GAMIFIED SELF-PACED E-LEARNING PLATFORM FOR COMPUTER SCIENCE COURSES

by

Yusuf Türk

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in
Computer Engineering

Yeditepe University

2017

GAMIFIED SELF-PACED E-LEARNING PLATFORM FOR COMPUTER SCIENCE COURSES

APPROVED BY:

Prof. Dr. Sezer Gören Uğurdağ (Thesis Supervisor)	[illegible] ....
Assist. Prof. Dr. Tacha Şerif	[illegible] ....
Assist. Prof. Dr. Tankut Barış AKTEMUR	[illegible] ....

DATE OF APPROVAL: ..../..../2017

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Prof.Dr. Sezer Gören Uğurdağ for her patience, help, and advice.

I also would like to thank my parents and my brother for their support.

ABSTRACT

GAMIFIED SELF-PACED E-LEARNING PLATFORM FOR COMPUTER SCIENCE COURSES

Gamification is the use of game elements in non-game context to increase engagement. One of the most important challenges in the last decade is to keep students engaged with the content while modernizing the education tools. E-learning systems are suitable to apply gamification techniques to increase student engagement. This work will introduce a self-paced e-learning platform for entry level Computer Science courses. For implementation purposes, Python is chosen since it is suitable for first year courses. Platform has modules for self-paced learning, laboratory homework (labwork), assignment, and management. It is implemented as a highly scalable web application with resources stored in the cloud. Course was tested with group of students submitting assignments and studying for their exam.

ÖZET

BİLGİSAYAR BİLİMİ DERSLERİ İÇİN OYUNLAŞTIRILMIŞ KENDİ-KENDİNE İLERLEMELİ ELEKTRONİK-ÖĞRENME PLATFORMU

Oyunlaştırma, oyun elamanlarını oyun dışı durumlarda kullanıp bağlılığı arttırmaktır. Son yılların en önemli sorunlarından bir tanesi de eğitim araçlarını modernize ederken bir yandan öğrencilerin bağlılığının devamlılığını sağlamaktır. Elektronik öğrenme sistemleri öğrenci bağlılığını arttırma amacıyla oyunlaştırmaya uygundur. Bu çalışma Bilgisayar Bilimi dersleri için kendi kendine elektronik öğrenme platformu sunacaktır. Uygulamayı geliştirme amacıyla ilk yıl derslerine uygun olan Python seçilmiştir. Platform kendi-kendine öğrenme, lab ödevi, ödev teslimi, ve yönetim için ayrı modüllerden oluşmuştur. Yüksek ölçeklenebilir ve verileri bulutta saklanan bir ağ programı olarak geliştirilmiştir. Sistem ödev teslimi yapan ve sınava çalışan bir grup öğrenci ile test edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

3D	Three Dimensional
AWS	Amazon Web Services
CPU	Central Processing Unit
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
edX	Education X
GB	Gigabyte
HTML	Hyper-Text Markup Language
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
MOOC	Massive Open Online Course
MVC	Model-View-Controller
NoSQL	Not only SQL
S3	Simple Storage Service
TCP	Transmission Control Protocol

1. INTRODUCTION

Gamification is the use of game elements in non-game context to increase user engagement. The rising popularity of gamification in the last decade introduced new approaches for the studies in various fields. One of the fields that gamification proved to be successful is learning. Gamification can increase the time students spend learning by using the game elements. Aside from introducing competition, gamification can also be used to form habits. Main elements used in gamified systems are rewards, badges, and leaderboards. Swarm (Foursquare) [1], Khan Academy [2], Duolingo [3], and Whale [4] are examples of popular platforms successfully implemented gamification.

Self-paced learning platforms allow students to interact with the system at their own speed. Popular e-learning platforms such as edX [5] and Coursera [6] are designed to emulate a classroom learning experience. In the case of Coursera, students are given lecture videos and resources and expected to complete the assignments before the deadline. Other platforms such as Code School [7], Code Avengers [8], and Hacker Rank [9] allow users to follow a curriculum at their own pace and does not have deadlines for projects. They also offer an online IDE which students can use to write and execute code. Programs are checked for correctness by the platform.

A platform to teach Python programming language is developed for this thesis. Python is chosen because it is easy to learn for people with little or no programming experience. It is also offered as the first programming language in many universities. The platform has a learning curriculum that students can follow at their own pace. In addition to that, there is also an exam module that students can work on assignments and submit them. Gamification is applied to the platform so that students can earn badges according to their accomplishments.

1.1. MOTIVATION

This research is providing a companion application that will improve the in-class performance of the students. The motivation for this research is the need to replace the traditional teaching methods for programming languages. By doing so, instructors can be more involved with the students by tracking their progress and act accordingly.

1.2. GOALS

The goal of this research is to increase engagement of students in learning and coursework. Although the platform is designed for the Python language, it can be easily applied to the other programming languages such as C, C++, and Java. By using the same application for different classes and programming languages this platform can become a standard for lab environment in universities. Also, having using the same platforms allows learners to start from the same experience level.

1.3. ORGANIZATION OF THESIS

Chapter 2 includes the background and literature review of the problem. Chapter 3 presents the methodology used to design the platform. Chapter 4 includes the implementation details of the platform. Chapter 5 includes the results and thesis is concluded in Chapter 6.

2. BACKGROUND AND LITERATURE REVIEW

This chapter describes e-learning systems, gamification, and user experience. Section 2.4. consists of the literature review of the projects using gamification.

2.1. E-LEARNING SYSTEM

E-learning is learning using technology to access educational material outside of the traditional classroom. There are two types of e-learning: synchronous and asynchronous [10].

Synchronous learning is the real-time learning that is mimicking a classroom environment using various tools such as video conferencing, webinars, and virtual classrooms. In synchronous learning, teacher and learners are in different locations but all participants are online at the same time. Coursera and edX are examples of this kind of learning.

Asynchronous learning is e-learning type that participants are following the same curriculum but they follow it according to their own schedule. Learners need not to be online at the same time. Self-paced online courses, forums, and message boards are examples of this kind of e-learning category. Khan Academy is an example of asynchronous learning.

2.1.1. Self-Paced Learning

Self-paced learning is an example of asynchronous e-learning platform in which the students are participating in classes or the given curriculum at their own speed. Learners can stop and continue studying independent of each other. Most of the self-paced learning platforms do not give a deadline to its students. Learners can complete their learning anywhere in the world and any time of the day. Self-paced learning can be a replacement for the traditional classroom for some cases. While being not appropriate for every learning scenario, self-paced learners can outperform the traditional learners. A study by Tullis and Benjamin [11] states that self-paced learners was more successful if they allocate their time according to the difficultly of learning material.

Learning paths can be customized according to each student. If students are not on the same level, students can take as much time as they want until they learned the curriculum [12]. They can also go back to previously covered topics if they need to. Khan Academy [2] also offers a self-paced programming module which teaches JavaScript. Students continue projects at their own pace.

2.1.2. E-Learning in Academia

E-learning is widely adapted in academia from blended and online learning to MOOCs. According to a survey study by European University Association [13], among the 249 participated institutions, 49 per cent of them have a institutional e-learning policies and strategies. Also, 53 per cent stated that the e-learning is widely used throughout their institutions. For the 27 per cent of the participant institutions give their objective of the e-learning as to provide more flexible learning, and 20 per cent wanted to increase the effectiveness of classroom time as seen on Figure 2.1.

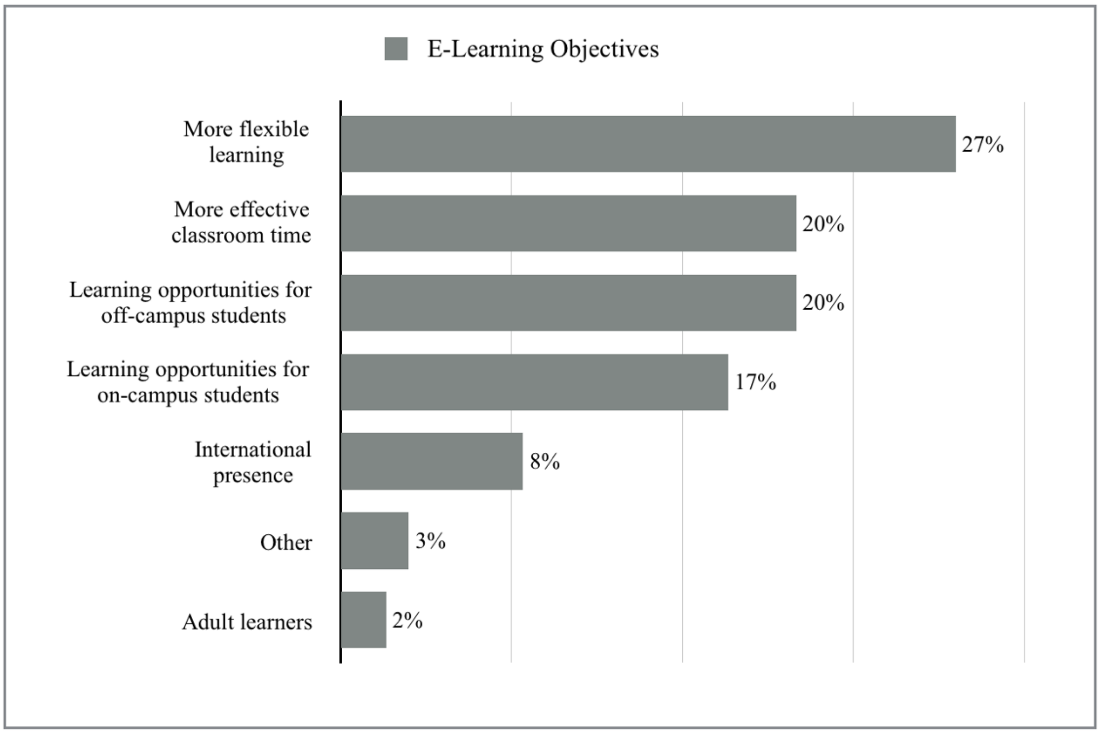


Figure 2.1. E-Learning objectives [13]

In addition to offering online learning systems, MOOCs are also popular among the universities. MOOCs offer learning for various topics including business, computer science, humanities, social sciences, psychology, and language learning. A MOOC is an online course often offered by universities which the students are given video lectures, reading materials, quizzes, exercises, assignments, and online exams. While some courses offer a certificate from the university, others choose to only award a certificate of participation. Each course has a syllabus that states the requirements of successful completion of the course. There are two types of timelines for these courses. First one is the self-paced approach that course is always available and students can take and finish the course any time they want. In the second model, course if only offered for a limited amount of time such as six weeks and the enrollment is closed after the first week. Most of the MOOCs are offered by timed basis, but some platforms are testing a self-paced model for the courses that previously offered on a timeline basis. While some universities offer their own platform such as MIT Open Courseware [14] and Stanford Online [15], others use a platform providing MOOC services such as Coursera and edX. Also according to the survey [13], only 12 per cent of the institutions are currently offering MOOCs. But, 46 per cent stated that they are planning to introduce them as seen on Figure 2.2. This shows that many institutions are trying to catch up and aware of the current trends.

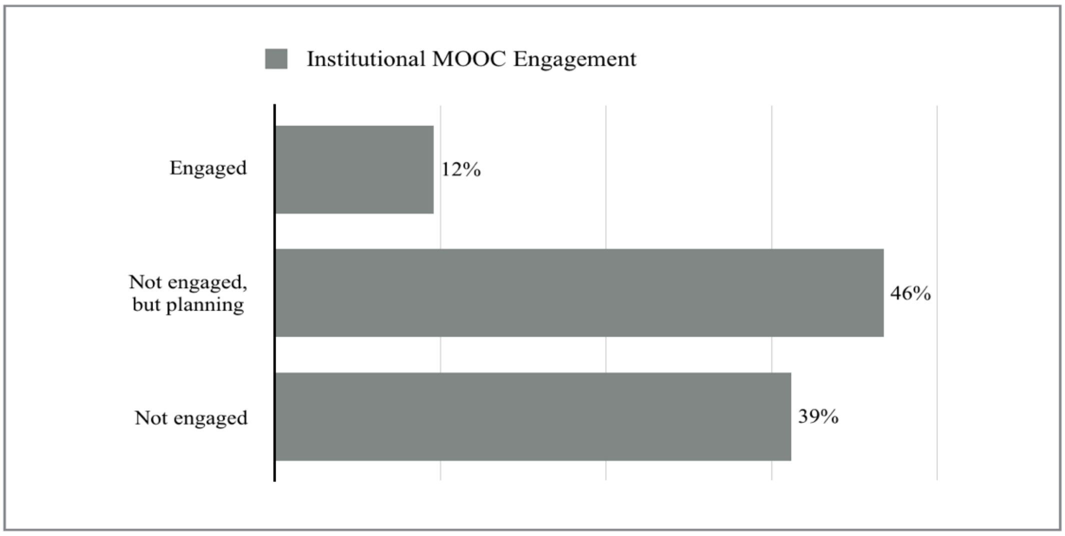


Figure 2.2. MOOC Engagement

2.1.3. E-Learning in Industy

Learning does not stop after the university education. Today, companies are also continuously educate their employees with corporate training. While some corporations get help from companies specialized in corporate training, others prepare their own materials since some information that will be thought to employees might be sensitive for the company. There are companies offering corporate training for every skill level from absolute beginner to expert. Platforms like Lynda [16] offer learning materials for thousands of subjects focused on the needs of the businesses, allowing employees to improve themselves on the specific subject. According to a study by Roland Berger [17], 77 per cent of the United States companies offered e-learning for their employees as professional development programs. In addition to that, MOOCs are also popular among the companies. MOOC providers partner up with the corporations to create courses specialized for their needs. According to an another industry report [18] for the corporate training professionals, 44 per cent of the companies intent to purchase online learning tools and systems for their employees.

Corporate training is now more important than ever. When a new technology comes out and it is evident that the old technology will be replaced, companies do not have a chance but the educate their employees to adapt to changes. This is where the e-learning shows its importance. When Google announced that the HTML 5 will replace the old HTML versions and Flash, employees needed to learn the new HTML 5 that has significant changes over the old versions. Google partnered with the Udacity [19] to create a new course on HTML 5 and enrolled 80,000 employees [20].

Companies also use e-learning to educate their customers on their products. SAP offers a training website [21] for their customers where they can attend to virtual live classrooms, e-learning materials, e-academy materials, and self-paced classroom. Training also includes study modules for the various certification programs for people wanting to specialized in their software applications.

2.1.4. Future of E-Learning

E-learning systems are replacing the traditional methods in some areas as mentioned in the previous sections. While replacing the traditional methods, it also creates completely new ways of using learning. For example, Udacity partnered with some companies such as Google, Amazon, IBM, Didi, Bosch, and Mercedes-Benz [22], to create a curriculum that responded to their needs from future employees. Some of the partnered companies agreed to have an interview with the students successfully completed the coursework and projects to hire them for the positions that respond to the technology learned in the class. There are also courses offered with contents that is not covered in universities. For example self-driving car engineering curriculum is not offered by universities. But, it is offered by the MOOC provider Udacity with industry partner Uber and Nvidia.

2.1.5. Traditional Classroom Teaching and E-Learning

E-learning is not replacing the traditional classroom teaching soon. Current university system is designed to teach students in classrooms and laboratories. And, the institutions are struggling to keep students in class. In some cases, attendance is mandatory to keep students in class. Universities are combining traditional and e-learning teaching rather than replacing them with each other. When e-learning is combined with the traditional system it is called blended learning. For example, the teaching is still conducted in classroom, but the assignments are submitted online or the lectures can also be followed from an e-learning platform. E-learning in this sense is not the same as distance learning. Distance learning is a learning method for the off-campus students so that

2.2. GAMIFICATION

Gamification: the process of adding games or gamelike elements to something to encourage participation. Gamification has become popular in the last decade. There are reasons that it is not considered as a simple tool and studied beyond the implementation.

First, it is an emerging business practice in various fields. From collecting airline miles for each flight to earning reward points for each cup of coffee, gamification is used to increase customer engagement and loyalty. Airline miles is an example of gamification focused on customer loyalty. In recent years, the sign up for loyalty programs become easier with mobile-enabled sign up. According to a study by 3Cinteractive [23], 59 per cent of customers made more store visits and purchases because of mobile-enabled loyalty programs. The reward might not be useful for the customer. According to a customer study by IBM [24], among the loyalty program offering brands, 70 per cent do not permit customers to choose their reward. For example, a coffee shop is only offering a small cup of coffee for a reward.

Second reason why the gamification is studied is that games are powerful things. In terms of the addiction, games and especially the mobile games became a subject of study. According to a study by Big Fish Games [25], 104 million people played mobile games in the United States only. Figure 2.3 shows the mobile game player statistics in several countries and estimations for 2020.

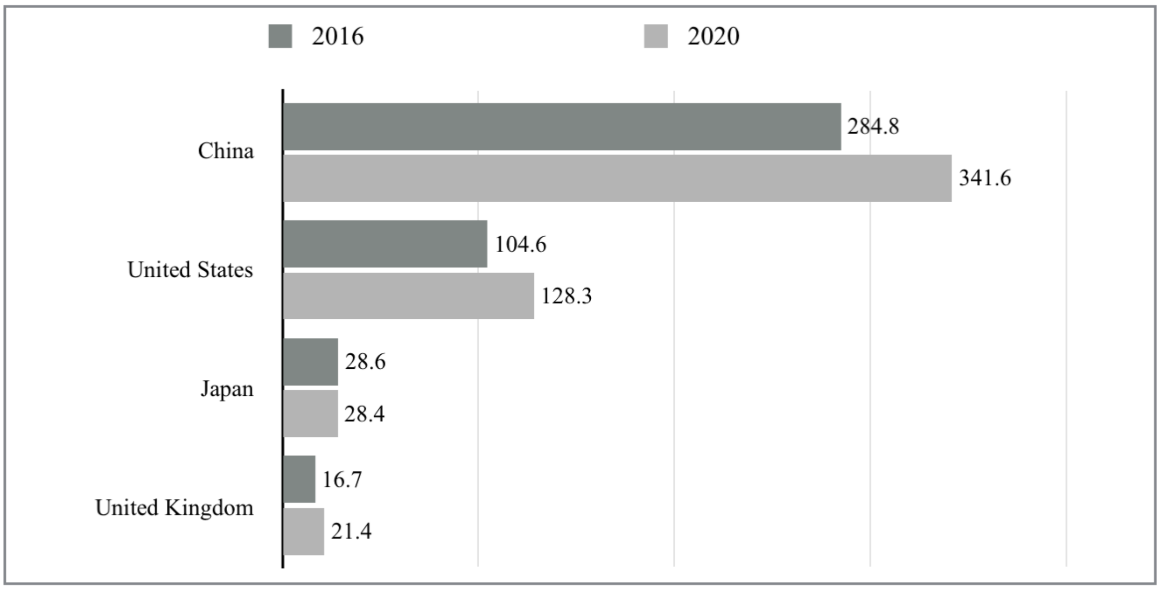


Figure 2.3. Mobile game users per country (millions) [25]

Increasing number of players shows that the mobile games have a decent following and the reasons why people keep playing them is a popular research subject. While not applicable for all cases, the main reason people keep playing them is because the games are addictive. An average user launches apps ten times a day, whereas a mobile addict launches more than 60 times a day [26].

Final reason gamification is studied is that there are lessons from psychology, design, strategy, and technology. Gamification systems are designed and optimized by analyzing multiple fields. For the psychological part, systems are designed to allow users to form a habit similar to addiction that allows the continual usage of the product. Systems are designed in a way that it should feel native to the user and this is a major design problem. After the product is designed, a strategy should be implemented so that the main goal is followed properly. This can increase the daily engagement of users.

Gamification is often misunderstood and compared to incorrect systems. It is important to know that what is not considered as gamification. Making everything a game or creating a 3D virtual world is not gamification because it does not create an experience that benefits an organization or individual. Use of games in workplace or business is not considered gamification if it is not intended to increase engagement. Also, simulations are not systems that is designed to increase individual engagement.

There are systems that include points and leaderboards, but they cannot be considered as gamified systems unless they provide a purpose to benefit an organization or an individual. There is also game theory that should not be mistaken with the gamification. Game theory is the study of human cooperation and conflict in competitive situations. It offers a mathematical model for competition. Game theory is not related to game design, whereas gamification uses game design elements.

2.2.1. Gamification Categories

There are three categories of gamification according to the book by Werbach and Hunter [27]. These are external, internal, and behavior change.

Figure 2.4 shows how the categories are formed. Internal is the type where organizations are benefitted from the gamification and users are either employees or the members of the community. External is the category where the organizational benefit is sourced from the individuals that are regular users of the system. Last category, behavior change is where the product is used to form a habit.

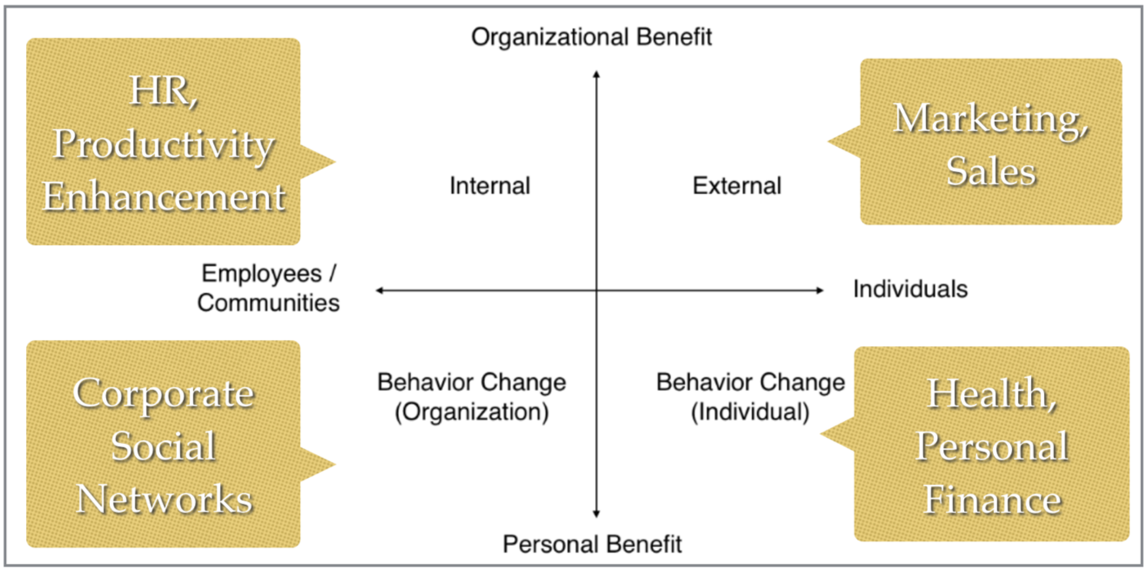


Figure 2.4. Gamification categories [27]

External category includes the systems that are prepared for the individual users which can be customers of the retail shop or users of the app. It is intended to increase marketing, sales, or customer engagement. .

Internal category includes the systems which provides value for the organization. The users are employees in the case of a company or a community in other cases. It could be an application for human resources or productivity enhancement for the company. .

Gamification is also used to form habits. The last category is behavior change which have the applications that are designed to either change the behavior of the community or individuals. Health, wellness, and personal finance applications are designed for individuals wanting to form a new habit.

2.2.2. Gamification Design Rules

The Gamification Toolkit [27] suggests that design rules should be followed to create a gamified system. There rules are gathered around three topics: experience, balance, and journey.

First, the system should offer the user an experience which can be something that user do to show their friends. For example, check-in application Swarm offers the experience of virtual check-ins for users to show it to their friends.

Second, the system should be balanced so that the each user can get some experience from the system. For example, if the system requires the user to use the application for a long time in order to get to a certain level, it would not be balanced for all users. If the Swarm application required users to use the app for a certain period after they checked-in to a place, some users would not get the same experience as other. Thus, the application is not balanced in the given cases.

Third, the user should be offered a journey or a path starting from beginner level to a mastery. It starts with creating a journey that all users starts with no experience. It should have a learning or experience curve that increase in knowledge of the system. When user masters everything in the system, the system will offer a different experience than someone who have just started. For example, in the case of the Swarm application, a user can master all the skills by going to different places and have different experiences. The mastery level is indicated with the number of badges.

2.2.3. Gamification Elements

Gamified systems have three elements: dynamics, mechanics, and components as seen on Figure 2.5.

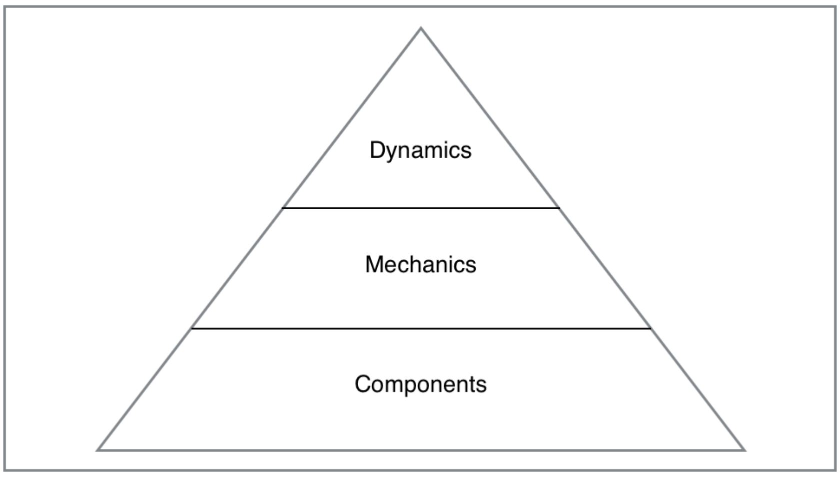


Figure 2.5. Pyramid of gamification elements

The dynamics of the gamification pyramid consists of constraints, emotions, narrative, progression, and relationships. Dynamics are at the top level of the design and they define the big-picture aspects. Mechanics level of the pyramid has the processes that drive the action forward. It consists of challenges, chance, competition, cooperation, feedback, resource acquisition, rewards, transactions, turns, and win states.

Finally, the components of the pyramid are specific instantiations of mechanics and dynamics. It consists of achievements, avatars, badges, boss fights, collections, combat, gifting, leaderboards, levels, points, quests, teams, and virtual goods. The most important elements of the system are points, badges, and leaderboards.

2.2.3.1.Points, Badges, Leaderboards

Elements points, badges, and leaderboards are the most important elements of the gamification systems. It is because they are the most visible elements to the users. The balance between them can completely change the user experience. In many cases users only see this elements, whereas other elements are implemented but not visible to the user. Figure 2.6 shows the elements of the Swarm application from the webpage.

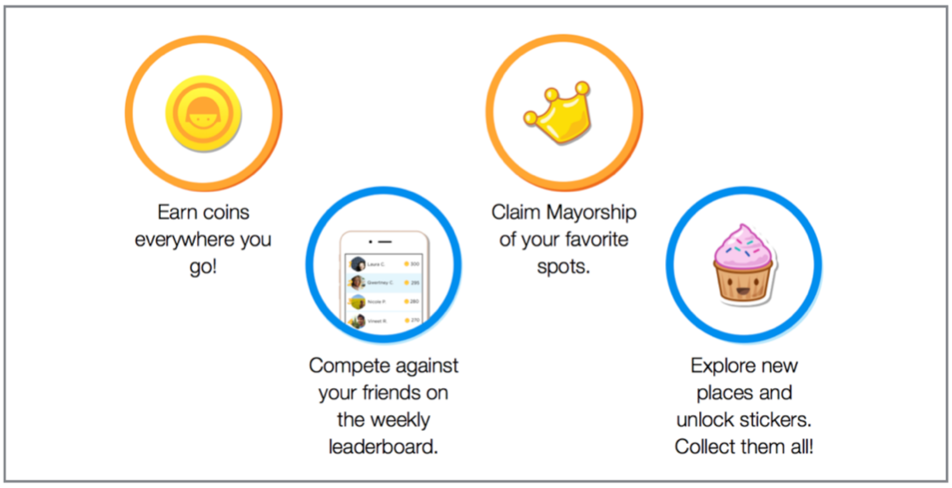
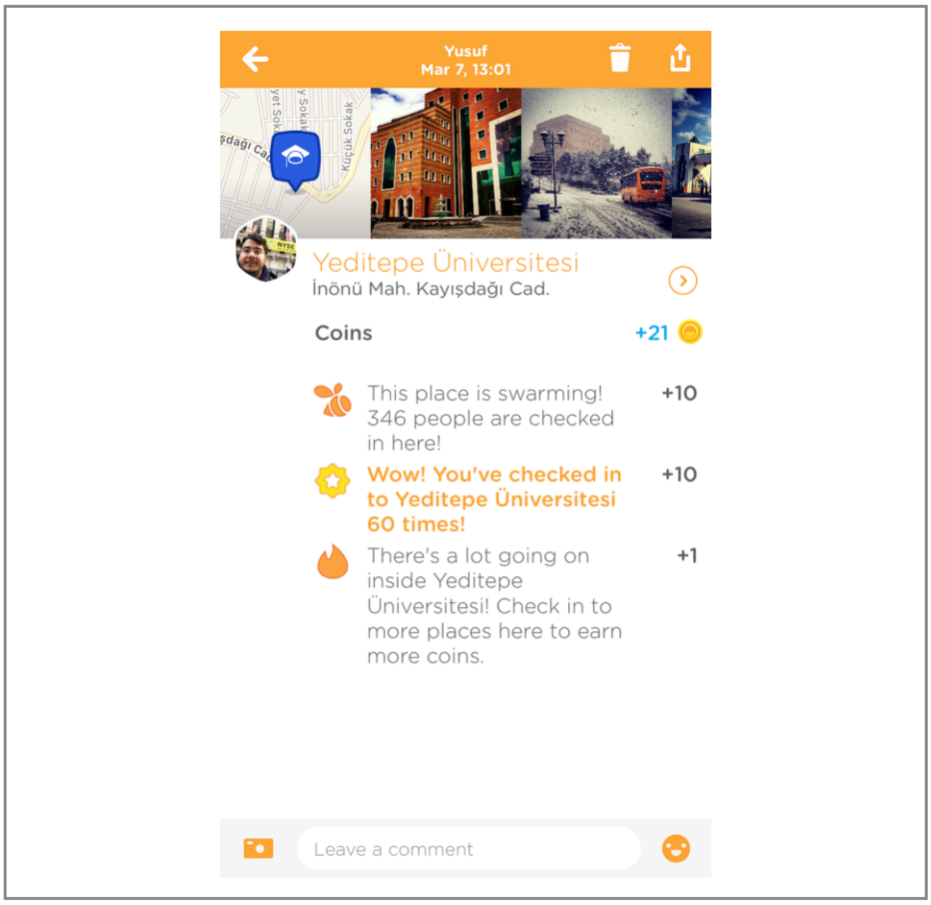

Earn coins everywhere you go!
Compete against your friends on the weekly leaderboard.
Claim Mayorship of your favorite spots.
Explore new places and unlock stickers. Collect them all!

Figure 2.6. Elements of the Swarm application

Points help to keep score when users are more active in the application. Each action has a point value that is designed to offer a balanced experience for all users. For example, in the case of the Swarm application, players earn points only when they check-in to a place as seen on the Figure 2.7. The amount of points determined based on the location and frequency.

Khan Academy [2] also offers a point based system where students earn points according to the length of the watched video.



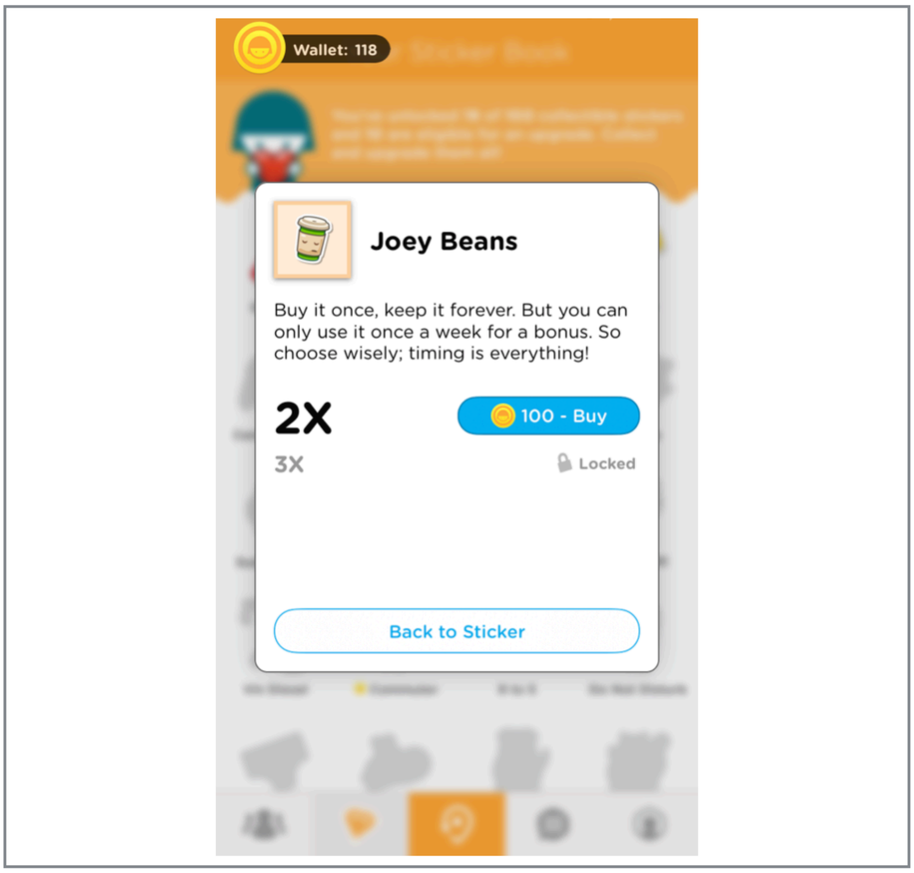
This place is swarming! 346 people are checked in here! +10

Wow! You've checked in to Yeditepe Üniversitesi 60 times! +10

There's a lot going on inside Yeditepe Üniversitesi! Check in to more places here to earn more coins. +1

Figure 2.7. Point system in Swarm application

When users earn points, they also have a change to earn bonus points or badges. System determines if they are in the win state which is a state of earning extra points or badges. Win states are determined by the designer of the gamification system. Users earn points to exchange them with the rewards. For example, Swarm application has a system where users spend points to upgrade the badges in the application as seen on the Figure 2.8. The points are both for exchanging for rewards and leaderboards.



Buy it once, keep it forever. But you can only use it once a week for a bonus. So choose wisely; timing is everything!

Figure 2.8. Upgrading a badge in Swarm application

Letting users earn rewards with their points increases the engagement with the product. For example, airlines offer miles with their own loyalty programs. For each flight, passengers earn miles according to their ticket group and distance they flown in the particular flight. The reason why airline passengers are signing up in the loyalty programs is because when they have earned enough miles, it can be exchanged with a ticket. Passengers use the same airline to collect more miles for rewards.

Point based system offer so much for the system designer. For example, designer can analyze how users earn and spend points to get a better understanding of their users. If this is a location based application, designers can learn which cities, neighborhoods, and type of places users use their system. If the users are using the system to get deals on hotel accommodation, it will be wise to promote the application for hotels.

Badge is an another important element of gamification. Badges are acquired when users complete a predefined achievement in the system. Badges can be visible to other users as a showcase for achievements of the user or it can only be visible for the current user. Badges are used to represent achievements that user is earned by continuously using the same system. They also give a sense of personalization. Figure 2.9 shows the badges offered by the application Swarm. Application offers stickers for frequent check-ins. For example, if user is frequently visiting airports, user receives a badge with a plane icon.

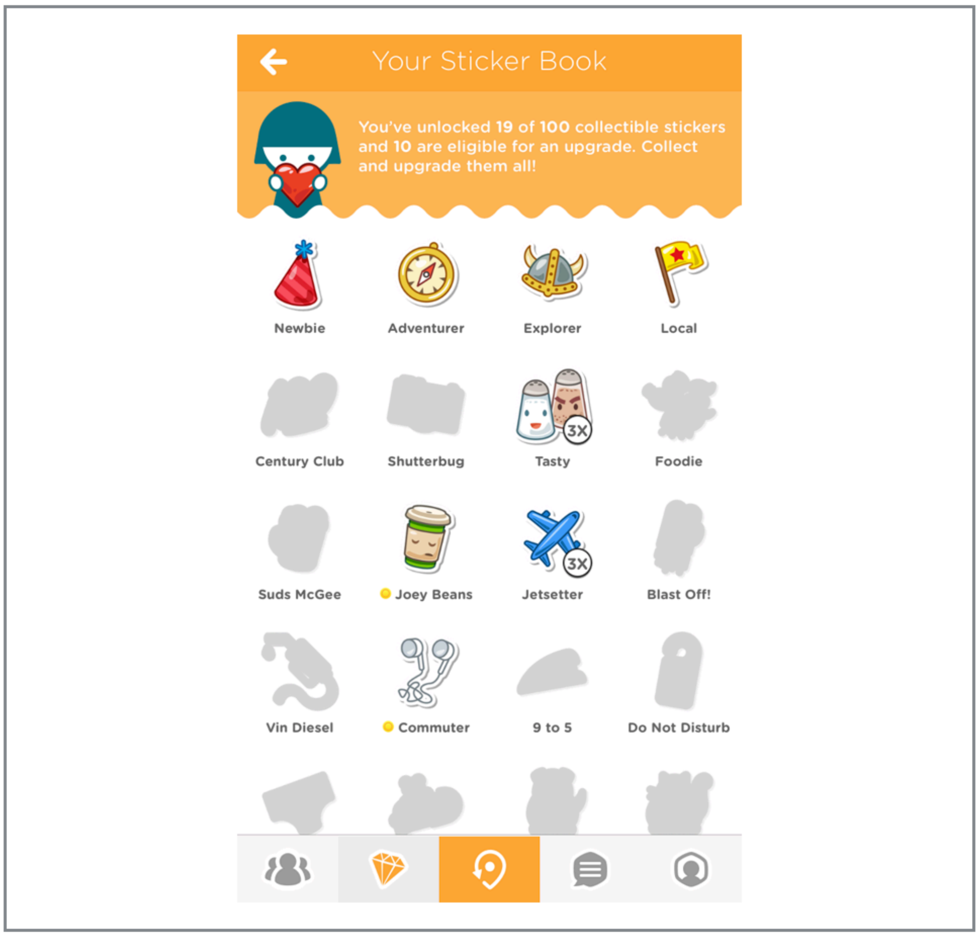
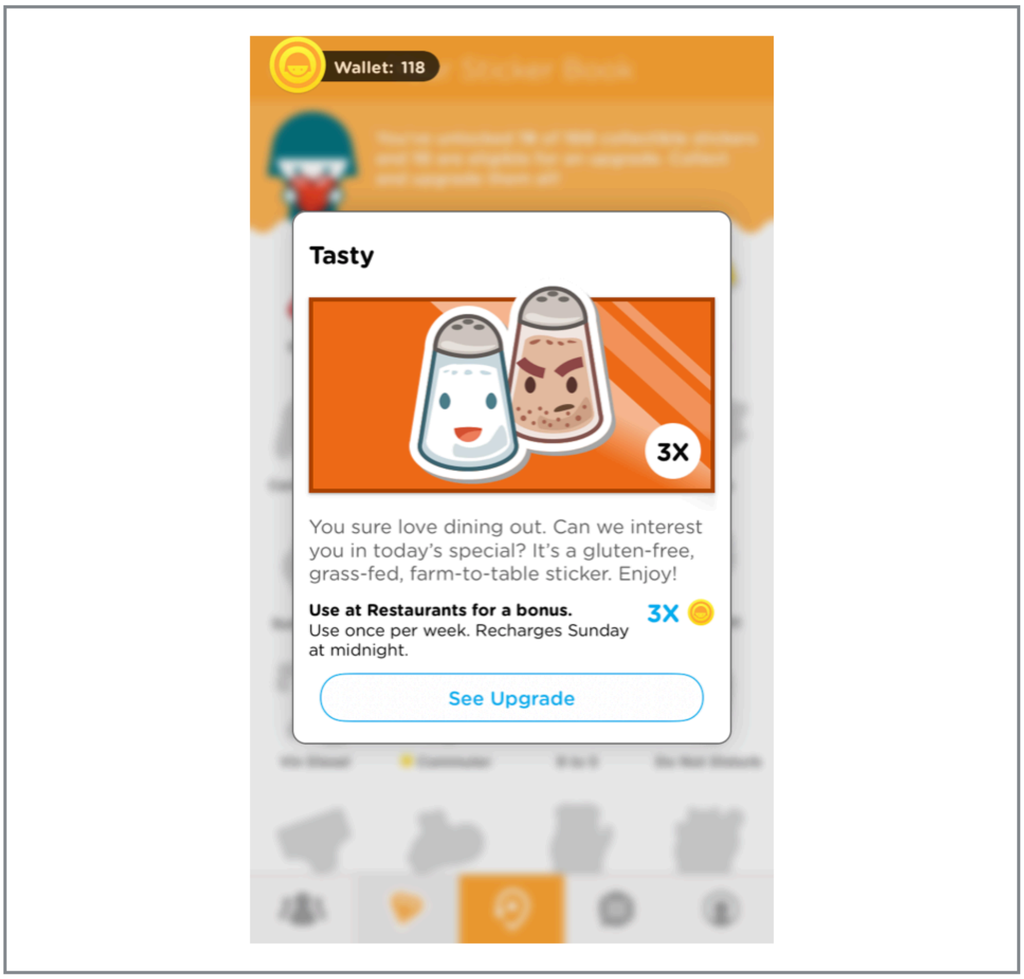


Figure 2.9. Swarm application stickers

Badges play an important role in keeping users constantly engaged with the application. Badges create an emotion bond between application and the user. Sense of ownership is the most important point of user loyalty. It is harder to switch applications because users feel that they will lose progress if they switch to another application. Badges are earned and showcased in a separate menu or in the user profile. Some applications offer engagement with the badge after they are earned. For example, in the application Swarm, when a badge is earned, it is not only used for showcasing. Users engage with the badge in two ways. First, the badge is used to increase points earned for each action. For example, if user uses a food related badge at a restaurant, it increases the points earned in that action. Second, badges can be upgraded with points to higher levels, so that they can give more points for each action. Figure 2.10 shows a badge in Swarm application used at restaurants.

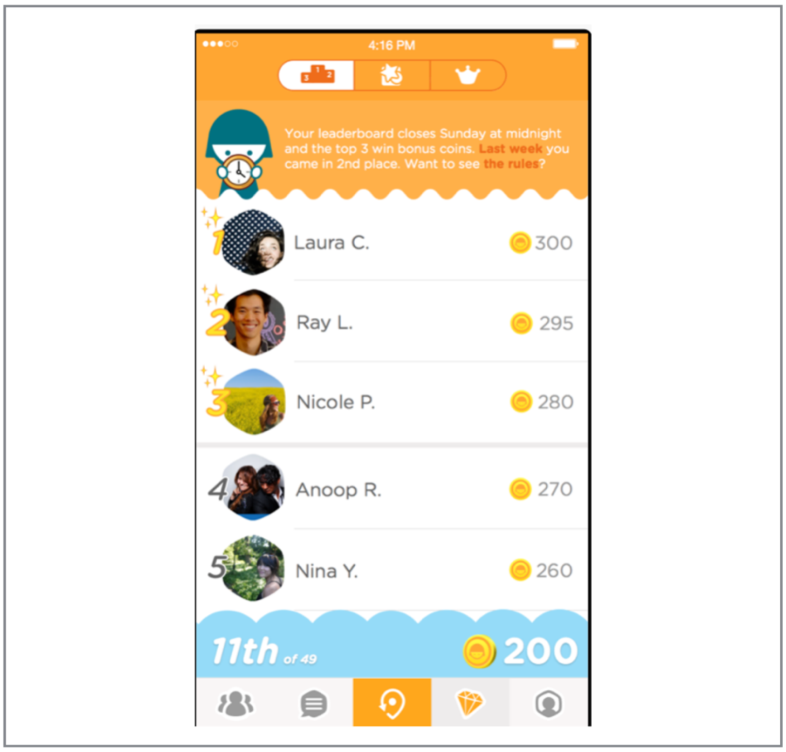


You sure love dining out. Can we interest you in today's special? It's a gluten-free, grass-fed, farm-to-table sticker. Enjoy!

Use at Restaurants for a bonus. 3X
Use once per week. Recharges Sunday at midnight.

Figure 2.10. Swarm application restaurant badge

Leaderboard is an another element of the gamification components. A leaderboard is simply a point based list showing points earned by each user as seen on Figure 2.11. Leaderboards are visible for all users. This allows competition among all users. While some platforms make the leaderboard the main element in the system, others do not use it. It depends of how the system is designed. If this is a system focused on personal development and habit forming, there is no need to include a leaderboard. The progress of the user can be visualized by different tools such as graphs, charts, and progress bars. The leaderboard ranking can be global, localized, or personalized. Global leaderboard includes all users from around the world. This is a useful case for highly competitive systems. Localized leaderboard consists of users from a certain country, city, or institution. For example, the career oriented social network LinkedIn [28] uses a separate leaderboard for the people in users institution. Users can see the most active users from their institutions. Personalized leaderboards consist of the users pre-approved by the user. This can be small community or a friend list. This can be used if the user is sharing personal information with the system. For example, in Swarm application users share their location and leaderboard in application consists of only the friends of the user which are pre-approved.Figure 2.11 shows a sample leaderboard from the application.



Your leaderboard closes Sunday at midnight and the top 3 win bonus coins. Last week you came in 2nd place. Want to see the rules?

1	Laura C.	300
2	Ray L.	295
3	Nicole P.	280
4	Anoop R.	270
5	Nina Y.	260

11th of 49 200

Figure 2.11. Swarm application leaderboard

2.2.4. Engagement

Increase in engagement is the main goal of the gamification. Engagement is maintained through two subjects which are rewards and motivation. They are the reason why gamified systems are successful.

Rewards are everything earned by the users interacting with the system and gaining more achievements. Rewards can be in different forms. According to Zichermann [29], rewards can be status, access, power, or stuff. Status is a position of a player in the system. For example, Swarm application is awarding a status called ‘mayor’ for the most frequent user in a place. Access is allowing users to have information or objects that is not available for other users. For example, the hotel booking website Booking.com [30] lets users to see special offers after users booked rooms using their website five times or more. Giving power over other players and information is an another reward. For example, more frequent users can earn more points for each action. Last type of reward is stuff which are the things only few or no other users can get.

Rewards are given on schedule to keep the system balanced. Also, if not balanced, system can be vulnerable to abuse. Common reward schedules are continuous, fixed ratio, fixed interval, and variable. .

There are two types of motivation in gamification. These are intrinsic motivation and extrinsic motivation. Intrinsic motivation is the self-desire to seek out new things and new challenges, to analyze one’s capacity, to observe and to gain knowledge. Extrinsic motivation is performance of an activity in order to attain a desired outcome.

2.2.5. Gamification Design Framework

Systems using gamification follow various design rules. One of the frameworks is the 6D framework [31]. The steps of the framework are given below.

i. Define business objectives

ii. Delineate target behaviors

iii. Describe your players

iv. Devise activity loops

v. Don’t forget the fun

vi. Deploy the appropriate tools

First step of the framework is defining business objectives. Every system must have an objective so that the gamified system to be designed accordingly. For example, objective can be improving sales or increase the time people spend in an application.

Second step is describing target behaviors. It should be as specific as possible. Targets are the specific actions players should take while using the system. Each target has a success metric such as a win state that shows if it has been successful or not. Most important analytics metrics of the target are virality and volume of activity.

Third step is based on the Bartle model of player types [32]. According to Bartle, there are four types of players which are killers, achievers, socialites, and explorers. Each platform should respond to needs of the multiple player types. Since gamification design is based on game design, player types are very similar to the user behavior in gamified systems.

Killer player type consists of users focused on winning, rank, and direct peer-to-peer competition. Achiever player type focuses on attaining status and completing all achievements. Socialite player type focuses on socializing rather than focusing on competition. They develop a network of friends and contacts while playing. Last player type explorer has a desire to explore and discover.

Fourth step is the defining the activity loops. An activity loop consists of action, motivation, and feedback. Figure 2.12 shows how the loop elements are related to each other. Motivation triggers the action. For example, if it is a weight loss application, action is defined by the motivation which is users desire to lose weight. Result of the action is analyzed and motivation is updated accordingly.

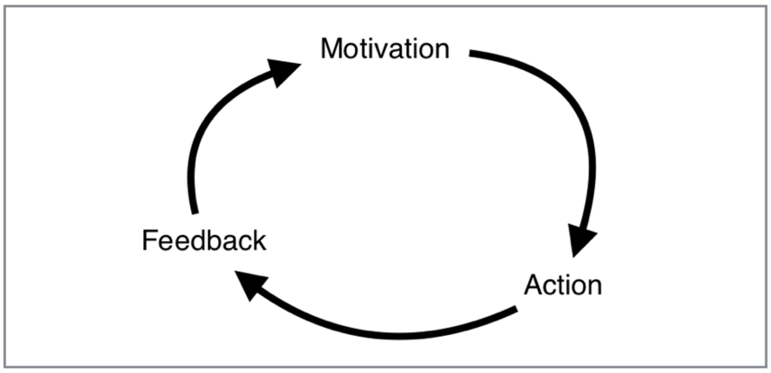


Figure 2.12. Engagement Loop

Fifth step is about not forgetting the fun. This might not look like a real step but it is important. Gamified systems are supposed to be fun and entertaining. The process of gamification is about making the non-game topic feel like a game.

Last step of gamification design framework is deploying the appropriate tools. When the design of the system is finished, designer should choose with tools to use for implementation.

2.3. LITERATURE REVIEW

Main goal of the gamification is to increase engagement of users using the system. Regalado et al. [33] prepared a virtual learning environment for teaching programming. Gamification is applied to their system by giving points to each video play, introducing leaderboards, and challenges. The study resulted in higher number of video plays, higher frequency of learning environment visits, and more competitive environment.

Ibanez et al. [34] prepared a gamified system that targets to teach C-programming language to students. It included work activities which students earned points, planning activities with exercise questions, and social activities with micro-blogging. Achievements of students are displayed in leaderboard, phrase showcase, and badge showcase. The study resulted in most students continue working even after they earned the maximum grade. It was because the students wanted to earn all badges.

A virtual 3D environment is created in a concept study by Lückemeyer [35] to decrease the drop out rates in university classes. A virtual classroom is created and students can take live lessons without going to a physical classroom. Aside from taking live lessions, students can also interact through their profiles. Although, the concept is not tested in classroom, study is planned to be used by advanced students for feedback sessions. By doing so, students can work on the same task to be completed in single session.

E-learning is combined with game learning in some cases. In a study by Maia and Graeml [36], a game called factory game is created for classwork. Authors stated that although the new generation is more familiar with the technology, the teaching methods are not using a language with the same technology familiarity. And, games and competitions can provide a long-term learning experience. In the designed game, students use distributed programming and concurrency to organize threads and teams complete each other for best production line. According to the study, students developed skills such as teamwork, communication, and problem solving. A survey conducted with students showed that their negative thoughts on this learning type is decreased.

Another example of learning with games is a work by Tsalikidis and Pavlidis [37]. An online multiplayer platform game is developed to teach JavaScript. The purpose of the project is to use role-playing game methods and apply them to programming language education. Platform offers three modes such as single player, multi player, and player versus player. The goal of the project is to engage learners with different problems and make them continue playing by solving problems. In role-playing games, despite being similar to each other, each encounter is unique. Thus, each problem students encounter is also unique.

Gamification could be unsuccessful even if it is designed properly. In the study by Berkling and Thomas [38], a platform is developed to increase engagement and motivation of students. Aside from confusing the users with gamers and assuming that there is a relation between playing games regularly and gamification, a system is designed by using gamification elements. According to their survey, students were motivated by relevancy, clear path, and grades. In the study evaluation, they found out that students are not gamers and also, students are not ready for their idea. Not all classroom environments and curriculums are suitable for gamification. In this case, it was viewed irrelevant towards exam study.

Jayasinghe and Dharmaratne [39] studied both game-based learning and gamification. In their study, they applied both techniques to a Computer Science course teaching algorithms. As a result, it was shown that when offered a gamified system, students follow a learning process known as Bloom’s taxonomy. But in the game based learning system, students did not follow each step which resulted in poorer learning experience. Gamified system proved to be better for the difficult academic subjects.

Gamified and non-gamified versions of the same are compared to each other in a study by Barata et al. [40]. The study applied gamification elements such as points, badges, levels, leaderboards, and challenges to a graduate level course. The results were compared to non-gamified version of the same course. Results showed that students were motivated while using the gamified version. Also, using challenges allowed more engagement such as forum participation and reading more slides.

Thesis work by Uzun [41] introduced a system to create gamified lessons. The work focused on dividing the lessons in small gamified elements. It offered an index to help the process of gamification. Aside from the lessons, system can also be used to gamifiy lectures, exams, grading, and assignments.

3. ANALYSIS AND DESIGN

The goal of this thesis is to create a self-paced e-learning platform that students will use to learn programming languages. In order to keep students engaged, gamification elements are applied to the system such as badges, and progress bars. The platform is a web application so that it is available for students regardless of their location. Python programming language is thaught using the application through self-paced learning module, assignment module, and labwork module. Following sections cover how the platform, architecture, and components are designed.

Python is selected because it is one of the most popular programming languages and many institutions teach Python in their first year curriculum. According to a research [42] to rank programming languages by popularity, Python ranked third among all languages. The popularity rankings were determined by amount of public repositories, number of questions asked on question and answer websites, and number of forked repositories.

3.1. PLATFORM OVERVIEW

The e-learning platform have the modules for self-paced learning, labwork, assignment, and management as seen on Figure 3.1. The self-paced learning module is the part where the learning curriculum of the Python programming language is offered. Learning module consists of sections. Each section has short explanations with examples, a quiz, and exercises. Labwork module is the part where students are offered exercises of the subject covered in class that week. Assignment module is the module where weekly or timed assignments are offered to students. This module can also be used as an exam module. Management module offers a dashboard and a labwork creator to the instructor.

The management module is only visible to the instructor. Self-paced learning module is always accessible to the students. Labwork module can be visible or not visible depending on the creation parameters. Exam or assignment module is only visible for the duration of the assignment. If instructor decides to close the submissions, this module becomes inaccessible.

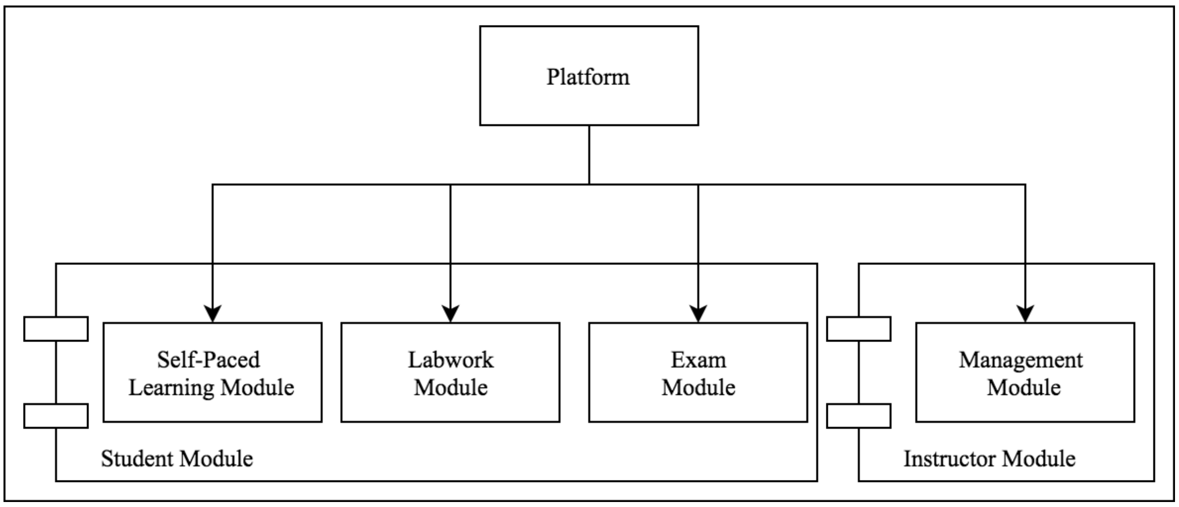


Figure 3.1. Modules of the platform

Self-paced learning module allows students to learn at their own speed. Figure 3.2 shows the use case diagram. Student logs into the module, follows instructions and complete the content, quiz, and exercises whose result updates the user progress. Instructor can access the user progress.

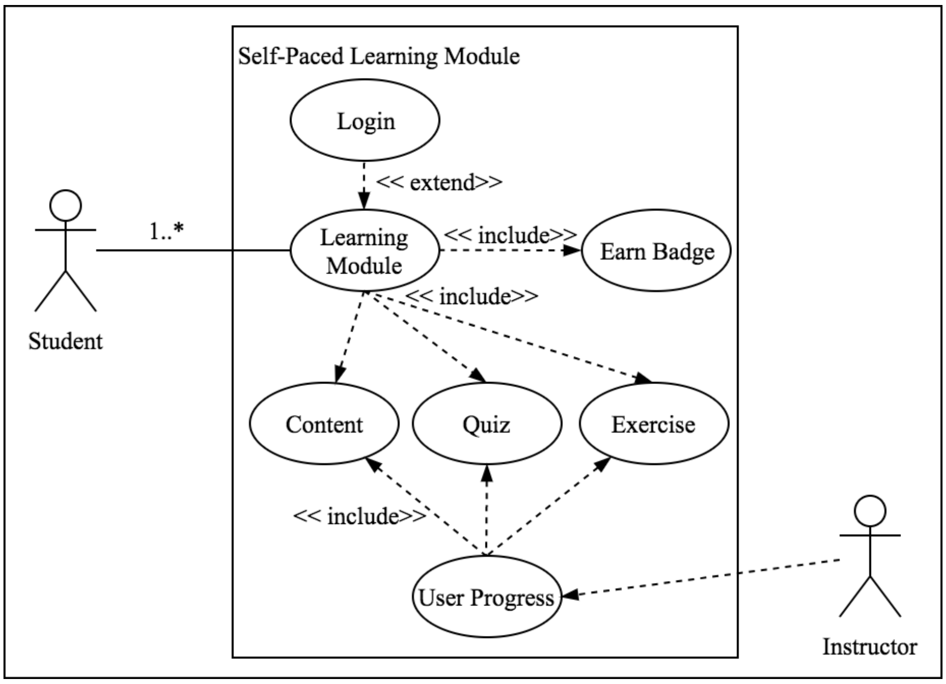


Figure 3.2. Use case diagram for learning module

Labwork module consists of a set of questions on subjects covered recently on class. Labwork questions can be changed by the instructor using the labwork creator. Figure 3.3 shows the use case for labwork module.

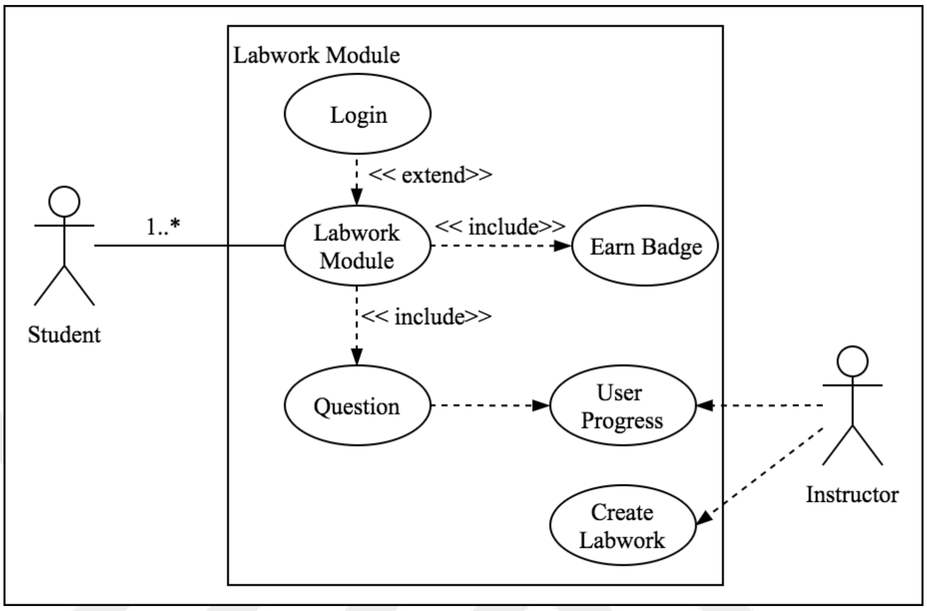


Figure 3.3. Use case diagram for labwork module

Gamification is implemented in the platform by using badges, and progress bars. Since this platform is emulating a classroom environment that will be used by the students of the same classroom, points and leaderboards were not used. Point based system is replaced with a progress bar that shows the completed percentage of the assignment. Also, a leaderboard is not used because in classroom setting, competition might discourage other students that are left behind. Self-paced learning systems are successful among the students because it is not discouraging for the slow learners, students falling behind of the class schedule, and introverted students. In the end, all classroom based learning systems have grading in effect which is the same as a leaderboard.

Progress bar is a replacement of a point based system that shows how far student is completed an assignment. For example, Figure 3.4 shows a progress bar for an assignment.

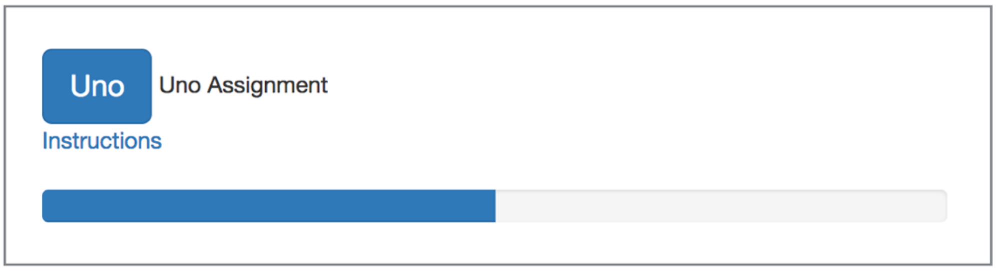
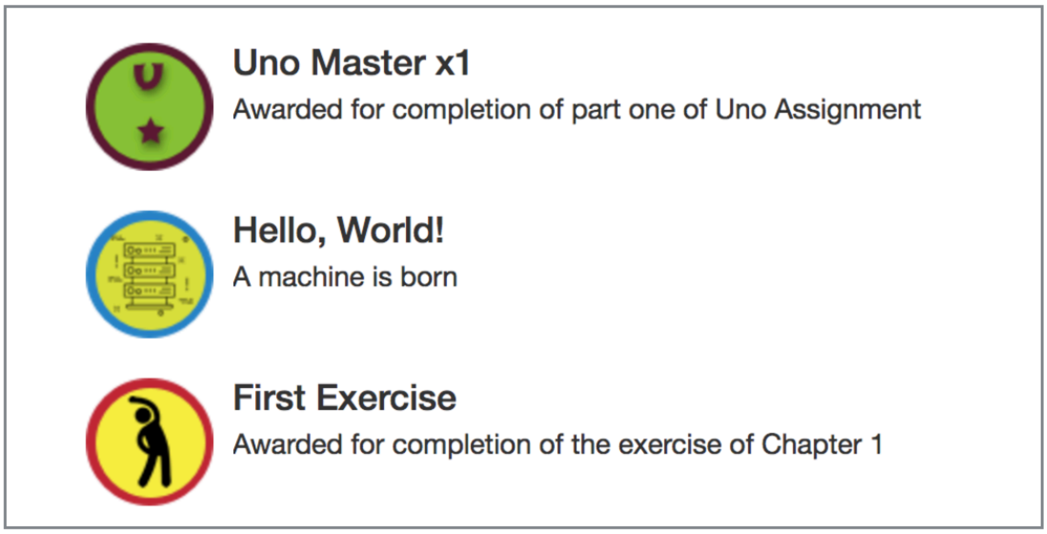


Figure 3.4. Progress bar for assignment

Badges are similar to physical tokens that users earn after completing certain achievements. For example, completing an assignment completely can have a corresponding achievement. Figure 3.5 shows example badges from the system. ‘Uno Master x1’ badge is awarded for completion of part one of the assignment. ‘Hello, World!’ badge is awarded when a student print their first ‘hello, world’ statement.



Uno Master x1
Awarded for completion of part one of Uno Assignment

Hello, World!
A machine is born

First Exercise
Awarded for completion of the exercise of Chapter 1

Figure 3.5. Badges from the system

3.2. ARCHITECTURE DESIGN

The platform designed in this thesis is a web application that uses a cloud database. So that students will use the system independent of their location. This is useful for two reasons. First, since the system is used to teach Python to first year students with little or no experience in programming, it is easier to skip the installation part for each student. By doing so, students have a chance to start learning immediately. Also, some students might have problems with installation or could not have a proper computer. Second, every code submitted by student is saved by the system. Students do not have to save their code manually.

Platform consists of a web-application framework, server scripts, and cloud database. Figure 3.6 shows the components of the platform. Application is used to serve website data, create and store profiles, and provide an online integrated development environment. Server scripts check the code written by the students for correctness and apply other tests. Also, scripts create containers in server. Cloud database is used to store content and user information.

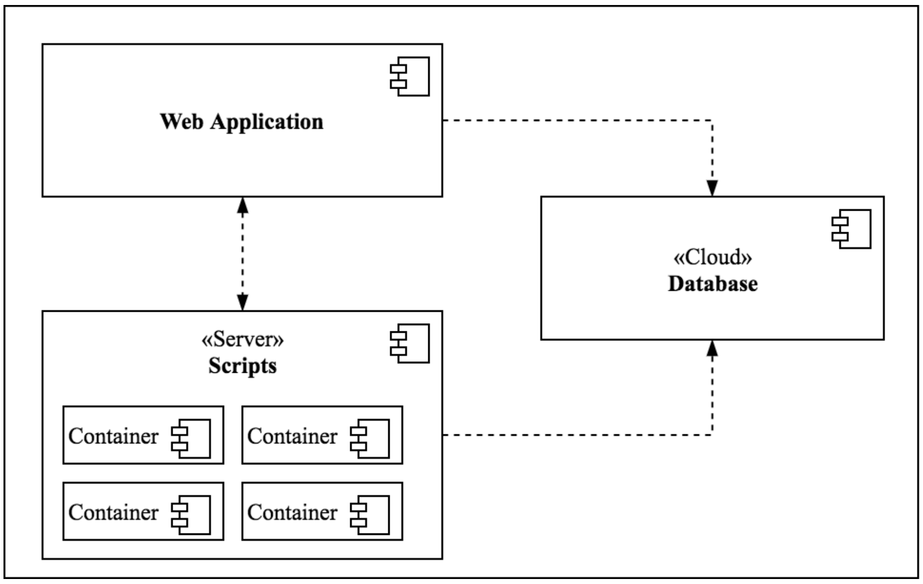


Figure 3.6. Platform architecture

Web applications must be designed to be scalable when needed. This is why the components of the platform is designed to be scaled as much as needed. Since the content and user information is stored in a cloud database, underlying architecture can be scaled as much as needed. For example if the server cannot handle the user load with its current containers, an another server can be added to the system without any problem as seen on Figure 3.7.

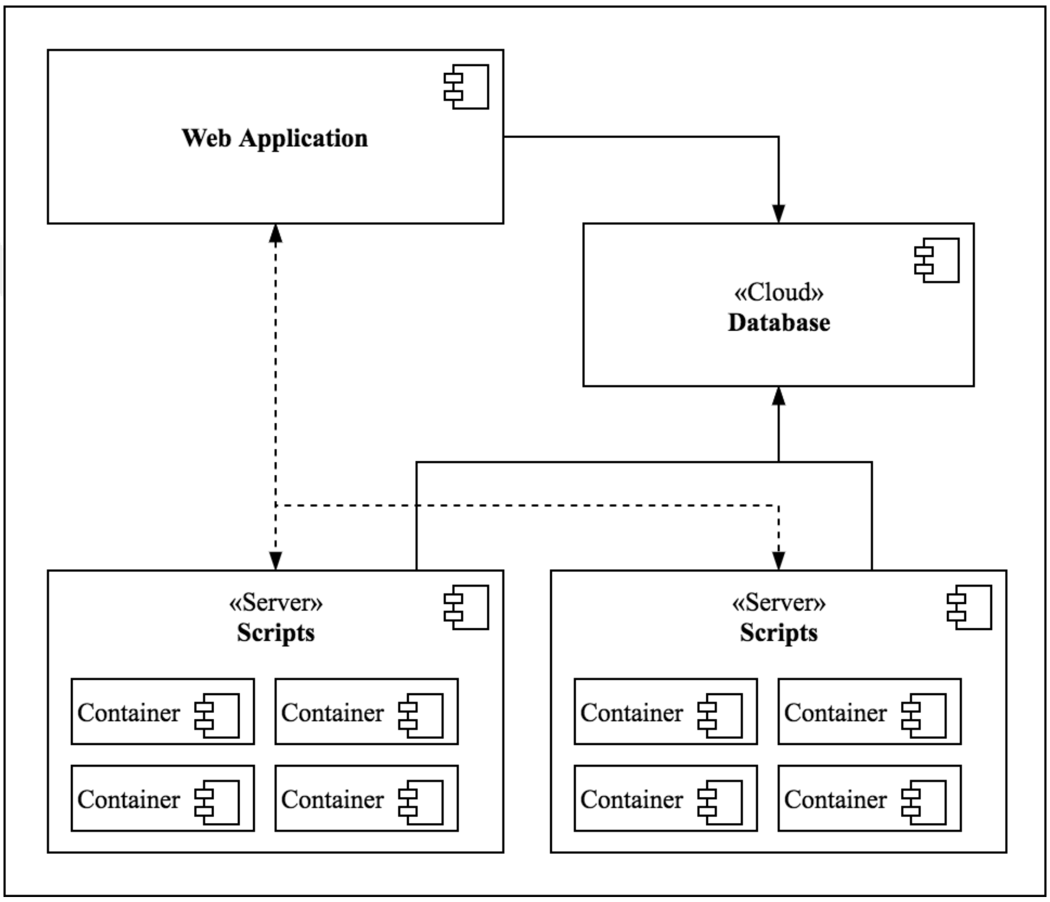


Figure 3.7. Platform architecture with two servers

Aside from scaling the server, the server storing the web application can also be duplicated to serve more users as seen on the Figure 3.8.

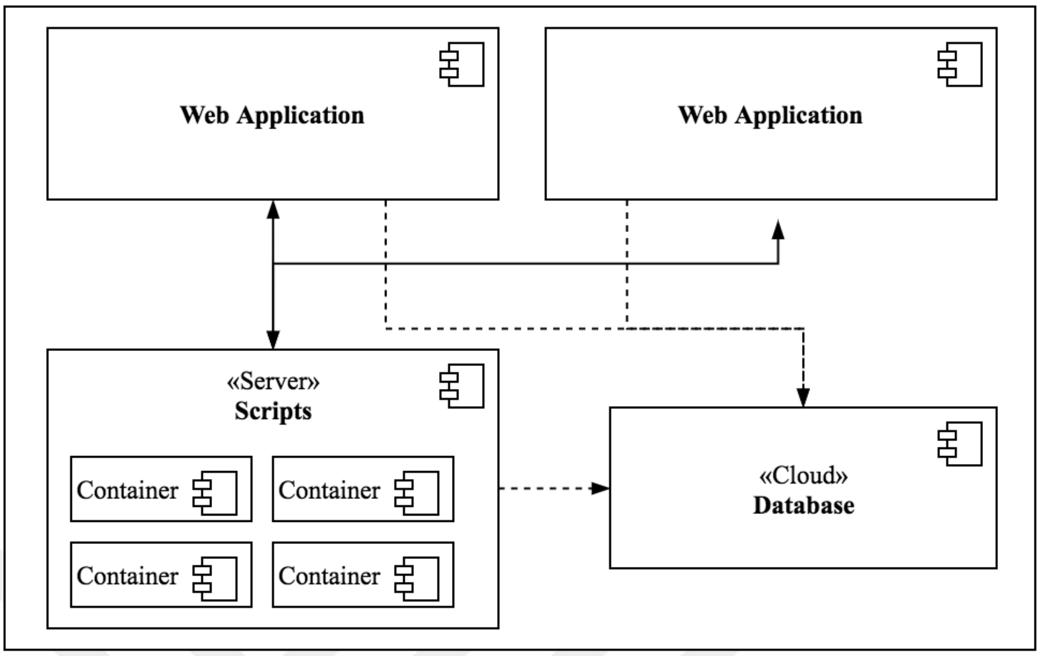


Figure 3.8. Platform architecture with two web application servers

3.3. COMPONENT DESIGN

The components visible to the user are self-paced learning, labwork, assignment, and management module. Aside from that there are user profiles, class dashboard, and labwork creator.

3.3.1. Self-Paced Learning Module

Self-Paced learning module is the module where students will follow a curriculum to learn Python programming language. The module consists of sections such as introduction, variables, lists, and operators. Sections have micro contents, quizzes, and exercises as seen on Figure 3.9. Micro contents have a detail about the current section and offers an example for student to try it out in the terminal.

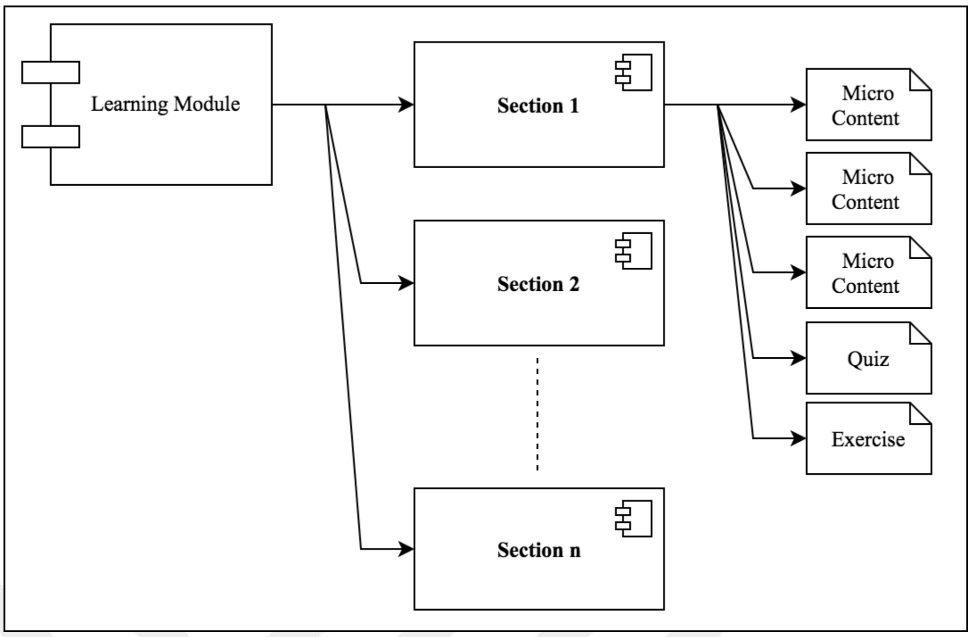


Figure 3.9. Learning module components

The user interface of the module consists of an instruction panel, code editor, console, and buttons for running code and continuing. Figure 3.10 shows the user interface mock up.

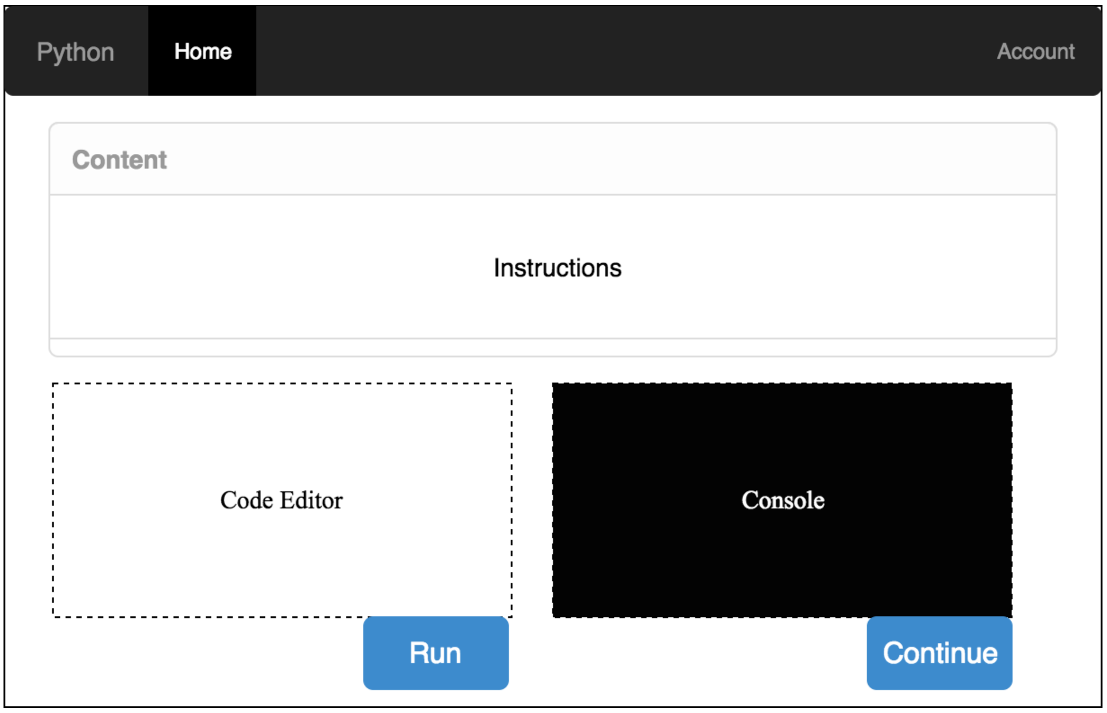


Figure 3.10. Learning module user interface

3.3.2. Labwork Module

The labwork module is used for labwork and exams. They are both timed events which means after some time, the submission is closed for all users. Also, the work submitted by students are graded and can be seen by the instructor in class dashboard. The exam module has three types of questions:

i. Python exercise questions

ii. Multiple choice questions

iii. True/false questions.

The user interface for this module consists of code editor, console, instruction panel, and help panel as seen on Figure 3.11. Help panel offers hints about questions.

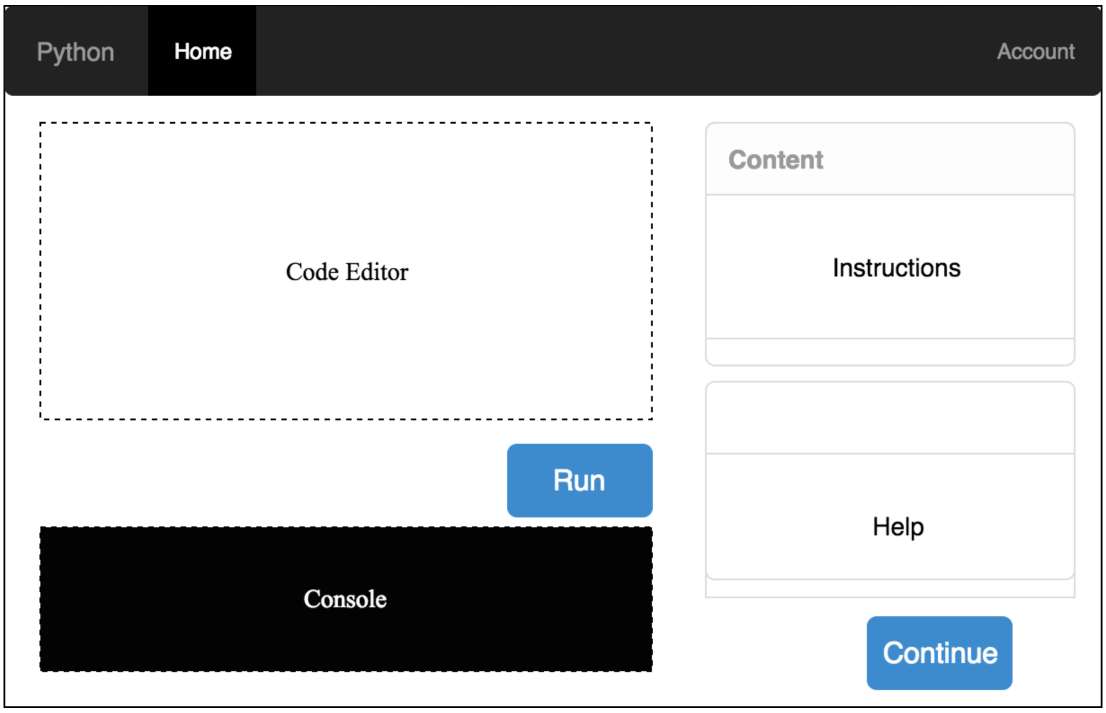


Figure 3.11. Labwork module user interface

Code editor is replaced with a multiple choice view when question type is multiple choice or true-false. Each labwork is created with grades for each question so that the grade can be calculated.

3.3.3. Assignment and Project Submission Module

User interface for assignment module is the same as the exam module. The difference between these two modules is that assignment module do not let student to skip questions. Because each assignment is prepared incrementally, students should find a correct solution to the current question in order to continue. For example, if an assignment to write a Python program that allows students to play blackjack card game were given, assignment module makes students write each function and continue incrementally. Student cannot play the game before creating a deck of cards and implementing the game logic.

3.3.4. Management Module

The management module is only visible to instructor. In this module two major interfaces are included: class dashboard and labwork creator. Class dashboard is the interface where instructor can see progress and grades of each student as shown in Figure 3.12. And, labwork creator allows instructor to create a labwork or an exam.

Instructor adds new content by using the content creator that takes instuction, help text, placeholder text, and unit tests as input. Content creator is used for each question separately.

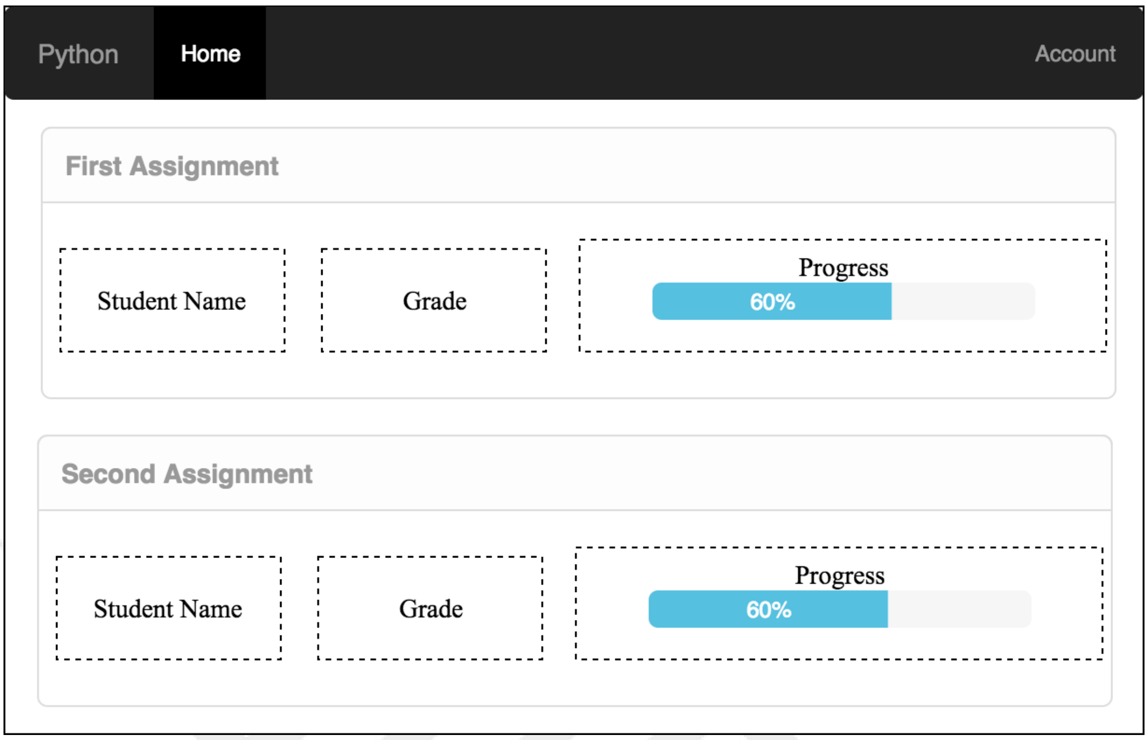


Figure 3.12. Class dashboard user interface mock up

4. IMPLEMENTATION

This chapter will cover the implementation of the platform. Contents are divided in three parts: web application, server side scripts, and database. First section will cover how the web application is implemented. Second section covers server side scripts and containers. Third section covers the database.

Figure 4.1 shows how components are related to each other and what technology is used for each component. Web application is implemented using an application framework Ruby on Rails [43]. Server script is implemented with the language Go [44]. The cloud database serving content and user data is NoSQL database DynamoDB [45]. The containers running in server are using docker [46].

Since testing groups will have less users than the production version, server scripts and web application are put inside the same server which is hosted on Amazon EC2 [47]. This is also done to reduce server costs.

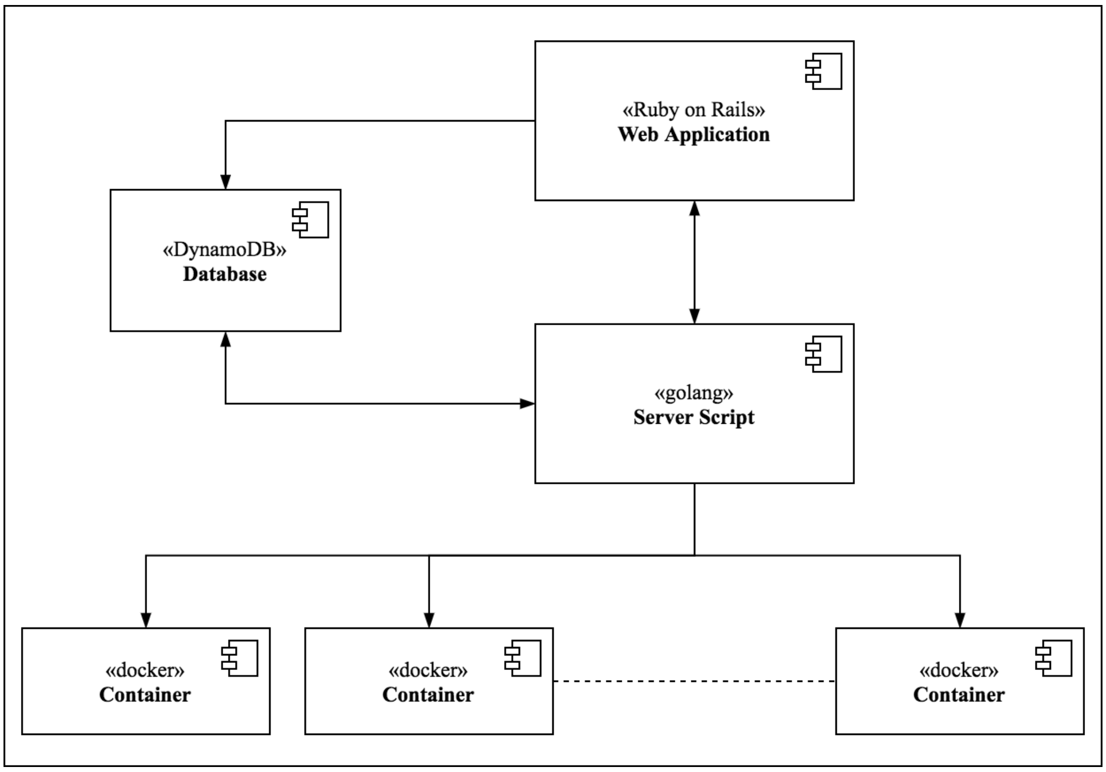


Figure 4.1. Component diagram of the platform

4.1. WEB APPLICATION

Web application is a client-server application where the client uses a web browser to access. Making a web application allows the service to be accessible from various devices. Because even if the device has a different architecture, almost all consumer devices have browsers.

The application is developed using Ruby on Rails which is a web application framework developed using Ruby programming language. Following chapters will describe how Rails, Bootstrap [48], and other components are used. The web application is designed to be as close as possible to the workspace of the students. Every code submitted to the server is saved.

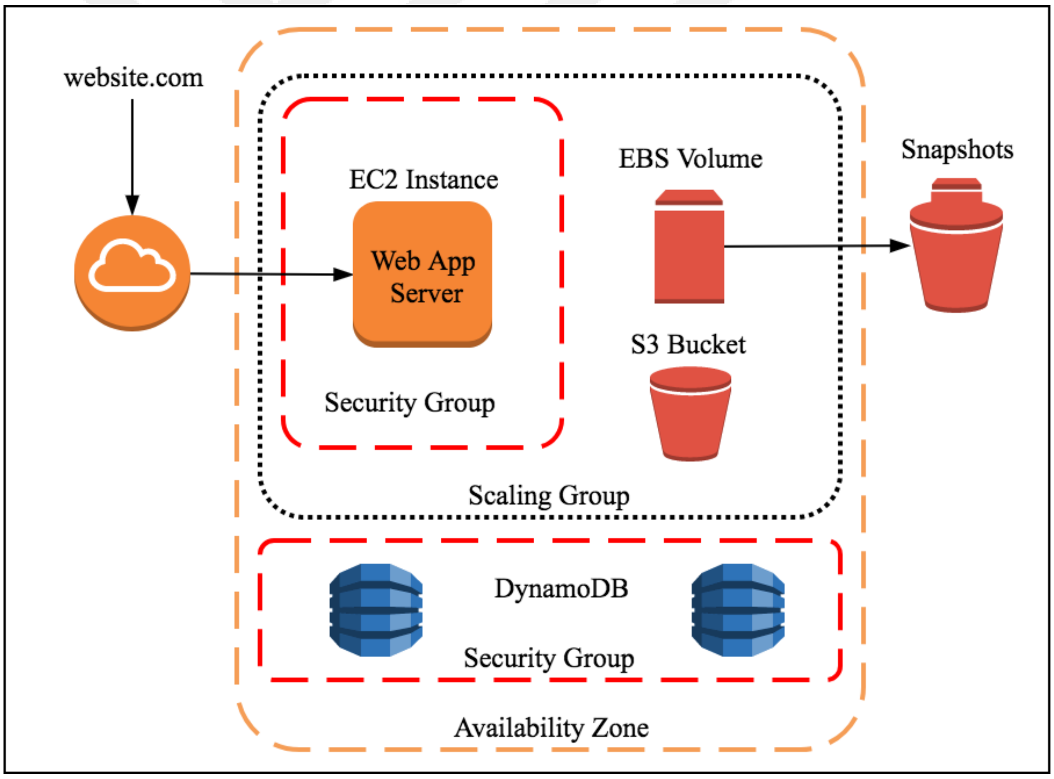


Figure 4.2. Web application on Amazon Web Services

Amazon Web Services (AWS) [49] is used for various parts of the application. AWS offers services for various fields such as web, cloud, and IoT. Figure 4.2 shows how the application is implemented using AWS components. Elastic Compute (EC2) modules are web servers with various customization options for CPU, memory, and storage. EBS (Elastic Block Store) is where the physical volume for instances are stored. S3 (Simple Storage Service) is where the static assets of the website is stored. EBS volumes are backed up using snapshots. DynamoDB tables store content and user data.

4.1.1. Ruby on Rails

Ruby on Rails is a web application framework developed using Ruby programming language which follows Model-View-Controller (MVC) pattern. In this pattern, model is the object that stores data. View is the visual representation of the model. Controller keeps the connection between models and views. Figure 4.3 shows how components of Rails are connected to each other.

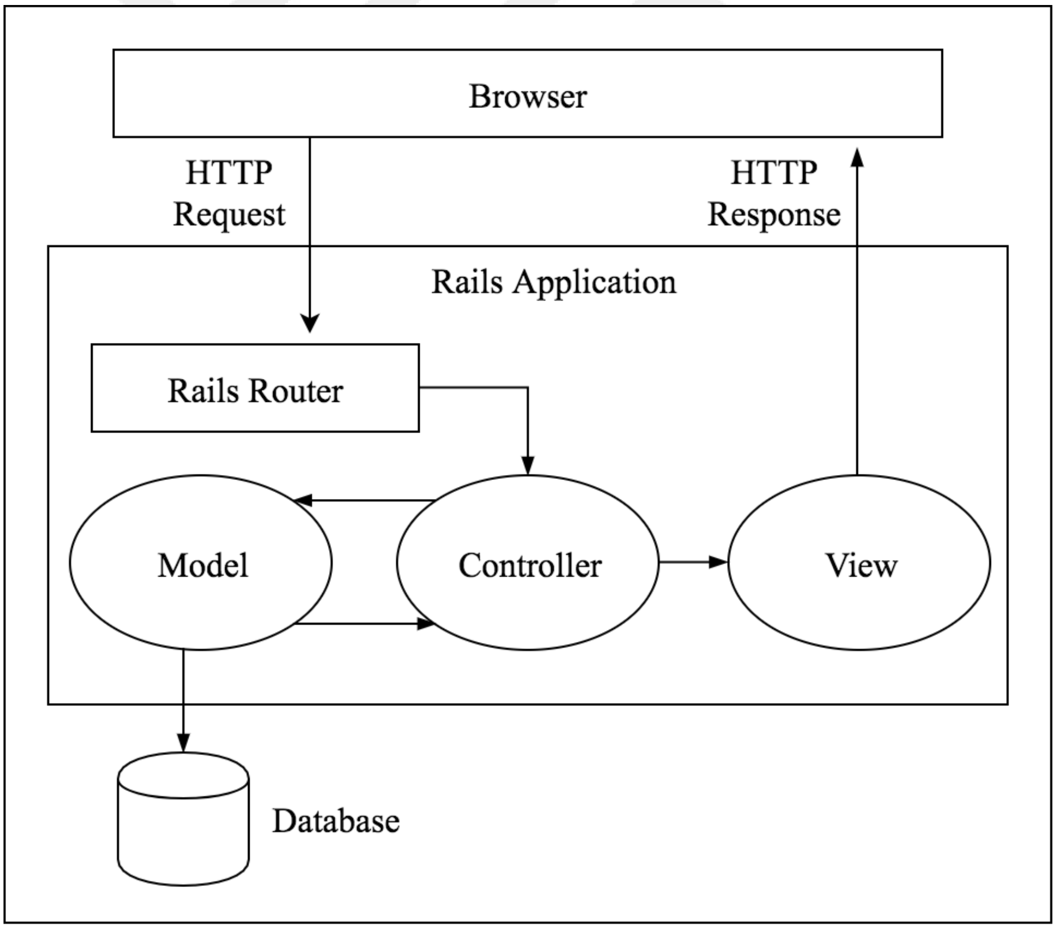


Figure 4.3. Rails application components

Rails applications consist of four parts which are router, controller, model, and view. Requests first received by the Rails router. Rails router recognizes requests and redirects them to the appropriate controller. Each route responds to a different request. Some of the route commands are get, post, patch, and resource. Command ‘get’ responds to the corresponding HTML request. Similarly, ‘post’ command sends the correct HTML response. The command ‘resource’ is used for models if all controller actions are needed. For example, if there is a separate page for each user, it is sufficient to type ‘resources :user’ and rails create the routes automatically. An example routes file generates the following text given below if ‘resources’ command is used with ‘users’ model.

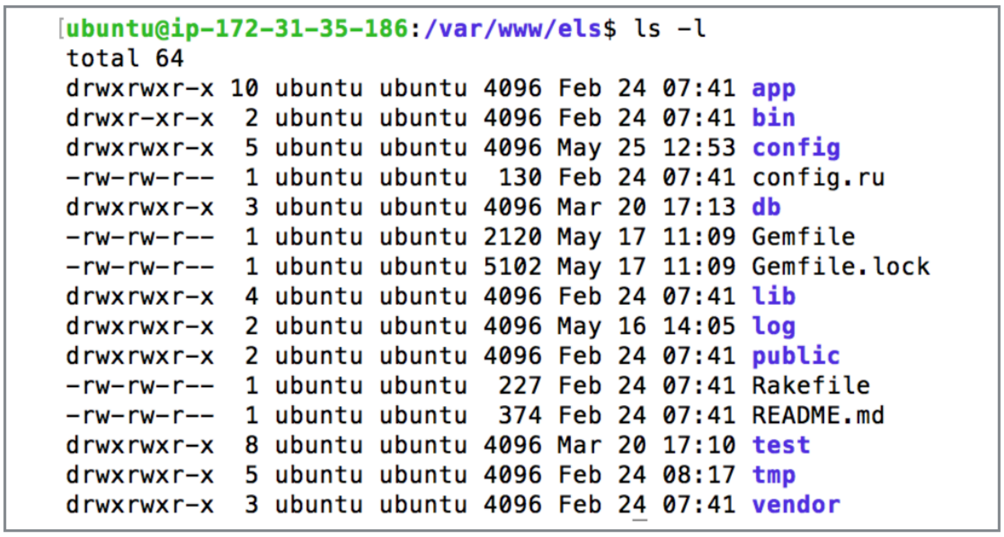
```
resources :users

get ‘/users’            =>      ( controller#index )

get ‘/users/:id’        =>      ( controller#show )

get ‘/users/new’        =>      ( controller#new )

post ‘/users’           =>      ( controller#create )

get ‘/users/:id/edit’   =>      ( controller#edit )

put ‘/users/:id’        =>      ( controller#update )

patch ‘/users/:id’      =>      ( controller#update )

delete ‘/users/:id’     =>      ( controller#destroy )
```

Controllers coordinate the interaction between model and view.Preparing correct values of models and querying them and then serving them to the view and creating the view serves time and resources. For example, if only one user profile is requested, there is no need to query all models containing users. Controllers have function for each corresponding route. Before the view is created, controller actions are called. Also, variables in views are created in controllers.

A model is a representation of data bind to a database. Each model creates a separate table in the database and model instances are the rows of this table. Despite using a simple database such as sqlite as a default, Rails support many database such as PostgreSQL, MySQL, Oracle, MongoDB, and SQL Server. In Rails, if a model attribute is changed and saved, it is automatically queried to the database. A migration is used when database needs an update. Migration files contain what changes needs to be done. The changes can be adding a new field to the table to adding a new foreign key. Then, 'rake' command is called to apply changes. Default model architecture is called ActiveRecord. For the purposes of this thesis work, a custom model class will be used.

Views are similar to a template that shows what will be created as an HTML file at runtime. Not all elements are rendered to each response of the same view. For example, some elements could only be visible to logged in users. View files end with extension 'erb' which means Embedded Ruby.

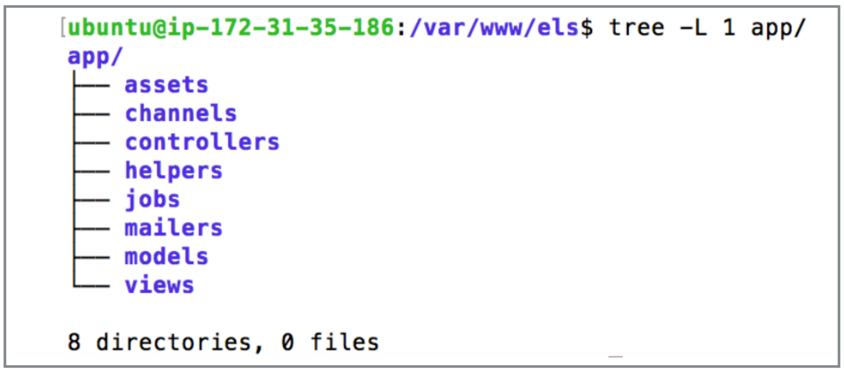
The directory structure of a rails application is given in Figure 4.4. The folder 'app' contains MVC files, 'config' folder contains routes.rb file, and 'db' folder contains migration files.



```
ubuntu@ip-172-31-35-186:/var/www/els$ ls -l
total 64
drwxrwxr-x 10 ubuntu ubuntu 4096 Feb 24 07:41 app
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb 24 07:41 bin
drwxrwxr-x  5 ubuntu ubuntu 4096 May 25 12:53 config
-rw-rw-r--  1 ubuntu ubuntu  130 Feb 24 07:41 config.ru
drwxrwxr-x  3 ubuntu ubuntu 4096 Mar 20 17:13 db
-rw-rw-r--  1 ubuntu ubuntu 2120 May 17 11:09 Gemfile
-rw-rw-r--  1 ubuntu ubuntu 5102 May 17 11:09 Gemfile.lock
drwxrwxr-x  4 ubuntu ubuntu 4096 Feb 24 07:41 lib
drwxrwxr-x  2 ubuntu ubuntu 4096 May 16 14:05 log
drwxrwxr-x  2 ubuntu ubuntu 4096 Feb 24 07:41 public
-rw-rw-r--  1 ubuntu ubuntu  227 Feb 24 07:41 Rakefile
-rw-rw-r--  1 ubuntu ubuntu  374 Feb 24 07:41 README.md
drwxrwxr-x  8 ubuntu ubuntu 4096 Mar 20 17:10 test
drwxrwxr-x  5 ubuntu ubuntu 4096 Feb 24 08:17 tmp
drwxrwxr-x  3 ubuntu ubuntu 4096 Feb 24 07:41 vendor
```

Figure 4.4. Rails application directory

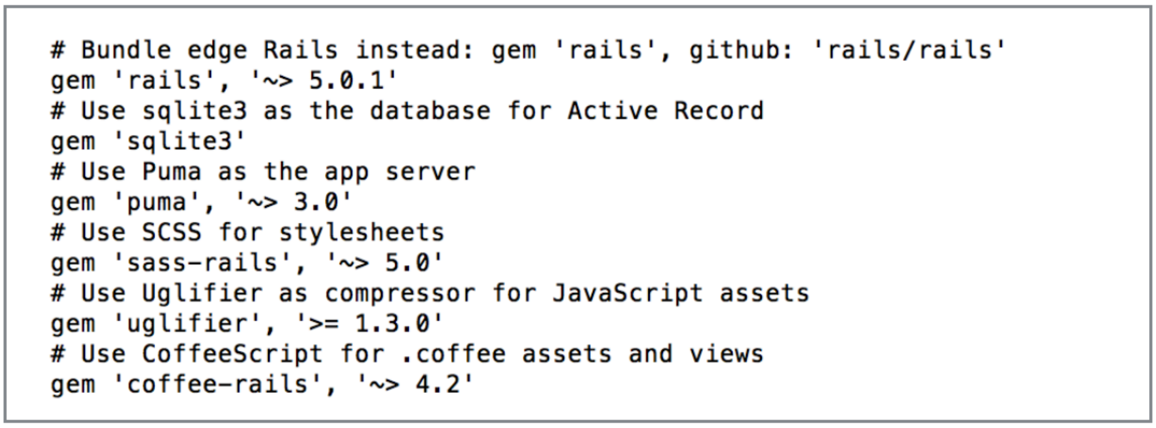
The directory 'app' contains folders such as model, view, controller, helpers, job, and assets as seen on Figure 4.5.



```
ubuntu@ip-172-31-35-186:/var/www/els$ tree -L 1 app/
app/
├── assets
├── channels
├── controllers
├── helpers
├── jobs
├── mailers
├── models
└── views

8 directories, 0 files
```

Figure 4.5. App directory structure

Rails applications use a package manager called RubyGems. In Ruby, a package is called a gem. Each rails application has a file called 'Gemfile' that contains the name and version of gems that will be used in the application. When a new gem is added to the file, 'bundle install' command is called and the new gem is installed. Figure 4.6 shows a portion of an example Gemfile which uses rails gem with version 5.0.1 and coffee-rails gem with version 4.2.



```
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '~> 5.0.1'
# Use sqlite3 as the database for Active Record
gem 'sqlite3'
# Use Puma as the app server
gem 'puma', '~> 3.0'
# Use SCSS for stylesheets
gem 'sass-rails', '~> 5.0'
# Use Uglifier as compressor for JavaScript assets
gem 'uglifier', '>= 1.3.0'
# Use CoffeeScript for .coffee assets and views
gem 'coffee-rails', '~> 4.2'
```

Figure 4.6. Part of a Gemfile

4.1.1.1.Ruby

Ruby [50] is an interpreted, dynamic and object oriented language that was designed with simplicity and productivity in mind. It has a easy to write syntax similar to Python. Ruby on Rails uses Ruby language. The purpose of the framework is to create powerful web applications using the simple and natural language Ruby. It also makes it simple to write repetitive tasks.

4.1.1.2.Model-View-Controller Pattern

The MVC is a pattern used in web frameworks that simplifies the process for developers. It is important to understand sequence of events before developing an application using this pattern. The process starts with a request from the browser and ends with a view rendered as a response. Figure 4.7 shows the sequence of events.

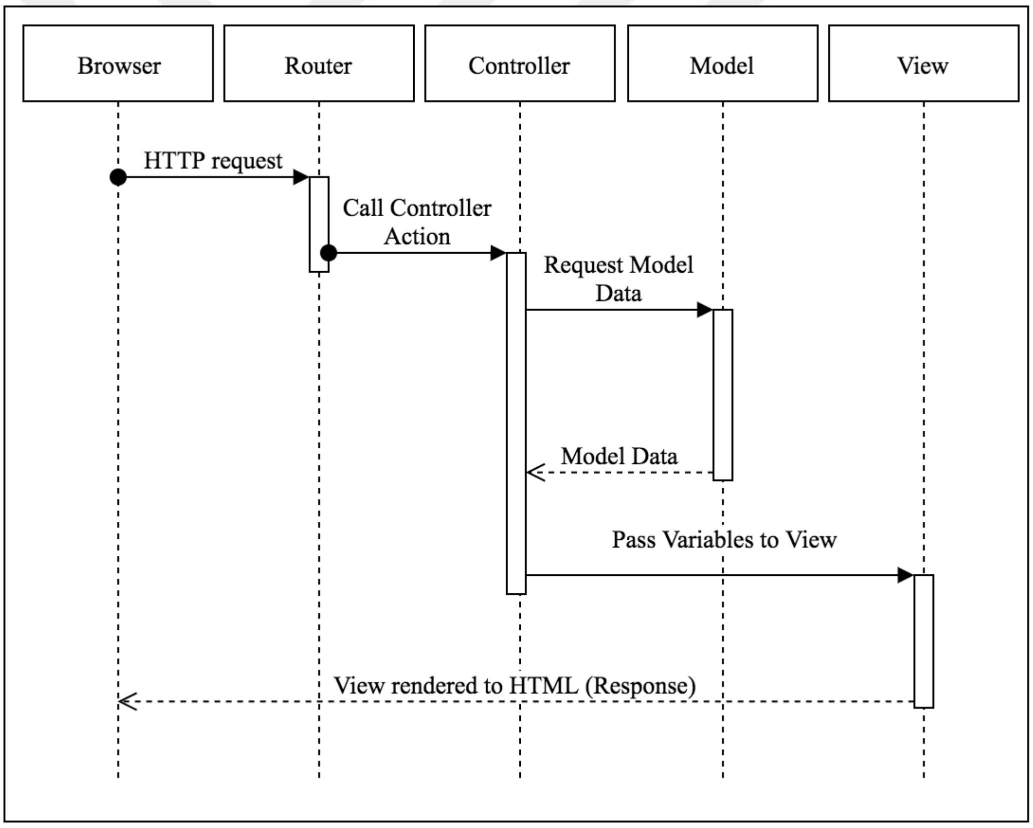


Figure 4.7. Sequence diagram of MVC

4.1.2. Session Management and Authentication

A session is an information that is available through the user's interaction with the website. After user enters the login information, a session is created if the information is correct. Sequence diagram of the authentication is given in Figure 4.8.

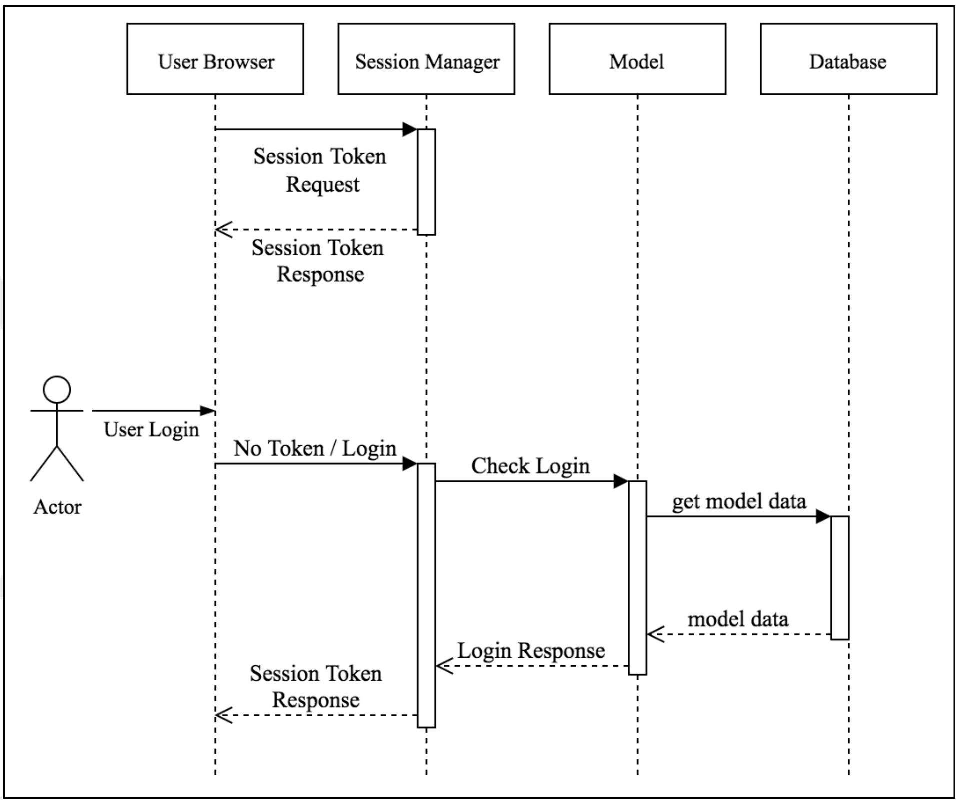


Figure 4.8. Sequence diagram of authentication

4.1.3. Controllers

Learning module, content, labwork, assignment, user, session, and static page controllers created for this application. Each controller creates the view of the module of the same name. Figure 4.9 shows the controller diagram of the application.

Learning module controller prepares the learning module by fetching user and learning content models. Content controller handles the creation of new content. Labwork controller fetches the user and content models and creates labwork view. Assignment controller is the same as the labwork controller except the progress iteration. User module handles create, edit, destroy, and update actions of user model. Session controller handles the session creation and authentication. Static page controller serves the information pages of the application such as contact and help.

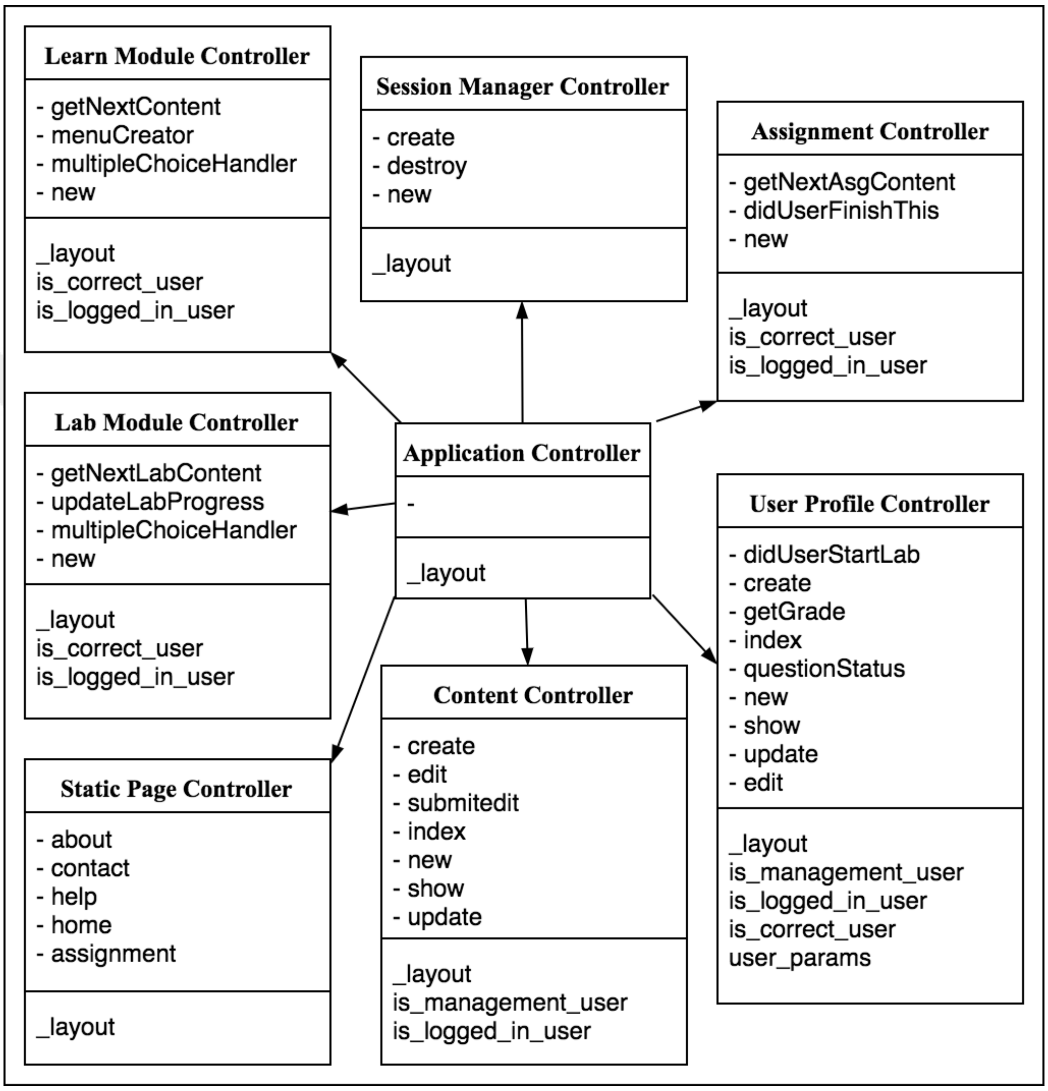


Figure 4.9. Controller diagram of the application

4.1.4. Models

There are four modules created for the application: user, exam, content, and learning content as seen on Figure 4.10.

User model contains information about the user such as name, email, password token, progress, and badges. Exam model contains information about exam and labwork types. It stores the exam id, grades of questions, and questions. Content model contains name, instructions, and test names that will be applied to the user code. Learning module does not have the unit test name because answers are not graded.

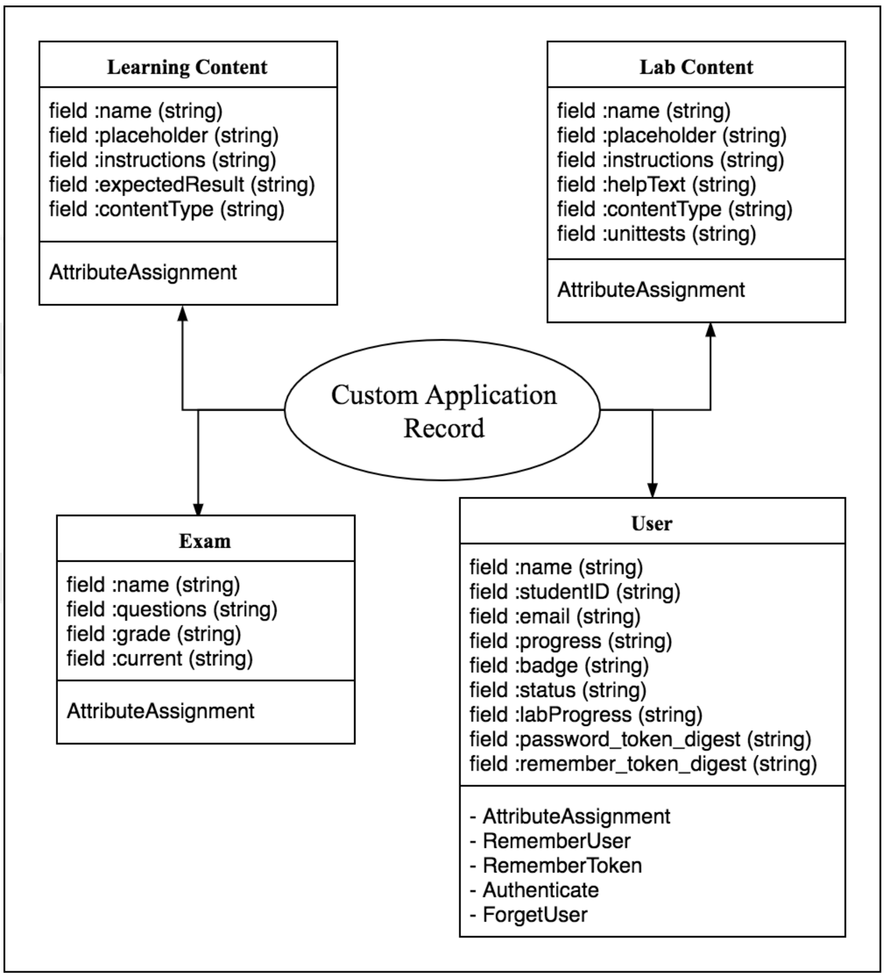


Figure 4.10. Models of the application

4.1.5. User Profiles

User profile shows the progress and achievements of the current user. Badges and micro-posts are also shown in the profile page of each user.

4.1.5.1.Badges

Badges are earned after completing pre-defined achievements. A badge consists of an image, a name, and a description as seen on Figure 4.11.

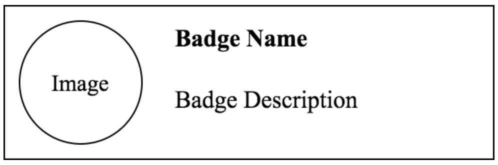


Figure 4.11. Badge in user profile

Badges are earned when milestones in assignments are completed. For example, students were offered an assignment to implement a game which was divided in three parts. For this assignment, there are three badges created for completion of each part. For the learning module, badges can be earned at random events such as first print statement containing 'Hello, World!'.

4.1.5.2.Micro-posts

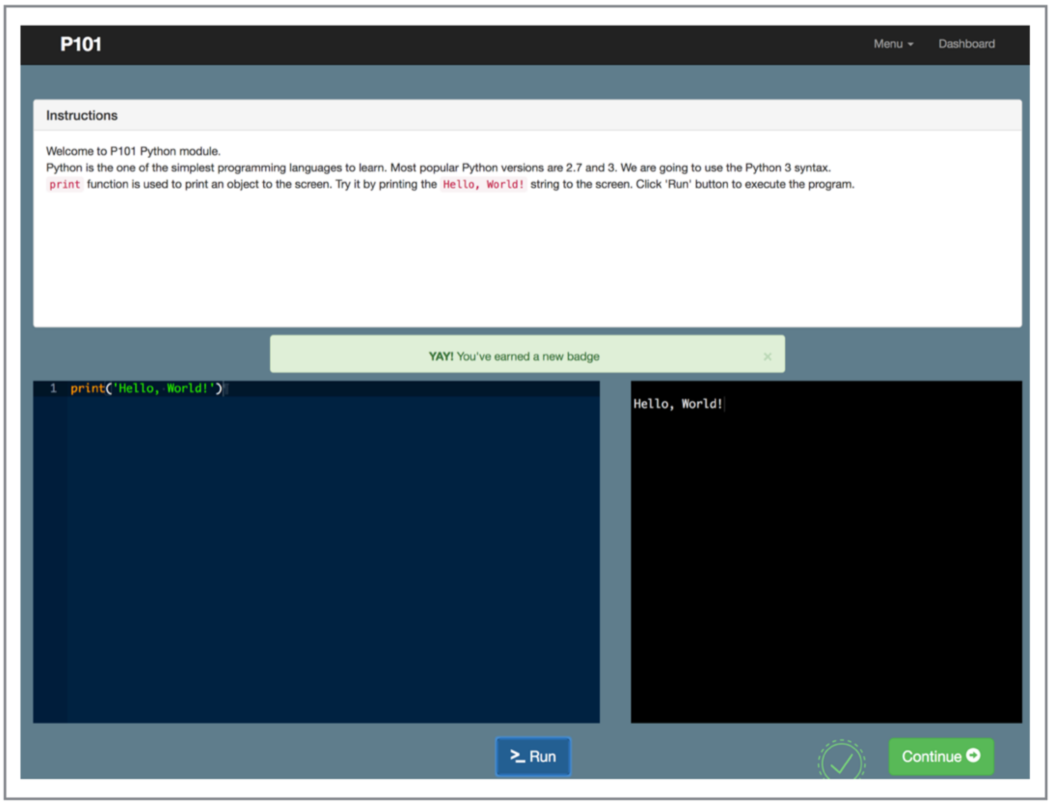
User actions are documented in profiles using micro-posts. Micro-posts are short updates that contains information about the actions taken by the user. For example, completion of a quiz is an update. Updates are similar to Tweets which are also short and placed in profile in similar design. Users only see their own post updates.

4.1.6. User Interface

This section describes screenshots of the user interface used in the system and how they are implemented. The interface elements include a code editor, a console showing output, instruction panels, and help panel. Interface is developed using a framework called Bootstrap. Bootstrap is a front-end framework that offers commonly used elements and grid layout system. It also has templates for typography, forms, buttons, interface components such as forms, and headers. Bootstrap is an open source project.

4.1.6.1.Interface for Learning Module

Learning module has three different interfaces that are content, quiz, and exercise. Content module is shown in Figure 4.12. It includes a instruction that explains the contents and an example command that can be run and user is expected to try it out.



Instructions

Welcome to P101 Python module.
Python is the one of the simplest programming languages to learn. Most popular Python versions are 2.7 and 3. We are going to use the Python 3 syntax.
`print` function is used to print an object to the screen. Try it by printing the `Hello, World!` string to the screen. Click 'Run' button to execute the program.

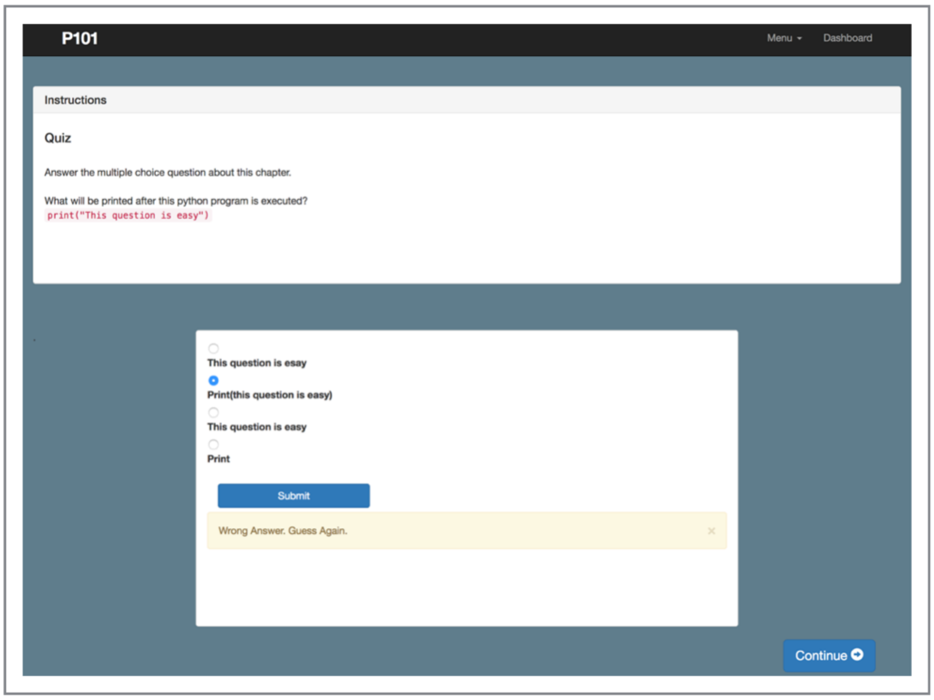
YAY! You've earned a new badge

```python
print('Hello, World!')
```

```
Hello, World!
```

Figure 4.12. Learning module content interface

Quiz module offers a multiple choice question about the current section. Figure 4.13 shows an example quiz from the learning interface.



Instructions

Quiz

Answer the multiple choice question about this chapter.

What will be printed after this python program is executed?

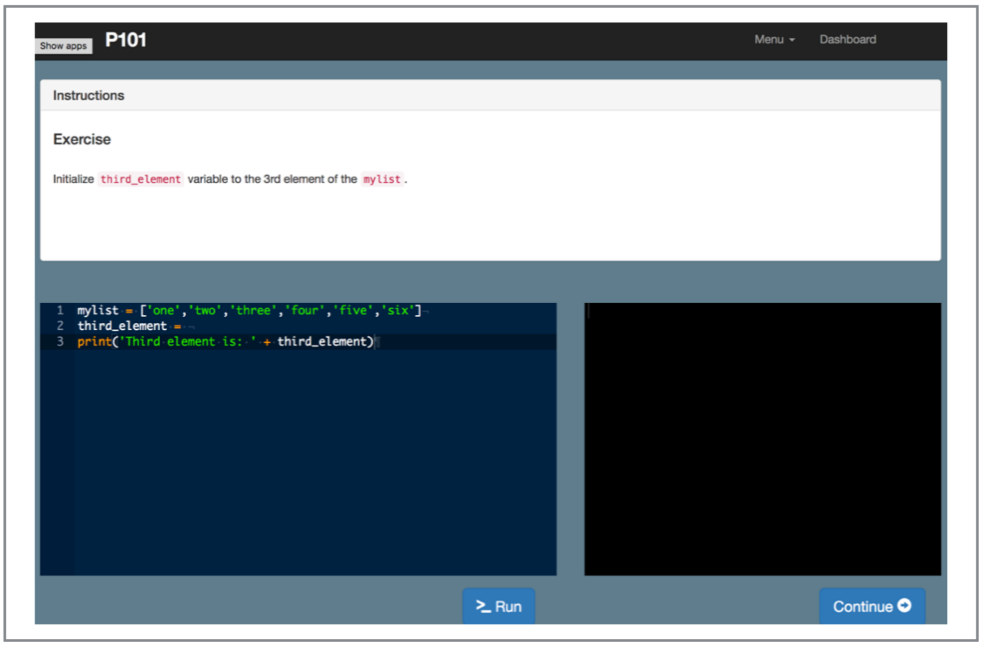
```
print("This question is easy")
```

- This question is esay
- [✓ Print(this question is easy)]
- This question is easy
- Print

Wrong Answer. Guess Again.

Figure 4.13. Learning module quiz interface

Exercise module has a question and a placeholder code that will be fixed by the student to output the expected result as seen on Figure 4.14.



Instructions

Exercise

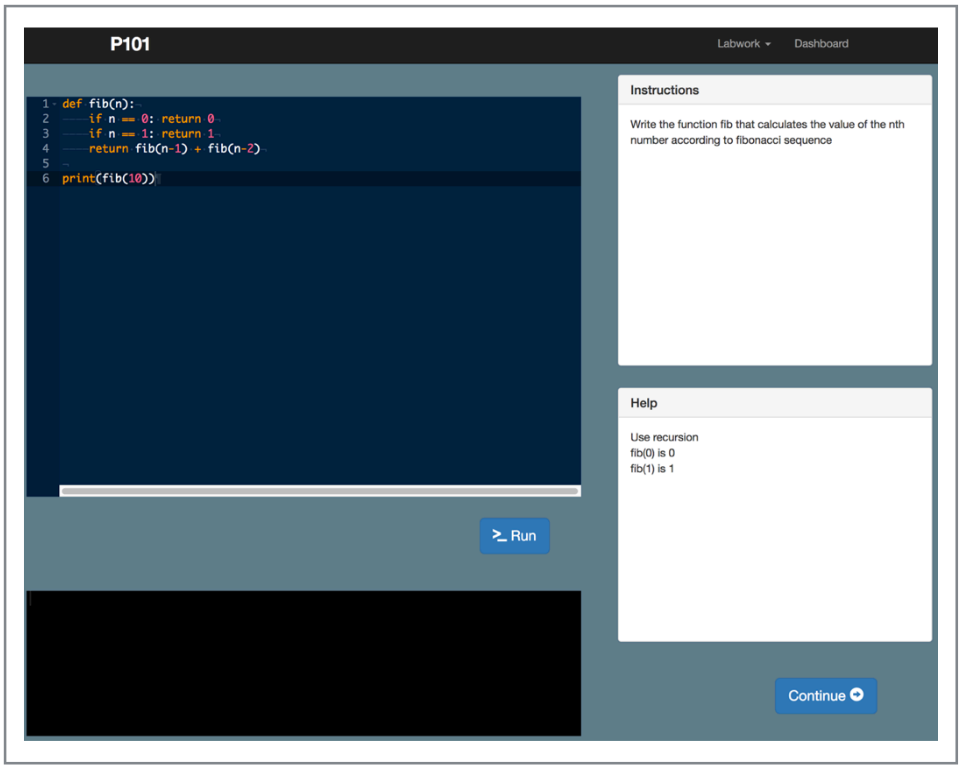
Initialize `third_element` variable to the 3rd element of the `mylist`.

```
mylist = ['one','two','three','four','five','six']
third_element = 
print('Third element is: ' + third_element)
```

Figure 4.14. Learning module question interface

4.1.6.2.Interface for Labwork Module

Labwork module is where the questions offered to the student for them to answer in a limited amount of time. It has a help and instruction panel in addition to the code editor and console. Figure 4.15 shows en example labwork module.



```python
def fib(n):
    if n == 0: return 0
    if n == 1: return 1
    return fib(n-1) + fib(n-2)

print(fib(10))
```

Instructions

Write the function fib that calculates the value of the nth number according to fibonacci sequence

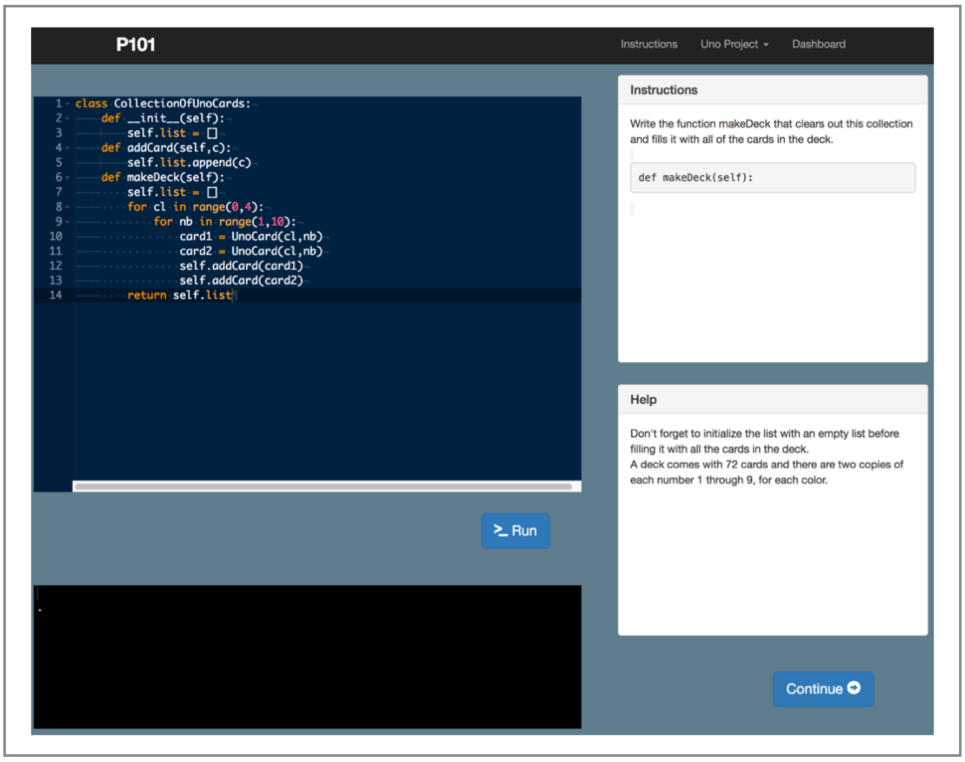
Help

Use recursion
fib(0) is 0
fib(1) is 1

Figure 4.15. Labwork module interface

4.1.6.3.Interface for Exam Module

Exam module is also used for assignments. The difference between labwork and assignment is student cannot continue to next question before completing the current question. The questions are prepared to be incremental and continues from the last completed one. Figure 4.16 shows an example assignment module.



```
class CollectionOfUnoCards:
    def __init__(self):
        self.list = []
    def addCard(self,c):
        self.list.append(c)
    def makeDeck(self):
        self.list = []
        for cl in range(0,4):
            for nb in range(1,10):
                card1 = UnoCard(cl,nb)
                card2 = UnoCard(cl,nb)
                self.addCard(card1)
                self.addCard(card2)
        return self.list
```

Instructions

Write the function makeDeck that clears out this collection and fills it with all of the cards in the deck.

```
def makeDeck(self):
```

Help

Don't forget to initialize the list with an empty list before filling it with all the cards in the deck.
A deck comes with 72 cards and there are two copies of each number 1 through 9, for each color.

Figure 4.16. Exam module interface

Since the students cannot continue to the next question until the current one is answered correctly, header menu shows what questions are answered and what is not as seen on Figure 4.17.

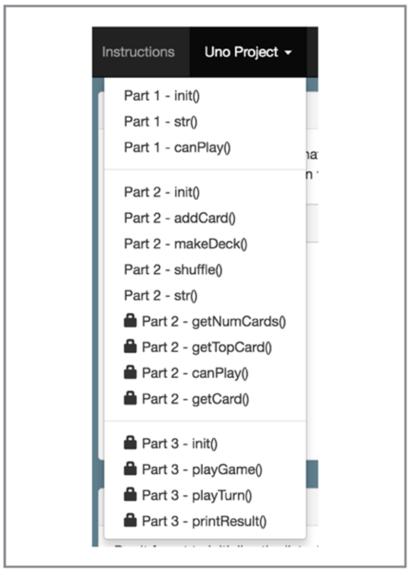
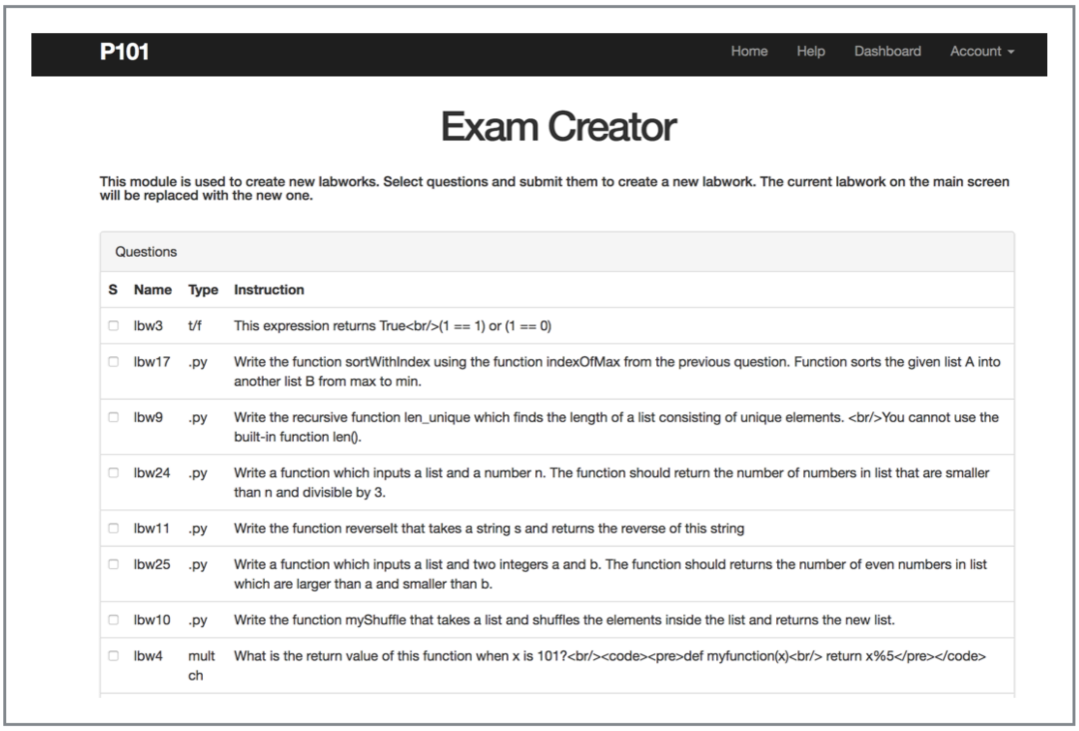


Figure 4.17. Exam module menu

4.1.6.4.Interface for Labwork Creator

Exam or labwork creation interface is only usable for instructor. Instructor select the questions that will be included in the next labwork and when a student starts a new labwork, new questions are served. Figure 4.18 shows the exam creator interface.



Exam Creator

This module is used to create new labworks. Select questions and submit them to create a new labwork. The current labwork on the main screen will be replaced with the new one.

Questions

S	Name	Type	Instruction
☐	lbw3	t/f	This expression returns True<br/>(1 == 1) or (1 == 0)
☐	lbw17	.py	Write the function sortWithIndex using the function indexOfMax from the previous question. Function sorts the given list A into another list B from max to min.
☐	lbw9	.py	Write the recursive function len_unique which finds the length of a list consisting of unique elements. <br/>You cannot use the built-in function len().
☐	lbw24	.py	Write a function which inputs a list and a number n. The function should return the number of numbers in list that are smaller than n and divisible by 3.
☐	lbw11	.py	Write the function reverseIt that takes a string s and returns the reverse of this string
☐	lbw25	.py	Write a function which inputs a list and two integers a and b. The function should returns the number of even numbers in list which are larger than a and smaller than b.
☐	lbw10	.py	Write the function myShuffle that takes a list and shuffles the elements inside the list and returns the new list.
☐	lbw4	mult ch	What is the return value of this function when x is 101?<br/><code><pre>def myfunction(x)<br/> return x%5</pre></code>

Figure 4.18. Exam creator module

4.1.6.5.Interface for User Profiles

The user profile is different for students and instructors. Students have a profile similar to Figure 4.19. It shows badges, progress of current work, and links to materials.

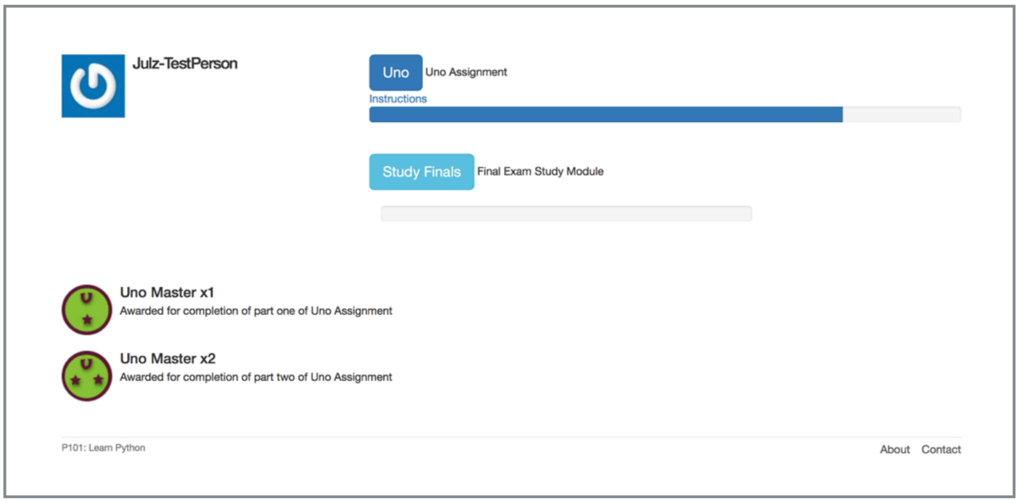


Figure 4.19. User profile

The instructor accounts are different from the student profiles. It includes a link to class dashboard which shows student grades and progress. Also, it includes a link to exam creator as seen in Figure 4.20.

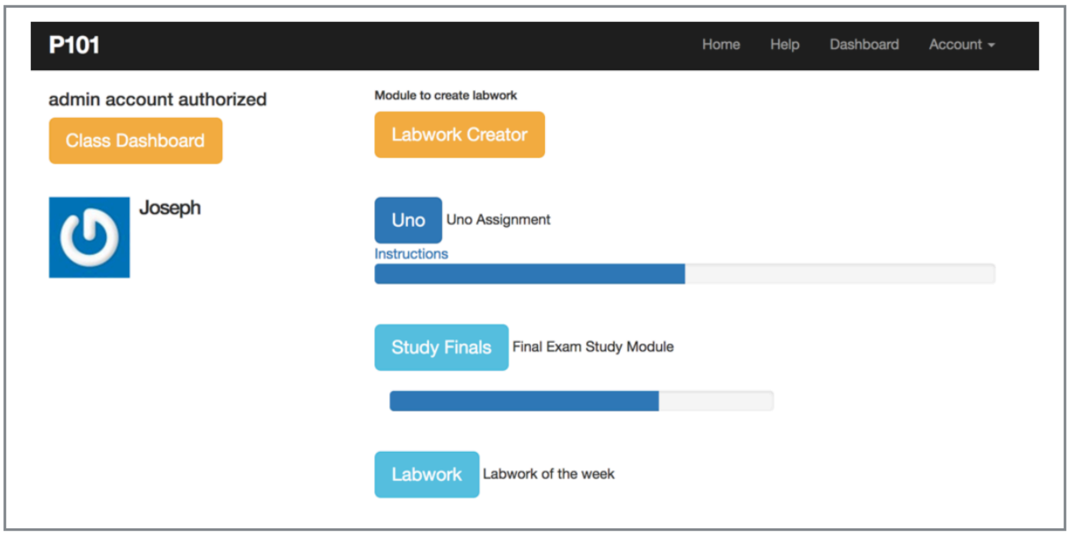


Figure 4.20. Instructor profile

4.1.7. Websockets

Websocket [51] is a full-duplex communication protocol that uses a single TCP connection. It works similar to TCP sockets. A socket is opened on the server for browsers to connect using sockets. It is the main technology used in push-notifications that allows asynchronous updates from the server. Websockets are used in this project to allow communication between Rails application and server scripts.

The code written by user is send to server using a websocket. Server runs the appropriate scripts and then the result is send back to the browser from the same socket. User gets a response according to correctness of the submitted code.

For easier reading and parsing JSON format is used in messages between browser and server. Code submitted by the user is retrieved as a string and combined with user and session information to create the message formatted as JSON. Figure 4.21 shows the sequence of events.

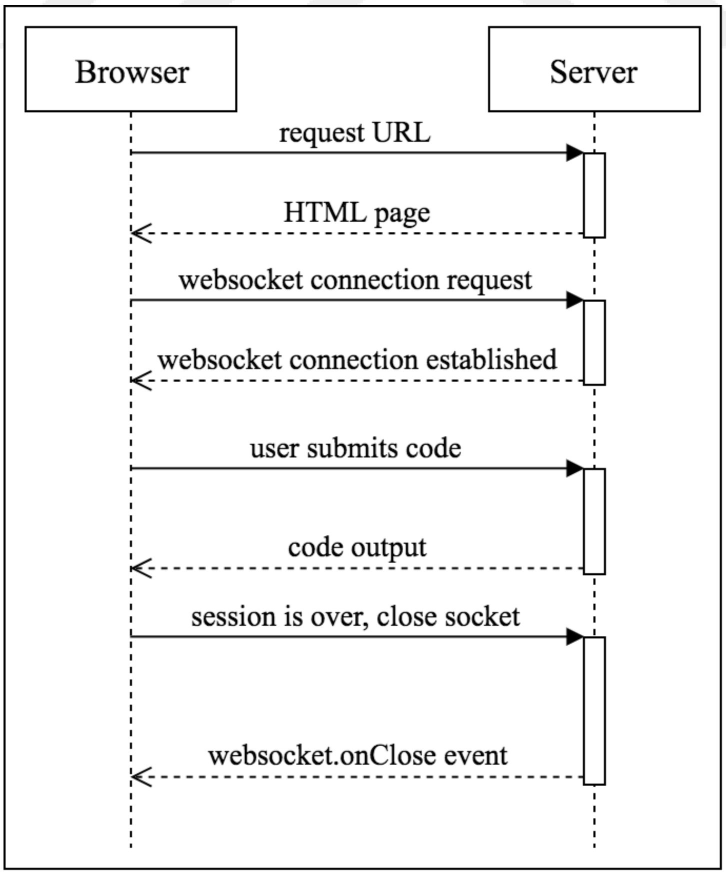

Browser
Server
request URL
HTML page
websocket connection request
websocket connection established
user submits code
code output
session is over, close socket
websocket.onClose event

Figure 4.21. Websocket sequence diagram

4.1.8. jQuery

jQuery [52] is a JavaScript library that makes event handling and animations easier. jQuery functions are called in case of an event. In the case of this work, functions are called on websocket events. Some of the events and functions are given below.

websocket.onOpen	=>	fillCodeEditor()
websocket.onClose	=>	serverConnectionChecker()
websocket.onMessage	=>	updateConsole()

4.2. SERVER

There is one main script running on the server which was implemented in Go programming language. The server scripts are also connected with the database to update and retrieve the user progress. Figure 4.22 shows the components of the server.

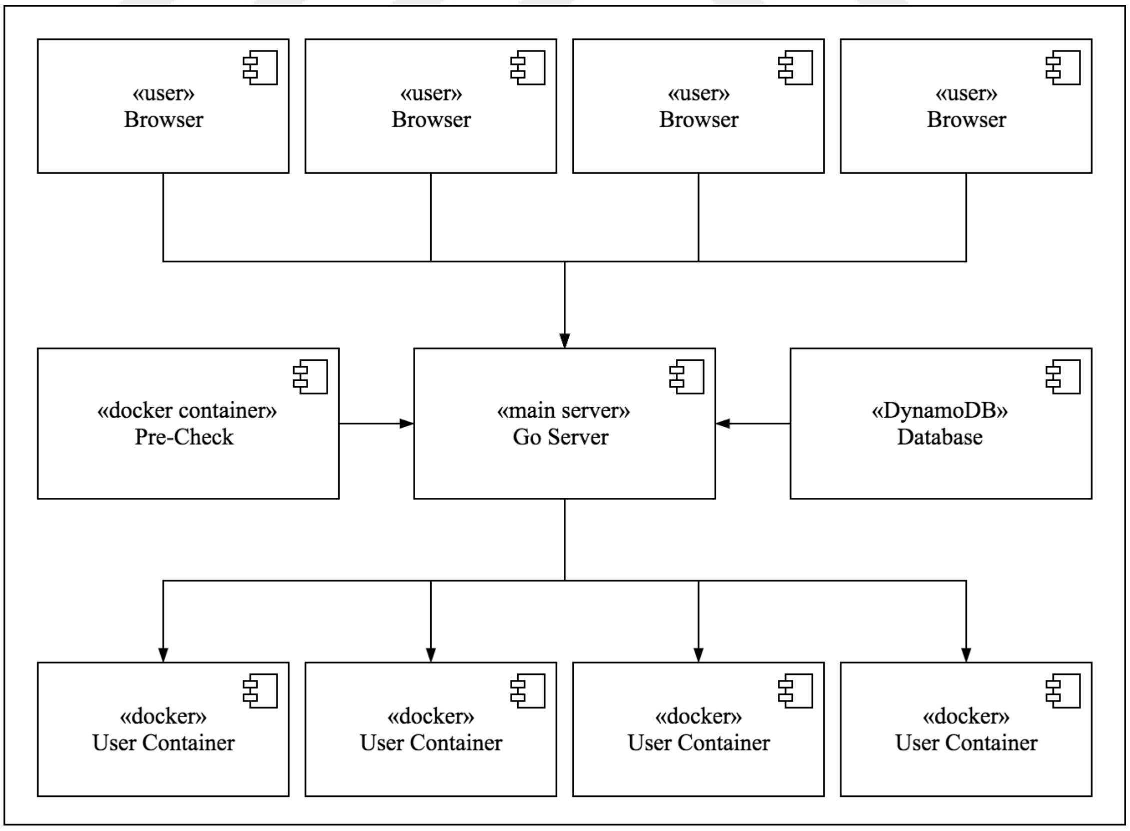


Figure 4.22. Server component diagram

Main server is the server that is always listening to connections from browsers. If the received code is valid according to the pre-check result, it is send to a user container. User container has two modules to execute. First, the unit tests are applied to the Python program. If the program passes tests, it is executed in user container. Then, database is updated according to the result from the container. Activity diagram can be seen on Figure 4.23.

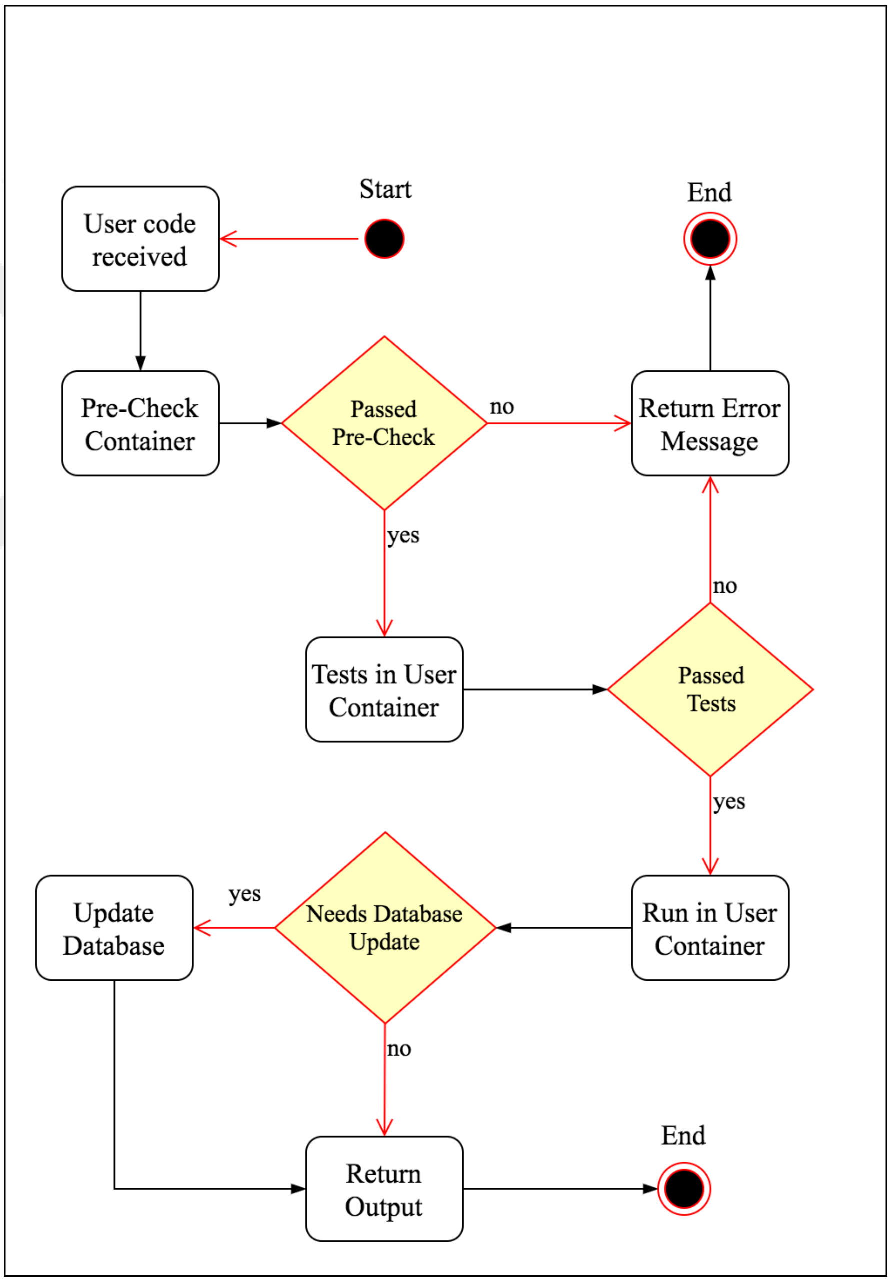


Figure 4.23. Activity diagram of the server

4.2.1. Web Applications

Web applications are designed to listen the same port continuously. The scripts written for this work are starting up at the same time as the hosting machine. The hosting machine uses Ubuntu Linux [53] with version 16.04. Scripts are gathered together as a linux service controlled by the ‘systemctl’ command. This allows administrator to only use reboot on the management console without logging in to the machine. Despite simple design, it allows administrator to write startup and reboot scripts on a separate master machine to control all machines in the architecture without any problem. For the purposes of cost cutting and simplicity, web framework Rails and server scripts are running on the same machine.

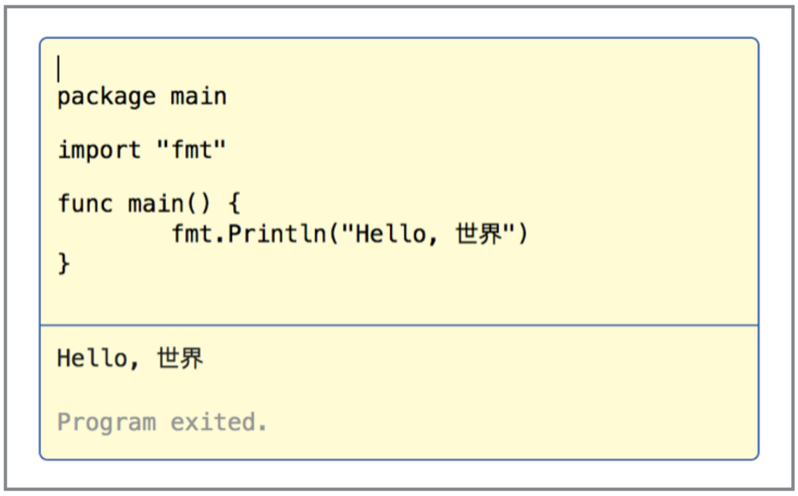
4.2.2. Go Language

The server program is implemented using Go programming language. Go is an open source and free programming language. It was created at Google in 2007. Despite being a new language, Go has become popular among developers. Most powerful applications written in Go are network and web servers. Go has a native support for concurrency using goroutines and channels. Goroutines are functions that can run concurrently with other functions.

Go language offers solutions to new problems that did not exist years ago. For example, in cloud computing, usage of virtual machines has become a standard. New machines are constantly added to the platform to meet the increasing demands. Since Go is an interpreted language, applications launch faster than traditional scripts.

In addition to developer popularity, many cloud based systems such as Google Cloud Platform [54], Heroku [55], and Amazon Web Services are supporting Go natively.

Go has a simple syntax similar to Python and Ruby. Figure 4.24 shows a simple ‘Hello, World!’ program in Go. Package name is stated with the ‘package’ command and dependencies are imported using ‘import’ command. One of the reasons why Go is used in the server is because the compiled binary can be used in another server easily. The binary contains the dependencies and other packages.



```
|
package main

import "fmt"

func main() {
	fmt.Println("Hello, 世界")
}
```

```
Hello, 世界

Program exited.
```

Figure 4.24. A simple Go program

4.2.3. Docker Containers

Docker is an open source project that automates software deployment by using containers. It is a platform for running, developing, and shipping applications. Containers are similar to old chroot based systems but they are much more enhanced and customizable. Containers have become standard for many developers shipping applications [56]. It eases process by minimizing the dependencies and libraries. A container is a closed box that contains libraries needed to run the application. A use case story is where multiple versions of the same library is tested with the new application and they can run in separate containers in the same system.

There is a difference between running an app in a virtual machine and a Docker container. A virtual machine is an emulated hardware running a guest operating system on a host operating system. Virtual machines use allocated resources of the host machine. The main difference is that even though the containers are isolated, they share the same operating system whereas virtual machines need their own guest operating systems. If appropriate, containers also share binary and libraries which is not possible in virtual machines. The size comparison between two systems are shown in Figure 4.25. For example, a guest operating system Ubuntu is as small as 1 GB, whereas a Ubuntu Linux image of docker container is only 188 MB [57]. There are even small docker operating systems for cloud computing less than 10 MB.

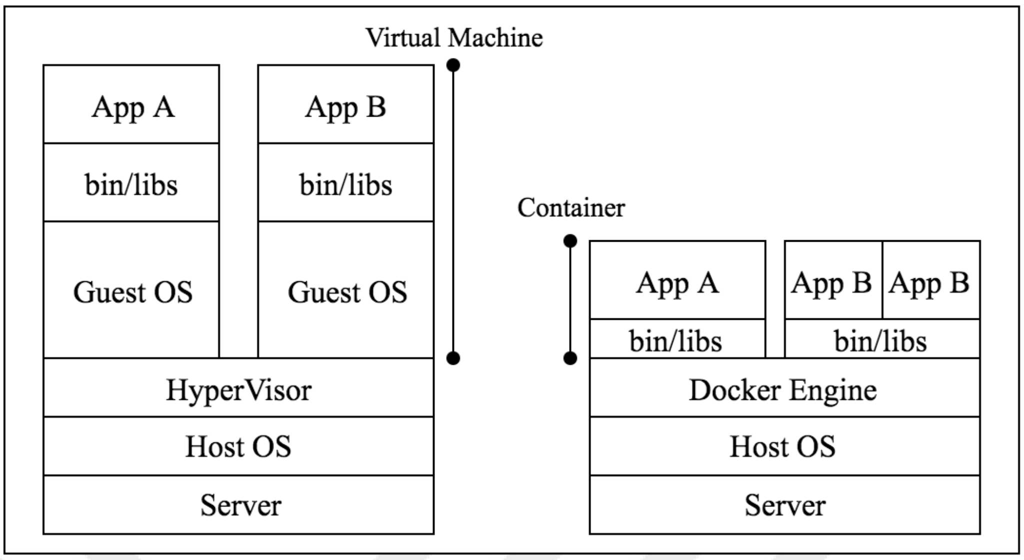


Figure 4.25. Virtual machine and containers

Another difference between virtual machines and containers is the startup time. It takes minutes to start a virtual machine whereas it takes seconds to start a container. Since the code from users will be executed in containers, startup time is important for performance.

Containers use images similar to virtual machines. An image is a file system that includes meta data on running the container. Image configuration file is called a ‘docker file’. The user containers used in this work contains Python version three for execution of scripts inside a container. The reason why user programs are executed in containers is they offer complete isolation from the host operating system. Each container is created whenever user is active. Aside from that, containers are lightweight and used for a set of defined tasks. This allows scalability for the e-learning platform. It is possible to use separate machines to run containers without mixing up the user code because each machine is isolated from each other. In order to scale the system to serve any amount of users, it is enough to add more machines to run containers as seen on Figure 4.26. For cloud and new generation web applications, being scalable is more important than performance. Because, in the long run, system performance will decrease by being not scalable. In addition to mentioned points, containers are easy to deploy. Many web services offer simple deployment options for docker containers.

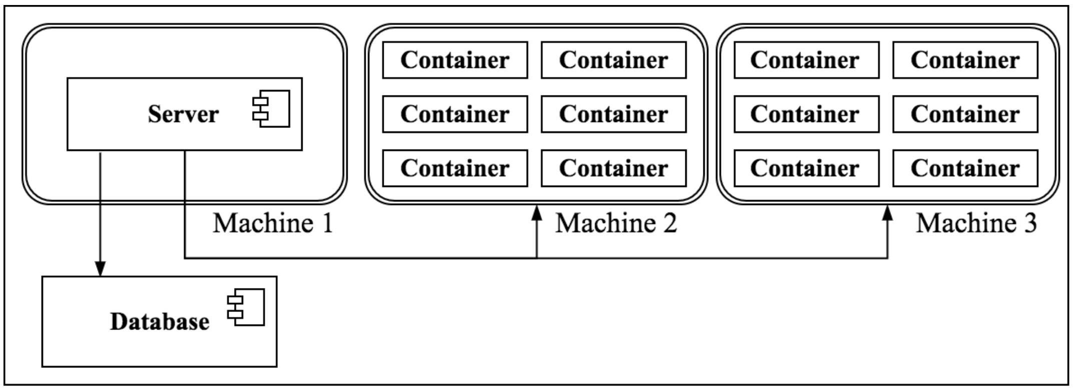


Figure 4.26. Adding containers to the system

Containers used in this work only consists of Python and libraries to run it. The program copies the user code into a container and runs it inside the container as seen on Figure 4.27. The main advantage of this system is isolation. For example, even though this is a learning platform, malicious code could be send to the server for execution and isolated containers are deleted with such code. The malicious code never runs on the host machine. In an another example, Python offers a library called ‘os’ that allows users to use system commands with functions similar to ‘exec’. Since the code is executed inside a container, user code cannot use the libraries like ‘os’. Finally, containers give more details on execution time, resources allocated for the program which provides analytics for the developers.

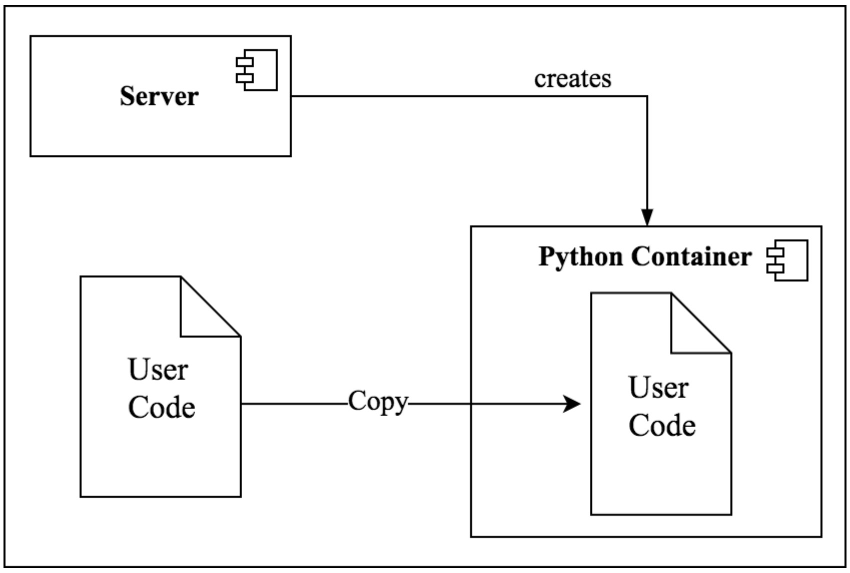


Figure 4.27. Python container

Containers are not kept alive to save machine resources since the startup time is under seconds. If the containers did not used in the system, one of the options was to create a service to execute Python programs separately as seen on Figure 4.28. If this method was used, there could be delays between message queue for starting the execution, and retrieving the results. Instead, docker containers give immediate control over execution.

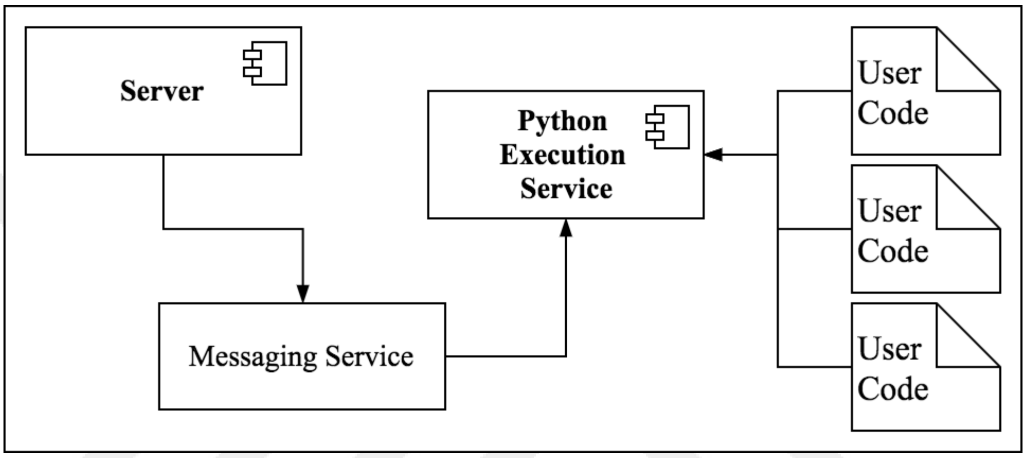


Figure 4.28. Python execution with messaging

4.2.4. Unit Tests and Evaluation

Unit test is a type of software test where each unit of the software is tested separately to check whether they work as expected or not. A common use of unit tests is writing tests for each function and class. Tests are applied whenever there is a change in source code. If tests are written at the same time as development, this is called test-driven development.

Test result is determined by the test type and the expected output. The test could be written to see if the unexpected value is received when a wrong parameter is given as input. For example, if there is a test written for the square function, it can be true if the result is 25 when the input is 5. Also, an another test can be true if the result is not 25 when the input is not 5. Function passes the tests in either case as seen on Figure 4.29.

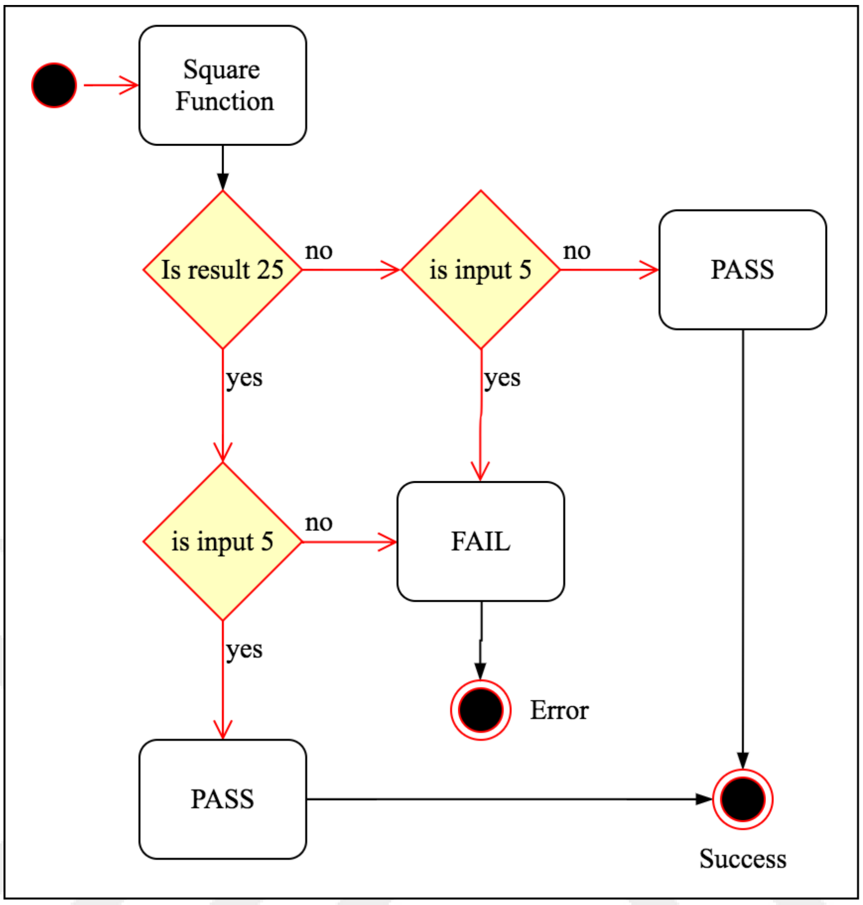


Figure 4.29. Tests for square function

4.2.5. Main Server Script

Main purpose of the server is to return the output of the received code. There are three stages of the server scripts. First, code goes through a pre-check. Then, unit tests are run with program. Finally, code is run if it passes the tests. Server code is written in Go programming language. The following code makes server start listening on the desired port.

```
http.ListenAndServe(:8080, nil)
```

Server controls the start and stop operations of containers. If a container belonging to the same user is still running and new code is submitted, container is forcefully stopped before running new code. Server also creates and calls unit tests. Test files are created separately for each program because each file name is different and contains the identification number of the owner. Tests are written as templates that are combined with the current user program information to create the test file. Unit tests are written when a new content is added to the system. For example, when instructor adds a new question, unit tests should also be added. Alternatively, platform also supports output comparison for simpler programs.

Tests are stored in the machine after creation. If user never attempts to a questions, tests are not created to reduce storage usage. Server also fills the code in the user interface if user was submitted code for the question. When user first connects to the server, previously written code is returned to the browser to fill the code editor. Results are also stored in the filesystem for instructor to see the results of the programs written by students. It also helps to gather analytics data such as most common mistakes and error types. Server interacts with the database if the user progress needs updating. Figure 4.30 shows the interaction between the server and other components.

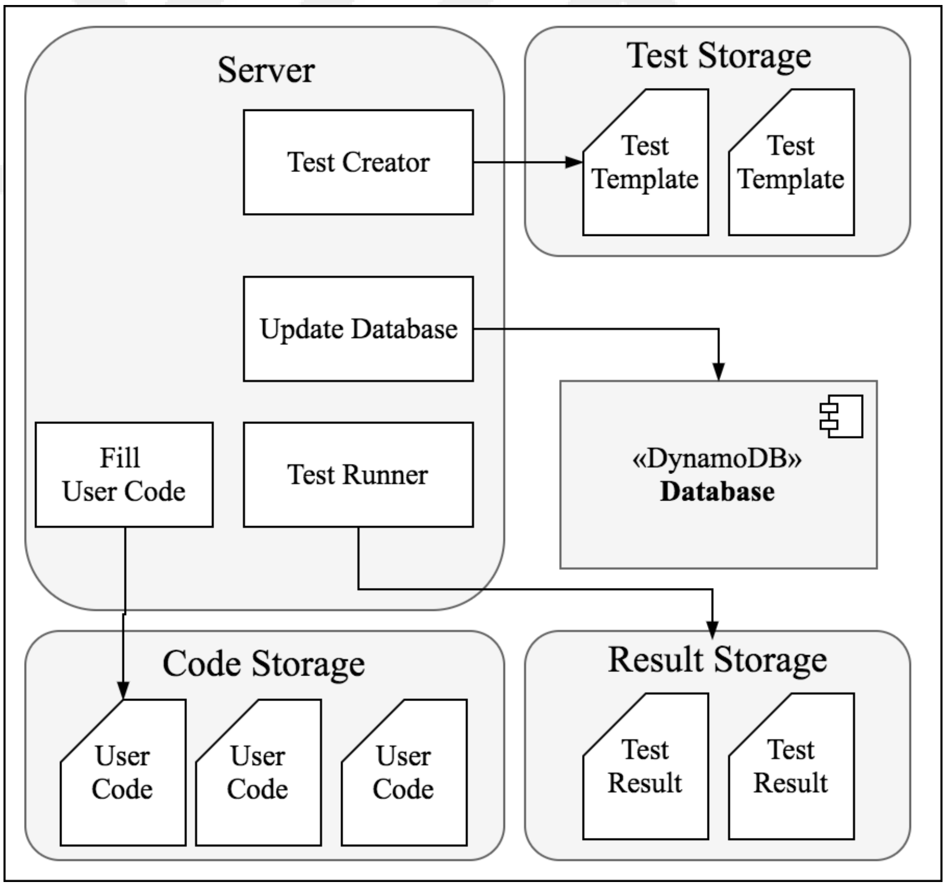
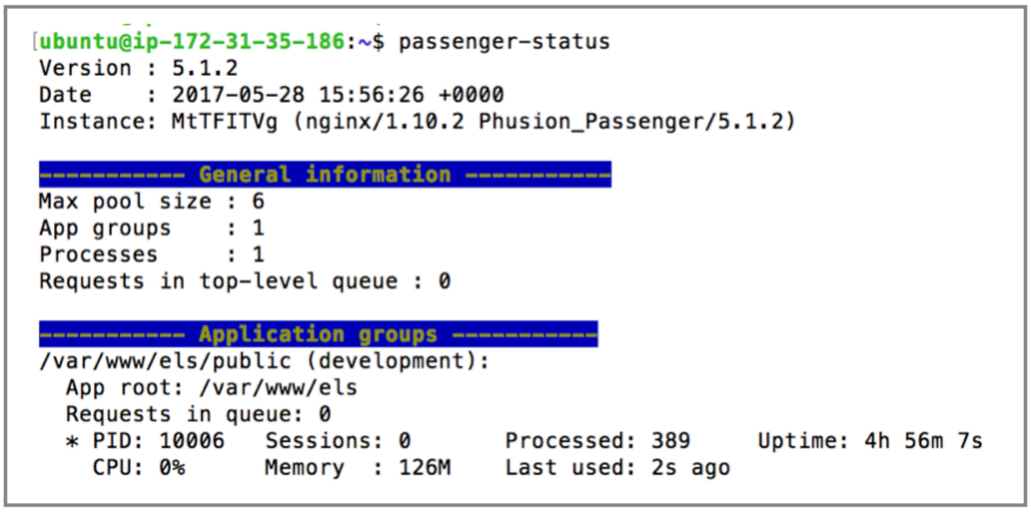


Figure 4.30. Server and the filesystem components

4.2.6. Passenger

Server machine also contains a service called Passenger [58] to manage the Rails application. The service allows application to start and stop at startup and handle traffic. Figure 4.31 shows status of an example Passenger server.



```
[ubuntu@ip-172-31-35-186:~$ passenger-status
Version : 5.1.2
Date    : 2017-05-28 15:56:26 +0000
Instance: MtTFITVg (nginx/1.10.2 Phusion_Passenger/5.1.2)

----------- General information -----------
Max pool size : 6
App groups    : 1
Processes     : 1
Requests in top-level queue : 0

----------- Application groups -----------
/var/www/els/public (development):
  App root: /var/www/els
  Requests in queue: 0
  * PID: 10006   Sessions: 0      Processed: 389    Uptime: 4h 56m 7s
    CPU: 0%      Memory  : 126M   Last used: 2s ago
```

Figure 4.31. Passenger server status

4.3. DATABASE

A database is a structured set of data that can be easily managed, accessed, and stored. In this project, a cloud database developed by Amazon called DynamoDB was used. Since the application is prioritizing scalability, cloud database is an appropriate solution. DynamoDB is a NoSQL database which will be summarized in the following sections.

4.3.1. NoSQL Databases

NoSQL (’non-SQL’ or ‘not only SQL’) database is a data store where the data storage and retrieval is different from the relational databases. NoSQL databases are non-relational, horizontally scalable, and open source. Some of the most popular ones are MongoDB, Cassandra, Couchbase, and DynamoDB. NoSQL databases are created for responding to demands of the new technologies. For example, new applications are dealing with massive amount of data that can be structured or unstructured. They both can be stored in the same table. There are four types of NoSQL databases. These are document databases, graph stores, key-value stores, and wide-column stores.

Database used in this work is used to store user data, code, tests, and content. Since they have different types, NoSQL was suitable to store them.

4.3.2. Traditional Databases and NoSQL

There are differences between traditional relational databases and NoSQL databases. First of all, NoSQL databases support sharding whereas relational databases does not. Sharding is separating data between servers or locations to balance the server load. Theoretically relational databases can also be sharded, but the NoSQL databases support auto-sharding that allows to move data between locations and servers.

Second, relational databases only store data defined in their schema whereas NoSQL uses dynamic schemas. It is not possible to push an item to a relational database that is not included in the schema. But, NoSQL databases accept an item with a column that is not in the table which will be automatically added to the table.

Lastly, NoSQL allows document store. Data in JSON, XML, or any other format can easily be stored and retrieved. These are the most popular data storage methods. They are especially useful for IoT applications.

4.3.3. DynamoDB

DynamoDB is a NoSQL database developed by Amazon and it is the database used in this work. It is an automatic scalable database based on SSDs and multiple availability zones. It is offered as part of Amazon Web Services. Since all services work with each other, it offers a backup option to Amazon S3 which is a storage service. DynamoDB is a key-value store type database.

Rails application and server both use the same database. While server only updates certain fields in the tables, Rails models are based on the database tables. There are four tables which are lab content, learning content, user data, and exam as seen on Figure 4.32.

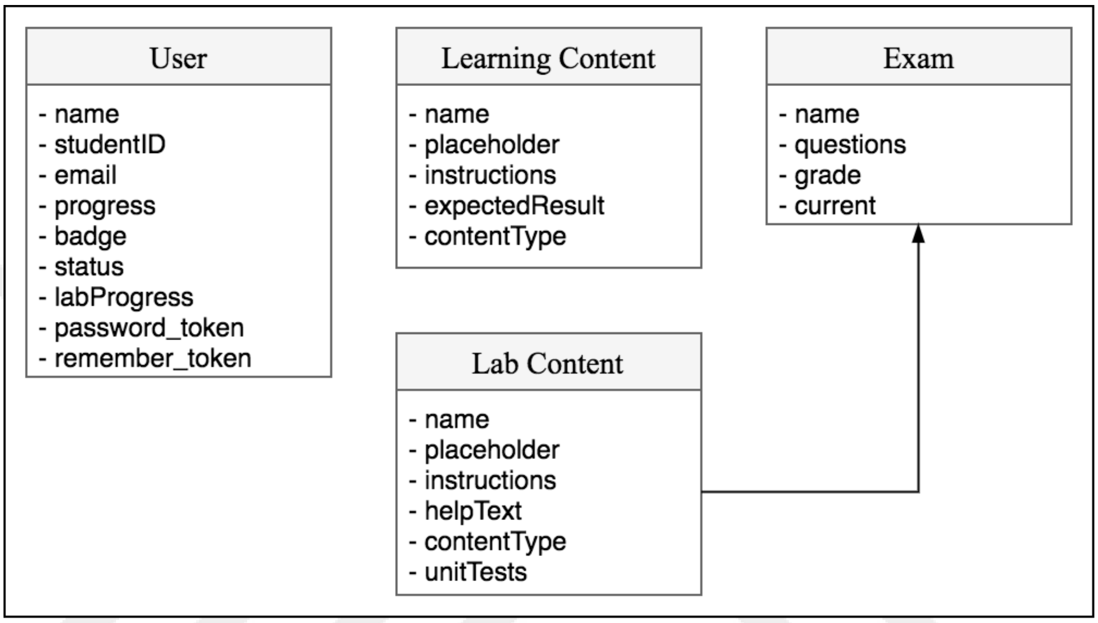


Figure 4.32. DynamoDB tables

5. RESULTS

This section will cover the results of the tests done in class and statistics gathered by following user behavior. There were 34 students in the classroom.

5.1. TESTING EXAM MODULE

The platform has been tested in class after the development is finished. It was used by the first year Computer Engineering students taking a class where Python is thought as their first programming language. First, students were given an assignment that could be submitted through the platform. The assignment was prepared using the exam module which can also be used for assignments. In this module, students cannot continue to the next question before completing the current one. It is because the most of the questions of the assignment require student to use the functions written in the previous questions.

Assignment given to students was to develop the game Uno so that the computer can play with itself and give the result after the game is finished. Uno is a card game where each player gets cards from the deck and plays them until one of them exhaust cards. The assignment is a simplified version of the real game.

The assignment has three parts. In first section, student implements the UnoCard class. Then, it is followed by implementation of CollectionOfUnoCards class which uses UnoCard class. Lastly, student implements the Uno class which implements the gameplay mechanics using UnoCard and CollectionOfUnoCards classes.

Each function of the given classes presented as a question to the student. Question contains the name of the function, instructions about the function input and expected output, and help text that offer hints about implementation. After student submits code, tests are applied in the server and students get a feedback on correctness. Student is allowed to continue to the next assignment after a successful response. Function names, sections, and definitions are given in Table 5.1.

Table 5.1. Functions of the assignment

Section	Function Name	Definition
1	def __init__(self, c, n)	Crates a card with color c, and number n
	def __str__ (self)	Returns the string that represents the UnoCard
	def canPlay(self, other)	Return true if other UnoCard is playable with the card
2	def __init__(self)	Creates an empty list
	def addCard(self, c)	Add card c to the end of the list
	def makeDeck(self)	Creates a deck with all cards
	def shuffle(self)	Shuffles the cards in the deck
	def __str__(self)	Return the string that represents the deck
	def getNumCards(self)	Return the number of cards in the deck
	def getTopCard(self)	Return the top card
	def canPlay(self, c)	Return true if there is a playable card with c in collection
	def getCard(self, index)	Return the card in location index
3	def __init__(self)	Creates a new game
	def playGame(self)	Plays the game
	def playTurn(self, player)	Executes one player turn
	def printResult(self)	Prints the game result

5.1.1. Class Performance

Students submitted their code by using the platform. Fifteen out of thirty-four students choose to submit their work through the platform as seen on Figure 5.1. It is typical to see fifty percent participation in this classroom as it is close to the number of regular participants in laboratory sessions.

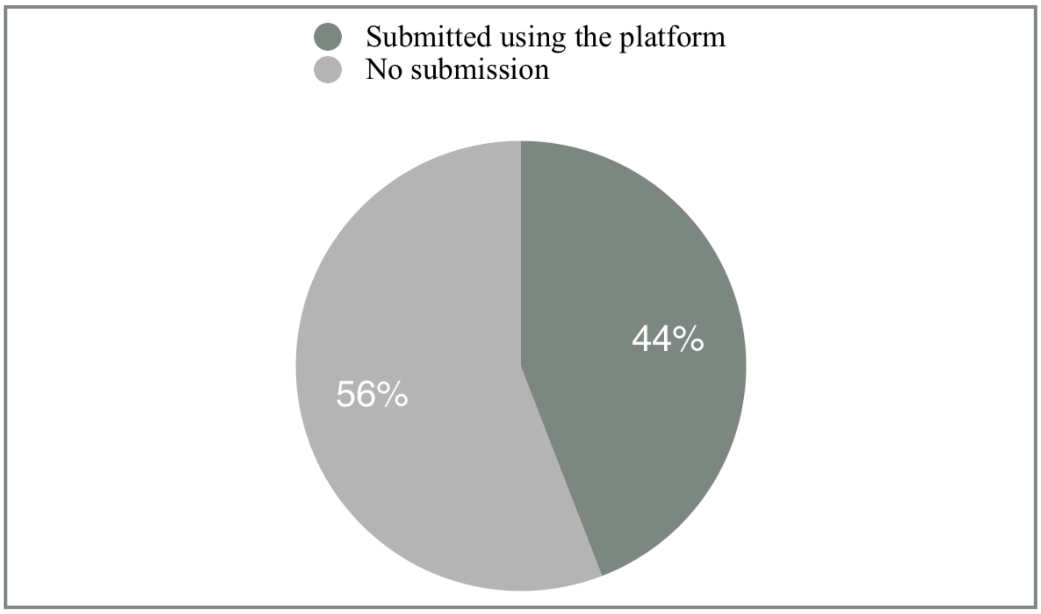


Figure 5.1. Assignment participation

In addition to the participation, completion rates were also calculated. 53 percent of the class completed more than half of the questions as seen on the Figure 5.2.

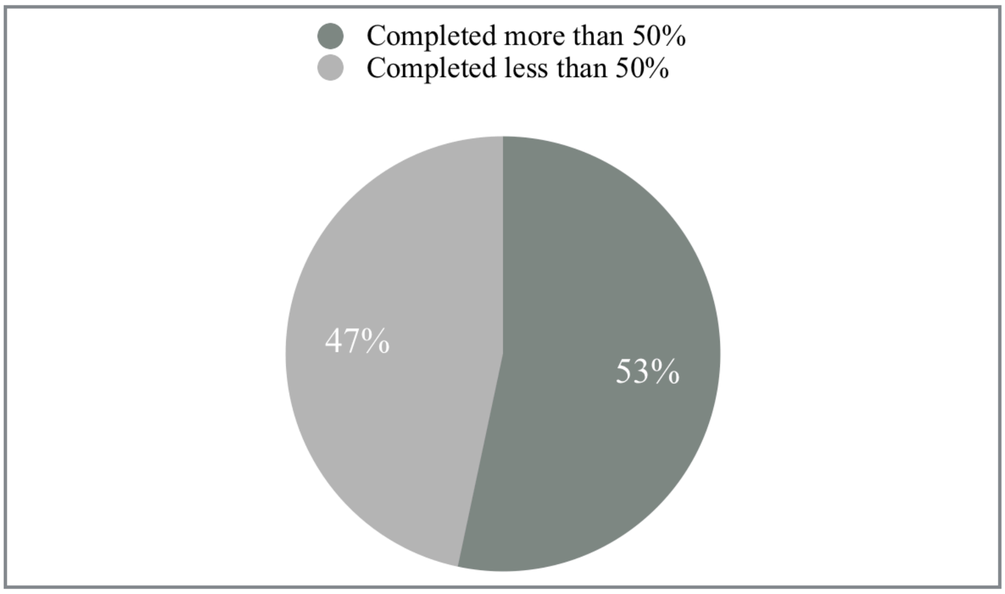


Figure 5.2. Completion rates

5.2. TESTING LAB MODULE

Lab module is prepared with a curriculum consisting of old exam questions for students to prepare for the final exam. There are twenty questions prepared with corresponding tests. The participation rates were fifty percent as given in Figure 5.3.

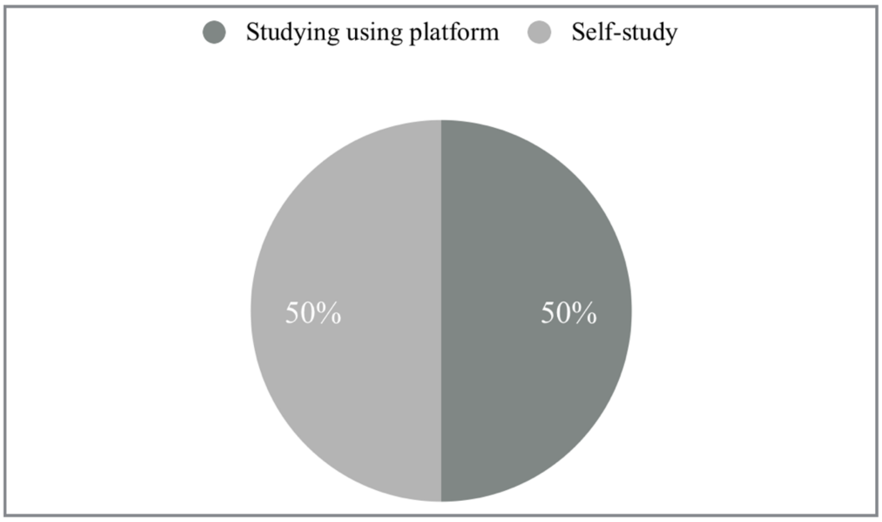


Figure 5.3. Participation rates of final module.

The results are grouped together by their grade. Grades between 0, 30, 70, and 100 created three groups as given in Figure 5.4.

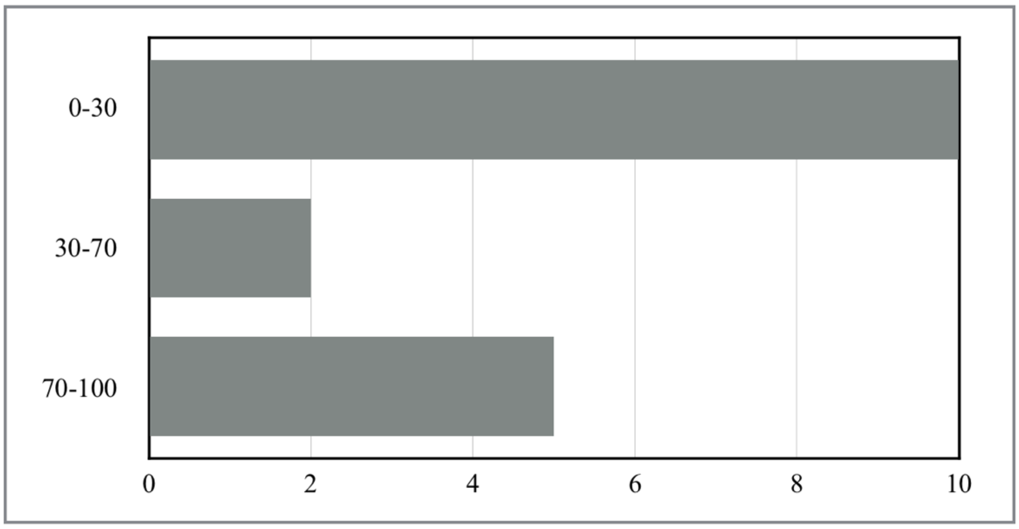


Figure 5.4. Grade distribution of final module

5.2.1. Questionnaire

Students were given questionnaires at the end of the semester. Aside from their participation, students were asked about their views on e-learning and gamification. The questions given to students were given below.

i. Did you use the platform for submitting Uno assignment?
ii. Did you use the platform while studying for the final exam?
iii. Did you aware of the badges you earned during the Uno assignment?
iv. Do you prefer to get immediate feedback for your assignment?
v. Do you trust an automated exam grading system?
vi. Do you prefer to work at your own pace instead of a group study or in-class learning?
vii. Is it hard to find exercises for studying programming exams?
viii. Does knowing the progress of your classmates motivates you to work more?

The answers students could give were definitely agree, mostly agree, neither agree or disagree, mostly disagree, and definitely disagree. Figure 5.5 shows the results of platform usage for assignment submission.

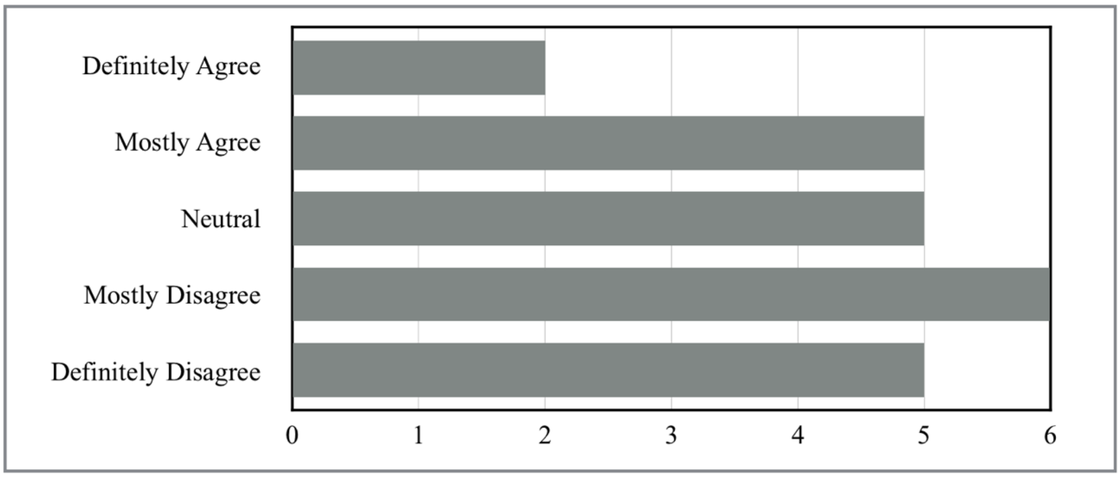


Figure 5.5. Platform usage for submitting assignment

Figure 5.6 shows results of the question regarding platform usage for studying the finals. Most of the students used the system for studying final exam.

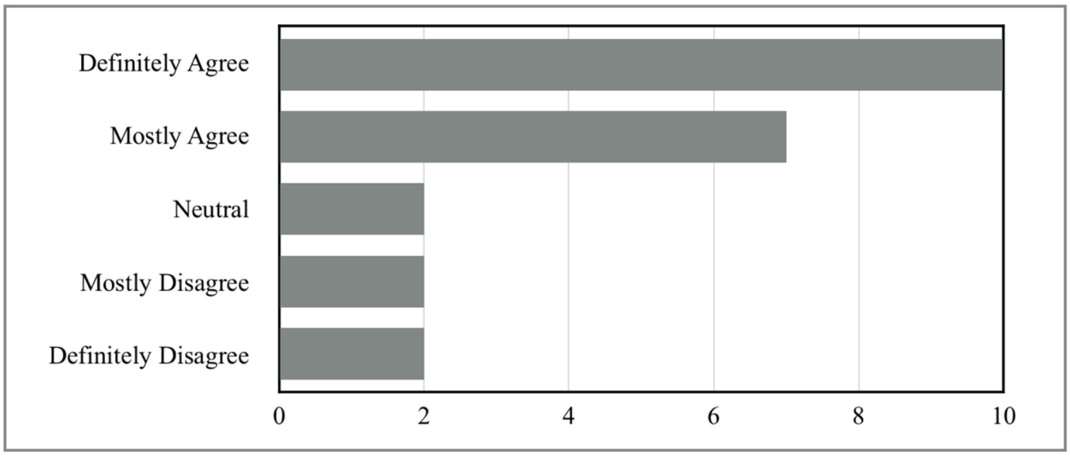


Figure 5.6. Platform usage for studying the finals

Most of the students did not acknowledge that they are earning badges while completing the Uno assignment. But, the number of students that completed more than half of the assignment were aware that badges were awarded for their progress. Some students answered that they did not acknowledge the badges were not using the platform for submitting the assignment.

According to Figure 5.7, students answered that they preferred to get immediate feedback for their assignments. The platform helps to achieve this by immediately showing the progress of the student and giving feedback on correctness of the student code. Usually, students do not know their grade and have to wait until they are graded by an instructor. By the time they receive the grades, it is already late for learning and correcting mistakes. Also, new material is introduced in class and students who are falling behind of the schedule have no change to catch up with the rest of the class.

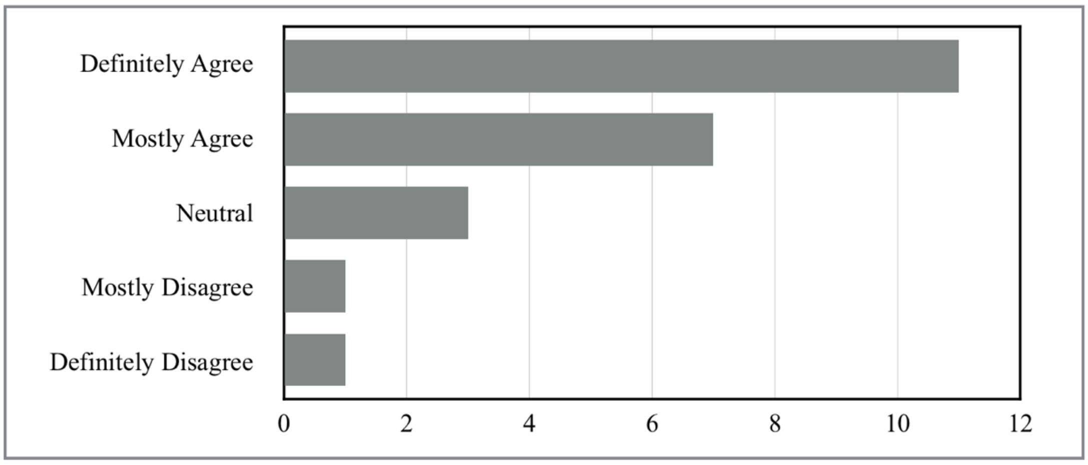


Figure 5.7. Immediate feedback for assignments

Another question in the questionnaire was about whether or not students trust an automated grading system. Most of the students answered that they do not trust an automated grading system as seen on Figure 5.8.

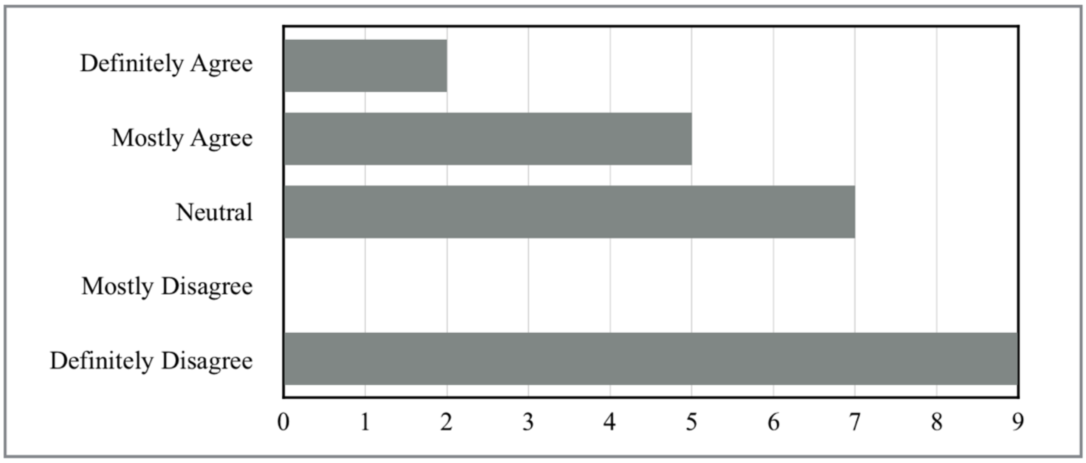


Figure 5.8. Trust on automated grading system

E-learning programs used in classes allow instructors to prepare a study curriculum that students can use for studying exams. While it can have disadvantages such as memorizing all answers to questions, it can also help to standardize exam studying. In this case, according to answers given by the students, they were having difficulties for studying exams. By using a platform similar to one designed in this work, students can access to study materials along with learning materials. Figure 5.9 shows number of answers to the question.

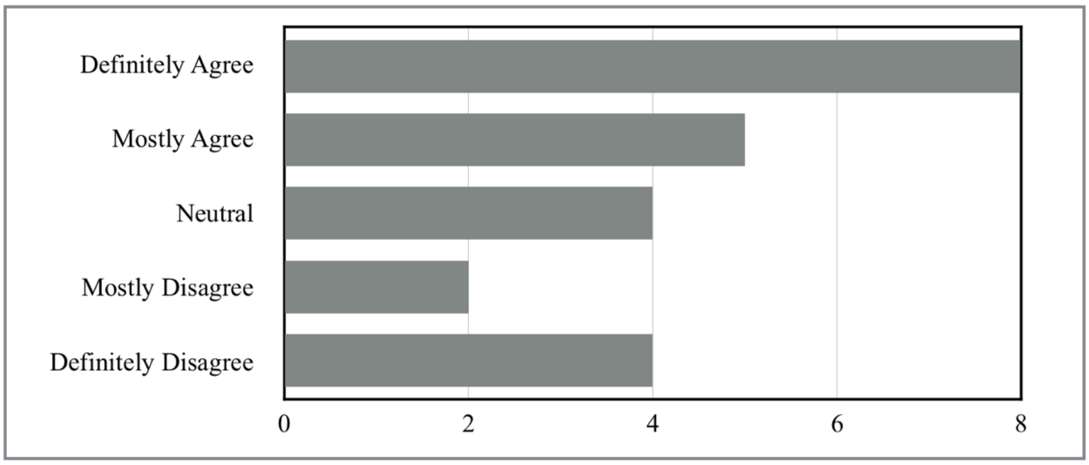


Figure 5.9. Students having difficulty finding exercises

Finally, a question about how knowing the progress of the classmates effect motivation were asked. Most of the students answered that it is more motivating when they know about the progress of others as seen on Figure 5.10.

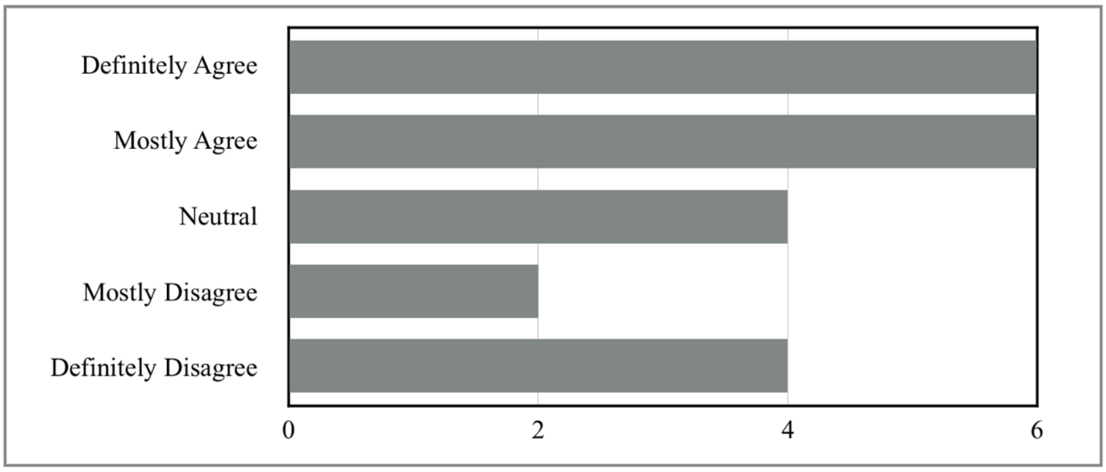


Figure 5.10. Effect of class progress on motivation

5.3. USAGE STATISTICS

Lab Student participation were also measured by using a statistics library called Google Analytics [59]. It was embedded to the website by adding a tracking script. The script tracks user location, system, duration of visits, frequency of visits, and visit source. It is one of the industry standard applications that used to measure user retention of websites.

Users visiting the website were almost 70 percent returning users and the remaining users were new visitors. A returning user means that same user has visited the website before, whereas a new visitor is a user that visited the website only once.

Session duration is the time spend on website for each visit. Average session duration for users was 12 minutes. Since the questions were short, students were using the system for answering couple of questions and then coming back to the system. Also, assignments were divided into sections and sections were divided into functions that allowed students to spend less time in the website while still continuing the learning progress. It was known that average page visits per session was 7 pages.

6. CONCLUSION AND FUTURE WORK

E-learning systems can be useful if used in classroom to teach programming languages. It helps to improve the teaching progress in many cases such as studying exams, submitting assignments, and learning the material. Even though the systems need time for adaptation by instructors and students, they are easy to adapt.

Gamified systems try to motivate and engage students by applying the known methods. But, each learning environment is different and gamification elements should be adapted accordingly. Preliminary results showed that students enjoyed earning badges. Also, they were motivated by knowing their progress and getting immediate feedback for their work.

A suggested use case of this work is where an instructor using the system to support the in-class learning. By doing so, instructor will know how the students are doing in real time. For example, if most of the students were having trouble at the certain part of an assignment, it will be wise to do a review session in class.

For the future work, the work will be improved to support exam taking. It is already grading the assignments and labworks, but the system can be used by students to take actual midterms and finals. Also, a similarity detection system to detect similar codes from the codes submitted by students will be useful for instructors. In addition to that, social sharing could be added to the system to make it more interactive for students.

REFERENCES

1. Swarm Application, https://www.swarmapp.com [retrieved 22 May 2017].

2. Khan Academy, https://www.khanacademy.org [retrieved 16 February 2017].

3. Duolingo, https://www.duolingo.com [retrieved 16 February 2017].

4. Whale, https://askwhale.com [retrieved 16 February 2017].

5. edX, https://www.edx.org [retrieved 22 March 2017].

6. Coursera, http://coursera.org [retrieved 22 March 2017].

7. Code School, http://codeschool.com [retrieved 22 March 2017].

8. Code Avengers, https://www.codeavengers.com [retrieved 22 March 2017].

9. Hacker Rank, https://www.hackerrank.com [retrieved 22 March 2017].

10. EduCause, "Asynchronous and Synchronous E-Learning", http://er.educause.edu/articles/2008/11/asynchronous-and-synchronous-elearning [retrieved 22 March 2017].

11. J.G. Tullis, and A.S. Benjamin. On the Effectiveness of Self-Paced Learning, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3079256/ [retrieved 22 March 2017].

12. The Knowledge Network for Innovations in Learning and Teaching at University of Albany, "Why Should I Use Self-Paced Learning?". http://tccl.rit.albany.edu/knilt/index.php/Why_Should_I_Use_Self-Paced_Learning%3F [retrieved 20 May 2017].

13. European University Association, “E-learning in European Higher Education Institutions”, http://www.eua.be/Libraries/publication/e-learning_survey.pdf [retrieved 20 May 2017].

14. MIT Open Courseware, https://ocw.mit.edu/index.htm [retrieved 20 May 2017].

15. Stanford Online, http://online.stanford.edu [retrieved 20 May 2017].

16. Lynda, https://www.lynda.com [retrieved 20 May 2017].

17. Roland and Berger Corporation, “Corporate Learning Goes Digital”, https://www.rolandberger.com/publications/publication_pdf/roland_berger_tab_corporate_learning_e_20140602.pdf [retrieved 20 May 2017].

18. Training Magazine, “2014 Training Industry Report”, https://trainingmag.com/sites/default/files/magazines/2014_11/2014-Industry-Report.pdf [retrieved 20 May 2017].

19. Udacity, http://udacity.com [retrieved 20 May 2017].

20. Extension Engine, “Putting Moocs to Work”, http://extensionengine.com/putting-moocs-to-work-recap-infographic/ [retrieved 20 May 2017].

21. SAP, “SAP Training”, https://training.sap.com/tr/tk/ [retrieved 20 May 2017].

22. Udacity, “Nanodegree Programs”, https://www.udacity.com/nanodegree [retrieved 20 May 2017].

23. 3C Interactive, “2017 Mobile Loyalty Report”, https://www.3cinteractive.com/resources/mobile-loyalty-report-2017/ [retrieved 20 May 2017].

24. IBM, “2017 Customer Experience Index (CEI) Study”, https://public.dhe.ibm.com/common/ssi/ecm/gb/en/gbe03818usen/GBE03818USEN.PDF [retrieved 20 May 2017].

25. Big Fish Games, “2016 Video Game Statistics and Trends”, http://www.bigfishgames.com/blog/2016-video-game-statistics-and-trends/ [retrieved 22 May 2017].

26. CNBC, “Mobile Addiction Growing at an Alarming Rate”, http://www.cnbc.com/2014/04/24/mobile-addiction-growing-at-an-alarming-rate.html [retrieved 22 May 2017].

27. Kevin Werbach, and Dan Hunter. *The Gamification Toolkit*, Wharton Digital Press, 2015.

28. LinkedIn, https://www.linkedin.com [retrieved 22 May 2017].

29. Enterprise Gamification, “SAPS”, http://www.enterprise-gamification.com/mediawiki/index.php?title=SAPS [retrieved 22 May 2017].

30. Booking.com, http://booking.com [retrieved 22 May 2017].

31. Coursera, “Gamification”, https://www.coursera.org/learn/gamification [retrieved 12 May 2017].

32. Richard Bartle. Hearts, Clubs, Diamonds, Spades: Players Who Suit Muds. http://mud.co.uk/richard/hcds.htm [retrieved 12 May 2017].

33. M.R. Regalado, E. Aranha, and T.R. da Silva. Gamifying an Online Approach for Promoting Game Development Learning and Contest: An Experience Report. *Frontiers in Education Conference (FIE), 2016 IEEE*, Eire, PA, USA. 1-8, 2016.

34. M. B. Ibáñez, Á. Di-Serio, and C. Delgado-Kloos. Gamification for Engaging Computer Science Students in Learning Activities: A Case Study. *IEEE Transactions on Learning Technologies*. 7.3:291-301, July-Sept. 1 2014.

35. G. Lückemeyer. Virtual Blended Learning Enriched by Gamification and Social Aspects in Programming Rducation. *2015 10th International Conference on Computer Science and Education (ICCSE)*, Cambridge. 438-444, 2015.

36. R. F. Maia, and F. R. Graeml. Playing and Learning with Gamification: An In-Class Concurrent and Distributed Programming Activity. *2015 IEEE Frontiers in Education Conference (FIE)*, El Paso, TX. 1-5, 2015.

37. K. Tsalikidis, and G. Pavlidis. jLegends: Online Game to Train Programming Skills. *2016 7th International Conference on Information, Intelligence, Systems and Applications (IISA),* Chalkidiki. 1-6, 2016.

38. K. Berkling, and C. Thomas. Gamification of a Software Engineering Course and a Detailed Analysis of the Factors That Lead to It's Failure. *2013 International Conference on Interactive Collaborative Learning (ICL),* Kazan. 525-530, 2013.

39. U. Jayasinghe, and A. Dharmaratne. Game Based Learning vs. Gamification from the Higher Education Students' Perspective. *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE),* Bali. 683-688, 2013.

40. G. Barata, S. Gama, J. Jorge, and D. Goncalves. Engaging Engineering Students with Gamification. *2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES),* Poole. 1-8, 2013. .

41. N.O. Uzun. *A Gamified Lesson Preparation System.* Master's Thesis, Ozyegin University, Turkey, 2017.

42. Redmonk, “The RedMonk Programming Language Rankings: January 2017”, http://redmonk.com/sogrady/2017/03/17/language-rankings-1-17/ [retrieved 12 May 2017].

43. Ruby on Rails, http://rubyonrails.org [retrieved 12 May 2017].

44. Go Programming Language, https://golang.org [retrieved 12 May 2017].

45. Amazon, “What Is Amazon DynamoDB”, http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html [retrieved 12 May 2017].

46. Docker, “Docker for Developers”, https://www.docker.com/what-docker#/developers [retrieved 12 May 2017].

47. Amazon EC2, “Amazon Elastic Compute Cloud Documentation”, https://aws.amazon.com/documentation/ec2/ [retrieved 12 May 2017].

48. Bootstrap, http://getbootstrap.com [retrieved 12 May 2017].

49. Amazon AWS, “What is AWS”, https://aws.amazon.com/what-is-aws/ [retrieved 12 May 2017].

50. Ruby, https://www.ruby-lang.org/en/ [retrieved 12 May 2017].

51. WebSocket, https://en.wikipedia.org/wiki/WebSocket [retrieved 12 May 2017].

52. jQuery, “jQuery API Documentation”, http://api.jquery.com [retrieved 12 May 2017].

53. Ubuntu, https://www.ubuntu.com [retrieved 20 May 2017].

54. Google Cloud Platform, https://cloud.google.com/products/ [retrieved 20 May 2017].

55. Heroku, https://www.heroku.com [retrieved 20 May 2017].

56. ZDNet Magazine, “What is Docker and Why is it so Darn Popular?”, http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/ [retrieved 20 May 2017].

57. Brian Christner. Docker Base Image OS Size Comparison. https://www.brianchristner.io/docker-image-base-os-size-comparison/ [retrieved 20 May 2017].

58. Phusion Passenger, “Ruby Deployment Tutorial”, https://www.phusionpassenger.com/library/walkthroughs/deploy/ruby/ [retrieved 20 May 2017].

59. Google Analytics, https://www.google.com/analytics [retrieved 20 May 2017].