UTILIZING CODEGRADE TO BUILD AN INTRODUCTORY WEB PROGRAMMING COURSE

Lappeenranta–Lahti University of Technology LUT

Bachelor's thesis, Software Engineering

2023

Author: Vili Huusko

Supervisor: University Lecturer Erno Vanhala (D.Sc.)

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Software Engineering

Vili Huusko

Utilizing CodeGrade to build an introductory web programming course

Bachelor’s thesis

2023

41 pages and 8 figures

Examiner: University Lecturer Erno Vanhala (D.Sc.)

Keywords: automatic assessment, CodeGrade, web programming, computing education, programming education

In autumn 2021, LUT University received an improved web programming course Web Applications. However, there were problems with the first version of this new and improved course, as the course required students to learn too many things in too little time. Because of this, the number of active students in the course dropped drastically in the early weeks.

This bachelor's thesis aims to create a solution to the steep learning curve of Web Applications by creating an introductory web programming course. The thesis also investigates what tools Cypress and CodeGrade have to offer, and how the automatic assessment should be implemented. The research method used in the study is design-science research methodology. The result of this thesis is evaluated from the continuous feedback accumulated during the course, the performance of the students, and the course feedback collected at the end of the course.

According to the course feedback, some of the students liked using CodeGrade, but some found it too difficult to use, especially on the part of submitting the exercises. The students thought that the weekly assignments were suitably challenging and interesting, but the assignment descriptions were sometimes unclear. The study also revealed that the automatic feedback was unclear to students who cannot interpret stack traces or regular expressions.

In the future, the course assignments and automatic feedback should be rewritten and made clearer. The course's coding environment should also be revised, as it would simplify the submitting process thus having the potential to improve students’ attitude towards CodeGrade and speeding up the testing process.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tietotekniikka

Vili Huusko

Web-ohjelmointikurssin rakentaminen CodeGradea hyödyntäen

Tietotekniikan kandidaatintyö

2023

41 sivua ja 8 kuvaa

Tarkastaja: Yliopisto-opettaja Erno Vanhala (TkT)

Avainsanat: automaattitarkistus, CodeGrade, verkko-ohjelmointi, tietotekniikan opetus, ohjelmoinnin opetus

LUT-yliopisto sai syksyllä 2021 uusitun verkko-ohjelmointikurssin Web Applications. Tämän uuden ja parannellun kurssin ensivedoksessa esiintyi kuitenkin ongelmia, sillä kurssilla tuli opetella liikaa asioita liian lyhyessä ajassa. Tämän vuoksi kurssin aktiivinen opiskelijamäärä putosi roimasti alkuviikkoina.

Tässä kandidaatintutkielmassa pyritään luomaan ratkaisu Web Applicationsin jyrkälle oppimiskäyrälle luomalla uusi verkko-ohjelmoinnin alkeiskurssi. Tutkielmassa myös selvitetään, mitä Cypress ja CodeGrade tarjoavat, ja miten automaatiotestaus tulisi implementoida. Tutkimusmenetelmänä tutkielmassa käytetään kehittämistutkimusmenetelmää. Tutkielman tulos arvioidaan kurssilla kertyneestä jatkuvasta suullisesta palautteesta, opiskelijoiden menestymisestä, sekä kurssilla kerätystä kurssipalautteesta.

Kurssipalaute osoitti, että osa opiskelijoista piti CodeGraden käytöstä, mutta osa koki asian liian hankalaksi muun muassa palauttamisen tiimoilta. Opiskelijoiden mielestä viikkotehtävät olivat sopivan haastavia ja mielenkiintoisia, mutta tehtävänannot olivat välillä epäselviä. Tutkielmassa myös selvisi, että automaattinen palaute oli epäselvää opiskelijoille, jotka eivät osaa tulkita pinojäljitystä tai säännöllisiä lausekkeita.

Tulevaisuudessa kurssin tehtävänannot, sekä automaattinen palaute tulee kirjoittaa uusiksi ja selkeämmäksi. Myös kurssin koodausympäristö tulee miettiä uudestaan, sillä se helpottaisi palautusprosessia nostaen mahdollisesti CodeGraden suosiota ja nopeuttaen testaamista.

Abbreviations

AAT	Automated Assessment tool
API	Application Programming Interface
CDN	Content delivery network
CLI	Command line interface
CSS	Cascading style sheets
DSRM	Design-science research methodology
HTML	Hypertext markup language
JS	JavaScript
JSON	JavaScript Object Notation
LMS	Learning management system
MOOC	Massive open online cours

Table of contents

1. Introduction

Lappeenranta-Lahti University of Technology LUT has offered before 2022 only one mandatory web-programming course, *Web applications*, for the software engineering degree program. This course gives the students the required tools and knowledge to build full-stack applications. This course, however, saw a massive drop in attending students after the third week due to having too many topics crammed into too little time. For this reason, the course was divided into two separate courses for the 2022-2023 term. The course was divided into *Introduction to web programming* and *Advanced web programming* based on the level of difficulty of the topics covered. The upcoming *Advanced web programming* is going to inherit most of the content from *Web applications* but the content to the new *Introduction to web programming* has to be made from scratch. New exercises must be built for the course and the assessment of the exercises have to be made utilizing the CodeGrade automated assessment tool (AAT).

The *Introduction to web programming* course covers the basics of client-side (front-end) web development. It will teach the students how to write Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, and how to use several JavaScript frameworks, namely Frappe-chart.js, Leaflet and Phaser. The course also teaches how to make application programming interface (API) calls and how to fetch and use JavaScript object notation (JSON) data.

The *Advanced web programming* course will focus on full-stack development. In addition to front-end development, it also consists of back-end (server-side) programming. Students will get familiarized with concepts such as Node.js JavaScript runtime environment and React.js front-end framework.

The intensiveness of the previous *Web applications* course was a factor in splitting the course into two parts to smooth out the difficulty curve. The course saw a major drop in attending students in week 4 during the introduction of Node.js back-end programming. The course feedback suggested that many students found the exercises of week 4 overwhelming and the increase in difficulty overwhelmingly rapid.

1.1 Objectives and goals

This thesis focuses on the making of the exercises to the new *Introduction to web programming* course and utilizing CodeGrade AAT and Cypress testing framework in the assessment of the exercises. Improving the original exercises from the *Web applications* course is important to achieve a more gradual increase in difficulty. The exercises also cannot be too easy in order to be engaging and promote learning.

The main research questions for this thesis are:

- How to construct new exercises for the upcoming web programming course?
- How to make the difficulty curve less steep?
- How to implement CodeGrade with Cypress to automatically assess the programming exercises?

The research method used in this thesis is design-science research methodology. This will be covered more in-depth in Chapter 3.

1.2 Structure of the thesis

This thesis consists of seven chapters. The first chapter is the introductory part of the thesis. It gives a small summary about the topic of the thesis and why the research was done. It also includes the thesis' research questions and remarks on how the research was done and what methodology it was done with. The second chapter gives background about the subject. It focuses on AATs in general and their history. It also covers a literature review on the topic of automatic assessment in education environment and analyzes it from the point of the course. The third chapter covers the research methodology.

Chapter four will focus on the tools used to test the assignments. It discusses what requirements the testing tools have, what features CodeGrade and Cypress have to offer, how the tests are made, what is being tested and how the tests integrate to the learning management system (LMS). The fifth chapter covers the upcoming *Introduction to web development* course and its structure and what is expected from a course itself and the reason

for some of the subjects taught. It also covers the programming exercises and the used technologies in each week’s exercises.

The sixth chapter discusses key findings of the research, and the created artifact – the exercises for the course and the tests for the exercises. The thesis concludes in the seventh chapter by summarizing the created artifact, key findings and discussing what the future holds for the research.

2. Automated assessment of programming exercises

This chapter covers automated assessment and its applications in education environments. The history of automatic assessment in education environment is also briefly covered to give background insight on the subject. Additionally, the advantages and disadvantages of automatic assessment is also covered based on research already done on the subject, and the findings of this research are analyzed. Based on the findings and the analysis of already done research, it is also weighted whether automatic assessment is suitable for the course, and what to endorse, and more importantly, avoid in the tests and assignments.

2.1 History of automatic assessment tools

Automatic assessment in education environments has a long history that goes back as far as 1960s and it has been in the interest of computer science educators ever since [1]. In the history of automatic assessment, three generations of assessment systems can be identified according to age and features [2].

First generation

The first generation of automatic assessment tools (AATs) dates back to the 1960s and 1970s, and it consists mostly of custom tools tailored for the courses themselves. Earlier models of AATs in this era did not have many features and mainly showed if the submitted code passed the predetermined specification or not, by matching output symbols. The later models had more features including a grade book, keeping track of students' progression over the course and a report generation function. Some ideas of plagiarism detection based on the size and execution time of the program were introduced, and concerns were raised about malicious code breaking the AAT software. [2], [3]

Second generation

Through the 1980s all the way to 2000s the AATs became increasingly easy to use and develop. During this period, the automated assessment systems were created using tools and utilities supplied by the operating system, and they started utilizing command line interfaces or rarely graphical user interfaces. They also adopted more pedagogical features, such as grading and assignment management. The most successful AATs from this era became widely used across several institutions, including BOSS and ASSYST. [2]–[4]

Third generation

The third and latest generation of AATs utilize the developments in web technology. Graphical user interfaces became increasingly common, and they also started to have more sophisticated features such as detailed feedback and different roles for students, admins, and teachers. Course management became a prominent feature of the AATs and plagiarism detection got developed further. During this period, the AATs evolved from simple tools to whole assessment systems. [2], [3]

2.2 Related research of automatic assessment

Integrating an automatic assessment system to a programming course involves a lot of manual labor. This is often compensated by the multiple benefits of the system, including reduced workload of the staff while assessing submissions. Automatic assessment systems, however, have also negative sides that need to be considered before integrating a system to a course.

As automatic assessment in education environment has been around for over sixty years, it has been the topic of numerous research. This section covers an integrative literature review of the topic of automatic assessment in a programming education environment and compiles relevant findings of different research. With the previous research covered, it is also discussed what needs to be considered and evaluated from the artifact of this research.

2.2.1 Advantages of automated assessment

Automatic assessment of programming exercises has multiple benefits compared to manual assessment. The main advantage is the time it takes to manually assess the assignments. Programming courses, especially introductory ones, can grow to enormous sizes. It is not uncommon for an introductory programming course to have hundreds of students. Therefore, manually assessing the assignments would take hours or even days [5]. With an automatic assessment system, the work can be done in a matter of minutes [5].

The time saved by automatically assessing the assignments has other benefits. The reduced workload of the teaching staff has the added benefit of focusing the time used to assess the assignments to other areas, such as guiding and instructing students [6]. As automatic tests by their nature require a more precise description and instructions without any ambiguity [2], [7] and teachers spending less time on assessing the submissions, more time and effort is spent creating the tasks, making them better learning tools [7].

The AAT can also handle multiple submissions at once, allowing students to receive instant feedback. This also allows students to submit their work multiple times in a short period of time. It is beneficial for the students to get instant feedback from their work especially if their submission required only minor adjustments [7]. The ability to assess multiple submission at once also has the effect on increasing the course capacity. This characteristic makes AATs especially useful in asynchronous massive open online course (MOOC) environments, where students do not directly interact with a teacher [6]–[8]. With online courses, especially MOOCs becoming increasingly common [9], AATs have proven to be valuable tools to use in such courses [6]–[8].

Automatic assessment is also a deterministic system, having a certain output every single time. This makes them objective, which has the benefit of eliminating human error due to fatigue or motivation [8]. It is also possible for a human assessor to have certain level of bias when assessing the assignments, which in turn invalidates the assessing process. This does not happen with an AAT [8].

2.2.2 Disadvantages of automatic assessment

Despite all the beneficial aspects of automated assessment, it also has some negatives to it. As stated before, AATs are deterministic systems, expecting a certain output with a given input. This expels a level of creativity from the assignments and makes them more one-dimensional [7]. In the example of a web-programming course, the input could be the elements and their attributes in a web site. This hinders the student's learning of program designing.

The AAT also requires a hefty manual process before being usable. This includes creating assignments that can be tested automatically, setting the AAT up, creating possible setup scripts, creating the tests themselves, testing the tests and troubleshooting. The tests may also need continuous maintenance if students find bugs in the tests. Pieterse [7] states that the time and effort saved using an AAT is merely moved from assessing the assignments to the manual labor caused by the AAT.

A lot more goes into programming than just getting the program to work. Code readability and structuring makes maintaining the code easier, and allows for other developers to get familiarized with the code much faster [10]. This makes it important to have sound source code that's well-structured and documented. Although CodeGrade supports code quality testing, documentation still has to be checked manually.

Automatic assessment also requires a higher level of skill from the students than traditional manual assessment [7]. The higher skill expectation can also reduce students' attitude towards the automatic assessment system, and reduce their study motivation [11]. This can be mitigated by giving extensive feedback on the tests that failed. On the contrary, experienced programmers may enjoy using the system increasing their studying motivation [11].

2.2.3 Analysis of related research

Weighting the advantages and disadvantages found in previous research, AAT suits the course well. Despite being an introductory course for web development, it is suggested to take late in the bachelor's program in LUT, during the third year. For this reason, the students

are expected to be somewhat experienced in programming thus benefitting more from the AAT.

Despite consuming time to setup and get working, the AAT process will save time in the long run. The assignments and tests can be reused in the future implementations of the course. Even if the assignments need to be corrected or revised in the future, it still equals to less work than having to manually assess every assignment in every implementation of the course.

The weekly exercises of the course are simple enough to be tested with a suitable testing framework. Järventausta created and evaluated the weekly exercises of the previous *Web Applications* course in his thesis *Piloting a new automated assessment tool in the Web Applications course* and found out that Cypress and CodeGrade are suitable tools to test simple front-end applications and full-stack applications [12].

Järventausta, however, found out that the feedback provided by the AAT was not sufficient in some cases, leading to students disliking the system on some level [12]. Manuel Rubio-Sánchez and Cheng found similar results in their research [11], [13]. For this reason, it is important to provide extensive feedback, and not just the default stack trace error that Cypress provides, as it can be hard to understand for some students. Extra attention must also be paid to assignment descriptions to avoid ambiguity [7].

As mentioned earlier, creativity is difficult to incorporate in assignments with automatic assessment. This has the effect of barring a level of freedom in the weekly exercises as they need to be done in a certain way. However, in the course project this type of freedom and creativity from a program design standpoint is desired, in order to test whether the students have learned enough to apply their knowledge in a bigger project. Therefore, the course project needs to be assessed manually.

3. Research methodology

This chapter covers the research methodology used in this thesis. The starting point of the research is discussed, and the required activities for the research are iterated and covered in-depth.

The research methodology used in this thesis is design-science research methodology (DSRM), which is used when aiming to solve a problem through creating an artifact [14]. In this thesis' case the problem is the too steep learning curve of the previous *Web applications* course, and the artifact is the new course, including assignment instructions, prototypes, and automatic assessments of the assignments.

Information technology researchers started to show interest in DSRM during the 1990's making it a relatively new research methodology [15]. DSRM has six main activities: problem identification, objective and solution definition, design and development, demonstration, evaluation, and communication. As with different steps, DSRM has also has different entry points: problem-centered, objective-centered, design and development-centered, and client/context-centered entry point [15]. An overview of the DSRM process is presented in Figure 1.

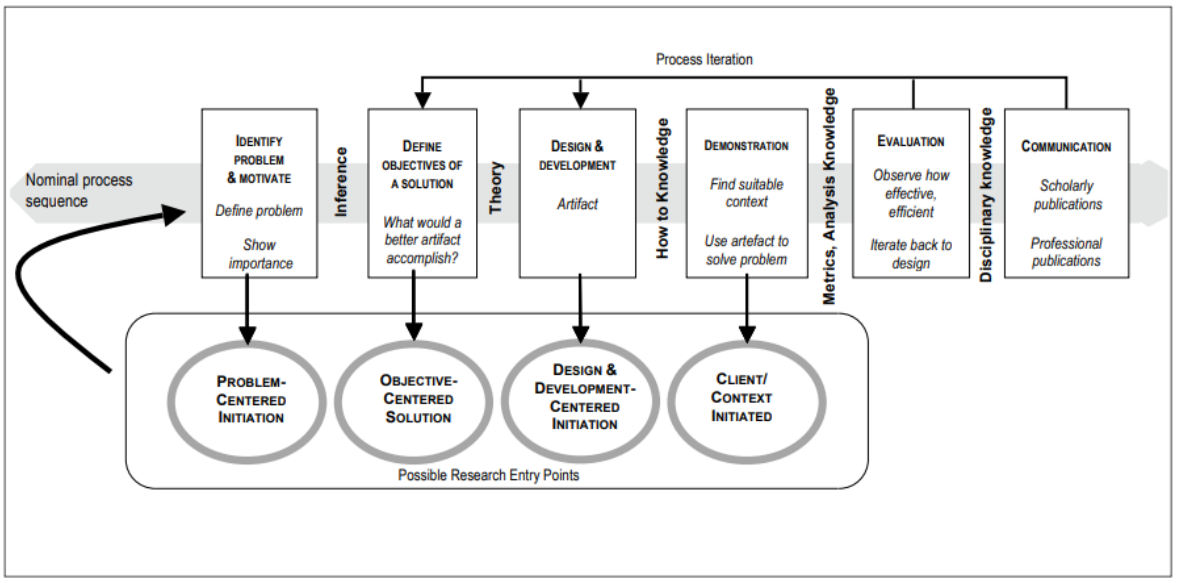

Figure 1: Design Science Research Methodology Process Model [15]

Before discussing how the research is conducted in every step of the DSRM model, the starting point must be evaluated, since the starting point determines the required steps of the DSRM process (Figure 1).

The problem of this thesis being known from the beginning supports it being objective-centered. The definition of an objective-centered solution according to Peffers et al. [15] is as follows:

> "An objective-centered solution, starting with activity two, could be triggered by an industry or research need that can be addressed by developing an artifact."

Given the definition of an objective-centered solution, this thesis aims to offer an objective-centered solution to the main research problem. As the thesis is objective-centered, the DSRM process starts from the 2nd step: objective and solution definition. The following section briefly covers all the methodology steps from the point of this thesis.

Objective and solution definition

The main problem in this research is the too steep learning curve of the *Web applications* course. This thesis aims to solve that by creating a new introductory web programming course focusing solely on front-end programming. This evens the workload allowing students to have an easier approach on the back-end development not having to learn everything from scratch in just a few weeks.

This thesis focuses on the design and implementation of the weekly exercises and the course structure and does not touch the lecture side of the course. The design of the weekly exercises is discussed and what is being required from the students each week. It is also investigated how the exercises are constructed in such manner that no week will have too much new subjects to learn, and how new knowledge is built on top of old knowledge. The content of the assignments is described more in-depth in Chapter 5.

Another problem the thesis aims to solve is how to implement CodeGrade and Cypress to the programming exercises. The problem is partly solved by Järventausta in his thesis [12], and the required features are presented in Chapter 4. The problem of testing JavaScript frameworks is discussed in Chapter 5.

Design and development

The structure and implementation of the artifact is defined beforehand. The course and weekly exercise structures are designed before implementing them and after that, the automatic assessments are developed before the start of the course. The process is, however, iterative, as the exercises get fixes and improvements throughout the course and any errors found in the automatic assessments get fixed as they are found. Additionally, student feedback is constantly considered, and the exercises see constant changes as per the feedback given by the students, if it is seemed fit.

Demonstration

The artifact is demonstrated in real-world use as the *Introduction to Web Programming* course including the weekly exercises and automatic assessment.

Evaluation

The artifact is constantly evaluated by multiple factors, including continuous student feedback, student performance and student attendance.

The artifact gets evaluated by students as the course progresses. The students share their opinions on the weekly exercises and the assessment system. The clearness of assignment descriptions and automatic assessment feedback gets mostly evaluated by this factor.

Another factor of evaluation is the student's performance: how many times did a student need to submit an assignment before getting a satisfactory number of points, and the mean points of every week's assignments. The difficulty of the assignments gets evaluated by this factor, and ideally the difficulty should rise as the weeks progress.

In addition to constant evaluation, a survey is to be conducted at the end of the course. In this survey students can give numerical and verbal feedback on how well the AAT tools and weekly exercises performed.

Communication

This thesis is used to communicate the findings of this research. It will be published in an academic platform where it will reach its target audience.

4. CodeGrade and Cypress

This chapter covers what is required from the testing environment from the course's standpoint. The selected testing environment is also presented, and their relevant features are discussed. The reason why the testing environments are suitable for the course is also briefly discussed.

4.1 Requirements for the testing environment

The tasks of programming courses that utilize command line interfaces (CLIs) are trivial to test in most cases. Almost every test can be conducted as input/output testing where the AAT only inputs some values to the program and expects a pre-determined output. In contrast, web programming needs a much wider spectrum of features from the AAT.

The website can have several elements with different properties and attributes, and these properties and attributes and even elements can change during the lifecycle of the web application. These elements also have different styling determined by CSS rules and these style rules can change during the lifecycle of the application. The website can also fetch data from certain servers that respond with data depending on the request. The request can also have a predefined body with POST requests, without which the server cannot respond properly.

To test all these features an end-to-end test must be conducted. This is easiest done using an end-to-end testing framework. The framework that was chosen for the course was the same used in *Web applications* – Cypress. CodeGrade is used in addition to Cypress to supply pedagogical features, such as Learning management system (LMS) integration, in order to grade the students automatically.

CodeGrade and Cypress offer various features relevant for the proper assessment and grading of the programming assignments. The next section covers what the tools offer and how they are relevant for the course.

4.2 CodeGrade features

CodeGrade is an automatic assessment tool and a blended learning application created in the University of Amsterdam. It supports various features that ease the grading process from teacher's and student's perspective. These features include automatic and manual assessment, automatic and manual feedback, plagiarism detection, code editor and executor and integration to LMS (learning management system). [16]

Automatic assessment

With CodeGrade, it is possible to automatically assess coding exercises. CodeGrade's automatic assessment system runs in a virtual machine that runs on Ubuntu (18.04.2 LTS) [17]. There are several ways to test a program in CodeGrade. In **I/O test** (input/output test), the system inputs a predetermined input to the program and expects a predetermined output. **Run program** -test runs a command and checks for a successful completion. **Code quality test** runs a static code analysis on the submitted code. **Capture points** -test runs a custom grading script that outputs a float between 0.0 and 1.0 which is used to give the student appropriate score. **Checkpoints** are special types of tests that passes and executes tests after it only if the tests before the checkpoint pass. [18]

Unit tests are test programs that usually utilize some form of testing frameworks that are uploaded to the file system and executed with a command. There are several built-in wrapper commands that install a testing framework. These built-in wrapper scripts are made for common testing frameworks such as Mocha, Jest, Junit 4 and 5 and Pytest. It is also possible to write own wrapper commands to run other testing frameworks, such as Cypress in this course's case, as long as it outputs a Junit XML file and works on Ubuntu (18.04.2 LTS). [19]

It is also possible to configure CodeGrade AutoTest features. It is possible to set the timeout, after which the test will automatically fail. It is also possible to allow the AutoTest to access the submission metadata, which includes, for example, submission date and submission

deadline. A very important feature for the course is, however, the ability to turn internet access on and off. Reasons for this is covered in Chapter 4.3.

Manual and automatic feedback

CodeGrade feedback can be divided into two categories: manual feedback and automatic feedback. Manual feedback can be done either as inline feedback for certain sections of code files, feedback for whole files or general feedback for the whole assignment.

Inline feedback can be written manually on any submitted code file on any line. It can be used to give precise feedback on any section of code the grader wants to comment on. If the submission is not a code file, the grader can give feedback for the whole file. On large submissions, such as projects, general feedback can be used. It supports markdown and prewritten snippets, that can be used to make grading tables or matrices [20]. It is also possible to give peer feedback in CodeGrade if the option to do so is enabled [21].

Automatic feedback is produced through automatic tests. It is generated by the software or testing framework through standard output streams and framework-specific logging functions and errors.

Plagiarism detection

CodeGrade offers plagiarism detection on submitted code and files. It supports the most common programming languages such as Java, JavaScript, Python and C++, but it also supports files without any code such as documentation files. The system also has several options to tweak, such as minimum similarity before a case is considered plagiarism, included file extensions, inclusion of old submissions and a base code that is going to get excluded from the plagiarism detection if the students are using a common library, for example. [22]

Built-in editor

In addition to submitting code files to turn-in an assignment, it is also possible to code the assignment straight within CodeGrade on the cloud. This eliminates the need to set up a local

coding environment that could be challenging for first year students that are just getting into programming. This also eliminates the task of building a file structure, as teachers can prepopulate the assignment with files that the students just need to edit. This also helps to guide inexperienced programmers and makes the subject, such as a new programming language, more accessible.

LMS integration

As a blended learning application, CodeGrade offers the ability to integrate to various LMSes, such as Moodle, Canvas or Brightspace [23]. This helps to track the progress of students since the points from CodeGrade automatically sync to Moodle. This makes it easy to manually grade assignments in addition to automatically doing so. Some options, such as deadlines however do not sync with CodeGrade and they need to be manually set in CodeGrade for students to be able to submit assignments [23].

4.3 Cypress features

As *Introduction to web programming* is a web programming course, the nature of the assignments requires a more complex testing environment that is capable of testing a variety of features other than a predetermined string in standard output stream. These features include whether certain user interface elements exist, whether they have the required functionality and whether network operations, such as API calls work. This is where Cypress testing framework comes into play.

Cypress is a front-end end-to-end testing framework [24]. It has various features, most notably being capable of simulating user input, controlling network traffic, testing document object model (DOM) elements and structure and the ability to alter how tests are conducted with commands and plugins [24]. This section will go over the most relevant features of Cypress regarding this course.

Testing DOM elements

Every website created during the course consists of HTML elements. For this reason, in order to even begin testing the requirements of the assignments, the testing environment must be capable of finding the elements and test their attributes and features. In cypress this is achieved with various commands.

A usual workflow with Cypress starts with checking whether an element exists. The element can be searched with jQuery selectors [25]. The element can then be saved to an alias if it is needed later. After the element is verified to exist, various commands to check the attributes can be chained, such as checking its classes, CSS rules or child elements.

Simulating user input

Nearly every assignment requires functionality and interactivity of some sort. For this reason, the ability to interact with elements is expected from the testing environment. In addition to asserting the elements and their attributes of the DOM tree, it is possible to interact with HTML elements with Cypress.

It is possible to simulate user actions in Cypress by firing the events a browser would fire by clicking the element using various commands. This means that Cypress does not actually click somewhere on the screen, so the clickable element must first be queried before it is possible to simulate a click event. It is also possible to simulate user typing text and inserting files, which are useful for filling out forms as seen in Figure 2. [26]

```
cy.get('#input-username').clear().type("Username");
cy.get('#input-email').clear().type("example@test.com");
cy.get('#input-password').clear().type("password(strong)");
cy.get('#input-admin').check();
cy.get('#submit-data').click();
```

Figure 2: Filling a form with Cypress

The figure shows how element interaction is done in cypress. First, the right element is queried using jQuery selectors. In this case, the HTML ids specified in the assignment description are queried. Then, the appropriate action is done to the elements depending on their type. The first three elements are text fields, the fourth element a checkbox, and the last element a button.

Controlling network traffic

Nearly half of the assignments make use of HyperText Transfer Protocol (HTTP) requests to different application programming interfaces (APIs). If every submission had access to internet, it would be possible for the API provider to temporarily revoke CodeGrade's access to the API rendering the tests useless. Internet access could also lead to more plagiarism, since the students could host a script file that others could include in their projects.

Cypress offers the ability to intercept HTTP requests. This works even if the program is not connected to the internet making the program and testing environment run completely locally. In addition to intercepting, Cypress can wait for the completion of HTTP requests to assert that certain requests are made before continuing the test, as seen in Figure 3.

To run tests without internet access, some fixed data needs to be sent back to the client from the intercepted request. This can be done in Cypress using fixtures, which are files with a fixed set of data that can be used by all tests. This also has the added benefit of always having the same data, thus making sure that the test works even if the original data in the API changes. [27]

```
1  cy.intercept('https://statfin.stat.fi/PxWeb/sq/4e244893-7761-4c4f-8e55-7a8d41d86eff', {
2      fixture: "week3/population.json"
3  }).as('population');
4  cy.visit("/");
5  cy.wait('@population');
6  cy.get('table tbody tr').should('have.length', 10);
```

Figure 3: Intercepting a network request with Cypress

Figure 3 shows an example of intercepting a network request to StatFin API. The fixture property is a JSON-file containing dummy-data, which is similar to the data found in the

actual API, but with different values to prevent hard-coding or bypassing the assignment specification. The as-command saves the link to an alias, in this case *population*. This makes programming the tests easier, but makes using the tests harder, because if the student's program fails to connect to the URL, Cypress, and thus CodeGrade, will just state that the program failed to connect to route *population* as seen in Figure 4.

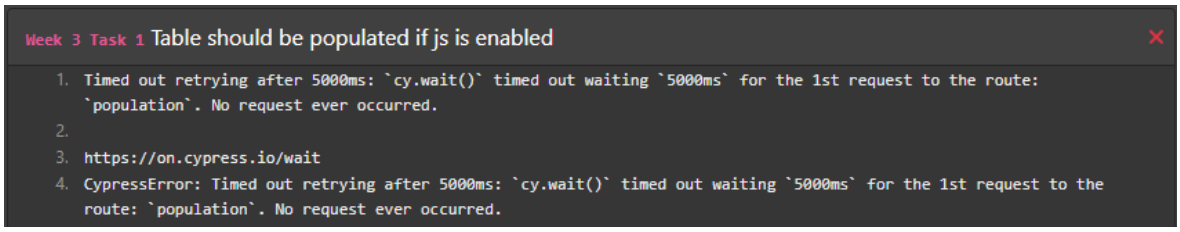

Figure 4: Cypress error message upon failing a request shown in CodeGrade

Configuration and plugins

Cypress is also widely configurable with numerous options to alter the behavior of the testing environment. For example, by default the tests run only once, and automatically fail if the testing environment runs into an error that originates from the program being tested. This can be avoided by altering Cypress' behavior upon detecting a raised exception. [28]

It is also possible to alter Cypress via plugins. Plugins allow the user to modify the internal workings of Cypress, and its node process. This allows the user to write custom code that runs on different parts of the Cypress lifecycle. [29]

5. Introduction to web programming

The main reason why *Web applications* was divided into two different courses was the intensive start of the course leading to a steep learning curve. The weekly exercises of *Web applications* consisted of 12 weeks, of which only 3 covered the basic topics – the same topics the whole *Introduction to web programming* focuses on. In the fourth week the focus shifted to back-end programming, which introduced a whole new dimension to web programming.

This intensive start was a reason why many students ceased to submit anything during later weeks. During weeks 1-2 on average 80% of enrolled students submitted the weekly exercise. On week 3 and onwards an average of 50% enrolled students submitted the weekly exercises [12].

The structure of the course was important to consider in order to make the difficulty level more gradual. In addition, the assignments need to have an appropriate level of difficulty in order to be challenging without being overwhelming. The assignments must also be able to be tested using the tools discussed in Chapter 4.

5.1 Requirements for the course

Introduction to web programming focuses only on the client-side front-end code that is run on the browser. Only the essential languages, HTML, CSS and JavaScript are covered. Therefore, languages such as TypeScript, and front-end frameworks that require a development server to be set-up and which offer more advanced features, such as React, Angular or Vue are not covered.

Web APIs are becoming increasingly common, with over 22000 public APIs being available for developers to use [30]. Many websites communicate with these web APIs in the form of Hypertext Transfer Protocol (HTTP) GET or POST requests. For this reason, HTTP GET, and POST requests are covered thoroughly. HTTP requests are usually sent to a Representational state transfer (REST) server, that works as an API. The communication

with these types of servers is done utilizing JSON data format. For that reason, JSON is in a crucial role during the course.

JavaScript has a multitude of different libraries that can be used on the front-end. They can be included in the application code either by downloading, adding them to the project structure and importing via JavaScript, or with an HTML script tag fetching from a geographically distributed network of content providing servers called Content Delivery Network (CDN) [31]. The course aims to get students familiarized with a few different frameworks to get familiarized in different APIs.

The course assignments need to also be testable using an AAT and follow the limitations of the used technologies. The tests must always work and have the same expected output. Therefore, the assignments cannot have any elements of randomness in them. Also, the assignments cannot have any visual testing, such as animations or certain pictures in them, since those are not possible to test with Cypress. Lastly, certain CSS selectors, like hover, cannot be tested without plugins because Cypress only triggers the JavaScript events and not CSS events.

5.2 Course structure

The structure of *Introduction to web development* was created with the goal in mind of avoiding the pitfall of having too many topics in too little time. The structure of the course is progressive, building new knowledge upon old knowledge. With this approach, the fundamentals are rehearsed continuously throughout the whole course.

The course also utilizes flipped classroom teaching, starting new topics with quizzes which test how well the student has studied the theory of the topic [32]. The programming tasks start with smaller and easier exercises that are faster to complete, and gradually rise in difficulty in order to test how well the student can apply their knowledge into practice.

5.2.1 HTML and JavaScript

During the first two weeks, the course has its focus on HTML and JavaScript. They are the foundation of all websites. HTML is used to create the basic structure of the website, whereas JavaScript is used to make websites have dynamic behavior, such as creating objects or creating HTTP requests. [33]

The assignment during the first week is simple enough to make the subjects easily approachable. The assignment consists of simple tasks, such as creating an HTML element, printing text to console, making DOM changes (changes to the website structure) and creating new elements. These tasks can be accomplished by using only few lines of code and little effort, in order to make the new technologies easily approachable.

The second week assignment assumes that the students are already familiar with the HTML and JavaScript environment. Therefore, the tasks are more advanced than in the first week. The second week assignment requires the students to use some of the more commonly used HTML elements, such as tables, and more advanced JavaScript concepts, such as loops and different types of element queries.

5.2.2 CSS and requests

Later in the course the students will also be implementing styling to their websites with CSS. CSS is a language used to describe the style of the HTML elements of the website [34]. In week 3, the students are required to make a table, that holds municipal data. The table is styled with CSS using different selectors.

As mentioned previously, web APIs are becoming more and more common. Therefore, one of the main topics of the course is making HTTP requests. In week 3 and onwards, the assignments include making HTTP requests to different web APIs. Therefore, in week 3, the data for the table is fetched from Statistics Finland (Statfin) API.

Week 4 introduces web fonts, responsive design, and Uniform Resource Locator (URL) query parameters. The week’s assignment is to create a TV show viewer website, that has a

search function. After the user searches for shows, the website connects to the TvMaze API using the search word as a URL query parameter.

The website will also use responsive design, which means that the website adjusts itself depending on the size of the screen it is viewed with. This is done using CSS media queries. The week introduces students to web fonts, which are fonts that are not preinstalled on the end-user's browser [35]. An example of such font is Roboto. The fonts need to be downloaded to the user from a server, usually from a CDN. Week 4 also briefly covers the usage of the Bootstrap front-end framework, which is used to easily make good-looking websites with pre-styled components [36]. It is also usually imported from a CDN.

5.2.3 Frameworks

Weeks 5 and 6 focus on different frameworks and HTTP requests. The fifth week focuses on Leaflet interactive map framework, as well as GeoJSON – a specially formatted JSON used to display geographical data.

> "GeoJSON is a format for encoding a variety of geographic data structures using JavaScript Object Notation (JSON) [RFC7159]. A GeoJSON object may represent a region of space (a Geometry), a spatially bounded entity (a Feature), or a list of Features (a FeatureCollection)." [37]

The assignment consists of fetching geographical data of the municipalities of Finland and coloring them according to their net migration. The map also will have some interactivity to it: if the user hovers his cursor over a municipality polygon, the name of the municipality should show up. If the user clicks on one of the municipality polygons, the positive and negative migration statistics should show up.

Week 6's assignment is built around the chart-drawing framework called frappe-chart.js. Additionally, the data shown in the chart is fetched using HTTP POST requests. The first two tasks focus on sending a static POST body to the Statfin API. On the subsequent tasks the POST body is modified by the user input to fetch different data from the API. The week also covers how navigation is implemented in HTML and JavaScript.

Week 7 will introduce the canvas element and an HTML5 game engine and JavaScript library *Phaser*. The goal of week 7 is to create a simple video game without strict assignment instructions to let students express their creativity and to learn a little game design.

5.3 Testing caveats

Most of the weekly exercises of the course are trivial to test. As seen before, Cypress is an excellent tool for querying and interacting with DOM elements, which is all that is required for weeks one and two. It was also learned in Chapter 4 that it is possible to intercept network requests and provide a fixture in Cypress, which eliminates the need for internet connection as it is disabled for various reasons explained in the same chapter. This makes it possible to test assignments with API requests and eliminates the possibility of the data changes in the API ruining the tests.

The frameworks selected for the weekly exercises are also suitable for automatic assessment. Leaflet draws scalable vector graphics (SVG) polygons based on the GeoJSON data, and the attributes, which determine the shape of the polygon can be tested. This can be used to determine whether the students have assigned the data correctly.

A major factor in the selection between Chart.js and Frappe-chart.js framework as the second JavaScript framework was the fact that Frappe-chart.js uses a div element in which an SVG chart was drawn. Chart.js, on the other hand, draws the chart on a canvas component, which uses raster images. It’s not possibly to feasibly test canvas elements with Cypress, so it would have been impossible to assess whether students would have mapped the chart correctly. Instead, with Frappe-charts.js the position of the data points can be tested and compared to pre-determined values.

Due to the nature of the week 7 exercise, it is not feasible to test automatically. *Phaser* draws the video game area on a canvas element, which, as stated before, cannot be tested with Cypress. Additionally, the exercise is designed to let students be creative, which would not be possible with the constraints of automatic assessment.

5.4 Programming environment

As Cypress requires the website to be hosted, some sort of live server needs to be set up on the CodeGrade system before the tests can be conducted [38]. For this reason, CodeSandbox was chosen as the main programming environment for this course. CodeSandbox is a cloud web-development platform with real-time rendering [39].

Codesandbox has its own built-in execution environment, removing the need for the students to set-up a development server [39]. CodeSandbox projects come bundled with a package.json file, which contains the dependency for parcel [39]. Parcel is a Node.js package, which builds a website and hosts it on a local server [40].

As students develop the weekly exercises on CodeSandbox, and submit the submission containing package.json with Parcel to CodeGrade, the virtual machine on CodeGrade will start a local server that hosts the student's website for Cypress.

This environment was chosen over a local programming environment, because setting up a server to host a website locally is out of this course's scope and more in-line with *Advanced web applications*.

6. Discussions

This chapter discusses how well the course was received by students, and its improvements and disadvantages compared to the previous course analyzing the course feedback. The viability of CodeGrade and Cypress is also further dissected and whether they are suitable tools for an introductory web programming course. Lastly, future work and the limitations of this study are also discussed.

6.1 Improvements and disadvantages

To accurately assess the improvements and disadvantages of the course, a course feedback survey was conducted. The survey had 34 respondents out of the 69 enrolled students, which is about half of the students. However, only 52 students submitted a final project, so it is safe to assume that the course had only 52 active students out of the 69. This bumps the response rate among active students to just over 65%, which is nearly two-thirds. Based on the rather large percentage of respondents, the feedback received can be considered as conclusive for the sake of this research.

Based on the survey, the course was overall well received, and it promoted learning. In the course survey, the students could answer from 1 – not at all, to 5 – very well with 3 being the average. Based on this, it can be considered that if a topic got an average score of over 3, it was understood by the students. Therefore 3 can be used as a reference number when evaluating each topic.

The students thought that the course promoted their learning all-around, as seen in Figure 5. The mean to the question *How well did you understand the following topics* was 3.68, which can be considered good. The only topic that got an average score below 3 was AJAX, which got an average score of 2.9. This can be explained by the fact that while the term was exercised extensively through practice by using the fetch API, the term itself was never used in the weekly exercises.

Compared on the previous course, *Web applications*, the intersecting topics, such as JavaScript, HTML and CSS were not understood quite as well, and the mean of the topics

was lower than in *Web applications*. This can be explained by the length of the courses. *Web applications* lasted twice as long as *Introduction to web programming*, so students had more time to comprehend the basic topics used thorough the courses.

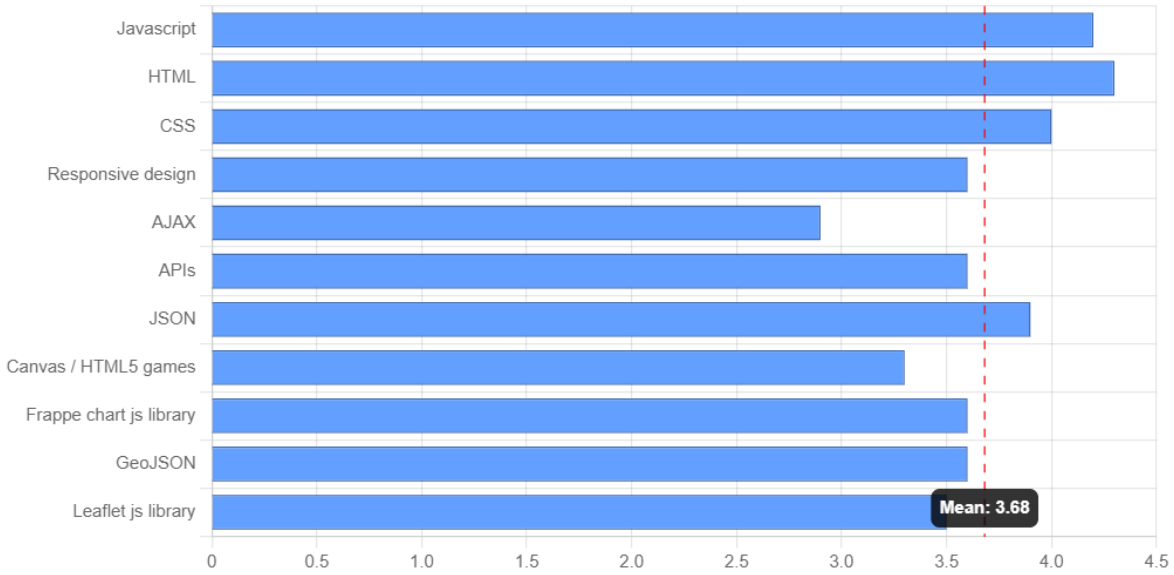

Figure 5: Graph illustrating the score distribution of the question "How well did you understand the following topics?" on the course feedback survey

6.2 Assignments

The assignments were well received by the students. The students found the exercises interesting and enjoyable, and they found them to be challenging enough. The only problem the students had with the exercises themselves was the fact that it was possible to gain enough weekly exercise points to pass the course from weeks 1-3 while not acquiring enough knowledge to be useful for the final project.

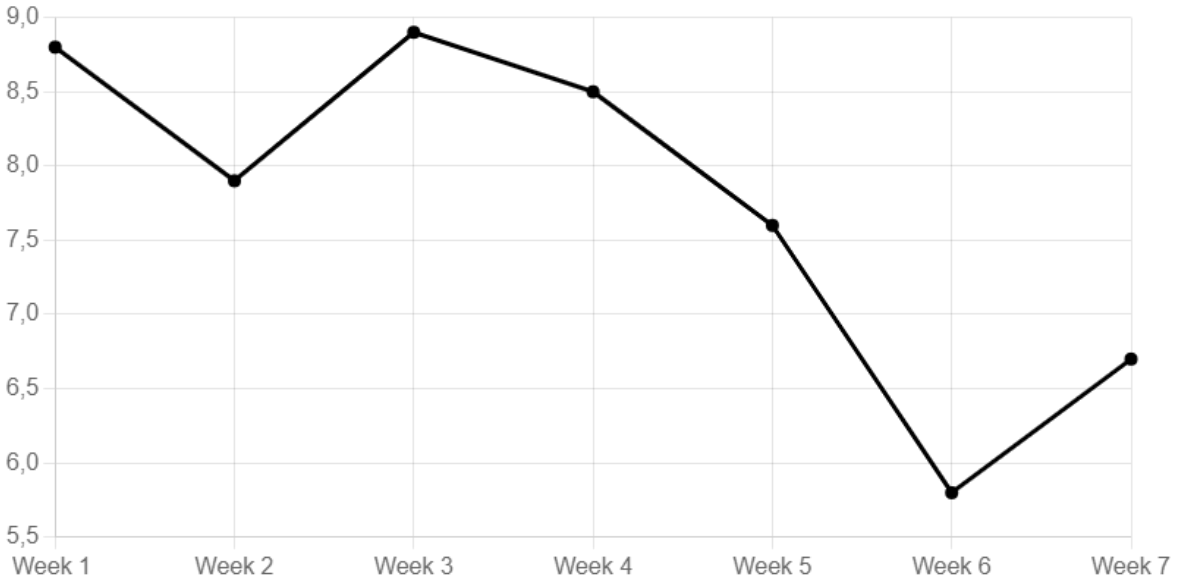

Figure 6: Graph showing the average score of students for each week's assignment

As for the performance of the students, the students seemed to be motivated thorough the course. The average score was smaller each week, which is expected as the assignments increase in difficulty. However, as seen in Figure 6, weeks 2 and 6 proved to be statistical anomalies as the average score of those weeks were unusually low.

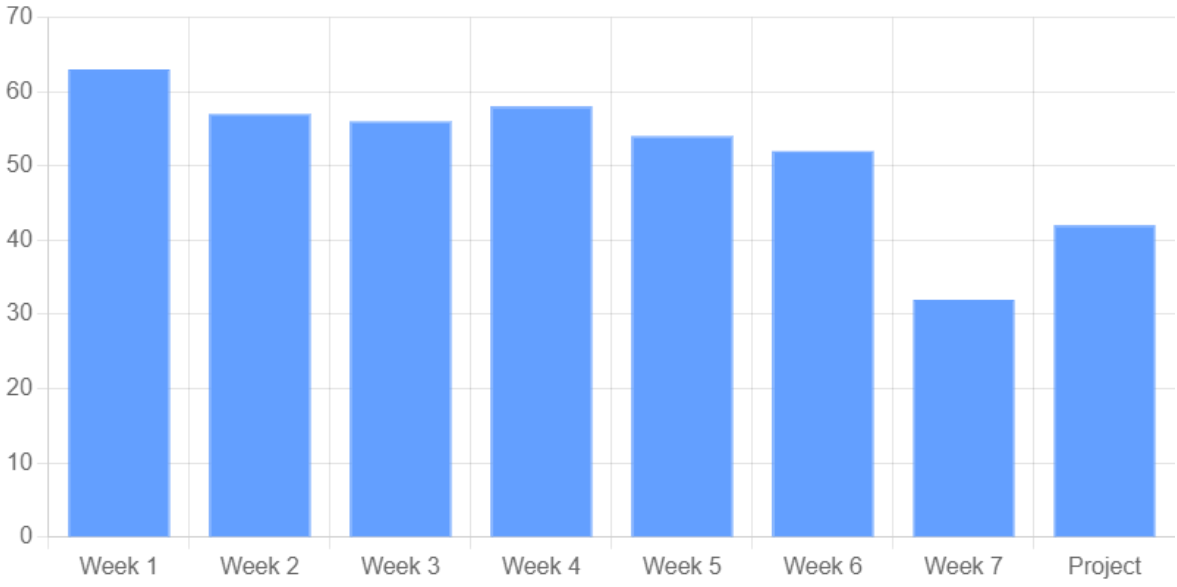

Figure 7: A graph showing the number of students who submitted at least 1 submission for each week

As Figure 7 shows, the number of students who submitted an exercise stayed relatively constant throughout the course. However, week 7 saw a major drop in the number of students who submitted. This can be explained by the fact that the project and week 7 were announced simultaneously, forcing the students to decide whether to focus their efforts fully on the project or share their efforts between week 7 exercise and the project. Another possible reason why week 7 saw such a drop in submissions was that it was possible to gain maximum weekly points from weeks 1-6 alone.

6.3 CodeGrade and Cypress

Overall, the reception of CodeGrade was varied. To the question *How useful did you consider the following events?* CodeGrade got an average score of 3.3, which can be considered good. On the contrary, the question *How hard did you find the following events?* CodeGrade got an average score of 3.5. On this question, the greater the score, the harder the students found the given subject, therefore an average score of under 3 would have been ideal.

The mixed reception of CodeGrade can be explained by various factors. First, the file submission system was seen as counter-intuitive by some students, and it's somewhat different compared to other courses such as CS1. The Cypress framework expects the website being tested to be served in a local server, and the project needs to be sent in a zip-file. However, being a front-end course, the students are not expected to have the knowledge of setting up the server, hence the usage of CodeSandbox. CodeSandbox bundles parcel package, which builds the project files and serves them on a local server. The project is also expected to be zipped without leading directories, which makes submitting with a system using macOS harder, since it zips the project with a folder *__MACOSX* which needs to be removed manually.

In addition, the feedback provided by Cypress was seen too vague in some cases. In most cases, Cypress' automatically generated feedback was used, but in some cases custom feedback was provided if the test fails. In addition to vague feedback, another problem with the AAT was that the tests would sometimes not follow the instruction due to human error, or the instructions were seen too unclear.

The usefulness of the feedback provided by the AAT also varied heavily depending on the skill level of the student. In most cases, the AAT provided a stack trace, which indicated the reason why the test failed and the position of the section of code that failed. This proved to be difficult for some students to understand, and they needed guidance on finding the issue in their source code. Furthermore, a lot of the tests that compared strings from the student's submissions to pre-determined specifications using regular expressions. These regular expressions were also shown in the test feedback, and this seemed to cause problems with students unfamiliar with regular expressions, as they could not understand what kind of output the test required.

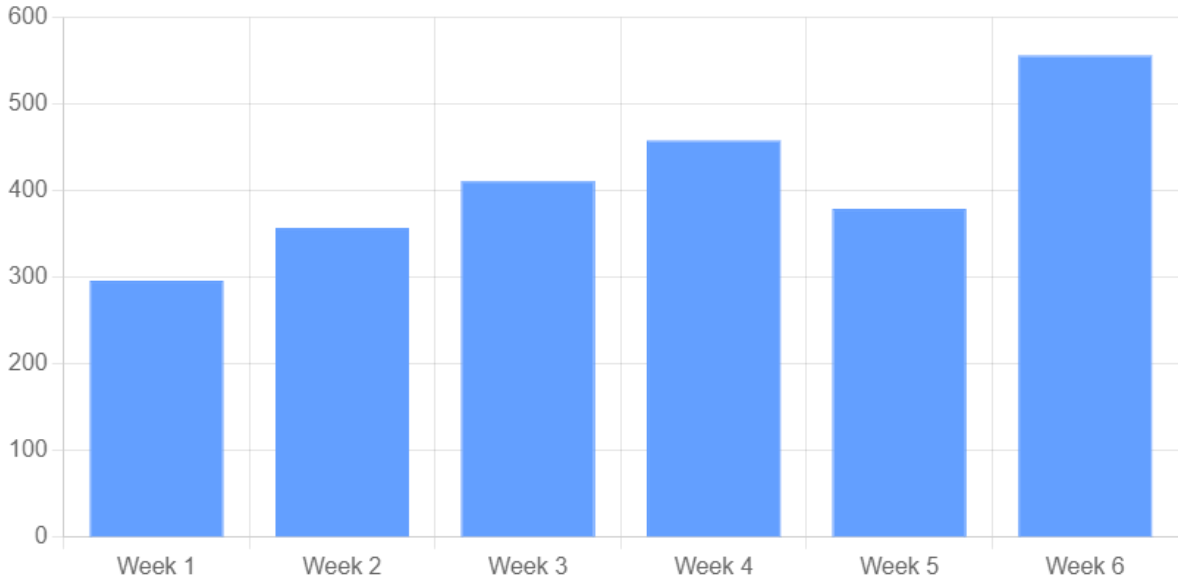

Figure 8: Total amount of submissions of each week

The sheer number of submissions made it impossible for one teaching assistant to manually assess them all in a reasonable time. As seen in Figure 8, the total amount of submissions of each week was in the hundreds. Week 1, being the easiest week still had 296 submissions, and week 6 being the most difficult one had 556 submissions. The total amount of unique submissions for weeks 1-6 was 2457. This proves that an AAT such as CodeGrade is mandatory for a programming course this size.

6.4 Reflections of the research questions

In the introductory chapter, the research questions of this thesis were presented. The research questions were the following: How to construct new exercises for the upcoming web programming course? How to make the difficulty curve less steep? How to implement CodeGrade with Cypress to automatically assess the programming exercises? This section gives answers to the research questions.

The assignments were constructed in a way that each week had a maximum of one or two new topics. Additionally, the assignments were designed to accumulate knowledge on top of already learnt subjects. This approach had the effect of easing out the learning curve, as the course didn't see a sudden drop in active students as seen in Figure 7.

The automatic assessments were implemented similarly to the previous *Web applications* course. Cypress was used as the testing framework, and CodeGrade was used as an automatic assessment tool and as an environment for submitting the exercises. The features of Cypress were suitable for testing the exercises of weeks 1-6, but week 7 and the course project had to be tested manually.

Based on the average grade of each week and the number of students who submitted at least one submission, the assignments worked well, and the difficulty of each week increased linearly. The course also didn't see a massive drop in active students as was seen in the previous *Web Applications* course.

6.5 Future work

What students found most problematic were the ambiguous assignment descriptions. Therefore, the single biggest improvement for the course would be to rewrite the assignment descriptions and fix all the logical and spelling mistakes. It would also be beneficial to provide some examples to the assignment descriptions, especially for assignments with CSS styling.

As for the AAT, all the automatic tests should be tested to find any logical mistakes, and the feedback should be made clearer, as it should give the student the exact problem with the submission, and not give excessive information about the failed test.

Furthermore, the programming environment and submission system should be revised, as it arose problems. The submission instructions should be made simpler and clearer, and the parcel package system should be replaced to a system where CodeGrade has a live server which serves the students submissions for Cypress to assess. The parcel building process takes some time and degrades the user experience of CodeGrade. It also unnecessarily complicates the submission process.

If the live server will be set up to CodeGrade in the future, the programming environment should be changed as well to Visual Studio Code. CodeSandbox had some problems being slow and sometimes stopping working if student accidentally changed the configuration. Another reason supporting the change to another platform is student preference. The survey indicated that roughly half of the students used Visual Studio Code as a programming environment, which was also the environment showed in the lecture videos.

The effects of the division of *Web applications* should also be evaluated using the upcoming *Advanced web applications* course. It should be investigated, if the upcoming course will see a similar drop in active students as *Web applications* did, as the same advanced topics are now spread out to a bigger timeframe, and the students have already acquired knowledge of JavaScript, HTML, and front-end programming from the introductory course.

6.6 Limitations

The artifact of this study is a new introductory web programming course, which is a first of its kind in LUT, so it is impossible to evaluate how it compares to a previous course, since no previous data is available. The closest candidate would be the previous *Web applications* course, which was twice as long, meaning students had more time to acquire relevant knowledge of most of the topics covered in *Introduction to web programming*.

Another limitation of this study is that it does not take into account any of the lecture materials of the course, as they have a big impact on the students' learning process as

opposed to the weekly exercises that test whether the students have learned the weekly topics.

7. Conclusions

This thesis studied about creating weekly exercises to an introductory web programming course and creating the weekly tests to it using CodeGrade AAT and Cypress testing framework. The study was done using design-science research methodology. The aim for this study was to create a working course, and to ease the learning curve of the previous *Web applications* course. Additionally, this study investigates how the weekly exercises are constructed, and how Cypress and CodeGrade are implemented and what tools they offer.

The study was evaluated by gathering continuous feedback from students. A feedback questionnaire was also conducted at the end of the course. The performance and attendance of students was also used to evaluate the weekly exercises. Based on the student feedback and performance, a working course was achieved. CodeGrade and Cypress also proved to be adequate tools in this environment.

The course and the exercises were largely well-received. The main problem with the exercises was the feedback provided by the AAT tool. The course questionnaire score of the topics that the course shares with *Web applications* course were also compared. The findings of this research are in line with previous findings. The more experienced students got more out of the automatic feedback the AAT provided than more inexperienced programmers.

The study was limited by the fact that the course was one of its kind and could not be compared to a previous introductory programming course. The closest candidate for this, however, was the first three weeks of the previous *Web applications* course. Another limiting factor was that the study only focused on the weekly exercises, leaving the lecture materials out of the scope of this study.

In the future, the assignment descriptions should be checked carefully to fix any mistakes. Additionally, it should be considered to make the exercise of week 6 easier to make the average score of week 6 more in line with the previous weeks. Also, the feedback of the AAT should be rewritten to give out more information, and the route names should be declared more clearly. The programming environment also needs to be reconsidered for future implementations of the course.

References

[1] S. Gupta, “Automatic Assessment of Programming assignment,” presented at the The First International Conference on Information Technology Convergence and Services, Jan. 2012, pp. 315–323. doi: 10.5121/csit.2012.2129.

[2] C. Douce, D. Livingstone, and J. Orwell, “Automatic test-based assessment of programming: A review,” *J. Educ. Resour. Comput.*, vol. 5, no. 3, Sep. 2005, doi: 10.1145/1163405.1163409.

[3] T. Delev and D. Gjorgjevikj, “E-Lab: Web Based System for Automatic Assessment of Programming Problems,” *ICT Innovations*, p. 10, 2012, doi: 10.13140/2.1.3025.9206.

[4] T. Rutanen, “Automatic Assessment of Programming Exercises,” Aalto University, 2021. Accessed: Aug. 18, 2022. [Online]. Available: http://urn.fi/URN:NBN:fi:aalto-202106207467

[5] Z. Ullah, A. Lajis, M. Jamjoom, A. Altalhi, A. Al-Ghamdi, and F. Saleem, “The effect of automatic assessment on novice programming: Strengths and limitations of existing systems,” *Computer Applications in Engineering Education*, vol. 26, no. 6, pp. 2328–2341, 2018, doi: 10.1002/cae.21974.

[6] H. Fangohr, N. O’Brien, A. Prabhakar, and A. Kashyap, “Teaching Python programming with automatic assessment and feedback provision,” Sep. 2015, doi: 10.48550/ARXIV.1509.03556.

[7] V. Pieterse, “Automated Assessment of Programming Assignments,” in *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*, Heerlen, NLD, huhtikuu 2013, pp. 45–56.

[8] J. Liebenberg and V. Pieterse, “Investigating the feasibility of automatic assessment of programming tasks,” 2018, doi: 10.28945/4150.

[9] K. Masters, “A Brief Guide To Understanding MOOCs,” *The Internet Journal of Medical Education*, vol. 1, no. 2, Dec. 2011, doi: 10.5580/1f21.

[10] T. Tenny, “Program readability: procedures versus comments,” *IIEEE Trans. Software Eng.*, vol. 14, no. 9, pp. 1271–1279, Sep. 1988, doi: 10.1109/32.6171.

[11] L.-C. Cheng, W. Li, and J. C. R. Tseng, “Effects of an automated programming assessment system on the learning performances of experienced and novice learners,” *Interactive Learning Environments*, pp. 1–17, Nov. 2021, doi: 10.1080/10494820.2021.2006237.

[12] A. Järventausta, “Piloting a new automated assessment tool in the Web Applications course,” 2022. Accessed: Jul. 03, 2022. [Online]. Available: https://lutpub.lut.fi/handle/10024/163918

[13] M. Rubio-Sánchez, P. Kinnunen, C. Pareja-Flores, and Á. Velázquez-Iturbide, “Student perception and usage of an automated programming assessment tool,” *Computers in Human Behavior*, vol. 31, pp. 453–460, Feb. 2014, doi: 10.1016/j.chb.2013.04.001.

[14] J. R. Venable, J. Pries-Heje, and R. L. Baskerville, “Choosing a Design Science Research Methodology,” *ACIS 2017 Proceedings*, Jan. 2017, Accessed: Mar. 07, 2022. [Online]. Available: https://aisel.aisnet.org/acis2017/112

[15] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management*

Information Systems, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.
[16] CodeGrade, “What is CodeGrade? — CodeGrade QuietStorm.1 documentation.” Accessed: Oct. 06, 2022. [Online]. Available: https://docs.codegra.de/about/whatis.html
[17] CodeGrade, “Websites.” https://help.codegrade.com/automatic-grading-guides/websites (accessed Nov. 10, 2022).
[18] CodeGrade, “Add Automatic Grading.” https://help.codegrade.com/setup-assignment/build-assignment/creating-automatic-tests (accessed Nov. 09, 2022).
[19] CodeGrade, “Creating Unit Tests.” https://help.codegrade.com/faq/creating-unit-tests (accessed Nov. 09, 2022).
[20] CodeGrade, “Give Feedback.” https://help.codegrade.com/setup-assignment/grade-your-assignment/grading-and-giving-feedback (accessed Nov. 09, 2022).
[21] CodeGrade, “Giving Peer Feedback.” https://help.codegrade.com/for-students/advanced-features/giving-peer-feedback (accessed Nov. 09, 2022).
[22] CodeGrade, “Detect Plagiarism.” https://help.codegrade.com/setup-assignment/analyze-your-assignment/checking-for-plagiarism (accessed Nov. 10, 2022).
[23] CodeGrade, “LMS Integration — CodeGrade QuietStorm.1 documentation.” https://docs.codegra.de/user/lms.html (accessed Nov. 11, 2022).
[24] Cypress.io, “Why Cypress?,” *Cypress Documentation*. https://docs.cypress.io/guides/overview/why-cypress (accessed Nov. 17, 2022).
[25] “Cypress.$ | Cypress Documentation.” https://docs.cypress.io/api/utilities/$ (accessed Feb. 02, 2023).
[26] Cypress.io, “Interacting with Elements,” *Cypress Documentation*. https://docs.cypress.io/guides/core-concepts/interacting-with-elements (accessed Nov. 24, 2022).
[27] Cypress.io, “Network Requests,” *Cypress Documentation*. https://docs.cypress.io/guides/guides/network-requests (accessed Nov. 24, 2022).
[28] Cypress.io, “Configuration,” *Cypress Documentation*. https://docs.cypress.io/guides/references/configuration (accessed Nov. 24, 2022).
[29] Cypress.io, “Plugins,” *Cypress Documentation*. https://docs.cypress.io/guides/tooling/plugins-guide (accessed Nov. 25, 2022).
[30] L. Liu, M. Bahrami, J. Park, and W.-P. Chen, “Web API Search: Discover Web API and Its Endpoint with Natural Language Queries,” in *Web Services – ICWS 2020*, Honolulu, HI, USA, 2020, pp. 96–113. doi: 10.1007/978-3-030-59618-7_7.
[31] J. D. Gagliardi and T. S. Munger, “Content delivery network,” U.S. Patent No. 7962580B2, Jun. 14, 2011 Accessed: Jul. 04, 2022. [Online]. Available: https://patents.google.com/patent/US7962580B2/en
[32] A. Herala, E. Vanhala, A. Knutas, and J. Ikonen, “Teaching programming with flipped classroom method: a study from two programming courses,” in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, Koli, Finland, Mar. 2015, pp. 165–166. doi: 10.1145/2828959.2828983.
[33] “HTML: HyperText Markup Language | MDN,” Jan. 18, 2023. https://developer.mozilla.org/en-US/docs/Web/HTML (accessed Feb. 07, 2023).
[34] “CSS: Cascading Style Sheets | MDN,” Sep. 25, 2022. https://developer.mozilla.org/en-US/docs/Web/CSS (accessed Feb. 15, 2023).
[35] M. Nygren, *Web Font Optimization for Mobile Internet Users : A performance study of resource prioritization approaches for optimizing custom fonts on the web*. 2019.

Accessed: Jul. 04, 2022. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-85481

[36] M. O. contributors Jacob Thornton, and Bootstrap, “Get started with Bootstrap.” https://getbootstrap.com/docs/5.2/getting-started/introduction/ (accessed Aug. 08, 2022).

[37] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and S. Hagen, “The GeoJSON Format,” Internet Engineering Task Force, Request for Comments RFC 7946, Aug. 2016. doi: 10.17487/RFC7946.

[38] Cypress, “Installing Cypress,” *Cypress Documentation*. https://docs.cypress.io/guides/getting-started/installing-cypress (accessed Dec. 15, 2022).

[39] “Overview – CodeSandbox.” https://codesandbox.io/docs/learn/introduction/overview (accessed Feb. 15, 2023).

[40] “Building a web app with Parcel.” https://parceljs.org/getting-started/webapp/ (accessed Feb. 07, 2023).