Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Odin Johan Vatne

Project Assignment Software

Specifications for an Improved Project Assignment Software for use by Universities with Larger Student Bodies

Master's thesis in Computer Science
Supervisor: Guttorm Sindre
July 2022

NTNU
Norwegian University of
Science and Technology

Odin Johan Vatne

Project Assignment Software

Specifications for an Improved Project Assignment Software for use by Universities with Larger Student Bodies

Master's thesis in Computer Science
Supervisor: Guttorm Sindre
July 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Each year, university students across the globe take on final year research projects to complete their scientific education. As class sizes grow, matching these students with appropriate project topics becomes increasingly difficult. In order to solve this problem, universities often turn to software systems to manage the allocation of projects to students. This paper examines one such system currently in use at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU).

This system is not meeting the needs of students or professors. It was implemented with limited resources for planning and design, and was not future-proofed for a larger student body or use by other departments. It relies on informal communications between students and professors for most processes, creating significant administrative work for both parties. The goal of this thesis is to produce and test a set of requirements for a new project assignment system which can reduce that load.

Unlike most research in the area of student-project assignment, this paper focuses on supporting manual matching at larger scales rather than replacing it with automated matching. The requirements and prototype were designed to help professors keep track of and evaluate a large number of students, and help students quickly find interesting projects from a large set of options.

This thesis first outlines a series of requirements for a new, more scalable system to alleviate the issues observed in the current one. It then documents the development of a prototype which implements some of those requirements. Finally, it evaluates the effectiveness of the prototype and requirements through usability testing and process analysis methods.

Based on our user testing, the prototype was very successful at improving the experience for students, but only moderately successful for professors. However, the feedback received from these tests implies that the requirements presented were sound, even though our implementation fell short.

Sammendrag

Hvert år gjennomfører universitetsstudenter rundt verden avsluttende forskningsprosjekter. Etter hvert som antallet studenter øker, blir det stadig vanskeligere å fordele prosjekter blant studenter etter ønske. Universiteter ser derfor i økende grad etter programvaresystemer som kan støtte denne prosessen. Denne masteroppgaven ser på ett slikt system i bruk av Instituttet for Datateknologi og Informatikk (IDI) ved Norges Teknisk-Naturvitenskapelige Universitet (NTNU).

Dagens system er ikke tilstrekkelig for studentene og veilederne sine behov. Systemet ble utviklet med begrensede ressurser for å håndtere akutte problemer, og er ikke tilpasset et økende antall oppgaver eller bruk av andre departementer og universiteter. Det baserer seg på uformell dialog mellom studenter og veiledere, som fører til betydelig administrativt arbeid rundt epost-tråder og møtetidspunkt for begge parter. Målet med denne masteroppgaven er å utvikle en kravspesifikasjon for et oppgavesystem som kan redusere dette arbeidet, og evaluere kravene gjennom en prototype.

I motsetning til mye annen forskning angående prosjekttildeling, fokuserer denne oppgaven på støtte for manuell tildeling av prosjektønsker på større skala, i stedet for automatisering av denne prosessen. Kravene og prototypen var utformet for å hjelpe veiledere å håndtere og vurdere et stort antall studenter, og for å hjelpe studenter kjapt finne interessante oppgaver blant et stort volum av forslag.

Oppgaven presenterer først en kravspesifikasjon for et bedre skalerbart system som motvirker disse problemene. Deretter dokumenterer den utviklingen av en prototype basert på disse spesifikasjonene. Til sist evalueres prototypen og kravene gjennom brukertesting og prosessanalyse.

Basert på brukertestingen, forbedret prototypen brukeropplevelsen for studenter i stor grad, men lyktes bare delvis for veiledere. Tilbakemeldingene vi samlet gjennom brukertestingen indikerer at kravene vi utredet utgjør en forbedring overfor det tidligere systemet, til tross for at prototypen ikke dekket alle kravene.

Preface

This thesis is the capstone of my five year Master of Technology in Computer Science at the Norwegian University of Science and Technology. The thesis was worked on throughout the spring of 2022, and expands on a project conducted in the fall of 2021.

I would like to thank my supervisor, Guttorm Sindre at the Department of Computer Science, for his advice and feedback throughout this project. I would also like to thank Joakim Danielsen Petersen for his partnership in the project of 2021. Finally, thank you to Brendan Walsh. Without his support I would not have made it through the last years of my degree.

Trondheim, July 2022

Odin Johan Vatne

Contents

Abstract . . . . . **iii**
Sammendrag . . . . . **v**
Preface . . . . . **vii**
Contents . . . . . **ix**
Figures . . . . . **xiii**
Tables . . . . . **xv**
1 Introduction **1**
1.1 Motivation 1
1.2 Previous Work 2
1.3 Research Questions 2
1.4 Scope 3
1.5 Contribution 3
1.6 Report outline 4
2 Background **5**
2.1 Master's Theses and Student-Advisor Matching 5
2.2 Existing IDI System 6
2.2.1 Features 6
2.2.2 Project Lifecycle 10
2.2.3 Issues and Omissions 12
3 Related Works **15**
3.1 Classifying Project Assignment Systems 15
3.1.1 Two-Way Negotiation Systems 16
3.1.2 One-Way Preference Systems 16
3.1.3 Two-Way Preference Systems 17
3.1.4 Student-Lead Systems 17
3.2 Existing Project Assignment Systems 18
3.2.1 Coordination of Student Project Allocation (2001) 18
3.2.2 A dynamic project allocation algorithm for a distributed expert system (2004) 19
3.2.3 Preference Based Final Year Project Title Selection System (2009) 20
3.2.4 Additional Works 20
3.3 Outcome of Review 21
4 Methods **23**

4.1 Development Methodology . . . 23
4.2 Interviews . . . 24
4.3 Task Backlog . . . 25
4.4 Time-to-Task Evaluation . . . 25
4.5 Keystroke-Level Modeling System . . . 25
4.6 User Evaluation . . . 27
4.7 Process Modeling . . . 28
5 Software Requirements and Specs . . . 31
5.1 Goals . . . 31
5.2 Non-goals . . . 32
5.3 Functional Requirements . . . 33
5.3.1 Project Lifecycle . . . 34
5.3.2 Application Lifecycle . . . 35
5.3.3 Student Views . . . 36
5.3.4 Professor Views . . . 37
5.3.5 Administrative Views . . . 38
5.3.6 Automatic Processing . . . 39
5.3.7 Notifications . . . 40
5.3.8 Filtering . . . 40
5.3.9 Thesis Proposals . . . 41
5.3.10 Exportable Data . . . 42
6 Implementation . . . 43
6.1 Choice of Technology . . . 43
6.1.1 Web vs. Desktop vs. Mobile . . . 43
6.1.2 Programming Language Choice . . . 45
6.1.3 Framework Choice . . . 45
6.2 Personas and Scenarios . . . 46
6.2.1 Professor Personas . . . 47
6.2.2 Student Personas . . . 47
6.3 Scenarios . . . 49
6.4 Revised Requirements . . . 51
6.4.1 Project Lifecycle . . . 52
6.4.2 Application Lifecycle . . . 53
6.4.3 Student Views . . . 54
6.4.4 Professor Views . . . 55
6.4.5 Automatic Processing . . . 56
6.4.6 Filtering . . . 56
6.4.7 Thesis Proposals . . . 57
6.5 Design Decisions . . . 58
6.5.1 Group Applications . . . 58
6.5.2 AJAX . . . 59
6.5.3 Profiles . . . 59
6.5.4 Automation . . . 60
6.5.5 Tags . . . 61

6.5.6 Filter Totals . . . 63
6.5.7 Ranking Interface . . . 63
6.6 Testing and Deployment . . . 64
6.6.1 Test Data . . . 64
6.6.2 Deployment . . . 64
6.6.3 Test Accounts . . . 64
7 Results . . . 67
7.1 Software Walkthrough . . . 67
7.2 Data Models . . . 77
7.3 Process models . . . 78
7.3.1 Project Search . . . 79
7.3.2 Project Listing . . . 79
7.3.3 Application Lifecycle . . . 80
7.3.4 Full Matching Process . . . 80
7.4 Time-to-Task . . . 93
7.4.1 Project Creation . . . 93
7.4.2 Finding a Project . . . 95
8 User Evaluation . . . 97
8.1 Student Responses . . . 97
8.1.1 Pre-test Questions . . . 97
8.1.2 Task Questions . . . 100
8.1.3 Post-test Questions . . . 109
8.2 Professor Responses . . . 112
8.2.1 Pre-test Questions . . . 112
8.2.2 Task Questions . . . 114
8.2.3 Post-test Questions . . . 122
9 Discussion . . . 127
9.1 Success of the Prototype . . . 127
9.1.1 Evaluation Criteria and Priorities . . . 127
9.1.2 Improvement over manual two-way negotiation . . . 129
9.2 Success of the Requirements . . . 130
9.3 Bias and Evaluation Problems . . . 131
10 Conclusion . . . 133
10.1 Future Work . . . 134
10.1.1 Unimplemented Requirements . . . 134
10.1.2 Impact of Popularity Metrics . . . 134
10.1.3 Student Selection System . . . 135
10.1.4 Tag Set . . . 135
Bibliography . . . 137
A User Evaluation Questions . . . 141
A.1 Student Questionnaire . . . 141
A.2 Professor Questionnaire . . . 145

Figures

2.1 The project listing page of the IDI system 7
2.2 The project ranking tool of the IDI system 8
2.3 The Administrator view of projects in the IDI system 9
2.4 The project creation page in the IDI system 10
2.5 A project's details page in the IDI system 11
2.6 The main page for professors in the IDI system 12
2.7 The archive section for professors in the IDI system 13

4.1 A legend of common BPMN elements 29

7.1 The main page of our prototype 69
7.2 The project creation page of our prototype 70
7.3 The project details page of our prototype 71
7.4 The application creation page of our prototype 72
7.5 The student applications page of our prototype 73
7.6 The professor applications page of our prototype 74
7.7 The professor projects page of our prototype 75
7.8 The profile page of our prototype 76
7.9 An entity relationship diagram of our database model. 77
7.10 Process model for finding a project without a system 82
7.11 Process model for finding a project with the IDI system 83
7.12 Process model for finding a project in our prototype 84
7.13 Process model for listing a project without a system 85
7.14 Process model for listing a project in either system 86
7.15 Process model for the manual application lifecycle 87
7.16 Process model for the IDI system application lifecycle 88
7.17 Process model for the prototype's application lifecycle 89
7.18 Model of the full matching process without a system 90
7.19 Model of the full matching process in the IDI system 91
7.20 Model of the full matching process in our prototype 92

8.1 Student usability rating of the IDI system 98
8.2 Number of projects each student applied for 99
8.3 Difficulty rating of our filtering system 102

8.4 Student test difficulty rating 110
8.5 Student usability rating for the prototype 110
8.6 Professor familiarity with the IDI system 112
8.7 Readability of the applications page 117
8.8 Professor test difficulty rating 123
8.9 Professor usability rating of the prototype 123

Tables

5.1 Functional requirements regarding project lifecycle. 34
5.2 Functional requirements regarding application lifecycle. 35
5.3 Functional requirements regarding student views. 36
5.4 Functional requirements regarding professor views. 37
5.5 Functional requirements regarding administrative views. 38
5.6 Functional requirements regarding automatic processing. 39
5.7 Functional requirements regarding notifications. 40
5.8 Functional requirements regarding filtering. 40
5.9 Functional requirements regarding thesis proposals. 41
5.10 Functional requirements regarding exportable data. 42

6.1 Revised requirements regarding project lifecycle. 52
6.2 Revised requirements regarding application lifecycle. 53
6.3 Revised requirements regarding student views. 54
6.4 Revised requirements regarding professor views. 55
6.5 Revised requirements regarding automatic processing. 56
6.6 Revised requirements regarding filtering. 56
6.7 Revised requirements regarding thesis proposals. 57

7.1 Approximate task time to create a new project with the IDI system . 93
7.2 Approximate task time to create a new project with our prototype . 94
7.3 Approximate task time to find a project with the IDI system 95
7.4 Approximate task time to find a project with our prototype 96

8.1 Student response about the large number of projects 98
8.2 Student responses about profiles saving them time 100
8.3 Student responses to questions about the first filtering task 102
8.4 Student responses to questions about the project application tasks . 104
8.5 Student responses about combining filters 106
8.6 Student responses about creating group applications 106
8.7 Student responses to questions about the application ranking task . 107
8.8 Student responses on whether they were more comfortable contacting professors via our prototype 108
8.9 Student responses to questions about accepting an offer 109

Chapter 1

Introduction

1.1 Motivation

Each year, university students across the globe embark on final year research projects to complete their scientific education. Students must find a topic to study, seek out a professor to act as their advisor, and form an agreement with that professor to develop the research. The student and professor then spend the next year working closely on delivering a thesis paper which contributes some new knowledge, process, or object to their field of study. Success in these projects is crucial for students' academic development[1]. It is also necessary for their degree progress; without a passing project, students may be unable to graduate. Additionally, the content of the final year of each student's education is dependent upon their choice of topic. Therefore the selection of this professor and project is deeply important for both the academic success of the student and the research output of the university.

Although this process may vary by university, we expect the problems they face to be similar in nature. The process of selecting a project requires frequent communication between students and professors, producing a large amount of work for both parties as they write and respond to applications. Professors need to keep track of which students they have spoken to, what projects they have already assigned out, and what administrative formalities need to be cleared to finalize each student-project pairing. Students have to sift through an extensive list of projects and sponsors to find ones applicable to their skills and interests, write applications for each project, and contend with invisible deadlines as other students reach out first and take projects out of circulation. Alternatively, students may suggest their own project topics, which requires finding professors with relevant experience and interests, and proposing their project to them. Out of the many projects a student may apply to, they will ultimately only pursue one, and each professor will only select a handful of students from the many that approach them.

As universities have grown, the student-project matching problem has only become more difficult. Reasons for this include the growing number of project

proposals and potential supervisors students have to familiarize themselves with in order to make informed decisions, and the increasing administrative load of managing such a large number of people and agreements. Enacting the matching process manually at these larger scales is prohibitively time-consuming and error-prone. As a result, many universities have implemented tools to facilitate this matching process[2]. These tools typically allow professors to create and edit project listings for students to browse, and link in with the school administrative systems to allow professors to assign students to projects.

None of these tools have found wide adoption. Individual universities or departments tend to develop their own systems which address only a subset of their most painful problems, then do not iterate on them due to a lack of resources and time. As a result, these systems make large-scale student-project assignment possible, yet fail to make it easy or efficient. In this project we will make a case study of one such system, currently in use at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU). By designing and testing an improved version of this system, we aim to develop a better understanding of the shortcomings of this type of tool and how to overcome them. We will provide a set of requirements for an improved manual matching system and a data-driven analysis of their effectiveness. The resulting knowledge will help other universities improve their own student-professor matching tools or implement new ones with ease.

1.2 Previous Work

The development of this software is a continuation of a project previously carried out by the researcher along with Joakim Danielsen Petersen under the supervision of Guttorm Sindre. The initial project was focused on interviewing professors and administrators from NTNU and other schools about the systems currently in place at their universities for student-project matching. Based on these interviews and the complaints each subject had about their current systems, a general list of requirements for student-project matching systems was devised. The project included a prototype of several of these requirements, which was made public on a GitLab site hosted by IDI. This project builds upon that previous prototype, expanding its capabilities so that its functionality can be analyzed and compared to other systems in this space. Several of our software requirements are also based on those interviews.

1.3 Research Questions

This section lists the research questions that we attempt to answer in this thesis. Two questions were defined:

Research Question 1: How can one design and develop a system for effective pairing of master students with thesis projects for universities?

Research Question 2: How does the software created for this paper compare to existing solutions in terms of usability, user satisfaction, and features available?

1.4 Scope

The aim of this project is to show that there exists a generic set of requirements which can be used to develop effective student-project matching systems without automated matching, that is, where all decisions are made manually. To this end, we will produce a prototype software which implements these requirements and then test it in comparison to IDI's matching system, which we take as representative of manual project matching systems in general. We will gather data on whether the improvements made result in a better user experience and a more effective tool. It is not the aim of this project to produce a fully-integrated software package ready to be used widely within NTNU or elsewhere. We have chosen to focus on demonstrating that the proposed requirements used to develop our application provide a meaningful improvement over those used for the existing IDI system, and argue that they should be integrated into any future systems put in use in this area.

1.5 Contribution

The contribution of this master's thesis is a set of functional and non-functional requirements for a manual student-project allocation system, along with an analysis of the effectiveness of that set of requirements in improving the user experience over a baseline implementation. These requirements can then be used by any university or team building a student-professor matching system in order to produce a higher-quality system. Ultimately, this project attempts to contribute to a better understanding of the challenges and solutions of this type of system. This project will also produce a web application which implements these requirements, the source code of which can be found at:
`https://gitlab.stud.idi.ntnu.no/odinjv/project-assignment-software`.
This software is published under the MIT open source license, to enable free use of the code in future work.

Additionally, this report presents findings from interviews with professors and students about the process of matching students with thesis projects. This information is made available in the interest of supporting future research and projects in this area.

1.6 Report outline

This report is structured as follows:

- Chapter 1 introduces the concept and motivation of the project.
- Chapter 2 gives an overview of the existing IDI system, which we will use throughout the report as a baseline implementation for comparison.
- Chapter 3 provides an overview of previous research into project assignment systems in order to provide context for this paper's contribution to the field.
- Chapter 4 describes the research, development, and analysis methodologies used to carry out the project.
- Chapter 5 lists the software requirements generated to guide development, and the goals we oriented them around.
- Chapter 6 details the process of implementing the system and the specific choices we made while doing so.
- Chapter 7 shows the results of the implementation and our analysis, including screenshots of the delivered software.
- Chapter 8 delivers the results of our user testing, and our interpretation of those results.
- Chapter 9 discusses the results of the system and the overall results of the project.
- Chapter 10 concludes this report and suggests future work.

Chapter 2

Background

This chapter introduces the IDI project matching system, which we use as a baseline reference implementation of a manual matching system throughout this paper. The current state of the system is described, along with analysis of its features and shortcomings based on interviews with school administrators and professors.

2.1 Master's Theses and Student-Advisor Matching

In order to graduate with a master's degree at NTNU, students must complete a one- or two-semester master's thesis under the guidance of a professor. The subjects of these projects are chosen by professors and then listed somewhere for students to browse and consider. Generally, students do not propose their own project topics, choosing instead from the options listed by their professors. Much of the student's final year of study is dedicated to pursuing this project, with the full final semester focused exclusively on performing the research, development, and writing necessary to produce a completed report. Upon delivery of an acceptable report, the students are then able to graduate with a master's degree in their field of study. This process is common at universities around the world which offer master's degrees[1].

Because the project chosen by each student defines their entire final year of education, it is very important that the process for choosing a project be as clear as possible. Mistakes and misunderstandings could lead to students running out of options and being required to work on a project which they are not qualified for or interested in, in order to graduate. This can result in lower-quality research and cause tension between students and their advisors[3]. Professors would prefer to work on research with students who are motivated by the project they have proposed, and school administrators would like to reduce the number of students with unexpected assignment issues as much as possible[4]. These are the stakes for the student-project matching process.

Without using any tools to facilitate project selection, the matching process can be generalized as follows:

1. Professors provide a set of project ideas to students in some form.
2. Students choose interesting projects and apply for them, or propose their own.
3. Professors respond to students' applications, accepting and rejecting them over the course of the matching period.
4. Professors and their accepted students finalize their choices and formalize a contract for the next year's research.

Some universities have systems in place to facilitate these steps to varying degrees. One such system is IDI's project assignment system, described below. For the purposes of this paper, we will use the IDI system as a representative reference of a project matching system.

2.2 Existing IDI System

NTNU's Department of Computer Science (IDI) has developed a web application to assist in the student-project matching process. This system was developed because the administrative load of managing the assignment of specialization projects manually grew too large for the department to handle as it expanded. To better understand this existing solution, the researchers previously reached out to faculty members at IDI to get an overview of how the system works, what functionality it covers, and what it fails to manage. This section will provide a description of that system as a point of comparison for our own, and outline some of the issues listed by its current users.

2.2.1 Features

The IDI system targets steps 1 and 4 in the matching process described above, assisting professors in listing projects for students to see and finalizing their choices in the school's academic records. It allows professors to create any number of projects which are added to a large central listing for students to browse. These projects can be duplicated from previous projects created by the professors, an important tool for ongoing research topics. Each project is assigned to at least one specialization from any of three programmes, which categorizes it in the internal records system. Projects also specify whether they are appropriate for a single student or a group of two, a title, and a description explaining the project's goals and academic requirements. The system also automatically archives projects eight months after their last activity, though they remain available for professors to reuse.

The project listing available to students, seen in Figure 2.1, displays all active projects created by professors. This list can be filtered by specialization and professor, allowing students to browse for interesting projects and find professors to contact about working together on their theses. The listing can also be sorted alphabetically by either title or professor name. Students can register interest in up to five projects, and remove them or reorder them by preference using the ranking

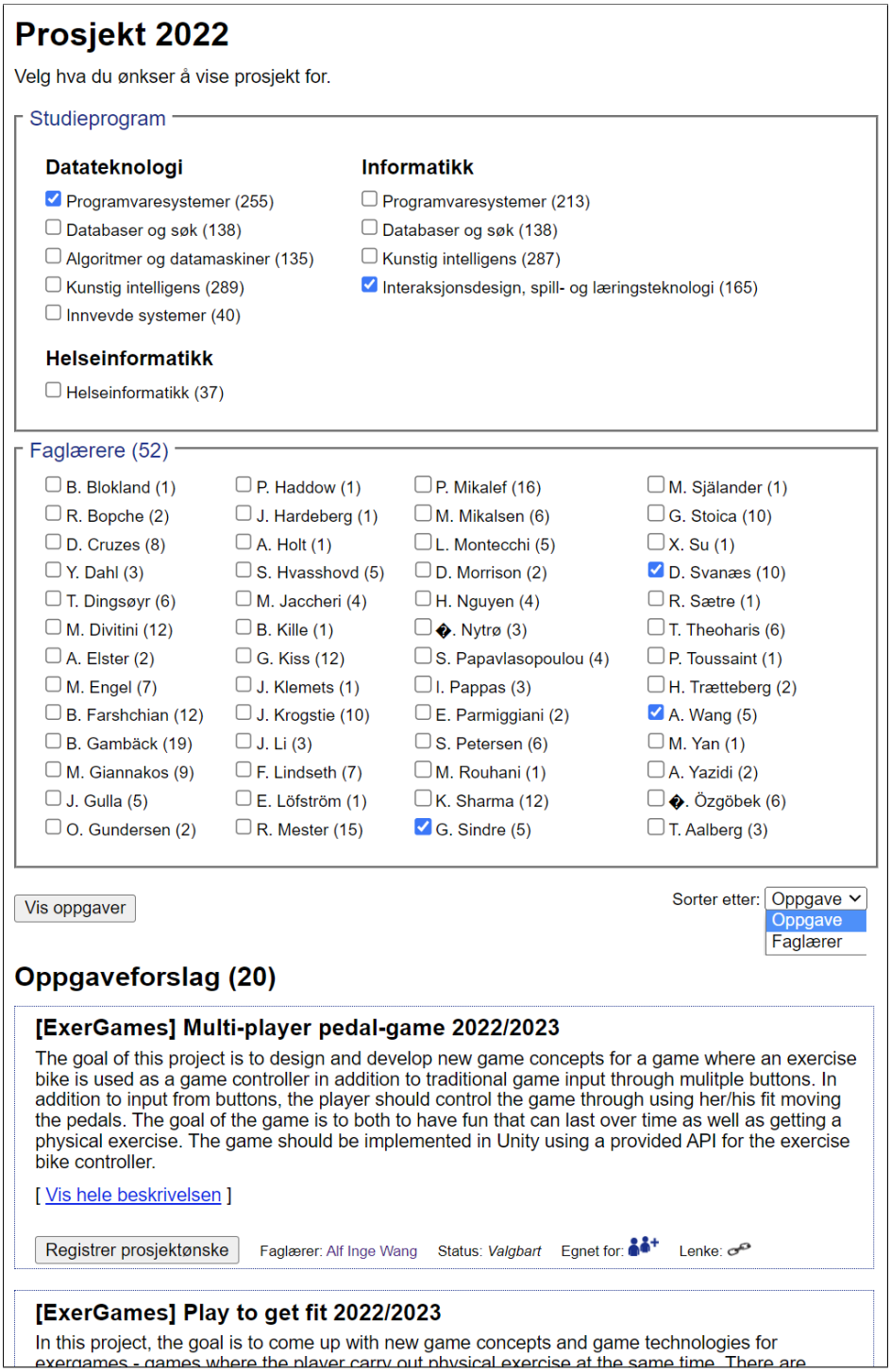

Figure 2.1: The project listing page of the IDI system. Projects can be filtered by professor and specialization.

Figure 2.2: The project ranking tool of the IDI system

tool in Figure 2.2. In turn, professors can see the rank each student assigned to their projects, as shown in Figure 2.3. When attempting to register interest the first time, students are prompted to log in via Felles Studentsystem (FS)[1], a nationally shared university login system used for all university services, to view this page.

After a student and a professor have agreed to work on a project together, the professor can extend an offer to the student within the system, given that the student has the project in their ranked list. If the project they decided on does not have a listing, as is the case for projects proposed by students, then the professor must create a new listing for it. New project listings can be added at any time, and will show up in the system immediately. Once the listing has been made, the student must find it and add it to their list. Only then can the professor extend an offer to the student. The student may at any time have multiple of these offers extended. The student can then choose to accept a single offer, which finalizes the decision.

The IDI system also has a number of useful tools for school administrators. It produces tables showing which students have not yet been assigned to a project, which projects are active and fully assigned, and which projects have been offered but are pending confirmation from the student. A section of this can be seen in Figure 2.3. These features allow administrators to gain an understanding of how well the process is proceeding, which enables them to make decisions about sending out broad reminder emails or checking in on specific students. The existing system also allows administrators to view any page as if they were any particular professor, which can help with resolving problems with listings and assignments.

Additionally, the IDI system handles "theory modules" alongside projects. Theory modules are specialized courses taken by students in the first semester of their final year, typically in connection with the subject of their master project. After a student has accepted a project offer, the web application will redirect to a page for selecting theory modules. The student can select these freely from two drop-down menus, though they are asked to discuss this with their advising professor, and many modules may have limited slots and require the approval of the teaching professor as well. The system allows for exporting the data of which students

[1] https://www.fellesstudentsystem.no/

In Progress (31)

Both supervisor and student have to accepted a project to move it to the list above.

Supervisor	Project	Students
[redacted]	ExAct: Tangible Interactive Technology for Physical Rehabilitation and Therapy	[redacted] - Offered [redacted] - Offered
[redacted]	Increasing awareness about privacy and personal data with games	[redacted] - Offered [redacted] ④
[redacted]	Network-Based Pandemic Modelling	[redacted] - Offered
[redacted]	Communication avoiding Conjugate Gradient algorithm programming on GPUs using CUDA	[redacted] - Offered
[redacted]	Creating a microkernel-based multi-server operating system Exploring Infrastructure Solutions for Edge Computing	[redacted] ① [redacted] - Offered
[redacted]	Adaptive Teaching Technologies to Create and Monitor Learning Activities	[redacted] - Offered [redacted] - Offered
[redacted]	VRP with trucks and drones for Health Service package delivery	[redacted] - Offered
[redacted]	Algorithmic problems	[redacted] - Offered
[redacted]	Manuell oppfølging av sau på beite Sau og rovdyr Sporing av sau ved hjelp av enkel radioteknologi	[redacted] ① [redacted] - Offered [redacted] - Offered [redacted] ①
[redacted]	AI-agents trained by Deep RL in simulated environments NAP-lab, Autonomous Vehicles (AVs) and Autonomous Driving (AD)	[redacted] ④ [redacted] - Offered
[redacted]	Bio-Inspired Artificial Intelligence for Computer Games	[redacted] ②
[redacted]	Building a simulation framework for autonomous driving	[redacted] ④
[redacted]	Blockchain-based Data Marketplace	[redacted] ③

Figure 2.3: A section of the administrator view of project statuses in the IDI system

are enrolled in which modules, for administrators to use when determining exam dates, to avoid scheduling collisions. For the purposes of this project, features regarding theory modules are considered out of scope.

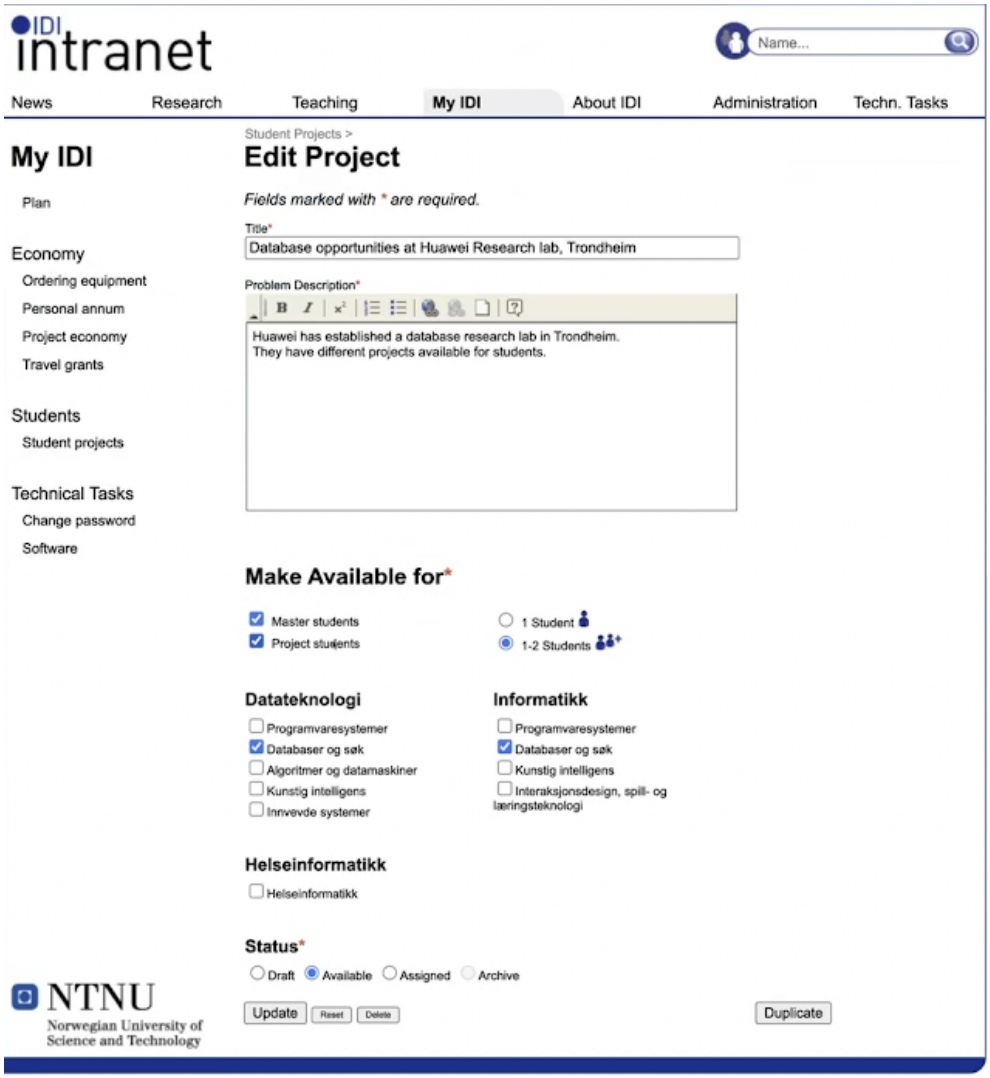

Figure 2.4: The project creation page in the IDI system

2.2.2 Project Lifecycle

This section describes the typical process a project goes through in the IDI system.

First, a new project is created by a professor, either from scratch or by cloning and editing an older archived project, via the interface in Figure 2.4. This may be published and added to the project listing, or saved in a draft state which prevents students from seeing it. Once a project is published, students can find it via the project listing shown in Figure 2.1. The system includes a details page for each

project as pictured in Figure 2.5, though it does not expose any new information that is not accessible to students from the main listing page.

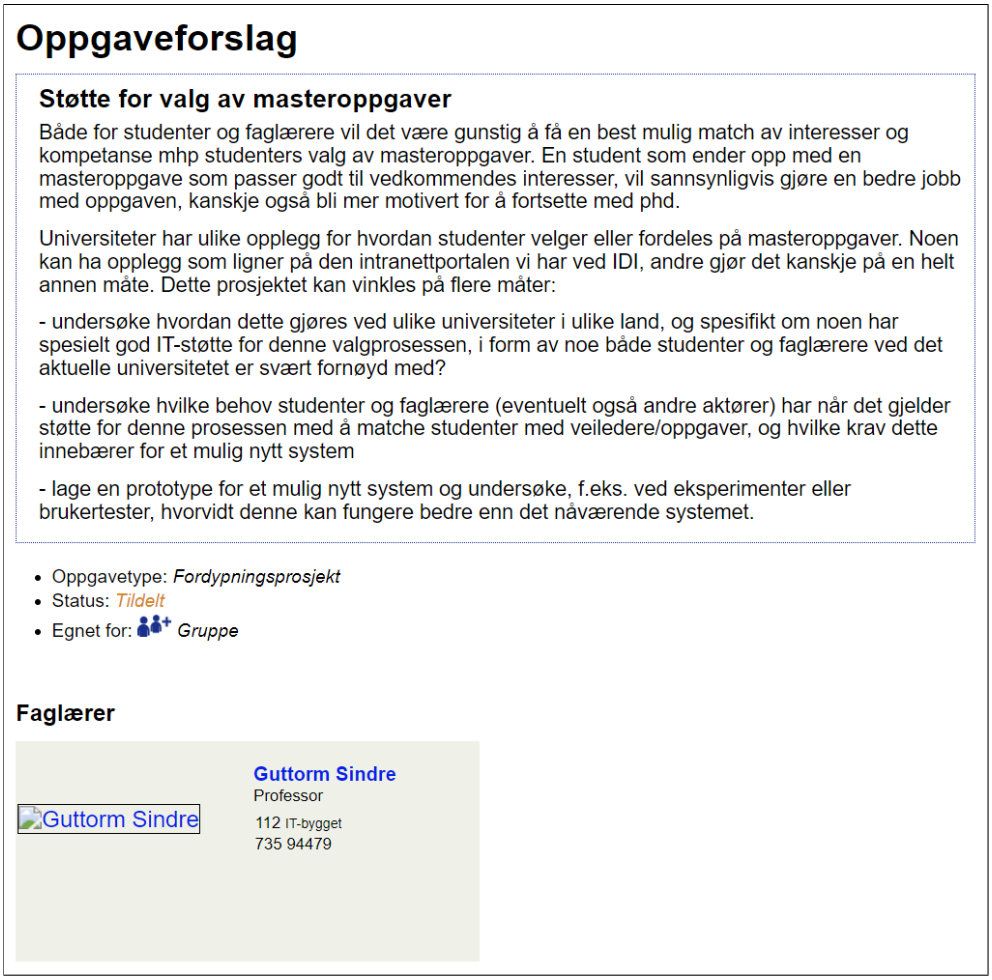

Figure 2.5: A project's details page in the IDI system

Formally, the only remaining step in the project's lifecycle is its assignment to a specific student. Informally, students must indicate their interest in the project somehow in order to get themselves assigned to it. One mechanism for this is the IDI system's ranking interface, which can be seen in Figure 2.2. Professors can see what rank each student has assigned their project, which can help them find and compare interested students. This is done on the main professor page shown in Figure 2.6. However, these ranks offer no further information about the student attached to them, so professors generally expect students to send an email application or set up an in-person meeting to discuss the project topic and the student's qualifications. This can be considered a part of the process, even though it is not an officially enforced policy or a part of the system. Following this, the professor assigns the project to the students they've chosen within the IDI system.

Once the students confirm the assignment, the project selection is recorded in the school's academic records system, and the process concludes. After eight months of inactivity with no further edits, the project is archived, at which point professors can access their archived projects via the interface shown in Figure 2.7.

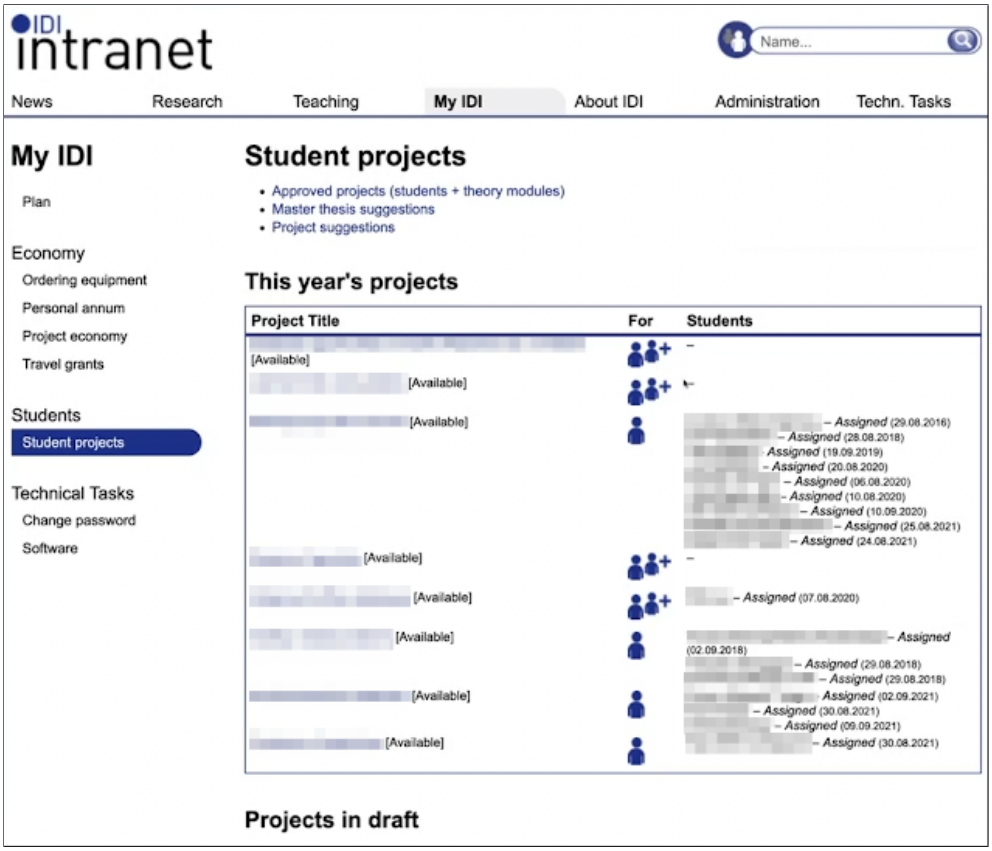

Figure 2.6: The main page for professors in the IDI system

2.2.3 Issues and Omissions

This section summarizes our findings on the failure points of the current system. These observations were derived from interviews we conducted with professors, administrators, and students, as well as our analysis of other project matching systems. These interviews were conducted according to the requirements elicitation methodology laid out in section 4.2.

Firstly, we note that the existing IDI system does not assist with step 3 of the matching process described in section 2.1. For step 2, although it generates a project listing for students to look at, it offers only limited tools to sort through the ever-growing number of projects in the system. Additionally, there is no way for students to act upon their project rankings within the system. As described in subsection 2.2.2, students can rank their top five projects, but they cannot apply for

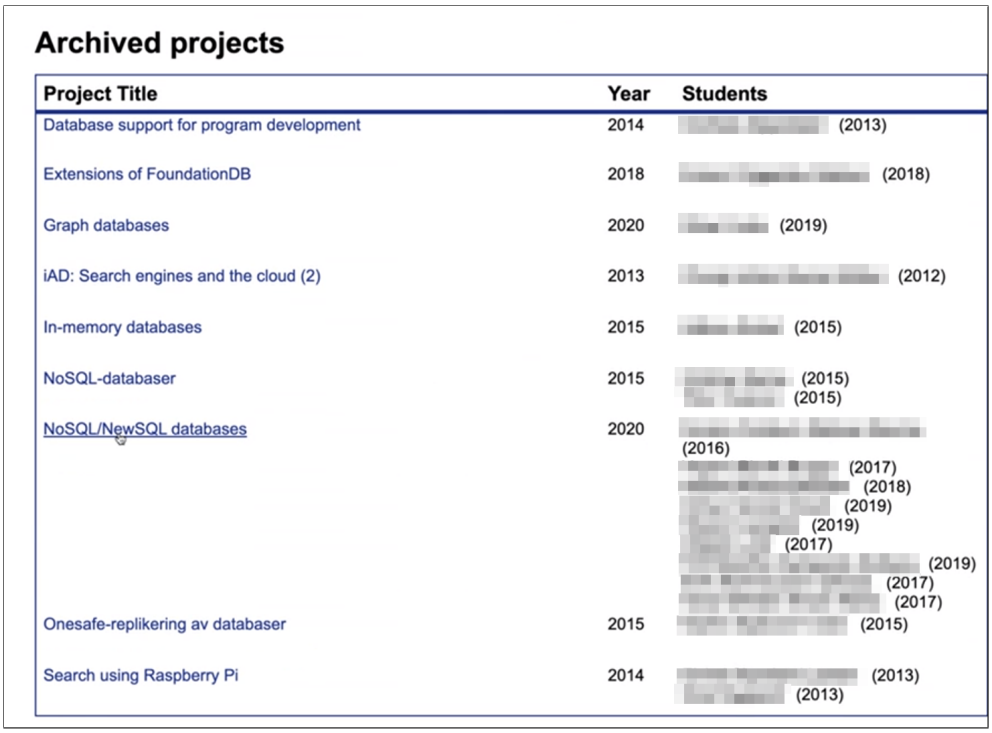

Figure 2.7: The archive section for professors in the IDI system

projects or ask professors questions through the system. Additionally, professors cannot see any information about students within the system, making it impossible for them to evaluate whether a student has the academic foundation necessary to work on the project without setting up a meeting. This means that professors are unable to act on rankings alone, necessitating additional work outside of the system.

As mentioned in subsection 2.2.2, professors expect that any student who ranks their project will also email them to discuss it further. Without such an email, most professors will not extend an offer, as they cannot assess how well the student fits their project. This expectation is not communicated explicitly to students, and its informal nature was stressful for many students who were not sure how many professors to contact or what to say. Because these applications occur over email, they are independent from the state of the student's rankings and the professor's project. This leads to professors wasting their time following up on applications from students who are no longer available, correspondence being lost in crowded inboxes, and both parties losing track of who is responsible for the next step.

There is also no formal process for students to propose a project within the IDI system. Students who want to work on their own project idea must find an advisor with no direction or assistance from the system. This requires them to either have prior familiarity with many of their department's professors, or read

through the public profiles of dozens of professors to figure out who may have an interest in their proposal. This is a problem that is particularly exacerbated as departments grow in size and more staff are hired. Because student proposals cannot be handled within the system, they too must happen over email, and are therefore subject to many of the same issues as applications.

The use of email in these cases means that information pertaining to application or proposal status is unavailable to the system. Professors cannot see a list of all students who have applied to each of their projects. Administrators have no idea what projects students have applied to and what their application status is. Students have no way to see how popular a project or gauge if its likely to be at capacity, and have to manually email each professor to retract their other applications after choosing a project.

Chapter 3

Related Works

In this chapter, we summarize the results of our literature survey. We searched for published works related to project allocation, final year projects, and student-project matching. This literature review is not intended to be a comprehensive overview of the entire project allocation problem, as that is not the goal of this paper.

3.1 Classifying Project Assignment Systems

As discussed in chapter 1, the concept of a software-driven system for assigning final year projects is not novel. A 2022 literature review performed by *Mubarak Ali et al.* found papers on the subject published as early as 2003[5], though our research turned up several papers even earlier than that. That literature review also found that the number of publications dealing with project assignment systems is trending upwards, implying that this topic is of increasing interest to the research community. The authors of the review do not attempt to attribute this increase to any particular factor, but we believe it is related to the issues caused by growing student bodies as described in our introduction.

In their 2019 review of project assignment methods for transnational engineering programs, *Hussain et al.* offered a comprehensive view of the issues and desires currently shaping the development of project assignment systems, as well as a scheme for categorizing existing assignment systems and a case study of two specific project allocation systems currently in use[2]. They identify the growing number of students in engineering programs as a major issue for project assignment systems, an assertion in line with our own observations. Specifically, they focus on the workload that matching large numbers of students with projects imposes on university faculty. The authors then enumerate the general approaches to project assignment that they observed and address how well they scale in size, along with analysis of their other strengths and weaknesses. The categories outlined below are based on those defined in the paper.

3.1.1 Two-Way Negotiation Systems

In a two-way negotiation system, students and professors discuss project allocation back and forth and come up with project assignments based on those discussions. This is equivalent to project assignment without any formal system at all. In a system like this, there may not even be a list of projects made available for discussion; students may be required to develop concepts directly with professors. As noted in *Hussain et al.,* this style of matching requires the most work from professors and students to execute[2]. It is also liable to leave students who are less confident about contacting professors without project assignments. However, it gives students and professors complete control over who they work with and what they work on, which is often a very desirable feature. For this reason, along with the fact that it does not require any technological infrastructure to set up, two-way negotiation is used in many universities despite its time costs.

3.1.2 One-Way Preference Systems

In one-way preference systems, students create a partial or complete preference ranking over some resource or group of resources in the matching system. Projects are then assigned using these rankings either manually or algorithmically. Some specific algorithmic implementations of this scheme are discussed below in section 3.2. One-way preference systems can be further broken down according to the resource ranked by students; the three most common things for students to rank are projects, areas of study, and professors.

By Title

In this type of project assignment system, students are provided with a list of project titles and descriptions. They then rank these project concepts according to which ones interest them the most. These ranks are then used to pick out projects for the students, either algorithmically or by using them to guide negotiation as described in subsection 3.1.1. The existing IDI project assignment system is a by-title one-way preference system where final assignments are made by negotiation, as is our prototype. Using rankings to assist matching makes this method faster than a pure negotiation system, but retains most of the student and professor control that comes from negotiation.

By Area of Study

In this type of assignment system, students rank areas of study rather than specific project proposals. Professors then reach out to students who ranked their area of study highly to attempt to arrange a specific project. According to a study produced by *Harland et al,* there was no significant difference in what factors students considered when choosing a project in this system, but professors were significantly happier with the resulting assignments because they felt they could

better match their projects with students[6]. However, this scheme requires more work on the side of the advisors than by-title preferences, as they have to manage the specific project assignment themselves.

By Advisor

The third type of one-way preference system has students rank specific professors instead of projects. The professors then get in contact with the students who ranked them highly and discuss projects. This system was proposed by *Calvo-Serrano et al.* to address the issue of matching a significantly larger number of students to a small number of professors[7]. According to the authors, no allocation system can guarantee that every student gets their first choice given this numerical imbalance. By shifting the students' ranking to the most limited resource in the system (professors), they are able to guarantee more high-ranking matches without negatively impacting student project selections. However, this scheme scales very poorly. As departments grow larger, it is more likely that students know a smaller set of their professors. In our user evaluation, all seven of the students who we spoke to said that they did not already know the professor they chose to work with prior choosing their project. This system would thus require extra work from students to research every professor in their department, and extra work from professors to then choose which projects to assign to which students.

3.1.3 Two-Way Preference Systems

In all of the one-way systems described above, students formally ranked some aspect of the system and then were matched with projects by negotiation or automation. In a two-way preference system, professors also rank their interest in students in order to produce a bidirectional ranking. Projects are then assigned algorithmically using both of these rankings. This is not a trivial problem to solve, and requires a complex multi-step optimization process which can only approximate the best match[8]. However, it does produce high-quality matches between students and professors without the need for time-consuming negotiation processes.

3.1.4 Student-Lead Systems

The final type of matching system outlined in *Hussain et al.* inverts the flow described in the previous matching schemes. In the student-lead scheme, students are required to propose their own thesis topics directly to professors, who then either accept their proposals or refer them to other professors who would be a better fit for the project. This system has the benefit of cutting out all of the work that would otherwise be necessary to generate and develop enough projects to cover the entire student body, and encourages greater student engagement because each student's project will be of personal interest to them. However, most

students also lack the research skills and knowledge of the field necessary to produce well-scoped, novel research topics. This can result in professors having to heavily alter student project proposals to make them feasible, which often makes students feel disenfranchised[2].

3.2 Existing Project Assignment Systems

In this section, we will outline some existing research on project assignment systems that we encountered in our literature review. This is only a small selection of the papers that are out there. They are chosen for their relevance to our implementation or demonstration of key automated matching methods.

3.2.1 Coordination of Student Project Allocation (2001)

This paper by *Dimitar Kazakov*, published in 2001 by the University of York, details the earliest implementation of a digital project assignment system that we were able to find. Prior to the implementation of this system, the University of York's Department of Computer Science handled final year project allocation entirely manually using a paper system[8]. Professors would deliver a printed page of project proposals to the department secretaries, who then compiled all of those proposals into a single document to be printed and distributed to students. Students then met with professors to discuss projects of interest and sign a final contract. Per the definitions above, this is an implementation of a two-way negotiation system.

The department struggled to maintain this process as the number of students needing projects grew past 150, leading Kazakov to design a new automated system. This system was then implemented by a student, Jamie Hodkinson. In this new system, professors turned their project definitions in to a system administrator who then entered them into a database. The matching process then operated in three phases:

1. Students search the database for interesting projects via a filtering system and rank them according to interest. Simultaneously, professors rank students according to how much they'd like to work with them. No matching could occur during this period.
2. Projects and students are matched algorithmically. First, perfect marriages (pairings where the professor and student are both each other's first choices) are matched, and those projects and students are removed from further consideration. Lower priorities are then shifted upwards, potentially creating new perfect marriages. These marriages are matched and this process repeated until no perfect marriages remain. Then an interactive, iterative process takes place where students and professors re-rank their remaining choices to produce new perfect marriages. This whole sequence repeats until the system runs out of ranked applications.

3. The remaining students are matched with the remaining projects at random, as no more rankings exist to guide the process. These matches are not made final until the students and professors who were randomly assigned can agree on the assignment via negotiation.

Interestingly, this system included a mechanism for ranking entire queries *en masse* – in addition to ranking their top *n* favorite projects, students could also rank entire sets of projects equally based on some filter query. This enables students to rank a much larger number of projects, which makes the matching algorithm more likely to converge on better results by reducing the number of random assignments. The paper did not include an analysis of the effectiveness of this matching system.

3.2.2 A dynamic project allocation algorithm for a distributed expert system (2004)

Another foundational paper on project allocation, this report by *Cheung et al.* describes the implementation of a front-end and back-end system for automated project matching[9]. Unlike Kazakov's implementation, which formatted the matching problem as an instance of the stable marriages problem, this system used an integer programming model across the one-way preferences provided by students to minimize the total ranking sum of all matched projects. In other words, this system finds the project allocation where the overall rank is as low as possible, which is not necessarily the same as the allocation where the most students get their first choice.

This system used three different front-end interfaces to enter data into its matching back-end: a student interface for creating project rankings, a professor interface for supplying project definitions, and an administrator interface for monitoring the submissions and activating the matching algorithm. This is quite similar to the system we propose in chapter 5. Notably, this system seems to require a very high number of ranked projects per student to resolve well; while its student interface demonstration showcases only five ranked projects, its data shows a student who got their 19^{th}-choice project. Obviously, it is a significant burden on students to find, read, and rank so many projects, especially if they have to discuss any details with professors.

Using this scheme, 30.8% of students received their top-ranked project, and 65.5% received one of their top five projects. This data was collected for only a single academic year of use, so it may have been skewed by a small number of highly-popular projects making it difficult for students to receive their top choices. However, if the data is representative of a typical usage of the system, then it seems likely that this allocation system could frustrate students if one third of them are not able to get one of their top five projects.

3.2.3 Preference Based Final Year Project Title Selection System (2009)

Hasan et al. outlined and compared several algorithms for allocating projects using one-way by-title preferences[10]. They asked each student to pick their top 10 projects from the set of project proposals provided by professors. They then fed these preferences into a program which ran three different matching algorithms over the data to produce project assignments. Following that, the authors analyzed these assignments both numerically and by gathering student feedback on satisfaction with the results.

The three algorithms compared were a naive rank sorting algorithm, a selection frequency sorting algorithm, and a network flow optimization algorithm. The authors found that the network optimization algorithm produced the most favorable results by far, with 40% of students getting their top-ranked project and 78.5% of students getting one of their top 3 projects. Additionally, students were more satisfied with the assignments generated by this algorithm and reported less stress during the process, compared to a first-come first-served system.

In order to implement the network flow optimization algorithm, the project allocation problem must first be interpreted as a directional network of nodes. This work was initially done for two-way preference systems by *Abraham et al.* in the influential 2007 paper "Two Algorithms for the Student-Project Allocation Problem"[11]. However, Abraham's paper was formalized for two-way preference matching, while Hasan intended to use student preferences alone. In the resulting modified formulation, nodes representing students are connected to nodes representing projects with edge costs equal to the students' preference rankings. This double layer of nodes is then sandwiched between a *source node* at the start, which is connected to every student with a cost of 0, and a *sink node* at the end, which every project connects to with a cost of 0. This creates a network of nodes which can accommodate any number of students and projects over which flows can be calculated. The authors then use a maximum flow, minimum cost algorithm to minimize the ranks of every project while ensuring that every student is matched.

Notably, while this system is able to easily encode additional constraints like the number of allowed students per project, it does not provide any guarantees on the maximum load of projects per professor. This is a serious issue, as a professor with popular projects could easily end up supervising far more projects than they could reasonably handle. The algorithms presented in *Abraham et al.* handle this, but that capability was lost by removing professor nodes. This could be a preventative issue for real-world uses of this scheme.

3.2.4 Additional Works

In 2017, The University of Hong Kong's Department of Computer Science published a paper titled *A Web-based Project Allocation System* by Lei Wan-Hong, which details the implementation of an automated student-assignment system in Django and Java[12]. We note this project here because its purpose and technology stack are very similar to our own.

Raihanah Binti Abdul Waha's 2012 paper *Final Year Project Online Management System* for Universiti Teknologi PETRONAS outlines the implementation of a project management system which implements an unranked two-way negotiation matching scheme[13]. This paper is of interest to us as it is one of the few that chooses to focus on manual matching instead of algorithmic matching. However, unlike our prototype, this system does not record student project preferences; it only acts as a portal for professors to post projects and for students to find them. Thus it is not of great use to study in depth.

3.3 Outcome of Review

Our literature review revealed that the vast majority of research into project assignment matching has been about algorithmic solutions. This is a reasonable result given the pressure of growing student bodies, and the desire for a system that produces the most fair outcomes possible. However, it reveals a lack of projects examining how to make manual assignment more effective and efficient without the use of algorithms. We consider this to be an important gap in existing knowledge, as improved manual matching may be an ideal solution for universities with a more manageable number of students, or that want more control over their matches. Additionally, none of the papers we reviewed detailed a formula for developing a project assignment system.

Our goal with this project was therefore to improve the matching system at IDI while remaining close to the existing implementation, and to document our process and rationale in a scientific manner which could aid other universities in implementing their own systems. Through this exercise, we hope to expand the knowledge of what assistive features can make manual negotiation-based systems more viable for larger student bodies.

Chapter 4

Methods

This section describes the methods used in the development and evaluation of the requirements and software presented in this thesis.

4.1 Development Methodology

In order to design software for research, an overarching approach must first be selected. The goal of our project was to develop a prototype which could address the issues we found from our interviews with users of existing project assignment systems. As these issues arose from the actual use of the system, they are considered practical problems. Software artifacts, when developed according to the design software process, can be an effective way to address practical problems while also providing generalizable scientific information to the field [14]. Therefore, we chose to use design science to organize our development process.

Design science research is a rigorous process for designing artifacts to solve problems [15]. There are many research strategies which can be used to actuate the design science process, but we chose to model our strategy on Action Research. The steps of the process are as follows:

1. *Explicate the Problem:* Identify a practical problem to be solved, as well as the underlying causes of that problem.
2. *Define Requirements:* Outline a solution to the explicated problem and elicit requirements for that solution. These requirements should be drawn from real practitioners who are impacted by the practical problem.
3. *Design and Develop an Artifact:* Create an artefact that addresses the explicated problem and fulfils the defined requirements.
4. *Evaluate the Artifact:* Determine how well the artifact fulfils the requirements and the extent to which it can solve or alleviate the practical problem that motivated the research.

We identified the problem of project matching through our personal experience with the IDI system, and began to determine the underlying causes through interviews with practitioners as described below. Through these interviews, we

generated a set of requirements which we believe any software solution to the student-project matching problem should implement. We then built a task backlog containing tasks corresponding to each requirement and developed the software artefact using that backlog as described below. Finally, we evaluated the effectiveness of the artefact and its requirements through user testing and process analysis.

4.2 Interviews

In order to understand what features are important for a project assignment software, we conducted one-on-one requirements elicitation interviews with professors and administrators at NTNU. Requirements elicitation is the process of seeking, uncovering, acquiring, and elaborating requirements for computer based systems [16]. Requirements elicitation is an essential part of the software design lifecycle; unclear requirements rank among the five most common reasons for software projects to fail [17]. There are many methods for requirements elicitation, including structured and unstructured interviews, card sorting and ranking exercises, and prototype presentations. The effectiveness of each method varies by project, but in most cases structured interviews are the most effective form of requirement elicitation, followed by unstructured interviews [18]. Methods for requirements elicitation generally provide information in five areas: understanding the application domain, identifying the sources of requirements, analyzing the stakeholders, selecting techniques to use, and gathering specific requirements. Interviews yield information in all of these categories, and are faster to set up and come more naturally to participants than other requirement elicitation methods [16].

We chose to run unstructured interviews based on the research above. We reached out to two professors and two administrators from IDI, as well as two members of NTNU's management who were also investigating solutions for master's thesis assignment. We asked each participant to discuss the workings of the current system from their perspective, what parts of it frustrated them, and what they thought it was missing. These questions were presented informally without a survey or other structured feedback system. Live notes on requirements and desires were taken during the meetings as participants brought them up. We combined these with requirements gathered from introspection on our own experiences with the system as students to get our final list of project requirements. Interviews by their nature yield a lot of extraneous information and requirements, which can be useful but distract from the final project [18]. Thus some filtering was employed to narrow down the final requirements to only those which aligned with our research questions.

4.3 Task Backlog

In order to track and prioritize work on the software portion of this project, we created a task board inspired by the Kanban system. Kanban is a system for controlling tasks and resource allocation first developed in the Japanese automotive industry [19]. To execute Kanban for software engineering, practitioners break up their proposed work into small, incremental tasks and lay them out on a backlog board. Individual developers then tackle these tasks one or two at a time, removing them from the backlog. This process keeps development focused on achievable tasks, limits the amount of work in progress at any one time, and helps reveal bottlenecks in production by providing a visual representation of tasks suck in pending states [19].

Tasks were added to the board to represent each of the functional requirements listed in chapter 5. These tasks were then grouped according to broad themes of development. These themes were then ordered by priority and developed one at a time. Because there was only one developer for this project, some aspects of the Kanban scheme relating to team management were not observed. Specifically, tasks in progress were not claimed by the individual developer, as there were no other developers to communicate this information to.

4.4 Time-to-Task Evaluation

RQ2 requires us to evaluate how our prototype compares to existing solutions in terms of usability. For the purpose of this project, we will assess this using task timing, or time-to-task evaluation. Time-to-task evaluation is a usability metric which is measured by setting out a discrete task or goal for the user, such as "create a new folder and move a document into it", and then timing how long it takes them to complete the task. This timing is then compared to the time it takes to perform the same task in another system [20]. By measuring this difference, we obtain quantitative data about how quickly the software works compared to another. Although this measure alone obviously lacks the nuance of full user sentiment data, "objective" performance measures such as task time are a reasonable predictor of actual user preferences [21]. Task timing data can additionally be enriched by tracking the task's success rate (whether the user is able to perform the task at all) and error rate (the number of mistakes the user makes when attempting the accomplish the task, such as visiting the wrong page or selecting the wrong item).

4.5 Keystroke-Level Modeling System

Due to technical limitations with the existing IDI system, we were not able to run tests within the software itself. Thus in order to evaluate task timings, we require a method to estimate task times without recording direct input. We chose the keystroke-level GOMS model to generate these estimates. GOMS takes its

name from "a set of **Goals**, a set of **Operators**, a set of **Methods** for achieving the goals, and a set of **Selections rules** for choosing among competing methods for goals"[22]. It is a model of human-computer interaction which attempts to predict the amount of time needed to use an interface to achieve a specific task by modeling the interaction as a series of actions taken conditionally, with mental processing time and physical input delay factored in. The prediction error of a well-constructed GOMS model is 21 percent for individual tasks[23] when compared to real user input, so in order to accurately compare the existing IDI system and our prototype, we chose to generate GOMS times for our own system as well rather than timing actual user input.

The keystroke-level model (KLM) is an instantiation of the GOMS system designed to predict interaction times with keyboard and mouse controls for an expert user of the system[23]. In the keystroke-level model, each interaction with the system is modeled as the sum of six operators:

$$T_{execute} = T_K + T_P + T_H + T_D + T_M + T_R \tag{4.1}$$

The operators are defined as follows:

- T_K*:* The time needed for individual keystrokes. This time varies per user based on their typing speed. We chose a time of 0.28 seconds per keystroke (40 words per minute) as our estimated typing speed. This is the speed presented for an average skilled typist in the keystroke-level model paper[23].
- T_P*:* The time needed to point the mouse at an individual target on the screen, such as a button or textbox. Although this time can be accurately modeled by Fitts' Law using exact pixel measurements[24], the KLM averages these pointing events to take 1.1 seconds based on the shortest and longest reasonable Fitts' Law values.
- T_H*:* The homing time needed for the user to place their hands on their keyboard or mouse. Card estimates this at 0.4 seconds.
- T_D*:* The time taken to draw a number of straight lines with the mouse in a drawing-based interface. This interaction is not used in either of our systems, so it will be omitted from our evaluations using the model.
- T_M*:* The mental preparation time the user takes before beginning the interaction. This models the time needed to read and process the page, search for relevant inputs and buttons, and decide what to do next. This quantity is highly variable per user and per task, but is estimated at 1.35 seconds in the keystroke-level model based on experiments performed by the authors[23]. It is worth noting that the real value of T_M is particularly high for creating new projects, as it would in theory cover the time required to think up a short description for a project. But because we keep this operator the same across our evaluations for both the old and new system, its actual value does not matter.
- T_R*:* The time it takes for the system to respond to the user's input. This is a real value that must be obtained from the system in some way. For most

web-based interactions, this is nearly instantaneous. However, actions like submitting a form may have more significant processing time. We chose to use real times from our prototype for this value, and estimated times for the IDI system based on our observation of recordings of the system in action.

Notably, *Card et al.*'s keystroke-level model does not take into account the time needed to scroll through a long page to find a specific item. This is a common interaction in modern web applications, but was not part of the user interface landscape when the paper was originally published. In order to account for this, we add an additional term to the equation, T_S. This scrolling action consists of a constant time for the user to assess if the information they are looking for is already on the screen, which we estimate at 1.5 seconds[25] and mark as mental preparation time M, plus 1 additional second for each full-screen scroll the user has to perform, which we mark as S. Thus our formula in practice is:

$$T_{execute} = T_S + T_K + T_P + T_H + T_M + T_R \tag{4.2}$$

The results of this evaluation can be found in chapter 7.

4.6 User Evaluation

RQ2 requires us to evaluate how our prototype compares to existing solutions in terms of user satisfaction. Quantitative methods of usability evaluation are significantly enhanced by pairing with qualitative methods of usability evaluation. There are many methods for evaluating ease of use and satisfaction from users, but for this project we chose to run an unmoderated user test. User testing is a highly effective way to gather data about the efficiency, effectiveness, and satisfaction of using a piece of software [26]. In a user test, participants are presented with a series of tasks which they must complete using the software. The user is asked to provide feedback on their experience using the software throughout these tasks. In a moderated user test, a researcher meets with the user in real time and prompts them through the tasks and questions. In an unmoderated user test, the user is instead given the directions and questions and asked to walk through them on their own, without a researcher present. We chose to run an unmoderated user test because it allowed us to reach more participants and fit our users' tight end-of-semester schedules. Unmoderated user studies produce less data than moderated user studies [26], as researchers are not able to watch the user's motions and get their moment-to-moment thoughts. However, we still felt it was the best choice for our project due to our scheduling constraints.

We designed our test following the steps outlined by *Whitenton, 2019* [27]. Our aim was to gather qualitative feedback on how the app compared to the IDI reference implementation in terms of features and usability. We wrote two separate test questionnaires, one for students and one for professors. The full text of both questionnaires is attached as Appendix A. In each questionnaire, we first asked a series of questions to establish the user's familiarity with the existing IDI

project assignment system. We then presented a sequence of tasks one at a time for the user to complete using a hosted version of our prototype. After each task, we present the user with a few questions about their experience of using the software to complete that task. We included at least one free response question with each task to allow users to express their thoughts in an unstructured way, mimicking the type of response elicited by an in-person structured interview. Finally, we ended each questionnaire with a series of questions about the overall experience of using the app and how it compared to using the IDI system. Student participants were recruited using posts in NTNU student Facebook groups and direct messages. Professor participants were recruited through email and in-person correspondence.

4.7 Process Modeling

Process modeling is the practice of analyzing and describing complex processes as visual flows. A simple and familiar example of process modeling is the flowchart, which breaks down a process into a simple sequence of conditional decisions. Creating and analyzing flows this way can help reveal inefficiencies, failure points, and bottlenecks in processes without requiring full implementations of the systems[28]. We use this technique to compare the methods used for project assignment in our prototype, the IDI system, and manually with no system. In this project, we chose to analyze our processes using the Business Process Model and Notation (BPMN), version 2.0. A business process is a set of activities that are performed in an organization, coordinated to jointly realize a business goal[29]. By modeling these processes, we can capture many of the constraints and requirements that govern them. The process of discovering and modeling these processes for an existing system is called process discovery. This task can be automated and performed at varying levels of rigor, but for our purposes we will develop our model through manual analysis. This allows us to produce models of the existing IDI system, our prototype, and a reference process for project matching with no system.

The Business Process Model and Notation specifies a number of nodes to represent real-world actions such as writing an application or waiting for time to pass, as well as logic operations like waiting for multiple processes to complete before continuing. Because many of these nodes have specific visual symbols and meanings, BPMN charts require some familiarity to be readable. We have produced an example graph in Figure 4.1, which uses a majority of the BPMN elements we will need to describe our processes in context and provides explanations for each.

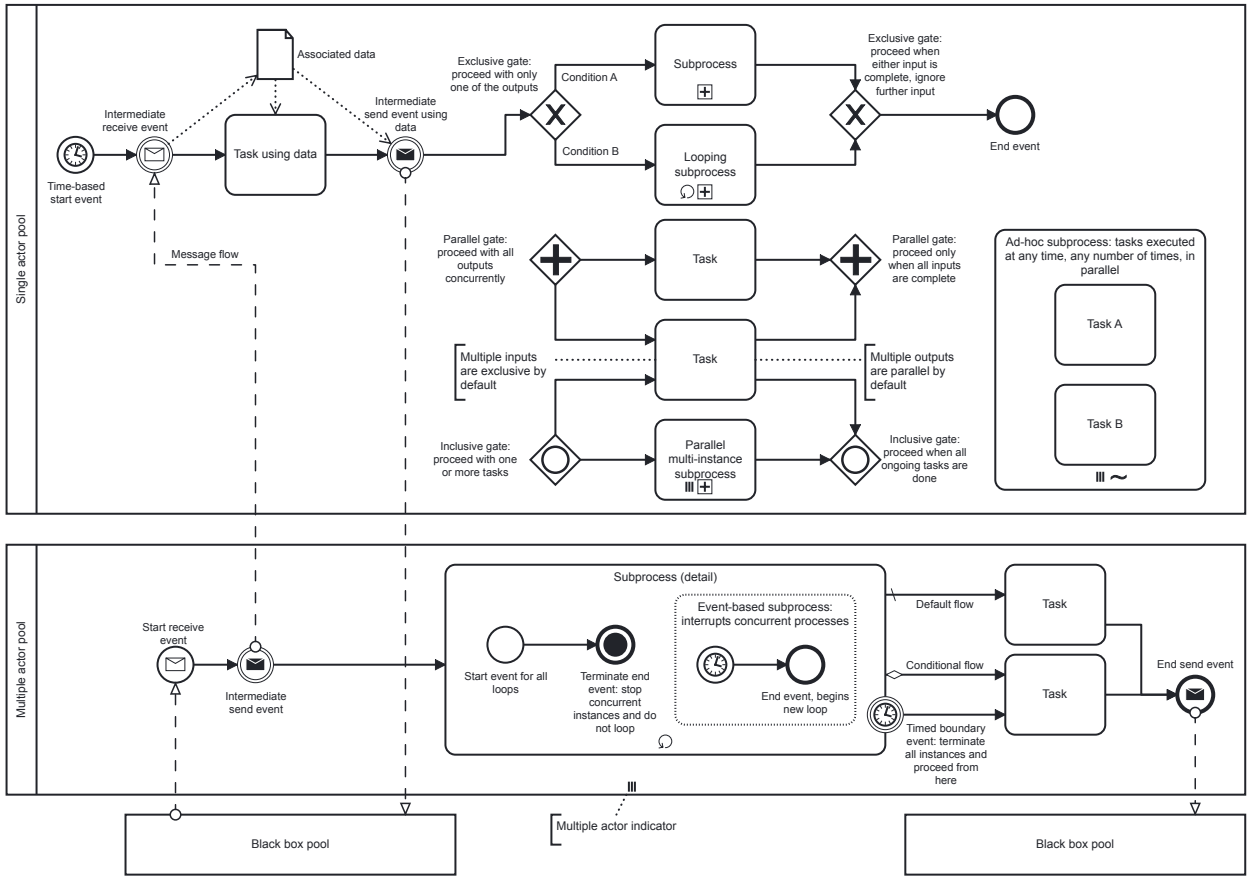

Figure 4.1: A legend of common BPMN elements

Chapter 5

Software Requirements and Specs

In this section we will outline the specifications for our program. These are the essential functions which we consider necessary for our prototype to be a reasonable solution to the problems presented in this report. They were developed using interviews with school administrators, professors, and students, along with our knowledge of the current system at IDI and gathered from literature review.

5.1 Goals

We aim to reduce the amount of ambiguous and time-intensive email communication necessary to match students and professors for thesis work in the current system. We have found this method of communication to be error-prone and poorly scalable for both professors and students. We address this by facilitating student applications and creating alternative contexts for communications, so that emails can be reserved for only the most necessary communication. This keeps all information about each project in one place for all parties, avoids the spam of dozens of application conversations going on at once in busy inboxes, and reduces the chance of applications being lost without followup.

We also intend to increase the quality of applications by making more information available to students and professors. Within the current IDI system, it is very difficult for students to find projects that are relevant to their skills, interests, and educational history. As discussed in section 2.1, this can be detrimental to the quality of research produced by the students. We fix this issue by introducing a comprehensive tagging system which allows students to filter by pre-requisite classes, areas of interest, fields of study, and more. To complement this, we also intend to support professor profiles where professors can specify information like their field of work and their supervision style, so that students can self-select for professors with complementary preferences and skills.

We also hope to reveal the popularity and availability of certain projects, to

encourage students to spread their applications more evenly among the available projects. This means professors won't need to turn down large amounts of students or take on more than they can handle, and mitigates students having to apply to other projects late in the application window because their desired project filled up.

For professors, we hope to give more information about students' preferences and options, to facilitate easier decision-making around which students are a better fit for the thesis, and which students already have other offers available. We address this by displaying useful metadata such as the student's priority for this application, as well as their specializations and preferred topics and tasks.

We aim to open up the possibility space to include more types of projects by allowing students to propose custom thesis projects, both directly to professors and generally to all professors with matching preference-tags. This is already a possibility at universities, but not something the old system accommodated well. The same framework will be used to accept proposals from third parties such as government agencies and private businesses, though this will be less self-selecting and instead managed by administrators. Accommodating passionate students with self-directed goals and academic partners with useful resources will help the university produce more impactful research.

Finally, we aim to design the system to be flexible, to be compatible with the different requirements of different institutes and universities, to whatever degree it is feasible. We also plan to make the data accessible in ways that make it convenient to integrate into the administrative environment of various institutes, primarily by allowing bulk-export of student data.

5.2 Non-goals

We do not plan to alter the flow of the university's existing thesis process. We do not propose any big changes to how and when projects are posted, how students choose projects, or how professors select students to work with. Such changes are beyond the scope of RQ1, and make evaluation of RQ2 more difficult. Therefore, our program is designed to facilitate the process as it currently stands. As a result of this, we also do not intend to implement automated matching. Although it is possible to automatically assign projects based on student preference rankings, it would significantly change the structure of the application process and remove the professors' ability to select students personally. A thorough assessment of the fairness of any kind of automated matching system should be undertaken before it can be considered.

We do not intend to develop a complete, comprehensive replacement for the existing IDI thesis assignment system. Due to the limited development time and the number of new features we aim to assess, we have decided not to re-implement some of the current system's existing functionality. We have attempted to only choose features which impact the matching process per RQ1. Features for handling theory modules will therefore not be included in our final product, as they

are specific to IDI and irrelevant to the process of pairing students with projects. For the same reason, we also will not integrate with any external university systems. We believe that such integration work would be very time-consuming and would prevent us from building out the new concepts that we plan to assess for RQ1. Similarly, we do not intend to focus on building a beautiful or optimized user experience for this tool. We will produce a usable interface which reasonably communicates all of the information the user needs, but the interest of this paper is in the improved process flow and information availability afforded by our new app, rather than refining the interface.

5.3 Functional Requirements

For this section, we use a set of priorities. A priority of "High" means the feature is necessary for a completed project. A priority of "Medium" means the feature contributes greatly to the quality of the prototype. A priority of "Low" means the feature has a smaller impact on the quality of the prototype.

5.3.1 Project Lifecycle

Table 5.1: Functional requirements regarding project lifecycle.

ID	Requirement	Priority
FR-PL-1	Professors must be able to create new projects within the system. Each project should have a title, description, tags for encoding useful metadata such as pre-requisite courses, and an expected group size.	High
FR-PL-2	Professors must be able to create drafts of new projects that are only visible to themselves.	Low
FR-PL-3	Professors must be able to view, edit, and publish drafted projects.	Low
FR-PL-4	Professors must be able to mark published projects as drafts, removing them from other views.	Medium
FR-PL-5	Professors must be able to define a total capacity for how many groups they intend to take on for a given project, though this should not limit applications or offers.	Medium
FR-PL-6	Professors must be able to duplicate old projects from previous years to set up projects for the current year, either as published projects or as drafts.	Low
FR-PL-7	Professors must be able to close any of their projects from further applications or edits, and removing them from other views.	High
FR-PL-8	Professors and administrators must be able to edit projects after creation.	High
FR-PL-9	Professors and administrators must be able to delete projects.	Medium

These requirements deal with the creation, updating, and deletion of projects. In our interviews with professors, we found that many of them preferred to copy closed projects from previous years, or clone a draft template that they kept from year to year, rather than create new projects from scratch every time. Thus we determined that in addition to basic creation and deletion of projects, it was necessary to allow users to create drafts and duplicate projects. This reduces the amount of time and repeated work needed to create projects, freeing up professors to correspond more with students, which is one of our goals.

5.3.2 Application Lifecycle

Table 5.2: Functional requirements regarding application lifecycle.

ID	Requirement	Priority
FR-AL-1	Students must be able to submit applications to projects listed on the site. These applications must include a text section for the student to explain why they are interested in the project and would be a good fit.	High
FR-AL-2	Students must be able to retract applications.	Medium
FR-AL-3	Students must be able to preemptively accept any project they have submitted an application to.	High
FR-AL-4	Professors must be able to respond to applications and ask for additional information from students.	Medium
FR-AL-5	Professors must be able to decline applications.	High
FR-AL-6	Professors must be able to extend final offers to students.	High
FR-AL-7	Students must be able to accept one final offer to confirm their selected project.	High

Applications require some automation to improve over the base model of emailing back and forth with a professor. We propose that applications should be directly tied to one specific project, with automated convenience features like preemptive acceptance, which accepts an offer as soon as a professor extends it. Applications should also have clearly-defined states which are visible to both the professor and student in order to minimize confusion about whether or not the application has been accepted. These features take the administrative load of managing applications off of the individual users and place them in the hands of the system instead. We believe this will reduce errors in the matching process and avoid the cognitive load of managing applications in an unstructured e-mail inbox.

5.3.3 Student Views

Table 5.3: Functional requirements regarding student views.

ID	Requirement	Priority
FR-SV-1	Students must be able to see a list of all projects. This list should allow them to browse through the available projects and enable them to visit the details page for each project.	High
FR-SV-2	Students must have a set of fixed academic tags that convey their academic background.	High
FR-SV-3	Students must be able to edit their personal interest tags.	Medium
FR-SV-4	Students must be given some indication of which projects have a high number of applications, and which professors or projects have low or no remaining capacity.	Medium
FR-SV-5	Students must be able to see a list of applications they have submitted to projects. This list should include the title, description, and professor of each project, as well as the status of the application, along with the text of the student's application and any follow-up conversation with the professor.	High
FR-SV-6	Students must be able to respond on applications where the professor has requested more information.	High
FR-SV-7	Students must be able to rank their top 5 projects based on their level of interest in each project. These ranks should be mutable, and visible to the professors of those projects alongside the student's application.	Medium

In our reflection on our own experience with the process and in speaking with other students, we determined that visibility and filterability of project data was crucial to making the system effective for students. By giving students tools to cut down the large pool of projects into smaller groups, we enable them to find the types of project they are looking for more quickly. Additionally, allowing them to manage all of their applications and discussions within the app cuts out the time lost searching for email threads and figuring out which projects they were about. This should help students find and apply to more relevant and interesting projects, leading to better research outcomes.

5.3.4 Professor Views

Table 5.4: Functional requirements regarding professor views.

ID	Requirement	Priority
FR-PV-1	Professors must be able to view a list of all of their projects. This list should include applications from students for each project.	High
FR-PV-2	Professors must be able to see and respond to all applications from students on a project's details page.	High
FR-PV-3	Professors must be able to get any student's email by clicking on their name.	High
FR-PV-4	Professors must be able to view the details of each application on their projects. These include the student's name, the title and description of each project, along with the text of the student's application and any follow-up conversations.	High
FR-PV-5	Professors must be able to see retracted applications listed separately.	Low
FR-PV-6	Professors must be able to define a total capacity for how many projects they intend to take on, though this should not limit applications or offers.	Low
FR-PV-7	Professors must be able to view a list of all third party thesis proposals that have been approved by an administrator.	High
FR-PV-8	Professors must be able to edit, decline, or accept approved third party thesis proposals.	High
FR-PV-9	Professors must be able to view all student thesis proposals.	High
FR-PV-10	Professors must be able to view student thesis proposals assigned to them separately.	Medium
FR-PV-11	Professors must be able to decline student thesis proposals assigned to them, which removes them from the proposal. The proposal remains in the view of all student proposals.	Low
FR-PV-12	Professors must be able to accept student thesis proposals.	High
FR-PV-13	Professors must be able to copy a student thesis proposal into a draft for editing.	Low

In our interviews we found that when evaluating student applications, professors often need to ask students for more information on their academic history and qualifications for a project. Enabling professors to have those conversations and see them attached directly to the applications and projects they pertain to is

therefore very important in a thesis matching system. In addition to applications and project listings, the administrators and professors we spoke to also expressed interest in systems for handling student project proposals and projects submitted by third-party industry partners. While these are important features, the majority of students will take on projects defined by professors. Thus the requirements for these systems are listed, but with lower priorities.

5.3.5 Administrative Views

Table 5.5: Functional requirements regarding administrative views.

ID	Requirement	Priority
FR-AV-1	Administrators must be able to view a list of students who have not yet been extended a final offer.	High
FR-AV-2	Administrators must be able to view a list of students who have been given offers but not accepted one.	Medium
FR-AV-3	Administrators must be able to get any student's email by clicking on their name.	High
FR-AV-4	Administrators must be able to override the academic tags of a student.	Low
FR-AV-5	Administrators must be able to view all thesis proposals submitted by third parties.	High
FR-AV-6	Administrators must be able to create accounts for third parties that are cleared to submit thesis proposals.	Medium
FR-AV-7	Administrators must be able to edit or decline third party thesis proposals.	Medium
FR-AV-8	Administrators must be able to assign professors to third party thesis proposals.	High
FR-AV-9	Administrators must be able to add relevant tags and prerequisites to third party thesis proposals.	High
FR-AV-10	Administrators must be able to approve third party thesis proposals only when they have been assigned a professor.	High
FR-AV-11	Administrators must be able to view student thesis proposals.	Medium
FR-AV-12	Administrators must be able to edit or decline student thesis proposals.	Low
FR-AV-13	Administrators must be able to view each page from the perspective of any other account in order to verify that projects, applications, and other features are set up correctly.	Medium

The university administrators we spoke to expressed a need to manipulate almost every aspect of the system in order to address errors and unusual cases. They also intended to be the managers of third-party projects such that third parties will not need familiarity with the institutes professors. Finally, they required diagnostic views of the system to let them find students who may need additional help. This resulted in a sizable set of requirements. Unlike with professors and students, the goal of these requirements is to create an interface that will get out of the way of administrators as much as possible. They are power users who want specific things from the system, and thus should be able to achieve those things with as little resistance as possible.

5.3.6 Automatic Processing

Table 5.6: Functional requirements regarding automatic processing.

ID	Requirement	Priority
FR-AP-1	All projects must be automatically closed and hidden before the next academic year.	High
FR-AP-2	Upon accepting a final offer, the student's other applications must be marked as retracted.	High
FR-AP-3	When a professor extends a final offer to a student who has preemptively accepted that project, the offer is accepted immediately.	High
FR-AP-4	Applications must be deleted fully after a project is closed.	Low

Automation is one of the main improvements gained from tracking applications and projects in a dedicated system instead of via e-mail. We identified that it was important to return the system to a clean state, with all projects closed and all applications deleted, at the start of each academic year in order to reduce the workload of system administrators. One common complaint from students was that after accepting one application, they would have to email every other professor they'd spoken with to rescind their other applications. And similarly, professors would need to inform other students that they would not be accepting more students to a project. Thus we propose automating that process to cut out that unnecessary, repetitive work.

5.3.7 Notifications

Table 5.7: Functional requirements regarding notifications.

ID	Requirement	Priority
FR-NT-1	The system must notify students of key events via email. These include extended offers, accepted offers, accepted or copied proposals, declined proposals, declined applications, and responses on applications.	High
FR-NT-2	The system must notify professors of key events via email. These include new applications, accepted offers, new assigned proposals, and responses on applications.	High
FR-NT-3	Notifications must be batched and messaged out daily or weekly to avoid inbox spam.	Medium

Although we believe that it is important to move application communication out of unstructured e-mail inboxes and into a system designed explicitly for that information, we also know from our interviews that professors and students use their inboxes as their main source of information on academic matters. Thus we conclude that in order for a matching system to perform well, it must still reach users in their inboxes, otherwise users are likely to miss key events and information. To avoid clutter, and emails getting lost, we also propose these notifications are sent in batches so that give a better overview.

5.3.8 Filtering

Table 5.8: Functional requirements regarding filtering.

ID	Requirement	Priority
FR-FL-1	Tags must exist for prerequisite classes, specialization or subject, topics of interest, and method of work.	High
FR-FL-2	The list of all projects must support filtering by the presence or absence of tags, and by assigned professor.	High
FR-FL-3	By default, students see a list of all projects which are relevant to their personal and academic tags.	High
FR-FL-4	The list of all student thesis proposals must support filtering by the presence of tags, and by assigned professor.	High
FR-FL-5	By default, professors see a list of all thesis proposals which are relevant to their personal tags.	Medium

One of the biggest pain points that we identified for students was difficulty in finding relevant or interesting projects. Searching for keywords was seen as unreliable and left students worrying that they weren't finding everything they wanted

to see. Thus we propose a system of specific tags which can be set by professors and filtered by students to make finding the right projects take less time. We also suggest allowing students to apply tags to themselves based on their interests and academic history, which are then used to automatically filter the list of projects. By making it easier to find projects about specific topics, students are more able to find interesting theses to work on, leading to better research outcomes.

5.3.9 Thesis Proposals

Table 5.9: Functional requirements regarding thesis proposals.

ID	Requirement	Priority
FR-TP-1	Third parties with cleared accounts must be able to submit proposals for theses. These require a title and description, with the option to add tags or suggest a professor.	High
FR-TP-2	Third parties must be able to retract thesis proposals, removing them from view completely.	Low
FR-TP-3	Students must be able to submit proposal for theses. These require a title, description, and tags, with the option to suggest a professor.	High
FR-TP-4	Students must be able to edit their thesis proposals. Copies of proposals will not be edited.	Medium
FR-TP-5	Students must be able to retract thesis proposals, removing them from view completely. This is not possible for accepted proposals. Copies of proposals will not be removed.	Low
FR-TP-6	Professors must be able to view projects proposed to them and coordinate with the proposer.	High
FR-TP-7	Accepted project proposals must automatically populate into the system. For external partners, this should create a project listing available to students. For student proposals, this should create a new project and automatically assign the student and professor to it.	High

The IDI thesis matching system which served as our reference implementation did not include any features for managing student-proposed projects. Students felt that the process the university had in place for these projects (emailing a pitch of your project idea to whichever professors they see fit) was challenging and hostile to project proposals. We believe this is a flaw, as self-directed projects from motivated students often produce better learning outcomes[4]. Thus we propose this set of features for managing project proposals from students.

5.3.10 Exportable Data

Table 5.10: Functional requirements regarding exportable data.

ID	Requirement	Priority
FR-ED-1	The system must have functions for exporting lists of student emails or usernames for any relevant grouping, such as all students without any extended offers.	High
FR-ED-2	The system must have a function for exporting a plain-text list of all finalized student-project pairings.	High
FR-ED-3	The system must have a function for exporting a plain-text list of all open projects, with their description and professor.	High

Based on discussion with administrators and professors, as well as being aware of our limitations in integrating with other university systems, we found a need for making the system data available in a simple format. Both administrators and professors mentioned wanting to contact single students as well as sets of relevant students based on some criteria. An administrator also mentioned wanting to export open projects, for example to compile into documents for easy viewing by third parties or students without access to the system. Further, for the system to integrate with other systems, such as Inspera for digital submission deadlines, it would be necessary to export student-project pairings especially.

Chapter 6

Implementation

This chapter details the decisions made and lessons learned during the implementation of the prototype. The content of this section may be of use to future researchers or individuals attempting to build effective project assignment systems, and is based on the experience of the researcher. The applicability of this information may be narrow due to its specificity, as it only regards the development of this particular software artefact. However, we believe that it is scientifically valuable regardless.

A presentation of the final state of our prototype can be found in chapter 7. The outcome of our user tests is documented in chapter 8, and a discussion of the overall success of the prototype can be found in chapter 9.

6.1 Choice of Technology

The technology selection for this prototype was done as part of the prior foundational work mentioned in section 1.2. We considered several application types and underlying frameworks before eventually settling on a Django web app. The reasoning for our choices is reproduced below.

6.1.1 Web vs. Desktop vs. Mobile

Our first major decision was choosing the platform on which we wanted users to access our prototype. The broad access paths we considered were a web application that runs in the browser, a desktop application that runs natively on laptops and desktops, and a mobile application that runs on smartphones and tablets. Our goal was to make the prototype as accessible to as many students as possible, in order to not advantage or disadvantage anyone based on access to technology. Our decision was narrowed down by our understanding of the system's very limited scope of use. We know the system will be utilized by professors, primarily in office settings, where a personal computer is the primary device. On the student side, they only need to access the system for a short duration of their entire university programme. This means they should have access to computers, either as

a resource provided on campus, or as the medium for remote learning. If the system is not accessible to other devices, such as phones, this should not limit what students are able to access the system. Beyond this, as there is much variation between personal computers, we wanted the system to be compatible with a wide array of devices, and easy to expand to mobile devices if necessary. Additionally, we wanted a platform that we could develop for very rapidly, as we were under time constraints to get the prototype to a testable state. Ultimately, we chose a web application to deliver our prototype.

Web applications have many beneficial features which make them ideal for this type of development. Web browsers act as platform-agnostic intermediaries which can interpret and display the same code on any device. By targeting web browsers as our access point, we were able to write one application and have it be accessible from any operating system and any device. This fulfills our goal of making the application as widely accessible as possible, although it requires additional work to adjust the user experience accordingly. Additionally, web application development is very amenable to rapid iteration. New versions of the prototype can be instantaneously deployed without requiring users to download or update anything. The separation of front- and backend code enforced by a web application's client-server architecture also makes it easy to update one side of the application without impacting the other, allowing us to focus on specific parts of the app and repeatedly update them without worrying about breaking unrelated pieces of the prototype. The researchers were also previously very familiar with HTML/CSS code and web application design, meaning that building the system as a web application would require less learning and start-up time than the other proposed methods.

The choice of a web application aligns with multiple constraints that we had already imposed on the project. For one, a web application would require all of our users to have internet access in order to use it. However, since our application would use its users' university accounts in a full implementation, it would require an internet connection anyway to do that authentication. Additionally, universities generally offer on-campus wireless networks which students can connect their devices to free of charge. For students without personal devices to access the internet, university library computers or borrowed devices are sufficient to access the site. Thus this constraint should not be a problem for this application.

Another major constraint of using a web application for our prototype is that it requires a hosting server to be maintained for the application to be accessible. If the server ever goes down, the application becomes completely unavailable. Additionally, running and maintaining a web server has recurring costs which must be considered. However, regardless of the front-end solution, every possible version of this service would require a server with a database to store and send project and application data. That server would be subject to the same server uptime constraints as the web application, and would require users to have an internet connection to access the data necessary to run the application. Thus there was no additional downside to using a server to manage our interface as well.

Our most restrictive constraint is that we require users to have a "modern" web browser. Web browsers have existed for over 30 years, and new standards of behavior are constantly added and adopted to different browsers at different times. There are thus inevitably browsers which will be incompatible with the features necessary to run and display a modern web application. In order to minimize the chance of this preventing some students from using our site, we chose to implement our front-end interface using basic HTML and only widely-adopted CSS and JavaScript features. Our metric for wide adoption was a usage score of 90% or more on *Can I Use*[1], a website which tracks the compatibility of CSS and JavaScript features with a wide range of web browser versions to produce data on how fully implemented a given feature is. Although this seems restrictive at first, this constraint comes up in mobile development as well; different operating system versions support different APIs, requiring some users to be left behind because their phones are too old. Furthermore, native mobile and desktop development both suffer from platform segmentation due to competing operating systems on both platforms. This requires developers to create two versions of their apps, or use frameworks which can bridge the gap to display one app on both operating systems. For web applications, supporting both mobile and desktop environments still has an additional cost to adapt the user interface, but a minimum level of functionality is guaranteed on all platforms by default. Given these constraints, it made the most sense for us to choose to build a web application, as the browser version requirement is less restricting than device or operating system restrictions.

6.1.2 Programming Language Choice

The next major question was what programming language to use for the server back-end. Our principles for this decision were similar to the ones we considered for our general application type. We needed a language which would allow rapid development, be easy to read and maintain, and which was broadly supported for server deployment. Ultimately, we chose to use Python for our web server. Because it is an interpreted language, Python is easy to run on almost any server. Additionally, its syntax is very readable by design, and has a wide set of features that make it easy to adapt to any problem. As with HTML/CSS above, the researchers also had past experience with working in Python, which meant that no learning time was needed for the language. There were no particular drawbacks that we identified with Python, or with any of the other languages that we considered, so this decision was made purely on the merits of Python rather than on avoiding issues presented by other languages.

6.1.3 Framework Choice

The final technology consideration was whether to use a framework for writing a Python web-application or develop from scratch, and if using a framework, which

[1] https://caniuse.com/

framework to use. From-scratch development is appealing, as it requires no time spent learning how to use a specific framework and gives the writer the most control over the application's behavior and code. However, writing a web server is a complex problem that is very error-prone and has been solved many times over. Thus we quickly chose to use a framework instead of developing from scratch. We evaluated a few Python frameworks, including Flask, Django, and Masonite. Flask is a very popular Python web server framework with a long history of support, having first been released in 2010 and updated continuously to the time of this writing. It provides useful mechanisms for handling requests and it is highly extensible and customizable. However, Flask does not provide any tools for interacting with databases, requiring developers to write their own database interaction code or install additional libraries to do it. Because our project was so heavily database-driven, we wanted a framework which would tie our database directly to our Python code, so we chose not to use Flask. Masonite is a much newer framework, first released in late 2017. Like Flask, it provides simple request handling features, but adds a database management system and a number of convenience features for common tasks like user authentication and sending notifications. Although Masonite is a powerful framework, we found its existing knowledge support (Stack Overflow Q&As, forum threads, etc.) lacking due to its young age. Thus we settled on Django.

Django is the second most popular Python web framework behind Flask[2]. Similar to both Flask and Masonite, it offers a suite of request handling tools. It also provides an authentication system and an administrator interface for managing the website right out of the box. As with Masonite, it adds a database layer that allows developers to write database models in Python and automatically handles the actual database logic. It also provides a convenient system for managing complex multi-object database relations that are difficult to implement without a helping tool. We found Django's database system to be very expressive and applicable to the things we needed to model, and its management tools very appealing, so we chose to work with Django.

6.2 Personas and Scenarios

After developing some of the basic functionality for our prototype, we considered our project goals and initial interviews to develop a series of professor and student personas, along with a set of core scenarios required by those personas to fulfil their needs within our system. We then used these personas and scenarios to determine what features we should implement with our remaining development time in order to produce the most complete, testable prototype we could. Those personas and scenarios are presented in this section.

[2] https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies

6.2.1 Professor Personas

Damian

Damian is a visiting computer science professor. This is his first year supervising thesis projects at the university. As such, he is only sponsoring a small number of projects in his area of study, and is not interested in advising students with their own topic proposals. His goal is to learn the system, post a few project listings, and choose a few applicants to advise.

Celine

Celine has been a professor in her department for over a decade. Over the years she has accumulated a large list of project topics for students to work on. She has numerous un-assigned projects from previous years which she would like to open again for this year, in addition to a small list of new ones. She needs a system that lets her manage a large number of projects and applications, re-use old project listings without trouble, and accommodate a power-user with years of experience.

Gregg

Gregg has been doing research into a very particular sub-topic of his field, and wants to work with a student who is very driven and interested in his specific niche of research. He has already formed tentative agreements to work together on research with some of the students who took his class. Gregg's goal with the system is to quickly create listings for these projects and get them assigned to students he's chosen.

Frank

Frank is a professor who is interested in mentoring students who propose their own project topics. Although he has some projects listed in the system, he encourages students to suggest their own research for them to collaborate on. His goal is to get those student projects into the project tracking system and school administrative systems as painlessly as possible, and keep those proposals and discussions together in one place to avoid any student ideas falling through the cracks.

6.2.2 Student Personas

Alice

Alice is a highly-motivated student who already has a strong working relationship with one of her professors. She has already discussed working together with this professor on her thesis. Her goal is to find her professor's projects in the system quickly, apply to the one she and the professor have already discussed, and then automatically accept the professor's offer as soon as it comes in.

Bob

Bob was busy with coursework and exams, and waited until the last minute to select a thesis project. He does not care about finding a project that suits his particular interests; he just needs to quickly find a project that he has all of the prerequisites for which has open slots and few applicants.

Emil

Emil is a deeply passionate student who only wants to work on a project that interests him. He is also very indecisive when it comes to choosing among those projects, and needs time to ask questions and mull over his choices before committing to one topic. He needs a system that will let him easily find projects that match his specific interests, communicate with professors about his questions, and prioritize his various applications as he thinks about what he would like to research the most.

Hanna

Hanna is a deeply introverted student who dislikes contacting professors and would prefer a very hands-off supervisor. Working style is more important to her than project content. She needs a system that exposes as much project information as possible so that she does not have to email professors asking for more detail. Her goal with the system is to find a project advisor whose style of supervision works for her.

Ingrid

Ingrid and her friend have worked together on a number of projects in the last few years, to great success. She wants to take on her thesis project together with that friend as well. Her goal is to find a project that can accommodate her and her friend, and communicate her group's interests, and experience to that project's professor.

Jakob

Jakob has been reading research papers over the summer as part of an internship and has a great idea for a project that he would like to pursue. Unfortunately, it is not an idea that any professor already has a listing for. Thus Jakob wants to propose his own project. He needs a system that will let him propose a custom project, find a willing supervisor, and finalize the assignment with whichever professor agrees to help research it.

6.3 Scenarios

Using those personas, we then developed some scenarios for how they might interact inside and outside the system. The individual steps of these sequences were given simple labels so we could refer to them later.

Alice and Gregg

Gregg has previously talked to Alice about her taking on one of his projects, and only needs to get it registered in the system.

AG-1 Gregg creates a listing for the project and then sends the link to Alice.
AG-2 Alice applies to the project and chooses to automatically accept the offer as soon as it comes in.
AG-3 Gregg sees Alice's application and approves it, triggering the automatic acceptance.
AG-4 Gregg closes the project now that Alice's offer is finalized.

Damian and Emil

Damian wants to write a small set of project listings for his specific part of the field, and make them easy to find for students who are interested in that specific work. Emil wants to work on a project related to this topic, but hasn't discussed it with Damian in advance.

DE-1 Damian creates a new project listing with tags to reflect his specific niche of the field.
DE-2 Emil searches for projects with this tag, finds Damian's listing, and applies for it.
DE-3 Damian decides that Emil is a good fit for this project based on his application and profile, but wants more detail on his personal history with the topic. He asks Emil if he has previous experience with the topic.
DE-4 Emil explains that he does, so Damian extends an offer.
DE-5 Emil waits to see if any of his higher priority applications are accepted before finalizing this one.
DE-6 Eventually, Emil accepts the offer for the highest priority project he can get. His other applications are immediately retracted.

Celine

Celine has a big backlog of previous projects to draw on for this year. She wants to quickly re-list them for this year's students (updating them where necessary), and then easily manage the large number of applications she will get.

C-1 Celine creates her projects by duplicating previous listings and editing them when necessary. She keeps track of new applications via the applications page throughout the matching period.

C-2 Because she has more listed projects than she can actually supervise in a semester, Celine waits to see which projects get the highest-ranked applications. She chooses students for each project based on their priority rank and application message, ignoring low-priority applications without rejecting them. She only rejects students once she has found enough high-priority matches to fill her workload, or once the matching period has ended.

Bob

Bob needs to find an appropriate project quickly. He cares more about availability and academic qualifications than topic.

B-1 Bob filters the list of open projects to remove any that require classes he has not taken, as he does not have time to negotiate the requirements with a professor. Additionally, he filters to only solo projects, as he has no time to find a partner.
B-2 Bob chooses projects that few students have already applied to in order to reduce his competition for the slot, and ranks his applications highly.
B-3 Bob accepts the first offer that he gets.

Hanna

Hanna dislikes frequent emails and meetings, and prefers to manage her own time as much as possible. This is the most important factor for her in picking an advisor.

H-1 Hanna filters to only see projects with working style tags like “infrequent meetings”, “biweekly meetings”, or “supervision by request” to show only those that mention minimal supervision, then applies to many of these projects and ranks them according to her interest in each topic.
H-2 Hanna receives an offer from a professor for one project, but rejects the offer because she finds out that the research methodology of the project involves user interviews.

Ingrid

Ingrid has worked together with two friends on numerous projects in the past, and would like to do her thesis project with them as well. She needs a project that can accommodate all of them.

I-1 Ingrid filters the available projects to those whose group size is 3 or more.
I-2 Ingrid applies to the project and adds her friends to her application.
I-3 One of Ingrid’s friends decides to withdraw herself from the application because she has found a solo project to pursue.
I-4 The professor evaluates Ingrid and her friends as a group and accepts all of them.

Jakob and Frank

Jakob has his own topic idea to propose, and needs to find a professor to advise his research into it. Frank is interested in advising student-proposed topics.

JF-1 Jakob browses the projects and profiles of a few professors, then decides that Frank and Celine are both good fits for what he wants to research. He creates a project proposal and assigns it to both professors.

JF-2 Frank sees the proposals and reviews them. After some more discussion with Jakob and the administrative staff over email, he approves the project. Frank now sees the project alongside all of his others, with Jakob as its assigned student.

6.4 Revised Requirements

After constructing these scenarios, we then took each labeled scenario step listed above and matched it up to the relevant functional requirements from chapter 5. This helped us prioritize which functional requirements to focus on for the final stretch of development on the prototype. Due to the limited number of working hours available for this project, we had to drop large parts of the ambitious original requirements. In this section, we will document the functional requirements that drove the second half of development.

The biggest change was the removal of the Administrative Views, Notifications, and Exportable Data requirements. Administrative Views and Exportable data are both feature sets that primarily affect the user experience and efficiency of administrators of the system, which is important for the system in practice, but largely revolves around large data volumes and edge cases that are difficult to test and infrequently affect our prototype. **FR-AP-1** was removed for the same reasons, as this is a feature that is only relevant once a year and is primarily an administrative task. Notifications, while extremely important in an actual implementation of our system, only impact the user experience on larger scales of time than a single demo test session. Additionally, they do not change the flow or process of any tasks, so we can still evaluate everything we need to in the prototype without notifications.

Notably, we chose to keep the Thesis Proposals requirements after this process, even though we ultimately did not get the time to implement them. While it was necessary to scope down due to the remaining time, Thesis Proposals aligned very well with our project's goals, and thus were worth keeping in mind as we moved towards the end of development in case any time opened up.

This process also allowed us to identify two missing functional requirements, things which we had already implemented or intended to implement but which were not described in our functional requirements table. Specifically, when evaluating scenario **JF-1**, we found that we were missing a functional requirement that specified the existence of profile pages for professors. We added this as **FR-PV-14**. And when evaluating **H-2**, we found that we were missing a functional

requirement that specified that students should be able to reject project offers from professors, even though we had already implemented it. We added that as **FR-AL-8**.

The full list of revised requirements from this process is summarized below.

6.4.1 Project Lifecycle

Table 6.1: Revised requirements regarding project lifecycle.

ID	Requirement	Priority
FR-PL-1	Professors must be able to create new projects within the system. Each project should have a title, description, tags for encoding useful metadata such as pre-requisite courses, and an expected group size.	High
FR-PL-2	Professors must be able to create drafts of new projects that are only visible to themselves.	Low
FR-PL-3	Professors must be able to view, edit, and publish drafted projects.	Low
FR-PL-4	Professors must be able to mark published projects as drafts, removing them from other views.	Medium
FR-PL-5	Professors must be able to define a total capacity for how many groups they intend to take on for a given project, though this should not limit applications or offers.	Medium
FR-PL-6	Professors must be able to duplicate old projects from previous years to set up projects for the current year, either as published projects or as drafts.	Low
FR-PL-7	Professors must be able to close any of their projects from further applications or edits, and removing them from other views.	High
FR-PL-8	Professors must be able to edit projects after creation.	High
FR-PL-9	Professors must be able to delete projects.	Medium

The requirements for project lifecycle remained largely the same, but with the removal of the role "administrator" from **FR-PL-8** and **FR-PL-9**. The project lifecycle requirements were so central to the functionality of the system that much of this functionality was already complete by the time of this analysis.

6.4.2 Application Lifecycle

Table 6.2: Revised requirements regarding application lifecycle.

ID	Requirement	Priority
FR-AL-1	Students must be able to submit applications to projects listed on the site. These applications must include a text section for the student to explain why they are interested in the project and would be a good fit.	High
FR-AL-2	Students must be able to retract applications.	Medium
FR-AL-3	Students must be able to preemptively accept any project they have submitted an application to.	High
FR-AL-4	Professors must be able to respond to applications and ask for additional information from students.	Medium
FR-AL-5	Professors must be able to decline applications.	High
FR-AL-6	Professors must be able to extend final offers to students.	High
FR-AL-7	Students must be able to accept one final offer to confirm their selected project.	High
FR-AL-8	Students must be able to decline offers that have been extended.	Low
FR-AL-9	Students must be able to name additional group members on an application.	Low
FR-AL-10	Added group members of an application must not be able to edit that application.	Low
FR-AL-11	Group members must be able to withdraw from the group application.	Low

We noticed the absence of specifications regarding group applications, and decided to flesh these out, as the current IDI system puts this work onto students and professors in emails, and additionally requires all students to submit their applications separately. Both professors and students mentioned this being inconvenient. We would also have liked to add features assisting in pairing up single students that want to work on projects scoped for groups only, but decided the added work of this would not fit into our development schedule at this point.

6.4.3 Student Views

Table 6.3: Revised requirements regarding student views.

ID	Requirement	Priority
FR-SV-1	Students must be able to see a list of all projects. This list should allow them to browse through the available projects and enable them to visit the details page for each project.	High
FR-SV-2	Students must have a set of fixed academic tags that convey their academic background.	High
FR-SV-3	Students must be able to edit their personal tags.	Medium
FR-SV-4	Students must be given some indication of which projects have a high number of applications, and which professors or projects have low or no remaining capacity.	Medium
FR-SV-5	Students must be able to see a list of applications they have submitted to projects. This list should include the title, description, and professor of each project, as well as the status of the application, along with the text of the student's application and any follow-up conversation with the professor.	High
FR-SV-6	Students must be able to respond on applications where the professor has requested more information.	High
FR-SV-7	Students must be able to rank their top 5 projects based on their level of interest in each project. These ranks should be mutable, and visible to the professors of those projects alongside the student's application.	Medium
FR-SV-8	Students must be able to see any group projects that they have been added to in their applications list.	Medium

No changes to the student view requirements were necessary.

6.4.4 Professor Views

Table 6.4: Revised requirements regarding professor views.

ID	Requirement	Priority
FR-PV-1	Professors must be able to view a list of all of their projects. This list should include applications from students for each project.	High
FR-PV-2	Professors must be able to see and respond to all applications from students on a project's details page.	High
FR-PV-3	Professors must be able to get any student's email by clicking on their name.	High
FR-PV-4	Professors must be able to view the details of each application on their projects. These include the student's name, the title and description of each project, along with the text of the student's application and any follow-up conversations.	High
FR-PV-5	Professors must be able to see retracted applications listed separately.	Low
FR-PV-6	Professors must be able to define a total capacity for how many projects they intend to take on, though this should not limit applications or offers.	Low
FR-PV-14	Professors must be able to view and edit the tags on their personal profile pages.	Medium

FR-PV-6 was ultimately not attained, as it would have required changes to the structure of the database that we did not have time to make towards the end of development. Various other requirements from Professor Views got moved to the Thesis Proposals requirements section.

6.4.5 Automatic Processing

Table 6.5: Revised requirements regarding automatic processing.

ID	Requirement	Priority
FR-AP-2	Upon accepting a final offer, the student's other applications must be marked as retracted or declined.	High
FR-AP-3	When a professor extends a final offer to a student who has preemptively accepted that project, the offer is accepted immediately.	High
FR-AP-5	When a professor closes a project, such as due to the capacity being met, any remaining applications to that project are rejected, and lower priority applications are shifted up.	Medium

We added **FR-AP-5** as an equivalent to **FR-AP-2** from the professor's side of the application. Just as we wanted to reduce repetitive work for students by automating retractions, we chose to automate rejections once a project is closed to eliminate that work for professors.

6.4.6 Filtering

Table 6.6: Revised requirements regarding filtering.

ID	Requirement	Priority
FR-FL-1	Tags must exist for prerequisite classes, specialization or subject, topics of interest, and method of work.	High
FR-FL-2	The list of all projects must support filtering by the presence or absence of tags, and by assigned professor.	High
FR-FL-3	By default, students see a list of all projects which are relevant to their personal and academic tags.	High

We moved **FR-FL-4** and **FR-FL-5**, which defined filtering of thesis proposals for professors, to the Thesis Proposals requirements section, and therefore did not complete them. The remaining filtering requirements are unchanged.

6.4.7 Thesis Proposals

Table 6.7: Revised requirements regarding thesis proposals.

ID	Requirement	Priority
FR-TP-3	Students must be able to submit proposal for theses. These require a title, description, and tags, with the option to suggest a professor.	High
FR-TP-4	Students must be able to edit their thesis proposals. Copies of proposals will not be edited.	Medium
FR-TP-5	Students must be able to retract thesis proposals, removing them from view completely. This is not possible for accepted proposals. Copies of proposals will not be removed.	Low
FR-TP-6	Professors must be able to view projects proposed to them and coordinate with the proposer.	High
FR-TP-7	Accepted project proposals must automatically populate into the system. For student proposals, this should create a new project and automatically assign the student and professor to it.	High
FR-PV-9	Professors must be able to view all student thesis proposals.	High
FR-PV-10	Professors must be able to view student thesis proposals assigned to them separately.	Medium
FR-PV-11	Professors must be able to decline student thesis proposals assigned to them, which removes them from the proposal. The proposal remains in the view of all student proposals.	Low
FR-PV-12	Professors must be able to accept student thesis proposals.	High
FR-PV-13	Professors must be able to copy a student thesis proposal into a draft for editing.	Low
FR-FL-4	The list of all student thesis proposals must support filtering by the presence of tags, and by assigned professor.	High
FR-FL-5	By default, professors see a list of all thesis proposals which are relevant to their personal tags.	Medium

As noted above, although we did not implement thesis proposal features in our prototype due to time constraints, we included these requirements in this re-ranking. We chose to do this because we believed that they were important, and we still intended to implement them when we reached this phase of development.

6.5 Design Decisions

This section explains some of the decisions we made about the specific implementation of various system features. These decisions were made according to our design principles and personas when possible. These decisions were details that we did not consider in our initial design, and had to make during implementation.

6.5.1 Group Applications

Security

Group applications require a lot of logic to prevent misuse from bad actors. Without proper safeguards, a bad actor could sign another student up for a group project that that student had no interest in without their knowledge. If that application were accepted, it would rescind all of the student's other applications and forcibly assign them to that project. The student would have to get that error corrected by an administrator and start over on the application process. Alternatively, a bad actor could flood another student's applications list with hundreds of false group applications to a huge number of projects, making it very difficult for them to manage their applications.

The implementation we settled on in our prototype did not have sufficient security features to prevent these attacks. Due to our time limitations, we only implemented the most basic safety feature, an option for group members to leave a group. This prevents the first kind of attack (malicious application acceptance) if the targeted user notices the application prior to its acceptance and removes themselves from the group. It does not do anything to prevent the second kind of attack (application spam). Our recommendation for a full-scale implementation of this system would be to require students to *invite* others to group applications rather than allowing them to be added without their express consent. Similarly, final acceptance of a project offer should require each student to accept or reject the offer, rather than just the owner of the application.

Ownership

As written, group applications in our system only have one owner. The owner is the person who creates the application. That student is the only one able to rescind the application or accept offers on it, and the only one who can set its priority. Group applications appear in group members' application lists, but do not show up as options in the priority list. The only interaction a non-owner group member can take with a group application is to remove themselves from the group application.

This ownership model was chosen for ease of implementation, but we do not believe it to be the best possible version of the group application system. Ideally, there should not be an individual owner for group applications. A group application should require each student to individually accept a professor's offer. Addi-

tionally, it should either have one priority per student, or take up the same priority slot for each student as a shared stake. Students should still be able to leave group applications, but no student should have the option to rescind the group's application. If every student but one leaves the group application, it should then be treated as a single application with the remaining student as its owner. We believe that this model would be much more intuitive for users based on our tests.

6.5.2 AJAX

AJAX, which is short for Asynchronous JavaScript and XML, is a web programming paradigm whereby a web page sends asynchronous requests to a server and receives data back rather than a new page to render. It then uses this data to update or manipulate the state of the page without reloading it. This is a very common technique for implementing web applications, as it allows for the interface to be updated without the interruption of reloading the whole page.

Despite its usefulness, we opted not to use AJAX as a main pattern in our web app. The AJAX pattern requires the client to handle displaying whatever information it receives itself, and we found it much faster to set up pages using server-side templates than by building components in JavaScript. Thus we chose to implement all of our data-based pages using server-side rendering. This let us build out complex pages which displayed a lot of data much more quickly, but meant that interactions like filtering and setting application priorities couldn't be instantaneous; instead, they required users to press a "submit" button which then fully reloaded the page to reflect their changes. This is not a common pattern in modern web applications, so some testers found it confusing. Ultimately, we feel that choosing not to use the AJAX pattern for rendering was the right choice for our prototype, but a larger-scale version of this system might benefit from including it.

We used AJAX in one place, validating usernames in the group application interface. In order to provide feedback on whether or not a username was valid without reloading the page, our only options were AJAX or including every username in the page source. Publicising the full list of student names and usernames was not acceptable to us from a privacy standpoint, so we set up an AJAX endpoint for validating username inputs instead. This was somewhat challenging to do with Django, as its cross-site scripting attack prevention required us to pass an authorization token into the JavaScript through the HTML templating system. However, the resulting code functioned very well, so it may indicate that we should have used AJAX more widely.

6.5.3 Profiles

Another privacy concern for our design was the visibility of user profiles. We chose to keep profiles very sparse and not include any personal details beyond name, email, and a set of personal tags chosen by the user. Still, those tags included academic history which we did not feel should be exposed to anyone who visits a

page. Thus we chose to require users to be logged in to view profiles. Furthermore, we did not want students to be able to view the academic history of other students, so we restricted student accounts to only be able to view professor profiles. We allowed professors to view the profile of any student, however. While we believe that this implementation struck a good balance of personal privacy and convenience, we believe that it would have been beneficial to allow students to view the profiles of students with whom they shared one or more group applications. A complete implementation of this system may want to include that detail.

6.5.4 Automation

Choosing what to automate in this system was a challenging process. When dealing with something as important as project assignments which determine a student's entire final year of education, we did not want to have any automated behaviors that would surprise users with unexpected results. We felt that most things in this system should be very carefully considered, and thus require manual execution. We did not automate matching or applying because of this. What we did choose to automate were user tasks we identified within the current system that were seen as repetitive and annoying.

Specifically, we chose to automate the withdrawal of other applications when a student accepts a project offer. In the IDI system, students would have to email each professor they had previously sent an application to and tell them that they are no longer pursuing that project. This was tedious and meant students might choose not to announce it, leaving professors in the dark about their interest. Thus, we set up automation in our system to withdraw all of a student's outstanding applications when they accept an offer.

Similarly, we chose to automatically reject outstanding applications to a project when it is closed. When a project is closed, it is considered inactive either because its capacity has been filled or because the professor has cancelled the project. Either way, in the IDI system the professor would need to inform applicants of this change one at a time. We automated this behavior because there was no benefit to doing it manually. As an extension of this, we made rejected applications automatically revoke their priority status, and shifted the priority of other projects accordingly. This means the priority of remaining applications should increase as the students options are limited, giving professors a proxy for judging if a student is especially in need of an offer.

Our last automation decision was to add a way for students to instantly accept an offer for a project. In our interviews, we found that it was somewhat common for students to discuss projects with professors prior to the projects being listed, and thus only went through the project listing system as a formality. In order to facilitate those students with pre-existing arrangements, we added a feature to allow students to instantly accept that professor's offer. This system also benefits students who are in a rush to get a project assignment, as they can set all of their applications to instantly accept the first offer they get, as well as students who are

sure of their priorities or indifferent about which project they get. In combination with the automatic withdrawal, this should narrow down the choices of professors faster, as applications are retracted without delay.

6.5.5 Tags

Custom Tags

When designing the tag system, we chose to disallow users from creating their own custom tags for projects or profiles. We made this decision to simplify the filtering experience and reduce confusion for students trying to find projects. New tags have to be added via the administrative pages. We believe that if users are able to create custom tags, the number of tags will increase significantly, duplicate tags will appear, and each individual tag will increase in specificity as users create the most perfect and accurate tag for their individual need. Over time, this makes filtering and searching for projects increasingly painful. For example, without some central authority to wrangle the system's set of tags, a student interested in machine learning projects could end up having to check tags like "deep learning", "convolutional neural networks", "generative adversarial networks", and dozens more specific subtypes of machine learning just to get a view of all of the available machine learning projects. This is further complicated if professors generate duplicative tags, such as tagging a project "CNNs" when the tag "convolutional neural networks" already exists. The interface for navigating this much larger set of tags would also need to be much more dynamic and complex in order to make it reasonable to find specific items from the resulting glut of information.

Systems to mitigate some of these issues do exist. The machine learning example can be solved by tag implication, which is the process of mapping a tag to some set of additional tags which must also apply if that first tag is true. By having all of those specific machine learning subtypes imply the "machine learning" tag, we would be able to retain the ability to quickly filter those projects even as professors became more specific with their tagging. Additionally, the issue of duplicative tags can be solved with tag aliasing, wherein equivalent tags like "convolutional neural networks" and "CNNs" are automatically resolved to one term or the other. This allows users to input useful shorthands or alternate spellings without actually creating confusing duplicate tags, and helps control the overall tag count at the same time. Finally, there is an established paradigm to solve the interface issue of finding and selecting tags from large sets, namely a search with auto-complete suggestions. With all of these systems in place, we could have allowed users to specify custom tags. However, with even a single one of them missing, we felt that custom tags would grow out of control. Because we did not have time to implement these systems, we chose to restrict custom tagging instead.

While the systems proposed above do make custom tags significantly more viable, it is worth noting that the tag aliasing and implication features require a system administrator to keep track of every new tag added each day and set up appropriate aliases and implications. This is a time-consuming task which requires

extensive knowledge of the field to correctly identify redundant tags and know what broader tags a specific tag should imply. Large-scale systems with custom tags use entire teams of volunteers or employees to manage their tags and maintain a clean working state. Any future implementations which are considering allowing custom tags should be aware of these costs when making that decision.

Tag Types

There are additional complications for custom tags that arise from the use of tags on both profiles and projects. The initial design for tags was based around using them solely for filtering projects by subject matter and prerequisites. Extending this behavior to user profiles seemed reasonable, but some tags were much harder to interpret in relation to profiles than they were for projects. When we extended the use of tags to encompass administrative metadata, like group size or collaborating third parties, we ended up with tags that did not apply to profiles at all, or which applied to students but not professors and vice versa. We then had to make a choice about whether to allow these tags on profiles, and how to hide them from the list of options if we chose to do so. Additionally, we found these administrative tags to be unintuitive to assign to projects using the tag menu, as opposed to separate menus, which would require hiding these tags from the tag menu. While we could expand our tag category model significantly to implement all of these distinctions with our pre-made set of tags, applying that level of separation to custom tags would require even more tag management work from an administrator. But without that separation, completely inapplicable custom profile tags could end up on project pages. One of our goals with the system was to avoid requiring administrators to edit the code or pages in any way, so asking them to set up lists of which tags can be used where was counter to our principles. In the end, we chose not to allocate time to this.

However, even in our first set of requirements we noted the need to separate out academic tags from preference tags. Academic tags represent a student's academic status in terms of classes and specialization, and cannot be edited by students because they are intended to be a true academic record to help professors assess whether the student has the prerequisites necessary to work on a project. In order to enable this, we added an "academic" Boolean field to our tag category model. This property was implemented before the other considerations mentioned above.

Tagging Interface

The tagging interface we chose to implement was a dropdown which allowed users to select from available tags. The selected tags would then appear in colored bubbles below the dropdown, which could be clicked to remove the tag. This interface was a compromise, as we did not have sufficient time to dedicate to finding and implementing a tried and tested design paradigm for tag selection. We eventually implemented a text field with value checking for adding group members,

which we believe to be a common design for tagging systems, and ideally we would have employed this for our tags too. Another consideration was to show a menu of all tags, separated by category, and allow professors to toggle them on and off like a switchboard. This would likely have become overwhelming in practice, when a full set of tags are defined. Our drop-down is similarly infeasible, as the length of the dropdown list increases linearly with the number of tags, and we only tested it with far fewer tags than we expect a full system to use.

Tag Colors

Throughout our interface, we use colored tag bubbles to help users quickly see which tags are from which tag category. However, color-coding is not accessible to users with various forms of colorblindness. While we intentionally chose our colors to be distinct even to most colorblind users, we also designed a number of fallback systems to help users distinguish tag groups without color. Firstly, all tags display their tag group as alt-text when hovered with the mouse. In addition, we always display tags sorted by category, and often group them under headers that display their category names. Lastly, we implemented these colors using clearly named css classes that can be easily overridden by users in their browsers to fit their personal needs.

6.5.6 Filter Totals

One nice feature of the IDI system's interface is the listing counts that it provides. After each filter option it displays a parenthetical number indicating the number of listings that meet that option. We wanted to replicate this in our prototype, however we found it difficult to do in a computationally efficient manner without significantly restructuring our database.

6.5.7 Ranking Interface

As described in section 2.2, the concept of project rankings already exists in the IDI system. We chose to keep the functionality of rankings largely the same in our implementation, with our only notable change being that we tied them to individual applications instead of just projects. As a result of this, we did not make it our goal to improve the ranking interface. We instead settled on a simple interface which was quick to implement. We present users with a ranked list of dropdown menus populated with the names of projects they have applied to. Users can then select titles at the appropriate list slot to rank them there. If a title already listed in another rank is selected, that title is moved up to that slot, leaving its old position empty. The other slots are unaffected.

This interface was received poorly in chapter 8's user tests. Users were confused by the re-ranking behavior as well as the additional unlabeled ranking slots past five which we provided. Additionally, our choice to use a "submit" button instead of updating priorities in real time confused some users. Based on our feed-

back and a survey of similar ranking interfaces, we believe that the best ranking interface would have been a drag-and-drop ordering list containing every project the student had applied to. The ranking display on each application would then automatically update whenever this list was changed. We recommend that any future developers consider building this type of interface instead.

6.6 Testing and Deployment

In order to test our prototype with users, we set up an unsupervised user test as described in section 4.6, as we were not able to get in-person interviews. This test included dynamically populated test data to fill out the system, and a temporary test account system in place of permanent authorization. The implementation of these features is detailed below.

6.6.1 Test Data

To test our system, we needed a way to fill it with arbitrary test data quickly. To this end we developed a Django command that can read in data from a group of spreadsheet files and create all of the objects specified within those spreadsheets in the database. We then created a base set of test data pulled from actual projects on the IDI project listing site. We used this test data to assess how well the site functioned with a significant number of projects, tags, and applications in place. This functionality could easily be adapted to import real data exported from other services, such as user accounts and student prerequisite classes.

6.6.2 Deployment

In order to enable remote user testing, we had to deploy our application to a production server for users to access from their own devices. We chose to use Heroku for this, as it was free and had a well-documented integration path for Django applications. This required us to make a number of changes to the application in order to facilitate hosting. The version of the app with these changes can be found in the *demo* branch of our repository. While most of these changes were related to app settings and did not change anything visible to the user, there were some new systems we implemented to make testing easier.

6.6.3 Test Accounts

In order to showcase all of the features of our prototype, we needed a way to give professors and students applications and offers to interact with. However, in the time we had available to set up the demo branch, implementing a system to dynamically detect projects created by professor testers and send them applications (or give offers to applications created by student testers) was not worthwhile.

Therefore, we chose to build a system that generated test accounts already populated with projects and applications in various states by extending the test data system described above.

We removed the standard login form from our site's top bar and instead presented users with two buttons, "Get Student Account" and "Get Professor Account". When selected, these buttons generated a new account with a unique name and username. They would then populate the system with new projects or applications created by that account in states that would otherwise be impossible without some kind of interaction from another user. The data describing these projects and applications was read in and instantiated using the same method described in subsection 6.6.1. This account generation system also generated new instances of every account that was used in the demo account's test data. These instanced accounts were only visible to the matching test user. This prevented the actions of one demo user from impacting any other demo users who might have been operating on the same data at the same time. While authenticated with a test account, the "log out" button is replaced with an "end test" button, which wipes all of the test data associated with the account and returns the system to a clean state.

Chapter 7

Results

In this chapter, we present the results of our development phase. This section includes a presentation of the prototype we created as well as an analysis of its performance using the keystroke-level GOMS and business process modeling notation discussed in section 4.5 and section 4.7. The version of the application shown here is the one which we presented to users for our user tests.

7.1 Software Walkthrough

In this section we present a series of screenshots documenting the final state of the application.

Figure 7.1 shows the main page of the prototype, which lists all active projects currently in the system. The filters on the right side can be used to show or hide posts with various tags, in order to make finding projects easier. At the top of the page, professors can see a button to add a new project. This button is hidden for students. The title of each project can be clicked to visit its details page, and each professor's name links to their profile. Tags are color coded by type, with a color key in the filter pane. Additionally, each filter section can be collapsed or expanded by clicking its title.

The project creation page in Figure 7.2 can only be accessed by professors. The dropdown directly above the tags section allows professors to select tags to add to their project. Clicking on a tag in the display removes it from the project. The status dropdown lets professors choose whether the project should be open, closed, or a draft. If this page is opened by copying or editing another project, the fields are all already populated. A project can be saved by itself, or one can save it and immediately begin on a new project or a new copy, by choosing between the "Create Project" and "Create and start another" buttons.

Figure 7.3 shows a project's details page. This page shows all of the information about the project that is recorded in the system. If the user viewing the page is also the creator of the project, they see a link to edit the project, as well as a list of all of the applications students have made to the project. The professor can

accept and reject applications from this page, or add comments to them. Students will only see their own application on this page, or if they have not applied to the project, a button inviting them to create an application.

The interface for applying to projects is shown in Figure 7.4. Students can enter the usernames of other students they want to work with to create a group application. If the entered username is valid, the name of that student will show up below the entry box. Group members can be removed by clicking on their names. Selecting the checkbox enables auto-accept for the application. With auto-accept enabled, the professor's project offer will be instantly accepted, skipping the final student verification step.

Figure 7.5 shows the student view of the applications page. This page displays all of the student's applications, grouped by status. At the top of the page is the ranking interface, which lets students set priorities for their applications. The titles of all of the projects the student has applied to are listed in each drop-down, and are swapped and re-ranked as necessary when one is selected. Each application is then shown in order of status. If the the application has been given an offer from a professor, students have the option to accept or reject the offer. Otherwise, they are given the option to toggle auto-accept and a button to withdraw the application. All comments attached to an application are also shown here.

Professors see an alternate version of the applications page, shown in Figure 7.6. Here, the professor can see all of their applications, grouped by project and sorted by recency. Professors can manage their applications from this page, including sending comments and accepting and rejecting students. Priority and status are shown for each application. Additional students are also shown on group applications, and all student names can be clicked to visit the student's profile.

Alternatively, professors can get an overview of their applications on their projects page, shown in Figure 7.7. This page does not show applications in full, but groups projects by status. This page lets professors easily change the state of their projects, and see how many applications they have in a compact format. In addition, the page has a new project button, and adds a copy button to each draft and closed project, making it easy for professors to populate the database with new or old projects.

Finally, Figure 7.8 shows a profile page. Each profile contains the user's name and email, role in the system (student or professor), and a self-selected list of tags describing the user's academic history and interests. As with other tag sections in the system, tags are chosen from the dropdown and can be removed by clicking on them. A profile's owner is the only person who is able to edit it.

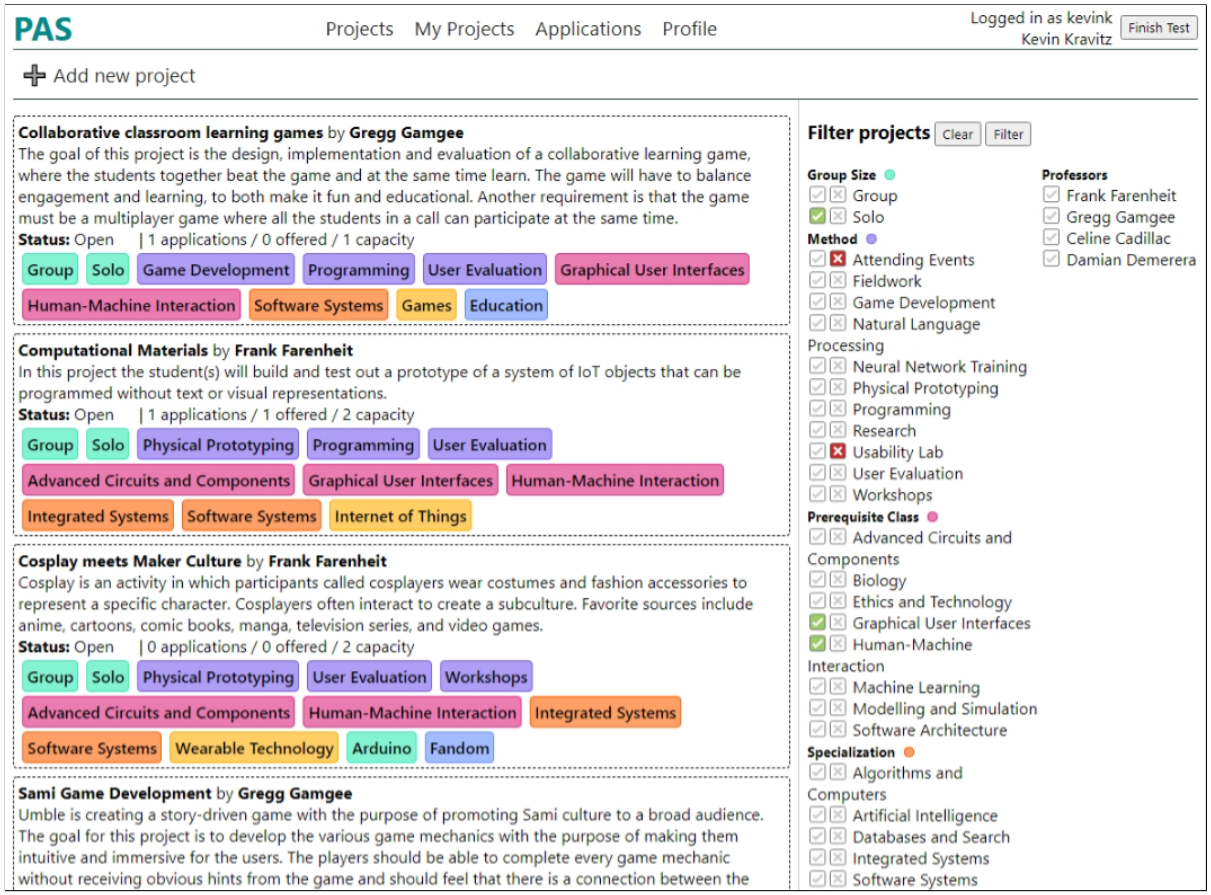

Figure 7.1: The prototype's main page. Filters can be selected to reduce the number of projects shown.

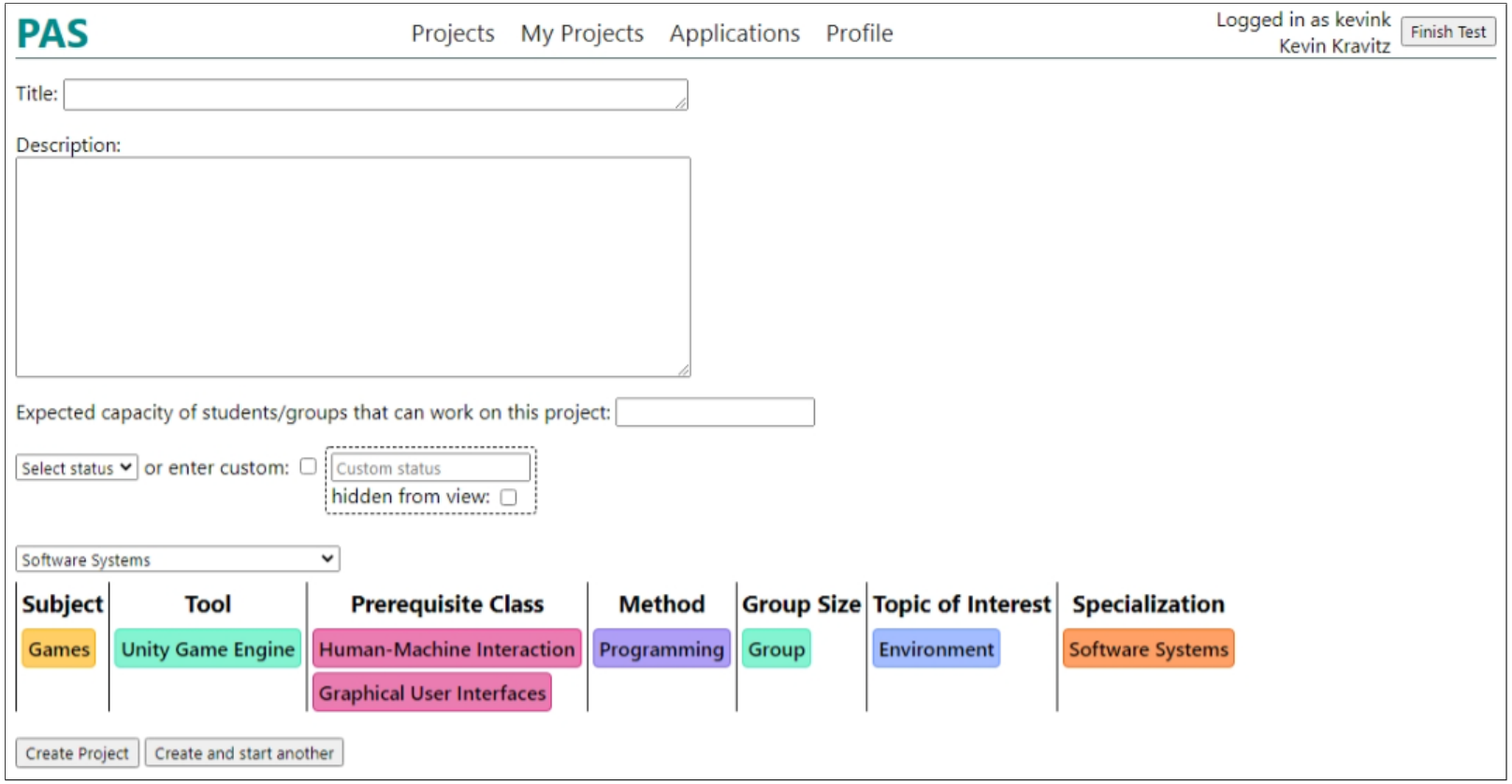

Figure 7.2: The project creation page. This page is only accessible to professors.

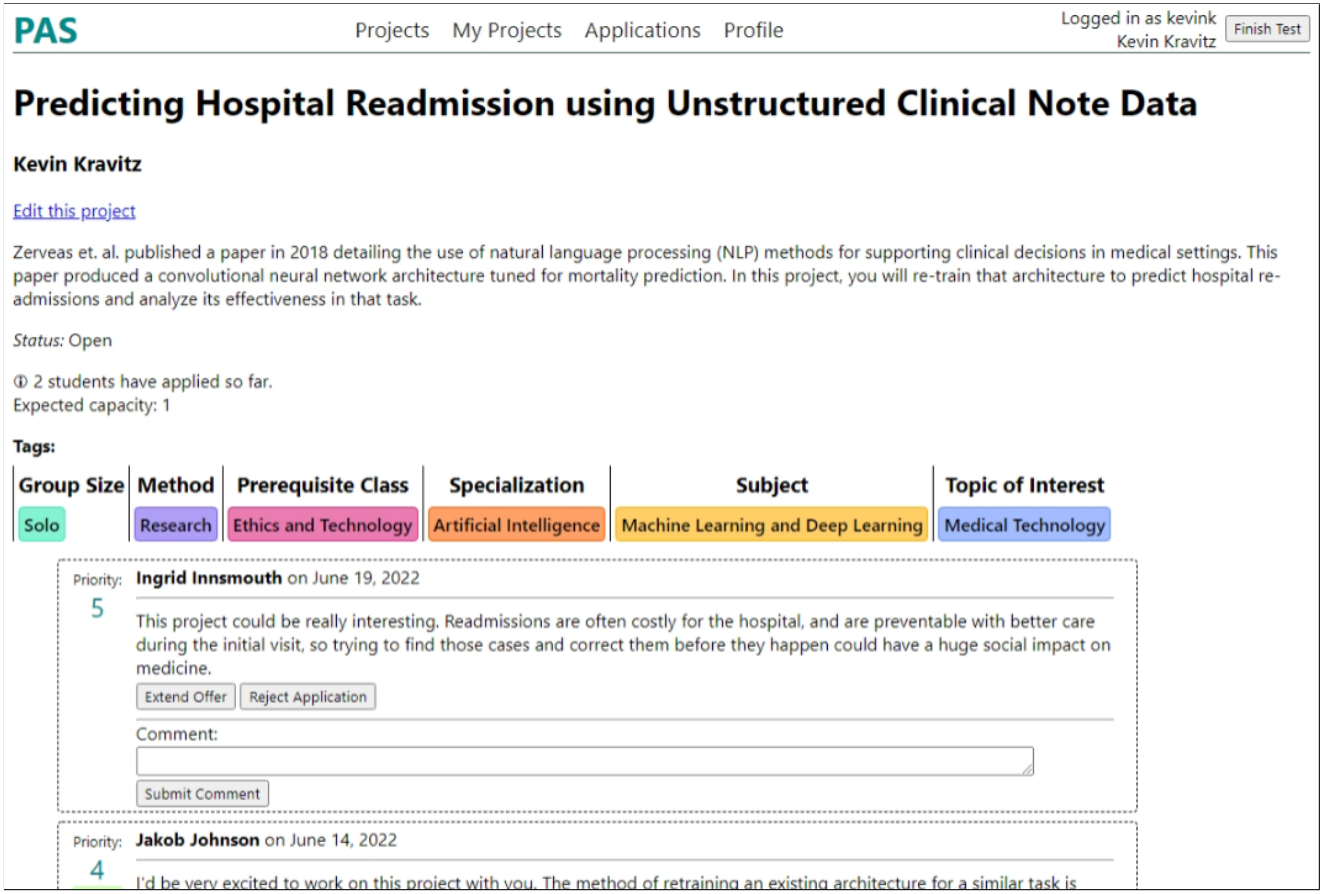

Figure 7.3: The project details page. Only the professor who owns the project can see student applications on this page.

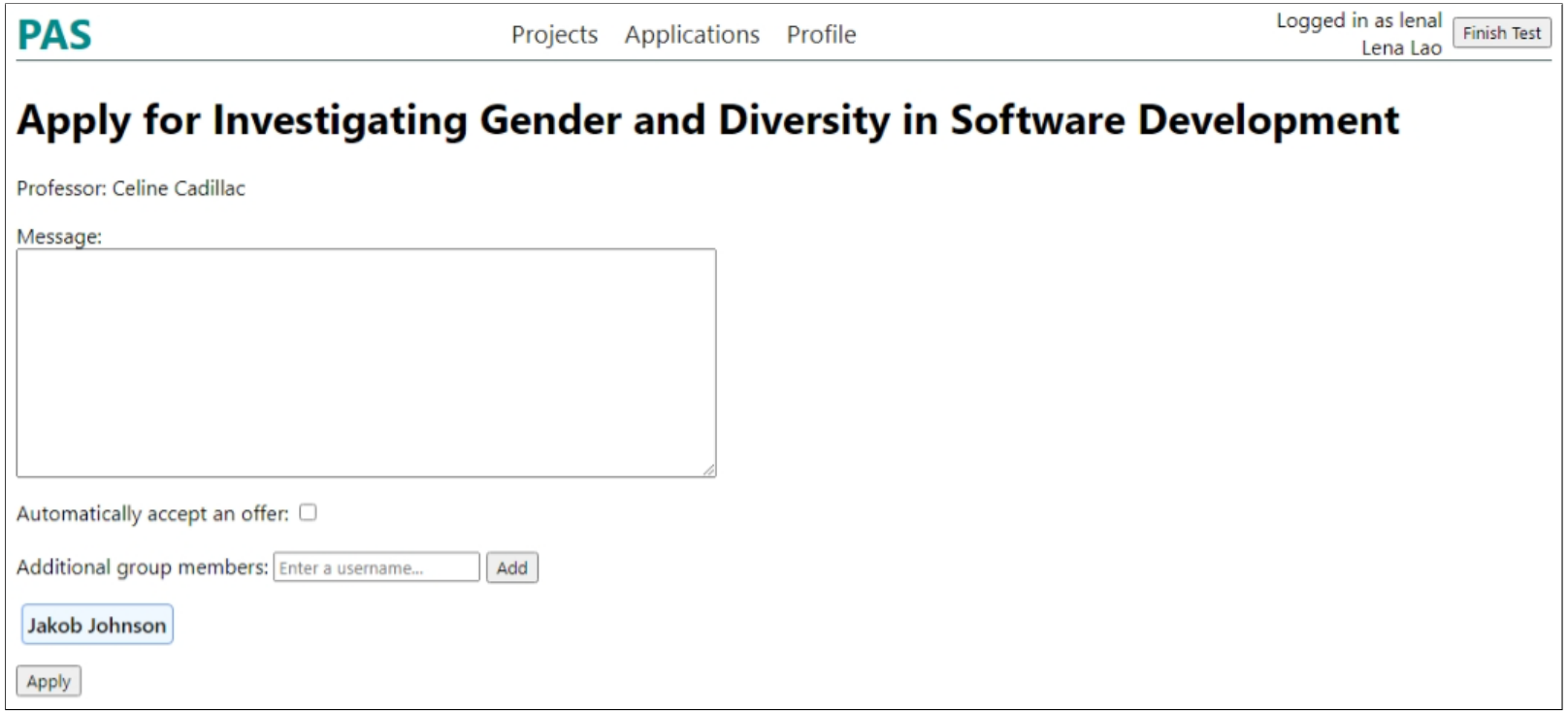

Figure 7.4: The interface for creating a new application to a project.

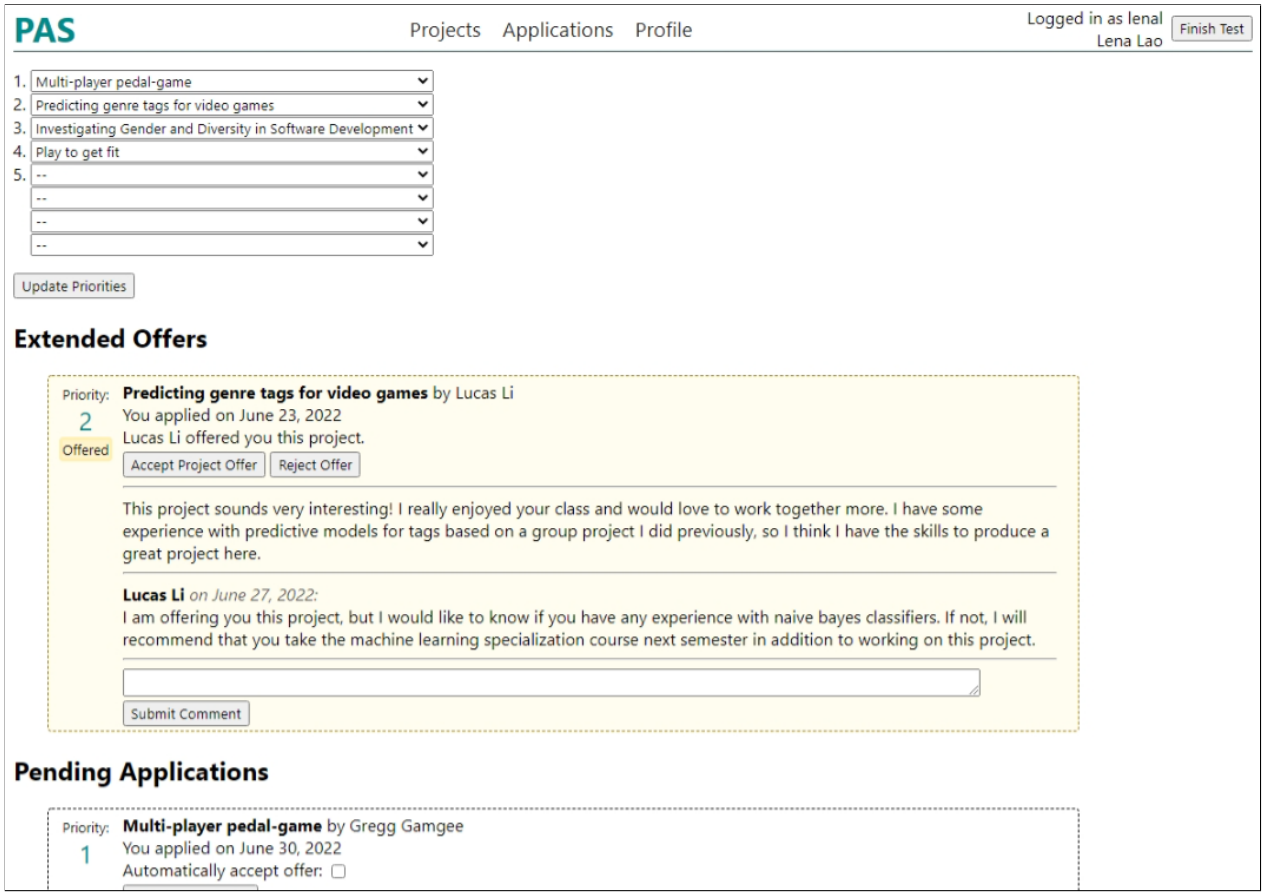

Figure 7.5: The applications page as it appears to students. All of the student's applications are shown on this page, grouped by status.

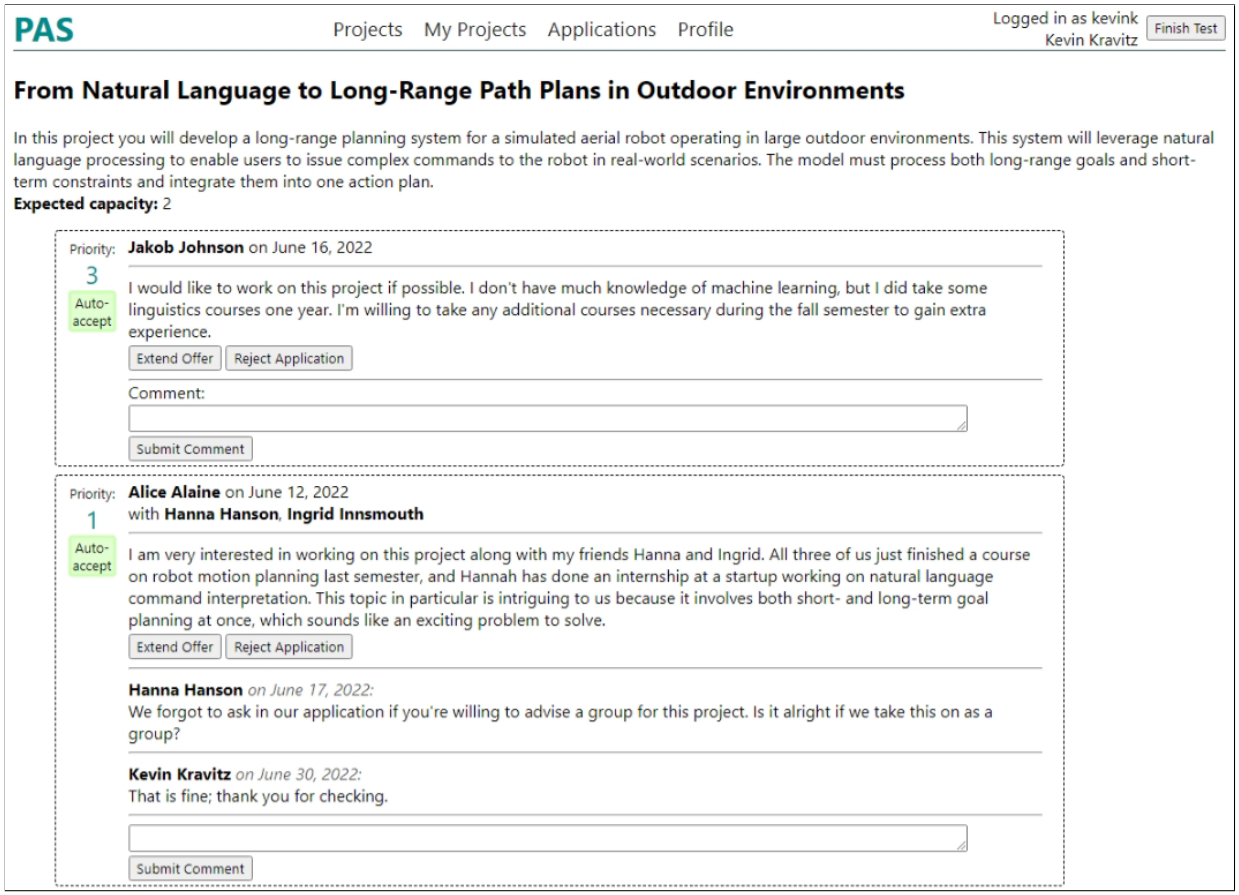

Figure 7.6: The applications page as it appears to professors. All applications to a professor's projects are shown on this page, grouped by status.

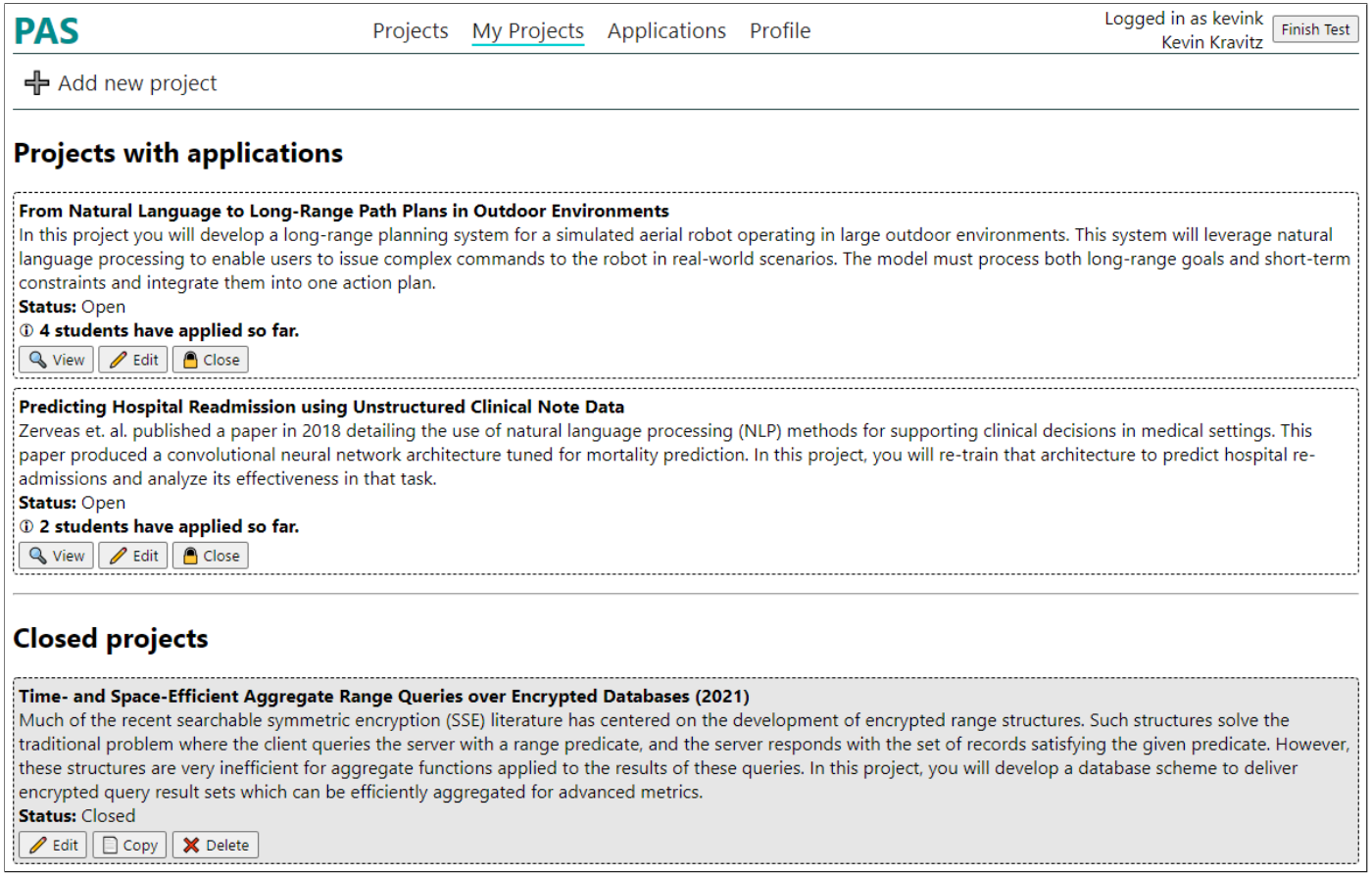

Figure 7.7: The professor projects page of our prototype.

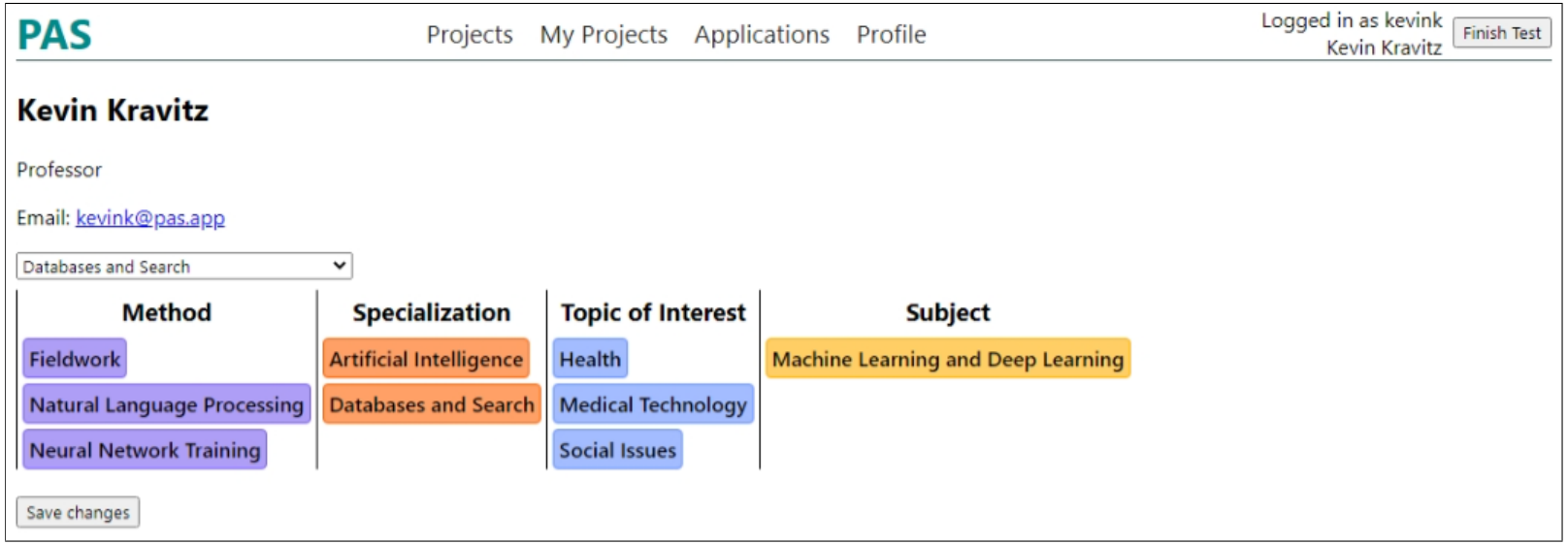

Figure 7.8: A profile page. Professor profiles are visible to all users, but student profiles can only be seen by professors.

7.2 Data Models

In addition to the frontend application, we needed to develop a data model to encode all of the system's users, projects, applications, and comments in our database. The resulting data model is shown in Figure 7.9. Due to the nature of Django, this diagram is also a class diagram of our system.

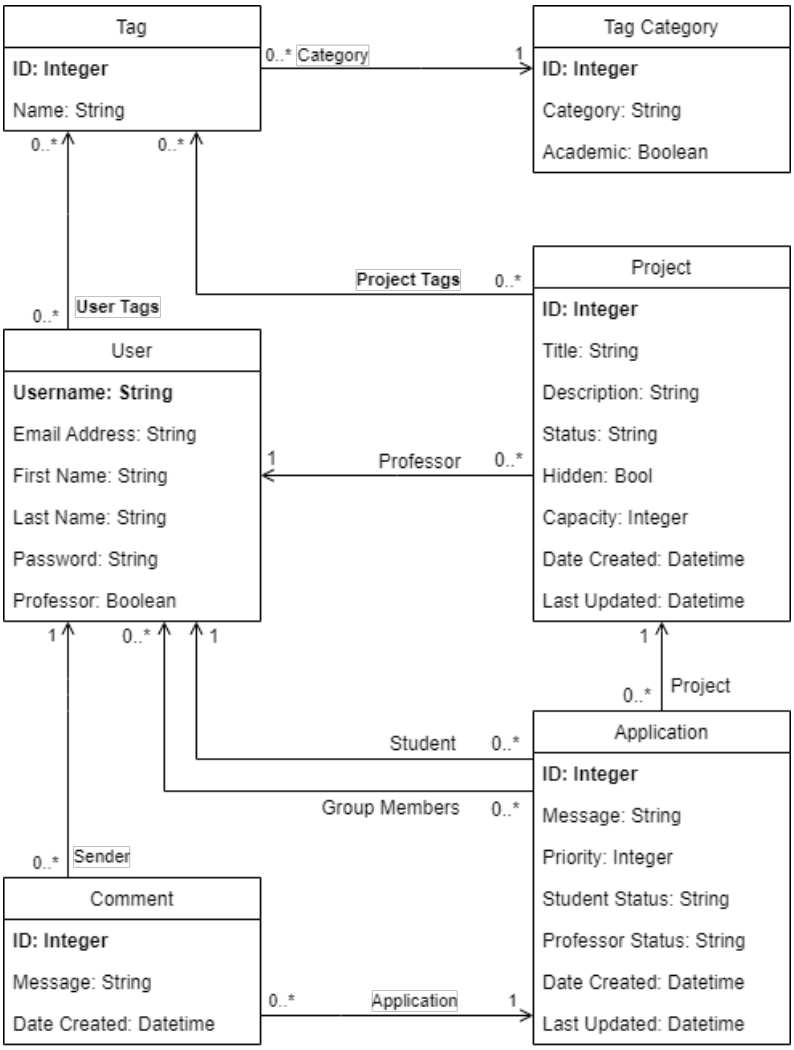

Figure 7.9: An entity relationship diagram of our database model.

We began by designing the user model. For this we leveraged Django's built-in user model, which already included all of the fields we needed and was integrated automatically with Django's authentication system. In addition to fields for a username, email, and first and last name, the model also supports assigning each user to an arbitrary number of user groups. We used these groups to divide users into students and professors. Functionally, this is a boolean value, and our

system treats it as such, so it is represented as such in the diagram.

We then developed the data model for projects. Each project is tied to one owning professor, with free text fields for professors to enter the project's title and description. We also store metadata about the project's creation date and last update time, which is used for sorting projects in some places. Projects have a status field which indicates whether they are open, closed, or drafts. This is implemented as a free text field so that it can also be set to an arbitrary text value, with the project's visibility encoded in the "hidden" Boolean field instead. We also gave projects an optional capacity field for professors to indicate how many students or groups they expect to take on for that project.

At the same time, we built out the data model for applications. Similarly to projects, each application is tied to one owning student, as well as one associated project.

Next, we set up the data structures for the tagging system. Each tag in the system has only one property, a name string, but is also associated with a named category that provides additional context for it. We group tags by category throughout the app. Additionally, projects and users can be associated with any number of tags, enabling project and profile tagging. We also added an "academic" Boolean to tag categories which prevents that category's tags from being edited on profiles when true, as described in section 6.5.5.

It should be noted that in the data diagram above, we have abridged two relational classes, UserTag and ProjectTag, which are present in the prototype. These tables simply relate a single project or user to any number of tags. This is a relation that could have been expressed more simply using Django's built-in features for one-to-many relationships, as we later did for group application members. Because these models were so simple, representing their effect inline as we did for group members was more reasonable than breaking them out into their own nodes.

Finally, we added the comment system. Each comment is simply a free text message with a sending user and a creation date. Any number of comments can be attached to an application, and they are displayed in order of creation.

Our model is simple, but sufficient, and can easily be expanded.

7.3 Process models

In this section, we present a series of process models to illustrate the similarities and differences between project matching with **no system**, the **IDI system**, and our **prototype system**. We use the process modeling system described in section 4.7 for these diagrams. All referenced diagrams are presented at the end of this section, as they are large and each require a full page.

7.3.1 Project Search

Our first comparison is the process of finding a project. Without any assisting system, as shown in Figure 7.10, administrators must first compile and release a list of project descriptions to students in some manner, at which point it is difficult to update the list with new projects. Students then read through all of these projects one by one until they find an interesting project. They must read through the entire list, as they have no way of knowing if projects later on in the list might be more interesting than the one they're currently most interested in. With the IDI system, professors enter projects directly into the database that students then access, and are able to add new projects throughout the matching process. Students then filter this list to remove any projects that aren't part of their specialization path, and then optionally filter further by professor. The same project selection process then takes place, but with a smaller list. Finally, in our prototype system, students first engage in an iterative filtering process as shown in Figure 7.12. This allows students to filter out any projects that they would not be eligible for, or that require any techniques they dislike. Once the filter conditions are acceptable, the student then browses projects in a similar manner as the other two systems.

The key process difference for finding projects is that in our prototype, students no longer need to check each project for unmet requirements, as they have already filtered those projects out during the initial filter selection step. Additionally, the number of irrelevant projects each student has to read through to make sure they are not missing an interesting topic is significantly smaller, because students are able to filter out subjects and methods that they do not care about. Both of these refinements significantly speed up the process of choosing projects for students.

7.3.2 Project Listing

Next, we examine the process of listing a project in the various systems. Figure 7.13 shows the manual implementation of this process. Professors email their project names and descriptions to an administrator, who then adds this information to a draft of the project list to email out later. With the IDI system, professors have a more complex process to list a project, though the complexity is necessary and makes a number of convenience features possible. The biggest benefit is reducing the administrative load, projects can be listed immediately without requiring the time of an administrator. This also enables professors to add new projects throughout the matching period. As shown in Figure 7.14, professors can choose to either create a new project or edit an existing one. They then fill out or edit all of the necessary fields, then publish the project immediately or save it for later. Our system replicates this process, with the only difference being the addition of tags to the project creation/editing step.

We chose to keep this process the same because we felt it was already a very minimal and expressive way for professors to control project listings. Much of the apparent complexity in this flow comes from splitting up work that already hap-

pens informally in the no-system version. In the formal systems, some additional metadata is required, and when copying other projects, each operation becomes optional. Comparatively, professors using emails to compose projects might save them as drafts or copy and paste text from other projects. The process described in Figure 7.14 simply enables that same behavior for professors who are interacting directly with a project database.

7.3.3 Application Lifecycle

The next set of charts chart describes the lifecycle of a single application. This lifecycle is similar across all three systems, but there are some differences worth modeling. Figure 7.15 shows the lifecycle of a student application without any system to assist it. In this scheme, the student contacts the professor directly and the two engage in a back-and-forth discussion process. When the negotiation is complete and the two parties agree on a project, the professor then registers this decision with an administrator who finalizes the pairing. It should be noted that although the message and response subprocesses use email nodes, they are also equivalent to in-person meetings or any other form of communication.

The main difference between this process and the IDI process modeled in Figure 7.16 is the final professor step; in the IDI system, registering the student-project match is done using the system instead of by emailing an administrator. Additionally, some key communications between student and professor such as sending project offers use the system as an intermediary instead of relying on direct emails.

Our prototype uses a similar process, as shown in Figure 7.17, but the various message events are always executed with the system as an intermediary. As a result, the negotiation and offer confirmation processes are much simplified. Additionally, state changes such as retracting interest or offering projects are done by simple buttons rather than composing emails. Our system is also able to provide automation for some steps to reduce redundant email steps; the "Accepted other project offer" event branch seen in Figure 7.16 is performed automatically, as is the "Decline student" branch when a project is closed and becomes unavailable. These are minor improvements on the scale of a single application, but since they are performed for all concurrent applications, they can save significant time and organization overall.

7.3.4 Full Matching Process

The final set of models tracks the full matching process from start to finish. In Figure 7.18, we model matching without a system to help. As mentioned above, this requires an administrator to manage the project listings and record student-project pairs. In the IDI system, shown in Figure 7.19, the administrator's role is automated via the system. However, there are still some crucial lines of communication which take place outside of the system. Specifically, these are pieces of the application lifecycle, which means that the status of an application cannot

be fully or accurately tracked. Figure 7.20 shows that in our system, all communication flows through the system, enabling better status tracking, automation, and preventing the possibility of critical peer-to-peer communication being lost in busy inboxes.

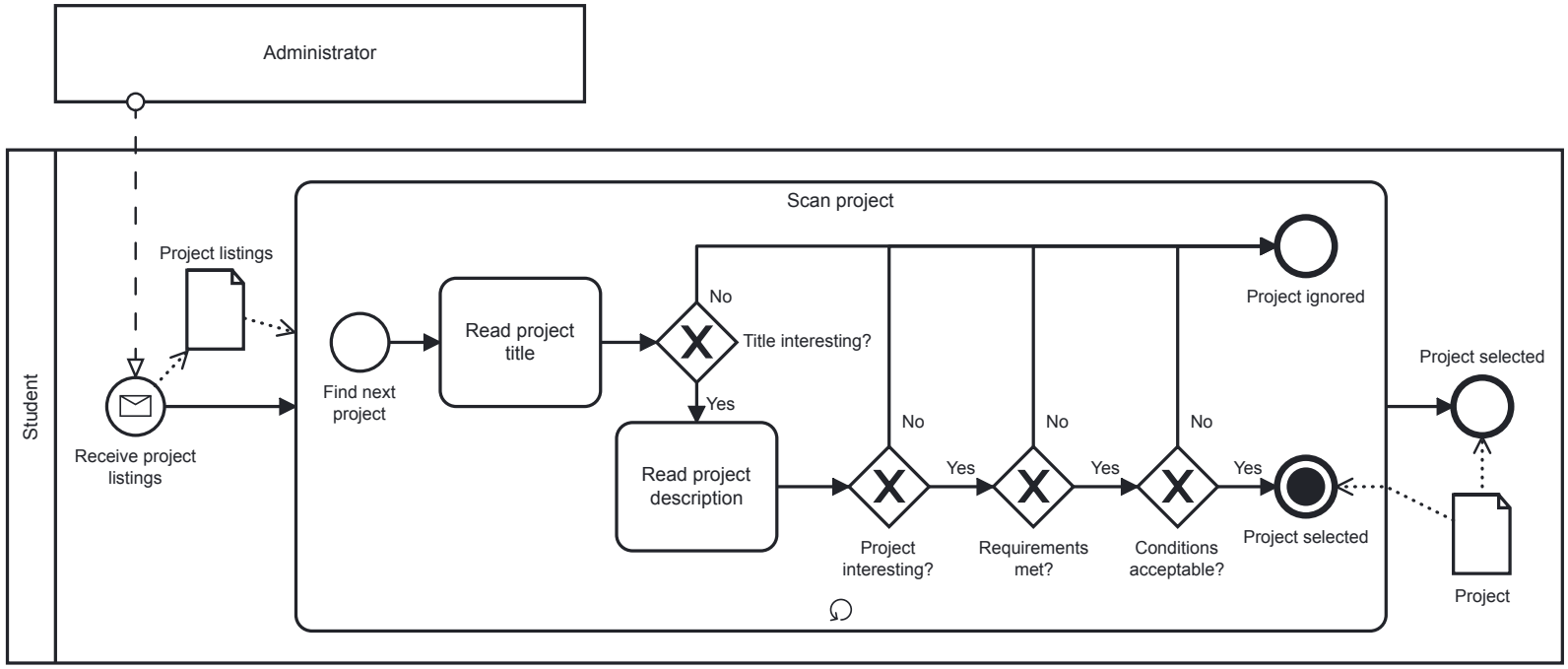

Figure 7.10: Process model for finding a project without a system

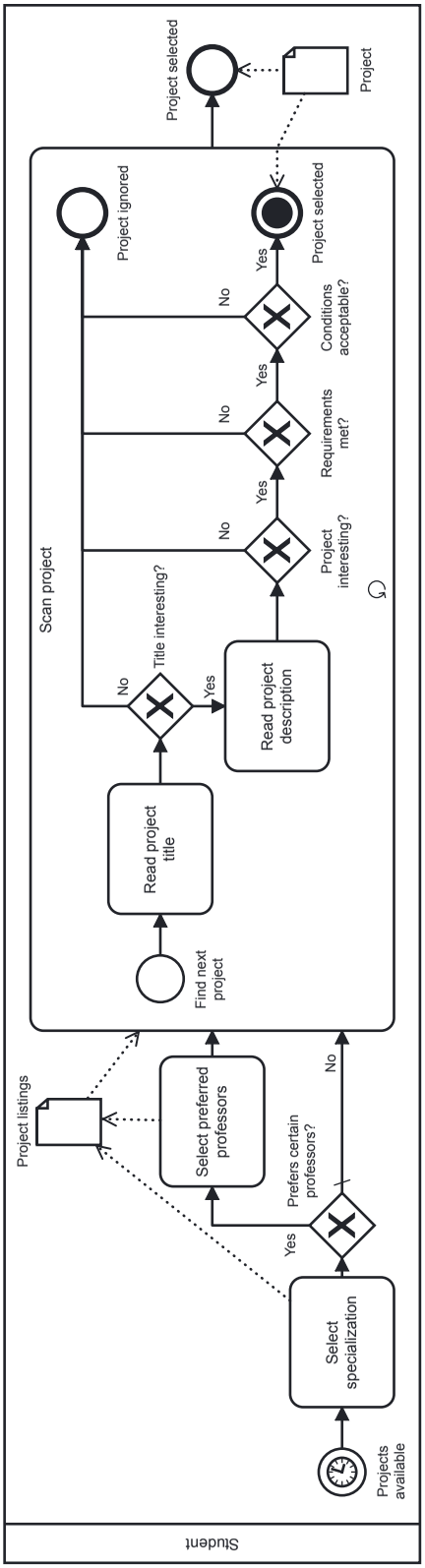

Figure 7.11: Process model for finding a project with the IDI system

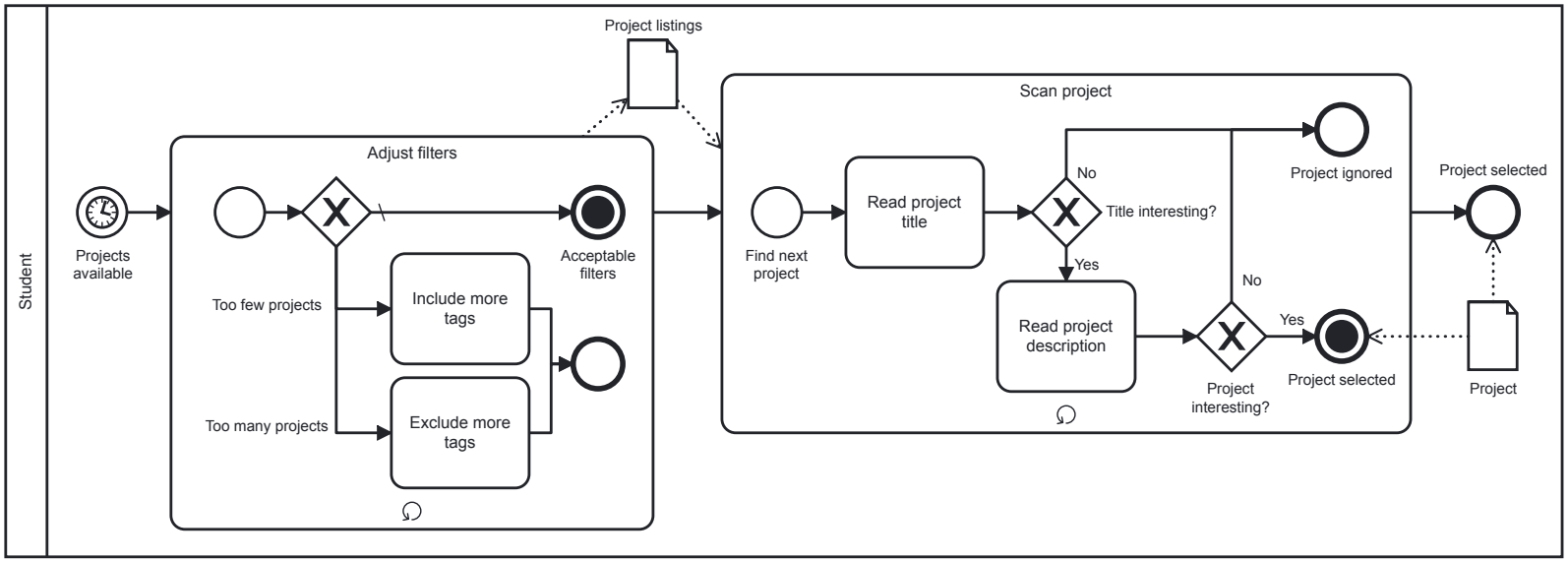

Figure 7.12: Process model for finding a project in our prototype

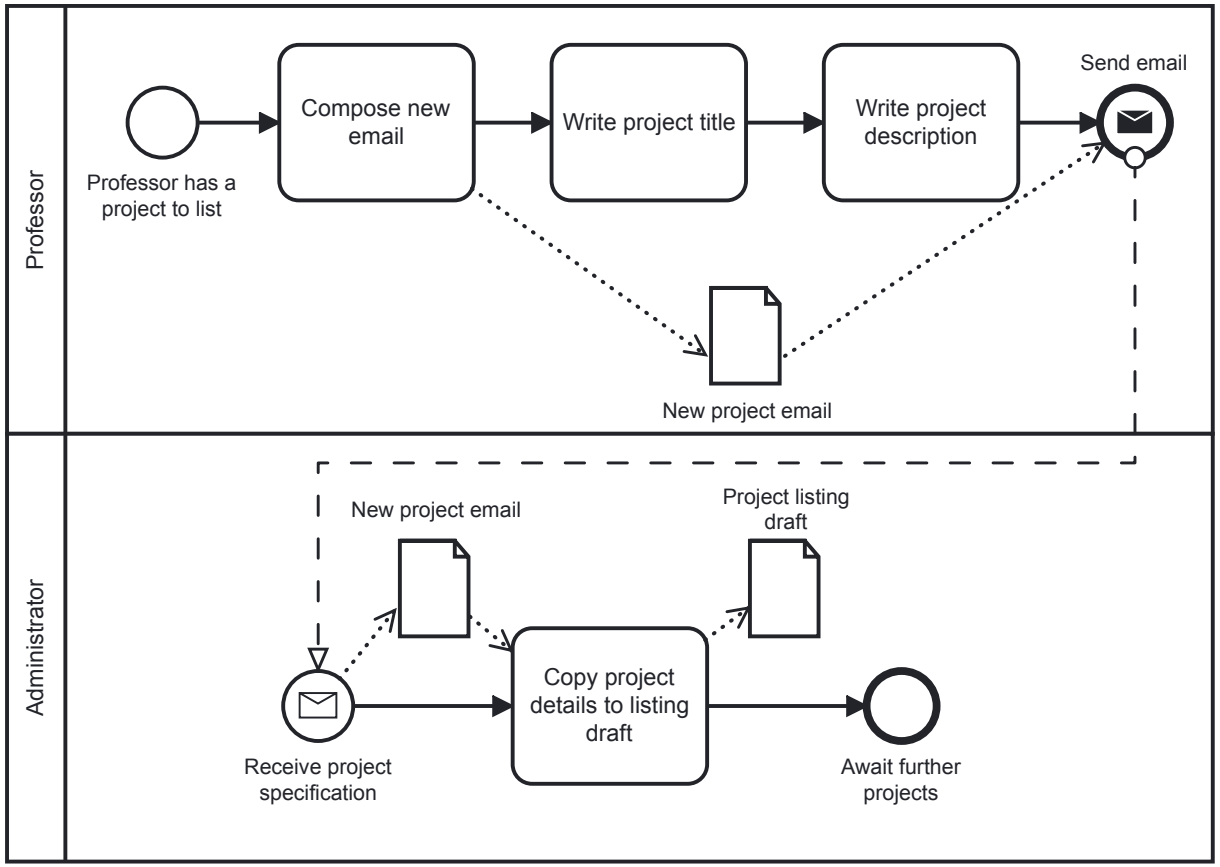

Figure 7.13: Process model for listing a project without a system

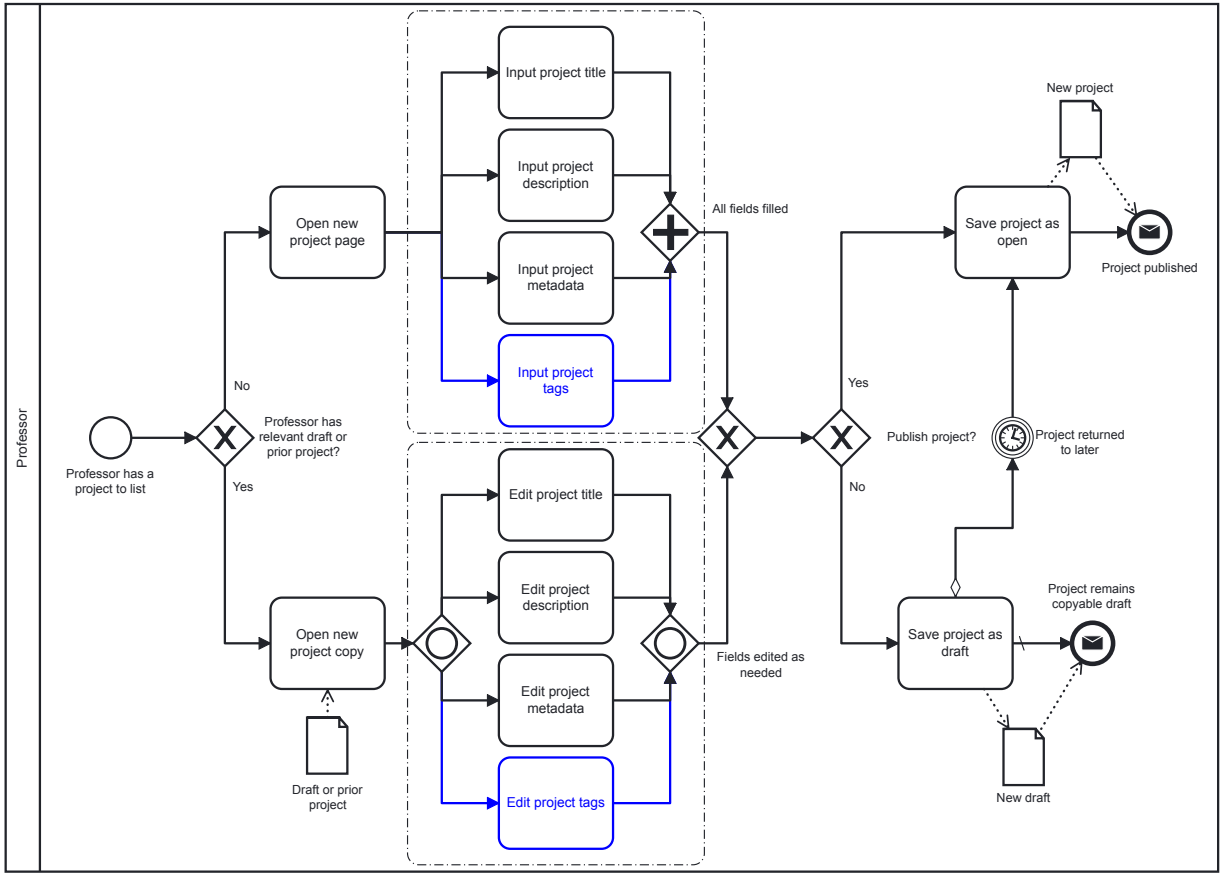

Figure 7.14: Combined process model for listing a project in both the IDI system and our prototype. Elements in blue are exclusive to our prototype.

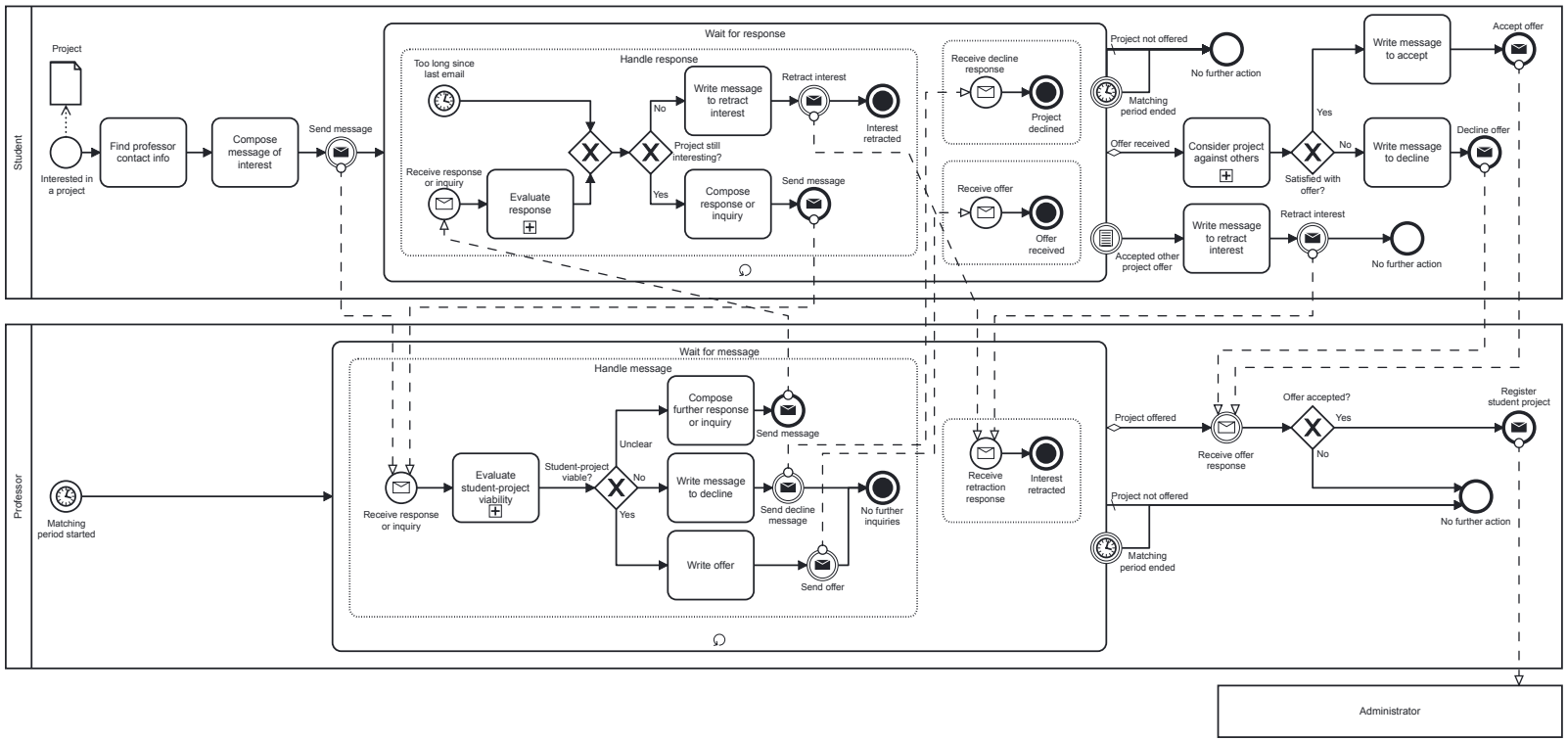

Figure 7.15: Process model for the application lifecycle with no system

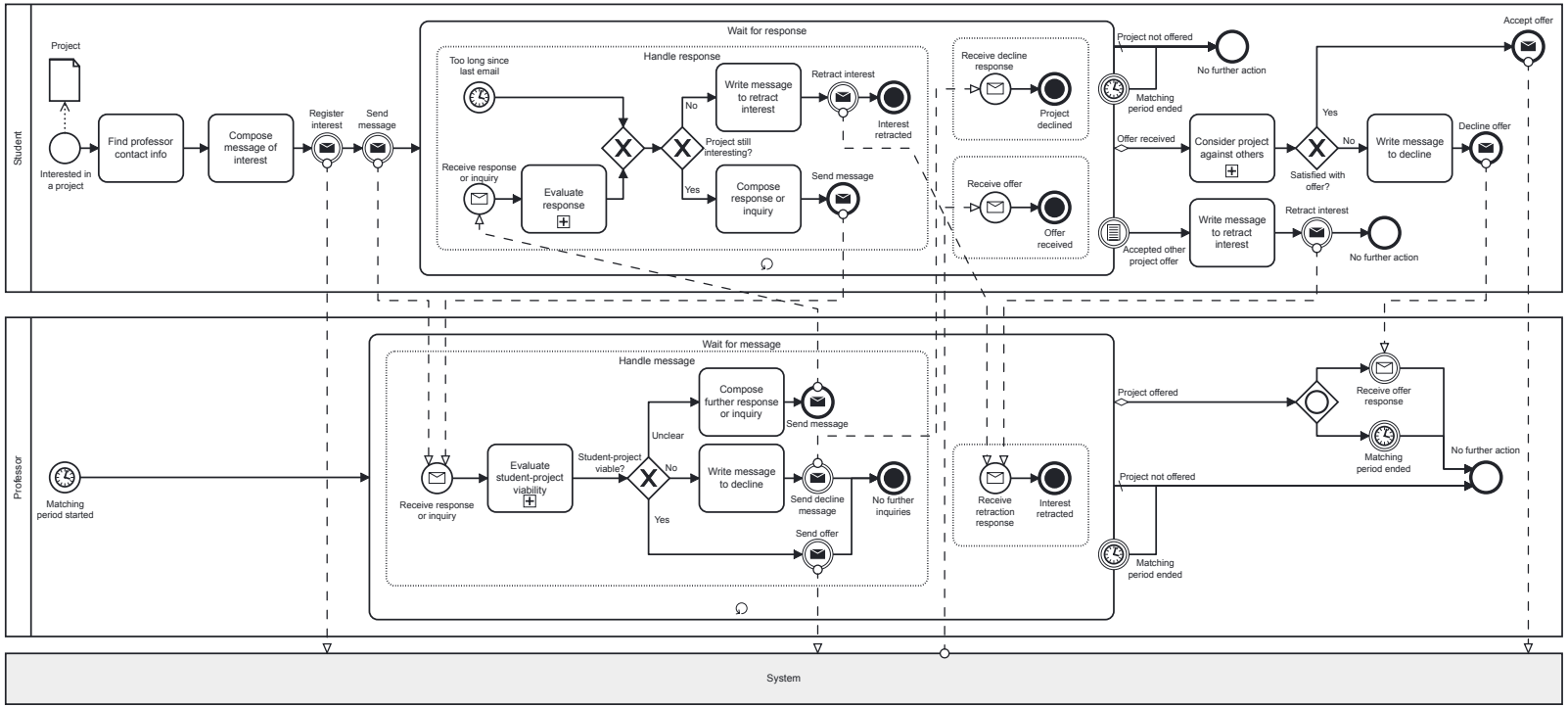

Figure 7.16: Process model for the application lifecycle in the IDI system

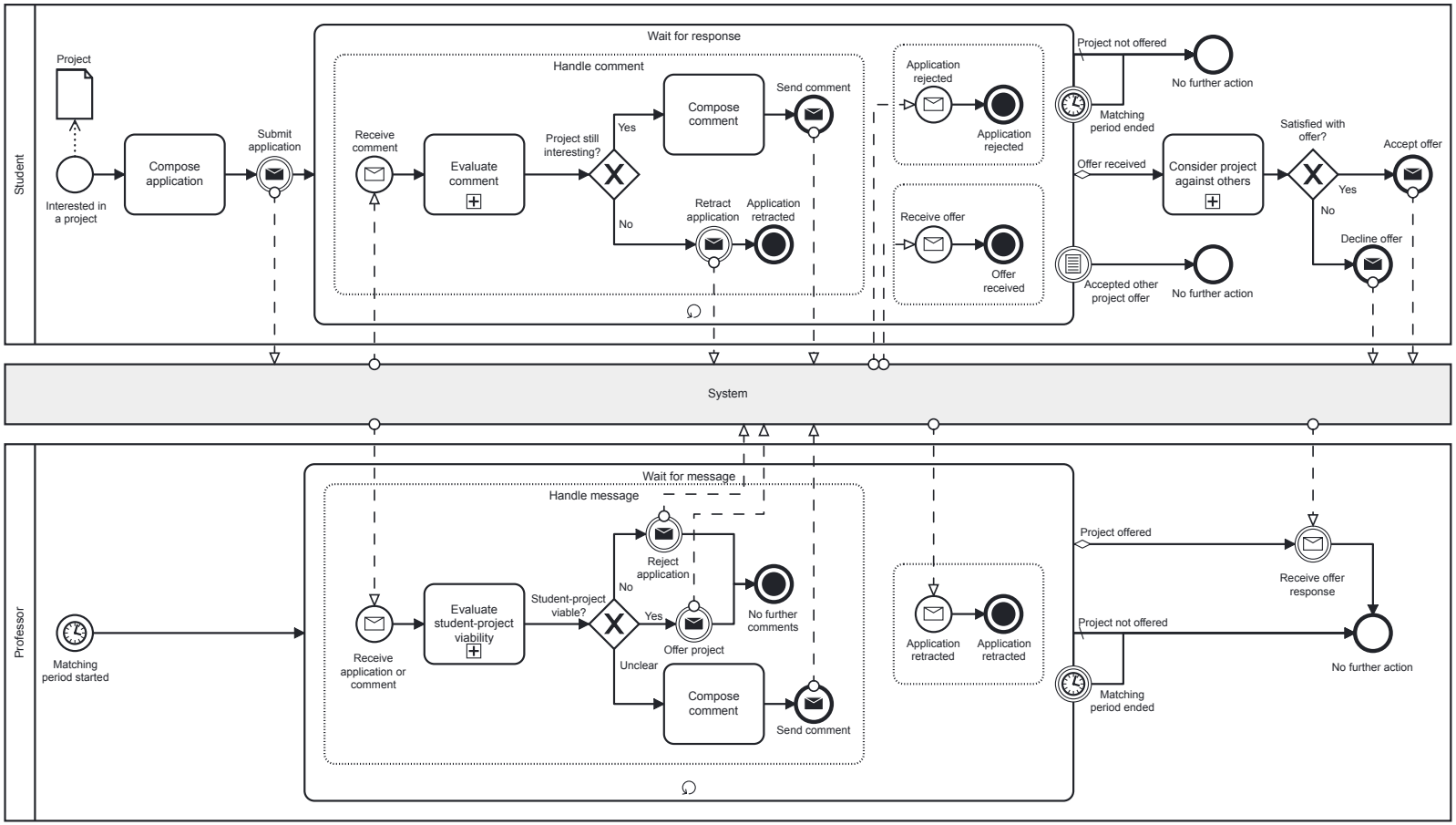

Figure 7.17: Process model for the application lifecycle in our prototype

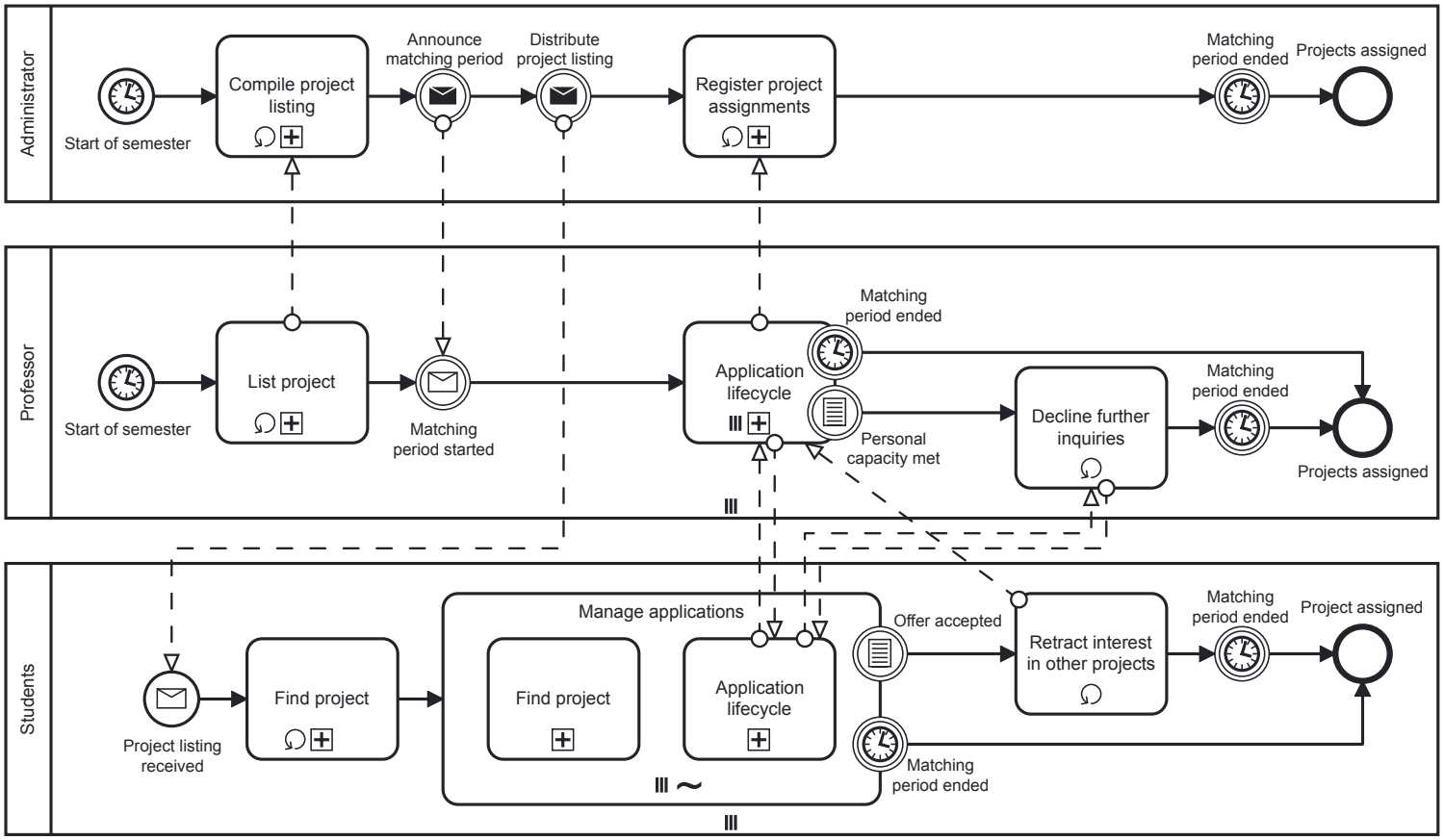

Figure 7.18: Model of the full matching process without a system

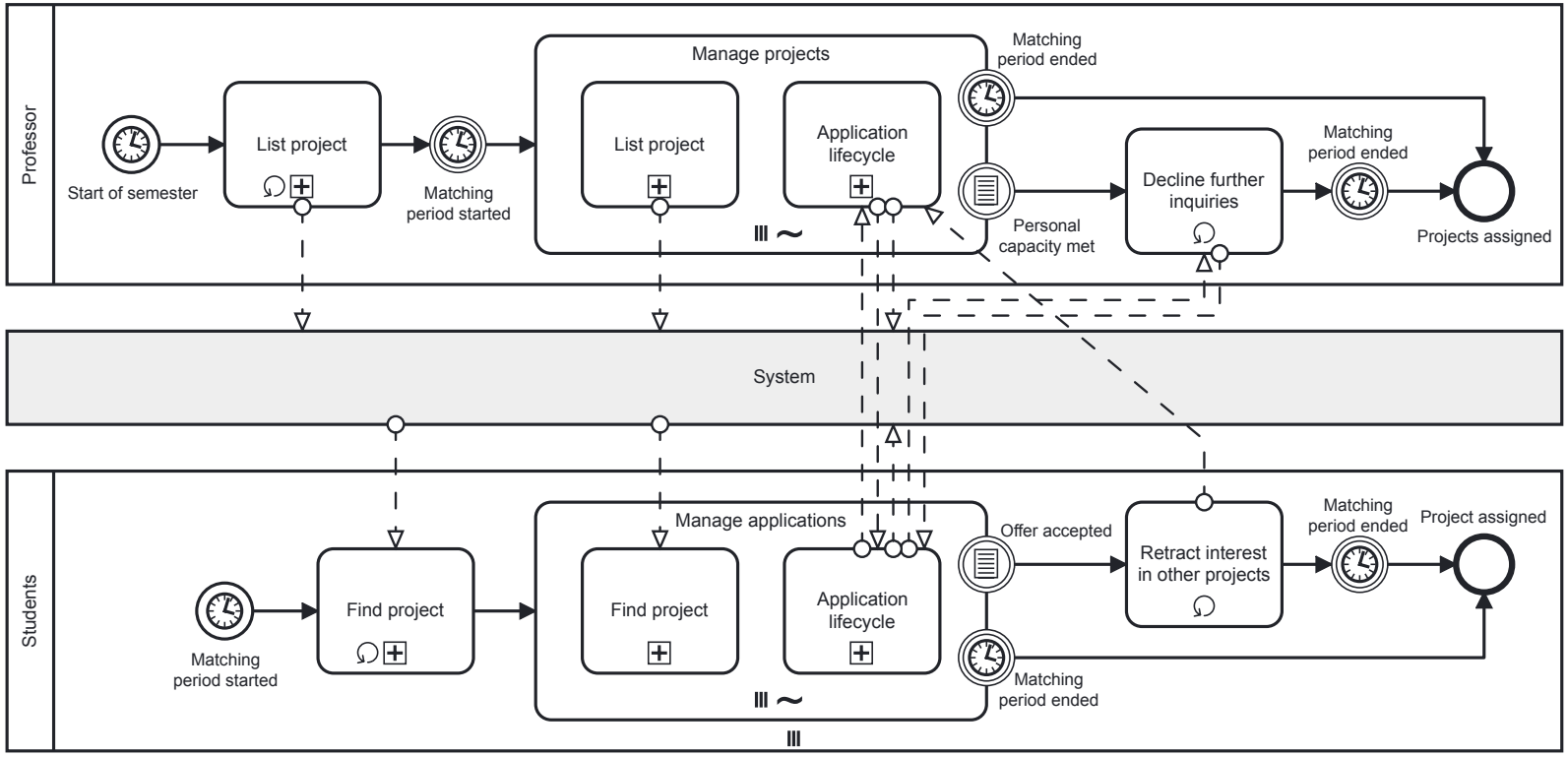

Figure 7.19: Model of the full matching process in the IDI system

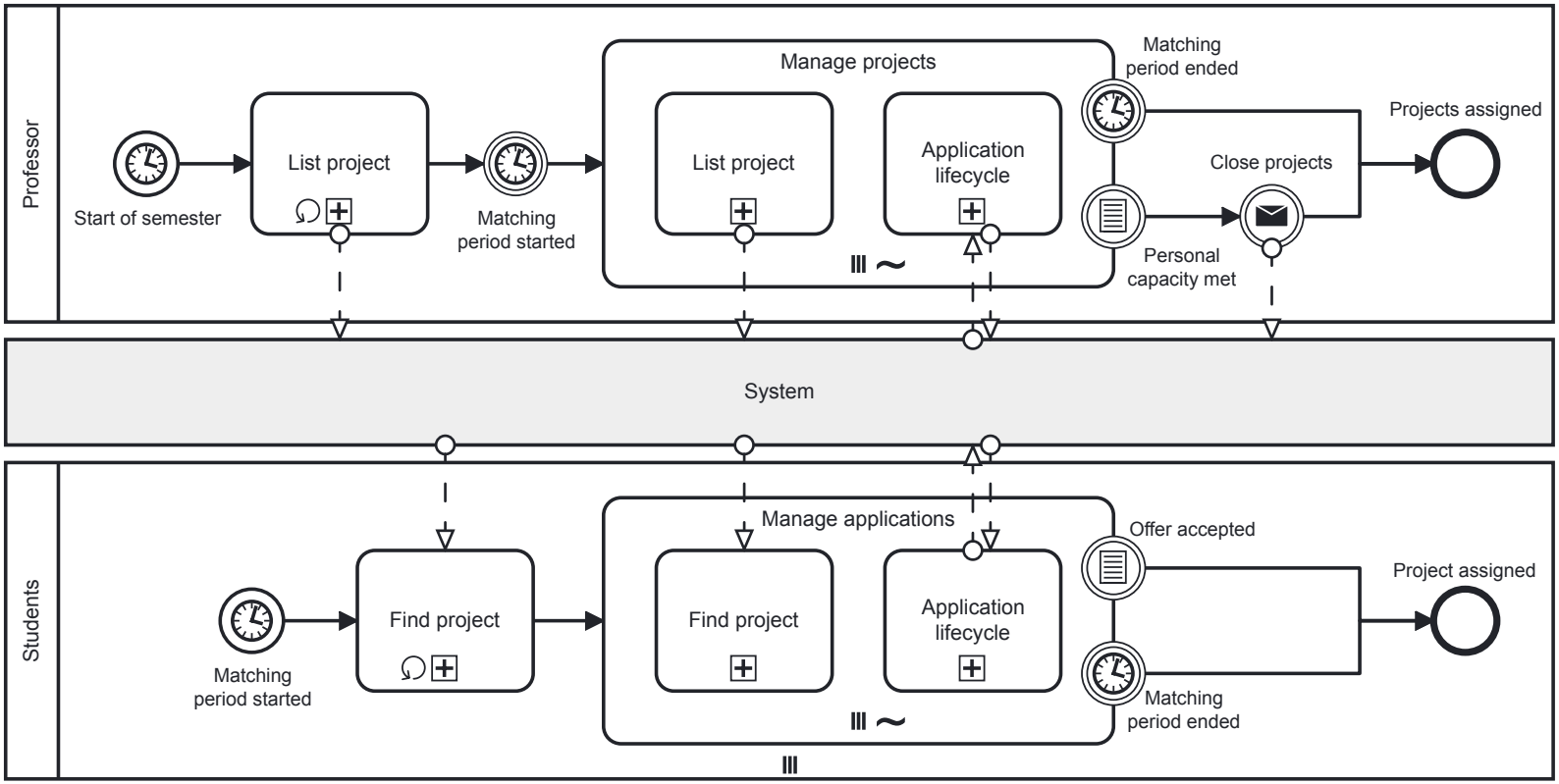

Figure 7.20: Model of the full matching process in our prototype

7.4 Time-to-Task

In this section we compare the time taken to perform certain key actions in the IDI system and our prototype using the keystroke-level model estimation framework described in section 4.5. We compare the systems on two key tasks: creating a new project and finding an interesting project. Other key flows such as applying to a project cannot be accurately compared because they take place partially or fully outside of the IDI system.

7.4.1 Project Creation

First, we look at the steps necessary to create a new project in both systems. For this task, we begin measuring on the main page of each application with the user already authenticated.

IDI System

Step	Description	Operator	Duration (sec)
1	Click the "Add Project" link	P	1.1
2	Page load time	R	0.7
3	Mentally prepare to create a project	M	1.35
4	Select the title field	P	1.1
5	Move hand from mouse to keyboard	H	0.4
6	Type a 65-character title	K	18.2
7	Move hand from keyboard to mouse	H	0.4
8	Select the description field	P	1.1
9	Move hand from mouse to keyboard	H	0.4
10	Type a 450-character description	K	126
11	Mentally prepare to add metadata	M	1.35
12	Move hand from keyboard to mouse	H	0.4
13	Select 6 relevant metadata items	P	6.6
14	Click the "Update" button	P	1.1
		Total	**160.2**

Table 7.1: Approximate task time to create a new project with the IDI system

The project entry process begins with the title and description of the project. For this estimation we assumed a 65-character title and 450-character description based on the lengths of titles and descriptions pulled from the real system. After entering this information, the user must then select a number of checkboxes and radio buttons to encode metadata about the project's status, associated specification, and group size. The resulting process time measurement was 160 seconds. Most of this time was spent typing, though a not insignificant amount is wasted moving between the mouse and keyboard to select various fields on the page. It

is worth noting that power users can cut down on the required pointing time by using the tab key to switch between fields.

Prototype

Step	Description	Operator	Duration (sec)
1	Click the "Add New Project" link	P	1.1
2	Page load time	R	0.5
3	Mentally prepare to create a project	M	1.35
4	Select the title field	P	1.1
5	Move hand from mouse to keyboard	H	0.4
6	Type a 65-character title	K	18.2
7	Move hand from keyboard to mouse	H	0.4
8	Select the description field	P	1.1
9	Move hand from mouse to keyboard	H	0.4
10	Type a 450-character description	K	126
11	Mentally prepare to add metadata	M	1.35
12	Move hand from keyboard to mouse	H	0.4
13	Select "Open" status (2 clicks)	P	2.2
14	Select 2 relevant tags (2 clicks each)	P	4.4
15	Click the "Create Project" button	P	1.1
		Total	**160**

Table 7.2: Approximate task time to create a new project with our prototype

The comparison between our systems reveals that despite our interface changes, it takes the exact same amount of time to use our system for a simple project entry. This was expected and intentional; we modeled our project creation process on the IDI system's process. However, there are some notable interface changes we made which actually make our process slower in most cases. We make use of drop-down menus for tags and status, which require one click to open and one click to select from. As seen in steps 13 and 14, this doubles the amount of time necessary to add one piece of metadata. The IDI system has fewer options for status and only a few tag-like metadata options, so it shows all of them on screen at once as a list of checkboxes. Even with our limited number of prototype tags, it would be infeasible to represent all of them on screen at once. Thus a user representing the same amount of metadata would require more time to input it. The only reason that this was not true of this example is that many of the metadata options in the IDI system are duplicative, so the same amount of information (six options) could be encoded in our system with half as many items (two tags and the status field).

7.4.2 Finding a Project

This section compares the time necessary for a student to find an interesting project within each system. We start measuring on the project listing page, with the student already authenticated.

IDI System

Step	Description	Operator	Duration (sec)
1	Mentally prepare for filtering projects	M	1.35
2	Select 1 specialization to find eligible projects	P	1.1
3	Wait for the page to reload	R	0.7
4	Mentally take stock of the reloaded page	M	1.35
5	Select another specialization to find additional eligible projects	P	1.1
6	Wait for the page to reload	R	0.7
7	Mentally prepare to read through projects	M	1.35
8	Scan the first visible group of projects for interesting keywords	M	1.5
9	Scroll the screen to show the next group of projects	S	1
10	Scan the visible group of projects for interesting keywords	M	1.5
11	Repeat 9 and 10 a total of n number of times	S + M	$(n-1)*2.5$
12	Select the first interesting project	P	1.1
		Total	$10.25 + n*2.5$

Table 7.3: Approximate task time to find a project with the IDI system

In the IDI system, there are very few filters available to students. Furthermore, these filters are inclusive, meaning that selecting additional filters from the same group expands the results list instead of shrinking it. As a result, the project lists returned by the system are usually long and unfocused. Thus students typically have to scroll a number of times before they can find an interesting project. In our tests, we found that it could take anywhere from 30 to 40 scrolls to read through all of the projects in a typical query in the IDI system without any professor preferences. Thus we added the note about steps 9 and 10 repeating numerous times to our model. Without professors, the range of totals is thus from 85 to 110 seconds. If the student has one or a few professors that they know of beforehand, that range goes down to 3 to 10 scrolls, or 17 to 35 seconds.

Additionally, each time the user selects a filter in the IDI system, the entire page reloads, which quickly adds up in waiting time. This is exacerbated by the fact that users need to re-situate themselves on the page each time it refreshes, which costs over a second each time.

Prototype

Step	Description	Operator	Duration (sec)
1	Mentally prepare for filtering projects	M	1.35
2	Select 4 filters to filter out uninteresting projects	P	4.4
3	Select the "filter projects" button	P	1.1
4	Wait for the page to reload	R	0.5
5	Mentally prepare to read through projects	M	1.35
6	Scan the first visible group of projects for interesting keywords	M	1.5
7	Select the first interesting project	P	1.1
		Total	**11.3**

Table 7.4: Approximate task time to find a project with our prototype

The final time in our prototype is significantly faster than in the IDI system in almost all cases. This is because our expanded filtering options allow students to cut down the projects list until it is very easily visible on one screen, or with one or two scrolls. This does, however, assume the student is very decisive about the filters they want and does not need additional mental preparation between selections. Figuring out the right filters would be a one-time cost, and so looking for projects afterwards should remain at the calculated cost. Additionally, we cut out the unnecessary refreshing that the IDI system performs each time a specialization is selected. It is worth noting that we still could cut out another reload and button press if we did our filtering on the browser side, but as discussed in subsection 6.5.2, we chose not to do this for ease of implementation.

Chapter 8

User Evaluation

In this chapter we present a summary of the feedback gathered from our unmoderated user tests. The data presented was gathered from seven students and five professors. All of the participants in our user testing had previously used the IDI project assignment system. As described in section 4.6, our user tests consisted of a series of opening questions to establish the user's familiarity with the IDI system, a set of tasks with associated questions, and a final group of questions about the user's overall experience with the prototype. For most of the multiple choice questions, we gave users the option to choose "Other" and provide a free response rather than choosing from the predetermined answers. These responses have been grouped according to sentiment. The results of this testing are explored below, beginning with student feedback and ending with professor feedback.

8.1 Student Responses

The results of every question asked as part of the student test are listed below. Additionally, some qualitative responses are quoted verbatim, but the complete response text of every participant to every question is not made public in this paper. The full set of student testing questions is attached to this paper as Appendix A.

8.1.1 Pre-test Questions

The questions listed in this section were presented to the participants before the testing tasks.

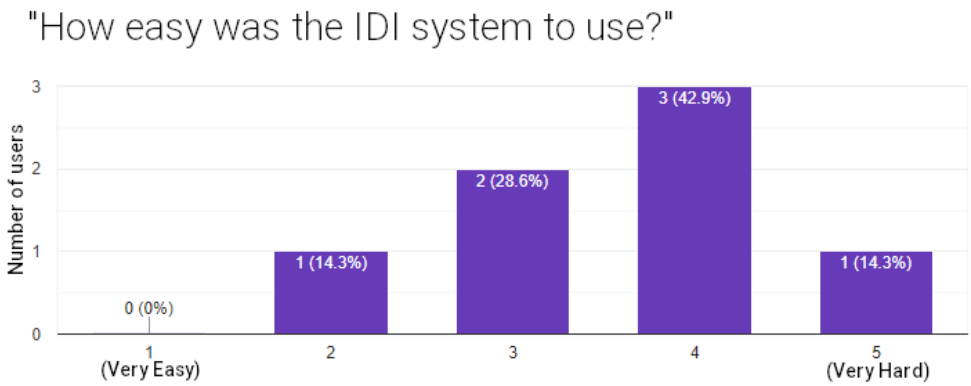

Figure 8.1: Student responses to the question "How easy was the IDI system to use?"
Mean: 3.57

The first question in our initial evaluation aimed to gather data about the user's experience of using the existing IDI system. This data is displayed in Figure 8.1. Sentiment about the existing system was negative. On a scale from 1 to 5, where 1 was very easy and 5 was very difficult, the average ease of use was 3.57. No respondents felt that the existing system was very easy to use.

Question	Yes	No
Has the large number of projects in the IDI system caused you any trouble?	4	3

Table 8.1: Student response to the question "Has the large number of projects in the IDI system caused you any trouble?"

The next preamble question aimed to measure whether project volume specifically was a pain point for students in the current system. This data is presented in Table 8.1. When identifying the underlying problems with the IDI system, we suspected that the growing number of projects was making it more difficult to find relevant projects in the IDI site. However, student response to the question was mixed. Only 4 of the 7 students who participated in the testing felt that the large number of projects specifically caused them problems.

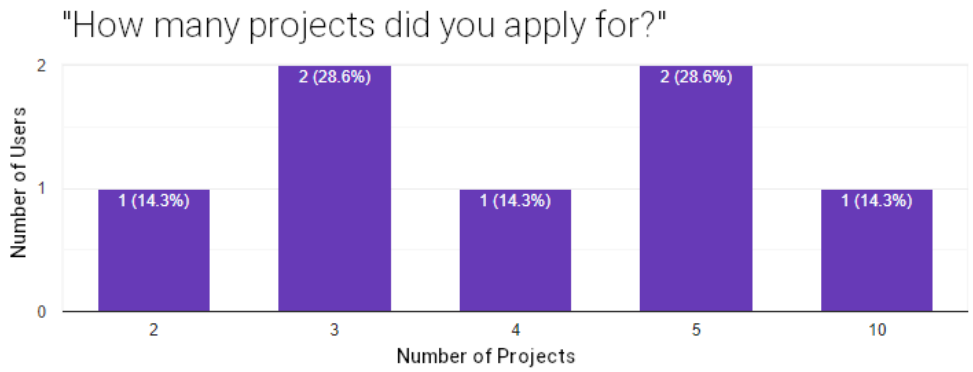

Figure 8.2: Student responses to the question "How many projects did you apply for?"
Mean: 4.57

Next, we tried to get an idea of how many projects each student had applied to. The results are displayed in Figure 8.2. Even among our small sample of students, we had one student who applied to 10 projects and one who only applied to 2. This shows a wider variance in project application numbers than we had expected. Our design decisions assumed four or five applications per student, as the IDI system only allowed each student to rank 5 projects. These assumptions seem to be mostly in line with the students we polled. We designed our system to expand easily to large numbers of applications per student, meaning that we would be able to accommodate the student who applied to 10 projects, but this data raises the question of whether we could have done anything to optimize the system more for students who put in only two or three applications.

We then asked the students what they both liked and disliked about the existing IDI system. Overall, responses to the question "**What did you like about the IDI project system?**" were not very enthusiastic; one user simply answered "Not too much," while another responded "Somewhat got the information I needed for some projects." Four of the six responses mentioned the IDI system's filtering capabilities, especially its ability to filter to specific professors. This shows that we correctly identified filtering as an important feature to expand upon. Three of the responses mentioned the project descriptions being useful. This implies that choosing to add additional descriptive information such as tags and project capacity was in line with student desire. It also highlights the importance of features that make descriptions more informative, such as rich text and image embedding. Although we did not implement these in our prototype, we consider them essential features for a full-scale implementation of the system.

Responses to the question "**What did you dislike about the IDI project system?**" were highly varied. Four of the six responses stated that the IDI system's filtering wasn't detailed enough, once again reinforcing our decision to focus on improving project filtering. In a similar vein, two responses expressed disappointment that the IDI system did not have a way to search projects. While we believe

that our improved filtering system alone should provide the tools necessary to find interesting projects, there is no reason that a system like ours should not be augmented with a search functionality. Indeed, when asked if they felt if anything was lacking from the system in the post-test questions, multiple students requested a search function.

In addition to those filtering responses, we also received a number of responses complaining about the user interface of the IDI system. Four replies said it felt bad to use, and two replies called it outdated. One student told the story of pressing an “Ok” button on a project without knowing what it would do and accidentally choosing it as their thesis project, which they attributed to the IDI site’s unclear interface. While we did not prioritize interface design as part of our prototype’s development goals, we did try to build according to modern web app paradigms when possible, which these results indicate was a good decision. Additionally, by making applications part of the system so they can be tracked separately from projects, we were able to make their status at any given time much more explicit.

8.1.2 Task Questions

These questions were presented to users during the testing, alongside corresponding tasks. The tasks will be briefly summarized here to provide context for the responses. After each task, users were asked the question "**Was there anything that was confusing or did not work correctly?**" This question will be omitted from discussion below for tasks where no students mentioned any issues.

Task 1 - Profile

In the first task, students were asked to edit their profiles within the prototype to include some tags relevant to their interests. Afterwards, we asked one multiple-choice question, presented in Table 8.2, along with a free response question to gather more detail.

Question	Yes	No
Do you think having a profile will save you time when applying to projects?	6	1

Table 8.2: Student responses to the question "Do you think having a profile will save you time when applying to projects?", attached to the profile editing task. One "Other" response interpreted as No.

Six of the seven responses to the free response question "**Do you like the idea of having a profile that’s visible to professors? What information would you want it to show?**" were positive, stating that student profiles were a good idea that could be useful. Four students mentioned wanting to have a more complete academic record on their profiles than just the specialization tags we provided. Additionally, two responses expressed a desire to have more control over their

profiles than the existing tags allowed, either by letting them create custom tags or giving them a free text field to write down additional information. These are both reasonable pieces of feedback, especially given our limited tag set in the demo. We believe that custom tags would end up being more confusing than helpful for professors, but allowing students to divulge as much or as little academic history as they want via a free-text box on their profiles is a reasonable way to address this complaint.

One student said that having a profile would save them time because they "don't have to write the same information to several professors", which is exactly what we were hoping to achieve. Conversely, another student doubted that the professors would even take the time to check profiles given the number of students they have to manage. Future projects may want to assess whether that is true, and look into incorporating profile information into applications in a more immediate way if so.

Another student said that "it could make it possible for professors also to find students for their project, and not only the other way around." This is not something we considered in our design, given our stated problem is a too large volume of students, putting this work on professors would likely not be acceptable. As such we did not make it easy for professors to view arbitrary profiles. However, it is an interesting idea. A future project could attempt to invert the relationship and make it possible for professors to filter and search students, view their profiles, and send them suggestions to look at certain projects. This would require giving professors a lot of tools to quickly filter students. We could see machine learning being a powerful tool for narrowing down options here, while mitigating the potential for bias in fully automatic matching.

Three of the seven students faced difficulty or felt that something was unclear in this task. Two students mentioned that the tag selection dropdown was a challenging piece of UI to use, with one saying they struggled to find it on the page and another calling it "not optimal". As discussed in section 6.5.5, we expected these complaints, as the tag input interface was a compromise to make the system testable within the time we had available.

One student called out a lack of informational/educational text on the site explaining what the profile actually was, or what could be edited. Given the time constraints on development, we did not implement any explanatory text on the site, so this was an expected piece of negative feedback. It is likely that a better visual design for the profile page could clarify these points on its own without the addition of text explanations.

Task 2 - Filtering

The next task required users to find a specific project within the system. This project was easy to find by filtering the large project list in various ways, though the task never explicitly directed the users to do so. According to the data in Table 8.3, every user found the projects by filtering, and all students thought that filtering

would be useful to them in a final year project search.

Question	Yes	No
Did you use the filtering system to find these projects?	7	0
If you were using this tool to pick your masters topic, would you use the filtering system to search for relevant projects?	7	0

Table 8.3: Student responses to questions about the first filtering task.
No "Other" responses.

Despite this positive feedback on our filtering system, the overall ease of use for finding projects as relayed in Figure 8.3 did not meet our expectations; four out of seven respondents rated the task's difficulty at 3 or higher on a 1 to 5 scale, giving us an average difficulty of 2.57. We suspect that some of the issues discussed below explains this rating.

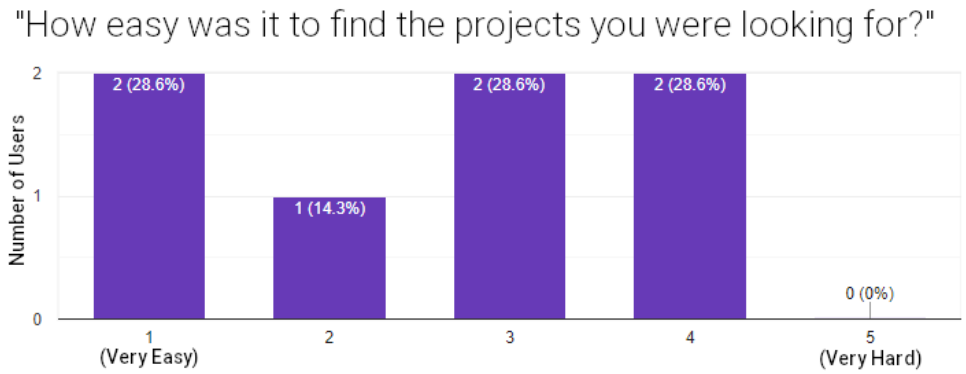

Figure 8.3: Student responses to the question "How easy was it to find the project you were looking for?"
Mean: 2.57

Following that, we asked the question "**By default, projects on the main page are filtered according to your academic history and interests as defined in your profile. Do you think this would be helpful when trying to find masters topics?**" Six of the seven responses were positive, with one student saying that this default filtering would save them time because they wouldn't "have to put in the same filters every time". Although many of our testers thought that default filtering was helpful, it might conflict with the students' desire to have a larger number of more expressive tags on their profiles, as it would lead to each student having such specific default filters that they would no longer see results by default until they cleared some of their filters away.

The one negative response said that it did not seem important to have this default filtering as students could just do it themselves, which is reasonable. One user also noted that the usefulness of this system is predicated on proper tagging

from professors. It is true that this system fails if the projects do not tag their work well, but we have no way around that short of auto-tagging projects based on keywords in their descriptions, which can be error-prone and lead to projects showing up under confusing or incorrect tags. Still, the perception that professors may not tag their projects appropriately could lead students to avoid filtering in order to not miss any projects.

Last, we asked students "**Is it helpful to see the number of applicants and offers on each project? What do you like or dislike about that?**" Responses to this question were unexpectedly mixed. Three responses were positive, three were negative, and one was neutral. The students who responded positively felt that it was helpful to see how many people they would be competing with for a slot, and said it would help them avoid wasting time applying to a project that was likely to be taken. The students who responded negatively felt the same, but noted that project assignments should be based on who is the *best* applicant, not on who applied first. Some students felt that if people who were qualified for a project saw that it was very popular, they might not apply for it, even though they would get an offer if they did apply. This is a fair criticism that revealed a bias in the design of the system; the researcher's experience was in line with the students who felt that they were unlikely to receive a slot for a popular project, and thus the feature was intentionally designed to help students avoid popular projects. However, this behavior encourages users to spread their applications among a wider pool of projects, which could lighten the load of individual professors. A more specific study of the impact of this information on user behavior should be carried out in order to determine whether or not it should be included in other project assignment systems.

Three respondents reported issues with this task. Two students mentioned being confused by the filters requiring them to press an "apply" button instead of applying instantly as soon as they were toggled. This was expected negative feedback, as we knew that the most common modern paradigm for filtering is instantaneous browser-side filtering via JavaScript. Unfortunately, our technical design forbade that approach, as the filter queries we constructed were complex and all of the database information available to the server. We could have implemented near-instant filtering via AJAX, but as discussed in subsection 6.5.2, we found AJAX unexpectedly difficult to manage with Django and thus not worth the time. Alternatively, we could have used client-side filtering, but this would also not leverage the strengths of Django, and require a lot of extra work. This feedback once again highlights the importance of UI in project matching applications, and we encourage future researchers to take this into account when choosing to allocate their time.

One student reported being confused by the default filters being applied from their profile, but quickly realized why those filters were there and cleared them in order to start searching for the project they needed. This is another element of the site that is not clear at first glance, and could be improved with explanatory text in the profile or above the filtering section.

Task 3 - Applying to a Project

Question	Yes	No
Is this process comparable to how you got a project in the IDI system?	1+1	4+1
If you chose to automatically accept an offer, then this project becomes your final choice the moment the professor offers it to you. Is this a feature you would make use of?	2+3	2

Table 8.4: Student responses to questions about the project application tasks. Respectively 2 and 3 "Other" responses were interpreted as yes or no, listed with + as prefix.

The next task asked students to apply to the project they just found. Our first question after this task attempted to assess whether users felt that the application process implemented in the prototype was an appropriate formalization of the informal application process they had previously engaged in. However, the question was poorly phrased, and did not particularly lend itself well to a simple choice format. From the "Other" responses we received, it seems that the students viewed this process as fundamentally different from the informal application process they used with the IDI system, and felt that it was a better process. This is an interesting difference from the professor responses, which largely regarded this process as no different from the IDI one.

We also explained the auto-acceptance feature and attempted to gauge interest in it. The responses to that question were in line with our expectations; all of the "Other" answers mentioned only wanting to use it if they were absolutely sure about a project, which is the intended use case.

In order to get more detail on the comparison we were driving at in our first question, we asked how applying to projects in this system compared to applying via email. Five out of seven respondents felt that using our system to apply was better than emailing. Multiple students appreciated having everything all in one place, with one saying they prefer this system because “emails make it messy”. The removal of the extra step of finding the professor’s email and setting context for which project you are interested in was appreciated. The negative respondents felt that the application within this system did not provide enough information on its own; one stated that they felt email was “more personal” and would probably send an email to the professor in addition to applying through the website, and another mentioned wanting to meet potential advisors in person before accepting their offer.

One student pointed out that it is awkward to apply to multiple projects made by the same professor in our prototype, as each application is tied to only one project. This leaves the student in the position of having to repeat their application several times as they apply to each of the professor’s projects, or work things out

over email instead of in the system. This was an oversight in the design, as we did not consider this a problem when designing our system. We could choose to incorporate this into the database model for applications, applying to multiple projects with one application. However, with the previous discussion of free-text in profiles, it may be more beneficial to move any application text that is true for multiple projects to a student's profile, so that the student only has to include information specific to each project in each application.

We also asked "**Do you like the idea of having formal applications within this system? Why or why not?**" and got mostly positive responses. Some students felt that our application system would prevent email applications being lost in professors' inboxes, while other students felt that these applications were *more* likely to get lost, and said they would email the professor if they did not get a response quickly. The students who felt that way believed that professors would not check the website frequently or thoroughly enough to keep track of every new application. It is interesting that these students thought about notification behavior for professors unprompted - this implies that we were right to create requirements for notifications, as they are clearly important to students both for themselves and for professors. Additionally, two responses again mentioned that they appreciated having everything relating to project selection gathered in one place.

One student expressed a desire to talk with the professors about their projects before submitting a formal application via our system. We originally considered including a functional requirement to allow students to comment on projects without sending applications, but ultimately removed that requirement because it seemed reasonable to do that over email, as an informal communication like that does not need tracking in a place where it cannot get lost. However, it may be worth considering making that possible within systems like this in the future, as several students expressed that they felt more comfortable contacting professors within this system than via email. If these questions could be posed publicly, this functionality could also cover the case of students wanting group projects without having groups. Students could informally ask if there were incomplete groups they could join under a project, and then submit a group application together, without requiring professors to interact.

Task 4 - Finding a Group Project

In the next task, we asked students to return to the main page and find a project that accepted group applications but did not require user evaluation. We felt that using both inclusion and exclusion filters at the same time might be confusing for some users, but the results recorded in Table 8.5 show that none of the users felt that it was unintuitive.

Question	Yes	No
Was it intuitive to combine positive and negative filters?	7	0

Table 8.5: Student responses about combining filters.
No "Other" responses.

We also asked students "**If you were using this tool to pick your masters topic, would you find it useful to filter out certain tags or requirements? If not, why not?**" Every response to this question was positive, with several students saying it would help them avoid “irrelevant” and “uninteresting” topics. This is a strong indicator that the addition of negative filters was a good way to address student feedback on interesting projects being hard to find. One student said specifically that it would reduce the time they spent reading uninteresting proposals, which was one of our goals.

Task 5 - Creating a Group Application

In the next task, we asked students to apply to the project they just found in the previous task. We also asked them to add two other students to their application as group members. Following this, we asked the question "**How does doing group applications this way compare to doing them via email?**" This question was poorly posed, as many of our respondents did not apply for any group projects and thus had no prior experience to compare to. Regardless, the users did provide speculative answers of how it compared to what they would have done to apply as a group in the current IDI system. Several students stated that the method presented in our prototype was easier and more intuitive than doing group applications via email, as it did not require professors to cross-reference multiple emails to make sure they knew everyone in a group. However, one student felt that this was a less personal approach, as it didn’t allow each group member to send their own introduction to the professor.

Question	Yes	No
Was it easy to add the other students to your application?	7	0

Table 8.6: Student responses about creating group applications.
No "Other" responses.

Per the results in Table 8.6, all students found the student adding interface easy to use. As mentioned in subsection 6.5.2, this was the only instant-feedback AJAX interface that we implemented in our system. The positive feedback here seems to imply that this type of interface is what users expected for this feature.

While working on this task, one user encountered a bug where the member name entry field was not cleared after pressing the button to add that user to the application. We fixed this bug before any further users could take the test. Otherwise, no users ran into any issues completing this task. However, one student

mentioned that adding students to an application should require the consent of each added student. This is something that we considered as part of our group projects implementation, but did not implement for the sake of the demo. The full explanation of the complexities of group project management and our choices around it can be found in subsection 6.5.1.

Task 6 - Ranking Applications

In the next task, we asked users to visit their applications page and rank their applications according to their interest. Application ranking is a concept that already exists in the IDI project system, so we expected the test students to be familiar with its operation. The results in Table 8.7 seem to match this assumption. Every student felt that it was easy to see how their applications were ranked, and the mechanism we introduced for re-ranking applications after a rejection was seen as a reasonable behavior by every participant.

Question	**Yes**	**No**
Was it easy to tell which application had which rank?	7	0
In this system, lower-ranked applications are automatically moved up when a higher-ranked application is rejected. Is that something you'd expect to happen?	7	0

Table 8.7: Student responses to questions about the application ranking task. No "Other" responses.

Unfortunately, our ranking interface was not well received. Four of the seven testers reported confusion or difficulty with this task. One user stated that updating the priorities of their applications did not work; this was most likely the result of them not noticing the submit button that must be pressed to save priorities. This was another case of our decision to not use AJAX patterns (see subsection 6.5.2) creating interface confusion. Another student was confused by the unlabeled priority fields below the first five, and the fact that they could assign priorities lower than their total number of applications. Two testers were also confused by the fact that they could not set a rank for the group application that they were a non-owning member of (see subsection 6.5.1). One student expected each project to be removed from the drop-down menus after it was ranked, rather than it remaining available and swapping to the new rank if selected somewhere else. Finally, one student requested a completely different interface where projects were dragged up and down in an ordered list instead of selected from drop-downs.

All of these criticisms are reasonable. The interface for this page was rather poorly considered, and our test data created additional confusion by providing students with very few projects to rank, as well as including a prominent group application that didn't show up in the menus as the test student was not its owner. Because ranking is a feature that already exists within the IDI system, we did not

focus on improving it as much as we should have, and had to compromise on the interface design due to time constraints. We explain this decision in section 6.5.5.

Task 7 - Comments

In this task, we asked students to respond to a comment from a professor on one of their applications. We then asked the question "**How does discussing a project with a professor in this system compare to doing it via email?**" Students were mostly positive about the comments system, again citing the ease of having everything together in one place on the app. Two of the respondents mentioned the lack of notifications as a deterrent in using this app to converse with professors - they want to respond promptly, but don't want to waste time checking the app multiple times per day. As described in chapter 6, we do consider notifications an important feature for this kind of application, but did not implement them due to our time constraints.

Question	**Yes**	**No**	**Other (Neutral)**
Would you feel more comfortable using this system to communicate with a professor about a project than emailing them directly?	6	0	1

Table 8.8: Student responses to the question "Would you feel more comfortable using this system to communicate with a professor about a project than emailing them directly?"
One "Other" response simply said "neutral."

One of the goals of moving communication from emails to the app was to make students feel more comfortable contacting professors by creating a dedicated space for it. Based on the results shown in Table 8.8, this appeared to work, as almost all respondents stated that they would feel more comfortable using this system than emailing professors directly. The one "Other" response simply said "neutral."

Task 8 - Accepting an Offer

Question	Yes	No
Was it clear which application had an offer?	7	0
After accepting an offer, your other applications were automatically revoked, and you were removed from the group application you were a part of. Is that something you'd expect to happen?	6+1	0

Table 8.9: Student responses to questions about accepting an offer.
One "Other" response on the last question was interpreted as a yes, listed with + as prefix.

In the final task, students were asked to accept an offer from a professor. As recorded in Table 8.9, none of the users indicated any difficulty finding the one application in their list that had an offer on it. After the students accepted an offer, their other applications got automatically retracted. This behavior was also considered reasonable by all participants, though the student who submitted the "Other" response noted that they'd expect the professors and group members to be notified in some way that the change was due to accepting another project, and not just a regular withdrawal. Our prototype did not indicate that in any way to the professor or to other students, which is an oversight.

8.1.3 Post-test Questions

The following questions were asked after the completion of all of the tasks listed above. By this point, the users had acquired a decent understanding of our system, so we could ask questions about the overall experience of using it.

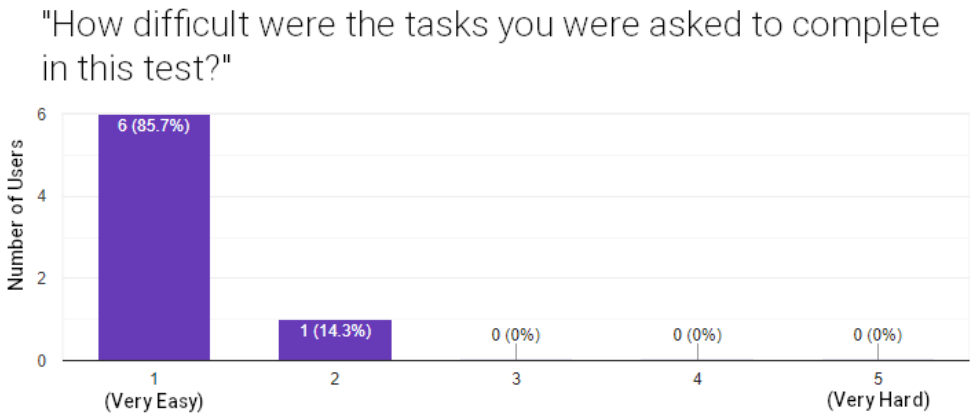

Figure 8.4: Student responses to the question "How difficult were the tasks you were asked to complete in this test?"
Mean: 1.14

The goal of the first post-test question was to assess whether our tasks had been unclear or frustrating for the user. The responses in Figure 8.4 were concentrated on the easy side of the scale, implying that the test was not unusually difficult. The professor responses in Figure 8.8 were similarly positive.

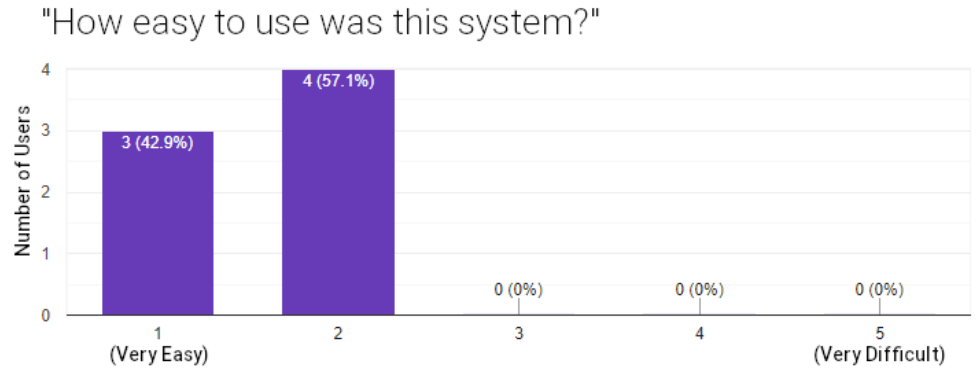

Figure 8.5: Student responses to the question "How easy to use was this system?"
Mean: 1.57

The next question asked users to rank the system's overall ease of use. The results in Figure 8.5 were entirely positive, although the majority of our testers felt the prototype could still be easier to use. These results were reasonable, considering that it is still only a prototype. They are also much more positive than the results shown in Figure 8.1, where we asked the same question about the IDI system.

Following those quantitative questions, we asked a series of qualitative questions beginning with "**How did this system compare to the existing IDI solu-**

tion?" The feedback on our prototype was very positive. Students called out filtering, tags, comments, and the cleaner user interface as improvements over the current system. Next, we asked "**Was there anything that surprised you about the tool during the tasks you completed?**" Only one respondent gave a negative answer to this question, repeating their earlier complaint that they could not set a priority for the group project they were in. As mentioned above, our discussion of that decision can be found in subsection 6.5.1.

The third question aimed to assess what things the students were worried might go wrong when the system was used at full scale, and their academic career actually depended on it. We asked the question "**What problems do you think you would encounter if you used this system to find a thesis project in a real semester?**" Four of the seven responses were either blank or said they could not think of any issues they would have with the system, which indicates that testers felt our prototype would be appropriate for real-world use. Two students mentioned that they thought the number of filters could get overwhelmingly large in a real instance of the system, and asked for a search bar to find filters instead of listing all of them out on the page. We noticed this issue during development, the filter menu is very long even when we only use a subset of the tags we expect a full system to use. We implemented collapsible categories for this reason, but did not make use of them by default, so a different solution for navigating the filters will be needed.

One student asked for more tools to find interesting projects - search, sort, and more filters. This would be a good set of features to explore in the future, however we suspect these features may quickly reach a point of diminishing returns due to their overlap, which is why we focused only on improved filtering for this prototype. This student also mentioned wanting to favorite or save projects for later in some way. We considered adding a requirement for this during the ideation phase, but decided not to because students could simply bookmark project pages in their browsers to save them for later. We still believe that this is a reasonable thing to leave up to users, but it is worth pointing out that many modern web applications which don't support bookmarks offer a favorites system instead, so users have grown to expect it.

Another student asked for a system for students to formally propose alterations or conditions to a project that the professor would have to accept. Specifically, they wanted to be able to auto-accept a project only if the professor accepted whatever change they laid out. While this could be an interesting feature, it seems an unnecessary formalization; students can lay out these sorts of conditions in their applications or in the comments, and professors would presumably only offer them the project if they agreed to those conditions. More often than not, these things are resolved in one-on-one meetings instead.

Next, we asked for the student's opinion of the project discovery systems in our prototype. Specifically, we asked "**Do you feel this tool would help you find interesting projects more quickly? Why or why not?**" Every response to this question was positive, with six of the seven responses mentioning filtering

as something that would help them find projects more quickly. One student was particularly excited about the tag system making it more explicit what research methods a project expects, as they personally took on a machine learning project by mistake because they thought it was a software programming project.

Finally, we asked the question "**Was there anything you felt was lacking or missing from the system**" in order to identify potential holes in our design. Most of the answers to this question were repeats of the suggestions explored two questions ago, in response to our question about possible problems at full scale. However, one student did mention that they felt the system was lacking a way for students to submit custom project proposals. As discussed in chapter 5, this is something that we believe is important for a full-scale project assignment system, but we did not implement it in our prototype. This answer reinforces our assertion that project suggestions are important for a system like this.

8.2 Professor Responses

The results of every question asked as part of the professor test are listed below. Additionally, some qualitative responses are quoted verbatim, but the complete response text of every participant to every question is not made public in this paper. The full set of professor testing questions is attached to this paper as Appendix Appendix A.

8.2.1 Pre-test Questions

The questions listed in this section were presented to the participants before the testing tasks.

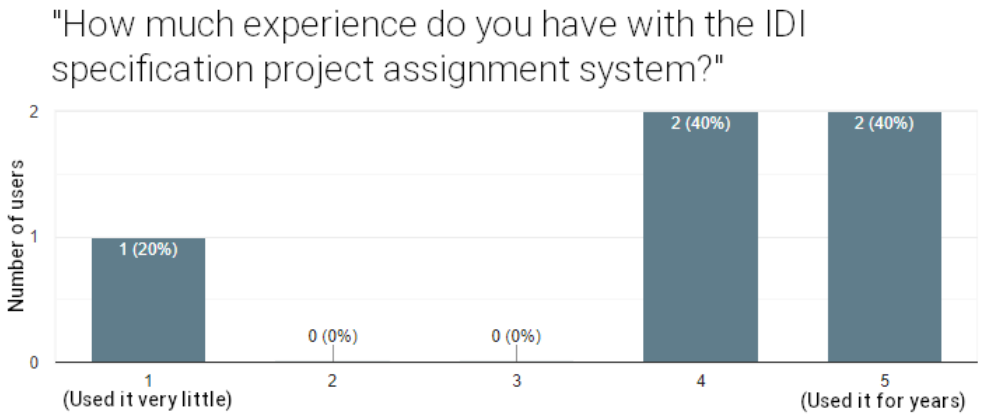

Figure 8.6: Professor responses to the question "How much experience do you have with the IDI specialization project assignment system?"
Mean: 3.80

Our first question aimed to get a sense of how familiar each professor was

with using the current IDI system. Because we used complaints from professors about the current system to guide our prototype's design, knowing how much each professor has used that system helps us assess if we identified and alleviated those complaints correctly. The results for this question are shown in Figure 8.6. Responses were polarized, with one professor expressing minimal familiarity with the IDI system while four others described themselves as somewhat or very familiar with it. While we would have ideally liked a wider spread of familiarity levels among our users, the one user who identified themselves as not very familiar with the system was still a useful point of comparison to the other professors' answers.

Next, we asked users if they had ever used any other systems for project assignment before, and to name them if they had. While this would have been helpful in providing additional points of comparison for our implementation, none of the respondents had used any other systems, so this question generated no data for us to analyze.

We then asked our participants what they both liked and disliked about the existing IDI system. Responses to the question "**What do you like about the IDI project system?**" covered a wide range of features. Across the five answers we received, the professors mentioned every available feature of the IDI system as positives – project statuses, student project rankings, draft and copy projects, and the ability to browse all projects. It is clear that we were correct to treat these as core requirements which had to be reproduced in our system. It is interesting how much more positive the professors were about this system than the students – one professor called it "easy to use" and stated that it had all of the features it needed, while one student said it just barely functional enough to sometimes meet their needs.

Despite this positivity, the respondents still had plenty of feedback to share when we asked the question "**What do you dislike about the IDI project system (or other systems you've used)?**" Two responses complained about the lack of a detailed categorization system making it difficult to find projects on specific topics. Three professors complained about the program's interface not being user friendly. Two of the responses mentioned the separation of old, outdated projects and new projects being insufficient, leading to confusion. Finally, one professor noted that it is unclear what action a professor should take after a student ranks their project unless that student also sends an email, and another complained about not being able to tell if the students ranking a project want to take it as a group or separately.

This negative feedback was was mostly in line with what we heard from professors during our requirements elicitation interviews, and confirms that we identified the IDI system's problems correctly. We addressed the categorization issues by introducing a tagging system, and the old project separation by formalizing the project state logic and hiding closed projects. The complaints about applications being unclear were part of our core motivation for developing these improvements in the first place, so our system addresses those issues by formalizing applications and tying ranks to those applications.

Next, we asked if the large number of students and projects specifically had given professors trouble in the IDI system. We posed this as a multiple choice question, but received different answers from every one of our five respondents. One selected "Yes", one selected "No", and the remaining three gave "Other" answers with additional details. Of those answers, one user abstained from responding due to lacking familiarity with the system. The two remaining "Other" responses both said that the volume of everything hadn't caused them any particular trouble, but that it was hard to get an overview of everything and required additional note-taking to manage. The last user selected the "Yes" option and did not elaborate. These responses are generally in line with our assertion that project and user volume was a pain point for professors in the IDI system, but they are somewhat less emphatic than we expected. This implies that the improvements we made in grouping information in one place and making it easier to track may not be as impactful for professors as we thought.

Finally, we asked professors what their primary method for managing project applications and questions from students currently is. This question was a bit unclear, resulting in us receiving different types of answers from every professor, but each answer was helpful in illustrating different techniques professors use to manage student applications. One professor described their strategy for choosing students, saying they picked whichever student ranked each of their projects the highest. Two professors described what we consider the standard pathway for the IDI system, listing projects on the site and waiting for students to email them applications. Two professors mentioned setting up in-person or virtual meetings with students to discuss questions, while two others mentioned discussing questions over email.

The prevalence of synchronous meetings in this feedback was unexpected; we optimized our system for text communications under the assumption that most student-professor contact took place via email, so we should emulate that process. The asynchronous nature of text communication also scales much better to large numbers of students, as it does not require shared free time between the students and the professor, so it made sense to optimize it given our theory that the growing size of the IDI department was straining the current system. However, this feedback raises questions about whether a system like this should assist in scheduling meetings between students and advisors in addition to the text communication mechanisms it already provides.

8.2.2 Task Questions

These questions were presented to users during the testing, alongside corresponding tasks. The tasks will be briefly summarized here to provide context for the responses. After each task, users were asked the question "**Was there anything that was confusing or did not work correctly?**" This question will be omitted from discussion below for tasks where no professors mentioned any issues.

Task 1 - Project Creation

In the first task, we asked professors to create a new project. Specifically, we asked them to fill in every project field, including capacity and appropriate tags. We then asked a series of questions, beginning with "**How did the process of creating a project in this system compare to project creation/listing in any project tracking systems you have used in the past?**" Responses to this question were positive. Two of the five professors called the process easy, and two complimented the addition of the tags. One said that it was very similar to the process they used in the old system. This shows that we did not regress the project creation process from what it was before with our changes, and that our additions were appreciated by the professors.

Next we asked if there was anything professors felt was missing from the project creation process. Only two of the five responses felt that something was missing. One mentioned wanting custom project tags, which are discussed in section 6.5.5. The other called for richer text in project descriptions, either via markdown or a full rich text editor. This supports our previous conclusion that it would have been valuable to implement a more expressive text system given how important project descriptions are.

Following that, we asked "**If you used this tool to track all of your projects, could you see yourself tagging all of them using this system?**" Our goal with this question was to see if professors felt the tagging was too annoying or time consuming to do for every project. One professor selected the multiple choice "Yes", one selected "No", and the remaining three selected "Other" and elaborated. All three of these responses were positive about the tag system and said they wouldn't mind using it, but expressed dissatisfaction with the selection of tags available in the prototype. Two professors requested the ability to create custom tags in response to this. As explained in section 6.5.5, we chose not to allow custom tags in our prototype due to the complexity they add to filtering. We suspect that if we had supplied a more complete set of tags, or explained that it was intentionally limited or could be added by administrators, sentiment about the tagging system would have been more positive among professors.

Three of the five professors reported issues or confusion while completing this task. One felt that the font size was a bit small. While we did design the site to work with web browser zoom functionality, we recognize that the default size may be too small for the wide range of users our site needs to work for. This was an oversight on our part. Another user was confused by the custom status option for projects. We implemented this feature because it was specifically asked for by multiple professors in our requirements elicitation interviews, but we could have done a better job clarifying what it did and how it worked in the interface. The third professor mentioned being unsure how to remove erroneous tags. We thought that having the tags change their outline when hovered should have been a clear enough affordance to show that they could be interacted with, but we suspect the interface would have been more clear with a close icon in each tag to

encourage professors to hover over them.

Task 2 - Profile

Our next task required professors to visit their profile pages and add some tags to reflect their academic interests. We then asked if professors were able to give themselves tags that accurately reflected their project topics and interests. Predictably, the responses to this question mirrored the responses to the previous one. One professor answered "Yes," one chose "Somewhat", and the remaining three answered "No." Although there were no "Other" responses with explanations in this answer set, we can surmise that the same issue of limited tags and no customization that made the previous task unsatisfying applied here. This is further confirmed by the response to our next question.

We then asked the users if there there was anything that they felt was missing from the profile screen. Two of the professors said they thought the profile was fine, but that the selection of tags was insufficient. As has been mentioned before, the set of tags present in our prototype was rather limited, and we expect this to not be a problem in a full implementation of the system with more tags. The other three professors felt that the profile was a bit empty, and wanted a place to enter a free-text personal description or link to their NTNU profile page. Some students shared this same feedback, so it is definitely a missed opportunity to have not included this functionality. Additionally, one professor was confused that the tags available on the profile were the same as those available for projects. The tags in our prototype were chosen with project pages in mind, so some do not have a clear meaning when applied to personal pages. However, we feel that using one consistent set of tags is the right choice for this system, as it keeps the relationship between personal tags and project tags immediately clear and consistent. The best compromise would be to use a system to hide the inapplicable tags from the profile editing screen. We considered such a system, but ultimately did not implement it due to time constraints.

Next, we asked if the professors would rather redirect students to an external profile entirely, and if so, why. The responses from professors favored both, requesting a profile within the matching system that also had a link to their NTNU profile or other external profiles. One professor mentioned specifically that their university profile includes their published research, which could be very helpful for students deciding on a project advisor. This could have been easily implemented by giving professors a field for links, or a free text field on their profile as discussed earlier.

Two users reported issues with this task. One repeated the earlier complaint about a lack of close icons on the tag bubbles in the tag editing section. The other user said "I believe the profile page did not work correctly," but did not elaborate further. Our guess is that this user did not press the save button to save their changes to the profile page, causing their edits to disappear when they next visited the page.

Task 3 - Survey of Applications

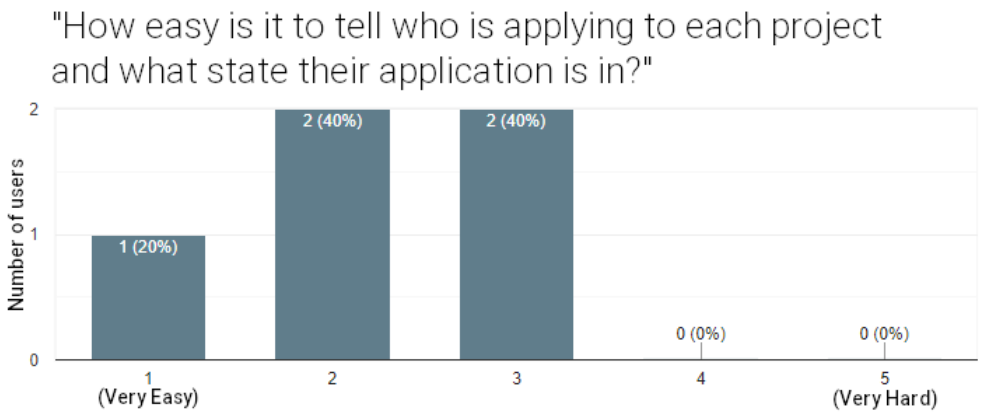

Figure 8.7: Professor responses to the question "How easy is it to tell who is applying to each project and what state their application is in?"
Mean: 2.20

In this task, we asked professors to visit their applications page and look over their outstanding applications, taking note of the applications' ranks and states. We then asked professors how easy it was to find that information. The results, displayed in Figure 8.7, were slightly less encouraging than expected. While no professors felt that it was difficult to get an overview of the applications, two of the four respondents felt neutrally about the interface, implying that it didn't make that information as clear as it could have. It may have been a mistake to pose this question in reference to our own interface only rather than asking professors to compare it to how they would accomplish the same thing in the current IDI system.

As a follow-up, we asked the professors if they were able to find their projects with open applications quickly. This gave us some more insight into the results discussed in the previous paragraph. Three of the professors responded positively, but one said they had to "get used to the interface" to find the open applications. We got one confused negative response from a professor who appeared to be looking at a specific project's details page rather than the application overview. Additionally, one of the positive responses said that while it was visible and readable, they still would have liked to be able to sort or filter this page by things like application status. We believe that these complaints are likely a response to the page's high information density. Every one of the professor's projects, along with all of their applications, are listed on this page, with very little spacing between them. Within each project, the application "cards" themselves are densely packed with student information, application text, and comments. When presented with this interface for the first time, especially if the page is already populated with data, users were overwhelmed.

The suggestion of using filtering to control this page was interesting. We did

not consider it in our requirements, but it would make it easier to manage large numbers of applications. By presenting the professor's applications as a static page separated by application status, we forced a specific order of information on the page which does not necessarily match how the professors want to see that information. Given this feedback, it seems that allowing them to filter by project, application status, and student would have been a better design.

Three professors reported confusion during this task. Two of the professors did not understand the "auto-accept" tag that some of the applications had. This is understandably confusing without context, especially for users who have never used this system as a student. Showing some explanatory text on hover would be a quick and effective way to address this. The remaining professor claimed that they were seeing other professors' project applications, which were irrelevant. This is not possible within the system. Most likely this user was seeing the pre-made projects already in the account and thinking they belonged to other professors, because they had not been created by the professor in task 1.

Notably, one professor was not sure what the "priority" number meant either. When designing the interface, we assumed that all of our users would already understand this concept from their past experience with the IDI system. However, if users do not have that experience or do not make that connection, they may mistake the priority number for something generated by the system, rather than a ranking chosen by the student themself. That misunderstanding could have an impact on which students get offers for projects, so some sort of explanation is definitely necessary.

Task 4 - Viewing a Profile

The next task required professors to view the profile of one of their applicants. The fastest way to get to this profile was to click on the student's name in the application, but the name is not visibly a link unless it is hovered. We chose to style the page this way because links naturally draw users' eyes when they are skimming pages, so cluttering a page with a large number of links makes it hard to read. To make sure that this wasn't confusing, we asked professors if they were able to find the student's profile easily. Four of the five respondents said yes, so it seems that this UI choice was mostly acceptable. It is worth noting that the one respondent who said no was also the user who was not on the right page in the previous task, which suggests that they may have run into difficulty because they were on a page with no profile links in the first place.

Next, we asked if the student's profile and written application provided enough information for the professors to evaluate them. Three of the four respondents said that they wanted more information from the student's profile than just tags. The information they requested ranged from technical academic information like grades and a full course history to personal statements like favorite course and extracurricular activities or hobbies. These are all things that the professors could ask for in comments or via email, but given the wide range of information that

professors asked for, the best solution is likely to add a free text box to the profile page as discussed before. The fourth respondent stated that they did not use profiles or personal information at all, and assigned projects solely based on ranking.

In response to our question about issues with the task, one professor left a very helpful comment about the tags on the student's profile. They explained that the headers on the tag categories were confusing, even though the tags themselves were understandable. In their example, the "Method" section on the student's profile was confusing, because it could either mean research methods the student has worked with in the past, or research methods the student wants to work with in the future. One possible way to fix this would be to use different tag group headings on student and professor profiles specifically to make it more clear how the tags should be read.

One professor struggled to find a profile because they were looking at their newly-created project's details page rather than the overall applications page. This was the same professor who incorrectly believed that they were viewing others' projects in the previous task. It seems likely that they clicked into the details of their own newly-made project, which had no applications, and thus had no student profile links to click on.

Task 5 - Responding to Comments

In the next task, we asked professors to respond to some comments left by students on their applications. After the users had completed the task, we asked "**How does answering followup questions and comments in the application compare to doing it via email? Would you have preferred it if the student emailed you?**" Overall, responses were positive, with four of the five professors saying that they appreciated having the whole history and context in one place. However, three of the five professors were concerned about having another communication system to have to check in addition to email, Microsoft Teams, Blackboard, Piazza, and more. This once again underscored the importance of notifications, so that checking this system can be a part of checking email rather than a separate task. Additionally, one professor simply said they would prefer email with no further detail.

Task 6 - Managing Applications

For this task we had the professors accept and reject a few pending applications. Following this, we asked the question "**With this system, all applications to a project are listed together on the same page. Did that change the way you approached evaluating the applications?**" Responses were mixed, which was expected for this question. One professor was pleased that they were able to see the exact list of members in each group, and also said the ranks were easier to see. Three other professors said that it did not particularly change the way they worked, while the last one declined to answer. This kind of question would be bet-

ter assessed by a long-term A/B test to compare differences in project assignment using our prototype and the IDI system.

Following that, we explained that when a student accepts an offer, their other applications are automatically retracted. We then asked professors if they thought this behavior would make managing a large number of applicants easier. Responses to this question were generally positive, but indicated some confusion on the part of the users. This was likely due the way the question was posed, as it attempted to explain a mechanic of the system that was largely relevant to students rather than professors, and that the professors could not see in action for themselves. One professor interpreted this as saying that accepting an application would close a project from future applications, and so asked for the ability to toggle this behavior, as some projects can accept multiple students or groups. Another stated that they wanted to know how many other applications the student had open in response to this question, which is a seemingly unrelated desire, as the student's other applications become irrelevant as soon as they accept an offer. We still believe that this behavior will be helpful to professors, as it automatically eliminates invalid candidates, but it is clear that it should either be explained to them more clearly or hidden from them entirely.

The following question was similar. We explained the reprioritization behavior for applications, wherein a student's application lower-priority applications are automatically ranked higher to fill the gap. We then asked the professors if they thought that behavior would make managing a large number of applicants easier. Responses were similarly mixed. Three professors responded "Yes", one responded "No", and the fifth abstained because they felt they did not understand the ranking system well enough to comment. Again, this implies that although these details may help professors select candidates for their projects, they may not want or need to know about them in such detail. The student responses to these features were much more clearly positive and less confused, which we believe is because it is more immediately obvious to students how these behaviors are helpful. It seems that what matters for professors is that applications have accurate statuses and ranks, which these systems help ensure. The exact mechanisms of how that comes to be are unimportant.

Finally, we asked if the professors found application priorities useful when selecting students. Four professors responded positively, while one did not feel strongly about priorities. One professor left some additional feedback on ranks that was particularly interesting. They said that although ranks are useful, they find them difficult to use in evaluation because there are no concrete protocols around them. The professor mentioned not knowing how long to wait for more applicants before choosing a student based on rank, or what to do when a student ranks a project but doesn't send an application. While we have solved the latter issue by tying ranks directly to applications in our system, we intentionally did not attempt to solve the former. Making changes to the process as a whole, such as enforcing a minimum open time for projects, was out of the scope of the project. However, this was a specific pain point which we heard from professors during our

requirements elicitation as well, and which we strongly believe that universities should address by setting some policies about open application periods in addition to using this system.

Task 7 - Closing Projects

In the next task, users were asked to close out any of their projects that had reached their application capacity. We then asked them if they would prefer for the system to automatically close at-capacity projects for them. Results were evenly split, with two respondents saying yes and two saying no. The fifth respondent declined to answer, saying they do not set capacities for their projects. This is the result that we expected to see from our test, and is why we chose not to automate this feature. From our interviews, we know that some professors view project capacities as hard limits which they won't go over, but for others they are just suggestions, or they don't set limits at all. While it would be possible to introduce a checkbox for professors to tick if they wanted projects to close automatically upon reaching capacity, we believe that keeping the process manual to avoid accidental project closures is the best option.

We also asked professors the question "**When a project is closed, it is removed from the main project list for students, but students can still visit its details page if they have the link. Is that what you'd expect to happen?**" and received similarly mixed answers. Two professors said yes and two said no, with one "other" answer asking what the point would be (which we interpret as another "no"). We asked a followup qualitative question which drilled into the details. One professor felt that allowing students to see closed projects would lead to them sending emails asking for slots in closed projects, which was annoying and a waste of time, while another felt that students should be able to see closed projects precisely because they might be willing to offer more slots in their closed projects. That second professor also mentioned that seeing closed projects could help students get a better understanding of their academic interests.

Next, we explained that students cannot see closed projects unless they have direct links to them, and then asked the professors if they would prefer for students to still be able to see closed projects on the project search page. Two professors responded negatively, with one saying that letting students see closed projects would just lead to time-wasting emails from students asking about them. One professor responded exactly oppositely, suggesting that students should be able to see closed projects precisely because the professor may be willing to take on additional students for that project. Additionally, they brought up the point that seeing closed projects may help students get a better sense of what types of projects the professor likes to work on. While the first point can be addressed by professors not closing their projects until the absolute end of the application period, the second requires system changes that are worth considering. It is reasonable that a professor could end up with only one open project left after their closing all of their others, which would make it hard for students to get an idea of the type of work

that professor does. It might make sense to allow students to search for closed projects on the main page, but keep them filtered out by default.

One professor reported being confused by the sorting of applications on the page in this task. We show applications in order of update time, so that the most recently sent or updated application appears first. We chose this sort order because we had no notifications system to alert professors to new interactions, so we tried to surface them as readily as possible. However, it may have been worthwhile to allow professors to choose the sort order here instead.

Task 8 - Copying Projects

The final task asked professors to duplicate a closed project, edit some details, and then save it as a draft. We then asked the question "**Do you feel that cloning and editing an older project is preferable to creating a new one? How did this process compare to the new project creation process?**" Four of the five professors responded positively, with one stating that they often reuse their projects and another saying it was "always preferable." The final professor's response was neutral. This is even more strongly positive feedback than we expected, and shows that we were right to call out project drafting and duplication as important features.

Following that, we asked "**Is it important to you to be able to create and edit listings as drafts before publishing them?**" Results were once again split, with two "yes" answers and three nos. This is again unsurprising, and lines up with what we heard in our pre-development interviews. Some professors like to create template projects and copy and edit them repeatedly, while others write each project from scratch. Accommodating both of those workflows was an important part of our design.

8.2.3 Post-test Questions

The following questions were asked after the completion of all of the tasks listed above. By this point, the users had acquired a decent understanding of our system, so we could ask questions about the overall experience of using it.

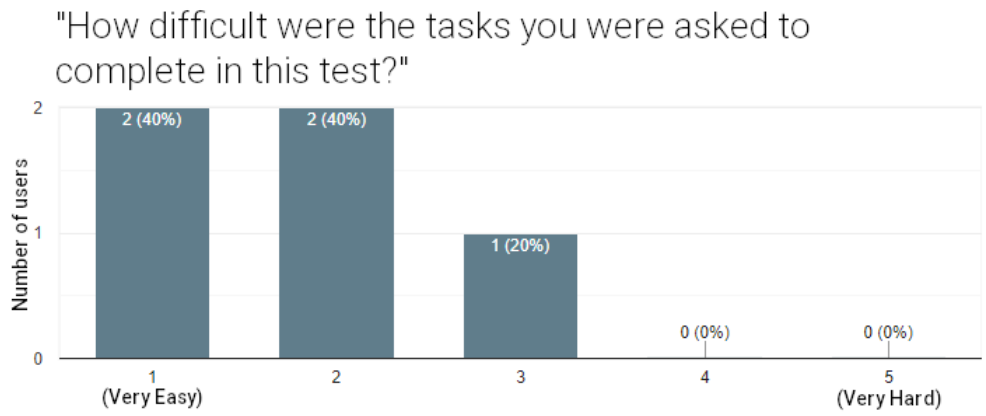

Figure 8.8: Professor responses to the question "How difficult were the tasks you were asked to complete in this test?"
Mean: 1.80

Students and professors were both asked this question as the first post-test question. The goal was to assess whether our tasks had been unclear or frustrating for the user. The professor responses, seen in Figure 8.8, were almost all on the easy side of the scale, implying that the test was not unusually difficult. The student responses in Figure 8.4 were similar, though slightly more positive.

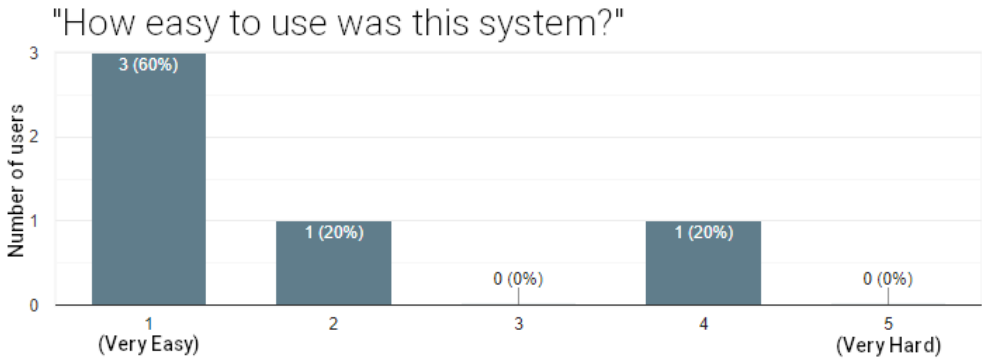

Figure 8.9: Professor responses to the question "How easy to use was this system?"
Mean: 1.80

That question was immediately followed by a question about the difficulty of using the prototype itself. The results in Figure 8.9 were mostly positive, with the majority of our testers giving the system the best possible rating. However, one user rated the system somewhat difficult to use. This was the same user who struggled with the application tasks early in the test, which we believe influenced this score. The positive result is slightly surprising given some of the negative feedback in earlier portions of the test, but it is reasonable given that the point of comparison for these professors was the existing IDI system, which is rather

difficult to use.

We then asked users if anything about the tool surprised them during the test. The goal of this question was to prompt users to discuss things about the app that were confusing or counter-intuitive. There were no negative responses to this question, and one professor praised the colors we gave to tags. This was surprising, as we expected more issues to come out in this question, but it shows that our implementation lined up with professors' expectations.

Next we asked what problems professors thought they might encounter if they used the system in a real semester. Most of the issues raised here were repeats of ones addressed earlier in this analysis – the selection of available tags, the lack of notifications, and some interface complaints. One professor mentioned the cost of onboarding professors to the new system. They also brought up the issue of not having clear guidelines about application periods from the university. As discussed before, we consider these details external to our prototype, but this response is another reminder of the importance of rigorously defined policies around the tool, as well as learning resources for professors to get started.

In our last quantitative question, we once again returned to the question of large user counts. We asked "**Do you feel this tool would help you manage a large number of students and projects better?**" Three of the four responses were positive, one was negative, and one was neutral. The neutral responder felt that followup conversations with students through this system may be difficult because the tool required a separate login. Our interpretation of this response is that the professor was worried about having to check a separate place than their inbox for conversations - a worry that students also shared about professors. We take this as yet another indicator that notifications are an important feature for this type of system, to make sure that professors do not miss any followups. We found that the professors we spoke to did nearly all of their communication via email, so bringing conversations to their attention in the place they already look for communications is important for any app following this scheme.

Following that, we asked professors what they thought was missing from the system. None of the feedback here was new. The selection of tags in the prototype came up once again, with the professor who brought it up saying that they would want meetings with both professors and students to determine the best possible list of tags. One professor requested free text fields in profiles again, and two mentioned our UI again. All of these complaints have been addressed already in this chapter. One professor did mention that they did not see how to search other professors' projects. The project listing is the main page of the website, but we did not have any professor tasks that interacted with it, so this confusion is understandable.

Finally, we asked how our system compared to the IDI system. The responses were mostly neutral; most professors felt that it was very similar to the IDI system, with a few improvements. None of the respondents were particularly excited about the prototype, or felt that it solved their problems significantly. This was unexpected, and not in line with the results from our student evaluation. Based

on our understanding of the problem, formalizing the application process and connecting applications directly to projects should have solved a number of the problems professors faced. However, it seems that most of the professors who took our test did not consider that to be a significant improvement. Students were noticeably more positive about it in contrast. This could imply that our choice of solution was biased in favor of students rather than being an effective solution for both parties.

Chapter 9

Discussion

This chapter synthesizes the technical results presented in chapter 7 with the user evaluation performed in chapter 8 to build a thorough analysis of the success of the prototype. This analysis is then extended to assess the requirements and design principles that guided the prototype, including our conception of the problem and overall choice of solution.

9.1 Success of the Prototype

In this section we will discuss the success of the prototype through the lens of the motivations we established in chapter 5. This analysis is based on the user feedback described in chapter 8, as well as the process comparison presented in section 7.3 and task timings in section 7.4.

9.1.1 Evaluation Criteria and Priorities

We evaluate the success of the prototype in three areas: information availability, ease of communication, and preventing lost applications. We believe that these are all important metrics of success, as they are all ways to address the issues brought on by an increasing number of students who need to be matched.

Information Availability

In order to make it easier for students to find the best projects and for professors to find the best students, we attempted to expose richer information about projects, professors, and students in our prototype. We did this by adding student and professor profiles, project tags, and project availability metrics.

Overall, we feel these features were successful in improving the amount of valuable information available to users. The students who tested our prototype felt that our professor profiles would make it much easier for them to find professors who would be good to work with. Similarly, they felt that the tags we added to projects made it significantly easier to find projects that were of interest to

them, and that those tags helped clarify a project's goals and methods when its description was unclear.

Student responses were more mixed on the application capacity information that we added to projects; some students found it very helpful, while others felt it would discourage them from applying to projects they were interested in. Ultimately, we believe that this feature positively impacts the matching system. One of our goals is to make it easier to manage large numbers of students. Encouraging students to spread their applications among projects and not overload one professor will ultimately reduce the amount of work necessary to balance workloads which normally has to be taken on by administrators.

The professors who tested our application were similarly positive about profiles and tagging, though they wanted more flexibility and expressiveness in both the tags and profiles. Most importantly, the professors did not feel that adding this information was a painful or unnecessarily time-consuming process, which means that the benefits of these features for students would be achievable in a real system. With a larger number of tags available and needing to repeat the process for each of their actual projects, the experience might become less tolerable however. Additionally, they were positive about having their own profiles to quickly communicate their research interests to students.

We also believe that the changes we made to project rankings to tie them to applications and automatically update them as students accept and reject other offers will significantly improve the decision-making process for professors. Although responses from professors about these features were mostly confused, we believe that is due to poor question formatting on our part rather than an issue with the features. These features allow professors to more quickly see the actual priority of every application and ignore applications from students who already have projects, which will speed up the decision process significantly.

Facilitating Communication

To make the negotiation process more efficient and less stressful for students and professors, we required students to write applications to each project they were interested in and introduced a comments system for followup discussion with professors. Previously, students could rank projects without actually applying, and all conversations with professors had to happen over email or in person.

Students were strongly positive about these changes. They were more comfortable sending comments within this system than via email, and felt that the added context of having their applications attached to a specific project made it easier to explain what they were interested in and why they were a good candidate. They also liked having their academic history in their profiles so that they would not need to repeat that information in multiple applications. We believe that these changes will make communication during the negotiation process easier for students who are less confident in emailing professors directly, mitigating one of the biggest weaknesses of a negotiation-based matching system.

Professors also liked the added context to their communications within this system. They felt that it was very helpful to have the student's application text and followups in the same place as the accept and reject options. They were less certain about not also having the conversations in their inboxes, as email is their main communication form, but we believe that the benefits of having all of the communication between students and professors formally organized into specific applications will outweigh the drawbacks. This is especially true if email notifications are put in place to make sure no new applications or comments are missed.

Preventing Lost Applications

One of the most concerning issues we identified with the existing IDI system was the possibility for student applications to be lost in professors' inboxes, as there was no way to organize and track them. We addressed this by building the formal application system, along with a page for professors that displays all of their applications grouped by project and status. As discussed above, professors were positive about formal applications because they removed a lot of ambiguity from the process. The formal status tracking also helps ensure that professors and students never have a misunderstanding about who needs to take the next step in the process at any point. While we were not able to test whether this actually solves the problem of forgotten applications, we strongly believe that we have fixed that issue with these features.

Overall, we find that our prototype is successful at addressing many of the problems identified with the IDI system, but could still be improved significantly.

9.1.2 Improvement over manual two-way negotiation

In this section, we compare our matching system to a basic two-way negotiation system as defined in subsection 3.1.1 and show that it is greatly improved in terms of speed of use and scalability. We consider the flow of one application through each system to illustrate this. First, students must find the project they wish to apply for, which can be done quickly and efficiently in our system using tag filtering. Without filters, students must peruse a long list of projects which grows even larger as student bodies increase in size. Then, the student applies for a project. In our system, they do this directly on the project page. Without it, they have to find the professor's email address somewhere or visit their office to discuss their interest. The professor then reviews the application and discusses the project with the student. Without our system, this discussion would take place over a series of emails with no formal tracking, or over a meeting or two. With our prototype, it is directly attached to the application for both professors and students, and the application has an informative rank to help the professor understand how interested the student is. Finally, the professor must accept the application, which is easily done in our prototype but may require more frustrating administrative work in a generic two-way negotiation system.

At each step of the way, our prototype is able to save time without altering the overall flow of the application. It makes projects easier to find with tagging, helps professors evaluate students more quickly with rankings and profiles, and makes accepting and rejecting applications simple. We believe that it is a clear improvement over simple two-way negotiation without a system, while simultaneously keeping the best features of two-way negotiation.

9.2 Success of the Requirements

In this section, we assess the success of the requirements we derived in chapter 5 at producing a useful prototype. As we noted in chapter 6, we found that we needed to modify our requirements in the middle of development. However, that modification was almost entirely about scaling down our goals due to time rather than our requirements being ineffective. Once we began implementing, we only found two requirements which we had missed in the initial set.

Overall, the user feedback reveals that the requirements we implemented in our prototype were well received. More crucially, the user feedback revealed that most of the requirements which we defined but had to cut were in line with user desires. Both students and professors expressed a desire for notifications and thesis proposals, which we have fully defined requirements for. The success of our prototype combined with the calls for these features implies that our specifications for the software were well-chosen.

It should be noted that our requirements for administrative features and data exports have not been tested at all, and thus we cannot be confident about their efficacy or correctness. We believe that these are crucial features for any full-scale implementation of this system, so we have chosen to publish our requirements for them regardless. Furthermore, we have some confidence in these requirements as they were generated by the same elicitation method as the rest of our requirements, and those other requirements proved to be useful and well-chosen.

A feature that was mentioned occasionally in elicitation, feedback, and deliberation, was student-to-student interaction. While we did not construct requirements for these features, as we did not have a strong idea of where they would fit into our system, we believe there is functionality to explore here. One particular idea we considered was enabling students with an interest in a particular group project to announce this under that project, which would free up professors from trying to coordinate single students into groups. Another idea was to have public spaces for students to request more information about projects, such that professors would not need to answer the same questions repeatedly. These features could potentially be implemented together in a single comment field for each project.

9.3 Bias and Evaluation Problems

Our testing setup had a number of issues which could have biased our results. This section details those issues and the impact we believe that they had.

First, we would like to point out the bias in our user recruitment process for testing. As discussed in chapter 4, we first used a series of posts in student Facebook groups to recruit. However, response was minimal, even after several posts. In order to recruit a sufficient number of students, the researcher reached out directly to students he had previously worked with in classes and asked them to take the test. The researcher only had a brief working relationship with each of these students, however the pressure of being contacted directly and knowing the researcher who developed the project may have made some students want to respond more positively. Additionally, all the students had graduated by the time of the test, which may mean they were not under particular stress. The professors who took our test were contacted via direct email or in-person meetings with the researcher's advisor. This direct contact method may have similarly biased the professors, although none of the contacted professors had previously worked with the researcher. The professors were all still working, with upcoming deadlines for grading exams, and may thus have found the test to be annoying during a stressful work period.

Many of the requirements for student-facing features in this project were derived from the researcher's reflection on his experience using the IDI system. As such, they may have been biased towards improving the student experience more than the professor experience, resulting in a prototype that favors students rather than being an effective solution for both parties. This may be the case given the fact that the student response to the prototype was more positive than the professor response.

In our test setup, we failed to set context on the state of the prototype or the goal of our tests before asking users questions. This was in part due to the shortcomings of remote testing; we felt that it would be too much text to ask users to read through a full explanation of what we were testing for, while in person this would have been a much quicker verbal explanation and could be reiterated if the feedback got too focused on the interface. As a result of this missing context, and some poorly-worded questions, many users evaluated our prototype on the merits of its interface and test data rather than the features available. This feedback was still useful, but because our primary goal was to evaluate the effectiveness of our feature requirements, it was not directly related to what we were hoping to test.

It is also worth noting that in all of our testing, we asked users to compare our prototype to the existing IDI solution. This makes sense for professors, who had recently assigned projects in the tool for next year's students at the time of our testing. However, our student testers had not used that system at all since selecting their projects one year ago, so their memory of its function and ability to use it as a point of comparison may have been limited. We provided a link to the listing page if they wished to refresh their memories, but the rest of the system

was unavailable. Additionally, students had used the IDI system to sort through a long list of projects, and had done so under time pressure, which likely colored their impression of it negatively.

The current IDI system is biased against students in a few ways. Professors deal with a much smaller volume of data on a regular basis, primarily their own projects, while students have to sift through possibly hundreds of projects. Professors also get familiarized with the system across multiple semesters, while students are using the system for the first time. The stakes are additionally higher for students, as the project they end up with decides who and what they will be working with full time for the last two semesters, and ultimately what they will be graded on as the capstone of a five year endeavour. On the other hand, while professors may feel significant responsibility for the success and satisfaction of their students, a bad match will only occupy a limited amount of their workload for the next two semesters, and not affect them past that. With all of this in mind, we should expect students to be more easily biased against the current IDI system, and eager to see the potential improvements of a new system, as they would benefit more from an easy-to-use system. Professors may be convinced if a new system addresses their problems, but any system will need to make up for the added cost of the professors having to re-familiarize themselves with it, and so a new system will generally face more criticism. These explanations align with our results, but this does not rule out the possibility that our development itself favored the student experience.

Finally, it is worth noting that all of our users interacted with our system in a very restricted and prescribed capacity. None of our tests asked for users to explore the system freely or interact with other users. There are numerous complexities and edge cases that can arise from this type of interaction which we did not attempt to test. We partially addressed this flaw by asking users to imagine what problems they might run into if they used the system in a real semester.

Chapter 10

Conclusion

As student bodies in engineering grow at NTNU and other universities, the final year project matching process will only become more difficult. We have observed evidence of this scaling problem at universities around the world. While many of those institutions solved this problem by implementing systems for algorithmic matching, we instead propose a solution that retains the control and personal attention of manual matching while still facilitating larger student bodies. We have presented a set of requirements for a scalable manual matching system derived from our research and interviews, as well as a prototype software artefact which implements some of those requirements. We then tested this prototype with users of the existing IDI solution to evaluate the effectiveness of our requirements at producing a high quality system that scales well to large student bodies.

Research Question 1: *How can one design and develop a system for effective pairing of master students with thesis projects for universities?*
Using the design science methods described in section 4.1, the researcher developed a set of functional and non-functional requirements for the creation of effective manual project matching systems. These requirements are specified in chapter 5. They were derived from interviews with students, professors, and administrators who had all previously engaged in the final year project matching process. The researcher then developed a software prototype based on these requirements with the goal of efficiently matching students to final year projects. This prototype was then evaluated with user testing and other analysis techniques in order to assess the accuracy and effectiveness of the requirements. Based on the results of this evaluation, we believe that our presented requirements and development methodology provide an effective way to develop a matching system.

Research Question 2: *How does the software created for this paper compare to existing solutions in terms of usability, user satisfaction, and features available?*
To evaluate the usability of the IT artefact produced for this paper, the researcher set up remote usability tests with 7 students and 5 professors using the prototype.

Using a questionnaire with both restricted and free response questions, the researcher obtained qualitative and quantitative data about the experience of using the application. Analysis of both types of data showed that the prototype software was an improvement over the existing IDI process, particularly for students. Students felt that our prototype was much easier to use than the IDI software, while professors found it only somewhat easier to use. Both students and professors were excited about the new features introduced in our prototype. Finally, students felt that the prototype solved almost all of the problems they encountered in the matching process, while professors were mixed on whether or not it would help. We conclude that our prototype was only moderately effective at addressing the issues of the current system. However, the feedback we received shows that our requirements were accurate to user desires, implying that it was our implementation that could have been improved.

10.1 Future Work

Through the course of developing this project, the researcher gained a much deeper understanding of project assignment systems. Due to time limitations in the research, not every avenue of exploration could be pursued fully for this project. Below we describe some possible areas of future work based on our learnings.

10.1.1 Unimplemented Requirements

As noted in chapter 6, we were not able to implement all of the requirements we specified due to our time constraints. We focused on the core student and professor experiences, and dropped the requirements relating to administrative views entirely. An obvious direction for future exploration would be to implement these remaining requirements and evaluate how well they meet the needs of administrators. Alternatively, designing and iterating on the underdeveloped features of the system, such as the tagging types detailed in section 6.5.5 or the student-to-student communication in section 9.2, could reveal entirely different ways to improve the experience. Additionally, repeating user tests with some of the cut features like notifications would be interesting, as those features were very heavily requested in our feedback.

10.1.2 Impact of Popularity Metrics

Some of the student feedback we recorded in chapter 8 expressed concern that the availability metrics we made available for projects (the number of open applications, accepted applications, and expected capacity) might deter the student who is the best match for a project from applying to it due to competition. We consider this an improvement to the matching system, as it should help spread the student load among professors and projects. However, a rigorous study of how this

information affects student behavior would probably be necessary before putting it into use at a large scale.

10.1.3 Student Selection System

Our requirements and prototype focused entirely on building a one-way preference system where students rank projects and professors can only accept or reject applications. It could be worthwhile to investigate inverting this relationship and allowing professors to search through students according to their personal tags and offer them projects, or invite them to apply. While we dismissed that idea in this paper as it seems like it would increase administrative load rather than decrease it, testing it as an alternate pathway could reveal unexpected benefits.

10.1.4 Tag Set

As discussed in chapter 6, we chose not to implement custom tags due to the amount of interface and operational complexity that they generate. Instead, we focused on a small set of hand-selected tags for our prototype. Professors and students were both interested in having more tags to express themselves better within our prototype. Some research and experimentation to determine the best set of tags and categories within various fields would be essential to make this sort of system work in real scenarios.

Bibliography

[1] A.-B. Hunter, S. L. Laursen and E. Seymour, 'Becoming a scientist: The role of undergraduate research in students' cognitive, personal, and professional development,' *Science Education*, vol. 91, no. 1, pp. 36–74, 2007. DOI: 10.1002/sce.20173. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sce.20173.

[2] S. Hussain, K. A. A. Gamage, M. H. Sagor, F. Tariq, L. Ma and M. A. Imran, 'A systematic review of project allocation methods in undergraduate transnational engineering education,' *Education Sciences*, vol. 9, no. 4, 2019, ISSN: 2227-7102. DOI: 10.3390/educsci9040258. [Online]. Available: https://www.mdpi.com/2227-7102/9/4/258.

[3] R. A. I'Anson, K. A. Smith *et al.*, 'Undergraduate research projects and dissertations: Issues of topic selection, access and data collection amongst tourism management students,' *Journal of Hospitality, Leisure, Sport and Tourism Education*, vol. 3, no. 1, pp. 19–32, 2004.

[4] T. Barber and V. Timchenko, 'Student-specific projects for greater engagement in a computational fluid dynamics course,' *Australasian Journal of Engineering Education*, vol. 17, no. 2, pp. 129–138, 2011. DOI: 10.1080/22054952.2011.11464055.

[5] N. I. S. I. Jailani, A.-F. Mubarak Ali and S. Ngah, 'Final year project allocation system techniques: A systematic literature review,' in *2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2022, pp. 99–104. DOI: 10.1109/ISCAIE54458.2022.9794501.

[6] J. Harland, S. Pitt and V. Saunders, 'Factors affecting student choice of the undergraduate research project: Staff and student perceptions,' *Bioscience Education*, vol. 5, no. 1, pp. 1–19, 2005. DOI: 10.3108/beej.2005.05000004. [Online]. Available: https://doi.org/10.3108/beej.2005.05000004.

[7] R. Calvo-Serrano, G. Guillén-Gosálbez, S. Kohn and A. Masters, 'Mathematical programming approach for optimally allocating students' projects to academics in large cohorts,' *Education for Chemical Engineers*, vol. 20, pp. 11–21, 2017, ISSN: 1749-7728. DOI: https://doi.org/10.1016/j.ece.2017.06.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1749772817300301.

[8] D. Kazakov, 'Coordination of student project allocation,' 2001. [Online]. Available: https://www-users.cs.york.ac.uk/kazakov/papers/proj.pdf.

[9] Y. Cheung, G. Meng Hong and K. Keng Ang, 'A dynamic project allocation algorithm for a distributed expert system,' *Expert Systems with Applications*, vol. 26, no. 2, pp. 225–232, 2004, ISSN: 0957-4174. DOI: https://doi.org/10.1016/S0957-4174(03)00137-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417403001374.

[10] M. Hasan, K. M. Sahari and A. Anuar, 'Implementation of a new preference based final year project title selection system for undergraduate engineering students in uniten,' in *2009 International Conference on Engineering Education (ICEED)*, IEEE, 2009, pp. 230–235.

[11] D. J. Abraham, R. W. Irving and D. F. Manlove, 'Two algorithms for the student-project allocation problem,' *Journal of discrete algorithms*, vol. 5, no. 1, pp. 73–90, 2007.

[12] L. Wan-hong, 'A web-based project allocation system interim report,' University of Hong Kong, 2017.

[13] R. B. Abdul Wahab, 'Final year project online management system (fypos): Client and server design,' 2012.

[14] P. Johannesson and E. Perjons, *An Introduction to Design Science*. Springer Cham, Jul. 2014, ISBN: 978-3-030-78132-3. DOI: 10.1007/978-3-030-78132-3.

[15] A. Dresch, D. P. Lacerda and J. A. V. Antunes, 'Design science research: A method for science and technology advancement,' in *Design Science Research*. Springer International Publishing, 2015, pp. 67–102, ISBN: 978-3-319-07374-3. DOI: 10.1007/978-3-319-07374-3_4. [Online]. Available: https://doi.org/10.1007/978-3-319-07374-3_4.

[16] D. Zowghi and C. Coulin, 'Requirements elicitation: A survey of techniques, approaches, and tools,' in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer, Berlin, Heidelberg, 2005, pp. 19–46. DOI: 10.1007/3-540-28244-0\{_}2.

[17] PM Solutions, 'Strategies for project recovery,' Tech. Rep., 2011. [Online]. Available: https://www.pmsolutions.com/resources/view/strategies-for-project-recovery/.

[18] A. Davis, O. Dieste, A. Hickey, N. Juristo and A. M. Moreno, 'Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review,' in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006, pp. 179–188. DOI: 10.1109/RE.2006.17.

[19] M. O. Ahmad, J. Markkula and M. Oivo, 'Kanban in software development: A systematic literature review,' in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 2013, pp. 9–16. DOI: 10.1109/SEAA.2013.28.

[20] J. Nielsen, 'Usability metrics,' Tech. Rep., 2001. [Online]. Available: https://www.nngroup.com/articles/usability-metrics/.

[21] J. Nielsen and J. Levy, 'Measuring usability: Preference vs. performance,' *Communications of the ACM*, vol. 37, no. 4, pp. 66–75, Apr. 1994, ISSN: 0001-0782. DOI: 10.1145/175276.175282. [Online]. Available: https://doi.org/10.1145/175276.175282.

[22] S. K. Card, T. P. Moran and A. Newell, *The Psychology of Human-Computer Interaction*. Laurence Erlbaum Associates, Inc., 1983, ISBN: 0-89859-859-1.

[23] S. K. Card, T. P. Moran and A. Newell, 'The keystroke-level model for user performance time with interactive systems,' *Communications of the ACM*, vol. 23, no. 7, pp. 396–410, Jul. 1980, ISSN: 0001-0782. DOI: 10.1145/358886.358895. [Online]. Available: https://doi.org/10.1145/358886.358895.

[24] I. S. MacKenzie, 'Fitts' law as a research and design tool in human-computer interaction,' *Human–Computer Interaction*, vol. 7, no. 1, pp. 91–139, 1992. DOI: 10.1207/15327051hci0701_3. [Online]. Available: https://doi.org/10.1207/s15327051hci0701_3.

[25] K. Pernice, 'Scanning patterns on the web are optimized for the current task,' 2017. [Online]. Available: https://www.nngroup.com/articles/eyetracking-tasks-efficient-scanning/.

[26] C. M. Barnum, *Usability testing essentials: ready, set... test!* Morgan Kaufmann, 2020.

[27] K. Whitenton, 'Unmoderated user tests: How and why to do them,' Tech. Rep., 2019. [Online]. Available: https://www.nngroup.com/articles/unmoderated-usability-testing/.

[28] D. R. Chand and A. M. Chircu, 'Chapter 3: Business process modeling,' in *Business enterprise, process, and technology management: Models and applications*, V. Shankararaman, J. Leon Zhao and J. K. Lee, Eds. Business Science Reference, 2012, pp. 187–212, ISBN: 9781466602502.

[29] M. von Rosing, S. White, F. Cummins and H. de Man, *Business process model and notation – bpmn*, 2015.

Appendix A

User Evaluation Questions

This appendix contains the full text of the user questionnaires used to generate responses in chapter 8. This includes the exact task instructions, as well as full text and response options for each question. A response option of "Other" is accompanied by a text field to let users specify. Some questions included URLs linking to specific web pages; these links have been removed, as some of them pointed to internal university resources.

A.1 Student Questionnaire

This questionnaire was given to our student testers. Each section description following this point is a verbatim quote from the questionnaire.

Pre-Task Questions

These questions are to help us understand your familiarity and history with project assignment systems. You may leave any questions you do not wish to answer blank.

Several of these questions make reference to the existing IDI project assignment system. This system is not fully accessible due to ongoing changes, but the project listing component can still be found at [link removed].

- Did you use the IDI specialization/master project assignment system to find a thesis project to work on?
  - *Response Options:* Yes, No
- How easy was the IDI system to use?
  - *Response Options:* Rank from 1 (Very easy to use) to 5 (Very hard to use)
- Did you already know the professor you chose to work with before you applied for their project?
  - *Response Options:* Yes, No

- How many projects did you apply for?
- Has the large number of projects in the IDI system caused you any trouble?
    - *Response Options:* Yes, No, Other
- What did you like about the IDI project system?
- What did you dislike about the IDI project system?

Task 0 - Get an Account

You will use an account that has already been populated with some data in order to complete these tasks. To begin, select the "Get Student Account" button at the top right. This will take a few seconds to complete as the system generates data for your account.

Task 1 - Profile

Before you start searching for projects, you should set up your profile to reflect your interests. This profile will help professors get to know your background and augment the information you include in your applications. Visit your profile and add tags to reflect your interests.

- Do you like the idea of having a profile that's visible to professors? What information would you want it to show?
- Do you think having a profile will save you time when applying to projects?
    - *Response Options:* Yes, No
- Was there anything that was confusing or did not work correctly?

Task 2 - Finding a Project

You want to apply to a project that involves games. You also want to work with Lucas Li, a professor whose course you really enjoyed. Find some games-related projects posted by Lucas Li, then visit the details page of the one with the least applicants.

- Did you use the filtering system to find these projects?
    - *Response Options:* Yes, No
- How easy was it to find the projects you were looking for?
    - *Response Options:* Rank from 1 (Very easy) to 5 (Very hard)
- By default, projects on the main page are filtered according to your academic history and interests as defined in your profile. Do you think this would be helpful when trying to find masters topics?
- If you were using this tool to pick your masters topic, would you use the filtering system to search for relevant projects?
    - *Response Options:* Yes, No, Other

- Is it helpful to see the number of applicants and offers on each project? What do you like or dislike about that?
- Was there anything that was confusing or did not work correctly?

Task 3 - Creating an Application

Now that you've found an interesting project, apply to it. If you'd like, choose to automatically accept an offer to work on this project. The content of your application does not matter. For example, you can paste "This project sounds interesting to me! I worked on something similar last summer."

- Is this process comparable to how you got a project in the IDI system?
  - *Response Options:* Yes, No, Other
- How does applying to projects in this system compare to applying via email?
- Do you like the idea of having formal applications within this system? Why or why not?
- If you chose to automatically accept an offer, then this project becomes your final choice the moment the professor offers it to you. Is this a feature you would make use of?
  - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 4 - Finding a Group Project

You also want to apply to a project with some friends, and none of you want to perform user evaluation. Find any project that accepts groups but doesn't include user evaluation, then visit its details page.

- If you were using this tool to pick your masters topic, would you find it useful to filter out certain tags or requirements? If not, why not?
- Was it intuitive to combine positive and negative filters?
  - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 5 - Creating a Group Application

Create an application to the project you just found, adding your friends Alice Alaine ("alicea") and Ingrid Innsmouth ("ingridi") to the application as group members. The content of this application does not matter. For example, you can paste "Alice, Ingrid, and I worked on a project like this in one of our classes last year. We'd really like to take on this project!"

- Was it easy to add the other students to your application?
  - *Response Options:* Yes, No, Other

- How does doing group applications this way compare to doing them via email?
- Was there anything that was confusing or did not work correctly?

Task 6 - Ranking Applications

At this point you have applied to a few projects. Now would be a good time to manage those applications. Visit your applications page and assign ranks to them in any order you want.

- Was it easy to tell which application had which rank?
    - *Response Options:* Yes, No, Other
- In this system, lower-ranked applications are automatically moved up when a higher-ranked application is rejected. Is that something you'd expect to happen?
    - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 7 - Responding to Comments

Lucas Li asked for more details on your academic history. Reply to his question (the content of your reply does not matter). For example, you can write "Yes, i had that topic in recommender systems last year."

- How does discussing a project with a professor in this system compare to doing it via email?
- Would you feel more comfortable using this system to communicate with a professor about a project than emailing them directly?
    - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 8 - Accepting an Offer

You have an offer on one of your applications! Accept that offer.

- Was it clear which application had an offer?
    - *Response Options:* Yes, No, Other
- After accepting an offer, your other applications were automatically revoked, and you were removed from the group application you were a part of. Is that something you'd expect to happen?
    - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 9 - Complete the Test

Thank you for your time! To conclude the test, press the "Finish Test" button in the top right. This will log you out of your test account and remove your edits from the system.

Post-Test Questions

Now that you are familiar with the system and have performed some core tasks in it, these questions aim to assess the system's successes and failures in providing useful accommodations for project matching.

- How difficult were the tasks you were asked to complete in this test?
  - *Response Options:* Rank from 1 (Very easy) to 5 (Very hard)
- How easy to use was this system?
  - *Response Options:* Rank from 1 (Very easy to use) to 5 (Very difficult to use)
- How did this system compare to the existing IDI solution?
- Was there anything that surprised you about the tool during the tasks you completed?
- What problems do you think you would encounter if you used this system to find a thesis project in a real semester?
- Do you feel this tool would help you find interesting projects more quickly? Why or why not?
- Was there anything you felt was lacking or missing from the system?

A.2 Professor Questionnaire

This questionnaire was given to our professor testers. Each section description following this point is a verbatim quote from the questionnaire.

Pre-Task Questions

These questions are to help us understand your familiarity and history with project assignment systems. You may leave any questions you do not wish to answer blank.

Several of these questions make reference to the existing IDI project assignment system. This system is not fully accessible due to ongoing changes, but the project listing component can still be found at [link removed]

- How much experience do you have with the IDI specialization project assignment system?
  - *Response Options:* Rank from 1 (Used it very little) to 5 (Used it for years)

- Have you used any other systems for project assignment before? If so, what were they?
    - *Response Options:* No, Other
- What do you like about the IDI project system (or other systems you've used)?
- What do you dislike about the IDI project system (or other systems you've used)?
- Has the large number of students and projects caused you any trouble with the IDI system?
    - *Response Options:* Yes, No, Other
- What is your primary method for managing project applications and questions from students currently?

Task 0 - Get an Account

You will use an account that has already been populated with some data in order to complete these tasks. To begin, select the "Get Professor Account" button at the top right. This will take a few seconds to complete as the system generates data for your account.

Task 1 - Create a Project

Now that you have logged in, you want to create a new project listing. The title and description of the project are not important, but you should try to tag it appropriately for whatever topic you enter. Your project should be open for applications and have a capacity of two groups. It should only accept groups, not single students.

As an example, the project could be called "Collaborative Games for IT and Sustainability", with a description like "There is a growing effort to integrate sustainability in IT education. As part of this larger effort, this task will focus on the development of collaborative games for increasing awareness about the role of IT in reaching the UN Sustainable Goals." This project would have tags like "Software Systems", "Games", and "Environment". You are free to use these example values verbatim if you wish.

- How did the process of creating a project in this system compare to project creation/listing in any project tracking systems you have used in the past?
- Was there anything you felt was missing from the project creation process?
- If you used this tool to track all of your projects, could you see yourself tagging all of them using this system?
    - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 2 - Edit your Profile

As a professor, you have a profile in the system that can help you communicate your interests to students. Users viewing your new project may want to know more about the professor they might be working with, so you should update your profile. Edit your profile to include a few tags that match your academic interests. For example, "Education", "Virtual Reality", or "Machine Learning".

- Were you able to give yourself tags that accurately reflect your project topics and interests?
    - *Response Options:* Yes, Somewhat, No, Other
- Was there anything you felt was missing from your profile page?
- Would you rather direct students to an external profile instead? If so, why?
- Was there anything that was confusing or did not work correctly?

Task 3 - Survey your Applications

Two of your projects have applications from students for you to review. Inspect the applications on both projects, but do not take any actions on them yet. Take note of the ranks of the applications and the students involved.

- How easy is it to tell who is applying to each project and what state their application is in?
    - *Response Options:* Rank from 1 (Very easy) to 5 (Very difficult)
- Were you able to find the projects with open applications quickly?
    - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Task 4 - View an Applicant's Profile

To get a better picture of one of your applicants, navigate to their profile and take a look at their interest and academic history tags.

- Were you able to find the student's profile easily?
    - *Response Options:* Yes, No, Other
- Did the student's profile, along with their written application, provide enough information for you to evaluate whether you would offer them the project? If not, what was missing?
- Was there anything that was confusing or did not work correctly?

Task 5 - Respond to a Comment

On the Natural Language to Long-Range Path Plans project in your project list, a student has asked for clarification on whether you will accept group applications.

Navigate to that project's page and send the student a reply (the content of your reply does not matter).

- How does answering followup questions and comments in the application compare to doing it via email? Would you have preferred it if the student emailed you?
- Was there anything that was confusing or did not work correctly?

Task 6 - Manage your Applications

For the Long-Range Path Plans project, reject the student who ranked it the lowest, and add a comment explaining why (the content of this comment does not matter). Then send an offer to the student who ranked the project the highest. Next, check the applications to your other project about Predicting Hospital Readmission. It should now have only one viable application for you to accept.

- With this system, all applications to a project are listed together on the same page. Did that change the way you approached evaluating the applications?
- When a student accepts an offer, their other applications are automatically retracted. Do you think this would make managing a large number of applicants easier?
    - *Response Options:* Yes, No, Other
- When a student's application is rejected, their lower-priority applications are automatically increased to fill the gap. Do you think this would make managing a large number of applicants easier?
    - *Response Options:* Yes, No, Other
- In the IDI system as well as this system, do you find application priorities useful to you when selecting students?
- Was there anything that was confusing or did not work correctly?

Task 7 - Close a Project

The group you offered the Long-Range Path Plans project to has automatically accepted your offer, which means you've hit the set capacity for that project. Close the project so that no more students can send in applications.

- Would you prefer to have the system close your projects automatically when they reach the capacity you chose?
    - *Response Options:* Yes, No, Other
- When a project is closed, it is removed from the main project list for students, but students can still visit its details page if they have the link. Is that what you'd expect to happen?
    - *Response Options:* Yes, No, Other

- Would you prefer for students to still be able to see closed projects on the main page? If so, why?
- Was there anything that was confusing or did not work correctly?

Task 8 - Clone an Old Project

There is another closed project from the previous year in your history that you would like to list again this year. Its title is "Time- and Space-Efficient Aggregate Range Queries over Encrypted Databases (2021)". Copy it and edit the year to match the current time, then set it as a draft. Afterwards, publish the draft.

- Do you feel that cloning and editing an older project is preferable to creating a new one? How did this process compare to the new project creation process?
- Is it important to you to be able to create and edit listings as drafts before publishing them?
  - *Response Options:* Yes, No, Other
- Was there anything that was confusing or did not work correctly?

Post-Test Questions

Now that you are familiar with the system and have performed some core tasks in it, these questions aim to assess the system's successes and failures in providing useful accommodations for project matching.

- How difficult were the tasks you were asked to complete in this test?
  - *Response Options:* Rank from 1 (Very easy) to 5 (Very difficult)
- How easy to use was this system?
  - *Response Options:* Rank from 1 (Very easy to use) to 5 (Very difficult to use)
- Was there anything that surprised you about the tool during the tasks you completed?
- What problems do you think you would encounter if you used this system for your thesis project management in a real semester?
- Do you feel this tool would help you manage a large number of students and projects better?
  - *Response Options:* Yes, No, Other
- Was there anything you felt was lacking or missing from the system?
- How did this system compare to the existing IDI solution, or other project listing systems you've used?

Odin Johan Vatne

Project Assignment Software

NTNU
Norwegian University of
Science and Technology