ADDIS ABABA UNIVERSITY

FACULTY OF NATURAL AND COMPUTATIONAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

AAU PUSH

FINAL PROJECT

BY

Addismiraph Abebe	NSR/2009/08
Biruk Tamiru	NSR/9151/08
Amanuel Shimeles	NSR/6860/08

Project Advisor: Berhanu Abebe

January 2019

DECLARATION

This is to declare that this project work which is done under the supervision of Berhanu Abebe and having the title AAU Push is the sole contribution of:

ADDISMIRAPH ABEBE NSR/2009/08

BIRUK TAMIRU NSR/9151/08

AMANUEL SHIMELES NSR/6860/08

No part of the project work has been reproduced illegally (copy and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: _____

Group Members:

Full Name	Signature
_____	_____
_____	_____
_____	_____

CERTIFICATE

I certify that this B.Sc. final project report entitled

AAU Push by:

ADDISMIRAPH ABEBE NSR/2009/08

BIRUK TAMIRU NSR/9151/08

AMANUEL SHIMELES NSR/6860/08

is approved by me for submission. I certify further that, to the best of my knowledge, the report represents work carried out by the students.

_____ _____

Date Name and Signature of Supervisor

Table of Contents

List of Figures

List of Tables

Definition, Acronyms and Abbreviations

Abbreviations	Definitions
AAU	Addis Ababa University
Push	Short version for the AAU Push system
CNCS	College of Natural and Computational Sciences

1. INTRODUCTION

1.1. Background

Addis Ababa University (AAU), established in 1950 is the oldest and largest higher learning and research institute in Ethiopia. Today, AAU has more that 33,000 undergraduate students and over 2,000 academic staff in its 14 campuses.[1] Creating a conducive learning environment is a multifaceted challenging feat. Part of the challenge comes from the number of moving parts involved in making the University function in harmony — the academic staff, supporting staff, student body, administration, leadership and many more.

One of these parts, important in the teaching-learning process, is the communication between students and their instructors. Most commonly, instructors relay message to students through a class representative, an assigned student for each section. This representative has the responsibility of making sure every student in the section receives the announcement, the handout copy or any other information. Aside from the class representative, staff members use the noticeboard to post announcements.

In November of 2015, during our first year at the Department of Computer Science, we noticed the inefficiency of the communication. With this problem in mind, we built AAU Push. A website that gives teachers a simple way to post announcements and upload course textbooks, references and assignments for students. Students receive these information through the website and the Android application. A similar system, made by an American company, called Piazza (piazza.com) is used by the Software Engineering Department at Addis Ababa Institute of Technology.

The utility of AAU Push was evident over the following two years as the staff of Department of Computer Science used it to communicate to students. It relieved unnecessary pressure from the class representative and empowered instructors to be more interactive in and outside of class. Currently this system is to be placed to all departments in the College of Natural and Computational Sciences (CNCS) under the recommendation of the dean of CNCS.

Although this system is good and helpful, it is not without shortcomings. In this project we seek to find out problems in this current system and improve its usability, practicality and adoption.

1.2. Statement of the Problem and Justification

AAU Push (Push) currently requires both instructors and students to signup and login to use the website. There, instructors share information, announcements or files like assignments, references and PDF handouts.

The system is faces multiple limitations:

1. The communication on the platform only allows flow of information in one way that is instructor to student. This is mean students don't have a way to ask questions or submit assignments.
2. The system only works over the internet. When file sizes exceed a certain limit, downloading the resources will be difficult. Large file sizes also incur higher costs for server space.
3. Although the system has had an Android app for students, it is now out of date and useless. Furthermore, there is not application available for iOS users and Instructors.
4. Instructors has reported several usability issues on the interface.

In addition to these limitations, there are many more features that can be added to the system to enhance the teaching-learning experience. Fixing these issues and developing the system further will help students be well informed and productive because AAU Push delivers, in one place, all information from teachers in the most organized manner. It will also help instructors be at ease and reassured because Push lets them reach all their students in the fastest, reliable and convenient way from anywhere at anytime.

1.3. Project Objective

1.3.1. General Objective of the System

The general objective of this project is to create e-learning communication platform to aid the instructor-student relationship.

1.3.2. Specific Objective of the System

- Document and analyze the current system
- List the shortcomings to the current system
- Gather list of features to add and errors to correct
- Develop features as needed
- Test and deploy the new system

1.4. Scope of the Project

This project will develop the current system to:

- Enable Instructors to edit and delete their posts
- Have local servers to increase speed and availability
- Have Android and iOS apps
- Allow students to submit assignments and ask questions
- Create accounts to be used by College Dean and Registrar Offices

1.5. System Development Methodology

Since our system is driven by user interface requirements, we will be using Agile as our software development approach. We will be putting less emphasis on planning and more emphasis on an adaptive process.

1.5.1. Investigation (Fact-Finding) Methods

We will be investigating the pros and cons of the current system by using the following methods:

- Survey: collect comments from students
- Interview: ask instructors and students in-depth of what they think of the product
- Observation: see how instructors use the product to find usability issues

- Research: review the landscape of similar e-learning solutions to make note of what could be missing from AAU Push

1.5.2. System Development Tools

The system will try to solve the problem using web and mobile technologies and hence the system shall have a website and applications that is capable of solving the problems listed above. The system shall be implemented using the following technologies.

1.5.2.1. Documentation Tools

The documentation tools are used to prepare the document for the system. They are listed below.

- Pages 6.2: We use this word processing software as the primary tool for writing the document.
- www.draw.io: This is a free website that enables us to draw UML diagrams easily. It syncs and saves to Google Drive so it's easily sharable. We also chose it because it works across different computer platforms as it is a website.
- Microsoft Office Visio: We have also chose this tool to make more detailed diagrams.
- TeamGantt: online tool to make time schedule diagram.

1.5.2.2.Communication and Collaboration Tools

These tools and technologies are the major one that we are going to use for version control and interaction with the team.

- Telegram: - We will use this to communicate with each other among team members and plan tasks.
- GitHub: - We will use this to keep version control of documentation. We use it to stay in sync with our advisor. We will be using it set up version control while we work on the development.
- Email: - We use this for general communication with our advisor.

1.5.2.3. Design Tools

The design tools are used to produce user interface samples for the system.

- Keynote: This product enables to draw fast and easily. We chose it because we were already familiar with it.
- Adobe Illustrator CC: We use this tool to make logos and colors for the interface design.

1.5.2.4. Environment

- Laptop with any operating system and recommended with Core i5 Processor, 4 GB RAM or higher.
- Any Android Phone with operating system Android 4.1 minimum
- iOS device with operating system iOS 9.0 minimum
- Python is used as the language to develop the backend of the system with Django framework. Django is a high-level web framework which was created for quick web project development. It delivers transparent and high-quality code writing, making it important for developers, and equally important for customers.
- HTML, CSS and Javascript are used to build frontend interfaces along with required libraries and frameworks.
- React Native is a framework that enables web developers to create robust mobile applications using their existing JavaScript knowledge. It offers faster mobile development, and more efficient code sharing across iOS, Android, and the Web, without sacrificing the end user's experience or application quality. We use this cross-platform framework to make Android and iOS apps.
- Python will be used to develop a Telegram bot. Because there are a number of APIs available that make development faster.

1.6. Significance of the Project

This solution will empower the academic staff to teach without limitations in communication. We will see an increased level of interaction with students.

Push will also let students focus their efforts on studying as opposed to stressing on how to get resources. It will help them stay informed and feel like they are on top of their work. This will result in boosted confidence and academic performance.

1.7. Beneficiaries

The beneficiaries of the new system are teaching staff members and their students.

1.8. Time Schedule

The timeline for our project is shown in Figure 1.1.

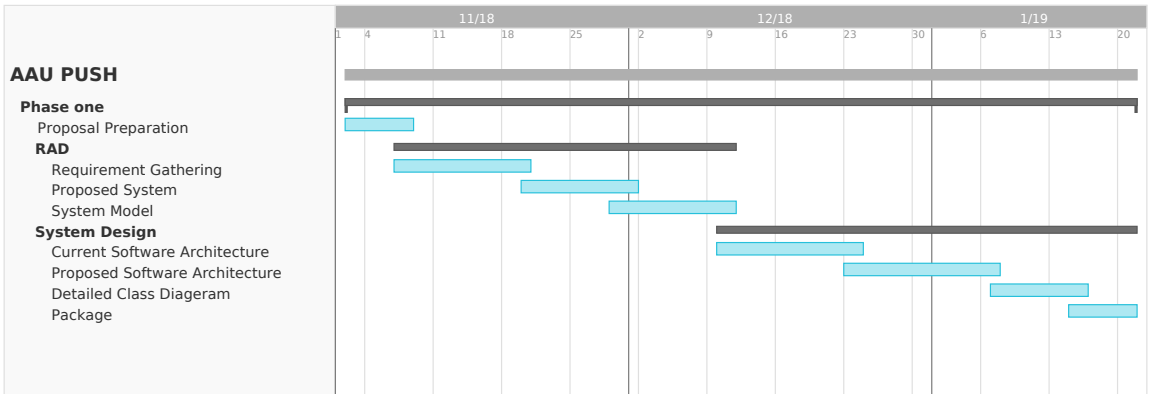

Figure 1.1 Project Schedule

2. REQUIREMENT ANALYSIS

2.1. Introduction

The purpose of AAU Push is to serve as a communication platform between teachers and students in the college environment. In this regard, we will need to review each aspect of this communication.

2.2. Current System

There are two major ways teachers and students communicate in Addis Ababa University.

Method One: Word of Mouth

The first method is communication using a class representative. This is most common practice used by teachers to reach their students. It works in such a manner: each section of class has an assigned student that is in charge of communicating with the teacher. This student, called the class representative (Rep), talks to the teacher via phone to get announcements. Once the Rep is up to date with the latest information, he/she announces to all students in any of the following classes or, if available, in the social media group for the class. This social media group can be on Viber, WhatsApp, Telegram or other group chatting application available. In addition, if the students has a question, for example to request a change of date for an exam, the Rep will be the one to ask the teacher. When the teacher needs to give out handouts to the students, there are several methods utilized. The first and most common is giving a printout of the handout to the students to copy for themselves. The teacher could give that printout to the Rep or leave it at a photocopy shop nearby that students can go to. Another way teachers distribute class materials is by copying it to students' flash drives. Typically the teacher would bring his/her laptop to the class and copy the material to all students with a flash drive at hand. Then students who did not have a flash would be responsible to get the files from their classmates. This method has one major issue that has been a problem for most teachers: the issue of computer viruses. As teachers plug and use multiple flash drives, they are vulnerable to attack from any virus present in these drives.

There is one other new way for distributing files: using social media applications. One of the most popular is Telegram, a social chatting application that has boomed in Ethiopia over he past two years.

Teachers have been observed sending class files to Reps over this platform. And then the Reps would in turn forward these into the class groups for all to download. Or teachers create what is called a Channel on Telegram, this is similar to a group chat however only permits one user to send messages to the group and that user is the creator of the group. Teachers create a channel and let their students join. This way of communication is one-way. The teacher sends information on one side and students get it on the other.

Method Two: AAU Push

The second method of communication and the focus of this project is AAU Push. A website used in Addis Ababa University, and more specifically in the Department of Computer Science. Development of this website started in late 2015 by computer science students and is still maintained by them. It is currently a one-way communication platform that seeks to bridge the communication gap between teachers and students with a focus on the undergraduate program. It allows teachers to post announcements and materials for all their students at once. For this system to serve its purpose, it requires two parties to be present, all the students of the department and their instructors. Let us see in detail how this system works.

The system is first initialized. The department provides its name, course offering and number of year of the program with the sections available. Once this is fed into the system, it is ready to onboard students and teachers.

Initially, all students of the department are made to signup on the homepage of the website. There each student will enter his/her name, ID, year, section and phone number. By signing up, they are automatically subscribed to all the courses their section is taking. Once they complete signing up, they are able to login using their ID and password. Upon login, they reach the Student Dashboard.

At the Dashboard, there are a number of tabs on the menu. The first enables them to view the latest announcements. On the next tab titled ‘Courses’, all the courses a student is subscribed to at listed. Students can click on the title of the course to see all the materials that have been uploaded by their teachers. They can download the material simply by clicking on the title of the document.

The following tab in the menu is ‘Add/Drop’. Although students are automatically subscribed to all courses their section takes, it does not mean that a particular student is taking all of them in reality. Students add and drop courses from one semester to another because of many factors. To reflect this reality, students can unsubscribe a course by ‘dropping’ it in the ‘Add/Drop’ menu. Conversely, they are also able to add a course by specifying from which year and section they are taking it from. That way, announcements and materials for that course will appear in their Dashboard.

The next tab is ‘Account’ where students can update their personal and contact information. Following the ‘Account’ tab is a logout button to exit the system. This concludes the list of functionalities students receive through Push. Let us now see how Push is used by teachers.

Teachers are registered on to the website by invite only. An Excel sheet with the list of email of faculty members (or instructors) of the department, for example Department of Computer Science, is collected from the department. Once Excel file is fed into the system, Push creates users and sends out automated invitations to all instructors to join and activate their accounts. This email contains the email and temporary password that the instructor uses to login. Once an instructor open this email, they can use the credentials provided to login into the system. On their first login, he/she is asked to fill in full name and create a new password for their account. After this step they will have access to the Instructor Portal where they are able to send announcements and upload materials to their students.

Once at the Portal, instructors see a form on a tab titled ‘Post’. This form has a field to write the message of the announcement. Once instructors write their message, they pick the section they would like to send it to by ticking on the section from the list in a section titled ‘Send To’. This section holds the list of sections the teacher gives a course to. However, when teachers have just activated their accounts, there is nothing listed in the ‘Send To’ section because they have not yet specified what courses they are giving. To do this, they go to the tab titled ‘Class List’. There, they can browse through the departments year, then section in a hierarchical order then finally tick the course they teach each section from the list. This will update the ‘Send To’ on the ‘Post’ form.

To send an announcement, instructors write their message in the Post form, pick a section to send it to and click on 'Send'. On the 'Post' instructors can attach or upload up to three 3 files and 2 images. The files will be listed as materials under the specific course on the student's Dashboard.

Instructors can see the viewership of an announcement or material from the 'Tracker' tab. It displays the announcement message and the number of students that have seen it. The same is true for materials and its number of downloads.

Similar to the student's Dashboard, the Portal has an 'Account' tab where instructors can update their personal information. Below that tab is a logout button to exit the system.

2.2.1.Major Functions of the Current System / Current System Description

- Allow students to sign up on system using their ID, year and section
- Allow students to login to the system to access Student Dashboard
- Allow students to view announcements from their teachers
- Allow students to download materials sent from their teachers under each course
- Allow students to 'add' a course to start receiving notifications and materials on that course
- Allow students to 'drop' a course to stop receiving notifications and materials on that course
- Allow students to modify their personal information
- Allow teachers to signup to the system by email invitation
- Allow teachers to login to the system to access Instructors Portal
- Allow teachers to pick which section they are teaching and what course
- Allow teachers to send announcements to students for specific course they teach
- Allow teachers to upload materials to students for specific course they teach
- Allow teachers to see the number of students that have seen a specific announcement
- Allow teachers to see the number of students that have download a specific material
- Allow teachers to modify their personal information

2.2.2.Problem of the Existing System

Over the time AAU Push has been in use, a number of issues have been seen.

Firstly, we can see that the system is not self-contained or fully automated. When teachers or students forget their passwords, the only way to reset it is by calling the administrators of the website. The admin will reset their password and give them a new one. Password recovery by email is missing from the system. Similarly, once an instructor has sent an announcement, there is no way to edit it or remove it unless an admin intervenes. This would mean that if a teacher posts the wrong information by mistake, correcting it will not be simple.

Secondly, communication over this platform is one-way only: from teachers to students. Even though there are many cases where students would need to give an input, this is not reflected on Push. For example, feature for student assignment submission and option for asking questions is missing.

Thirdly, using AAU Push to transfer large files over the internet is difficult. Push currently only works when teachers and students are connected to the internet. On normal cases, this should not be an issue. However, teachers often want to send files with large sizes like video or bulky textbooks. In this situation, it becomes difficult or time taking for teachers to send it. And then students will have to look for Wi-Fi to download that large file because it will be too expensive to do it via mobile data. So Push does not solve the need of sending large files in a fast and cheap manner.

Finally, the system is limited to a website. In todays fast and connected world, doing work on-the-go is demanded. For Push, mobile applications are currently not available.

2.2.3. Requirement Gathering Methodologies

1. User Observations: We have gathered requirements by watching teachers and students interact with the system in different use cases like sign up, login, post announcement, download materials.
2. Survey: Used a Google Form to collect reviews from students.

3. User Interviews: We have collected requirements by interviewing teachers and students to find what is missing. Issues of the system and feature lists have been developed from this activity.

4. Review Other Systems: to develop feature lists, we have also researched other systems to see what needs to be added to ours.

2.3. Proposed System

2.3.1.Overview

This part of the document specifies a system that can be used to solve the current systems problems that were mentioned earlier.

2.3.2. Definitions

Year: is the number identifying the level of the student. For example, a student entering the University would have a year of 1. The following year will be 2. The limits of the year will depend on the program the student takes.

Section: a distinct group within a larger body of all students in a specific year. For example, a first year student in the second section would have a Section value of 2.

Course: a series of lectures or lessons in a particular subject. For example, Introduction to Computer Science

Class: is a course being given to a specific year and section by an Instructor. For example, Introduction to Computer Science being given to Year 1 Section 1 students is one class. The same course given to Year 1 Section 2 students is another class.

Post: is a message with text and optionally files and images

Staff: an employee working for the University under a specific department or in the administration.

Portal: is the index page a Staff views when logging in to AAU Push. From there the user can access a number of functionalities.

Dashboard: is the index page a Student views when logging in to AAU Push.

Forum: is a chatting area for both students and staff members. It is used to ask and answer questions or discuss on topics.

Reminder: is a post that an Instructor gives to Students to set a time deadline for a certain activity. For example, an upcoming test or assignment submission.

2.3.3. Functional Requirements

1. Students, Teachers (Instructors), Department Heads, Registrar Officers and the College Dean should all have accounts to access the system.
2. All users should be able to login.
3. All users should be able to log out.
4. All users should be able to recover their password through the email.
5. All accounts must be verified using the personal information provided.
6. All users should be able to participate in and create discussion forums.
7. Forum participants should be able to send and receive messages and images.
8. All users should be able to search forums by their names or forum ID.
9. Students should be able to submit assignments.
10. Students should be able to access posts sent to them from their teachers, their department head, program coordinators, registrar and the university dean.
11. Students should be able to mark posts as ‘Read’ to indicate that they have read it.
12. Students should be able to add classes to get information from teachers and drop classes to discontinue information from a specific class, so that the information displayed will be relevant.
13. Students should be able to update their account information, but they shouldn’t be able to update their name, ID or department.
14. Students should be able to access Reminders set by their teachers, for upcoming assignments, tests and projects.
15. Students should be able to submit assignments to their teachers using Reminders set for assignments.
16. Teachers should be invited to sign up via email by Department Head.

17. Teachers should be able to add classes they are teaching in order to post announcements and upload materials.

18. Teachers should also be able to remove classes that they are not teaching anymore. Teachers should specify the year and section of the courses they are giving to add their classes.

19. Teachers should be able to send posts for announcements and uploading materials. They should choose the classes they want to post to.

20. Teachers, Department Heads, College Dean and Registrar Officers should be able to see how many students got their post, how many of them marked it as 'Read' and how many times the material they posted was downloaded.

21. Teachers should be able to edit their announcements.

22. Teachers should be able communicate over a group chat where they can post messages to members of their departments.

23. Teachers should be able to update their account information, but they shouldn't be able to update their department.

24. Teachers should be able to set Reminders for upcoming assignments, tests and projects.

25. Department heads should be able to post announcements and upload materials to students in his/her department. They should choose the recipient, either year or section in their department.

26. Department heads should be able to send single invites to teachers.

27. Department heads should be able to invite a single teacher or many teachers using Excel Sheets.

28. Department heads should be able to disable teacher accounts.

29. Department heads should be able to post in their department group chat.

30. Department heads should be able to participate in and create discussion forums.

31. Deans should be given accounts by the administrators. They should change their password when they login for the first time.

32. Deans should be able to post announcements and upload materials to students in his/her campus/university. They should choose the recipient, either department, year or section in their campus/university.

33. Deans should be able to only post in department group chats not read messages.

34. Registrar offices should be given accounts by the administrators. They should change their password when they login for the first time.

35. First, registrar offices should be able to post announcements and upload materials to students in his/her campus/university. They should choose the recipient, either department, year or section in their campus/university.

2.3.4. Non-Functional Requirements

2.3.4.1.User Interface

- All users should only be required to have basic knowledge of using computer in order to use the system.
- Reminders for students should be displayed at all pages when the student is logged in.

2.3.4.2.Access Control

- Student should login using his/her ID
- Discussion forums should be either closed or open. Closed forums should be joined either by providing a code set by the forum creator or based on characteristics of the students, like year, section, department, courses and campus. Every forum should have a forum ID, name and description.
- Teachers should be able to add only courses given by their department.

2.3.4.3. Availability and Speed

- Local servers placed within the university's network should host files sent by teachers. This is will increase availability and download speeds of course files. Whenever a student is within the university, Internet connection is not required to download files. This should feel seamless to the students. The data stored in local servers within the University should be up to date with the data on the cloud server.

- The system should have mobile applications and Telegram bot in sync with the website.

2.3.4.4. Documentation

- Full documentation of the system including the Project Proposal, System Requirement Analysis Document, System Design Document and the Object Design Documents will be compiled to facilitate future reference and system maintenance.

2.3.4.5. Usability

- To students, delivery of information from teachers is more important than delivering general announcements from the university dean, registrar or their department head. Therefore, when displaying information there should be a distinction between information from their teachers and information from their department head, program coordinators, registrar and the university dean, giving more emphasis to the first one.
- Students should specify the year and section of the courses they are taking to add their classes. Students should be able to add only courses given to their department. Students should be able to access files in a simple and organized way. Files sent from teachers to students should be classified under the class they were uploaded to and files that aren't from teachers should be classified together.

2.3.4.6. Physical Environment

The system is going to be deployed at Addis Ababa University, where the system is not required to withstand any special condition.

2.3.4.7. Security

- Students should be able to signup online by providing their ID, Name, Department, Year and Phone number. The given information should be checked against the student information list provided by the University. And if all the information provided by a student match's with a student record in the list, that isn't already taken, the student should be signed up and given an account.
- When inviting teachers via email the system should create accounts and temporary passwords for teachers before inviting them. They should set up their account and change their password when they first login.
- Teachers should login using the email they were invited with.

- Department heads should be given accounts by the administrators. They should change their password when they login for the first time.

2.2. System Model

2.2.1. Scenarios

2.2.1.1. Current (as-is) Scenarios

Table 2.1 Scenario - recoverForgottenPassword

Scenario name	recoverFogottenPassword
Participating actors	abebe:Instructor, addis:SystemAdminstrator
Flow of events	1. Abebe, an instructor at the Department of Computer Science, wants to use AAU Push to send PDF books to his student. 2. Abebe gets on the login page for Push and tries to login. After multiple failed attempts he realizes that he has forgotten his password. 3. He presses the Forgot Password link at the bottom of the form but it is no help because it is a broken link. 4. He soon notices at the bottom ‘Need help? Call Addis’ with a phone number listed. 5. He called Addis, a system administrator and explains the situation. 6. Addis logs into Push as an administrator and resets Abebe’s password to another password and sends the password to him. 7. Abebe uses the password to login and achieves his goal of sending the books.

Table 2.2 Scenario - collectAssignmentAnswers

Scenario name	collectAssignmentAnswers
Participating actors	leilt:Instructor
Flow of events	1. Leilt, an Instructor at the Department of Computer Science, has used AAU Push and now would like to give an assignment to the class for them to submit on Push. 2. She logs into the website to send an announcement about the assignment and attaches the problem statement. But realizes that Push has no feature that lets students upload their answers. 3. She then goes to create a Google Form that would be accepting files and shares that link in the announcement. So she collects the submission from students there.

Table 2.3 Scenario - sendAnnouncment

Scenario name	sendAnnouncement
Participating actors	temesgen:Instructor, robel:Student
Flow of events	1. Temesgen, an Instructor at the Department of Computer Science, wants to announce to his students that there is an upcoming test. 2. He logs into the AAU Push system and navigates to the Post area. 3. He types his message and sends it. 4. Robel, one of Temesgen's students, is at the computer labs that evening. He logs in to Push. 5. There he sees the announcement for an upcoming test and starts to prepare accordingly.

Table 2.4 Scenario - sendWrongAnnouncement

Scenario name	sendWrongAnnouncement
Participating actors	temesgen:Instructor, robel:Student, addis:SystemAdminstrator
Flow of events	1. Temesgen, an Instructor at the Department of Computer Science, wants to announce to his students that there is an upcoming test. 2. He logs into the AAU Push system and navigates to the Post area. 3. As he types his message, he thinks to add a helpful video from YouTube. He adds the link and sends. 4. Robel, one of Temesgen's students, is at the computer labs that evening. He logs in to Push. 5. There he sees the announcement for an upcoming test with then link. 6. But when he clicked on the link, it opened Zeritu's music video. 7. Robel immediately tells Temesgen that he shared the wrong link. 8. Temesgen realizes the mistake but feels helpless as there is no way to delete or edit the posted announcement. 9. He then calls Addis, a system administrator and haves his remove the post from the database. 10. Temesgen then carefully posts again the announcement with the correct link.

2.2.1.2.Visionary Scenarios

Table 2.5 Scenario - addInstructorToPush

Scenario name	addInstructorToPush
Participating actors	tamrat:DepartmentHead, selam:Instructor
Flow of events	1. At the Department of Computer Science, Selam, a new instructor, has just joined as a teaching staff member. 2. Dr. Tamrat, the head of the department, wants to add Selam in to the AAU Push system so that she will be able to use it in her teaching. 3. Dr. Tamrat is recognized as department head by the Push system and he logs in. 4. He types in the email address of Selam and sends her an invite. 5. Selam receives an invitation email with username and temporary password. 6. Selam uses the invitation email sent to her to register to the website and setup her account. She is now able to use Push.

Table 2.6 Scenario - marthaAsksQuestion

Scenario name	marthaAsksQuestion
Participating actors	martha:Student, abebe:Instructor
Flow of events	1. Martha, a second year computer science student, wants to ask her teacher Abebe why the program she wrote is not working. 2. Abebe teaches her class and has setup a class discussion forum to talk about problems they might be having. 3. She logs into Push and opens the class forum and posts in there her question and attaches her program file to it. 4. Abebe reviews the file and gives her a reply the next day.

Table 2.7 Scenario - callDepartmentMeeting

Scenario name	callDepartmentMeeting
Participating actors	tamrat:DepartmentHead, StaffMembers
Flow of events	1. Dr. Tamrat, the head of the Department of Computer Science, would like to call all staff members to a Department Faculty Meeting. 2. He logins to Push and opens the Department Group Chat area. 3. There he sends the message with the agenda for the meeting attached in a PDF file. 4. All staff members see his message and reply to confirm the date.

Table 2.1 to 2.7 describes real-world examples of how one or more actors interact with the system. Some scenarios have been omitted due to similarity, since they follow a similar path and only differ on the action they perform, and for brevity of the document.

2.2.2. Actors

Table 2.8 Scenario - List of Actors

Actor	Description
Student	A person who is studying at the college
Instructor	A member of staff of a department that teaches at least once course to a section
DepartmentHead	An Instructor in charge of leading a department, its program and staff members
CollegeDean	An Instructor assigned to head the college
RegistrarOfficer	A person holding a position at the Registrar Office

Table 2.8 lists the actors that were discovered during the requirement gathering.

2.2.3. Use Case Model

2.2.3.1. Use Case Diagram

Figure 2.1 Use Case Diagram - AAU Push. Note: to reduce complexity and readability, notice that each actor is repeated as needed.

2.2.3.2. Use Case Descriptions

Table 2.9 Use Case - InviteInstructor

Use case name	InviteInstructor
Participating actors	Initiated by DepartmentHead Communicates with Instructor
Entry condition	The DepartmentHead is logged in the website
Flow of events	1. The DepartmentHead clicks on ‘Manage Department Data’ from the Portal. 2. Push presents a form presenting two options. First is a file upload button that accepts an Excel file. And second is a short-text area field titled ‘Invite a Staff Member’ to write the email of the instructor. 3. The DepartmentHead uploads an Excel file containing the emails of the Instructors and clicks ‘Send Invite’ button. 4. Push creates a user objects based on each email provided. Then sends a username and temporary password to each email address. 5. An invited Instructor opens the email that presents their email for login, temporary password, and a button titled ‘Setup Account Now’ which redirect to login page. 6. Push presents a login form requesting for email and password for the Instructor. 7. The Instructor fills the credentials provided in the email and is logged in. 8. Push then presents a form requesting: Title (Prof., Dr., Mr…), First Name, Last Name, Email, New Password and Confirm New Password. 9. The Instructor completes this form and clicks on ‘Complete Setup’.

Exit condition	The DepartmentHead has received a confirmation of the invitations that are sent and of those that are already existent in the system. The Instructor is forwarded to the Instructor Portal.
Alternative Flow	If the DepartmentHead chooses to invite one Instructor, he/she writes the email of the Instructor into the text field and clicks on 'Send Invite'.

Table 2.10 Use Case - SignUp

Use case name	SignUp
Participating actors	Initiated by Student
Entry condition	The Student is on the sign up page of Push
Flow of events	1. Push presents a sign up form. 2. The Student fills the sign up form on the page by inputting first name, last name, ID and Department. Then clicks on 'Sign Up'. 3. Push then displays a form asking for year, section, password, phone number and email. 4. The Student picks their year and section from the list of available options in the drop-downs. Enters their preferred phone, email and password. Then clicks on 'Complete Sign Up'. 5. Push registers the user.
Exit condition	The user receives a confirmation that a successful signup is done and is forwarded to the Student Portal.
Quality requirements	

Table 2.11 Use Case - Login

Use case name	Login
Participating actors	Initiated by any user
Entry condition	The user has opened the login page of Push
Flow of events	1. Push presents a form with two tabs ‘Student' and ‘Staff’. Under the ‘Staff’ tab are two fields, ‘Email’ and ‘Password’ and a button ‘Login’. Under ‘Student’ tab are two fields, ‘Username’ and ‘Password’. 2. If the user is a Student, user inputs their ID and password and clicks on ‘Login’. If the user is an Instructor, DepartmentHead, CollegeDean or RegistrarOfficer, user inputs their email and password and clicks on ‘Login’. 3. Push verifies that the login credentials are correct.
Exit condition	The user is advanced to the Portal or Dashboard
Quality requirements	
Alternative Flow	3. If the login credentials fail verification, the user is given a notice on the login page that the email and/or password is wrong. 4. The user can then click on ‘ForgotPassword’ button to initiate the ForgotPassword use case.

Table 2.12 Use Case - ForgotPassword

Use case name	ForgotPassword
Participating actors	Initiated by any user
Entry condition	The user is on the login page.
Flow of events	1. The user clicks on 'Forgot Password' link below the login form. 2. Push presents a form with 'Email' field and a button 'Reset Password'. 3. The user inputs their email and clicks on 'Reset Password'. 4. Push sends a password reset link to the email provided. 5. The user opens the link from their email. 6. Push presents a webpage with a form containing two fields 'New Password' and 'Confirm New Password' and a button 'Reset Password'. 7. The user fills the form and clicks on 'Reset Password'.
Exit condition	The user receives a confirmation and redirected to the login page.
Quality requirements	

Table 2.13 Use Case - Logout

Use case name	Logout
Participating actors	Initiated by any user
Flow of events	1. At the Portal or Dashboard, the user clicks on 'Logout' from the menu presented. 2. Push log out the user.
Entry condition	The user is logged in the system
Exit condition	The user is redirected to the login page
Quality requirements	

Table 2.14 Use Case - SendPost

Use case name	SendPost
Participating actors	Initiated by Instructor, DepartmentHead, RegistrarOfficer, CollegeDean
Flow of events	1. At the Portal, the user clicks on 'Send Post' tab from the menu. 2. Push presents a form requiring post text, three optional file fields, two images and which class to send it to. And a button titled 'Send Post'. 3. The user fills the form and clicks on 'Send Post'. 4. Push records the post and sends it to students with a notification.
Entry condition	The user is logged in the system
Exit condition	The user is given confirmation that the post is successfully sent.
Quality requirements	

Table 2.15 Use Case - ViewPost

Use case name	ViewPost
Participating actors	Initiated by Student
Entry condition	Student is logged in and at the Dashboard
Flow of events	1. The Student clicks on 'Your Push' from the menu available. 2. Push displays all posts from Instructors sorted by date with the latest on top. Each post shows the title and name of the Instructor that sent it, the text content of the post, files and images attached, and the date and time of the post.
Exit condition	
Alternative Flow	The Student clicks on 'PushBoard' to view posts from the DepartmentHead, CollegeDean and RegistrarOfficer

Table 2.16 Use Case - ClickReadPost

Use case name	ClickReadPost
Participating actors	Initiated by Student
Entry condition	Student is logged in and at the Dashboard
Flow of events	1. The Student clicks on 'Your Push' from the menu available. 2. Push displays all posts from Instructors sorted by date with the latest on top. Each announcement has a button 'Read' used to confirm that the Student has read it. 3. The Student clicks on 'Read'. 4. The button is removed from the post. Push updates Tracker for the Instructor that posted it.
Exit condition	
Alternative Flow	

Table 2.17 Use Case - TrackPost

Use case name	TrackPost
Participating actors	Initiated by Instructor, DepartmentHead, CollegeDean or RegistrarOfficer
Entry condition	The user is logged in the system
Flow of events	1. The user clicks on 'Tracker' tab in the Portal. 2. Push presents a list of all the posts sent by the user. For each post, 'Delivered To' and 'Read By' fields indicate the number of students the announcement is sent to and number of students that have confirmed to have read it respectively. For any files attached on the post, the 'Downloads' label shows how many times the files have been downloaded. 'Edit Post' and 'Delete Post' buttons are also available. 3. The user views the statistics provided.
Exit condition	

Quality requirements	If the user clicks on 'Edit Post' button for a specific post, the EditPost use case is initiated. If the user clicks on 'Delete Post' button for a specific post, the DeletePost use case is initiated.

Table 2.18 Use Case - EditPost

Use case name	EditPost
Participating actors	Initiated by Instructor, DepartmentHead, CollegeDean or RegistrarOfficer
Entry condition	The user is logged in and has initiated TrackPost use case
Flow of events	1. The user clicks on the 'Edit Post' for a post from the list provided. 2. Push makes the text for the post an editable form. 3. The user makes the edits required to the text and clicks on 'Update Post' button. 4. Push updates the post and notifies the recipients of changes.
Exit condition	The user gets a confirmation that announcement has been edited.
Quality requirements	

Table 2.19 Use Case - DeletePost

Use case name	DeletePost
Participating actors	Initiated by Instructor, DepartmentHead, CollegeDean or RegistrarOfficer
Entry condition	The user is logged in and has initiated TrackPost use case
Flow of events	1. The user clicks on the 'Delete Post' for a post from the list provided. 2. Push prompts the user for a confirmation asking 'Are you sure you want to delete this post?' 3. The user confirms by clicking on 'Yes' 4. Push deletes the post.

Exit condition	The user gets a confirmation that announcement has been edited.
Quality requirements	

Table 2.20 Use Case - AddClass

Use case name	AddClass
Participating actors	Initiated by Instructor or Student
Entry condition	The user is logged in the system
Flow of events	1. The user clicks on 'Class List' tab in the Student Dashboard or Instructor Portal. 2. Push presents a form that lets the user pick courses by specifying year and section. 3. From the form the user ticks on the courses desired and clicks on 'Add Class'. 4. Push updates the list of classes the user is engaged in.
Exit condition	The user receives a confirmation that the classes are added
Quality requirements	If the user is an Instructor, Push adds the list of classes the Instructor is teaching and makes them available for sending posts. If the user is a Students, Push adds the class for the Student and he/she will receive posts from that class.

Table 2.21 Use Case - DropClass

Use case name	DropClass
Participating actors	Initiated by Instructor or Student
Entry condition	The user is logged in the system

Flow of events	1. The user clicks on 'Class List' tab in the Student Dashboard or Instructor Portal. 2. Push presents a table that displays the list of classes the user is currently engaged in. 3. From the table the user clicks 'Drop Class' on the course to be removed. 4. Push updates the list of classes the user is engaged in.
Exit condition	The user receives a confirmation that the classes are dropped.
Quality requirements	If the user is an Instructor, Push removes the class from the list of classes the Instructor is teaching. If the user is a Student, he/ she will stop receiving posts from that class.

Table 2.22 Use Case - GiveAssignment

Use case name	GiveAssignment
Participating actors	Initiated by Instructor
Entry condition	The Instructor is logged in the system
Flow of events	1. At the Portal, the Instructor clicks on 'Assignments' from the menu presented. 2. The tab displays a list of assignments that have been given by the Instructor and a button, 'New Assignment'. 3. To give a new assignment the Instructor clicks on 'New Assignment' button. 4. Push opens a form requiring the name, assignment definition file, Reminder date (day, month and year), which class it is for and comments related to the assignment. 5. After filling the form, the Instructor clicks on 'Give Assignment' button. 6. Push saves the data and notifies the recipients.
Exit condition	The Instructor gets confirmation that the assignment is successfully given.

Quality requirements	

Table 2.23 Use Case - SubmitAssignment

Use case name	SubmitAssignment
Participating actors	Initiated by Student
Flow of events	1. Student opens "Assignment" tab from the student dashboard or clicks "Submit" on one of the assignment Reminders available 2. Push list all assignments concerning the student with a form containing a "comment" text field and "assignment" file field for the each assignment he/she didn't submit 3. The student fills the form of the assignment he/she wants to submit and clicks "Submit" 4. Push stores the assignment data under the given assignment
Entry condition	Student must be logged in.
Exit condition	The student gets confirmation that the submission was successful .
Quality requirements	

Table 2.24 Use Case - DownloadAssignmentSubmission

Use case name	DownloadAssignmentSubmission
Participating actors	Initiated by Instructor
Entry condition	The Instructor is logged in the system

Flow of events	1. At the Portal, the Instructor clicks on 'Assignments' from the menu presented on the left. 2. The tab displays a list of assignments that have been given by the Instructor. 3. The Instructor clicks on 'View Submissions' button for any of the assignments given. 4. The button opens a list of submissions each one detailing the name of student, submitted assignment file, date of submission and comments left by the Student. 5. From this list, the Instructor clicks on 'Download File' button for each submission to review the Student's work.
Exit condition	The Instructor downloads the files successfully.
Quality requirements	

Table 2.25 Use Case - CreateForum

Use case name	CreateForum
Participating actors	Initiated by any user
Flow of events	1. At the Portal or Dashboard, the user clicks on 'Forums' from the menu presented. 2. The tab displays a list of active forums. 3. To give a new one the user clicks on 'Create Forum' button. 4. Push opens a form requiring the name of the forum, description and forum privacy (open or closed). 5. After filling the form, the user clicks on 'Create Forum' button.
Entry condition	The user has logged into the system
Exit condition	Push opens the forum
Quality requirements	If the forum privacy is closed then the user provides a forum code or a user attribute (year and/or department and/or section) and value that can be used to specify users that can join.

Table 2.26 Use Case - SearchForum

Use case name	SearchForum
Participating actors	Initiated by any user
Flow of events	1. At the Portal or Dashboard, the user clicks on 'Forums' from the menu presented. 2. The tab displays a list of forums the user has joined. And a button titled 'Browse More Forums'. 3. The user clicks on 'Browse More Functions'. 4. Push presents a list of forums with title, description and forum privacy (open or closed). And a search bar on top with a text field. 5. The user types the name of the forum required. 6. As the user types, Push lists the related forums.
Entry condition	The user is logged in the system
Exit condition	
Quality requirements	

Table 2.27 Use Case - OpenForum

Use case name	OpenForum
Participating actors	Initiated by any user
Flow of events	1. At the Portal or Dashboard, the Instructor clicks on 'Forums' from the menu presented. 2. The tab displays a list of forums the user has joined. 3. The user opens the forum by clicking on its name. 4. Push displays the forum
Entry condition	The user is logged in the system
Exit condition	The user gets confirmation that the Forum is successfully destroyed.
Quality requirements	If the user is the creator of the forum a button titled 'Destroy Forum' is presented in forum.

Table 2.28 Use Case - JoinForum

Use case name	JoinForum
Participating actors	Initiated by any user
Flow of events	1. At the Portal or Dashboard, the user clicks on 'Forums' from the menu presented. 2. The tab displays a list of forums the user has joined. And a button titled 'Browse More Forums'. 3. The user clicks on 'Browse More Functions' 4. Push presents a list of forums with title, description and forum privacy (open or closed). 5. The user clicks on the title of an open forum. 6. The forum opens to show latest messages. A button titled 'Join Forum' 7. The user clicks on 'Join Forum'.
Entry condition	The user is logged in the system
Exit condition	The user gets confirmation that the Forum is successfully joined and opens the forum
Quality requirements	5. The user clicks on the title of a forum closed with password. 6. Push opens a dialog requesting for the password and a button titled 'Join Forum'. 7. The user enters the password and clicks on 'Join Forum'.

Table 2.29 Use Case - DestroyForum

Use case name	DestroyForum
Participating actors	Initiated by any user
Entry condition	The user has opened the forum by use case OpenForum

Flow of events	1. The forum is displayed with the latest massages and presented with a button titled 'Destroy Forum'. 2. The user clicks 'Destroy Forum'. 3. Push removes the forum instance
Exit condition	The user gets confirmation that the Forum is successfully destroyed.
Quality requirements	If the user is the creator of the forum a button titled 'Destroy Forum' is presented in forum.

Table 2.30 Use Case - UpdateAccountInformation

Use case name	UpdateAccountInformation
Participating actors	Initiated by any user
Entry condition	User is logged in
Flow of events	1. From the Portal or Dashboard, the user clicks on 'Account'. 2. Push presents a form pre-filled. If the user is a Student, the form has first name, last name, ID and Department, year, section, phone number and email. If the user is an Instructor, DepartmentHead, CollegeDean or RegistrarOfficer, the form has Title (Prof., Dr., Mr…), First Name, Last Name and Email. At the bottom of the form is a password field and a button titled 'Save Changes'. 3. The user can then change the values of the form, enter their password and click 'Save Changes'. 4. Push updates the changes.
Exit condition	The user is given confirmation that the changes have been successfully made.
Alternative Flow	If the given password is wrong, an error message is displayed and user is asked to try again.

Table 2.31 Use Case - SetReminder

Use case name	SetReminder
Participating actors	Initiated by Instructor
Entry condition	The Instructor is logged in the system

Flow of events	1. At the Portal, Push presents the list of Reminders set by the Instructor and a button titled 'Set Reminder' on the right side. 2. The Instructor clicks on 'Set Reminder' button. 3. Push presents a pop-up form requiring the name, Reminder date (day, month and year), which class it is for and comments related to the Reminder. 4. After filling the form, the Instructor clicks on 'Set Reminder' button. 5. Push saves the data and notifies Students on their mobile application.
Exit condition	The Instructor gets confirmation that the Reminder is successfully set.
Quality requirements	

Table 2.32 Use Case - ViewReminder

Use case name	ViewReminder
Participating actors	Initiated by Student
Entry condition	The user is logged in the system
Flow of events	1. At the Dashboard, Push presents the Reminders on the right side.
Exit condition	
Quality requirements	
Alternative Flow	

Table 2.33 Use Case - EditReminder

Use case name	EditReminder
Participating actors	Initiated by Instructor
Entry condition	The user is logged in the system

Flow of events	1. At the Portal, the user clicks on 'Tracker' from the menu. 2. Push presents the list of the names of the Reminders set by the Instructor. With buttons titled 'Edit' and 'Delete'. 3. The user clicks 'Edit' on the Reminder to be edited. 4. Push expands the Reminder to show a pre-filled editable form with the text of the Reminder, the date (day, month, year) with a button titled 'Update Reminder'. 5. The user clicks on 'Update Reminder'. 6. Push records the changes.
Exit condition	The user is given confirmation that the Reminder is edited
Quality requirements	
Alternative Flow	

Table 2.34 Use Case - DeleteReminder

Use case name	DeleteReminder
Participating actors	Initiated by Instructor
Entry condition	The user is logged in the system
Flow of events	1. At the Portal, the user clicks on 'Tracker' from the menu. 2. Push presents the list of the names of the Reminders set by the Instructor. With buttons titled 'Edit' and 'Delete'. 3. The user clicks 'Delete' on the Reminder to be deleted. 4. Push prompts the user for a confirmation asking 'Are you sure you want to delete this post?' 5. The user confirms by clicking on 'Yes' 6. Push deletes the Reminder.
Exit condition	The user is given confirmation that the Reminder is deleted
Quality requirements	
Alternative Flow	

Table 2.35 Use Case - ViewDepartmentChat

Use case name	ViewDepartmentChat
Participating actors	Initiated by Instructor, DepartmentHead
Flow of events	1. At the Portal, the user clicks on 'Department Chat' tab from the menu. 2. Push presents the messages in chat ordered by date.
Entry condition	The user is logged in the system
Exit condition	
Quality requirements	

Table 2.36 Use Case - MessageDepartmentChat

Use case name	MessageDepartmentChat
Participating actors	Initiated by Instructor, DepartmentHead
Entry condition	The user is logged in the system and is on the ViewDepartmentChat use case
Flow of events	1. Push presents a form at the bottom of the chat. The form has a text area, a file field and an image field with a button titled 'Send Message'. 2. The user fills the form and clicks on 'Send'. 3. Push updates the chat and adds the message as the latest message in the group chat.
Exit condition	
Quality requirements	

Table 2.37 Use Case - DeanMessageDepartment

Use case name	DeanMessageDepartment
Participating actors	Initiated by CollegeDean
Entry condition	The user is logged in the system

Flow of events	1. At the Portal, the user clicks on 'Message Departments' from the tab presented. 2. Push presents a form requiring post text, three optional file fields, two images and which Department to send it to from the list by ticking checkboxes. And a button titled 'Send Post'. 3. The user fills the form and clicks on 'Send Post'. 4. Push adds the message into the Department Chat areas of the departments receiving the messages.
Exit condition	The user is given a confirmation that the message is successfully sent
Quality requirements	

Table 2.38 Use Case - MessageForum

Use case name	MessageForum
Participating actors	Initiated by any user
Flow of events	1. The open forum displays the latest messages and a text area and a file attachment field with a send button. 2. The user writes the message in the text area, optionally attaches a file and hits 'Send'. 3. Push records the post and notifies the recipients
Entry condition	The user has opened the forum by use case OpenForum
Exit condition	The user gets confirmation that the message is sent.
Quality requirements	

Table 2.9 to 2.38 are the descriptions of the use care diagram shown in Figure 2.1. These show the full scope of the system to be developed.

The next section shows sequence diagrams for chosen use cases. Some have been omitted due to similarity, since the follow similar path and only differ on the action they perform, and brevity of the document.

2.2.4.Sequence Diagram

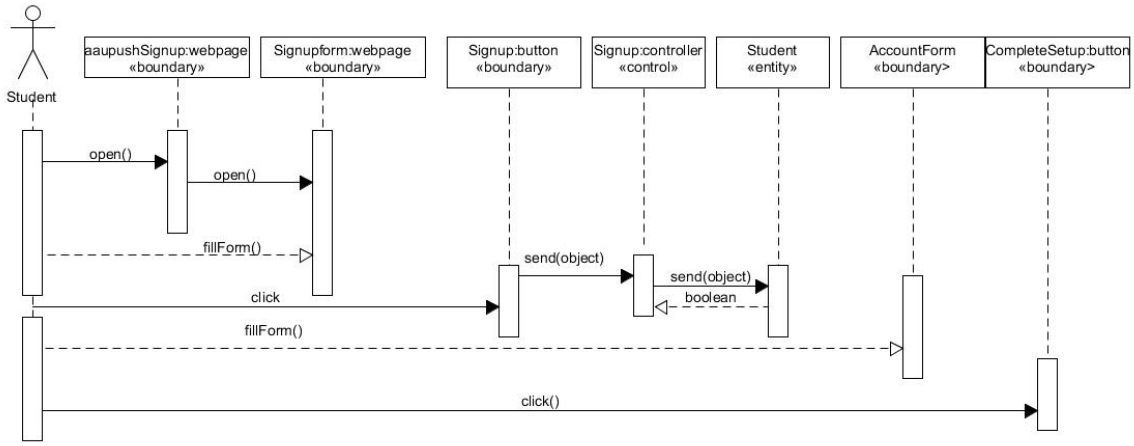

Figure 2.2 Sequence Diagram - SignUp

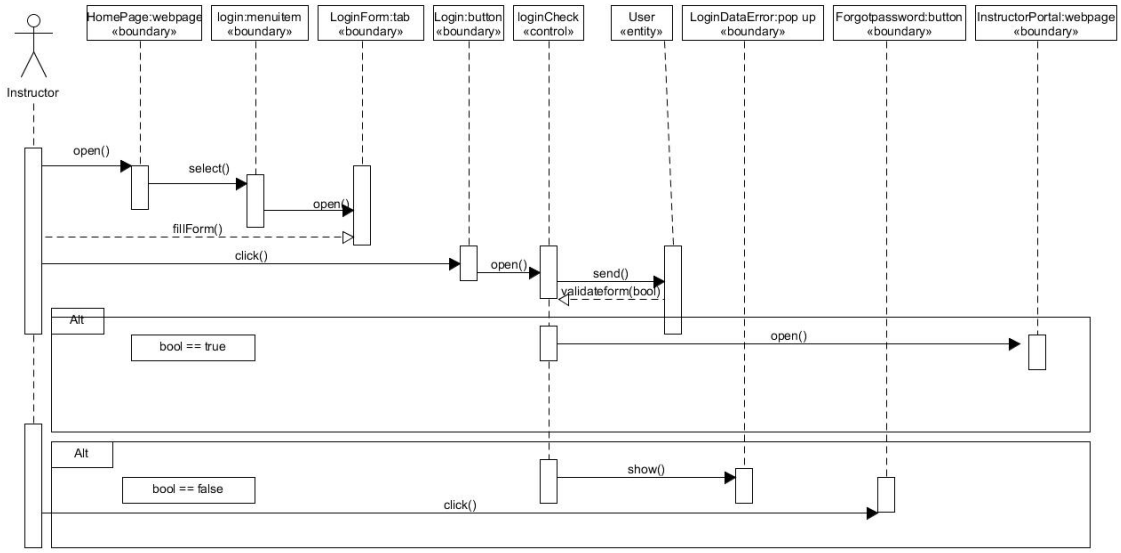

Figure 2.3 Sequence Diagram - Login

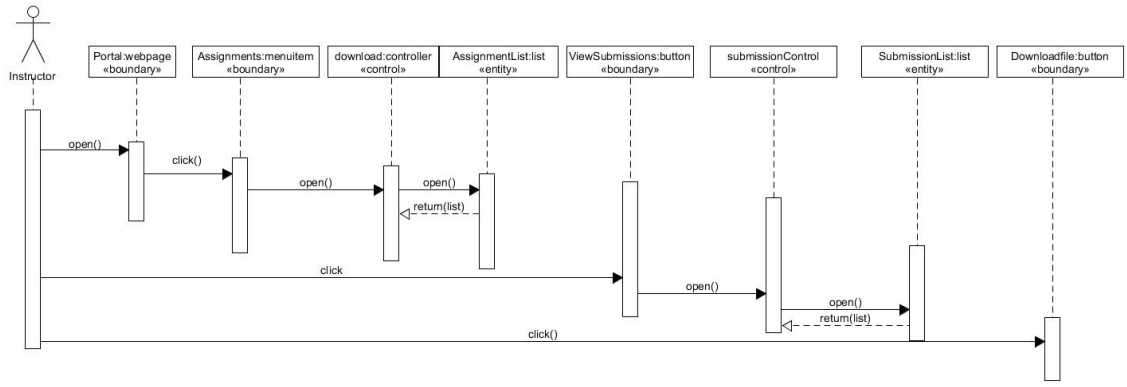

Figure 2.4 Sequence Diagram - DownloadAssignmentSubmission

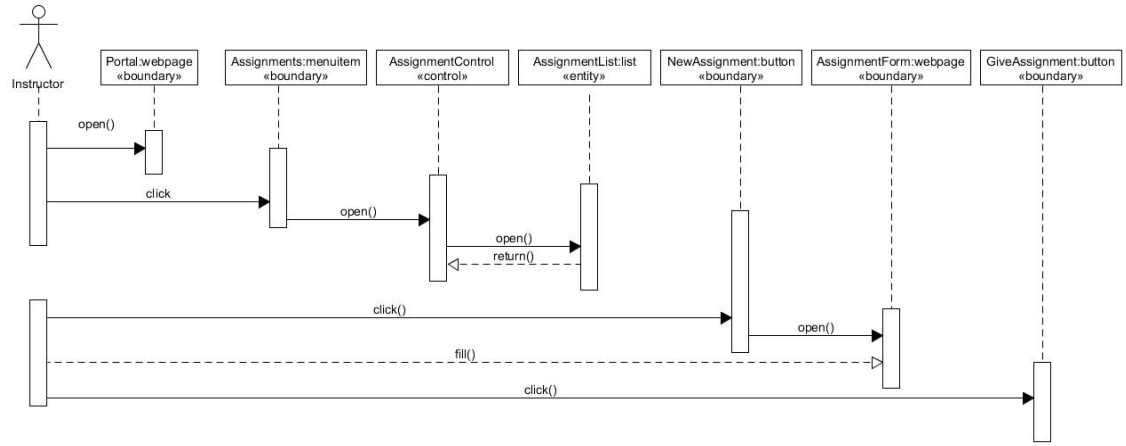

Figure 2.5 Sequence Diagram - GiveAssignment

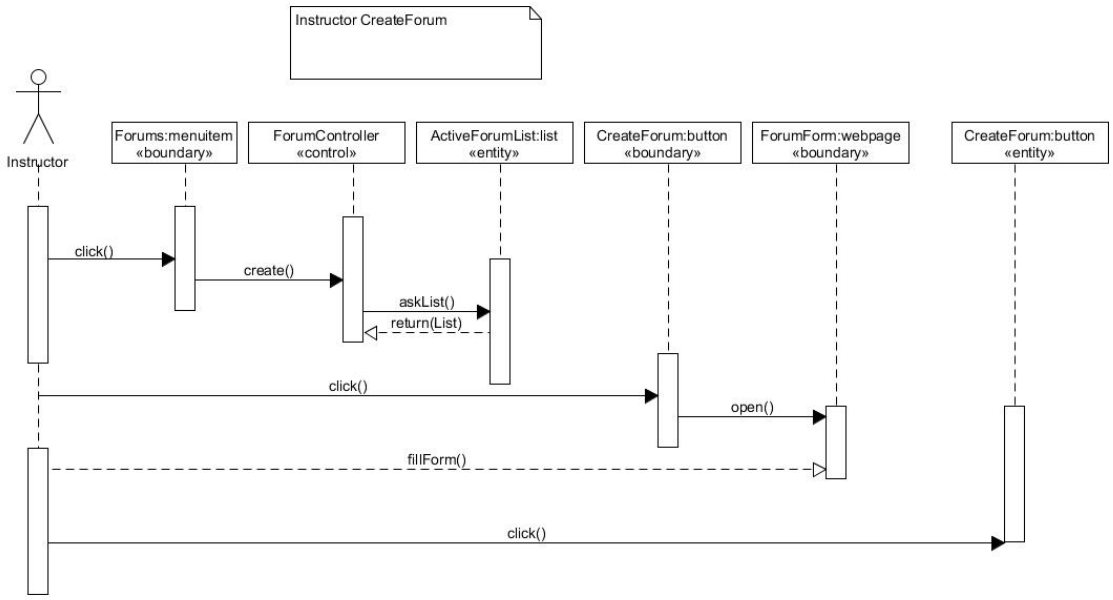

Figure 2.6 Sequence Diagram - Create Forum

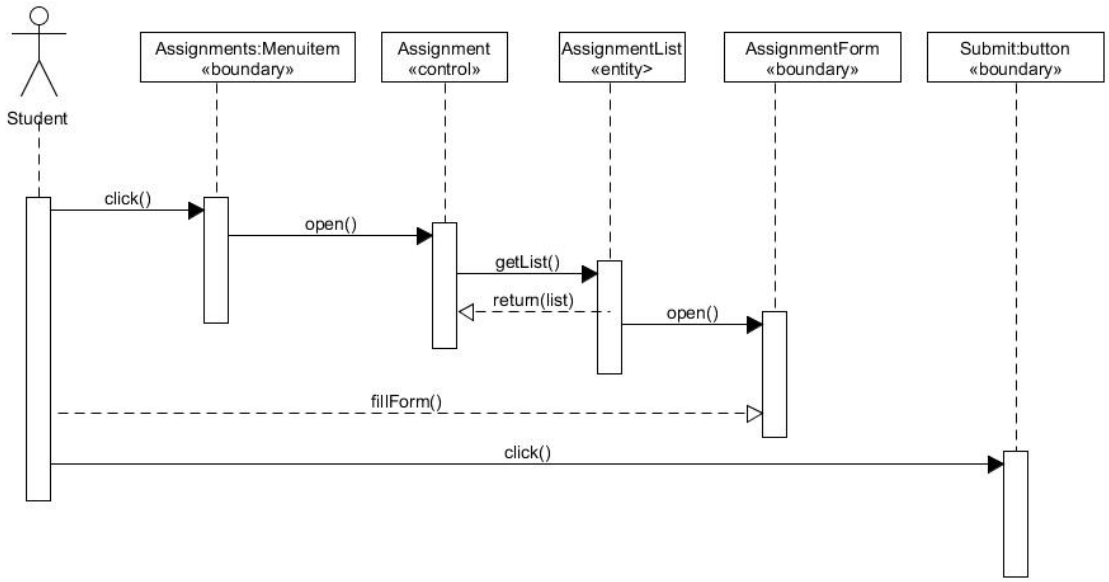

Figure 2.7 Sequence Diagram - Submit Assignment

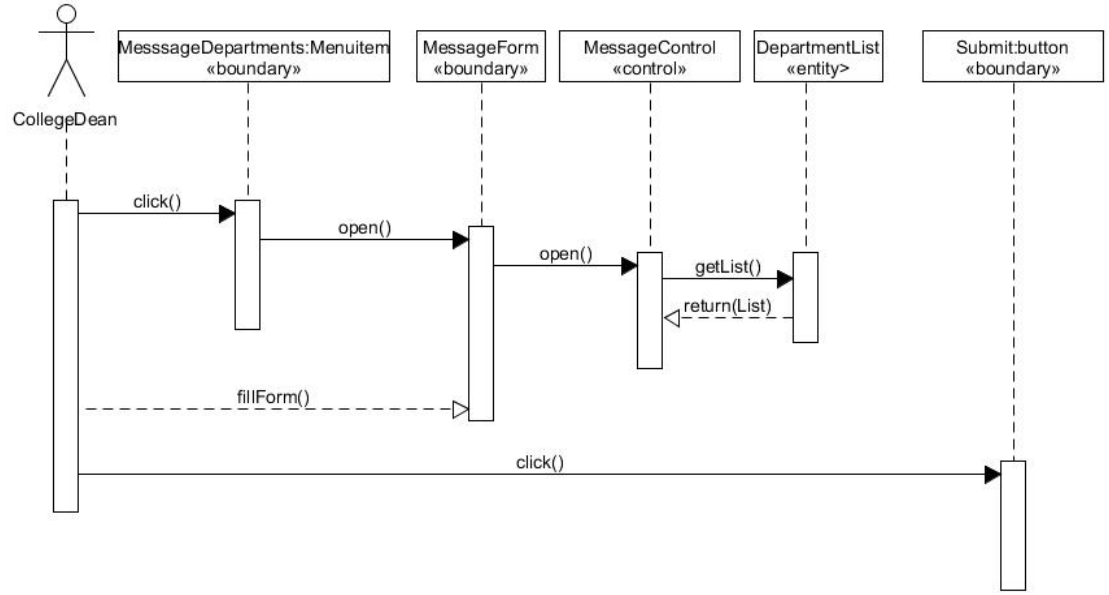

Figure 2.8 Sequence Diagram - DeanMessageDepartment

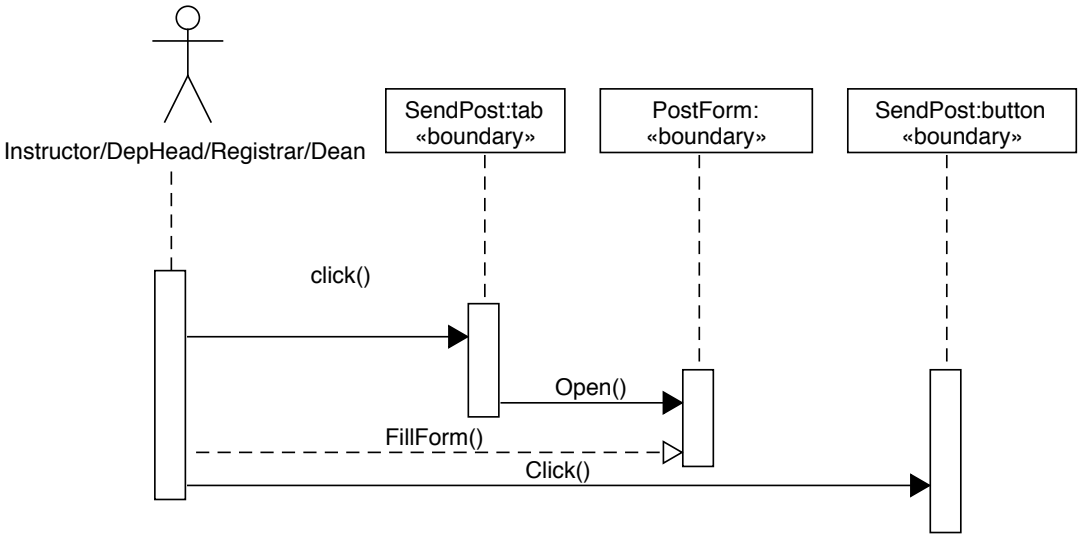

Figure 2.9 Sequence Diagram - SendPost

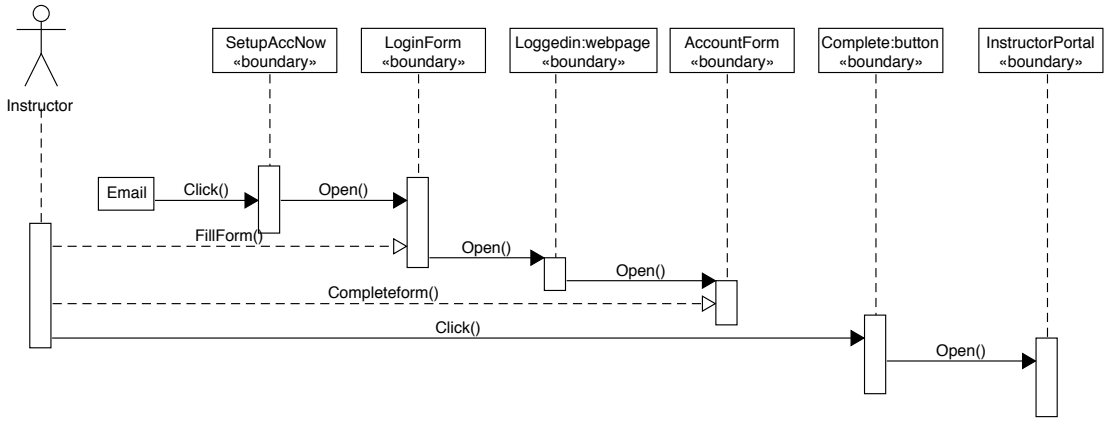

Figure 2.10 Sequence Diagram - InviteInstructor

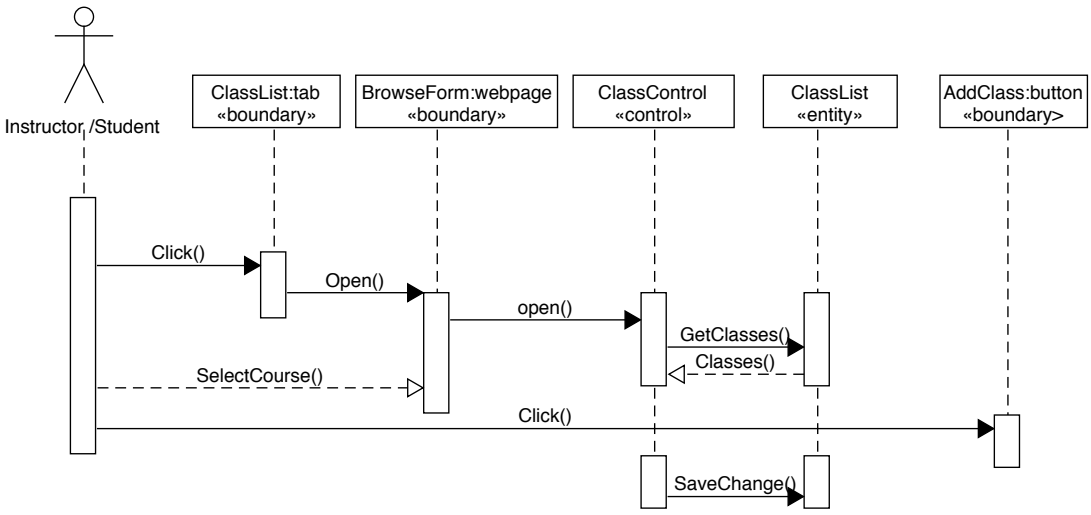

Figure 2.11 Sequence Diagram - InviteInstructor

2.2.5. Activity Diagram

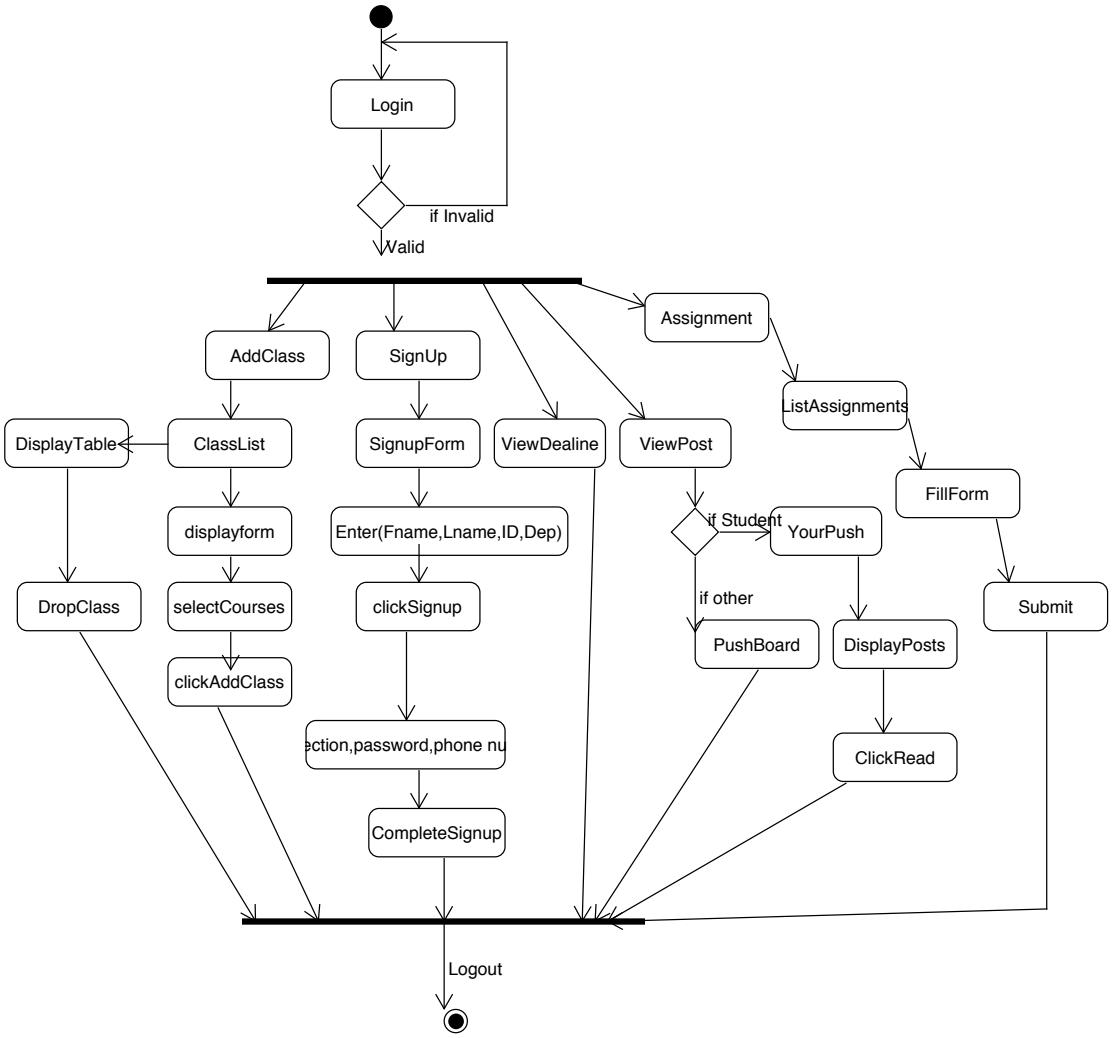

Figure 2.12 Activity Diagram - Student

The above Figure 2.9 shows the actives a student undertakes on the system.

2.2.6. State Chart Diagram

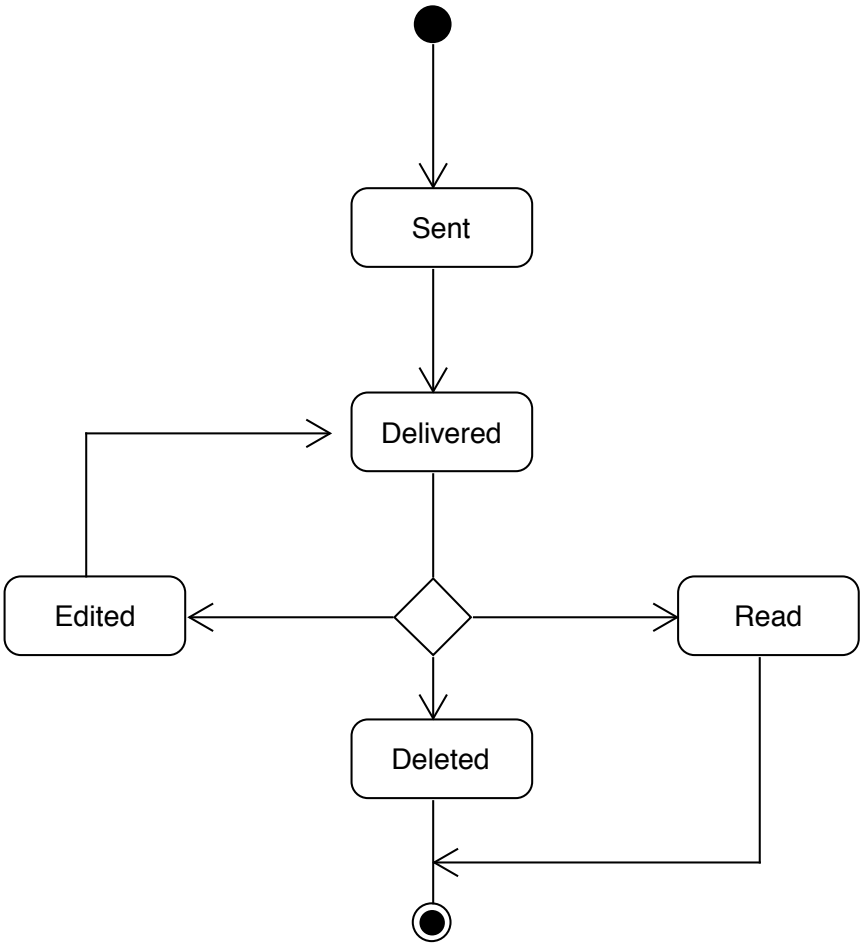

Figure 2.13 State Chart Diagram - Post

2.2.7. Object Model

2.2.7.1. Class Modeling

Figure 2.14 found on the next page shows the Class Diagram for AAU Push. Due to the complexity of the diagram, we have printed it on A3 paper.

Figure 2.14 Class Diagram - AAU Push

2.2.8. User Interface

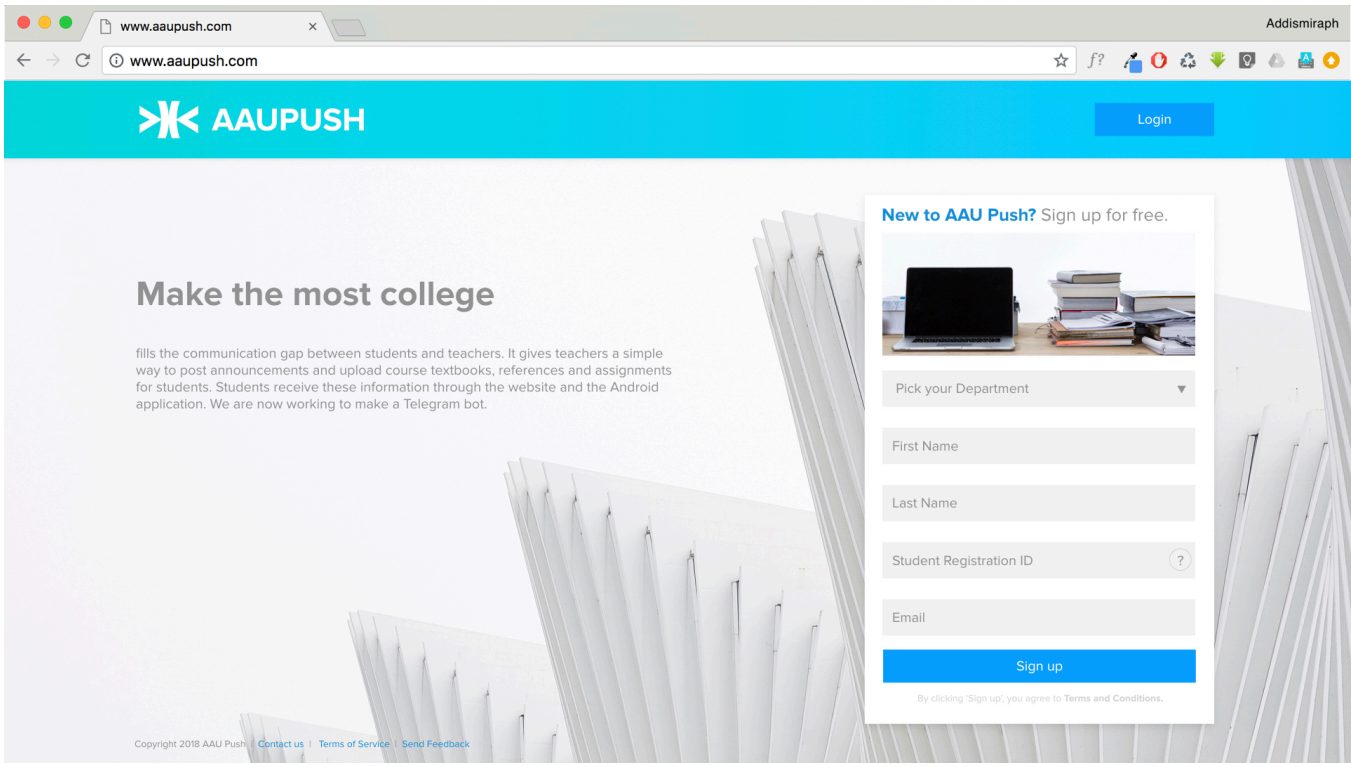

Figure 2.15 Landing Page Mockup

The mockup, Figure 2.15, was made in Keynote showing the home web page of the Push. Notice on the home page is the sign up form and the login button.

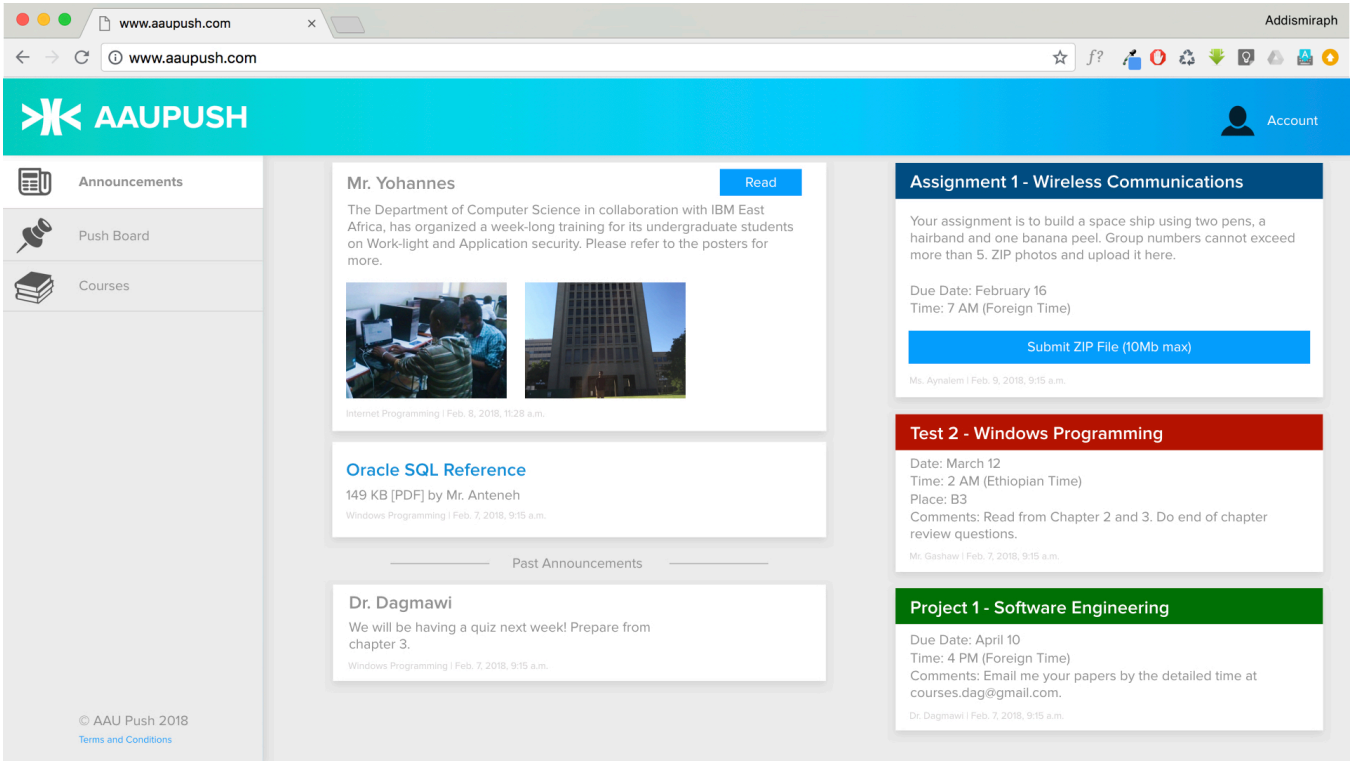

Figure 2.16 Student Dashboard Mockup

The mockup, Figure 2.16, was made in Keynote showing the web Student Dashboard area. This mockup shows multiple features:

1. The ‘Read’ button for a post described in ClickReadPost use case is seen.

2. The form to submit an assignment is seen on the right hand side.

3. Under the assignment form there are other Reminders for a test and project.

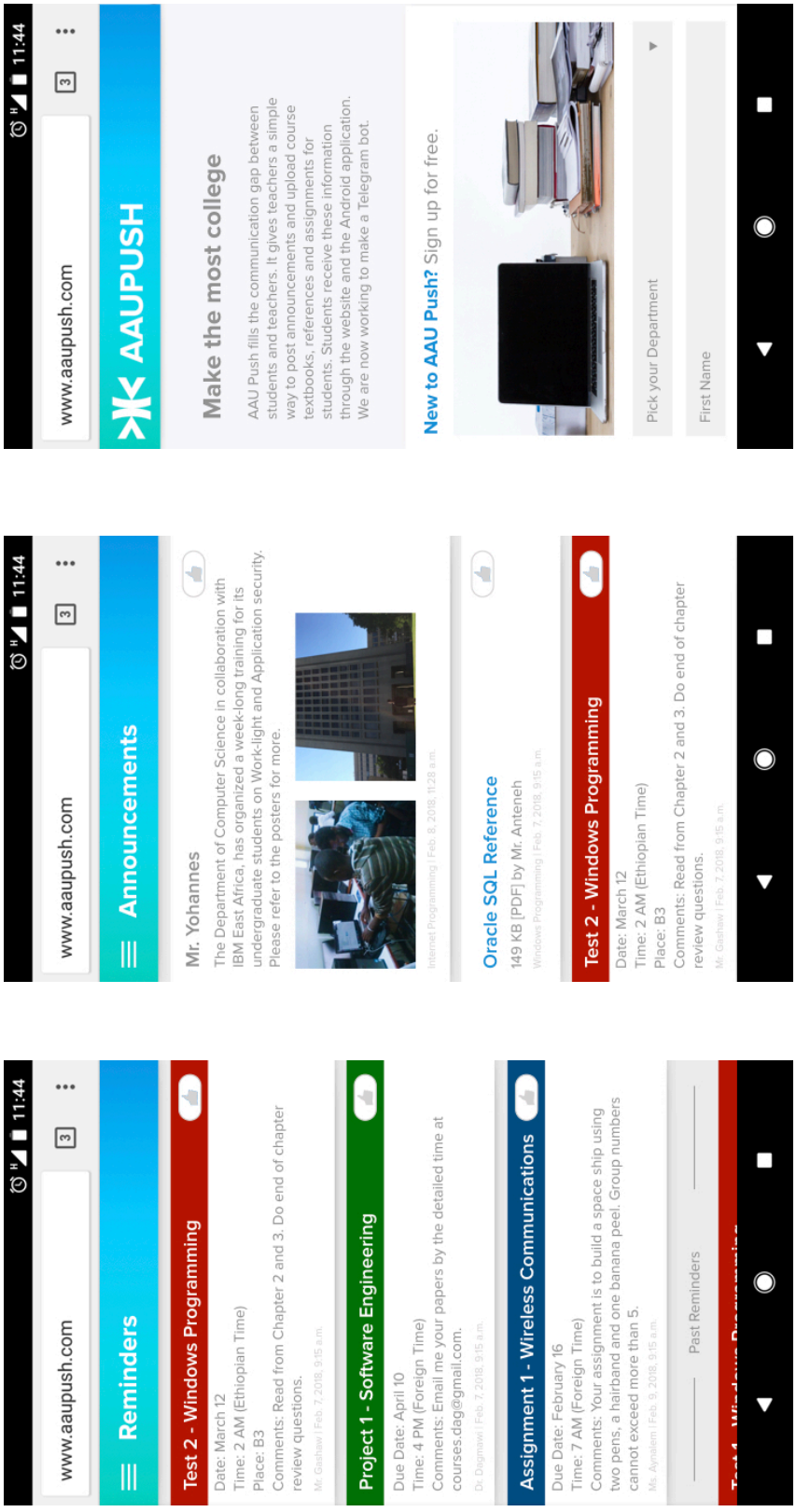

Figure 2.17 Mobile Responsive Interface

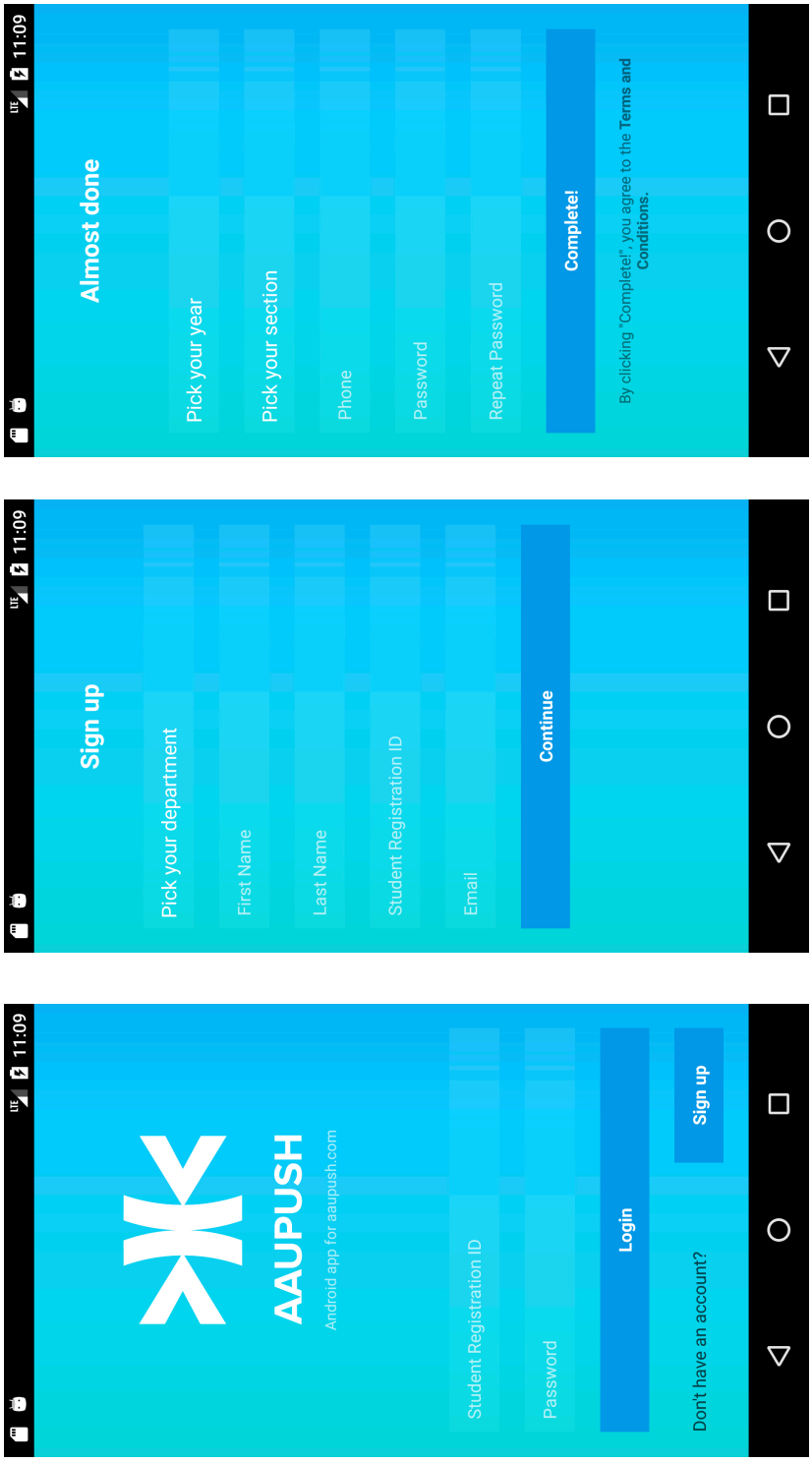

Figure 2.18 Android Mobile Application Interface

Figure 2.15 to 2.18 show proposed user interfaces for the system.

3. SYSTEM DESIGN

In this chapter we will discusses the internal structure of the system and its detailed architecture. We will cover the core parts of the system and its architecture with all the methodologies it is going to apply. System design deals with the functional design of the system and its internal structure from the subsystems and data management to the hardware and software mapping.

3.1. Introduction

The system design will include the overall view of the system from the functional viewpoint. The system is a web and mobile application that is concerned with making the teaching learning process easier.

3.2. Current Software Architecture

The current system of AAU Push is built using the Django framework. Django is an MVC. MVC stands for Model-View- Controller. The model stands for the data the system is going to operate. The View stands for the user interface that the actors of the system will interact with. The controller acts as a bridge between the model and the view. It will send and receive messages between the view and the model.

The current system uses a three-tiered architecture. The following diagram, Figure 3.1, shows how it works.

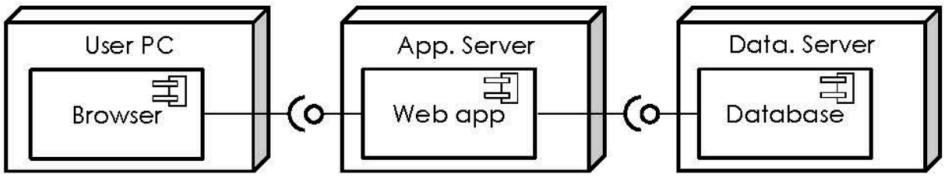

Figure 3.1 Current System Architecture

The user interacts with the web server that is currently hosted on pythonanywhere.com. That web server is responsible to communicate with the MySQL database.

3.3. Proposed Software Architecture

3.3.1. Overview

The proposed system will have a MVC architecture, the same as the current system, since we are using the Django framework. This architecture is highly used for building web applications.

This MVC architecture is chosen because it allows us to develop the system in a flexible and manageable way.

The system is composed of 12 subsystems. Each subsystem and its functionality is described in following subsections.

3.3.2. Subsystem Decomposition

The subsystem decomposition describes the assignment of functionalities and responsibilities to mutually related subsystems. Each subsystem has relationships with one or more subsystems. We have decomposed the system into three main components: the System User Interface, Application Logic and Data Storage.

The envisioned AAU Push system contains the following subsystems:

- UserInterface Subsystem
- AssignmentManagement Subsystem
- ReminderManagement Subsystem
- DepChatManagement Subsystem
- AccountManagement Subsystem
- ForumManagement Subsystem
- PostManagement Subsystem
- Notification Subsystem
- DatabaseInteraction Subsystem
- AddDropManagement Subsystem
- DataStorage Subsystem

Below there is Figure 3.2 that shows subsystem decomposition and Interactions between each subsystem of the new system. After the figure, each subsystem is described briefly.

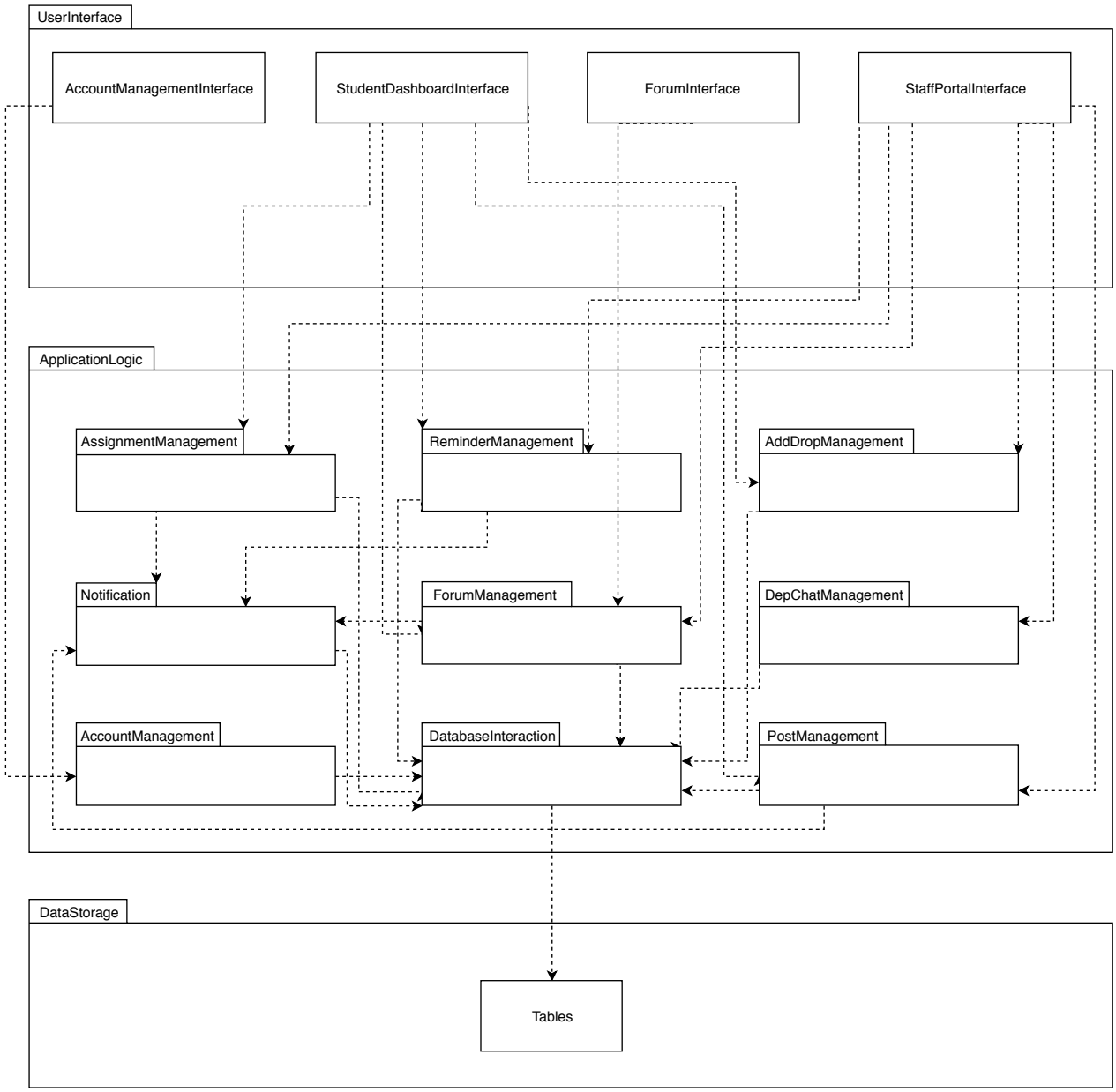

Figure 3.2 Subsystem Decomposition

UserInterface Subsystem

The SystemUserInterface subsystem is responsible for enabling users to interact with the system. It is the outer layer of the system that is viewed and manipulated by system users. It accepts user inputs and interact with other components of the system in order to present an output for the user. Figure 3.3 shows the subsystem. This subsystem is composed of the following classes:

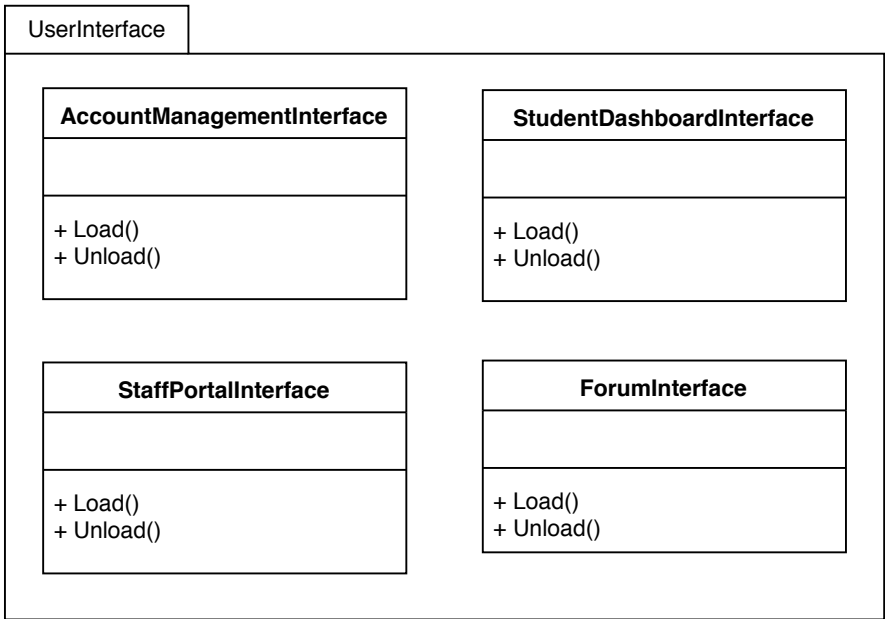

Figure 3.3 Subsystem - UserInterface

AccountManagementInterface: This class is the first point of interaction for the user with the system. It is responsible for enabling the user to login, signup, start password recovery process and update their account information. It interacts with the AccountManagement subsystem in order to execute the required functions.

StudentDashboardInterface: This class is responsible for the interaction between a student and the system. It lets the student to view posts from teacher and confirm having read it by clicking 'Read Post'. It lets students view and submit assignments. It interacts with the AddDropManagement subsystem to let students add and drop classes and with ReminderManagement to view reminders.

ForumInterface: Responsible for displaying all activities related to using the forum feature by using the ForumManagement subsystem. That is to search, join, create, destroy, view forums and send messages.

StaffPortalInterface: This class is responsible for displaying for Staff members, that is Instructor, DepartmentHead, CollegeDean and RegistrarOfficers. All of their functions will be visible for interaction. Activities like sending posts, editing posts, giving assignments, setting Reminders, tracking posts, adding and removing classes. It works by interacting various subsystems needed for each task.

AssignmentManagement Subsystem

The AssignmentManagement subsystem is responsible for managing tasks related to assignments on the platform. It is a application logic layer that makes sure business rules are applied. It interacts with the DatabaseInteraction class to access the database — to execute queries, send and retrieve data. Figure 3.4 shows the subsystem. It gives service to the UserInterface subsystem.

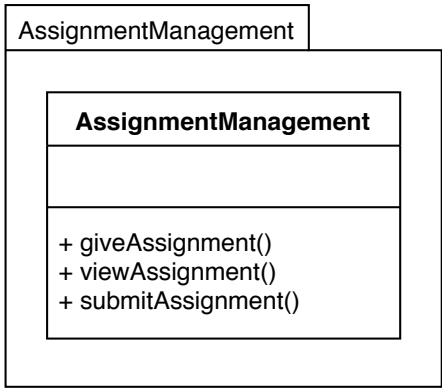

Figure 3.4 Subsystem - AssignmentManagement

ReminderManagement Subsystem

The ReminderManagement subsystem is responsible for handling logic involved in the setting, viewing, editing, and deleting of Reminders on AAU Push. It interacts with the DatabaseInteraction and Notification subsystems to handle data and notify students respectively. Figure 3.5 shows the subsystem.

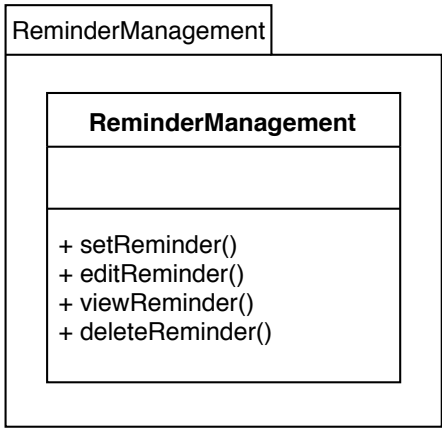

Figure 3.5 Subsystem - ReminderManagement

DepChatManagement Subsystem

This subsystem is responsible for handing tasks involved in the department chat. This is the group chat platform that is available for teachers in the Portal. This subsystem decides who views what and who can send messages where. Figure 3.6 shows the subsystem.

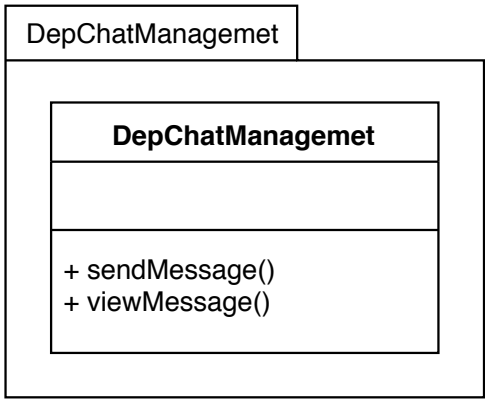

Figure 3.6 Subsystem - DepChatManagement

AccountManagement Subsystem

This is a subsystem that gives basic utility to the system. It handles logic for logging in or out, signing up, recovering password and update account information. It works with the DatabaseInteraction subsystem and Django's built-in authentication systems. Figure 3.7 shows the subsystem.

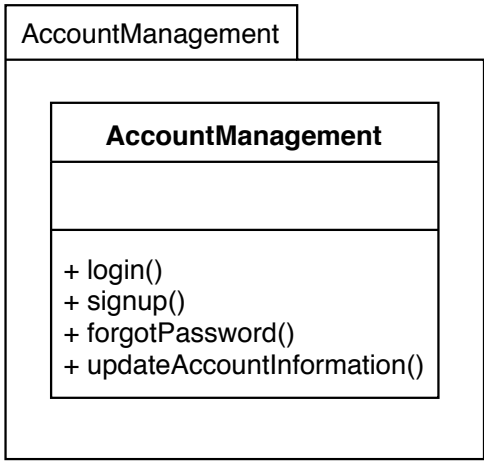

Figure 3.7 Subsystem - AccountManagement

ForumManagement Subsystem

The ForumManagement subsystem controls logic on tasks in the conducting a forum chat. It enables users to create, join, search, open, message and destroy forums. It interacts with the DatabaseInteraction subsystem and gives service to the UserInterface subsystem. Figure 3.8 shows the subsystem.

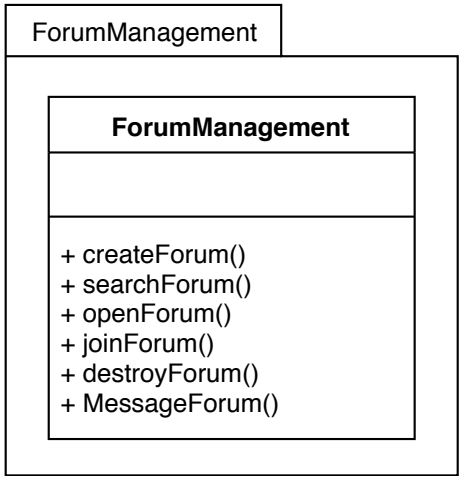

Figure 3.8 Subsystem - ForumManagement

PostManagement Subsystem

PostManagement is a subsystem that handles the core functionality of the AAU Push system. It allows Staff users to send, edit, track and delete posts. And it allows Students to view announcement and mark them as 'Read'. It works by interacting with the Database-Interaction and UserInterface subsystems. And uses the Notification subsystem to notify students as new posts are released. Figure 3.9 shows the subsystem.

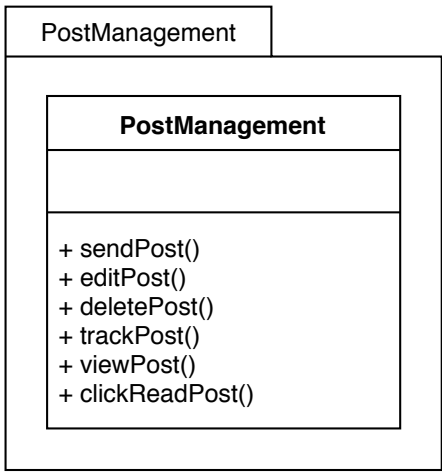

Figure 3.9 Subsystem - PostManagement

Notification Subsystem

The Notification subsystem is responsible for notifying students of new posts and other alters in general. This could be on mobile phone application or other platforms. This subsystem takes service from the DataInteraction subsystem and gives service to ForumManagement, PostManagement, AssignmentManagement and ReminderManagement. Figure 3.10 shows the subsystem.

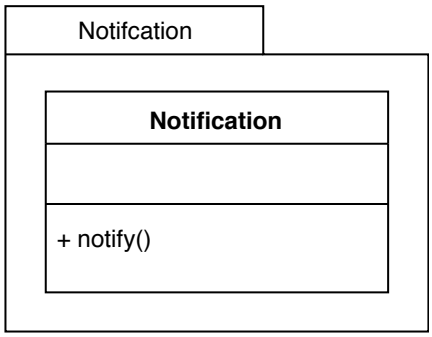

Figure 3.10 Subsystem - Notification

DatabaseInteraction Subsystem

This subsystem is the one which interacts with the data storage area of the system. It takes service from the DataStorage subsystem and gives service to all subsystems in the ApplicationLogic component. Figure 3.11 shows the subsystem.

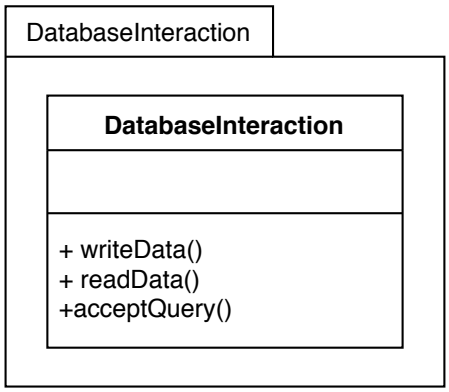

Figure 3.11 Subsystem - DatabaseInteraction

AddDropManagement Subsystem

This subsystem is responsible for handling logic for allowing Instructors and Students to add and drop or remove classes. It works in accordance to the business rules detailed in the nonfunctional requirement specification. Figure 3.12 shows the subsystem.

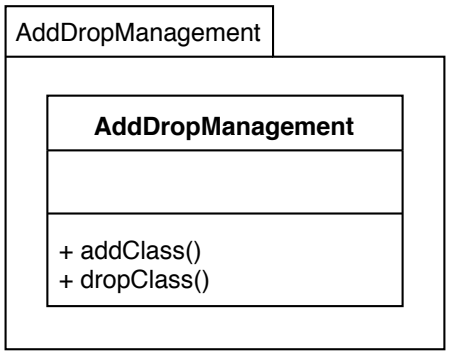

Figure 3.12 Subsystem - AddDropManagement

DataStorage Subsystem

The DataStorage subsystem is the storage area of data used by the system. Any changes to this storage area will be handled by the DatabaseInteraction subsystem. Figure 3.13 shows the subsystem.

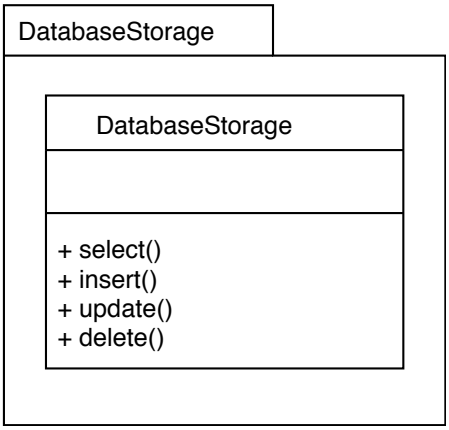

Figure 3.13 Subsystem - DataStorage

3.3.3. Hardware/Software Mapping

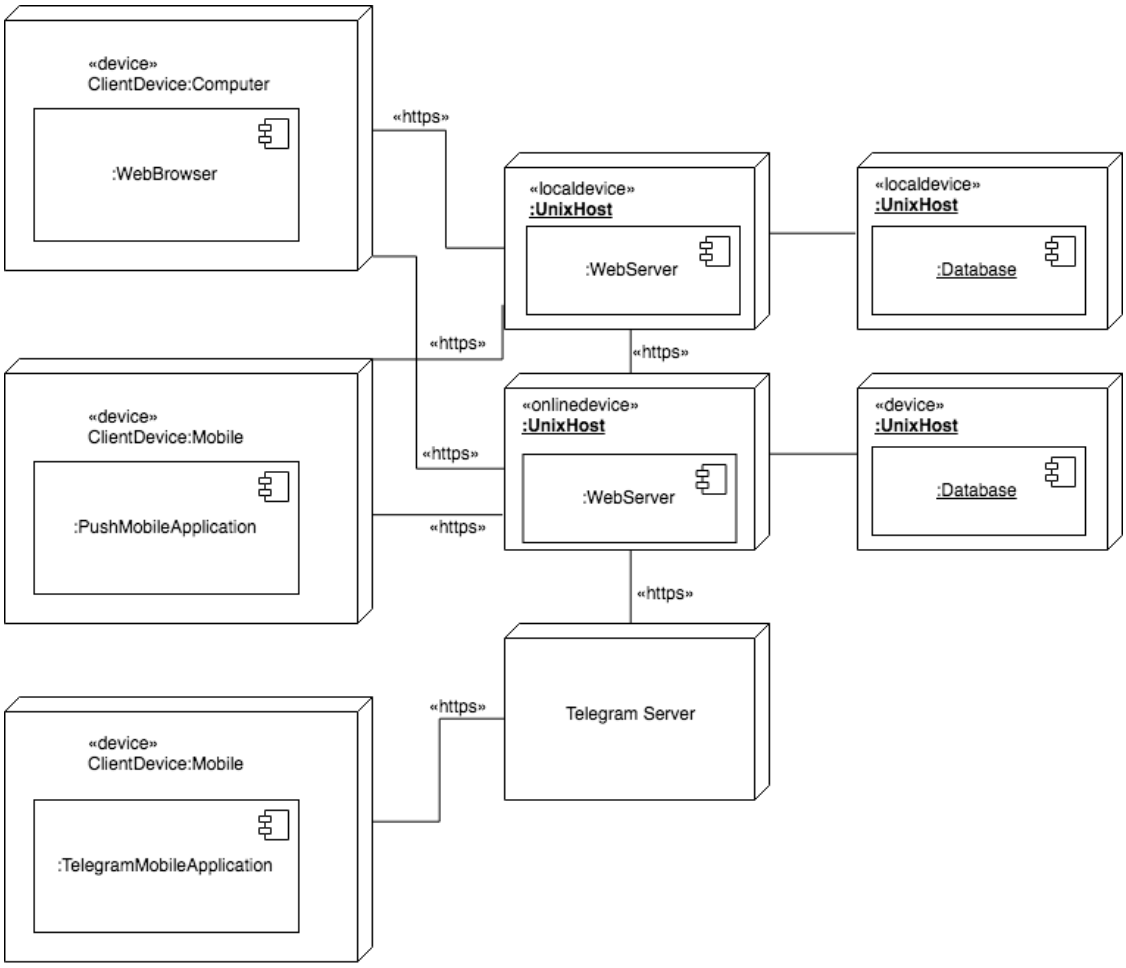

Figure 3.14 Hardware/Software Mapping

As seen in Figure 3.14, we will be using a three-tiered architecture. The three-tiered architectural style organizes subsystems into three layers: The interface layer includes all

boundary objects that deal with the user, web pages, mobile application interfaces, and so on. The application logic layer includes all control and entity objects, realizing the processing, rule checking, and notification required by the application. The storage layer realizes the storage, retrieval, and query of persistent objects.[2]

We will be using the Django framework for our backend. This makes the three-tiered architecture ideal because Django is an MVC framework. Let us look at each layer is closer detail:

1. Interface Layer

Our users of clients will have three main ways of interacting with the system. First is by using the web app which gives all users all necessary functions. Second is by using the Android and iOS applications. Third is by using a Telegram bot. Telegram is a popular instant messaging application in Ethiopia that has a feature called bots. These are automated chatting machines that any user can create and program. We will be using it to connect with our servers and provide students with posts right to their chats.

2. Application Logic Layer

This consists of our servers that are in between the database and the interface. This web server will have API (Application Program Interfaces) to enable the mobile application and the Telegram bot to work. There are two features to notice here:

Firstly, as seen in the figure, the Telegram bot is not directly connected to our servers. Instead, data will always have to flow through Telegram's own server. This is a security protocol they have implemented. So our servers will only be communicating with Telegram's servers.

Secondly, we will be having two web servers. One placed locally within the university's network and another placed on an online hosting service. The reason is to satisfy the Availability and Speed nonfunctional requirements detailed. These two servers will be communicating in order to keep the database up-to-date and correct.

3. Storage Layer

Similar to our Application Layer, our databases will be located in two locations. One within the direct reach of the local server and another one placed on the online hosting server.

3.3.4. Persistent Data Management

3.3.4.1. E-R Diagram

Figure 3.15 found on the next page shows the ER Diagram for AAU Push. Due to the complexity of the diagram, we have printed it on A3 paper.

Figure 3.15 ER Diagram - AAU Push

3.3.4.2. Description of Classes

User - an entity that stores the identity and credentials of a user

Table 3.1 Class Description - User

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the user
Username	Varchar	255	Unique Not null	Holds the username of the user
First name	Varchar	255	Not null	Holds the First name of the user
Last name	Varchar	255	Not null	Holds the last name of the user
Email	Varchar	255	Unique	Holds the email of the user
Password	varchar	255	Min length : 7 Not null	Holds the password of the user

Student - an entity that stores student specific information

Table 3.2 Class Description - Student

Attribute Name	Data Type	Length	Constraints	Description
User Id	Integer	-	Primary Key Foreign Key	Holds the primary key of the user instance the student is asso-ciated with
Entry year	Integer	-	Not null	Holds the year the student joined campus
Section number	Integer	-	Not null	Holds the section number the student is in
Registration number	Varchar	20	Unique Not null	Holds the registration Id of the student
Phone	Varchar	20	Not null	Holds the phone number of the student
Department Id	Integer	-	Foreign Key Not null	Holds the primary key of the department the student is in

Staff - an entity that stores staff specific information

Table 3.3 Class Description - Staff

Attribute Name	Data Type	Length	Constraints	Description
User Id	Integer	-	Primary Key Foreign Key	Holds the primary key of the user instance the staff is associated with
Title	Varchar	15	Not null	Holds the title of the staff
Role	Integer	-	Not null	Holds information to specify the role of the staff
Department Id	Integer	-	Foreign Key	Holds the primary key of the department the staff is in
College Id	Integer	-	Foreign Key Not null	Holds the primary key of the college the staff is in

College - an entity that stores college information

Table 3.4 Class Description - College

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the college
Name	Varchar	255	Unique Not null	Holds the name of the college
Registrar officer Id	Integer	-	Foreign Key	Holds the primary key of the staff that is the registrar officer of the college
College dean Id	Integer	-	Foreign Key Unique	Holds the primary key of the staff that chairs the college

Department - an entity that stores department information

Table 3.5 Class Description - Department

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the department
Name	Varchar	255	Unique Not null	Holds the name of the depart-ment
College Id	Integer	-	Foreign Key Not null	Holds the primary key of the college the department is in
Department head Id	Integer	-	Foreign Key Unique	Holds the primary key of the staff that heads the department

Section - an entity that stores section information

Table 3.6 Class Description - Section

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the section
Year	Integer	-	Not null	Holds the year the section is found in
Section number	Integer	-	Not null	Holds the section number of this section
Department Id	Integer	-	Foreign Key Not null	Holds the primary key of the department the section is in

Course - an entity that stores course information

Table 3.7 Class Description - Course

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the course
Name	Varchar	255	Not null	Holds the name of the course
Course code	Varchar	50	Unique Not null	Holds the course code of the course
Ects	Decimal	-	Not null	Holds the ECTS of the course
Credit hour	Decimal	-	Not null	Holds the credit hour of the course
Department Id	Integer	-	Foreign Key Not null	Holds the primary key of the department the course is in

Post- an entity that stores post information

Table 3.8 Class Description - Post

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the post
Content	Text	-	Not null	Holds the content of the post
Type	Integer	-	Not null	Holds information used to specify the type of the post
Pub date	Date	-	Not null	Holds the date the post was posted
Forum Id	Integer	-	Foreign Key	Holds the primary key of the forum the post was posted to if it was intended for a forum
User Id	Integer	-	Foreign Key Not null	Holds the primary key of the user that posted the post

Reminder- an entity that stores reminder information

Table 3.9 Class Description - Reminder

Attribute Name	Data Type	Length	Constraints	Description
Post Id	Integer	-	Foreign Key Primary Key	Holds the primary key of the post instance the reminder inherited from
Title	Varchar	255	Not null	Holds the title of the reminder that specifies what the reminder is for
Due date	Date	-	Not null	Holds the due date of the project/ assignment/presentation… that the reminder was set for
Place	Varchar	255	Not null	Holds the place of the project/ assignment/presentation… that the reminder was set for
Submission file type	Varchar	10	-	Holds the type of file the user should submit for the assignment if the reminder is for an assignment

File- an entity that stores file information

Table 3.10 Class Description - File

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the file
File	File	-	Not null	Holds the data of the file
Name	Varchar	255	Not null	Holds the name of the file
Extension	Varchar	10	Not null	Holds the file extension of the file
Is image	Boolean	-	Not null	Holds information to specify whether the file is an image or not

Post Id	Integer	-	Foreign Key	Holds the primary key of the post that contains this file
User Id	Integer		Foreign Key Not null	Holds the primary key of the user that posted the post

Forum- an entity that stores forum information

Table 3.11 Class Description - Forum

Attribute Name	Data Type	Length	Constraints	Description
Forum Id	Varchar	255	Primary Key	Holds the unique Id used to identify the forum
Name	Varchar	255	Unique Not null	Holds the name of the forum
Description	Text	-	Not null	Holds the description of the forum
Privacy	Boolean	-	Not null	Holds information that specifies if the forum is either open or closed
Join code	Varchar	20	-	Holds the join code to be used when joining the forum if the forum is closed

Instructor_teaches - an entity that stores the course and section that an instructor teaches

Table 3.12 Class Description - Instructor_teaches

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the relationship
Staff Id	Integer	-	Foreign Key Not null	Holds the primary key of the staff that teaches
Section Id	Integer	-	Foreign Key Not null	Holds the primary key of the section taking the course

Course Id	Integer	-	Foreign Key Not null	Holds the primary key of the course being taught

Student_attends - an entity that stores what courses a student is taking

Table 3.13 Class Description - Student_attends

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the relationship
Student Id	Integer	-	Foreign Key Not null	Holds the primary key of the student attending the class
Instructor_teach-es Id	Integer	-	Foreign Key Not null	Holds the primary key of the class the student is attending

Post_posted_to - an entity that stores the recipients of a post

Table 3.14 Class Description - Post_posted_to

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to iden-tify the relationship
Post Id	Integer	-	Foreign Key Not null	Holds the primary key of the post being sent
Recipient Id	Integer	-	Foreign Key Not null	Holds the primary key of the intended recipient(section, class or department primary key)
Recipient type	Integer	-	Not null	Holds information that specifies which entity the post was intend-ed for

Post_delivered_or_read - an entity that stores the view history of posts

Table 3.15 Class Description - Post_delivered_or_read

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the relationship
Post Id	Integer	-	Foreign Key Not null	Holds the primary key of the post that was viewed by the student
Student Id	Integer	-	Foreign Key Not null	Holds the primary key of the student that viewed the post
Is read	Boolean	-	Not null	Holds information that specifies whether the post was read

File_downloaded - an entity that stores the download history of files

Table 3.16 Class Description - File_downloaded

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the relationship
File Id	Integer	-	Foreign Key Not null	Holds the primary key of the file that was downloaded by the student
Student Id	Integer	-	Foreign Key Not null	Holds the primary key of the student that downloaded the file

File_submitted_for - an entity that stores the download history of files

Table 3.17 Class Description - File_submitted_for

Attribute Name	Data Type	Length	Constraints	Description
Id	Integer	-	Primary Key	Holds the unique Id used to identify the relationship

File Id	Integer	-	Foreign Key Not null	Holds the primary key of the file that was submitted by the student
Student Id	Integer	-	Foreign Key Not null	Holds the primary key of the student that submitted the post
Assignment Id	Integer	-	Foreign Key Not null	Holds the primary key of the assignment the file was submitted for

3.3.5. Access Control and Security

Access Control describes the access rights to each operation of a class respective to each actor of the system. Generally, it is used to describe policy definitions with regard to user's access rights throughout the subsystems.

Based on the use case descriptions and subsystem specifications Access control and security of the AAU Push system is modeled with access matrix as shown in Table 3.18.

Table 3.18 Access Control - AAU Push

Subsystem	**Class**	**Operation**	**Actor**				
			Student	Instructor	DepartmentHead	CollegeDean	RegistrarOfficer
UserInterface	AccountManagementInterface	Load() Unload()	Y	Y	Y	Y	Y
	StaffPortalInterface			Y	Y	Y	Y
	StudentDashboardInterface		Y				
	ForumInterface		Y	Y	Y	Y	Y
ReminderManagement	ReminderManagement	setReminder() editReminder() deleteReminder()		Y			
		viewReminder()	Y				
PostManagement	PostManagement	sendPost() editPost() deletePost() trackPost()		Y			
		viewPost()	Y				
		clickReadPost()	Y				

Nofication	Nofication	notify()	Y				
ForumManagement	ForumManagement	createForum() searchForum() openForum() joinForum() destroyForum() messageForum()	Y				
DepChatManagement	DepChatManagement	sendMessage()		Y	Y	Y	
		viewMessage()		Y	Y		
DatabaseStorage	DatabaseStorage	select()	Y	Y	Y	Y	Y
		insert()	Y	Y	Y	Y	Y
		update()	Y	Y	Y	Y	Y
		delete()	Y	Y	Y	Y	Y
DatabaseInteraction	DatabaseInteraction	writeData()	Y	Y	Y	Y	Y
		readData()	Y	Y	Y	Y	Y
		acceptQuery()	Y	Y	Y	Y	Y
AssignmentManagement	AssignmentManagement	giveAssignment()		Y	Y		
		viewAssignment()	Y				
		submitAssignment()	Y				
AddDropManagement	AddDropManagement	addClass()	Y	Y	Y		
		dropClass()	Y	Y	Y		
AccountManagement	AccountManagement	login()	Y	Y	Y	Y	Y
		signup()	Y				
		forgotPassword()	Y	Y	Y	Y	Y
		updateAccountInformation()	Y	Y	Y	Y	Y

REFERENCES

[1] AAU at a glance [Online]. Available: http://www.aau.edu.et/about/aau-at-glance/ [December 27, 2018].

[2] B. Bruegge, A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Third Edition. 2010.