University of Nevada, Reno

A Web-Based Application for Automatic Evaluation of Programming Assignments

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Aswathi Mohan

Dr. Frederick C. Harris, Jr., Thesis Advisor

August, 2017

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

ASWATHI MOHAN

Entitled

Web Based Application for Automatic Evaluation of Programming Assignments

be accepted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Dr. Frederick C. Harris, Jr., Advisor

Dr. Sergiu M. Dascalu, Committee Member

Dr. Jeffrey C. LaCombe, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

August, 2017

Abstract

The assessment of programming assignments are complicated and time-consuming. Also the number of students enrolling for programming courses is increasing tremendously. The instructors or graders have to dedicate more time to grade the assignment manually with tasks such as downloading source files, compiling and running the program, and entering the grade. This thesis presents an online assignment submission and automatic evaluation web platform for programming classes (Submit). This application allows instructors to create new courses, manage enrolled students for the course, create assignments, etc. The students are able to enroll in courses with a specific join token given by the instructor. Then the students are able to access the assignment and upload the program files for the assignment. They are also able to test the code against the public test cases created by the instructor. The source code compiles and runs automatically and compares the student output with the expected output created by the instructor. The student will get instant feedback for their submission such as output, whether the test is passed or not, what output was expected, and the difference between those two outputs. The instructors are able to grade the assignment and comment on the submitted program via a web interface. The application has been developed with Ruby on Rails and Flask and can utilize the features of both frameworks.

Dedication

I dedicate this thesis to my family and especially my husband who have encouraged and supported me throughout my journey.

Acknowledgments

I would like to express my sincere thanks to my adviser, Dr. Frederick C. Harris, Jr. and my committee members Dr. Sergiu M. Dascalu and Dr. Jeffrey C. LaCombe for their time and suggestions. I would like to thank Nolan Burfield for the initial work on the system that lead to the thesis. I am grateful to several of my colleagues for their passionate participation, inspiration, valuable suggestions, and support during the integration and testing and providing valuable comments on this research project. Finally I would like to express gratitude to my family for their support throughout my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The popularity of computer science courses is continually increasing. As a result, the number of students enrolling in computer science classes is also increasing. Computer science instructors assign many programming assignments to enhance the outcomes of the learning process by students, but the evaluation process of programming assignments is not straightforward. The assignment grading became a burden for instructors. It involves time-consuming steps like compiling and testing programs with various inputs. These kind of evaluations limit the number of programming assignments that an instructor can assign during a course period. One way of alleviating this problem is to automate the evaluation process. This thesis introduces an online assignment submission and automatic evaluation web platform for computer science programming classes. The goal of this project is to be able to easily create and rapidly evaluate student programming assignments, in a consistent manner with a low error rate.

The Submit application is capable of creating courses, managing enrolled students, creating assignments and their test cases, and automatic grading of students submission. The students can enroll in courses, view the assignments, and submit the solution for the assignments. The students are also able to test their assignment with the public test cases created by the instructor before submitting it. The system gives instantaneous feedback for the test with outputs and the difference in output as compared to instructor's expected output. The instructors/graders are able to view and grade the submission and are able to add comments on each line of the code.

Finally, the instructor can upload the grade which creates a pdf file with grades, grader's comment, and details of each test case such as input data, pass or fail the test cases. There is an option to download the assignment as a zip file and download the grades of students as an Excel file.

The application has been developed with Ruby on Rails and Flask. The front-end is designed with Rails which handles user's request for each page. The back-end is handled by both Rails and Flask. A MySQL database is used for storing user data and other information. The Rails App handles the user login, course creation, assignment creation, managing enrolled students, and assignment submission. The Flask App handles the generating expected output based on test cases created by the instructor and testing the student's submission. The Flask App compiles and runs the program automatically and compares the output with instructor's output and gives the difference between two.

This application reduces the effort and time needed to grade the assignment, so the instructor can include more assignments for their courses. Since the application runs on two different frameworks and on different systems, it can utilize the resources and advantages of both. This makes the application faster and more efficiently handles multiple users.

The rest of this thesis is organized as follows. Chapter 2 covers background of other related applications that are used for automatic assignment grading, frameworks, and libraries used for implementing the application. Chapter 3 discusses the design of the application, the software engineering functional and non-functional requirements, the use case model for the application, and a detailed design of the system architecture and database. Chapter 4 goes into detail of how the system was implemented. Chapter 5 goes over an application walkthrough of its functionality. Finally, the thesis wraps up in Chapter 6 with a discussion of the conclusion and the future work.

Chapter 2

Background and Related Work

2.1 Overview

The advancement in computer technology popularizes the computer science and engineering course. As a result, there are many people attracts into this field and many students are enrolling for programming classes. The instructors have to assign more homework to enhance the student's skill. This will increase the number of submissions. In this situation, the assignment grading became a burden for the instructors and graders. The automatic assignment evaluation can alleviate this problem to a certain level. This chapter will discuss other application developed to automate the assignment evaluation and features and improvements in our new application.

2.2 Related Work

The popularity of computer science and engineering attracts more people into this field and more people are interested in learning different programming languages. The increase in the number of students enrolling for programming classes will increase the effort for the instructor to manage the class. The instructor has to add more problems to enhance the student's skill, but the programming assignment grading becomes a burden for instructors or graders since it takes a lot of time. There are some applications available online and some Universities developed theirs on applications which focus on grading programming assignments for computer science classes and help graders to grade the assignment easily with less time spent on grading. A few

examples of such online assignment evaluation applications are WebCAT, Stepik, Instructure, Marmoset, and automatically grading programming homework platform developed by MIT, etc. One of the advantages of these systems is that the users get instantaneous feedback for their answers. Even if a user fails to pass the test cases for the problem at the very first attempt, the feedback inspires the students to re-attempt the problem until he reaches the correct solution.

Marmoset, developed by University of Maryland [25, 26],is a grading system for handling student programming project submission, testing, and review of the code. The system evaluates student submission against public test cases and test cases that are not visible to students. This system also takes a snapshot of a student's work every time and saves it in a file. This data is useful for studying how students learn programming languages. It works in different programming languages and is designed to work with both small and large projects. Currently, the instructors and teaching assistance of the University of Mary Land using this system but not available to public.

WebCAT is an open-source tool for automatically grading programming assignments [6, 32]. Along with the automatic grading, WebCAT uses a different approach to grade a student's code.It supports grading of the assignments where students are asked to submit their own test cases. It is a language independent tool that focuses on test driven development. This assessment approach is proposed by Stephen H. Edwards of Virginia Tech. But this system cannot be used for managing courses and students. It is only for assignment submission and evaluation. Also, students have to submit their own test cases.

The automatically grading programming homework platform developed by MIT can automatically identify errors in student programming assignments and suggest corrections [14]. This system will identify the minimum number of corrections necessary to get a program working rather than the way the student approached and solved the problem. This application provides specific feedback, including the line numbers of specific errors with recommended corrections. Currently, teaching assistance of the

University is using this system but it still on developing and testing stage and not available to public.

Instructure is an online software for managing courses [10]. This platform provides access for students to their courses, materials associated with the course, and a submission option. Instructors are able to create assignments, upload file, grade, etc. Students are able to view and submit assignment. But these applications are not able to automatically evaluate the submission as Submit does. This application is available online.

Submit is an online platform for assignment submission and evaluation for computer science classes. It is designed for evaluating programming assignments. This application is made for instructors, students, and graders. Instructors are able to create a course, manage students, create assignments and evaluate the assignment. The students are able to view the assignment, submit the source code for the assignment, and test the assignment against test cases defined by the instructor. Some test cases are visible to students and some are invisible. The instructors or graders can evaluate the assignment automatically and upload the grades. The student gets instant feedback for their submissions like the error in their outputs, comments for their submission, and grades. It is a simple and powerful platform for computer science programming classes.

The previous version of this application, 'Submit: An Online Submission Platform for Computer Science Courses' [18] is able to manage assignment submission and automatic evaluation of assignment and grading. The old system manages the course, grades assignments, handles files and provides grades and feedback. However, the system does not support multiple users at a time and cannot able to manage infinite loops and errors. This will break the application.

The new version of the Submit is able to handle multiple users and handle infinite loops and errors. Now the new application is developed on Ruby on Rails and Flask frameworks. Both the framework are configured in two different systems. So the application can utilize the features of two frameworks and resources of the two

systems. A celery queue is configured for scheduling jobs in Flask App. So the incoming request from the users are added to this queue and serves one by one. So the system can handle multiple users without breaking the application. The new system also handles the infinite loops and errors. The Flask App is responsible for creating instructor outputs and evaluating assignment against different test cases. If there any infinite loop or faulty code in the submission, the application stops the further processing of the code and send a feedback to the user. The feedback includes the message showing the error or exceeds max time limit for the process in the case of infinite loops. This application is now implemented with a RESTful API which can ensure the interoperability between the Rails and Flask framework configured on two different systems on the web. This REST API handles the incoming RESTful requests with data [1] and the project controller parses the data from JSON format and saves that into the database. It assures the security using defensive programming software design to continue functions in unpredicted situations.

2.3 Libraries and Frameworks

Submit depends on multiple libraries and frameworks. These include Ruby on Rails, Flask, Celery, services for networking, and user interface styling. These libraries and frameworks are listed in this section.

Rails is a web application development framework running on Ruby language which is designed to make programming web applications easier. It is an open source software and has an elegant and compact design which effectively creates a domain specific language for writing a web application. It allows us to write less code while achieving more than many other languages and frameworks. The common web programming tasks such as generating HTML, making data models, and routing URLs are easy with Rails, and the application code is concise and readable. Rails combines the Ruby programming language with HTML, CSS, and JavaScript to create a web application running on a web server. In this project, Ruby on Rails is used to develop the application's front end and back-end with Model View Controller design [24, 37].

Rails official website is very useful to develop the application on Rails. We have to install Ruby language 2.2.2 or newer before installing Rails and other dependencies. Tutorialspoint [27] and Codeacadamy [4] are very good websites to learn Ruby on Rails for beginners.

Ruby is a dynamic, object oriented, general purpose, open source programming language. It is very flexible and easy to read and write. This language was influenced by many other languages and developed as an alternative to scripting languages like Perl and Python. Ruby on Rails is developed with Ruby language. In this application development Ruby is used as the scripting language [23, 24]. The beginners can learn Ruby language from Codeacademy [4] which provides an interactive tutorial for Ruby.

MySQL database offers a multi threaded and multi-user structured query language (SQL). MySQL is designed to handle heavy loaded systems. The data is stored in structured tables, which can be accessed via multiple threads. The MySQL database was chosen for its robustness, easy setup, and it is using common language SQL [19]. We have to install MySQL server and client prior to application development and create tables in the database. MySQL documentation [19] and codeacademy [4] helped to learn SQL queries.

Hyper Text Markup Language (HTML) is the language that a web browser reads to render the display. HTML uses tags to describe to the web browser how exactly to display the content. HTML is used in this application to design the web page for the user interface. The W3 school will be a great place to learn and practice HTML [29].

Cascading Style Sheets (CSS) is used to describe how to style HTML content. CSS is used to define layout, design, and variations. This is used in the application to add style to the user interface and not have a standard display that would be uninteresting to the user. W3 school [28] and Codeacademy [4] are a good place to learn and practice CSS.

JavaScript allows for dynamic web page content. JavaScript is a necessary part of this application in order to do the data visualization that is present to the user and the data handling to the front-end. In this project, the JavaScript is used for data

handling and automatic updating of web pages. W3 school [30] and Codeacademy [4] helped me to learn and understand JavaScript in an interactive way.

Flask is a popular extensible web framework for developing web applications written in Python. It is based on Werkzeug toolkit and Jinja2 template engine [8]. Since it does not require particular tools or libraries it is called a micro framework. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Flask framework supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools [33]. It consists of features such as development server and debugger, integrated support for unit testing, RESTful request dispatching, and much more. In this application, the Flask is used to create outputs for test cases and run test cases. We need to install Python2.6 or newer and configure virtual environment before installing Flask. The Flask documentation will provide steps to install Flask [8].

Celery is an asynchronous task/job queue which is based on distributed message passing. Celery is written in Python, but the protocol can be implemented in any language. Celery focuses on real-time operation but also supports scheduling. Celery communicates via messages, usually using a broker to mediate between clients and workers. To initiate a task the client adds a message to the queue, and the broker then delivers that message to a worker. Celery can consist of multiple workers and brokers, giving way to high availability and horizontal scaling. The execution units are called tasks and are executed concurrently on a single or multiple worker servers using multiprocessing, evenlet or gevent. Tasks can execute asynchronously in the background or synchronously wait until it is ready [3]. In the Submit application, the Celery queue handles the request from the Rails App. We have to install the message broker RabbitMQ before installing Celery, which is running in background. The installation steps for Celery queue is provided in the Celery documentation. Celery is also running in background of Flask App [15].

RabbitMQ is a lightweight feature-complete, stable, durable, and easy to install open source message broker. Its an excellent choice for a production environment. RabbitMQ is the default message broker with celery so it doesn't require any additional dependencies or initial configuration [22].

Python is used for Flask application development. It is a high level, object oriented, general purpose, powerful programming language. Its features like dynamic type system and automatic memory management together with its interpreted nature make Python an ideal language for scripting and application development in most of the platforms. Python has an extensive and comprehensive standard library which is freely available and makes the application development easier [21]. Biginners can learn Python from Tutorialspoint and Codeacademy [4]. The python ibrary also help to develop the application.

JavaScript Object Notation (JSON) is used for storing and exchanging data between a browser and a server [11]. It is a lightweight data-interchange format and easy for humans to read and write. It was derived from JavaScript but it is a language independent data format and easy to generate and parse [31]. In Submit it is used to exchange between Rails and Flask. W3 school and JSON documentation will help to learn the structure and create JSON object. Python and Ruby have different functions to encode data in JSON format.

Chapter 3

Design

3.1 Overview

The software requirement specification is the basis for software development. It describes the functional and non-functional requirements, and includes a set of use cases that describe user interactions that the software must provide [7]. This chapter discusses the functional requirements, non-functional requirements, and use-case modeling of the Submit web application.

3.2 Requirements Specification

3.2.1 Functional Requirements

The functional requirements describe the functionality that the system is supposed to perform. It depends on the type of software, expected users and the type of system where the software is used. The functional requirements include the technical details, data manipulation, data processing, data integration, security requirements, performance, data migration, and conversion [34]. Functional requirements for the Submit web application are listed below in Table 3.1. Functional requirements are labeled as FR.

Table 3.1: The functional requirements of Submit

FR01	Allow users to login to their account
FR02	Allow users to edit their account
FR03	Allow users to edit files within the web page
FR04	Allow instructors and admins to create courses
FR05	Allow instructors to edit courses
FR06	Allow instructors to open or close course enrollment
FR07	Allow instructors to create new assignments
FR08	Allow instructors to specify start and due dates on an assignment
FR09	Allow instructors to specify how to compile and run uploaded submissions
FR10	Allow students to enroll in courses
FR11	Allow students to submit assignment files
FR12	Allow instructors to mark student as a grader of the course
FR13	Allow graders to function as an instructor without the ability to create courses
FR14	Allow students to view assignment grades and grader comments
FR15	Allow graders to run submission against test cases
FR16	Allow graders to create comments on a students submission
FR17	Allow instructors to edit assignments

3.2.2 Non-Functional Requirements

The non-functional requirements describe the specific behavior or functions that the system is supposed to perform. It can also be referred to as the quality attributes of the system architecture. Response time, scalability, reliability, maintainability, usability, etc., are some of the examples of non-functional requirements [35]. Non-functional requirements of the Submit application are listed in Table 3.2. Non-functional requirements are labeled as NFR.

Table 3.2: The Non-functional requirements of Submit

NFR01	The website will be developed on Ruby on Rails.
NFR02	The test cases will be running on flask.
NFR03	Be fast at serving pages and files.
NFR04	Be intuitive and quick to learn.
NFR05	Be compatible across all major web browsers.
NFR06	Support many students in multiple courses.
NFR07	Data and user information shall be stored in MySQL.

3.3 Use Case Modeling

A use case is a written description of the list of actions or event steps of how the user will perform tasks, typically the interactions between a role and a system, to achieve a goal. It outlines a user's point of view and a system's behavior as it responds to a request. Use cases help us to explain how the system should behave and react under given condition by successfully achieving the goal of that system, set by stakeholders. [38] In the Submit system, there are four types of users: Administrator, Instructor, Grader, and Student. Use cases are labeled as UC.

UC01: Log In

All users must have an account to access the website. The system is linked with University's NetID and password, so all users should be login with those credentials.

UC02: Create Courses

Instructors and administrators will be able to create new courses. They will be able to specify course name, description, term and year and whether the course is open or not.

UC03: Edit Courses

The instructors will be able to edit courses.

UC04: Delete Courses

The instructor will be able to delete courses.

UC05: Enroll Courses

Students and graders must be enrolled in a course to view and access the assignments. Each course has a twelve character registration token that must be given by the instructor to enroll in that course.

UC06: Manage Enrolled Students

Instructors will be able to view all enrolled students in a course and edit their roles or remove them from the course.

UC07: Edit Users

The Submit administrator will be able to view all enrolled users and edit their information and roles.

UC08: Create Assignments

Instructors will be able to create new assignments in courses. The instructor will be able to specify assignment name, possible points, assignment available date and due date.

UC09: Create Test Cases

Instructors will be able to create run methods and test cases which consist of inputs and expected outputs, for which all submitted code for a given assignment will be run and tested.

UC010: Edit Test Cases

The instructors will be able to edit and update test cases.

UC011: Create Run Variables

The instructors will be able to specify run constraints such as maximum CPU run time and maximum core size.

UC012: View Assignments

The students will be able to view all the assignments for the course. This includes assignment name, available date, due date, and time.

UC013: Submit Assignments

Students will be able to submit their assignment once they are satisfied with their testing against the given test cases. Once the assignment is submitted, then the files may not be edited any more by the student.

UC014: Edit Submissions

Students will be able to edit their assignment by adding, removing, or modifying their uploaded files. Besides that, the files may be edited directly with the application's built in text editor before submitting the assignment.

UC015: Run Test Cases

Instructors, students, and graders will be able to run assignment submission code against a set of test cases. Certain test cases may be created as hidden and this will be only visible to instructors and graders.

UC016: Unsubmit Assignment

Instructors will be able to change the status of submission from submitted to not submitted. Then the students are able to edit or resubmit the assignment.

UC017: Grade Submission

Instructors and graders will be able to grade student's submissions. It can be done either one at a time or all at once. They will be able to create grade file based on the results.

UC018: View and Comment on Submissions

Instructors and graders should be able to view code submitted by students. They can open the files in the built in text editor and provide feedback through comments for each line. This will be visible to students.

UC019: Upload files

Students and instructors will be able to upload their file by 'click' or drag-drop.

UC020: Download Files

Instructors and graders will be able to download the assignment as a zip file and grades of all students as an Excel file.

Use Case Diagrams

The use case diagram for the Submit application is shown below. It shows the users or actors for various use cases and its association and relationship between an actor and use cases. There are four different kind of users in this application. The use case diagram for the instructor is shown in Figure 3.1. The use case diagram for the administrator is shown in Figure 3.2. The use case diagram for the grader is shown in Figure 3.3 The use case diagram for the student is shown in Figure 3.4.

3.4 Database

Submit stores all data associated with the website in a MySQL database. The database requires the definition of tables and the type of values stored in tables. This includes user data with encrypted and salted passwords, uploaded data files, course information and so on. Here the database structure is relational because there is a relationship exist between database tables such as 'has-many', 'belongs-to' and 'has-and-belongs-to-many' [5]. The relationship that between database tables is shown in Figure 3.5. The tables in the Submit application database are as follows:

Figure 3.1: The Use Case diagram for the Instructor

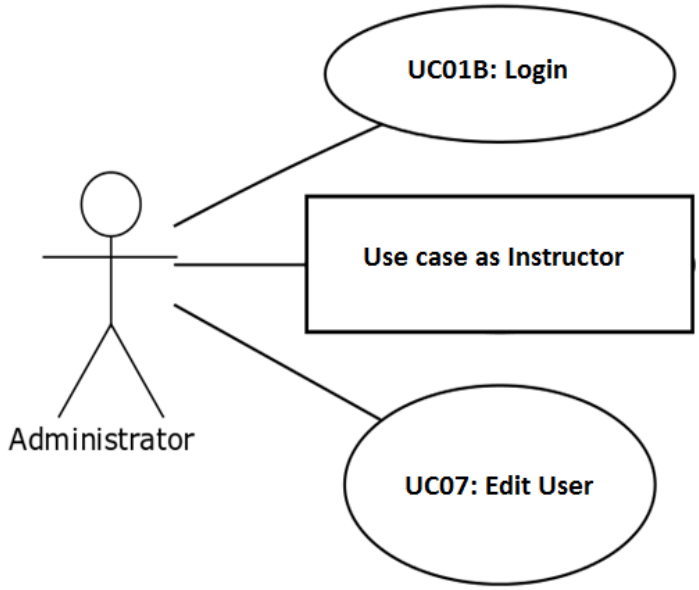

Figure 3.2: The Use Case diagram for The Administrator

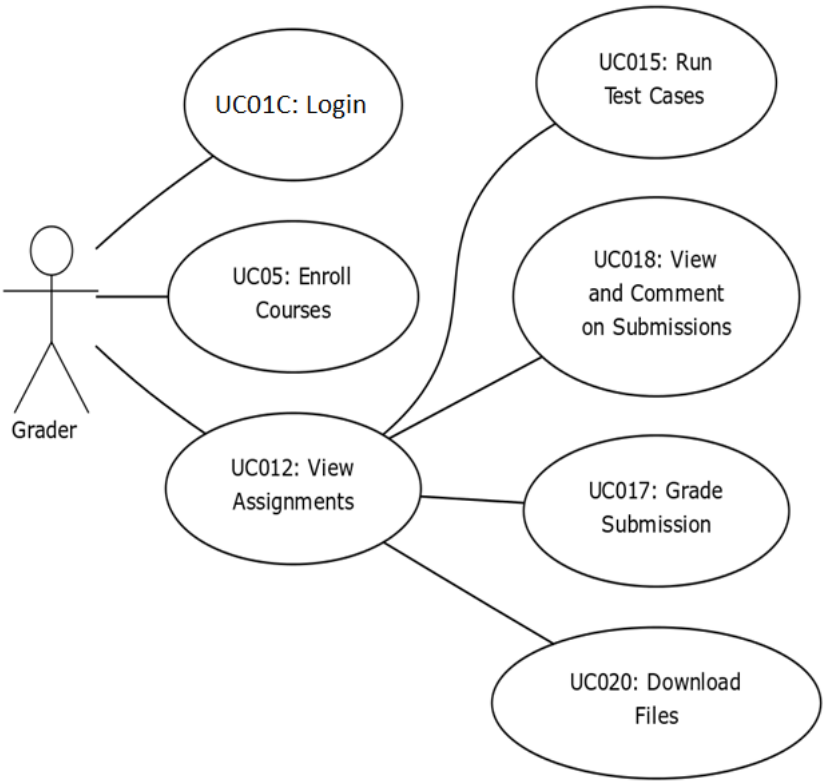

Figure 3.3: The Use Case diagram for the Grader

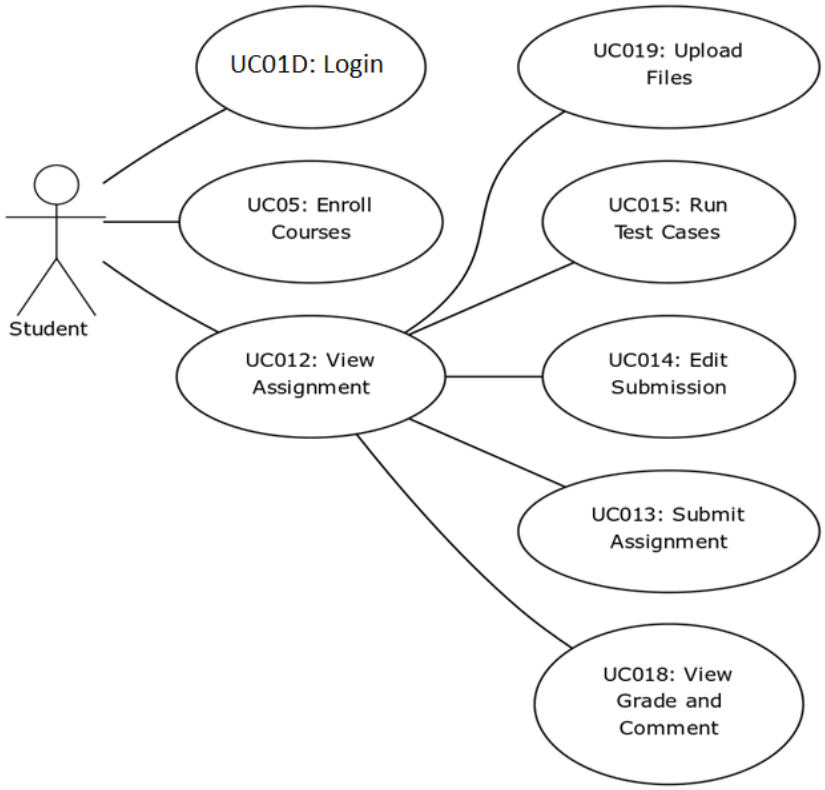

Figure 3.4: The Use Case diagram for the Student

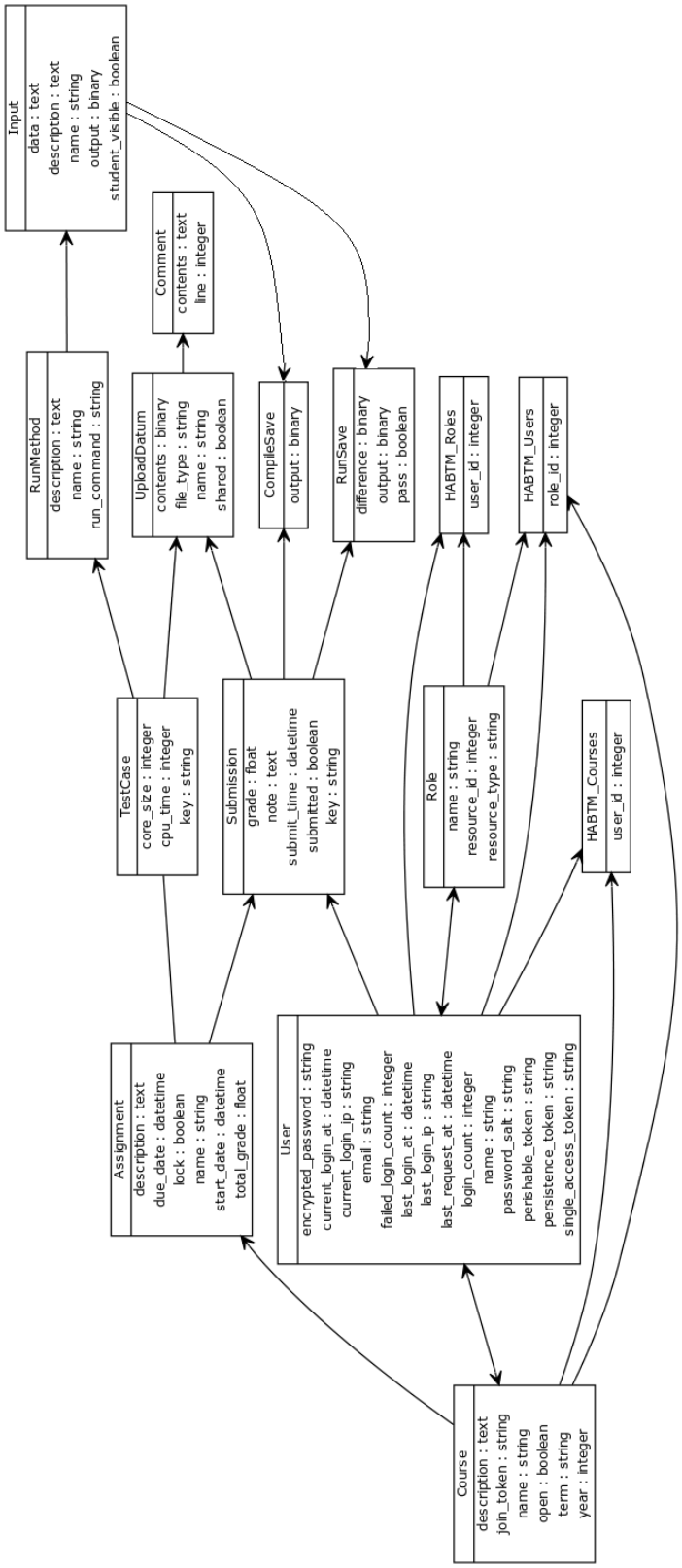

Figure 3.5: Diagram representing the relationship between Database Tables

- Users Data table will store details of each user such as first name, last name, Netid, and email. Also, it stores the account creation time, update time, login count, failed login count, last requested time, current login time, current login IP, and last login IP to ensure security. The ID is used as a primary key. This table has relations to courses, submissions, and roles tables. (Figure 3.6)

users		
primary key	id	int (11)
	email	varchar (255)
	persistence_token	varchar (255)
	single_access_token	varchar (255)
	perishable_token	varchar (255)
	login_count	int (11)
	failed_login_count	int (11)
	last_requested_at	datetime
	current_login_at	datetime
	last_login_at	datetime
	current_login_ip	varchar (255)
	last_login	varchar (255)
	created_at	datetime
	updated_at	datetime
	first_name	varchar (255)
	last_name	varchar (255)
	netid	varchar (255)

Figure 3.6: Users Table

- Assignments table will store the information of assignments such as assignment name, description, course id, start date, due date, and total grade. It also stores the record of assignment creation time, updating time, and whether the assignment is submitted or not. The ID is used as a primary key. This table has relations to submissions and test cases tables. (Figure 3.7)

assignments		
primary key	id	int (11)
	lock	tinyint (1)
	due_date	datetime
	start_date	datetime
	description	text
	course_id	int (11)
	created_at	datetime
	updated_at	datetime
	name	varchar (255)
	total_grade	float

Figure 3.7: Assignments Table

- Courses table will store the information of each course such as name, id, description, term, and year. It also stores the record of assignment creation time, updating time, whether the course is open to students or not. The join token associated with each course will be stored in this table. The ID is used as a primary key. This table has relations to users table, assignments table, and different user roles. (Figure 3.8)

- Test cases table will store the information regarding test conditions of each assignment. It includes run constraints such as CPU time, max core size, assignment id, creation time, and updating time. This table has relations to run methods and upload datum tables. (Figure 3.9)

- Inputs table will store the inputs for each assignment, its corresponding outputs, and description. It also stores the record of inputs creation time, updating time, and the input is student visible or not. This table has relations to run saves and compile saves tables. (Figure 3.10)

- Run methods table will store the details of run command, name and description, and other details that required to run the assignment. The ID is used as a

courses		
primary key	id	int (11)
	name	varchar (255)
	open	tinyint (1)
	description	text
	join_token	varchar (255)
	created_at	datetime
	updated_at	datetime
	year	int (11)
	term	varchar (255)

Figure 3.8: Courses Table

test_cases		
primary key	id	int (11)
	assignment_id	int (11)
	created_at	datetime
	updated_at	datetime
	cpu_time	int (11)
	core_size	int (11)
	column_name	varchar (255)
	key	varchar (255)

Figure 3.9: Test Cases Table

inputs		
primary key	id	int (11)
	name	varchar (255)
	description	text
	data	text
	output	blob
	student_visible	tinyint (1)
	run_method_id	int (1)
	created_at	datetime
	updated_at	datetime

Figure 3.10: Inputs Table

run_methods		
primary key	id	int (11)
	name	varchar (255)
	run_command	varchar (255)
	description	text
	test_case_id	varchar (255)
	created_at	datetime
	updated_at	datetime

Figure 3.11: Run Methods Table

primary key. This table has relations to the inputs table. (Figure 3.11)

- Submissions table will store user id, assignment id, submission time, key, the status of submission, and grade. It also stores the record of assignment creation time and updating time. The ID is used as a primary key. This table has relations to upload datum, run saves and compiles saves tables. (Figure 3.12)

- Run saves table will store the final results of the assignment evaluation such as whether it passed the test cases, output, the difference between student, and instructor output. The ID is used as a primary key. (Figure 3.13)

submissions		
primary key	id	int (11)
	grade	float
	note	text
	assignment_id	int (11)
	user_id	int (11)
	created_at	datetime
	updated_at	datetime
	submitted	tinyint (1)
	submit_time	datetime
	key	varchar (255)

Figure 3.12: Submissions Table

run_saves		
primary key	id	int (11)
	submission_id	int (11)
	difference	blob
	pass	tinyint (1)
	output	blob
	created_at	datetime
	updated_at	datetime
	input_id	int (11)

Figure 3.13: Run Saves Table

compile_saves		
primary key	id	int (11)
	submission_id	int (11)
	output	text

Figure 3.14: Compile Saves Table

upload_data		
primary key	id	int (11)
	name	varchar (255)
	contents	mediumblob
	created_at	datetime
	updated_at	datetime
	submission_id	int (11)
	test_case_id	int (11)
	file_type	varchar (255)
	shared	tinyint (1)

Figure 3.15: Upload Data Table

- Compile saves table stores the information of compiled output. That means if there is a compile error, the details of that error is stored in this table. (Figure 3.14)

- Upload data table will store files related to the assignment, its type, contents, and id. It also stores whether the file is shared or not, submission id, test case id, created time, and updated time. This table has relations to the comments table. (Figure 3.15)

- Comments table stores comments for students' program codes. It includes line number, content, upload datum id, created time, and updated time. (Figure 3.16)

comments		
primary key	id	int (11)
	name	varchar (255)
	contents	text
	line	int (11)
	upload_datum_id	int (11)
	created_at	datetime
	updated_at	datetime

Figure 3.16: Comments Table

3.5 Architecture

The web application is built on Ruby on Rails. It is designed to run on two back-ends, Rails and Flask framework. Each of them has its individual design and then is interconnected. The website was built with Ruby on Rails which uses the Model-View-Controller (MVC) architecture [12]. Rails offers simple integration of data retrieved from the controller with the MySQL database. The models are responsible for validating data in order to create consistency and then saving data to a database.

Figure 3.17 shows the architecture of the Submit Ruby on Rails framework design. The view will be the front-end to the user which generates an output for the user. Each view is connected to a controller action that will communicate with data. The controllers are responsible for linking the front-end views with the back-end models and database. The model maintains the relationship between objects and database which handles data validation [17].

The following steps explains the flow of working of the Submit application:

Step 1: The user request the page.

Step 2: The router identifies the controller and action

Step 3: The controller instantiates or manipulate the models.

Step 4: The model add or retrieve data from the database.

Step 5: The model renders data to the controller.

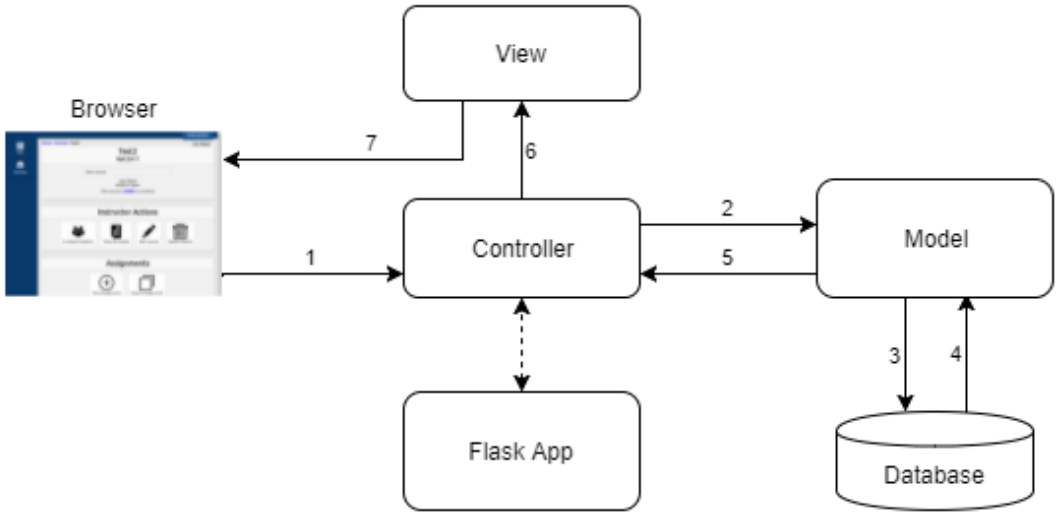

Figure 3.17: An architecture diagram of Submit Rails App

Step 6: Controller renders the view.

Step 7: The rendered view is send to the user.

The creation of test cases and testing assignments against test cases are handled by the Flask server. The data required to create correct outputs and run the test cases are sent to Flask in JSON format from Rails using HTTP request. The Flask handles all the requests and parses the content. After that, it compiles and run the program and generates outputs. The result is sent back to Rails in JSON format. The RESTful service handles the incoming RESTful requests for data. The controller gets the data and saves it into the database and updates the web page with the new result.

The asynchronous job queue Celery is used to schedule the jobs. The tasks are executed concurrently on one or more worker servers and the task can be executed asynchronously in the background or wait until ready(synchronously). RabbitMQ is used as the message broker to communicate between server and client. Figure 3.18 is the architecture of Submit's design for creating outputs and run test cases.

The following steps explain how Submit handles 'Run Test' request and 'Create

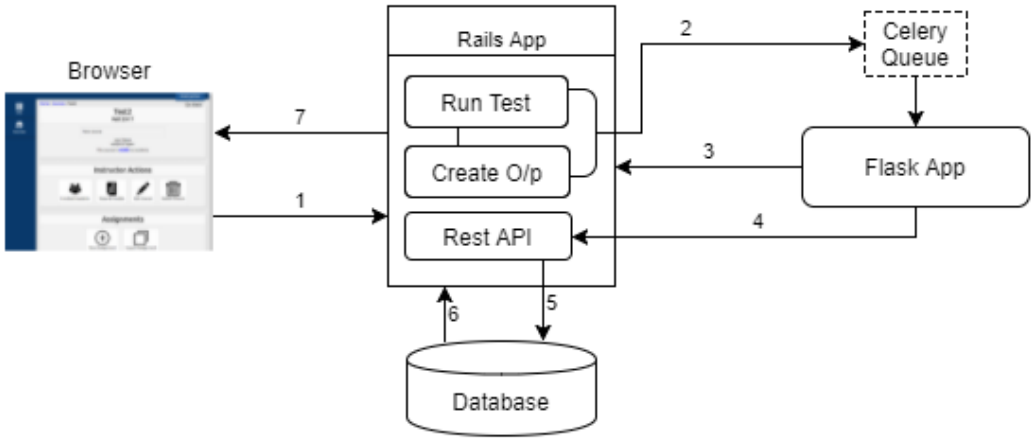

Figure 3.18: An architecture diagram of Submit

Output' request.

1. The user requests 'Run Test'/'Create Output'.

2. The router identifies the controller and action. The controller sends an HTTP request to Flask App with data in JSON format. The request will be add to the queue in a Celery job queue.

3. The Flask handles the request as a Per-Request After-Request callback. It sends back a unique key back to the controller and saves it into the database and returns to the Flask App.

4. The Flask App the processes the request and sends the results back to Rails as an HTTP request. The data will be encoded in JSON format.

5. The RESTapi handles this request and checks for the key and consistency. Then it parses the data and saves the results into the database.

6. Controller renders the view.

7. The rendered view is sent to the user.

Chapter 4

Implementation

4.1 Submit web application

4.1.1 Overview

The Submit web platform is developed to support instructors to manage their courses and programming assignments more easily and sufficiently. It helps instructors and graders speed up assignment grading and upload the comments and grades very quickly and easily. The students also get instant feedback on their submissions. This web application is built with the advantages of Ruby on Rails and Flask. Its Model-View-Controller framework provides the default structure for a database, web service, and web pages. The MVC design aids in the separation of tasks for similar page requests, user interaction, and database implementation. The views section holds the information on how to display the web page, the controllers handle the requests, and the models handle the data. The MySQL database is what the model integrates with and stores all the data associated with the Submit website [13].

This section discusses the implementation of the Submit web application. It includes the implementation of different subsystem and controllers, models, and views associated with it, along with some screen shots.

4.1.2 View Implementation

In Ruby on Rails, web requests are handled by the Action Controller and Action View. The controller is concerned with the communication with the database and the view

is responsible for compiling the responses. For each controller, there is an associated views directory which holds the files that make the views associated with that controller. These files are used to display the view that results from each controller action. The view generates an output for the user. It is built with HTML/Embedded Ruby files which allows HTML/CSS integration with Ruby. Every web connection to a Rails application results in the displaying of a view [17]. Figure 4.1 shows a sample view implementation in Submit.

```
<% breadcrumb :assignment, @assignment %>
<% content_for :title, @course.name + ' assignment ' + @assignment.name %>
<header>
  <h1><%= @course.name %> assignment <%= @assignment.name %></h1>
</header>
<div class="content-group">
  <%= @assignment.name %><br>
  <%= @assignment.description %><br>
  <%= "Total Points " + @assignment.total_grade.to_s %><br>
  <%= @assignment.start_date %><br>
  <%= @assignment.due_date %><br>
</div>
<header>
  <h1>Your Submission</h1>
</header>
<div class="content-group">
  <%= @submission.user.netid %> - <%= link_to 'Edit',  submission_url(@submission) %><br>
</div>
```

Figure 4.1: Sample View implementation

4.1.3 Model Implementation

In MVC architecture, the model keeps the relationship between the object and the database and it handles data validation, association, and transaction. A model illustrates a singular object in Rails. This is implemented in Active Record library, which provides an interface and binding between the tables in a relational database and the Ruby program code that manipulates database records. An Active model is a base library which contains various modules used for developing classes. There are many models associated with submission application development which connect the controller actions and the database. User sessions, courses, assignments, submission, etc. are examples of some models implemented in this application [27]. Figure 4.2 shows a sample model implementation in Submit.

```ruby
class Assignment < ActiveRecord::Base
  belongs_to :course
  has_many :submissions
  has_one :test_case

  validates :name, presence: true
  validates :description, presence: true
  validates :start_date, presence: true
  validates :due_date, presence: true
  validates :total_grade, presence: true
  validate :due_date_after_start_date

  after_save :create_submissions_for_students
  after_save :create_test_case

  def create_submissions_for_students
    course.users.select { |user| user.has_local_role? :student, course }.each do |student|
      submission = submissions.create(user: student) unless submissions.exists?(user: student)
    end
  end
```

Figure 4.2: Cutout of a Model implementation

4.1.4 Controller Implementation

In MVC architecture, the Controller coordinates the actions between model and view. The controller directs the flow of data. It receives the user commands and works with models to process the commands and displays the web page properly by the view. The controller is responsible for routing external requests to internal actions and manages sessions which gives the impression of progressing interaction with our application. The Submit web application is driven by a set of controllers. Each controller is responsible for certain actions of Submit such as user sessions, courses, assignments, submission, etc. The controllers are further divided into actions and these actions are linked to the applications routes. When a specific route is called, the associated controller action finds the required models and passes them to an associated view [27]. The controller uses JavaScript and AJAX for reloading page content for tasks such as deleting a file or updating a page. The controllers that the Submit application use are described below with its functionalities.

Application Controller

The Application controller is the base controller created by Rails which is inherited from the Action controller. All the other controllers are inherited from the Application controller. The Application controller is not associated with any views or models.

Assignment Controller

The assignment controller manages everything related to the assignment. The assignment model validates whether all the required fields are set, the due date is valid, and all students can view the assignments. The assignment controller is related to creating new assignments, grading all assignments, editing and updating assignments, downloading all grades and the instructor can make the submission status from submitted to not submitted. There is a view and a model associated with this controller.

Comments Controller

The comments controller manages the creation and deletion of comments. The instructor can comment on the student's submission on the corresponding line of code. There is a favorites list associated with this. This controller checks whether the comment is unique for each line. Both the students and graders can view the comments.

Courses Controller

The course controller handles the course. The course model validates and makes sure that all required fields are set and all the data are in an acceptable range like the term, year, and, open. The course controller manages the functions such as create new courses, edit, update and delete courses, manage the students joining the courses with join token, manage enrolled students (edit, update or remove users), view, and download grades etc. There is a view and a model associated with this controller.

Inputs Controller

The inputs controller is undertaking the activities related to inputs for test cases of each assignment. Each input is associated with a name, description, and data. The instructor can edit, update, and delete inputs. These are handled by the inputs controller.

Run Methods Controller

The run method controller acts as a holder for multiple inputs and runs the command. The run command describes how to compile and run the program. We can also edit, update, and delete run method.

Submission Controller

The submission controller is responsible for all actions related to assignment submission and grading. It contains all the files and information related to a student's assignment submission. Submission controller can run, edit, delete, submit, and grade. The submission will be locked once it is submitted, and then students can't modify after that unless the instructor unlocks the submission. It is also responsible for run-save updates, output deletion, and the creation of pdf grade files.

Test Cases Controller

This module handles the test cases associated with the assignments. The instructor composes the test cases and sets constraints like CPU time and max core size. It also generates required outputs based on the instructor's code and inputs. The instructor can edit, update, and delete test cases.

Upload Data Controller

This controller manages all uploaded files related to each assignment. It handles the program files uploaded by the instructor, assignments submitted by the students, shared files like make files, grade files, shared uploaded data in test cases, comments, and downloading files.

User Session Controller

This controller manages the user authentication. The user must have a valid NetID and password to login or register to the course. This is a part of the AuthLogic gem which handles the session logic.

Users Controller

The users controller handles user account information such as user name, and email id. Users can create, update, and delete the account.

4.2 RESTful API

The RESTful API is an application program interface used to communicate with two different web services which will help to manage the application easily. It ensures the interoperability between the computer system on the web. It uses HTTP requests to GET, PUT, POST, and DELETE data. The REST API service has routes and controller setup. We can define the end points in the form of routes. The controllers of the system handle the incoming RESTful requests with data [1]. Then the project controller parses the data from JSON format and saves that into the database.

Base API Controller

The Base API controller is used to handle the authentication and extract the common API functionalities. This approach requires re-authentication on a per-request level. It is a very simple approach and ensures statelessness. Figure 4.3 is the Base API controller of the Submit application. It checks whether the JSON object is empty or not before parsing the request [1].

```
1  class BaseApiController < ApplicationController
2    skip_before_action :verify_authenticity_token
3      before_filter :parse_request
4
5    private
6      def parse_request
7          if not request.body.read.empty?
8              @json = JSON.parse(request.body.read)
9          end
10     end
11 end
```

Figure 4.3: REST Base API controller for parsing JSON

API Submission Controller

It is the project controller for the REST API. The project controller parses the data from the JSON format and saves that into the database. It assures the security using defensive programming software design to continue functions in unpredicted situations. It checks whether the JSON object has the required key using 'has_key' before continuing to parse the JSON request. Unless it has the key, it stops parsing and considers it as a bad request [1]. Figure 4.4 is the API submission controller of the Submit application.

```
1 class ApiSubmissionController < BaseApiController
2   before_filter only: :data do
3     unless @json.has_key?('submission')
4       render nothing: true, status: :bad_request
5     end
6   end
7
```

Figure 4.4: API_submission Controller for parsing JSON request

4.3 Flask App

Flask is a micro-framework for Python which is based on Werkzeug, Jinja2 libraries [8]. In this project, Flask is used for creating test cases and running and testing the test cases for the assignment instead of running on Rails. This App is running in the background which speeds up the processes. When the instructor creates the test cases, they are sent to the Flask App and then Flask is generating corresponding outputs based on the program and inputs. In the case of testing the assignment, when the user hits the run button, then the request is sent to the Flask App. The Flask App then compiles and runs the program and compares the outputs with the instructor's output and sends back the result to Rails. Figure 4.5 shows an example of the Flask App used in this project.

```
1  from flask import Flask, request, g
2  import json
3  import os
4  import shutil
5  import subprocess
6  import difflib
7  from json import JSONEncoder
8  from flask import Response
9  import http
10 import hashlib
11
12 static_directory = os.environ["SUBMIT_COMPILE_PATH"]
13 app = Flask(__name__)
14 @app.route("/submission", methods=['POST','PATCH'])
15 def submission():
16     app.logger.debug("JSON received...")
```

Figure 4.5: Example for Flask App used in Submit

4.3.1 Requests

The Rails App sends the data to the Flask App in the form of HTTP request. The data is encoded in JSON format because it is a lightweight data interchange format and easy to generate and parse. Here we used Per-Request After-Request callbacks because the REST API requires re-authentication on a per-request level [9]. When the Flask App gets a request from the Rails App, it generates a key and sends it to Rails App. Rails saves this key into the database and returns to the Flask App. After that, the Flask sends back the results to Rails, and the outputs are saved into the database.

In order to create the test cases, the instructor defines the run methods and inputs in the specific fields and uploads a sample code for each assignment. When the instructor hits the 'Create Output' button, an HTTP request is sent to the Flask App. The data required to generate the output is sent as a JSON object. It contains details such as course name, assignment name and id, current username and id, run constraints such as CPU time, max core size, run name, run command, input name,

input data, program file, and make file. Figure 4.6 shown below is an example of sending a request to the Flask App.

```
# Test Case sending to Flask app
uri = URI('http://hpcvis6.cse.unr.edu:5000/testcase')
http = Net::HTTP.new(uri.host, uri.port)
req = Net::HTTP::Post.new(uri.path, 'Content-Type' => 'application/json')
req.body = {:details => @details, :RunMethods => @RunMethods,
            :sourcefiles => @sourcefiles, :sharedfiles => @makefile}.to_json
res = http.request(req)
test_case.key = res.body
test_case.save
```

Figure 4.6: Example for sending HTTP request from Rails to the Flask App

On the Flask side, the App parses the JSON object and then compiles and runs the program with the given data. The subprocess Popen is used to compile and run the program. This will generate the required output. This output is again encoded into JSON format with JSONEncoder() function, and the response is sent back as an HTTP request. On the Rails side, the REST API handles the request, then parse JSON and saves the outputs into the database.

For running the test, the user has to click the 'Aswathi thesis reportRun Test' button. Then the request sends as an HTTP request to the Flask App with required data in JSON format. In this case the JSON file contains details such as course name, assignment name and id, current username and id, submission id, run constraints such as CPU time, max core size, run name, run command, input name, input data, outputs associated with input, program file submitted by the student, and make file.

The Flask App handles this request the same as it handles the output generation. Here it parses the JSON data and compiles and runs the student's programs. After creating the output, then it is compared with the instructor's outputs associated with each input. The difflib module is used to compare each character of the output. Then the result encoded into JSON format and the response is sent back as an HTTP request. The output associated with each input is sent back to Rails one by one as it completes each test case. On the Rails side, the REST API handles the request and then parses the JSON and saves the outputs into the database. The

view associated with the submission controller displays the outputs. The page will be updated automatically when the outputs are saved in the database. Figure 4.7 shown below is an example of Per-Request After-Request callbacks used in the Flask App.

```
@app.after_request
def call_after_requests(response):
    for callback in getattr(g, 'after_request_callbacks', ()):
        callback(response)
    return response
def after_this_request(func):
    print ("function", func)
    if not hasattr(g, 'after_request_callbacks'):
        g.after_request_callbacks = []
    g.after_request_callbacks.append(func)
    return func
```

Figure 4.7: Example for Per-Request After-Request callbacks used in the Flask App

4.3.2 Request Queue

An asynchronous task/job queue is used here to schedule the jobs in the Flask App. Celery is used here for managing job queue. Celery can execute the tasks in the background and leave the application free to respond to other requests. After finishing each tasks, the Celery sends the result back to the Flask App. This will help to run each test without interruption. Celery requires three components: The Celery Client, Celery Workers, and a Message Broker. The Celery Client is used to provide background jobs, in Flask it runs with the Flask application. The Celery Worker processes the jobs in the background. The Celery Client uses message broker to communicate with the message queue. RabbitMQ is used as the message broker which is the default message broker for Celery. Implementation of Celery job queue is shown in Figure 4.8.

```
celery = Celery(app.name, backend='rpc://', broker='pyamqp://')
celery.conf.update(
    task_serializer='json',
    result_serializer='json')
@celery.task
def my_submissiontask(json, key):
    directory = static_directory + "/" + json['details']['username']
    # mk directory
    if not os.path.exists(directory):
        os.mkdir(directory, mode=0o777)
```

Figure 4.8: Celery task queue used in Flask App

The main feature that Submit has from the previous application is that Submit's back-end is running on two frameworks: Rails and Flask and the implementation of the job queue. Two major processes that this application has is the test case creation and testing of the submission, both are running on Flask. This makes the application faster, and it could use the resources of both frameworks. Also, the queue implemented is running in the background which doesn't require a lot of system resource and can handle more users.

Chapter 5

Application Walkthrough

The Submit's user interface for all users is designed very simply and easy to use and understand. It removes the repetitive navigation from one page to another and has more interfaces to access each function. The interface is basically designed for all four types of users: instructor, student, admin, and grader. Mainly there are two different views in this interface: one for students and the other for instructors. The student view is the default view for the application when a new user is logged in. This chapter will serve as a walkthrough of the Submit application, by showing the app's interface and describing its functionality for different roles: instructor, student and grader.

5.1 Instructor Walkthrough

5.1.1 Login

When a user browses the application, it shows the login screen as shown in Figure 5.1. The user can login to the Submit application with the University's NetID and password. The database verifies the credentials and then allows users to access the account. Since it is linked with NetID, the user's details such first name, last name, and email id are available in the database. Only the admin can edit the user information. The student role is the default role assigned. Only the admin can assign the role of an instructor. The instructor or admin can assign the grader role.

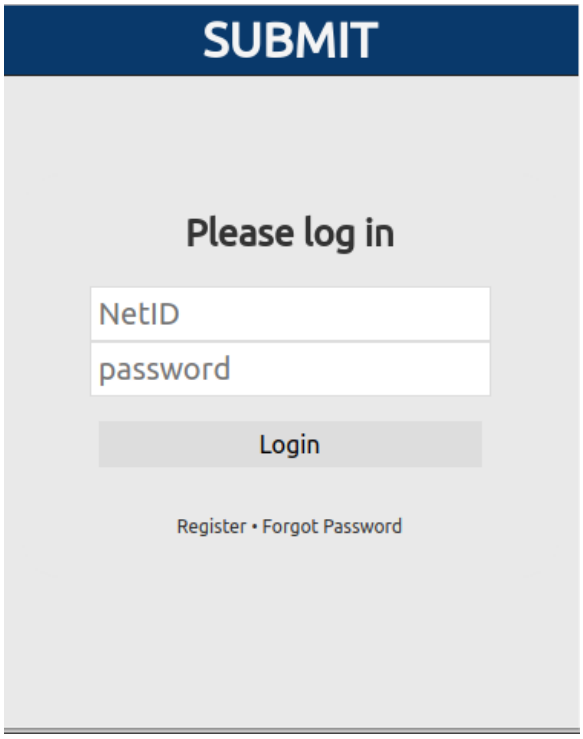

Figure 5.1: Submit Log In Page

5.1.2 Instructor Home Page

Figure 5.2 is the view of the instructor home page. The instructor can create a new course and view all the courses that the instructor is teaching for that term. The course table shows the course name, term, and year. By clicking on each page, user can go to the instructor actions page of that course.

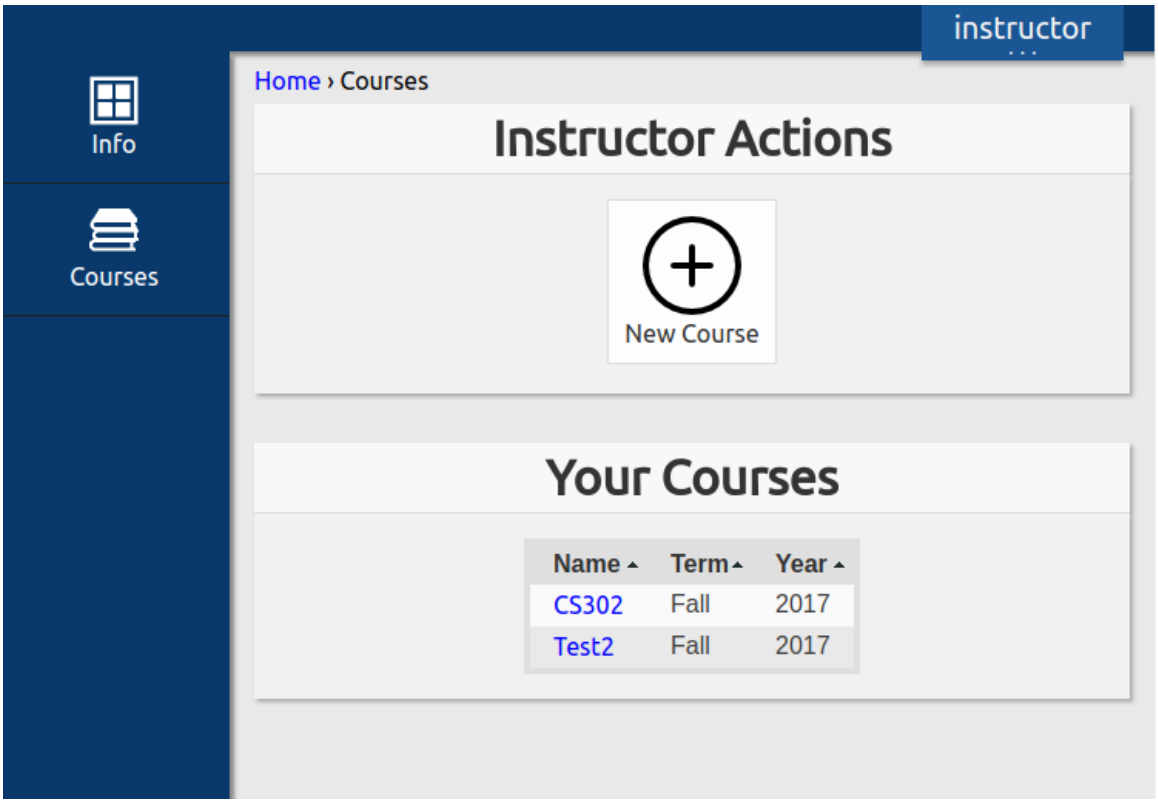

Figure 5.2: The Instructor Home Page

5.1.3 Create New Course

The instructors are able to create new courses. Figure 5.3 is the view of creating a new course. For the new class, the instructor can add course name, details and description, term, year, and specify whether the course is open for students or not. The instructors are able to edit or delete the course. There will be a unique enrollment token generated with each new course. The instructor sends this token to students who want to join the class. Students with this token are allowed to join the class.

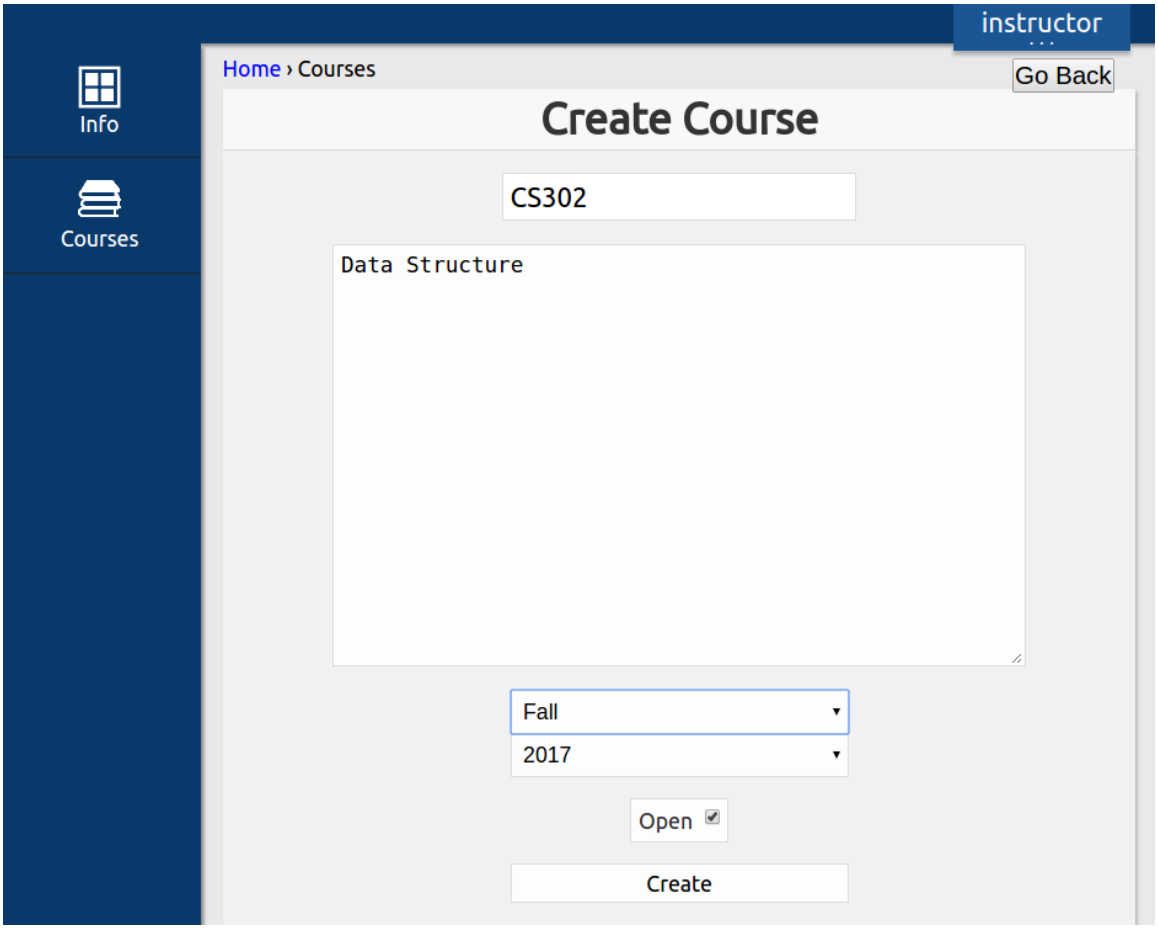

Figure 5.3: The New Course page

5.1.4 Instructor Actions

The instructor actions page has features such as manage enrolled students, edit course, delete the course, view all the student grades, create new assignments, and the assignments associated with the course. The join token for the course is also visible on this page. Figure 5.4 shows the view of the Instructor Actions page.

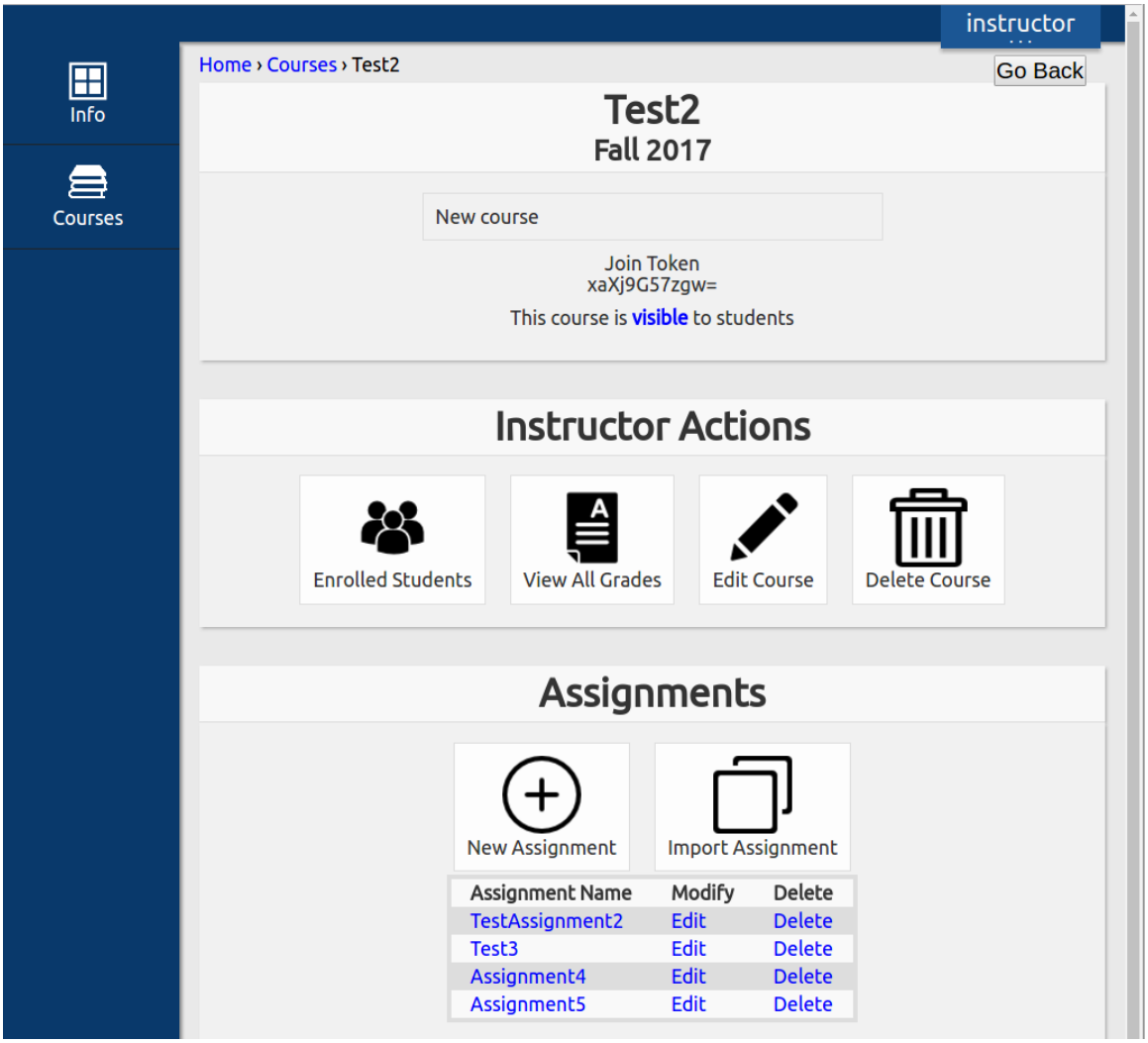

Figure 5.4: Instructor Action Page

5.1.5 Manage Enrolled Students

The instructors can manage all the enrolled students for that class. Figure 5.5 is the view of the enrolled student's page. The table contains student details such as first name, last name, email id and the roles the students have. There is an option for the instructor to remove a student from the course. The table can be sorted by student's first name, last name, email id, or role.

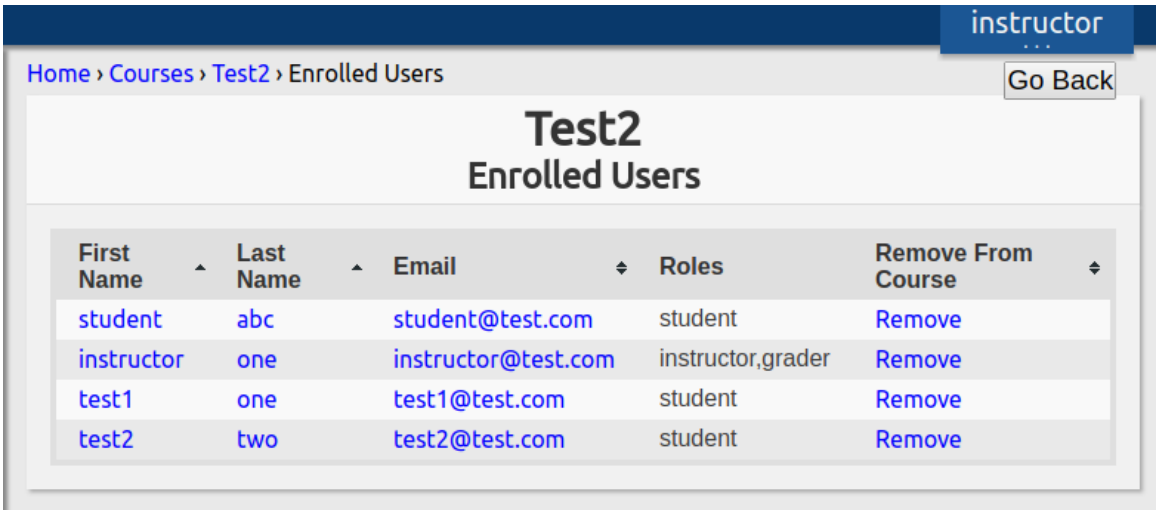

Figure 5.5: View of Manage Enrolled Students page

5.1.6 Create New Assignment

The instructors are able to create assignments associated with the course. Figure 5.6 shows the view of creating new assignment. While creating a new assignment, the instructor can add assignment name, description, available date, due date and time, and maximum score. This page is very user-friendly. There is a built in calendar for selecting the available date and due date. The instructors are able to edit or delete assignments.

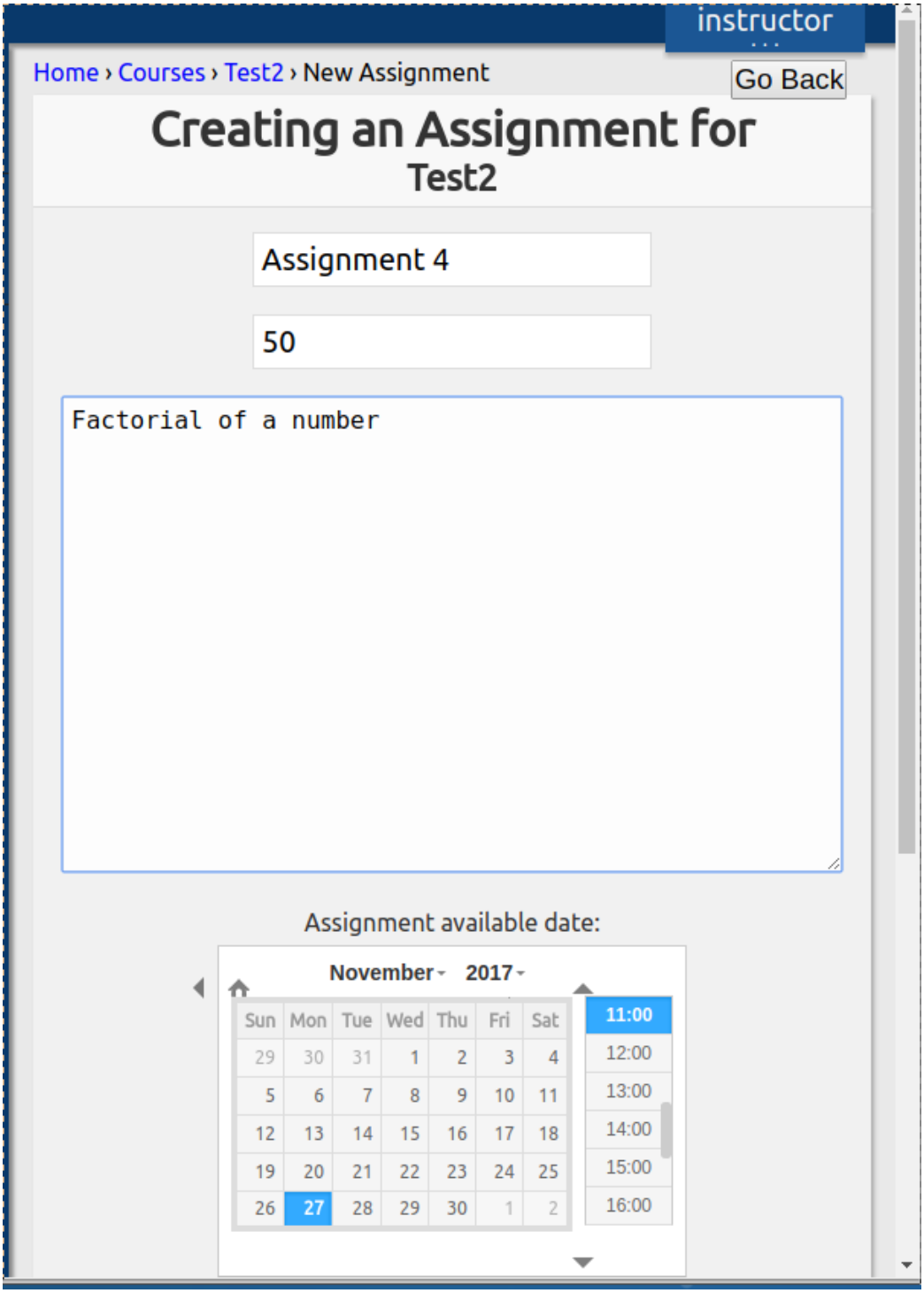

Figure 5.6: View of creating new assignment

5.1.7 Instructor Actions for Assignment

The instructor has an actions page for managing assignments. Here the instructor can edit test cases, test all student submissions, view all grades, unsubmitted all student assignments, and view each student's submission and grade. Figure 5.7 shows the view of Instructor Actions page for managing assignments.

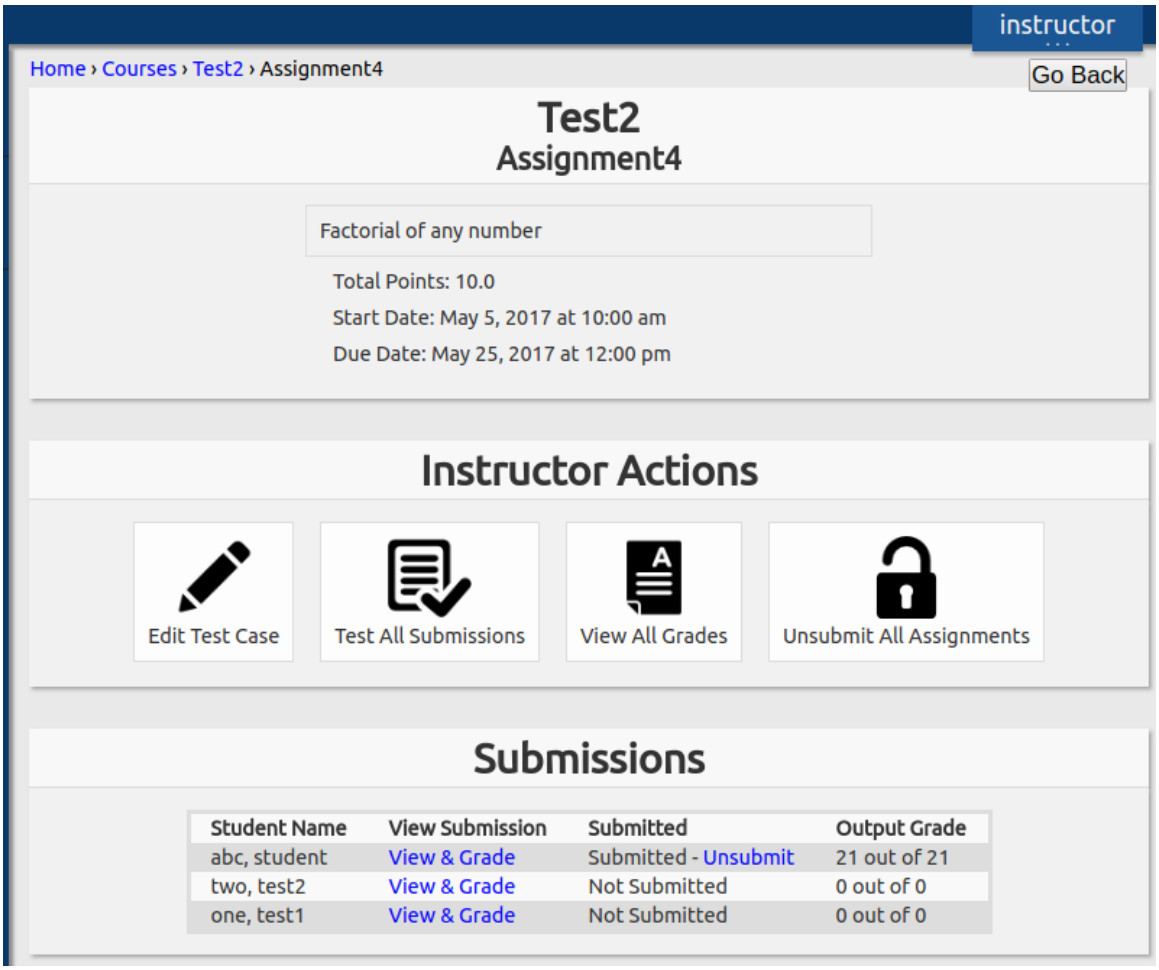

Figure 5.7: Instructor Actions Page for managing assignments

5.1.8 Create Test Cases

One of the main features of the Submit application is automatic grading of programming assignments. The instructor creates the assignment which contains a number of test cases that specify how to run the program. To create the test cases for the

assignment, the instructor uploads the program files that generate the correct output. After that, the instructor defines the run method and inputs to generate output. The run constraints, max CPU time and max core dump size are also specified to prevent infinite loops and avoid the system from storing large files. Figure 5.8 shown below is the instructor view of the test case page.

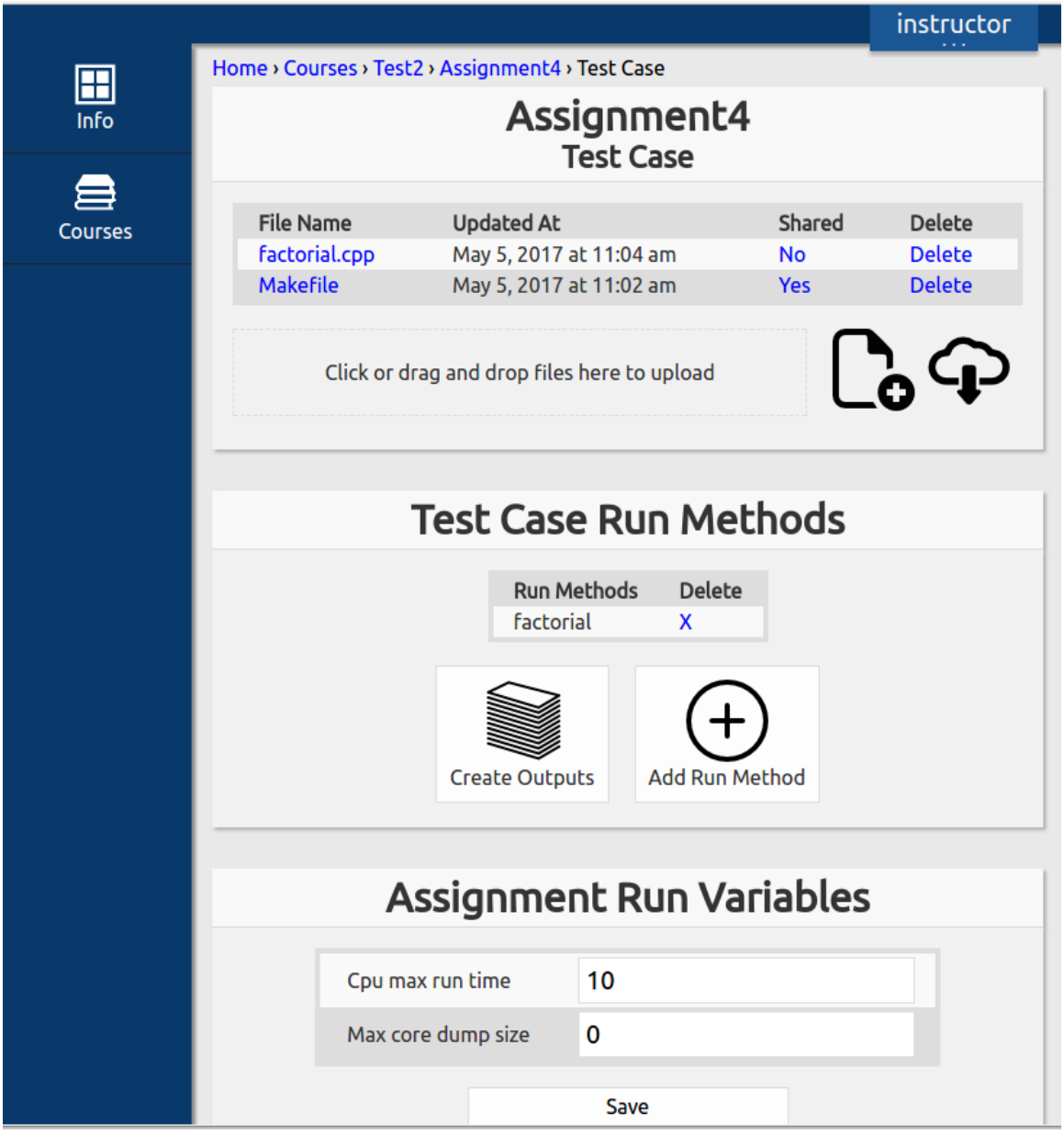

Figure 5.8: View of Test Case page where instructor uploads program, creates run methods, adds inputs and sets program parameters

Create Run Method

The run method defines the command line call to run the program. To create the run method, the instructor has to specify run name, run command, and can include a description about how to run the program. Figure 5.9 shown below is the view of creating a run method for the assignment.

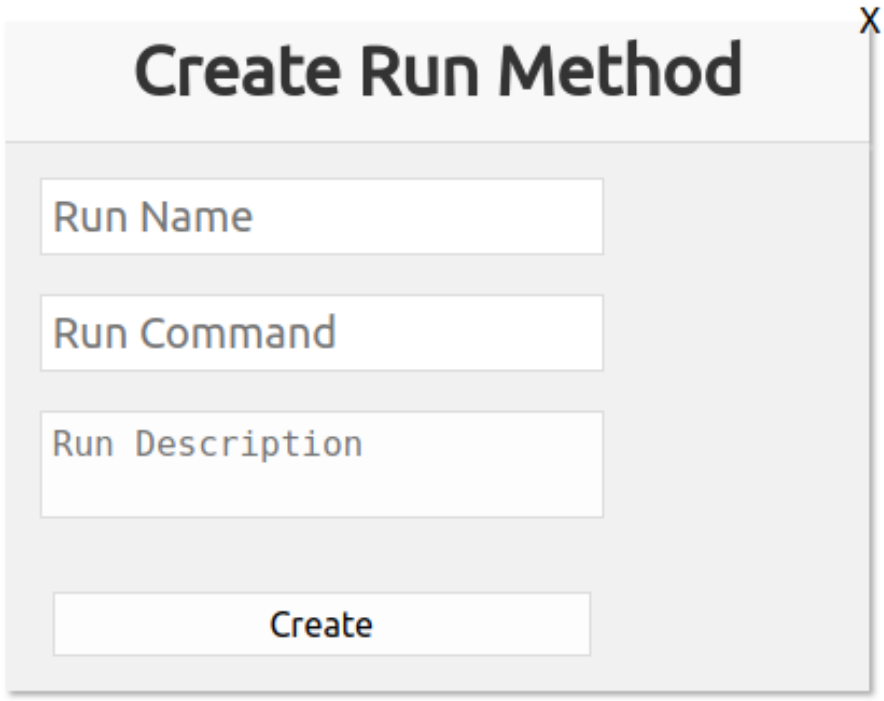

Figure 5.9: View of creating run methods

Create Inputs

The instructor specifies inputs to create expected output. The input object has name, description, and data. The instructor can add as many inputs as he wants. The outputs are obtained from the inputs within the run method. Figure 5.10 shown below is the view of the input created by the instructor.

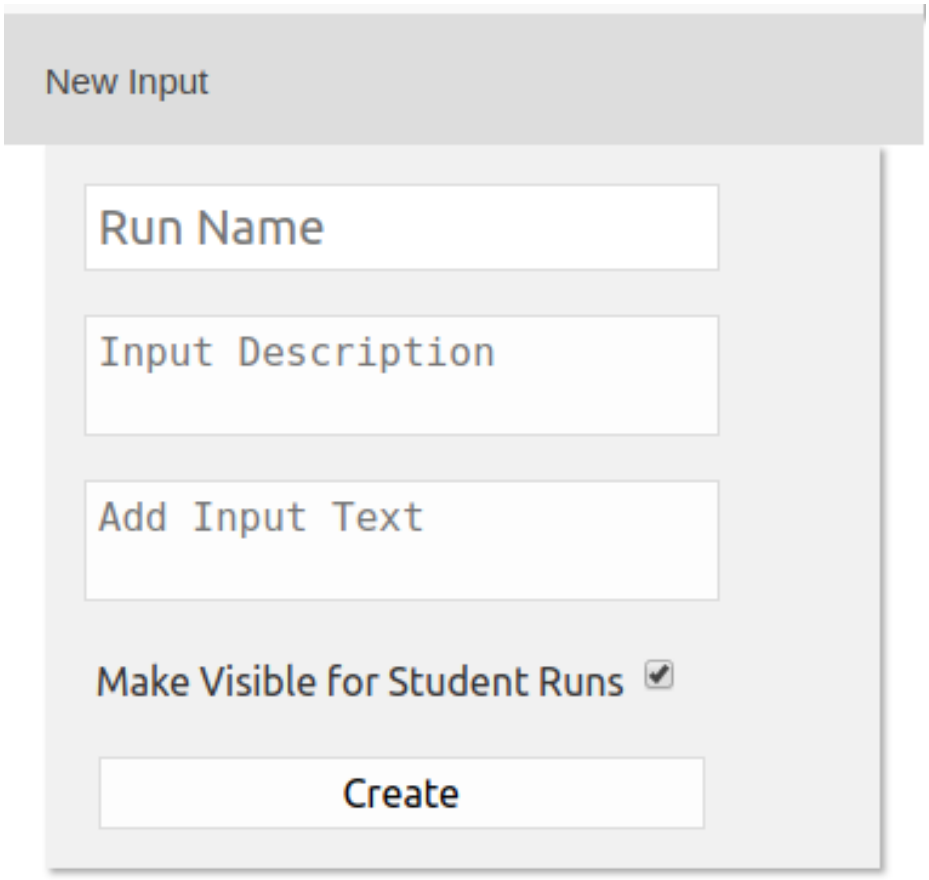

Figure 5.10: View of input page

Create Outputs

The instructors are able to create outputs associated with each input. By clicking on 'Create Output' button, the application send request to create output. Figure 5.11 shown below is the view of creating outputs.

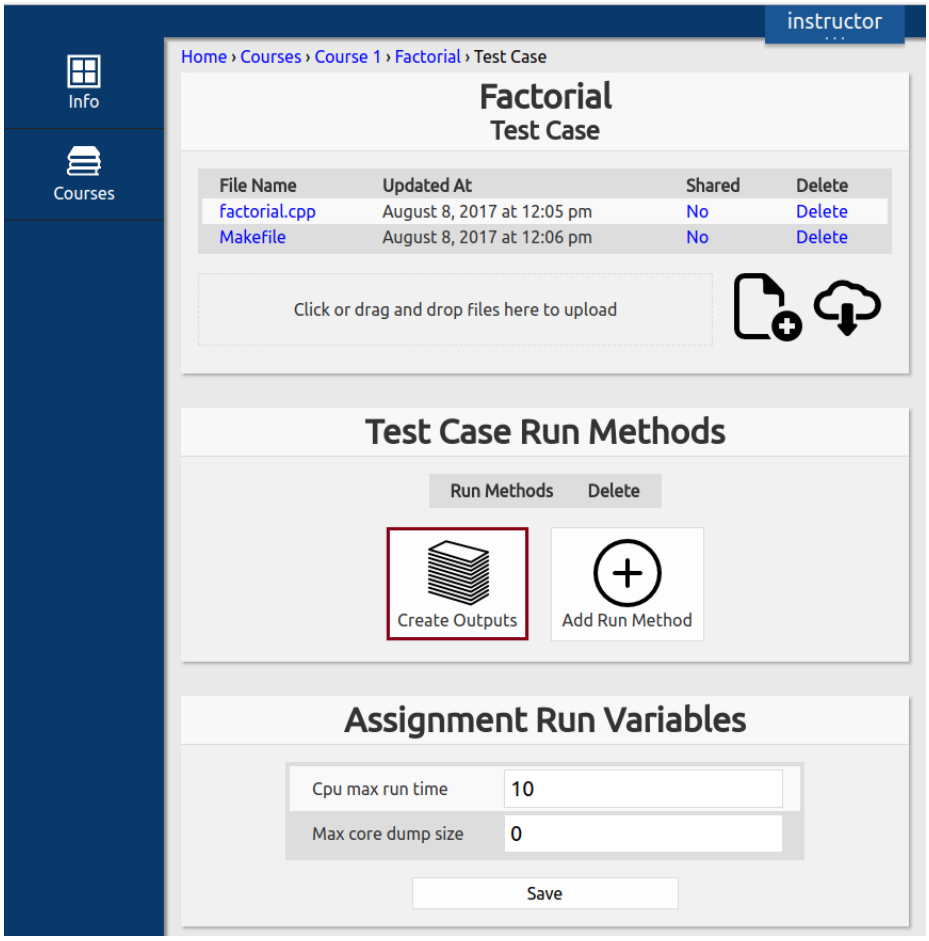

Figure 5.11: Create Outputs

5.2 Student Walkthrough: Part1

This section is discussing about the students point of view of this application. This part includes student's enrollment, view assignment, submit solution for the assignment, test assignment and submit assignment.

5.2.1 Student Home Page

Figure 5.12 is the home page for the students. All students must enroll into the course to access the assignments associated with the course. There is a unique enrollment token for each course. The instructor sends this token to students who want to join the class. Students can enroll in the course with this token. Figure 5.13 shows the enrollment into the course. The home page shows all the courses that the student is enrolled in. The course table shows the course name, term, and year. Student can navigate to each course page by clicking on course name.

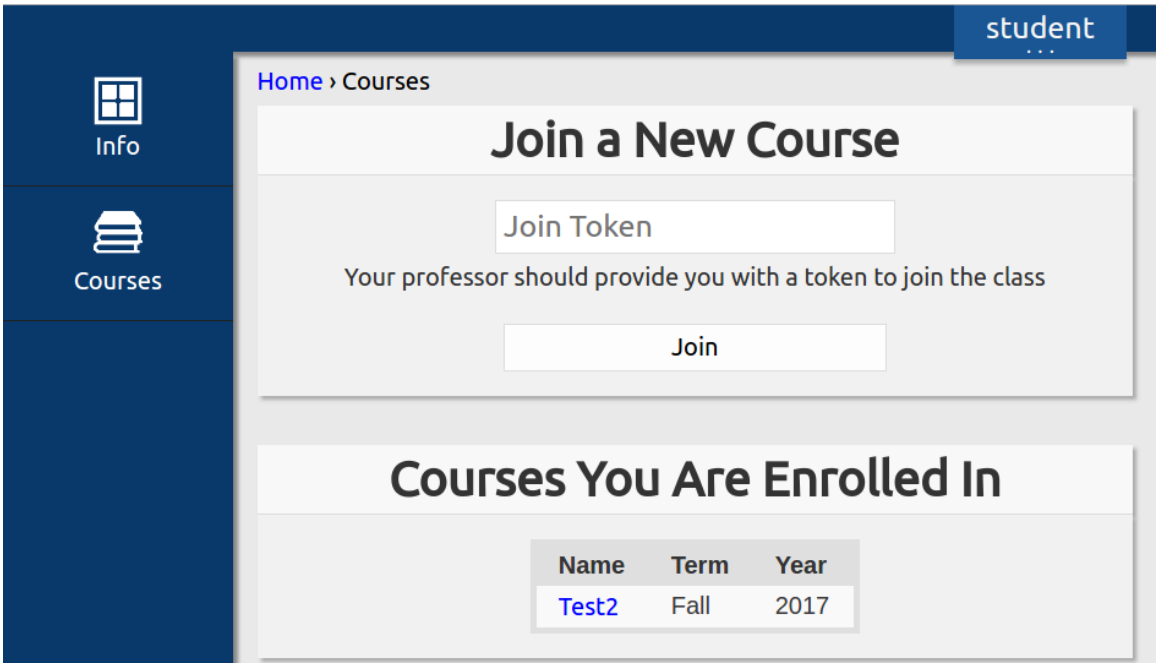

Figure 5.12: View of the Student page of Submit

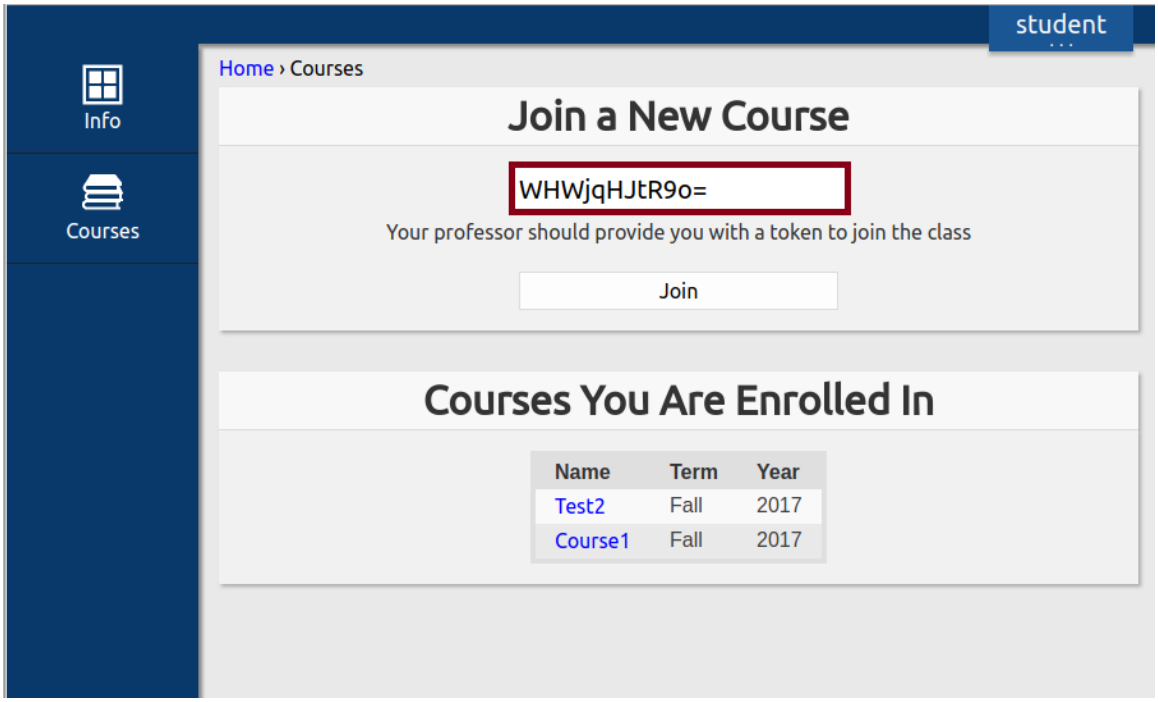

Figure 5.13: View of Student enrolling into a course

5.2.2 View Assignments

The student who is enrolled in the class can see all the assignments created for that class. This page specifies the start date and due date for the assignment. The students can access each assignment by clicking on it. Then they can view the assignment and submit the solution for the assignment. The students are also able to test their assignment with the public test cases created by the instructor before submitting it. Figure 5.14 shown below is the view of assignments page.

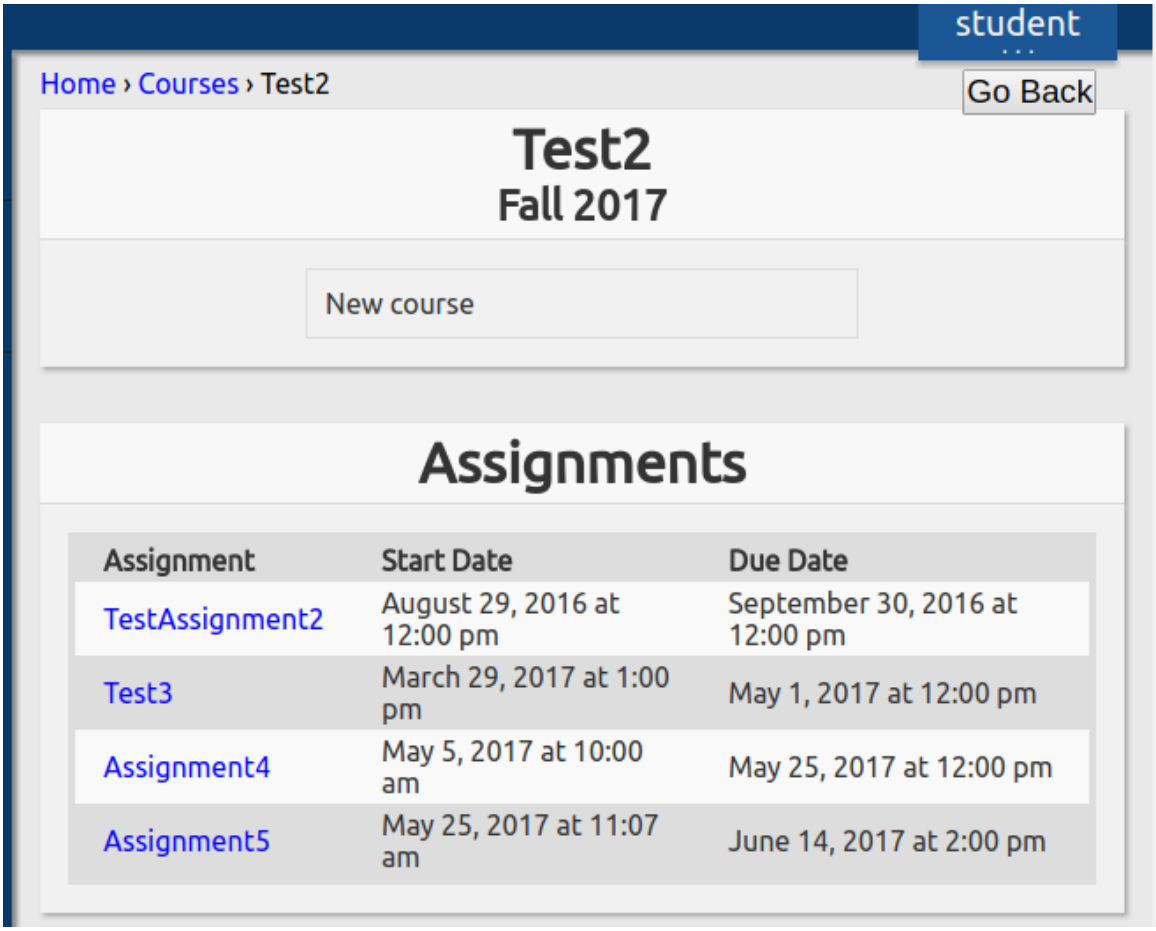

Figure 5.14: The Assignment page

5.2.3 Assignment Submission and Testing

The students are able to access the assignment and upload the program files for the assignment. The assignment can be tested against the test cases by clicking the 'Run Tests' button. The 'Run test' compiles and runs the program automatically and compares the output with the expected output created by the instructor. The user will get instant feedback for the test such as whether the test is passed or not, what output was expected, and the difference. This is another feature of Submit. Submit figures out the difference between the student's outputs and instructor's output and then generates a formatted output for the user to find the correct and incorrect results. Some of the results are set as invisible for students which are only visible to instructors and graders. If the student is satisfied with the output, then they can 'Submit and lock' the assignment. Once the student submits the assignment then it cannot be edited or run by the student. Figure 5.15 is the view of submitting solution for assignment and Figure 5.16 shows the view of the program output with correct and incorrect results. The input data, output, expected output, and difference are shown in the figure. The incorrect output will be shown in a red box.

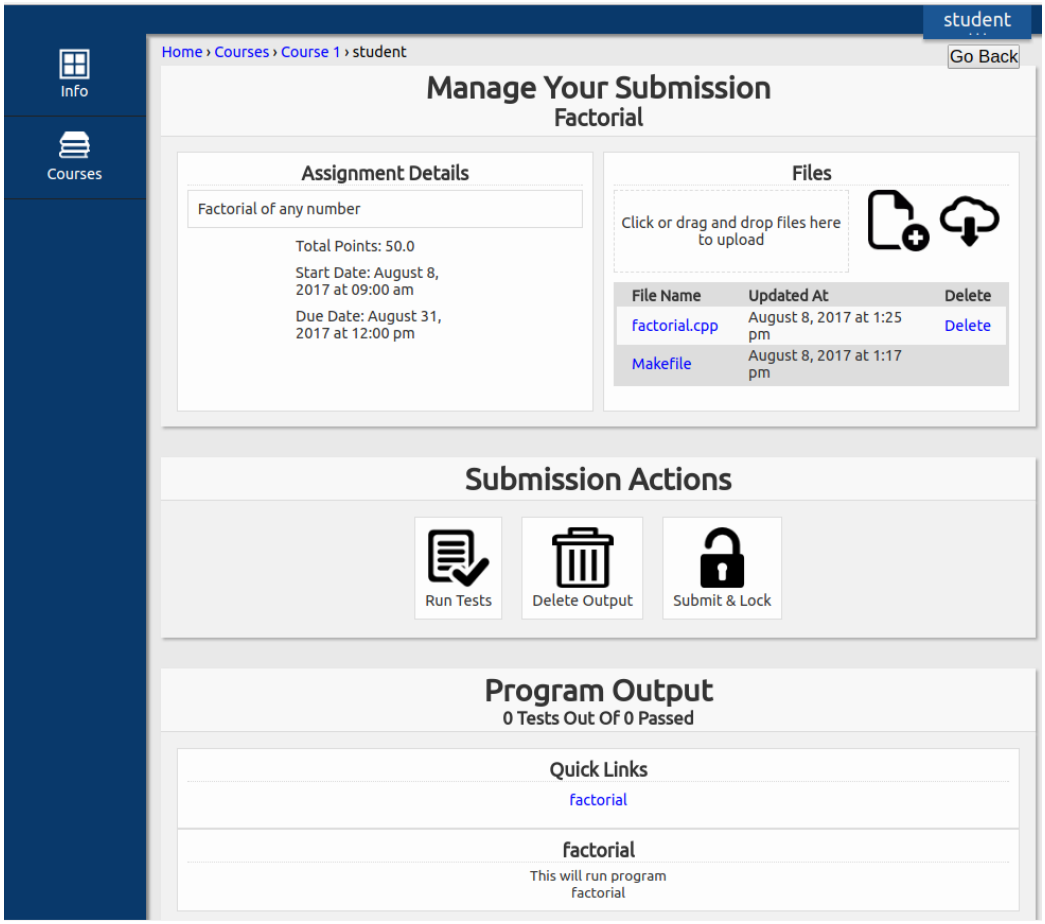

Figure 5.15: The view of student uploading solution for the assignment

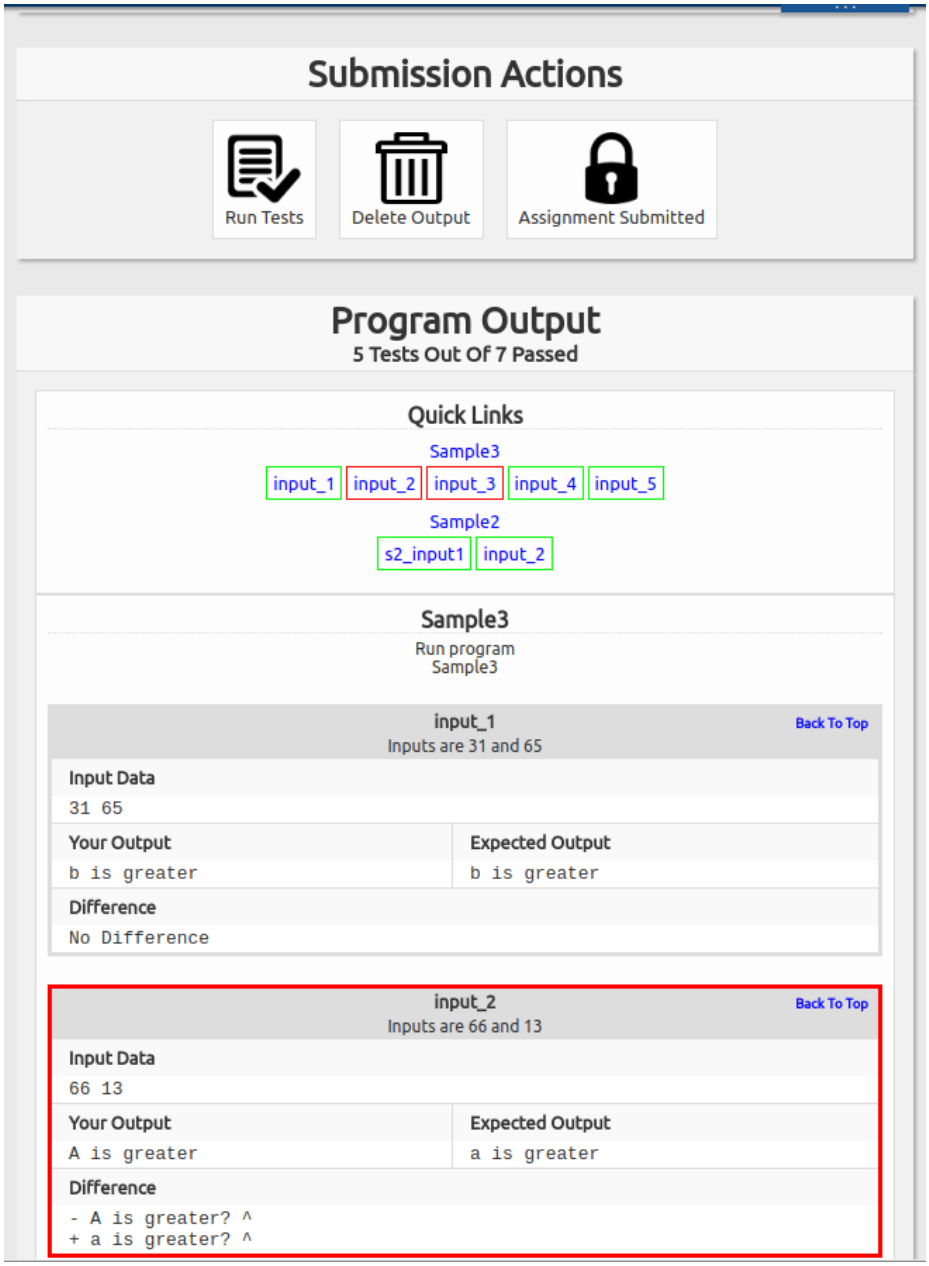

Figure 5.16: The output view for running program

5.3 Grader Walkthrough

This section will discuss the grading of students assignment. This part includes grader's enrollment into the class, view assignment, test assignment, comment on students code, and grade assignment.

5.3.1 Grader Home Page

Figure 5.17 is the home page for the grader. All graders must enroll into the course to access the assignments associated with the course. There is a unique enrollment token for each course. Grader can enroll in the course with this token. The home page shows all the courses that the grader is enrolled in. The course table shows the course name, term, and year. Grader can navigate to each course page by clicking on course name to view all the submitted assignments.

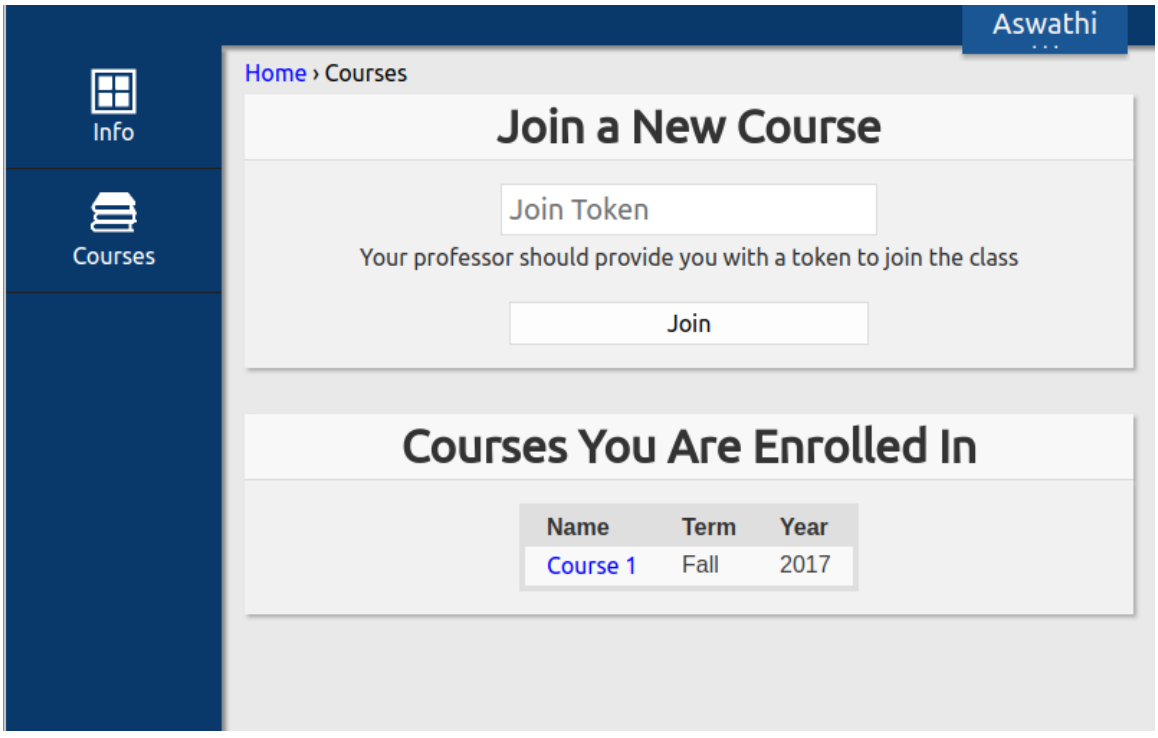

Figure 5.17: View of the Grader home page of Submit

5.3.2 Grader Actions Page

The grader has an actions page for managing and grading assignments. Here the grader can edit test cases, test all student submissions, view all grades, unsubmitted all student assignments, and view each student's submission and grade. Figure 5.18 shows the view of Grader Actions page for managing assignments.

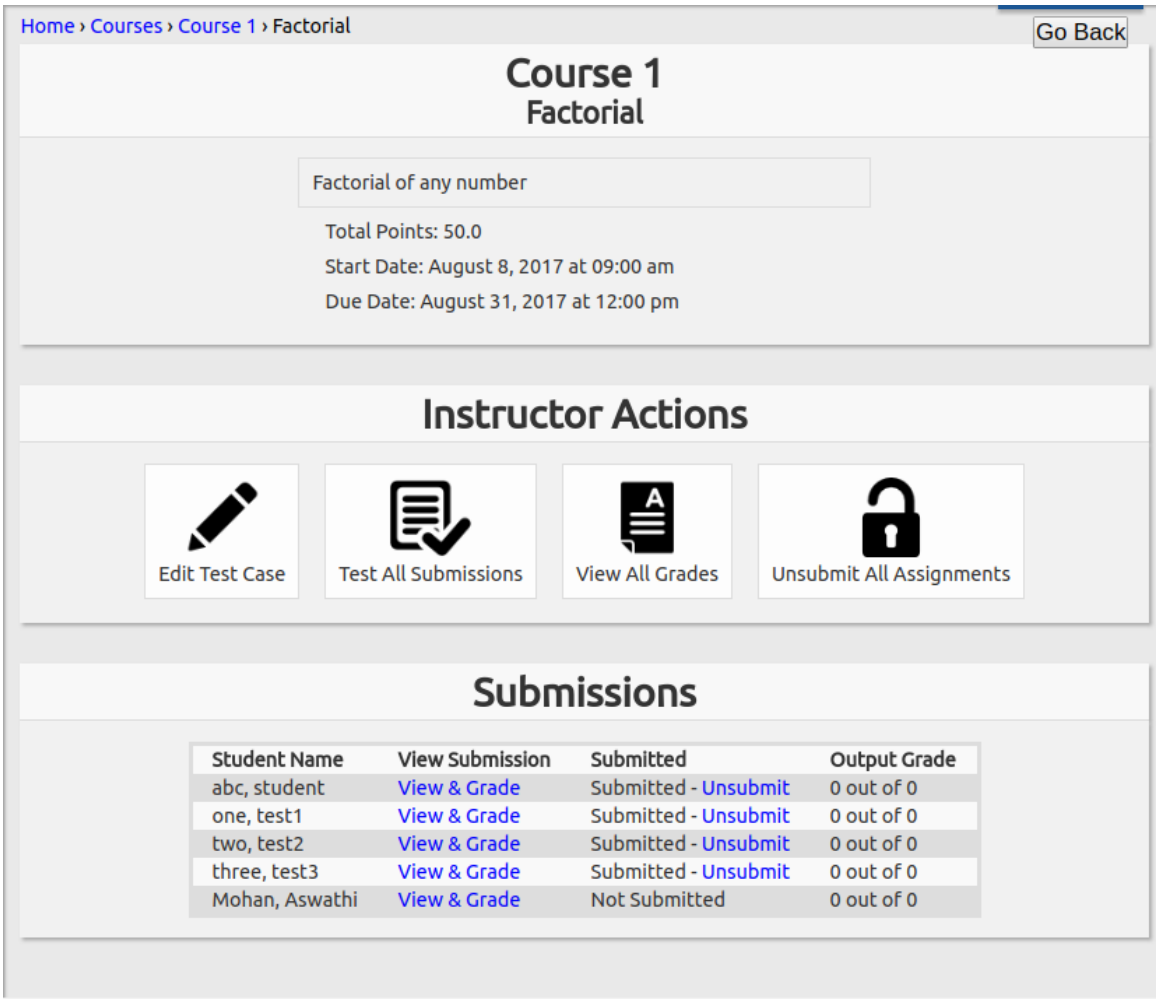

Figure 5.18: Grader Actions Page for managing assignments

5.3.3 View and Grade Assignment

The grader is able to view and grade each student assignment. This page shows the submitted program files and whether the submissions are on time or late. The grader can test the submission by clicking 'Run Tests' button. The grader will get instant

feedback for the test. It generates a formatted output for the user to find the correct and incorrect results. In this page, the grader can comment on students code and create the grade file based on the test. Figure 5.19 shows the view of test result generated by the grader.

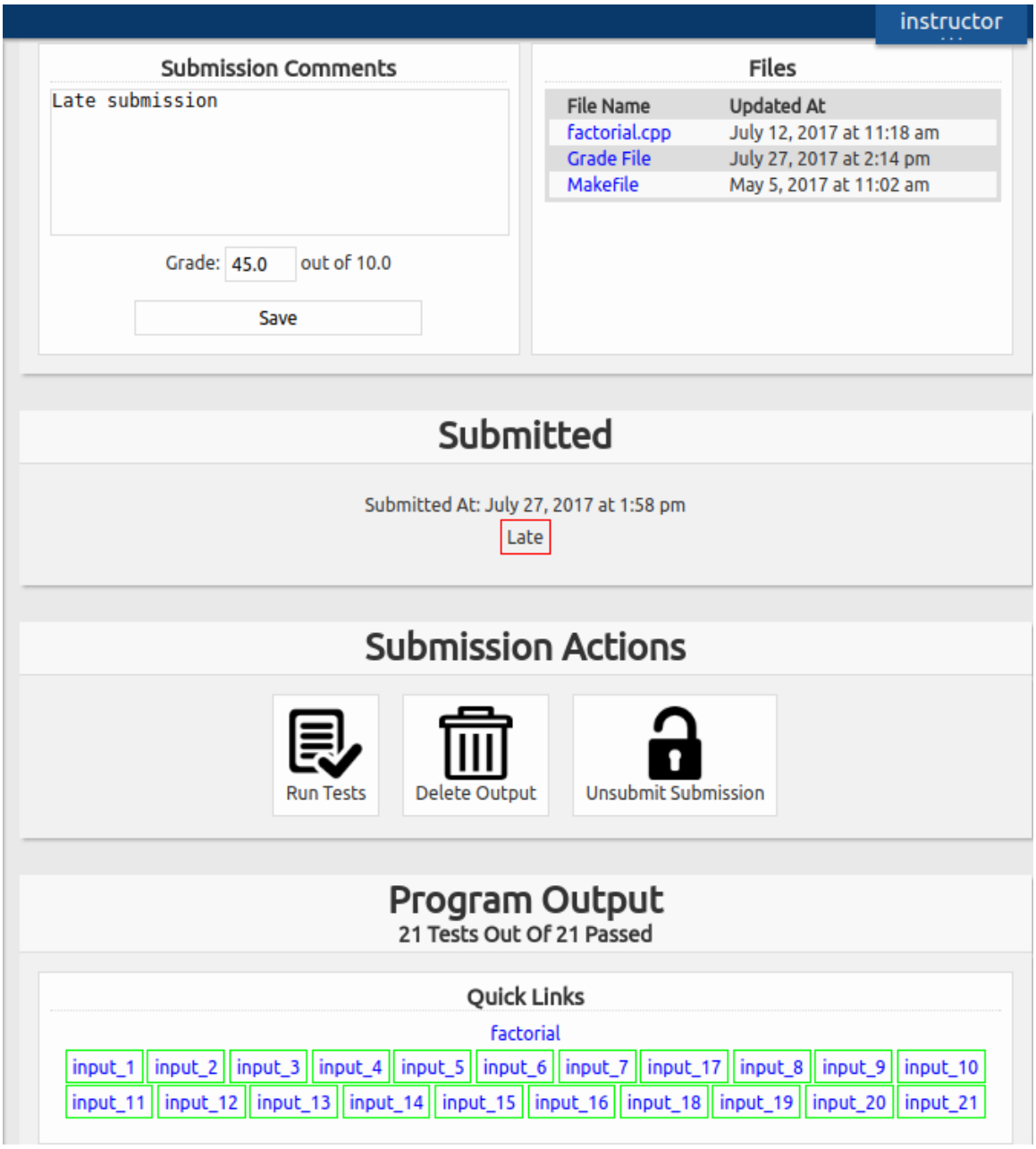

Figure 5.19: View and Grade Assignment

5.3.4 Comments

One of the other major features of the Submit application is built-in online text editor, Ace [2]. It is a high-performance code editor. Its built in features make the code editing easy. All users have access to it. This adds more support to students and graders. A student is able to make a minor change to their code in order to fix the error. This will eliminate the need to upload new code for fixing minor errors. The instructors and graders are able to add comments on each line of the student code. These comments can be viewed by students. This will make grading easy and students will get instant feedback. Figure 5.20 shown below is an example of text editor along with comments added to it.

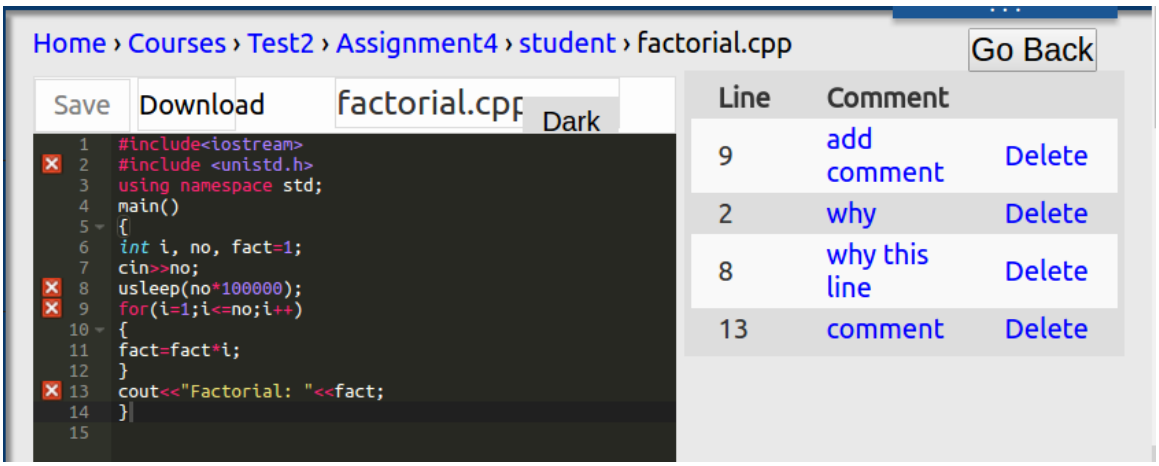

Figure 5.20: The Online Text Editor in Submit

5.3.5 Grade File

Submit will create a pdf report based on the student's grades and other details associated with testing. After testing the code, the instructor/grader can add grades and comments for the assignment. Once the grader uploads the grade file, it is also visible to students. The grade file includes students name, grades, comments, the list of inputs, and indicates whether the test is passed or not. Submit will create a pdf report based on the student's grades and other details associated with testing.

5.3.6 View All Grades

The instructor/grader is able to view and download grades of all student. The file shows the student name grade and comments for the submission. Figure 5.21 shown below is the grade file for all students. The downloading file will be in Exel format.

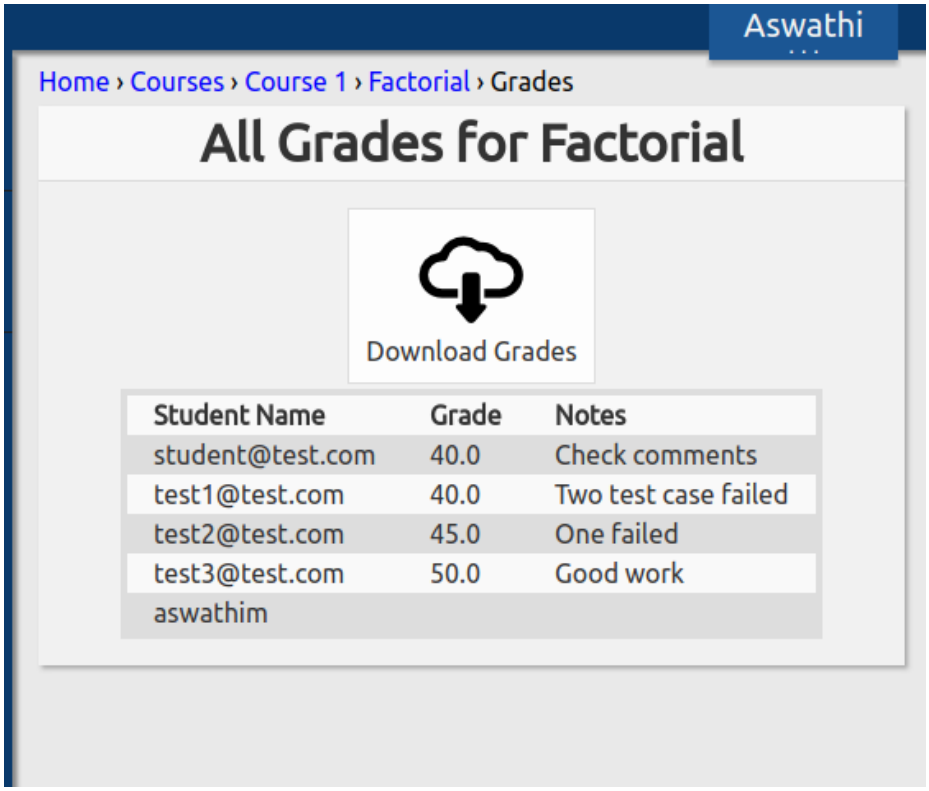

Figure 5.21: Sample View Grade file

5.4 Student Walkthrough: Part2

This section is discussing about how student can view grades and comments for the submission after grading.

View Comments and Grade

The studets are able to view the comments and grade for the submission after grading. They can access this from the manage submission page. By clicking on the source file, the student can view the comments for their code and by clicking on the grade file they can view the pdf grade file uploaded by the grader. Figure 5.22 shown below is the view of comments and grade Page for the student.

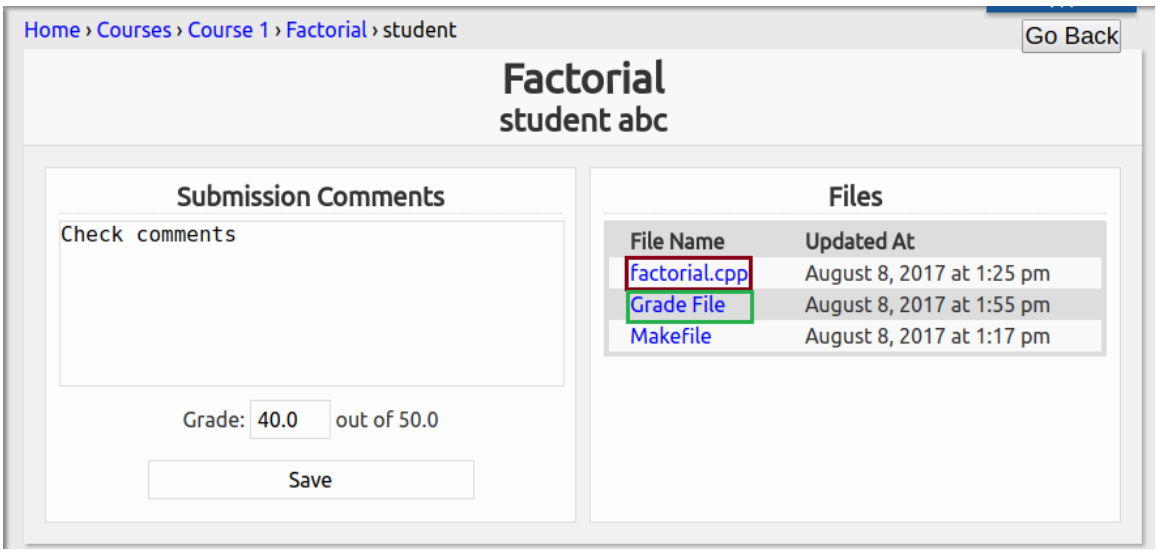

Figure 5.22: Sample View of Comments and Grade Page

View Comments

Figure 5.23 shown below is the student's view of comments for their code. It shows the comment for each line with line number.

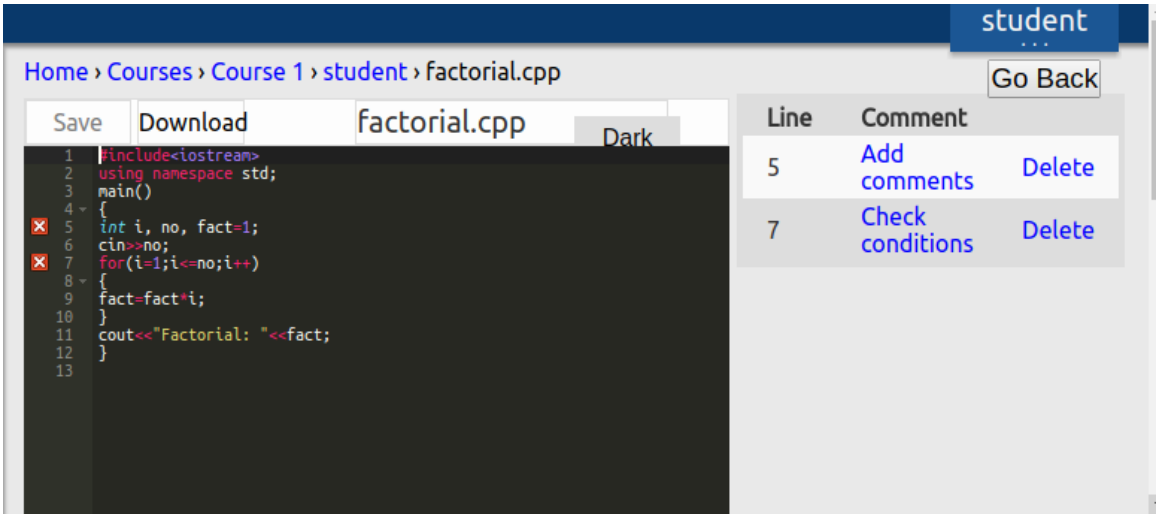

Figure 5.23: Sample View of Comments for the student code

View Grade

The student can view the grade file when the grader upload it. The grade file includes students name, grades, comments, the list of inputs, and indicates whether the test is passed or not. A sample grade file is shown in Figure 5.24.

GRADE
25.0 out of 50.0

NOTES
Late submission.

INPUT NAME	INPUT DESCRIPTION	PASS/FAIL
input_1	Inputs are 31 and 65	PASS
input_2	Inputs are 66 and 13	FAIL
invisible_input	Inputs are 45 and 46	PASS
input_3	Inputs are 56 and 23	FAIL
input_4	inputs are 678 and 2544	PASS
input_5	inputs are 567856 and 1321547	PASS
s2_input1	inputs 12 and 23	PASS
input_2	inputs are 5678 and 5656	PASS

Figure 5.24: Sample Grade file

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis presents a web based application for automatic evaluation of programming assignments for computer science classes which enables simple code submission and quick assignment grading. Submit has been built with Ruby on Rails and Flask and can utilize the features of both frameworks. This makes the application faster and efficiently handles multiple users. Rails has built-in security features and protect user data from outside and internal attack. Rails also offer simple integration with MySQL database, Jquery, Ajax, and JavaScript. The application design is simple and easy to use for instructors, graders, and students. This application allows instructors to create courses, manage courses, create assignments, and automatically grade assignments. The students are able to enroll in courses, submit their assignments, and test their programs against the public test case. The user will get instant feedback for their testing such as whether the test is passed or not, what output was expected, and the difference. This application reduces the effort and time needed to grade assignments. The Submit has an efficient and easy way to submit and grade assignment.

The software size and complexity has a great impact on the effectiveness of an application development. The software complexity depends on different interfaces used in the application, complex requirements, maintainability of the code, SQL complexity, use of frameworks, data communication and algorithms. This application is developed on two frameworks: Ruby on Rails and Flask. Rails provide a simple

and easy platform for web application development. The extensive libraries of the Flask reduce the size of code and reduce the complexity of the application. MySQL database is used to store all data and file in the application. JavaScript and AJAX are used for reloading and automatic updating page. HTML/CSS is used for creating web pages. It offers simple integration with MySQL database, Jquery, Ajax, and JavaScript, and HTML/CSS.

Software size measurement evaluates the application at the source code level and it helps to understand the size of the critical systems. A reliable software size measurement helps to implement a better managing programming practices. This application is designed with a Model-View-Controller architecture. There are thirteen controllers is used to develop this application. Also, thirteen model associated with the controller to manipulate data and communicate to the database. There is a view associated with each action. There are about five thousand lines of code is used to develop the application which includes model design, controller design, view design, integration of database, page updations, RESTful services, Flask App development, and integration of Celery and RabbitMQ with the application. Since the system is developed with two different frameworks and has many inputs, outputs, logic files, inquiries, interfaces, and data communication, the system can be considered as an average complex application.

6.2 Future Work

The Submit application is an efficient way to evaluate student programming assignments. But there are many more features and enhancements that can be added to provide the best system for automating assignment grading.

Plagiarism Checker

Plagiarism checking is the process of detecting the occurrence of plagiarism within the source code or documents [36]. In the case of assignment submission, the students may have copied the source code from the internet or from their peers, so it is necessary

to spot the plagiarism. This can be done by checking whether the submitted code matches with other submitted codes for the same assignment. Adding a plagiarism checker will make the application more efficient.

Monitor Job Queue

Currently, Submit has a job scheduling queue for creating outputs and testing submissions. In the future, the administrator should be able to view the tasks in the queue and manage them individually.

Peer Review

In the future, students shall be able to review the code anonymously and comment on other students code to give feedback and suggestions.

Upload zip file

In Submit currently, the user can upload their file one by one. In the future, users should be able to upload zip files for assignment submissions.

User Study

To study the effectiveness of the application, we are planning to conduct a user study in future. The first step to evaluate the application is to upload the website in the University domain, so all users can access the website. As part of the experimental user study, the response from participants is checked both quantitatively and qualitatively. For the user study, the users are divided into two categories: instructors, graders, and students and each group consist of 10 members. We prefer users with prior experience in teaching and grading in computer programming course. As part of testing, there will be a testing questionnaire and application testing session and receive feedback and suggestions from the users.

Prior to the experiment, each participant will complete a pretest questionnaire which includes demographic data as well as their familiarity with the computer programming assignments and their experience as an instructor, grader or a student for

computer programming courses. The experimenter explained about the application and demonstrate different use cases. After providing the instructions, the user will start the actual experiment. The users will be asked to do the assignment creation, submission, and grading with the newly developed assignment submission and automatic evaluation platform.

The users as instructors will ask to create a new course, create new assignments and test cases for the assignments. The user study for the student role has two parts. For the first part, the users as students will be asked to enroll into the new course and submit the solution for the assignment and test the assignment. If they are satisfied with test result against the public test cases, ask them to submit the assignment. If they want to edit the submission, ask them to open the code in a built-in editor for minor corrections. When they are satisfied with the result, ask them to submit. The users as graders will ask to enroll into the course with the join token. They can view all the student submission and grade. For grading, they will be asked to test the assignment with all visible and invisible test cases. Based on the result, they will be asked to comment on the student's code and upload grade for the assignment. For student testing second part, the users will be asked login and check the grade and comments for their submission. Finally, each participant will complete a post test questionnaire which includes usefulness of the application, the accuracy of the application, easiness to use the application and their suggestions and feedback. This user study will be helpful to check the usability of the application and robustness of the application in different conditions.

Bibliography

[1] Abraham Polishchuk. Building a restful api in a rails application. URL: https://www.airpair.com/ruby-on-rails/posts/building-a-restful-api-in-a-rails-application. [Accessed on 17 August 2017].

[2] AlloyUI. Tutorial-ace editor. URL: http://alloyui.com/tutorials/ace-editor/. [Accessed on 17 August 2017].

[3] Celery. Celery: distributed task queue. URL: http://www.celeryproject.org/. [Accessed on 17 August 2017].

[4] Codecademy. URL: https://www.codecademy.com. [Accessed on 17 August 2017].

[5] David Aragon. Introduction to database design on rails. URL: https://quickleft.com/blog/introduction-to-database-design-on-rails/. [Accessed on 17 August 2017].

[6] Stephen H. Edwards. Teaching software testing: automatic grading meets test-first coding. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, pages 318–319, Anaheim, CA, USA. ACM, 2003. ISBN: 1-58113-751-6. DOI: 10.1145/949344.949431. URL: http://doi.acm.org/10.1145/949344.949431. [Accessed on 17 August 2017].

[7] Eriksson U. Functional and non-functional requirements. URL: http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/. [Accessed on 17 August 2017].

[8] Flask. Welcome — flask (a python microframework). URL: http://flask.pocoo.org/. [Accessed on 17 August 2017].

[9] Flask Snippets. Per-request after-request callbacks. URL: http://flask.pocoo.org/snippets/53/. [Accessed on 17 August 2017].

[10] Instructure Inc. Instructure inc. URL: http://www.instructure.com/. [Accessed on 17 August 2017].

[11] JSON. Introducing json. URL: http://http://www.json.org/. [Accessed on 17 August 2017].

[12] Kalid Azad. Intermediate rails: understanding models, views and controllers. URL: http://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/. [Accessed on 17 August 2017].

[13] Shivprasad Koirala and Marla Sukesh. Learn mvc (model view controller) step by step in 7 days day 1. URL: http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in. [Accessed on17 August 2017].

[14] Larry Hardesty, MIT News Office. Automatically grading programming homework. URL: http://news.mit.edu/2013/automatically-grading-programming-homework-0603. [Accessed on 17 August 2017].

[15] Miguel. Using celery with flask. URL: https://blog.miguelgrinberg.com/post/using-celery-with-flask. [Accessed on 17 August 2017].

[16] MySQL. Documentation. URL: https://dev.mysql.com/doc/refman/5.7/en/entering-queries.html. [Accessed on 17 August 2017].

[17] Nascenia. Ruby on rails - controller. URL: //http://www.nascenia.com/ruby-on-rails-development-principles/. [Accessed on 17 August 2017].

[18] Nolan Burfield, Sergiu M. Dascalu, Hardy Thrower, Brandon Worl, Frederick C. Harris, Jr. Submit: an online submission platform for computer science courses. In *2015 INTERNATIONAL CONFERENCE ON COMPUTER APPLICATIONS IN INDUSTRY AND ENGINEERING*, CAINE 2015, pages 89 –95, San Diego, California, USA, 2015. URL: http://searchdl.org/index.php/conference/view/1003. [Accessed on 17 August 2017].

[19] Oracle. Mysql 8.0 reference manual. URL: http://dev.mysql.com/doc/refman/8.0/en/introduction.html. [Accessed on 17 August 2017].

[20] Pavel Pevzner. Smart teaching solutions. URL: https://stepik.org. [Accessed on 17 August 2017].

[21] Python Software Foundation[US]. Python 3.6.2rc2 documentation. URL: https://docs.python.org/3/. [Accessed on 17 August 2017].

[22] RabbitMQ. Using rabbitmq. URL: http://docs.celeryproject.org/en/latest/getting-started/brokers/rabbitmq.html. [Accessed on 17 August 2017].

[23] Ruby. Documentaion. URL: https://www.ruby-lang.org/en/documentation/. [Accessed on 17 August 2017].

[24] Ruby on Rails. Ruby on rails. URL: http://rubyonrails.org/. [Accessed on 17 August 2017].

[25] Jaime Spacco, David Hovemeyer, William Pugh, Fawzi Emad, Jeffrey K. Hollingsworth, and Nelson Padua-Perez. Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITICSE '06, pages 13–17, Bologna, Italy. ACM, 2006. ISBN: 1-59593-055-8. DOI: 10.1145/1140124.1140131. URL: http://doi.acm.org/10.1145/1140124.1140131. [Accessed on 17 August 2017].

[26] The marmoset project . URL: http://marmoset.cs.umd.edu/. [Accessed on 17 August 2017].

[27] Tutorials point. Ruby on rails development principles, explained. URL: https://www.tutorialspoint.com/ruby-on-rails/rails-controllers.htm. [Accessed on 17 August 2017].

[28] W3Schools. Css introduction. URL: http://www.w3schools.com/css/css_intro.asp. [Accessed on 17 August 2017].

[29] W3Schools. Html introduction. URL: http://www.w3schools.com/html/html_intro.asp. [Accessed on 17 August 2017].

[30] W3Schools. Javascript introduction. URL: http://www.w3schools.com/js/js_intro.asp. [Accessed on 17 August 2017].

[31] W3Schools. Json introduction. URL: https://www.w3schools.com/js/js_json_intro.asp. [Accessed on 17 August 2017].

[32] Web-cat home page. URL: http://web-cat.org/home. [Accessed on 17 August 2017].

[33] wikipedia. Flask (web framework). URL: https://en.wikipedia.org/wiki/Flask_(web_framework). [Accessed on 17 August 2017].

[34] Wikipedia. Functional requirements. URL: https://en.wikipedia.org/wiki/Functional_requirement. [Accessed on 17 August 2017].

[35] Wikipedia. Non-functional requirements. URL: https://en.wikipedia.org/wiki/Non-functional_requirement. [Accessed on 17 August 2017].

[36] Wikipedia. Plagiarism detection. URL: https://en.wikipedia.org/wiki/Plagiarism_detection. [Accessed on 17 August 2017].

[37] wikipedia. Ruby on rails. URL: https://en.wikipedia.org/wiki/Ruby_on_Rails. [Accessed on 17 August 2017].

[38] Wikipedia. Use-case. URL: https://en.wikipedia.org/wiki/Use_case. [Accessed on 17 August 2017].