.8 Technological overview 24
 - 3.8.1 HTML 24
 - 3.8.2 CSS 24
 - 3.8.3 JavaScript 25
 - 3.8.4 JavaScript frameworks 25
 - 3.8.5 Cross-platform mobile development tools 28
- 4 Related works 35
 - 4.1 Personalization on mobile devices 35

4.2	Designing news application.....	36
4.3	Mobile news applications	39
4.3.1	Flipboard	39
4.3.2	Pulse	40
4.3.3	Taptu.....	40
4.3.4	Zite.....	41
4.3.5	News Republic	42
4.3.6	Newspapers	42
4.3.7	Features of the popular news apps.....	43
4.3.8	User experience aspects of the popular news apps.....	44
4.3.9	Reflection	45
Part III	46
5	Application life cycle	47
5.1	Scenarios and use cases	47
5.1.1	Scenarios	47
5.1.2	Use Cases	48
5.2	Requirements (functional and non-functional).....	50
5.2.1	Functional Requirements.....	50
5.2.2	Non-Functional Requirements.....	52
5.3	Presentation of the Proof of concept (POC) prototype.....	53
5.3.1	POC description	53
5.3.2	POC test scenario	56
5.3.3	POC test feedback	57
5.3.4	Additional requirements	58
6	Architecture and implementation	60
6.1	Architecture	60
6.1.1	Strategy.....	60
6.1.2	Pattern.....	61
6.1.3	The way of Angular.....	62
6.1.4	Application structure	65
6.1.5	API.....	66
6.2	Implementation.....	67
6.2.1	Web storage	67
6.2.2	Google Map.....	68
6.2.3	Social networks	68

6.2.4	JSON	68
6.2.5	Libraries.....	69
6.2.6	Events	69
6.2.7	Profile	70
6.2.8	Geographical location.....	70
6.2.9	Application flow.....	70
6.2.10	Development	72
6.2.11	Hybrid application.....	74
6.3	Application Graphical User interface.....	76
6.3.1	Application description	76
6.3.2	Application layout	82
Part IV	105
7	Evaluation.....	106
7.1	Introduction	106
7.2	The evaluation process	107
7.3	Evaluation results	110
7.3.1	First phase.....	110
7.3.2	Last phase.....	112
7.4	Suggested improvements.....	113
8	Conclusion and further work.....	115
8.1	Conclusion.....	115
8.2	Further work.....	117
9	Bibliography.....	118
Appendix A	122

List of Figures

Figure 1.1: The main focus of the project	2
Figure 2.1: Architectural view of the Smart Media system [19]	8
Figure 2.2: Screenshot of the iOS news app.....	10
Figure 3.1: List of popular mobile operating systems	12
Figure 3.2: Screenshots of app stores for iOS and Android	13
Figure 3.3: Maslow's Hierarchy of Needs	21
Figure 3.4: Design hierarchy of needs.....	22
Figure 3.5: Technology Acceptance Model (TAM) [9]	22
Figure 3.6: Mobile Services Acceptance Model (MSAM) [17]	23
Figure 3.7: Interest over time. Web search. Worldwide, past 12 months (Google Trends)	27
Figure 4.1: Screenshot from 247 showing the individual UI design along with an example of individual usage [7]	37
Figure 4.2: Screenshot from 247 showing the peripheral UI design along with an example of peripheral usage [7]	37
Figure 4.3: Screenshot of News Sync showing results for the query "Watergate" in the catching-up scenario [52]	38
Figure 4.4: Screenshots from Flipboard showing top categories, articles related to the category "Technology", and a single news article	39
Figure 4.5: Screenshots from Pulse showing the start page, list of categories, and a single news article	40
Figure 4.6: Screenshots from Taptu showing the welcome page, categories and a single news article	41
Figure 4.7: Screenshots from Zite showing the categories, top stories and a single news article	42
Figure 4.8: Screenshots from News Republic showing the categories, articles related to one category, and a single news article.....	43
Figure 4.9: Screenshots of some newspapers	44
Figure 5.1: Use case diagram	49
Figure 5.2: Site map of the POC prototype	55
Figure 6.1: The connection between clients and the server.....	60
Figure 6.2: Overview of the layers and their roles	61
Figure 6.3: MV* pattern	62
Figure 6.4: Screenshot of Service dependencies	65
Figure 6.5: Rough sketch of the application's architecture.....	66

Figure 6.6: Image showing the API methods	67
Figure 6.7: Screenshot of the application's code hierarchy (shown in WebStorm).....	73
Figure 6.8: Screenshot of the application files during run time displayed in the browser's development tool.....	73
Figure 6.9: Screenshots of the Hybrid application running on a smartphone	75
Figure 6.10: Screenshot of the application's Home page	76
Figure 6.11: Screenshot showing the notification feature	77
Figure 6.12: Screenshot showing the loading of news articles.....	79
Figure 6.13: Screenshot from the application showing the search feature	80
Figure 6.14: Site map of the application	82
Figure 6.15: Image of the application's design structure	83
Figure 6.16: Image showing different loading types.....	84
Figure 6.17: Screenshot of the Index view	84
Figure 6.18: Two screenshots of the Home page	85
Figure 6.19: Screenshot of the Basic view in large screen.....	87
Figure 6.20: Screenshot of the Basic view in small screens (such as smartphones)	88
Figure 6.21: Screenshot of the Basic view in medium screens (such as tablets).....	89
Figure 6.22: Screenshot of the Card view in large and medium screens (such as desktop and tablets)	89
Figure 6.23: Screenshot of the Card view in small screens (smartphones).....	90
Figure 6.24: Screenshots of the Details view in different screen sizes	91
Figure 6.25: Screenshots of the Image view in different screen sizes.....	92
Figure 6.26: Screenshots of the List view in different screen sizes	93
Figure 6.27: Screenshots of the Main view in different screen sizes	94
Figure 6.28: Screenshots of the Map view in different screen sizes	95
Figure 6.29: Screenshots showing collapsed and un-collapsed right panel.....	96
Figure 6.30: Article view (step to turn the geo-location on and increase text-size).....	97
Figure 6.31: A single article displayed in Modal window	97
Figure 6.32: Screenshots of a single article in modal window with different background colors.....	98
Figure 6.33: Screenshots of the rating and sharing features (with hovering effects)	98
Figure 6.34: Screenshots of the Profile widget	99
Figure 6.35: Step 1 of the authentication process.....	100
Figure 6.36: Step 2 of the authentication process (input validation).....	101
Figure 6.37: Screenshots showing the validation process in case of success or failure	101
Figure 6.38: Left-side navigation menu showing different user status and view	102
Figure 6.39: Right-side navigation menu	103

Figure 6.40: Left-side navigation menu during log-out process (news vs. article view) 104

List of Tables

Table 3.1: Overview of different development approaches [30].....	16
Table 3.2: Advantages and drawbacks of PhoneGap	29
Table 3.3: Advantages and drawbacks of Appcelerator Titanium	31
Table 3.4: Criteria for evaluation of the cross-platform development tools.....	32
Table 5.1: Textual use case 1	49
Table 5.2: Textual use case 2	50
Table 7.1: Advantages and disadvantages of some evaluation methods [25][36][42]	107

List of Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
DI	Dependency Injection
DOM	Document Object Model
HTML	Hyper-Text Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MVC	Model-View-Controller
OS	Operating System
POC	Proof Of Concept
RS	Recommendation System
RSS	Rich Site Summary
SDK	Software Development Kit
SM	Smart Media
TAM	Technology Acceptance Model
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

Part I

1 Introduction

The use of mobile devices and mobile applications is growing at a tempestuous pace. Smartphones play important roles in people's lives as mobile applications are used more than ever for reading news, entertainment, socializing etc. In addition, smart phones make that most people have access to information anywhere and anytime. This growing portion of information can create problems as more and more data is available. When the information is plentiful, the knowledge of what information is useful and valuable matters most.

Compared to desktop/personal computers, smart phones have some inherent limitation such as screen size, limited bandwidth and storage capacity which can be challenging. Therefore, it is important to both filter the information and present the information in a user friendly way when creating mobile applications.

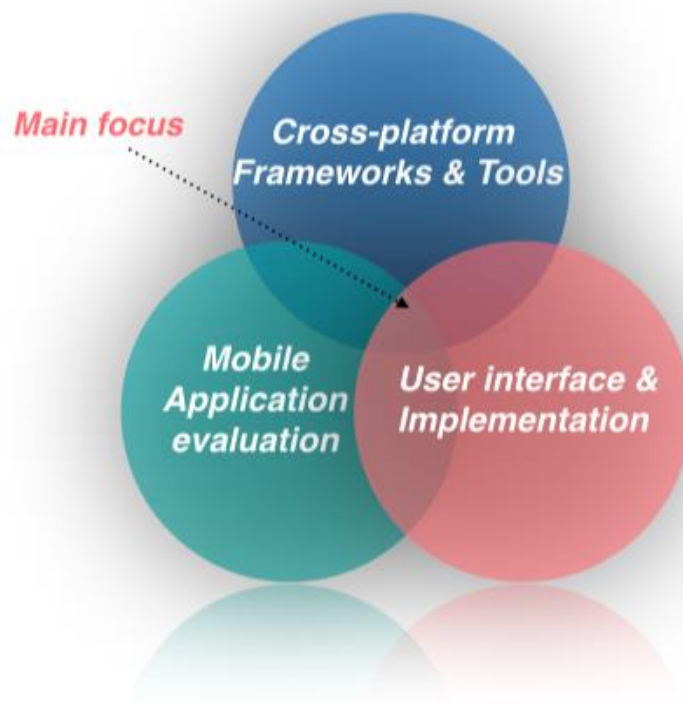


Figure 1.1: The main focus of the project

1.1 Problem

The smartphone space is dominated by three big platforms: iOS, Android and Windows. In order to reach many users, mobile applications must be designed and built for all three of them. Traditionally it means using each platform's provided technology and SDK, i.e. Objective-C for iOS, Java for Android and .NET for Windows.

This thesis will concentrate on making a responsive web application and a hybrid application using cross-platform tools. The result will be a news recommender system that can run on different platforms and devices, where the main focus will be on the user interface part of the system and how this part can be created.

1.2 Research questions

The following are the research questions that need to be answered in this thesis/study.

Q1. What characterizes the user interfaces of successful and user friendly mobile news apps?

The concept of usability and principles that form the foundation of a successful and user-friendly application will be presented. It will also be discussed considerations that have to be taken into account when designing applications for mobile devices. An application will be developed based on the usability principles. The application will be tested and evaluated by representative users.

The back-end logic and concerns related to the API is out of scope for the thesis.

Q2. How can existing frameworks and tools be used to reduce development and maintenance costs and ensure consistency across platforms and devices?

Different approaches for developing applications (advantages and disadvantages of these) will be presented. A list of various tools/framework will be listed. There are many frameworks on the market that can be used to develop applications across various platforms and devices. It is out of scope to look at all of these. Therefore, the focus will be to look only at the most popular and recognized tools available. Based on some criteria, some of these tools/frameworks will be selected to develop the application. Finally they will be evaluated to see if they were the right choices to fulfil the various criteria that were set.

Q3. How should a news recommender system user interface be designed to minimize user interaction and support implicit user profiling?

This research question is about how information (in this case, news articles) can be presented in an intuitive way so that it does not require so much user interaction. For example, without giving any explicit details, the application should display relevant news stories based on the user's implicit behavior and preferences such as reading habits.

Different perspectives to present the news will be presented. In addition, a handful of popular and well-known existing news applications will be considered during the development of the application.

1.3 Approach

The research approach in this thesis is a theoretical study of web design, frameworks and tools, and a practical experiment of the chosen frameworks/tools in use.

First, a theoretical study will be performed by reading papers about design (especially for mobile devices) and cross-platform development. In addition, it will be done a study examining the positive and negative aspects of user interfaces for existing news application.

Second, based on the findings in the first part, a web and a hybrid news application will be designed and developed. The news applications will make use of an already developed recommendation system that provides real-time news from Norwegian newspapers.

Third, the applications will be tested with respective users. Finally, an evaluation of the application and the findings will be carried out.

1.4 Results

The result of this project focuses on identifying characteristics for a successful and user-friendly mobile news application by looking at existing commercial news application at present time and published articles on the subject, and especially how they are realized in terms of design, recommendation, and the mobile features to support this. Development of two use case application (a web application and a hybrid application) are also part of the results in form a practical approach on how to realize a user-friendly mobile news recommendation application.

Various usability principles and rules will be considered to achieve an intuitive and user-friendly application. First and foremost, it is important that the application is easy and pleasant to use. Other important quality attributes is that the application is flawless and that the users understand and to some extent can control the user interface.

The application will be tested by both usability experts and ordinary users. To compare our application with other news applications and to evaluate it, we will let users use both our application and some popular commercial news applications. After using the applications in a certain time, the users shall answer a set of questions related to the user interface. This will give us a good idea of what is bad or good with our application compared to existing news applications.

In the use case application, one of the focus areas was to limit the steps to perform a simple action, and make the most important actions more visible and present at all times. Therefore, buttons and links are created in such a way that they are more visible and more appealing than other items on the application. It is placed side menus that allow the users to always have access to the most important functions in the application.

1.5 Report outline

Below is given a short overview of the different chapters of the report.

Chapter 2 Smart Media (SM) gives a presentation of the Smart media project.

Chapter 3 Background presents an overview of the technologies chosen for developing the application. It also introduces the theoretical background of the project.

Chapter 4 Related works presents work from related projects and research.

Chapter 5 Application life cycle starts with user scenarios and requirements, and ends with a presentation of the POC prototype.

Chapter 6 Architecture and implementation contains the details of the completed application, including its architecture and the implementation process.

Chapter 7 Evaluation gives an evaluation of the system based on testing with users.

Chapter 8 Conclusion and further work marks the end of the report by giving some concluding thoughts about the outcome of the project. It also contains a discussion of possible further work.

2 Smart Media (SM)

The SM is a program at the Department of Computer and Information Science at the Norwegian University of Science and Technology, which was established in 2012 in close collaboration with the regional media industry and the Telecom operator Telenor Group. The program focuses on exploring new technologies to help the news industry with the explosion of online information and looking for ways to deliver news more efficiently and appealing to their readers.

Central to the SM program are technologies like:

- Semantics
- Mobile platforms
- Big Data architectures
- Text analytics and sentiment analysis
- Information retrieval and recommendation

The architecture of the SM is quite complex and consists of several technologies and APIs. As shown in figure 2.1, it is a combination of a standard Solr index for news articles, a Hadoop cluster for generating user profiles, and a MangoDB for event logs and generated user profiles.

On the server side, RSS news streams from different Norwegian newspapers are continually analyzed and indexed in Solr for subsequent recommendations. The indexing process accesses the RSS news before it retrieves the corresponding HTML documents. After extracting the body texts from these HTML documents, the system starts to extract named entities from the texts. Afterwards it identifies the locations of the news with Google Maps, and then it stores the information about every news article in a structured Solr index [19].

A middleware layer is used on the client side to retrieve recommended news articles from the Solr index and present these to the user. The user's activities, like user clicks or when the user opens the article in full view, are logged by the client. The client sends these events/logs to the server for storing in the MangoDB database.

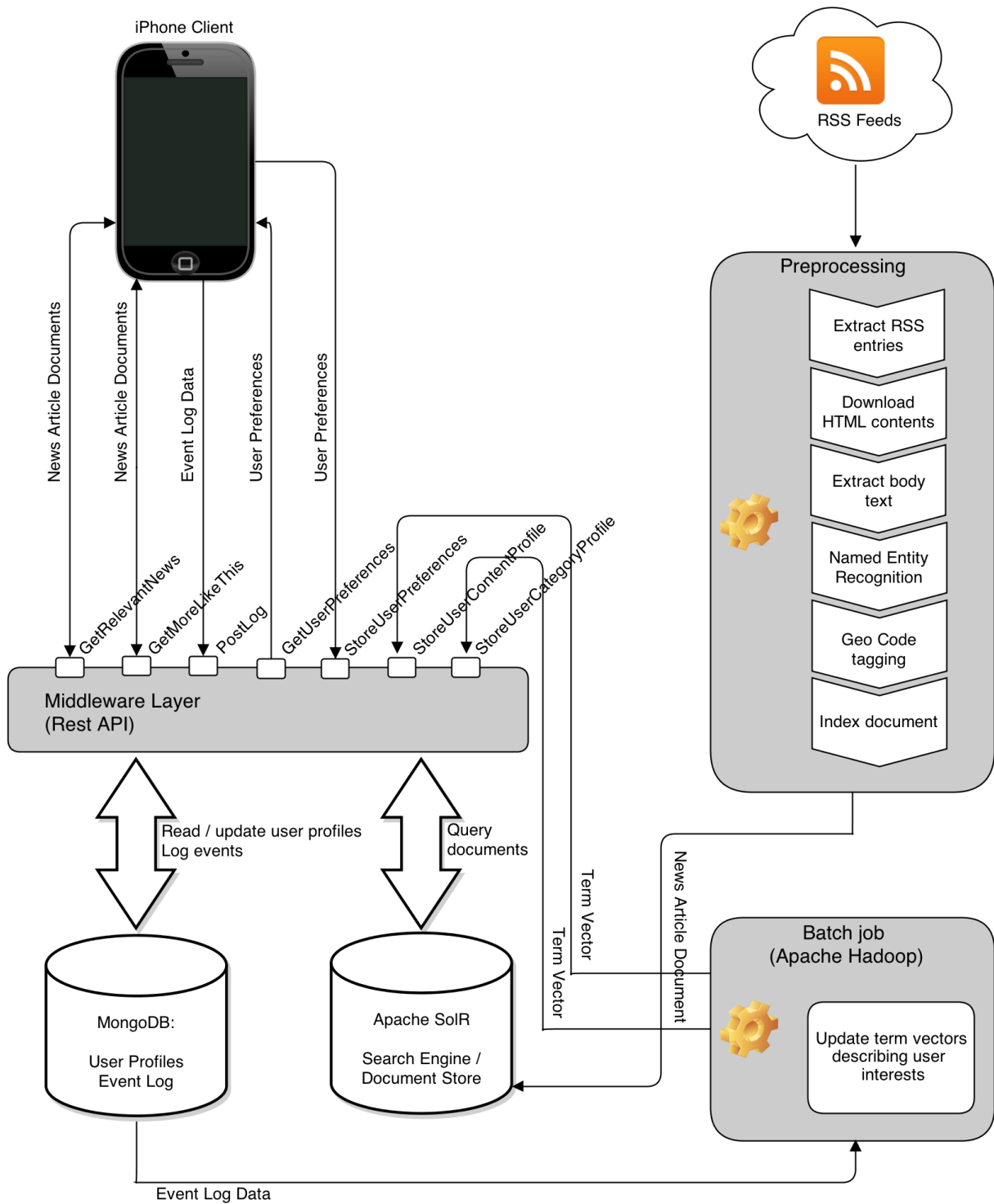


Figure 2.1: Architectural view of the Smart Media system [19]

Different existing techniques for recommendation were explored to serve as background for this solution. Some of these techniques were collaborative filtering, content-based filtering, freshness and location [48]. While collaborative filtering approaches build a model from a

user's past behavior (articles previously desired by the user) as well as similar decisions made by other users, content-based filtering approaches utilizes a series of characteristics of an article in order to recommend additional articles with similar properties. Freshness of news is a challenge with news recommendation. Since news have a short lifespan and are best served fresh, making approaches like collaborative filtering a difficult task considering scalability, sparsity and the cold start problem. Regarding location, the news articles are ranked higher based on the closeness of their location to the user's current location.

In the present time, there is only one client application that makes use of the SM's back-end. The application is a native iOS app developed for iPhone running on iOS 6.0 and above [21]. A hybrid approach to news recommendation is adopted in this prototype. Freshness and locational information extracted from news events and users mobile are part of the recommendation strategies to make sure that news events in users' presence are given adequate attention [19].

The iOS news app has a pure gesture-based user interface. As shown below in figure 2, swiping movements are used to turn pages and navigate in the news app. When you start the application for the first time, you get asked whether you want to share your position or not. You get recommended news articles based on your location and user profile. Therefore the application allows you to log in if you want to share your profile across multiple devices. In the start page you get popups which guide you through the use of the application. In the top left corner you will see/get the number of unread recommended articles. The main page of the app gives a short summary of the latest news. Sliding-side menus are available from the main page to configure the app or select particular news categories. By swiping down, you get recommended news. While swiping horizontally moves you from one recommended article to the next, swiping vertically takes you either to the main page or to the full view of the article. If you double click an article, you will get a map showing where the news took place. In addition, the app allows you to share articles in social media like Facebook and Twitter. This can be done by holding down one finger on the article, which gives you the share view.

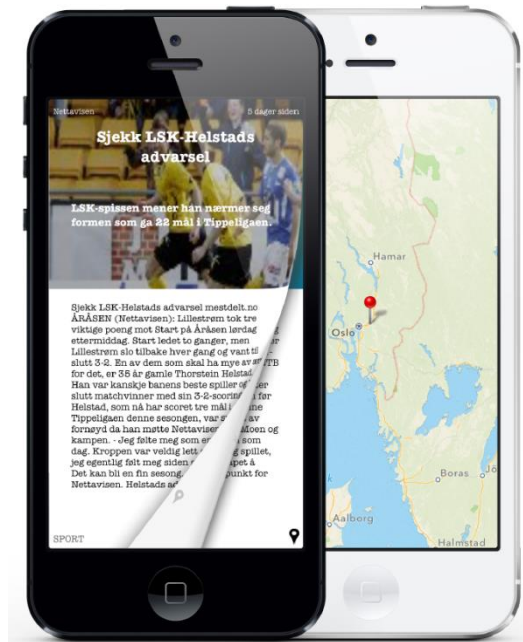


Figure 2.2: Screenshot of the iOS news app

Each news story has a number of views that may need to be supported by the user interface (UI). These are

- RSS text: This view shows full or summarized text of a news article. It includes metadata like title, lead-text, image, publishing date and author's name.
- Full text: As the name indicates, this view normally holds all the information that is accessible for a particular news story.
- Entities: An entity view shows a set of keywords extracted from a news article by using a keyword extracting algorithm or other technologies.
- Location on map: The map view shows the place where the news story is related to.
- Category: This view displays only those news articles that belong to the same category.

The plan ahead is to create a web application and a hybrid application. The goal is to reach out to more users and to investigate whether cross-platform application is better than the traditional way of making native mobile application.

Part II

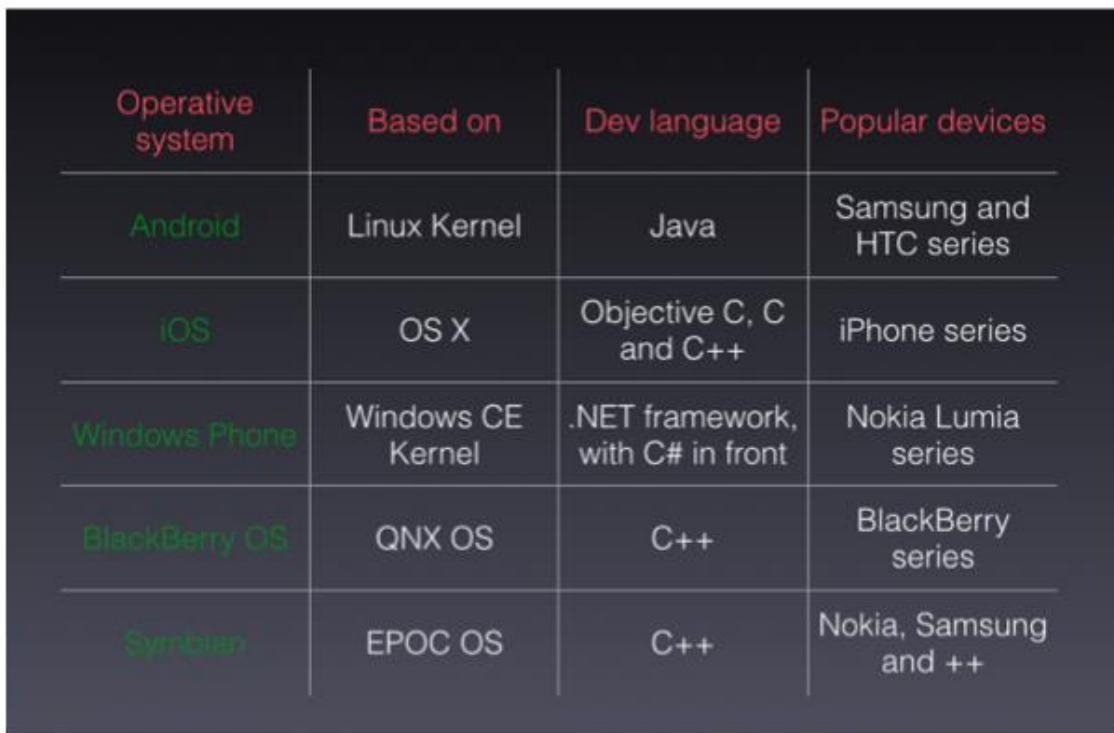
3 Background

This chapter describes the theoretical basis for the project. It provides a brief explanation of the technologies used to develop the application. It also gives an insight into the theories, models and related studies of personalization.

3.1 Mobile applications

A mobile application, most commonly referred to as an app, is a software application/program developed to run on a mobile device, such as smartphones and tablets. Mobile applications can come preloaded in the device as well as can be downloaded by users from app stores or the internet. Mobile applications allow users to customize their devices to their specific set of wants and needs, e.g. playing music and games or socializing with others through social media applications.

Mobile applications can be developed by using one of three different approaches: web development, hybrid development, and native development. All these development approaches will be explained more thoroughly later in this chapter.



Operative system	Based on	Dev language	Popular devices
Android	Linux Kernel	Java	Samsung and HTC series
iOS	OS X	Objective C, C and C++	iPhone series
Windows Phone	Windows CE Kernel	.NET framework, with C# in front	Nokia Lumia series
BlackBerry OS	QNX OS	C++	BlackBerry series
Symbian	EPOC OS	C++	Nokia, Samsung and ++

Figure 3.1: List of popular mobile operating systems

3.1.1 Native development

By using the native development approach, you can create applications that are written for a specific platform and runs on that platform only. To support platforms such as iOS, Android and Windows Phone, you have to develop separate applications with different programming languages and environments, such as Objective-C and XCode for iOS, Java and Eclipse or any other IDE for Android, and C# and Visual Studio for Windows Phone. Therefore this type of application is expensive to develop because it is tied to one type of operating system. However, native applications can fully use all platform and device functions such as accessing camera or address book/contact list. For native applications, the platform provides real native UI (based on the machine language of the respective platforms) for iOS, Android and Windows Phone. Typically, native applications are distributed through an application store, e.g. iOS App Store and Google Play store for Android.



Figure 3.2: Screenshots of app stores for iOS and Android

Advantages of native applications [13][30]:

- They offer a best-in-class user experience, offering a rich design and tapping into device features and offline use.
- They provide full access to the device's hardware, such as its GPS sensor, contact list, camera, microphone, and accelerometer.
- They offer the best graphic and animations.
- They are relatively simple to develop for a single platform.
- The code can load and run faster.

Disadvantages of native applications [13][30]:

- They cannot be easily ported to other mobile platforms.
- Developing, testing, and supporting multiple device platforms is incredibly costly.
- They require certification and distribution from a third party that you have no control over.
- They require you to share revenue with the one or more third parties.
- They require the most future maintenance when compared with the other two mobile options.
- When placed in an app store, a native application is completely controlled by the app store's owner (like Apple or Google).

3.1.2 Web development

With the web development approach, you can use standard technologies such as HTML, CSS, and JavaScript to create an application that runs inside the browser of the mobile device. Since your application is platform independent, you do not need to develop a new application to support a new mobile platform. But some modifications might be required to support a different browser engine. Unlike native applications, mobile applications cannot access the platform functions because they depend only on the browser. In addition, mobile web applications are not distributed through an app store. They are available via a link on a website or a bookmark in the user's mobile browser.

Advantages of mobile web applications [13][30]:

- They are simple to deploy across multiple handsets.
- They are easy to create, using basic HTML, CSS and JavaScript knowledge.
- They offer a better user experience and a rich design, tapping into device features and offline use.
- Content is accessible on any mobile web browser.
- Since one mobile app works on every platform, future maintenance is simple.
- Since mobile web apps run in a browser, and are not installed on the device itself, all updates instantly reflect in the application. This means that users do not need to install the latest update or do anything at all. The app instantly updates for all users.
- Mobile web apps offer complete freedom since they are distributed through the web and not through a proprietary "app store" controlled by a third party.

Disadvantages of mobile web applications [13][30]:

- The optimal experience might not be available on all handsets.
- They can be challenging (but not impossible) to support across multiple devices.
- They do not support native application features, like access to the device's camera, microphone or contact list.
- They do not handle heavy graphics as seamlessly as native apps.

3.1.3 Hybrid development

With the hybrid development approach, you can create applications that combine elements of both native and web applications. Hybrid applications, which are written with web technologies (HTML, CSS and JavaScript), run inside a native container and leverage the browser engine to render HTML and process JavaScript. Unlike web applications, hybrid applications can access device capabilities such as the accelerometer, camera and local storage. Similar to native applications, hybrid applications are distributed through the application store of the platform.

Advantages of hybrid applications [30]:

- Native look-and-feel (without the native cost).
- They offer full device access, including the "native-only" features, like the camera, microphone, and address book.
- They can be deployed in the platform market for applications such as AppStore for iPhone and Android Market for Android.
- Building hybrid apps are considerably cheaper than building native apps.

Disadvantages of hybrid applications:

- Different support for the different OS.
- Functionality that is not supported by the framework must be written as plugins to the framework in native code.
- Often provide access to the lowest common denominator of features. If Android comes with a new feature, it is not certain that the framework will support it if it does not exist on other platforms.
- Hybrid apps offer the same graphical abilities as mobile web apps [30].

- Turning a mobile app into a hybrid app require familiarity with a mobile framework [30].

	Native App	Mobile Web App	Hybrid App
Skills/tools needed for cross-platform app development	<ul style="list-style-type: none"> • Objective-C • Java • C++ • C# • VB.net 	<ul style="list-style-type: none"> • HTML • CSS • Javascript • Web programming language (i.e., Java) 	<ul style="list-style-type: none"> • HTML • CSS • Javascript • Web programming language (i.e., Java) • Mobile development framework
Apps needed to reach all major smartphone platforms	4	1	1
Installed on device?	Yes	No	Yes
Distribution	App Store/Market	Internet	App Store/Market
Device integration	Full integration: (camera, microphone, GPS, gyroscope, accelerometer, file upload, contact list)	Partial integration: (GPS, gyroscope, accelerometer, file upload)	Full integration: (camera, microphone, GPS, gyroscope, accelerometer, file upload, contact list)
Best used for	<ul style="list-style-type: none"> • Highly graphical apps • Apps that need to reach a large consumer audience 	<ul style="list-style-type: none"> • Data-driven apps • B2B apps • Internal business apps 	<ul style="list-style-type: none"> • Cross-platform apps that need full device access. • Business apps that need app store distribution

Table 3.1: Overview of different development approaches [30]

3.2 Mobile User Interface

In the recent years, the use of smart phones for activities such as internet and app usage, games, music and others have exploded. According to a report from ZenithOptimedia (2013), Norway was the country in the world that lay at the front when it came to use of digital media in 2012. In addition, this report showed that 65 percent of Norwegians own a smartphone [58]. This trend will continue to increase and many of the users are and will expecting exceptional experiences from the applications they use.

Even though mobile devices brings with it many opportunities, it has many constraints too. The most obvious constraint going from desktop to mobile is screen size. Mobile screens are smaller and therefore one should take that into consideration when designing and developing applications for mobile devices. Another obvious constraint is difference in certain input mechanisms, e.g. there is no mouse in mobile devices. No mouse means no hover states. It also means that there must be some other means (such as the user's finger) of clicking and navigating content. This difference in input method can be quite exciting because it opens the door to new possibilities with various touch gestures. Other important constraints are thing like battery life and processing power [46].

3.3 Personalization & Recommendation systems

Personalization can mean different things to different people in different fields. But roughly speaking, personalization can be defined as the use of technology to meet the individual needs of users. Neil Thurman and Steve Schifferes (2012) define personalization as "*A form of user-to-system interactivity that uses a set of technological features to adapt the content, delivery, and arrangement of a communication to individual users' explicitly registered and/or implicitly determined preferences*" [50]. Personalization where the user can manage the system is called explicit personalization; whereas personalization that is done automatically by the system is called implicit personalization [11].

According to Fan and Poole (2006), there are four ideal types of personalization: architectural, relational, instrumental, and commercial. Each type represents a different philosophy regarding the motivation behind personalization and what personalization tries to achieve. These four ideal types are also different in two other dimensions. First of all, they can be differentiated in terms of utilitarian or affective orientation. The instrumental and commercial types are utilitarian, as they accentuate task performance, while the architectural and relational perspectives are affective because of their emphasis on user's emotions. Secondly, they can also be differentiated in terms of assumptions of use. While architectural and instrumental personalization are individual, and are concerned with the individual interaction with the system, commercial and relational personalization are considered interactional since they focus on the relations among multiple entities [11].

"Recommendation systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user" [41]. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read. A recommender system normally focuses on a specific type of item (e.g. movies or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.

RSs have become an important research area since the appearance of the first papers in collaborative filtering in the mid-1990s. Recommender systems are usually classified into the following categories, based on how recommendations are made [1]:

- Content-based recommendations: The user will be recommended items similar to the ones the user preferred in the past;
- Collaborative recommendations: The user will be recommended items that people with similar tastes and preferences liked in the past;
- Hybrid approaches: These methods combine collaborative and content-based methods.

3.4 Usability

A system's overall quality is a sum of many quality attributes, which one of them is usability. ISO 9241-11 (1998) defines usability as "*The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*" [24].

Usability consultant, Jakob Nielsen (1993) defines usability in terms of five quality attributes [32]:

1. Easy to learn: the user should be able to go quickly from not knowing the system to do some work.
2. Efficient: the system should let the user attain a high level of productivity. User efficiency is a measure of how fast users can use the system to perform their tasks.
3. Easy to remember: after a period of inactivity, the user should be able to return to do some work without having to learn everything all over.
4. Relatively error-free or error-forgiving: users do not make many errors or if they do make errors, they can easily recover from them.
5. Pleasant to use: the users like the system and they are satisfied by using it.

The first and the last quality attribute are the most important ones for us to achieve when it comes to the user interface of the application. But we will take into consideration all the attributes that are mentioned above.

3.5 Usability rules and principles

In this section we will present some important usability issues, terms, rules and principles.

3.5.1 7±2 Principle

Due to some limits on its capacity for processing information, the human brain deals with complexity by dividing information into chunks and units. According to George A. Miller (1956), humans' short term memory can retain only about 5-9 things at one time [31]. This fact is often used as an argument for limiting the number of options in navigation menus to 7. There are heated debates about "Seven, Plus or Minus 2" principle and therefore it is not clear how this principle can, could or should be applied to the web [16].

On this basis, the navigation of the SM application shall not contain more than seven elements so that users find what they may be looking for.

3.5.2 2-Second Rule

In accordance to this principle, a user should not wait more than 2 seconds for certain types of system response, such as application switching and application launch time.

To increase the user's efficiency and the user experience, the SM application shall not use more than few seconds to present news articles or switch between different pages/views.

3.5.3 3-Click Rule

This rule emphasizes the importance of clear navigation, logical structure and easy-to-follow site hierarchy. It suggests that a user of a website should be able to find any information or access the site features within three mouse clicks. Otherwise, the users will stop using the website [16].

The application will be developed in such a way that the users are able to find any information or access certain features within three mouse clicks.

3.5.4 Inverted Pyramid

The inverted pyramid is a writing style where the conclusion or the summary of the article is presented in the beginning of the article. This approach is well-known in journalism where writers try to give their readers an instant idea about the topic they are reporting [16]. According to Nielsen (1996), the inverted pyramid becomes even more important for web writing and for better user experience since users usually do not scroll, which means that they read only the top part of an article. By presenting the summary of the article at the top, the readers can stop at any time and will still get the most important of the article [35].

The application shall follow this principle with its own twist. It shall initially display image, headline and lead text to give an indication of what the article is about. If the article seems interesting for the user, then the user can tap/click on it to be sent to the full article view where all information related to the article will be displayed.

3.6 Psychology of Web Design

One of the most famous theories in psychology is Maslow's hierarchy of needs, which was proposed by the Russian-American psychologist Maslow Abraham in 1943 [56]. According to Maslow, if you try to satisfy the needs of one level in the hierarchy without having first met the needs of the prior level, your place in the hierarchy will be unstable. Lower levels in the hierarchy serve as the foundation for higher levels.

Based on this theory, the idea of a design hierarchy of needs rests on the presumption that in order to be successful, a design must meet basic needs before it can satisfy higher-level needs. Before a design can impress us, it must work as intended. It has to meet some minimal need or nothing will really matter. As shown in figure 7.2, Maslow's hierarchy can be translated to design, for which the hierarchy from low to high would be functionality, reliability, usability, proficiency and creativity [8].

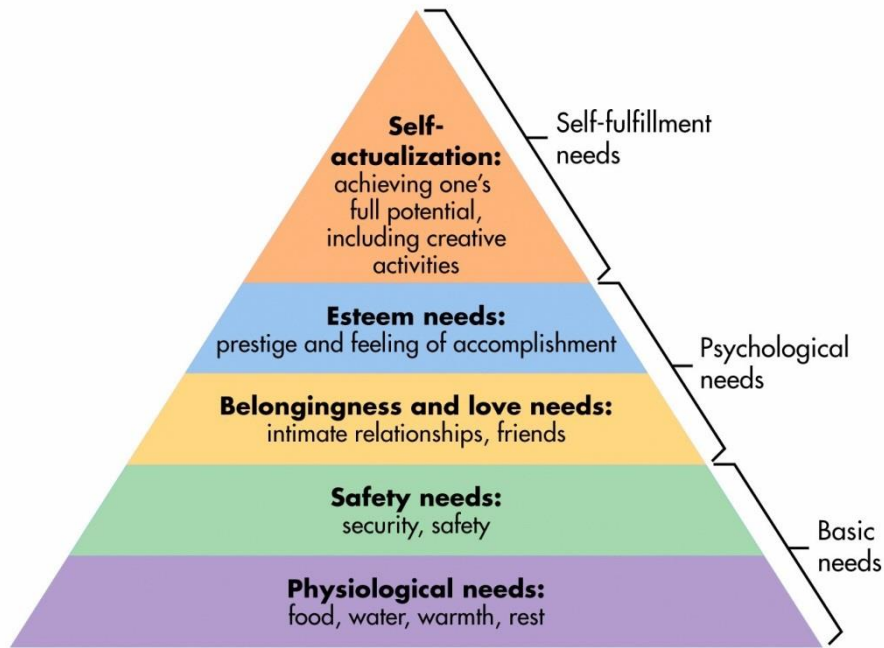


Figure 3.3: Maslow's Hierarchy of Needs

A design must be able to function before anything else. Once the design has met the functional needs, it can move up to the next level in the design hierarchy: reliability. In this level, the design should offer stable and consistent performance. It should not only work, but work again and again. Once the design works consistently, then one can focus on the next level which is usability needs. It should be relatively easy to learn to perform basic tasks quickly, without a lot of learnability. For instance, a usable website should have a navigation system that is easy to understand and use. Among other things, it should have a readable text and a layout in which orienting one is straightforward. The next level in the hierarchy is proficiency. A design that lets people do things that formerly were not possible and to expand on basic functionality is considered to be great and proficient. When all the lower-level needs have been met, the design can move to creative needs. Here, the design can explore and create things that expand the product itself [8].

While Maslow's hierarchy makes sense, it has been challenged and criticized because of lack in empirical evidence and particularly the order of needs specified by the model (the assumption that lower levels must be satisfied before higher levels). The same could be said of the hierarchy of design needs. Although there is little evidence to support the hierarchical aspects, meeting lower-level needs before attempting to satisfy higher-level needs makes sense most of the time. For example, if your website is not usable, you will probably fix that before giving visitors more ways to be proficient [8].

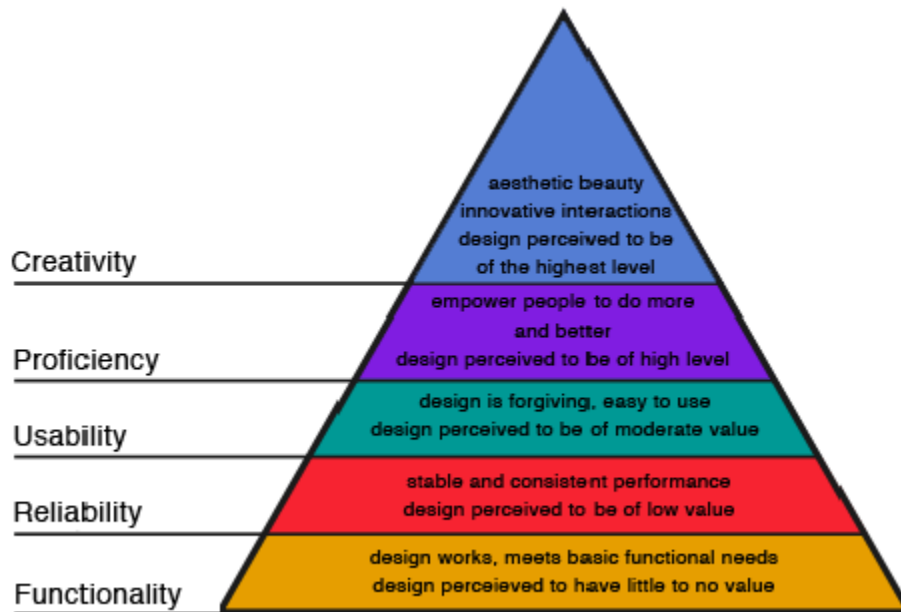


Figure 3.4: Design hierarchy of needs

3.7 Acceptance Models

To test the user's attitude and intention to adopt new technologies, there has been developed a number of models. Technology Acceptance Model (TAM), which is an extension of the Theory of Reasoned Action (TRA), seems to be one of the most accepted models [17]. A key objective of TAM is to provide a basis for determining the impact of external variables on internal beliefs, attitudes and intentions. The TAM suggests that perceived ease of use (PEOU) and perceived usefulness (PU) are the two most important factors in explaining system use. The relationships between the various factors of TAM are shown below in the Figure 3.3.

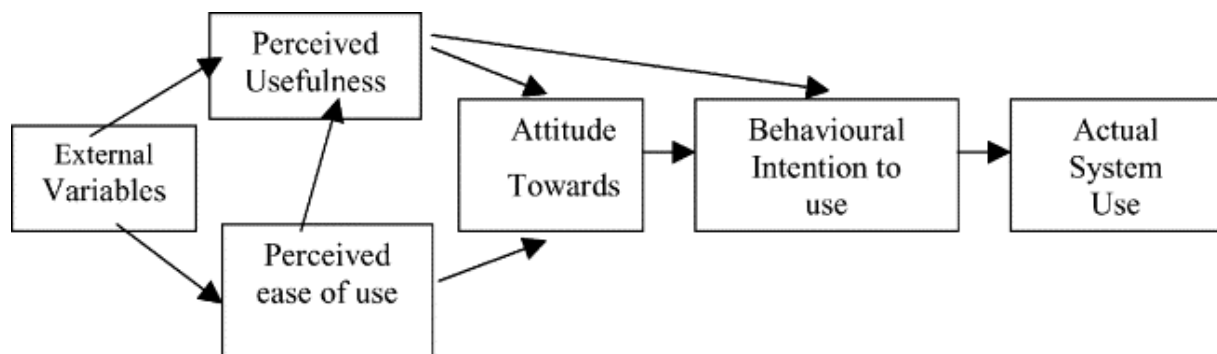


Figure 3.5: Technology Acceptance Model (TAM) [9]

Based on acceptance models and observation of people's views on mobile applications, proposed [17] an extended technology acceptance model called the Mobile Services Acceptance Model (MSAM). The model is based on TAM and it is added with other factors such as context, trust, personal initiatives and characteristics.

The factors Perceived Usefulness and Perceived Ease of Use were embraced from TAM.

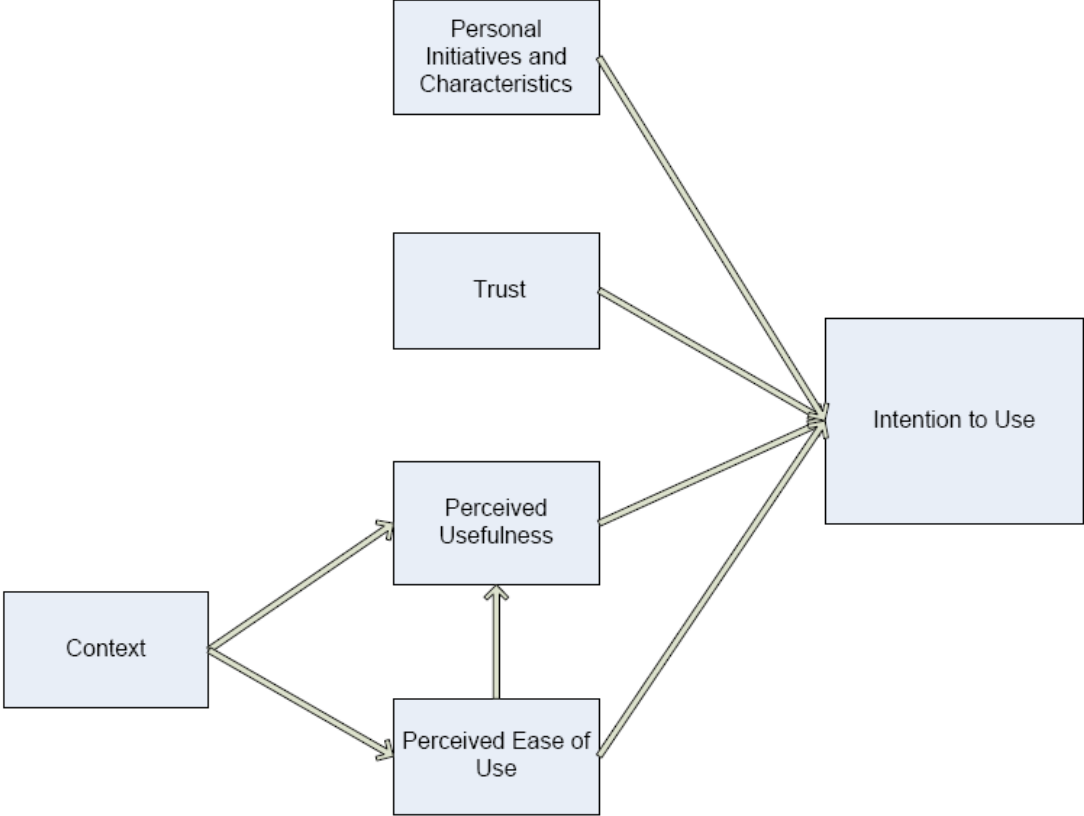


Figure 3.6: Mobile Services Acceptance Model (MSAM) [17]

As shown in Figure 3.4, context (which includes typically location and identity) has a direct impact on Perceived Usefulness and Perceived Ease of Use. For example, if people cannot access desktop computer, they will perceive information systems via mobile devices as useful. Personal initiative can be defined as the user's willingness to try out new applications. Personal characteristics include the elements like age, gender, educational background, knowledge and skills, culture, and preference. Trust can be defined as user's beliefs or faith in the degree to which a specific mobile application can be regarded to have no security and privacy threats. According to [17], perceived usefulness and perceive ease of use have a direct positive effort on user's intention of use.

3.8 Technological overview

To create a web application, one should at least have knowledge of the following technologies: HTML, CSS and JavaScript. These are also used to build a hybrid application, where they are wrapped inside a thin native container that provides access to some native platform features. Therefore, the first part of this subsection provides a brief explanation of these technologies. Then we will look at various JavaScript frameworks and cross-platform development frameworks/tools before we choose some of them to create a web- and hybrid-application on the basis of some specific criteria.

3.8.1 HTML

The HyperText Markup Language (HTML) has been around since 1990s and is the main markup language for creating web pages and other information that can be displayed in a web browser [55]. It defines the structure and layout of the information that are being presented by using a variety of tags and attributes.

HTML5 is the latest version of the HTML standard which tries to bring order, structure, and enhancement to a critical technology that has finally matured after years of difficult adolescence [15]. HTML5 introduces new features such as the `<canvas>` element for 2D drawing, `<video>` and `<audio>` elements for media playback, content-specific elements like `<article>`, `<footer>`, `<header>`, `<nav>` and `<section>`. Other interesting features in HTML5 are new form controls like date, time, email, url, and support for local storage [53].

3.8.2 CSS

Cascading Style Sheets (CSS) is a style sheet language that defines how HTML elements are to be displayed [54]. It is used to keep the content and design (look and formatting) separately. CSS can be connected to an HTML file in three different ways. The usual procedure is to have an external style sheet with all the CSS code and then links it to the HTML document. The code is placed in the head section of the HTML document. The advantage of this method is that one can use the same style sheet into several HTML files. The other way to do it is to write the CSS code directly in the head section of the HTML document between the start and the end tag, which is called internal CSS. The last way is to write it as an attribute directly on the element you want to style; this is called inline CSS.

CSS3 is the latest version of the language, and gives Web developers more flexibility to the design of web pages. The most important feature to CSS3 might be the media queries which allow developers to tailor to different resolutions without having to change or remove content.

3.8.3 JavaScript

JavaScript is a scripting language used to make web pages interactive. The majority of modern websites use JavaScript and all modern web browsers support JavaScript. JavaScript derives its syntax from Java, its first-class functions from Scheme, and its prototype based inheritance from Self [12].

It runs on the user's computer and does not require constant downloads from the website. It also allows the user to carry many tasks relating to the data in the received page, including performing calculations, changing display colors and styles, checking the user input and much more [6].

3.8.4 JavaScript frameworks

Today there are many JavaScript frameworks/libraries out there. Therefore it is not an easy task to choose one among the crowd. We tried three of the most common JavaScript frameworks/libraries to find out which one fit for our project. These were also chosen because we wanted to structure our code for MVC or similar patterns, decoupling logic or data handling from the views, which all these frameworks provide.

3.8.4.1 AngularJS

AngularJS, created in 2009 by Google Inc, is an open source JavaScript framework that uses Model-view-controller (MVC) architecture, data-binding, client-side templates, and dependency injection to create a structure for building web apps [18]. It allows developers to use HTML as their template language and lets them to extend HTML very easily by simply adding attributes, elements or comments. Angular's two-way-data binding and dependency injection eliminate much of the code that has to be written. And it all happens within the browser, which makes it an ideal partner with any server technology [4].

3.8.4.2 *Backbone.js*

Backbone came out in June 2010, and its community is nearly as large as AngularJS. It is a lightweight JavaScript library that allows code to be organized into Models, Views and Collections. Backbone is referred to as an MV* because it does not have the concept of controllers. Rather the functionality of controllers is spread across views and models [27]. Developers commonly use Backbone to create single-page applications (SPAs). Backbone focuses on giving developers helpful methods for querying and manipulating data rather than re-inventing the JavaScript object model. The entire Backbone source is so small that it can be read and understood in just a few hours [37].

3.8.4.3 *Ember.js*

According to the Ember.js website, Ember.js is a framework that enables you to build "ambitious" web applications [10]. It is another JavaScript MVC framework (initially released in 2011) that helps developers to organize and structure the source code for large web applications. In comparison to other popular JavaScript application frameworks, it delivers a more complete MVC pattern which includes the following five concepts: Models, Views, Controllers, Templates and Routers. Ember.js comes with a steep learning curve if the developers are familiar and used to write server-side web applications [45].

Unlike Backbone, Ember.js is much more tightly integrated and suited to larger applications with complex UI requirements [27].

3.8.4.4 *Conclusion*

There is no shortage of choice when it comes to frameworks for web development in JavaScript. Since the main focus of the thesis is not about evaluating JavaScript frameworks, we did not spend so much time on this part. We could probably use each of the mentioned frameworks to get the job done. But we had some requirements/criteria in our mind when deciding on a framework. These were:

- The framework should be easy to learn, so we could have a steep learning curve.
- The framework should have a large community and there should be a lot of material (books, tutorial/videos, blogs etc.) about it in case we needed help.
- The framework should facilitate the organization and structure of the code.
- The framework should help to improve the coding time, i.e. code more efficiently.

In addition to read about and be able to obtain knowledge on the frameworks, we also used TodoMVC. TodoMVC [51] is a site that are made to help developers to choose the right MV* framework for their project. On this site they have created a TodoMVC project that offers the same Todo application implemented using MV* concepts in most of the popular JavaScript frameworks available today.

In the short period that we had to acquire knowledge and try the frameworks, we found the following:

- A characteristic that makes Backbone special is that the framework is remarkably hands-off. This means that experienced developers can quickly get started, but less experienced developers find them writing a lot of repetitive code.
- Ember.js has great strengths such as library size and support network, but if you are only trying to create a single-page application, it might be overkill for you. If you are working on a multipage, navigational, long-term project, Ember.js might be your pick.
- AngularJS, which was released in 2009, is the oldest of the three frameworks. Probably as a result, it also has the largest community. We noticed a lot of news sites using AngularJS on their front pages, like the Guardian, the Huffington Post and MSNBC.

We also used Google Trends autocomplete suggestions to find which of the frameworks is looked for the most in the past year. As shown in the figure below, the most frequently searched framework is AngularJS.

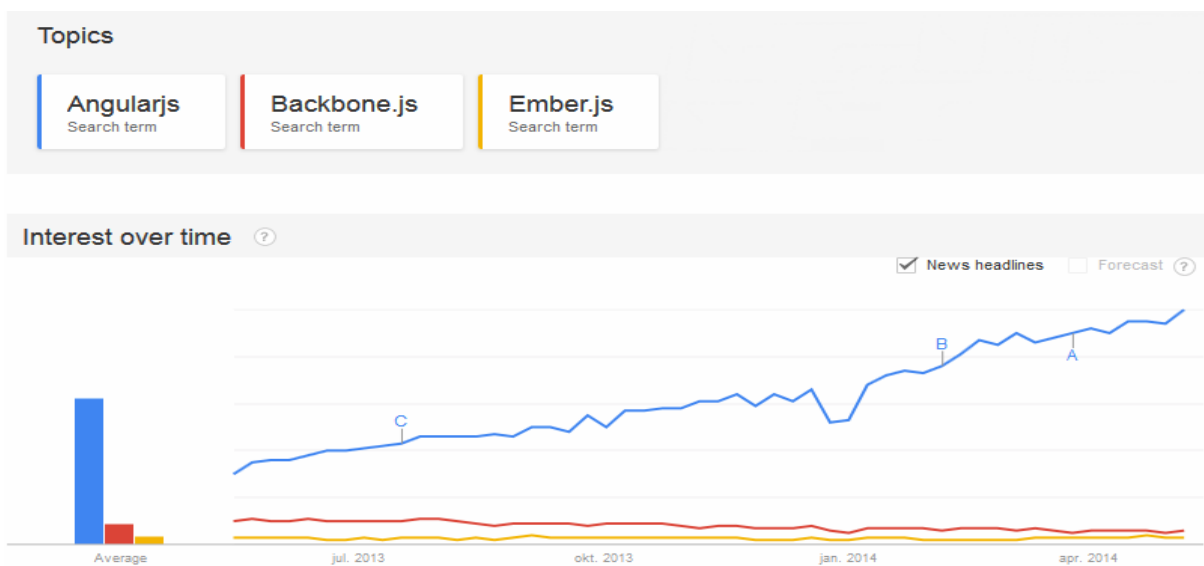


Figure 3.7: Interest over time. Web search. Worldwide, past 12 months (Google Trends)

Based on our founding's, AngularJS met our requirements. Therefore we chose AngularJS for development of the application.

3.8.5 Cross-platform mobile development tools

In the past few years, many cross-platform mobile development tools have emerged. There has been an explosion of activity in this area as mobile devices have become faster and more widely adopted. In this section we will first present two of the most popular cross-platform development tools available on the market. Then we will elaborate on a list of criteria for evaluating these tools. The selection of these criteria is based on and has been influenced by various sources like literature research and a collection of typical problems apparent in online developer communities. Additionally, important experiences regarding necessary features have been obtained from developing prototypical applications. The tool that meets well the criteria will be used to develop the hybrid application.

3.8.5.1 PhoneGap

PhoneGap is an open source cross-platform mobile development framework for building mobile applications using HTML5, CSS3 and JavaScript. It supports the following mobile platforms: iOS, Android, Windows Phone, Blackberry, Symbian, Bada and webOS. The key advantage of creating a native mobile application with PhoneGap is that you can use a mobile web application and build it into a native application that end users may install on their mobile devices. As a native application, it can access certain hardware features/capabilities including camera, geo-location, accelerometer, sound and more using PhoneGap's JavaScript APIs. PhoneGap is well-suited for mobile web application. But like all of the cross-platform frameworks that leverage the browser for UI, it is not well-suited for applications that require intense math calculations or 3-D animations [2].

You have the ability to submit your application to the platform's app store since PhoneGap is wrapping the HTML and JavaScript in a native shell. This is not the case for a typical web application [44].

Your PhoneGap application will be more like a web application than a native application because it will be running in a web browser. The user interface (UI) you design will not use the native controls and will be subject to the limits and speed of a web browser.

The tooling for PhoneGap depends entirely on the environment you want to build the application with. You can develop in whatever environment you would like and basically use a plugin for the IDE in most cases. Some of the most common IDEs such as Netbeans, Eclipse, XCode and Visual Studio have good support for PhoneGap applications.

Pros	Cons
<ul style="list-style-type: none"> • Development skills required only in technologies like JavaScript, HTML5, CSS3 rather than knowing all the languages needed to develop mobile applications for smart-phones. • Cross-platform code that can work for all major mobile platforms. • Support for distribution and integration on app stores. • Access to native hardware capabilities. • Great community around it, good documentation and tutorials. 	<ul style="list-style-type: none"> • Low performance, if the mobile app includes a lot of graphics. • Does not support all native features. • The app will not have native look and feel. • Missing pre-build UI widgets and touch features such as drag, drop, copy, paste, slide etc.

Table 3.2: Advantages and drawbacks of PhoneGap

One big benefit to PhoneGap is PhoneGap build, which takes the pain out of compiling PhoneGap apps. It allows you to upload your project in whatever environment you created in, but build it automatically for the other mobile platforms [38].

There is an active developer community around PhoneGap that is constantly discussing pertinent features of the language.

3.8.5.2 *Appcelerator Titanium*

Appcelerator Titanium is an open source platform for developing native cross-platform applications using web technologies. Source code is released under the Apache 2 license. The platform was introduced in December 2008 by Appcelerator Inc.

Appcelerator Titanium consists of an SDK that provides a platform-independent API to access native UI components including navigation bars, menus, dialog boxes, alerts, and native device functionality including the file system, sound, network and local databases. You code in JavaScript against their SDK which includes UI components as well. This means that you can write a cross-platform user interface [2].

Appcelerator Titanium supports different mobile devices and OSs including iOS, Android, Blackberry, as well as hybrid and HTML5 [5].

For example, in Titanium you can programmatically declare a button and specify its layout and some attributes about that button. When you compile your application, the button will appear as a real native Android and a real native iOS button on iOS. But this does not mean that you can build a completely cross-platform application including the UI with 100% code reuse and do it in JavaScript. Many of the UI elements and interaction paradigms are cross-platform, but some parts of it are not. For example, in iOS you have the idea of a Navigation Controller which keeps track of what screen you navigated through and lets you go back. Android does not have such a control. But, Appcelerator Titanium does have support for platform specific controls. It means that you have to make some of your code conditional based on the platform [47].

Appcelerator Titanium has an Eclipse-based integrated development environment (IDE) that simplifies the development process. It is simple to use and lets you build a web application out of the same codebase. Furthermore, Appcelerator has a development framework called Alloy, which follows a model-view-controller (MVC) architecture and uses XML and CSS. It provides a simple model for separating the application user interface, business logic and data models.

Appcelerator also lets you to have access to their complete backend of cloud services, which is a fast and easy way to build connected mobile applications. Here you can choose from a library of services such as push notification, status updates, photo storage and social integration [5].

Pros	Cons
<ul style="list-style-type: none"> • Robust and efficient. • API that uses available native features. • Uses standard native UI elements and is compatible with touch gestures like drag, drop, slide etc. • Large and fast growing community. Good documentation and a lot of tutorials can be found on their web site. 	<ul style="list-style-type: none"> • Increasing complexity. • Complex toolkit. • All device resources are not available. • New way of coding that takes time to get used to it. • Large cost fee.

Table 3.3: Advantages and drawbacks of Appcelerator Titanium

3.8.5.3 *Criteria*

In order to compare and assess which cross-platform mobile development tool is best suited to the project, we set up the following criteria as shown in the Table 3.3.

As mentioned earlier in the beginning of the chapter, these criteria are based on and strongly influenced by different sources such as online development communities and literature study. One can also find these criteria in the research work done by [22]. In their paper, they structure their criteria into infrastructure and development perspective. Since a detailed evaluation of cross-platform development tools is out of scope for this thesis, we believe that the following criteria are enough for us to take the right decision.

3.8.5.4 *Conclusion/Summary*

There are obviously many more options out there than the mentioned ones, but these two were picked because they are the most serious widely used offerings. Both PhoneGap and Appcelerator Titanium are good choices to make with regard to the development of hybrid applications. They also have weaknesses as well as strengths. In the following, we will describe how they met the criteria in Table 3.4.

<p>Development environment</p> <p>Evaluates how easy it is to set up the development environment for a framework/tool and a desired platform. In addition, it will look at the tool support such as IDE, debugger and emulator that exists for the framework.</p>
<p>Ease of development</p> <p>This criterion summarizes the documentation's quality and the learning-curve. Accordingly the quality of the API and documentation is evaluated. This part of the criterion is accomplished if good tutorials, code examples, link to similar problems, user-comments, etc. are available. The learning curve describes the subjective progress of a developer during his/hers first examination of the framework.</p>
<p>License and costs</p> <p>This criterion examines whether the framework is distributed as free software or open source. It will look into the cost for creating commercial software and whether costs for support inquiries appear.</p>
<p>Supported platforms</p> <p>Considers the number and importance of supported mobile platforms. It will focus on whether the solution supports the platforms equally well.</p>
<p>Access to platform-specific features</p> <p>Includes access to device hardware features such as camera or geo-location and to platform functionalities like contacts or notification. It will also compare them according to the API and the web site of the framework.</p>
<p>Distribution</p> <p>Evaluates how easy it is to distribute the created app with the framework. Furthermore, this criterion assess whether updates are possible.</p>

Table 3.4: Criteria for evaluation of the cross-platform development tools

Development environment

Developers are not limited in their choice of development environment when using PhoneGap. However, not all IDEs offer auto-completion for PhoneGap's API. PhoneGap has PhoneGap Build which is a service that compiles an app for different platforms in the cloud, so that developers do not have to install the platform SDKs. After providing the source code

of a PhoneGap app, the app can be compiled and signed for all chosen platforms which can easily be downloaded.

Appcelerator Titanium is tightly integrated into Appcelerator's IDE called Titanium Studio, which is based on Eclipse. Titanium Studio offers auto-completion for the whole Titanium API. Furthermore, it supports deployment to emulators or devices as well as distribution to app stores. Even though that the platform SDKs have to be installed separately, setting up the development environment for Titanium is straightforward.

Ease of development

Both frameworks have a good and comprehensive documentation that clearly are structured. The documentations provide a number of code samples which make it easier to begin with. Compared to PhoneGap, Titanium is harder to get used to and the progress is relatively slow, since a high degree of framework-specific knowledge must be acquired.

License and cost

PhoneGap is distributed under Apache License Version 2.0, and commercial software can be created free of charge. But it costs to deploy and compile your application to other platforms using PhoneGap Build [39].

Appcelerator is licensed by Apache Public License v2. The standard Titanium framework is free and open source. But the enterprise editions which offer additional functionality are only available with a subscription.

Supported platforms

While PhoneGap supports seven mobile platforms (iOS, Android, Blackberry, Windows Phone, WebOS, Symbian and Bada), Titanium supports only five platforms (iOS, Android, Blackberry, Tizen and Mobile Web).

Access to platform-specific features

Both frameworks provide easy access to most platform-specific features such as accelerometer, geo-location, storage, etc. These are more than enough for our case.

Distribution

Both PhoneGap and Titanium apps can be distributed via app stores without problems.

As we see above, PhoneGap meets the criteria better than Appcelerator Titanium and seems to be the preferable option for cross-platform development. Therefore we chose PhoneGap for developing the hybrid application.

4 Related works

This chapter will give an overview of the state of the art for mobile cross-platform application development tools. In addition, topics like personalization and designing mobile news application will be presented.

4.1 Personalization on mobile devices

Mobile devices have become a primary platform for information access. As more and more people use mobile devices for communication and information retrieval, and the amount of information and online services increases, it becomes difficult for users to find the right information needed to complete a task. Recommendation systems (RSs) can contribute to address such problems by providing recommendations tailored/personalized to users' needs and preferences.

According to Ricci (2010), recommendation approaches that prove to be successful for pc users cannot be straightforwardly applied to mobile users. On the one hand, mobile RSs must overcome the barriers that usually are present in mobile usage environments: the limitations of mobile devices, limitations of wireless networks, the effects of the external environment and the behavioral characteristics of mobile users. On the other hand, the mobile RSS can utilize two distinctive characteristics of mobile information systems. The first property is "location-awareness", i.e. knowledge about the user's physical location at a specific time which can be utilized as an important source of information to adapt the information delivered by the system. The second property is "ubiquity", i.e. the ability to deliver information and services to mobile users wherever they are, and whenever they need it [40].

Ricci (2010) lists several issues that must be considered when designing RSs for mobile devices:

- Compared to desktop computer, the screen is smaller which can impact on users' performance. For instance, on a small screen the user may be forced to scroll a lot and the more a user scroll down, the smaller the chances are of an item being clicked.
- Mobile devices offer limited input and interaction capabilities which makes the effort of querying higher than on a regular keyboard.

- Mobile Internet browsing sessions are often shorter than on desktop computers, i.e. in the range of some minutes.
- The cost of retrieving data using mobile data through Edge or 3G is usually higher than on personal internet connections at home or office.

4.2 Designing news application

In the past, researchers have looked into generating a personalized webpage of relevant news based on the users' needs and preferences.

One of the early studies on presenting an interactive newspaper online was conducted by Kamba et al. (1995). They proposed a system that automatically downloads articles as the user is reading the newspaper. In addition to being able to browse and rate an article, the user can also control the layout of the page flexibly [26].

Anderson & Horvitz (2002) designed and developed a system called Montage. The Montage system builds personalized web portals for its users. It learns which pages are being seen regularly at a certain time period and presents content based on the user's interests and browsing pattern [3].

A lot of research has also taken place in the realm of interface design. For instance, Shneiderman (1994) suggests the use of dynamic queries to show information and update search results in response to movements of sliders, buttons or other UI elements/widgets [43]. Teitler et al. (2008) present a system called NewsStand which collects, analyzes and displays news stories in a map interface. NewsStand retrieves articles by monitoring RSS feeds from different online news sources. It provides users with geographic perspective of where the news coming from and helps them to explore news based on location [49].

Marshal (2007) suggest RSS based news reading technology, the Times News Reader application to enhance the readership of traditional newspapers by presenting news in a personal and searchable interface, leading to a better reading experience. The application has several functionalities that one associates with news websites, such as the ability to share and save the articles. Additionally the users can download the entire newspapers overnight, so that they can read it offline [29].

Benckert van de Boel (2011) presents a recommendation on how to design an aggregating news application on iPad called 247. In her paper she focuses on combining the editorial credibility, a social media layer and a fluent interaction experience in order to provide

overview and accessibility. A major concern is how to move the conventional newspaper into a digital medium, and how the digital medium can be used in everyday life to access news, either as an individual keeping up on the latest news or as a peripheral device presenting the latest headlines to the users [7].

In her research she does a field work that examines how people use the iPad as a device, and how, when and where people have access to news during the day. This results in a design that allows both individual use and peripheral use as shown in figure 4.1 and 4.2.



Figure 4.1: Screenshot from 247 showing the individual UI design along with an example of individual usage [7]



Figure 4.2: Screenshot from 247 showing the peripheral UI design along with an example of peripheral usage [7]

[52] present a news service called News Sync, a system that addresses the following scenarios where a user wants to:

- catch up on news from a particular time period.
- keep in touch with news from specific locations.
- follow the lives of celebrities.

They propose a use-case driven approach to create an appealing, sticky news consumption experience, using techniques from search, language processing, visualization and learning.

Figure 4.3 shows a screenshot of News Sync after a search for "Watergate" has been executed. The left panel lists the clusters ordered by popularity and relevance. It displays additional properties about the cluster, such as tag clouds showing keywords and locations related to the news articles. The right panel however gives supplementary information about the highlighted cluster. It shows a brief summary, followed by a list of relevant articles. The list shows metadata such as headline, publication date and lead paragraph for each article.

The News Sync's interface allows users to share summary, articles or stories on the most popular social networking sites. It also gives users the opportunity to save their queries and results for later access.

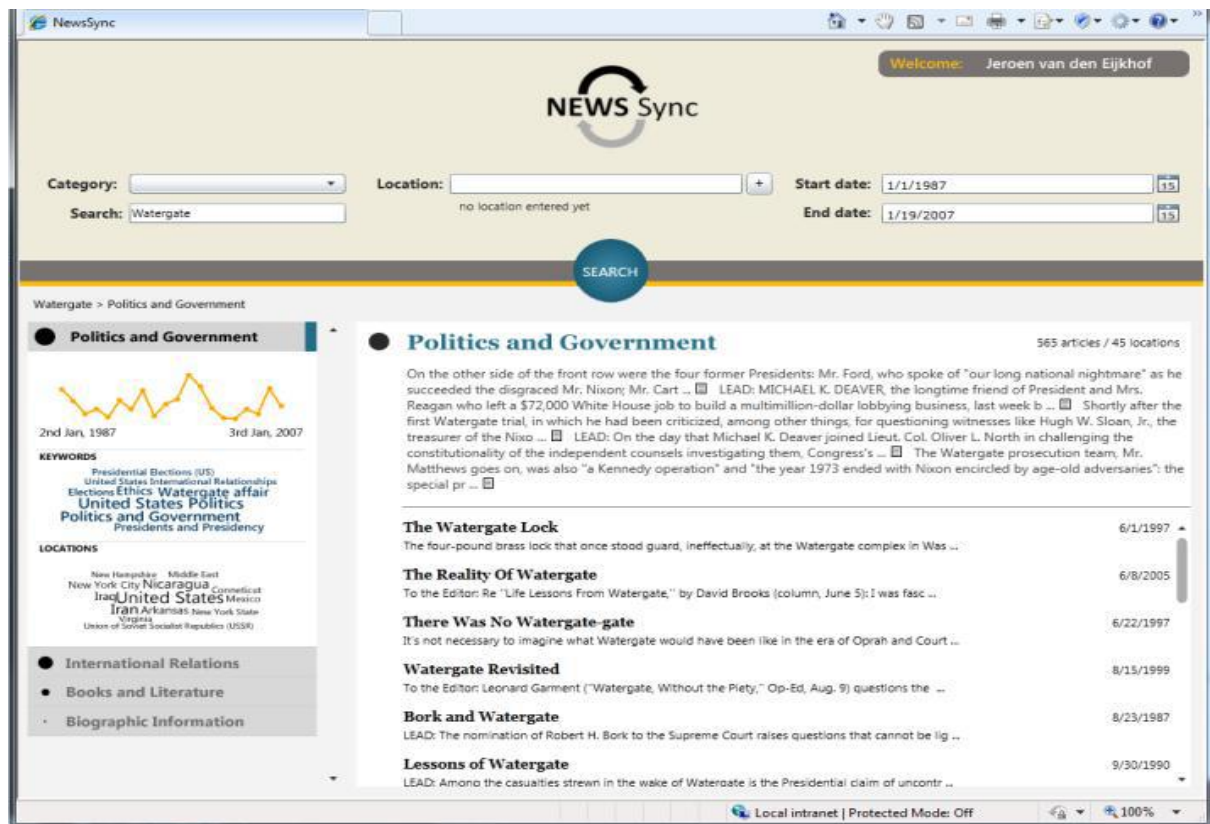


Figure 4.3: Screenshot of News Sync showing results for the query "Watergate" in the catching-up scenario [52]

4.3 Mobile news applications

At the moment of writing there are dozens of news apps out there. In this sub-chapter we will look at some of the most famous and popular apps that stand out from the crowd. There will be given a brief description of these apps and the way they look and work.

4.3.1 Flipboard

Flipboard is a personalized news aggregation app for iOS, Android, Windows 8 and Blackberry that collects content from social media and other websites. It lets users pick topic they are interested in as well as popular publications and sorts all of that news by category. Users can create, edit and share magazines via social networks or with other Flipboard users. The users have also an option to select a specific area of interest and the app will provide them with news related to that. The app offers syncing which gives the users access to their top news on any device.

The application's user interface is easy to use. It is designed for intuitive flipping through content. One can search for everything such as people, topics and favorite sites, and then flip through them in a magazine-style format [14].

Flipboard was called "iPad App Of The Year" by Apple in 2010 [57].

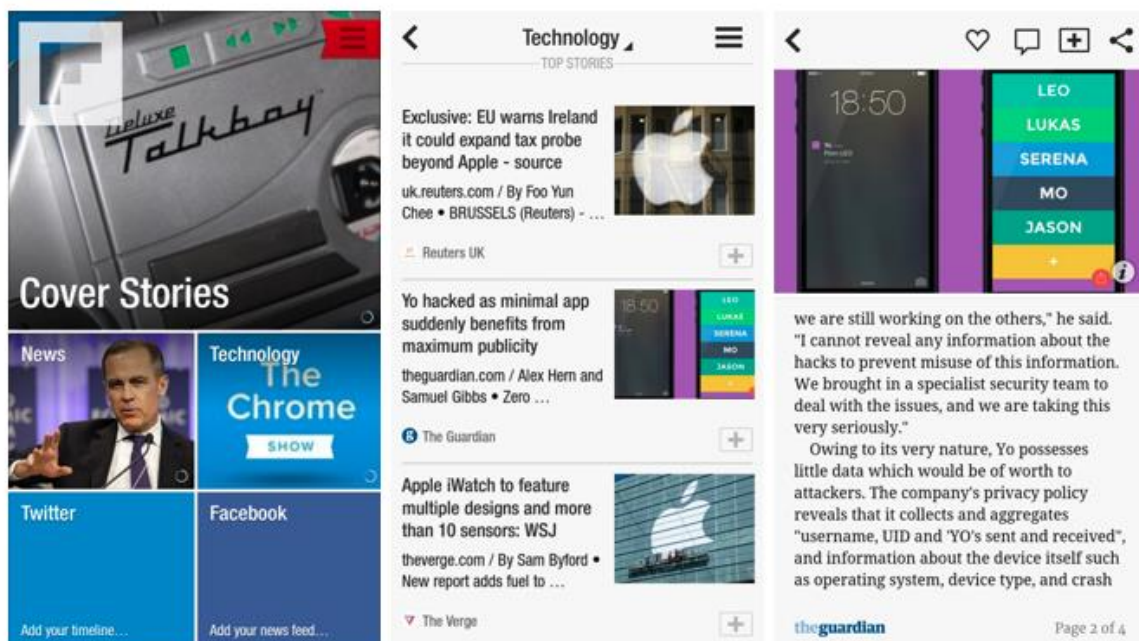


Figure 4.4: Screenshots from Flipboard showing top categories, articles related to the category "Technology", and a single news article

4.3.2 Pulse

Pulse is a news application for iOS, Android and web browsers that supports HTML5. It was selected as one of TIME's top 50 iPhone apps for 2011 and honored with the Apple Design Award [28].

Pulse aggregates the users' favorite blogs, magazines, social networks and newspaper, so they can follow top relevant contents from different sources. The users can sync with social networks such as LinkedIn, Facebook, Twitter and Instagram. In addition to give the users the ability to save stories, it also supports offline sync so the users can read news stories whenever and wherever they are. Since Pulse syncs the users account across all devices, the users can for instance start to read a story on their smartphone and finish it later on their tablet. The users can customize feeds based on their interests to get fresh, personalized content delivered daily.

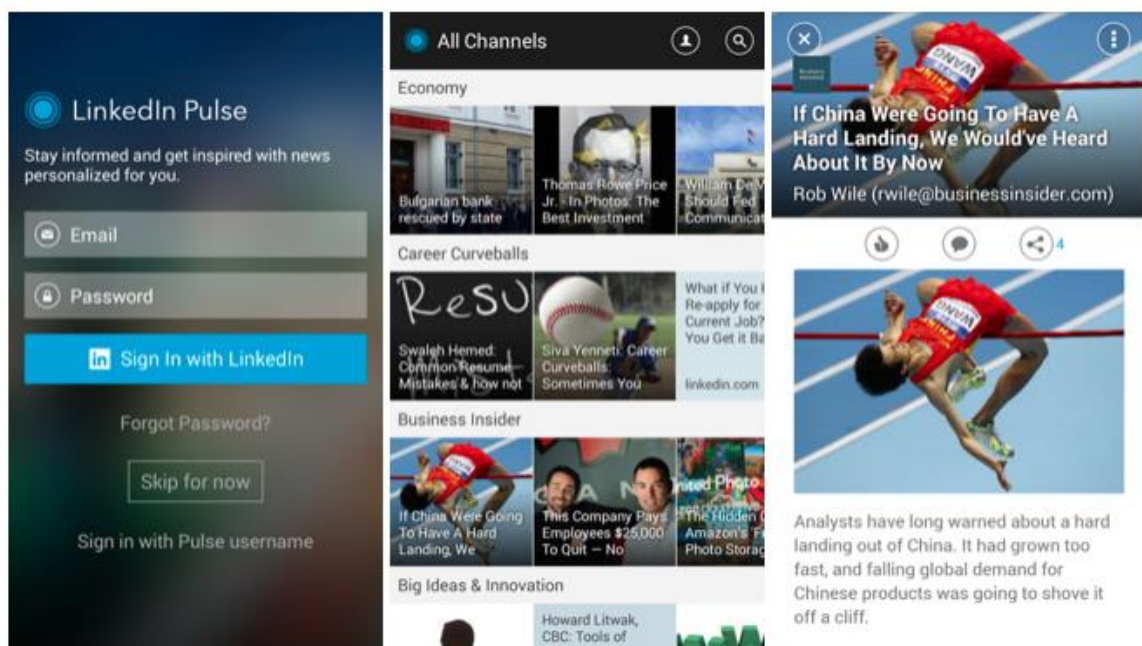


Figure 4.5: Screenshots from Pulse showing the start page, list of categories, and a single news article

4.3.3 Taptu

Taptu is a social news reader that lets users pull all the news they are interested in into one app, plus content from social network like Facebook or Twitter. If the users are logged in to the app via Facebook, Twitter, LinkedIn or Google Reader, they can sync their content

between all their devices and they can share news or streams they like with friends via these social networks or email.

Besides letting users to access a large number of news sources in an easy to read interface, it also allows them to personalize and customize their feeds. Taptu is available for iOS, Android and web.

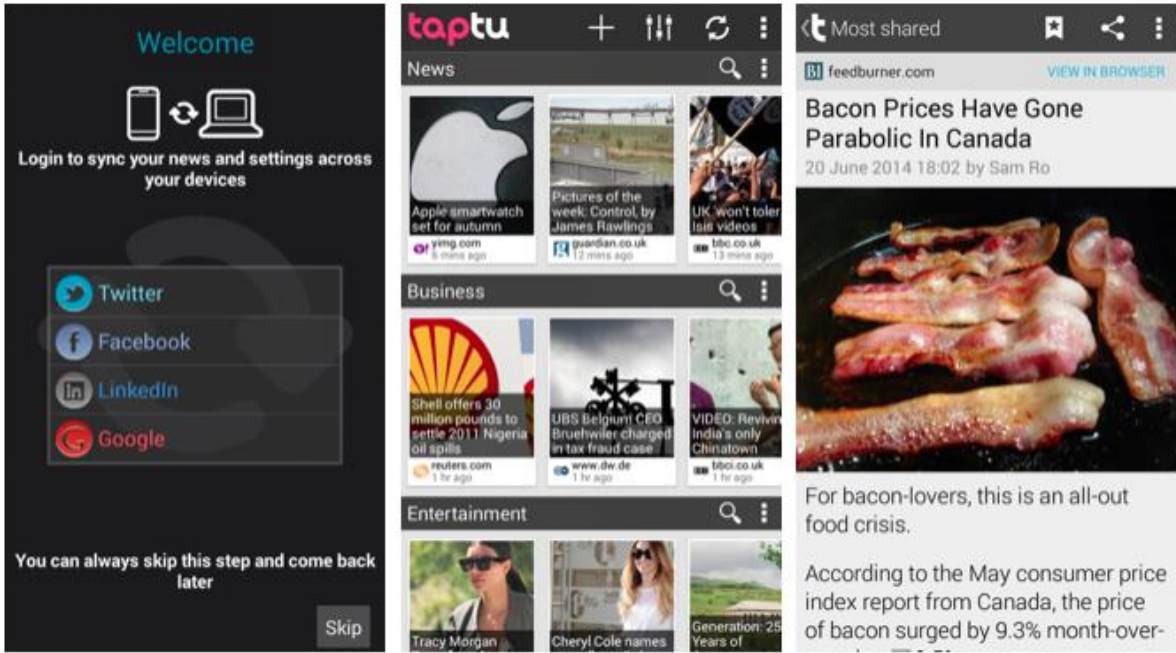


Figure 4.6: Screenshots from Taptu showing the welcome page, categories and a single news article

4.3.4 Zite

Zite is a personalized magazine available for iOS, Android and windows phone. Based on the users' interests and reading habits, Zite learns what to send in the future and delivers the news, articles, blogs, videos and authors that they like. To indicate what they like, users can thumb articles up and down, and Zite will get more personalized as they go. The app features more than 40000 topics to choose from [59].

Zite supports offline article viewing by offering users the ability to share articles to a third party service like Instapaper, Pocket, Evernote or Email. The users can also view the last 50 articles that they have viewed as well as access to all of the articles that they have thumbed up/down or shared.

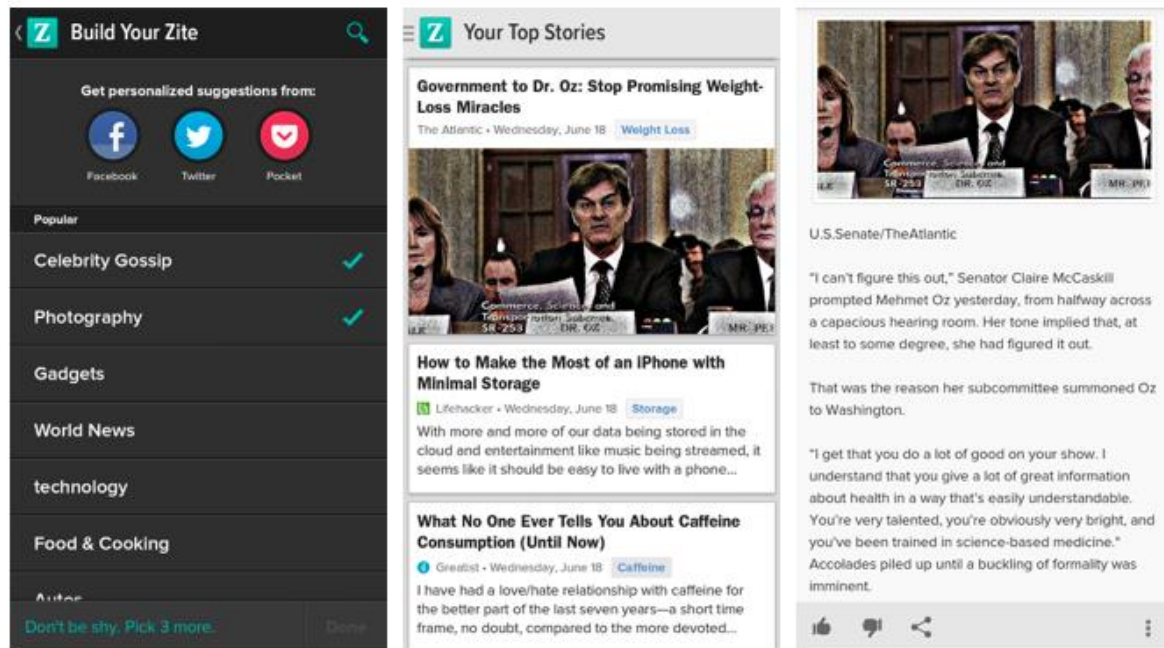


Figure 4.7: Screenshots from Zite showing the categories, top stories and a single news article

4.3.5 News Republic

News Republic lets users get a visual overview of the latest news that matters to them. Users can even personalize their home screen by moving topics around screens and resizing their icons. It displays Articles lists that allow for quick news browsing while an improved catalog of popular choices and sources expand the depth of news users can view.

News Republic learns what the users like and personalize their news experience automatically. Users can read any article and share it with friend by email, Twitter or Facebook.

4.3.6 Newspapers

Besides the applications mentioned above, we have studied and looked at both national and international newspaper sites such as Aftenposten, Adressa, VG, BBC, CNN and The New York Times. None of them are designed as a responsive web application. Instead they contain multiple versions of the same web page that supports both tablet and smartphones, and they request the web page via browser with very similar behavior.

The differences between the newspaper sites and the commercial apps is that the newspaper sites does not offer the users the opportunity to create their own profile where they can tailor

the content or decide what kind of news they want to see more of. Nor they do give the users the ability to share articles via social media. The newspaper sites simply offer different news content in form of articles and videos. Furthermore, besides showing news content, there are a number of commercials in their sites.

What is common for both commercial apps and the newspaper sites is that they divide the articles into categories. Additionally they present the articles more or less equally. This means that every article shown in RSS view has data like headline, photo or a brief summary of the article. While in the full text view, they show all the information about each article.

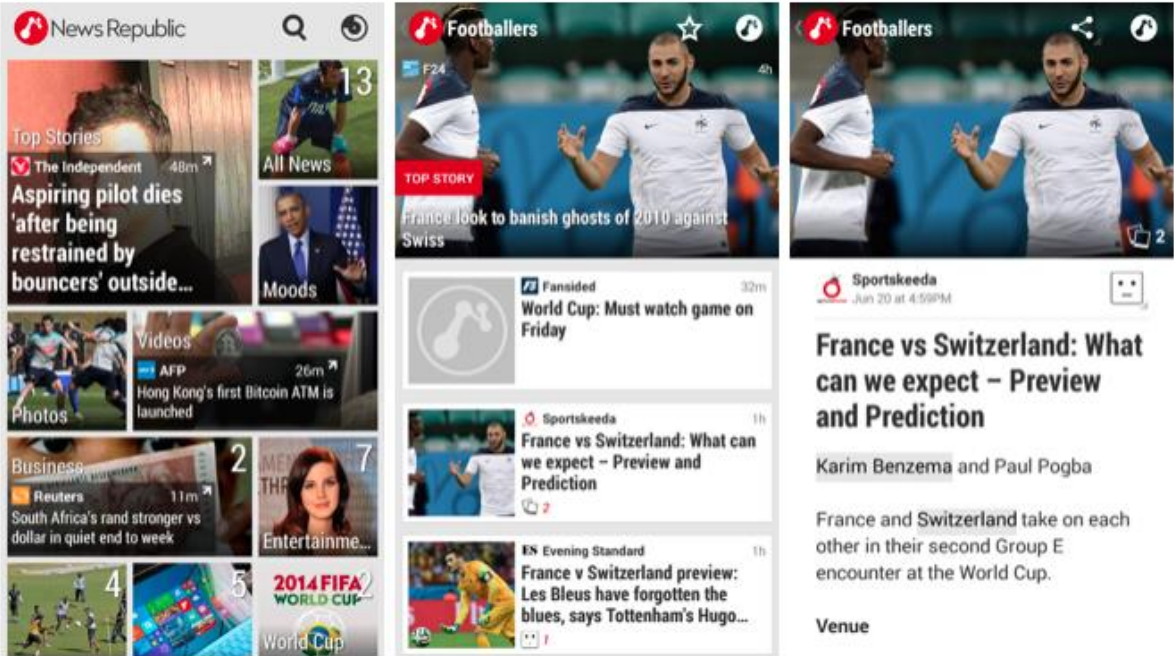


Figure 4.8: Screenshots from News Republic showing the categories, articles related to one category, and a single news article

4.3.7 Features of the popular news apps

Many of the news apps that were used and studied during the project have much in common in terms of functionalities that they offer. First of all, you must create a profile in order to use all the app's features. One common feature is that users can select news articles that are most relevant or interesting to them. As users have used the apps for a while, then they get personalized news based on their interests, preferences or reading habits. Since the apps are easy to use and require less explicit information from the users, this led the users to spend

time on what is really important to them, namely to read news. Among other things, many of the apps support offline article viewing, which means that users have access to the articles without having internet access. Additionally, users have ability to share articles via social networks like Facebook or Twitter. They also allow users to store and sync their favorite items across their devices, so they can access them anytime and anywhere. Last but not least, almost all of these apps have a search feature which let users to search for content.

A feature that is only reserved News Republic is that it shows entities inside articles, which allows users to click on those entities to redirect them to related news stories.

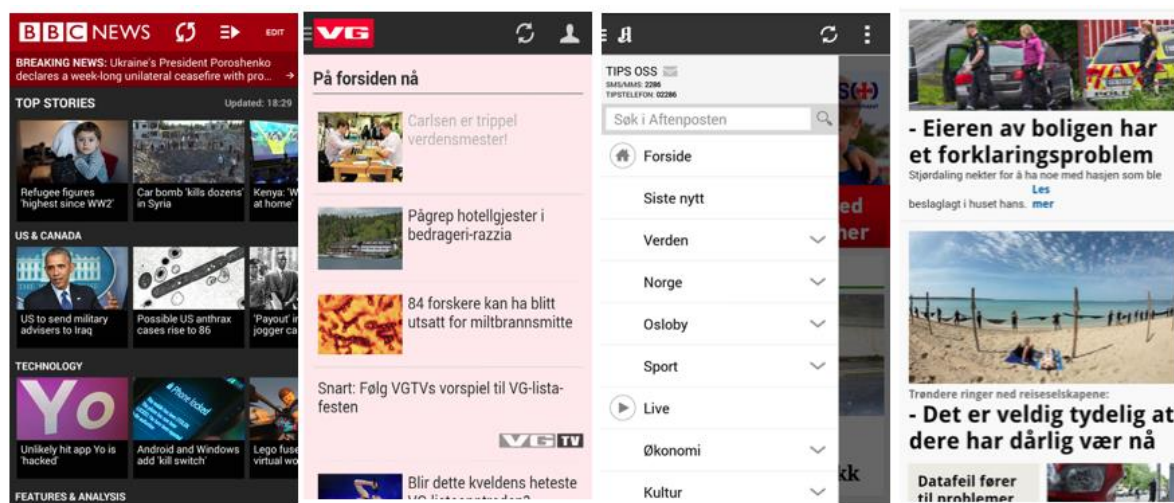


Figure 4.9: Screenshots of some newspapers

4.3.8 User experience aspects of the popular news apps

One of the most important reasons that these apps are among the most popular news apps is that they have a user-friendly and intuitive user interface. These apps have much in common in the way they present the news. After launching the apps, users are in most cases greeted with an RSS perspective that keeps several lists of different categories and feeds. The categories/feeds are often presented in square boxes where image and category name are shown. After selecting a category, one is sent to the articles belonging to the chosen category. In this view, each article contains elements such as image, headline, lead text, publisher and date. The articles are frequently ranked by publication date. Each page shows between 2 and 6 articles. If users want to see more articles, they can swipe to get more related news stories. By clicking on the desired article, one is presented with the full article perspective, where all information about the selected article is displayed.

In the full article perspective, users are also able to like, comment, save and share article through social networks. To perform such actions, one can click on the buttons (with icon) that are shown at the bottom of the article page. Users have access to the menu and the search box (which are shown at the top bar of the page) anywhere in the app except in full article perspective.

There is also useful thing that is not offered by all the apps. For instance, the News Republic app is the only one that allows users to change the font size of the articles.

4.3.9 Reflection

The mentioned news apps are popular and successful because of their user-friendly user interface and the features that they provide. All these contribute to a good reading experience for the users, which in turn makes them satisfied to use the apps.

We have got inspiration from both the commercial apps and the newspaper sites after trying them out. The way the Smart Media application looks and behaves is strongly influenced by the knowledge and the lessons we have got after studying these application's design and functionalities.

Part III

5 Application life cycle

This chapter starts with presentation of the scenarios and use cases that were used to develop the functional and non-functional requirements for the application. The functional and non-functional requirements are presented in section 5.2. The chapter ends with a presentation of the POC prototype.

5.1 Scenarios and use cases

This section explains the system by proposing two scenarios that demonstrate the application in use. The scenarios helped to form a basis of features that could be of interest to implement. The personas in the scenarios are made to try to detect user groups of the system. These are fictitious profiles of people who would use the system and was used to better understand the needs of the various stakeholders.

5.1.1 Scenarios

Scenario 1

Ole is a 20 year old computer science student from Trondheim. He is very interested in technology and has many gadgets. Besides his desktop computer at home, he also has a laptop, a tablet and a smartphone. In addition to spending a lot of time to be active in social media, he likes to read news and catch up on what's going on in his area and elsewhere in the rest of the country. He is a little skeptical and does not always believe in what is being mentioned in the newspapers. This causes that he often checks several newspapers daily to get an accurate picture of an issue that interests him. Furthermore, he often posts news articles on social media like Facebook and twitter to get his friends' and others' opinions on such matters. But all in all, this is a demanding task and takes a lot of time. Therefore he begins to check online for a news application that makes this task a little bit easier.

He comes across the Smart Media application and tries it out. He sees that the application gives him the opportunity to share the articles he read on social media. He thinks it is cool. Now he does not need any more to copy and paste the url of the articles to share it with others. Instead, he just uses a few keystrokes before everything is shared without switching between different pages/sites.

Scenario 2

Kari is a 60 year old teacher from Oslo. The school that she works at is located in the downtown area and do not have many free parking lots. Therefore she uses public transport. She usually tends to read the news on her laptop while she is at home, but it also happens that she uses her smartphone when she is sitting in the bus on the way to and from school. One morning before she goes to school, she keeps on reading a case concerning Norwegian schools compared with other European schools. To catch the bus, she reads quickly through the entire article. After a hard working day, on the way home she takes up her mobile phone to read news instead of being bored. She opens a news story and start reading. After reading halfway, she notices that she reads the same article that she read this morning. Because of the busy work day, she often tends to read the same article twice. She wishes that there was something that could help her with this problem.

The next day at the staff room, she hears from a younger colleague about the Smart Media application. She opens the Smart Media application by entering the url on the browser and creates an account there. She tries reading a few articles and finds out that the application shows her the read articles in an intuitive manner. She thinks that the application is amazing and become satisfied by using it.

5.1.2 Use Cases

While a user scenario describes a real-world example of how one or more people interact with the system, a use case typically refers to generic actors. Use case represents an objective user wants to achieve with a system. The use case diagram gives a graphical overview of the functionality provided by the system in terms of actors and their goals. The textual use cases provide a detailed description and explain the basic and alternative flows of the events.

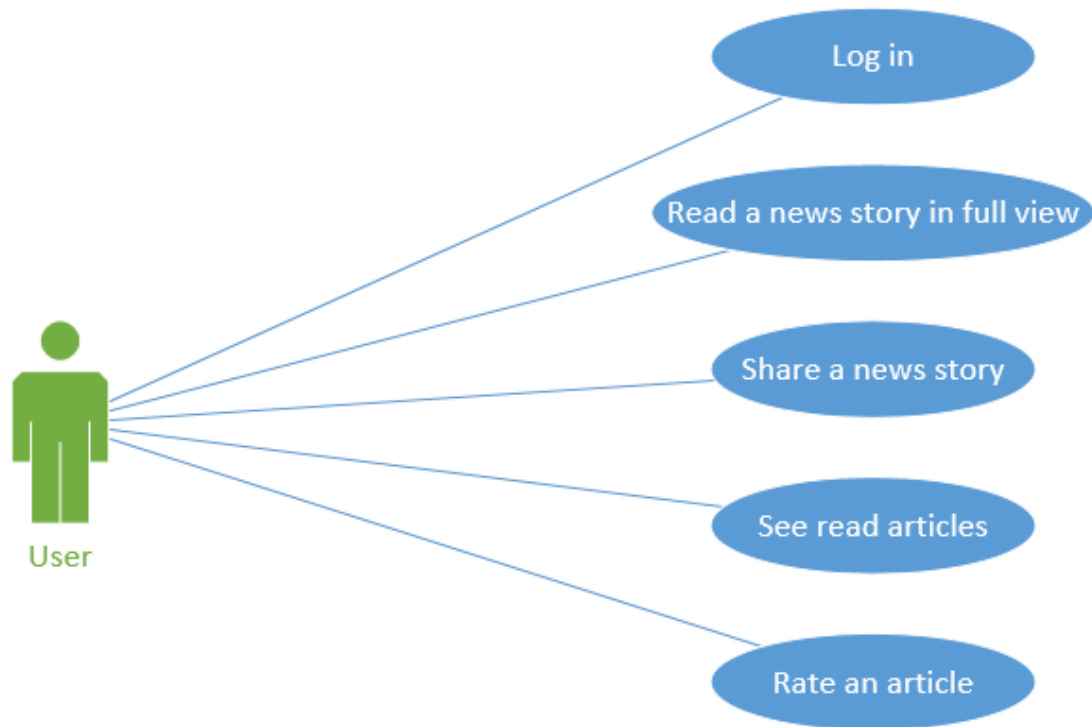


Figure 5.1: Use case diagram

Below are shown the textual use cases for two of the most important events.

	Read a news story in full view
Description	The user wants to read the chosen article in full view.
Precondition	The user has started the application and is logged in.
Basic Flow	<ol style="list-style-type: none"> 1. The user presses the "Start Reading"- button. 2. The application redirects the user to the news page. 3. The application displays a list of news articles. 4. The user chooses the desired article. 5. The application navigates the user to the chosen article. 6. The user sees the article in full view. 7. The user closes or logs out of the application.
Alternate Flow	<ol style="list-style-type: none"> 1. In step 1, the user can also press the left side-menu button and then press the News-link to go to the news page.

Table 5.1: Textual use case 1

	Share a news story
Description	The user wants to share a news article on social networks.
Precondition	The user has started the application and is logged in.
Basic Flow	<ol style="list-style-type: none"> 1. The user presses the "Start Reading"- button. 2. The application redirects the user to the news page. 3. The application displays a list of news articles. 4. The user chooses the desired article. 5. The application navigates the user to the chosen article. 6. The user sees the article in full view. 7. The user clicks on the "thumb up" link. 8. The application slides out the social media buttons. 9. The user presses on the desired button (for example the Facebook icon). 10. The application opens up a popup window. 11. The user fills out necessary info before clicking on the share button. 12. The application shares the article on the social network (in this case Facebook). 13. The user closes or logs out of the application.
Alternate Flow	<ol style="list-style-type: none"> 2. In step 1, the user can also press the left side-menu button and then press the News-link to go to the news page.

Table 5.2: Textual use case 2

5.2 Requirements (functional and non-functional)

The Smart Media application is developed and based on a set of requirements and specifications. This section describes the functional and non-functional requirements for the application.

5.2.1 Functional Requirements

FR1: The application must be able to consume data and news content from the back-end API.

FR2: With a unique email address users must have opportunity to register a valid user account (Sign up).

- Email address should not be pre-registered inside Smart Media database.
- At least a six characterized password must be chosen.

FR3: With a valid user account, users must have opportunity to log into the Smart media app (Sign in).

FR4: Sign up should automatically cause sign in of the user.

FR5: When signed in, users must have opportunity to register/update information (like the one shown below) about them self.

- First name
- Last name
- Birth date
- Occupation
- Gender
- Place of residence

FR7: The application must visualize news articles based on different views.

- Main view RSS (multiple)
 - Users must be able to see more than one article
- Full page article view (single)

FR8: The application must produce data such as user activities, duration of time spent reading an article and user rating of articles.

- Time spent reading an article should contain following data:
 - User id
 - Article id
 - Time occurrence of the event
 - Time spent while users are located on the article page
- Rating an article should include:
 - User id
 - Article id
 - Rating value

The generated information about these events must be posted to the back-end server and be stored in the database for future analysis.

FR9: Users must have opportunity to rate articles based on five star category rating system.

The rating system must be based on the following:

- Hated it (One star)
- Disliked it (two stars)
- It was different (three stars)
- Liked it (four stars)
- Loved it (five stars)

FR10: Users must have opportunity to share an article on social media like Facebook, Twitter, LinkedIn and Google Plus.

FR11: The application must localize and visualize the location of both the user and the occurrence place of an article, simultaneously on a single map.

5.2.2 Non-Functional Requirements

NFR1: The application must be designed based on responsive design principles.

- Page content and layout shall response and react to both horizontally and vertically changes on the screen (self-changing content).

NFR2: The application shall complete several logical front-end tasks simultaneously without interfering each other or the user.

NFR3: The application shall offer the same user experience on different platforms, devices and browsers.

NFR4: The application shall give users feedback based on their interaction and application state such as error.

NFR5: The application should be easy to use and learn.

- Necessary information should be readily available
- Easy to navigate through
- Recognizable behavior, elements and widgets
- Access to main functionality through multiply access points

NFR6: Users shall have the minimal amount of click before performing or completing a task.

NFR7: The application shall have a low threshold for cross-domain requests such as:

- User registration
- User authentication
- News article consuming
- Profile update

NFR8: All cross-domain requests shall occur asynchronously without locking or freezing the GUI.

5.3 Presentation of the Proof of concept (POC) prototype

The Proof of concept prototype were developed to determine possible errors and failures of chosen system architecture, designs patterns, frameworks/development tools, fast user testing, generation and collection of user data.

Due to limitations of time, the main requirements that had to be completed in order to generate data were implemented. POC established foundation for the final application. It was developed as a Single-page application (SPA) to provide a better user experience and performance. SPAs can benefit of redrawing specific parts of the user interface without refreshing/reloading the entire page or requesting new templates. This behavior increases the performance and minimizing the document object model (DOM) dependency.

Based on predefined scenarios and tasks, users were asked to follow and complete the tasks. Feedback from test-users was collected and analyzed; new functional and non-functional requirements were added on or customized. Because of its responsive design capability, the application were accessed and launched through a web browser on different devices.

5.3.1 POC description

The prototype was developed with JavaScript (AngularJS and jQuery) for front-end logic, HTML5 and CSS3 with scheme template libraries for the layout and style of the prototype. The idea was to discover advantages and disadvantages with the chosen frameworks and tools as fast as possible in the application lifecycle. Following the directions and customs of each framework, the core functionality of the application was developed.

The POC application contains five different views:

- Index
 - Where the application normally launches and deploys from. All necessary files, images and templates will be requested by the browser and cached for later use. In this view, there is nothing for the user to click on. Only a logo and an empty progress-bar with a completion percentage that is visible to the user.

While loading, the percentage and progress-bar content increases. This indicates the user to wait. When this process is completed, the content of the page is set to the Home view.

- Home
 - This is the applications welcome view; users have access to the navigation and application menu, quick-links and information about the application. The background automatically changes with randomized pictures. New users have the opportunity to create a membership account through the app menu or quick-link at the end of the page. Members can sign in through the same process. The news articles can be accessed in three different ways; through the menu, the button that are placed in the middle of the view or quick-link at the bottom of the view. By clicking on either of these elements, the content of the page is set to the News view.
- News
 - This is the application's main view; at this moment the application is requesting/downloading news articles asynchronously from the back-end server. This action is indicated with animation. When the download is completed, the animation ends and the content of the view appear to the user. The recent top-ten news articles containing image is placed at the top of the view. Users can shuffle through these articles by using the left or right arrow. The articles title and lead text is placed over each image. This is the Smart media top-ten carousel-bar. Beneath the carousel follows the rest of downloaded articles in individual containers with image, title, publisher and date of publicity. At the center of the view, a management button is placed where users can either update (forcing a new request to the back-end server, looking for new news stories to download) or get more articles (adding more news stories to the page from the server). Both the carousel and these article containers act as link to the article view. By clicking on them the content of the page is set to the Article view.
- Article
 - Entering this view will present the article to the user in Full-article and Map/geo-location perspective. The left side of the view containing the title, image, publisher, time, lead- and article story, and the place related to the article on a map. The right side of the page is the articles control panel; users

can see the amount of time used on the view, rate or share the article on social media. Only signed-in user's rating value will be submitted to the back-end server. Other users will get a warning, reminding them to sign in.

- Settings
 - Only accessible through the navigation menu and to the signed-in users. Users can add or edit information about them self.

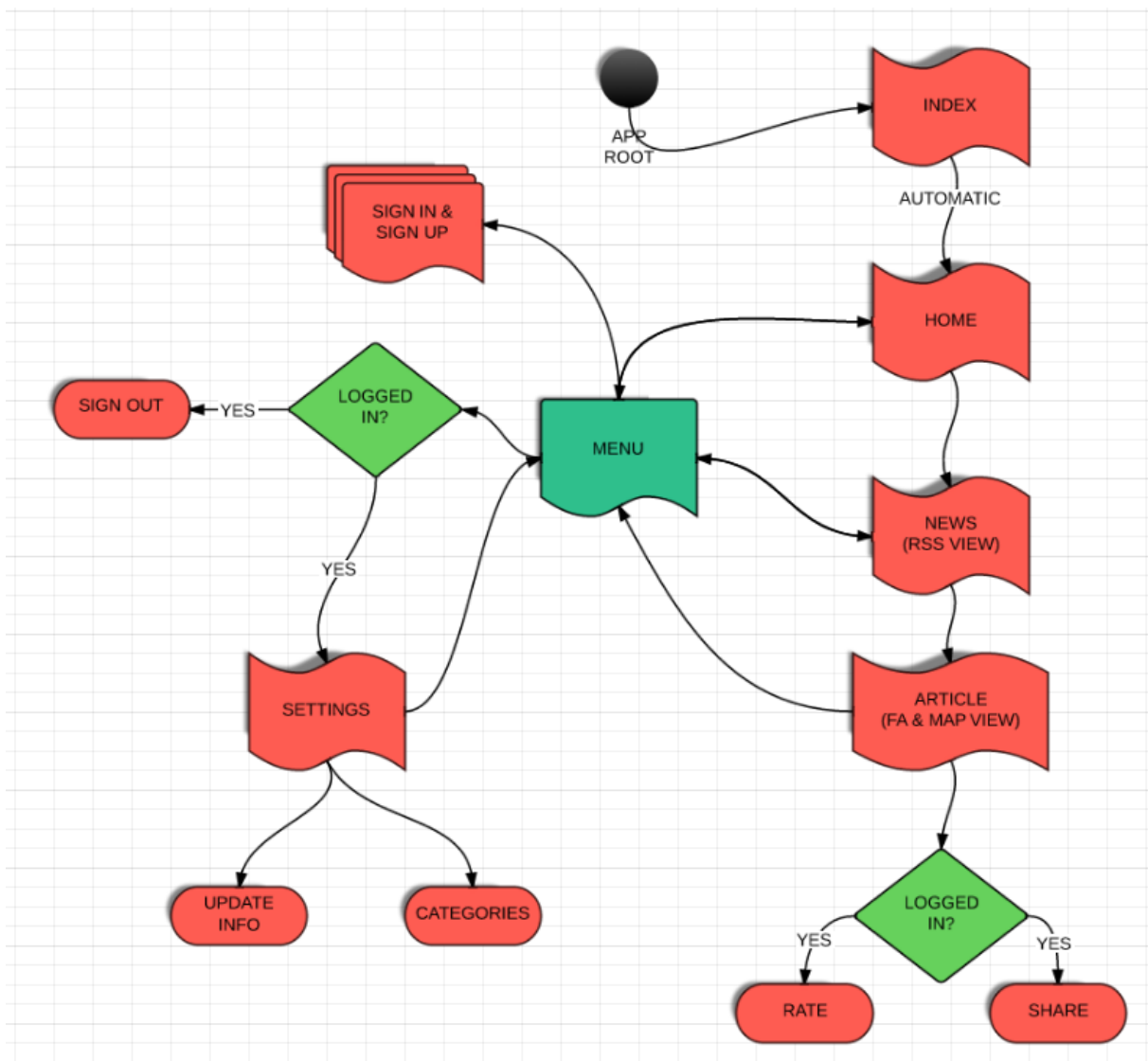


Figure 5.2: Site map of the POC prototype

The navigation menu is placed on the left side of all views except the Index view. The actual menu is hidden until the menu button is pressed. It forces the menu to slide from left to right

and become visible to the user. From the menu, users can sign in/up, log out, accesses the Settings view and related links.

These links/buttons that appear after the menu-button has been pressed, vary based on whether the user is logged in or not. When the user is not logged in (which is the case the first time the user visits the site), the sign in/up link is shown at the top followed by the latest news, home, and other links concerning the project. But if the user has already created an account and logged in, then it will be displayed two links instead of the sign in/up link. The two new links that appear are settings and sign out link. Just as the name implies, the settings link can be used to change or update user info, and the sign out link to log out of the application.

5.3.2 POC test scenario

The prototype was deployed to the server on NTNU for testing and data generation after the completion. Due to its simplicity, two scenarios were chosen for the test-users to test the GUI and interactions with the application. Main focus was on the navigation menu, news, article and settings view.

For the scenarios, the users were asked to follow and perform the scenarios that are shown below.

- Scenario 1
 - Step 1: Click on the menu button
 - Step 2: Click on the Sign In/Up link and create an account
 - Step 3: Go to the News page by either pressing on the Start reading button on the main page or the Latest news link on the menu
 - Step 4: Select an article to read
 - Step 5: Rate the selected article
 - Step 6: Log out of the application by clicking on the Sign out link on the menu

- Scenario 2
 - Step 1: Click on the menu button
 - Step 2: Click on the Sign In/Up link and log in
 - Step 3: Navigate to the Settings view by clicking on the Settings link

- Step 4: Add/edit your information and press the Update button afterwards
- Step 5: Navigate to the News view by clicking on the Latest news link
- Step 6: Press the dropdown menu called Manage and choose between "Get more articles" and "Update"
- Step 7: See if you get more articles or updated/recent articles

The two scenarios above were tested on three devices, a personal computer, Samsung Galaxy S3 and iPhone.

5.3.3 POC test feedback

The prototype was analyzed according to the functionalities of each view. We found the following positive and negative aspects of the prototype:

- Index
 - + Creates a faster download phase for the application.
 - Should start cross-domain request at this moment to reduce download time in later lifecycle?
- Home
 - + Good starting point for the application, feeds the user with information and useful links.
- News
 - + The user has the opportunity to see several articles simultaneously.
 - Requests the API for news articles each time rendered.
 - Adding more news articles to the view forces a new cross-domain request.
- Article
 - Scrolling over Google map container, forces scroll on the actual map and not on the view.
 - Users are forced to click back to the News view for selecting a new article. Repeating this step was annoying for the users after a period of time.
 - Control panel content is visible to the users at all time, even when the users are not interacting with it.
- Profile
 - Too broad layout.

- Adds an extra switch between views.
- Navigation
 - + Necessary links are available to the user.
 - Sometimes confusing the user with too many links.

Overall feedbacks were positive, but it seemed annoying to switch between the News and Article view for the majority of the users when selecting new articles to read. There were also complaining about the format of the article stories, which were caused by the back-end API. We got mixed feedback on the font-size, style and colors. Users' opinion regarding the layout was totally different from each other.

Users were not aware of the application's ability to store their geo-location; this created some skepticism. They tend to have the ability to turn geo-location on and off at any time. Clearly the application needed a performance lift-off. Most of the users expected to scroll down at the end of the page to see more stories, doing this task via a button seemed confusing. After using the POC for a while, users lost counts of which articles that were read or not.

5.3.4 Additional requirements

After conducting feedbacks from first test-users of the POC and own analyses of the application, previously added requirements were customized and additional requirements were added.

One of the directions that were given high priority was to give the users the ability to customize the application as much as possible.

Functional requirements

FR2.1: The application must start requesting news articles from the back-end server before entering the News view.

- FR2.1.1: Downloaded articles must be stored locally on the client for later use (possibility for offline reading, reducing cross-domain requests).

FR2.2: The application must be able to update news articles automatically after an interval of time.

FR2.3: The application must automatically add additional article stories to the view, without interaction by the user (FR2.2 and FR2.3 eliminates management button and reduce confusion).

FR2.4: The user must be able to search for articles.

FR2.5: The application must keep track of which articles that have been read on a particular client.

- FR2.5.1: The user must be able to see the rating of a rated article, when the article is selected again.
- FR2.5.2: The user must be able to change a rated article's rating star, even the article has been rated previously.

FR2.6: The users must be able to change or switch:

- FR2.6.1: The way the news articles are displayed.
- FR2.6.2: Font-size of the article story.
- FR2.6.3: Background and font color for article stories.
- FR2.6.4: Color of the navigation menu.

The customizable preferences mentioned above must be saved both on the client and the server.

FR2.7: The user must be able to turn geo-location on or off.

FR2.8: The user must be able to share the whole application automatically on social media (see FR10).

6 Architecture and implementation

This chapter provides a basic explanation of the features of the final application, along with description of its architecture, implementation and design process.

6.1 Architecture

It has not only been important to choose several proper frameworks and libraries for the implementation of the Smart Media application, but it has also been a high priority and emphasis on how to use these frameworks in a proper manner both separately and together.

For certain, some of them have "best practice" way of coding and performing styles. These were taken into account when developing the Smart Media application. In addition there were some main strategies and focuses that were carried on through the whole implementation phase.

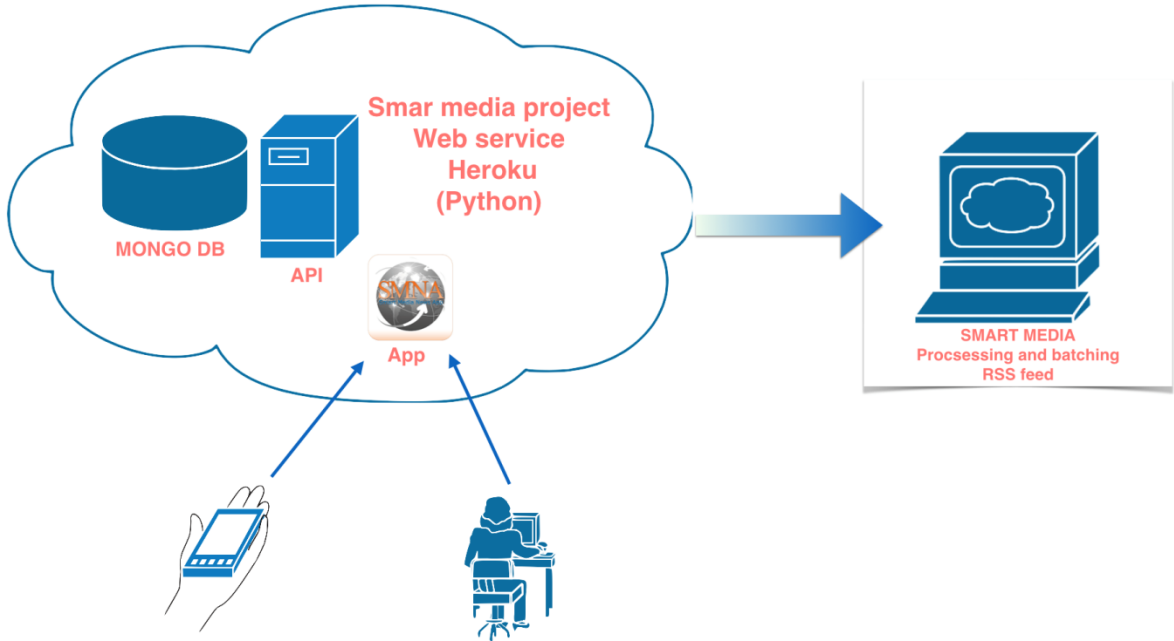


Figure 6.1: The connection between clients and the server

6.1.1 Strategy

The main strategy was to split the application into separate layers and parts, where each part has specific concerns, roles and duties to fulfil. This gives the application the advantage of separations of duties, reduces coupling between layers and parts, and to keep the code cleaner.

Each module was developed as separate code units without relying on other parts of the system. The application was built in such a way that the main source code could be reused to create the hybrid application with minimal customizations.

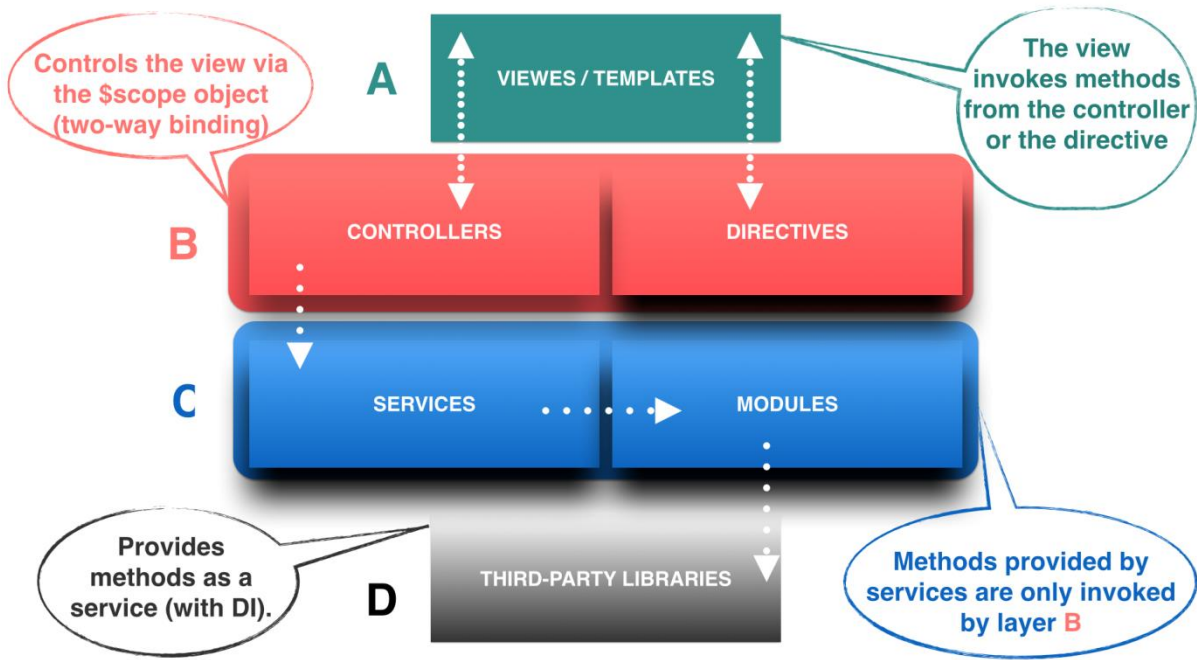


Figure 6.2: Overview of the layers and their roles

6.1.2 Pattern

There are used several patterns in the Smart Media application; some of them were imposed by the chosen framework, some preferred and some strategically chosen.

Angular is already featuring MVC/MV* pattern, but the commonly approach to develop Angular applications is to use the MVC pattern. This design pattern is supposed to make the front-end and testing development easier. It also separates responsibilities in between entities, modules and enforcing isolation, which was one of the Smart Media strategy goals.

The Model is the main object and domain-specific. The View/Template could render the Model in one or several different views and the Controller controls the Model and changes the View. Simply described, the pattern separates the Model, View and the logic between them through the Controller.

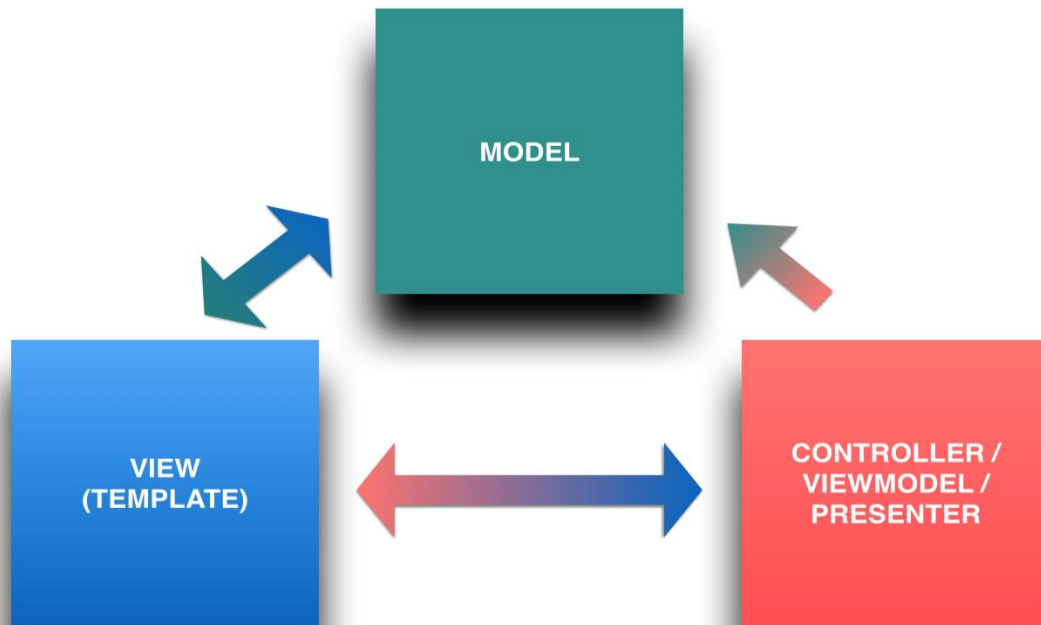


Figure 6.3: MV* pattern

From a high level, the Smart Media application is built with MVC, Singleton and Revealing prototype pattern. All other modules are built separate as plugins, which can be added or removed from the application.

6.1.3 The way of Angular

Angular is modular based, which means developers can accurately use the framework as is or they can extract some of the properties/modules they need to achieve for a certain functionality. These modules are injected to the application at run time. When the application increases in code size, this behavior is appreciated. Unlike other frameworks or libraries where developers are forced to include the whole framework/library before they can use a specific functionality. This may overload the system with unnecessary unused code and decrease the performance of the application. This is not the case when using Angular. All of Angular's modules are prefixed with an “ng” tag to indicate their affiliation to the Angular framework. Since Angular is an open-source project, there are a large amount of third party modules built for Angular available.

The Angular modules that were used in the Smart Media application are:

- *ngAnimate*
 - For complex animation and document object model manipulation.
- *ngCookies*
 - For adding, editing and removing Cookies on the client-side.
- *ngRoute*
 - For routing and linking controllers to views when developing Single Page applications.
- *ngSanitize*
 - For parsing and serializing html.
- *ngTouch*
 - For touchscreen devices.

Beyond ng-modules, Angular provides a set of components such as Services, Directives, Filters, and Controllers. These components are created as singletons, and invokes only once at run time. Some of these components are built-in components and originates with Angular framework. All these components can be used together to organize and share objects, methods etc. across the whole application. The Smart Media application is built upon both own-built and built-in components such as Directives, Services, Controllers and Filters.

Directive:

Directives can be used as markers on or attached to the document object model. Directives represent a small or larger piece of code, which transforms or specifies a certain behavior on the DOM element. There are more than approximately 80 own customized built and 30 built-in Angular Directives used in Smart Media application. Directive's primary task is DOM manipulation.

Service:

Services are standalone objects that can be wired with Directives, Controllers and Filter by using dependency injection (DI) at run time. DI means that Angular provides the Service to other components as requested. By manually reading or using automated test, developers can determine which dependencies are used inside a Controller or Directive.

Creating own-built Services are an easy task. There are several different types of Services, but all of them must be registered inside the application. Angular Service's primary task is to

represent a set of methods and objects to the rest of the application, so that they can be shared or used. There are approximately 20 Services running on the Smart Media application.

Controller:

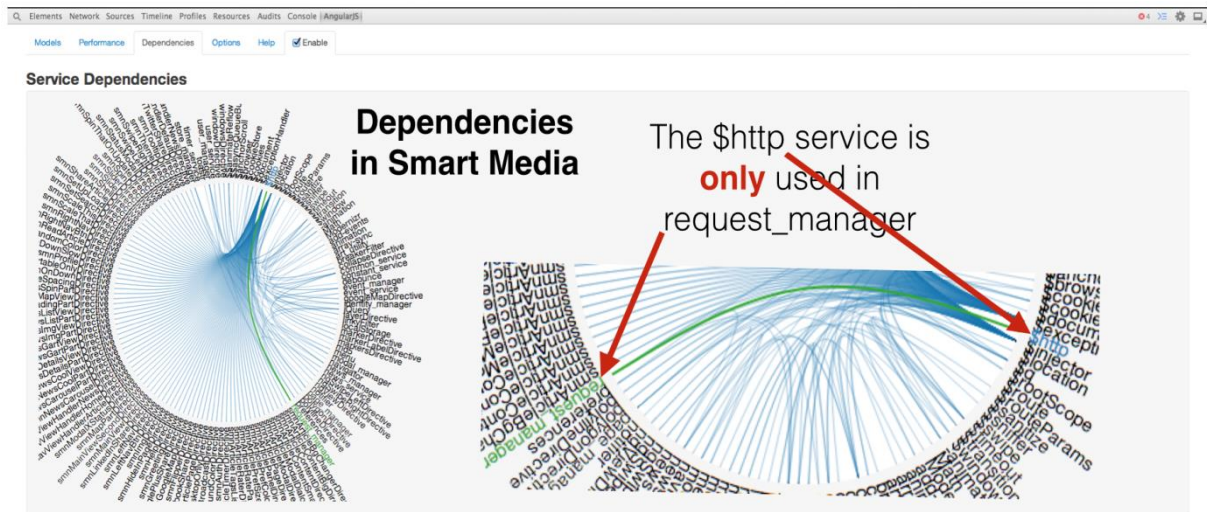
The Controller is the glue between the View/Template and the models inside that View. A Controller initializes the state of the models and adds behavior to it. Controllers are mostly used for data binding or to complete business logic. Other operations are not recommended. For instance, when a Controller performs a task, the actual code for performing the task is located somewhere inside a Service and the Controller only invokes that method. It is not recommended to use Controllers to format or filter inputs/output, or to manipulate the DOM elements.

Scope

The Scope is the actual glue between the Controller and the View. The Scope object is the most common used object in any Angular made application. There is only one root scope, but many child scopes. Scope is injected to every Controller or Directive, and refers to the application's model via one- or two-way data binding. The scope objects are arranged in a hierarchical structure based on the DOM structure of the application. Some scopes are inherited from, shared between or isolated from the rest of the application. Scope arranges connection between the controllers and directives.

Broadcast and Watch

With Broadcast and Watch, the application can listen on events fired by other parts of the application or property-changes inside controllers or directives. The Watch expression can notify the Controller or the Directive about changes in their properties. In addition, one can register an event-listener and method to be executed whenever the broadcast events occur. With these expressions, the application can render the View without reloading the whole page, or complete asynchronous tasks behind the scenes.



Batarang, plugin for chrome

Figure 6.4: Screenshot of Service dependencies

6.1.4 Application structure

The Smart media application is structured into three larger parts. From bottom up, services are running in the background, and then the directives (which manipulate the DOM elements) are followed by controllers and views that are visible to the end-users. For instance, there is only one service for cross-domain requests. This service is registered by the name "request manager" and is the end-point of the application (see Figure 6.4). This service is only used when the application is about to perform a request to the API. Code for error handling, asynchronous task management and cross-domain request conventions are defined in this service. This enforces separation of duties, and makes testing easier.

Other services, which perform a cross-domain request, must go through this service. The remaining services are structured in exactly the same way and use the same principles. In general services prefixed with "_manager" are the most underlying components and each domain specific models has an associated manager service that handles all logic about the Model.

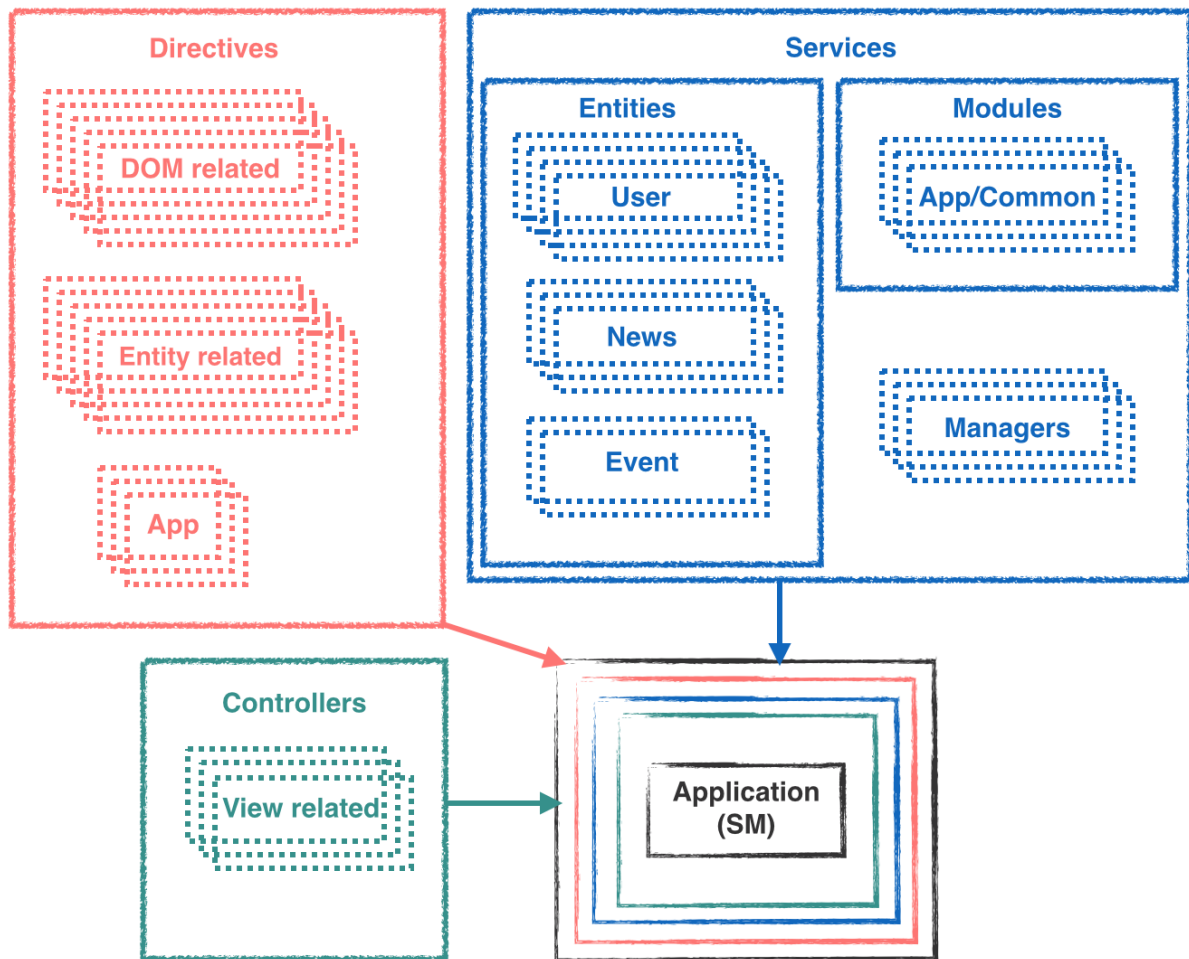


Figure 6.5: Rough sketch of the application's architecture

6.1.5 API

The Smart media application is relying on two Web services, with representative APIs. A custom made Web service has been developed only for the Smart Media application by the collaborators at IDI. This service is deployed on Heroku and can be accessed via HTTP conventions. The API is very small and provides the application with the most needed and necessary features only. The API is built in such way that only applications served on the same domain can make use of its available methods. The application is therefore hosted at the same server; this will also decrease the API request compilation time and increase performance.

The second service is the Smart media project services used for indexing and fetching news objects only. This service is the news-service provider for the application. The news objects

are first requested by the web service at the Heroku, then filtered and customized before forwarded to the Smart Media application for use.

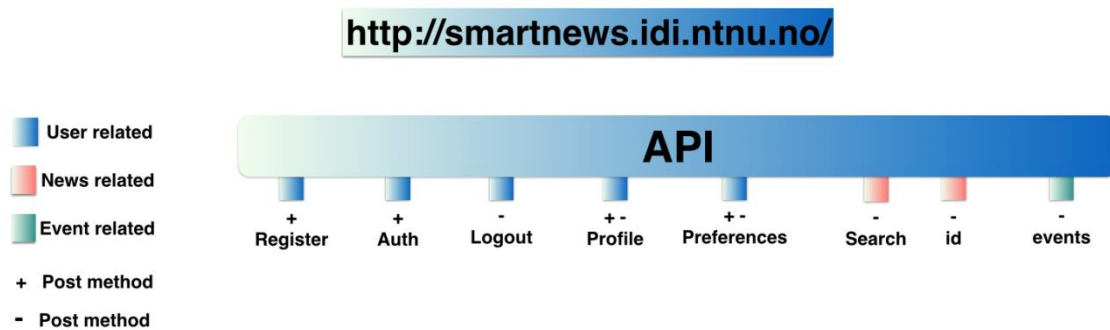


Figure 6.6: Image showing the API methods

6.2 Implementation

This section describes the implementation of the application. Some of the most important parts of the system will be explained in details.

6.2.1 Web storage

One of the greatest advantages of using HTML5 is the Web/Local Storage feature. Unlike SQL databases and without any complex operations or transactions, objects can easily be stored within the client's web browser until they are deleted. These objects are stored based on key and value pairs. The right key gives accesses to the associated value. Any object can be stored in the Local Storage, but the actual value is stored as String (data type).

Store manager service is the only service accessing the HTML5 Local Storage; this service handles the transformation when adding or retrieving data for the storage. Other services do not need to worry about the type of the retrieved or stored objects. Furthermore, all objects are stored as JavaScript Object Notations (JSON). This gives the user the ability of reading articles offline. Additionally, the application can keep track of user preferences; read articles and a lot more. If the user deletes the data manually via developer's tools, the application will reset the data state automatically.

6.2.2 Google Map

The geographical location (geo-location) is visualized in two different ways in the SM application. The original idea was to display the location of the user and the location of where an article story had taken place. To do so, the application needed latitude and longitude properties for both the user and the article. But this had to be changed along the way, as the API did not provide sufficient information or additional details about the article properties.

Instead of geo-location properties for occurrence place, the API could provide the application geo-location properties for entities mentioned in the article. There is no guarantee for the entities belonged to an article to have geo-location properties. Therefore articles are being filtered on the client-side, and only articles with representative entities that contain such properties can be shown as markers on the map.

To be able to access the Google Map's API, one must first create an online-generated Google Map API key, which must be referred to from the application at run time. The process of collection of this key is very easy and simple. Google presents high quality documentation for achieving this.

6.2.3 Social networks

An important requirement was to be able to share articles via social networks. This could be achieved in two ways. The first way was to develop separate plugins for each of the desired social network by downloading each of their provided third-party libraries/plugin and customize them for the Smart Media project. To use any of them, the application had to be registered on their app registry and would be assigned a unique app id before it could make use of their API. The second way was to use only one plugin for all of them. Obviously the second option was preferable, and therefore was chosen. Different modules/directives are adopted for Facebook, Twitter, LinkedIn and Google plus in the application.

6.2.4 JSON

Every request by the application to the server requires sending or receiving of data. Any data sent or received by the application have the JSON (JavaScript Object Notation) format. These objects are either a single object or an array of objects. This makes the manipulation of these

objects easier. Among other objects, the news objects are filtered on the client, and any unused properties are deleted before forwarding these values to the rest of the system.

6.2.5 Libraries

The application's complexity increases as third-party libraries are added on, especially because of the way JavaScript handles the source code. Lots of bugs and errors can be hidden inside several of libraries. Therefore libraries were added on with care and caution to the Smart Media project, using Angular's DI system. Creating modules inside JavaScript code-scopes contains the properties outside of the global JavaScript scope, which reduces errors, failures and possibility of property conflict between third-party libraries.

The most popular third-party JavaScript library used in any modern web application is jQuery, which is written in pure JavaScript. It delegates the intermediaries of completing various tasks such as DOM manipulation, finding DOM elements or cross-site requests.

jQuery is definitely the largest third-party library used in the Smart Media application. Frameworks such as Bootstrap, which depends on jQuery, are also used in the application. Some other small jQuery-dependent plugins are also used for animation and notification.

6.2.6 Events

User events are a large part of the Smart Media application's focus. It concerns both the application and the whole Smart Media project team at IDI. The events are created each time the users select a news article and they are passed between different controllers and directives, and are modified before they are saved on the users' browser. To minimize application request to the server, the application saves up to twenty event objects before it tries to store them on the web service's Mongo database. If this is successful, the local storage is rewriting. If not, the system waits until it has Internet access before the application repeats this process again until event objects are saved on the server. This means that offline reading and event-registering process can be achieved simultaneously without having internet access.

6.2.7 Profile

A new user profile object is requested each time user clicks on the Profile link. If there are none-stored values on the server, the user is met with empty fields. Else, the downloaded properties are replaced on the input fields. The user can change/overwrite these values and save them on the server again. Both this technique and the way the UI is built make it difficult to overwrite unsaved profile properties by the user.

6.2.8 Geographical location

Unlike web applications, mobile applications give users ability to turn on and off their geo-location in a simple way. There are several techniques and libraries that can be used to collect the users' geo-location on a web application, but this can add more code layers than necessary. Usually this is achieved by pure JavaScript code, using the Navigator object of the browser. The users are met with a confirmation box (which differs from browser to browser) on the top of the browser. By either to grant or refuse permission, the outcome of this is saved as setting property on the users' browser (only for current domain). Next time the user launches the application, the same stored value for this property is used to complete the necessary tasks. Changing this value is not an easy task, at least for regular pc users. The user must go inside the browsers settings tab, locate this property and change it. This was not really practical for Smart Media project since some of the views depended on the users' geo-location information. Because of this, there were decided to add an extra property to the user-preference object. With this property, the application can decide to either locate the user or not proceed with this action.

For the Smart Media Web application, the users' geo-location value is based on the IP address. The Hybrid application uses a new Navigator object to access the device's geo-location to locate the user.

6.2.9 Application flow

The application is dependent on HTML5's Web Storage. Both news objects and user preferences are stored inside this storage. For the news objects there are three possibilities:

- The local storage is empty
 - Forces a request to the server, storage up to four hundred articles. If the request is successful, the application saves the collected data and a timestamp for the occurrence of the request to the server.
- The local storage is not empty, but it has been six hours since last news storage update
 - Forces a news request, updates the data and the timestamp.
- The local storage is either empty and it is less than six hours since the last news storage update
 - Falls back to normal state, and does not execute any requests

These three possibilities can occur before the user visits the News view; this will give the application enough time to load the content before it is needed. This increases the performance of the application. When the user visits the News view, news articles have already been downloaded or updated by the application and the user does not need to wait until the application request the server for new articles. The content is downloaded faster via local storage. The last opportunity for the news objects will only occur when the user is located inside the News view and the application have already downloaded the news objects either from the storage or the server. If this is the case, then the application is using an internal service to encapsulate the news objects for reuse. The service can increase in size and keep track with added articles as long as the application's state stays normal and the user is not forcing an entirely reload of the application (manually reloading the browser). A reload will of course reset the application's state and the service. Since the application is built as a single page application, this will not occur when the user navigates between different views. In addition, if the user navigates to the News view before the application has received the news articles from the server, the application will not force a second request. The application waits until the request is completed before it requests the storage and not the server for downloading news articles.

For the user-preferences there are three possible outcomes:

- The local storage is empty
 - At first-time launch, the application creates default values for these properties and saves them inside the user's browser.
- The local storage is not empty

- Loads the preferences from the browser's storage and uses them to set the UI or other preferences.
- Change of the user status
 - Occurs only when a user signs in. In this case, user-preferences (unique to each user) are requested from the server. If these properties exist, they will overwrite the existing properties stored in the local storage. The application reloads the view and rewrites the preferences based on new information.

Directives watch the preference object and listen for events to change the UI or DOM manipulation. After completing one or several of possibilities mentioned above, the application falls back to standby state. It listens to events, watches properties and reacts to various user interactions.

There are some small changes between the application running on a portable device (such as smartphones and tablets) and desktop computers. There is a service specific for this task, which sets the amount of requested news object from the server (Local Storage for mobile devices have limited capacity). This service is also making sure of which images are used; smaller images are preferred for smaller screen size. There are also directives, which remove or add DOM elements based on the user's device.

6.2.10 Development

The application was entirely rewritten from scratch with new architecture and patterns. The application followed an iterative and incremental development methodology. The process was divided into several sprints, where functionality was tested and evaluated before being added to the application.

The final application contains 3917 lines with own-built customized JavaScript code, 2215 CSS3 code, and 1010 lines with HTML5. Both the CSS and the JavaScript code are minified to give the application faster file-download time. All other libraries or plugins (CSS and JavaScript) are also minified and merged to the application code.

WebStorm and NetBeans IDE were used for the development of both the web application and the hybrid application.

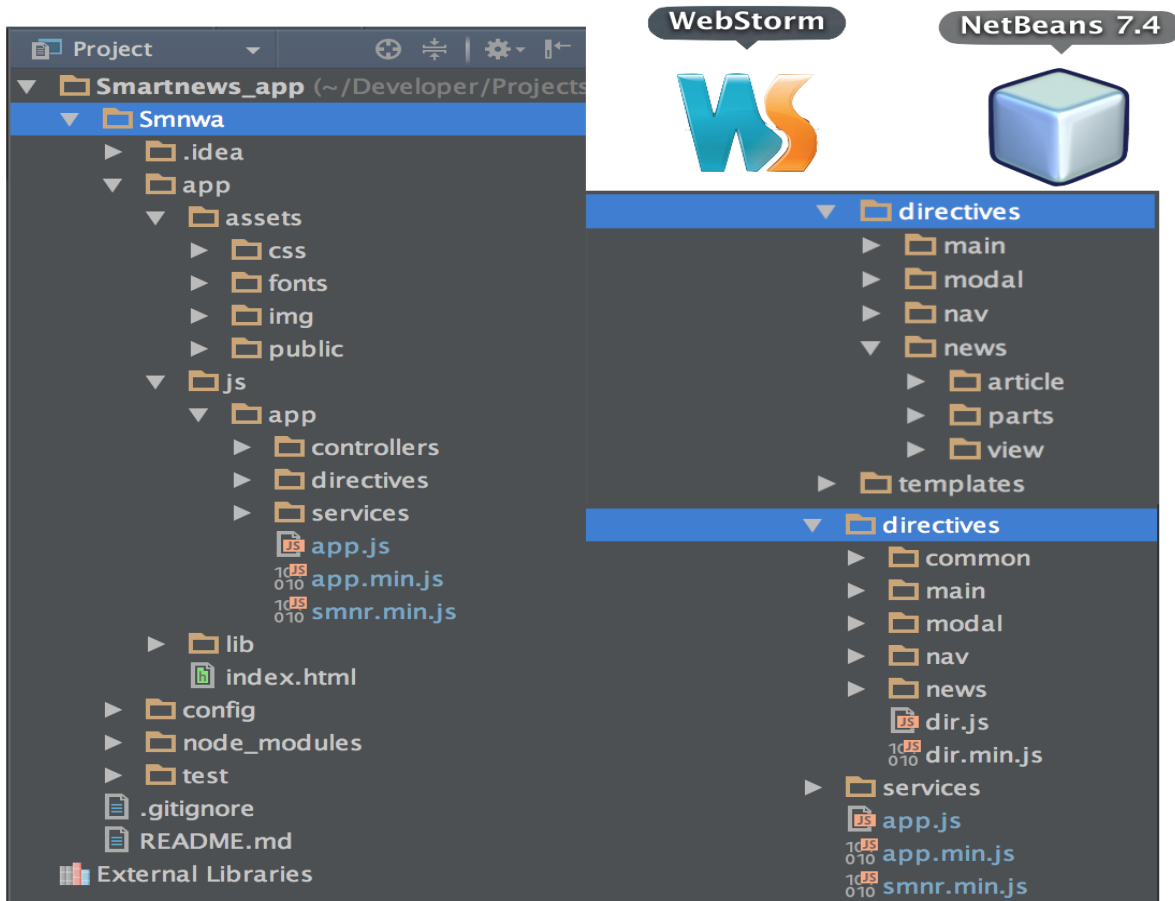


Figure 6.7: Screenshot of the application's code hierarchy (shown in WebStorm)

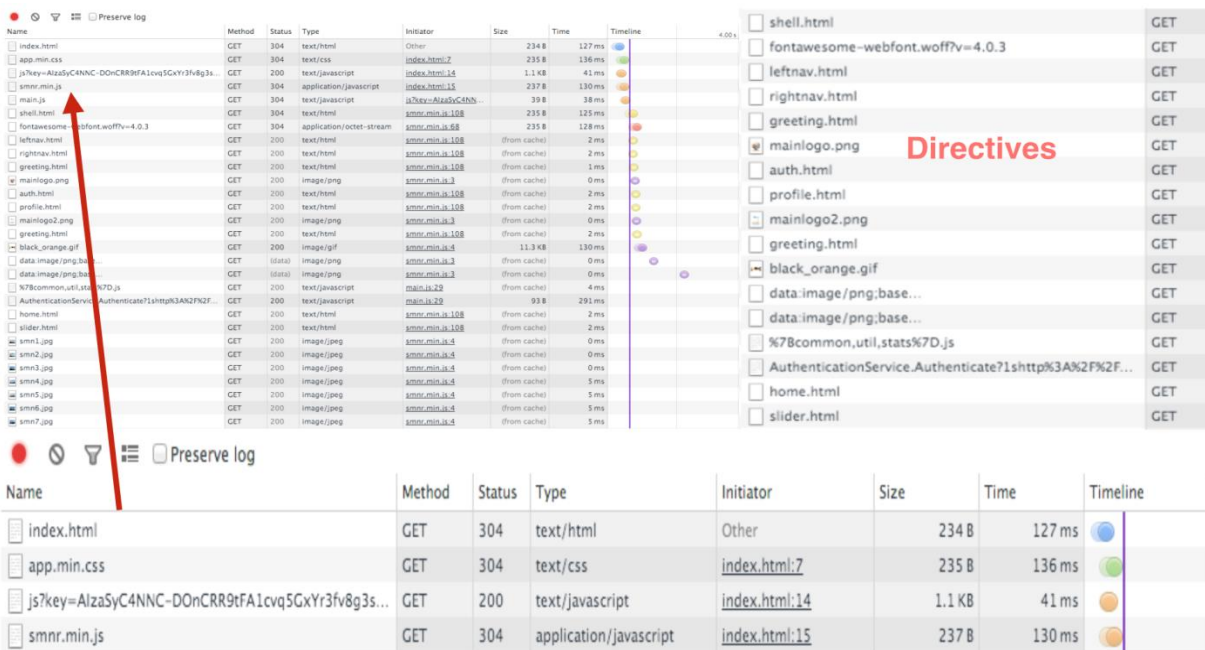


Figure 6.8: Screenshot of the application files during run time displayed in the browser's development tool

6.2.11 Hybrid application

The hybrid application is a supplement app to the Smart Media project. The original plan was that this application could complete the Smart Media Web application and deliver the same functionalities. Unfortunately because of the API and the way it works, the Hybrid application was not able to offer all the functionalities as the Web application. Since they both drive from the same source code, removing only unwanted modules were enough to disable not working functionalities. Modules removed from the hybrid application relative to the web application are:

- The event saving module
- The user-profile module
- The user preferences module

Users are allowed to create new user account and sign in to the application, download news articles (for offline reading), change between views and so on.

The hybrid application was developed using the PhoneGap API and NetBeans. There are a lot of configurations that must take place before the application can be transferred to an actual device. Platform such as iOS expect a cost fee before this transfer can be completed, but there are all types of different emulators that can be used for testing and running of the application.

As the configuration is quite strict and all of the folder name-prefixes must correspond to the constant of what the system expects, the setup can be quite difficult to achieve. Source code should be at a folder named "www". Native plugins for each platform will be located in the folder named "platforms". These plugins can be downloaded separately using the command line tool, and it causes that unnecessary code is not added to the project. The setup is automatically done by NetBeans (own Cordova apache plugin). There are both third-party plugins and PhoneGap developed plugins that can be used by developers. The Smart Media hybrid application uses the device's geo-location, which is located in the Geo-location plugin delivered by PhoneGap.

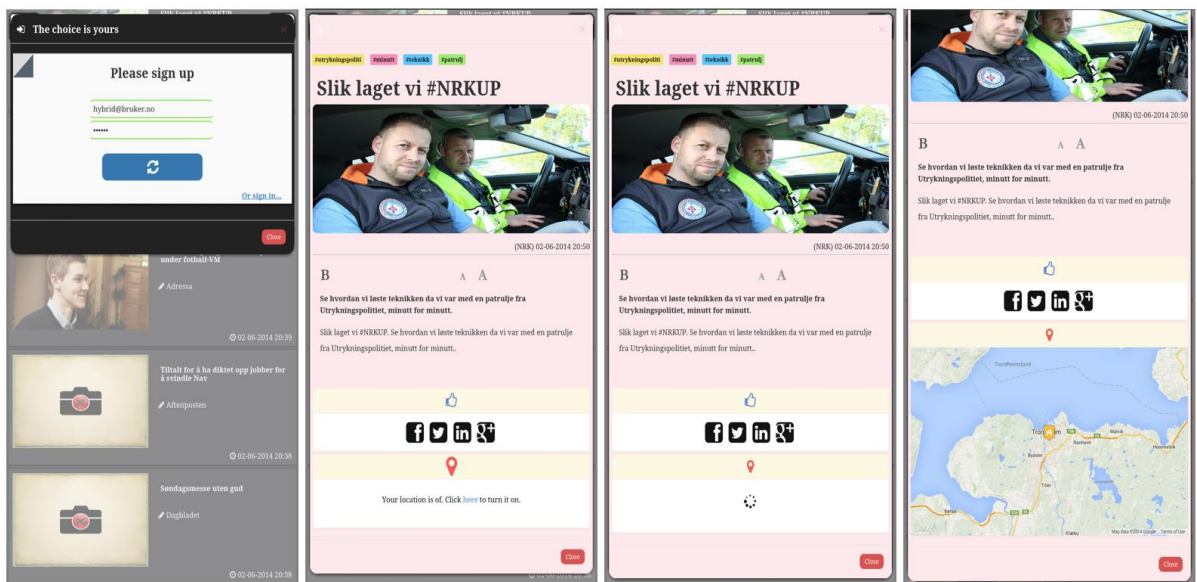
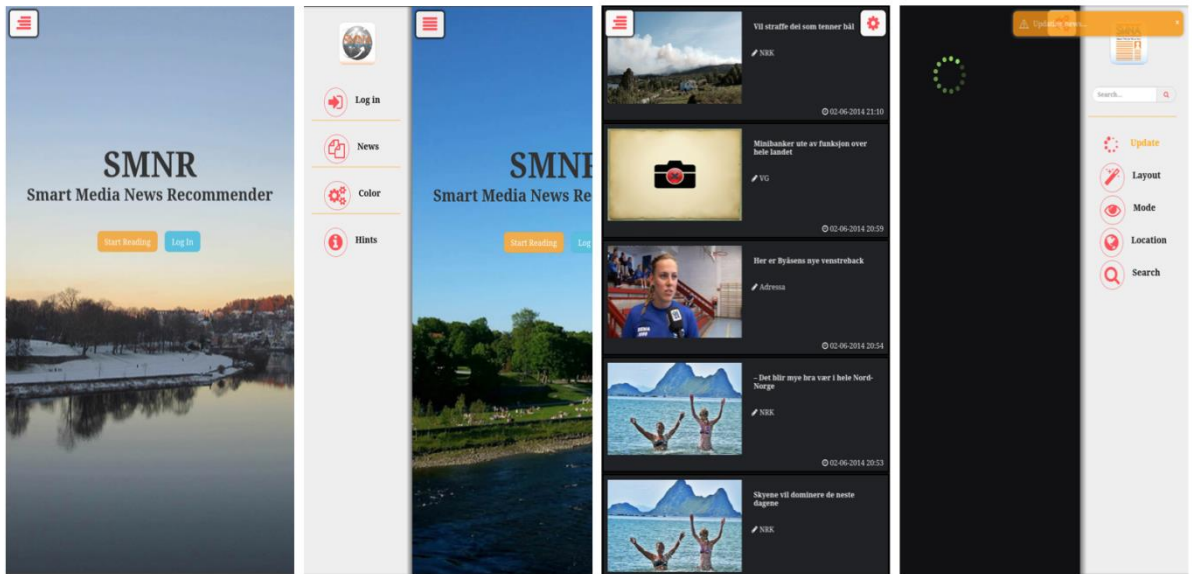


Figure 6.9: Screenshots of the Hybrid application running on a smartphone

6.3 Application Graphical User interface

This section describes the user interface of the application. First, it will be given an introduction of the final product, then each view of the application will be explained with text and corresponding images.

6.3.1 Application description

The final application is developed based on a set of rules and design principles. As a result of experience from development of the POC prototype, the final application reduced the amount of different views. The application contains the following four views: index, home, news and article.

The way that the different views are being changed between each other is that usually the page becomes blank before it is replaced with another view. Existing views are being moved before a new view is taken place from right to left. The animation creates a slide-out and slide-in effect when users switch between views (see Figure 6.10).

Elements, such as buttons appear on the page with animation. This gives them a fade-in effect. There are also a set of other CSS effects like hovering, scaling and transforming on different elements. The UI is compact, and all noisy elements have been removed from the DOM. This gives users ability to concentrate on what is important to their perspective.

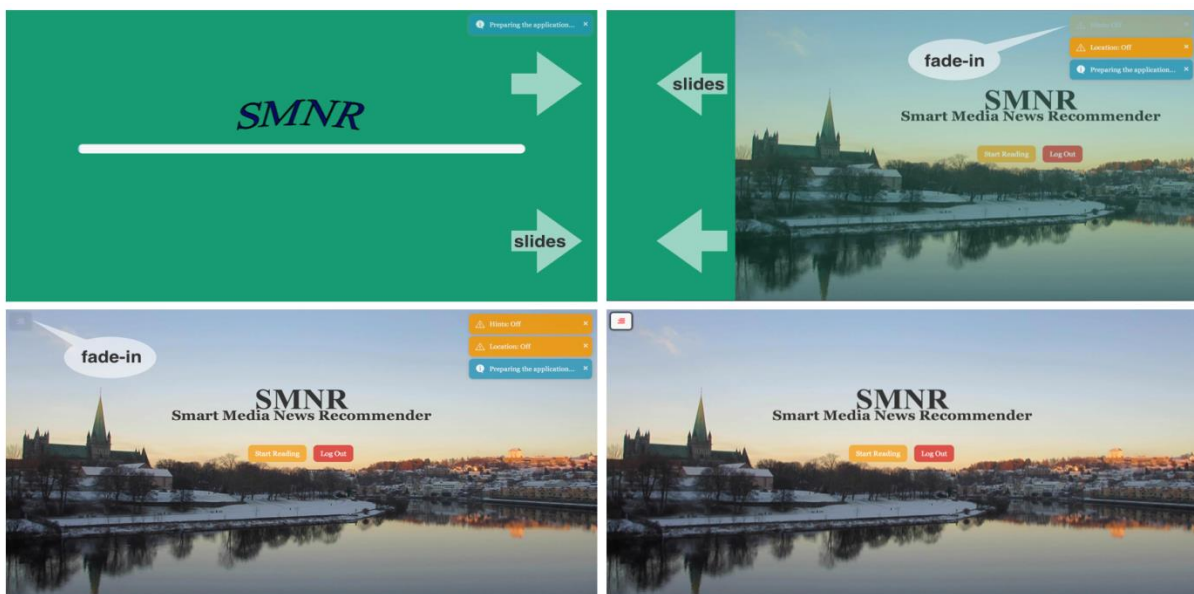


Figure 6.10: Screenshot of the application's Home page

The application has a notification feature which gives users feedback based on application state and their interaction with the application. The notifications appear as a pop-up window at the top-right of the page. By using CSS, the notification window will always appear above all other layouts. There are four different notification appearances with corresponding background color and icons. Blue color indicates notification for information, green indicates success, warning notification has orange color, and error notification has red color.

The notification module is handled by the system and will notify the user what it supposed to do in case of error or failure. The application also offers a user-controlled notification feature called "Hints". This can be turned on or off by the user. When "Hints" is on, the user will get notifications by clicking on different DOM elements or input fields. The notifications will appear only once with information about the element or what the user have to/should/could do. Any type of notification will appear for four seconds on the screen. During this period, the user are able to remove this pop-up box by clicking on the close button on the top right of the box.

The UI is also touch-optimized and response better to touch events than a regular UI.

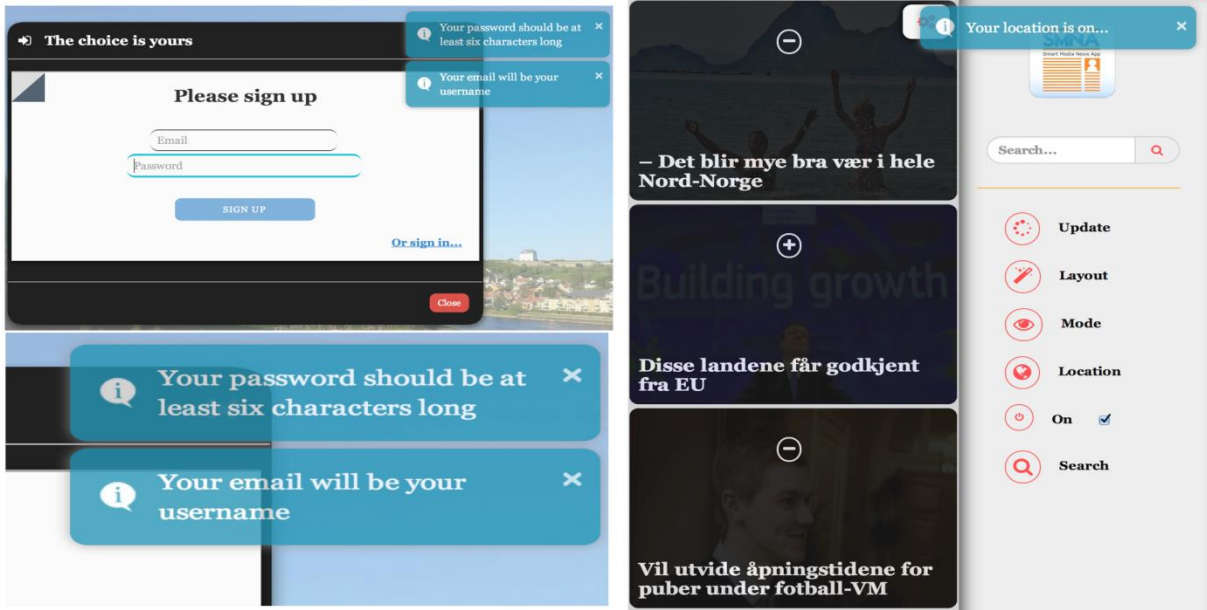


Figure 6.11: Screenshot showing the notification feature

As mentioned earlier, the final application contains four different views. Below, it will be given a brief description of the various views.

Index

The index view has not been changed since the POC prototype. The system sets a random color for every reload (every time the application launches). This view is the application's first view when visited, and the user is forwarded to the Home view after end loading. The system checks the user-preference object, notifying the user which property is on or off. In this way, the user is aware of which preferences are on or off (such as location). Furthermore, if the user accidentally tries to redirect to an invalid URL on the same domain, the system will always redirect the user to this view.

Home

This is the application's welcome view. For the first time the user is presented to the menu button (fades in at the top-left of the view). There are two buttons placed in the center of this view. The first one is a link to the News view (called Start reading) and the second one (called Log in) is a link to the authentication drop-down window (Modal window). The second button is also a dynamic button, which means that it is changed based the user's status and executes different actions. For example, if the user is logged in, the button becomes red with the text "Log out". Beneath these buttons is some additional information about the project, followed by a Google Map marking the IDI's place in Trondheim.

Users can easily use one of the four underlying buttons to share the application on preferred social media. The aim of developing this view was to provide the users important and necessary information. Therefore the buttons have a strong color and are centered to make them stand out in order to find them easy. Simultaneously this is the first view where the user can navigate through. Therefore it was important to make a decent impression of the application.

News

The second menu button appears (fades in at the top-right) to the user in this view. Users are presented with loading-animation (getting article from the server/local storage) before the downloaded news article take place on the view. The rich site summary (RSS) perspective is the only perspective for presenting articles in this view. But there are seven different types of RSS perspectives that users can switch between. This can be done by using the right navigation menu. Switching between types will change the entire representation of the view.

At first, there are displayed only 50 news articles in this view. But by scrolling down to the end of the page, the system will automatically notify the controller to load more articles. Then the controller loads 50 more articles and adds them to the page. Before the articles are downloaded, there are shown a rotating loading-animation indicating that articles are being retrieved. Since the articles are sorted by date (newest at front), these articles will appear at bottom of the page/view (see Figure 6.12).

If the local storage is empty (the user have requested over the limit), the application will request more articles from the server as this is needed. The user can shuffle through these articles and select one for reading by clicking on the RSS container/article box. Only signed-in users have access to the full news story. If the user is offline, a status modal window will appear on the view notifying the user about his/hers current status. In order to be able to read the selected article, the user will be asked to sign in. The user can refuse to sign in/up by clicking on the cancel button. This will cause the application to fall back to standby state. But if the user choses to sign in/up, the status window will disappear and the authentication window will appear.

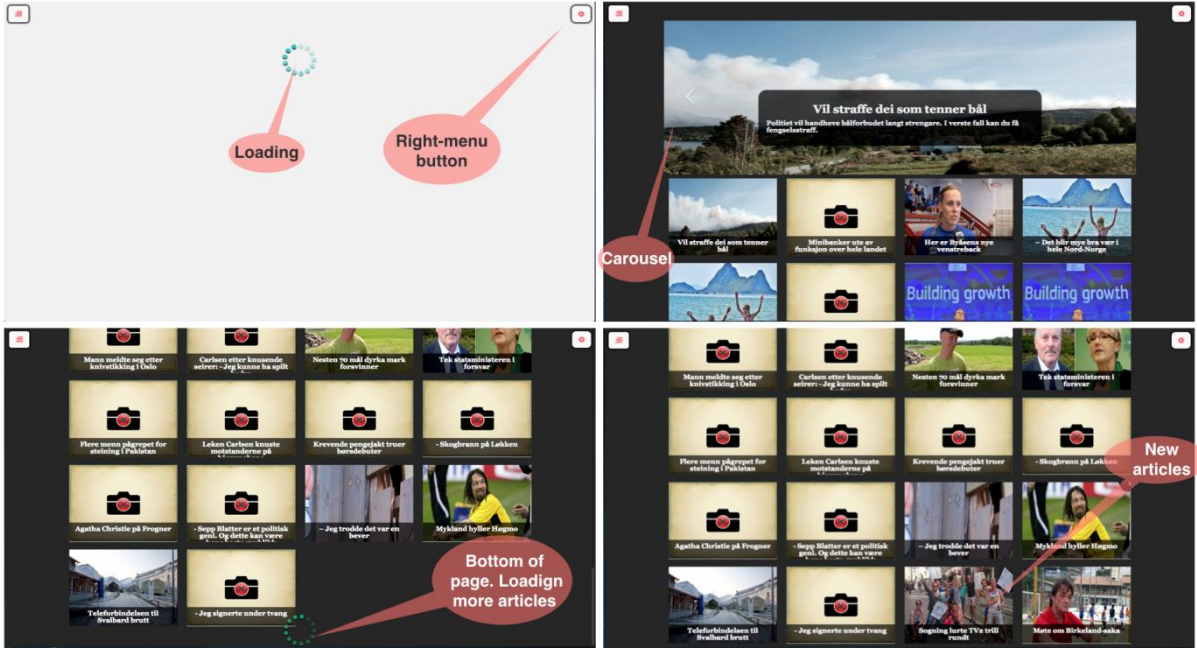


Figure 6.12: Screenshot showing the loading of news articles

The user must log in to the system before it responds to the user clicks. Since the application proceeds with the same selected article, the user does not need to select the article again.

There are three different ways to display the selected article (called "Mode" in the application). The first one is to display the article on a Modal window without freezing the UI or changing the application's state or view. In this case the user stays on the same view. The second way is to change the current view (News) with the Article view, which forces a redirect. The last option opens a new tab on the user's browser. Like the first option, the News view stays untouched.

There are a few actions that the user can achieve by using the right-side navigation menu. The user can for instance turn the geo-location on or off, force an update (reload the article storage), change the layout, and search for articles.

The search feature goes through every downloaded article in the current view when the user types in the search field. The search functionality can be accessed in two ways, either by directly typing in the search field or by typing on the search-modal window input field as it appears. The list of articles will be filtered on the fly as the user types in different characters into the input field. The articles that do not contain the word typed by the user will be removed from the list.

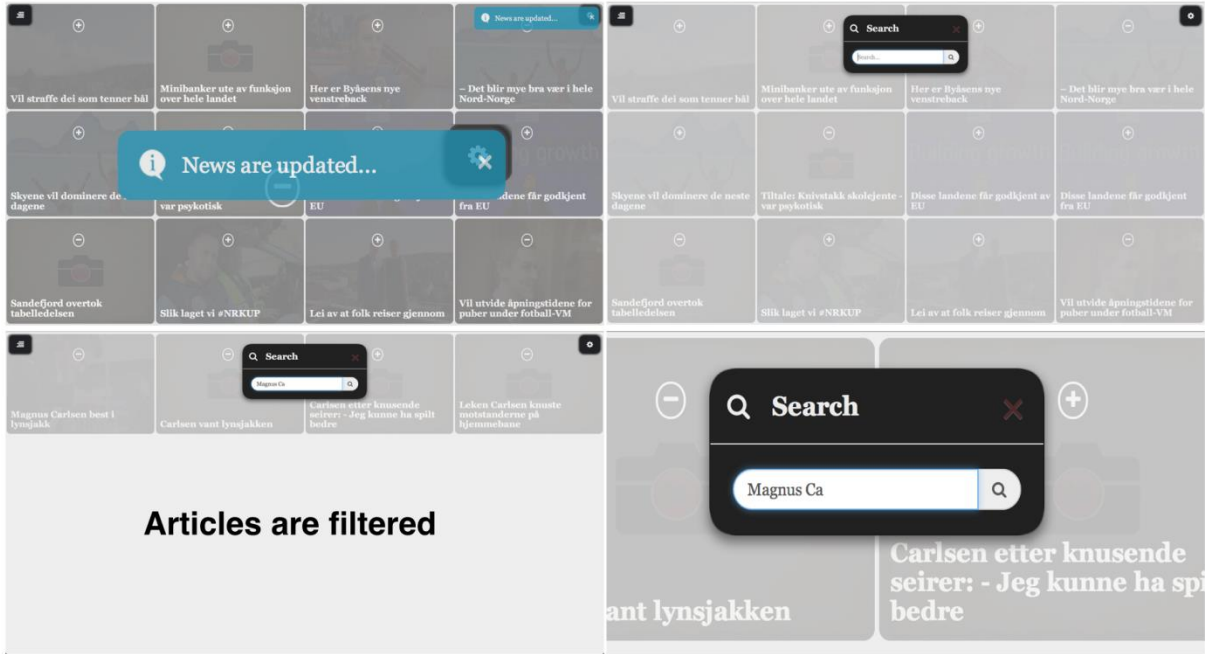


Figure 6.13: Screenshot from the application showing the search feature

Article

This view renders the selected article in full perspective. Each news object contains 1...n tags. These tags are unique and are searchable. When users use the search functionality, the tags appear in order at the top of the page followed by the article's title, image, publisher, date and time, lead-text and body text. On the right side of this view, the articles' control panel (containing three separate panels) is visible. These panels are drop-down panels where they are collapsed by default. The user can un-collapse them by clicking on the desired panel.

The first two panels are reserved for rating and sharing of the articles via social networks. If an article has been rated, the corresponding rate value (amount of stars with yellow color) is displayed to the user. The last panel is the geo-location panel showing the user's location on a Google map. The map is hidden inside the panel. Beneath the control panel, there is a back button. This button is connected to the scroll bar, meaning that it will follow the users as they scroll downward the page to read the full news article.

The users can change the article view in two different ways. First of all, they are able to change the background – and the font color. By clicking on the "B" button, one can choose between four different colors. Secondly, the users can increase or decrease the font-size of the article by clicking on the buttons specified with small and large "A". Changing the color or the font-size occur with a fade-in and fade-out effect where the old value is replaced by the new value.

Profile

The profile is a compact widget that appears on a modal window, and can be accessed any time during the application's run-phase through the left-side navigation menu. Only signed-in users have access to their profile. The button named "Edit profile" will replace the user's current profile properties such as first name, last name, birthdate etc. into corresponding input and HTML select fields. If this is the first time, the user does not have any stored values and these fields will be empty by default. Users can change any of these properties and store them. They will be notified in case of success or failure.

The entire UI is divided into two parts. The first part is the navigation menus which are static. They are placed hidden for the users on the left and right side of the view, and slides out only on demand. Between the navigation menus is the dynamic container (second part) that frequently changes its content. New views are removed or replaced with other views into the container with a slide-in and slide-out animation. The user interface is responsive, flexible and can adapt to any screen size. Last but not least, the layout of the application is developed to be simple to use and easy to learn.

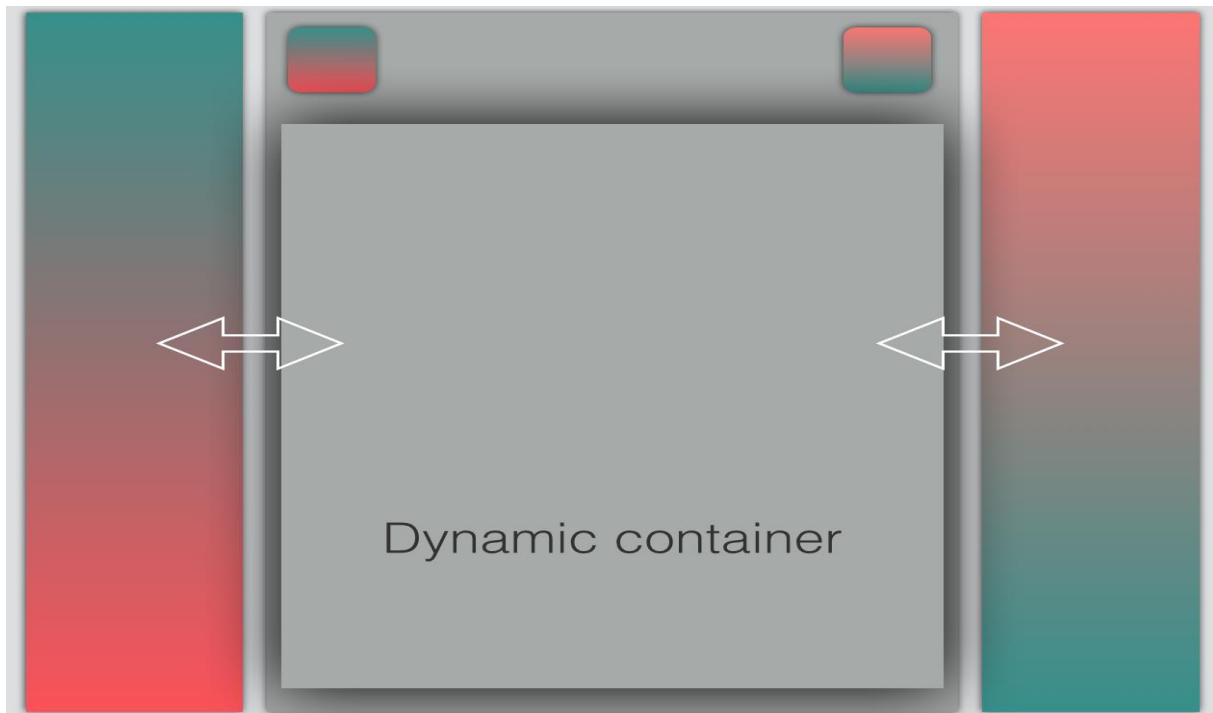


Figure 6.15: Image of the application's design structure

Loading

The UI of the application never freezes. Regardless of the application's state, the user can always interact with the application. All requests to the back-end server and other part of the system are done asynchronous without depending on the graphical user interface.

It is used two types of loading animation in the application. These are the progress bar and the spinner animation. The progress bar shows the user how long it is left to the next action. It starts empty with a white background color and ends with another background color that takes place in the bar from left to right. The spinner rotates for a period of time and is used to notify the user to wait, meaning that something is going on in the background. Both of these loading techniques are known to most users, since they are frequently used in different applications and mobile devices today.



Figure 6.16: Image showing different loading types

UI - Index

This view is a pre-main view for the application and shows the SM application logo to the user. Both the background color and the logo differ each time the view is rendered on the page. The content on the view is centered on the page, and the assigned color will be the background color for the rest of the run time.



Figure 6.17: Screenshot of the Index view

UI - Home

This view represents the SM application's main view. It adds a more personal and cultural meaning to the application by shuffling through different pictures of Trondheim and NTNU. All pictures fills hundred percent of the page, both horizontally and vertically.

The layout is noise-free and contains two quick action links to the most important actions that the user can complete (Log in/Log out, reading articles and sharing the application). All these actions are accomplished only by one click. Due to the buttons' importance and functionality (making them easy to spot and hard to forget), they are colored with strong colors. There is just enough amount of textual information on the page about the project (not confusing the user with too much information). In addition, this part is hidden by default and the user must scroll the view to be able to see it.

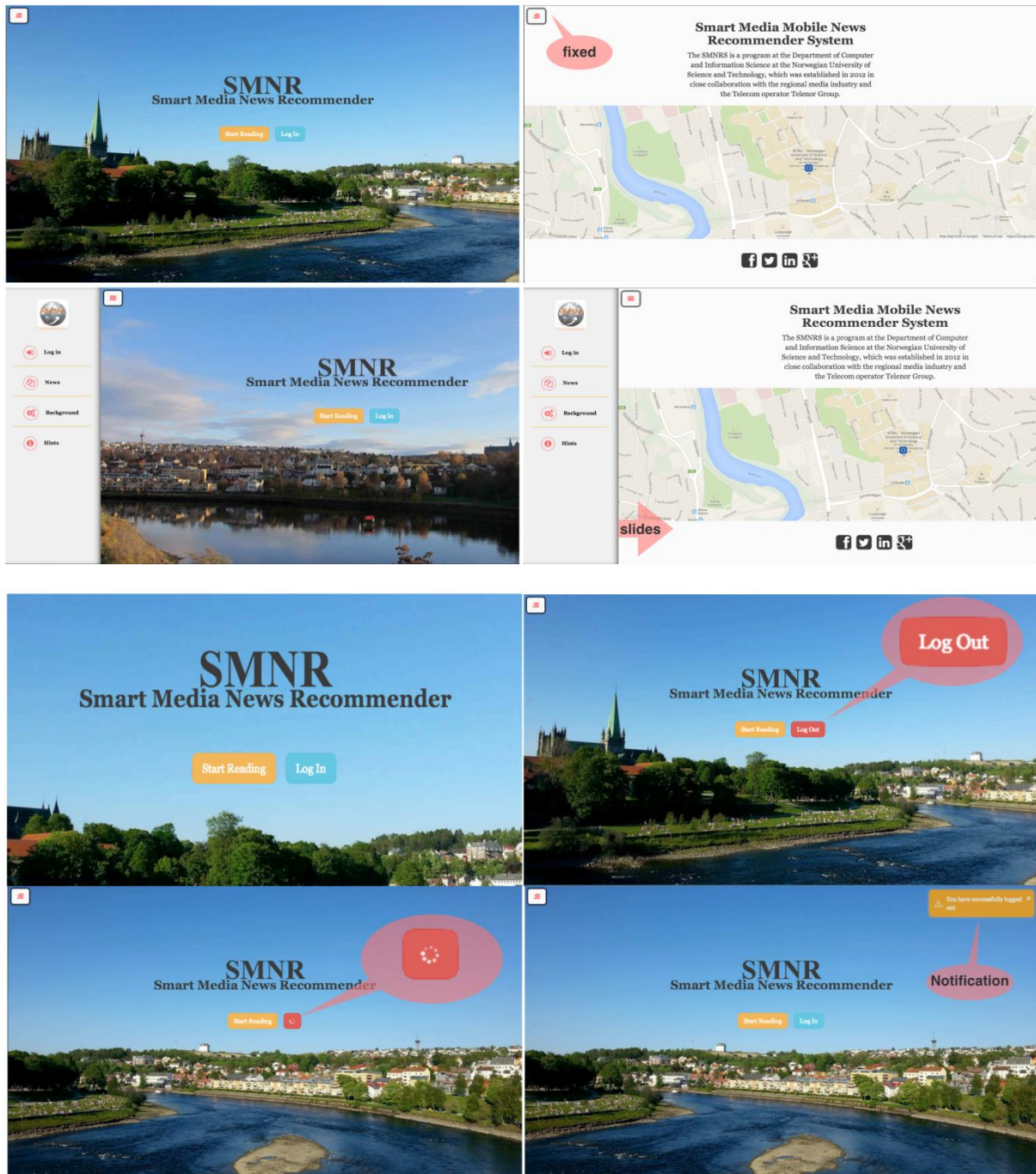


Figure 6.18: Two screenshots of the Home page

UI - News

The News view is definitely where the most amount of information is presented to the user. As mentioned earlier, news articles are presented with the RSS perspective and the user is able to see more than one article at time. When the view is finished loaded and content takes place in the page, the view slides downward (a fall-in-place effect). The user can switch between seven different views/layouts.

Each news object on the view is assigned a container/box (this is the article box). To make the separation between each article more visible to the user in a graphical matter, the articles are always separated by a clear, consist box-border color and space between them (left, right, top and bottom separation). Furthermore, it is implemented a hover effect on the articles which makes it easier for users to see if they have the mouse over the selected article. The articles are always separated in rows and columns. The separation is vertically and horizontally aligned and gives the user feeling of order. Each row will always have the maximum amount of given articles, until all of the fifty articles have taken place. Mathematically this means that if the current view contains of four columns, then it must be at least twelve full rows positioned under each other, plus one more row containing only two articles ($50/4 = 12 + 2$). Some views contain more or less than four columns, but the logic is still the same.

News views

Below it will be given a description of how news articles can be presented in different layouts. Users are able to choose these layouts or views as you may call it by clicking on the right navigation menu and select the layout link in order to change the news view. The names of the different views/layouts are the following: basic, card, details, image, list, main and map.

In this section where it is referred to different screen sizes, it is important to be aware of the following conventions:

- Large screen ≥ 1200 pixels (normal sized desktop and above)
- Medium screen ≤ 992 pixels (smaller sized desktop)
- Small screen ≤ 767 pixels (tablets such as iPad)
- Extra small screen ≤ 480 pixels (smartphones, iPhone)

Basic

It is a clean and ordered layout which is very simple to use. The image of the article is in focus, followed by the title and the lead-text. The layout is centered and has grey-dark background. It suits well for touch devices, but definitely most suitable for desktop devices

where the cursor/mouse is visible to the user and there is a lot of animation on each article box.

The news articles that contain a valid source image are displayed at the top of the page in a carousel bar which automatically shuffles through the news object list for each ten seconds. Each article slides out from left to right. The user can also control this shuffling by hovering over the carousel to pause/stop it, which makes users to be able to read the title and the lead-text in their own pace.

The images of these articles are the background of the carousel container. The title and the lead-text of the articles are placed above each image. Since the color of these images is based on coincidence, it is unpredictable to define a specific text-color for the title and the lead-text. Therefore the text (white colored) is placed on a transparent black box (the background image is still visible). As the carousel changes content (switches to the next article), the container shifts position from a higher value on the Y-axis to a lower value. This gives the box a slide-down effect (see Figure 6.19). The transition takes four seconds.

On a touch device, the user can control the carousel by dragging left (to go to the next article) or right (for previous article).

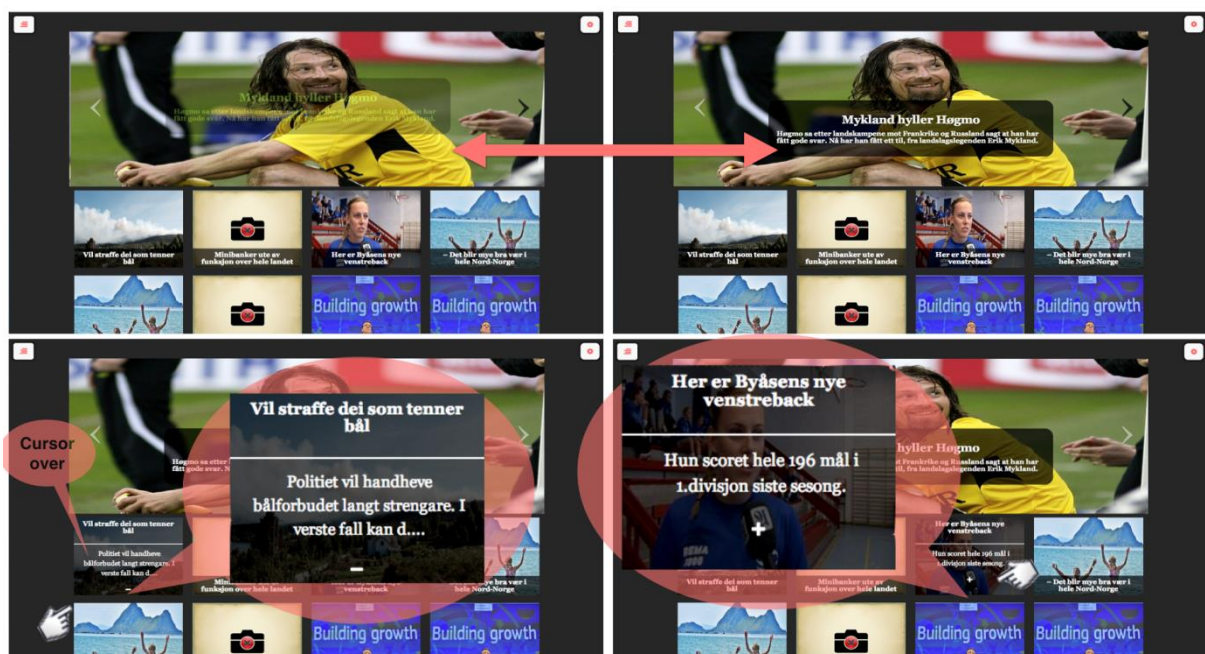


Figure 6.19: Screenshot of the Basic view in large screen

The remaining articles are divided into four columns. For medium size screens, each row contains three article boxes. For small screens two and for extra small screen one article per row. The images of these articles are in the background of the article boxes. Only the title is visible to the user (using the same colors as mentioned above, white text on black transparent). This is the foreground of the box. The foreground slides from bottom to up when the mouse enters a single article box. It fills the entire article box. Since the foreground is transparent, the user can still see the image which is in the background. At the same time, the article's lead-text become also visible and the user can see if the article has been read before (plus sign indicating read article and minus sign for no-read article).

Figure 6.20 shows that the application can load new articles while CSS animation occurs.

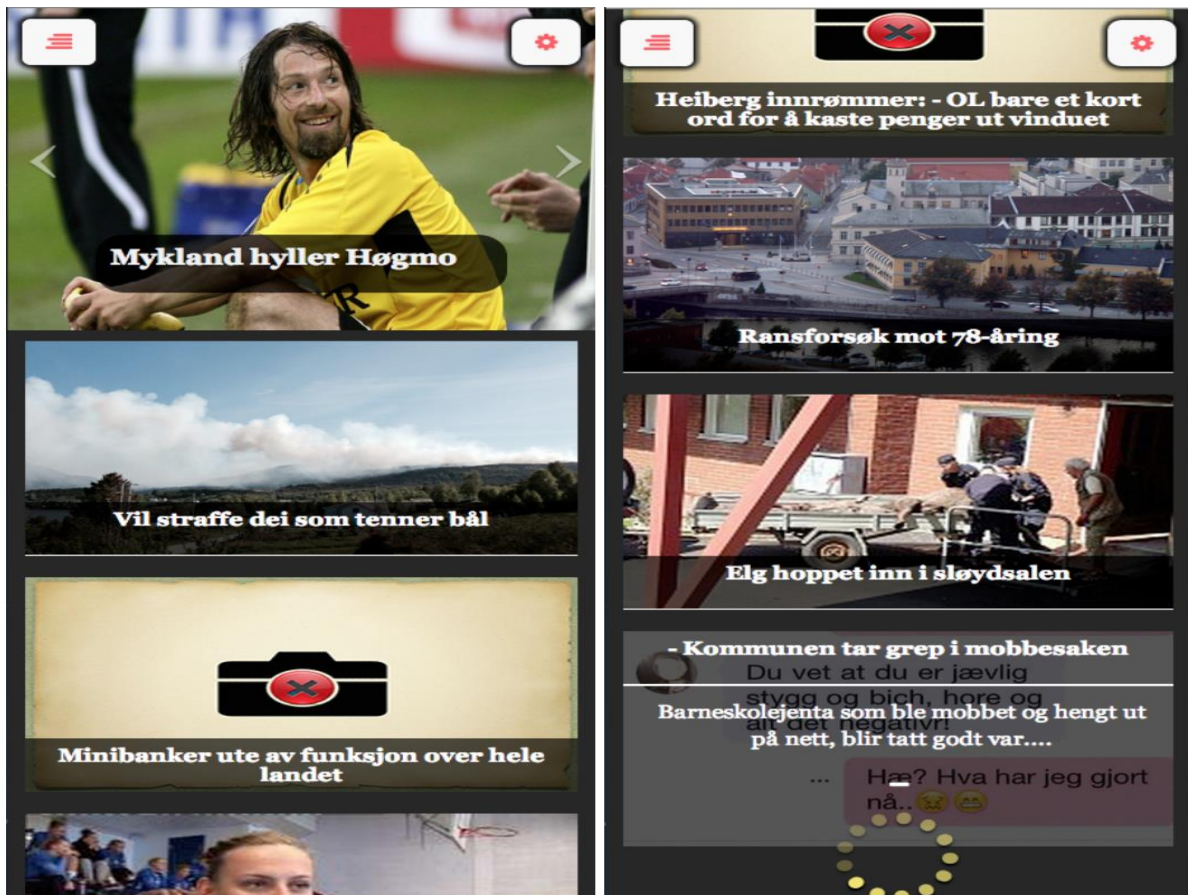


Figure 6.20: Screenshot of the Basic view in small screens (such as smartphones)

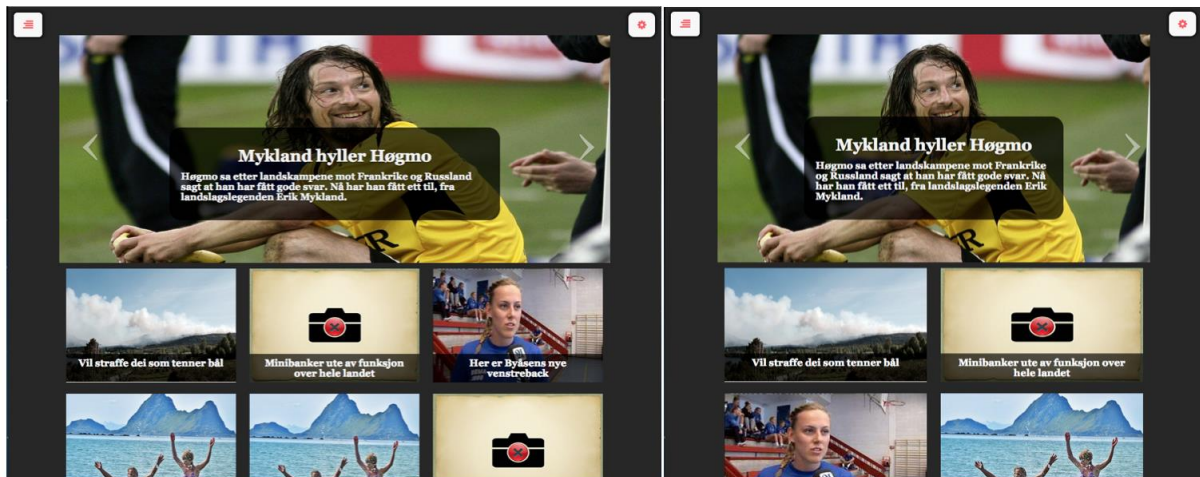


Figure 6.21: Screenshot of the Basic view in medium screens (such as tablets)

Card

This is the smallest and simplest of all the news layouts in the application. It suits perfectly for smaller sized screens and touch devices. It has been used bright colors in both the background and the rest of the layout. The content is centered.

Each article box displays the news articles with title, a small size image (at the right), and ten percent of the lead-text at the bottom of the article box. There are enough spaces for three article boxes at most, two on medium and small sized screens, and one on smartphones.

It has not been used any CSS effects on the article boxes..

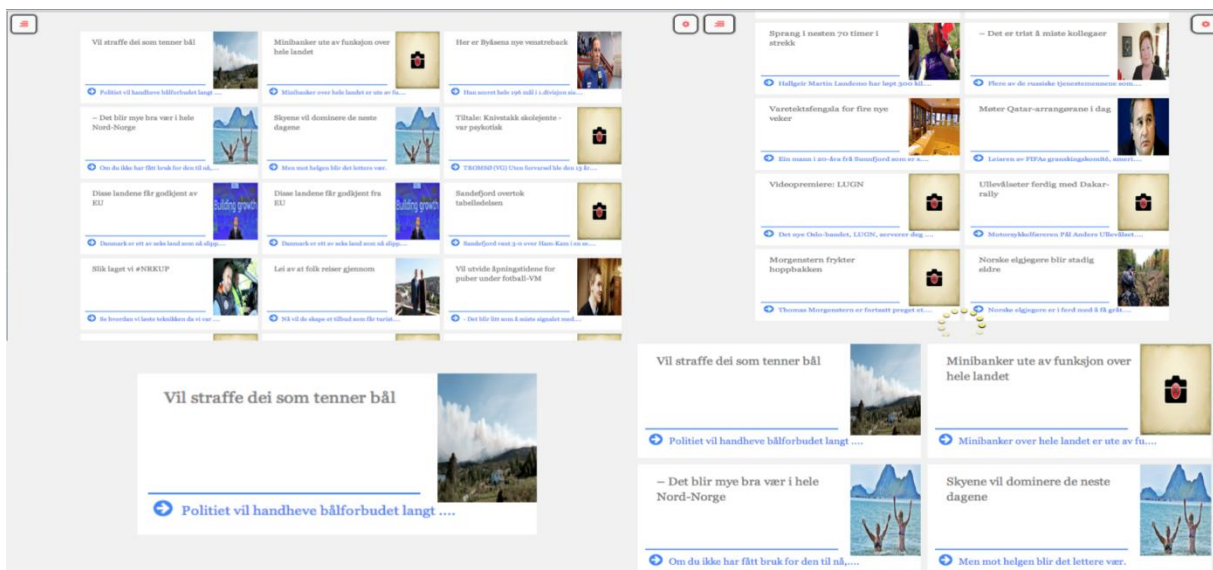


Figure 6.22: Screenshot of the Card view in large and medium screens (such as desktop and tablets)

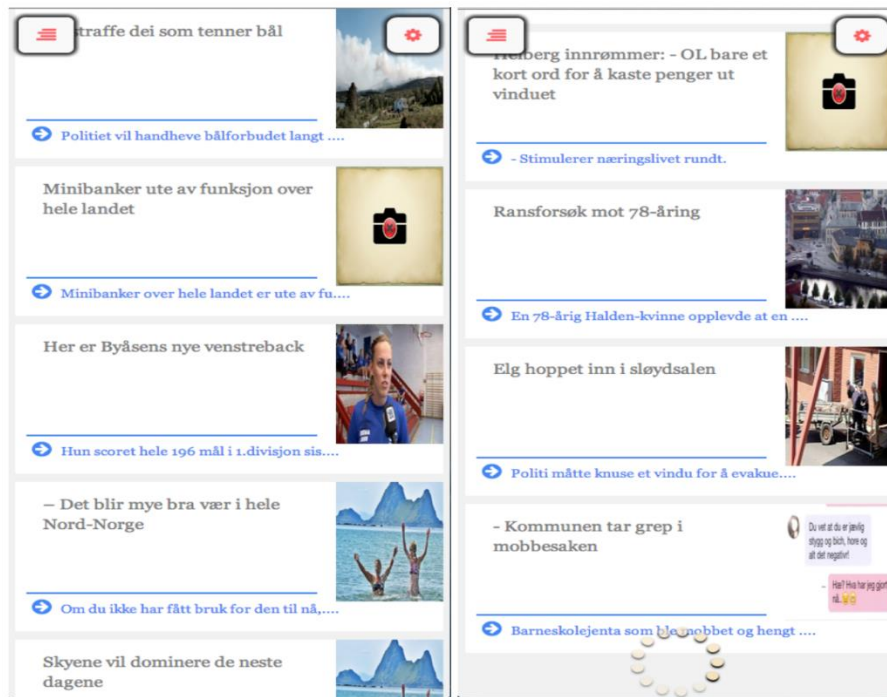


Figure 6.23: Screenshot of the Card view in small screens (smartphones)

Details

This is the most complex view in terms of having information attached to it. The RSS perspective is expanded with more information in this layout. It works perfect on any devices and screen sizes. The background color is black, but nearly visible due to the closeness of each article box. Since there is a lot of information on each article box, each row has a maximum of two articles for large screens and one article for all other screen sizes.

The article box contains source image, title, lead-text, publisher, and published date. The background color for each article box is dark-grey with a black border (same color as the layout's background color). This gives a frame-attached appearance to the whole view. When the mouse is over the article box, the border becomes clearly grey which makes the selected article to stand out.

It has been used different icons to mark the publisher and published date of the articles. Furthermore, the CSS effects for this view are minimal.

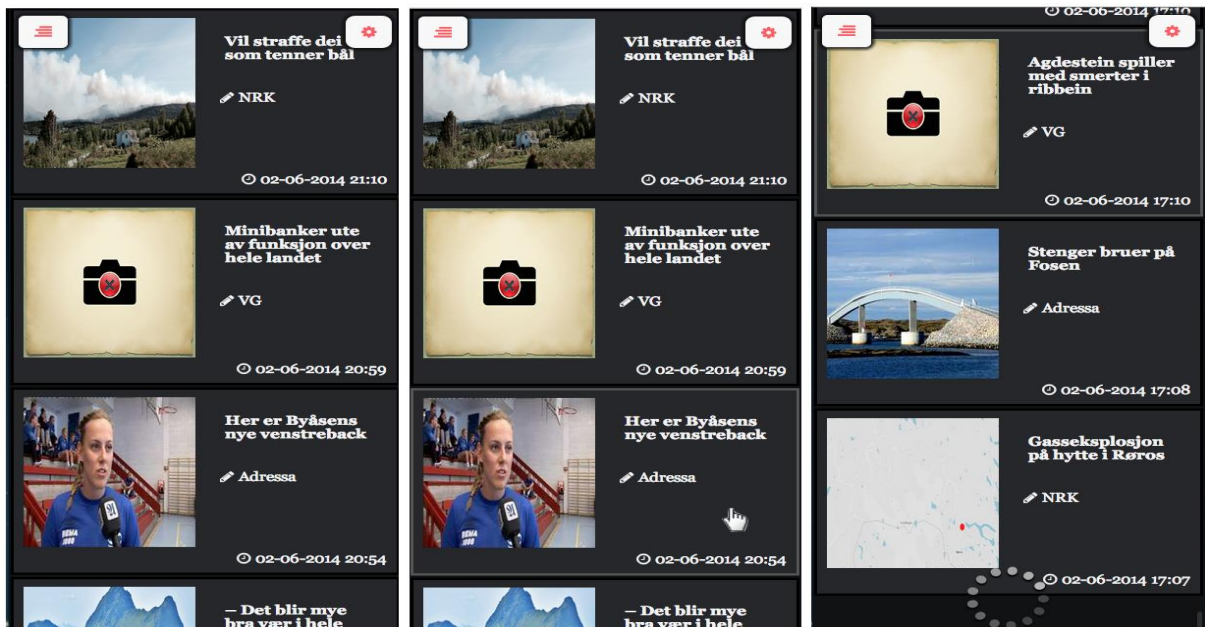
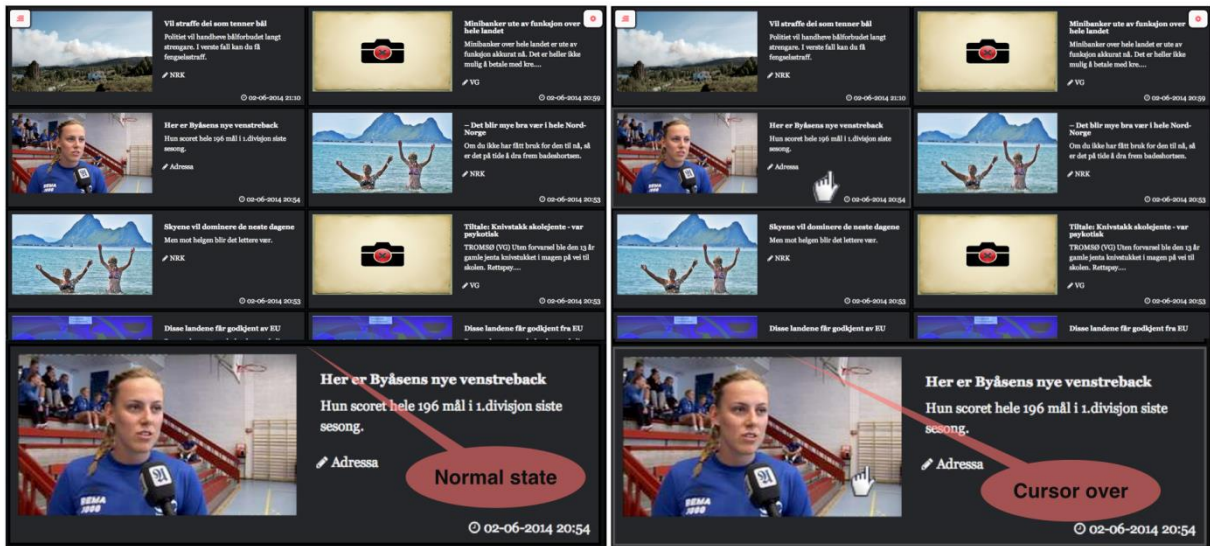


Figure 6.24: Screenshots of the Details view in different screen sizes

Image

The image layout is frame-based with rows containing up to four article boxes depended on the screen size of the device. It fits best for desktop devices. Besides presenting the news stories, the main goal of this layout is to display the articles' status (if the article has been read or not) and to create a modern layout. As the name expresses, the articles' images is in focus, but the images are not entirely visible to the user at first. Each article box is presented with the title on the foreground of the image, and an icon which tells the user if the article has been read or not. By hovering over the article box, the foreground fades out and only the articles' images become visible to the user.

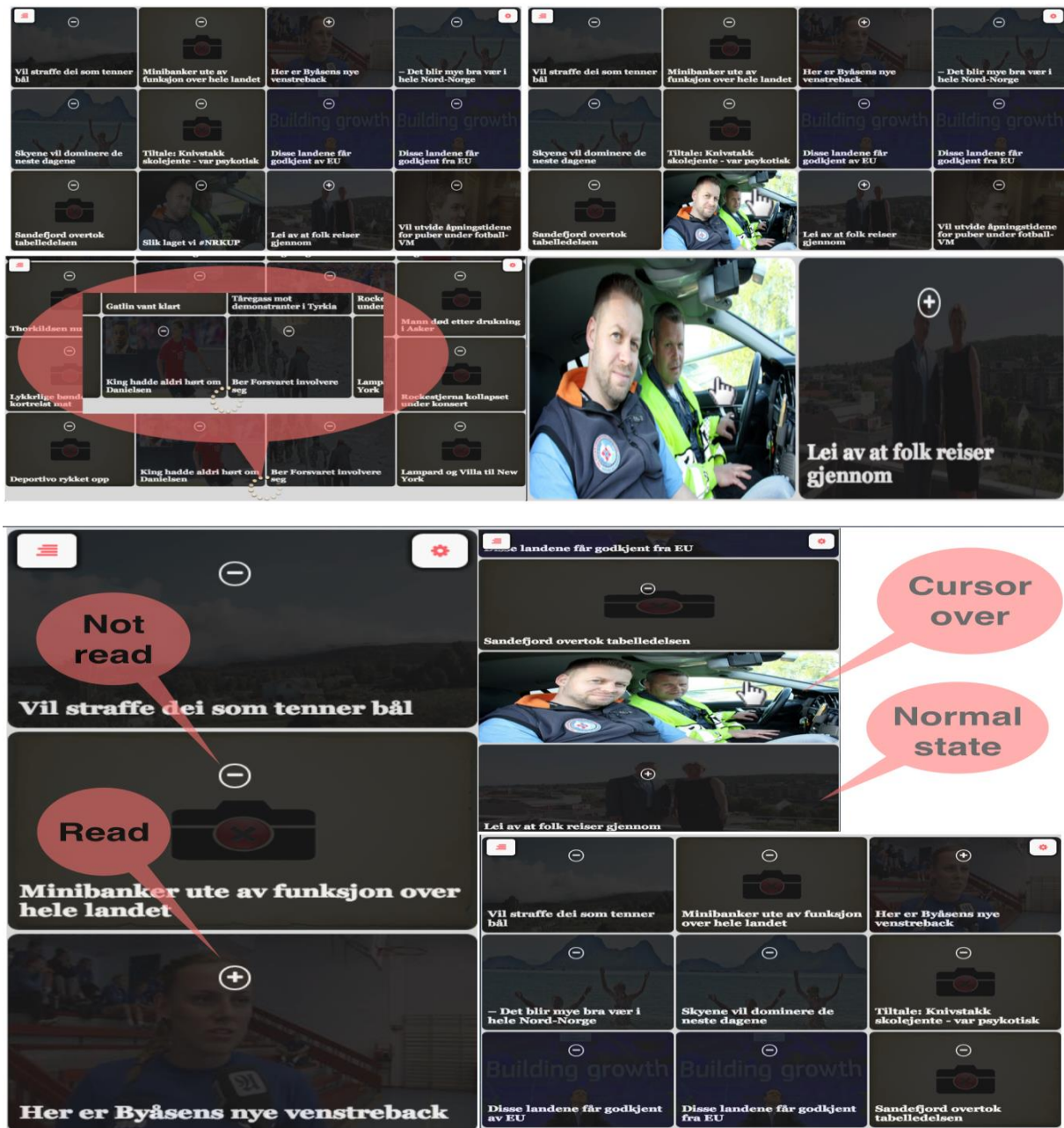


Figure 6.25: Screenshots of the Image view in different screen sizes

List

This layout has the largest amount of articles in each row. It starts with displaying twelve articles per row for large screens and two articles for small screens. It uses earlier mentioned techniques to create a frame-based layout. The user is almost able to see up to fifty articles at the same time on large screens. This makes it easier to select the desired article. For optimum use, this layout fits best for devices with cursor or large screens.

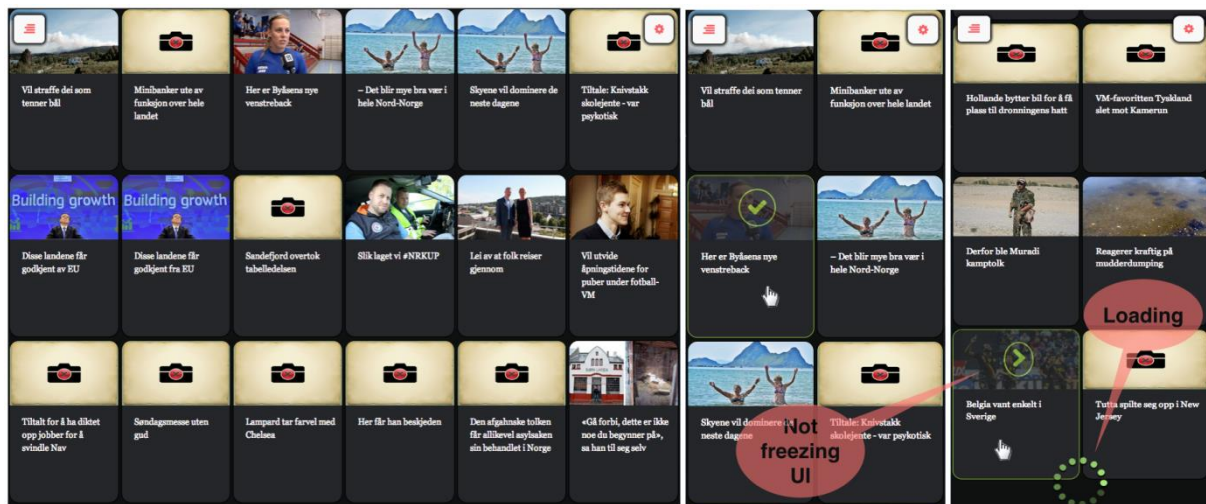
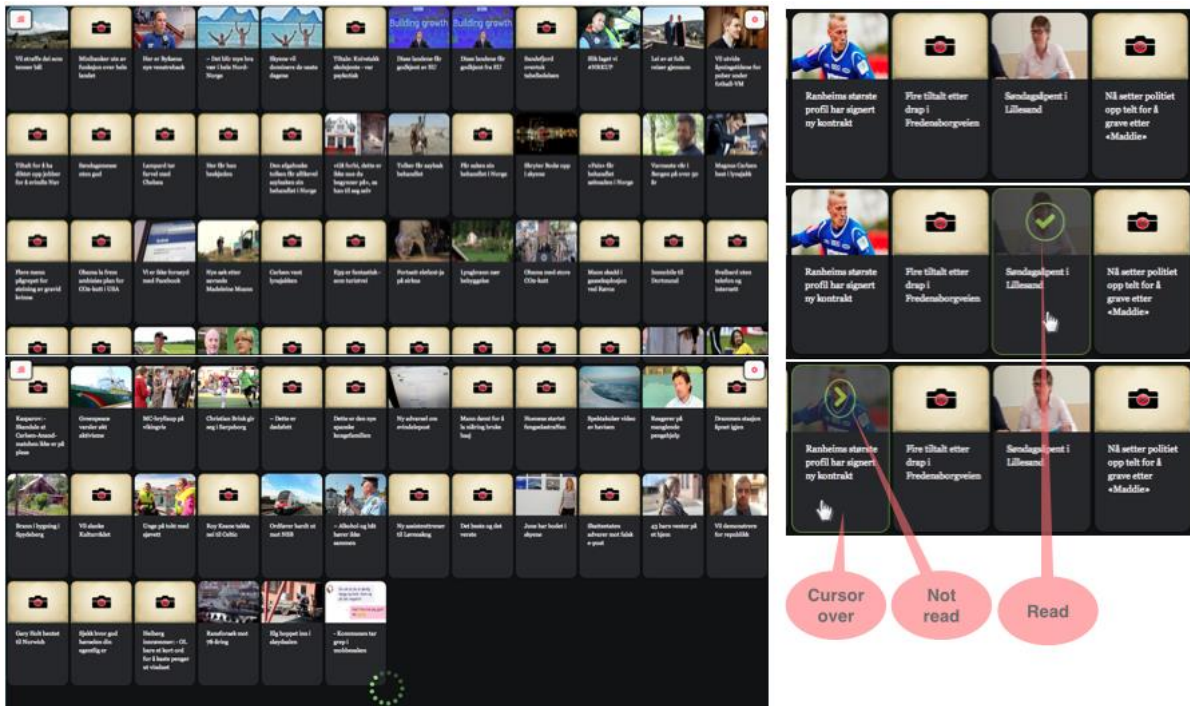


Figure 6.26: Screenshots of the List view in different screen sizes

The images are minimized to fit the container. Each box is having the minimal margins on each side relative to each other. Since there are so many articles visible to the user at the same time, only the title is shown in the container to increase readability.

The text color is white on dark grey. On hover, a transparent foreground will appear just on the hovered article image. The title is still clear and readable. On this foreground, the user is notified if the article has been read previously or not. In case of read article, there is shown a checked icon. If the article has not been read, a forward icon is used to indicate that.

Main

This layout is designed almost like the Basic layout. It has a carousel bar at the top and the remaining articles beneath it. The only difference is that the remaining articles are listed as the Image layout.

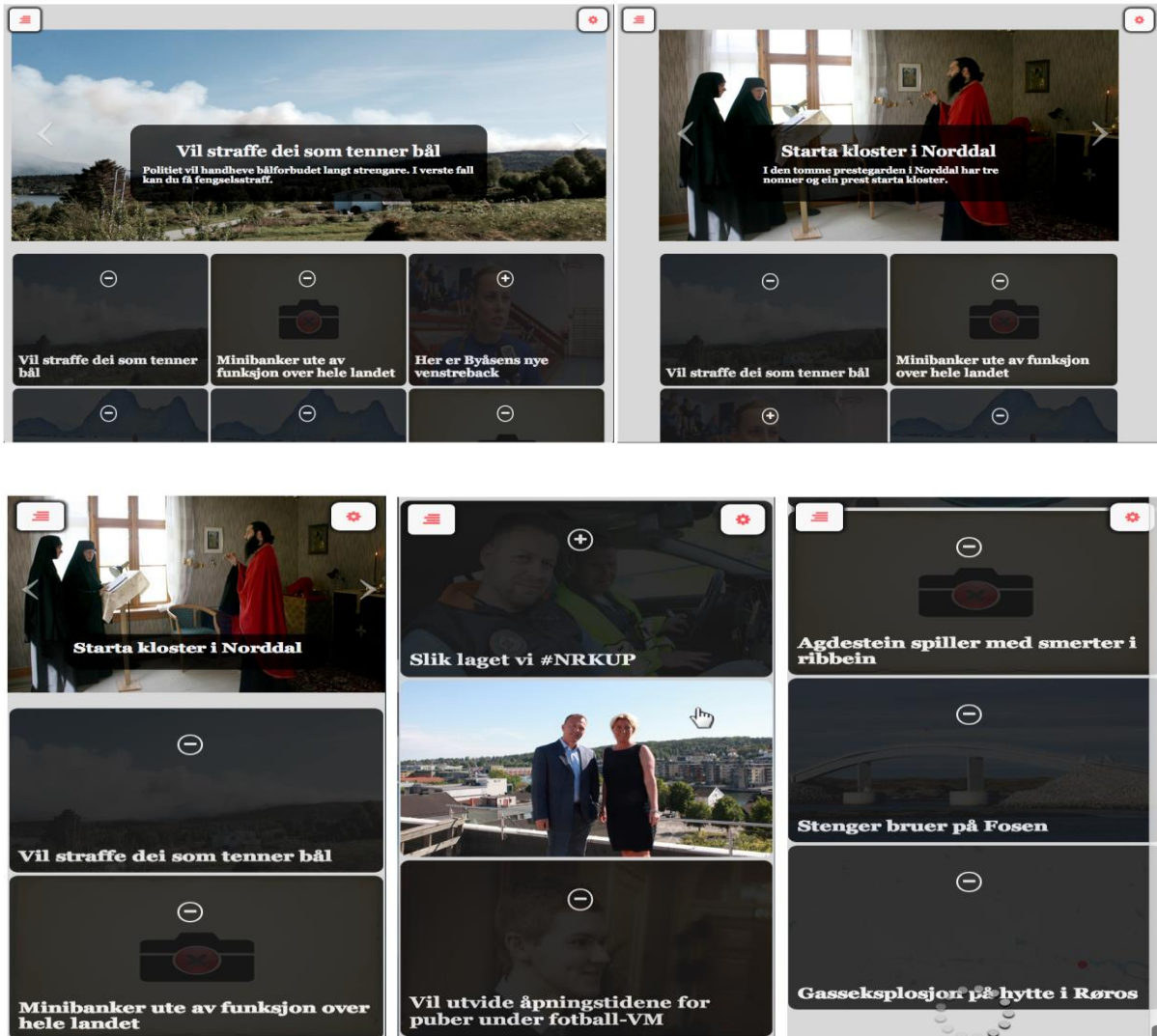


Figure 6.27: Screenshots of the Main view in different screen sizes

Map

The Map layout is probably the most recognizable layout to most of the users. It is a full page Google map with Google marker icons showing where the entities of the article have occurred. It works on all devices, but suits best for large screens. By hovering on any markers, the title and the lead-text appears in a standard HTML window just beneath the marker. By clicking on the marker, the browser opens new tab with the selected article and the user are redirected to it. Navigating back to the Map layout, an information window (pop-up) is

marking the selected article. Now the user is aware of which articles that have been read (see Figure 6.28). The default Google zoom control-panel has been removed, but the user can zoom in the map by mouse click. Double click will zoom out.

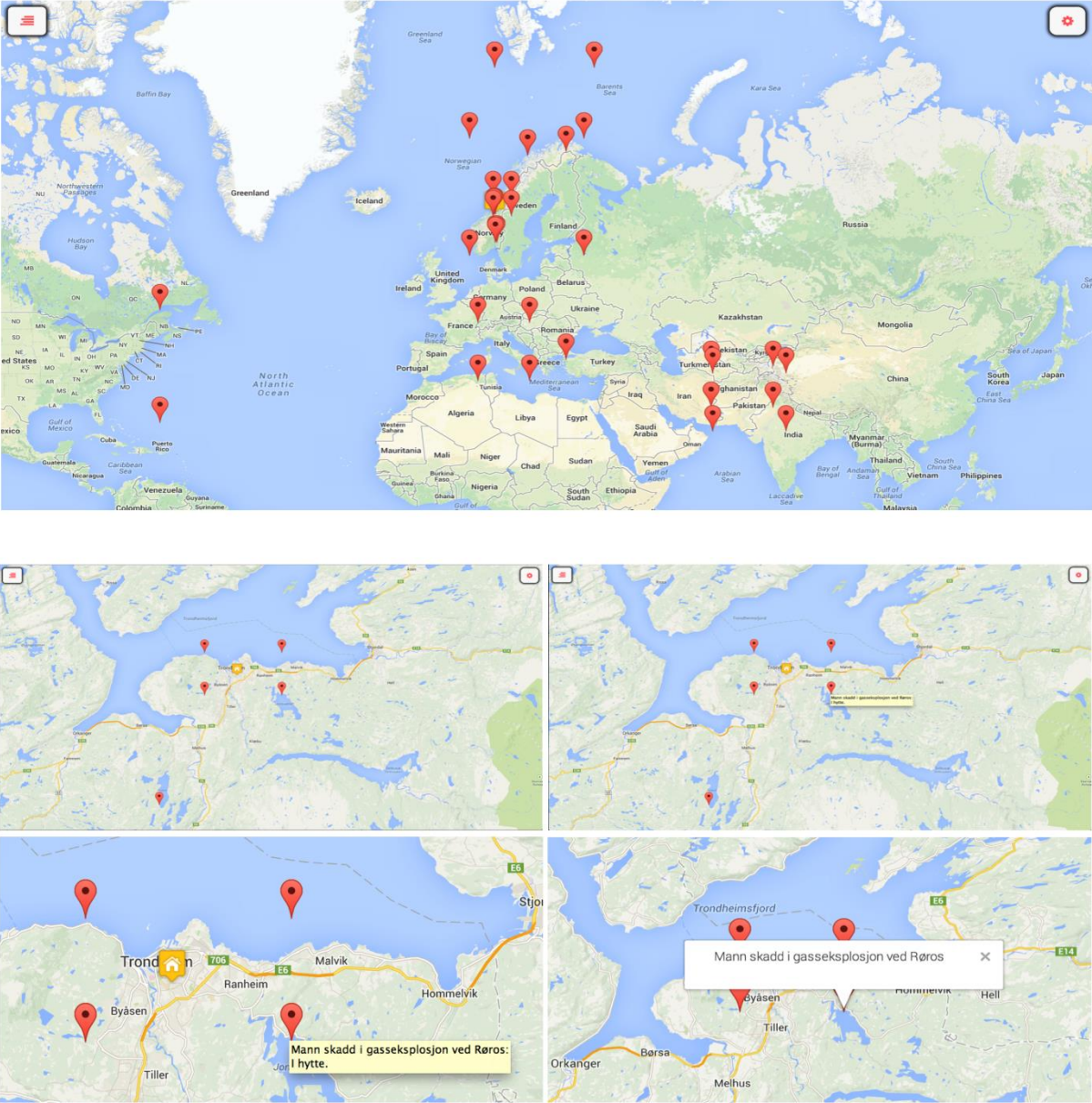


Figure 6.28: Screenshots of the Map view in different screen sizes

Summary of the news layouts

There are several similarities between the layouts (except the Map layout which is standalone). First of all, they are usually centered in the browser's page or are stretched out completely at the available space. Second of all, they are flexible and react to the browser-size changes on run time. There is a large contrast between the background and the lead-text

colors which increases the readability. Usually, there is a small or large amount of hovering effect on the article boxes/containers to make the articles stand out.

They have different types of the RSS perspective. With one click on the entire article box, the user can start reading the article. Recognizable elements such as the carousel bar and the Google map layout are being used to increase the usability of the application.

Last but not least, all the layouts share the same data, so it is up to the user to choose the desirable layout.

UI - Article

Since the articles can be viewed on a single page and a Modal window, the user interface is tied together. The panels to the right are collapsible and are not expanded at first. This is done in order to not confuse the users with all interactable elements and to not take the focus from the articles. Since the article tags are unique, they all have a random background color to separate them from each other. Other than that, the article view is a standard news application view where the user is always met with the title and image, followed by the story. This is the way all users are used to. Therefore changing it would just confuse the users.

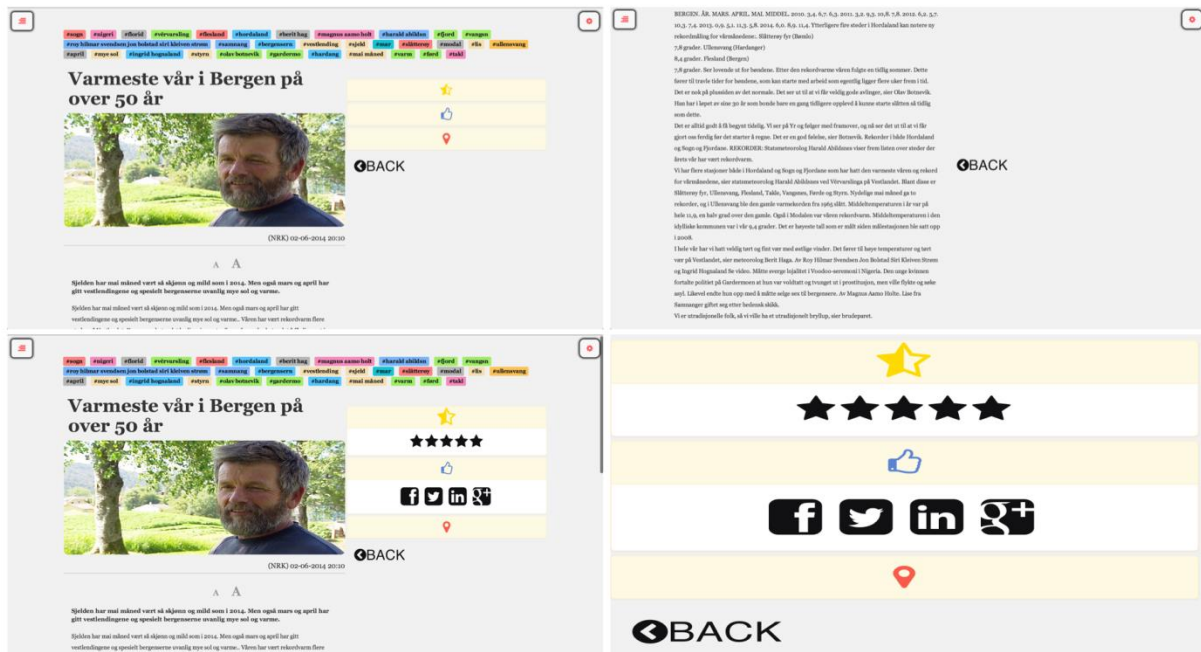


Figure 6.29: Screenshots showing collapsed and un-collapsed right panel

There are a lot of empty spaces in this view. Usually these spaces are used for advertising, recommended news or related articles. Since the backend API did not support or offer

recommended news, it was decided to leave these spaces blank instead of filling it with random articles that were not related at all to the desired/selected article.

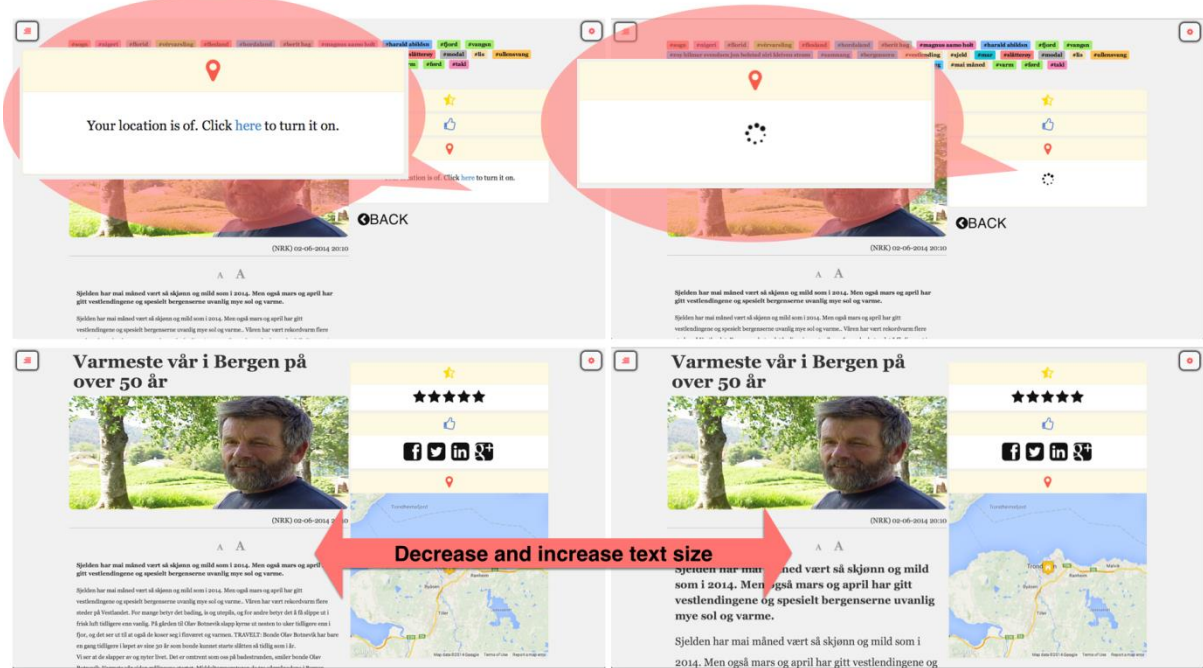


Figure 6.30: Article view (step to turn the geo-location on and increase text-size)

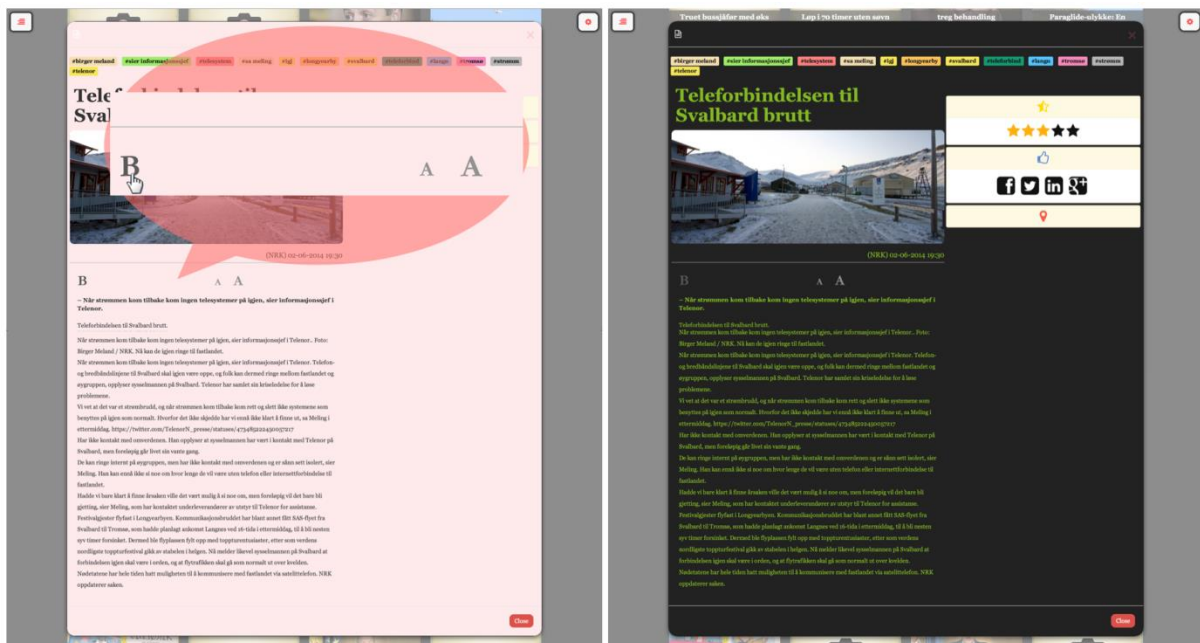


Figure 6.31: A single article displayed in Modal window

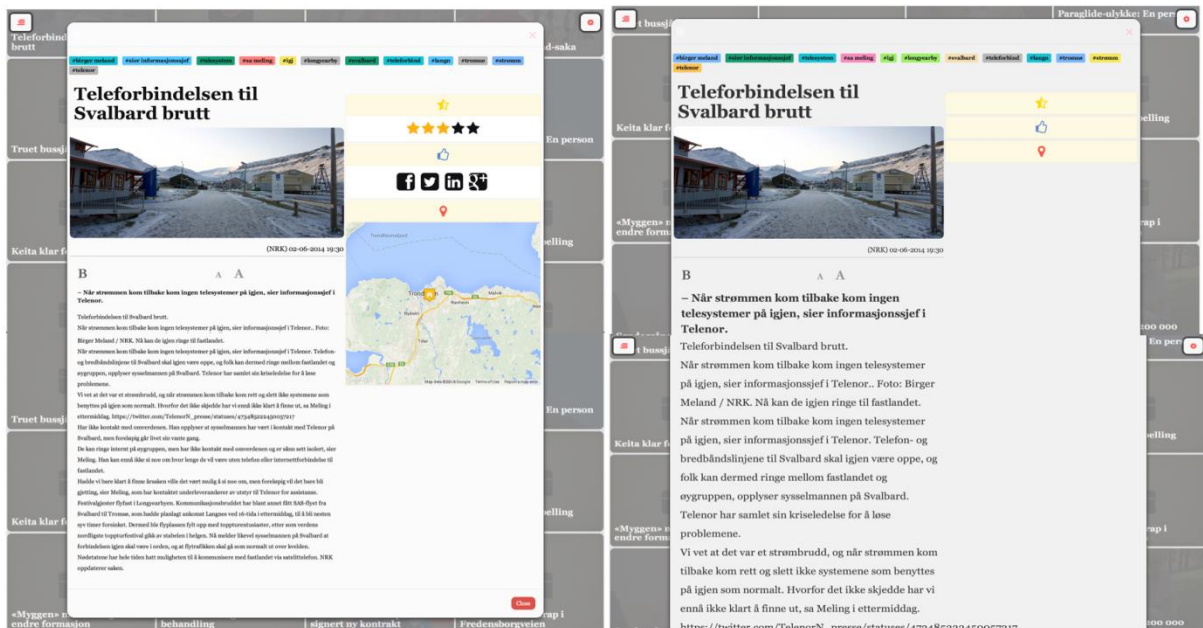


Figure 6.32: Screenshots of a single article in modal window with different background colors

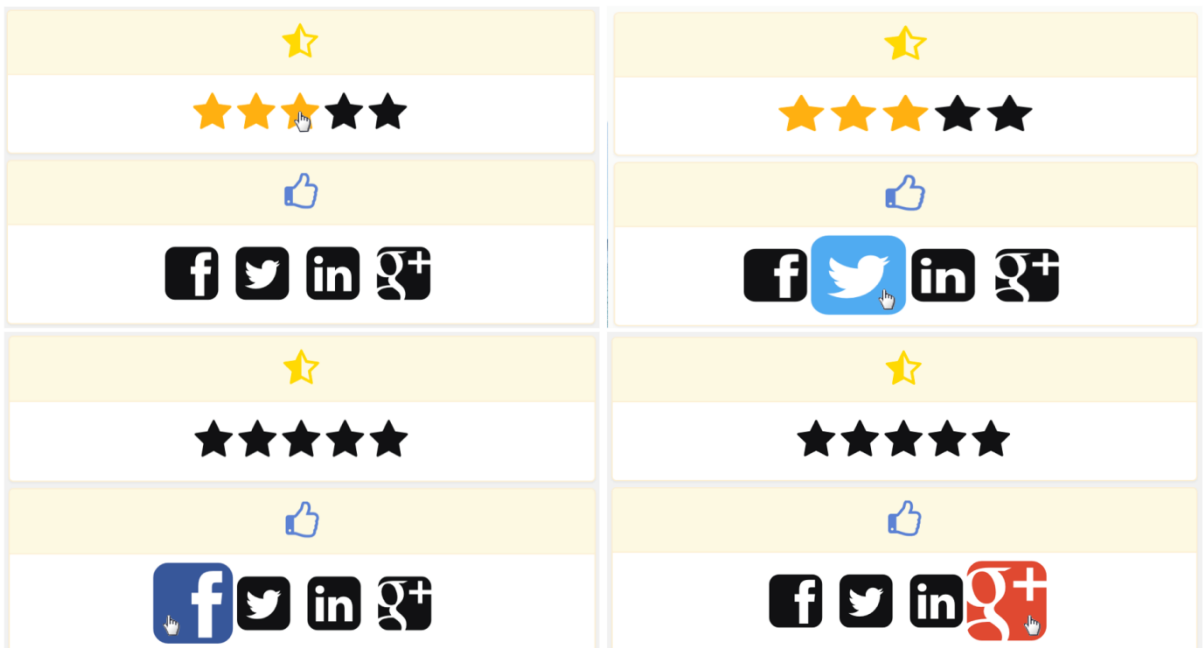


Figure 6.33: Screenshots of the rating and sharing features (with hovering effects)

UI - Input fields

The input fields used in the SM application are large and wide. This gives the user a better readability of what they are typing. Each input field can have four states.

- The standby state (default state)
 - When the field is not selected, this state is colored with black border and text.
- The selected state (when cursor enters the field)
 - When the user selects the field, the length of the field increases, and the borders becomes light blue (text color stays black).
- The error state (when the validation fails)
 - When the field is selected or not, but contains validation errors (email validation, to many or few characters etc.). This state is marked with red border and text color. The length of the field stays the same as it is selected.
- The success state (when the typed text is valid)
 - No matter if the field has been selected or not, if the typed value is valid, the border color becomes light green and the length of the field transforms to the default state.

Users will be able to see these states as they start or end the typing in the input fields.

UI - Profile

This view is two-sided. On the first side (default side), the user can see their information. The information is presented in a white background with black text. The second side is where the user can edit the profile.

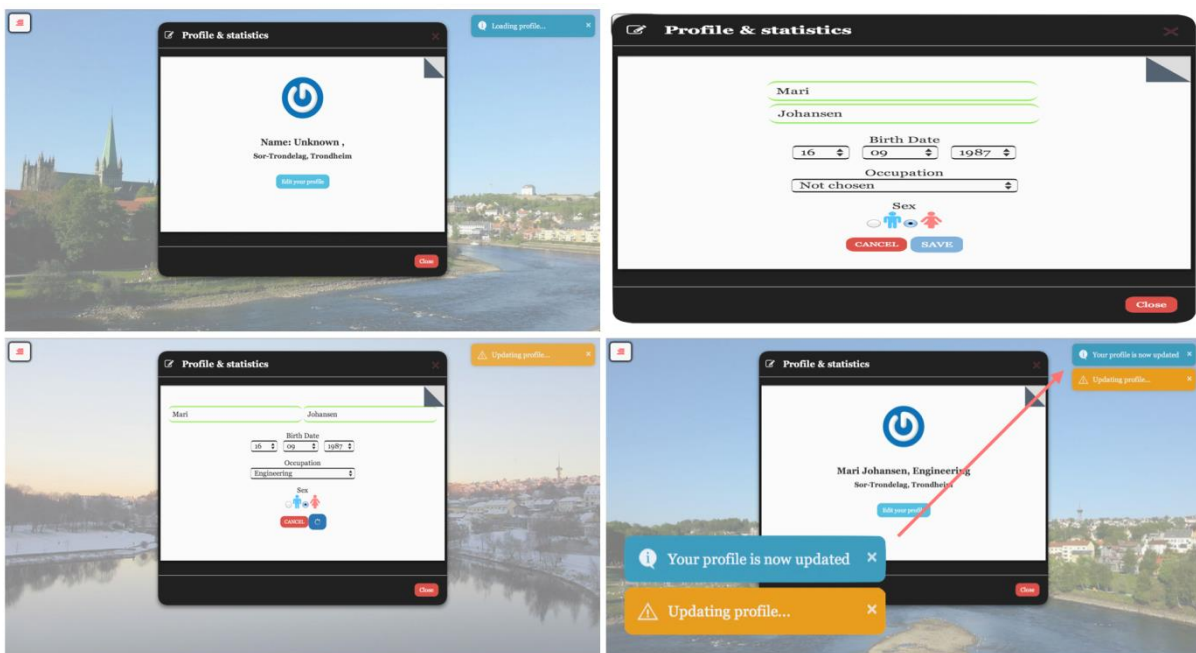


Figure 6.34: Screenshots of the Profile widget

While the first side slides upward, the edit side slides downward (collapsing). Users can either cancel (slides back to the default side) or save (where the label on the "Save" button changes to a rotating circle). When the information is saved successfully, the user is notified by a success notification pop-up and the profile view slides back to the default side.

On medium and large screen sizes, the input fields are horizontally aligned. But as the screen size decreases, these fields will lie on top of each other.

UI - Authentication

This process focuses on simple and fast authentication. The authentication widget is small and easy to use. If the "Hints" setting is on, users will get notification about the username and password as they select the corresponding input fields. These fields are validated using the above-mentioned techniques.

Users can switch between signing in and up by using the corresponding links or click on the grey top-right corner. By clicking on any of them, the widget flips from side to side (180 degrees transformation like a two-sided coin, see Figure 6.35). The "Username" field is auto selected by default when users click on the "Log in" link or when the users switch between "Sign in" and "Sign up" page. Therefore it is not necessary to select this field manually. This makes it easier for the user to type, especially on touch and small size screens.

As shown below in Figure 6.35, the system asks the user to sign in in order to read the selected article.

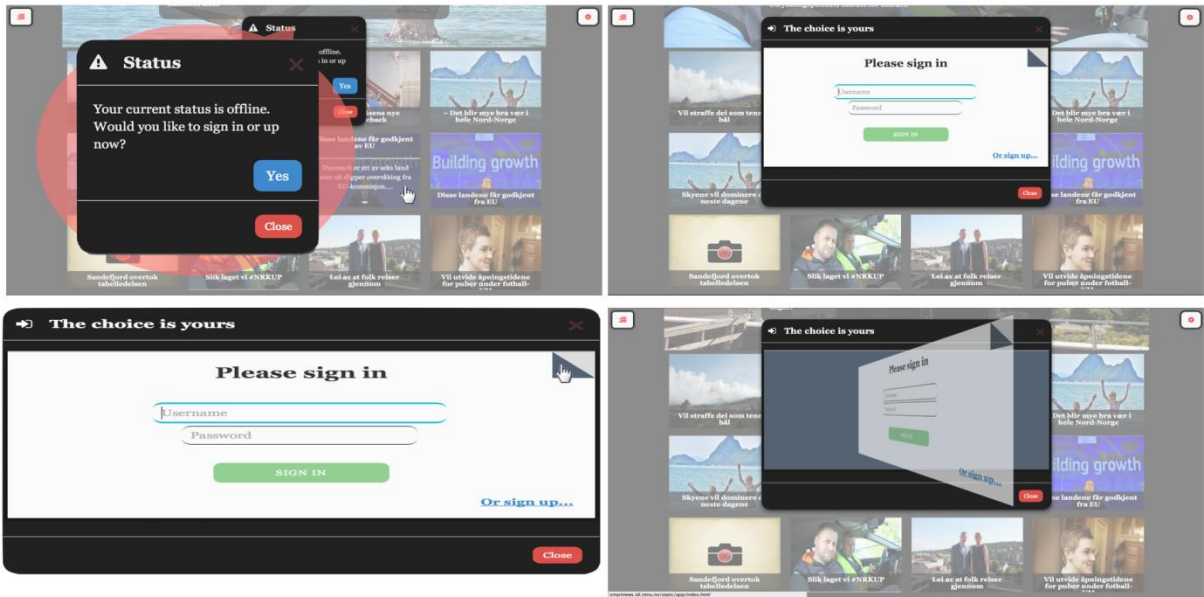


Figure 6.35: Step 1 of the authentication process

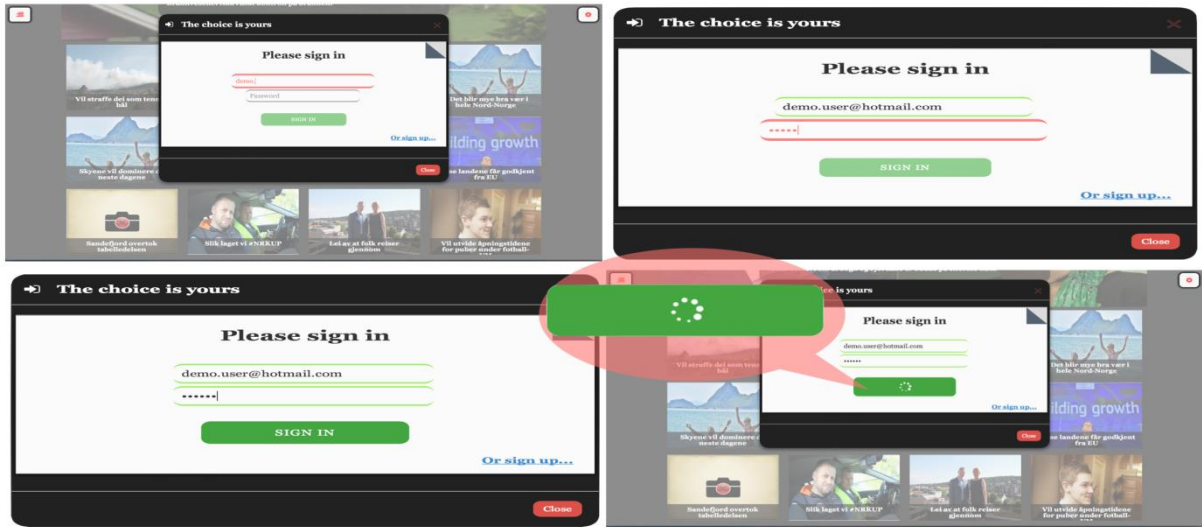


Figure 6.36: Step 2 of the authentication process (input validation)

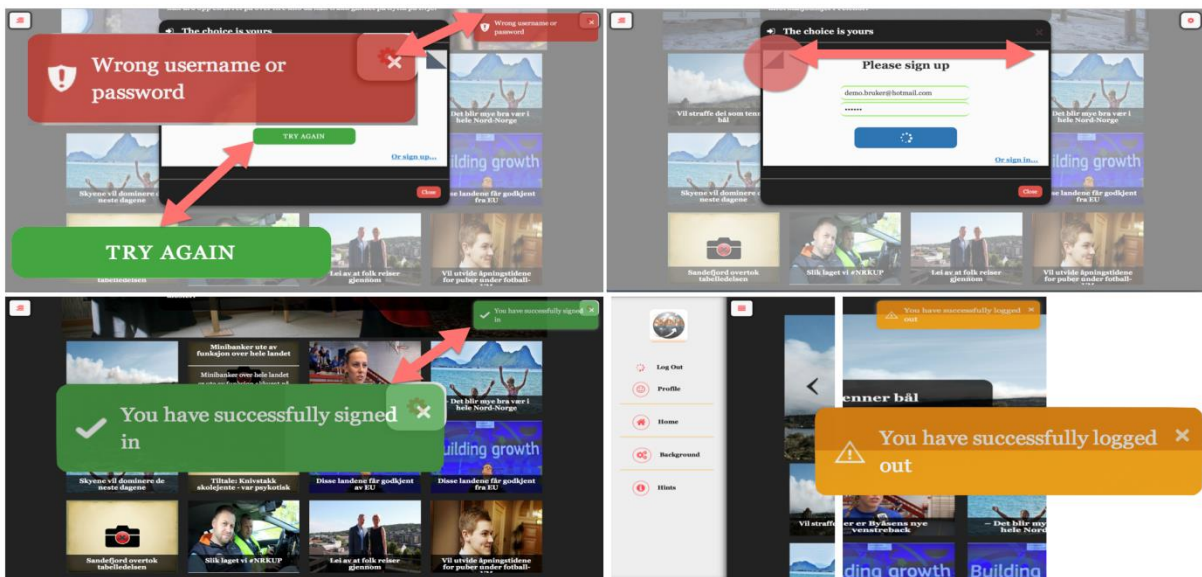


Figure 6.37: Screenshots showing the validation process in case of success or failure

UI - Navigation menus

The navigation menus are based on slide-to-left and slide-to-right navigation. This type of navigation menus can usually be found in application such as Facebook, Twitter, Google, LinkedIn etc. Most users are familiar to uses this kind of menus. By clicking on the menu buttons, the navigation menu will appear from left-to-right and right-to-left pushing the entire view along the process. Additionally, the menu -button icons change since it is different icon for close and expanded menus. Because of this behavior, it is not possible for the user to

slide out both menus at the same time. When one of the menus is expanded, the other menu button becomes invisible.

The menus are compact and flexible. They add or remove different links and buttons to and from themselves based on the user's current position. Each view has a different link anchored to it. If the user is located in the Home view, then the link to the Home view is removed because the user is already in the Home view.

Users can change the background color of the menus at any moment during the run time. Changing the color will take effect immediately without any need of reloading the page/view.

The navigation menus are following design principles such as 7±2 rule and three-click-rule. For instance, the user can perform any task within three mouse clicks. And the navigation menus contain less than 7 links.

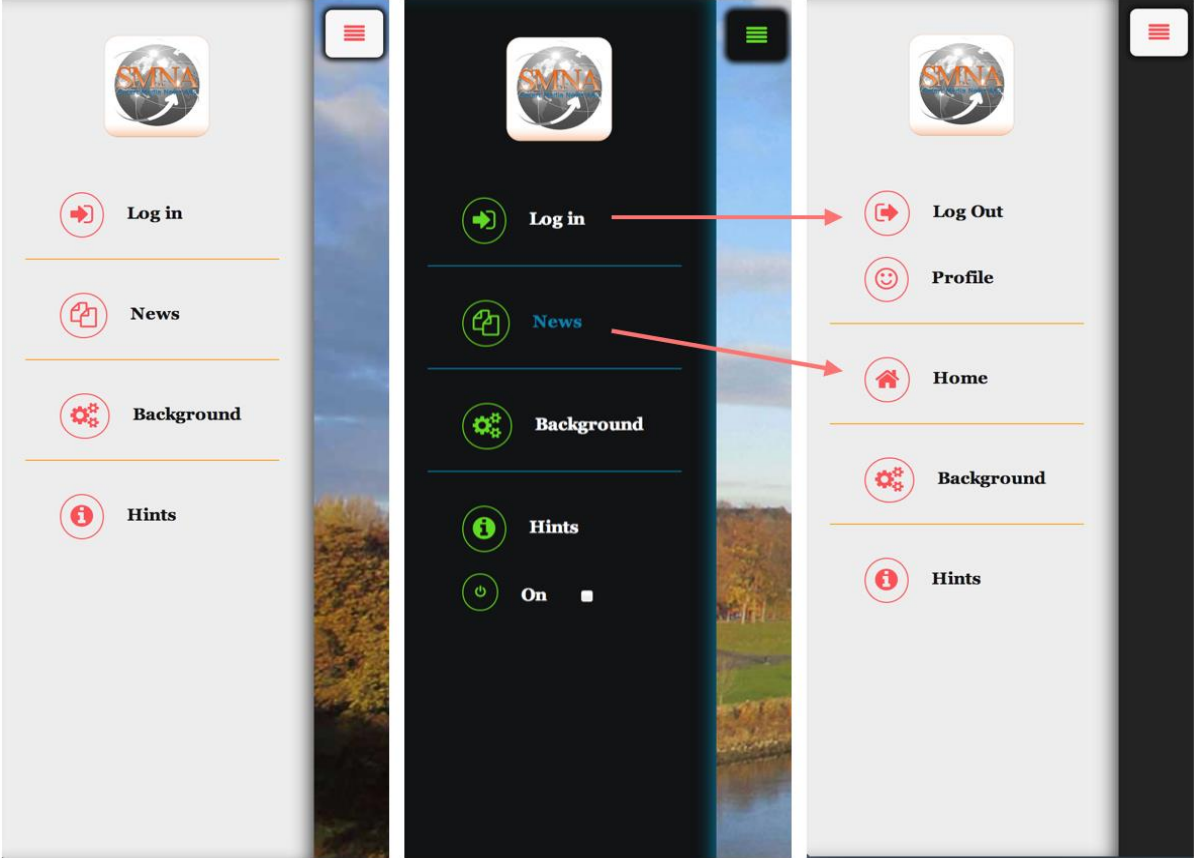


Figure 6.38: Left-side navigation menu showing different user status and view

The left-side navigation menu and its items (links and buttons) are related to the application and the user account. If the user is offline, then there is a button for signing in. if the user is online, then the button is replaced with the sign out button. Besides of changing from sign in

to sign out, the link to the profile widget is also visible. To change the background color of the navigation menus, the user have to click on "Background" and choose one of the preferred colors. By clicking the "Hints" checkbox, the application's hint system turns on.

The right-side navigation menu is related only to the news articles. Button for forcing news articles to update, changing layout, opening articles in different mode, turning the location on and off, and searching for article are available through this menu. Any actions caused by user-clicks on these buttons will be performed immediately.

In addition, the menus can be expanded on touch screen without using the menu buttons. The user can just drag with finger-touch from the edge of the view and inward to the middle to slide out the menu. Drag from left to right open the left side menu and vice versa. When the menu is open, then drag left to right will close it.

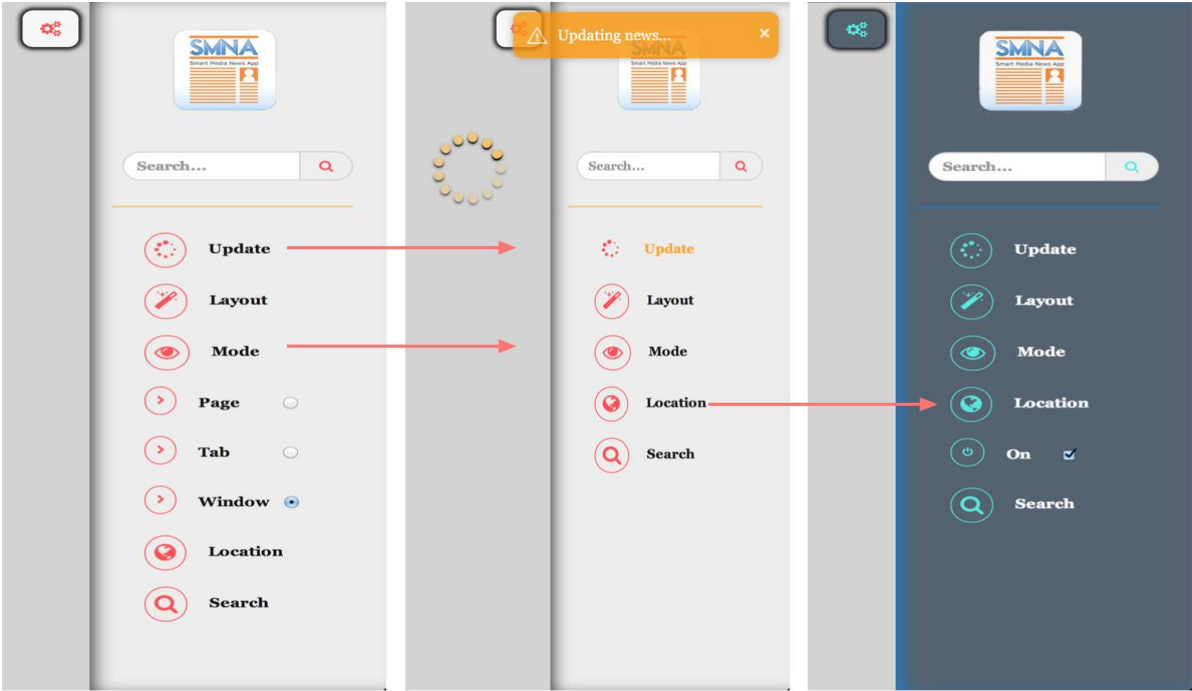


Figure 6.39: Right-side navigation menu

When a new view is taking place in the application, any expanded navigation menu will automatically be closed. This is done to facilitate the whole view to the user. Furthermore, performing tasks such as logging out the user or updating the news articles will be displayed

with loading animation on the navigation menus. The corresponding icons will change to a rotating spinner-animation.

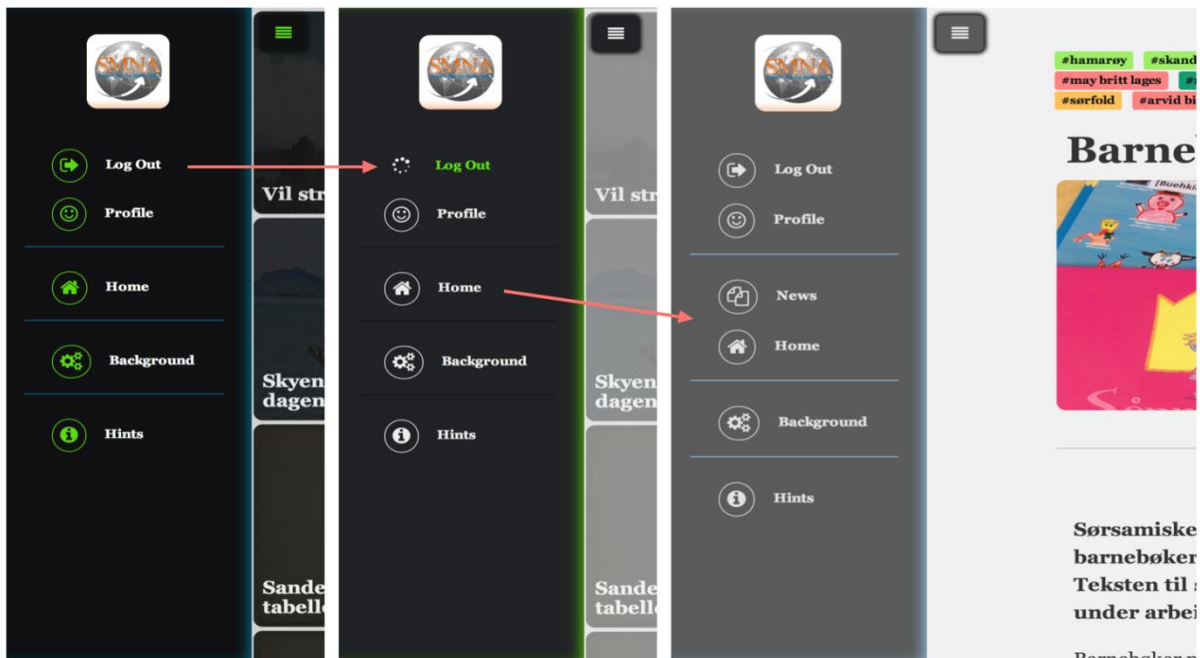


Figure 6.40: Left-side navigation menu during log-out process (news vs. article view)

Part IV

7 Evaluation

This chapter elaborates the evaluation process for the application. The first section provides a brief introduction of different evaluation techniques. The next section gives an overview of the evaluation process. The last section presents the results of the evaluation.

7.1 Introduction

There are basically four ways to evaluate a user interface. These are *formally* (using analysis techniques to calculate usability measures), *automatically* (usability measures computed by running a user interface specification through some program), *empirically* (usability assessed by testing the interface with real users), and *heuristically* (by looking at the user interface and passing judgments based on the skill and experience of the evaluators) [36][33].

According to Nielsen (1994), empirical methods are the main way of evaluating user interfaces, with user testing probably being the most commonly used method. Several studies have shown that usability inspection methods are able to find many usability problems that are overlooked by user testing, but that user testing also finds some problems that are overlooked by inspection, meaning that the best results can often be achieved by combining several methods [33]. Unfortunately in most practical situations, people do not conduct empirical evaluations because they lack the time, expertise or inclination to do so [36].

Jeffries et al. (1991) did a research work where they let four groups to evaluate the user interface for a software product, each applying a different technique/method: heuristic evaluation, software guidelines, cognitive walkthroughs, and usability testing. They found out that the heuristic evaluation by several UI specialists uncovered the most serious problems with the least amount of effort. Additionally, they also reported a large number of low-priority problems by using heuristic evaluation [25]. The advantages and disadvantages of the four usability evaluation methods can be seen in Table 7.1.

The most widely used inspection method is heuristic evaluation. As mentioned earlier, heuristic evaluation uses a small set of evaluators who judge a user interface for compliance with usability design principles. Five evaluators are the recommended number for critical systems and no fewer than three evaluators for any heuristic review [42]. Researchers have shown that five participant evaluator is sufficient to find approximately 80% of the usability problems in a system [20].

	Advantages	Disadvantages
Heuristic evaluation	Identifies many more and serious problems Less expensive and less time-consuming It is intuitive and easy to motivate people to do it It does not require advance planning It can be used early in the development process Requires 3-5 evaluators	Requires UI expertise It sometimes identifies usability problems without providing direct suggestions for how to solve them
Usability testing	Identifies serious and recurring problems Avoid low-priority problems	Require UI expertise High cost Misses consistency problems
Guidelines	Identifies recurring and general problems Can be used by software developers	Misses some severe problems
Cognitive Walkthrough	Helps define users' goals and assumptions Can be performed by individuals or by groups	Need task definition methodology Tedious Misses general and recurring problems

Table 7.1: Advantages and disadvantages of some evaluation methods [25][36][42]

7.2 The evaluation process

To make an intuitive and user-friendly interface, the application was tested and evaluated by several users during the development of the application. We used a combination of the usability evaluation methods that are mentioned in the previous section.

The main advantage of these kinds of evaluations is the involvement of users during the design and development process. Results are based on seeing and finding out what aspects of the user interface cause problems for representative users. The downside is that user evaluation are extremely time consuming.

Before we started to develop the application, we made sketches/mock ups of the user interface. These were used to detect usability problems in the user interface. We conducted a test with graduated computer science students by using Cognitive Walkthrough (CW) method. CW is primarily suited to evaluate design before testing with users become feasible and as a supplement to user testing in situations in which users are difficult or expensive to recruit [23]. In this methodology, the developers of an interface walk through the interface in the context of core tasks a typical user will need to accomplish. The actions and feedback of the interface are compared to the user's goals and knowledge, and inconsistencies between the user's expectations and the steps required by the interface are noted [25]. Initially we followed the rules when carrying out such a method, but we realized later that we might overlook problems since we were only two. We believed it was better to try CW with more people who had similar backgrounds (in terms of education) as us. Therefore we persuaded some of our classmates who were about to graduate in computer science to perform the test. At this early stage we were focusing on to find out how easy it was for new users to accomplish tasks with the system. The procedure for CW consisted of two phases; a preparation phase and an execution phase. In the preparation phase, we chose some specific tasks to be evaluated and constructed a correct action sequence for each task. When this was done, the execution phase began. For each action in the action sequences we tried to find answer for the following four questions [23]:

- Will the user try to achieve the right effect?
- Will the user notice that the correct action is available?
- Will the user associate the correct action with the effect trying to be achieved?
- Will the user see that progress is being made toward solution of the task, if the correct action is performed?

During the execution of the test, we followed with the order to check whether the tasks they completed led to success or failure. In case of failure, we considered it as a usability error/problem. After completion of the test, we used the time for a brief discussion with test users to find out more about their thoughts on the interface and the way they completed the

tasks. This was also done to avoid misunderstanding and problems around the design of the application.

After the POC was done, we conducted another experiment where family members and friends who had no it/usability background to analyze the user interface empirically. The test sessions were administrated by both authors and consisted of an introduction to make the users familiar with the test situation, the actual test and a debriefing of the users. In the introduction, we taught the users to think out loud because this was an unnatural thing to do for the users, and experience indicates that without teaching – and some encouragement during the session – only few users are capable of giving valuable verbal reports about their work. The actual test was initiated by reading the first task out loud and handing it over to the users who solved it while thinking out loud. After finishing the first task, the second was presented in a similar manner, and so forth. When the users had completed all tasks, they were debriefed to provide any additional insight into the system. We noted the cases where they had problems to get thing done or found it difficult and used longer time than necessary to perform a desired task.

The two tests that were done made us realize what was important for the users. Furthermore, it gave us an idea of what problems the user interface had and what could have been done better. With users and their feedback in mind, the development of application started. Before the application was done, we got in touch with people from Adressavisen who were willing and interested in taking a look at the application and provide feedback. They were skilled UI professionals, with many years of experience in evaluating interfaces. We conducted a heuristic evaluation where they were given access to the application and were asked to write a report pointing out the usability problems in the user interface as precisely as possible. Ideally people would conduct such evaluations according to certain rules, such as those listed in typical guidelines documents. Some of those guidelines documents have many rules to follow and therefore seen as intimidating by developers. Most people probably perform heuristic evaluation on the basis of their own intuition and common sense instead. By doing so, the evaluators are free to say what they think about the application without complicating the whole process. We found out that the evaluators discovered some problems that which we had not originally identified ourselves.

Since the application was finalized quite late at the end of the project, it was difficult to recruit many test users during this period. But we managed to get five users to test and

compare the finalized application with two other popular news apps (Flipboard and News Republic). The test users did use the apps for a limited time, and then they received a questionnaire where they were asked to complete. The questionnaire contained ten specific/close-ended questions and two open-ended questions regarding the user interface and features of the application. A 5-point scale, with 1 being negative end of the scale (strongly disagree) and 5 being the positive end of the scale (strongly agree), was used to measure the test users' responses to elements in the questionnaire. These ratings were then analyzed to estimate the individual's intention to use the application. The last two open-ended questions were used to obtain the users' thought and ideas in case the first ten questions did not uncover the weaknesses/strengths of the application compared to the two other apps. The questionnaire can be found in Appendix A.

7.3 Evaluation results

This subsection is divided into two parts. While the first part deals with the findings and feedbacks of the tests that were performed during development of the application, the last part describes the last test result performed after the application was fully developed.

7.3.1 First phase

After conducting several tests, we have found that the design of a good user interface usually requires the use of various usability methods. Involvement of both experts in the field and potential users has made it possible to achieve a user interface with a good starting point which apparently is satisfying. This has also caused that the user interface has undergone an iterative process with changes along the way. The user tests gave us a lot of valuable information on what was missing or could be improved with the design. Most of the feedbacks were taken into consideration during the development of the application.

The evaluations of the application detected a variety of usability problems, most of which were related to the following heuristics developed by Jakob Nielsen (1995).

Aesthetic and minimalist design: Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Match between system and the real world: The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

Help users recognize, diagnose, and recover from errors: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

The majority of problems encountered were related to the *Aesthetics and minimalist design* heuristic. The news page's header, which is a carousel, took a substantial portion of the visible area of the page. Besides reducing the visible area which in many cases forced the users to scroll more than they would need to, the carousel seemed noisy for some users since it moved itself too quickly. The way we solved this was to allow the carousel go slower and made some more views which did not have carousel. By presenting the news articles in several ways gave the users opportunity to choose the view that they wanted.

Some of the pages had dialogue problems. As an example we can mention the login window. When the users tried to sign up and entered invalid values in the text fields, this was indicated in the application by showing the fields with a red background. Not all users did understand it. Because of this we implemented warning/alert messages to give users feedback on what was wrong. The users also got feedback or indication if the registration was successful.

When it came to *language problems*, there were dropdown menus with radio buttons that did not give a hint of what they stood for. Some of their names included words that were not commonly understood by users. For instance, "themes" and "views" were two words that were unknown for all most all users except those with it-backgrounds. Because of this, some users were not aware of that the news could be presented in different ways and that they had the ability to change colors and fonts on the page. As has been pointed out above by Nielsen, a system should speak the user's language [34]. By this he means that one must use words and phrases that are familiar to the user. And that is why we tried to change the name and phrases that users had no knowledge of or were familiar with.

Another problem that was detected during the evaluation was related to handling of error messages. Sometimes during the tests, we experienced that the server was down or that the users were not able to log into the application. In such cases, the users got messages that were written in code, something that did not make any sense to them. On the basis of this we created error messages that were expressed in plain language. Additionally, these error messages were indicated precisely what the problem was.

7.3.2 Last phase

This sub-section describes the results of the last user test. As it was mentioned earlier in this chapter, it was used a scale of 1-5 for responses. Because of this, one cannot assume that the users perceive the difference between adjacent levels as equidistant. Intentions, opinion and personal characteristics are all construct which are thought to vary in degree between individuals.

The aim of the first question was to determine if the app is user-friendly and simple to use. It got a score between 4 and 5. The users were agreed or strongly agreed that the SM application is easier to use than the other two apps. The next two questions which were about finding necessary information and to perform tasks in an easy way got a score of 4 (better compared to the other two apps). The fourth question which concerned if the application did everything that the user would expect it to do, got a lower score compared to the other two apps. The reason for this was that that the users had an expectation that the SM application should also give them the opportunity to log in with their Facebook or Google+ profile. This was something that was offered by the two other news apps. In addition, they thought it was silly that the SM application did not have a reset password feature in case they forgot or lost their password. News Republic received top score, followed by Flipboard and the SM app. Users found News Republic better since they could read news without having created an account/profile.

Regarding question number five, all users agreed that it required only few steps to achieve what they wanted to do with the apps. They thought that there no differences between the three apps. In question number six, the users were asked if they could use the apps without written instructions. In this case, the SM app got top score in comparison to the other two apps (which got a score of 4). They thought the other two apps had some features that they had no clue what it could do. Question number seven which was about the system's way of

handling errors apparently had a neutral response. Except one user, all users provided a score of 3 for all the apps. The one who gave a score of 4 to the SM app was because he failed to sign up in the first place. When he wrote an invalid password, he got error messages on what was causing the problem.

The last three questions dealt largely on whether the apps met users' needs and if it was comfortable to use them. All users agreed that all apps met their needs to a great extent and that they were pleasant to use.

During the last two (open-ended) questions, we discovered that users felt that the SM app was easier to understand and use. But they missed a few functionalities such as login via Facebook/Google+ and being able to reset their password. They meant the way the SM app presents new articles is exciting. And the fact that they have opportunity to change the way news stories are displayed, the font size and the background color is a plus.

7.4 Suggested improvements

During the evaluation phase, a few weaknesses with the initial design were identified. This section lists the enhancements that should be implemented in the next version of the Smart Media application.

The first thing that should be in place is a well-functioning API. Due to the API's limitations, the Smart Media application has some weaknesses that need to be improved. One of the shortcomings of the application is that it does not provide users ability to reset their password or get a new password in case of forgotten or lost password. This functionality is incredibly important to implement since it is common that people forget their passwords. Another useful functionality that is important to implement is to provide articles based on categories.

To gain a great value in a recommendation system, one should focus on providing readers with obvious ways forward from the article. The goal is to keep the users on the site furthest possible and that each user give the largest possible number of page views.

Another functionality that we have implemented, and which are not particularly relevant and useful at the present time, is showing the location of users when opening/reading an article.

Instead of displaying the users' locations, one should present the location of where the story took place. But based on users' locations, one can also offer users news stories that are related to their neighborhood.

Last but not least, to make the overall experience better for the end-user, the application should provide a set of unique features. These could be, for example to display statistics about users' reading habits (number of read articles, total time for reading articles) and statistics about articles (number of readers per article, average rating, rank articles based on top rated article). Additionally rank articles based on rating and provide personalized news articles to the user.

8 Conclusion and further work

This chapter begins by presenting the research questions, and then describes the result of the study in relation to them. The last section contains a discussion of further work.

8.1 Conclusion

During the project a news application was created by using web and hybrid approaches. The application has gone through different evaluation methods and has been tested with a small set of users. The three research questions that guided the work during the project were:

1. What characterizes the user interfaces of successful and user friendly mobile news apps?
2. How can existing frameworks and tools be used to reduce development and maintenance costs and ensure consistency across platforms and devices?
3. How should a news recommender system user interface be designed to minimize user interaction and support implicit user profiling?

To answer the first research question, we looked at and studied various applications to find out what made them successful and user-friendly. We found the following characteristics:

- They are intuitive and easy to use
- They do not require much explicit information from users
- They tailor news articles based on users' interests and preferences
- They allow users to share articles through social networks
- They allow users to store and sync articles across devices, making sure that the users have access to news articles whenever and wherever.

In addition, we discovered that most of these applications had followed usability principles and rules such as clear and uncluttered navigation, space between elements on the page, readable texts, two-second rule and three-click rule. Having tried the applications in a short period of time, we also figured out that Nielsen's five usability qualities seemed to be present for these applications. The five usability qualities described by Nielsen are: learnability (easy to use), efficiency (can perform tasks quickly), memorability (easy to remember), error-free or error-forgiving, and satisfaction (pleasant to use). We believe that user interfaces for

applications should meet these five usability components in order to be successful and user-friendly.

The second research question is about how existing framework and tools can be used to reduce maintenance costs and ensure consistency across platforms and devices. We tried several frameworks and tools, and concluded that PhoneGap as a tool to develop hybrid application are better suited to our project. By using technologies such as HTML5, CSS3 and JavaScript (in our case AngularJS) will lead to reduce the development time and maintenance costs compared to creating native applications. Although we were not familiar with these technologies and had not used these tools/frameworks before, we managed to create a responsive web application, and a hybrid version of it. The application can run and behave pretty much the same on all major platforms and devices. If we were using a native approach, we would probably have spent at least twice as long to develop an application that could run on different platforms since it requires different programming languages and development environments to achieve something similar.

It depends on what you want to achieve. If you want to reach more users and your application does not need high-end graphics or calculations, then many of the cross-platform tools provide a good basis to build an application that can run on different mobile platforms and devices. But if you want to make a game app or want the native look and feel, then the native approach is the way to go.

The last research question which concerns how the user interface for recommendation system should be designed to minimize interaction and support implicit user profiling, is hard to give an exact and precise answer to since the API prevented us to implement elements/functionalities that could help us with this. But it is possible that logging user information/events in the background and displaying relevant stuff on the client side can make things easier for the user. In this way, one can personalize and show content based on user-clicks/-history, interests and preferences. Something that we wanted to implement (which was not provided by the back-end API) was recommending articles based on users' reading habits or articles related to users' position. At the present time, we present on one of the views articles placed on the world map. This gives the users the option to read news concerning the various part of the world. We have designed the app in such way that a user can see read/visited articles. Furthermore, we present 50 articles in the first place. But once a user

scrolls down the page, the news page updates and several (50 more) articles will be retrieved without the user having to do anything.

Logged in users can store preferences such as article layout, color, font size, geo-location etc. At later occasion, signed on a different client, the user's preferences is still unchanged. For instance, if the user has turned the geo-location on, the new client will also have this on. This will prevent the user to go through the same process again.

During the tests we discovered that to minimize the interaction and support implicit user profiling is not particularly necessary as long as the user interface is intuitive and easy to use. As long as the users of the application understand and to some extent can control over what is being presented, then they are happy and satisfied with the application.

8.2 Further work

The application we developed and evaluated showed great potential. In a nutshell, we are pleased with the results of this research work and we believe that it will act as a solid foundation for future research.

Unfortunately, due to lack of time, we did not perform a major test for both test users and design experts after the application was completed. It would be interesting to examine if the users' ability to change the presentation of news articles, background color and text size has a positive effect on user experience.

It would also be interesting and useful if one could make an application using different approaches for measuring development time and costs for the various ways to build an application.

Another useful thing that is worth investigating is to create two completely different designs, and then test it with representative users to check out which one satisfies the test users' needs.

9 Bibliography

- [1] Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey to the State-of-the-Art and Possible Extensions. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 17, NO. 6, 734-749.
- [2] Allen, S., Graupera, V., & Lundrigan, L. (2010). *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution*. Apress.
- [3] Andersen, C. R., & Horvitz, E. (2002). Web Montage: A Dynamic Personalized Start Page. *Proceedings for the 11th International Conference on World Wide Web*, pp. 704-712.
- [4] AngularJS. (n.d.). Retrieved April 6, 2014, from <http://docs.angularjs.org/guide/introduction>
- [5] Appcelerator Titanium. (n.d.). Retrieved May 15, 2014, from <http://www.appcelerator.com/titanium/>
- [6] Ballard, P., & Moncur, M. (2008). *Sams Teach Yourself Ajax, JavaScript, and PHP All in One*. Sams Publishing.
- [7] Benckert van de Boel, A. (2011). Designing the future of the newspaper.
- [8] Bradely, S. (2012). Designing For A hierarchy Of Needs. In A. Maier, C. Chapman, D. Travis, D. Fadeyev, F. Inchauste, S. Bradely, & V. Friedman, *Psychology of Web Design* (pp. 33-52). Freiburg: Smashing Media GMBH.
- [9] Davis, F. D. (1989). Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, Vol. 13 No. 3, pp. 319-340.
- [10] Emberjs. (n.d.). Retrieved May 11, 2014, from <http://www.emberjs.com>
- [11] Fan, H., & Poole, M. S. (2006). What Is Personalization? Perspectives on the Design and Implementation of Personalization in Information Systems. *Journal of Organizational Computing and Electronic Commerce*, pp. 179-202.
- [12] Flanagan, D. (2011). *JavaScript: The Definitive Guide, 6th Edition*. O'Reilly Media.
- [13] Fling, B. (2009). *Mobile Design and Development*. O'Reilly Media.
- [14] Flipboard. (2010, December). *Flipboard*. Retrieved May 7, 2014, from <https://www.flipboard.com>
- [15] Freeman, A. (2011). *The Definitive Guide to HTML5*. Apress.
- [16] Friedman, V. (2012). 30 Usability Issues To be Aware Of. In A. Maier, C. Chapman, D. Travis, D. Fadeyev, F. Inchauste, S. Bradley, & V. Friedman, *Psychology of Web Design* (pp. 72-87). Freiburg: Smashing Media GMBH.

- [17] Gao, S., Krogstie, J., & Gransæther, P. A. (2008). Mobile Services Acceptance Model. *International Conference on Convergence and Hybrid Information Technology 2008*, pp. 446-453.
- [18] Green, B., & Seshadri. (2013). *AngularJS*. O'Reilly Media.
- [19] Gulla, J. A., Ingvaldsen, J. E., Fidjestøl, A. D., Nilsen, J. E., Haugen, K. R., & Su, X. (2013, September). Learning User Profiles in Mobile News Recommendation. *J. Print Media Technol. Res., Vol. II (No 3)*, pp. 125-214.
- [20] Hartson, H. R., Andre, T. S., & Williges, R. C. (2001). Criteria For Evaluating Usability Evaluation Methods. *International Journal of Human-Computer Interaction*, 13:4, pp. 373-410.
- [21] Haugen, K. R. (2013). *Mobile News: Design, User Experience and Recommendation*.
- [22] Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2013). Evaluating Cross-Platform Development Approaches for Mobile Applications. In J. Cordeiro, & K.-H. Krempels, *Web Information Systems and Technologies* (pp. 120-138). Springer Berlin Heidelberg.
- [23] Hertzum, M., & Jacobsen, N. E. (2001). The Evaluator Effect: A Chilling Fact About Usability Evaluation Methods. *International Journal of Human-Computer Interaction*, 13:4, pp. 421-443.
- [24] ISO 9241-11. (1998). Ergonomic requirements for office work with visual display terminals (VDTS) -- Part 11: Guidance on usability.
- [25] Jeffries, R., Miller, J. R., Wharton, C., & Uyeda, K. M. (1991). User interface evaluation in the real world: a comparison of four techniques. *Proceeding CHI '91 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 119-124.
- [26] Kamba, T., Bharat, K., & Albers, M. C. (1995). The Krakatoa Chronicle - An Interactive, Personalized, Newspaper on the Web. *Proceedings of the 4th International World Wide Web Conference*.
- [27] Kamran, S. (2013, January 31). *Safari Books Online*. Retrieved May 11, 2014, from <http://blog.safaribooksonline.com/2013/01/30/web-application-frameworks-for-javascript/>
- [28] LinkedIn. (n.d.). Retrieved May 8, 2014, from <http://www.linkedin.com/company/pulse-news>
- [29] Marshall, C. C. (2007). The Gray Lady Gets a New Dress: A field Study of the Times News Reader. *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, pp. 259-268.
- [30] Michaels, ross & cole, ltd. (mrc). (n.d.). *mrc-productivity*. Retrieved April 23, 2014, from <http://www.mrc-productivity.com/Research/whitepapers/NativeAppsWrongChoice.pdf>
- [31] Miller, G. A. (1956, Mars). The Magical Number Seven, Plus otr Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, Vol 63(2), p. 8197.
- [32] Nielsen, J. (1993, November). Iterative user-interface design. *IEEE Computer Vol. 26, No. 11*, pp. 32-41.

- [33] Nielsen, J. (1994, April). Usability Inspection Methods. *Proceeding CHI '94 Conference Companion on Human Factors in Computing Systems*, pp. 413-414.
- [34] Nielsen, J. (1995, January 1). *NN/g Nielsen Norman Group*. Retrieved from <http://www.nngroup.com/articles/ten-usability-heuristics/>
- [35] Nielsen, J. (1996, June 1). *Nielsen Norman Group*. Retrieved June 6, 2014, from <http://www.nngroup.com/articles/inverted-pyramids-in-cyberspace/>
- [36] Nielsen, J., & Molich, R. (1990, April). Heuristic evaluation of user interfaces. *Proceeding CHI '90 Proceeding of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 249-256.
- [37] Osmani, A. (2013). *Developing Backbone.js Applications*. O'Reilly Media.
- [38] PhoneGap. (n.d.). Retrieved May 13, 2014, from <http://www.phonegap.com>
- [39] PhoneGap Build. (n.d.). Retrieved May 15, 2014, from <http://phonegap.com/2011/10/13/phonegap-build-pricing-plans-update/>
- [40] Ricci, F. (2010). Mobile Recommender Systems. *Information Technology & Tourism*, 205-231.
- [41] Ricci, F., Lior, R., Shapira, B., & Kantor, P. (2011). *Recommender Systems Handbook*. Springer.
- [42] Scholtz, J. (2004). *Usability Evaluation*. National Institut of Standards adn Technology.
- [43] Shneiderman, B. (1994). Dynamic Queries For Visual Information Seeking. *IEEE Software* 11(6), pp. 70-77.
- [44] Shotts, K. (2013). *PhoneGap 2.x Mobile Application Development Hotshot*. Packt Publishing.
- [45] Skeie, J. H. (2014). *Ember.js IN {{ACTION}}*. Manning Publications.
- [46] Smashing Magazine. (2012). *Essentials Of Mobile Design*. Freiburg: Smashing Media GmbH.
- [47] Sonmez, J. (2013, July 1). *Simple Programmer*. Retrieved June 1, 2014, from <http://simpleprogrammer.com/2013/07/01/cross-platform-mobile-development/>
- [48] Tavakolifard, M., Gulla, J. A., Almeroth, K. C., Ingvaldsen, J. E., Nygreen, G., & Berg, E. (2013). Tailored News in the Palm of Your HAND: A Multi-Perspective Transparent Approach to News Recommendation.
- [49] Teitler, B. E., Lieberman, M. D., Panozzo, D., Sankaranarayanan, J., Samet, H., & Sperling, J. (2008). NewsStand: A New View on News. *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advance in Geographic Information Systems (GIS)*, pp. 18.1-18.10.
- [50] Thurman, N., & Schiffers, S. (2012). The Future Of Personalization At News Websites. *Journalism Studies*, pp. 1-18.
- [51] TodoMVC. (n.d.). Retrieved May 11, 2014, from <http://www.todomvc.com>

- [52] Vydiswaran, V., Eijkhof, J. v., Chandrasekar, R., Paradiso, A., & George, J. S. (2012, January 11). News Sync: Enabling Scenario-based News Exploration. *Proceedings of the American Society for Information Science and Technology*, pp. 1-10.
- [53] W3Schools. (n.d.). Retrieved Mars 29, 2014, from http://www.w3schools.com/html/html5_intro.asp
- [54] W3Schools. (n.d.). Retrieved Mars 30, 2014, from http://www.w3schools.com/css/css_intro.asp
- [55] Wikipedia. (n.d.). Retrieved Mars 29, 2014, from <http://en.wikipedia.org/wiki/HTML>
- [56] Wikipedia. (2014, May 7). *Wikipedia*. Retrieved June 5, 2014, from http://no.wikipedia.org/wiki/Maslows_behovspyramide
- [57] Yarow, J. (2010, December 10). *Buisnessinsider*. Retrieved May 8, 2014, from <http://www.businessinsider.com/apple-calls-flipboard-ipad-app-of-the-year-2010-12#!KrrbN>
- [58] ZenithOptimedia. (2013, February 11). *ZenithOptimedia*. Retrieved Mars 18, 2014, from <http://www.zenithoptimedia.com/zenithoptimedia-publishes-new-media-forecasts/>
- [59] Zite. (n.d.). Retrieved May 8, 2014, from <http://www.zite.com>

Appendix A

Questionnaire

	Strongly disagree	1	2	3	4	5	Strongly agree
1. It is user-friendly and simple to use.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
2. It makes the things I want to accomplish easier to get done.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
3. It is easy to find the information I needed.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
4. It does everything I would expect it to do.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
5. It requires the fewest steps possible to accomplish what I want to do with it.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
6. I can use it without written instructions.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
7. The system gives error messages that clearly tell me how to fix problems.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
8. It meets my needs.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
9. I am satisfied with it.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
10. It is pleasant to use.		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	

List the most negative aspect(s):

List the most positive aspect(s):