University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Computer Science	Spring semester, 2015 Open / ~~Restricted~~ access
Writer: Heine Furubotten	.... (Writer's signature)
Faculty supervisor: Hein Meling External supervisor(s):	
Thesis title: The Autograder Project: Improving software engineering skills through automated feedback on programming exercises	
Credits (ECTS):	
Key words: Digital Learning, Continuous Integration, Version Contol, Test-Driven Development, Autograder, Grading	Pages: .... + enclosure: .... Stavanger, ... Date/year

University of Stavanger

The Autograder Project: Improving software engineering skills through automated feedback on programming exercises

by

Heine Furubotten

A thesis submitted in partial fulfillment for the degree of Computer Science

in the
Faculty of Science and Technology
Department of Electrical and Computer Engineering

June 2015

Declaration of Authorship

I, Heine Furubotten, declare that this thesis titled, 'The Autograder Project: Improving software engineering skills through automated feedback on programming exercises' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

“When you talk, you are only repeating what you already know. But if you listen, you may learn something new.”

- J. P. McEvoy

Abstract

Many students find it difficult to learn programming skills. One reason for this difficulty is that feedback from teaching staff is often slow. The Autograder aims to improve student learning through rapid feedback and to stimulate self-learning. The Autograder project provides a web-frontend and a server back-end that has been developed for automatically correcting and evaluating solutions to programming exercises submitted by students. Correcting and evaluating student submissions are based on teacher written test cases, which the submitted solutions should pass. From this the students get rapid feedback and a score on the programming assignments. Autograder depends on a custom built continuous integration service, test-driven development and a version control system to deliver its services.

The Autograder has been successfully used in a master-level course at the University in Stavanger. Students and teaching staff was monitored and interviewed through their assignment work. Autograder provided an efficient way for the students to reach their potential, through rapid feedback on submitted exercises. The teaching staff got a better overview of the students progress, which made it easier to follow up each student. They was relieved from the burden of manually correcting assignments and could more easily identify pain points in the exercises. Together with oral examination in the lab, the test results obtained through the Autograder was used as the basis for grading the lab assignments. The lab submissions made it much easier for the teaching staff to prepare for the oral examination. Although we cannot draw strong conclusions at this point in time, we have some data points that seem to indicate that student learning has improved also on the final written exam.

Acknowledgements

The author would like to thank Hein Meling for valuable discussions and input throughout the making of the Autograder application and project details. Would also thank him for using his course Distributed Systems as an use case in this thesis. Tormod Erevik Lea created all the test cases used in the Distributed Systems course and was a valuable asset in designing the project and application. The author thanks for his efforts, which was a huge contribution for making the full scale use case possible. When investigating how the software industry work with their software stack, Stig Rune Malterud and Leon Mwazange took the time to answer our questions. For taking their time to help, the author is thankful. A special thanks are going out to all the students of 2015 in the Distributed Systems course for valuable feedback during the project and helping improving the application.

Contents

1 Introduction

When teaching advanced programming courses at colleges and universities, it is common to give the students a practical way of learning the academic material, often in the form of lab assignments. Lab assignments provides a good way of letting the students get hands on experience, allowing them to get a deeper understanding of the course material. However letting all the students dive into the assignments and creating lots of diverse solution to the problem will often, especially for advanced programming assignments, lead to less control over what the individual students actually have learned and what they have accomplished through the assignments. From a students perspective it can also be hard to get all of the aspects in an assignment right while waiting for feedback on their work. Teachers usually have limited time for each course and cannot always deliver detailed feedback on the solution to every student on demand. With todays work process the students might miss some details of the curriculum.

The idea of having the students solutions digitally corrected is not a new one. Solutions for having the students manually upload their codes for static analysis and then getting a report back about the state of the code is one of these solutions. These systems means the students will have to finish their software and then upload their solutions to a website and it will be analysed in this system. Another system is to have students solution runned through test cases in their editor and upload the results to a remote server, through plugins. These solutions have an impractical way to hand in and process the solutions, and are not rooted in how the students will work when they have graduated.

In this thesis we try to systemize the workflow for both students and teachers for the workflow practiced in the software industry, and make the process, from students working on their solutions to teachers receiving the finished solution, more user friendly. This new workflow has also been monitored to investigate the effect this has on the students and teaching staff. The system developed for this thesis also tries to generalize the system for use in as many courses and lab environments as possible. From this an application named Autograder was created. This application have a custom continuous integration service for automatic building of students solutions, and a web service to deliver the results back to both students and teaching staff. This application is also fully implemented with GitHub, a git management system. This integration makes the course management as automatic as possible and gives a practical way of managing solutions.

The workflow and application was put in use at the University of Stavanger in a master-level course. From this course the lab project was fully managed through the Autograder project. Students worked on their assignments through GitHub and got feedback on their code through the continuous integration service. The teaching staff could monitor student progress and had better data to base their grading on.

During the test period at the University of Stavanger the students and teaching staff was interviewed and asked about how the new workflow and application affected the lab work. Students got asked how their learning was affected by Autograder. Teachers was asked how Autograder had an impact on their view on students needs and the grading process. The lab project was also finished with an anonymous survey among the students.

From the responses given from the students it was revealed the students benefitted from the feedback by being able to faster finding areas where they needed improve their solution. There was also an agreement among the students they benefited positively from getting more experience from the workflow used in the industry. Teacher reported from interviews they benefitted from Autograder through easier management of student solutions, relieved from the burden of manually correcting the assignments, could easier find pain points in the assignments and obtained more data for the grading process. The potential for improvements for both student learning and data for grading process was also reflected in the grades given while using Autograder.

The successful use at the University of Stavanger shows good potential from using systems like Autograder. Both students and teaching staff both benefited positively from this project, and made the students more independent from the teaching staff. With further development the services delivered from Autograder might benefit more for courses on colleges and universities. Such environments as massive open online courses, where students have limited connection to the teaching staff, might also have a great benefit from this.

2 Motivation

Developing, managing, and providing feedback on lab exercises for advanced software systems is demanding in a number of ways. The teaching staff may invest a huge amount of resources in developing and managing such lab exercises. In addition, manually reviewing and grading such lab exercises is also a huge undertaking, and requires a substantial amount resources, especially in courses with many students. This often leads to long delays between student submission of a lab exercise, and when the student gets feedback on their code. Furthermore, this model of manually correcting handins is not rooted in how software is developed in the real world. When students heads out to work in the software industry they do not have teachers telling them if their work is correct or not. The students need to be prepared to make those decisions themselves.

As established above, being able to give the required feedback as early as possible is of significant importance. Student surveys conducted at Augustana College shows that early assignments and fast feedback will give the students a better perspective [9]. Getting the feedback early enough helps them to focus on which part of their assignments need work in order to get a more well defined answer. This claim is supported by a survey conducted at the University of Stavanger as well [8]. Furthermore, teachers capacity is also limited, as they often have other responsibilities than teaching one class, and that limits how much immediate feedback can be given. Getting through all the answers from every student will take time, especially in courses with a lot of students attending.

This may place a significant constraint on how quickly the necessary feedback will reach the student. The longer it takes, the less valuable the feedback will be for the student. Resorting to digital learning and software to do some or most of the work going through student handins would give a unique advantage for both students and teachers.

Providing the students with a tool that they can interact with at any given time, to tell them if their work produces the correct answers, will make it possible to give rapid feedback. The students can get feedback while working on the assignment, and with this feedback may have more relevant questions to ask teachers or teaching assistants when necessary. When it comes to assignment feedback, the teachers also have a tool that will make this work easier and saves resources that can be used to guide the students in the right direction.

Another difficulty this will solve is preparing the students for working life, where the software industry is increasingly adopting continuous integration and test-driven development to validate their software. By providing rapid feedback on assignments, this process will be highly coupled with the use of a continuous integration service. The importance of continuous integration in industry has been highlighted in Morten Mossige's work on this topic [11]. Thus silently training the students to use such a system holds great potential for the industry and preparing the students for how the industry operates, or should operate.

With these ideas in mind we propose an automatic feedback system for lab assignments that can be verified automatically. The primary target will be computer science courses, where programming assignments are widely used. By specifying test cases that can validate the submissions of each student or group of students for the desired behavior of the submitted code. This system would also need to provide the students a way to view their test results at any given time and place. Students should be able to complete their assignments both on and off campus. When students commit a code change, this would need to be picked up the system in order to generate feedback to the students instantly.

3 Background

This section will explain some of the techniques and work processes behind the Autograder project. The Autograder project build upon well known tools, techniques and processes to get the students on the right track. The background for chosen work process in the Autograder project is discussed below.

3.1 Industrial Workflows and Tools

To better prepare the students for working life in the software industry, some informal talks with the commercial sector in Stavanger has been initiated, specifically with developers from Bouvet and Capgemini. The topic of the conversations was how they work on a day-to-day basis and what kind of tools are important, both on an individual basis and when working together as a team. The talks uncovered that there is several problematic areas where students would benefit from better practice before beginning their careers in the software industry. In their day-to-day work, these companies practice pair-programming, code-reviews and test-driven development. These strategies have help them build up robust code bases in their projects, and while practicing said strategies better solutions became possible.

Another issue discussed was the practice of dividing large tasks into smaller, more manageable tasks. Thus, teaching students to maintain a task board and work on smaller tasks one at the time would simplify the workflow and make the overall task appear more manageable. Finally, we also discussed the use of tools such as version control and continuous integration as a convenient way to easily integrate the functionality that the developers commit to version control. With these tools they have found that it becomes much easier to keep track of how far everyone have gotten with their work, and how much of this has been correctly implemented.

3.2 Version Control

Version control is a system that allows you to track file changes over time. Version control has become a valuable tool for many software projects, and especially when many people work on the same project [5]. Examples of frequently used version control systems for software projects are Git, SVN and Mercurial [6].

Version controlling is also highly used in the industry today, which makes it valuable to use in the Autograder project. This system will not only be a good way to get students familiar with using such a tool, but will also be helpful in keeping track of the source code, for both students and teachers.

When designing the automated feedback system the specific version control system chosen was Git. Git is a widely used version control system and is a system most students will encounter after finishing their degree. Git has also many good options when it comes to hosting of repositories. One such hosting service is GitHub. Github provides an online service, with an excellent web interface. Through this interface you can track your repositories and also look into other open source repositories.

GitHub will enable the students to quickly see the value of using version control systems. This is due to their well designed user interface. The user interface they use are focused on core elements in the version control process. It easily show which files are currently tracked, and from these files you can find all previous versions. It is also easy to compare earlier versions. Another great thing with using GitHub is their highly available REST API. This API makes the process of making third party application easy. They also provide OAuth through their API and thus makes it easy to log users in using Github accounts.

3.3 Continuous Integration

A continuous integration (CI) environment include the use of automated building and testing of a shared code repository. Developers use this to frequently integrate their work, possibly many times a day. For each of integration, a build and test process verifies for the developer how well the integration worked. Potential errors that occurs can then be found by tests in this progress. The use of CI provides a way to develop cohesive software more rapidly and reduces integration problems at release time [7].

Normal use of CI involves an integration server. This server waits and listen to eventual updates to the code repository. When an update arrives, the system extracts the code and starts a build process. This also includes running all the tests present in the repository. A build is considered successful when the code compiles and all the tests pass. If any of them fail you will have an unsuccessful integration, and you need to resolve the problem that make one or more of the tests fail. The integration process can also measure how much of the code is covered by the current test and fail an integration on how well the code is covered. When a successful build is achieved, the integration server can also start a deployment of the new integration. This is called continuous delivery.

Today there are many CI tools that can be set up to track a software stack. Many is developed to work together with a specific set of version control systems, and some programming languages even has this built in to their development tools. Services with such goals are systems like Travis CI [3], which is an online CI tightly linked to GitHub repositories, and Jenkins [1], which is a CI developed to support a lot of different build jobs and can run on self-hosted infrastructure.

The ideas behind CI is a valuable concept for this project. To be able to provide the students with feedback quickly, the continuous integration can automatically start integration of work done by the students. When this is linked with a version control system, any edits to the code repository could trigger a new build and analysing how well the integration went, this can be used to tell the student how well that particular integration went. Continuous integration is also valuable and widely used in the industry. Introducing the students to such a tool will give them practice in using these tools when starting work for the software industry.

3.4 Test-Driven Development

Test-driven development or TDD is a way to develop software where you make implementations in small iterations at a time by defining the desired functionality first. When starting a new function or feature for your programming stack, you start with defining what this feature will do. When this is done a test case is built. These test cases are built to check for the desired functionality and will fail at first. Then the functionality is developed and tested against the test case. This is then repeated until the tests passes.

This process is often affiliated with extreme programming and similar agile development, where it is a core practice [12]. No matter how the programming process is, the test-driven development can be adopted. However it might not be in a central part of the programming process. Using TDD can be a valuable tool controlling the condition of a software stack, and it can limit having a long line of bug trails [4]. If the tests are carefully crafted, the main problem, which introduced a bug, could be found at an early stage. TDD is used to provide the continuous integration process with test cases to check the integration validity.

This practice can be used in the Autograder project to test if the code implemented by the students give the desired effect. Teachers write test cases to go through code implemented by the students, defined by a code template. If the students makes functionality not tracked by the teachers test case, they should write their own tests to meet the required code coverage. This makes it easy to test what the students write and also if the students wants to write some of the code outside the template, it allows them to do so while practicing use of TDD.

4 Hypothesis

As described in the motivation section, there is a slow feedback problem in today's teaching of software development. Where the students need to wait several days or even weeks before getting feedback whether their solution is a good one or not. To tackle this slow feedback problem, we propose the following hypothesis: *by applying software engineering practices from the software industry, the learning process can be improved through rapid feedback.*

By using a continuous integration service to automatically evaluate students work, rapid feedback can be delivered to the students. Using test-driven development to test for wanted behavior and implementations can make it possible see in detail what each student has been able to complete correctly and what remains.

With faster testing of students code and the possibility to give students feedback through these tests, the theory is that students can more quickly identify and close their gap in knowledge. The rationale for this idea is that when the students are working on their assignments, getting feedback there-and-then makes it easier to understand the feedback. Moreover, teachers can also identify pain points in the assignments that can be used to shift the focus in lectures to these pain points, or to improve the assignment description. While students work on their assignment, they often need to know where they stand in order to know if their solution need more work or not. This is only possible with some kind of feedback. Students can of course create their own test to see if it fits within their understanding of the course material. However if their understanding is incorrect or incomplete, they cannot detect this on their own. This feedback has to come from a member of the teaching staff, and today's work process forces them to either wait for an answer over email or wait until there is a offical help session at the lab facilities, typically once a week. While waiting for help the students might be left with a lot of unused time, time they could use on learning the specifics of the course material. With rapid feedback given through continuous integration and test-driven development, it is reasonable to say students will be left with more time to actually work on aspects of the course where it is needed. With a fast way of verifying if they have understood something correct or not, lets them divert their attention to where it is needed, instead of waiting for the teaching staff telling them so.

This rapid feedback gives the teaching staff a huge advantage too, as they do not need to go through all of the student submissions to give needed feedback and answering the same questions multiple times. In the automatic feedback given through testing students code automatically answers the students right away, when they needed this feedback. Having the students code automatically evaluated gives good feedback to the teaching staff as well. The teaching staff can see over the test results from all the students and find common problems the students make. With this information, teachers time can be spent on those problem areas, thus diverting the teaching over on areas where the students need it the most.

In the grading process, the teachers have another advantage. All the students have handed in the assignments and put them through the tests, and with this information, the teaching staff can make a more informed decision when grading. All of the students are put up against the same set of tests and are thereby judged more fairly, assuming that they don't cheat. Also having access to the students code through GitHub, allows the teaching staff to view the code and can possibly identify which students have a better solution compared to other solutions. Another way of easing the grading process.

Students are not only at universities and colleges to learn, it is also a place to prepare them for a working environment. This means the universities and colleges need to train their students in the methods and practices used in the software industry. After examining how the software industry works, it becomes apparent that it is not only the pure programming part that is important. The ability to use the tools and work processes used in the industry is also important.

In the industry, employees need to be able to work independently and as a team. They also need to have knowledge about how they work together on larger project and to validate the correctness of the programs that they develop. Setting the Autograder to an environment similar to what they will face after graduation, has theoretically a huge potential for preparing the students for the industry. Having the working environment and handin procedure through a version control system, like git, providing the feedback through a continuous integration process and validating the solution through tests, gives the students the necessary training needed to understand the work process.

The Autograder project also wants to stimulate more students to collaborate with each other to improve their understanding and simplify the learning process of the curriculum. When the students work on their assignments in an online environment, it is also natural to ask questions and get help online. There is also a huge potential from students learning from each other. Students often work in groups and might be fragmented. The problem that needs to be solved is how to make the students learn from each other across different groups in a good way. A way to make the students ask each other questions and stimulate them to answer each others questions is to reward students for this. The students can ask their questions openly to the entire class in order to start a discussion about the topic, and any student can contribute to this discussion. Giving the students rewards on each step in this process can possibly motivate the students to ask more questions and thus learn more from the other students.

5 Autograder Prototype

As established in the hypothesis, the setting of an automated feedback system, Autograder, needs to be closely tied to the industrial work method. Because of this, the autograder application was built to use git and a online git repository management system called GitHub. Git would give the students training in the use of a version control systems, and also serve as a storage system for their assignments. The online nature of GitHub and the ability to control organization, repositories and members through their REST API makes them a natural choice. GitHub is also chosen because of their user interface. Files uploaded are the main focus within their user interface and from the file view it is also easy to find earlier versions, version history and comparison of versions. Github also provides supplementary systems to the version control system, such as issue tracker, statistics, and contribution systems.

The hypothesis also states that students need to have a continuously delivery of feedback on their assignments to be able to maximize their time while working on assignments. To deliver this service a custom continuous integration service has been built into the Autograder prototype. Whenever a student solution to an assignment is uploaded, the Autograder will build the solution. Autograder's continuous integration service will be notified from GitHub when an upload of a new solution has occurred and from this information the service can immediately start a build. This build will download the newly submitted student solutions and teacher written test cases, and merge them for

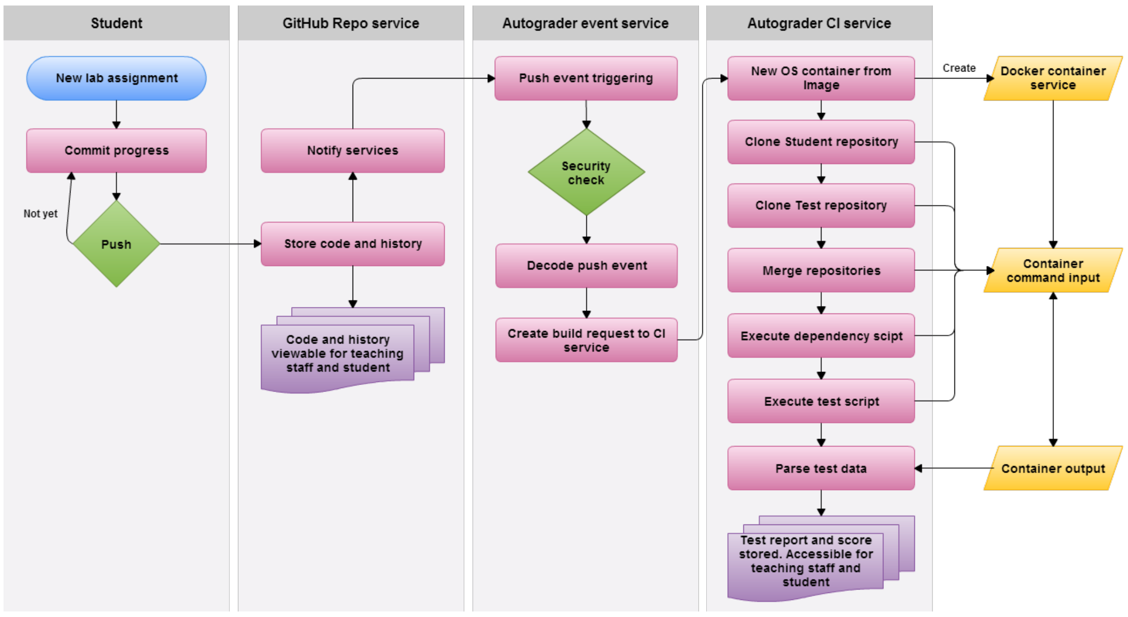


FIGURE 5.1: Flow chart showing the steps between students getting a new assignment to finished test report.

testing in a shared work space. For security reasons this execution takes places in a virtual machine environment called Docker, and at the end of execution the finished output is parsed for presentation to students and teaching staff. The build process is described in more detailed in the continuous integration chapter.

The results and detailed build logs from the test process are available for students and teaching staff from a web service in the Autograder prototype. This lets the students and teaching staff access their courses from anywhere in the world at any time. The results are presented to the students on a page called student panel and will be personalized for the students for each course. This gives the students a good overview and status of their assignments. The teachers are also able to access the students results in a web page called teachers panel, which allows the teaching staff to manage their course and get a quick summary of all the students progress. From this summary page the teaching staff can easily gain access the more detailed information about each student, by accessing the build log. The teaching staff can also fully manage the course from this page, with the settings page and which students and groups are allowed in the course. From the web service the teaching staff can also manage users (students and teaching staff) and repositories on GitHub. This is an automated process. More detailed information about the automated GitHub management can be read in the chapters user management, course management and GitHub communication, and a more detailed description of the web service can be found in the web service chapter.

5.1 Installation

The Autograder prototype is a open source project on GitHub. The source code is located under the name hfurubotten/autograder on GitHub, and have to be cloned from this location in order to install it. The Autograder prototype is written in Go, and rely on a few third-party libraries. These third-party libraries include go-github, goauth2, go-dockerclient, diskv and github-gamification, and need to be cloned as well before compiling the Autograder prototype. Simplest way to streamline this process is to use a go get command on the Autograder repository on GitHub. This will download all the necessary dependencies. When all needed resources is cloned, the prototype can be compiled. These are the commands needed to download and install Autograder, assuming you have installed the Go programming language.

```
# Clone source code
go get github.com/hfurubotten/autograder
# Compile code
go install github.com/hfurubotten/autograder
```

This will produce an executable file called autograder, in the $GOPATH/bin folder. When starting the Autograder for the first time, we need to provide a number of parameters. These include:

- Application codes obtained from GitHub.
- The domain name that the application is running on.
- The GitHub username that should be used as the primary administrator for this instance of Autograder.

The GitHub application codes can be obtained by creating a new application on GitHub's web services, and will contain two different codes. How to generate the application codes from GitHub can be read about in appendix A. The first code that we need to obtain from GitHub is the application identifier, and the second is a secret code for the application. This secret code can be viewed as a password for the application and must be kept strictly confidential.

The Autograder application also needs to know the domain name it is running under. This is necessary to allow GitHub to redirect back to appropriate location (the specified domain name), when using the OAuth protocol.

The primary administrator of the Autograder application has the ability to give other users to teacher and administrator privileges when they sign in to the Autograder application. The user privileges and how the users interacts with the Autograder application are explained in the user management chapter.

```
# Prints instructions
$GOPATH/bin/autograder -help # prints instructions
# First start up
$GOPATH/bin/autograder -clientid=0987654321abcdef
  -secret=abcdef0123456789fedcba
  -domain=http://autograder-app.com
  -admin=hfurubotten
# Normal start up
$GOPATH/bin/autograder
```

5.2 GitHub Communication

All management of the repositories and users in the organization on GitHub are done through GitHub's REST API. This API lets the application perform actions on behalf of a user and lets the Autograder application create or change repositories, teams, and users within the organization. When requests using the API are executed an access token from the teacher are used to authenticate the request. This request can then, when accepted, perform needed actions in the Autograder application.

In the Autograder prototype GitHub's REST API is accessed by using a third-party extension created by Google, under the name go-github. This implementation makes the requests callable through simple function calls, simplifying the process of managing the organization on GitHub. This extension makes it possible to connect to GitHub's different services, such as the repository service, issue service, organization service, and user service. These services makes it possible to create, edit, or remove repositories and users within selected organizations. It also enables the Autograder to add, read, alter or remove the content within the repositories.

5.3 User Management

When users are accessing the Autograder application they have to authenticate themselves with their GitHub account. This authentication is done through the OAuth protocol. This protocol works by first send the user to GitHub's web service with an identifier specific for the Autograder application. This identifier lets GitHub's services

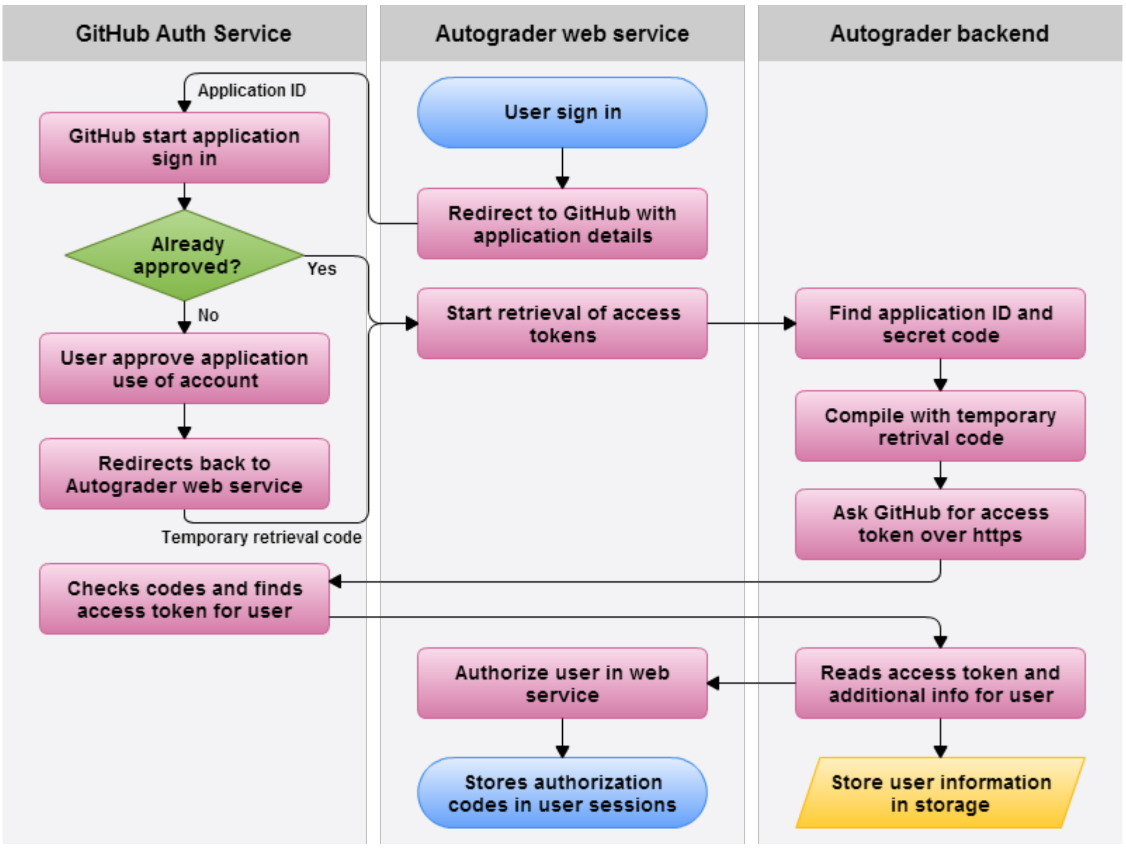

GitHub Auth Service
Autograder web service
Autograder backend
Application ID
GitHub start application sign in
User sign in
Redirect to GitHub with application details
Already approved?
Yes
No
Start retrieval of access tokens
Find application ID and secret code
User approve application use of account
Compile with temporary retrival code
Redirects back to Autograder web service
Ask GitHub for access token over https
Temporary retrieval code
Checks codes and finds access token for user
Authorize user in web service
Reads access token and additional info for user
Stores authorization codes in user sessions
Store user information in storage

Figure 5.2: Autograders use of the OAuth protocol.

know which application is requesting to have a user authenticated. The user must then approve that Autograder can access their account. This approval process only needs to be done the first time, when the user is signing up for the Autograder application. For students the approval restricts the Autograder application to only look up the GitHub profile information. Teachers, on the other hand, need to approve access to organizations and repositories as well. When the user have approved the use of their GitHub account, they will be directed back to Autograder's web service. In the backend of the Autograder application an exchange for an users access token to GitHub is started. This is done with a user identifier acquired in the redirect back to Autograder's web service and a secret application token from the Autograder application. When this is approved by GitHub's systems, the access token is received from GitHub. '

This access token is then used to request for the profile information from the user's GitHub account. From the profile information Autograder will try to establish username, real name, email address, profile avatar and profile link from GitHub's information, and store this as user information. If any needed information, e.g. real name and email address is missing, this will be manually collected through the profile page in the Autograder web service. When this is collected the access token gets encrypted and stored with a link to the users GitHub username. The user information will then be stored separately, and can be retrieved using the access token returned from GitHub while signing in to Autograder. Also when the user signs in, the username is stored in a session in order to recognize the user between page requests in the web service.

A user can have three different access levels or privileges within the Autograder application. This is student access, which all user have as standard, teacher access and administrator access. The students access gets the user access to signing up for courses already created in Autograder. A user can only be promoted to the two other access levels by a user which already holds administrator access. The administrator have access to a page named admin panel. In this admin panel the administrator can upgrade another user with teacher access or administrator access, or both access levels. With teacher level the user can create new courses within Autograder.

The user information also stores which courses the user is attending, any groups the user is member of, if the user is a teaching assistant and which courses the user is an assistant, and which courses the user is a primary teacher for, if the user have teacher access.

5.4 Course Management

To meet the goal for an industry work setting for the students the Autograder application has been fully integrated with GitHub. This integration mean the application can manage all repositories and members within an organization on GitHub. Organizations on GitHub can be compared with user on Github where a collection of repositories are registered under a specific name. However an organization can have a collection of members under its name, as a user only have itself as member. This organization is the basis of a course in the autograder application. When a user with teacher access rights registers a new course through the web service an organization on github will be linked to this course.

When teachers creates a new course, their GitHub access token will be used to find all the organizations the teacher have administrator rights to GitHub, through GitHub's REST API. From this information the teacher selects the organization which will be used in the course and links them together. In this linking process the new course will be registered in Autograders course register. In the registration process the organization name and the access token for the teacher will be stored with this course. The organization name will be used as a reference to which organization the autograder application will perform action against. The access token for the teacher is also stored as an administrator token and will be used to perform actions on behalf of the user on later stages. The access

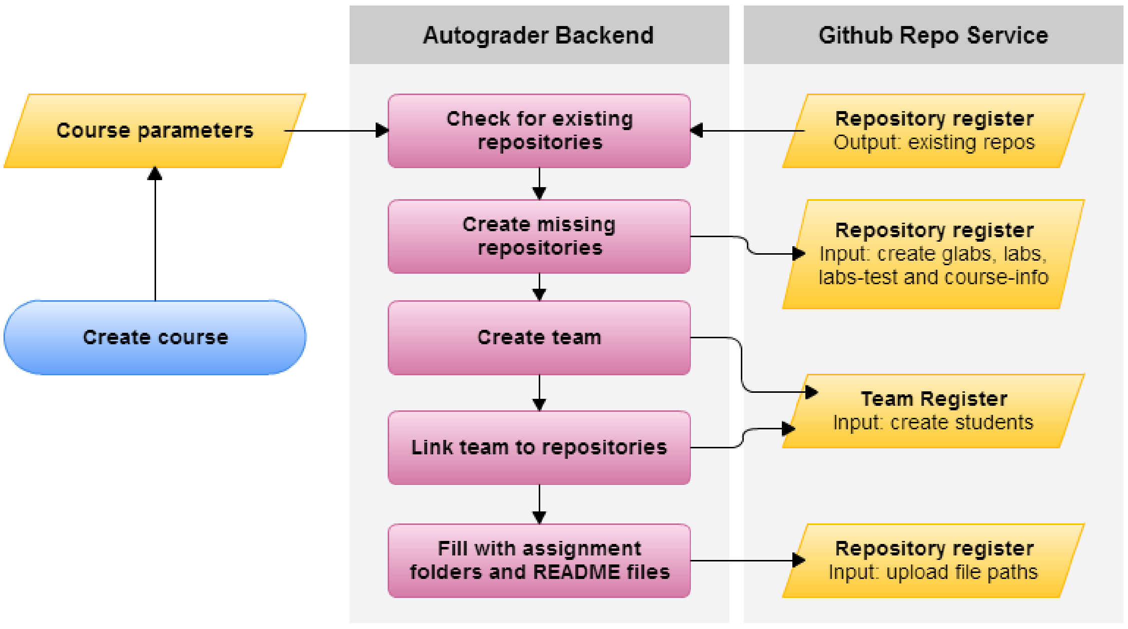

Autograder Backend
Github Repo Service
Course parameters
Create course
Check for existing repositories
Create missing repositories
Create team
Link team to repositories
Fill with assignment folders and README files
Repository register Output: existing repos
Repository register Input: create glabs, labs, labs-test and course-info
Team Register Input: create students
Repository register Input: upload file paths

FIGURE 5.3: Steps for taken internally by Autograder in order to create needed repositories and teams on GitHub.

token are stored to actually be able to perform actions also when the original teacher is offline. GitHub is also restricting the API to be used through a user token and not only the application token.

In the creation process there is also a restriction on GitHub's organization API, where it does not allow the creation of new organization from an application. This limitation means the teacher will have to create the organization themselves on GitHub's own web pages. When the organization is created in GitHub the organization will also be visible in the Autograder application.

In the registration process the Autograder application starts up a process of creating a set of standard repositories and teams. This process creates the repositories with course information, test cases and assignments. These repositories are named course-info, labs-test, labs and glabs, and are the primary repositories used by the students and the teaching staff. Information about the course, any updates to the course and general information to the students are uploaded to the course-info repository. The course-info repository is not required to use, but recommended as this will let the students get all information from a common place. The three other repositories, labs, glabs and labs-test, are necessary to use for the autograder application to perform optimal while building students solutions. labs and glabs are used to distribute the assignments to the students, where glabs are used for group assignments and labs are for individual assignments. The teaching staff upload the assignments and any supplementary information, such as skeleton code, to this repository. Students can then easily clone the data and start

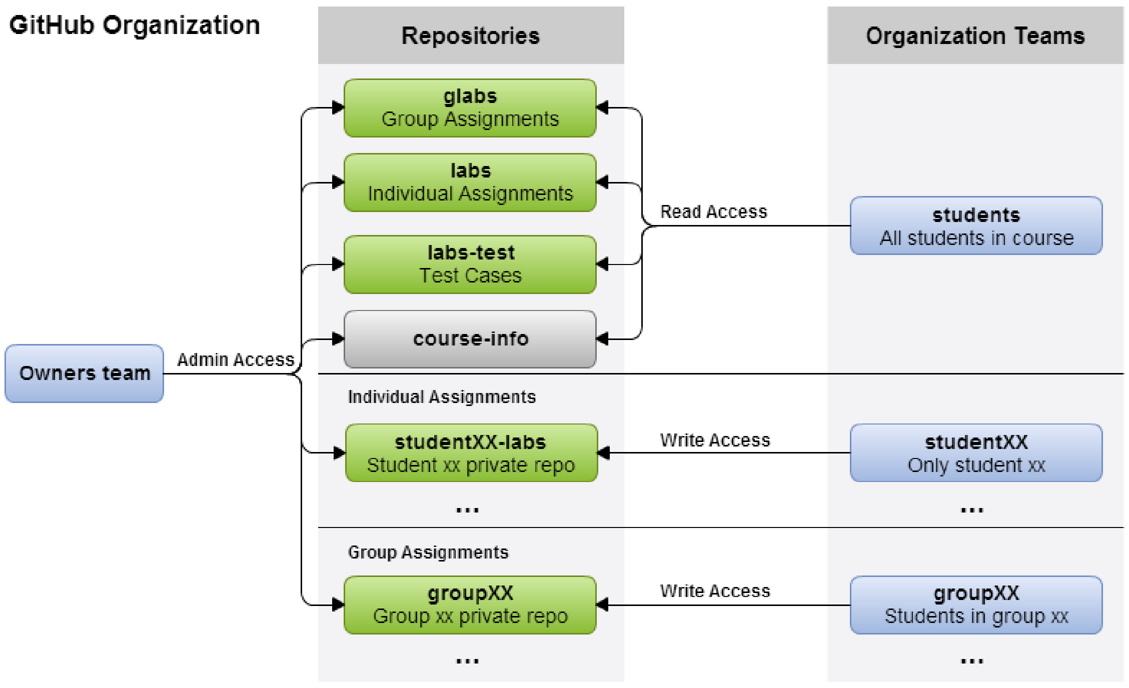


FIGURE 5.4: Relationship between different teams and repositories on GitHub, including access rights.

working on their assignments. The labs-test repository will be the repository which stores test cases for execution on students solutions. This repository will be hidden from the students and can only be access by the teaching staff and the continuous integration service.

The creation process will also create a standard team in the organization, the students team. This team will hold all the students in the course and lets the students have read access to the course info and labs repositories. New students signing up for the course will be added to this team automatically. When students are added to this team GitHubs systems sends out an invitation to join the organization. From this invitation the user accepts to be a member of the organization on GitHub.

After the course has been created in and the GitHub organization has been set up, the students and teaching assistants can start signing up for the course in the web service. When students signs up for a new course the Autograder application will add the student to the attending list on the course and the list of courses for said user. This signup process will also add the student to the students team in the GitHub organization. When this is done the student have to accept the invitation sent from GitHub. When this invitation is accepted, the teaching staff gets the user up on a list of students waiting for access to the course. The student have to wait on a teacher to approve the student for the course. The approval process of students is a control mechanism for the teachers. It will let the teaching staff control exactly who gets access to the course and who gets a repository of their own.

When a member of the teaching staff accepts a new student. This student need their own repository on GitHub, a place where the student can upload their solutions. The moment the students gets approved by the teaching staff, Autograder will connect to GitHub and create a repository with the student's username and the suffix "-labs", and a corresponding team with write access to the said repository. This will let the student have their own repository for submitting solutions.

Some courses practice group assignments as well, and to enable this a solution for creating multi user repositories is also implemented in the Autograder prototype. When the students submit a group request through the web service, the teaching staff can approve or discard the group, and when approved the application will create a repository for the group of students. The group repository will be created with the prefix "group" and end with the group identifier at the end. After creating the repository the application will also create a team with the same name on GitHub. This team will be filled with all the members of the group and given write access to said repository.

A course always begins with one teacher as member when it is created. However some courses on Universities and colleges have multiple teacher and can also have multiple teaching assistants. The prototype is constructed to handle these scenarios as well. The user who created the course is the primary teacher, but a set of other users can be upgraded to teaching assistants in the course. For a user to be upgraded to a teaching assistant the user need to first sign up as a student. The primary teacher on the course then have the option to choose a student to be upgraded. When this student, or user, gets upgraded, this user will be added to a special team on GitHub. This team is called the owners team, and will have admin access to the whole organization. Being a member of the owner team will give the user the same access to the organization and course in the autograder application as any teacher.

5.5 Continuous Integration Service

In the Autograder prototype there has been developed a custom continuous integration tool. This tool is the core which the continuous integration service is relying on. When a student upload their progress to github, the web service gets a notification about this and will start a build request to the continuous integration service.

The event notification from GitHub gets processed in the web service and will find in which repository, and in which course, a student has uploaded a new version of their

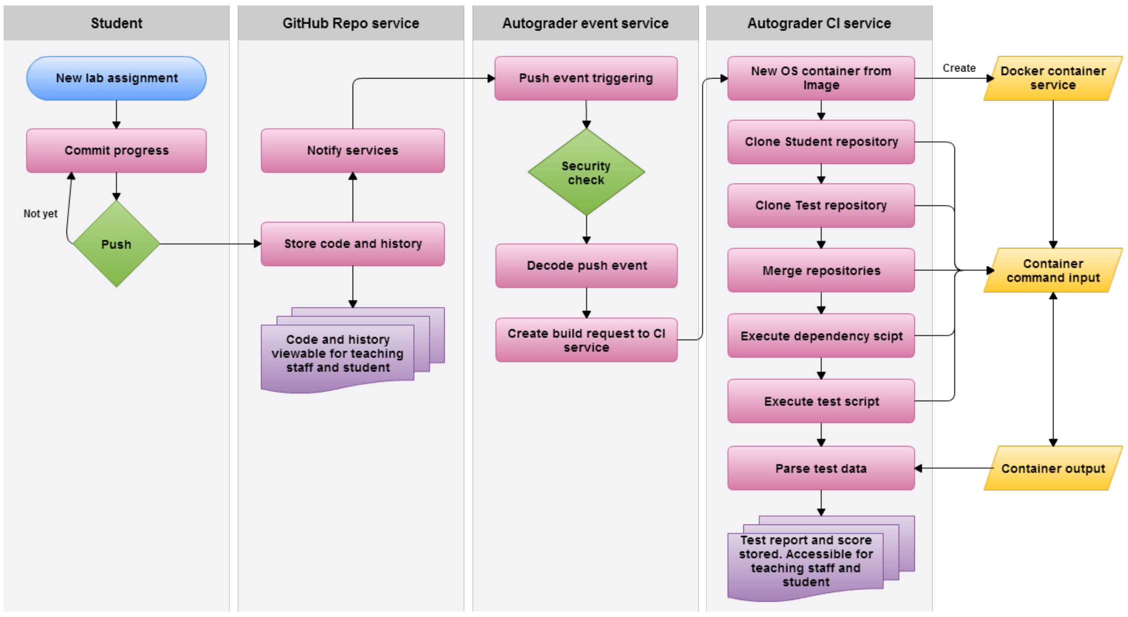
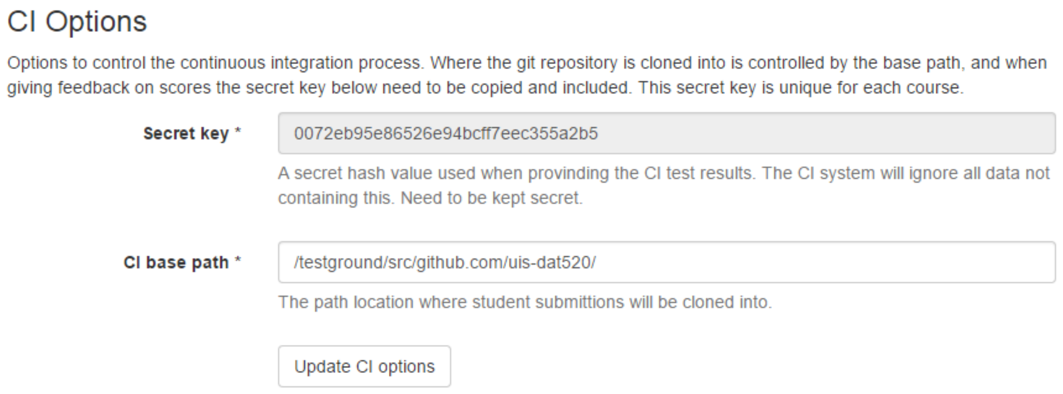


FIGURE 5.5: Representation of how the continuous integration service receives, process and deliver build requests.


CI Options

Options to control the continuous integration process. Where the git repository is cloned into is controlled by the base path, and when giving feedback on scores the secret key below need to be copied and included. This secret key is unique for each course.

Secret key * 0072eb95e86526e94bcff7eec355a2b5

A secret hash value used when provinding the CI test results. The CI system will ignore all data not containing this. Need to be kept secret.

CI base path * /testground/src/github.com/uis-dat520/

The path location where student submittions will be cloned into.

FIGURE 5.6: Settings for the continuous integration service in the web service.

assignment. The usernames and course data are checked against stored data, to confirm the legitimacy of the event, and can then start a build request. This build request tells the continuous integration service which repository to use and how to build the student solution.

When the continuous integration service get the build request, it first starts up a new docker container, where it can run the solution in a clean environment. A docker container is a simulated linux operating system which is started from a common virtual image. This image contains the needed compilers and programs used to compile programming code from different languages. This docker container is used each time a new command will be called for this particular build, and each build has its own container.

After starting up a new container, to run the build in, the base path for where the build should be located in the file system is created. This is an option the teaching staff can change in the course setting in the web service. The path settings given from the teaching staff enables them to decide exactly where the repositories should be cloned and where the build should be executed from in the file system.

The next operation executed from the build process are to clone two of the repositories on GitHub. First it clones and downloads the student repository, or the group repository, in the predetermined path, and then clones the labs-test repository which contains the teacher written test cases. These two repositories then gets merged by copying the content of the labs-test repository over to the student repository. This operation will let the test cases be located in the same file location as the files in the student solution, but this also means that files with the same name in the both repositories will conflict and result in students version is overwritten. Files in the teacher restricted repository labs-test are prioritized in order to prevent students overwriting any test cases.

When all the files are present in the same file structure and ready to start testing. The continuous integration service is built to have custom build and test procedures for each assignment. This is done through a standard call to script files, written by the teachers. For each assignment there is a linux shell script which need to be implemented by the teaching staff to execute their test cases. Script files required are one shared script file for all the assignments and one for each lab assignment. The first script file is named dependencies.sh and is ment to run any commands which would be common for all the assignments. This file can also be empty if not needed. The second file is the test script itself. This script file will be in a folder named the same as the assignment, and must contain the commands to start up the testing of students code. These two script files gets called by the build process in order to get the test results from the students solutions, with the test file as the last runned script file.

Throughout the build process, the continuous integration service records the standard output from the commands executed within docker. After the test script is done executing the output gets analysed. The analysis finds out how many tests failed and how many passed. The analysis also looks for any score objects, with more detailed test data, in this output. If any score objects are present, and have valid course hash ID, it will use these to calculate a score for the student or group, and if no score objects are found the basic test data will be used in the score calculation. How the score objects work and are calculated can be found under the chapter score reporting.

After analysing the data from the standard output, this data is also kept for reviewing of the students and teaching staff. The score objects are only for giving back information back to the autograder application about internal score progress and are removed from the test data. However there can be much information in the test data itself for students and teaching staff to better understand why a test case failed or did not build correctly. The standard output data gets sanitized for any potential sensitive information, and then stored in a build report. This build report is the basis of what later on will be presented to the students in the web service. In the build report, together with the output data, the score objects and calculated score are stored. This report are then written to disk under the specific student or group, to be presented for teaching staff and students when you visits the web service.

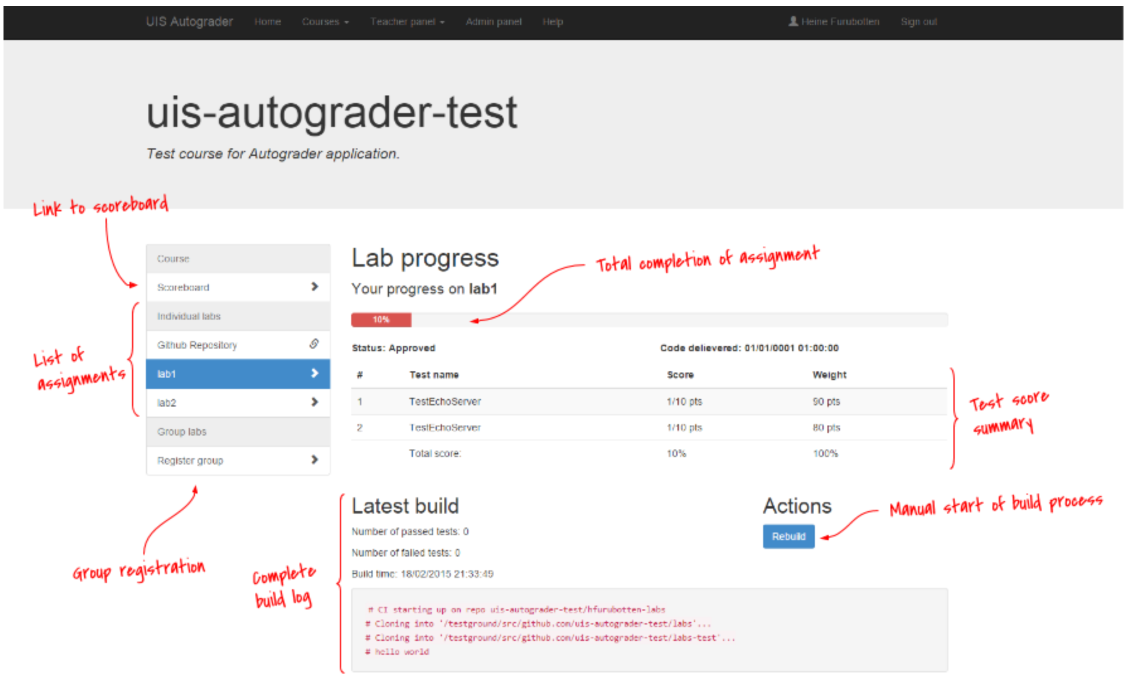
5.6 Web Service

Students and teaching staff need to access the test data generated from the continuous integration service and other services from the autograder application in a simple way. This also needs to be accessible for them from anywhere at any time. The best way to deliver these specifications is to have a web service. These web services have been implemented directly into the prototype and works closely with the background processes.

The web service is designed with a simplicity in mind and have been created with help of the standard theme from the twitter bootstrap library. This does also make the web pages mobile friendly, where the pages will automatically scale for any screen size. Students and teaching staff can then access results from any device they which. The web pages deliver for the most part only a shell to present the content on the pages. When the user access a page they download a html file which needs to be filled with content, and this is controlled by javascripts in the background of the page. Raw data is retrieved from the web service and then presented in the design on the web pages. This allows the same data to be reused on multiple web pages, without the need to render this into the page on each page request, and the data can be refreshed without having a hard refresh of the web page itself.

This chapter explains in more detail each of the available pageviews in the web service.

5.6.1 Students Panel



uis-autograder-test

Test course for Autograder application.

Lab progress

Your progress on **lab1**

10%

Status: Approved **Code delievered: 01/01/0001 01:00:00**

#	Test name	Score	Weight
1	TestEchoServer	1/10 pts	90 pts
2	TestEchoServer	1/10 pts	80 pts
	Total score:	10%	100%

Latest build

Number of passed tests: 0

Number of failed tests: 0

Build time: 18/02/2015 21:33:49

```
# CI starting up on repo uis-autograder-test/hfurubotten-labs
# Cloning into '/testground/src/github.com/uis-autograder-test/labs'...
# Cloning into '/testground/src/github.com/uis-autograder-test/labs-test'...
# hello world
```

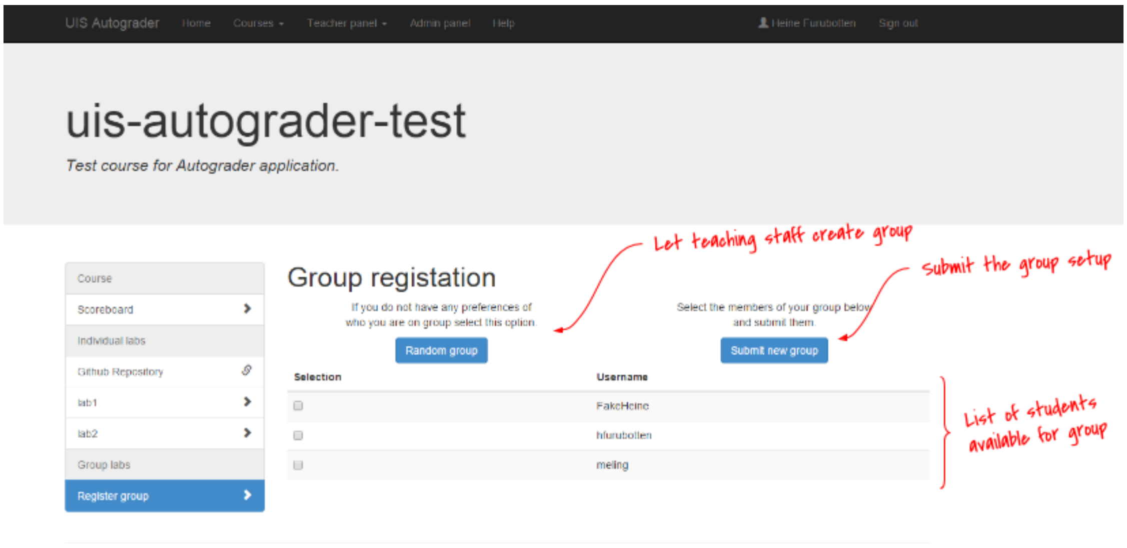
Actions

Test scores and build requests are available for students and the teaching staff through the web service. Each student have their own personal page for each course they attends. This page gives the student a list of all the lab assignments and using this list the students can look up latest build for each assignment. The presentation of test results are generated from a build report created in the continuous integration service. From a build report the page generates three different sections with information to the student, a total assignment score and status, a individual test score summary, and the complete build log.

Test results from each assignment is presented first of all with a process bar, which shows a quick summary of the overall progress of the assignment. Under the process bar the student can find the status of the lab assignment, a status which is shown either as an active assignment or as approved by the teaching staff, and when the last time the build was triggered from the student uploading their solution to GitHub.

Together with the overall process information, a summary of the individual test scores is shown. This is a table of all the data gotten from score objects generated in the teacher written test cases. This table will show which test generated the score, how many points the student has gotten within the specific test and how much the test results from the said test weigh on the total result.

Rest of the page follows with the complete build log. The student can see how many failed and passed test, or build failures, have been detected. Also the students can read the pure output from the build and test process. Within this output, the students can look more closely on potential error messages and any feedback generated by the test, in order to improve the solution.



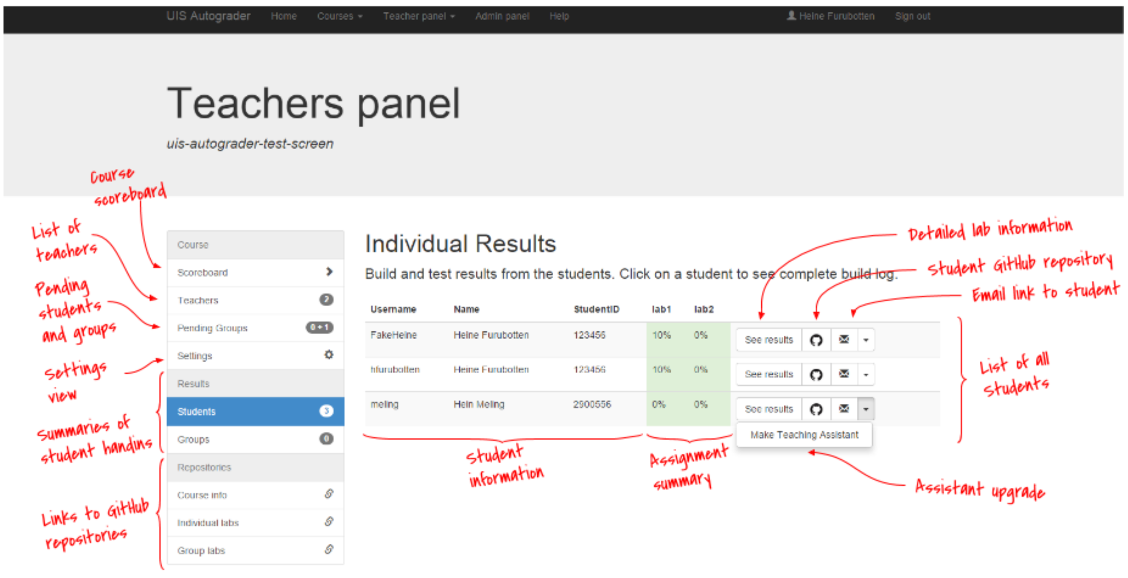
If you do not have any preferences of who you are on group select this option.

Select the members of your group below and submit them.

Selection	Username
☐	FakeHeine
☐	hfurubotten
☐	meling

In the case of group assignments in the lab project, the students also has the possibility to form groups through the web service. The students can form groups in two ways, through selecting the students they wants to work with or telling the teaching staff they want to be put together with any student. When the students have submitted their group selection, the teachers need to confirm that this is a valid group. When the group has been approved for lab work, the students gets access to build their solutions and see results from their shared lab assignments, in the same way as their individual assignments.

5.6.2 Teachers Panel



Individual Results

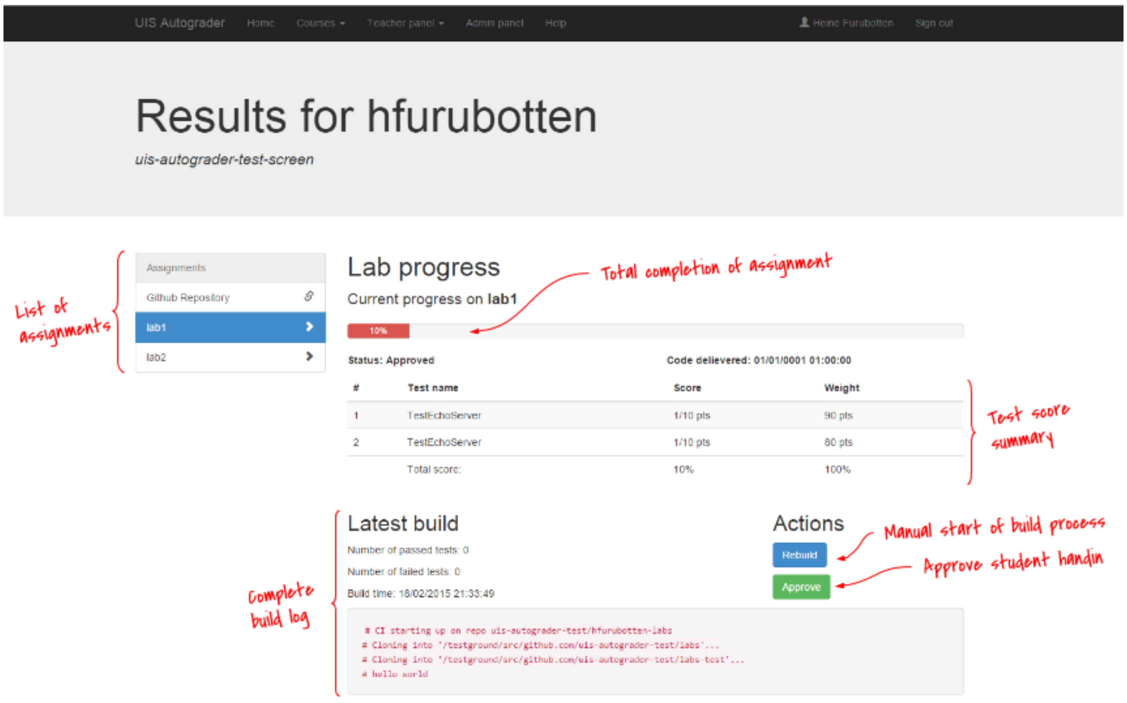
Build and test results from the students. Click on a student to see complete build log.

Username	Name	StudentID	lab1	lab2
FakeHeine	Heine Furubotten	123456	10%	0%
hfurubotten	Heine Furubotten	123456	10%	0%
meling	Hein Meling	2900556	0%	0%

The teaching staff also have a common page for their course, called teachers panel. This page is designed to give the teaching staff full control over their course and see a summary of students results. Teachers panel lets the teaching staff manage the course and the students from one place. From teachers panel the teaching staff can change the setting of a course, manage students and groups, look up more detailed and upgrade users to teaching assistants.

The main view from in teachers panel is the summary of students or groups. There is two summary views which can be viewed. First summary is of all the students in the course, and will always be present on teachers panel, even with or without individual assignments. The students are presented in a table. This table contains student information, the total score on their assignments and a menu for the individual student. Information showed about the student are student ID, name and GitHub username. The results from their assignments are presented with a column for each as percentage completed of their assignment. If the course have group assignments, there will also be a possible to see the same view for all the groups in the course. The student information is then changed out with information about the group, and shows group name and ID and all students whom are members. The last column in the summary table is a menu for actions the teaching staff can access for the student or group. This menu enable them to view the complete build log for the students, send out emails to individual students or groups, see their GitHub repository, remove groups, and upgrade students to teaching assistants. Which of the two views, groups or students, are showing when a teacher is visiting the teachers panel is controlled from the deadlines on the assignment. The view is selected through calculating the nearest deadline of an assignment. The teachers can also switch between the two in the main menu on teachers panel.

When students sign up for the course or the students submits a new group, this has to be approved by the teaching staff. Each time the students sign up for course or a group, they gets added to a list which needs approval. When there is content in this list a new menu item appears to the teaching staff there is a student or group which needs to be approved. When a member of the teaching staff toggles this view they get a list of the students or groups which needs to be approved. They can then either deny or approve the student or group for the course. Upon approval the system will notice the server in the background of this and start the creation of repositories and access policies on GitHub.



Results for hfurubotten

uis-autograder-test-screen

Lab progress

Current progress on **lab1**

Status: Approved

Code delievered: 01/01/0001 01:00:00

#	Test name	Score	Weight
1	TestEchoServer	1/10 pts	90 pts
2	TestEchoServer	1/10 pts	80 pts
	Total score:	10%	100%

Latest build

Number of passed tests: 0

Number of failed tests: 0

Build time: 18/02/2015 21:33:49

Actions

```
# CI starting up on repo uis-autograder-test/hfurubotten-labs
# Cloning into '/testground/src/github.com/uis-autograder-test/labs'...
# Cloning into '/testground/src/github.com/uis-autograder-test/labs-test'...
# hello world
```

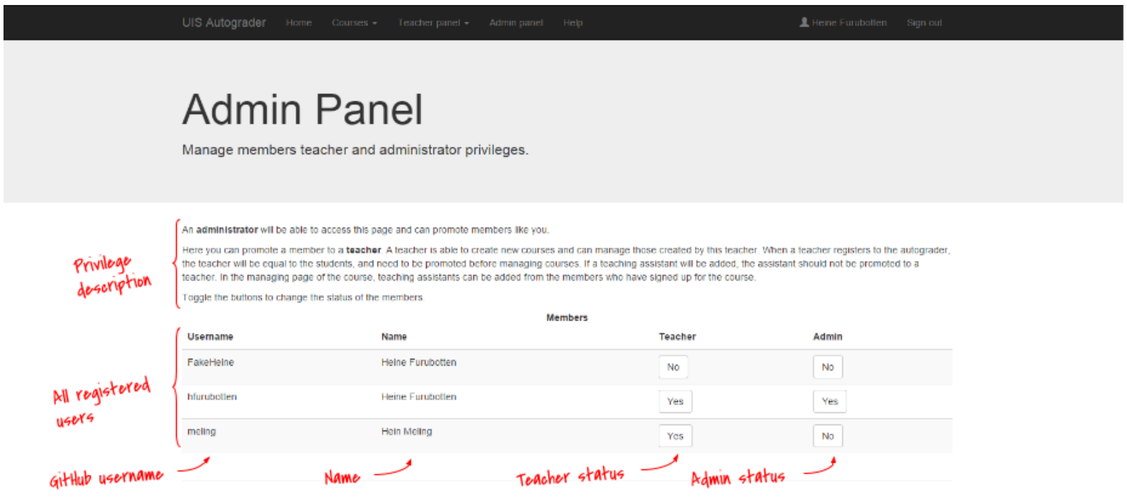
Often it is not only the summary which is important for the teaching staff. The complete build log needs to be reviewed as well. From the list of students or groups the teaching staff can access this information. From the menu on each student they can get access to a page which shows the same complete information which is shown on students panel. Here the teaching staff can see the complete build log, the score summary and the overall progress. All the different assignments are accessible from this page as well, and the teacher can select an assignment to view in the main menu on the page.

To control the course as best as possible, the teachers panel have a settings page. In the settings the teaching staff can change parameters for the course information, the continuous integration service and the assignments. First section in the settings are the standard course options. These options is the control over course descriptions, number of assignments and how to interact with GitHub. The description of the course includes a longer text explaining the course and the name shown within the Autograde application. The number of assignments, both individual and group assignments, control how many assignments the system should look for in the data from GitHub repositories. These setting also controls whether or not the course should make the repositories on GitHub private, on other words make them only viewable for members with access right to the repositories.

The next section is options for controlling the continuous integration service. From the setting in this section the teaching staff can control where in the file system the students repository should be cloned to. This makes it possible for the teachers to adjust needed paths for the building process. The other piece of information in the continuous integration sections is the read only score hash ID. This is the hash value needed to authenticate score objects from test cases. The hash ID need to be kept confidential in order to not let the students create fake scores.

The last two section is the settings view is assignment options. These two sections collect deadline dates and the folder name it goes under within the student's repository. The folder names are collected for use in the continuous integration service. When starting up a build, the continuous integration service needs to know where the different script files are located and are done through this information. The assignment deadlines are collected for knowing which assignment is current active one. This information is collected separately for the group assignments and the individual assignments.

5.6.3 Admin Panel



Admin Panel

Manage members teacher and administrator privileges.

An **administrator** will be able to access this page and can promote members like you.

Here you can promote a member to a **teacher**. A teacher is able to create new courses and can manage those created by this teacher. When a teacher registers to the autograder, the teacher will be equal to the students, and need to be promoted before managing courses. If a teaching assistant will be added, the assistant should not be promoted to a teacher. In the managing page of the course, teaching assistants can be added from the members who have signed up for the course.

Toggle the buttons to change the status of the members.

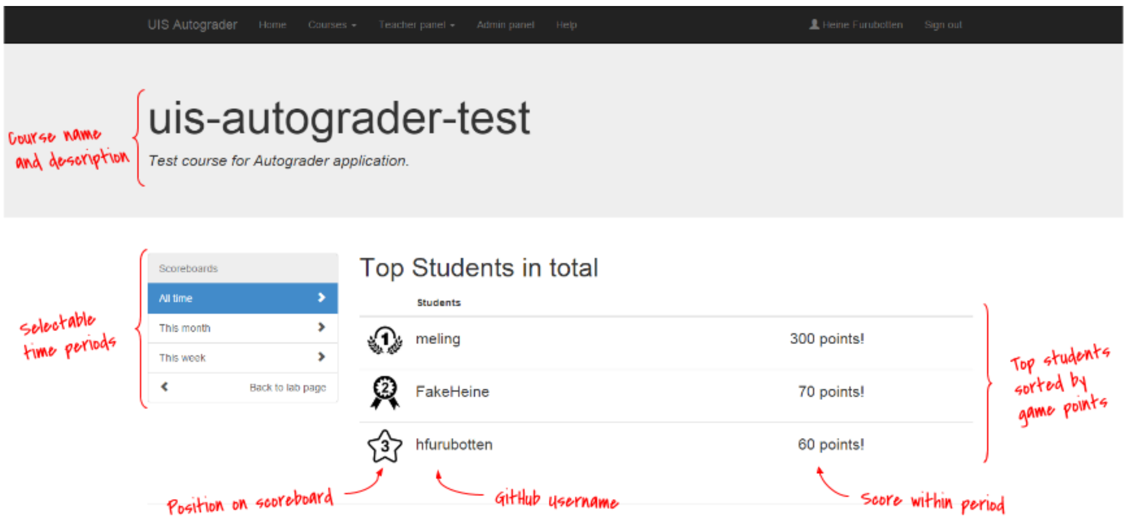
Members

Username	Name	Teacher	Admin
FakeHeine	Heine Furubotten	No	No
hfurubotten	Heine Furubotten	Yes	Yes
meling	Hein Meling	Yes	No

Administrators are the users who have access to manage all the other user in the Autograder application. With an administrator status the user will have access to a web page named admin panel. From this page the user can see a list of all user registered in the autograder application, and basic information about the user, eg. the real name of the user. The admin panel allows the administrators manage the privileges of each individual user. This allows the administrators to promote or revoke administrator or teacher privileges from a user. Teacher privilege will let the user create new courses within the autograder application, and the administrator privilege lets the administrators share the workload of managing the users.

All users shown in the admin panel will have listed which status the user currently have. The administrators will have the option to change any of the two higher access levels. This is done through a button for each of the options. Changing the status is as simple as pushing the button and then the web page will send a request in the background about the user getting new privileges. The updated user will be able to see the new options on their own pages the moment after the administrator have upgraded the privileges.

5.6.4 Scoreboard



uis-autograder-test

Test course for Autograder application.

Top Students in total

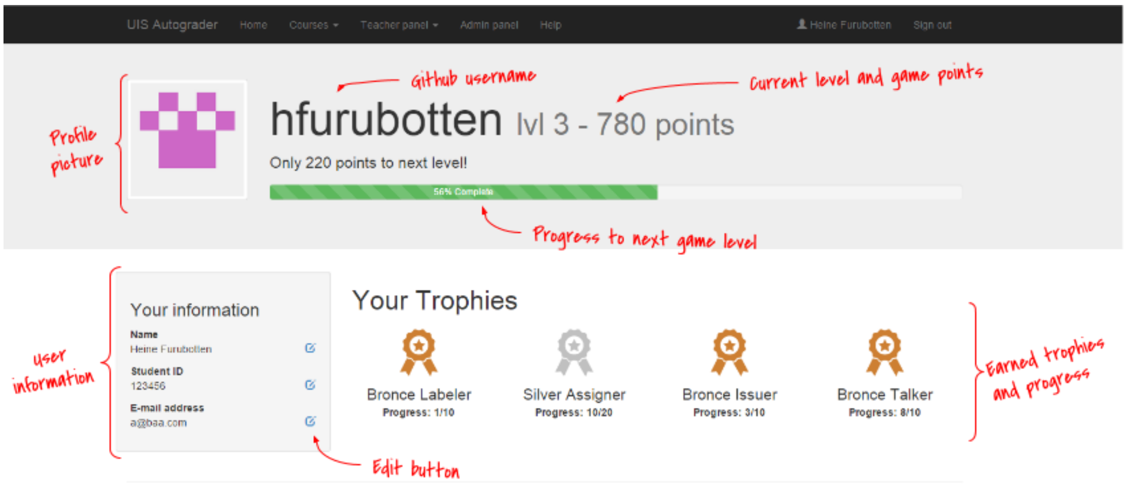
	Students	
1	meling	300 points!
2	FakeHeine	70 points!
3	hfurubotten	60 points!

With the game engine all courses have their own scoreboard accessible from the student panel and the teachers panel. The scoreboard shows the top 9 students generating conversations on GitHub. This page lets the students compete to be the most helpful student in their class, and to get motivated to generate more to keep their place on the list. The scoreboard is accessed by a menu item in on either the teachers panel or the student panel.

When students or teachers access this page, the total score for the course is the standard view. This shows the scores for the students through the total duration of the course. The list is also decorated with different trophies to make the list seem more privileged to be on one of the higher positions on the list. When the list loads a animated fade in effect is implemented. This is created to make a more “wow”-effect from the students, and making it more fun to interact with. The total course score can be changed to view last week or month too. This is done through the page menu. The page then loads the desired score data and pushes them into view for the user on the page.

The data used to present the scoreboard is loaded dynamically when the student pushes any of the menu items. This loading is done in the background with help of javascript, and loads the latest information from the server each time.

5.6.5 Profile Page



hfurubotten lvl 3 - 780 points

Only 220 points to next level!

Your information

Name
Heine Furubotten

Student ID
123456

E-mail address
a@baa.com

Your Trophies

Bronce Labeler
Progress: 1/10

Silver Assigner
Progress: 10/20

Bronce Issuer
Progress: 3/10

Bronce Talker
Progress: 8/10

Each user in the Autograder application have their own profile, which stores information needed by the system to have optimal service for the user. The first time the users log into the application they will automatically be redirected to this page for collecting needed information. Most of the information can be collected from GitHub, but in the cases where only limited information can be obtained the users have to input this manually in the profile page. Needed information in the profile is the user’s real name, student or employer identification and an email address. If any of these pieces are missing the user will always be redirected back to the profile page, and missing information will be highlighted. The user can edit or input their information at any time from the profile page. Each line of information has a edit button and lets the user input new information, and then upload this for saving on the server.

The profile pages does not only show the information added or collected information from GitHub user profiles. The profile page also show the user's game status. In the headline of the profile page you can see the user's avatar on GitHub with current game level and progress towards next level, represented by a process bar. All badged the user has earned will is also viewable in the profile page. More about how the levels and badges works can be read in the game engine chapter.

5.6.6 Help Pages

To offer support for queries from students and teacher a help page is available. These pages are available even if the user is signed in or not. The help section gives explanations about what the autograder project is and how the students can register and sign in to the Autograder application. The help section is also easily extendable. Finding pages to view on the help section is fully automatic and a new page only need to be added to the help folder and the page can be loaded. Provide a new link in the menu and a new html file and the Autograder application will be able to find and provide the page.

5.7 Game Engine

The Autograder project aims to stimulate more student learning from each other and collaboration outside their own their usual circles. In order to get the students properly stimulated a gamification approach was implemented and will give the students points for actions performed on GitHub, mainly in the issue section of their repositories. The gamification was implemented in a game engine in the Autograder prototype. The game engine takes in any action performed by the students on github and process it for point. Points will be awarded to the students and they can climb on a scoreboard.

The prototype will get notifications about student actions from a web hook created on each of the students repositories and the assignment repository. Web hooks is a service from GitHub where an application can get a push notification from GitHub when a certain action has occurred on their services. For said repositores the webhook have been edited to give a notification about any action performed on GitHub.

When the game engine got a push notification from GitHub it sorts out the notifications relevant for the point system. Push events, where students have pushed a new solution to their repository, are forwarded to the continuous integration service for building. Actions relevant for giving points to students will be decoded and the student completing the action will be awarded with points. These points for different actions can be seen in

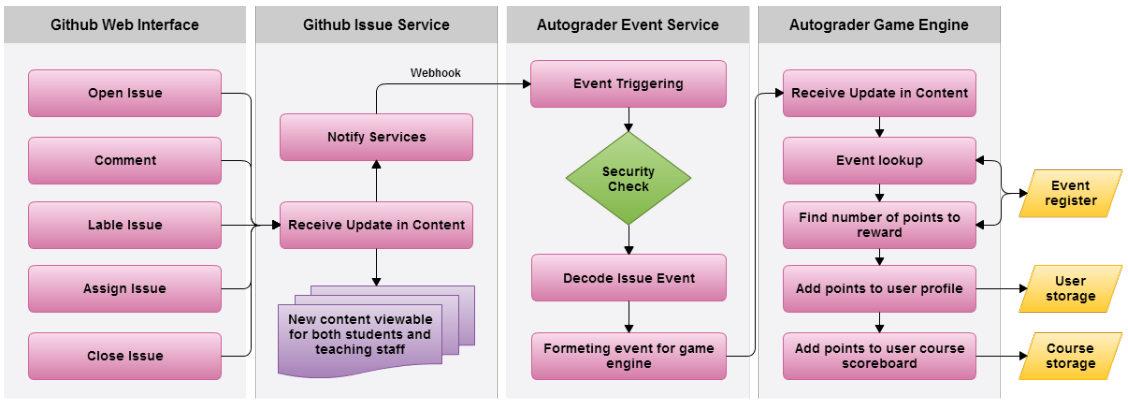


FIGURE 5.7: Flow chart of how the game engine receives action notifications and rewards students for their efforts.

Action	# of points
Comment	50
Open issue	70
Close issue	20
Reopen issue	20
Issue assignment action	10
Issue label action	10

TABLE 5.1: List of points earned from each type action on GitHub.

table 5.1. Here comments, giving actual value back to other students, give most points, and organizing and sorting the issues, for better overview, gives less points. Currently supported actions are within the issue service on GitHub.

When a student gets points this will be stored in the user profile. This process of giving points will also result in giving the students higher and higher game level and badges. These game levels and badges will start off easy and get harder and harder to conquer. The game levels can be viewed on their profile page and current level are represented alongside their name and a process bar shows the progress towards next level. Game levels are purely calculated from the number of points the student have obtained, and the number of points needed for each level can be seen in table 5.2.

Badges on the other end is calculated by number of times a certain action is completed. When an action is completed by a student this counts on the different badges. Current badges accessible in the game engine are talker badge for creating comments, issuer badge for new issues, assigner badge for assigning issues and the labeler badge for labeling issues. These badges will be unlocked one by one from when the student completes one of the said actions. Badges are shown on the profile page with a trophy for each of them, and they can range in value by how many actions are completed on each of them. The values the students can obtain on them are bronze, silver, gold, platinum,

Game level	**# of points needed**
1	0
2	100
3	500
4	1 000
5	2 500
6	6 000
7	10 000
8	14 000
9	20 000

TABLE 5.2: Points needed for each level in game engine.

and onyx, where bronze is the lowest one and onyx is the most valuable. The progress on the badges are also shown underneath the badge themselves.

Points obtained by students will also be registered within the course the action was created. These points will be stored in a list of all the students. The points will be the basis for a scoreboard, where the students can compete for a top position among their peers for collaboration efforts. This scoreboard stores points gained per user overall, for each month and each week, and have a individual scoreboard for each of these periods. The overall scoreboard will count all the points generated by the students and shows a descending list of the nine students currently on the top. The two other scoreboards will reset itself for each new month and week, and start counting points within the new period.

6 Autograder Work Process

This section will explain the normal work process within the Autograder project. How the students will work on their assignments and get rapid feedback and how teachers can write best possible test cases for their courses will be gone through.

6.1 Submitting Test Cases

In order to be able to supply feedback in Autograder's continuous integration service, it relies on teacher written test cases. Through the test cases written by the teaching staff, the Autograder application gets supplied results which tells Autograder how to present progress for the students. When the students upload their solutions to GitHub and the continuous integration service are notified to start a build, it needs a set of test cases to execute on the student's solution. These tests are the teacher written test cases.

The teaching staff have to develop a set of test cases to use when testing student code. The continuous integration service expect the test cases to either be in students git repository or in the git repository named labs-test. The labs-test repository is a repository only accessible for the teaching staff, and will be hidden for the students, if the course is a private course. This extra git repository for teacher specific code and test cases is a security implementation. On private courses the labs-test repository will be hidden, thus preventing the students reverse engineering the test cases when trying to pass the assignments.

The hidden labs-test repository enables the teachers to also test for extra behavior, or have extended behavior testing, on students solutions. A good way to get extra insurance on what the students actually know about the course material. The hidden labs-test repository insures that the students can not reverse engineer the test cases to pass the assignments, but it is however a good practice to give the students some test cases which is open and clear for the students. This lets the students have some sort of confirmation of their work before uploading their code. A smaller test sample will let the students test if their code is on the right track before having it runned through the continuous integration service. The labs-test repository files will also have higher priority than students written files. If the students tries to overwrite the test files with new data, this will be stopped by the prioritizing of the files in the labs-test repository. This also has to be in the mind when teachers writes test cases. There cannot be any files with the same name in both students repository and the labs-test repository. The student's file would be overwritten.

In order to be able to run all the teacher written test, and to be able to support as many languages as possible, the teachers also need to supply a way to start up all the test cases. This is solved through having custom shell scripts runned as standard instead of calling the test cases directly. There are two script files for each lab assignment which will be runned when the continuous integration service starts up. One for the dependencies, which is intended to contain commands common for all the lab assignments. The second script file is intended to run the tests it self. These script files need to be supplied as well, from the teacher, in order to get the continuous integration service set up for running test cases on students solutions.

After running the test cases, the test results will be parsed and give a score back to the students with how many test cases passes or fails. It is a very basic way of giving a progress reports back to the students, and it will only show any progress when a test passes perfectly. However using score reporting the teachers can report back more detailed and specified data to the continuous integration service. This can be reported back by using a JSON object. When a test is running it could collect more detailed test informations, for instance each tested value, and summarize the a partial score. This score object can then be sent to the output stream where it will be picked up by the continuous integration service. Teachers can then, with the JSON score object, send back more detailed progress to the students from within the test cases.

The continuous integration service can also work without the need for teacher written test cases. Students can supply the test cases themselves. Then the students write test cases for their own code and those will be runned through the continuous integration service. The teaching staff still need to supply the script files, which runs the test cases. This script need to be able to start up any test the students mye write to actually run them. In the case of student written test cases, the autograder application will not supply any feedback about the course material. Due to the fact that all the test cases is supplied by the students, they will only test for behaviour in their knowledge and show which test cases passes on any given time. Testing through students written test cases can however be valuable for the teaching staff. It will immediately be clear, for the teaching staff, what test cases have been written by the students, and can therefore be helpful in the grading process, for instance when looking on code handed in by the students. If the course material includes use of students written code, it is recommended to use both teacher and student written test cases.

6.2 Score Reporting

When the continuous integration service have runned the test cases started up by the script files, it will try to interpret the data given back through the standard and error output. The most basic parsing of test data is to only look after the test itself, and report back to the teaching staff and student how many test passed and failed. However the continuous integration service also have a way to be delivered more specific test data.

When the test cases are executed the test itself can count up how many of the subset values a student's solution is passing, and give a more detailed picture of how the progress is of each student. After a test is finished, the test can output a specific JSON object in order to tell the integration service about the internal progress of a test. This JSON object, call a score object, contains information about which test it comes from, max score possible on the test, actual score reached by the student, how much weight the specific test should have on the total progress and lastly a security code.

If the continuous integration service finds a JSON object which can be parsed into a score object, it will switch over from the basic pass/fail test reporting to the more detailed test reporting. When the continuous integration service detects one or more score objects in the output data, it will use this information to calculate the score given through the web service to the student and teaching staff. When reading the different score objects a percentage between 0 and 100 will be calculated, using the following formula:

$$weight_{total} = \sum_{i=0}^{n} weight_i \tag{6.1}$$

$$score_{taski} = \frac{score_i}{maxscore_i} \tag{6.2}$$

$$score_{total} = \sum_{i=0}^{n} score_{taski} \cdot \frac{weight_i}{weight_{total}} \tag{6.3}$$

Through the formula the Autograder web service knows how the progress can be presented to the students and teaching staff in form of a process bar in the web service. This gives the students and teachers a simple way to get updated on their assignment progress.

The score objects also provides an uninformed way of telling how each test scored individually, and this also gets presented in the web service. Besides the process bar, giving a overall progress representation, each of the individual test will be presented as a list. This list shows the different test names, and the total influence the test matter on total progress.

```
{
  Secret   string // the unique identifier for your course
  TestName string // Name of the tests that is covered
  Score    int    // The score the student has accomplished
  MaxScore int    // Max score possible to get on this specific test(s)
  Weight   int    // The weight of this test(s)
}
```

Listing 6.1: JSON structure of score object.

The structure of the score object can be seen in listing 6.1 To be able to pick up on the score object it need to be in JSON, and have the properties TestName, Score, MaxScore, Weight and Secret. The TestName property tells the name of the origin test. Score, MaxScore and Weight tells how much the student has scored on the test of the possible max score, and the Weight tells how much of the total progress the score should count. The property Secret is a hexadecimal hash value specific for the course. This value ensures score objects only comes from test cases controlled by the teaching staff, and not from students trying to get a higher score. The hash value can be found in the course setting in the web service, and must not be shared with the students. Any score objects containing a different hash value than the designated for the specific will be ignored.

6.3 Student Work Process

When the students work on the assignments they receive all their assignments and supplementary code through the shared repository called labs for individual assignments and glabs for group assignments, on GitHub. From this repository the students will either copy the content over to their repository, or clone this repository directly from GitHub and put their own repository up as a remote location in git. The students can then download their assignments from a common place, managed by the teaching staff, and upload their assignment progress to their own private repository.

Setting up their git repository clone on their own computer is the only action needed for the students to actually begin on their assignments. They have the assignments and can start work on it as soon as they downloads this from the designed repository. While working on their assignment the procedure will be as normal, where they pick up the assignment, read up on the material needed to complete the tasks and implements their solution.

After a while the students feel confident they have a full or partial solution for the task presented in the assignment and can upload the implementation to GitHub. When they do this Autograder is notified and starts the testing of their solution. As soon as the test is finished running their solution, the results are presented to the students in the web service. The build log and the score will tell the students if there are need for improvements, and where the test cases shows an incomplete implementation. From this information the students can immediately see where they need to focus their energy to achieve a better solution. After reading the build log and test data, the students can go back to working on their assignments or do more research on the assignment and start over until they are satisfied with their solution.

Uploading their solutions to GitHub is not only encouraged to do when students want to test their solution. Often the course practice group assignments, which means students need to collaborate with other students in the same repository. With multiple students working on the solution for an assignment the students should upload their progress for sharing with other participants. If the students uploads their progress regularly it is easier for everyone involved to keep track of progress.

7

Use Cases

Two use cases has been set up to find out how well the theory of the Autograder project works and how capable the autograder application are in different situations. First use case have tested the autograder application for different programming languages and different ways to test the code. The second use case was on the course Distributed system at the University of Stavanger. In the last use case the autograder application was tested over a whole semester on actual students.

7.1 Test Frameworks

First use case in the Autograder Project was to test the capabilities of the application itself. Three different possible ways to use the Autograder application was put in use. The first of all the golang test framework was tested with autograder, this was also the test framework used in functionality testing while developing the prototype. The second usability test was to use plain shell scripts to report back students progress. With shell scripts the solutions was tested simply by checking output generated by the program, and what exit status it reported back. Third usability test was on the programming language Java, and is the most extensive usability test in this use case.

The most widely used programming language at colleges and universities in Norway is Java. Java also has an extensive testing framework called JUnit. Thus, it is important that the Autograder can perform automatic testing and parse results for Java-based lab assignments.

In order to validate the capability of the Autograder to test Java-based assignments, we took some of the exam exercises from the year 2010, and adapted them to a simple introductory course for Java. These exercises was created to test a student's ability in the most important aspects in object-oriented programming in Java, ranging from inheritance, composition, interfaces and abstract classes. The basic tasks was to create different types of mathematical sequences, one class for each of the following: Fibonacci, geometrical and arithmetical sequences.

In order to test how many of the assignments a potential student had managed to complete, a set with unit test was adapted to validate the answers submitted by a student. These unit tests also contained different score counters, which is reported to the autograder system and used to present the progress of the student submitting a solution.

The functionality test of this test case of the autograder project was done in house, which means the introductory Java course put into the autograder application has not been presented to a live set of students. Simulated students have gone through the lab exercises and tested the different stages through the lab work process. The results from this experiment confirms that the Autograder is fully capable of working with Java-based test cases.

7.2 Distributed Systems

One of the main objectives of this thesis is to find out how the students react to having better and faster feedback on their assignments. This use case explores the process of how the students work with their lab assignments and how their learning are affected by rapid feedback. The effect on students always having their current progress tracked also had to be explored. The psychological effect on students wanting to learn is of great importance. The interest to actually learn the details of the curriculum among the students holds large potential, where they can learn much from actually study course material in more detail. Another aspect which was in large interest is the possibility to have the students learn from each other and to the stimulate the students to ask more questions, not only to the teaching staff, but also to each other.

To explore these aspects of the autograder project, the autograder application has been used in the course Distributed systems [DAT520] at the University of Stavanger. The distributed systems course features a large lab part in distributed programming. This lab part of the course expected the students to create a distributed system in order to get a closer connection to how it works in practice. More specifically the students are have to create a leader and failure detector, and make this work with a paxos algorithm.

The lab project given to the student was a graded project which weighed 40% of their total grade in the course. Within the lab project the implementation of the paxos, failure detector and leader detector algorithms was weighed 40% of the grade, and the remaining 60% was weighed for the implementation of a network layer to transport the paxos messages between different machines running their software. In the lab project only the core curriculum was tested with help from the Autograder application. This means only the paxos, failure detector and leader detector algorithms was provided feedback for and confirmed with the continuous integration service. The remaining network layer was kept open for student interpretation.

For this use case the assignments was rewritten to fit into the Autograder projects work process. The assignments was adapted to be checked within the Autograder application. Manuals for the how the students should sign up for the course, understand their test results, use GitHub and new policies was created. Thanks to Tormod Erevik Lea a complete set with test cases was developed for the paxos lab. These test cases was upload to the labs-test repository and was subjected to the students solutions, and also a set of dumbed down test cases was delivered out to the students, to enable the students to partially test their solutions locally.

A new practice started from this use case, which has not been used before, was to have the students getting their assignments approved by a member of the teaching staff at the lab facilities within the lab hours. With this practice the teaching staff got a presentation of the work done by the students and could see if the students knew what they had implemented. Before a member of the teaching staff would approve a lab, the students needed to have at least 60% completion of the assignment.

At the end of the lab project the complete work done by the students would be tested in a lab exam. This exam was a 15 min presentation and question session. At the lab exam the students would present their solution one by one, and not as a group used in the assignment approval process. The individual question session tested the individual student for the knowledge needed to complete the lab project, and to be able to tell if the work is due to a single individual or a actual collective group effort.

8 Results

This section will go through the findings from the two use cases for the Autograder project. The usability chapter will go through the usability for different test frameworks and the findings from this testing of the application. Most of this chapter will however be based on the findings from applying the Autograder project on the Distributed Systems course at University of Stavanger. In this course the lab project was fully managed from the autograder application and assignments was confirmed through the test cases developed for the course and the in lab approval process. From the use of the Autograder project in distributed systems results was collected through interviews of both students and teaching staff, and an anonymous survey at the end of the course.

Throughout the course periodical interviews of the students was performed. In this interviews the students was asked questions about how the Autograder project affected the their work with the assignments. Among these questions it was asked if the students was able to work more structured with the rapid feedback in Autograder, if they felt the approval process and grading had improved or not, if they felt more motivated to go more in detail about the curriculum and if they felt more confident of their solutions.

The teaching staff was also interviewed on regular intervals. In these interviews the subject was about how they were able to follow up on the students, how well they knew the different students solutions, if there was easier to detect problem areas in the assignments and if they easier could find sufficient basis to grade the students in a fair way.

At the end of the lab project the students was also asked to fill out an anonymous survey [2]. This survey let them answer questions about the use of version control, continuous integration and test driven development in the Autograder project and how it could impact their future, how the Autograder project affected how they worked on the assignments, how they think the autograder could be improved and how assignments was compared to earlier years. This survey was created to get the most honest and open opinions from the student's total impression.

8.1 Usability

The usability of the continuous integration service was tested through three different test frameworks. These test frameworks was junit in Java, Golang test framework and a simple shell script to verify the solutions. These three possible ways to deliver test data back to the autograder application was tested by writing simple test cases, uploading them to a test course on GitHub and then simulating a student working on passing these test cases. When simulating the student working on the solution the continuous integration service delivered fast and steadily results back from the build process. While testing different test frameworks within the continuous integration service the Autograder application started and executed the build each time. This suggest that the continuous integration service are able to run many different test frameworks without problems. Since the test frameworks in this test are very different from each other also shows that it would probably be able to run many more types of test frameworks as well.

Another perspective the usability use case showed was the ability to parse the lower level of test data generated. The test cases was as mentioned very different from each other, and the continuous integration service was mainly developed with the Golang test framework in mind. For Golang test data, it could easily pick up and the number of passed, failed and build failures within the test data. However when the two other test frameworks was tested, the continuous integration service struggled at times to pick up on the right keywords in the test data. This lead to missing data in the presentation of failed, passed and build failures.

However in higher level of test feedback, where internal test scoring was used, the continuous integration service managed to pick up the correct information each time. This internal test scoring is the more detailed information returned about the internals of a test case. This score reporting follows a strict format for reporting back information and showed itself to work at all times with any of the test framework.

Even though the lower parsing of lower test data failed at some frameworks, the Autograder application gave back the complete build log for the test cases and did not limit the student from getting the feedback they needed and with use of the score reporting the student would also not be blocked from knowing their score.

In the Distributed Systems use case an important aspect about the feedback itself also emerged. Test cases developed for the Distributed systems course followed the natural feedback format from the Golang. The feedback informing about which action failed within a test was given by telling which function failed, what the expected values was and what was gotten instead. After using these test cases with the students, it showed itself not to be the best practice to use for students trying to learn. Many students reported back they often struggled to actually understand what was wrong in their solution by simply getting information about what was expected and what was gotten instead. They could not easily understand what part of the algorithm they had a wrong understanding off with this limited information.

From the problems students had with the feedback they got, it showed itself that any feedback are not always good feedback. When using test driven development to educate students about something new they need to have a reference to educational material with their normal feedback. It can be hard to find out what actually is expected from a algorithm by only examin values coming in and out of simple functions. Best practice found while testing in the distributed systems course was to have a combination between references to the curriculum and a natural test feedback for the test framework. With this combination the students could look up an aspect of the algorithm if they needed better understanding of it, and also be able to find minor errors in their code, for instance finding typos.

8.2 Student Learning

Through the use case in distributed systems course the students the students worked hard on the assignments and truly put the Autograder application up for testing. Statistics from the version control system and the web service showed that the students work steadily with their assignments throughout the semester, with peaks around the delivery dates, as illustrated in figure 8.1. The students ended up generating 2567 commits to their repositories and generating at least as many build requests in the continuous integration service. The statistics also showed the web service had 2365 sessions on the

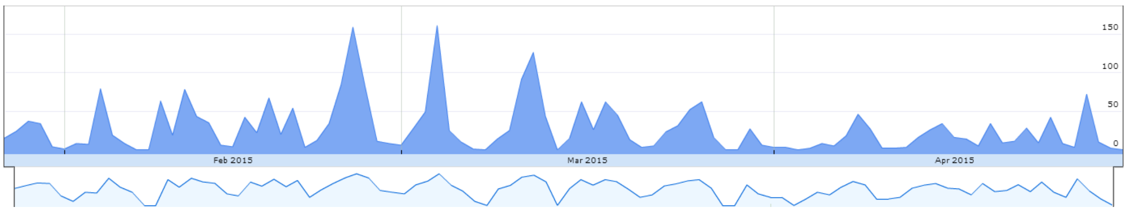


FIGURE 8.1: All commits created by students distributed over time.

autograder web site and 9975 pageviews. The number of commits and sessions are very close compared to each other and could suggest the students often visited the autograder web service for a closer inspection of their score. The number of page views generated by 31 students and a teaching staff of three suggests that the autograder application var frequently used and an important part of the lab project.

While the application was frequently used the total impact on students work process are also of importance. Through the interviews the students could report the rapid feedback had a positive impact on their work.

From the beginning the students the students felt they could easier start up with their assignments when they already had some parts put together for them. Often when the students get a completely open assignment, it is hard to know exactly know where to start in order to build the best possible solution. The test cases used in the course demanded the students to implement their code in some predetermined name spaces. This gave them a good starting point to start implementing their solutions. While they normally had to wonder about where to start, the students reported they could start on learning the material important for the course much sooner. Thus giving them more time to understand the material.

Better management of their time was also reported through other parts of the lab project. The student praised the possibility actually get their lab assignments checked while working on the. This gave them a quick way to see if they had understood the algorithms in the course or not. In the cases where aspects of the algorithms was unclear for them, they easily found out they had not fully understood the algorithm. With the feedback given they could find gaps in their knowledge and they could find the areas where they needed to research the algorithms more closely. Many students reported they could understand the lab material better when they got rapid feedback on what they had done correctly or not.

Through the lab project the students also reported back they managed to use their time better. Usually when they got stuck on their assignment they would use much time on requesting help to solve their problems. With the feedback they got from testing their assignments they could easier find out why the problem had occurred, and with the information gathered from the test data they managed to faster find their way back. These two scenarios given back from the students makes a strong connection to the reasoning in the hypothesis, where the students can free up time to learn the curriculum in more detail with rapid feedback.

Another important aspect with the assignments is the stress level the students have before handing in their solutions. This stress often comes from not knowing if they actually have understood the material correctly when handing their solution in. With the normal handin procedure they also need to wait a longer period before getting an answer to this. The long wait is often because the teaching staff actually need to go through all the answers afterwards, before giving feedback. Many students reported much of the stress was removed when they could test their solution on beforehand, and thus get more confident on how good their solution is before they handed it in.

The rapid feedback has shown itself to be very useful in helping the students to learn the curriculum better, and make them more confident with their answers. However there was also reported back a positive side with having a score given back to the students while they worked. This total assignment progress showed them how far they have gotten on the assignments, and this was also a good motivation factor. With the percentages rising higher and higher on their process bar, they got really motivated to get more. One of the students mentioned in the survey it was good to see how far they were from the goal, and even if it was on 98%, which well above approved, they still wanted to get those last two percentages. This shows the actual representation of how much they have completed drives them to actually learn more just to get full completion score.

The students also had a written exam in the course covering many of the same aspects as the lab project. After correcting the answers given in these exams there could also be seen an increase in knowledge among the students compared to earlier years. This increase in knowledge showed itself in the questions covering the same parts of the curriculum as the lab project. When taking the fact that the students wanted to dig a little deeper into the curriculum while working on the lab assignments into account, it looks like the improved learning in the lab project also has influenced what the student knew on the written exam. A powerful signal that the students also manages to carry with them the knowledge they obtained in the lab project.

8.3 Student Follow Up

One important task for the teaching staff is to follow up and help the students where it is needed. The students often ask the teacher about problem areas they have in the assignments, and might need further guidance from the teaching staff to resolve the problem. If there is only one or two who get such a problem, there not much time lost from guiding these students. However it raises a problem when many students ask about the exact same problem, and often each of these students use a lot of time to research their problem beforehand.

With Autograder in place the teaching staff can monitor the students more closely since they have direct access to each student's test data. When this while having this data on the teacher's fingertips, the teachers can easier see if the students are having problems with one particular area. When one or two students asked about a specific problem, the test data made it easy to research if more students had the same problem, but had not asked the teaching staff.

This scenario was also a problem during the distributed system course. The students was struggling with implement a certain part of the network layer and looking at the implementation submitted already by the students and using the test data, the teaching staff could detect a area where students needed a bit of extra guidance. Resolving this a longer description of how to solve the problem was published to the students and made the problem more manageable for the students.

In the distributed system course the test data showed itself useful for easily finding pain points in the assignments, and made teaching staff able to resolve this at a much earlier point in the assignments process.

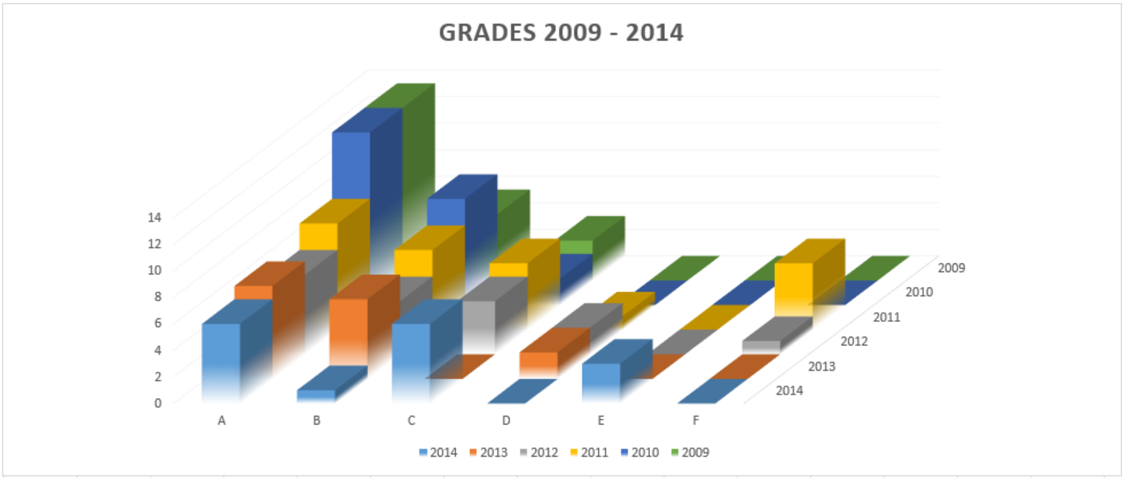
Another useful side with the information the autograder application delivered was when students asked for help with specifics in their programming code. From the test data the teaching staff could see which test cases failed and for what reason, and from the autograder web page the teaching staff could easily trace the problem to a section in the code. This made the process of helping the students much faster as the test results lead to a better insight to their code. From the autograder application it also was easy to find the students code for review, helping speeding up the process of finding a problem in the students programming code.

8.4 Grading Improvements

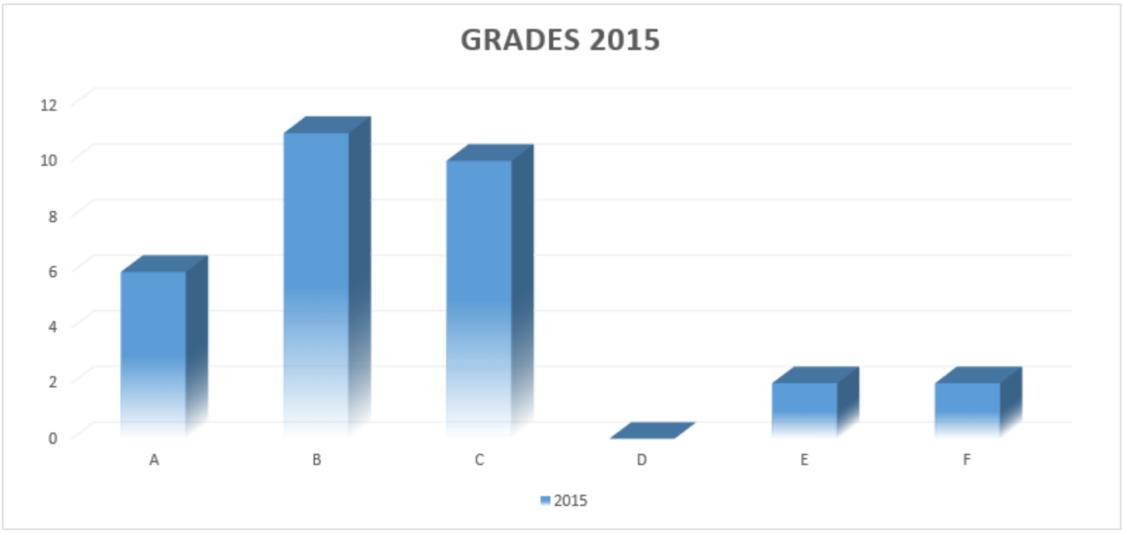
The process of grading students final product can be a tiring process. The teaching staff need go through all of the students solutions and find out how each student solved the assignments, and how well the solution is. This process takes time, which often is limited, and can in many circumstances lead to too little overall knowledge of the details within students solutions.

From the interviews of the teaching staff in distributed systems it was revealed they had earlier years had too little overview of what each student had implemented and if they actually had created a solution which contained all the details needed for a complete solution. As with this year's course there was also a lab exam previous years at the end of the lab project, but in the limited time of 15 minutes it is hard to reveal if the students had implemented all details within the parameters of the algorithms. With such a huge lab project as in distributed system it is also hard for the teaching staff to read through and understand the entire implementation of all the students within the grading deadline.

The teaching staff reported they got a much better overview of the work of each student with the Autograder project. Compared to earlier years they knew which details the student managed to solve of each algorithm. The overview of test results given through the Autograder application made it possible to see which part of the assignment each student had managed to complete and which parts they had less success rate with. This new immediate knowledge about what work the students had completed enabled the teaching staff to ask more specific questions during the lab exam. Having more specified questions to ask students let the teaching staff explore the knowledge of the student more closely and could make a better judgement of which grade the student deserved.



The effect of better knowledge about the work completed by the students are reflected in the grades given in this years course. The ideal grade curve would follow a normal distribution. This means number of grades would peek around the grade C. Previous years does however not reflect such a distribution of grades. The statistics for previous years show a clear peak around the grade A, and decreases in lower grades. According to the teaching staff this trend had developed because of limited insight in how the students solution worked. This lead to most of the grading work was based on what the students was able to show during the lab exam. This grading process lead to students easier getting a higher grade because they could have supposedly good implementations based on the what they could show in the lab exam.



Looking at the grades from this years students, a clear change in the distribution has occurred. The distribution peak of the students have shifted to a place between B and C. This years grades have moved towards the more ideal distribution center of C. Grades given will always change each year because there are new students. This can account for some of the changes in the distribution this year. However compared with the grades from previous years there is a much larger change this year from what has been seen earlier. According to teaching staff most of the change was due to better understanding of the solutions given by the students. This allowed the teaching staff to judge more from how the implementation worked. This could be observed from the test data in the Autograder application, and made a more accurate assessment of the grade.

Also the students could report they felt the Autograder project had a positive effect on the grading process. With test cases the students said they felt the judgement of their assignments where more fair. The test cases were the same which was used on every student, and this made the sensation of a more fair assessment of their assignments. Students could also report they felt they would end up with a more justified grade after the lab assignment, both in negative and positive grades, and removed much of the stress behind the grading process. The students knew immediately what they had achieved through their lab project and could easily make a personal estimate about where they would end up. This is also a positive improvement in the grading process for the student. Students often have to wait weeks before getting a grade back on their work without knowing exactly how well their solution represented the expected in the course.

8.5 Gamification

One of the goals in the Autograder project is to stimulate more to student collaboration in order to get them to learn the course material from each other. Often the students sit in smaller groups and talk to each other there and does not communicate much with the whole class. This can lead to some students burning in with questions another student could answer. One way to let the students ask each other questions, learn from each other on a larger scale and let them contribute to the course, is to let them have online discussions. However, in order to getting the students to actually start discussions online without getting pushed into it is to give some sort of motivation. In an attempt to motivate students to start discussions online on their own, a game engine was implemented for the Autograder prototype. This game engine rewarded the students with points and a position on a scoreboard for comments, issues and similar on GitHub.

The gamification process let the students get points and badges on their Autograder profile for helping each other and contributions in the course. They can also compare their contribution efforts on the scoreboard, and see if they have the position they want among their peers. The scoreboard is implemented to let the student get more competitive, fighting to be on top of the list.

Through the use case in Distributed systems course very limited results was obtained. The game engine was introduced for the student on the two last lab assignments and left them with only a month to familiarize themselves with the new reward system. Some of the students had in that period managed to put the game engine to use. Some of the students reported back they felt motivated to have more conversations on GitHub, help out with the course material and improving the test cases used in the course. Many of these students reported they were motivated by the scoreboard and actually had the desired effect. The teaching staff could also report back a increase in issues being published, comments from the students and help requested online after the game engine was introduced. This suggests a reward system for the students collaboration, in order to learn from each other, could function very well.

On a more negative side, some students reported back they felt compelled to create fake conversions in order to get points. The students who mentioned this problem felt they needed to come up with conversation topics to collect points to better their effort in the course. This is not a desired effect, the students should not feel compelled to make up things to ask about, but actually ask when they have questions and answer when they have knowledge which could help answer questions from other students.

While there is collected some information about how the game engine influence the students, it was introduced too late in the distributed systems course for an accurate measurement. Some preliminary results has presented itself. These results show the students can be stimulated to help each other and present their questions to the other students instead of waiting for help from the teachers. However requiring the students to use this system can limit the positive effect of a game engine. To truly see the full effect of a gamification system on online collaboration would need more testing and might need a improved game engine. How the gamification process can be improved and further tested is explained in the further work chapter.

8.6 Preparation for Industrial Workflows and Tools

An aspect in the education of students are to prepare them for the challenges they meet when heading out in the industry they are learning about. This aspect is one of the foundations for how the Autograder prototype has been built. The use of the version control system git, use of continuous integration to deliver builds and use of test driven development to test the students solutions are the root of the application. This allows the students to work in a similar environment to what they will be faced with after they graduate.

Through the Distributed Systems course at University of Stavanger the students need to control their code through git and have their solutions built and tested through the continuous integration service. The students also needed to collaborate in groups while working on their assignments, which meant they had to use the version control system to sync up their code to each others progress. The collaboration in groups simulates a team which many companies have working on the same codebase at the same time.

While starting up the course most student showed they had limited experience from using git or git like systems. They needed some tutoring before managing to set up their git repositories and connecting this to the remote storage. Some of the work needed to set up correct git connections in the course are considered advanced, but these aspects was documented in detailed manuals [10] to give the students best possible chance to manage this on their own. Manuals provided through private assignment descriptions are provided in Appendix D. Through the setup process, the teaching staff got many questions about the basic aspects of git, suggesting the students did not have a good enough foundation in their knowledge of git. Many students also asked often about how to upload and save their progress to the remote storage. Knowing this, its a clear sign students need more practical experience with using git, and quite possible version control systems. Through the course the students also showed they managed the process of uploading and controlling their repositories better.

The industry does not only use a version control system to manage their codebase, they also use testing and often automatic testing. This is where the test driven development and continuous integration comes into the picture. In the distributed system course the students was introduced to this through teacher written test and having these automatically runned in the Autograder application. Students have the possibility to interact with a continuous integration service and see how it works, and see how test can check how their programs. It can certainly teach the students something, but through the use in the distributed system course they did not really get a first hand try at the use of these practices. This is an opinion among the students also, the students reported they did not get a direct contact with the test driven environment.

It is not within the topic of this course to actually teach the students of how test driven development works or how the continuous integration can benefit them, but it is a goal of the Autograder project to silent train the students to use such practices. From the results found while using autograder in the distributed systems course it is clear the students could learn much of the use of version control in their courses, but in the other end silently training the students in test driven development can be a harder task. In order to solve this the students need a more hands on approach, where the students can write their own tests. To actually achieve this goal fully, courses could let the students write their own test. This will let the students get practical experience with test driven development. A setting requested by some students [2], and in their opinion get a better preparation before graduating.

9 Future Work

Several teachers at the University of Stavanger are already planning to adopt Autograder in their programming courses in the Fall semester of 2015, and the project has been awarded a grant from Prekubator TTO to continue development. This section will explain some beneficial extensions, which can make the Autograder project more practical and better equipped for the future.

9.1 Code Coverage

In most colleges and universities teaching test driven development is also a part of the education. This means students will need to write their own test cases in order to pass the course. In current version of the Autograder application, test cases are used to validate students solutions, and a way to test how well the students have implemented their tests are needed.

One way of testing how well the test cases work is to measure the coverage they have. Coverage is the amount of original code the test cases actually go through when runned. This can tell the teachers how much of the students code are actually tested.

Most of the test frameworks have the option to deliver coverage analysis, and Autograder should be able to read the coverage reported back from the test framework. When reading the standard output from the build and search for the format the code coverage is given in can let the continuous integration service know how much coverage the test cases have. This information can then be stored with the test data and be available for both students and teachers.

9.2 Advanced Cheat Detection with Code Analysis

Often there is a problem that students getting a bit behind on their assignments and get the temptation to cheat. Autograder project have been created to make the lab environment easier to manage and easier to grade, and a ways to easily detect acts of cheating is a huge step in this direction. An extension to the autograder application which makes it possible to detect where students have copied each other need to be created.

The easiest way of checking if a solution is a copy of another is to do a differential check. This checks how many of the lines in the code is similar to each other. This solution is however fairly easy for the students to work around, and most common way to trick it is to change function and variable names.

To overcome this challenge a more advanced analysis is needed. While changing the names of entities within the code does nothing to the compiled version of the programming code, changes to the structure however will have a larger impact. It is also the different structures of the programming code which makes the solutions from students different from each other and are the basis of the assignment. Because of this fact the structure itself need to be analysed, and a cheat detection extension to Autograder application should include a structural analysis rather than just a differential analysis.

9.3 Task Board

Task boards are widely used in the industry. When given a task to implement new software or feature, this main task is broken up into many smaller tasks. While breaking it down to smaller tasks this larger task is made more manageable. With each smaller task you have one simple thing that needs to be implemented and could be finalized in a short amount of time. Tasks are also often set up with a deadline where it needs be finished implemented.

This way of working is most useful when there is a team working on a common goal. Then each team member can pick a task to complete and pick a new task to work with after this. It becomes a easy way to divide the main task between all of team members.

This task board can be implemented in the automatic feedback system by having a designated page for showing a task board for each group or user in a course. These tasks can be added by the teacher or by the users themselves. When a teacher put up tasks, it will be distributed to all the taskboard in that course. User will only post task to their own task board. As the users work through their task, these task get marked with the username and a in development tag, to show that a user is working on that task, and when a task is finished, it gets tagged with a ended tag.

9.4 Improved Game Engine

In the current version of Autograder there is implemented a game engine. This game engine are not fully tested or optimized for full use. Some preliminary test results have been collected about using gamification to stimulate improvement in course software and online collaboration. These results showed conflicted results where some students thought they needed to invent conversions to get points and some felt motivated to ask more and help with the course material.

Even though the game engine was a partial success, it was introduced too late to actually get accurate results. Also the game engine should be further developed to get the best possible impact on a possible use case.

Improvements to the game engine can be done by include it better in the assignments and actual work process of the assignments, such as making it count at the final score of the assignment. One of the concerns from the students was that they did not get points for helping each other face to face. Letting the students get point for attending the lab facilities at help sessions, getting points for asking the teaching staff and similar scenarios.

9.5 Specified Test Data Parsing

In current Autograder prototype the continuous integration service searches the standards output for certain keywords in order to find out how many test cases have been runned, or if there is a build failure. The same technique is also used to find out how many test cases has failed or passes. Every test framework has their own way to represent this information, and the current version of Autograder is mostly optimized to look for the Golang test framework.

The continuous integration service can run any test framework, which can be started from command line, and would be able to calculate scores given in any test case. The problem presents itself when the pure build log is represented. The build log counts the occurrences of keywords represented by a test case in Golangs test framework, and because there is different keywords and patterns given in each of the different test frameworks these new patterns and keywords need to be in the search as well.

With other test frameworks in mind a specified test framework parsing extension is proposed. This extension should be able to pick up more of the patterns given by other frameworks and present them in a general way. Often many of the test frameworks also give data back in form of JSON or XML data as alternatives. This form of data should also be detected and parsed into readable test data.

In order to achieve the wanted parsing of different test frameworks it is possible to create a library of different patterns from test frameworks. This library can be implemented by having a list of text formats which corresponds with the expected output. Running the build log through this library can let the continuous integration service find out which type of test framework is used in the build process and decode data accordingly. More detailed parsing of the build log will make it possible to give more detailed feedback to the students.

Some test frameworks has the option to deliver the build log in other formats than pure text. An example of this is the test framework jUnit in Java, where test data can be delivered in XML format. Other formats are also used by different test frameworks. Methods for decoding these formats will make the continuous integration service open for wider usability from test frameworks.

With more specific parsing of test data it also possible to present more specific test feedback to the students and teaching staff. This can be done through grouping the test name, status and internal printout from the tests together. With the grouped test information, it is possible to represent this as individual sections instead of the raw build log in current version. Since the test data can be shown as individual sections, the representation of test data can be made more organized. Changing the interface into sections lets the students and teaching staff to select and read the output from specific test cases one at a time. This does not only make it more organized, but it would make the test data easier to read. In current version of autograder represent the test data as a complete build log of raw text. With sectioned data, students and teaching staff can find and investigate a specific test without actually searching through whole build log.

9.6 Automated Test Case Generation

One of the largest tasks with using the Autograder project in different courses is the amount of work needed to develop test cases for checking the solutions submitted by the teaching staff. These test cases are also one of the most important part for actually delivering the feedback to students. If the amount of work needed to create these test cases could be reduced, a substantial usability improvement of the Autograder would result.

In many courses the teachers have a solution to their lab assignments ready. By using these solutions to automatically create test cases for use in the Autograder would save a substantial amount of time for teachers.

To support automating the generation of test cases from an existing solution, a thorough survey of existing frameworks is needed. There might exists many good frameworks for automated test case generation from existing solutions. Using existing frameworks to execute this job will make it easier to create a good featured solution of this problem. The different possibilities for test generation need to be mapped and documented for easy use for the teachers within the Autograder project. Another possibility is to also build in the most common ones in the Autograder application itself.

10 Conclusion

In this thesis the objective was to improve learning through lab projects more valuable for both students and teacher. The students need better and more rapid feedback to faster improve their solutions and be able to learn more through the lab project. The Autograder project resulted in an application which can be used to automatically build solutions from the students and stimulate them to easier learn the course curriculum. Autograder was also tested on a master-level course at University of Stavanger with success. The students managed to easier work through the lab project and autograder gave a good way for the students to reach their potential. Through the application the teaching staff could easily manage their course, was relieved from the burden of manually correct lab assignments, easily find pain points in the assignments, and have better material for grading the students.

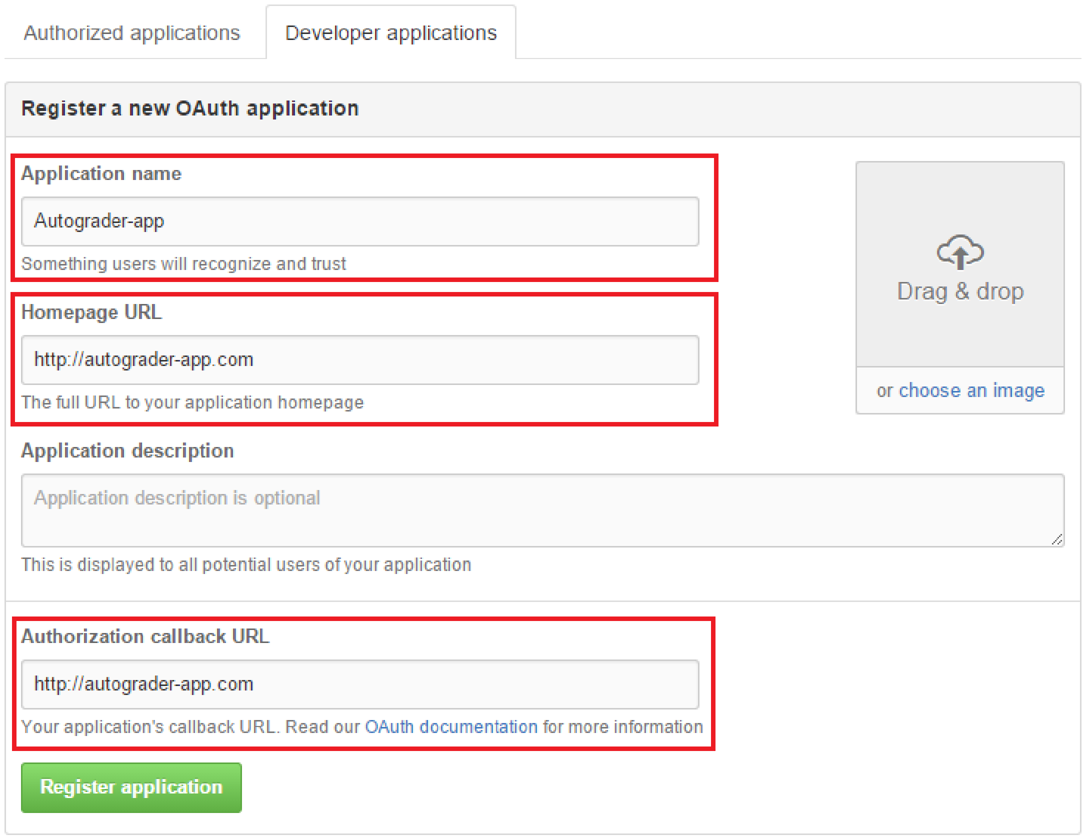
A

Github Application Codes

When installing a new instance of the Autograder application the installer need to get application codes from GitHub. These application codes a used through the OAuth protocol and are used to authenticate an application when signing in users through GitHubs services. Following is a step by step guide to obtain these codes.

1. Go to the page https://github.com/settings/applications
2. Click into “Developer applications”.

3. Click the "Register new application" button.



Register a new OAuth application

Application name

Autograder-app

Something users will recognize and trust

Homepage URL

http://autograder-app.com

The full URL to your application homepage

Application description

Application description is optional

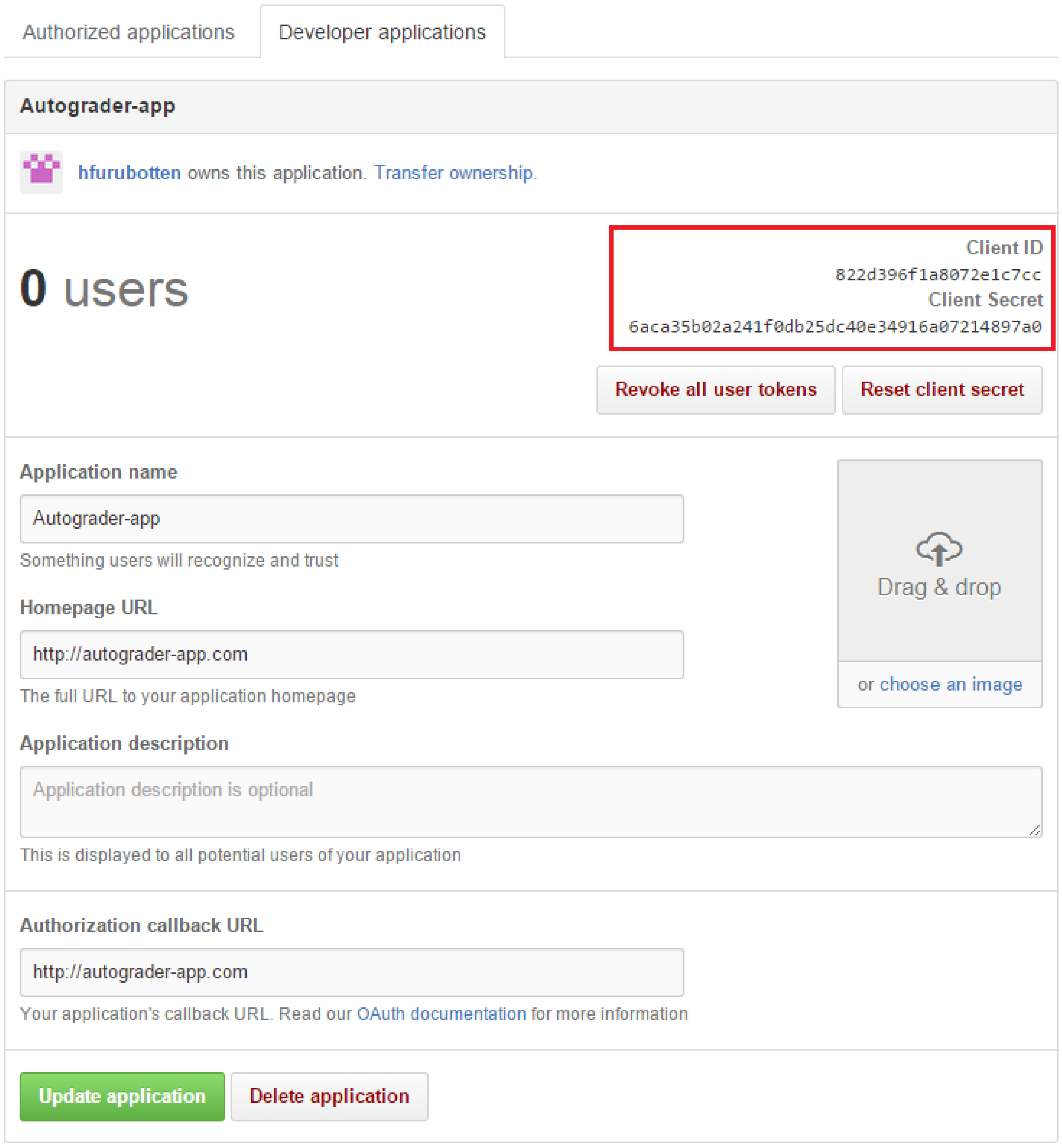
This is displayed to all potential users of your application

Authorization callback URL

http://autograder-app.com

Your application's callback URL. Read our OAuth documentation for more information

4. Application codes will then be available for the user.



Authorized applications | Developer applications

Autograder-app

hfurubotten owns this application. Transfer ownership.

0 users

Client ID
822d396f1a8072e1c7cc
Client Secret
6aca35b02a241f0db25dc40e34916a07214897a0

Application name

Autograder-app

Something users will recognize and trust

Homepage URL

http://autograder-app.com

The full URL to your application homepage

Application description

Application description is optional

This is displayed to all potential users of your application

Authorization callback URL

http://autograder-app.com

Your application's callback URL. Read our OAuth documentation for more information

B

Web Service Statistics

Following is a summary report of web site statistics, collected through Google Analytics.

Google Analytics

autograder.ux.uis.no - http://autograde...
All Web Site Data

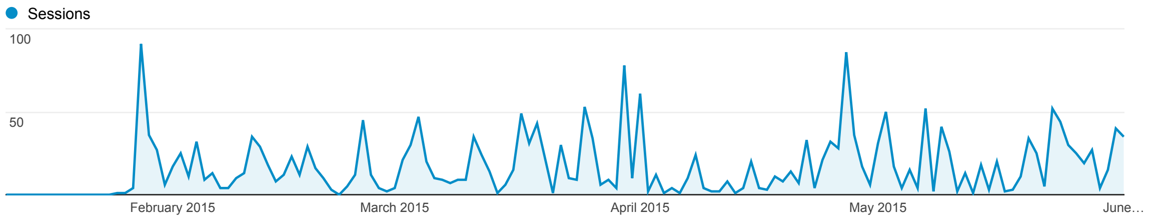
Go to this report

Audience Overview

Jan 11, 2015 - Jun 1, 2015

All Sessions
100.00%

Overview



Sessions
2,365

Users
1,187

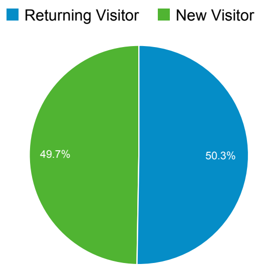
Pageviews
9,975

Pages / Session
4.22

Avg. Session Duration
00:03:57

Bounce Rate
49.13%

% New Sessions
49.68%



	Language	Sessions	% Sessions
1.	(not set)	973	41.14%
2.	en-us	741	31.33%
3.	nb	298	12.60%
4.	en-gb	130	5.50%
5.	nb-no	94	3.97%
6.	ru	64	2.71%
7.	en	34	1.44%
8.	fr	18	0.76%
9.	zh-cn	10	0.42%
10.	nn-no	2	0.08%

C
Git Setup Instructions

This section offers step-by-step instructions on how to complete and hand in Lab 1. Please refer to the workflow described below also for future labs unless otherwise noted. The tasks will introduce you to some basic programming in Go. You may find them easy if you have previous experience with the language, but they serve as a good example of how to work with Autograder.

1. You will have access to two repositories when you have registered using Autograder. The first is the labs repository, which is where we will publish all lab assignments, skeleton code and additional information needed. You only have read access to this repository. The second repository is your own repository named username-labs. Username should be substituted with your own GitHub username. You have write access to this repository. Your answers to the assignments should be pushed here.

2. To get started with the Go part of this lab, you can now use the go get command to clone the original labs repository. Here is how to do it: On the command line enter: `go get github.com/uis-dat520/labs` (ignore the message about no buildable Go files). This will clone the original labs git repo (not your copy of it.) This is important because it means that you don't need to change the import path in the source files to use your own repository's path. That is, when you make a

commit and push to submit your handin, you don't have to change this back to the original import path.

3. Change directory to: `cd $GOPATH/src/github.com/uis-dat520/labs.` Next, run the following command:

```
git remote add labs https://github.com/uis-dat520/username-labs
```

   where username should be replaced with your own GitHub username. The above command adds your own username-labs repository as a remote repository on your local machine. This means that once you've modified some files and committed the changes locally, you can run: `git push labs` to have them pushed up to your own username-labs repository on GitHub.

4. If you make changes to your own username-labs repository using the GitHub web interface, and want to pull those changes down to your own computer, you can run the command: git pull labs master. In later labs, you will work in groups. This approach is also the way that you can download (pull) your group's code changes from GitHub, assuming that another group member has previously pushed it out to GitHub.

5. As time goes by we (the teaching staff) will be publishing updates to the original labs repo, e.g. new lab assignments. To see these updates, you will need to run the following command: git pull origin master.

6. For the first set of labs we will provide you with skeleton code and a set of tests. Thus, you will have to implement the missing pieces of the skeleton code, and verify that your implementation passes the available tests. Note that Autograder will run an additional set of test cases to verify your implementation. Not all tests must pass to get a passing grade.

D
Autograder Usage Survey

Following is the survey sent to all students in Distributed system course at the end of the lab project.

Autograder usage survey

After the use of the prototype application of Autograder in DAT520, we would like to ask you what you think.

This survey only covers the part of the assignments covered in the autograder application and the autograder project. This means the code which was validated within the autograder application online. In other word, the paxos part. The focus will be on the learning process and if the application and work methods have helped you to get through the DAT520 course material.

Within the survey autograder is referenced in two ways, and might need a bit of clarification. We reference towards the autograder application and the autograder project. The autograder application is the web pages you get up when you go to look at your handed in answers and test scores. The autograder project is the overall project where the university is experimenting with giving instantaneous feedback on your assignments. This project also tries to stimulate to more cross group and student learning where students can learn from each other online and asking questions from anywhere, to anyone and at any time.

All answers are anonymous and are not traceable back to any single person. This survey will be a part of the master thesis Heine Furubotten is writing, and might be used in other papers released regarding the autograder project. We thank you for all answers.

*Required

Your first thoughts

1. **What is your first and honest opinion? ***
   Fist of all what is your first thought and honest about the autograder project? Do you have any thoughts about the strong and weak sides of having your assignments within an automatic feedback system, such as the autograder application?

Backgroud

2. **Have you used any automatic validation frameworks before?**
   In connection to your education have you used any frameworks to validate your assignments automatically? If yes, please note them below.

3. **Have you worked with git/GitHub before this course?** *
   Mark only one oval.
   - Yes
   - No

4. **Have you work with Test Driven Development(TDD) before?** *
   Mark only one oval.
   - Yes
   - No

5. **Have you worked with Continuous Integration(CI) before?** *
   Mark only one oval.
   - Yes
   - No
   - Unsure

6. **Have you taken this course earlier?** *
   There is quite a few students who took this course previously. If you are one of those who took the class before, we would like to ask some questions to be able to compare it to previous years.
   Mark only one oval.
   - Yes
   - No *Skip to question 12.*

Compared to earlier years

7. **Did autograder provide better feedback compared to earlier?** *
   Compared to earlier years, have the autograder project given you more or the needed feedback to understand the course material?
   Mark only one oval.
   - Yes
   - No
   - Unsure

8. **Do you think it was easier to see the correctness of your assignments this year?** *
   Mark only one oval.
   - Yes
   - No
   - Unsure

9. **Was the course material easier to understand this year?** *
Was it easier to understand the course material with the feedback given from autograder?
Mark only one oval.

- Yes
- No
- Unsure

10. **Is there anything you think was done better in the lab assignments last year?**

11. **What are your view on the changes done to this year lab assignments?**

Assignment handling

12. **Do you think using git to handle your assignments is a good solution?** *
Mark only one oval.

- Yes
- No
- Unsure

13. **Do you think using GitHub to storing your assignments is a good solution?** *
Mark only one oval.

- Yes
- No
- Unsure

14. **Do you think using git is an advantage for you long term?** *
Git and other version control systems are heavily used many places in the industry. Do you feel its an advantage to use git at the university, before heading out in working life?
Mark only one oval.

- Yes
- No
- Unsure

15. **Do you think using GitHub is an advantage for you long term?** *
GitHub is a much used resource in the open source and other communities. Do you feel its an advantage to use GitHub at the university, before heading out in working life?
Mark only one oval.

- Yes
- No
- Unsure

16. **Any comments on the use of git and GitHub?**

17. **Do you think using Test Driven Development (TDD) to validate assignment tasks is a good solution?** *
Mark only one oval.

- Yes
- No
- Unsure

18. **Do you think your experience with TDD within autograder give you an advantage long term?** *
Writing test to validating the correctness of code is often used in the industry. Do you think you get an advantage by having TDD as a part of the validation process, regards to using this later?
Mark only one oval.

- Yes
- No
- Unsure

19. **Any comments to the use of TDD?**

20. **Was it valuable for you to have your assignments continuously built in autograder?** *
The autograder application uses an adapted Continuous Integration environment to always build your solutions. Was it valuable for you to always get your code built and tested each time you pushed it to github?
Mark only one oval.

- Yes
- No
- Unsure

21. **Do you think it is an advantage for you to have worked within a CI environment long term?** *
CI environments are often used in the industry to continuously check the state of a code base at all time. This is also often used together with TDD. Your experience with this in the autograder project, will you say this gives you an advantage before heading out to working life?
Mark only one oval.

- Yes
- No
- Unsure

22. **Any comments to the use of CI in autograder?**

Assignment work process

23. **Did the test let you better/faster understand the needed parts with the assignments?** *
With the feedback and the continuous builds in the autograder application, was it easier to understand the needed parts of the assignements?
Mark only one oval.

- Yes
- No
- Unsure

24. **Do you feel you got the feedback needed to understand the different task?** *
Mark only one oval.

- Yes
- No
- Unsure

25. **If No to previous question, what was missing in the feedback?**

....

....

....

....

....

26. **To get the tests to pass what did you focus on?**
Tick all that apply.

- ☐ Reverse engineer the tests
- ☐ Understanding the algorithms
- ☐ Replicating the pseudo code given
- ☐ Other: ....

27. **Is it better to reference the algorithm themselves?** *
Would it be easier to understand the algorithms themselves if feedback given had referenced to what part of the algorithm was failing?
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Unsure

28. **Is it better to have a combo between algorithm reference and todays feedback?** *
Would you say it is better to use both reference to the algorithm themselves and the test failure data used today?
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Unsure

29. **Any comments on the feedback given through autograder?**

....

....

....

....

....

Alternative work processes

30. **Do you believe completely open assignments had worked better?** *
Do you believe completely open assignments would have been a better choice? This practice has been used earlier years and featured the possibility to do the assignments freely within golang. By completely open assignments, we mean assignments without sceleton code and any previously handed out code from the teaching staff.
Mark only one oval.

- Yes
- No
- Unsure

31. **Do you think the lab assignments would be easier without the sceleton code and locked in code setup?** *
Mark only one oval.

- Yes
- No
- Unsure

32. **If Yes to previous question, would it still be simpler without any automatic feedback at all?**
Mark only one oval.

- Yes
- No
- Unsure

33. **If autograder had been used in other courses on the university would you see that as an advantage?** *
We are looking into using autograder in other courses on the university. If autograder was used in other courses at the university, would that make it easier to use this work process?
Mark only one oval.

- Yes
- No
- Unsure

34. **Is there certain courses at the university where you think autograder could be put to good use?**

35. **Any other comments about how autograder could be used?**

Point system

In the middle of the course we introduced a point system, which could influence your grade positive. This system was put in place in an effort to stimulate questions being asked and tried solved by other students. The points where given in order to give you as a student something back for helping each other.

The point system was not meant to be extra work towards the assignments, but a way for you as a students to help each other getting the answers needed, even when the teaching staff was not available.

This system was introduced a bit late in the course, but we would like to ask for some feedback in order to improve it for later use.

36. **Do you feel the point system worked as intended?** *

    Mark only one oval.

    - Yes
    - No
    - Unsure

37. **Did you feel more motivated to ask your questions with the point system in place?** *

    In other words, did you have more motivation to put your questions on github when you knew you would get points back for doing so?
    Mark only one oval.

    - Yes
    - No
    - Unsure

38. **Did you feel a barrier to post your questions knowing everyone could see your post?** *

    Mark only one oval.

    - Yes
    - No
    - Unsure

39. **With the ability to see the top contributors in the autograder application, did you feel competitive to get on top of this list?** *

    Mark only one oval.

    - Yes
    - No
    - Unsure

40. **Do you think it would be a good idea to get points for lab attendance also?** *

    Mark only one oval.

    - Yes
    - No
    - Unsure

41. **Was there a technical barrier preventing you from posting your questions?** *

eg. problems with github or similar

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Unsure

42. **If there was a technical barrier, then what was your problem?**

43. **Any comments you can give towards the point system?**

Last comments

44. **Any last thoughts you want to share?**

After going though all the questions and had the time to think more about certain aspects about the autograder project, is there anything, good or bad, you want us to know about?

Bibliography

[1] Meet Jenkins - Jenkins - Jenkins Wiki. URL https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins.

[2] Autograder Usage Survey. URL https://docs.google.com/forms/d/1VYHWTpBig8os5AqXF7p4jXDJrFJrw1j0uQpI9vsh1Dk/viewanalytics.

[3] Travis CI: Getting started. URL http://docs.travis-ci.com/user/getting-started/.

[4] Kent Beck. *Test-driven Development: By Example*. 2003. ISBN 0321146530. URL http://www.google.no/books?hl=no&lr=&id=gFgnde_vwMAC&pgis=1.

[5] Collabnet. *Rapid Subversion Adoption Validates Enterprise Readiness and Challenges Traditional Software Configuration Management Leaders*. Collabnet, 2007. URL http://www.open.collab.net/news/press/2007/svn_momentum.html.

[6] Shaumik Daityari. Version Control Software in 2014: What are Your Options?, 2014. URL http://www.sitepoint.com/version-control-software-2014-what-options/.

[7] Martin Fowler. Continuous Integration, 2006. URL http://martinfowler.com/articles/continuousIntegration.html.

[8] Heine Furubotten. Feedback to DAT320 Operating Systems - Google Forms, 2014. URL https://docs.google.com/forms/d/1CrTc7peFM6CXPfFrLECixmNqB1qw6jRBO5QVAyUrcMI/viewanalytics#start=publishanalytics.

[9] Marksalisbury. What if early feedback made your students work harder? (Spoiler Alert) — Delicious Ambiguity, 2014. URL http://www.augustana.edu/blogs/ir/?p=1100.

[10] Hein Meling, Tormod Erevik Lea, and Heine Furubotten. GitHub manuals. URL https://github.com/uis-dat520/course-info.

[11] Morten Mossige, Arnaud Gotlieb, and Hein Meling. Testing robot controllers using constraint programming and continuous integration. *Information and Software Technology*, 57:169–185, January 2014. ISSN 09505849. doi: 10.1016/j.infsof.2014.09.009. URL http://linkinghub.elsevier.com/retrieve/pii/S0950584914002080.

[12] Craig Murphy. Improving Application Quality Using Test-Driven Development (TDD), 2005. URL http://www.methodsandtools.com/archive/archive.php?id=20.