Project Proposal

Noël Keijzer Jan-Jaap Korpershoek Tom Leemreize Joost Prins

March 14, 2018

1 Project Requestor

A. Fehnker
University of Twente, Faculty EEMCS
Building Zilverling, room 3120
PO Box 217
7500 AE Enschede
The Netherlands

2 Beneficiaries/Stakeholders

The primary stakeholder is:

A. Fehnker
University of Twente, Faculty EEMCS
Building Zilverling, room 3120
PO Box 217
7500 AE Enschede
The Netherlands

Beneficiaries of this project are:

M. Huisman
University of Twente, Faculty EEMCS
Building Zilverling, room 3039
P.O. Box 217
7500 AE Enschede
The Netherlands

L. Ferreira Pires
University of Twente, Faculty EEMCS
Building Zilverling, room 4098
PO Box 217
7500 AE Enschede
the Netherlands

3 Statement of Problem or Need

In the first module of the Bachelor programme Creative Technology, the students are introduced to programming. Currently the students' only way to ask for help is during the assigned hours, or outside of the assigned hours on Slack or blackboard, from either the student assistants or the professor. The main issue with Slack or blackboard is that there is no real way to discuss your code aside from posting a screenshot or text copy of the code with another seperate message describing the issue or question. This makes it hard to clearly indicate what you are talking about, as it is not possible to attach comments to specific lines of the code.

For novice programmers giving feedback line by line is more clear than feedback over a whole file. This feature is lacking in the current method of giving feedback.

Because of the above problems, the client is in need of a system in which students can receive line-based feedback on their programs.

4 Project Deliverables

The main deliverables of the project are the design report, the usage manual, a test plan, documntation and most importantly a system which satisfies the requirements specified during the requirements engineering phase of the project. The requirements for the system can be found in section 5.

Base system The base system will satify all user stories listed in 5.3.1. Below a few usage scenarios will be listed to describe the functionality of the base system.

A student logs in to the system. The student will now be able to see all the courses that they are enrolled in. The student clicks on one of the courses and ends up on the exercises page. The student clicks an exercise and ends up on the page of this exercise. Here the student is able to upload their project. The student's files are then displayed within the left sidebar of the exercise overview. The student can click one of the files and comment on lines in the code in the selected file. Teaching assistants and moderators will be able to comment on the files (line by line) as well as on the comments of the student. Depending on the permissions of other students they might also be able to view the files and comment on them/other comments.

A teaching assistant (hereinafter referred to as "TA") logs in to the system. The TA gets redirected to a dashboard that shows recent activities and notifications after logging in. The TA can see all the courses (s)he is enrolled in (as a TA as well as as a student). The TA can comment on exercises of all students for courses (s)he is a TA for, as well as reply to comments of other users.

An administrator logs in to the system. The administrator can now see all the courses that are in the system. The administrator can create new courses as well as new exercises for existing courses. The administrator can create groups of users. These groups are assigned roles that contain certain rights on the system. The rights that can be assigned to groups are listed in the user stories in 5.3.1.

The design report of the project will contain an overview of all design choices that have been made during the course of this project, including our thought processes and justification behind each of these design choices. These design choices consist of a detailed specification of all the requirements, which have been created in cooperation with all of the stakeholders of this project. The requirements have been converted to the user stories mentioned before, which can be found in section 5. Aside from the requirement specifications for this project, the design report will also include a detailed test plan and test results to guarantee the validity of the implementation of the project to a certain degree, making sure it has been tested sufficiently for an official release.

Design report The design report will describe the design of the system, including the reasoning behind the decisions that were made.

Usage manual The usage manual will describe how a user can interact with the system.

Test plan Description of how we test and which test we perform.

Documentation Description of how the system works, how it has to be installed and how it can be extended.

5 Requirements

5.1 Explanation MoSCoW Method.

The MoSCoW[2] method is used to describe the requirements for the system that will be delivered. All user stories listed under ”must” will be implemented as basic functionality of the system. The base system will only fulfill these requirements. All user stories listed as ”should” will be implemented if time allows it. All user stories listed under ”could” will be implemented if after implementing the ”must” and ”should” requirements there is time left. Realisticly the ”would” user stories will not be implemented in the given time frame. For each of the listed categories testing and documentation will be done as well. Thus testing and documenting the system that implements the ”must” requirements has priority over starting with a should requirement, likewise testing the ”should” requirements has priority over starting with the could requirements.

5.2 Definitions

Term	Definition
Course	A structure in which a number of people and exercises can be added.
Exercise	A structure in which projects can be started.
Project	A number of code documents that belong together, this can correspond to the work of a project group in the lectures.
User	Anyone using the system, i.e. student, teaching assistant, lecturer.
Student	A student, has the least rights within a group.
Reviewer	Someone who reviews the code in a course, i.e. teaching assistant or tecturer.
Moderator	Someone who can manage everything in a course i.e. a lecturer.
Administrator	Someone who can create new courses.
Comment	A remark on a piece of code or another comment.
Thread	A comment and other comments that react to it.

5.3 User Stories

5.3.1 Must

- As a user I must be able to log in and out of the system.
- As an administrator I must be able to create a course.
- As a moderator I must be able to add users to a course.
- As a moderator I must be able to add exercises to a course.
- As a user I must be able to create a project in an exercise of a course that I am a member of.
- As a user I must be able to upload code to a project that I am a member of.
- As a user I must be able to create a comment on a project on which I have that right.
- As a user I must be able to view a personal page with comments that are interesting to me.
- As a reviewer I must be able to see projects in a course I am a member of.
- As a reviewer I must be able to comment on a project in a course I am a member of.
- As a moderator I must be able to assign different rights to user groups within a course, so that they can have limited or extended rights.
- As a moderator I must be able to assign different rights to user groups within an exercise, so that they can have limited or extended rights.
- As a comment I must be able to be attached to a line of code.
- As a comment I must be able to be attached to another comment.
- As a system I must be able to show code with syntax highlighting and line numbers.

5.3.2 Should

- As a user I should be able to close a comment thread I started.
- As a moderator I should be able to add other users to a project.
- As a user I should be able to remove a user from one of a project I created.
- As a user I must be able to remove a comment that I made.
- As a user I must be able to remove a project I created.
- As a reviewer I should be able to attach a tag to a comment.
- As a reviewer I should be able to close any comment thread.
- As a moderator I should be able to control the visibility level of comments of different user groups (i.e. who can see what.).
- As a moderator I should be able to control the visibility level of projects (i.e. who can see what.).
- As a moderator I should be able to add users to any project in a course.
- As a moderator I should be able to remove a user from any project in a course.
- As a moderator I should be able to remove any project in a course.
- As a moderator I should be able to remove any comment in a course.
- As a moderator I should be able to choose a template for user rights in a course.
- As a moderator I should be able to create a new template for user rights.
- As a moderator I should be able to make the comments of different user groups invisible until I approve them.
- As a comment I should be unapproved if my author has limited permissions to post comments.
- As a moderator I should be able to approve unapproved comments.
- As a moderator I should be able to remove unapproved comments.

5.3.3 Could

- As a moderator I could be able to make a comment sticky to make it appear at the top of a thread.

- A system could be able to receive comments generated by a tool (i.e. checkstyle, etc.).
- As a user I could be able to upvote/downvote comments.
- As a user I could be able to connect my project to a git repository to receive the code from there.
- As a moderator I could be able to add a deadline to an exercise.
- As a moderator I could be able to add a final verdict to a project.
- As a user I could be able to view a dashboard for a project.

5.3.4 Would

- An external tool would be able to act as a reviewer.
- A user would be able to comment on the project of another user.

6 Project Risks

Below the possible risks of the project are discussed. For each possible risk the likelihood and impact of the problem on the project have been added as well as precautions and solutions for the problem. This analysis has been added to prevent unforeseen risks from delaying the project. The precautions and solutions will make sure that the project will always be finished in time and that a working version will be delivered.

Problem	**Precautions**	**Impact**	**Solution**
A member may unexpected have to leave the group	Develop a project plan where the basic system can be implemented with just 75% of the workforce, define optional extensions that can be added when there is time left after the basic implementation. Make a clearly defined plan of the tasks of every member such that it is easy to know what the tasks of this member were.	5/5	Distribute the work equally over the remaining students such that the increased workload is kept to a minimum
The laptop of a member may break	Make sure every member makes backups of their work after every meaningful change and it is easy to migrate the work to a new environment.	2/5	Force the member to get a replacement device, otherwise assume the member has to leave the group.
A member may forget to bring his laptop	Set up a daily reminder for this.	1/5	Send him back home to pick up his laptop and make him work overtime.
We might underestimate the amount of work needed to deliver the project	Plan for this, set out to have a basic implementation that should be completely finished within 7 weeks and then define optional extensions that can be done if there is time left.	4/5	Worst case scenario only finish the basic implementation. Work more to finish the basic implementation in time if necessary.

Figure 1: Risk Analysis

7 Planning

The project will be delivered before Friday 13th of April 2018.

Week (start day)	**Task**	**Deliverables**
05-02	Acquiring requirements and selection of tools.	Draft userstories.
12-02	Finalizing intial requirements. Project proposal.	
19-02	Project proposal. Test plan. Design.	Project proposal.
05-03	Project design. Implementation.	
12-03	Implementation	Test plan.
19-03	Design document. Implementation.	
26-03	Implementation.	First version design document, first prototype
02-04	Implementation. Presentation.	
09-04	Poster. Documenting/Writing manual.	Presentation slides
16-04	Presentation, poster, documentation.	Final Poster and presentation. Final Product.

8 Project Organization

8.1 Responsibilities

In this section the responsibilities are divided over the team members. The table with the responsibilities can be found in table 2.

Item	Description	Noël	Jan-Jaap	Joost	Tom
UT login	Connect our system to the login system of the university of twente			x	
Front end	Presentation part of the system		x		
Comment system	Making users able to view and create comments.				x
Database connection	Self explanatory.	x			
Database	Design and management of the database.	x			
JSON REST api	Creating an API so that information retrieval from server is simplified			x	x
Roles/rights system	Assign rights to users for viewing, commenting, etc.		x		
Testing	Self explanatory.	x	x	x	x

Table 2: Responsibilities

8.2 Procedures

In this section we will describe how we do things and what conventions or methods we use.

MoSCoW [2] We will make use of the MoScoW method to properly clarify the system that the client wants. In section 5 all user stories have been listed.

Agile Development [1] In this project we will work according to the Agile development system. Most notably we use the following principles:

1. Frequent delivery of working systems.
2. Close cooperation with client and stakeholders.
3. We work together daily (colocation).
4. Working software is the primary measure of progress.
5. Continuous attention to technical excellence and good design.

Test driven development We will make use of a test driven approach during development. Thus we will first create tests that guarantee that the system satisfies all initial requirements and then develop the system based on these tests. This guarantees that the system is thoroughly tested and functioning properly. Obviously we will also do other testing when the system is finished, but this should give us a good basis to begin developing the system.

References

[1] *Manifesto for Agile Software Development.* URL: `http://agilemanifesto.org/`.

[2] *MoSCoW Method.* URL: `https://en.wikipedia.org/wiki/MoSCoW_method`.