

Bachelor Project System (BEPsys)

by

Sarah Bashirieh - 1523259

Nima Rahbari - 1515659

Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

October 2013

Abstract

In this report we represent the detailed procedure of developing a web system for the faculty of EEMCS of Delft University of Technology. The *Bachelor Project System* - BEPsys¹- makes it possible for companies to propose projects and for students to find a project in an easy and convenient way. In chapter one we have described a short research about similar existing systems, methods and web application frameworks to start a project, and we made a comparison between them. Chapter two describes the design of the database and the front-end of the system and different decisions that we made. In chapter three we added more details about the implementation of the system. In chapter four the usability testing of the system and the method we used to test is explained. Finally the last chapter contains our conclusion of the project in short.

¹BEPsys stands for Bachelor Eindproject Systeem

Contents

Abstract	i
-----------------	----------

List of Figures	v
------------------------	----------

1 Orientation	1
1.1 Introduction	1
1.2 Current system investigation	1
1.3 Future system	2
1.4 Comparison with existing systems	3
1.4.1 Customer Relationship Management systems	3
1.5 Choice of software methodology	5
1.5.1 Adaptation to change	5
1.5.2 Increased productivity	5
1.5.3 Variable requirements	6
1.6 Implementation	6
1.6.1 Back-end	6
1.6.1.1 Choice of framework	6
1.6.1.2 Framework comparison	9
1.6.2 Front-end	9
1.7 Deployment	10
2 Design	11
2.1 Introduction	11
2.2 Main activities in the application	11
2.2.1 Company wants to propose a project	11
2.2.2 Student wants to do a project	12
2.3 Back-end of the system	12
2.3.1 Database design	12
2.3.1.1 The domain model	12
2.3.1.2 Entities	13
2.3.1.3 User roles	13
2.3.1.4 First approach	14
2.3.1.5 Second approach	14
2.3.1.6 Final approach	15
2.4 Front-end of the system	16

2.4.1	Introduction	16
2.4.2	Use cases	17
2.4.3	Register	20
2.4.4	User's home page	21
2.4.5	Projects page	21
2.4.6	Ads page	22
2.4.7	Coordinator's special pages	23
2.4.7.1	Static information pages	23
2.4.8	Informing users	24
3	Implementation	25
3.1	Introduction	25
3.2	Back-end	25
3.2.1	Models	26
3.2.1.1	DataSources	27
3.2.1.2	Behaviors	27
3.2.1.3	Data association	27
3.2.1.4	Data validation	28
3.2.2	Views	29
3.2.3	Controllers	29
3.2.3.1	Components	30
3.2.3.2	PagesController	31
3.2.4	Dispatcher	31
3.2.5	Routes	31
3.2.6	AJAX	31
3.3	Front-end	32
3.3.1	jQuery	32
3.3.2	Bootstrap	33
3.3.2.1	Dropdown button	33
3.3.2.2	Tooltip	33
4	Testing	34
4.1	Introduction	34
4.2	Test Plan	34
4.2.1	Goals	34
4.2.2	Participants	35
4.2.3	Test Method	35
4.2.4	Tasks	35
4.2.5	Test Environment	35
4.2.6	Evaluation	36
4.2.6.1	User interface	40
4.2.6.2	Effectiveness	43
4.2.6.3	Overall satisfaction	45
4.2.6.4	Conclusion	47
5	Conclusion	48

A	Project Plan	49
A.1	Introduction	49
A.1.1	Background	49
A.2	Project description	50
A.2.1	Introduction	50
A.2.2	The client	50
A.2.3	Contacts	50
A.2.4	Problem description	50
A.2.5	Goal	50
A.2.6	Product	50
A.2.7	Preconditions	50
A.3	The approach	51
A.3.1	Introduction	51
A.3.2	Software Engineering Methodology	51
A.3.3	Technical details	51
A.3.4	Proceedings and schedule	51
A.4	Quality Assurance	52
A.4.1	Introduction	52
A.4.2	Quality	52
A.4.2.1	Evaluation	52
A.4.2.2	Versioning	52
A.4.2.3	Pilots	52
B	Requirements Elicitation	53
B.1	Introduction	53
B.2	Current system	53
B.3	Proposed system	53
B.3.1	overview	53
B.3.2	Stakeholder analysis	54
B.4	Functional requirements	54
B.4.1	Company contacts	54
B.4.2	Coordinators	55
B.4.3	Supervisors	56
B.4.4	Students	57
B.5	Nonfunctional requirements	58
B.5.1	Usability	58
B.5.2	Security	58
B.6	Scenarios	58
B.6.1	Company contacts	58
B.6.2	Coordinators	59
B.6.3	Supervisors	60
B.6.4	Students	60
B.7	Flow charts	62
B.7.1	User register	62
B.7.2	Company	64
B.7.3	Coordinator	65
B.7.4	Student	67

Bibliography

69

List of Figures

1.1	Scrum	5
1.2	Disciplined agile requirements change management process	6
1.3	How the MVC pattern works	7
2.1	The approach without a User table.	14
2.2	Bachelor Project System database EER diagram	15
2.3	Use case diagram of company contact	17
2.4	Use case diagram of the student	18
2.5	Use case diagram of the coordinator	19
2.6	Use case diagram of the supervisor	20
2.7	Registration page	21
2.8	List of projects in student view	21
2.9	A project's page	22
2.10	Ad page	23
2.11	List of student available for coordinators	23
2.12	Registration successful alert	24
2.13	Delete a student from the system	24
3.1	A typical CakePHP request	26
3.2	Models of the system	26
3.3	Controllers of the system	30
3.4	Bootstrap dropdown button	33
3.5	Bootstrap tooltip	33
4.1	Navigation	40
4.2	Understandability	40
4.3	Website's look	40
4.4	Help messages	41
4.5	Clarity of the forms	41
4.6	Website's organization	41
4.7	UI satisfaction	42
4.8	Task performance steps	43
4.9	Task performance time	43
4.10	Website's learning curve	43
4.11	Productivity	44
4.12	Helpful to perform the task	44
4.13	Use in future	45
4.14	Recommend to colleagues	45

4.15 Ease of use	45
4.16 Satisfaction about the whole website	46
B.1 Definition of icons	62
B.2 Registration (except supervisor)	63
B.3 Proposes/edits project	64
B.4 Closes a completed project	65
B.5 Fills in presentation date	66
B.6 Project approval	66
B.7 Post Ad	67
B.8 Join/leave project, Invite supervisor, coordinator approval	68

Chapter 1

Orientation

1.1 Introduction

This section is the result of the orientation phase. In order to obtain a better understanding of the problem, we interviewed the client¹ several times. We compared a number of existing systems which are similar to our product. With the result of interviews and the research about the existing system we made workflow examples of the future system². We will discuss the software methodology that was used in the project. Furthermore we choose a suitable web application framework for our project based on our desired features and comparing possible frameworks.

1.2 Current system investigation

The current bachelor project system is part of the blackboard environment, and it is treated like a course. There is no direct interaction between the company and the bachelor project system. The interaction is done via the bachelor coordinator. Companies interested to work with the university need to contact the coordinator, who then will inform the company about the project proposal layout. A company needs to make a document which fulfils the desired requirements. After sending the document to the coordinator, the project proposal is uploaded in the blackboard page. If the suggested project does not meet the requirements of TU Delft the coordinator will contact the company via email or phone in order to change the project description. In the current system there are three types of project:

¹See Appendix A section 2.2

²see appendix B section 3

- Open projects: projects available to be chosen by students.
- Active projects: projects which are not available any more to choose, with contact information about the supervisor and student team.
- Completed projects: projects that were successfully finished, with contact information about the supervisors and student team.

When a project is accepted by the coordinator, it is put in the open projects list by the coordinator. In order to start a project, the students contact the coordinator and a supervisor about a chosen project. They must print the project proposal form and add their name and the supervisor name to the form. This form should be signed by students and company contact person. The students submit the form to the coordinator for approval. After approval the project is put in the active project list with the corresponding information. If the project is completed it will be moved to the completed project list. In the current system there is an opportunity for the students to find a suitable group. This functionality is called *Group Finder*³. The students are able to publish an ad⁴. The students may send desired message to the coordinator, which will be uploaded in the Group Finder by the coordinator.

1.3 Future system

In this section we give an overview of functionalities of the future system that fulfils the client's needs. The structure of the system will be like the current system with project lists and group finder functionality. In BEPsys, due to the direct interaction between users and the system, the printing and signing the proposal as mentioned in previous section will be reduced. The company can upload the project proposal directly in the system. When a company proposes a project, the coordinator will be notified about it. After approval the project proposal will be added to the open project list. In this system we have another type of project besides open, active and completed projects:

- Proposed projects: a list of proposed projects by companies, to be approved by the coordinator. These projects are visible to the company who proposed it and the coordinator. After approval the status of the project will be changed to open.

³A part of the system where students can post ads to seek a partner or group.

⁴An ad in this context is a message that can be posted by students to help them find a group or a partner.

The students can join a desired project. When all the group members have joined and a supervisor is invited to the group, they can request approval from the coordinator. Once approved, the project will be moved to the active projects list. The Group Finder functionality will be almost the same as the Group Finder on the blackboard, but students can now post an ad without contacting the coordinator.

In the new system users⁵ can edit their content. For instance companies can edit their project information and students can edit their registration information in the system, so there is no need to send an email to the coordinator for each minor change. However the coordinator will be notified by email about each change. In the future a search functionality could be implemented in the system for the coordinator to have a better overview about the different projects and project members. For example the name of the student can be searched in the system for obtaining information such as contact information and group number. This feature is however not a must have.

1.4 Comparison with existing systems

In this section a comparison is given between two existing *Customer Relationship Management* systems with our system, BEPsys. Customer Relationship Management systems automate interaction between companies and their customers. We have decided to investigate CRM systems, because BEPsys is also a CRM system where TU Delft is the company and students and companies play the role of customers. First, the features of each CRM system are given and second some of them are compared in a table.

1.4.1 Customer Relationship Management systems

We have investigated two open source CRM systems, namely *Zurmo* [1] and *Splendid-CRM* [2] that have the following features:

- Contact management: Both systems let users⁶ view contacts⁷ in the system and there is a search function to find any contact.
- Activity management: A list of actions that have to be done or were done is viewable by the users, i.e. activities, notes and meetings. This way there is a summary available which is easy to view.

⁵See appendix B section 3.2

⁶Staff of the organization

⁷Customers of the organization

- **Reward system:** A reward program to keep users more involved and to teach them to use the system better or let them try new functionality. Users can for example earn achievements or badges after they have completed a particular action. This feature is found only in Zurmo system.
- **Reporting:** Reports about all records in the system. These reports can be saved or exported or viewed in form of charts.
- **Security:** Both systems also take care of different roles⁸, permissions and policies. i.e. who can view a particular document. This feature is especially important when there are different roles that have different permissions in a system.
- **Product management⁹:** Lets users create product catalogs and the items of the product with their quantities.
- **Internationalization:** Both systems offer many language packs and support many timezones, so that the calendars can be customized per time zone.
- **Personalizing feature:** Zurmo has custom fields and layouts that allow developers to customize the environment. Furthermore users can personalize their home page which exists of different panels. So the users are able to delete a panel if it is not usable for them.
- **Portability:** Both system are compatible with different browsers. Zumro also runs on Android and iOS.

In the following table we have compared features that are important for us in Zurmo, SplendidCRM and BEPsys.

CRM System/Features	Zurmo	SplendidCRM	BEPsys
Activity management	yes	yes	yes
Product management	yes	yes	yes
User management	yes	yes	yes
Security	yes	yes	yes
Portability	yes	no	yes
Reporting	yes	yes	no

With respect to the comparison we came to the conclusion that except minor differences they are very much alike.

⁸For instance admin or regular user

⁹Product can be any product or service that is sold by the company

1.5 Choice of software methodology

In the beginning of the project we looked at Scrum method of Agile software development framework.

In this project the requirements are not known in the first place, we had several interviews with the client to gather all the requirements. Due to variability of the requirements we have decided to use Scrum method.

At the beginning of every cycle, which is a short period of working on a specific functionality, we discuss what has to be done and when the deadline is, the so called *sprint backlog*. When a cycle starts, we will have daily meetings where we very shortly talk about what we have done and what we are doing right now. In the next subsections we explain in detail why we have chosen Agile for this project.



FIGURE 1.1: Scrum

1.5.1 Adaptation to change

Doing the project in cycles or sprints enables us to be ready for possible changes in the requirement. After each sprint the work can be shown to and user-tested by the client and changes could be made on time to meet the client needs. This is much better than finishing the project entirely, risking to have to do the project all over again if the client is not happy with the result.

1.5.2 Increased productivity

Daily meetings encourage the developers to discuss what they are doing and what problems they are facing. This prevents the developers from wasting much time on problems

which otherwise would be solved in a couple of minutes. Also if someone has done something the way it should not be done, it can be detected early.

1.5.3 Variable requirements

As we mentioned before the requirements are gathered from several interviews with the client. During the project, changes in the requirements are possible, therefore we need to be prepared. We can benefit from The Agile Change Management Process which is suited for this kind of projects[3]. It focuses on the priority of the requirements, meaning the requirement with the highest priority are going to be implemented first. Each part of the project can be re-prioritised at any time. The priority of each part is decided by the client or by ourselves. For example if the client would like to see a part of project first or if many parts of the project are dependent to a particular part, we will increase the priority.

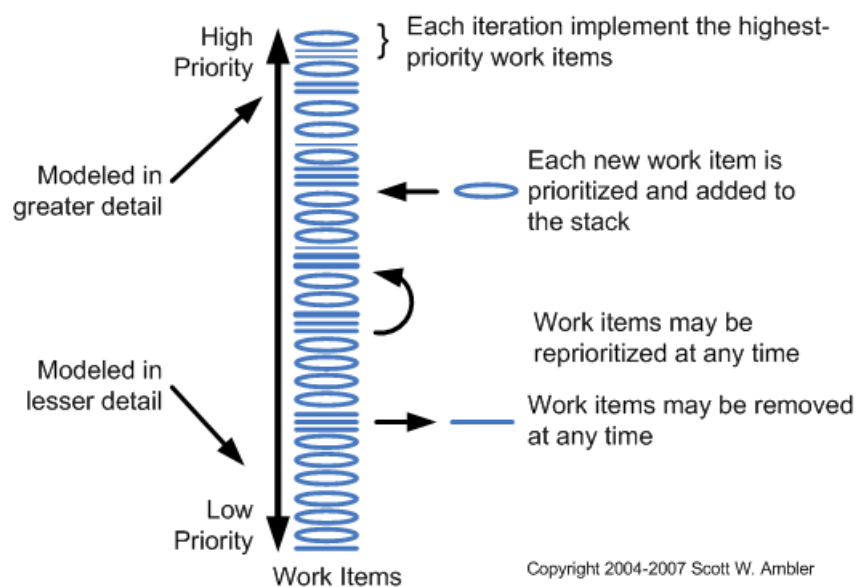


FIGURE 1.2: Disciplined agile requirements change management process

1.6 Implementation

1.6.1 Back-end

1.6.1.1 Choice of framework

We think it is important that the code of this project is maintainable and upgradable and that the project should be finished within our desired time frame. In order to achieve the

aforementioned we decided to use a web application framework which will help us concentrate on the specific functionality and components instead of wasting time on making general functionality which is available in a framework. There are many considerations when it comes to choosing a suitable web application framework to build a web application. In our opinion the following considerations are important in choosing a framework:

Software pattern

Almost all web application frameworks use the MVC pattern. MVC stands for Model, View and Controller. Using the MVC pattern helps you keep your data (Model), the logic of your application (Controller) and the user interface (View) separate. It also makes it easier to re-use code as different components are separate and there is more flexibility when one wants to apply a change in the system. In other words, MVC improves code maintenance in the future.

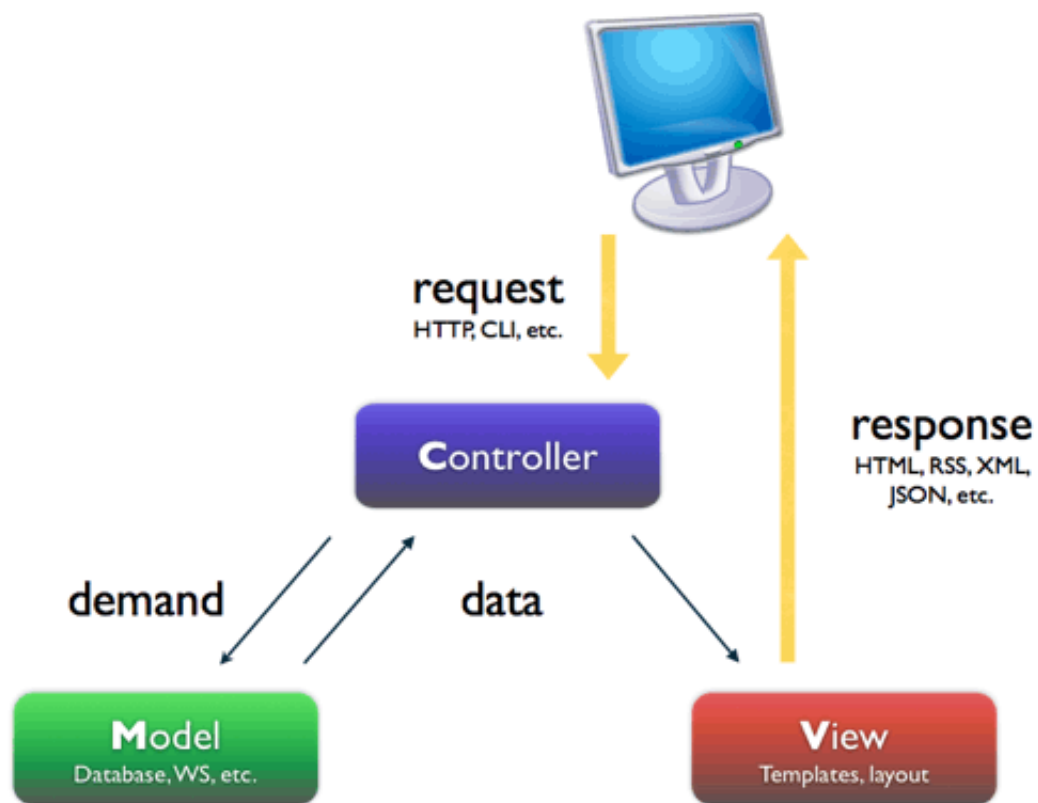


FIGURE 1.3: How the MVC pattern works

Learning curve

Every framework has its own structure and conventions¹⁰. Some of them strictly follow the general conventions, while others may sometimes do things in their own special way.

¹⁰For instance the way how controllers and models should be named in a framework.

The more they deviate from the general conventions the steeper the learning curve is and the more time it takes to get familiar with the environment and up to speed.

Core features

It is important to know what the core features of a framework is, so you choose the one with the features you would like to have. The following features are important to us:

- Data validation: making sure input data have the right format.
- Data sanitization: cleaning input data from potential dangerous content.
- Authentication: recognizing the user.
- Authorization: what the user can access and what not.
- AJAX: built-in AJAX functionality.
- Templating: views in different formats

Documentation

The quality of the documentation plays a very important role in the success of a web application framework. The better explained the documentation the more productive developers will be and the more they support the framework. This way it will be easier for new developers to pick up the framework relatively fast.

Community

No matter how well a framework is documented, you will run into problems which you simply cannot solve without the help of other experienced developers. This is simply because documentation cannot cover all possible problems you will come across in the framework. With a bigger community it is easier as a newbie to start coding in a framework and that makes the learning curve less steep, which then will result in more time to work on the project.

License

First of all it is important to know what kind of licence a certain framework is distributed under. Some licenses allow you to use the framework for commercial purposes if you mention the copyright owners in the code or have a copy of the license agreement file some where in the system. While most of the web frameworks allow you to use the framework in a commercial environment, it is important to first know what the license is before you start using it to prevent surprises. Some of the most used free software licenses for web application frameworks are MIT license, GPL license and BSD license.

1.6.1.2 Framework comparison

We have decided to compare 3 popular frameworks: CakePHP which was written in PHP, Django written in Python and Ruby-on-rails written in Ruby. After comparing we will draw a conclusion based on a mix of our desired features and the comparison.

When someone is already familiar with web development and there is not much time, it could be wise to just use CakePHP and not look further. In case of *Ruby on Rails*, not only there is a new programming language to learn - Ruby - but there are also a lot of framework specific stuff to learn. Hence the use of Ruby on Rails is only recommended when there is time to learn a programming language from scratch.

Django, written in Python, requires one not only to know Python, but also the so called *Django Template Language*, which is not really straight-forward and requires time to get used to. Below is a comparison table, which compares CakePHP, Django and Ruby on Rails on features that we would like to have.

Framework/Characteristic	CakePHP	Django	Ruby on Rails
Data validation	✓	✓	✓
Data sanitization	✓	✓	✓
Pattern	MVC	MTV(model template view)	MVC
Ajax	AjaxHelper	✗	JavaScriptHelper
Authentication	✓	✓	✗
Authorization	✓	✓	✗
Learning curve	Slight	Slight	Steep
Documentation level	High	High	Very high
Community	Huge	Big	Big

CakePHP has all the features that we want. Furthermore we have experience both in PHP as a language and CakePHP as a web application framework. The two other frameworks require us to learn a programming language from scratch, which will take much more time than we can afford in this project.

1.6.2 Front-end

BEPsys should facilitate the interaction between companies and the university. The front-end implementation plays an important role here. It should be fast, reliable and also simple to use, in short the front-end design should be user friendly. Keeping in mind the time span of the project and the need of a robust front-end implementation we decided to use Bootstrap which is a toolkit for developing user interfaces for web applications. The technical advantages of Bootstrap, which make it suitable for our project are:

- Fast: Bootstrap is built with Less which is a flexible pre-processor that offers much more power and flexibility than regular CSS[4].
- Customizable: it has the ability to be tailored for the project in short time.
- Cross-browser: there is no need to test whether tables and forms are compatible with different browsers.
- Device compatibility : it has wide support for different devices including mobile phone and tablets
- Responsive: it adapts itself to different platforms.

1.7 Deployment

The client has chosen *Heroku* to deploy the application. Heroku is a cloud platform as a service (*Paas*) which supports several programming languages. Heroku has many advantages including:

- No need to manage server infrastructure.
- Easily scalable.
- Support for variety of database management systems.
- Free basic account.
- Only pay for what you use.

In short, with Heroku developers can deploy their applications written in several supported programming languages with very little configuration. Heroku takes care of everything server-side except for special customizations.

Chapter 2

Design

2.1 Introduction

In this chapter the design of the system will be explained in details. Our domain model will be explained as well as the front-end of the system along with what choices we have made throughout this process and when relevant, why we have chosen for a certain approach or method. In the subsection dedicated to the front-end, different user interactions and important pages are shown.

2.2 Main activities in the application

In order to understand the benefits of the new system we discuss two most important activities in the system from the start to the end. For detailed explanation and charts see appendix C.

2.2.1 Company wants to propose a project

A company wants to propose a new project. The system can be accessed via an email address. If it is the first time that a company accesses the system, the contact information form must be filled in. Whereafter the company is redirected to the company home page. In this page the company can propose a new project by filling in the project proposal form. After submitting the form an email is sent to the company about the proposed project. See figure [B.3](#)

The coordinator receives an email about the new proposed project. The information about the project can be seen in the email, so the coordinator is able to approve the

project. If the project proposal needs change the coordinator will contact the company outside of the system or remove the project. See figure B.6. A company contact can edit a proposed or an open project in the company home page if it is needed, however an approval is always required.

2.2.2 Student wants to do a project

In order to start a bachelor project, students need to have a group of two to four people. Students agree to be in a team outside of the system. When students access the system for the first time, they need to fill in their information such as name and student number. They can go to the projects page and see the open, active and completed projects. Joining the project is possible if the status of the project is open. To start a project students must invite a supervisor¹ to the project. In the project page the email address of the supervisor can be added, after which he/she receives an email with the information about the project and an approval link, which also registers² and logs the supervisor into the system. After supervisor's approval, students can request approval from the coordinator. The coordinator can approve the project group or delete students from a project. In case of approval students can start the project. See figure B.8.

2.3 Back-end of the system

In this section the design of our database will be explained in details using the domain model we have created using our requirements elicitation document. Furthermore, different approaches that were considered are explained and finally what was decided to carry on with. When necessary the approach is justified.

2.3.1 Database design

This subsection is dedicated to explanation about the structure of the database of BEPsys and how it was designed.

2.3.1.1 The domain model

In our requirements elicitation document (Appendix B), we have seen that BEPsys consists of four different stakeholders: **Company contact**, **Student**, **Coordinator** and

¹This is also agreed beforehand outside of the system.

²In case the supervisor is invited to the system for the first time.

Supervisor. Each student can do one **Project** at the same time, while company contact and supervisor can be attached to many projects at the same time. On the other hand, each project has and belongs to many users, thus we decided to add a model **User** to our design to be the general type of specific roles. We use inheritance to let the specific roles inherit characteristics from the parent User model. Students should be able to post Group Finder³ ads, so we needed a table **Ad** to hold information about an ad. Each student can post many ads, hence there is a one-to-many relationship between **Student** and **Ad**. In the next subsections the structure of the database is explained in details.

2.3.1.2 Entities

The database of the system exists of 8 tables of which 7 are entities and 1 is a many-to-many join table:

- **User:** a User is a general user of the system and contains the common attributes from different special users of the system. Most importantly User contains a *role* which determines what kind of special user a User is.
- **Student:** contains individual attributes of a student, such as a study number, etc.
- **Company contact:** contains individual attributes of a company contact, such as address, etc.
- **Coordinator:** contains individual attributes of a coordinator.
- **Supervisor:** contains individual attributes of a supervisor.
- **Project:** a Project contains the attributes of a project, which is proposed by Company contact, done by Student, supervised by Supervisor and coordinated by the Coordinator.
- **Ad:** contains attributes of a Group Finder message
- **User_Project:** a join table which matches users to projects.

2.3.1.3 User roles

BEPsys consists of four users, each of them having a role in the system. Based on their roles is determined how the authorization of each of the actors is done. Although the roles of each user is unique, all 4 of them are *users* who access the system. Therefore, we had to choose the best approach to create the database without being inefficient or slowing the system.

³See Appendix C for more information

2.3.1.4 First approach

The first approach we thought of was to have one table **User** for all users of the system. This would not require any joins when selecting information from the users and thus be processed faster. On the other hand there would be a lot of redundant storage in the table. This would produce a table with a huge amount *NULL* values, which eventually would decrease the performance of the system. Therefore this approach was not the approach we looked for.

2.3.1.5 Second approach

The second approach was to have no table **User** at all. The system would consist of only special tables representing different roles without a general **User** table:

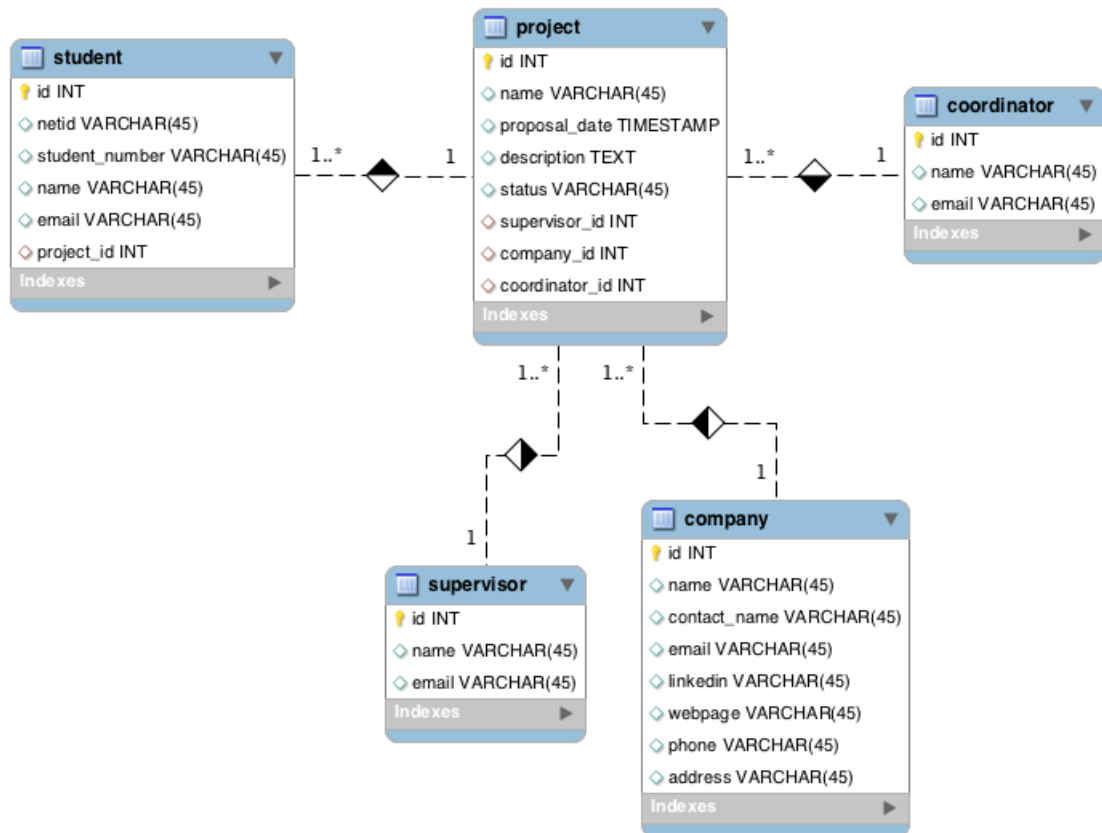


FIGURE 2.1: The approach without a User table.

With this approach there would be redundant columns in `company_contact`, `student`, `coordinator` and `supervisor`, namely *name* and *email*.

2.3.1.6 Final approach

In our final approach, we wanted to avoid having *NULL* values in columns, which would decrease the performance of the queries and secondly we wanted to avoid having the same attributes in different tables which are handled the same. We wanted the *User* table back, but to avoid *NULL* values we decided to have a general *User* table which would contain the attributes that are common between all the users and each special user's table would contain attributes that belong to each special user. Having separate tables for each kind of user would not be a performance trade-off because there would not be any expensive joins involved with these tables. Therefore we thought it was wise to normalize. Below is a EER diagram of the final approach after proper normalization of the database:

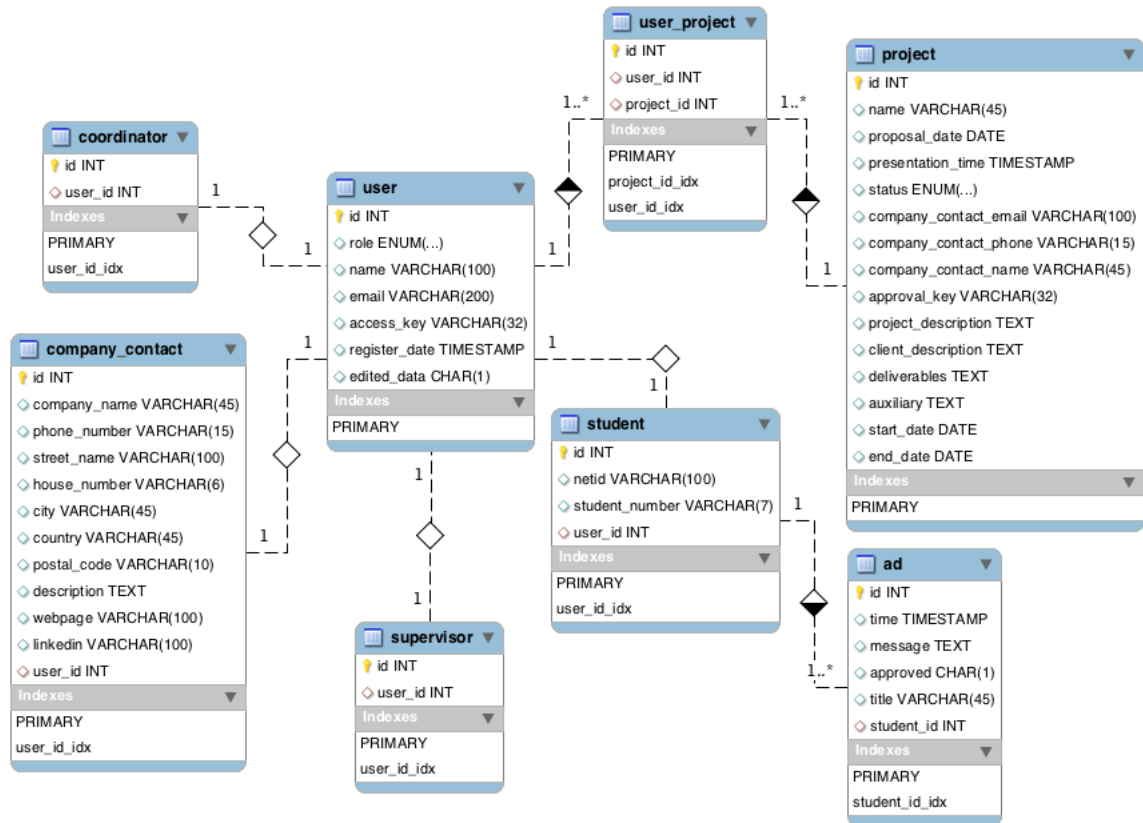


FIGURE 2.2: Bachelor Project System database EER diagram

2.4 Front-end of the system

2.4.1 Introduction

The front-end of the system enables the interaction between different users and the system. The goals of our front-end system:

- Navigate users to the different part of the system: after accessing the system, users can go to the project page, by clicking the *projects* button in the navigation bar.⁴
- Inform users about different actions that they can perform in the system and its consequences: Suppose a supervisor accepts the invitation of the students to start the project. The invite supervisor input is not going to appear for that particular project. So the students know they can't invite another supervisor to the project.
- Make the users aware of the state of the system: When a company contact submits a project proposal, he/she will be informed by an alert⁵ that the submission was done.
- Show the changes that users make in the system: for instance when a student joins a project, the join button is going to be changed to the leave button.
- Let the users know which information they can change in the system: for instance a company contact can only change his project information when the status of the project is proposed or open.
- Contains information pages that help the users to understand the rules and regulations of the system: For instance there is a page dedicated to information for the companies about how to use the system.
- Restrict access for different users: for example the front-end enables the coordinator to change the status of the projects while it is hidden from other users.

Besides fulfilling targets mentioned above the system's front-end is considered successful when it works without any bug in different user environments. To achieve this we used Twitter Bootstrap⁶. During the rest of this chapter we going deeper in the front-end of each specific part of the system.

⁴Users home page

⁵We discussed the alerts later in subsection Informing users

⁶See 1.6.2.

2.4.2 Use cases

As we mentioned before we have four different types of actor in the system which can perform different actions. As a result of that the front-end of the system looks different for each role he/she has in the system. Before designing the front-end we define the actions of each user by providing the use case diagram separately for each role.

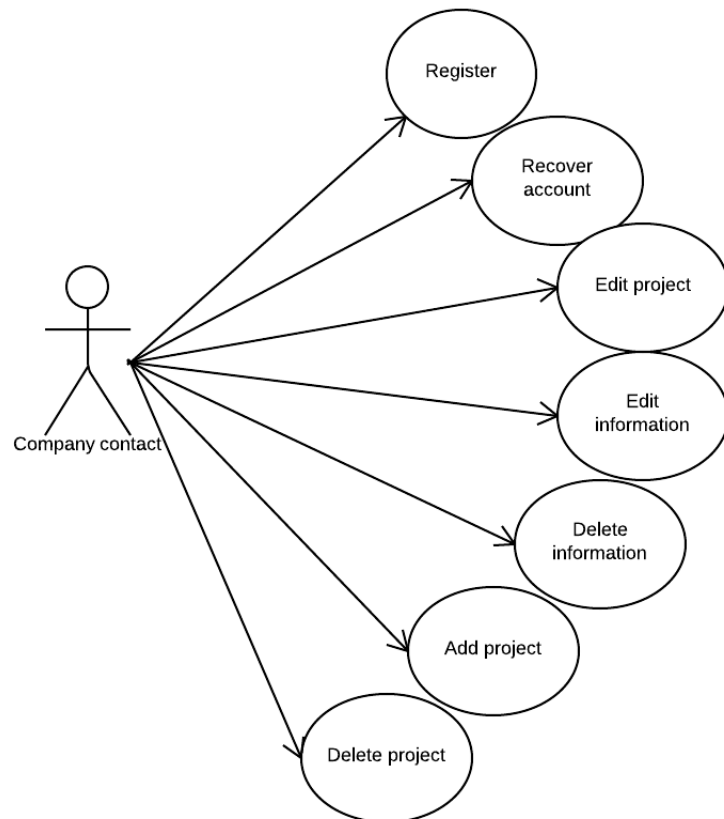


FIGURE 2.3: Use case diagram of company contact

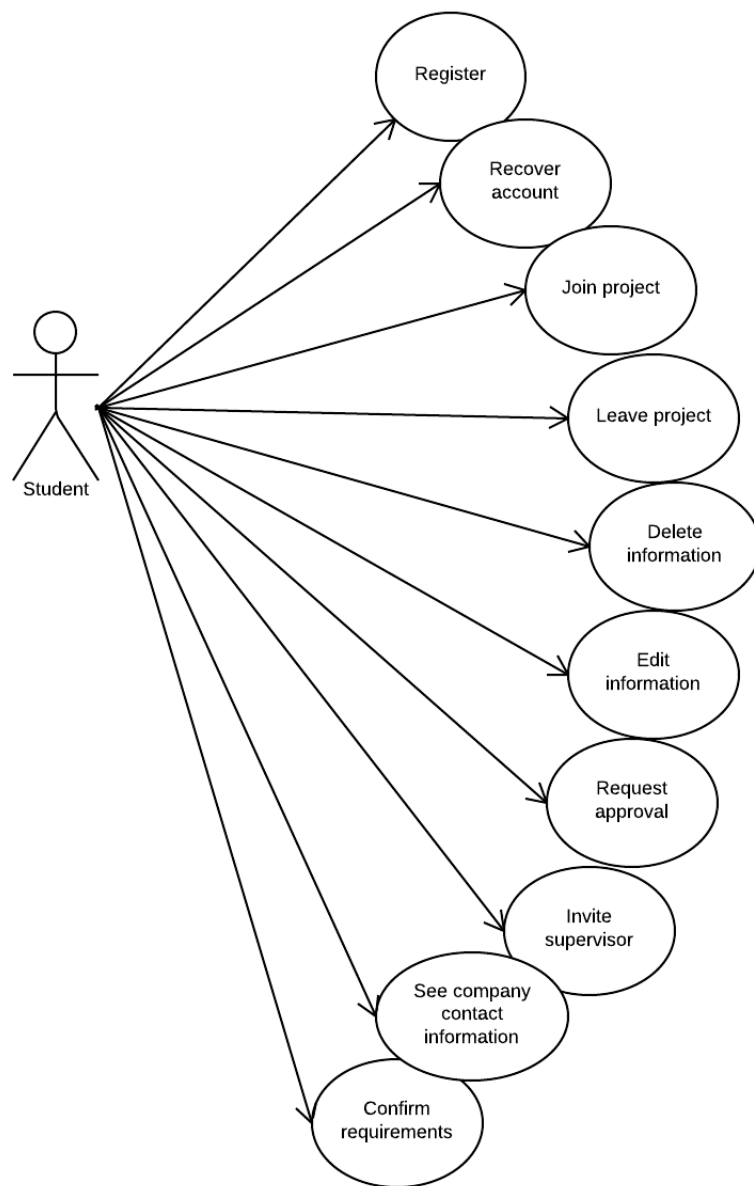


FIGURE 2.4: Use case diagram of the student

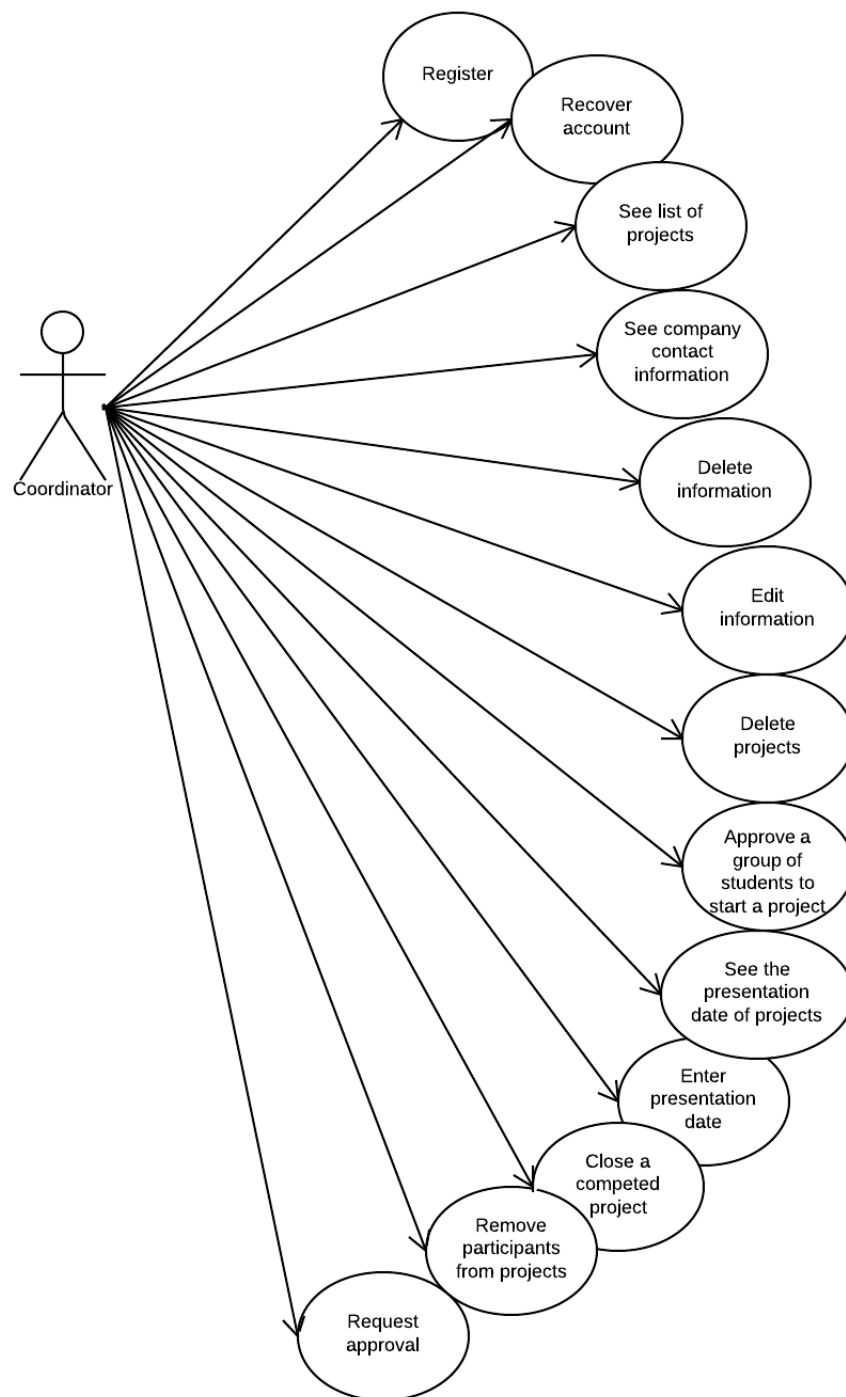


FIGURE 2.5: Use case diagram of the coordinator

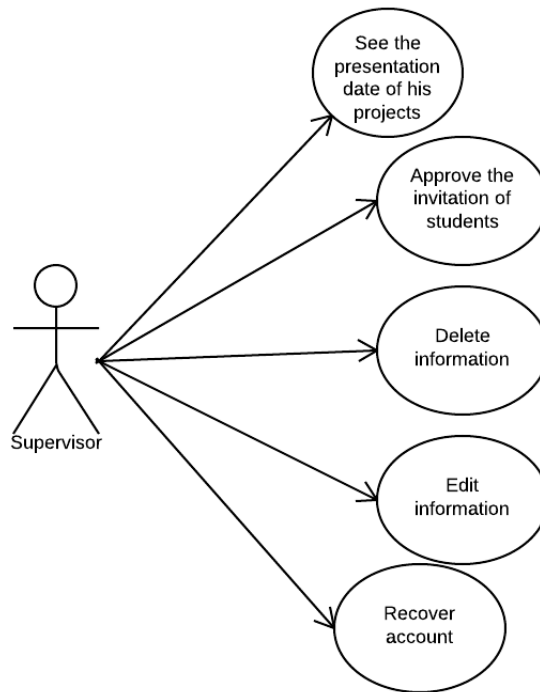


FIGURE 2.6: Use case diagram of the supervisor

2.4.3 Register

Using a drop-down button users can indicate what role they have. They can choose one of the three roles : student, company contact and coordinator. After filling the email field and clicking the submit button they will be registered in the system, after which an email is sent to the user containing a private access URL. By accessing this URL the user is logged into the system. In our database design we have one table for all users. Therefore each user needs to be unique. If an email address is already used to register, the system suggests to recover the account by showing a message. Recovering an account works in a similar way.

Please submit your email address to register

Who are you? ▾

Type here your email address ...

Student

Company

Coordinator

Get private URL

[Already have an account? Recover here.](#)

FIGURE 2.7: Registration page

2.4.4 User's home page

In our system each user has his/her own home page. We divided the home pages in four groups namely: student, company contact, coordinator and supervisor homepage. Students, supervisors and companies can see their related projects in their home pages. Via the home page users can perform all the actions possible. For example via navigation bar in home page⁷ a student is navigated to the list of projects or to the Group Finder page. Furthermore, a coordinator can navigate to the list of companies, list of students and list of supervisor's page. Companies can propose a project via their home page.

2.4.5 Projects page

The projects page accessible from the navigation bar, contains a list of open, active and completed projects. Coordinators can in addition see another list in this page, namely proposed projects. The company contact responsible for a project can still edit the

Home

Projects

Group Finder

Signed in as **Student4**

Logout

Open projects

Active projects

Completed projects

ProjectA

Project1

projectx

project1

FIGURE 2.8: List of projects in student view

information of an open or proposed project in his home page. Coordinators can edit the information of open, proposed and active projects and delete all type of projects of the system. By clicking each project all information about that project will be shown in a

⁷Note that when a user logs in to the system the navigation bar will be available in all pages

separate page. So each project has it's own page. It this page besides the information available, there is a possibility for students to join⁸ the project, and invite a supervisor to the system.

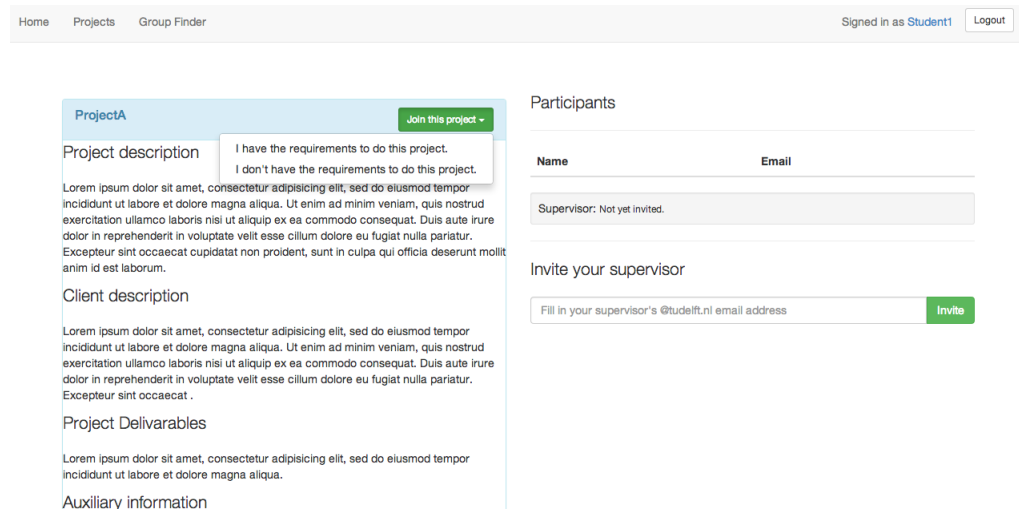


FIGURE 2.9: A project's page

2.4.6 Ads page

In this page students can find a partner or a group for the bachelor project by posting an Ad, they can also delete or edit their Ads. Coordinators can monitor this page and delete the Ads from the system. In the picture below we see the view of the Ads page for a student. He/she can view all of the Ads and delete his/hers ad via *Action* button.

⁸or leave the project if they have already joined

Time	Title	View
Monday August 26, 16:37	partner needed	view this Ad
Today, 15:31	Project for summer	view this Ad Actions ▾

Need a group or a group member?

[Place an Ad](#)

FIGURE 2.10: Ad page

2.4.7 Coordinator's special pages

Via navigation bar coordinators can access list of companies, students and supervisors. In these lists coordinators can see all information about these users. Moreover coordinators can delete a user from the system.

Home	Projects	Take me to ... ▾	Signed in as Coordinator	Logout
------	----------	------------------	--	------------------------

List of students

Name	Email	NetID	Student Number	Project	Action
Student1	Student@example.com	edjcm	134	ProjectA	Actions ▾
Student2	sara.bashiri@gmail.com	Student2NetID	2222222	No project yet	Actions ▾
Student3	student3@example.com	Student3NetID	333333	No project yet	Actions ▾

FIGURE 2.11: List of student available for coordinators

2.4.7.1 Static information pages

The users should be aware of the rules and regulations of the system. Extra information about how to propose a project or how to use the system in order to start a project is explained in a special information pages. In this pages extra information for companies, students and supervisors will be provided, in order to be able to use the system.

2.4.8 Informing users

The system's front-end is responsible for informing the user about changes that he has made or might make in the system. This can be seen as a feedback from the system to the user. For instance when a user is registered in the system an alert will appear to inform the user that the registration was successful.

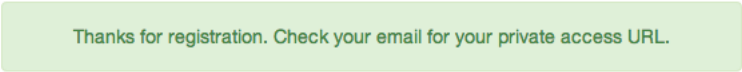


FIGURE 2.12: Registration successful alert

Another example is when a coordinator deletes a user from the system, the system asks the coordinator if he/she is sure about this action.

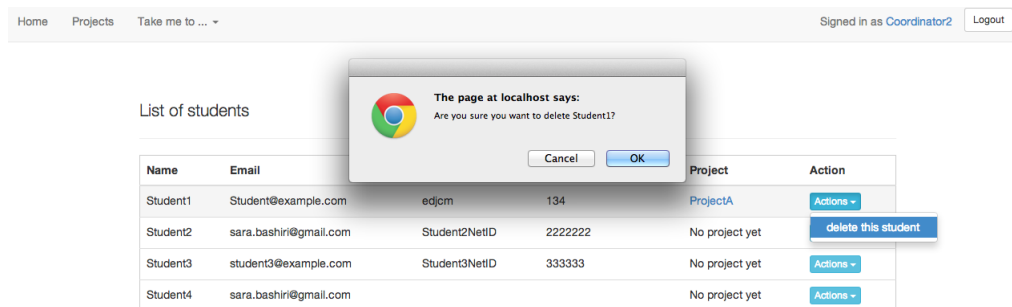


FIGURE 2.13: Delete a student from the system

Chapter 3

Implementation

3.1 Introduction

After designing the system, in this chapter we will go into more details about how we implemented BEPsys. We will divide the chapter into two parts, namely back-end and front-end.

3.2 Back-end

As explained in detail in chapter 1, we used the CakePHP web application framework which uses the MVC software architecture pattern. In this section an overview is given about the three layers of MVC, namely Model, View and Controller in our system. These three layers work together to finish a request from the user. The following figure shows a typical CakePHP request:

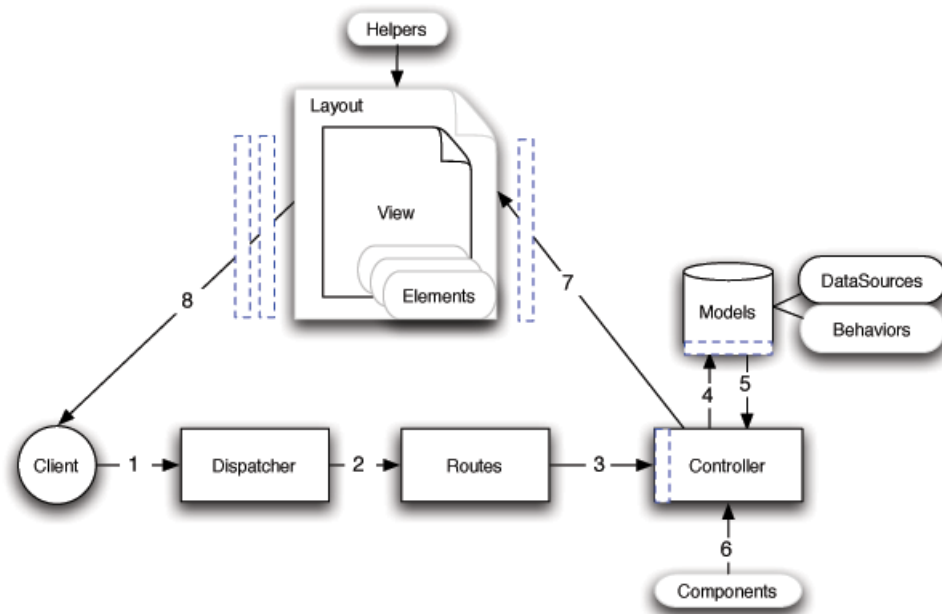


FIGURE 3.1: A typical CakePHP request

The details in the above figure will be explained in the coming sections.

3.2.1 Models

The *Model* layer is responsible for retrieving, saving, processing, validating and associating data. The model is the first layer of interaction with the database. We have made a model class for each entity in the database. There is no need to make a model class for the many-to-many table `user_project`, because data association and validation happens in user and project model separately. The following figure shows the model classes of BEPsys.

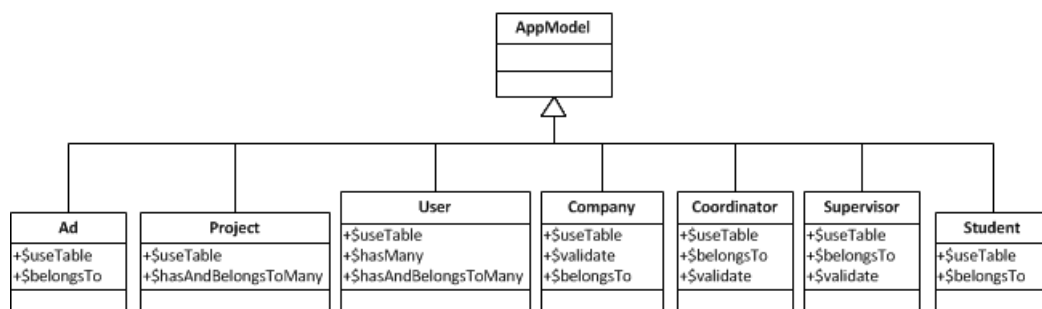


FIGURE 3.2: Models of the system

3.2.1.1 DataSources

DataSources are responsible for linking the data with the Models. For our database which is described in chapter 2, we needed a RDBMS¹. CakePHP datasources support MySQL, Postgres, Sqlite and Sqlserver by default. Because Postgres is supported by CakePHP and Heroku has an official plugin for it, we decided to chose Postgres as RDBMS .

3.2.1.2 Behaviors

Behaviors in CakePHP are part of the model layer. They allow us to organize frequently used functionality to be able to reuse them, without requiring inheritance. For instance we can let a model behave like a **Tree** structure. However, we did not need any special behavior in BEPsys and thus we did not use this part.

3.2.1.3 Data association

As we mentioned earlier, data association is also part of the model layer. In a CakePHP model, data associations are defined using an associative array which accepts several conditions as its elements. There are 4 different kinds of associations that can be used to link models:

- `hasOne`
- `hasMany`
- `belongsTo`
- `hasAndBelongsToMany`

For instance, data association in **Project** model is done as follows:

```
/**
 * hasAndBelongsToMany associations
 *
 * @var array
 */
public $hasAndBelongsToMany = array(
    'User' => array(
```

¹ RDBMS stands for Relational Database Management System

```

        'className' => 'User',
        'joinTable' => 'user_project',
        'foreignKey' => 'project_id',
        'associationForeignKey' => 'user_id',
        'unique' => 'keepExisting',
        'conditions' => '',
        'fields' => '',
        'order' => '',
        'limit' => '',
        'offset' => '',
        'finderQuery' => '',
        'deleteQuery' => '',
        'insertQuery' => ''
    )
);

```

3.2.1.4 Data validation

Data validation happens before saving data. For instance when a company contact fills in the information of a new project² and submits the form, the `save()` method of `Project` model is called after data is validated successfully. CakePHP has several validation rules built in such as *notEmpty*, *email* and *alphaNumeric*. The example below is part of the validation from `project`:

```

public $validate = array(
    'name' => array(
        'notEmpty' => array(
            'rule' => array('notEmpty'),
            'message' => 'The name field should not be empty!',
            'allowEmpty' => false,
            'required' => true,
        ),
    ),
    'project_description' => array(
        'notEmpty' => array(
            'rule' => array('notEmpty'),
            'message' => 'The description of the project should not be empty!',

```

²Propose a project

```
        'allowEmpty' => false,  
        'required' => true,  
    ),  
    ),  
);
```

In the above example we see the *notEmpty* rule which makes sure that users fill in the name and the `project_description` field. If users leave these field empty, the 'message' makes sure that users get informed properly.

3.2.2 Views

The *View* layer is responsible for presenting the user with information from the model. The view can provide a user with different types of result. The result can be a rendered *HTML* page or *JSON* encoded or even a *XML* view. For views that need to be rendered CakePHP uses its own `.ctp` extension which stands for *CakePHP Template*. These view pages (e.g. *index.ctp*) can be written in plain PHP or a mix of PHP and HTML. The view layer consists of 4 parts:

- Views: each view is unique for each controller and belongs to an action from that controller. It is the base of each response.
- Elements: these are smaller, reusable views which can be included in every view file to prevent coding something you already have done.
- Layouts: these files are containers of views. Most of the views are rendered inside layouts. For example the `head` tag of the rendered HTML pages are inside layouts and not inside views.
- Helpers: these are the available logic to use in the view files. For example `HtmlHelper` makes it easy to construct Html elements and `TimeHelper` has methods available for displayig date and time in the views.

More details about the implementation of the views can be found in the *font-end* subsection.

3.2.3 Controllers

The *Controller* layer is responsible for handling the user's request and send back a response to the user. It does that with the help of the Model and the View. The controller

is like a manager. It waits for a request from the user, sends the work to the correct part of the system, checks for authorization and authentication rules and after checking the response type, sends back the response to the user. BEPsys consists of the following controllers:

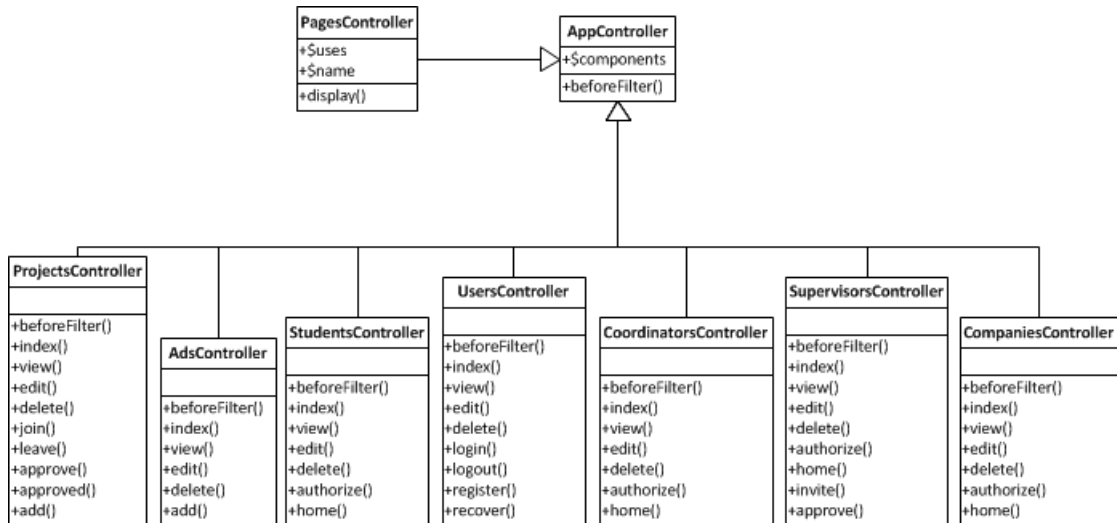


FIGURE 3.3: Controllers of the system

3.2.3.1 Components

Components are logic packages shared between all the controllers. Components are comparable with helpers in the view layer, with the only difference being that helpers provide functions for data presentation. We used *Auth* and *Session* components.

```

public $components = array(

    'Auth' => array(
        'authenticate' =>
        'loginRedirect' => array('controller' => 'users', 'action' => 'login'),
        'logoutRedirect' => array('controller' => 'pages', 'action' => 'display', 'home'

    ), 'Session'

);
  
```

Auth component

Auth component is responsible for identifying and authorizing users. As we see in the image above for authentication we added *loginRedirect* and *logoutRedirect* handlers. When for visiting a certain page a login is required, *loginRedirect* with the users controller and

login method is called. So the user needs to login to the system in order to access the page. Furthermore when users logout of the system the *logoutRedirect* is responsible to display the home page of the system.

Session component

The Session component of CakePHP is a wrapper of `$_SESSION` in PHP, which persists user data when logged into the system. In the above code we added 'Session' to components array to tell CakePHP to remember the logged in user.

3.2.3.2 PagesController

The `PagesController` is used to serve static pages. Static pages do not require a model and a controller. We used this controller for our main homepage and our information pages.

3.2.4 Dispatcher

Dispatcher is responsible to convert requests to calls of the controller methods. After it has found the requested controller, it calls the requested action in that controller. ³

3.2.5 Routes

Routes enable us to connect different URLs to controller actions. In the example below which is a part of our `routes.php`, we map `/register` to `/users/register` to make the URL shorter.

```
Router::connect('/register/*', array('controller' => 'users', 'action'=>'register'));
```

3.2.6 AJAX

In BEPsys we have used AJAX technology to perform certain actions asynchronously without refreshing pages. These actions are actions that change the page's content in real time where the user expects to see this right away instead of seeing an unnecessary page reload. An good example is a student joining a project. After successfully done so, the server send a response back to the script that will change the state of the button to *leave* and shows the name of the student in the list of participants.

³<http://api.cakephp.org/2.3/class-Dispatcher.html>

3.3 Front-end

A combination of a good front-end design and an efficient front-end implementation leads to a proper front-end in the system. In this section we are going to explain our approach for front-end implementation in more details.

3.3.1 jQuery

jQuery is a very popular open source JavaScript library which makes it easier for developers to write front-end code using much less code than pure JavaScript; that is the reason we used jQuery instead of pure JavaScript. The main advantages[5] of jQuery are:

- Cross platform compatibility: jQuery takes care of JavaScript compatibility in different web browsers. There's no need to worry whether your code works well in different browsers.
- Great event handling engine: catches and triggers any event with little code and takes care of browsers that use their own custom event system.
- Short DOM manipulation: making a page dynamic with pure JavaScript requires tens of lines of code, which takes jQuery a couple of lines to do. jQuery's `toggle()` function hides a DOM element if it's visible and shows it otherwise:

jQuery

```
$('.foo').toggle()
```

JavaScript

```
foo = document.getElementsByClassName('foo');
for (i = 0, len = foo.length; i < len; i++) {
    el = foo[i];
    if (el.style.display === 'block') {
        el.style.display = 'none';
    } else {
        el.style.display = 'block';
    }
}
```

Clearly jQuery's version of the above code contains much less code while it does the same with the DOM element.

3.3.2 Bootstrap

In orientation phase we decided that for the front-end of our system we would use Bootstrap framework. We have used several Bootstrap components to make the pages cleaner and more user-friendly.

3.3.2.1 Dropdown button

Bootstrap has a useful *Dropdown button* component which allows us to assign multiple actions to one button. This makes the users interface less complicated. In the image below you see can an example of this component.

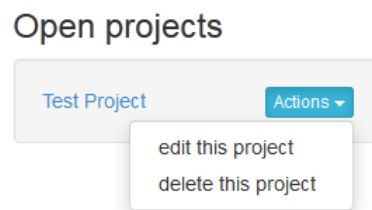


FIGURE 3.4: Bootstrap dropdown button

3.3.2.2 Tooltip

The tooltip component shows what action can be taken by an element when hovering on that element. This enables the user to see the consequences of an action before performing the action. An example of using the tooltip component:

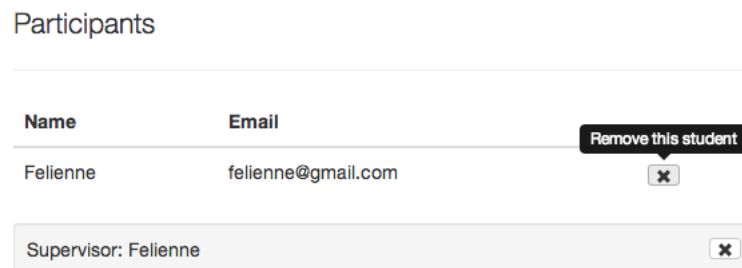


FIGURE 3.5: Bootstrap tooltip

If a mouse cursor is on the delete button, the user can see that this button will remove the corresponding student.

Chapter 4

Testing

4.1 Introduction

To test our system we decided to conduct *Usability Testing*. That is, we provide users¹ with particular tasks to perform to see whether they were able to complete those actions. However, there is also the matter of how well users think they can complete an action. That is the user experience of the system. This chapter covers the aforementioned in detail.

4.2 Test Plan

4.2.1 Goals

In order to use the results of usability testing properly, we had to set usability goals[6]. Usability goals must contain usability concepts such as effectiveness and satisfaction[7]. We also decided to measure how users experience the user interface of the system as a part of usability testing. These 3 concepts are explained below:

- Effectiveness: By effectiveness is meant how useful users find the system, that is how the system helped users perform their task and whether the system made it possible for them to complete their task in an effective² way.
- Quality of User Interface: How users experience the user interface of the system and whether or not different parts of the user interface was clear to them. This also

¹see participants subsection

²Effectiveness here is defined by test users.

covers the overall organization of the system and the ability to navigate through the system in users' opinion.

- Satisfaction: To what extent users are satisfied about the whole system, e.g. the ease of use of the system. This is about the whole system in general.

4.2.2 Participants

We let 10 people each taking a role in the system, test the usability of our system. All of our test users have different occupation and age.

4.2.3 Test Method

For usability testing we have decided to use the hall way method[8]. There is a mathematical model that claims that letting 5 people test the system is sufficient with as large as possible number of tasks to perform. However we decided to have more people test the system, because testing in different roles with the same users would give us almost the same answers for our questionnaire. We eventually let 10 people test the system in a random role.

4.2.4 Tasks

For testing our system, we ask testers to complete some tasks. We have decided to divide tasks for each role:

- Student: register, join a project, leave a project, invite supervisor
- Coordinator: register, close a completed project, delete a project, delete a project participant, edit project, approve a project
- Company: register, propose project, edit project, delete project, delete account
- Supervisor: approve a project

4.2.5 Test Environment

Although our system is compatible with mobile devices, we decided to test the functionalities of the system using the personal computer and laptop of the testers.

4.2.6 Evaluation

After the tasks are performed by our testers, we asked them to fill in a questionnaire about their experience with the system. The questionnaire consists of 3 sections with the 3 usability goals introduced earlier in this chapter. In the remainder of this chapter the questions and results are shown, after which a short conclusion is given.³

³The colors are there to differentiate between different score and do not have a special meaning.

BEP system questionnaire

[Edit this form](#)

* Required

What is your role in the system? *

- ☐ Coordinator
☐ Student
☐ Company
☐ Supervisor

User interface

The following questions are about the system's look and feel.

It was easy to navigate through the website. *

1 2 3 4 5

very difficult ☐ ☐ ☐ ☐ ☐ very easy

Texts on the website were understandable. *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so

The website looked appealing to me. *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so

Help messages were helpful during my task. *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so

It was clear for me what I had to fill in the forms *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so

The website was well organized. *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so

How satisfied are you about the user interface of the website *

1 2 3 4 5 6 7 8 9 10

not satisfied at all ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ very satisfied**Effectiveness**

The following questions are about how useful you think the website is.

I could perform my tasks with few steps. *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so**I could perform my task in little time. ***

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so**It was easy to learn to use this website. ***

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so**I was productive using this website. ***

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so**The notification system was helpful to perform my task. ***

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so**Overall satisfaction**

The following questions are about your overall experience with the website.

I am going to use the website in the future. *☐ Yes☐ No**I am going to recommend the website to colleagues ***☐ Yes

☐ No

The website was easy to use. *

1 2 3 4 5

not at all ☐ ☐ ☐ ☐ ☐ very much so

How satisfied are you about the whole website? *

1 2 3 4 5 6 7 8 9 10

not satisfied at all ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ very satisfied

Do you have any comments?

Never submit passwords through Google Forms.

100%: You made it.

Powered by


This content is neither created nor endorsed by Google.

[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)

4.2.6.1 User interface

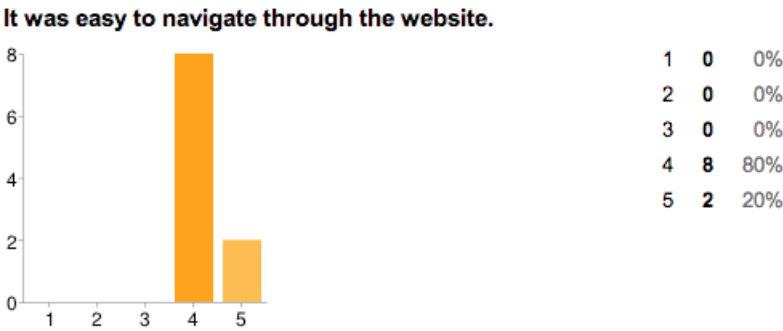


FIGURE 4.1: Navigation

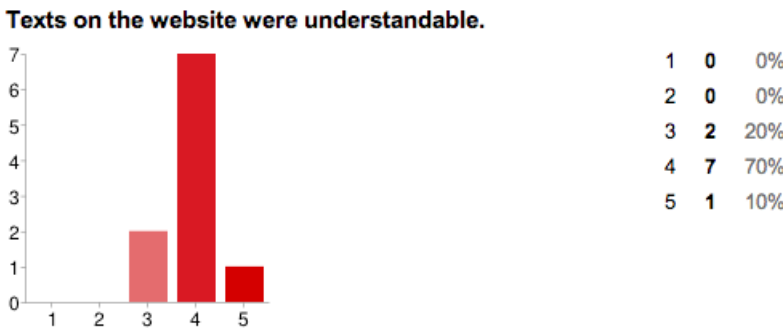


FIGURE 4.2: Understandability

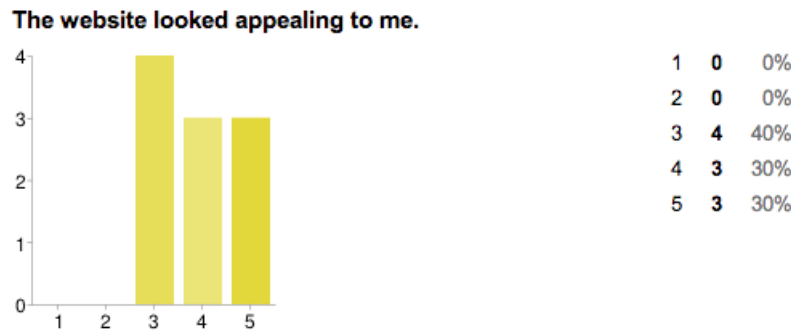


FIGURE 4.3: Website’s look

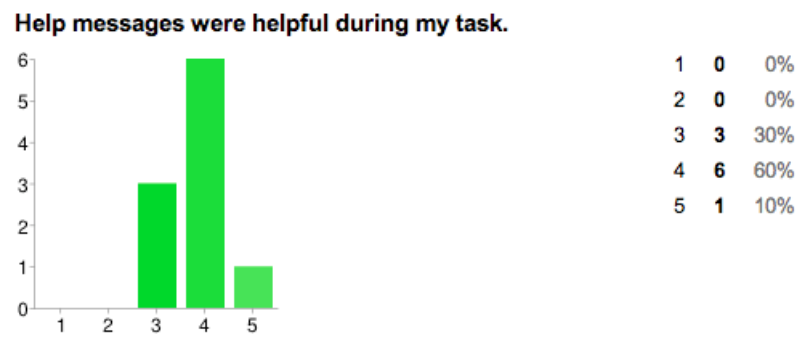


FIGURE 4.4: Help messages

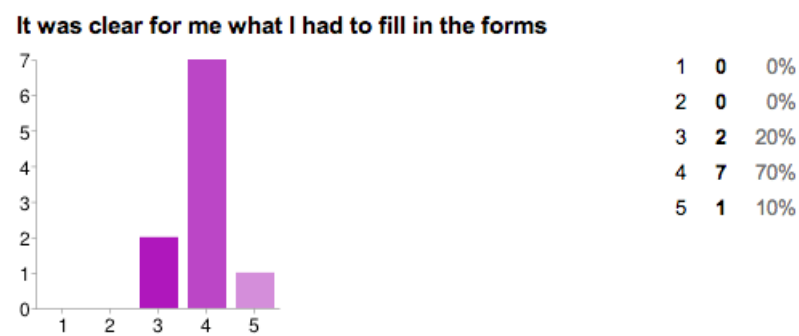


FIGURE 4.5: Clarity of the forms

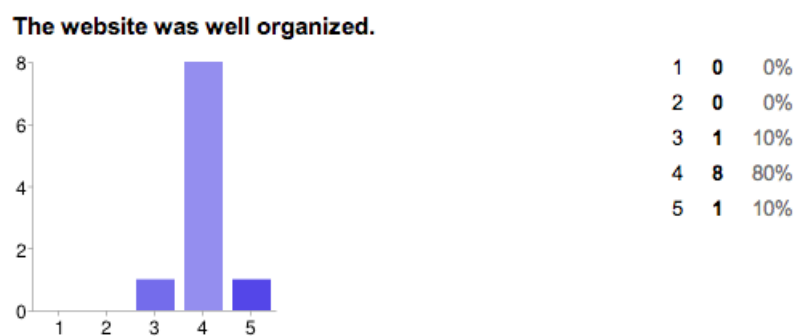


FIGURE 4.6: Website's organization

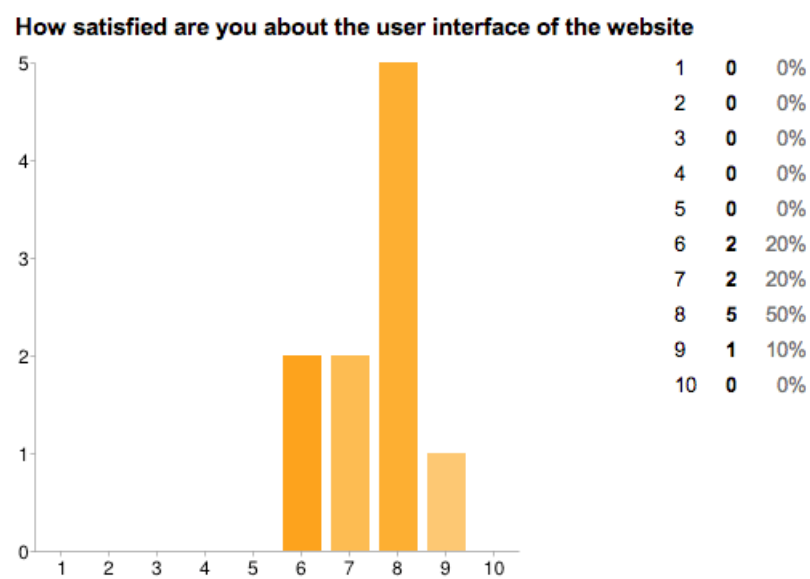


FIGURE 4.7: UI satisfaction

4.2.6.2 Effectiveness

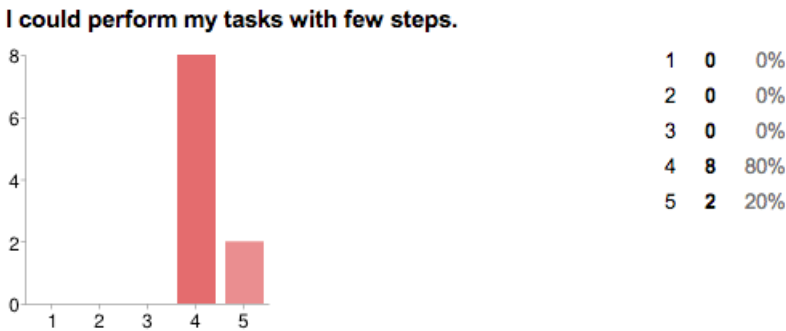


FIGURE 4.8: Task performance steps

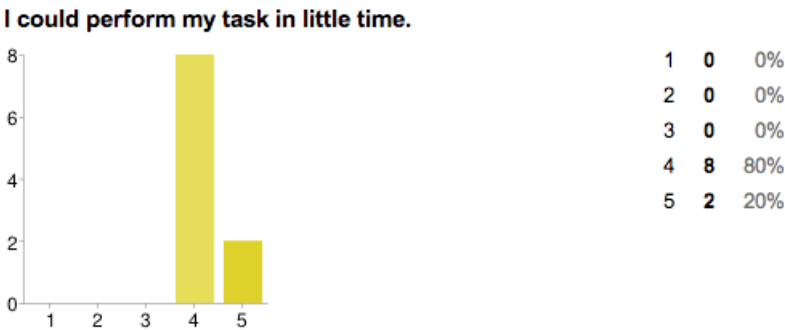


FIGURE 4.9: Task performance time

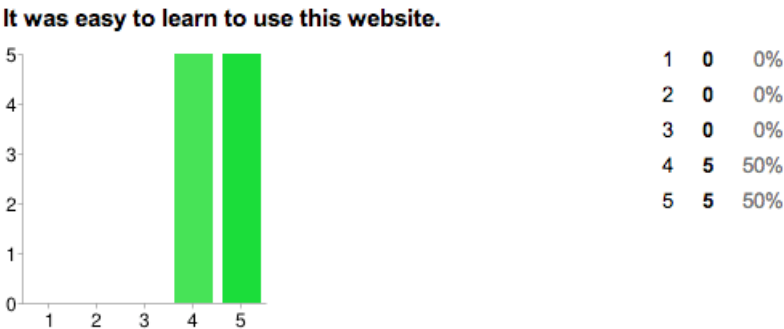


FIGURE 4.10: Website’s learning curve

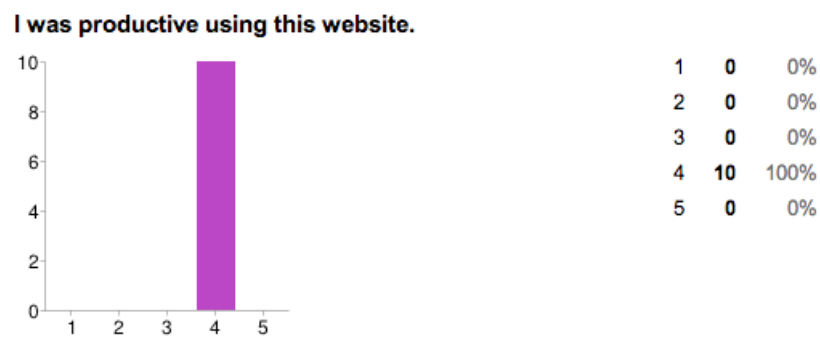


FIGURE 4.11: Productivity

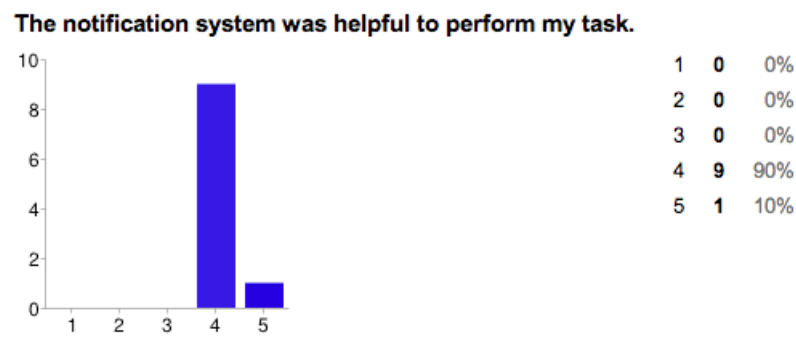


FIGURE 4.12: Helpful to perform the task

4.2.6.3 Overall satisfaction

I am going to use the website in the future.

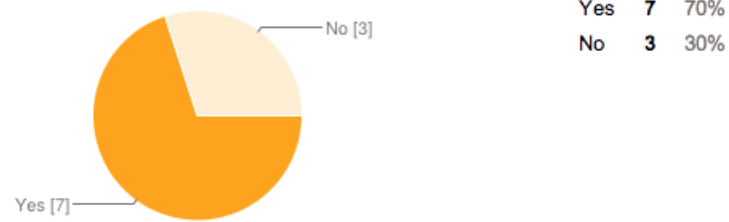


FIGURE 4.13: Use in future

I am going to recommend the website to colleagues

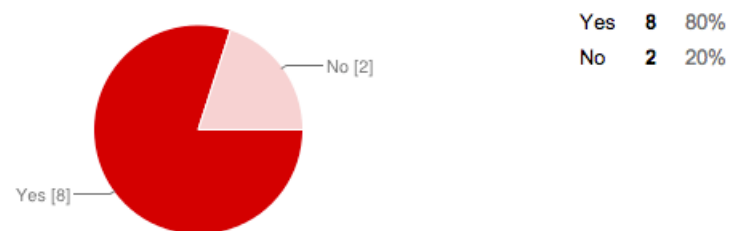


FIGURE 4.14: Recommend to colleagues

The website was easy to use.



FIGURE 4.15: Ease of use

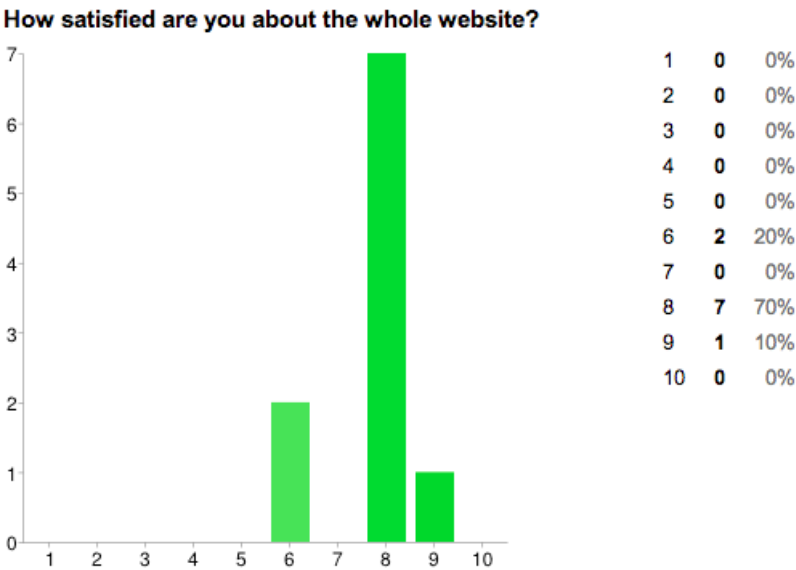


FIGURE 4.16: Satisfaction about the whole website

4.2.6.4 Conclusion

From the results we can see that the testers found that the system made them more productive and that using the website was easy for them, hence it took them not much time to perform their tasks. They also thought the system is well organized. As expected they found the system not very appealing, as the user interface has yet to be improved in the future. One of the testers suggested usernames and passwords to use for login, something we have already thought of and will build in the near future. The same goes for search functionality for the system.

Chapter 5

Conclusion

The goal of this project was to develop a system that should be an improvement to the current system. The future system would make proposing, finding and joining projects faster and more organized. The coordinator would no longer have to make each small change in the system himself/herself. Instead this should easily be done by the concerning user. In addition the system would help easing the process of approval which is done several times in different work flows. During this project we successfully implemented the *must have* requirements. The delivered product is a working system¹ which was mentioned in the requirements elicitation document. Before the system is completely ready for production, some improvements had to be done such as authorization. After testing we received positive feedback about the system. The user interface needs some improvements in order to make the system more appealing to the users. In addition we think some functionalities need to be added to the system before production:

- Login with user name and password
- All approval possible within the system: coordinator should be able to approve actions in the system and not in an email.

And some suggestions for later versions are:

- Notification within the system: in-system notifications as an alternative to email notifications.
- Document upload system.
- Messaging system which enables users to interact within the system.

¹Development version

Appendix A

Project Plan

A.1 Introduction

In this section a planning of the project is given. It will become clear what the background of the project is, what the project exactly contains, what are the requirements and restrictions, what approach we will to use and finally a schedule of important events during the project.

A.1.1 Background

At the moment the communication about potential bachelor project between the interested companies and TU Delft happens through email. A company sends an email to someone from TU Delft (not always the coordinator), in many cases this person (if not the coordinator) notifies the Bachelor's final project coordinator. The coordinator will send a project proposal template to the company to fill in (if the company does not already have it). The company will send the proposal document back to the coordinator. When the project meets the rules, it will be posted on the related BlackBoard page. The aforementioned procedure could be improved a lot if there is more automation in parts of it. If the project is carried out successfully, the process should be more organized and less waste time.

A.2 Project description

A.2.1 Introduction

This section is about the project specifically. We will describe who is the client, who are the contacts, what are the goals of the project and what exactly has to be delivered.

A.2.2 The client

The client is Dr.Ir. Felienne Hermans from TU Delft.

A.2.3 Contacts

Dr. Martha Larson is the coordinator of the project, the project is supervised by Ir. Hans Geers.

A.2.4 Problem description

As mentioned in the introduction section the current system requires a lot of unnecessary wasting of time and sending emails. Companies are not able to update their project proposal without the help of the coordinator.

A.2.5 Goal

Develop an organized web system that should improve the communication between the companies, the university and the students by easing the process of proposing a project, registering for a project and approving the project proposal by the coordinator.

A.2.6 Product

The product is a web application which will be available on a public domain.

A.2.7 Preconditions

There are no special preconditions defined. Although, the system has to be maintainable and has to be finished very rapidly because of the lack of time. The focus is more on having a basic working system.

A.3 The approach

A.3.1 Introduction

This chapter describes methods, techniques and approaches that will be used in the project.

A.3.2 Software Engineering Methodology

This project will to be carried out using the Agile framework and specifically the Scrum method. In each *sprint* there will be worked on a new part of the system and at the end the new functionality will be added to the whole system. We will have two kinds of meetings. The *daily meeting* during which we will shortly discuss what we have done and what we are doing right now. If there are any problems we will try to solve them. The other meeting is a *sprint meeting* at the end of a new sprint or milestone of the project. If there is any problem with the functionality another spring will start, only shorter in time. We have chosen this method because of the fact that the project has to be finished very rapidly.

A.3.3 Technical details

We will be using a web application framework which is most suitable to our situation. We will investigate this in the *orientation report*.

A.3.4 Proceedings and schedule

Week/Activity	1	2	3	4	5	6	7	8
Orientation								
Requirements Analysis								
Front-end design								
Technical Design								
Implementation								
Testing								
Final Report								
Presentation								

A.4 Quality Assurance

A.4.1 Introduction

In this chapter we discuss the different methods that we will use to guarantee the quality of the entire project.

A.4.2 Quality

The quality of this system is assured by working according to the software methodology mentioned in the chapter *The approach*. We will also organize weekly meetings with the client and test during the implementation phase to get the right feedback on time.

A.4.2.1 Evaluation

The written code will be evaluated by *SIG*, after which we are able to improve the quality of the implementation according to the feedback.

A.4.2.2 Versioning

We are using *Git* as a version control technology to ensure the quality and maintainability of the code we are writing.

A.4.2.3 Pilots

Because of the time restriction and to avoid surprises after the product delivery we need to get the proper feedback from the client. We decided to let the client test the system as much as possible during the implementation phase. When new functionality is added to the system, we let the client test and give feedback. During the final user-testing we let all the actors test the product. At last we could tweak the system a bit using the relevant feedback.

Appendix B

Requirements Elicitation

B.1 Introduction

This appendix contains functional and nonfunctional requirements of the system, including user stories and scenarios.

B.2 Current system

At the moment registering for a bachelor project is done by filling in a form (on paper) by student teams, that has to be signed by both the company and the university. Furthermore, a company interested to work with the university needs to contact the coordinator first, who then will upload the project proposal. This will happens many times until the proposal is approved to appear in the list of open projects. This method is inflexible, because the proposal cannot be updated/edited by the company and the coordinator needs to be contacted again in order to update the proposal.

B.3 Proposed system

B.3.1 overview

The bachelor project monitoring system is a web application that lets students, coordinators/supervisors and companies communicate in a well organized and reliable web environment. It enables all involved parties to access contact, project and company information in a fast way. The new system should make the change to the project proposal flow more flexible as it lets the companies edit their projects. On the other hand it make

it easier for coordinators as proposing and joining the projects happens in one central system, where input from other users can be approved using very little time.

B.3.2 Stakeholder analysis

Four different stakeholders are distinguished:

Company contacts: They are people within a company who have project available for the students of the university.

Coordinators: Coordinators are responsible for approval of the proposed projects. Furthermore they are able to see the information of all users of the system and also to delete users from the system.

Supervisors: Supervisors get invited by a project group, they will assist the group throughout the project.

Students: The students executing a project or looking for a project or a partner.

B.4 Functional requirements

In Agile software development methodology user stories are used as an alternative for a requirements list. In order to get a more clear overview we decided to use user stories for each actor. To prioritize the requirements we will use the MoSCoW method.

B.4.1 Company contacts

Must have:

- Register in order to access the system.
- Fill in a project proposal form in order to propose a project.
- Edit or delete the content of the project proposal.
- Edit or delete registration information.
- Recover account

Could have:

- See the list of the projects that I have proposed so far, regardless of their status.
- Get notified when a group is approved to work on the project.

- Receive a confirmation via email that the submission is done.
- Receive a confirmation via email that the project is approved.

B.4.2 Coordinators

Must have:

- Register in order to access the system.
- See the list of open/active/completed projects.
- See the company contact¹ information of projects.
- Be able to remove projects.
- Approve a group of students in order to begin a project.
- See the presentation date and time of each project.
- Enter a presentation date and name as a presiding coordinator for each project.
- Close a completed project.
- Remove a participants from a project.
- Edit or delete registration information.
- Recover account.

Could have:

- Find the contact information of companies and students.
- See a a list of supervisors and their projects.
- Get notified by email that a project has been approved.
- Receive a confirmation email containing email addresses of the involved students when a project is closed.
- Get notified by email when a project is edited.
- Get notified by email when a students posts an ad in the Group Finder. This email contains a link to that ad.

¹By company contact is meant that each company can be represented by different contacts. Each company page belongs to a company contact.

- Receive a CC of the confirmation email that a project was submitted by a company.
- Receive a CC of the confirmation email that an ad was submitted.
- See the list of students participating in a project.

Won't do:

- Use the system to send a feedback to the company if the project does not meet the requirements.
- Approve or reject a proposed presentation date by students.

B.4.3 Supervisors**Must have:**

- See the presentation date and time of my projects.
- Approve² the invitation of students.
- Edit or delete registration information.
- Recover account.

Could have:

- Receive a confirmation email that supervisor has just joined a project in the system.
- See the list of all open/active/completed projects in one page.
- See a list of projects that supervisor have supervised so far.
- See the company contact information of projects.
- Get notified by email that project was approved.

Won't do:

- Approve or reject the proposed presentation date by students.

²The supervisor can only approve the invitation, because the students and the supervisor agree to work on the project prior to this invitation. This invitation also registers the supervisor on the system.

B.4.4 Students

Must have:

- Register in order to access the system.
- See the list of all open/active/completed projects.
- See the company contact information of projects.
- Confirm³ that student has all the requirements to begin the project.
- Join an open project.
- Leave a project.
- Request approval to start a project.
- Invite a supervisor to the system.
- Edit or delete my registration information.
- Recover my account

Could have:

- Post an ad in the Group Finder.
- Get notified by email when my group has been approved.
- Delete my ads when student has a project.
- Get notified by email that ad was submitted.
- Get notified by email that ad was approved.
- Get notified by email that ad was deleted and that student has to post another one if student wants.

Won't do:

- Propose a presentation date in the system.

³The student is asked to check a check box. It is the students' responsibility to check if they have all the requirements. The system has no way of checking this.

B.5 Nonfunctional requirements

Besides the main functionality and behavior of the system, in other words: what the system is supposed to do, it is also important to specify how the quality of the system is supposed to be.

B.5.1 Usability

The system should have a simple and understandable interface for the Bachelor's final project coordinator. Furthermore, the system should let coordinators approve student groups, projects, Group Finder ads.

The system should let companies submit their project proposal without spending too much time. Right from their home page companies should be able to edit or delete their info and their project proposals.

B.5.2 Security

The system requires all the input from users first to be approved by the coordinator. Relevant input should be validated and cleaned before saving. Passwords should be encrypted before saving.

B.6 Scenarios

B.6.1 Company contacts

- **Register:** when a user goes to the website and wants to register, when user fills in an email address and submits the form, they receive an email with a private access URL to access the system.
- **Proposes project:** when a company contact goes to *add a project* page, the company contact can fill in the project proposal form and submit it.
- **Edit/delete project:** when company contacts go to the edit page of their project, they see a project proposal form that they have filled in before. This form is editable, so they can update the information. If they submit the data, it will be also updated in the system. Company contacts can also delete their project by *delete* button.

- **Edit/delete registration information:** when users click on their name in navigation bar they access their *home* page. In this page they can edit or delete their registration information using edit or delete button in action drop down.
- **Recover account:** via *recover account* page, users can fill in their email address in to receive their private access URL by email.

B.6.2 Coordinators

- **Register:** see *Company contacts register*
- **See list of projects:** coordinators are able to access the *projects* page via navigation bar. In this page they see 4 lists of project, namely: open, active, completed and proposed. All of the projects in the system belongs to one of these lists.
- **See company contact information** coordinators can access the company contacts page by *company contacts* button in navigation bar. In this page they can see all the company contacts with their information.
- **Remove project:** in *projects* page, coordinators are able to remove each project using *delete* button.
- **Approve a group of students in order to begin a project:** when a *request approval email* ⁴ is sent to the coordinators. They can access the project by a link which is provided in email. They can either approve the request by changing the status of the project from open to active, using status drop down; or delete the project (See B.6.2 subsection, remove project part) or remove the participants from the project (See B.6.2 subsection, remove a participant from the project).
- **See the presentation date:** see *See list of projects*. In this page the coordinators can see the presentation date and time of each project in the active project list.
- **Enter presentation date:** when coordinators are in project's page, they are able to submit a date and time for the presentation of the project along with the name of the presiding coordinator.
- **Close a completed project:** when coordinators are in *projects* page, they are able to close a finished project by changing the status of the project from active to completed.

⁴This email is sent by student who wants to start the project. It contains a link to the project page. For more information about request approval see subsection C.6.4, Request approval part

- **Remove a participant from a project:** in project's page, coordinators is able to remove a student or a supervisor from the project using remove button which is located next to each participant.
- **Edit/delete registration information:** see *Edit/delete registration information* in Company contacts.
- **Recover account:** see *Recover account* in Company contacts.

B.6.3 Supervisors

- **Sees presentation date and time:** in supervisor's home page, the supervisors can see their projects. Next to each project the corresponding presentation date and time is also available.
- **Approves invitation/joins group:** see *Student invites supervisor*. An email should be sent to the supervisors containing an approval link, which also registers the supervisors on the system.
- **Edit/delete registration information:** see *Edit/delete registration information* in Company contacts.
- **Recover account:** see *Recover account* in Company contacts.

B.6.4 Students

- **Register:** see *Company contact register*.
- **See project list:** students are able to access the *projects page* via navigation bar. In this page they see 3 lists of open, active, completed projects.
- **See company contact information:** in projects page students can select one project to see more information about it⁵. By clicking the company name the students can see more information about the company and the company contact who offers the project.
- **Confirm requirements:** in project's page when students want to join a project, there is a check box where they can confirm that they have the requirements to start the project.

⁵We call this page *project's page* which contains the project proposal and other information such as company name.

- **Join an open project:** in project's page when the status of the project is open, students can join the project using join button. Then their name will be added to the list of participants in this page.
- **Leave a project:** see above paragraph *Join an open project*. Students can leave the project using *leave* button in this page. Then their name will be removed from the participants of the project.
- **Requests approval:** when a group of students already have a supervisor and are ready to start⁶, they are able to request approval from the coordinator, by *request approval* button in project's page.
- **Invite a supervisor:** in project's page when students fill the email of their supervisor and submit it, the supervisor will be invited to the project by email⁷.
- **Edit/delete registration information:** see *Edit/delete registration information* in Company contacts.
- **Recover account:** see *Recover account* in Company contacts.

⁶The project should consists of at least two students

⁷This email consists of project's information and a link which enables the supervisor to join the project

B.7 Flow charts

In this section workflows of the system are displayed. This is done to divide all user stories in more understandable processes and thus making user stories more clear for the reader. Note that actions in grey font are actions which don't belong to must have requirements. Below is the definition of the icons which appear in the workflows.

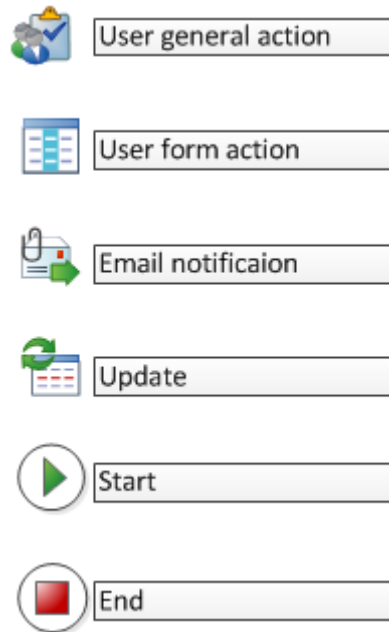


FIGURE B.1: Definition of icons

B.7.1 User register

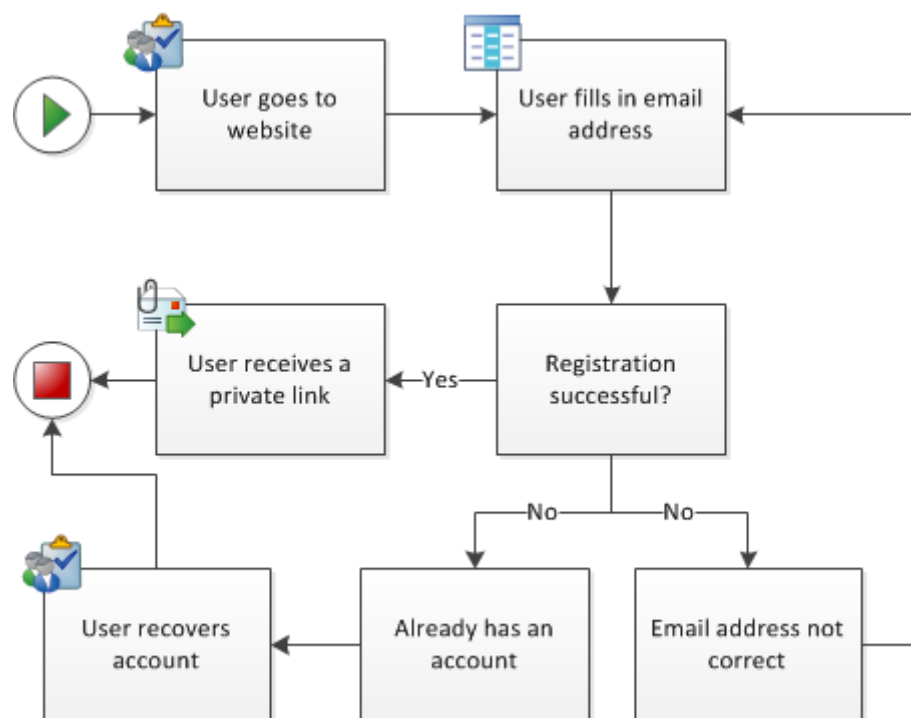


FIGURE B.2: Registration (except supervisor)

B.7.2 Company

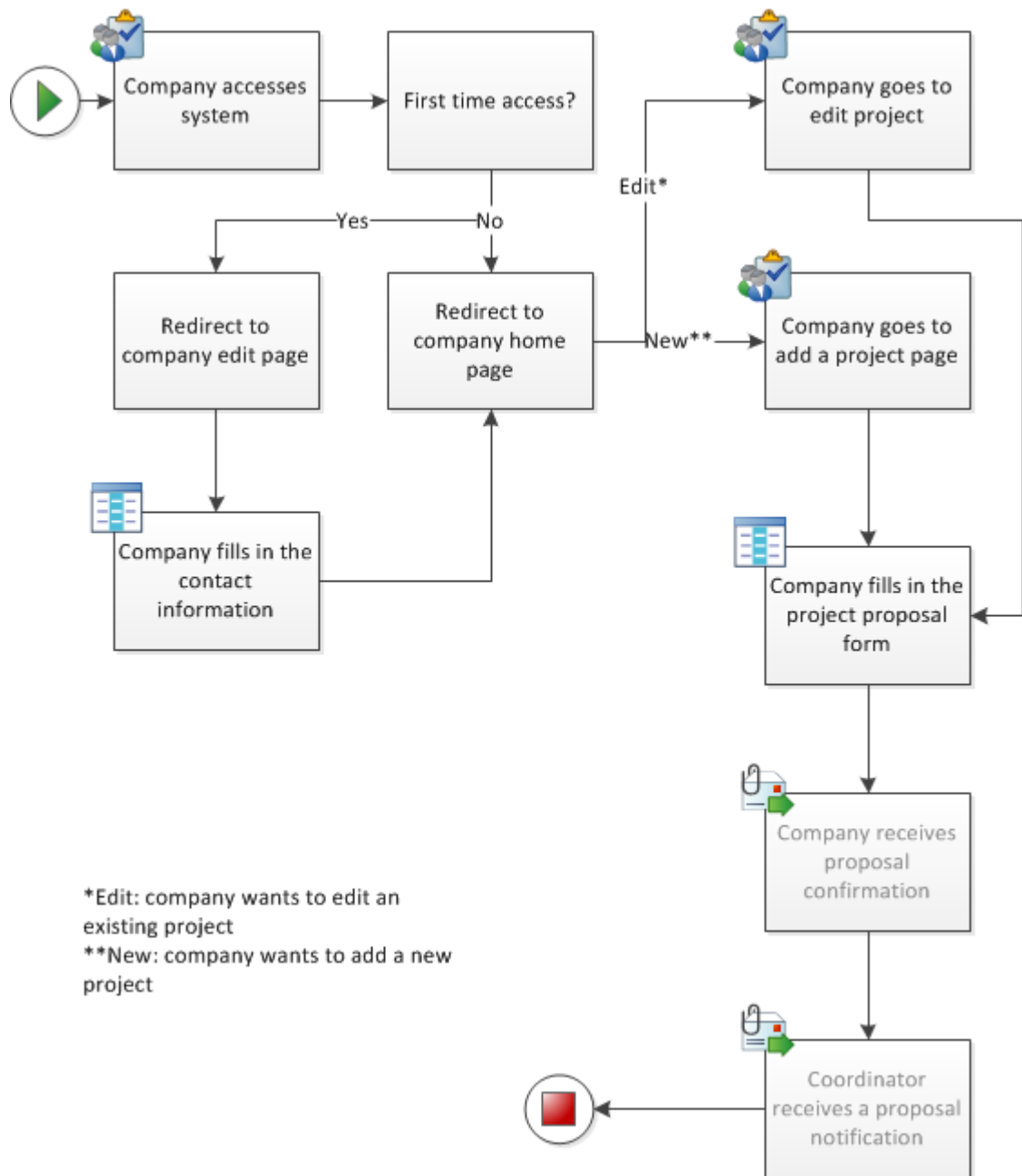


FIGURE B.3: Proposes/edits project

B.7.3 Coordinator

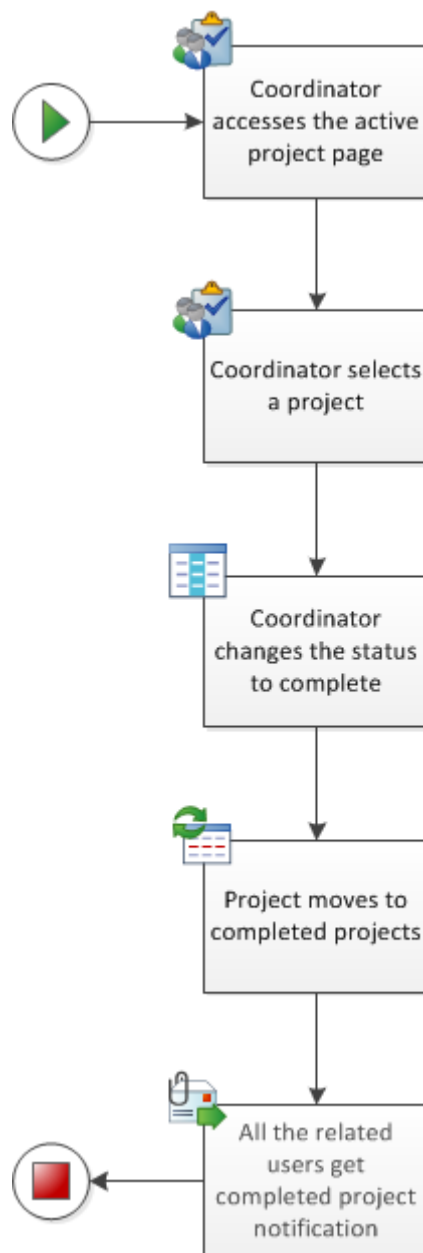


FIGURE B.4: Closes a completed project

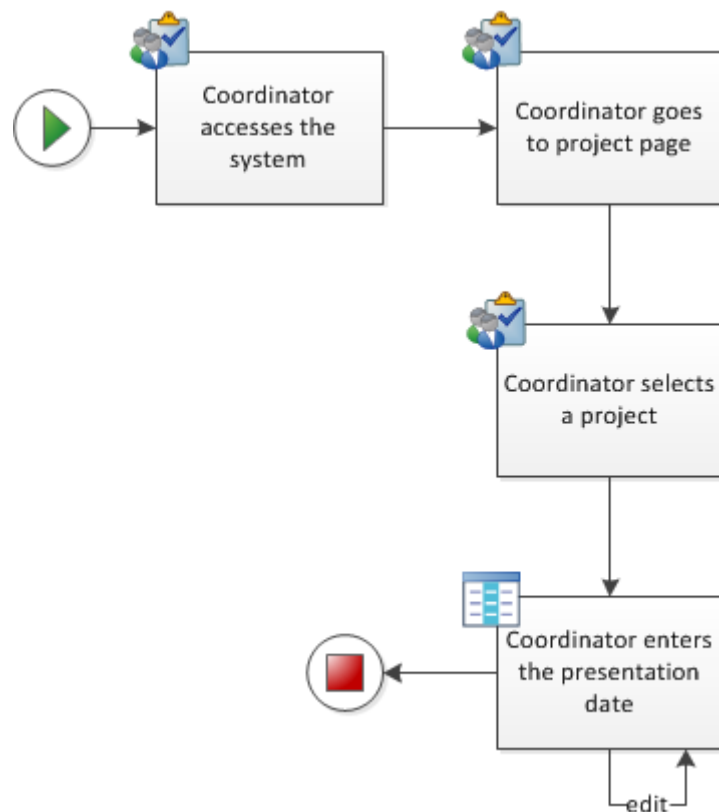


FIGURE B.5: Fills in presentation date

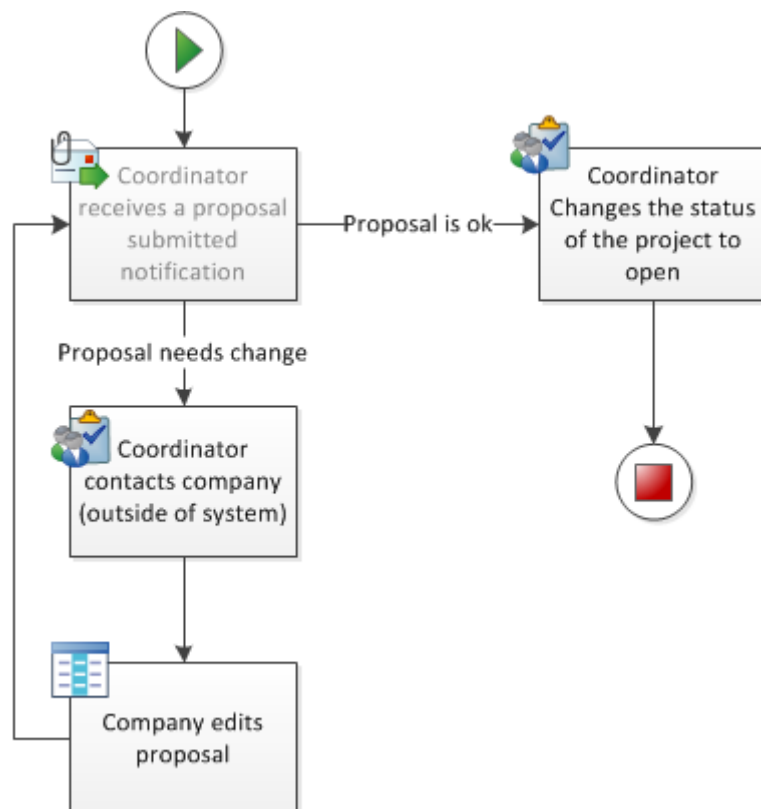


FIGURE B.6: Project approval

B.7.4 Student

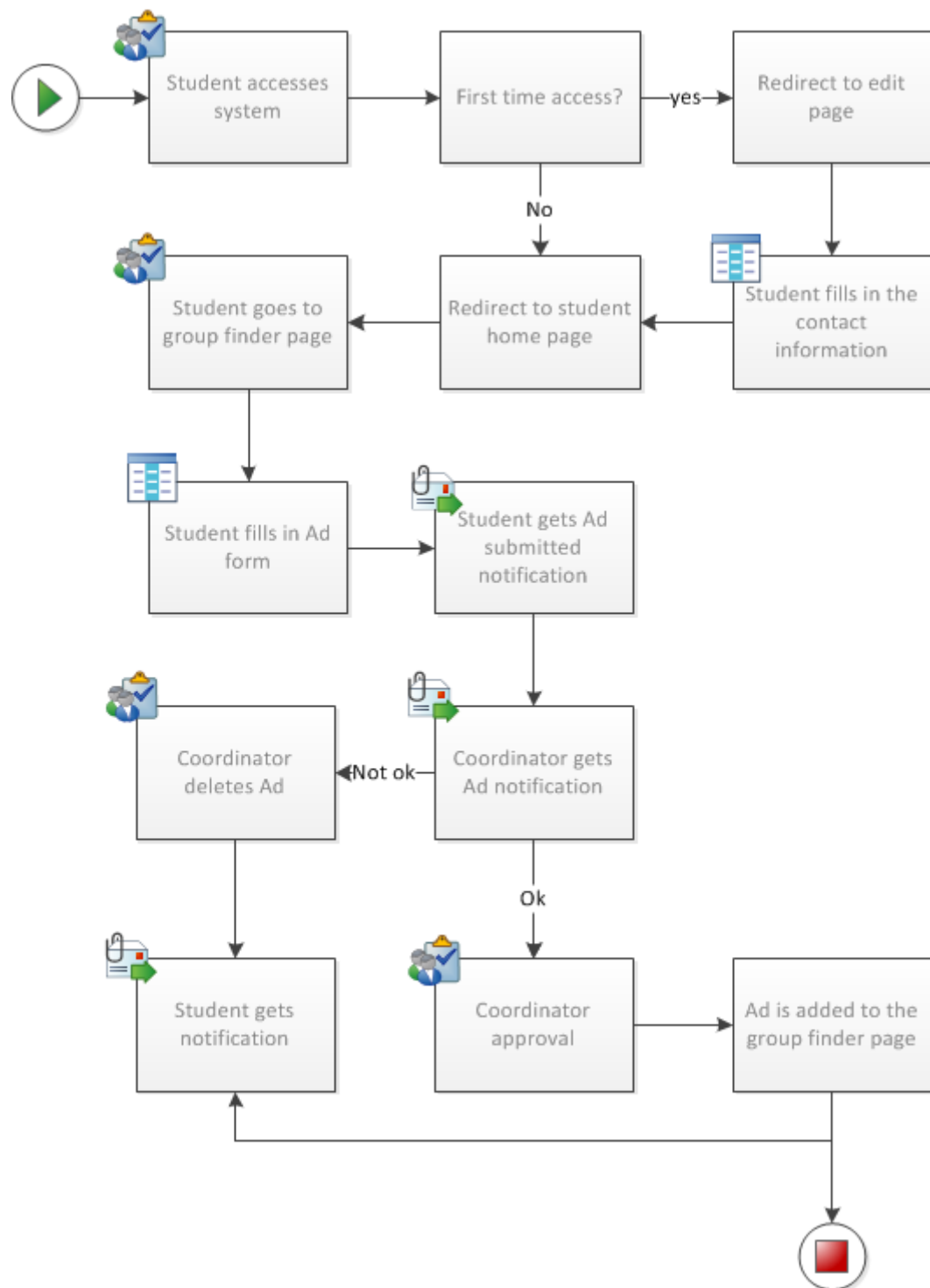


FIGURE B.7: Post Ad

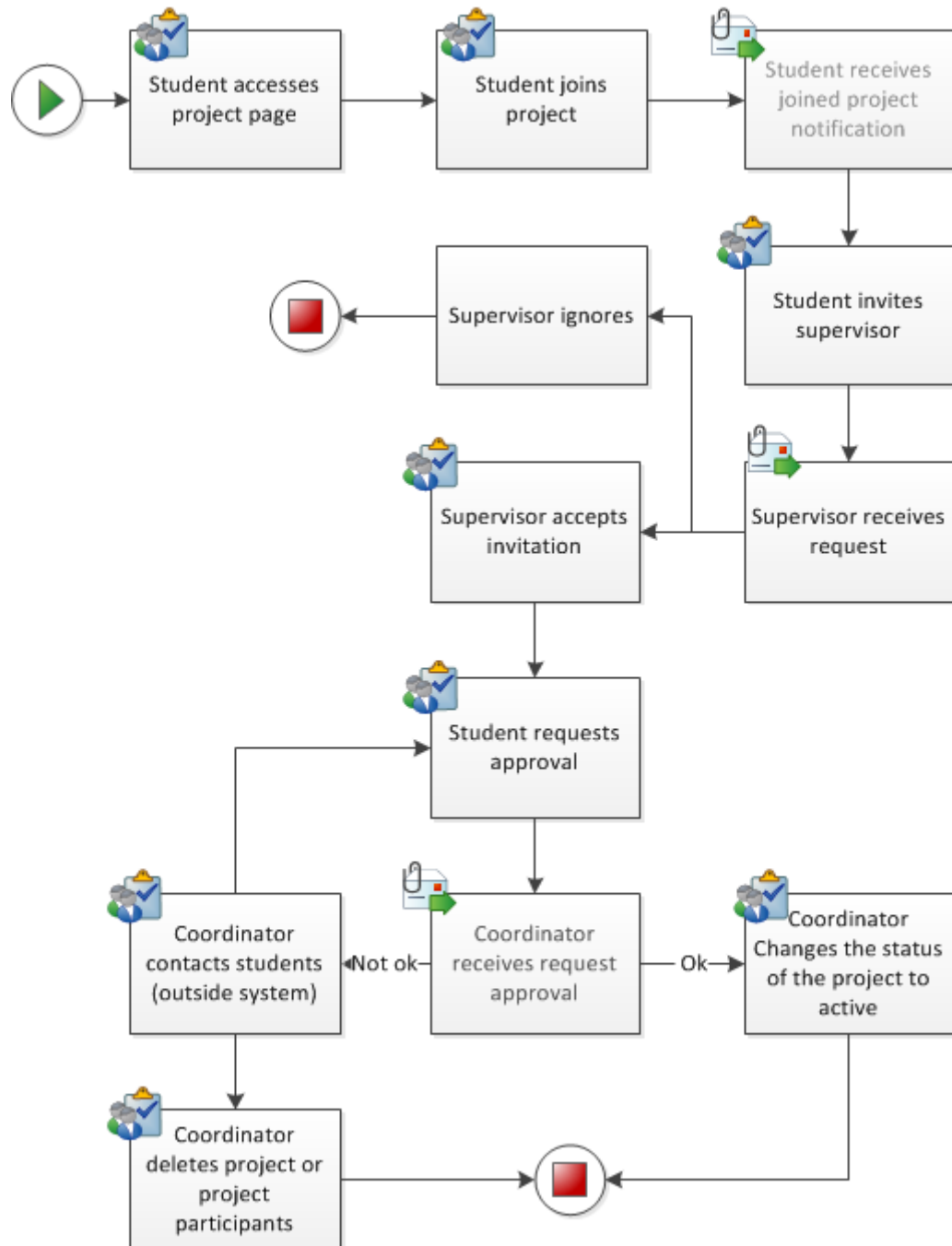


FIGURE B.8: Join/leave project, Invite supervisor, coordinator approval

Bibliography

- [1] Zurmo - open source crm, .
- [2] Splendidcrm, .
- [3] Tom DeMarco and Tim Lister. *Peopleware: Productive Projects and Teams*. 2013.
- [4] Mark Otto. Bootstrap from twitter. <https://dev.twitter.com/blog/bootstrap-twitter>, 2011.
- [5] What are some of the pros and cons of using jquery?, .
- [6] Jeffery Rubin and Dana Chisnell. *Handbook of usability testing*. Academic press, 2008.
- [7] Nielsen. *Usability Engineering*. Academic press, 1994.
- [8] Jacob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems, 1993.