Open Learning Management System Lite

Final Documentation

**Team:**

Kym Haywood II, Byron (Noah) Hickey, Cameron Morgan,

Austin Simmons, Chinonyerem Ugwumadu, & Shamarcus Walker

CSCI 5530-A Software Engineering

Department of Computer Science

Georgia Southern University

Spring 2019

Table of Contents

[**Introduction**](#_gvpi2fwnz3zw) **5**

[Team Members](#_9grn72g83515) 5

[**Feasibility Report**](#_o1npc8no64v) **6**

[Scope](#_c1lorif0ycw9) 7

[Project Requirements](#_9mk435cfm3os) 7

[Risk Assessment of Viable Options](#_mgudqe1rl4t) 10

[Feature Risk Reasoning](#_c0vzs1rdqjsr) 11

[Executive Summary](#_3o8v7ch3po4y) 14

[**Software Requirements**](#_gq4rtytwuuap) **15**

[1. Introduction](#_f5kimeckx5if) 16

[1.1 Purpose](#_lo4w6t7qjpbw) 16

[1.2 Scope](#_clhb36jcees8) 16

[1.3 Definitions, Acronyms, and Abbreviations](#_r9hkoauvw4mm) 16

[1.4 References](#_vj06pyr6wg06) 17

[1.5 Overview](#_j24sb5aeyrjt) 17

[2. The Overall Description](#_7wnkhpjy8e09) 18

[2.1 Product Perspective](#_om7h3hk7fdzf) 18

[2.1.1 System Interface](#_3f3oop233207) 19

[2.1.2 Interfaces](#_3ent0d51kzlq) 19

[2.1.3 Hardware Interfaces](#_b17iq446puve) 20

[2.1.4 Software Interfaces](#_ywnuzynetqly) 20

[2.1.5 Communications Interfaces](#_xlfs5eg7bx1m) 21

[2.1.6 Memory Constraints](#_u8sexvnvuigt) 21

[2.1.7 Operations](#_z9affqf3pbjz) 21

[2.1.8 Site Adaptation Requirements](#_5lmisofi5ty3) 21

[2.2 Product Functions](#_fo9jb9hcxrb3) 22

[2.2.1 User Use Case](#_d3xh4uckin0) 22

[2.2.2 Professor Use Case](#_fkjyr2lwrdvl) 23

[2.2.3 Student Use Case](#_o5yw8fikrf9k) 28

[2.2.4 Admin Use Case](#_61kyz5c6pl4t) 32

[2.3 User Characteristics](#_jt2dniwdl5zf) 37

[2.4 Constraints](#_kg92ll98rprv) 37

[2.5 Assumptions and Dependencies](#_kij9laitdi8m) 37

[2.6 Apportioning of Requirements](#_grso9su4vtmy) 37

[3. Specific Requirements](#_6rguf8nba8v) 38

[3.0.1 Profile](#_72en1kejmubh) 38

[3.0.2 Quizzes/Surveys](#_oufu9ucnoson) 38

[3.0.3 Reports](#_z2dtzuuoz66k) 39

[3.1 External Interfaces](#_20vtj6lwm0i9) 39

[3.1.1 Grade Export](#_xgljpsbb3mrg) 39

[3.1.2 Quiz/Survey Export](#_202vjuakhugk) 39

[3.1.3 Quiz/Survey Import](#_c8awqyot9jub) 40

[3.2 Functions](#_k3ek37g18p3v) 41

[3.2.1 User Logs In](#_q3gmopmte6yu) 41

[3.2.2 Professor Uploads Homework](#_gx4nnd3j1piq) 41

[3.2.3 Professor Creates Quiz/Survey](#_mlbgb6fw23ez) 42

[3.2.4 Professor Imports Quiz/Survey](#_tsjdi13olosa) 42

[3.2.5 Professor Views Students Grades](#_ej6bscnv4emo) 43

[3.2.6 Professor Exports Quiz/Survey](#_fm26lykafig5) 43

[3.2.7 Professor Exports Grades](#_3qefp8kie7tk) 44

[3.2.8 Student Takes Quiz/Survey](#_r8iuahee2yfi) 44

[3.2.9 Student Downloads Homework](#_1x9o6i6r7cr) 45

[3.2.10 Student Submits Completed Homework](#_rjefri25s3ih) 45

[3.2.11 Student Views Their Grades](#_1pk97nu5hcy7) 46

[3.2.12 Admin adds a User](#_v9pwhof7oqop) 46

[3.2.13 Admin deletes a User](#_rkpt8cegg0ng) 46

[3.2.14 Admin edits a users’ password](#_tg4c5cze37hr) 47

[3.2.15 Admin adds a new course](#_v43l152peeql) 47

[3.3 Performance Requirements](#_eyb4c8bhxthm) 48

[3.4 Logical Database Requirements](#_ccidwo92ggj4) 48

[3.5 Design Constraints](#_w0hnnlp3culo) 49

[3.5.1 Standards Compliance](#_mxu2h436b6k8) 49

[3.6 Software System Attributes](#_ezncpsfy9ypj) 49

[3.6.1 Reliability](#_18mindlsqsyg) 49

[3.6.2 Availability](#_yu4429h8auw5) 49

[3.6.3 Security](#_9k1gl89e7zb8) 49

[3.6.3.1 Password Encryption](#_dcku9bsmkwtb) 49

[3.6.3.2 xAPI IP Restriction](#_pvqm9py4wvy4) 50

[3.6.3.3 CSRF Token](#_p2zc2ic4cx2g) 50

[3.6.4 Maintainability](#_1nmqcz6cya3e) 50

[3.6.5 Portability](#_jsbbdjgmvuaz) 50

[3.7 Organizing the Specific Requirements](#_vcaxyf8k6frh) 50

[3.8 Additional Comments](#_vhwu7rix30sr) 51

[4. Change Management Process](#_rwiau5dzrj6i) 51

[**Project Schedule**](#_j1uum3l70rbk) **52**

[Gantt Chart](#_5sw1ozt8t7wz) 53

[LOC Estimation](#_ep9lbhthgee1) 55

[**Design Document**](#_oup5de4xjhc4) **56**

[Abstract](#_1fob9te) 57

[Introduction](#_tyjcwt) 57

[1.1 Purpose of System](#_zww3n2hvly9) 58

[1.2 Functional and Nonfunctional Requirements](#_1t3h5sf) 58

[1.3 Design Methodology](#_7jtgex6adkmn) 59

[1.4 Definitions, acronyms, and abbreviations](#_17dp8vu) 60

[1.5 Overview](#_3rdcrjn) 60

[Proposed Software Architecture](#_9sj48m50mh4n) 61

[2.1 Overview](#_35nkun2) 61

[2.2 Subsystems](#_3j2qqm3) 61

[2.3 UML Profiles](#_1ci93xb) 62

[2.4 Subsystem Decomposition](#_41mghml) 63

[Object Design](#_3fwokq0) 63

[3.1 Overview](#_3058rqf5y83r) 63

[3.2 Object Interaction](#_sdmaq7csajod) 65

[3.3 Detailed Class Design](#_3cqmetx) 65

[3.3.1 Courses Package](#_f2scoxtr5rr) 66

[3.3.2 Accounts Package](#_4bvk7pj) 66

[**Testing Document**](#_9gbobtbaqhe6) **67**

[Introduction](#_t3r3xb06553) 68

[1.1. Test Plan Objectives](#_3yb30rdpg32l) 68

[1.2 Scope/Plan](#_xtw4ey3iby8k) 68

[Test Strategy](#_f1f4xeb0ugjt) 68

[2.1. Data Entry](#_1faocq9j3004) 68

[2.2. Performance Test](#_j3enfiqi961m) 69

[2.3. Security Test](#_o7m8faqhyb2x) 70

[2.4. Automated Test](#_oduc89vh6bd3) 72

[2.5. User Acceptance Test](#_b27dzw5bpo70) 75

[Functions To Be Tested](#_3jvajsfr1mjk) 78

[Dependencies](#_jmp4r4vlgabw) 78

[Schedule](#_1o7kp69wr8gy) 78

[Tools](#_o7k48ukduigv) 79

[Test Cases](#_skykweu93dd8) 79

[2.2.1 User Use Case](#_pfpe6ssi4y) 79

[2.2.2 Professor Use Case](#_j9ew2p2774is) 80

[2.2.3 Student Use Case](#_xz5b1faqgzcl) 86

[2.2.4 Admin Use Case](#_sl59bmld6h5y) 89

[Code Metrics](#_d5eqhdx8bj78) 93

[**Post Mortem**](#_p99hpbu7snlz) **94**

[**Appendix**](#_mwylwcldpybx) **95**

[Meeting Minutes](#_ldviwqjvh5a) 95

[**References**](#_li9gtdf5tgoy) **117**

# Introduction

## Team Members

Name: **Kym Haywood II**

Role: Project Lead/Backend Developer

Responsibilities: Maintain and facilitate group tasks and keep project moving along, Set Up meetings with client, Backend developer for django views and modules

Name: **Byron (Noah) Hickey**

Role: Test Lead

Responsibilities: Design document and Testing

Name: **Cameron Morgan**

Role: Frontend Developer (HTML/CSS)

Responsibilities: Frontend developer for html pages, Customizing web pages using CSS

Name: **Austin Simmons**

Role: Frontend Developer (HTML/JS)

Responsibilities: Frontend developer lead for Javascript and HTML

Name: **Chinonyerem Ugwumadu**

Role: Backend Developer

Responsibilities: Backend Developer lead, Django specialist

Name: **Shamarcus Walker**

Role: Backend Developer

Responsibilities: Backend Developer for python functions

Open Learning Management System Lite:

# Feasibility Report

## Scope

This project will be designing and creating a Learning Management System (LMS). A LMS is a software application that allows administration, documentation, tracking, reporting and delivery of educational courses. The LMS system will have two main roles, an educator and a student. An educator is the one who will administer the course work and a student is the one who will be using the course work.

## Project Requirements

The project requirements come from a combination of client specific requirements and researched features that most LMS’s have. These items will be the possible features of the application.

* Profile

A representation of the user’s account displaying the user’s information including his/her email. Each one will be created upon the user creating an account.

* Task view for Students

This feature will allow students to see what current tasks need to be done. The task view will assist the student in keeping track of the current assignments, quizzes, surveys, etc. that are yet to be completed until the decided due date/time of the task.

* Email Notification

With the previous version of the project we implemented an email system using google’s free smtp server. This will make research very minimal due to prior knowledge. However implementation will be a little higher because we may have to replicate this in another language or framework.

* Reports

There will be reports for professors and students. Professors will have the ability to view a grade report and a response report for each quiz. The grade report will show the grade for each student that took a quiz. The response report will show the student responses for each quiz question. Another report available for a professor will be live reporting for surveys. This will be a visual representation that will show in real time the student responses as they submit the survey. Students will be able to view a list of their quiz scores.

* Secure

Given that our software is a LMS, we have to create a learning environment for students. The easiest way for us to do so is to allow the users to create accounts. We have to secure the passwords for each account. We plan to do this by using base64 encoding and custom salts that can be decrypted with our algorithm. Many of our members have knowledge on base 64 encoding and how encryption works so research is minimal, and implementation will consist mainly of simple math and string concatenation.

* Assignments

This feature allows professors to upload assignments for students to access them. These assignments that are upload will be downloadable for the student to complete off of the web. There will also be a preview feature so the student will be able to see the assignment before downloading it. Previewable assignments with be of doc., docx., or txt format.

* Quizzes and Surveys

This feature allows for students to take surveys and quizzes online. Professors will be able to set time limits for quiz access and they can also set a restriction on IP addresses if they want their student to access the quiz in a specific place. Professor has the option on whether they want to upload the questions from a text file or if they want to enter questions manually.

* xAPI Compliant

xAPI contains an extensive list of requirements for software to be considered compliant. Hence, we use a reputable github repository as a reference guide to what aspects we can use in our project in order to meet the xAPI requirements.

<https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Communication.md#applicationjson>

* Study Tools

Students will have access to different study tools within this program. Flashcards are an available feature for students to create and access within their profile. Slideshows, meaning files of ppt, pptx, and pdf, will uploaded by professors and able for download by students. Videos will also be uploaded by professors and able for download by students. There will be no preview functionality for this feature

* Data Migration

The data migration feature will allow data to be extracted and transferred from a database and/or storage system, to the lms. Validation of completeness and the withdrawal of old data will be apart of the migration process.

* Auto Grade

This feature will allow the calculation of the grade each student makes on each multiple choice quiz based on certain preferences such as the number of questions and the worth of each question. This feature will also have a curve function in which the grade of each student on a certain quiz will be adjusted to help boost each student’s score depending on whether the course is based on percentages or points.

* IP Restriction

This feature will allow professors to control where students are able to take surveys and/or quizzes. By filtering the IP addresses to only those connected to the nearest network, Professors will be able to restrict quizzes and/or surveys to students that are physically in the class.

* Quiz Codes

Similar to the IP restriction feature, the quiz code feature will allow Professors to restrict quizzes and surveys to only the students in the classroom. Professors will be able to provide an access code to their students, to access the given quiz and/or exam.

## Risk Assessment of Viable Options

Pulling from the project requirements, this section will analyze the risk of these options and see which ones are viable given the time limitation of the project. This analysis will factor in time needed to research unknown areas and time for actual implementation. A high risk key is between 8-10, this means that the feature will either take a large amount of time to research or implement. A medium risk key is between 4-7, this key means that the feature will take a moderate amount of time to research or implement. A low risk key is between 1-3, this key means that the feature will take a low amount of time to research or implement. N/A refers to a category that will not be applicable in the specified area.

|  |  |
| --- | --- |
| **Heat Map Risk Key** | |
| **High** | **8-10** |
| **Medium** | **4-7** |
| **Low** | **1-3** |
| **N/A** | **N/A** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Feature | Research | Implementation | Time | Total Risk |
| 1 | Quiz Codes | 1 | 2 | 2 | 1.666666667 |
| 2 | Profile | 3 | 3 | 3 | 3 |
| 3 | Auto Grade | 3 | 3 | 3 | 3 |
| 4 | Task view for Students | 5 | 4 | 4 | 4.333333333 |
| 5 | Notification | 5 | 4 | 4 | 4.333333333 |
| 6 | Data Migration | 5 | 3 | 5 | 4.333333333 |
| 7 | Quizzes and Surveys | 3 | 7 | 5 | 5 |
| 8 | IP Restriction | 6 | 3 | 6 | 5 |
| 9 | Secure | 7 | 6 | 5 | 6 |
| 10 | Reports | 8 | 7 | 8 | 7.666666667 |
| 11 | Assignments | 8 | 8 | 8 | 8 |
| 12 | Study Tools | 8 | 8 | 8 | 8 |

## Feature Risk Reasoning

* Quiz Codes
  + The reasoning for choosing a one for research is because members of the project team have experience with quiz codes and functionality within the program.
  + With implementation, the team already has an understanding of the complexity behind getting this portion done and what it entails to build successfully. Through experience the team understands that this isn’t to complex of a task.
  + With little research to be done and with implementation complexity being low, thus the time to complete this feature will be low.
* Profile
  + The team has had some experience with front end web development so the team would have to do little research when it comes to this.
  + The team understand how to implement this feature, and the rigorousness of this implementation is not high.
  + Do to little research and low complexity implementation, the time taken for completion should be low.
* Auto Grade
  + The team has an understanding of how to implement a program that auto grades test and quizzes.
  + Do to the team’s experience with past projects there is an understanding on how to implement this feature efficiently.
  + This project take little effort to research and little complexity to implement but to make sure everything works properly but this feature as a whole should not take long to complete.
* Task view for Students
  + The team would have to look up a good amount of information involving this.
  + The implementation of this feature should be simple however due to the lack of knowledge the implementation could be more difficult than expected.
  + With research and implementation in consideration, time to do this is moderate.
* Notification
  + The team has a moderate understanding of how to implement notifications into the project which thus there will be a noticeable amount of research that is needed to perfect this feature.
  + Having a few members that have implemented this feature before it has been deemed a 4 on the risk scale due to members with past experience.
  + This feature will take a slightly below average time to complete which is why we rated it low in comparison to some complex features presented.
* Data Migration
  + Although team members that have worked with similar functionalities it has been assessed as an average risk because there will be research to implement in the manner need to complete this feature.
  + The team found this feature shall be fairly easy to complete after gather further information about the feature in regards to our project.
  + This feature was given a 5 for time due to the need of other information from the project to complete it.
* Quizzes and Surveys
  + The team has built previous projects in the past that use this feature therefore we decided to give research a 3 for this part of the project.
  + Previously the team ran into problems with implementation of this feature so in order to error on the side of caution we rated this a 7 for implementation.
  + Although implementation can be hassle for the team at least the knowledge of how to do it is there thus this feature has been given a 5 for Time.
* IP Restriction
  + With the team having some but little knowledge of using this feature on a large scale project, it has been given a 6 for research.
  + The team is confident that after further research implementation of this feature will not cause many problems which is why it has been rated a 3 for implementation.
  + For time, this feature has been given a 6 because it will take time to research and we fill if that is done in a timely manner we should be able to complete this in an acceptable time.
* Secure
  + The team decided to give this a 7 for research because we have little to no knowledge of how to secure the LMS in a standardized manner.
  + After doing discussion the team rated implementation a 6 for this feature because we think that it will take a noticeable time due to the lack of experience with standardized measures.
  + This was rated a 5 for time because we feel as if research is don’t properly we should be able to conduct implementation in a timely manner.
* Reports
  + Reports rated out at 8 because everything involving the topic will still be researched even if we have prior knowledge to insure it’s correctly done to the xAPI standard.
  + Implementation for reports rated out at 7 due to the fact we have to use xAPI standard to fulfill requirements for reports
  + Time rated out at an 8 because we will have to research and implement as well as wait for other data/features to be added into the project.
* Assignments
  + Assignments was given an 8 for research because The team lacks knowledge needed for implementation therefore there will be a lot of research.
  + The team feels as if it will be hard to implement assignments as well as the add one that go with assignments therefore we rated them an 8.
  + Time was also rated 8 because for the team to research and learn the material and also implement it is going to take a large amount of time.
* Study Tools
  + The team rated research for study tools to be an 8 because we lack knowledge of how to implement therefore we have to do a lot of research.
  + Due to the fact we have to research many of the topics involving Study Tools we feel as if implementation will be difficult therefore we rated it an 8.
  + Time was also rated an 8 due to the fact that there will be a lot of research and difficult implementation involved in this feature.

## 

## Executive Summary

This project will incompasses building a LMS for use by students and teachers to provide a learning platform to promote positive learning. This report describes and explains possible designs and solutions to the project. Through intensive research, the project’s designs and features have been evaluated based on the risk involved with completion of each. Although some of the main functionality of the project have been deemed high risk, the team and client will discuss possibilities to further ensure the completion of the project. It is concluded that building an LMS with the above requirements is well within reasonability. With that being said the projects main functionality we be completed and things that the team and client agree are too risky will be changed accordingly. With the information given in this report, the project is found to be feasible.

Open Learning Management System Lite:

# Software Requirements

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to present a detailed description of the Learning Management System Lite. The document will explain the purpose and features of the LMS system, what the system will do, the limitations that the system must operate under, and how the system will react to outside forces. This document is intended for the use of stakeholders and the developers of the system. It will be proposed to Dr. Andrew Allen for its approval.

### 1.2 Scope

This software system will be a Learning Management System Lite for a professor at Georgia Southern University. This system will be designed to assist the professor in administering educational tools by an alternative means than the current university system. By providing an additional educational tool, the system will meet the professor’s needs while remaining easy to use.

More clearly, this system is designed to allow the professor to post quizzes, surveys, and homeworks for students to access. The software will also act as a storage space for text, pdf, and word documents that the professor may wish to distribute among classes. The LMS Lite system will have an export and import feature that will help to integrate this new system with the current university system. To facilitate the data being collected, the information will be stored in a relational database.

### 1.3 Definitions, Acronyms, and Abbreviations

|  |  |  |
| --- | --- | --- |
| **Term** | **Abbreviation** | **Definition** |
| Assignment |  | A quiz, survey, and/or homework |
| Experience API | xAPI | An e-learning software specification that allows learning content and systems to speak to each other in a manner that records and tracks all types of learning experiences. |
| Georgia Southern University | GSU or University | The educational institution that the client teaches at |
| Homework |  | A .doc, .pdf, or .txt file |
| Hypertext Transfer Protocol | HTTP | Hypertext Transfer Protocol is a protocol which allows the fetching of resources. |
| Learning Management System | LMS | A software application for the administration, documentation, tracking, reporting and delivery of educational courses, training programs, or learning and development programs |
| Mean Time Between Failure | MTBF | Predicted elapsed time between inherent failures |
| Simple Mail Transfer Protocol | STMP | Simple Mail Transfer Protocol is an Internet standard for email transmission. |
| Software Requirements Specification | SRS | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. |
| Stakeholder |  | Any person with an invested interest in the project who is not a developer |
| Structured Query Language | SQL | Standardized query language for requesting information from a database |
| Survey |  | A quiz that is not graded |

### 1.4 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements*

*Specifications.* IEEE Computer Society, 1998.

Teamleader, Joan, et al. *Software Requirements Specification.*

[www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc](http://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc), 2004.

### 1.5 Overview

The next section, The Overall Description, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements in the preceding chapter. This chapter will be most beneficial to the customer and any using the system.

The third section, Specific Requirements, of this document is written primarily for the developers and describes in technical terms the details of the functional requirements of the product. Both section two and three describe the same software product, but are intended for different audiences and therefore use different language.

# 

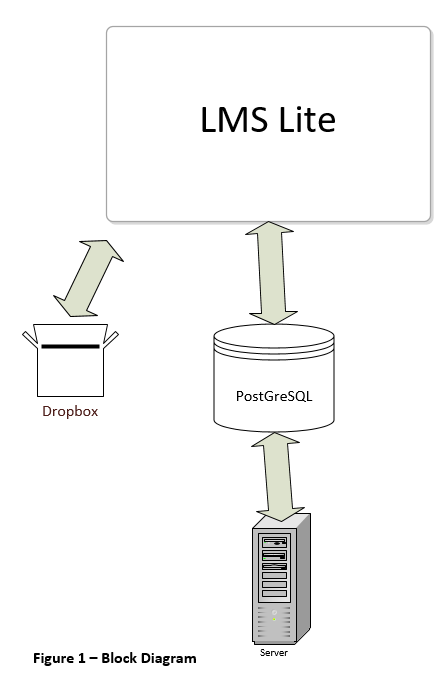
## 2. The Overall Description

This Learning Management System shall simulate the classroom environment within the web browser. There should be features that provide ease with professor and student interaction. The learning environment shall consist of testing, quizzing, and surveying within the application. This application will also allow for professors to upload homework for their students to access, complete, and upload for their submission. Overall, this application can be used efficiently between users for good learning and teaching experience.

### 2.1 Product Perspective

A learning management system is a software application that is used for the betterment of education electronically. This should provide ease of reporting, documentation, and simplify the use of providing educational courses online. This product can be in comparison with online educational resources such as Blackboard, Folio from desire to learn, and Moodle in which they will have similar features to assist with the educational process. The difference between this software and others is that this software.

The software will communicate with the following structure (Figure 1). There shall be the functioning LMS Lite application on the web browser. This application will communicate with a third party storage resource Dropbox, and the SQL Database PostgreSQL. Lastly, the database shall share communication with the web server.



#### 

#### 2.1.1 System Interface

This section was not added to the report because the LMS lite will only be used on a web browser and there is no real concern of system interfaces. The client will not have to use any other program, and there will be no integration of two systems.

#### 2.1.2 Interfaces

Three types of users will interact with the system: administrators, professors and students. A user of the system shall see the log-in page upon visiting the web portal. If the user is not registered, the only way to be add is administratively. Every user shall have a profile page where system use is user specific.

Administrators are able to add and remove students and/or professors to the system. They also have the ability to change passwords, to the students or professors request.

The functionality provided to the professor webpage are more extensive. Data migration is an important feature for professors. Professors shall be able to import quizzes and surveys in the form of .txt files, and homework as .txt, .pdf, and .doc files. Grades will be exported as a .csv file to the professor. The system shall create reports of grades, and assignments done by students, and professors shall be able to export them. A gradebook will be made for professors based off of results from assignments.

The student view will allow for quizzes and surveys to be taken. There will be an interface for homework to be downloaded and uploaded to a dropbox like setting. Students will also have the ability to view grades, due dates and times presented by the  
professor.

#### 2.1.3 Hardware Interfaces

This section was not included because the LMS lite will not run on any foreign hardware interfaces. The only hardware interface that the program will run on is a web browsers which is more software based and doesn’t rely solely on hardware capabilities.

### 

#### 2.1.4 Software Interfaces

1. Google Chrome
   1. Version: 72.0.3626(Current)
   2. Source: Google LLC
   3. Google Chrome is a open-source web browser that is displayed in plain text. The web browser allows for access to the world wide web.
   4. Google Chrome is necessary because the LMS lite relies on a web browser to run. Google Chrome will allow for the most up to date web browser and best server to client communication.
2. PostgreSQL
3. Version: 11.2
4. Source: The PostgreSQL Global Development Group
5. PostgreSQL is a open-source SQL database management tool that is displayed in plain text, and queried using SQL.
6. PostgreSQL is necessary to the LMS lite because it allows for database management which will be needed to store data that is essential to the programs such as assignments, users, and courses.

The system must use google chrome in order to access the LMS. Each user should have the latest version of google chrome installed on their respective devices.

#### 2.1.5 Communications Interfaces

The LMS lite will use multiple communication protocols because it is a web based application. HTTP will be used to transfer files from the web server. TCP/IP will be used to restrict IP during quiz taking process. STMP is the protocol that will be used to send notification via email.

#### 2.1.6 Memory Constraints

It is anticipated that each school computer has a minimum of 8GB ram, which is well above the requirement to run a web browser, there is no memory constraint that concern the operation of the LMS lite.

### 

#### 2.1.7 Operations

This section is covered in section 2.1.2. If needed, this section can be used at a later date to elaborate further.

#### 2.1.8 Site Adaptation Requirements

There will be no site adaptation needed for the LMS Lite. It solely needs a web browser to function. This is why this section is left off of the SRS report.

## 

### 2.2 Product Functions

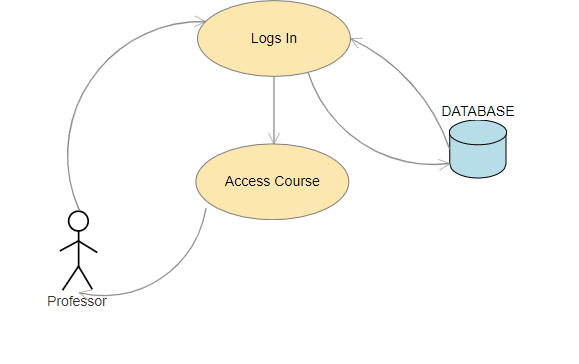
This section outlines the use cases for each of the functionalities of the LMS Lite application.

#### 2.2.1 User Use Case

Use case: **User Login**

Statement: The system shall allow a user to login.

Diagram:



**Brief Description**

The user can be either a professor or student, they will log in with username/email and password. Upon successful credentials store in the database the user will go to their designated page based upon user type.

**Initial Step-By-Step Description**

User has clicked the log in button on the homepage of the application

The User enters his/her username.

The User enters his/her password.

The User clicks the login button.

Database is called to find the user credentials that were entered.

If profesor type go to professor profile, if student type go to student profile.

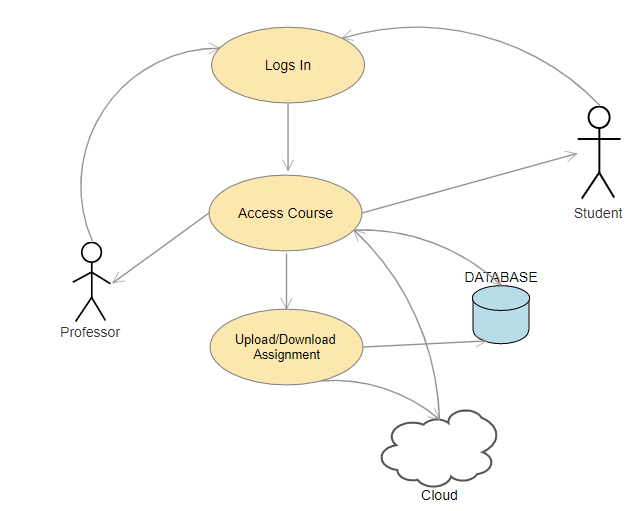
### 2.2.2 Professor Use Case

A.

Use case: **Professor Uploads Homework**

Statement: The system shall allow a Professor to upload homeworks.

Diagram:



**Brief Description**

The professor user has logged in and uploads homework by the click of a specific button on professor homepage screen. User has already navigated to the class in which to add homework to.

**Initial Step-By-Step Description**

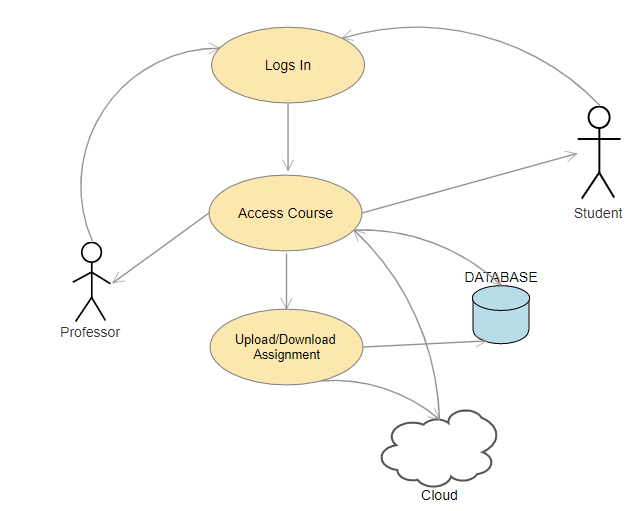
1. The clicks the upload homework button.
2. File Explorer dialog box opens.
3. Professor navigates and selects the homework file.
4. Professor clicks ok button on file explorer dialog box.
5. Professor sets start and stop time/date for the assignment.
6. Professor clicks the upload button.

B.

Use case: **Professor Creates Quiz/Survey**

Statement: The system shall allow a Professor to create a quiz and survey.

Diagram:



**Brief Description**

The professor user has logged in and creates a quiz/survey by the click of create quiz/survey button. User has already navigated to the class in which to add quiz/survey to.

**Initial Step-By-Step Description**

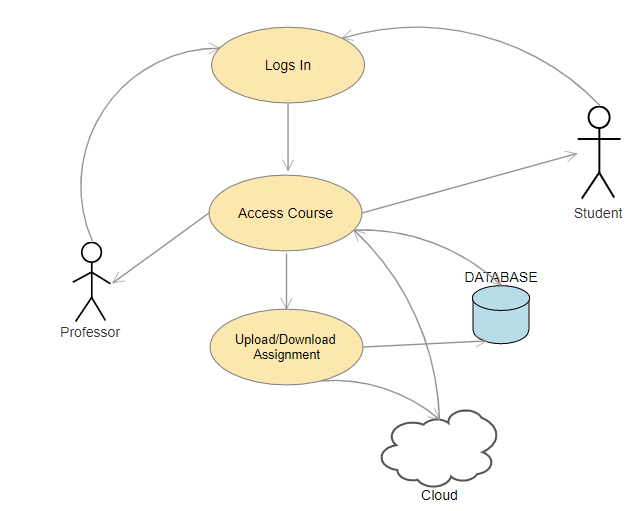
1. Professor selects create quiz/survey.
2. New pane/window opens.
3. Professor enters quiz/survey name.
4. Professor enters the start and end time/date.
5. Professor builds quiz/survey.
6. Professor clicks review.
7. Professor click upload.

C.

Use case: **Professor Imports Quiz/Survey**

Statement: The system shall allow a Professor to import a quiz and survey.

Diagram:



**Brief Description**

The professor user has logged in and Imports Quiz/Survey by the click of the “Import quiz/survey” button. User has already navigated to the class in which to add quiz/survey to.

**Initial Step-By-Step Description**

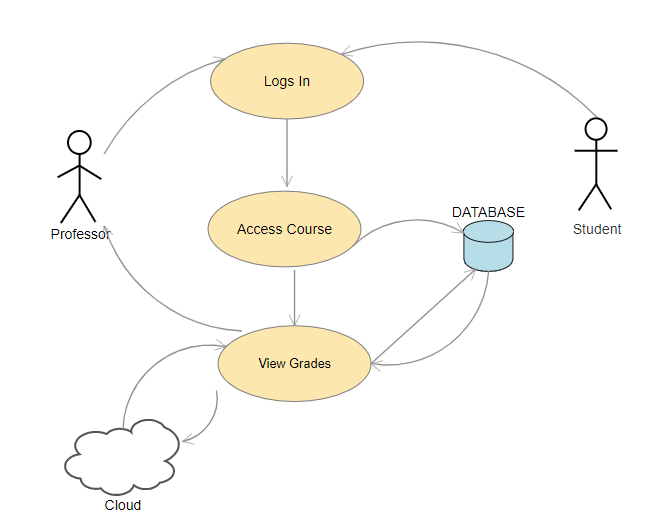
1. Professor selects import quiz/survey.
2. File Explorer dialog box opens.
3. Professor navigates and selects the quiz/survey file.
4. Professor clicks ok button on file explorer dialog box.
5. Professor is able to view the assignment and add points to each question
6. Professor sets start and stop time/date for the assignment.
7. Professor clicks the upload button.

D.

Use case: **Professor Views Students Grades**

Statement: The system shall allow a Professor to view students grades.

Diagram:



**Brief Description**

The professor user has logged in and clicks the“View Grades” button. User has already navigated to the class in which grades they want to view.

**Initial Step-By-Step Description**

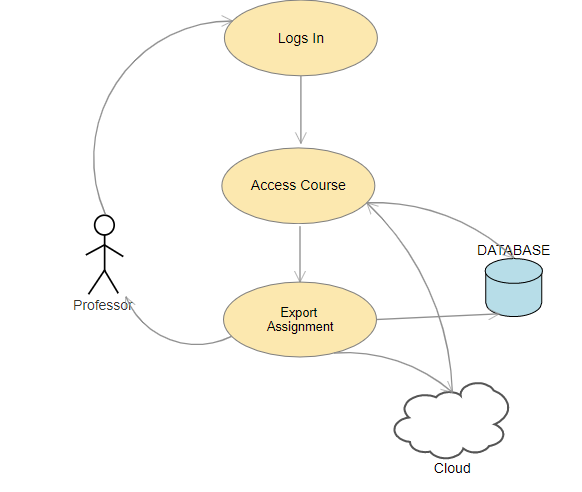
1. Professor click view grades.
2. Database is queried to populate grade report specific grades from the table with that classes grade in them.
3. Grade book is displayed with list of assignments.
4. Specific assignment is clicked and breakdown of student grades is displayed.

E.

Use case: **Professor Exports Quiz/Survey**

Statement: The system shall allow a Professor to export a quiz and survey.

Diagram:



**Brief Description**

The professor user has logged in and clicks the“Export Quiz/Survey” button. User has already navigated to the class in which the quiz/survey is located.

**Initial Step-By-Step Description**

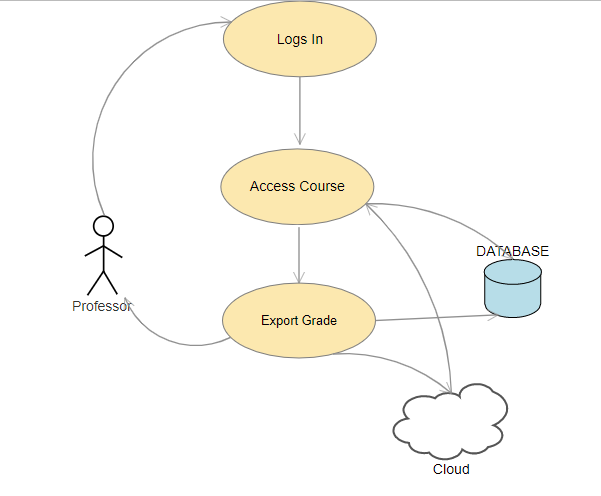
1. Professor clicks export button.
2. Professor chooses export quiz/survey.
3. Professor selects quiz/survey.
4. Professor clicks download.
5. File explorer opens.
6. Professor chooses the file location and name of file.
7. Professor click save button in file explorer window.

F.

Use case: **Professor Exports Grades**

Statement: The system shall allow a Professor to export grades.

Diagram:



**Brief Description**

The professor user has logged in and clicks the“Export Grades” button. User has already navigated to the class in which grades they want to view.

**Initial Step-By-Step Description**

1. Professor clicks export button.
2. Professor chooses export grade.
3. Professor selects assignment(s).
4. Professor clicks download.
5. File explorer opens.
6. Professor chooses the file location and name of file.
7. Professor click save button in file explorer window.

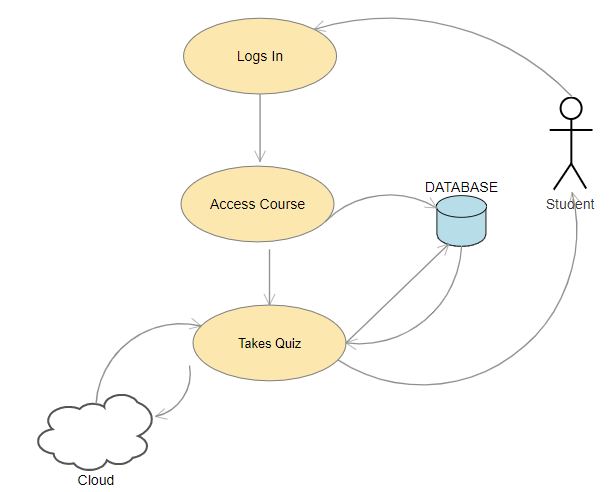
### 2.2.3 Student Use Case

A.

Use case: **Student Takes Quiz/Survey**

Statement: The system shall allow a Student to take a quiz and survey.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for Quiz/Survey.

**Initial Step-By-Step Description**

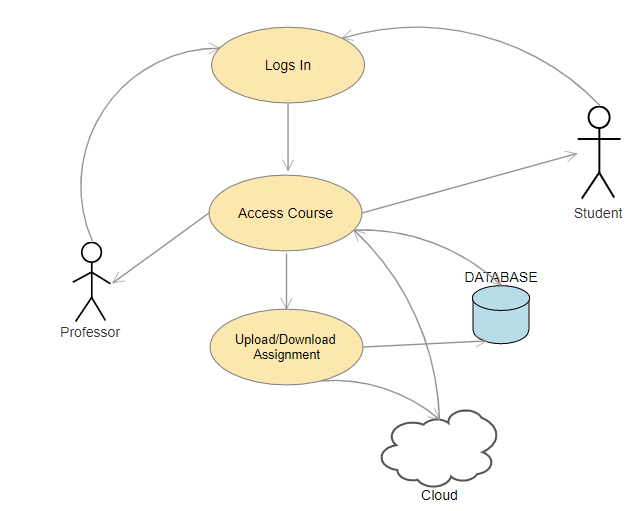
1. The Student navigates and selects the class
2. New window opens
3. Student selects the quiz/survey that they want to complete
4. The quiz/survey opens up
5. Student completes the quiz/survey
6. Student click submit button

B.

Use case: **Student Downloads Homework**

Statement: The system shall allow a Student to download a homework.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for homework.

**Initial Step-By-Step Description**

1. The Student navigates and selects the class.
2. New window opens, showing a list of the homeworks.
3. Student navigates and selects the homework file.
4. File location is sent from database
5. LMS takes address searches in dropbox and finds file
6. Student downloads homework file.

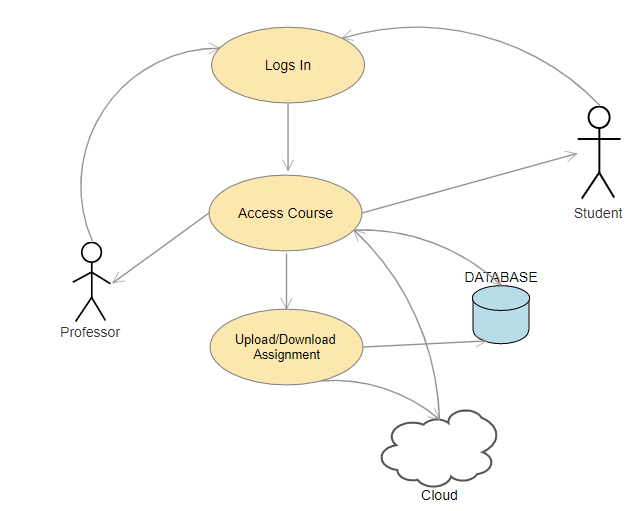
## 

C.

Use case: **Student Submits Completed Homework**

Statement: The system shall allow a Student to upload a completed homework.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for homework.

**Initial Step-By-Step Description**

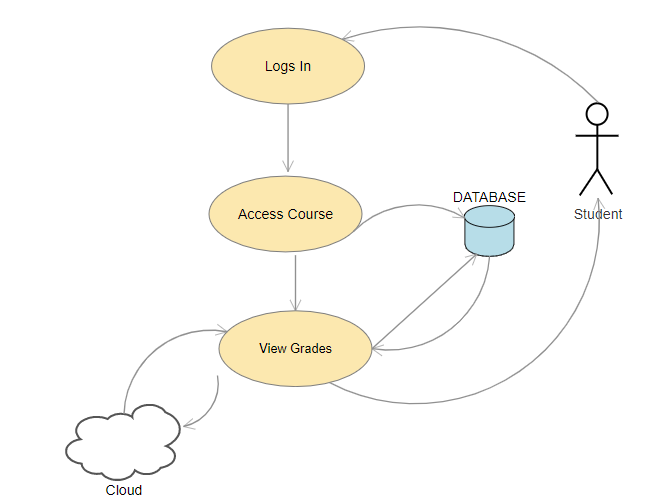
1. The Student navigates and selects the class.
2. New window opens, showing a list of the homeworks.
3. Student navigates and selects the homework file.
4. Student selects “add a file”, then “upload”.
5. Student selects their file name.
6. Student clicks add.
7. Student clicks submit.
8. Address of file is sent to database
9. File is stored in dropbox

D.

Use case: **Student View Grades**

Statement: The system shall allow a Student to view their own grades.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for viewing grades.

**Initial Step-By-Step Description**

1. The Student navigates and selects grades.
2. Database is queried for student grade report
3. New window opens, showing a list of the grades for each assignment in each class.

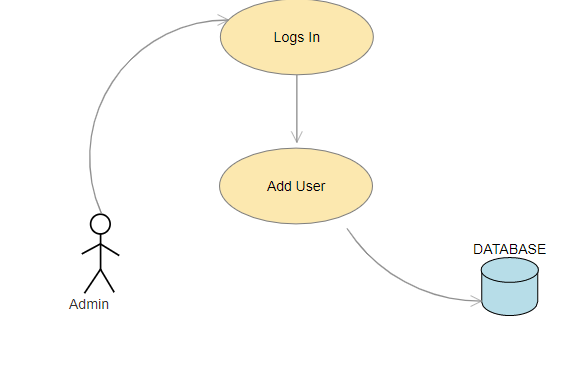
### 2.2.4 Admin Use Case

A.

Use case: **Admin adds a User**

Statement: The Admin shall be able to add a User to the database from the admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to User window.

**Initial Step-By-Step Description**

1. The Admin navigates to the user window.
2. Admin Selects “Add User”
3. Admin enters user information
4. Admin specifies User Type
5. Admin Submits User information
6. User information is sent and added to the database

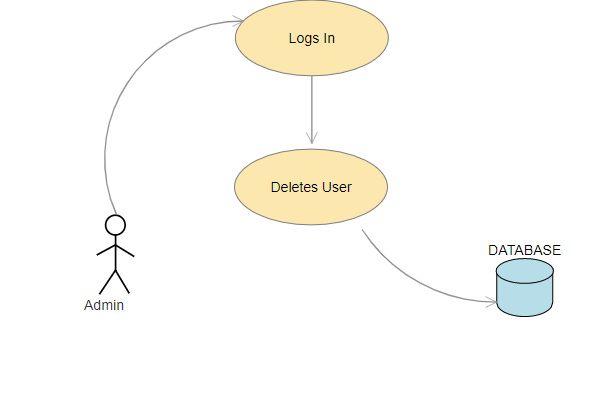
### 

B.

Use case: **Admin deletes a User**

Statement: The Admin shall be able to delete a User from the database within the admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to User window.

**Initial Step-By-Step Description**

1. The Admin navigates to the user window.
2. Admin filters by user identification
3. User appears
4. Admin clicks Delete
5. User information is deleted from the database

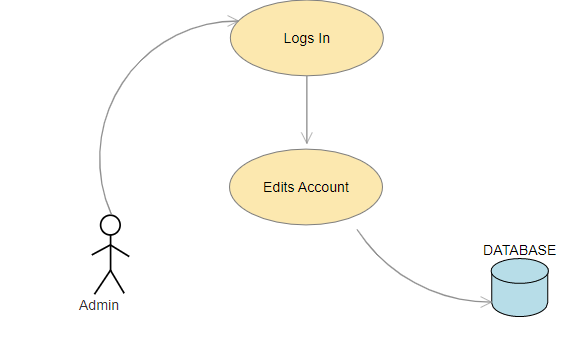
## 

C.

Use case: **Admin edits User password**

Statement: The Admin should be able to edit a User’s password within admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to User window.

**Initial Step-By-Step Description**

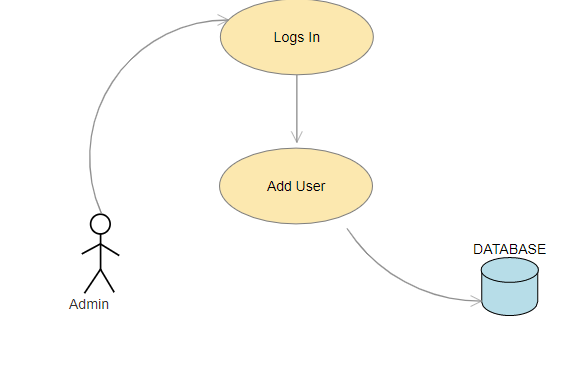
1. The Admin navigates to the user window.
2. Admin filters by user identification
3. User appears
4. Admin selects edit users information
5. Admin changes password
6. Admin clicks submit
7. Edited information is sent to database

D.

Use case: **Admin adds a User**

Statement: The Admin shall be able to add a course to the database from the admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to Course window.

**Initial Step-By-Step Description**

1. The Admin navigates to the Course window.
2. Admin Selects “Add Course”
3. Admin enters Course information
4. Admin designates professor to course
5. Admin Submits Course information
6. Course information is sent and added to the database

### 2.3 User Characteristics

Of the three types of users in this system, none of the users will be required to have an advanced understanding of technology to use the application. Each user will have different roles and responsibilities that will have to shown to them. However, any user should have a basic understanding of how to log onto a computer, get on the internet, and find the webpage.

### 2.4 Constraints

No additional constraints are needed at this time. If additional constraints are needed at a later time then they will be added to this section.

## 

### 2.5 Assumptions and Dependencies

One assumption about the product is that it will be able to import and export data from/to a different system. If the format of the files are compatible, for example a quiz uploaded being a .txt file, the file should not only be able open in the system, but also a different LMS.

Another assumption is that professors will have the ability to edit quizzes and surveys after importing them, before upload. For example, if a professor uploads a .txt file of a quiz or survey, and there is a syntax error the professor should be able to correct that error before uploading the quiz or survey for the students view.

## 

### 2.6 Apportioning of Requirements

All requirements specified in this document are to be implemented into this version of the system. If a feature is deemed at a later date to be unobtainable then Haywood Technology Consultants reserves the right to postpone the feature if it is not vital to the functionality of the system.

# 

## 3. Specific Requirements

#### 3.0.1 Profile

The profile feature will be implemented for all users. There will be an interface for users and depending upon which type of user, there will be certain privileges given within the program.

**Administrator:** Ability to create and delete Professor and Student users, add course to professor list.

**Professor:** Add course to student list, upload assignments, view/edit student grades, export grade data, and import/export assignment data.

**Student:** From this window the user will be able to see what they want to access and at the click of a button they will be able to navigate to wherever they wish within the program.

#### 3.0.2 Quizzes/Surveys

The Professor user will be able to upload quizzes and surveys. They will have the ease of creating quizzes by uploading a .txt file. Once the .txt file is uploaded, it will be read by the program and an edit window will pop up in order for the user to make sure everything uploaded properly. This same format will be allowed for surveys.

**Quiz/Test:** An assignment given to the student user where responses are graded and stored based on the specification of the professor user.

**Survey:** An assignment given to a student user that are ungraded, but responses are recorded and can be viewed against other student responses by professor users.

**Homework:** Homeworks that are stored on a third party cloud service that can be retrieved by student users.

### 

#### 3.0.3 Reports

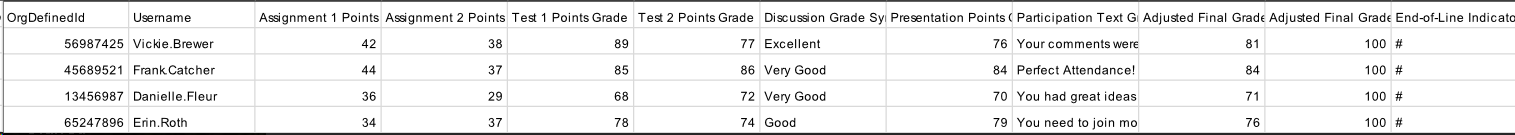
The reports feature will be implemented for students and professors. Professors will receive a response report, displaying students responses to a quiz or survey. Grade reports will be sent to professors, displaying a grade for each student that has completed said quiz. Professors will also receive live reporting from surveys. This will be a visual representation that will show in real time the student responses as they submit the survey. Students reports will consist of a list of scores from quizzes taken.

### 3.1 External Interfaces

### 

#### 3.1.1 Grade Export

The following is the format that the grades will need to be exported from the LMS Lite in order to upload into the university system. The file for this information will be a .csv file.



#### 3.1.2 Quiz/Survey Export

The following is the format that quizzes and surveys will need to be exported from the LMS Lite to upload into the university system. The file for this information will be a .txt file. MC is multiple choice. SR is short response. MA is multiple select. FIB is fill in the blank. TF is true or false. ESS is essay.

MC What is the sixth planet from the Sun? Mars Incorrect Saturn Correct Jupiter Incorrect Mercury Incorrect

SR What is the opposite of subtraction? addition

MC Which of the following is not a warm color? Red Incorrect Orange Incorrect Blue Correct Yellow Incorrect

MA Which of the following are multiples of 3? 12 Correct 27 Correct 15 Correct 31 Incorrect

FIB \_\_\_\_\_\_\_\_ is the first African-American president of the United States. Barack Obama Barack Obama

MA Which of the following are programming languages? Java Correct C++ Correct C# Correct C% Incorrect

TF The capital of Georgia is Atlanta. True

TF There are a total of 10 continents in the world. False

FIB \_\_\_\_\_\_\_\_ is the opposite of day. Night night

SR What is the first priority in the order of operations, PEMDAS? parentheses

ESS Write a response of no more than 200 words, explaining your favorite historical event. Ex: The Civil War

#### 3.1.3 Quiz/Survey Import

The following is the format that quizzes and surveys will need to be formatted to be imported into the LMS Lite. The file for this information will be a .txt file.

MC What is the sixth planet from the Sun? Mars Incorrect Saturn Correct Jupiter Incorrect Mercury Incorrect

SR What is the opposite of subtraction? addition

MC Which of the following is not a warm color? Red Incorrect Orange Incorrect Blue Correct Yellow Incorrect

MA Which of the following are multiples of 3? 12 Correct 27 Correct 15 Correct 31 Incorrect

FIB \_\_\_\_\_\_\_\_ is the first African-American president of the United States. Barack Obama Barack Obama

MA Which of the following are programming languages? Java Correct C++ Correct C# Correct C% Incorrect

TF The capital of Georgia is Atlanta. True

TF There are a total of 10 continents in the world. False

FIB \_\_\_\_\_\_\_\_ is the opposite of day. Night night

SR What is the first priority in the order of operations, PEMDAS? parentheses

ESS Write a response of no more than 200 words, explaining your favorite historical event. Ex: The Civil War

## 

### 3.2 Functions

#### 3.2.1 User Logs In

|  |  |
| --- | --- |
| **Use Case Name** | User Logs In |
| **Requirement Level** | The system shall allow a user to log in. |
| **XRef** | Section 2.2.1 User Use Case |
| **Precondition** | A User is on the Login page of the LMS system |
| **Basic Path** | User = Professor  The Professor enters his/her username.  The Professor enters his/her password.  The Professor clicks the login button. |
| **Alternative Paths** | User = Student   1. The Student enters his/her username. 2. The Student enters his/her password. 3. The Student clicks the login button. |
| **Postcondition** | The User is redirected to their specific homepage. |
| **Exception Paths** | The User may abandon the login anytime before the button is pressed. |
| **Other** | N/A |

#### 3.2.2 Professor Uploads Homework

|  |  |
| --- | --- |
| **Use Case Name** | Professor Uploads Homework |
| **Requirement Level** | The system shall allow a Professor to upload homeworks. |
| **XRef** | Section 2.2.2.A Professor Use Case |
| **Precondition** | The Professor is logged in and selected a course. |
| **Basic Path** | 1. The clicks the upload homework button. 2. File Explorer dialog box opens. 3. Professor navigates and selects the homework file. 4. Professor clicks ok button on file explorer dialog box. 5. Professor sets start and stop time/date for the assignment. 6. Professor clicks the upload button. |
| **Alternative Paths** | N/A |
| **Postcondition** | The homework is uploaded to the system. |
| **Exception Paths** | The Professor may abandon the upload anytime before the upload button is pressed. |
| **Other** | Email notification is sent to students that homework is uploaded. |

### 

#### 3.2.3 Professor Creates Quiz/Survey

|  |  |
| --- | --- |
| **Use Case Name** | Professor Creates Quiz/Survey |
| **Requirement Level** | The system shall allow a Professor to create a quiz and survey. |
| **XRef** | Section 2.2.2.B Professor Use Case |
| **Precondition** | The Professor is logged in and selected a course. |
| **Basic Path** | 1. Professor selects create quiz/survey. 2. New pane/window opens. 3. Professor enters quiz/survey name. 4. Professor enters the start and end time/date. 5. Professor builds quiz/survey. 6. Professor clicks review. 7. Professor click upload. |
| **Alternative Paths** | The Professor clicks review and finds an error. Professor clicks edit, makes the change, and clicks review again. |
| **Postcondition** | The quiz/survey is uploaded. |
| **Exception Paths** | The Professor may abandon the upload anytime before the upload button is pressed. |
| **Other** | Point per question is defaulted to 1.  Email notification is sent to students that homework is uploaded. |

#### 3.2.4 Professor Imports Quiz/Survey

|  |  |
| --- | --- |
| **Use Case Name** | Professor Imports Quiz/Survey |
| **Requirement Level** | The system shall allow a Professor to import a quiz and survey. |
| **XRef** | Section 2.2.2.C Professor Use Case |
| **Precondition** | The Professor is logged in and selected a course. |
| **Basic Path** | 1. Professor selects import quiz/survey. 2. File Explorer dialog box opens. 3. Professor navigates and selects the quiz/survey file. 4. Professor clicks ok button on file explorer dialog box. 5. Professor is able to view the assignment and add points to each question 6. Professor sets start and stop time/date for the assignment. 7. Professor clicks the upload button. |
| **Alternative Paths** | N/A |
| **Postcondition** | The quiz/survey is uploaded. |
| **Exception Paths** | The Professor may abandon the upload anytime before the upload button is pressed. |
| **Other** | Point per question is defaulted to 1.  Email notification is sent to students that homework is uploaded. |

### 

#### 3.2.5 Professor Views Students Grades

|  |  |
| --- | --- |
| **Use Case Name** | Professor Views Students Grades |
| **Requirement Level** | The system shall allow a Professor to view students grades. |
| **XRef** | Section 2.2.2.D Professor Use Case |
| **Precondition** | The Professor is logged in and selected a course. |
| **Basic Path** | 1. Professor click view grades. 2. Grade book is displayed with list of assignments. 3. Specific assignment is clicked and breakdown of student grades is displayed. |
| **Alternative Paths** | N/A |
| **Postcondition** | Grades are viewed by Professor. |
| **Exception Paths** | N/A |
| **Other** | N/A |

#### 3.2.6 Professor Exports Quiz/Survey

|  |  |
| --- | --- |
| **Use Case Name** | Professor Exports Quiz/Survey |
| **Requirement Level** | The system shall allow a Professor export a quiz and survey. |
| **XRef** | Section 2.2.2.E Professor Use Case |
| **Precondition** | The Professor is logged in and selected a course. |
| **Basic Path** | 1. Professor clicks export button. 2. Professor chooses export quiz/survey. 3. Professor selects quiz/survey. 4. Professor clicks download. 5. File explorer opens. 6. Professor chooses the file location and name of file. 7. Professor click save button in file explorer window. |
| **Alternative Paths** | N/A |
| **Postcondition** | Professor can use saved file. |
| **Exception Paths** | The Professor may abandon the export anytime before the save button is pressed. |
| **Other** | File is a .txt |

### 

#### 3.2.7 Professor Exports Grades

|  |  |
| --- | --- |
| **Use Case Name** | Professor Exports grades |
| **Requirement Level** | The system shall allow a Professor to export grades. |
| **XRef** | Section 2.2.2.F Professor Use Case |
| **Precondition** | The Professor is logged in and selected a course. |
| **Basic Path** | 1. Professor clicks export button. 2. Professor chooses export grade. 3. Professor selects assignment(s). 4. Professor clicks download. 5. File explorer opens. 6. Professor chooses the file location and name of file. 7. Professor click save button in file explorer window. |
| **Alternative Paths** | N/A |
| **Postcondition** | Professor can use saved file. |
| **Exception Paths** | The Professor may abandon the export anytime before the save button is pressed. |
| **Other** | File is a .csv |

#### 3.2.8 Student Takes Quiz/Survey

|  |  |
| --- | --- |
| **Use Case Name** | Student Takes Quiz/Survey |
| **Requirement Level** | The system shall allow a Student to take a quiz and survey. |
| **XRef** | Section 2.2.3.A Student Use Case |
| **Precondition** | The Student is logged in and selected a course. |
| **Basic Path** | 1. The Student navigates and selects the class 2. New window opens 3. Student selects the quiz/survey that they want to complete 4. The quiz/survey opens up 5. Student completes the quiz/survey 6. Student click submit button |
| **Alternative Paths** | The student abandons the quiz/survey for a period of time, then comes back to it at a later time. They then complete it and press submit. |
| **Postcondition** | The Quiz/Survey is submitted |
| **Exception Paths** | The due time expires before the student has completed the quiz/survey, and the quiz automatically submits the answers completed. |
| **Other** | N/A |

### 

#### 3.2.9 Student Downloads Homework

|  |  |
| --- | --- |
| **Use Case Name** | Student Downloads Homework |
| **Requirement Level** | The system shall allow a Student to download a homework. |
| **XRef** | Section 2.2.3.B Student Use Case |
| **Precondition** | The Student is logged in and selected a course. |
| **Basic Path** | 1. The Student navigates and selects the class.  2. New window opens, showing a list of the homeworks.  3. Student navigates and selects the homework file.  4. Student downloads homework file. |
| **Alternative Paths** | N/A |
| **Postcondition** | The Homework is downloaded. |
| **Exception Paths** | The due date has passed and the student is unable to open the homework file. |
| **Other** | N/A |

#### 3.2.10 Student Submits Completed Homework

|  |  |
| --- | --- |
| **Use Case Name** | Student Submits Completed Homework |
| **Requirement Level** | The system shall allow a Student to upload completed homework. |
| **XRef** | Section 2.2.3.C Student Use Case |
| **Precondition** | The Student is logged in and selected a course. |
| **Basic Path** | 1. The Student navigates and selects the class.  2. New window opens, showing a list of the homeworks.  3. Student navigates and selects the homework file.  4. Student selects “add a file”, then “upload”.  5. Student selects their file name.  6. Student clicks add.  7. Student clicks submit. |
| **Alternative Paths** | N/A |
| **Postcondition** | The Homework is submitted. |
| **Exception Paths** | The due date has passed and student is unable to submit their homework file. |
| **Other** | N/A |

## 

### 

#### 3.2.11 Student Views Their Grades

|  |  |
| --- | --- |
| **Use Case Name** | Student Views Their Grades |
| **Requirement Level** | The system shall allow a Student to view their own grades. |
| **XRef** | Section 2.2.3.D Student Use Case |
| **Precondition** | The Student is logged in and selected a course. |
| **Basic Path** | 1. The Student navigates and selects grades.  2. New window opens, showing a list of the grades for each assignment in each class. |
| **Alternative Paths** | N/A |
| **Postcondition** | The Homework is downloaded. |
| **Exception Paths** | The due date has passed and the student is unable to open the homework file. |
| **Other** | Email notification is sent to students of grades for each assignment. |

#### 3.2.12 Admin adds a User

|  |  |
| --- | --- |
| **Use Case Name** | Administrator adds User to system |
| **Requirement Level** | The system shall allow an Administrator to add a user to the database from the interface |
| **XRef** | Section 2.2.4.A Administrator Use Case |
| **Precondition** | The Administrator is logged in and have administrative rights. |
| **Basic Path** | 1. The Administrator navigates to the user window.  2. Admin selects “Add User”.  3. Admin enters user information.  4. Admin specifies the user type (professor or student).  5. Admin submits user information.  6. User information is sent and added to the database. |
| **Alternative Paths** | N/A |
| **Postcondition** | New user is added to the system. |
| **Exception Paths** | The administrator does not have admin rights to add a user. |
| **Other** | N/A |

#### 3.2.13 Admin deletes a User

|  |  |
| --- | --- |
| **Use Case Name** | Administrator deletes User to system |
| **Requirement Level** | The system shall allow an Administrator to delete a user from the database from the interface |
| **XRef** | Section 2.2.4.B Administrator Use Case |
| **Precondition** | The Administrator is logged in and have administrative rights. |
| **Basic Path** | 1. The Admin navigates to the user window.  2. Admin filters by user identification  3. User appears  4. Admin clicks Delete  5. User information is deleted from the database |
| **Alternative Paths** | N/A |
| **Postcondition** | A user is deleted to the system. |
| **Exception Paths** | The administrator does not have admin rights to add a user. |
| **Other** | N/A |

#### 3.2.14 Admin edits a users’ password

|  |  |
| --- | --- |
| **Use Case Name** | Administrator edits a users’ password |
| **Requirement Level** | The system shall allow an Administrator to edit/change a users’ password, per request. |
| **XRef** | Section 2.2.4.C Administrator Use Case |
| **Precondition** | The Administrator is logged in and have administrative rights. |
| **Basic Path** | 1. The Admin navigates to the user window.  2. Admin filters by user identification  3. User appears  4. Admin selects edit users information  5. Admin changes password  6. Admin clicks submit  7. Edited information is sent to the database |
| **Alternative Paths** | N/A |
| **Postcondition** | The users password is changed. |
| **Exception Paths** | The administrator does not have admin rights to add a user. |
| **Other** | N/A |

#### 3.2.15 Admin adds a new course

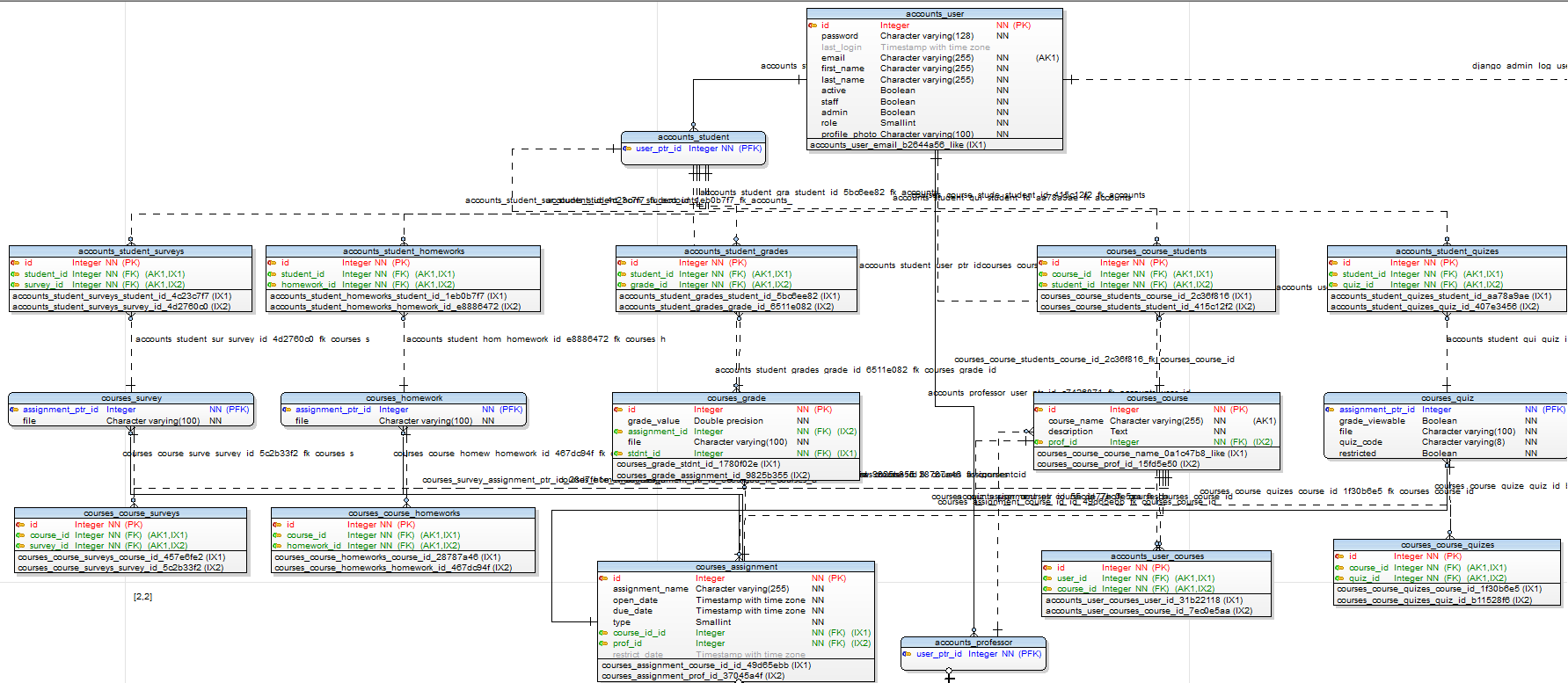
|  |  |
| --- | --- |
| **Use Case Name** | Administrator adds a new course |
| **Requirement Level** | The system should allow an Administrator to add a course to the database. |
| **XRef** | Section 2.2.4.D Administrator Use Case |
| **Precondition** | The Administrator is logged in and have administrative rights. |
| **Basic Path** | 1. The Admin navigates to the Course window.  2. Admin Selects “Add Course”  3. Admin enters Course information  4. Admin designates professor to course  5. Admin Submits Course information  6. Course information is sent and added to the database |
| **Alternative Paths** | N/A |
| **Postcondition** | New course is added to the system |
| **Exception Paths** | The administrator does not have admin rights to add a user. |
| **Other** | N/A |

### 3.3 Performance Requirements

Performance requirements are not a concern for this system. Due to the fact that the LMS is solely ran on web browsers it will be up to the client to ensure that the university and web service provider is responsible for the speed and reliability of the connection.

### 3.4 Logical Database Requirements

The database collects the following information users, grades, assignment info, and Django system information.



### 3.5 Design Constraints

There are no design constraints for this system aside from time. Other standards nor hardware limitations are a constraint for this project. If any constraints arise each one will be analyzed and necessary action may be taken.

#### 3.5.1 Standards Compliance

The standards provided for the system are covered in section 3.1 regarding formats for grades and quizzes and surveys being extracted for import into the university system and quizzes and surveys being imported into the LMS Lite system.

### 3.6 Software System Attributes

In this sections various requirements and attributes will be discussed. In order to get a more in depth understanding of the software overview.

#### 3.6.1 Reliability

The system must be running current version of Google Chrome and allow there must be some sort of internet connection. There is no MTBF requirement for the LMS lite. The software chosen for the LMS to run on is reliable and there should be no issues related to reliability.

#### 3.6.2 Availability

The LMS lite will be available 24/7 because it is a web based program. This means that upon server restart if information is being uploaded to the servers then there could be potential for loss of information.

#### 3.6.3 Security

##### 3.6.3.1 Password Encryption

**SHA256:** Per the software framework used, the user password are defaultly encrypted using SHA256 encryption. Using SHA hashing functions is the security standard of SSL certificates

##### 3.6.3.2 xAPI IP Restriction

**xAPI IP Restriction: Professors will have the opportunity to restrict a quiz/survey to being only taken on campus.**

##### 3.6.3.3 CSRF Token

**CSRF Token:** Due to the use of the django framework a CSRF Token is given to login sessions so that an https connection can be established between the user and the server. This prevents attackers from being able to send from a malicious site a request that is not authenticated from the server.

#### 3.6.4 Maintainability

There are different features in this system that allow for maintainability. The program will be implemented using xAPI which has built in features and a set format that creates a maintainable environment. xAPI allows for a uniform code structure that many modern LMS programs follow, this alone allows for anyone to change or tweak based on needs if they research features of xAPI Also the project will have multiple user profiles which by nature allows for constant supervision of program.

#### 3.6.5 Portability

Portability is not a concern because the LMS will be ran on a web browser and any host-machine with an internet connection and Google Chrome should be able to use the LMS.

### 3.7 Organizing the Specific Requirements

This sections was not used because section 3.2 covers all relative information corresponding to this section. If any additional information is to be required this section may be used at a future point.

### 3.8 Additional Comments

No additional comments are needed at this time. If additional comments are needed at a later time then they will be added to this section.

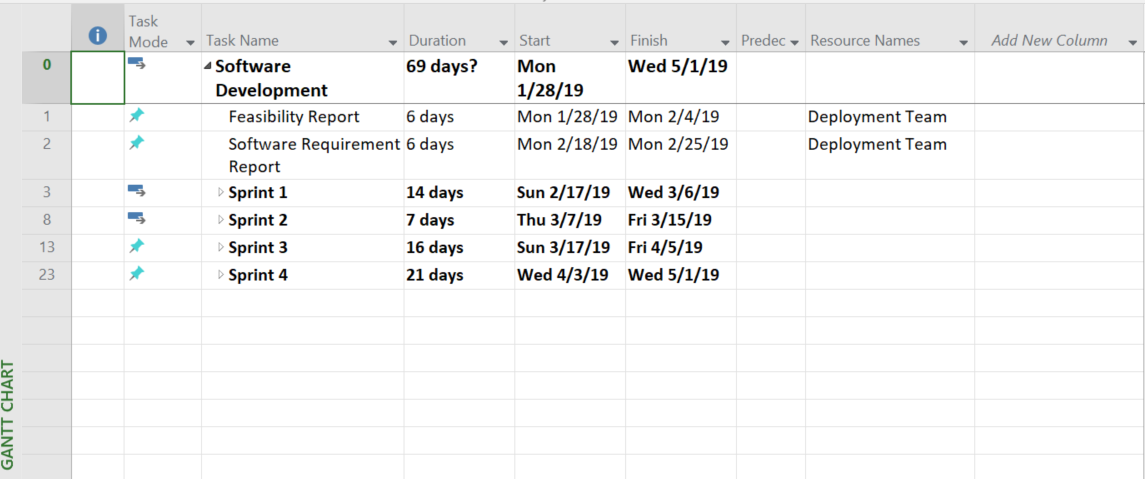
## 4. Change Management Process

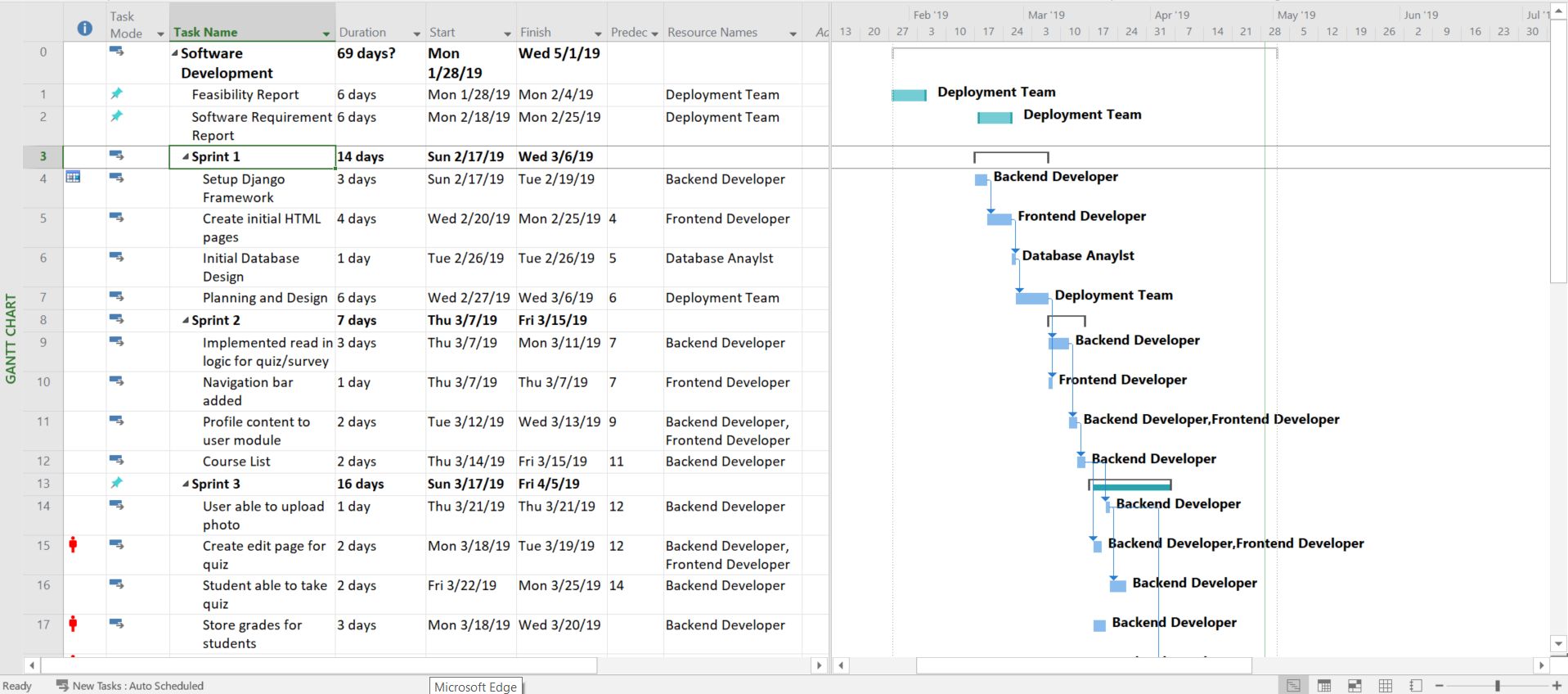
Changes will need the approval of the project manager, Kym Haywood II. All changes need a majority consensus among the core project team. Any change will be analyzed by the core project team and seen if it will be feasible during the duration of the project. Haywood Technology Consultants reserve the right to deny a change if the change will take more time than given in the project deadline. When a change is made, the change will be updated in this document with the date/time beside the change (Ex. 3:00 pm 2/24/19). This will provide a thorough revision history with the use of Google Drive track changes.

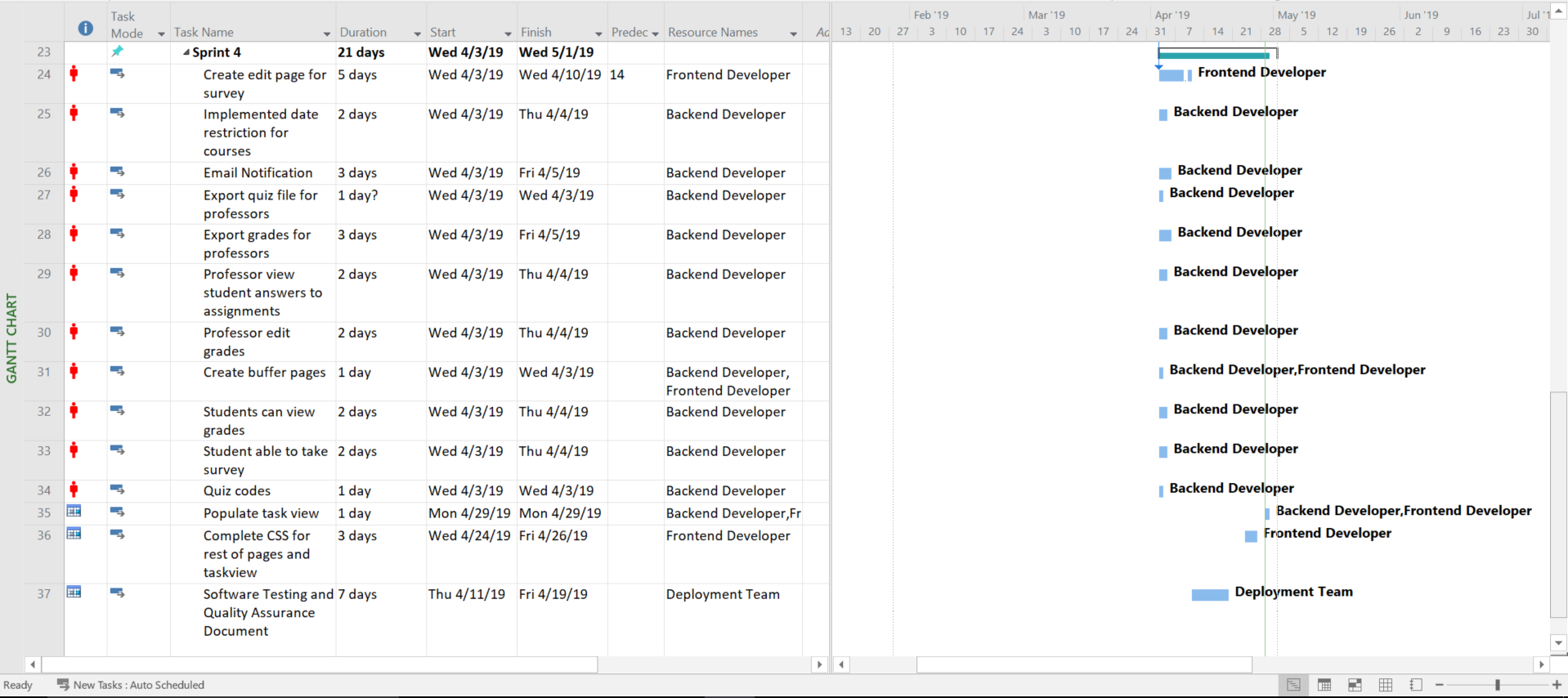
Open Learning Management System Lite:

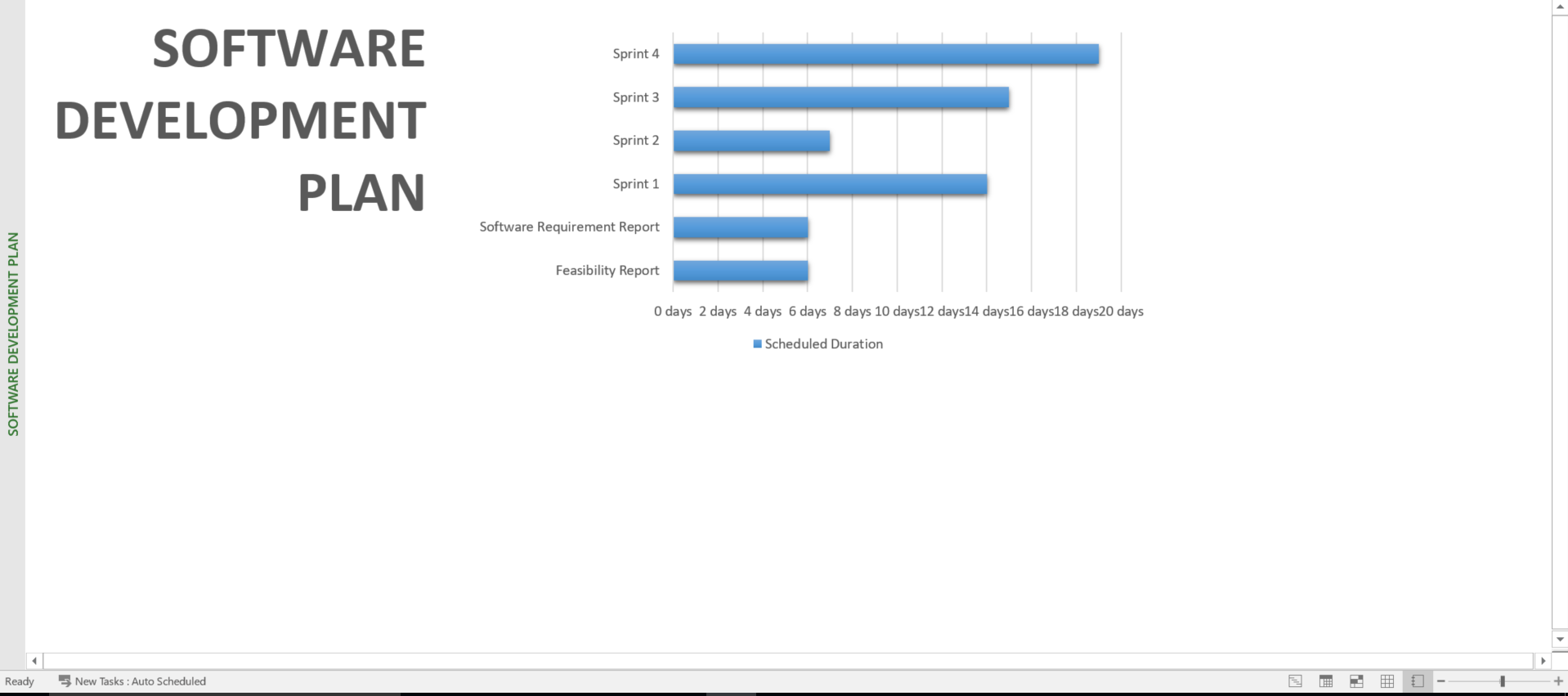
# Project Schedule

## Gantt Chart



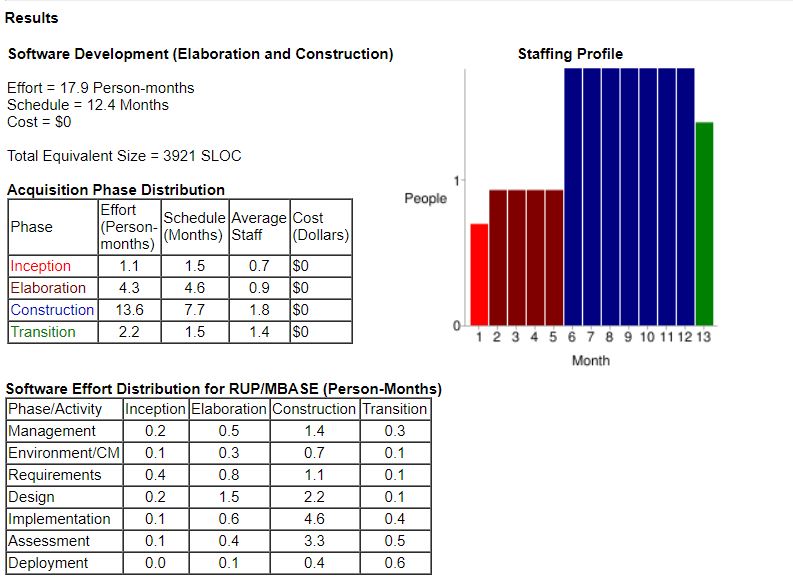






## LOC Estimation

## 



Open Learning Management System Lite:

# Design Document

## Abstract

Educators and students alike run into similar problems with tracking course work as well as maintaining an organized base of information. As such problems have become even more prevalent with the advancement in the education field there has a need for a simple way to keep track of class servicing both educators and students. This is where the Open LMS Lite helps alleviate such issues that arise by enabling a productive, organized, and user-friendly work environment. The Open LMS Lite offers an efficient way to deal with course loads enabling students to further their education in the most productive way as possible by given them access to a easy to use platform that both teachers and students are able to navigate. As with ease of use the Open LMS Lite allows for students to stay on top of their grades by storing grades of assignments that are taken on the system. This document discusses the design considerations of the Open LMS Lite application.

## Introduction

The Open LMS Lite propose is to help students and teachers maintain a productive work space by removing issues with organization and user-friendliness. To achieve such a goal, this chapter will clearly define the purpose of the Open LMS Lite system by defining the functional and nonfunctional requirements of the software, introduce common jargon, and present a general overview of this document. The purpose of the system is to offer teachers and students an all in one user friendly application to keep track of their courses as well as do assignments related to those courses. With this in mind, The Open LMS Lite application is designed in such a way to mitigate issues with accessibility as it is web based and requires minimal base requirements to run. To better understand the Open LMS Lite there will be a walkthrough of the basic thoughts and possibilities involved in making such system. This walkthrough will cover the proposed software architecture, object design, glossary, and the appendix.

### 

### 1.1 Purpose of System

The reason the Open LMS Lite was created was to allow teachers and students to maintain in a productive workspace while still being able to do assignments and keep the information for all of their course organized in one consolidated application. This is important to keep in mind, the Open LMS Lite focus is to create a platform to be easy to use and an one stop for all of your class material. As technology has advanced a lot of course worked has tried to move online. Although this may seem like a great idea it comes with its drawbacks. One of these drawbacks causes a situation making it extremely hard for students to keep all their courses organized. This is that all teachers don’t use the same software to conduct their courses. The Open LMS Lite was made to fix this drawback by allowing for all teachers to use this software to administer their course work to students. As well other drawbacks occur such as ease of use for students and teachers. Open LMS Lite fixes this by making the application easy for users to understand while still being able to maintain the features that allow for a productive learning environment.

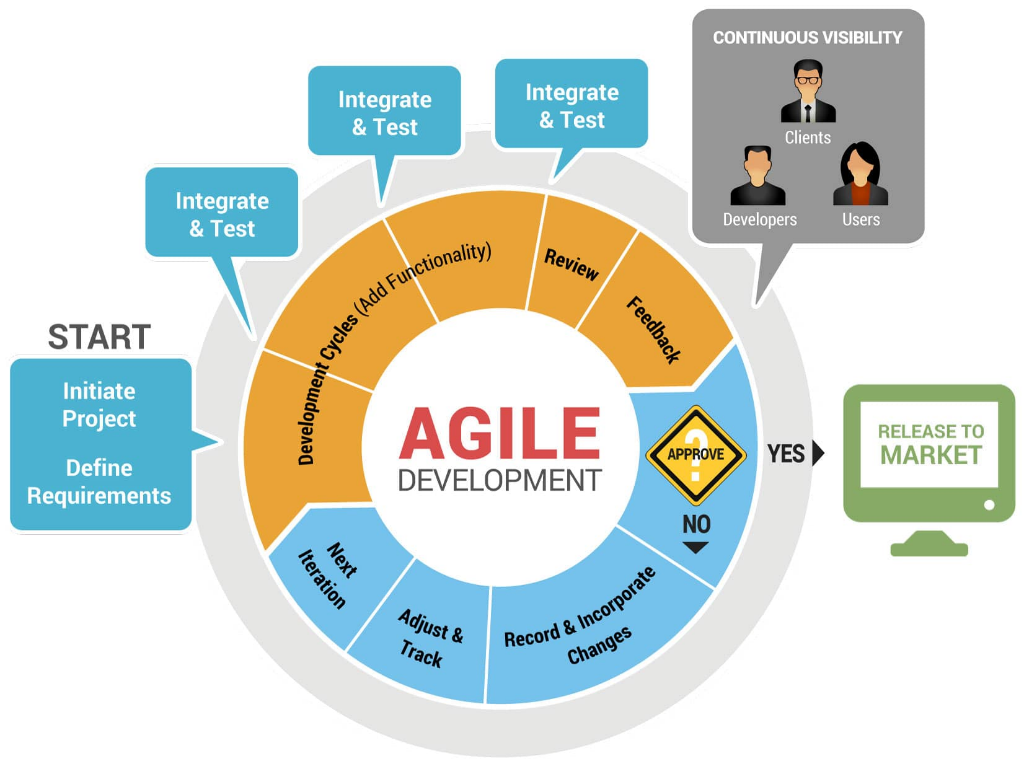
### 1.2 Functional and Nonfunctional Requirements

The system shall allow students to be to keep track of all of their coursework while keeping the application as easy to use as possible. The system shall also allow for students to upload and download documents as well as take quizzes on the application. The constraint for these requirements is that the student must be enrolled in a course that uses the Open LMS Lite..(see appendix B)

The system will allow for teachers to distribute assignments as well as due dates. The teacher will also be able to grade and update student profiles with posted grades. The system will also keep up with the classes grades for assignments for the teacher to view. The system shall also allow for teachers to upload and download documents as well as administer quizzes on the application. The constraint for these requirements is that a student must be enrolled in the course for the use case to complete. (see appendix B)

The system shall allow for students and teachers to receive email notification about upcoming assignments in order to remind them about what has to be done as well as show a task view in the application for both teachers and student. The constraint for these requirements is that a student must be enrolled in the course for the use case to complete. (see appendix B)

### 1.3 Design Methodology

The Open LMS Lite was designed with the Agile Software Design method. Using the Agile design method allowed for a productive methodical approach to the creation of the application. Agile allowed for the team to asses the situation and approach accordingly. Many of the issues that arose during the creation of the application were able to be moved to different sprints enabling the team to continue the production of the Open LMS Lite. This design was chosen because of this allowing for all group members to meet and stay on the same page moving forward to create the application. This design also allowed us to stay closer in touch with the client making the creation much easier then using other methods that would distant the client from the production of the application. 

### 1.4 Definitions, acronyms, and abbreviations

***LMS***: A software application for the administration, documentation, tracking, reporting and delivery of educational courses, training programs, or learning and development programs

***STMP***: Simple Mail Transfer Protocol is an Internet standard for email transmission.

***SQL***: Standardized query language for requesting information from a database

***xAPI***: An e-learning software specification that allows learning content and systems to speak to each other in a manner that records and tracks all types of learning experiences.

### 1.5 Overview

In later chapters this document will go more in depth giving more information with the process of creating the Open LMS Lite application. Chapter 2 will discuss the architecture of the systems giving a more in depth view of how the system operates and how it functions. Chapter 3 will be about the object design of the application.

## 

## Proposed Software Architecture

This chapter will discuss the overall design of the Open LMS Lite application. It will go in depth and explain the design of the application and what handles certain functionality of the system. The purpose of this chapter is so that the reader can better understand the system in terms of how it works and how it is designed.

### 2.1 Overview

The Open LMS Lite uses Django a web framework software to help with development and the application architecture. This ensured rapid development as well as clean organized code. Django takes care of all the structure of the application as well as process data that is pushed through the application. Django is a high- level Python web framework hence the application being written in python. Open LMS Lite is constructed with three different packages, and it uses the model-view-controller. Further in this chapter those packages will be discussed.

### 2.2 Subsystems

Within the Open LMS software there exists two major subsystems: Courses and Accounts. The Accounts system is used mainly for user authentication and sessions. Uses of this system entail; users logging in and out of the software, as well as serving the correct data to each user that is logged in. Within accounts there are three user types: Admin, users that can alter the models and overall structure of the data stored. Professors, users that can alter the course model by either adding assignments to them or removing them. Professor users also have the ability to alter the grade models for students. Students, who have the ability to only alter models that contain references to their primary keys.

The Course subsystem handles any course data that is held to be served to users. This includes grade data for the students. References to the quiz, and survey data. From a hierarchical standpoint the Course subsystem is the crux of the software that give it the most functionality.

### 2.3 UML Profiles

The Open LMS Lite system has two primary UML profiles Courses and Accounts that can be displayed below.

**Figure 2.3-1: UML Profile for Courses package.**

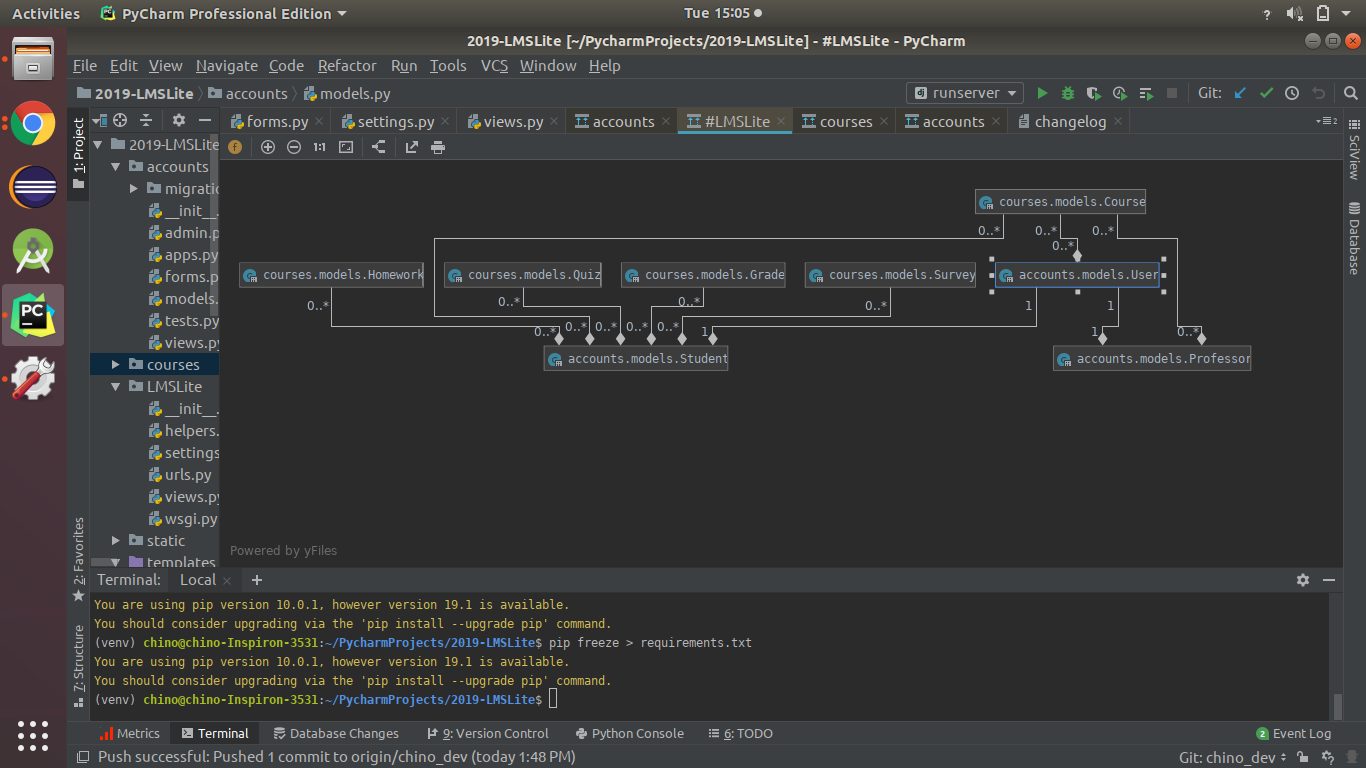
****

Fig 2.3-1 consists of the Courses UML in the Open LMS Lite system. Django has an auto populate feature that allows for views of a project in a UML diagram which is seen above.

**Figure 2.3-2: UML Profile for Accounts package**

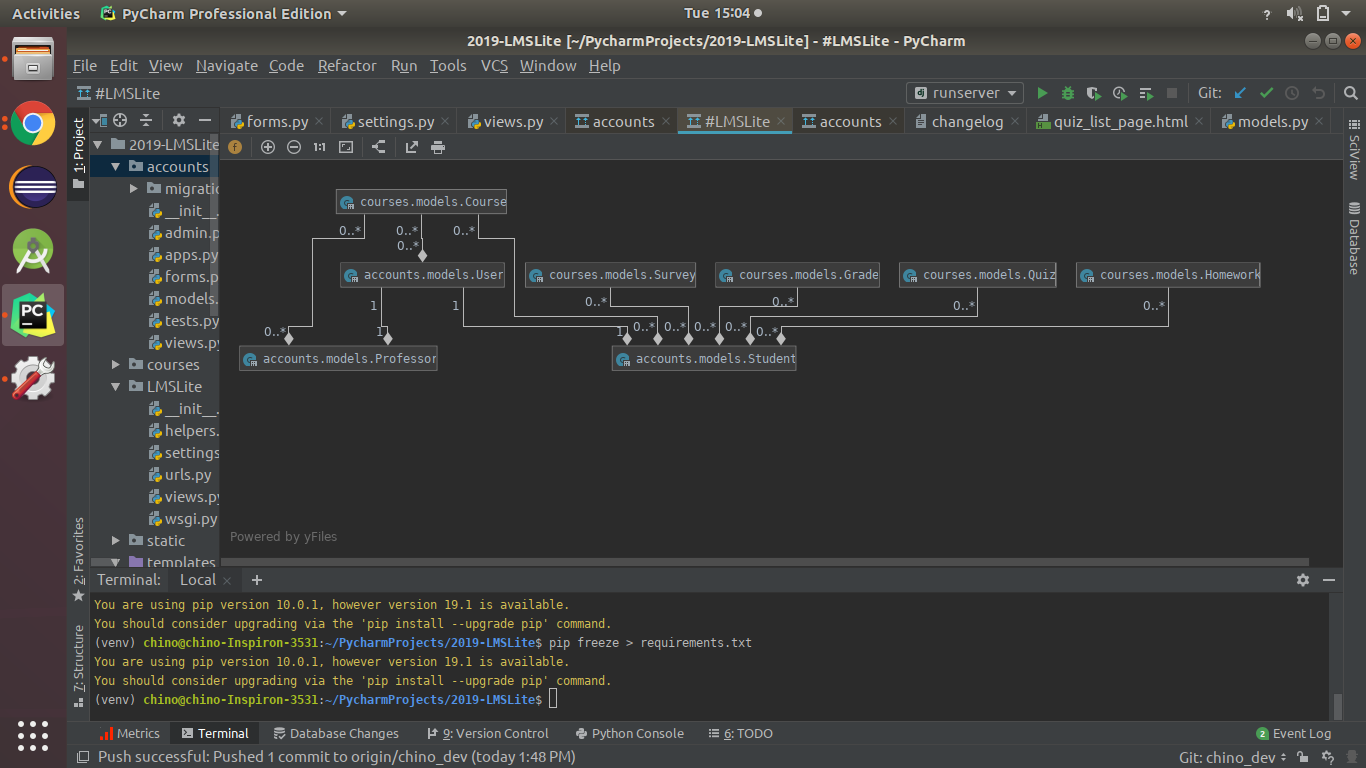


Fig 2.3-2 shows the UML Profile for the Account Package of the Open LMS Lite.

### 2.4 Subsystem Decomposition

The Open LMS Lite system is decomposed into three packages. As previously described, these packages are the Accounts and Courses. The first access point is the Account package which is where the base HTML and login page is populated. The second access point is also the Account package which communicates with the Courses package and the database to populate data for the Student or Professor that is accessing the Course package. The third access point is where the Course package updates or retrieves information from the database to push to the account package and populate information for the student or professor account that is accessing the courses package.

## Object Design

In this chapter the object design of the application will be shown. A proper design of objects makes a application more sustainable and overall an easier to maintain system. This chapter will try to help the reader better understand the design of the application and give an overview that will sum up what was discussed in the previous chapters.

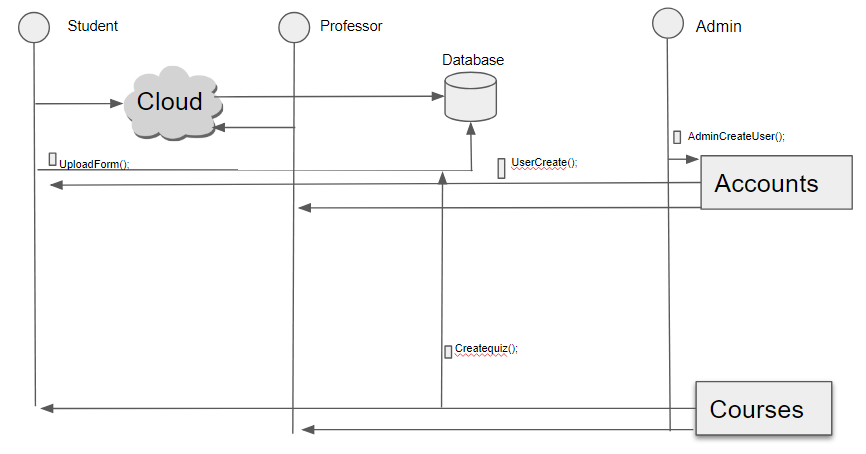
### 3.1 Overview

The Open LMS Lite has for main packages that function together to give you the whole application. The packages are as such Courses and Accounts. All of the course information including quizzes, uploading and downloading .docs , and task view is managed by the Courses package which communicates with the database which is generated via Django through the model.py file within package. The Accounts package handles all of the data created to from the user’s profile, as well as the grades for each student and communicates with the database that is generated via Django through the model.py .

### 

### 3.2 Object Interaction

The Open LMS Lite’s three packages communicate with each other although they all have their own specific job in order to allow for the application to run properly. The diagram below shows a visual representation of how they communicate with each other. The Accounts handles all of the information that goes on the profile for each user which is a link to the Courses package which allows for the Account package to know exactly who to assign things. The sequence diagram shows some of the methods involved with the interactions between the classes.



### 3.3 Detailed Class Design

In this section, the detailed class design will be discussed. The purpose of each class in each package will be explained, and its appropriate class diagram and code will be referenced in Appendix C and Appendix D respectively.

### 

### 3.3.1 Courses Package

The Courses package is in charge of everything involved with course information. This package takes care of assignments, notifications, due dates, task view. This package is in place in order to keep everything as organized as possible while allowing students to be able to work on course work and for professors to assign course work to people that are in their classes. The package also communicates with the database through the model.py class where it updates information for the Accounts package.

### 3.3.2 Accounts Package

The Accounts package function is to keep track of all the users in the system. Its job is to create users and hold information about that user as well as populate the profile with information that is stored about the user. This includes Name, Student ID, grades for courses. All of this is done by the package communicating with the database that is generated via Django through the model.py.

Open Learning Management System Lite:

# Testing Document

## Introduction

### 1.1. Test Plan Objectives

The objective of the testing that was done on the Open LMS Lite system was to insure that the application operated and functioned in a way that it could be released to the public being that it was open sourced. The main focus of the testing of the application was the special functionality that was added to the Open LMS Lite system for example email notification, proper login measures, and upload/download files. The Open LMS Lite was tested mainly using automated testing. Automated testing worked best for testing this application because of the software design technique that was used to develop the application. Agile allow for multiple sprints where focus could be placed on testing features that had been completed while still working to complete other features. Also due to the application being web based there were many tools that were vetted to choose from to test the application. The application was mainly tested using Selenium which will be discussed further in the document.

### 1.2 Scope/Plan

The team decided that most of the testing will be automated testing and that the easiest way to accomplish this would be by creating code to test after functions were in place. In order to do this in a timely manner upon completion of a function and implementation it was test at a team meeting. The schedule later in the document will give a better understanding of when the testing took place but this was the pattern that we used to do all of the testing for the Open LMS Lite system.

## Test Strategy

This section will go more in depth showing a few test cases with actual code and screenshots of the actual test case being performed on the Open LMS Lite system. Below are examples of different test strategies that we used.

### 2.1. Data Entry

The Open LMS Lite system uses a database to store data that is collected by users when an account is created. In order to test this test cases were created to see that information was properly placed inside of the database. The data was then confirmed by team member by checking the database and also looking at the webpage for both professor and student.

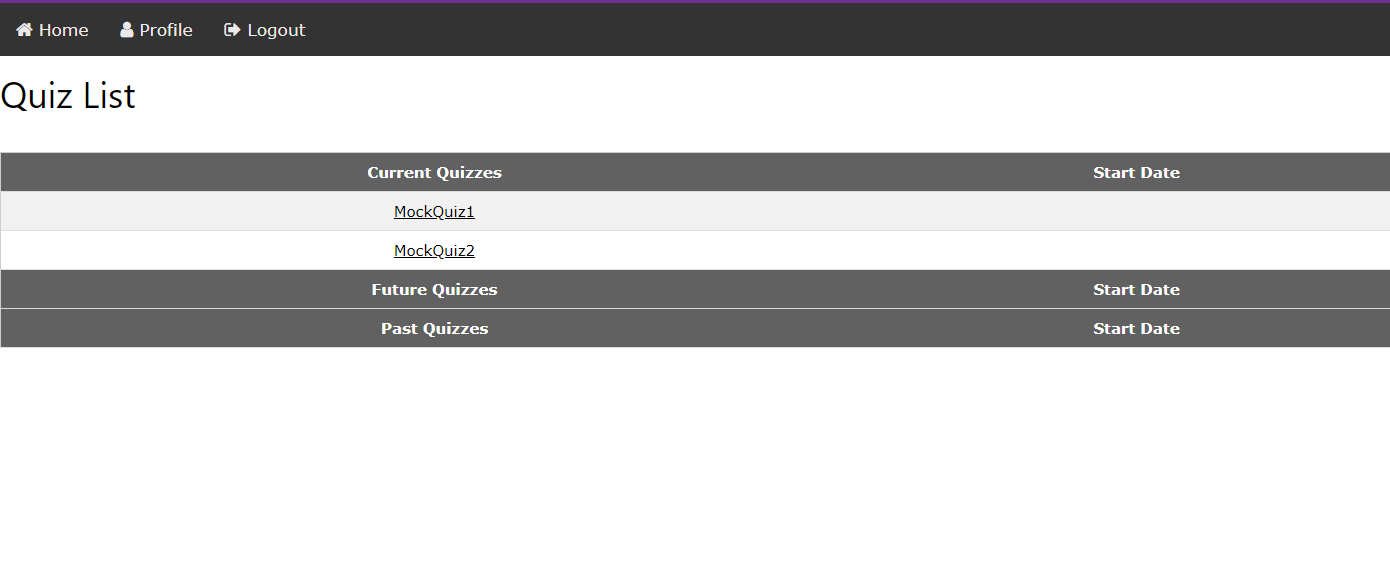
### 2.2. Performance Test

In order to test the performance of the Open LMS Lite system test cases were made to repeatedly add quizzes to make sure that users were able to perform the task multiple times without error. This is also our stress test due to the built in web features that add stress protection to the system itself.

**Scope :** To Insure that system can withstand small to moderate user errors without any errors of system failures.

**Test Case 1**

**Senario:** Professor makes the same quiz twice, and continues to make more quizzes.

1: Professor pulls up webpage

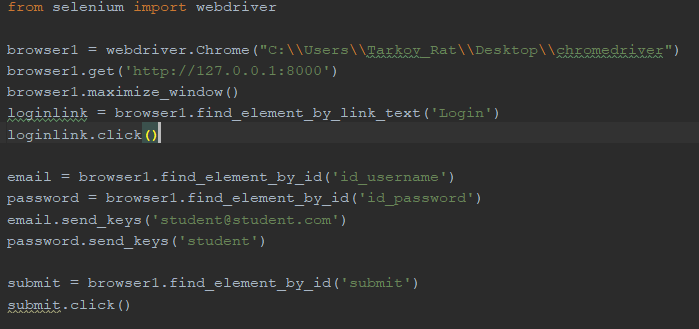
2: Professor goes to quiz page.

3: Professor creates a quiz

4: Professor submits quiz

5: Professor creates another quiz.

### 2.3. Security Test

****

In order to test security of the Open LMS Lite system the following test cases were made to ensure that the login screen would not allow for the users to access website without having a correct username and login.

**Scope :** To Insure that system has enough integrity to ensure that users are using a legitimate system while protecting their data.

**Test Case 1**

**Senario:** Student opens the website and logs in using the incorrect user name and password.

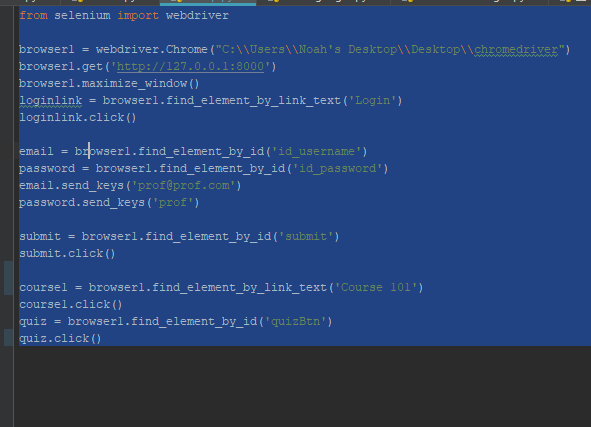
1: Student pulls up webpage

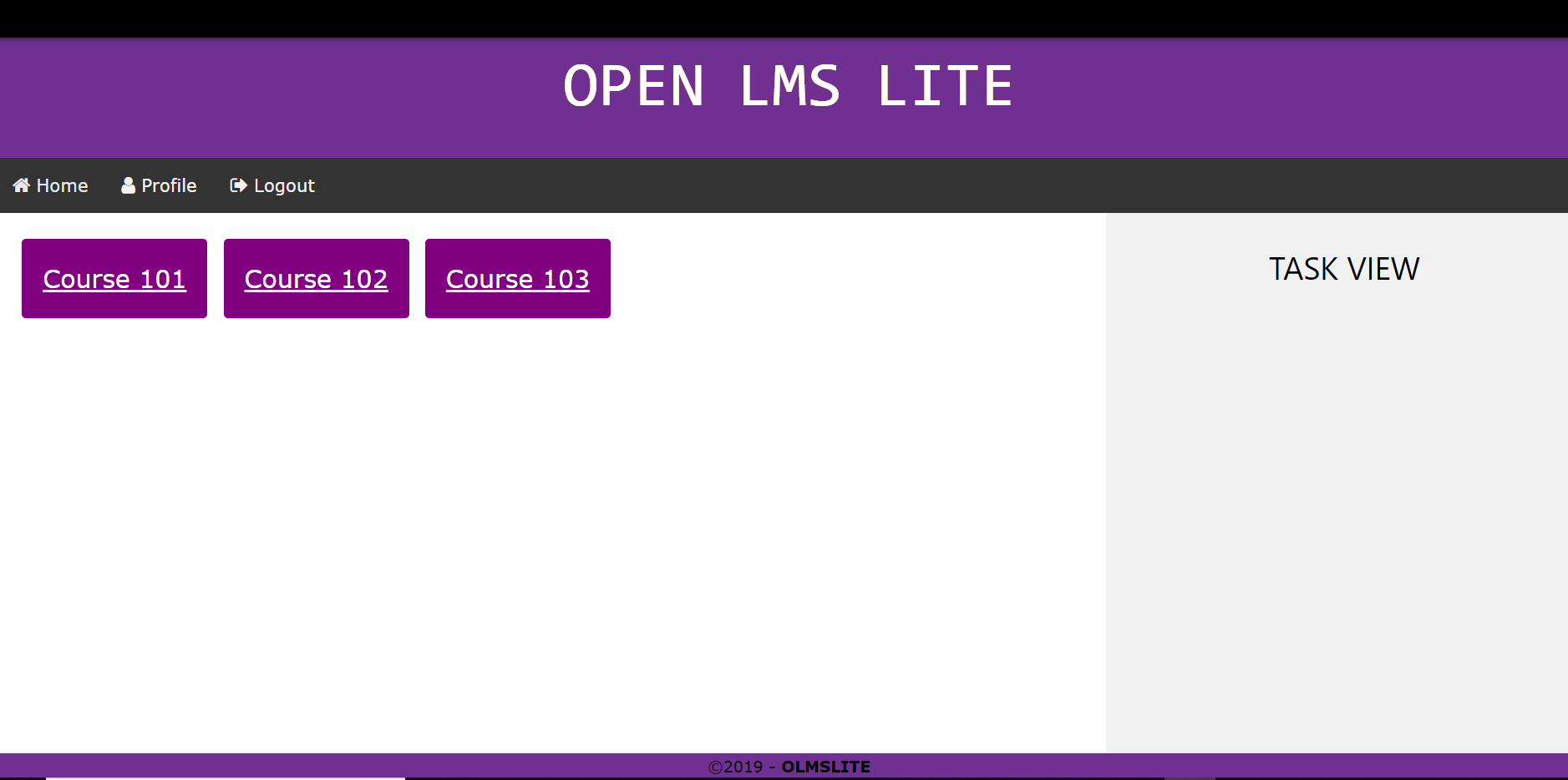
2: Student clicks login link.

3: Student enter credentials

4: Student clicks login button

5: Error page is displayed.





**Test Case 2**

**Senario:** Professor opens the website and logs in using the incorrect user name and password.

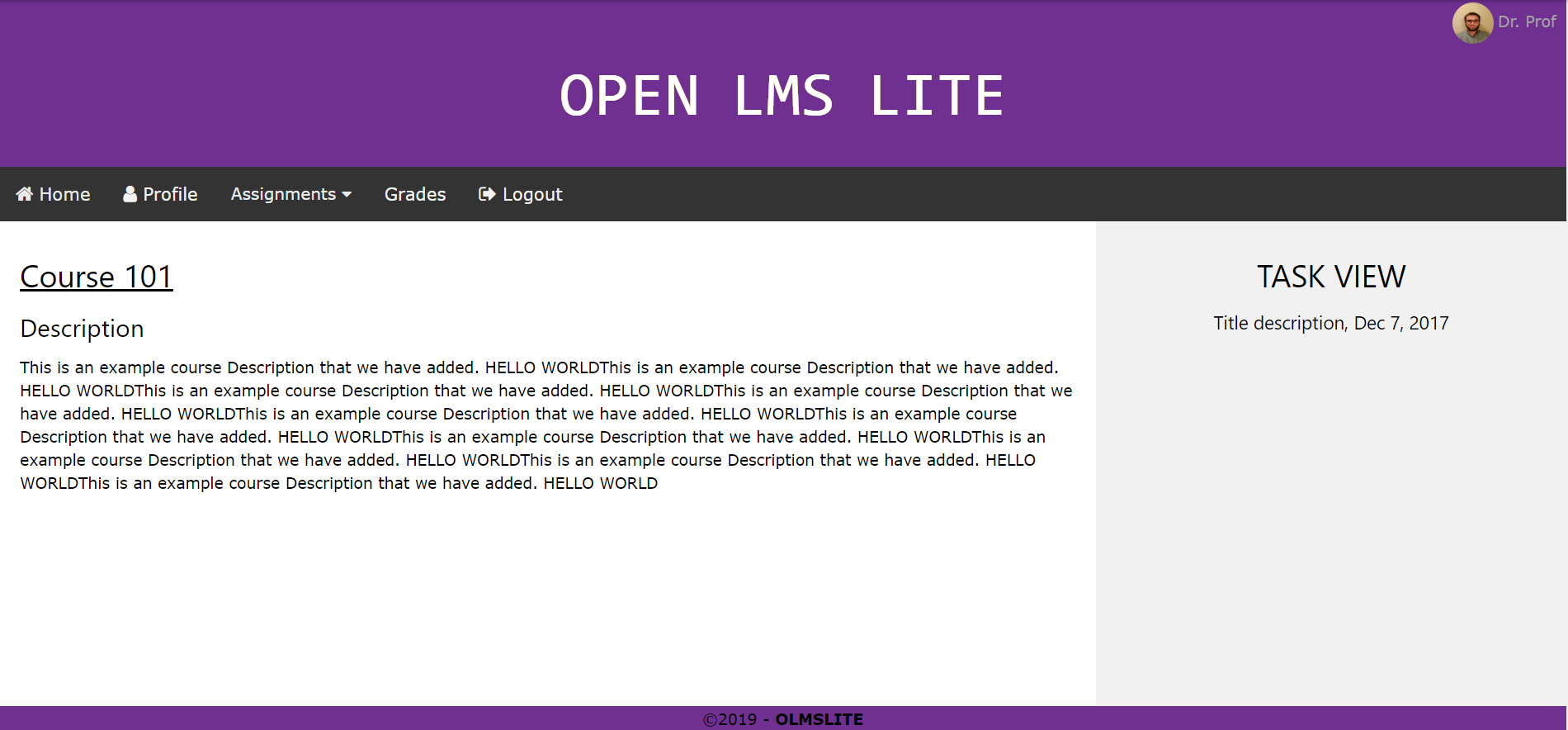
1: Professor pulls up webpage

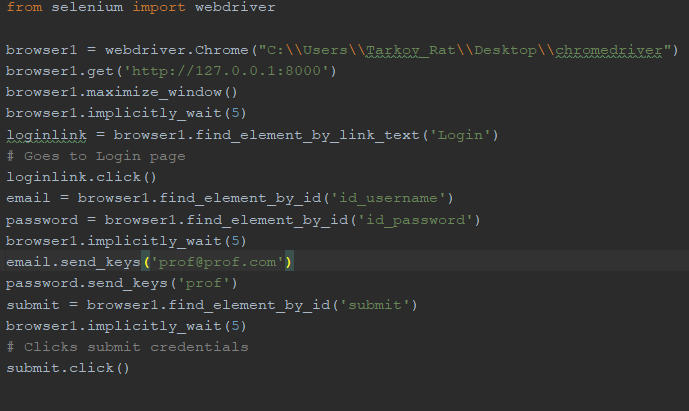
2: Professor clicks login link.

3: Professor enter credentials

4: Professor clicks login button

5: Error page is displayed



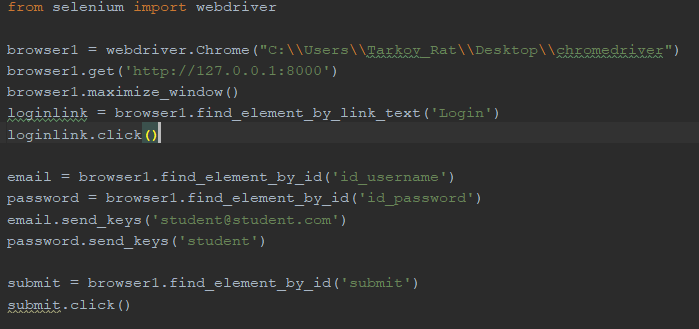


### 2.4. Automated Test

Selenium was the platform used for automated testing of the application. Due to the application being web based and selenium being designed for web based application it was decided to do most of the testing using the platform in order to cover as many lines as possible. The following test cases were made for the automated testing portion of the testing.

**Scope :** To Insure that systems main functionalities are running correctly and that the system is ready for user acceptance testing.

**Test Case 1**

****

**Senario:** Student opens the website and logs in using the correct user name and password.

1: Student pulls up webpage

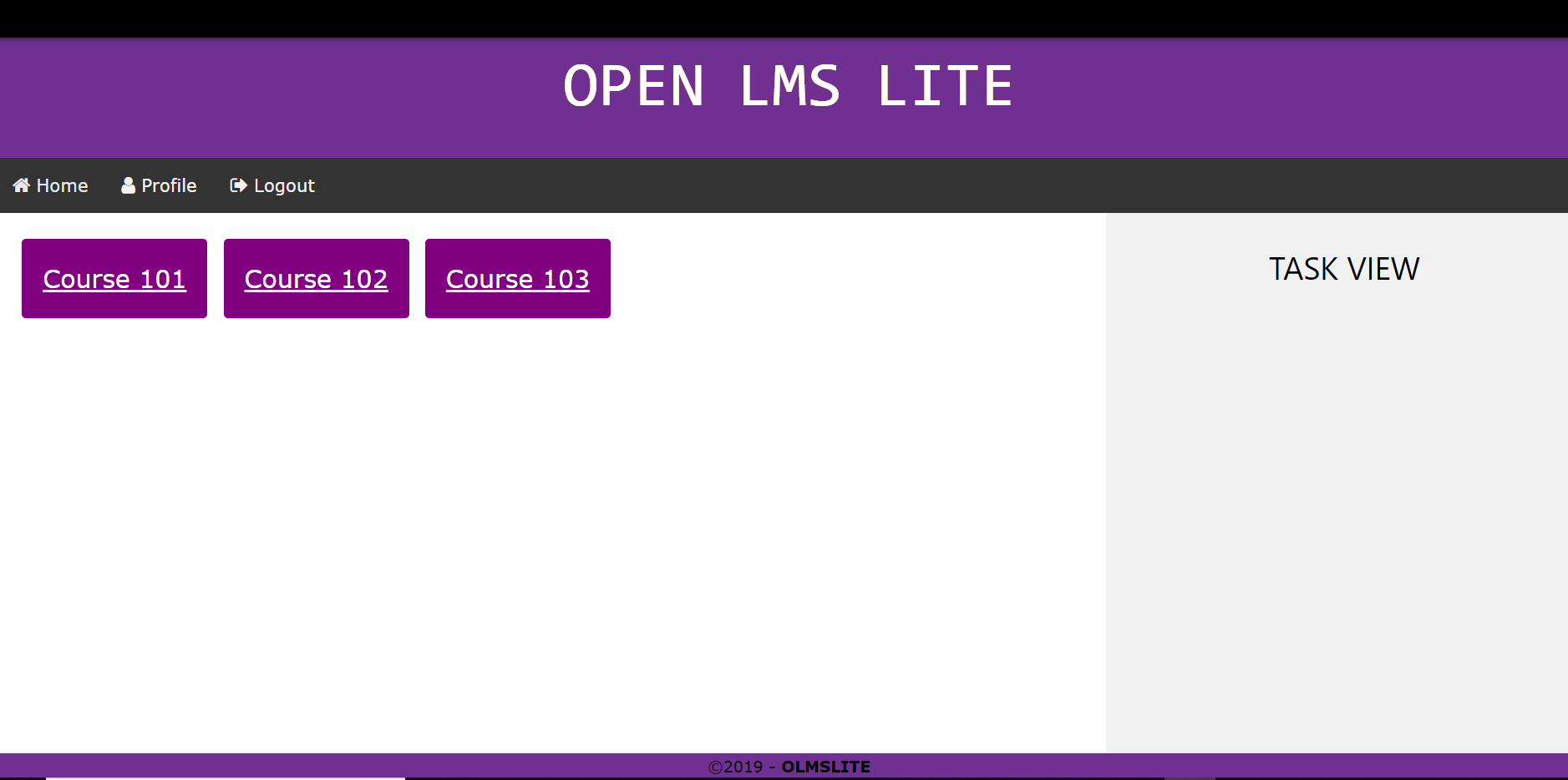
2: Student clicks login link.

3: Student enter credentials

4: Student clicks login button

5: Home page populates with correct information for student.

**Test Case 2**



**Senario:** Professor opens the website and logs in using the correct user name and password.

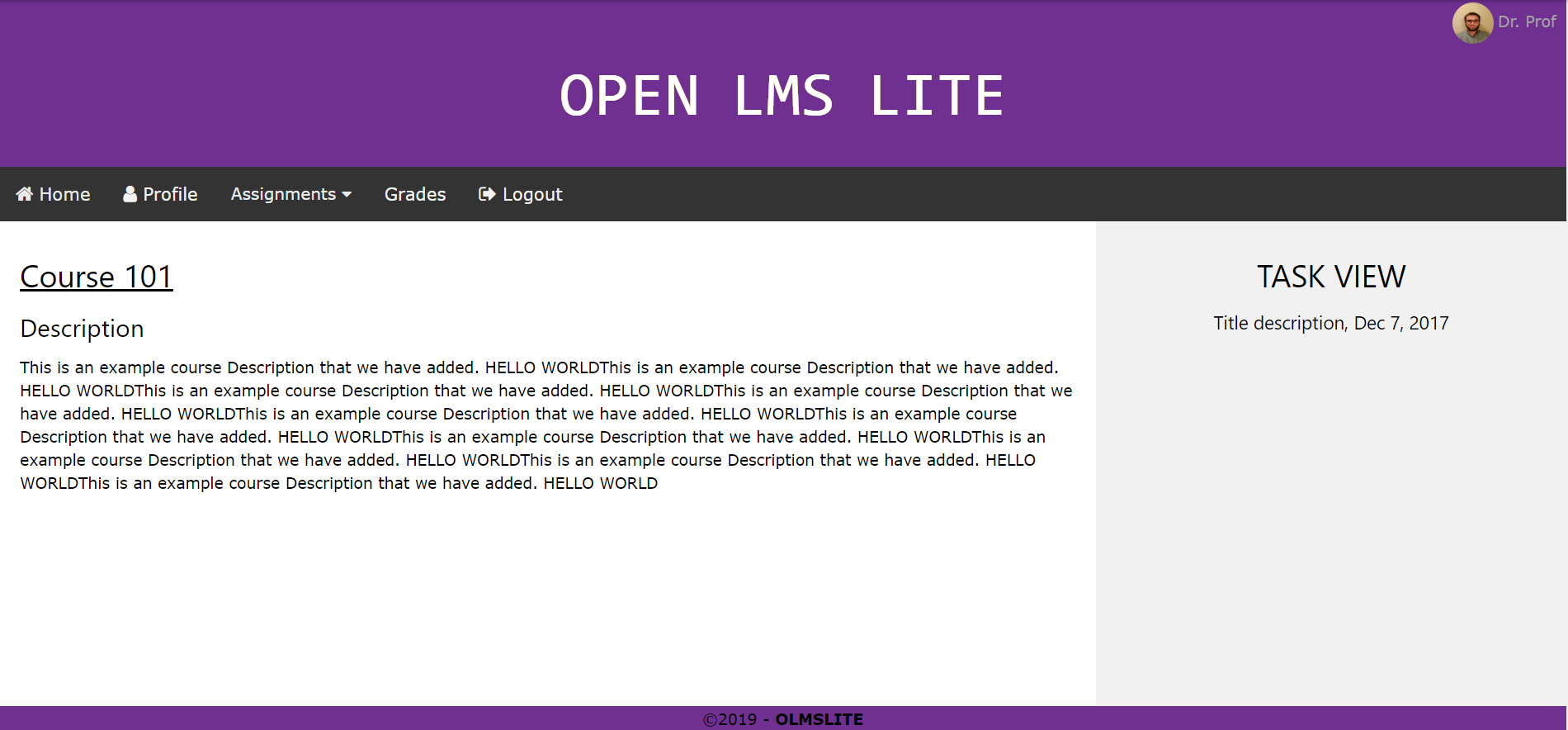
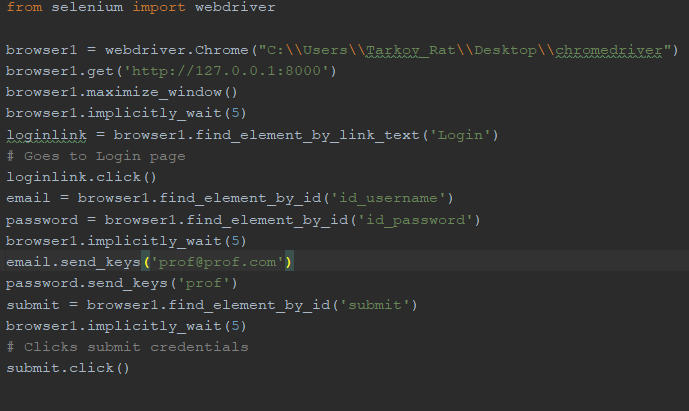
1: Professor pulls up webpage

2: Professor clicks login link.

3: Professor enter credentials

4: Professor clicks login button

5: Home page populates with correct information for Professor.



### 

### 2.5. User Acceptance Test

For the User Acceptance Testing the following test cases were chosen in order to simulate the most real-world application as possible for the system.

**Scope :** To Insure that system can withstand real world test case scenarios and function without error or system failures.

**Test Case 1**

**Senario:** Professor goes to a course and creates a quiz for students to take.

1: Professor pulls up webpage

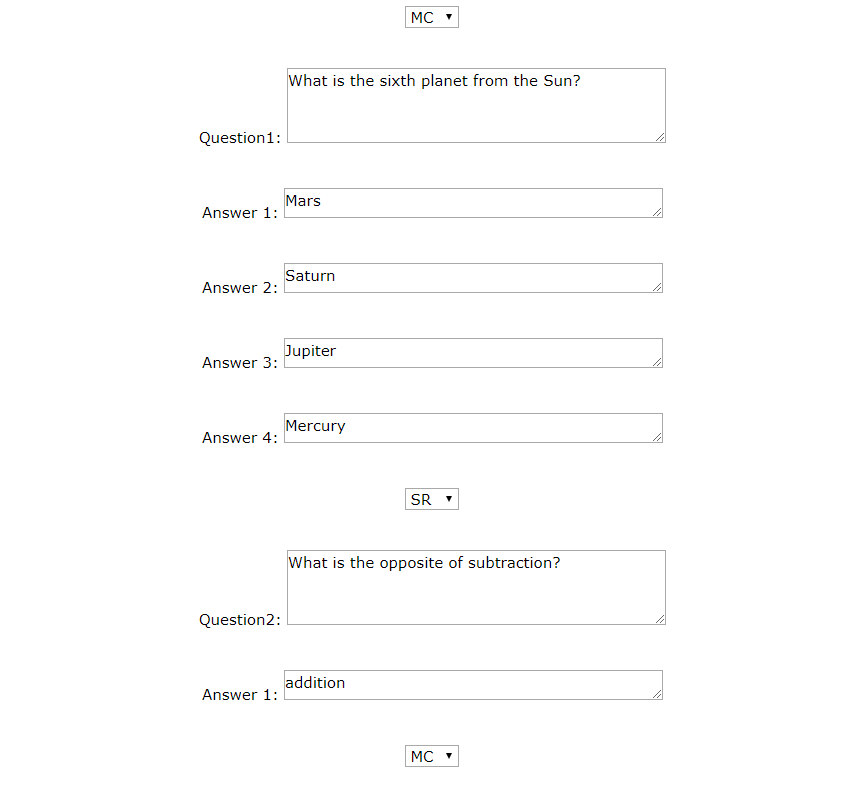
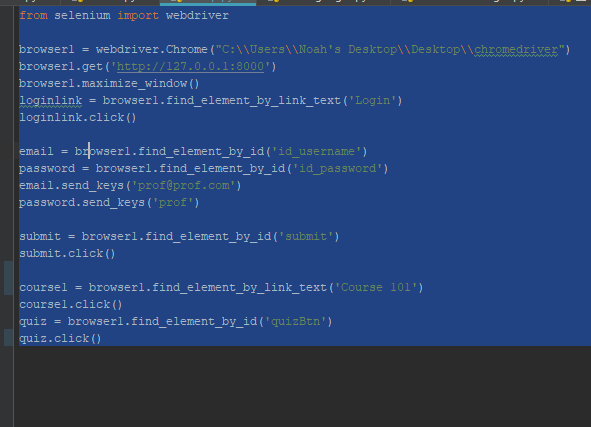
2: Professor clicks login link.

3: Professor enter credentials

4: Professor clicks login button

5: Professor clicks on courses/quiz

6. Professor chooses add quiz and creates quiz and submits.



**Test Case 2**

**Senario:** Student goes to a course and takes a quiz .

1: Student pulls up webpage

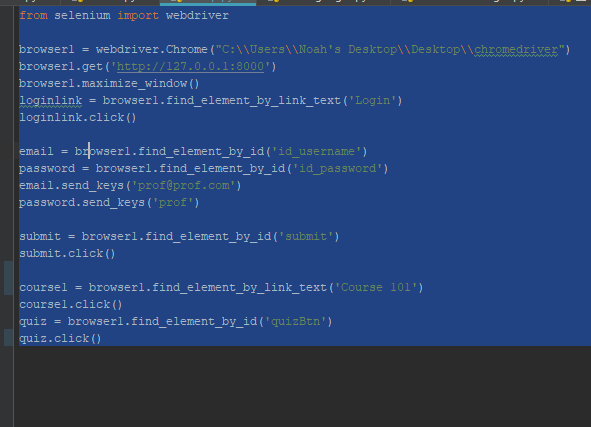
2: Student clicks login link.

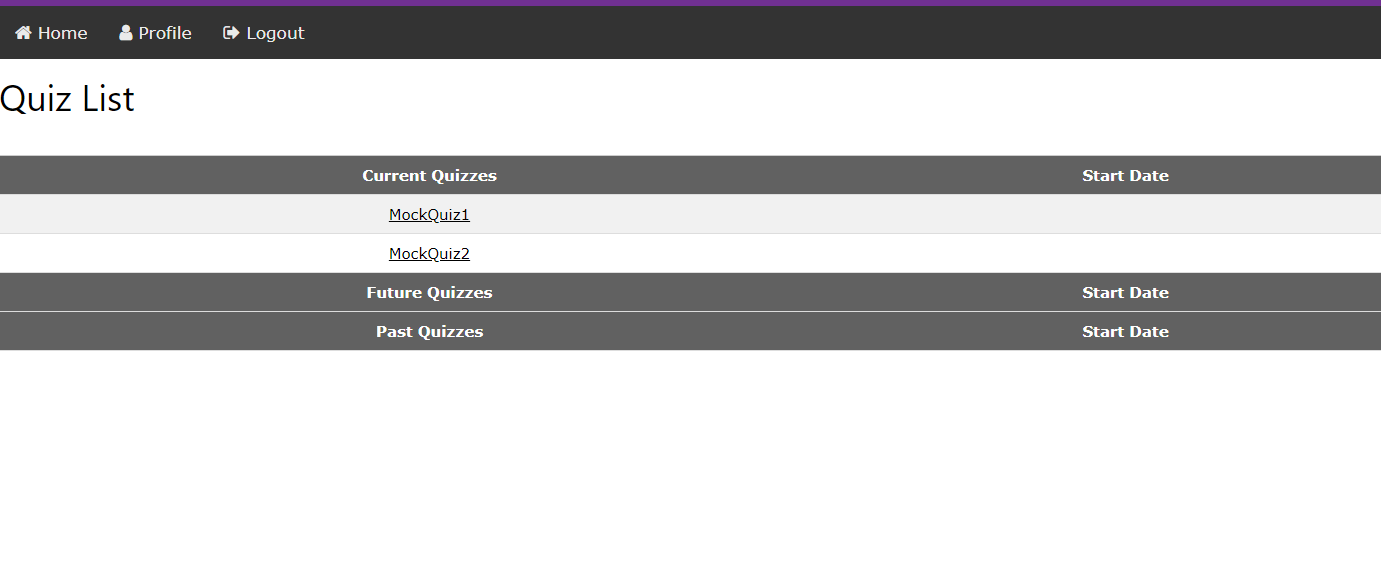
3: Student enter credentials

4: Student clicks login button

5: Student clicks on courses/quiz

6. Student chooses quiz and takes it.





## 

## Functions To Be Tested

The functions that have been tested on the Open LMS Lite are listed below.

* Email Notification
* User Profile/Login
* Quiz Feature
* Web Page Navigation
* IP restriction

## Dependencies

The main dependency linked to the Open LMS Lite system is the database and cloud which it uses to store data needed for the system to run properly. Admins are the only ones able to access this database via their profile log in. Other than the database and cloud there are no notable dependencies associated with the system.

## Schedule

Testing for the Open LMS Lite started on 4/10/2019 and the schedule is as follows below.

|  |  |  |
| --- | --- | --- |
| DATE | TYPE | DES. |
| 04/10/2019 | Automated | User Add/Remove |
| 4/13/2019 | Automated | User Login |
| 4/15/2019 | User Acceptance | Quiz Feature |
| 4/19/2019 | User Acceptance | Email Notification |
| 4/23/2019 | Automated | Quiz Feature |
| 4/26/2019 | User Acceptance | Quiz Feature |
| 4/28/2019 | User Acceptance | Upload/Download |
| 4/30/2019 | User Acceptance | IP Restriction |

## Tools

The Open LMS Lite was tested using selenium which is a high-level python web framework testing tool. Selenium is automated testing which allowed for the maximum amount of line coverage. Selenium was a great tool that expedited the testing process while also allowing the testing to focus on important functionality of the application.

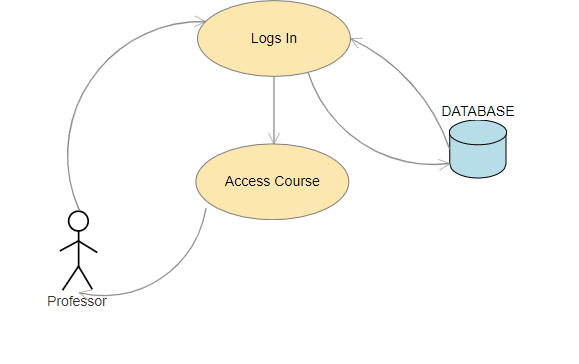
## Test Cases

### 2.2.1 User Use Case

Use case: **User Login**

Statement: The system shall allow a user to login.

Diagram:



**Brief Description**

The user can be either a professor or student, they will log in with username/email and password. Upon successful credentials store in the database the user will go to their designated page based upon user type.

**Initial Step-By-Step Description**

User has clicked the log in button on the homepage of the application

The User enters his/her username.

The User enters his/her password.

The User clicks the login button.

Database is called to find the user credentials that were entered.

If profesor type go to professor profile, if student type go to student profile.

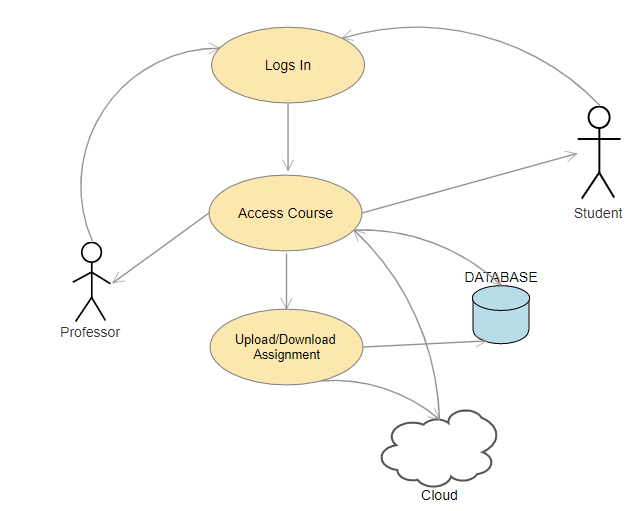
### 2.2.2 Professor Use Case

A.

Use case: **Professor Uploads Homework**

Statement: The system shall allow a Professor to upload homeworks.

Diagram:



**Brief Description**

The professor user has logged in and uploads homework by the click of a specific button on professor homepage screen. User has already navigated to the class in which to add homework to.

**Initial Step-By-Step Description**

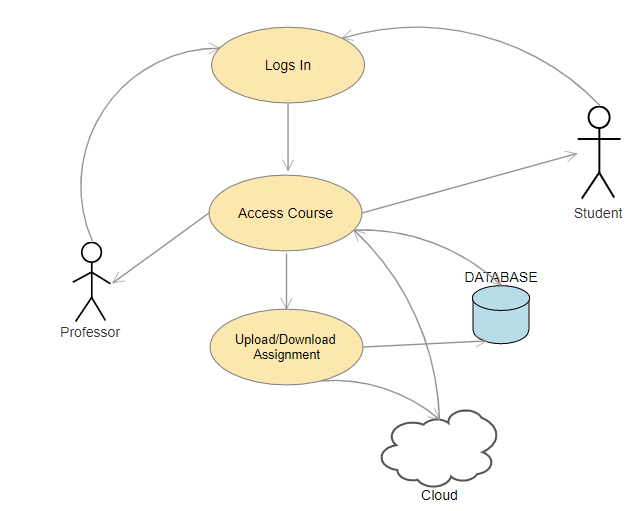
1. The clicks the upload homework button.
2. File Explorer dialog box opens.
3. Professor navigates and selects the homework file.
4. Professor clicks ok button on file explorer dialog box.
5. Professor sets start and stop time/date for the assignment.
6. Professor clicks the upload button.

B.

Use case: **Professor Creates Quiz/Survey**

Statement: The system shall allow a Professor to create a quiz and survey.

Diagram:



**Brief Description**

The professor user has logged in and creates a quiz/survey by the click of create quiz/survey button. User has already navigated to the class in which to add quiz/survey to.

**Initial Step-By-Step Description**

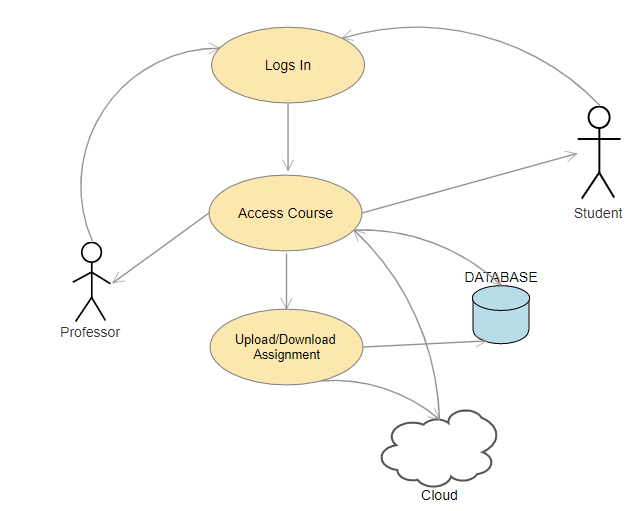
1. Professor selects create quiz/survey.
2. New pane/window opens.
3. Professor enters quiz/survey name.
4. Professor enters the start and end time/date.
5. Professor builds quiz/survey.
6. Professor clicks review.
7. Professor click upload.

C.

Use case: **Professor Imports Quiz/Survey**

Statement: The system shall allow a Professor to import a quiz and survey.

Diagram:



**Brief Description**

The professor user has logged in and Imports Quiz/Survey by the click of the “Import quiz/survey” button. User has already navigated to the class in which to add quiz/survey to.

**Initial Step-By-Step Description**

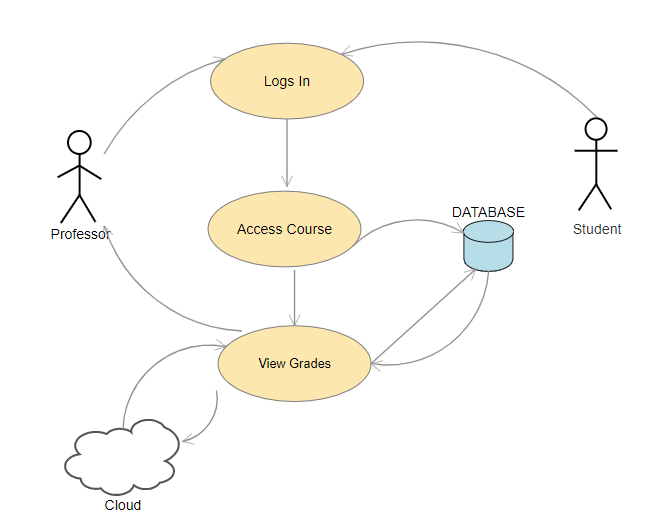
1. Professor selects import quiz/survey.
2. File Explorer dialog box opens.
3. Professor navigates and selects the quiz/survey file.
4. Professor clicks ok button on file explorer dialog box.
5. Professor is able to view the assignment and add points to each question
6. Professor sets start and stop time/date for the assignment.
7. Professor clicks the upload button.

D.

Use case: **Professor Views Students Grades**

Statement: The system shall allow a Professor to view students grades.

Diagram:



**Brief Description**

The professor user has logged in and clicks the“View Grades” button. User has already navigated to the class in which grades they want to view.

**Initial Step-By-Step Description**

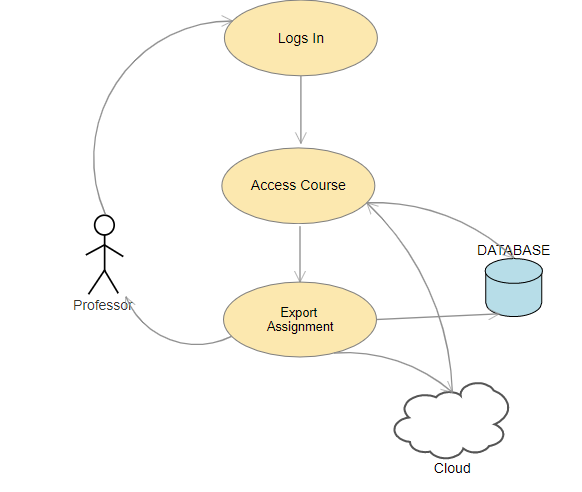
1. Professor click view grades.
2. Database is queried to populate grade report specific grades from the table with that classes grade in them.
3. Grade book is displayed with list of assignments.
4. Specific assignment is clicked and breakdown of student grades is displayed.

E.

Use case: **Professor Exports Quiz/Survey**

Statement: The system shall allow a Professor to export a quiz and survey.

Diagram:



**Brief Description**

The professor user has logged in and clicks the“Export Quiz/Survey” button. User has already navigated to the class in which the quiz/survey is located.

**Initial Step-By-Step Description**

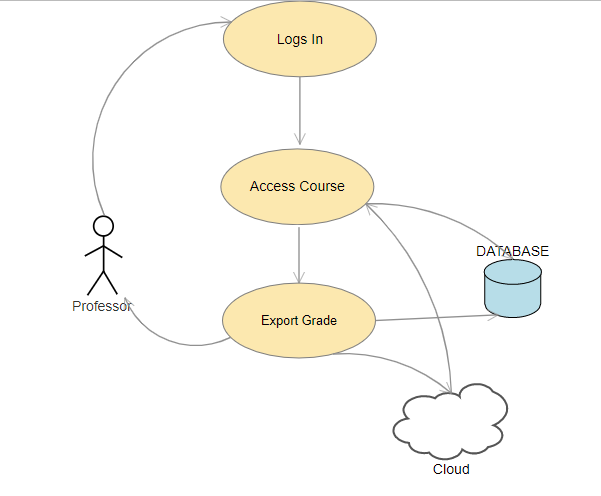
1. Professor clicks export button.
2. Professor chooses export quiz/survey.
3. Professor selects quiz/survey.
4. Professor clicks download.
5. File explorer opens.
6. Professor chooses the file location and name of file.
7. Professor click save button in file explorer window.

F.

Use case: **Professor Exports Grades**

Statement: The system shall allow a Professor to export grades.

Diagram:



**Brief Description**

The professor user has logged in and clicks the“Export Grades” button. User has already navigated to the class in which grades they want to view.

**Initial Step-By-Step Description**

1. Professor clicks export button.
2. Professor chooses export grade.
3. Professor selects assignment(s).
4. Professor clicks download.
5. File explorer opens.
6. Professor chooses the file location and name of file.
7. Professor click save button in file explorer window.

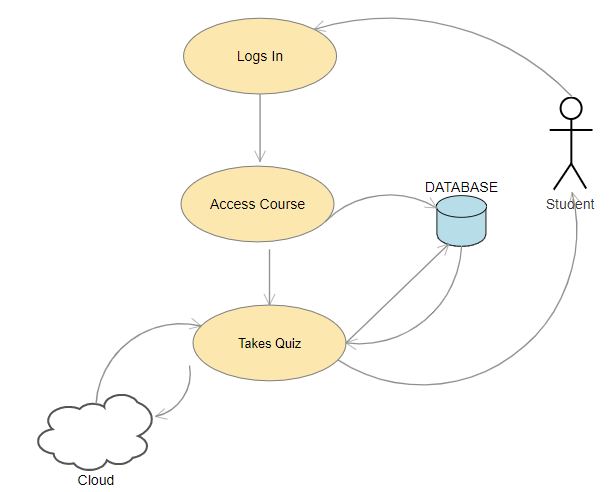
### 2.2.3 Student Use Case

A.

Use case: **Student Takes Quiz/Survey**

Statement: The system shall allow a Student to take a quiz and survey.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for Quiz/Survey.

**Initial Step-By-Step Description**

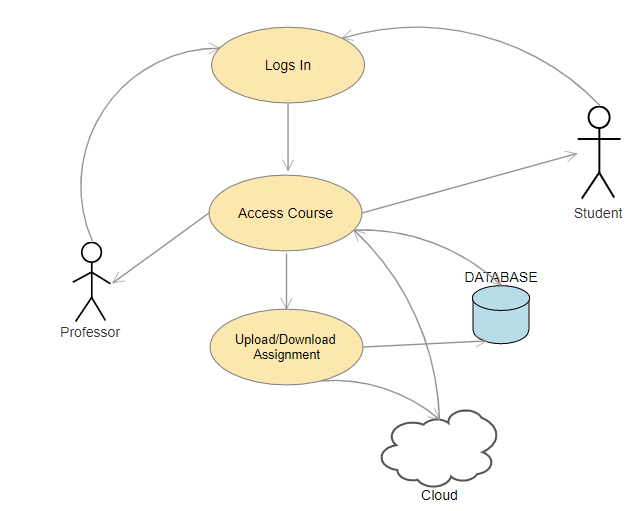
1. The Student navigates and selects the class
2. New window opens
3. Student selects the quiz/survey that they want to complete
4. The quiz/survey opens up
5. Student completes the quiz/survey
6. Student click submit button

B.

Use case: **Student Downloads Homework**

Statement: The system shall allow a Student to download a homework.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for homework.

**Initial Step-By-Step Description**

1. The Student navigates and selects the class.
2. New window opens, showing a list of the homeworks.
3. Student navigates and selects the homework file.
4. File location is sent from database
5. LMS takes address searches in dropbox and finds file
6. Student downloads homework file.

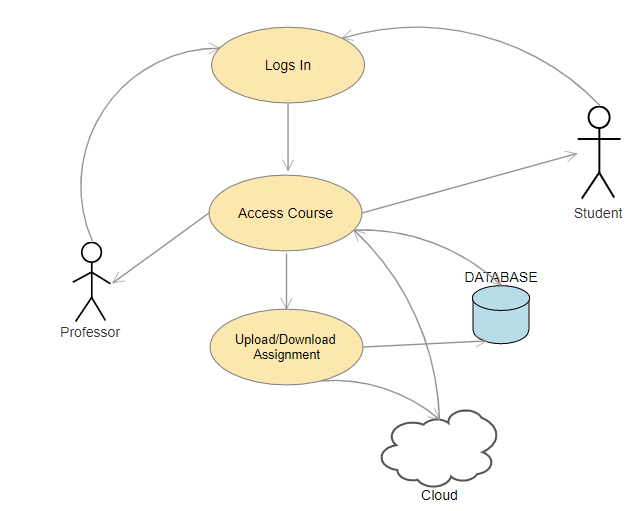
## 

C.

Use case: **Student Submits Completed Homework**

Statement: The system shall allow a Student to upload a completed homework.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for homework.

**Initial Step-By-Step Description**

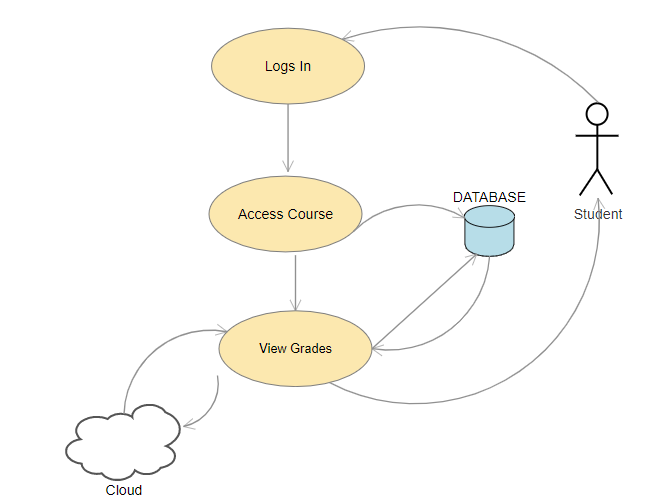
1. The Student navigates and selects the class.
2. New window opens, showing a list of the homeworks.
3. Student navigates and selects the homework file.
4. Student selects “add a file”, then “upload”.
5. Student selects their file name.
6. Student clicks add.
7. Student clicks submit.
8. Address of file is sent to database
9. File is stored in dropbox

D.

Use case: **Student View Grades**

Statement: The system shall allow a Student to view their own grades.

Diagram:



**Brief Description**

The student user has logged in and navigates to the desired class for viewing grades.

**Initial Step-By-Step Description**

1. The Student navigates and selects grades.
2. Database is queried for student grade report
3. New window opens, showing a list of the grades for each assignment in each class.

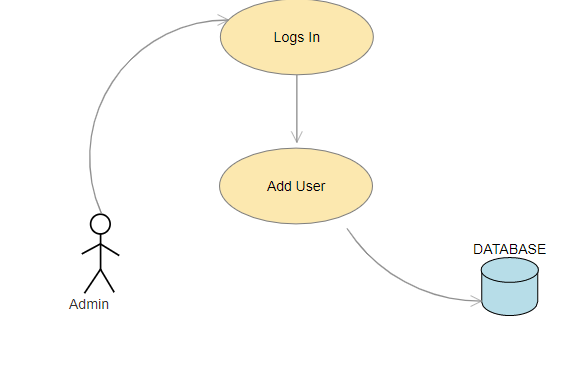
### 2.2.4 Admin Use Case

A.

Use case: **Admin adds a User**

Statement: The Admin shall be able to add a User to the database from the admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to User window.

**Initial Step-By-Step Description**

1. The Admin navigates to the user window.
2. Admin Selects “Add User”
3. Admin enters user information
4. Admin specifies User Type
5. Admin Submits User information
6. User information is sent and added to the database

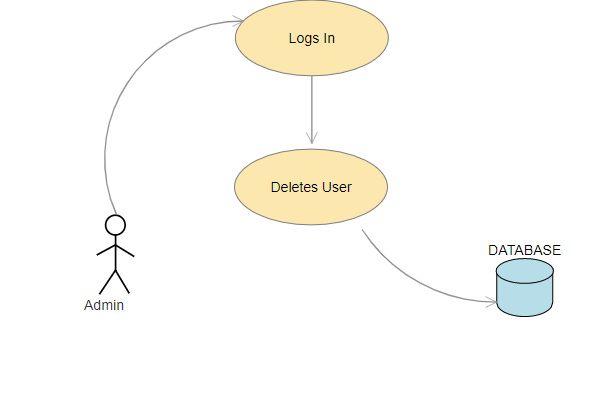
### 

B.

Use case: **Admin deletes a User**

Statement: The Admin shall be able to delete a User from the database within the admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to User window.

**Initial Step-By-Step Description**

1. The Admin navigates to the user window.
2. Admin filters by user identification
3. User appears
4. Admin clicks Delete
5. User information is deleted from the database

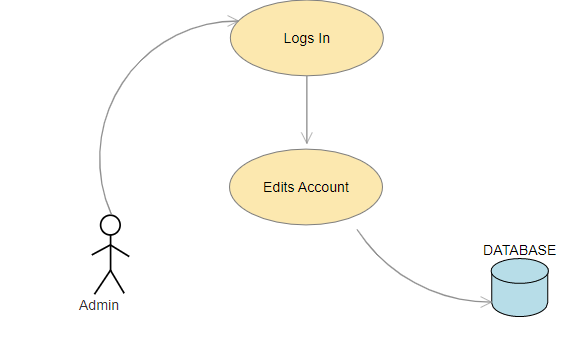
## 

C.

Use case: **Admin edits User password**

Statement: The Admin should be able to edit a User’s password within admin interface.

Diagram:



**Brief Description**

The Admin has logged in and navigates to User window.

**Initial Step-By-Step Description**

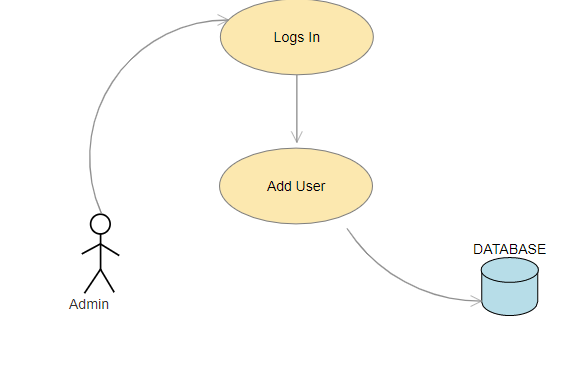
1. The Admin navigates to the user window.
2. Admin filters by user identification
3. User appears
4. Admin selects edit users information
5. Admin changes password
6. Admin clicks submit
7. Edited information is sent to database

D.

Use case: **Admin adds a User**

Statement: The Admin shall be able to add a course to the database from the admin interface.

Diagram:



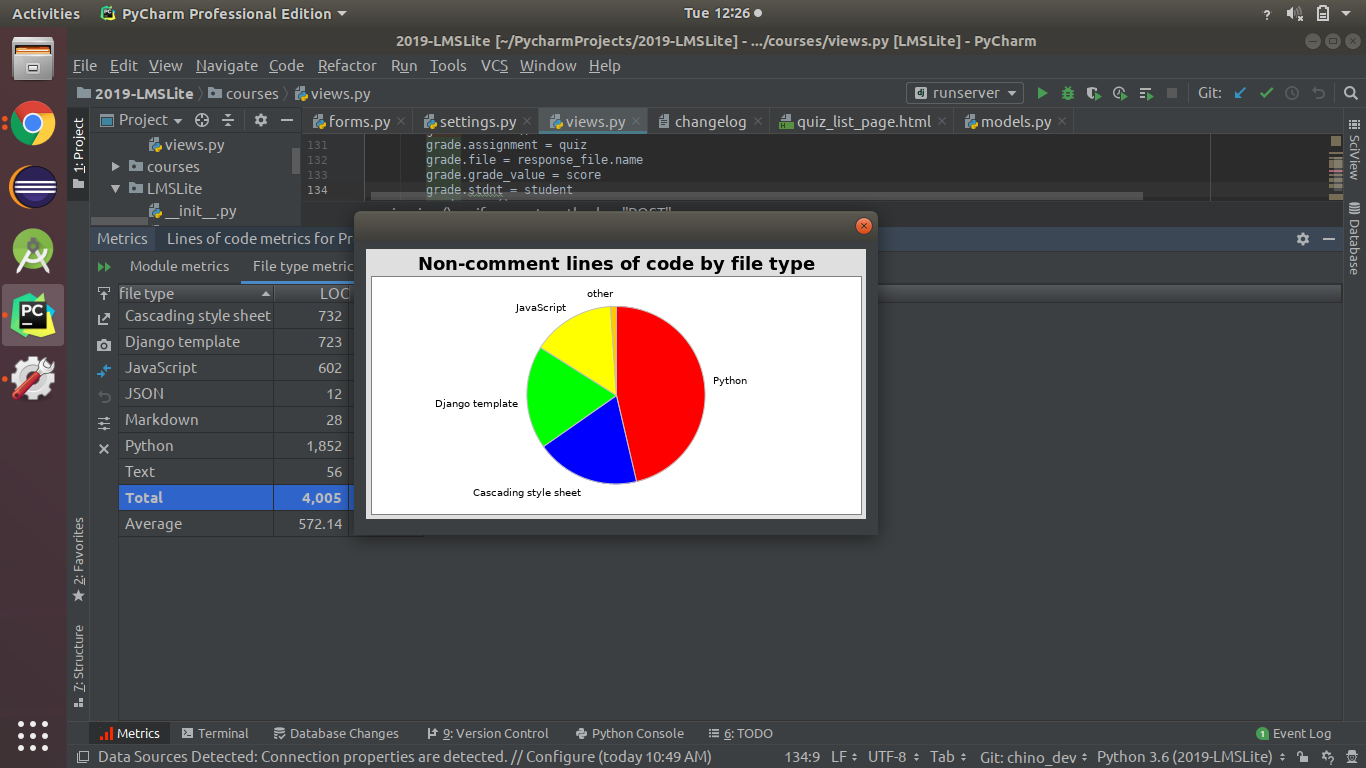
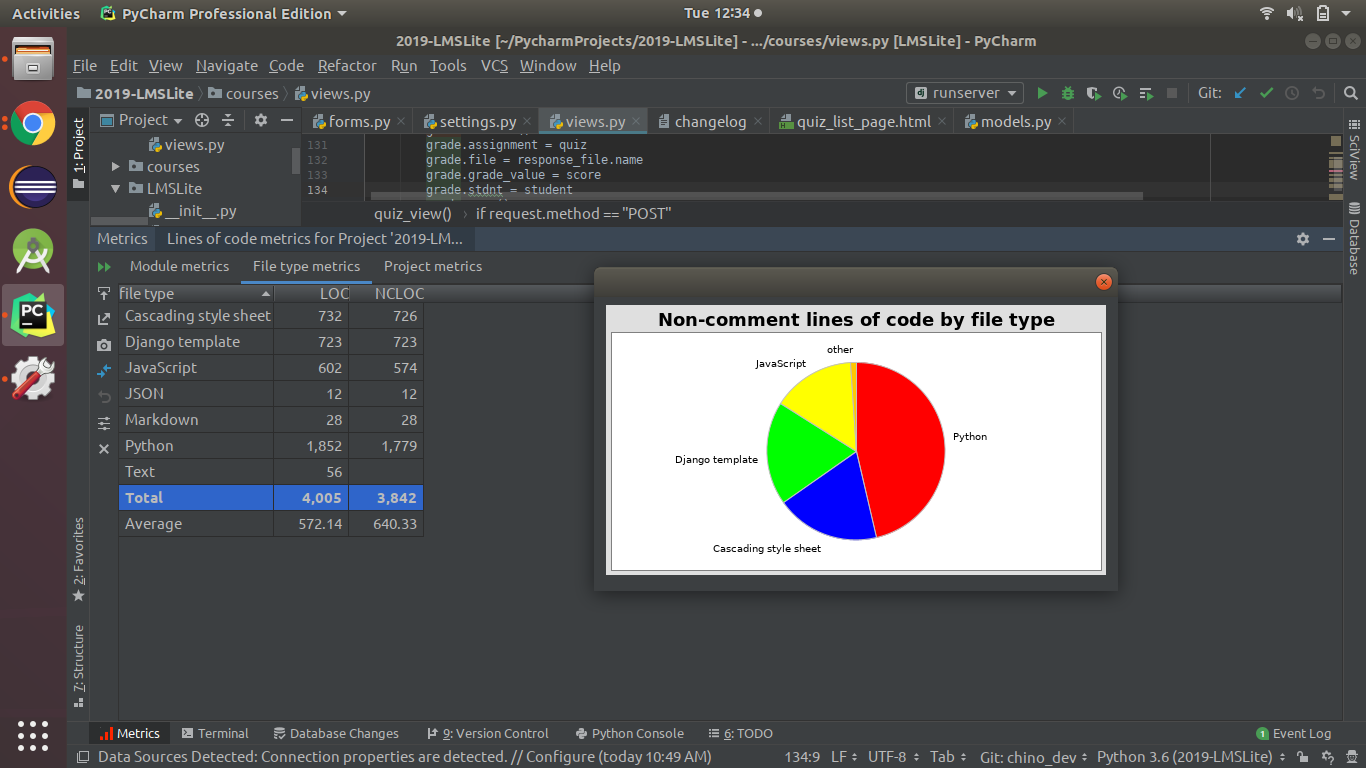
**Brief Description**

The Admin has logged in and navigates to Course window.

**Initial Step-By-Step Description**

1. The Admin navigates to the Course window.
2. Admin Selects “Add Course”
3. Admin enters Course information
4. Admin designates professor to course
5. Admin Submits Course information
6. Course information is sent and added to the database

## Code Metrics



# Post Mortem

The Open LMS Lite application was a very interesting and educational experience for all group members. Members were challenged by learning a lot of information in a very limited amount of time and having to display that knowledge that was learned in order to provide the best product possible for our client. Members also faced many difficulties rooting from class schedules and other priorities within the semester that had to be taken care as the group approached graduation. Through all of the challenges that met our group throughout the semester, we were still able to come together and work efficiently to complete a quality project.

Prior to the project, the Open LMS Lite team as a whole, had little experience with the Python coding language and using it within the Django framework. This learning curve definitely through a few wrenches in our plans and thought us a valuable lesson we all will take from this project which is do not bite off more than you can chew. Although there was a lot of research that had to be done to be able to use Django and code efficiently in Python the group was able to work together in order to accomplish the projects intent. We feel as if it was a good thing that we were able to get the experience of working with a new language and platform due to the fact that many of us will move on to jobs where we will have to adapt to the environment there and this project gave us a that experience with having to adapt to what was around us. All in all the group is more than proud of the effort and the result of that effort in the software we represent. We all know that this is the experience that will start off our careers and we were glad to be able to conclude our undergrad with The Open LMS Lite.

# 

# Appendix

## Meeting Minutes

**MEMORANDUM**

Date: January 27, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey

Cameron Morgan

Austin Simmons

Chino Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:00 pm. The following items were discussed:

* Feasibility Study

**Project Requirements**

Our project requirements come from a combination of client specific requirements and things that every LMS should have. These items will be the minimum features of our application.

* Task view for Students
* Notification
* Reports
* Secure
* Customizable
  + Slideshows
  + Videos
  + Assignments
  + Tests, Quizzes, and Surveys
  + Auto Grade
* xAPI Compatible

**Assessment of Options**

This section covers additional features that will be nice to add to the LMS.

* Flash Cards (Study Tool)
* Data Migration
* Auto Curve
* Student View Class Average for Quiz
* Live Reporting for Surveys
* IP Restriction
* Quiz Codes

**MEMORANDUM**

Date: January 30, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey

Cameron Morgan

Austin Simmons

Chino Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Updates to the Feasibility Study

Links:

<https://xapi.com/the-layers-of-experience-api-xapi/>

<https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Communication.md#applicationjson>

* Finish Feasibility by Friday (2/1/2019)

**MEMORANDUM**

Date: February 7, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey- Left Early

Cameron Morgan

Austin Simmons

Chino Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 3:30 pm. The following items were discussed:

* We discussed the features that would be stored in the database(tables)
  + Users
    - userID (int)
      * email(txt)
      * fName(txt)
      * lName(txt)
      * role(int)
      * password(#txt)
  + Course
    - courseID(int)
      * (FK to UserID) as professorID(int)
      * studentList[int]
      * CourseName
  + AssignmentInfo
    - AssignmentInfoID(int)
      * (FK to Assignment)AssignmentID(int)
      * (FK to CourseID) as CourseID(int)
      * dateAva(date)
      * dateDue(date)
      * fileAddress(txt)
  + Assignment
    - AssignmentID(int)
      * type(int)
      * AssignmentName(txt)
      * quizCode(txt)
  + Questions
    - questionsID(int)
      * question(txt)
      * pAns[]
      * cAns[]
      * (FK)AssignmentID(int)
  + Responses
    - responseID(int)
      * (FK to Users)UserID (int)
      * (FK to Assignment)AssignmentID (int)
      * studentAnswers[]
      * grades(double)
* Things we are going to store on the server
  + Assignments from Professor
  + Submission from Student
* We will be coding the front end in HTML, CSS and the back end in Python
* Use Case 1
  + Professor uploads quiz to both server and db.
  + Professor adds quiz data to table
  + Notification sent
  + Student logs on.
  + Takes quiz
  + Answers saved on answer click
  + Submit quiz/grade sent to database/add info to log.(xapi)
  + Due date expires, notify professor
  + Generate report

**MEMORANDUM**

Date: February 13, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey

Cameron Morgan

Austin Simmons

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:00 pm. The following items were discussed:

* Expand on xAPI in Feasibility Report
  + Track learning (compare previous tests to current test)
  + Limit ability to cheat
* Requirements Report
  + Describe what is going on in each requirement
  + Sign Off from client
  + Requirements
    - Profile
    - Quizzes and Surveys
    - Reports
    - Secure
      * Hash passwords
    - Auto Grade
    - xAPI
      * Assignment - Layer 2
        + Uploading - Prof
        + Turnin - Student
        + Store
      * Email Notification - Layer 1
        + Automatically email a student when a assignment or quiz is uploaded
      * Cheating prevention - Layer 1
        + IP restriction
        + Lockdown browser
        + Quizcode
      * Data migration - Layer 3
        + **Ask Client format for data extract**
      * Task View for Students - Layer 1
        + Upcoming events (quizzes due)
      * No Layer 4 because no social media

**Client Meeting 2/14/2019**

Profiles

Need 3 Profiles - Students, Professors, and Admins

Students:

Professors:

Admin: Add Professors/Students

**Don’t be ambiguous!!!**

Client Requests

Student - Quiz (multiple choice, multi select, short/long answer)

Professor - Create quiz questions (upload from a specific format or create online)

Multiple choice and Multi select need to be auto graded

Questions by default set to 1pt each but can be changed

Assignments can be downloaded as word/pdf/txt (assignments that a professor uploads)

Start date - Due date

Professor needs to pull student submissions

Think more about how files will be stored (files with same names - Student)

Close of assignment/quiz on due date (think about more)

xAPI

Data Migration (professor)

Download quiz/assignment to upload to folio/blackboard

Be able to upload quiz/assignment from blackboard/folio

Be able to export grades to folio

**MEMORANDUM**

Date: February 17, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:00 pm. The following items were discussed:

* Setup Web Server Framework
* Created 3 mock web pages(LMS home, Login, Professor Welcome)
* Made edits to database schema

Next Meeting

* Be able to login is as a User professor/student
* Have requirement sign off complete with basic edits(DETAILS)
* Set up Database

**MEMORANDUM**

Date: February 20, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Cameron Morgan

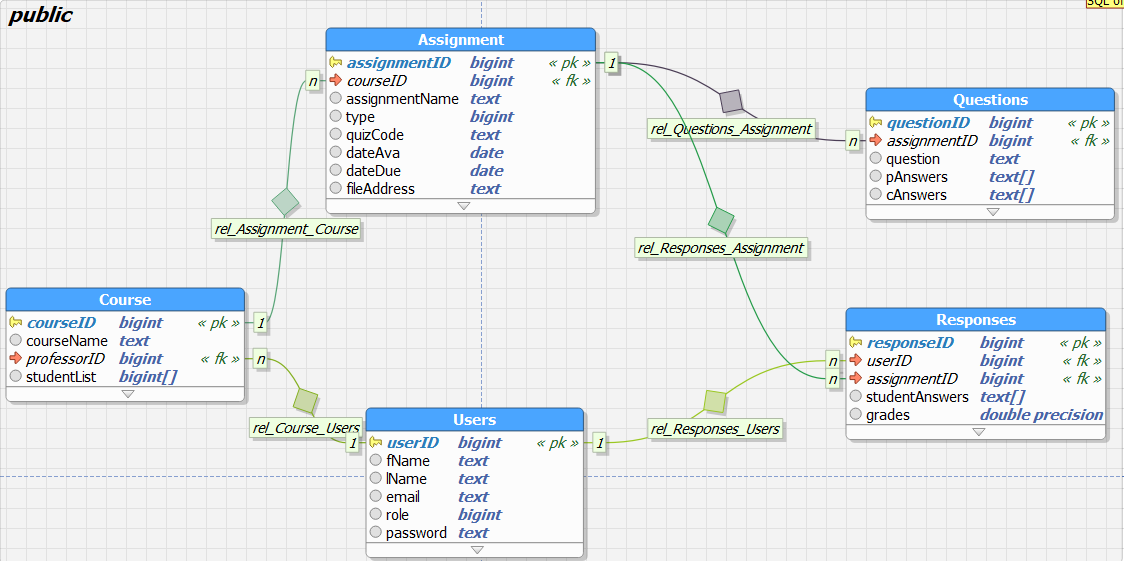
Austin Simmons

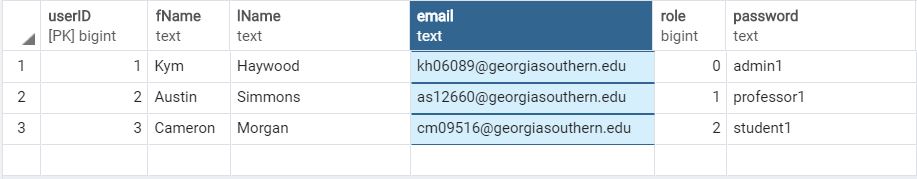
Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:00 pm. The following items were discussed:

* Users
  + userID (int)
    - email(txt)
    - fName(txt)
    - lName(txt)
    - role(int)
    - password(#txt)
* Course
  + courseID(int)
    - (FK to userID) as professorID(int)
    - studentList[int]
    - courseName
* Assignment
  + assignmentID(int)
    - type(int)
    - assignmentName(txt)
    - quizCode(txt)
    - (FK to courseID) as courseID(int)
    - dateAva(date)
    - dateDue(date)
    - fileAddress(txt)
* Questions
  + questionID(int)
    - question(txt)
    - pAnswers[]
    - cAnswers[]
    - (FK)assignmentID(int)
* Responses
  + responseID(int)
    - (FK to Users)userID (int)
    - (FK to Assignment)assignmentID (int)
    - studentAnswers[]
    - grades(double)





* Login Page and Home page are functional

Things to be done by 2/24/2019 @ 2pm:

* SRS Document- assigned different portions of the document to group members
* <https://help.blackboard.com/Learn/Instructor/Tests_Pools_Surveys/Reuse_Questions/Upload_Questions>

Things to be done by 2/27/19:

* Login page
* Connect database to framework

**MEMORANDUM**

Date: February 24, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey- Left Early

Cameron Morgan

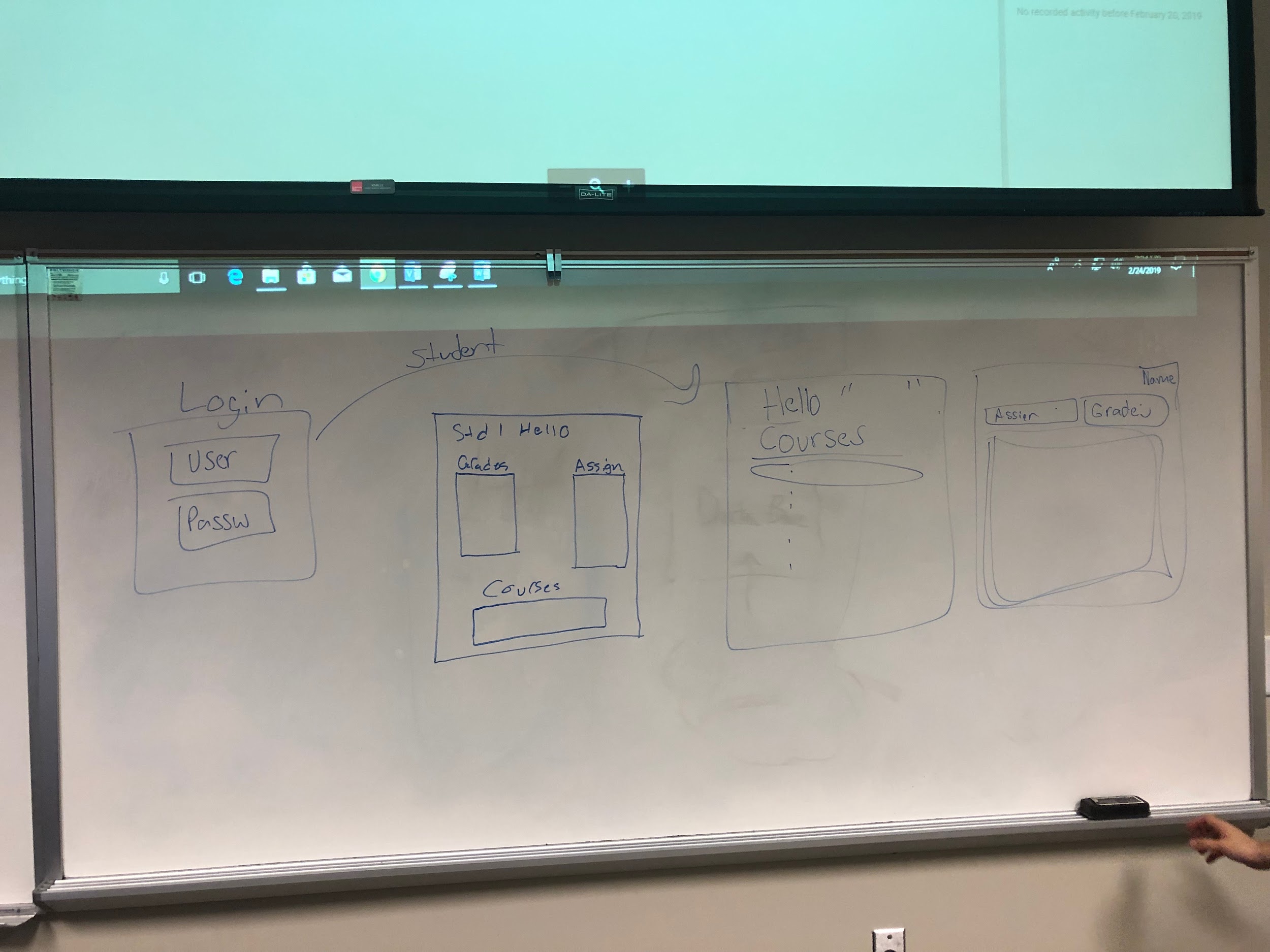
Austin Simmons

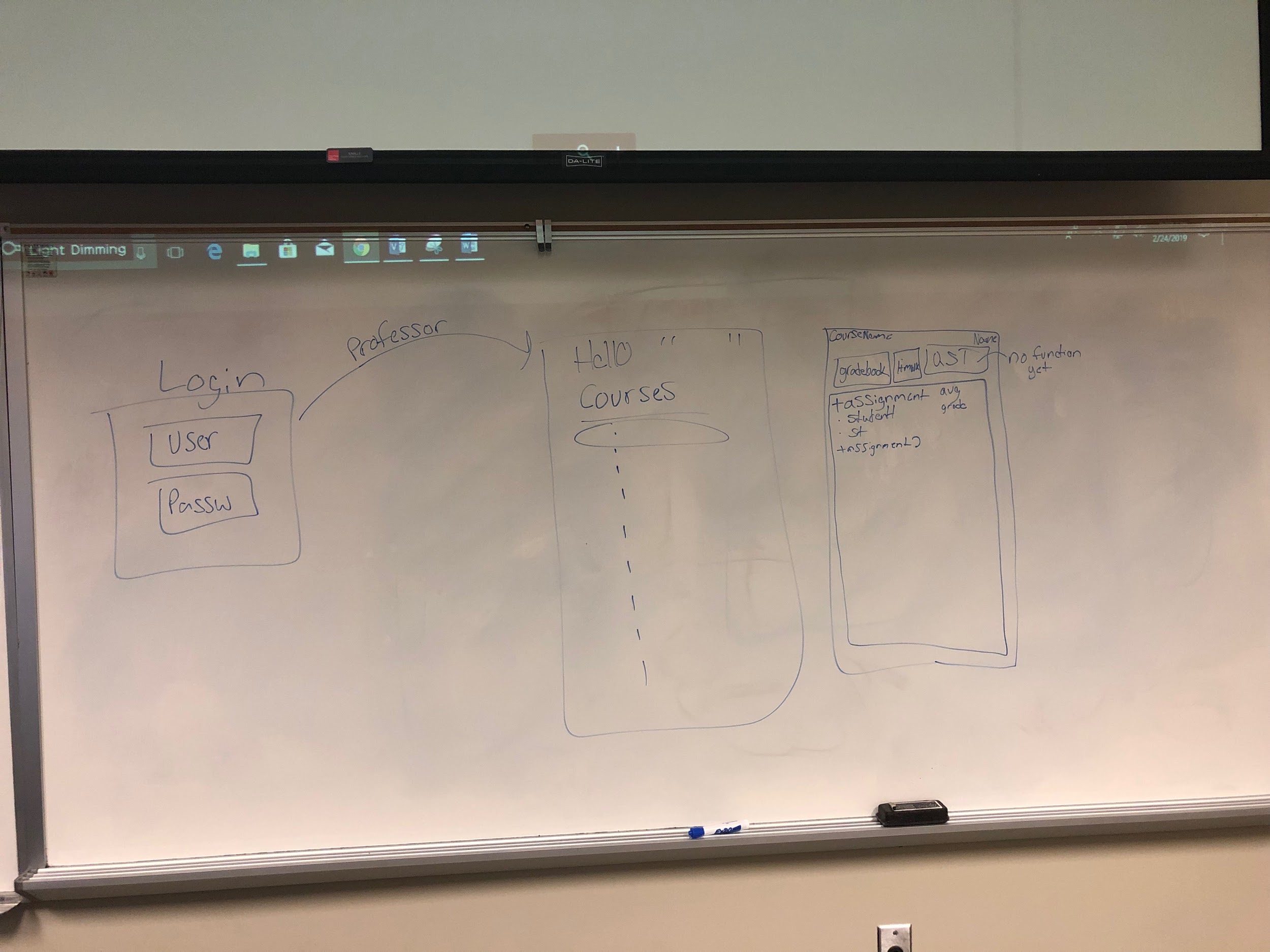
Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 2:30 pm. The following items were discussed:

* SRS
* Tasks
* Gantt Chart
* Basic profile design





*Basic outline of profile pages*

Meeting adjourned at 5:00pm

**MEMORANDUM**

Date: February 27, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Cameron Morgan

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Reviewed SRS
* Edited SRS
* Submit SRS 2/28/2019
* Everyone needs to download PyCharm on Computer (Professional)
  + Download as student user
* Explanation on how to push to personal branch(NOT MASTER)
  + When you make edits push to branch and name the branch “*Thing that was worked on”.* Ex. HomepageEdits
  + Comment specifically when pushing what was worked on
  + When working on code (PLEASE COMMENT WITHIN CODE)
* Project Files are in the GitHub Repository
  + Edit the HTML files

**MEMORANDUM**

Date: March 6, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Cameron Morgan

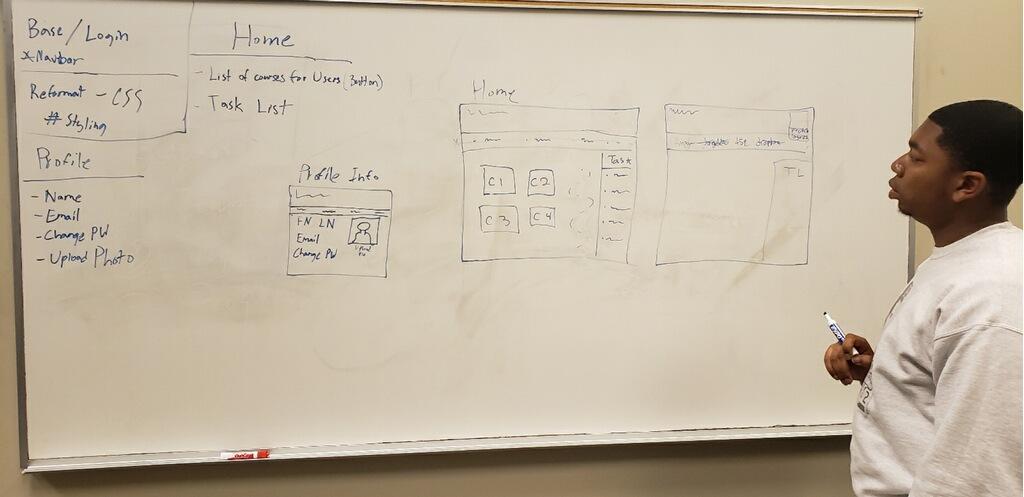
Austin Simmons

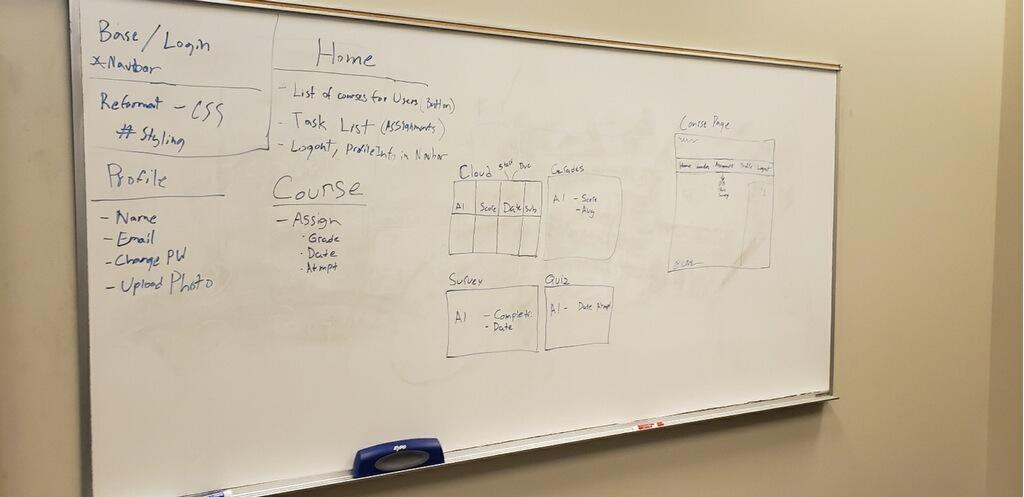
Chinonyerem Ugwumadu

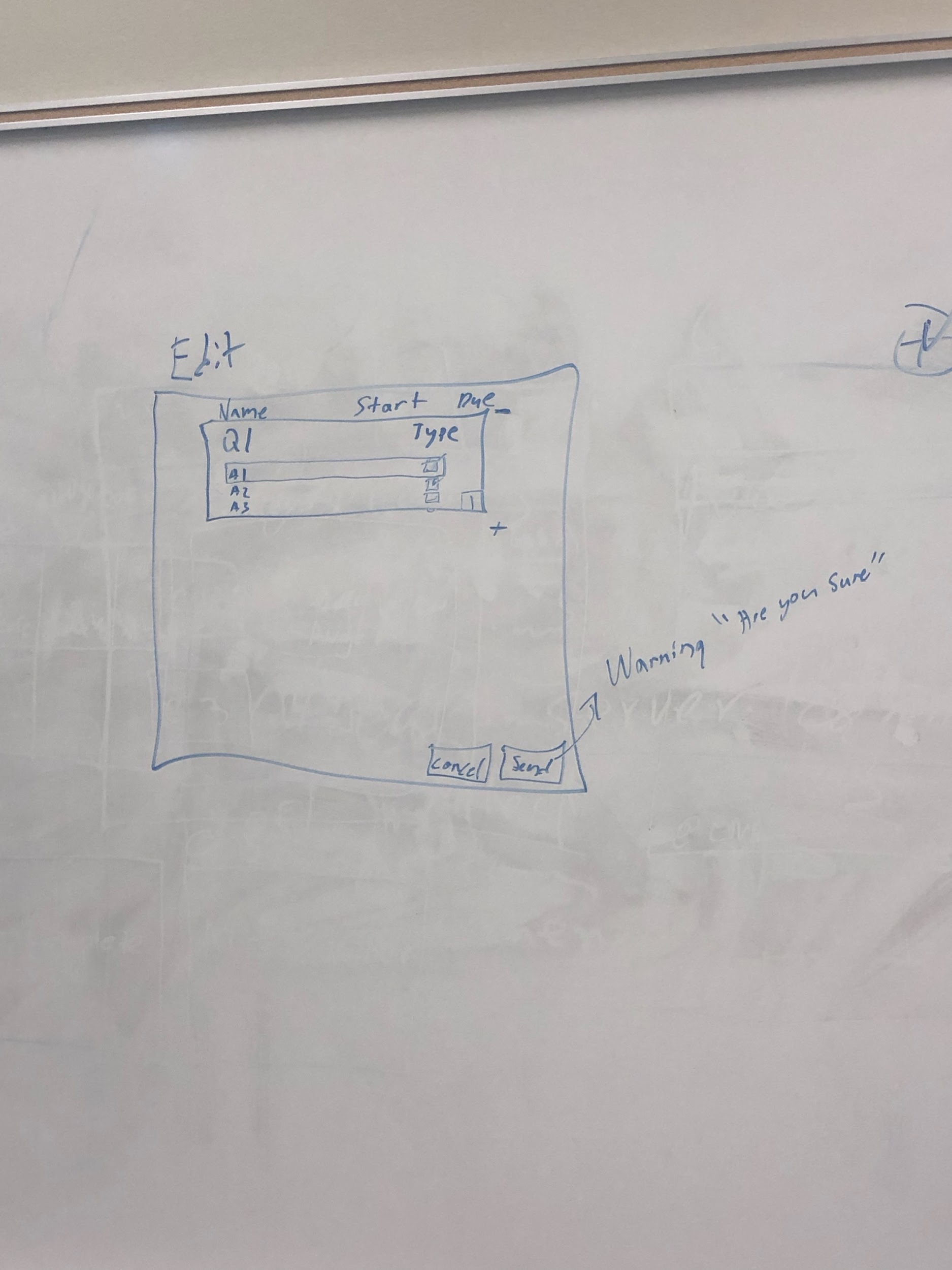
Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Repository has been forked to correct location(Re-Pull project to Pycharm)
* Discussed admins are the only ones who can register/create users
* Using SHA-256 for password encryption instead of base64
* Discussed data we need for student pages
  + Base/Login
    - Navbar
  + Home Page
    - List of courses for users (button)
    - Task List (Assignments)
    - Logout, Profile information in Navigation Bar
  + Profile Page
    - Name
    - Email
    - Change Password
    - Upload Photo
  + Course Page
    - Navigation Bar
      * Home
      * Grades
      * Assignments(DropBox, Quiz, Survey) Dropdown menu
      * Profile
      * Logout
  + Cloud
    - Assignment
    - Score
    - Date (Start, Due)
    - Submissions
  + Grades
    - Assignment
    - Score
    - Average
  + Survey
    - Assignment
    - Complete (or not)
    - Date
  + Quiz
    - Assignment
    - Date (Start, Due)
    - Number of Attempt







**MEMORANDUM**

Date: March 13, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Cameron Morgan

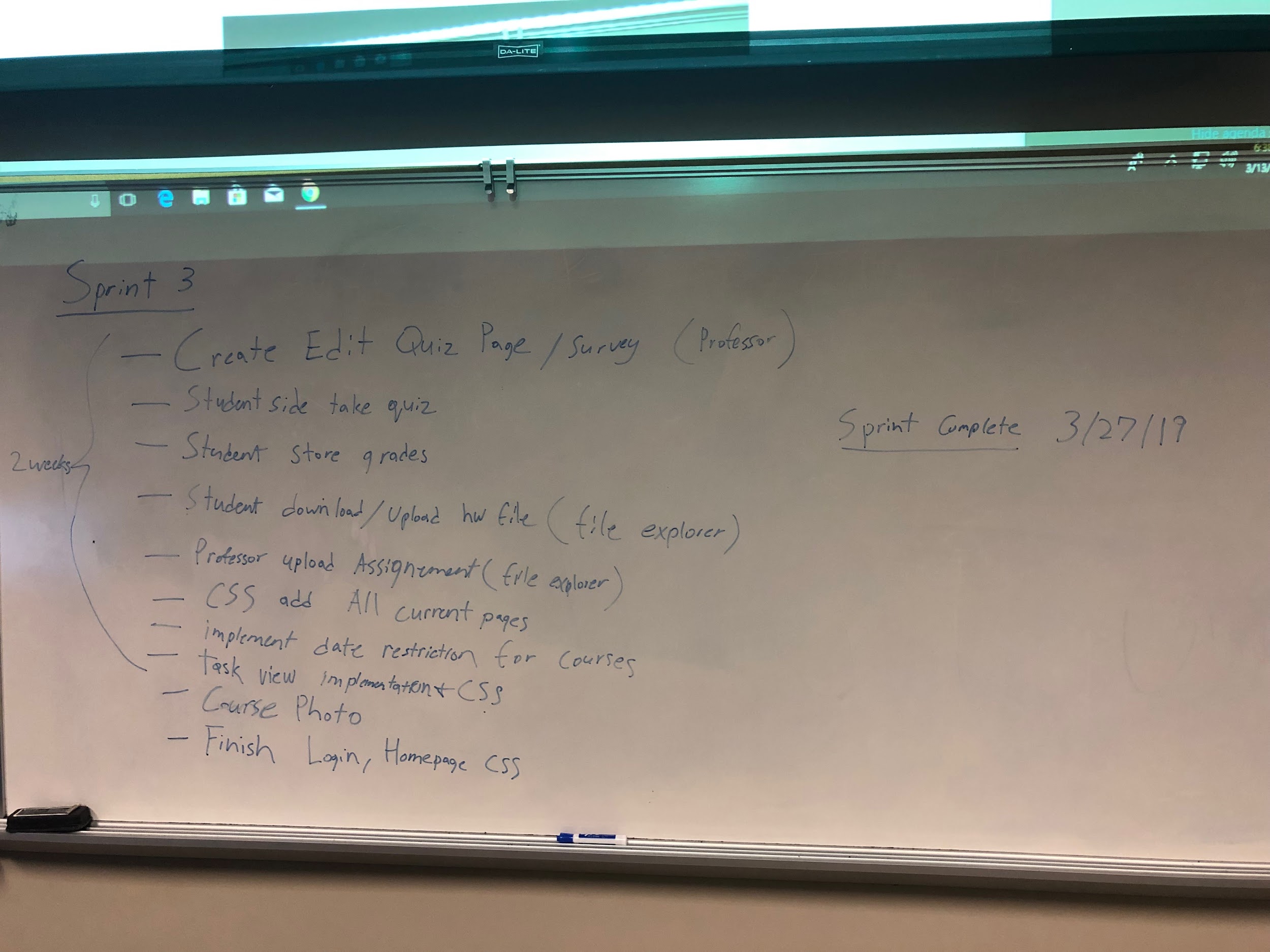
Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Trello has officially been linked to GitHub
* Reading from text file logic is complete
* Navigation bar set up for web page complete
* Profile content added to user module
* Need to add Course start/end (datetime) - unique identifier (ex. 201808 = Fall 2018)-Per client meeting
* EDITS TO BE MADE:
  + Course List tab added to Navigation bar
  + Profile should have(Name, Email, Change Password, Upload Photo)
* Planned Sprint 3 (See trello Assignments to cards), SEE PHOTO
  + Due March 27, 2019



**MEMORANDUM**

Date: March 13, 2019

A meeting was held for the referenced project. The following were in attendance:

Andrew Allen

Kym Haywood II

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 12:30 pm. The following items were discussed:

Progress Update

Database setup to store user data

Tables for the associations of the users

Students. Professors, Admin

Working on file read in

Showing sprint 1 progress

Client

Homework/Quiz deadlines lock (datetime)

Live tracking surveys

Course start/end (datetime) - unique identifier (ex. 201808 = Fall 2018)

**MEMORANDUM**

Date: March 27, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey- Left Early

Cameron Morgan

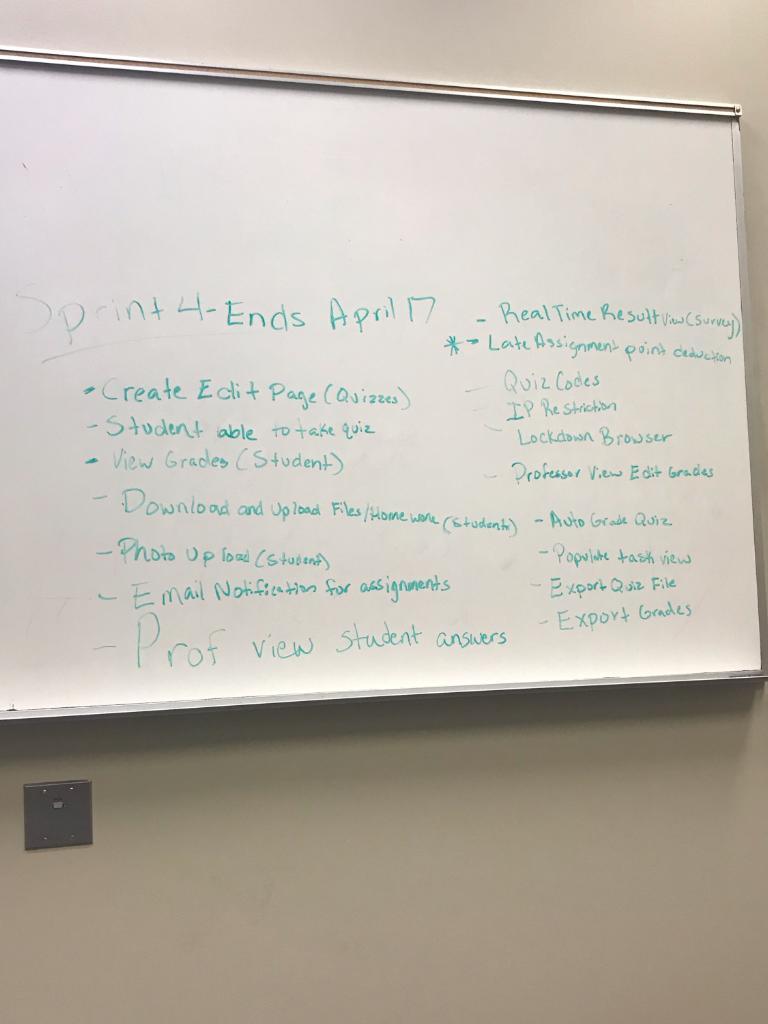
Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Planned Sprint 4 (See trello Assignments to cards), SEE PHOTO
  + Due April 17, 2019
* Discussed spring break progress
* Made CSS edits



**MEMORANDUM**

Date: April 3, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 2:00 pm. The following items were discussed:

* Worked on Edit Page(Should be done tomorrow)
  + Dynamically can create question
* Worked on Auto Grade logic
* Upload Photo/quiz/hw works
* Change password works

**MEMORANDUM**

Date: April 7, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 2:00 pm. The following items were discussed:

* Continued work on Sprint 4
* Working on Design Document
* List of Quizzes viewable for students
* Started Logic for auto grading
* Researching email notification logic

**MEMORANDUM**

Date: April 10, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey- Left Early

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Continued work on Sprint 4
* Edit page is complete
* Auto grade has been implemented into code
* Working on Sprint 4 checkpoints
* Code to send basic email has been created.
  + Needs to be implemented into code
* Hit checkpoints for tonight’s meeting

**MEMORANDUM**

Date: April 14, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II (tardy)

Cameron Morgan

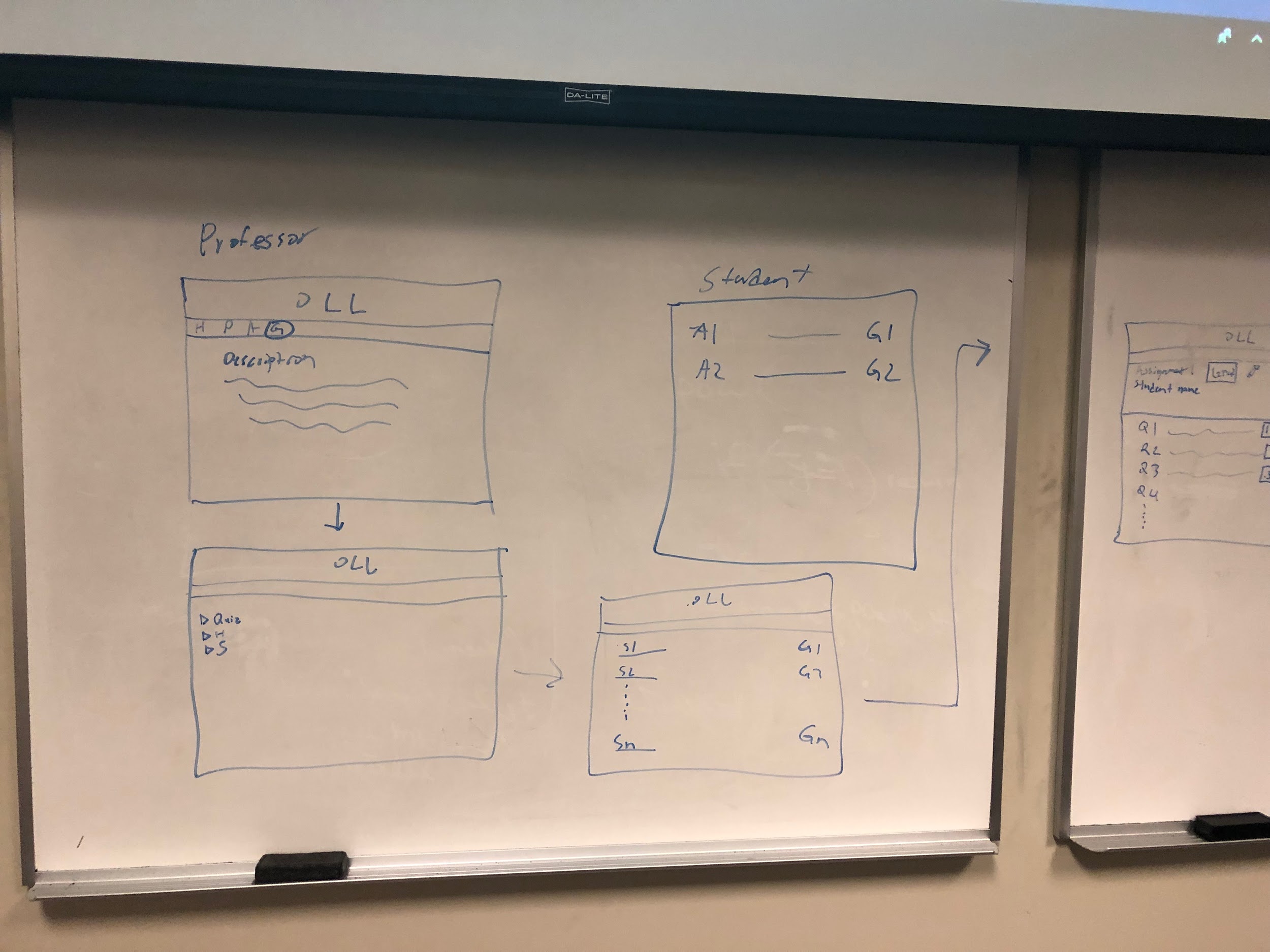
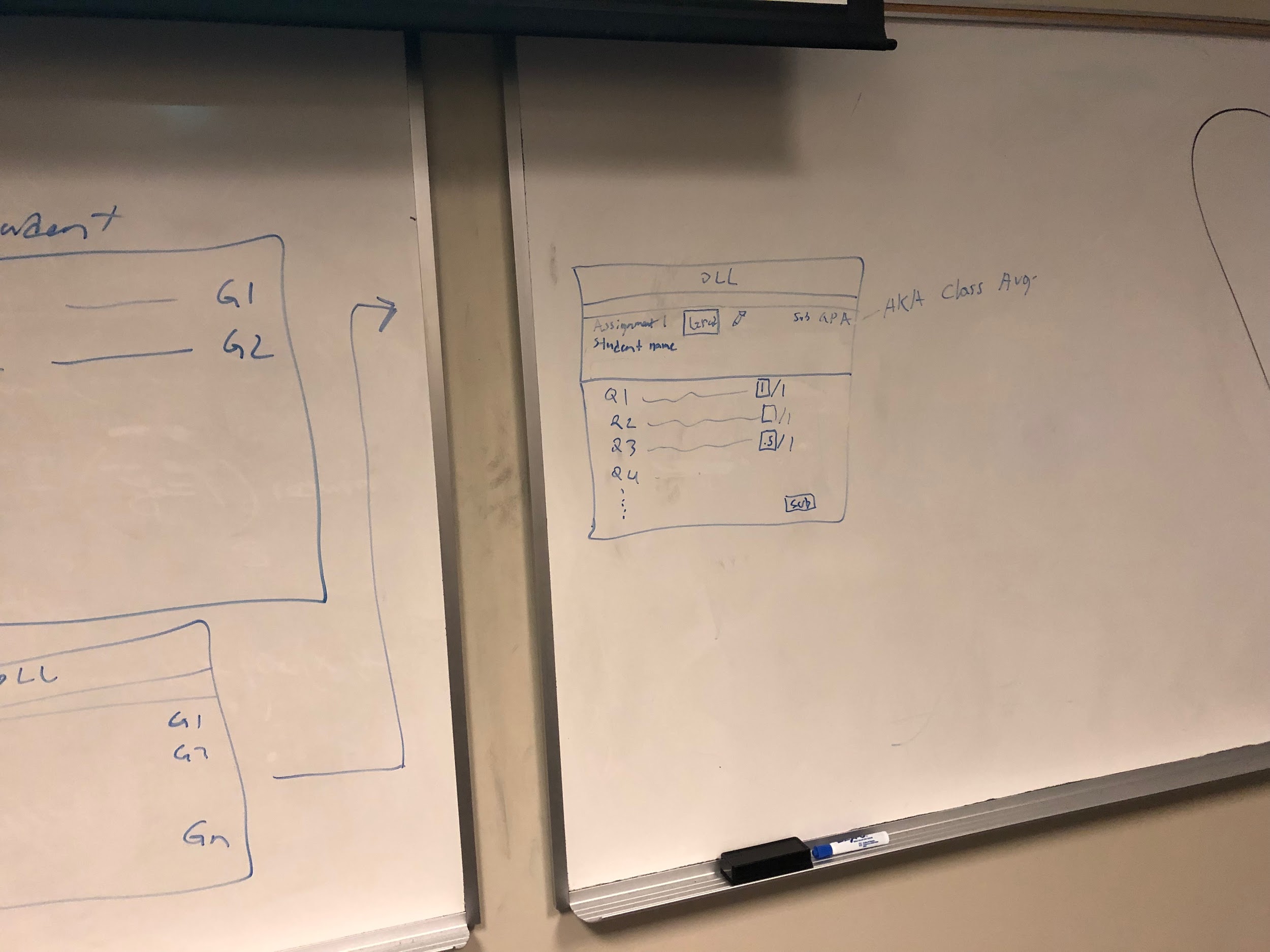
Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 2:30 pm. The following items were discussed:

* Continued work on Sprint 4
* Discussed grade pages and their setup (SEE PHOTO)
* Discussed Senior Project Symposium Poster



**MEMORANDUM**

Date: April 17, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey- Left Early

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 4:00 pm. The following items were discussed:

* Student grades viewable
* Working on Project

**MEMORANDUM**

Date: April 24, 2019

A meeting was held for the referenced project. The following were in attendance:

Kym Haywood II

Noah Hickey- Left Early

Cameron Morgan

Austin Simmons

Chinonyerem Ugwumadu

Shamarcus Walker

The meeting was held at Georgia Southern University in the IT Building at approximately 5:30 pm. The following items were discussed:

* Working on Project
* Finish Sunday (4/28/2019)

# References

1.Omer, A. H. (2018, April 20). LMS Features: What To Look For In A Learning Management System. Retrieved from <https://elearningindustry.com/lms-features-what-look-learning-managent-system>

2. Elearning Home. (n.d.). Retrieved March/April, 2019, from <https://elearning.adobe.com/>

3. XAPI.com Homepage: What is xAPI (the Experience API). (n.d.). Retrieved from <https://xapi.com/>

4. Moodle - Open-source learning platform. (n.d.). Retrieved from https://moodle.org/