Gamified Programming Platform

a project in Information- and Communication Systems

Luleå Tekniska Universitet
971 87 Luleå, Sverige

Nils Fitinghoff
Software Engineer
Tester, Backend

Anton Jerhamre
Software Engineer
Frontend

David Sandström
Software Engineer
Backend

Mikael Hedkvist
Software Engineer
Backend

Henrik Nilsson Harnert
Software Engineer
Tester, Backend

Axel Sundbom
Software Engineer
Tester, Frontend
Repository Maintainer
Hardware Maintainer

Niklas Fuks
Project Manager

Tim Granström
Software Engineer
Backend

Christopher Rosenvall
Software Engineer
Frontend

Fredrik Bostrand
Software Engineer
Backend

Linn Danielsson
Software Engineer
Frontend

Tobias Axelsson
Software Engineer
Backend

Rickard Nordlander
Software Engineer
Tester, Frontend

Axel Vallin
Software Engineer
Frontend

Anders Mikkelä
Software Engineer
Frontend

Supervised by
Peter Parnes

Abstract

This report presents how gamification could be applied when learning programming. The key aspects of gamification were investigated by conducting interviews with teachers at Luleå University of Technology and a theoretical study. The produced platform shows how some parts of gamification can be used when learning programming. The platforms primary target is university students taking computer science courses at Luleå University of Technology. The product is a platform for increasing learning by gamification. In the platform it is possible for teachers to create courses with gamification elements such as being able to choose between an adventure map, a leader board, a progress bar and badges. The teacher can also create assignments and choose how the assignment should be tested, input/output test and/or lint-test the code. The student can join courses, solve assignments and get feedback if the programming assignment have been correctly solved.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Goals . . . 3
- 1.3 Expected result / requirements . . . 4
  - 1.3.1 Initial requirements . . . 4

2 Working methods **5**
- 2.1 Sprints . . . 5
- 2.2 Groups . . . 5
- 2.3 Roles . . . 6

3 Prestudy **7**
- 3.1 Interviews . . . 7
- 3.2 Platform . . . 8
- 3.3 Theory . . . 9
- 3.4 Comparison interviews and theory . . . 10
- 3.5 Workshop . . . 11

4 Design **14**
- 4.1 Decisions . . . 14
- 4.2 Use-cases . . . 16
- 4.3 Frontend . . . 17
- 4.4 Backend . . . 18
- 4.5 Tester . . . 18

5 Future work **19**
- 5.1 Frontend . . . 19
- 5.2 Backend . . . 19
- 5.3 Tester . . . 19

6 Discussion **20**
- 6.1 General . . . 20
  - 6.1.1 Work . . . 20
- 6.2 Conclusion . . . 21
- 6.3 Acknowledgements . . . 21

A Interview questions **22**

B Workshop scenarios **23**

C Installation instructions **25**
C.1 Tester 25
C.2 Backend 25
C.3 Frontend 26

D Cas-diagram **27**

Chapter 1

Introduction

1.1 Background

At Luleå University of Technology, there are many courses wherein programming is a major subject. Students sometimes face difficulties knowing whether or not they have programmed correctly until a lab supervisor or examiner checks the code. This dependency on human interaction generally leads to slow feedback, and reflections regarding the programming tasks are often forgotten between the time of writing and the actual checking.

Apart from that, students frequently find that their studies in programming are useless, because they do not understand where- and how it can be used. In other words, they lack the context they would need to actually motivate themselves to learn programming. By providing a context, the students can be 'fooled' into thinking they have a use for their knowledge until they actually see the real-life uses.

To improve the time required for feedback and make the assignments more fun, an interactive on-line tool for coding practice was requested. There are many different applications on the internet that already does parts of this, but none that are freely available and suitable for holding university courses in.

None of the tools that were investigated were able fill all of the niches that teachers may want to use to create exercises. They either were built like a game and had no pedagogic parts or provided pedagogy only and were quite boring. Also, many of the tools investigated were too complicated for teachers to implement into existing courses, so they would need to be simplified before taken into use, or too much time would go into maintenance.

1.2 Goals

The purpose of gamification is to take elements from games and apply them in other areas to create an increased commitment within the applied area. The target group is mainly students studying at Department of Computer Science, Electrical and Space Engineering with a programming direction but also other educations within LTU that contain programming parts, for example teacher education where programming will become a central part in the future because of a revised curriculum for elementary school and high school

that passed in the fall of 2017.

To reach the goal of the project, a platform that embodied these ideas was needed. The goal was to have a platform that was easy for both users and teachers alike. Gamification was a big part of the platform but a realization that not all teachers wanted to use these elements was reached and because of this it needed to be simple to activate and disable gamification features for courses as the teacher saw fit. As such, a big goal during the development of the platform was to make features very modular.

1.3 Expected result / requirements

The expected result was to deliver a web-based platform for learning programming. The platform should include gamification methods that stimulates the student so that they want to program more and be better at programming. The students should be able to write their code in the browser and then be able to validate if they have the correct outcome of the assignment. The platform should support progress and different levels so that every student is challenged. The platform shouldn't be limited to LTU even though LTU is the primary source that will use the platform.

1.3.1 Initial requirements

- Full solution for writing and testing code.
- Immediate feedback.
- Show progress/statistics.
- Involve gamification.
- Support multiple programming language.
- Teachers should be able to construct assignments.
- Easy navigation.
- Log in by CAS.
- Primary target is LTU students.

Chapter 2

Working methods

2.1 Sprints

Agile work flow was used during the project. This was done by dividing the weeks into sprints. A sprint was expected to be one week but sometimes sprints become longer due to heavy workload. Every sprint was planned before the current sprint was ended. The group leaders planned the sprints, one from each current sub group along with the project leader. taiga.io, a sprint planning tool kept track of the backlog and different task assigned to individual group members. This showed which task was completed and which needed to be helped with. During the project some of the members evaluated the tool taiga.io and found that it could have been better if it was possible to add tasks to more than one person, if it was easier to assign points to a task and move a task to other sprints. A decision was made to keep taiga.io because the members had gained experience with it and it would be impractical to split the backlog to a new tool.

2.2 Groups

All of the different task sections were split into groups. During the project it was laborated on how many number of groups would be the most efficient for the project. The working model that was found was to have three groups with 4–5 members each as long as there were enough tasks in different sections for three groups. Every week at least one time all members met and discussed the previous week and the upcoming week. This became a meeting with 15 people that ended up talking over each other which felt ineffective. This was solved by minimizing the meeting members to 3–4 people: the project leader along with one group leader each from the frontend, backend and tester group. The leaders were democratically chosen in each active group and their task was to tell the other group leaders what they have and will accomplish during next period. They then went to their own groups and told them about the meeting. This worked well during the first weeks but less and less information were distributed to the group member as time went by. When this problem was then identified it was to short time left on the project to change the meeting method and instead it was simply decided that everyone needed to try and keep the quality of the distributed information up like it was in the first few weeks.

The different groups that has been presented are:

- Interviews
- Workshop
- Platform
- Theory
- Frontend
- Backend
- Tester.

2.3 Roles

All the project members had a responsibility that the project should succeed and it was important to contribute to the extent that each member was capable of.
The three major roles have been assigned during the project, these are
Project leader — Niklas Fuks which had an overview of the project and its sub tasks. Planned the meetings, sprints and held presentations with demos.
Groupleader and software engineers -
Axel Sundbom (tester)
Tobias Axelsson (backend)
Axel Vallin (frontend)
Software engineers -
Linn Danielsson
Nils Fitinghoff
Mikael Hedkvist
David Sandström
Anton Jerhamre
Christopher Rosenvall
Fredrik Bostrand
Henrik Nilsson Harnet
Tim Granström
Rickard Nordlander
Anders Mikkelä

Chapter 3

Prestudy

Since the goal of the project is a platform that can increase learning using gamification it was need to conduct a study about which parts of gamification is relevant for the platform and how should they be represented. Peter Parnes, the project owner suggested that the information could be collected by reading literature, having a workshop with teachers at LTU and some teachers outside the university and by interviewing a few teachers teaching in courses that could have use for the platform as a part of their course. An investigation was done about what tools and existing platforms are already available and if they could be reused and built on. This investigation was done by the platform group.

3.1 Interviews

The interviews were done by interviewing a couple of teachers teaching in courses at LTU. A group of four members set up individual meetings with each respective teacher where they asked relevant questions regarding a gamified coding platform. The most comprehensive and general questions during the interviews can be found in appendix A. The interview answers from the teachers were summarized to identify the most important elements that could be used. The most important elements were those which were repeated during several interviews and those that the interview group found most interesting. The identified and summarized elements are:

- Mandatory assignments would increase the participation frequency but would be less fun.
- Public scoreboard may be demotivating.
- Best result could be displayed by a score.
- A balance between school and fun.
- Levels / different difficulties.
- Even the weaker students should have a chance to solve assignments on the easier levels.
- No points or badges.

- Anonymous questions.
- Good for the lower grades.
- Must be easy to adapt the system into a course.

These summarized answers was a good starting point to see which parts should be included later on in the platform and which parts should be excluded.

3.2 Platform

To meet the requirements and goals of the project it was clear that a platform that could compile and verify code was needed. When the project started, the project owner had suggested tech.io which is a collaborative platform to share coding assignments through open-source "playgrounds". At a first glance, the platform seemed to match the needs of the project, but it was unsure if the platform would have an open API which the project communicate and work with. The developers of tech.io, CodinGame was contacted to find out if they would be open to sharing their API.

While waiting for an answer from tech.io, an investigation into alternative platforms to tech.io was launched and after some time of searching for a suitable alternative over the internet it was concluded that no platform found really matched the requirements that were set.

After waiting for several days, tech.io finally responded where they mentioned the possibility of using "code snippets" to embed coding assignments onto other sites than on tech.io, but they were evasive with the question if they would make their API open and never really answered this. After some time of consideration, a decision to not use tech.io was made because of the risk of becoming too dependent on their platform.

Because of this, creating a code verification platform/API from scratch was considered. While discussing about making a code verification platform from scratch, a new alternative was brought into light. A computer science student at LTU was currently creating a platform named Sockr for code verification and once again, it was considered to use a third party platform.

After some discussions with the student, he explained that he would be open for the project to use Sockr. A thorough investigation of the Sockr source code was however needed to make sure that it was actually useful for the project. When diving deeper into the source code of Sockr it became clear that once again, the project would not be able to use this in a good and sustainable way. While Sockr did have the functionality that the project needed, the source code was mostly undocumented, didn't have a clear API and the backend and frontend were too tightly coupled to be used in a sustainable way.

As such, after much research it was concluded that the best and most sustainable thing for the project would be to develop a code verification/compilation platform from scratch, effectively making the project completely independent of other services.

3.3 Theory

A theory study was conducted where the task was to find what articles and the books have learned about gamification. A group with a couple of members sat down and read books and articles mostly found on internet. The group found a number of useful elements from gamification theory that could possibly be implemented into the platform, it was however realized that not all these elements could be implemented all at once and it was later needed to identify which elements who would gain priority. The elements that the theory group found useful were:

- Reward
  - Reward should give positive information since it increases the motivation.
  - Rewards of controlling type is demotivating.
  - If rewards ends it may be demotivating later on.
  - Customization, collect points and be able to spend them.
- Progress
  - Progress should be in increasing order.
  - Be careful with which parts should be graded.
  - Large numbers are better.
  - Bonus for overachieving.
- Freedom
  - Freely choose between assignments.
  - Possibility to redo assignments.
- Game mechanics
  - Chance?
  - Duels?
  - Take turns?
- Leaderboard
  - Anonymous leaderboard.
  - No disincentive
  - No ranking, demotivated to have low rank
  - Populate the leaderboard with fake users

3.4 Comparison interviews and theory

To find the key aspects of gamification from both interviews and theory a comparison was made where a group compiled the data from the previous received feedback and other notes collected into a new list that could act as a basis for future decisions about gamification. When compiling the list it was also important to take into account what parts that could actually be implemented during the project and which ones could be possible future work. The newly formed list contained the following key elements:

- A lot of small assignments
  - A lot of structure, easy to understand the feedback.
  - Levels.
- Progress
  - Progress in increasing direction.
- Freedom
  - The mandatory assignments have highest participation but may be demotivating.
  - Students should have different options to solve a task.
  - Deadlines may be demotivating but also sometimes necessary.
  - Redo an assignment multiple times.
- Rewards
  - Should act as a positive information about the student skill level.
  - Rewards should not act as a controlling factor.
  - Redeemable points, e.g. avatars.
- Feedback
  - Teachers should be able to check if students have understood the assignment or not.
- Leaderboard
  - Divided opinions about how good leaderboards are.
  - Anonymous leaderboards.
  - Fake users so that no one is last.

3.5 Workshop

A workshop was presented as an idea from the project owner Parnes as a way to find out how different teachers think that the system would look like and what features they feel should be included. A group of five people started to work by writing different scenarios for the workshop, see appendix B. As a preparation for the real workshop, these scenarios were then tested on a few members of the project group where the members were acting as teachers and they had to implement the planned workshop scenarios. This gave the workshop group some thoughts about what could be improved before the ordinary workshop with the teachers. The workshop with the teachers was made by splitting the teachers into two different groups and then they had 10–15 minutes to discuss the scenarios in the group and then another 10 minutes for discussions with the other group. The first scenario task was to discuss which type of system and elements could be available that would have motivated them as students. In the scenario 2 the groups had to imagine themselves being the examiners of a course and the groups got to discuss which limitations should be done in the system but also once again, how they could motivate their students. During the scenario 3 the task was to focus on embodying the ideas and thoughts from scenario 1 and 2 into an e-tool and how the general design should for said tool look like. A compiled list from both groups together from what was discussed and concluded during the workshop:
Result from scenario 1

- Let assignments interlink.
- Be able to choose which assignments should be done.
  - Hard to grade, every goal of the course need to be met.
  - Assignments with similar content.
- Constructive alignment, how goals, assignments and examination interacts.
- Early and quick feedback.
- Formative assessment, continuous feedback.
- Groups assignments, hide the personal result.
- AI Avatar that gives feedback.
- Lot of visualization of the result.
- Present the purpose of the assignment.
- Different angle of the design depending on the user which is solving the assignment.

Result from scenario 2

- Not everything in a course should be change at same time.
  - Old assignments could be divided.

  - Shorter feedback loop.
  - old assignments with new context.
- Updating the course introduction is cheap and makes the goals of the course more clear.
- When the knowledge should be tested.
  - Thoroughly, min/max, median or average.
  - In the end there is a written exam.
- Invoke from elder students.
- Peer-to-peer feedback.
- Test one implementation even if it's bad.
- Students create assignments for each other.

Result from scenario 3

- Anonymous questions live
  - "Did you understand what I said?"
  - Direct feedback.
- Direct chat, students and teacher.
- Auto generated feedback to the teacher.
  - Aggregate data from relevant measurements.
- A student knowledge bank.
  - Wiki.
- Code correction / evaluation.

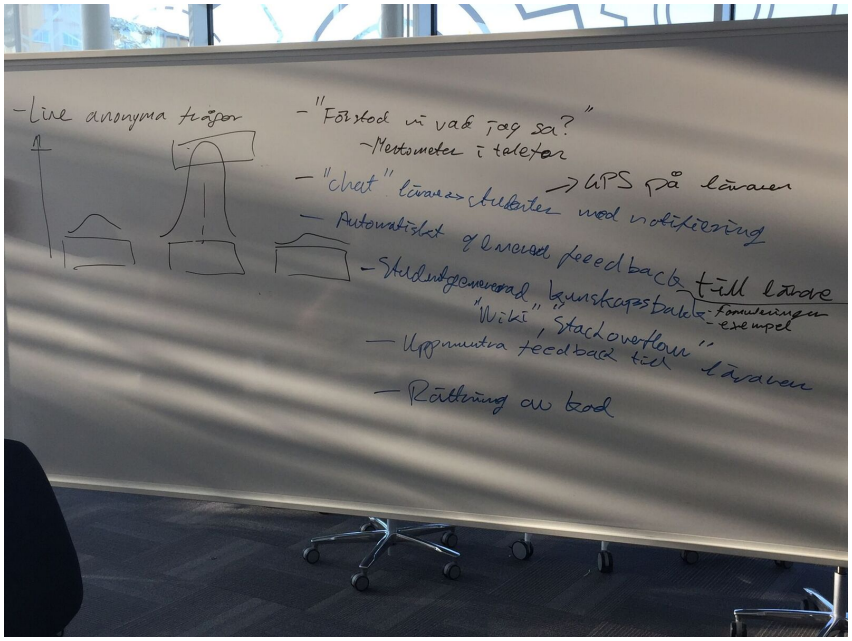

Figure 3.1: A mockup over interface from workshop-group 1.

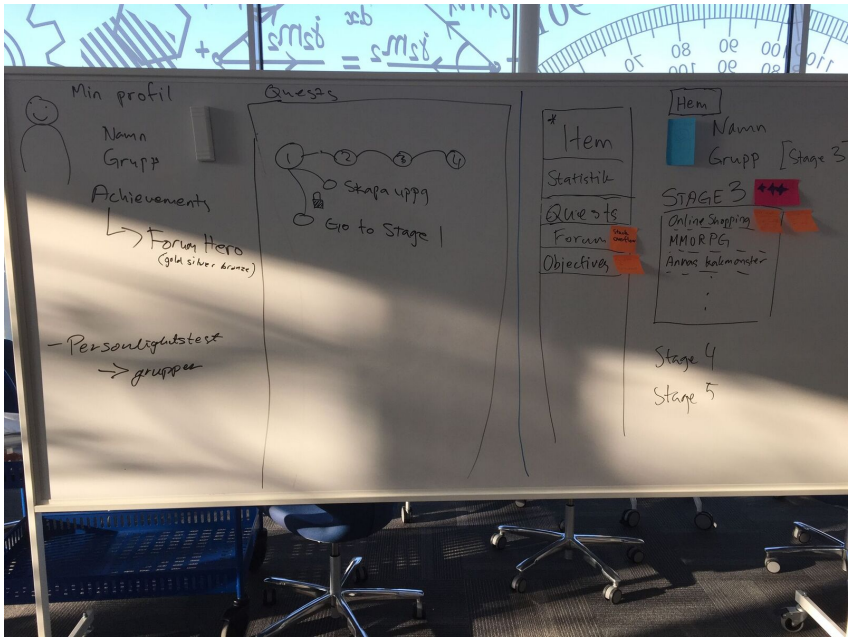

Figure 3.2: A mockup over interface from workshop-group 2.

Chapter 4

Design

4.1 Decisions

When the prestudy was completed it was time to make some decisions about the content of the platform, how it should look and how different parts should be implemented. To give a picture of how the design on the user interface should look like, a group sketched some simple mockups so that every member could have the same starting point and vision.

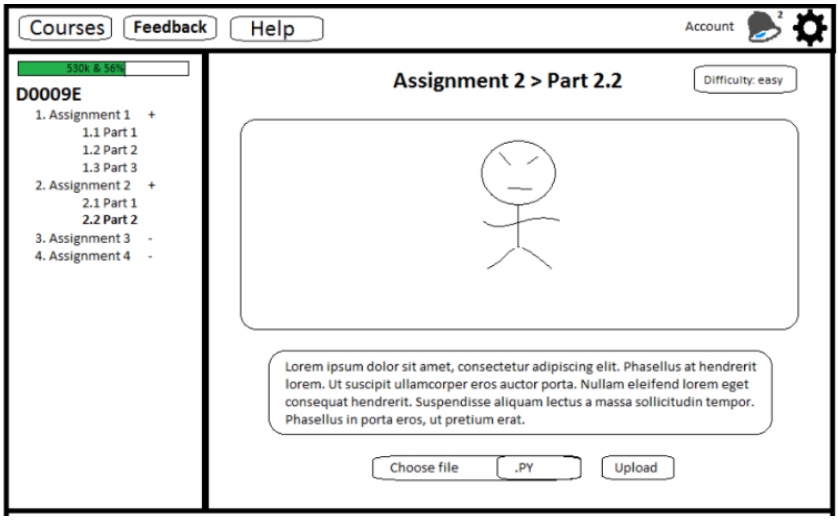

Figure 4.1: An initial mockup for the assignment page.

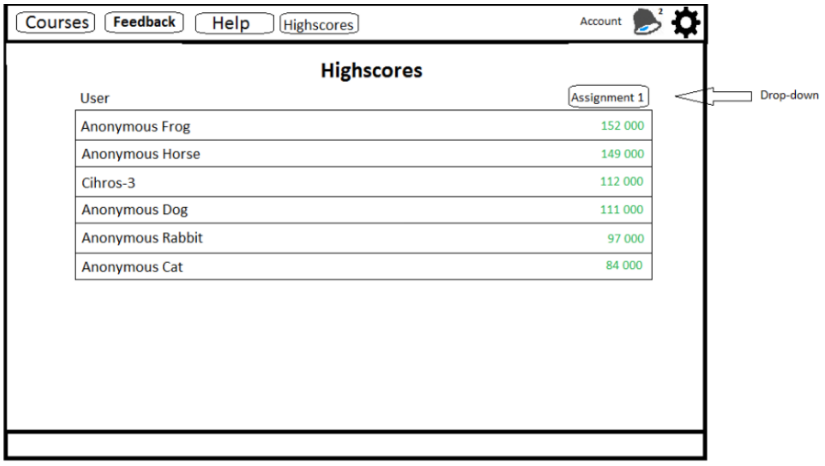

Figure 4.2: An initial mockup for the leaderboard.

The mockups were only used to have a base and the rest of the design would grown during the implementation time and it was mostly the frontend group that designed the pages. During the prestudy it become clear that the whole system flow was needed to be implemented. No one of the found an existing solution that would fit with the system and it couldn't be ensured that it will work over time. The system was divided into three parts. Tester whose main task was to implement a system that is able to test code that students write and send back some feedback. The frontend should implement the user interface and the gamification visualization. The backend is the in the middle and it will take the code from frontend and load a test from the database, send the submitted code along with the tests to Tester and then give the response (feedback) back to the frontend.

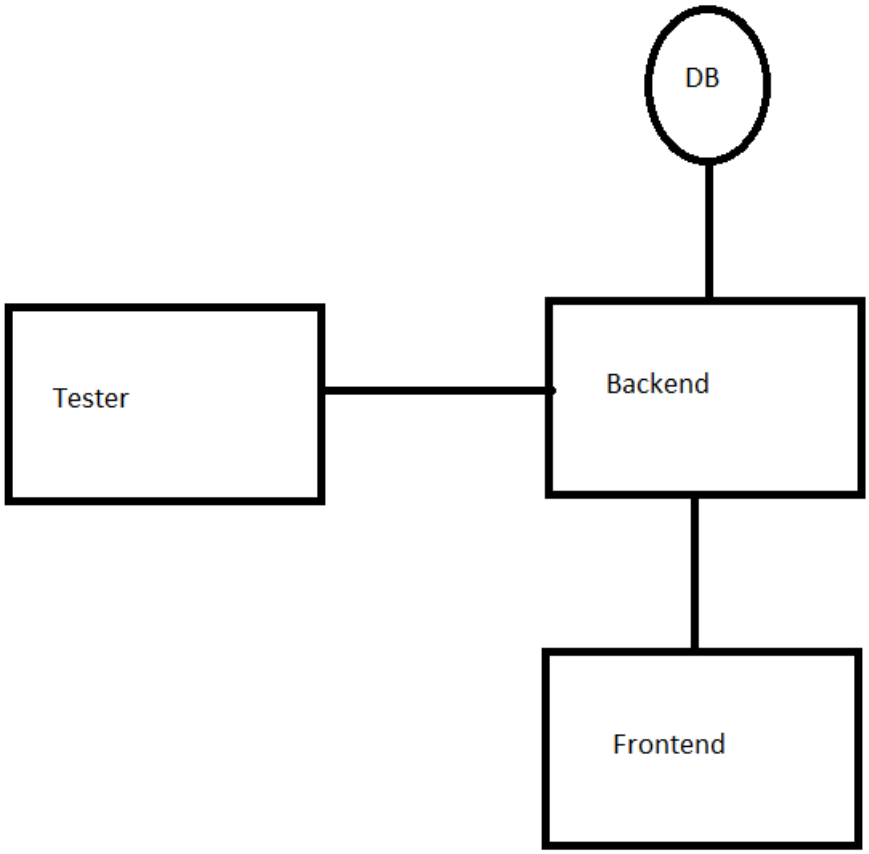

Figure 4.3: A simple sketch over the system parts and how they are connected.

The time estimation for every separate part was difficult to estimate since the concept was fairly new for everyone. The decision of what programming language to use was based on the criteria, it should be fullstack meaning the same language should be used in every part so that if there is any interaction between the three groups it should be easy to understand the code. It should also be fairly modern programming language. The two alternatives was Python with frameworks or JavaScript with other frameworks. A democratic choice become JavaScript using Node.JS for backend and Tester, MongoDB as a NoSQL database and Angular4 for frontend.

4.2 Use-cases

In the system there are two different roles, teacher and student. The teacher is able to create courses, assignments and invite students to the courses. The student can solve assignments. Students who want to solve an assignment go to the site, log in, go to the specific assignment and write the code required in the assignment and submit it. The code is sent to the backend that saves the code to the database for future changes and loads the tests that belong to the assignment from the database. It then sends both the tests and the code to Tester which tests the code and sends back the output through the backend which in turn calculates the correct number of points that the student should be

given for their solution. The output is now served to the frontend and the student can see if the code has passed or not with feedback.The teacher that wants to create a course with assignments log in, clicks create a course where the teacher writes the descriptions and chooses the gamification parts that should be available for that course. Then the teacher creates an assignment where the teacher writes the assignment and chooses and writes the output that should be tested.

4.3 Frontend

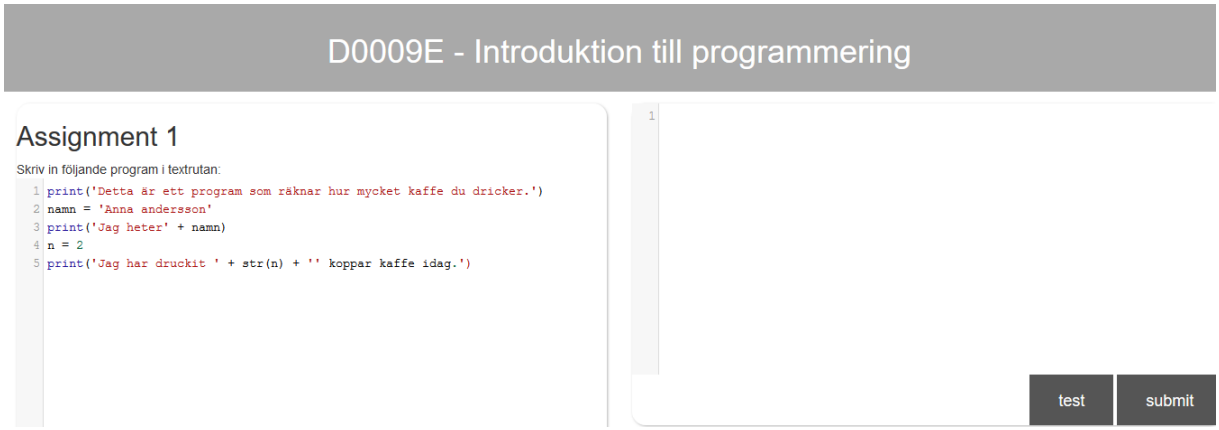

Figure 4.4: A screenshot over how assignment looked when using codemirror.

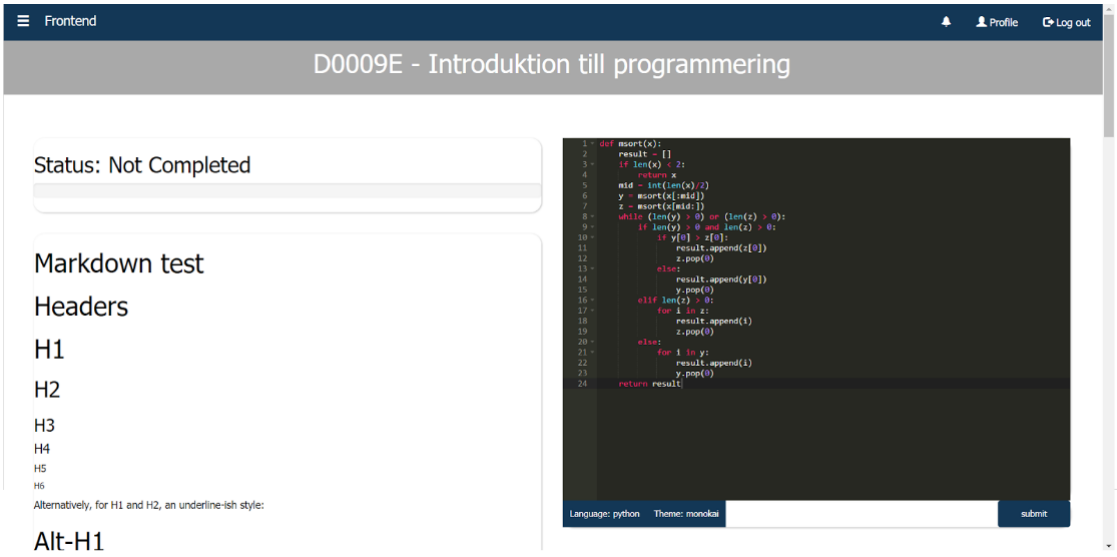

Figure 4.5: A screenshot over how assignment looked when using Ace.

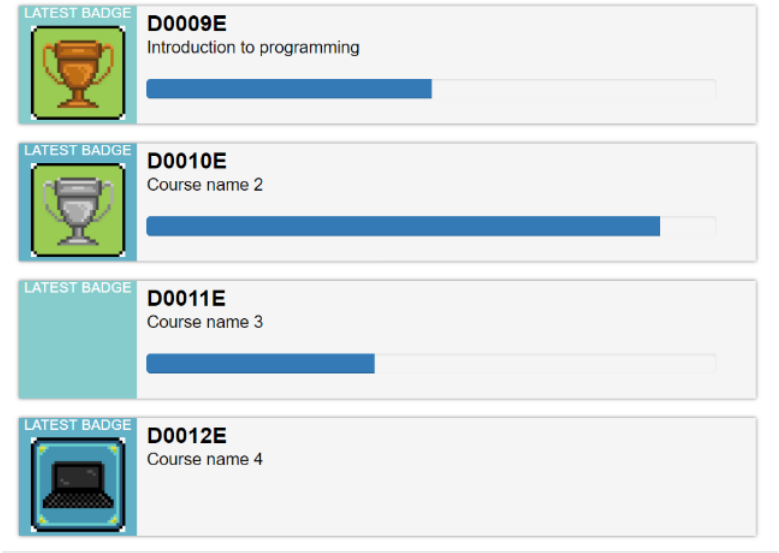

Figure 4.6: How the progress and badges look like.

4.4 Backend

A backend group was formed after all the prestudies were finished. Beginning to work on the backend, the first decision that had to be made was what it was going to be developed in. Since it had been decided previously that the project would be a full-stack JavaScript solution the natural decision was to use Node.JS as server framework with express.js as a web framework to handle routing, for database it was decided to use the document database MongoDB.

The first few days were mostly spent on getting acquainted with Node.JS and its work flow. After the group had gotten a general grasp of things, a coding standard was decided on and the group was divided into sub-parts in the backend. The parts that the group was divided into were routes, database and continuous integration.

It was decided that it was very important for continuous integration to be a part of the project. The reasoning for using continuous integration was that having a modern workflow where you could push changes to git and have them available on a live test server would be important to keep the project effective. Using continuous integration was also a good way to effectively separate production and development builds and a way to ensure that production builds always kept a certain standard and robustness. However, while the idea and reasoning for using continuous integration was good, the concept wasn't put to as good use as it could have been. For the first half of the project, the continuous integration wasn't really used they way it was intended too, mainly due to communication errors and the fact that it wasn't needed as much, but for the second part of the project it was used more.

4.5 Tester

Chapter 5

Future work

5.1 Frontend

5.2 Backend

5.3 Tester

Chapter 6

Discussion

6.1 General

The projects outcome can be considered successful. The system that has been delivered has the ability to test code and give feedback to students and has elements of gamification. The gamification part can be further developed but we have built a good basis for it. The gamification and quick feedback that is readable is very important when someone is new at coding. The standard outputs from a compiler can be difficult to understand if you're new. The availability to see if the code written is correct and and matches the teachers output is important since sometimes feedback from the teacher or lab supervisor takes long time and an assignment can take long time to complete. Many new students are afraid to ask dumb questions since they don't want to be classified as bad. A anonymous question handler would be suitable so that everyone can ask questions and answer them completely anonymously. The major difficulty here is to choose which parts should be included. Everyone that was asked thought different and we choose the parts based on repetitive answers from teachers and our own preferences. We also thought that when a teacher creates a course, the teacher should have the ability to choose what gamification parts should be included for the specific course. This may, based on the prestudy, be a bit bad since it's shown that when a reward that is usually given for a something is removed, the motivation decreases. There is still a lot of things that could be implemented from gamification and from the question form and hopefully someone will continue implementing features and test this system on students. There is a large part left which isn't covered and that is to set up a workshop with students were they test and gives feedback about the system. It was planned to be done but due to prioritizing functionality this was left.

6.1.1 Work

During the project there has been many confusions what should be done, why and how. We were 15 persons in the project although the project was dimension for approximately 7-9 persons. Non of the project members have ever worked in a such large group so it was difficult to start the project due to some people took more place than others. The projects outcome was unclear for a very long time and projects specification and requirements should be more detailed, this for help the project to start faster. The prestudy part

took too long time, there was many parts to cover and required. We was told to have a workshop but the workshop was held when the implementation should started so the project stayed up for a while waiting for the result of the workshop. Due to another heavily course simultaneously in period 1 most of the project members prioritized that course and that meant that some task wasn't completely finished that time it should be done and become overhanging to next sprint. Since there was 15 persons in the project it become some groupings and the communicating lacked between the groups. This was a hard problem to solve and the only way to solve was to encourage everyone to sit in the classroom and talk to each other. Two different meeting methods were tested, the whole group and leaders from groups. When the whole group was collected the meeting become more that one or few talked and the other sat quiet and the meeting become long and nothing felt concluded after the meetings. So we changed that every group selected a leader that went to leader meeting. this felt good but the leaders didn't bring all information to the meeting and didn't always remember everything from them so that the information the group got from their leaders may not be complete. So the conclusion is that the projects groups should be smaller, about 7-9 persons in a project is more suitable since the persons taking this course mostly doesn't have experience working in projects and more less in large groups.

6.2 Conclusion

Gamification has become a large part of motivation. By getting rewards for performed works the motivation and satisfaction increases. Gamification with immediate feedback in programming is fairly new since learning programming has grown and spread to lower ages, a tool for learning programming quickly is needed. The platform provided here is dynamic in that way that there is no limitation in the difficulty in the assignments, the creator of the assignment is that one which sets the level. By that said, even since the target is university students, the platform is also applicable to younger students. There are endless choices for new gamification parts that can be implemented, only the imagination puts a stop to which parts can be implemented. The platform delivered is an initial working product that is usable but can be expanded to fairly low cost.

6.3 Acknowledgements

We would like to express our gratitude to the teachers that have been available for interviews and attended the workshop that has helped us understand different gamification aspects. We would like to thank Prof. Peter Parnes for the platform idea and the tuition during the project.

Appendix A

Interview questions

- Do you find that student appreciate e-tools?
  - Which tools do you think is needed to engage students?
- A large problem (at least at the computer science program) is that may students are shy and doesn't dare to ask questions. Do you think that an e-tool in the classroom could simply this. e.g. anonymous feedback services?
- Do you experience that there is different types of students that response on different types of motivation?
  - If so, which trends have you experienced?
  - Do you think that it is possible to reach more students by e-tools?
- Are you familiar with gamification?
  - How did you get it?
  - Which pro and cons do you see?
  - Which pitfalls have you experienced?
- How much time would you as a teacher be able to spend to create assignments?
  - How much time do you spent on creating regular assignments?
- What do you need available to see that this tool will work?
  - Which difficulties do you see by using a e-tool, e.g. Tech.io?
- How can we motivate you as a teacher?
- What would you think about if you should create a similar tool?

Appendix B

Workshop scenarios

Introduction 10min We are studying the fifth year at computer science program at LTU and for now doing our project. As a part of our prestudy are we hosting this workshop where we will investigate possibilities through a couple of scenarios. You will do a couple of assignments in groups and then present your result and discuss these with others.

Scenario 1 10 min + 10 min discussion
You have just started a course where you will learn to program in a language you don't find interesting. The teacher has given you lot recommended assignments but non seems interesting. Which type of system or moments would motivate you to do the assignments. Dream free.

- Which subject do you think has this problem?
- Do the assignment needs to be presented? If so how?
- Which material is needed?
- What should be on a list with highlights?

Scenario 2 10 min + 10 min discussion After many years you are the teacher for that boring course. The boss comes in to your office a Friday afternoon and says that the course moments are to few and you need to update them till Monday. You sits down as the motivated teacher you are and thinks about a system than would motivate students, you remember your old concept.
You want to make this work for a long time, which limitations must you as a teacher do?

- What is most time consuming?
- Is it possible to sort your highlight after time consuming?
- How long time can the moment take to develop and maintain?

Scenario 3 15 min + 10 min discussion Now when you have a working system and it scales along the new courses you teach in, but the students wants a e-tool. How does this tool look like?, Think web besides details and try to catch the users experience.

- How do you do inputs?
- How does your highlights look like practical?
- Do you see any limitations?

Appendix C

Installation instructions

Learning as a service was built and tested on debian linux 9 (stretch) with node 8.6.0 and npm 5.3.0.

C.1 Tester

Tester consists of two components; Manager and Runner. Managaer replies to requests from the backend and manages docker containers that run arbitrary code. Containers are used to ensure that some test A does not interfere with some later test B by modifying the executing environment.

1. Clone the repo: **git clone https://github.com/ax-rwnd/d7017e-project**
2. Change directory to the Manager folder: **cd d7017e-project/tester**
3. Install the dependencies for the Manager: **npm i**
4. (Optional) Select languages by adding/removing dependencies in **Makefile**. For instance the line **all: python27 python3 java c # haskell** selects the languages Python 2.7, Python 3, Java, C, but not Haskell (since it's commented out).
5. Run the Makefile: **make**
6. (Optional) Set preferences for Runner in **config/default.js**. There, things like queue lengths and ports may be configured.
7. Move back up to manager: **cd ..**
8. Start Manager: **node server.js {PORT}**

C.2 Backend

Backend is the state-managing component. It uses MongoDB to store information that it receives while processesing frontend requests and tester results.

1. Install och configure MongoDB.
2. Clone the repo: **git clone https://github.com/ax-rwnd/d7017e-project**
3. Change directory to backend: **cd d7017e-project/Backend**
4. Install dependencies: **npm i**
5. onfigure database address/port in **Backend/config/default** and **Backend/config/production** IP/Port may differ between the files, should you want to use different databases for testing and production. To select one of these files, set the **NODE_ENV** environment variable to **production** or **development**.
6. Start the backend daemon: **npm start**.
7. (Optional) Start start backend in the foreground: **node ./bin/www**

C.3 Frontend

Frontend is the part that the users see. It builds on Angular for UI and contacts backend for functionality.

1. Clone the repo: **git clone https://github.com/ax-rwnd/d7017e-project**
2. Change directory to frontend: **cd d7017e-project/frontend**
3. Redirect frontend to backend: **sed -i "s/ \(backend_ip: \)'.*'/ \1'https://your_backend'/" src/environments/environment.prod.ts**
4. Tell were the global ip for frontend is: **ed -i "s/\(frontend_ip: \)'.*'/\1'https://your_frontend'/" src/environments/environment.prod.ts**
5. (Optional) Repeat step 2 and 3 for **src/environments/environment.ts**
6. Move or link your ssl-certificates **ln -s encryption/private.key.default encryption/private.key && ln -s encryption/server.crt.default encryption/server.crt**
7. Start server: **–ssl 1 –ssl-cert ./encryption/server.crt –ssl-key ./encryption/private.key –live-reload false**

Appendix D

Cas-diagram

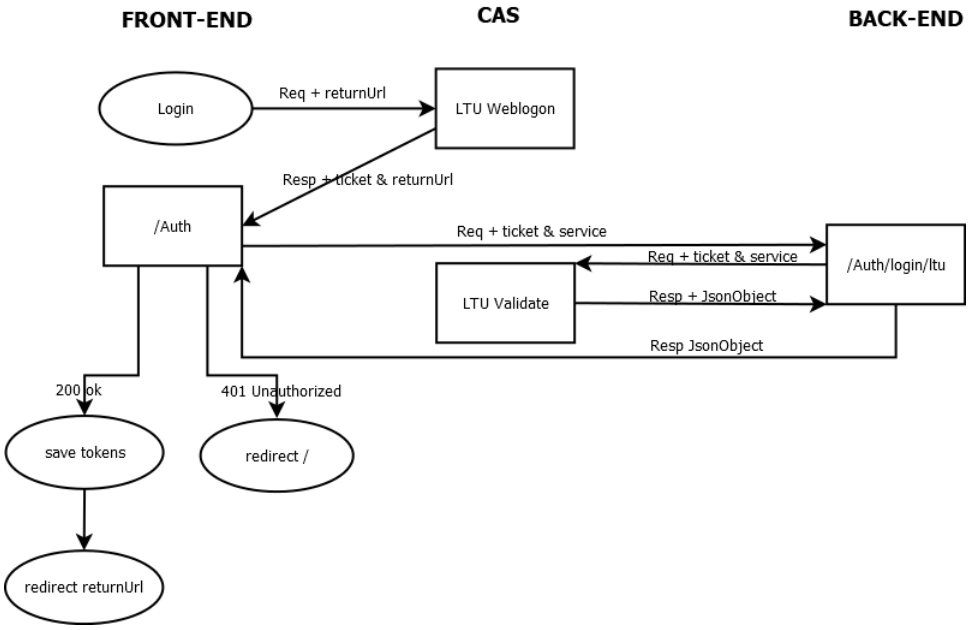

Figure D.1: A diagram showing how CAS-login is implemented.