CoCo: Collaborative Courses

by Team ; DROP DATABASE;--

SLACK CHANNEL: https://csc483w18.slack.com

Final Document

For: Dr Mark Allison

Table of Contents

Abstract

The CoCo system is designed for students to access course materials online, and outside of the normal course times. The students will have access to each of their registered courses for the semester. This will range from students being able to find their instructor's contact information, to accessing online homework assignments, to chatting with students, tutors, and instructors in real-time.

There will also be a feature in the system that will allow students to take notes, and also be able to give access to other students to be able to collaborate during the class time to take notes, enhancing the students' learning capabilities. The instructor(s) and tutor(s) will also be able to access these notes in order to help clarify any unclear topics of the lecture and to make sure that the student is learning the techniques correctly.

1. Introduction

1.1. Purpose of System

The purpose of CoCo is to allow students to have access to their course resources online. This will range from being able to view digital assignments, to being able to view grades that the grader or instructor have posted for a given assignment, to being able to chat and share collaborative notes with other students that are also in the class.

1.2. Scope of System

The scope of the system will be restricted to students, tutors, graders, the instructor(s) for the course, as well as the administrators of the system. While the system will only be designed for courses that plan to implement digital resources for their students, it will still be able to be used if the instructor only plans on tracking grades.

1.3. Development Methodology

Our team will be developing the CoCo software utilizing the agile methodology. This will allow for rapid development cycles, while focusing on producing a quality piece of software for the customer. This will also allow our team to adapt to any changes in the requirements, should they come up during development. Choosing agile development will increase the chances of our team producing a high-quality product in the end that satisfies all of the needs of the customer.

1.4. Definitions, Acronyms, and Abbreviations

Actors: External entities that interact with the system.
Advisor: A faculty member at the University of Michigan who gives advice to the students, looks over their class schedule and guides them in getting their intended degree.
DD: Design Document.
Deliverable: Work product for client.
Active Courses: Courses that are offered and running for the current semester

1.5. Overview of Document

The rest of the document to follow is our project plan, requirements of the system, software architecture, object design, and testing process. Each of these sections are broken up into their own chapters.

2. Current System

Not applicable due to this system being completely new.

3. Project Plan

3.1. Project Organization

Phase 1 1/20/2018-2/19/2018

Tyler Elton	Leader and System Designer
Gary Landrum	Documentation and Backend Setup
Zach Nelson	Backend Lead
Brandon Krug	Programming
Jason Moehlman	Programming Documentation

Phase 2 2/20/2018-4/1/2018

Tyler Elton	Testing Lead
Brandon Krug	Testing
Jason Moehlman	Testing and Testing Documentation
Gary Landrum	Backend Testing and Documentation
Zachary Nelson	Backend Testing and Database Management

3.2. Software and Hardware Requirements

Hardware: At least 3 virtual machines with 1 dual core processor, 4GB of RAM and 20GB Hard disk space each. Machines need interconnectivity between each other and all machines require internet access.

Software: PHP 7.1 (using the Laravel 5.5 LTS framework), Composer, NPM, MySQL, Apache

3.3. Work Breakdown

Task #	Task	Description	Duration	Dependencies
1	Project Plan	Form team. Agree upon project idea and formulate PRD.	7 days	
2	Create use cases	Identify use cases for common actors. Compare use cases for teach team member together. Present Use Cases to Dr. Allison	10 days	1
3	Develop Personas	Further develop personas. Develop test cases for personas.	7 days	1
4	Identify Architecture	Identify the architecture of the system. Create a class diagram based on test cases and architecture	10 days	2/3 (M1) (D1)
5	Finalize design document	Finalize the design document based on class diagram created in 4. Assign subsystems of the program to develop for team members.	7 day	4(M2)
	Develop Core	Divide project into subsystems, identify		5(D2)

6	Functionality	objects, complete design document,	21 days	
7	Unit Test Core Functionality	Transition of software models into source code.	14 days	6
8	Implementation of Core Features	Database design. Deploy software on backend.	14 days	27 (M3)(D3)
9	Create SFD	Compile SFD based on documentation created during semester. Create powerpoint for SFD	16 days	8
10	Present SFD	Present powerpoint and submit SFD.	1 days	9(M4) (D3)

M – Milestone D – Deliverable

4. Requirements of System

4.1. Functional and Nonfunctional Requirements

Functional Requirements:

- Users must be able to sign into the system successfully. This applies to all actors.
- The student user/actor must be able to
    - View all active courses they are a member of
    - View all available content within those courses
    - Upload assignments within due date window
    - View their grade in the course
    - Participate in the Live Chat Feed for a course and tutoring appointments
    - Sign up for tutoring sessions for each of their courses
        - Manage their appointments
- The Grader user/actor must be able to
    - View all active courses they are a member of
    - View all available content within those courses
    - View all assignments that are due and assigned in each of those courses
    - Grade all assignments that are uploaded for a given course
    - View all grades that are posted within the course
    - Notify the instructor when an assignment(s) have been graded
- The Tutor user/actor must be able to
    - View all active courses that they are a tutor for
    - View all available assignments within each of those courses
    - Manage tutoring sessions/appointments
    - Create Live Chat Feeds for tutoring appointments
    - Participate in the Live Chat Feed for each course and in tutoring appointments
    - Create notes on tutoring session
- The Instructor user/actor must be able to
    - View all active courses they are teaching or assisting on
    - Create content for their courses
    - Grade all assignments that are uploaded for each course
    - Approve grades that graders submit for each course
    - Assign students to the course and section for active courses
    - Export Grades for each course they are teaching
    - Participate in the Live Chat Feed for each course they are assigned to
    - Create Live Chat Feeds for courses they are teaching

- The Admin user/actor must be able to
    - Have full management rights over the system
    - Access to any course to troubleshoot any issue that may arise

Nonfunctional Requirements:

- The system must provide secure and reliable log on capabilities for users.
- The system must respond to user interactions quickly and efficiently.
- The system must be properly secured from a physical standpoint
- The system must encrypt all traffic and its databases
- The system must be FERPA Compliant

4.2. Identified Personas

- Mike - Student / Tech Savvy / High Speed Internet @ Home / Likes to customize / Compulsively checks current LMS for updates
- Gertrude - Student / Not Tech Savvy / Lives out of town / Wants to spend as little time on the system as possible.
- Kim - Instructor / Puts a lot of effort into blackboard courses / Uses third party tools to keep in contact with students / Grades quickly
- Jebediah - Instructor / Does not customize shell / Only communicates via phone / Grades at a set time / No after hours activities

4.3. Use Case Diagram

The next figure depicts the interaction between the actors and the previously described use cases. A description for each actor follows.

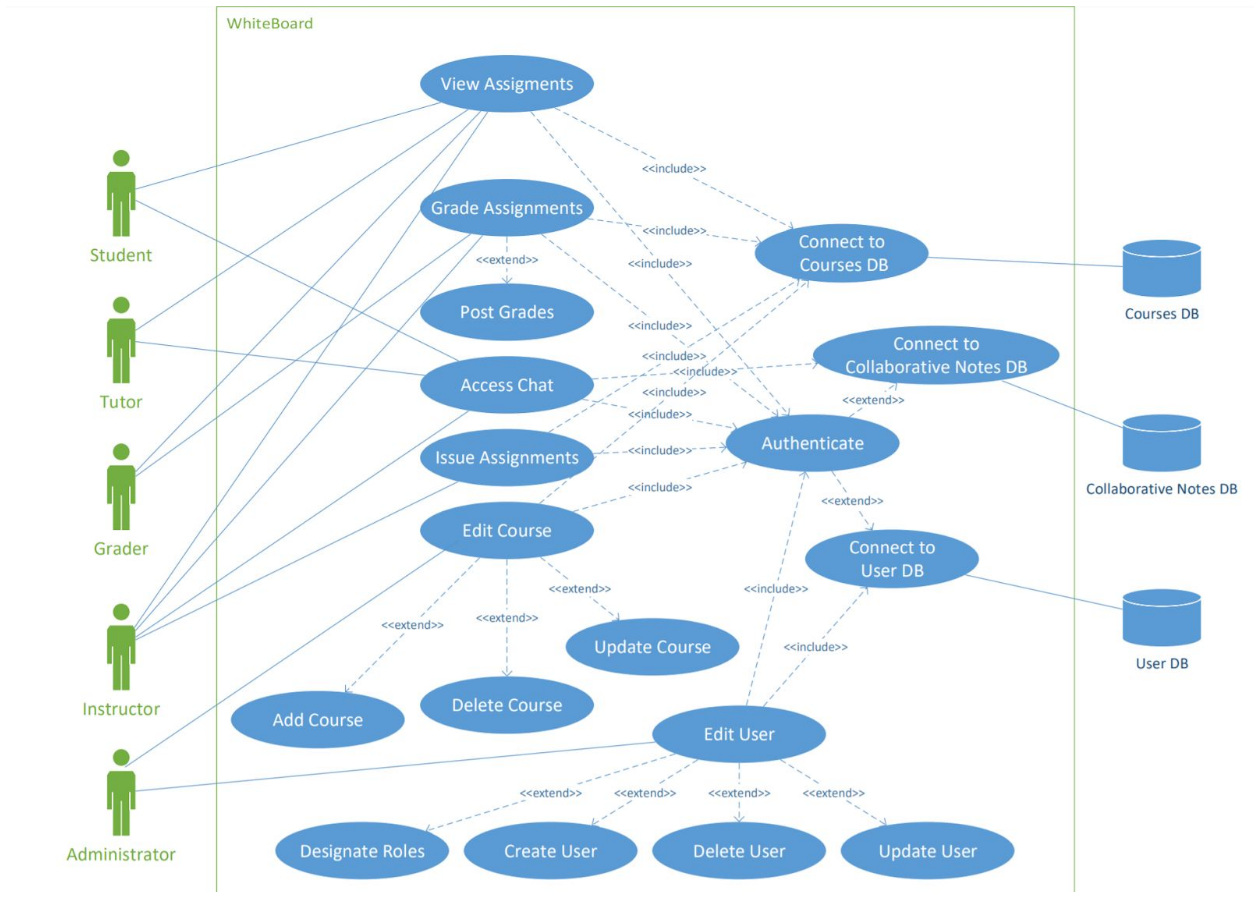


Figure 1: Use Case Diagram

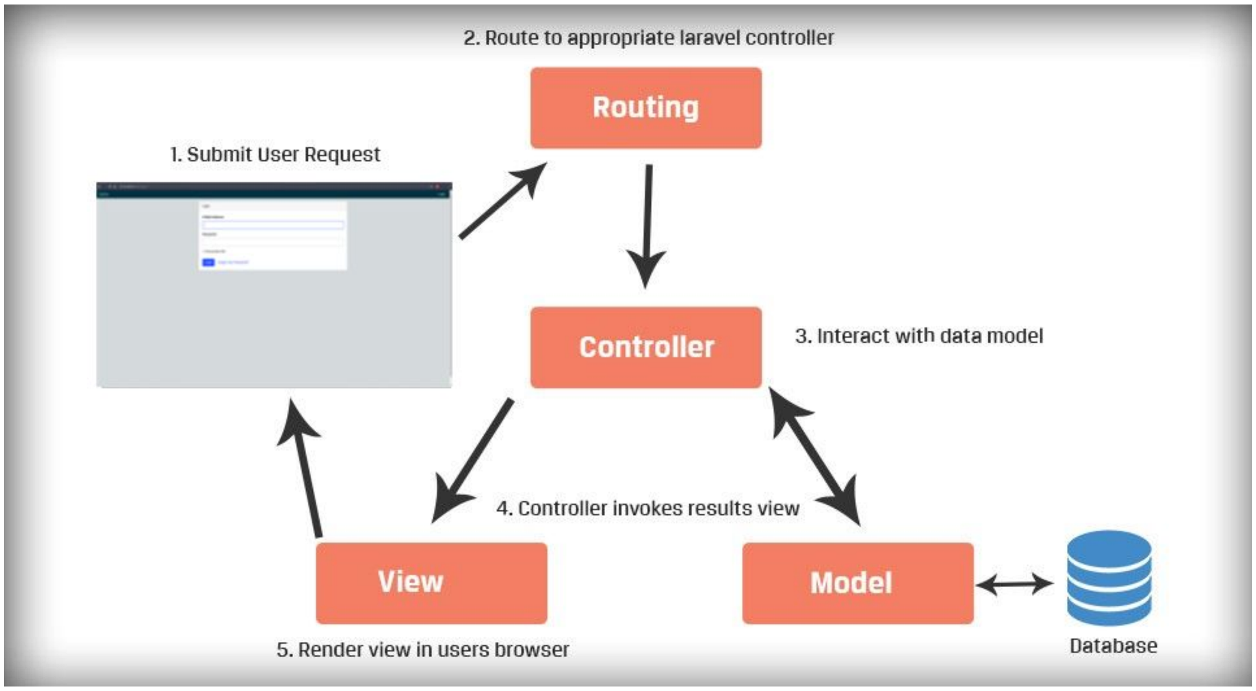
4.4. Requirements Analysis

After identifying the client's requirements we, as system designers, are in a position to devise the most appropriate solution. By using the use cases to capture all potential requirements, the team will devise the most suitable software architecture, which meets the user's needs. This will ensure the attainment of customer's expectations as well as providing a functional final product.

5. Software Architecture

5.1. Overview

After evaluating and analyzing the main structure and functionalities of our project, the highly utilized Model View Controller (MVC) architecture was chosen for our project. This architecture facilitates the separation of data, logic, and visual elements, and is particularly suitable for web applications such as this. In this architecture, objects are assigned one of three roles: model, view, or controller. Model objects encapsulate the data which is specific to the application, while also defining the computational logic which will manipulate said data. View objects pertain to the way the data is displayed on the screen to the user. Controller objects act as a middleman between the Model and View objects, updating views as the data changes and vice versa.


2. Route to appropriate laravel controller
Routing
1. Submit User Request
Controller
3. Interact with data model
4. Controller invokes results view
View
Model
Database
5. Render view in users browser

5.2. Subsystem Decomposition

Login Subsystem - This is composed of the user class and its generalizations, as well as the login screen, login authenticator, and userDbConnector class. This subsystem will be the landing point for a user first entering the system and is responsible for authenticating the user to the system as well as authorizing the user to access the appropriate courses.

Course Subsystem - The course subsystem is the largest subsystem in CoCo and contains the classes that carry out most of the business logic of the system. This includes the courseShell, Course, Assignment, gradeCalculator, courseDbConnector, fileCreator, and schoolInfoSysDbConnector. The subsystem will also contain the screen and menu classes for the aforementioned processes (course menu, assignments menu, course creation menu) This subsystems contains all the courses needed to create and deliver a course through coco.

Chat Subsystem - The chat subsystem will be operating in parallel with other CoCo subsystems. As any time a user is authenticated to CoCo their chat feed will be updated regularly for all enrolled courses. The chat subsystem contains the chatDbConnector, chatChannelLoaded, as well as the chatMenu.

5.4. Persistent Data Management

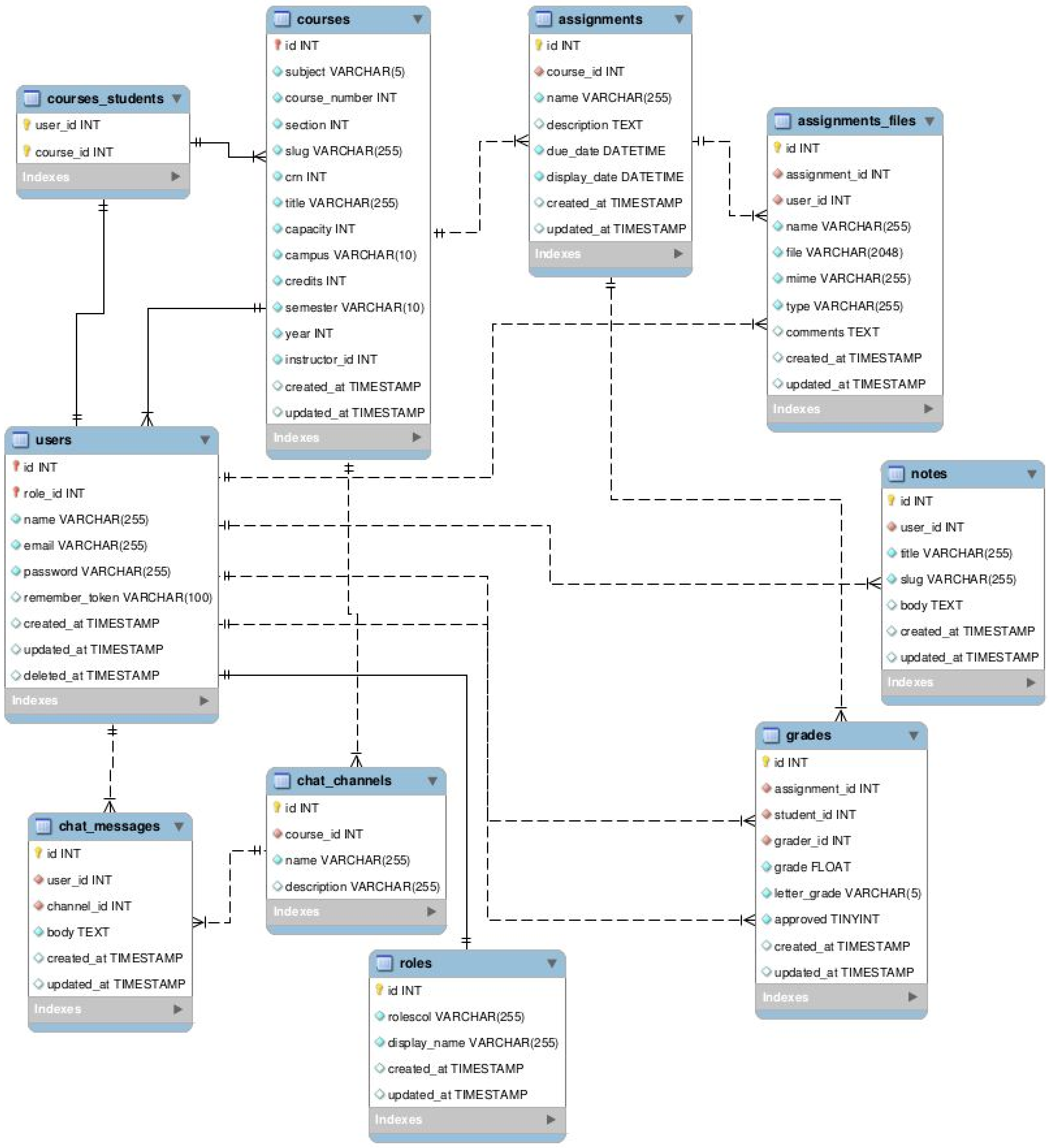


Figure 1: Entity Relation Diagram

All data will be stored in a MySQL database. The application will connect to the database in order to retrieve records and save new/existing records.

6. Object Design

This chapter will build on the architecture and design.

6.1. Overview

The primary class objects revolve around users, courses, assignments, grades, chat, menus, utility functions, screens, and connectors. Users consist of the students, tutors, instructors, and administrators who use the system, each with their own permissions and attributes. Connectors are used to interact with database, facilitating the reading and writing of data. Utility functions are classes which perform some kind of computation or task, such as the grade calculator, file creator, and chat channel loader. The users will navigate the system via the screens/menus.

6.2. Object Interaction

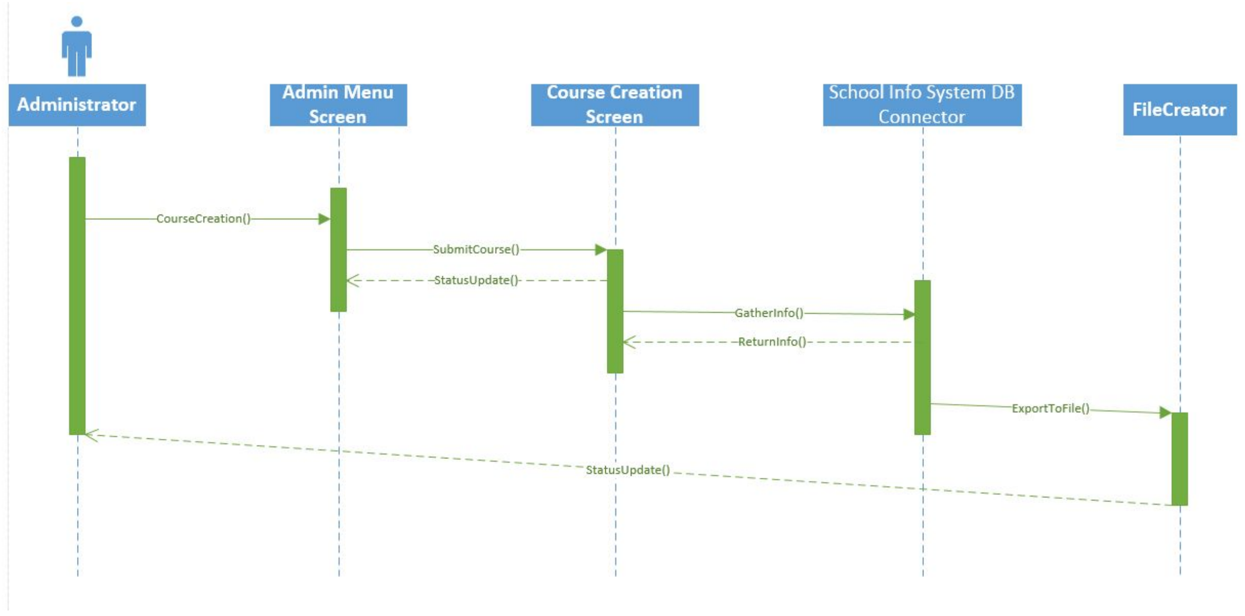


Figure 2: Sequence Diagram for Course Creation

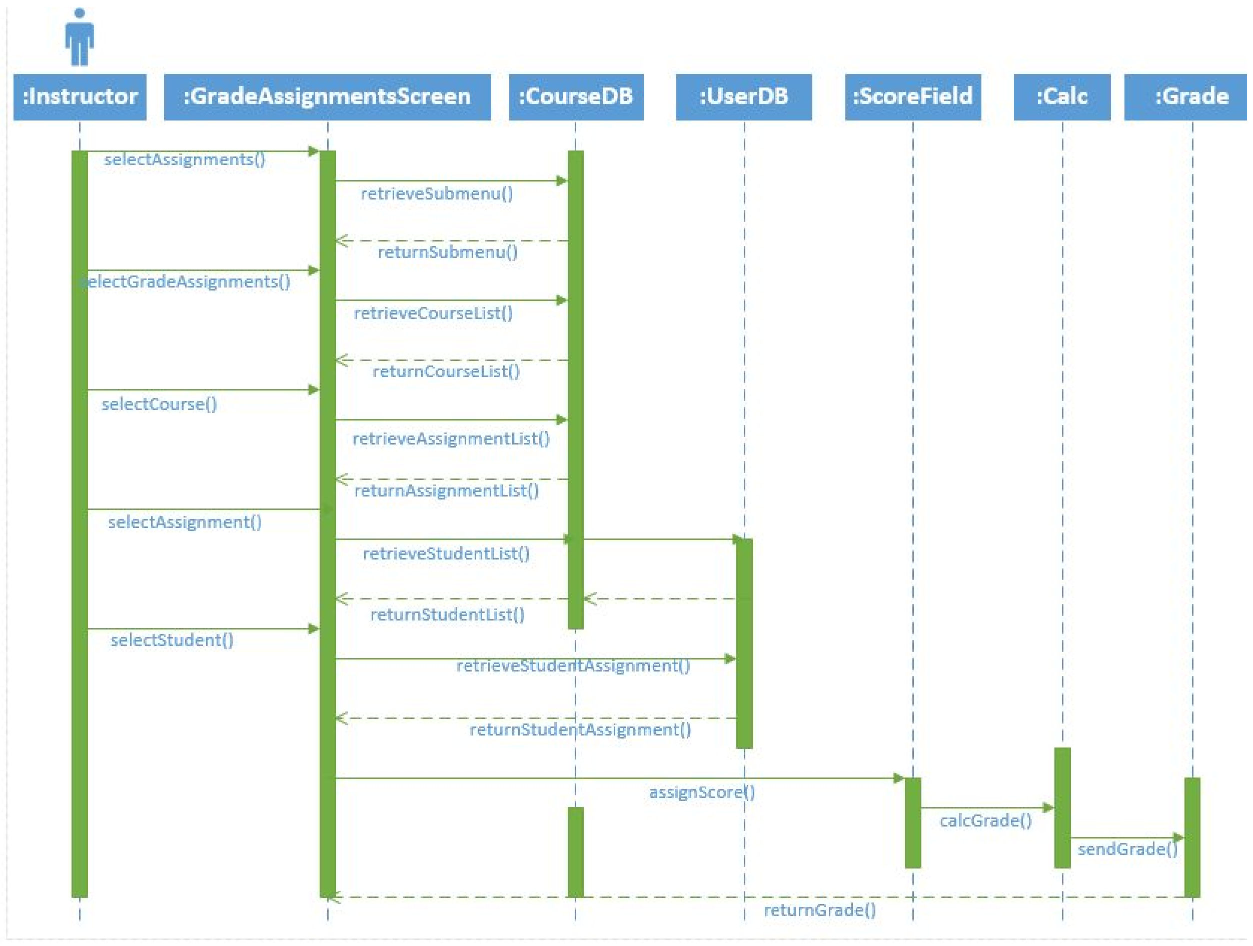


Figure 3: Sequence Diagram for Instructor Grading

6.3. Detailed Class Design

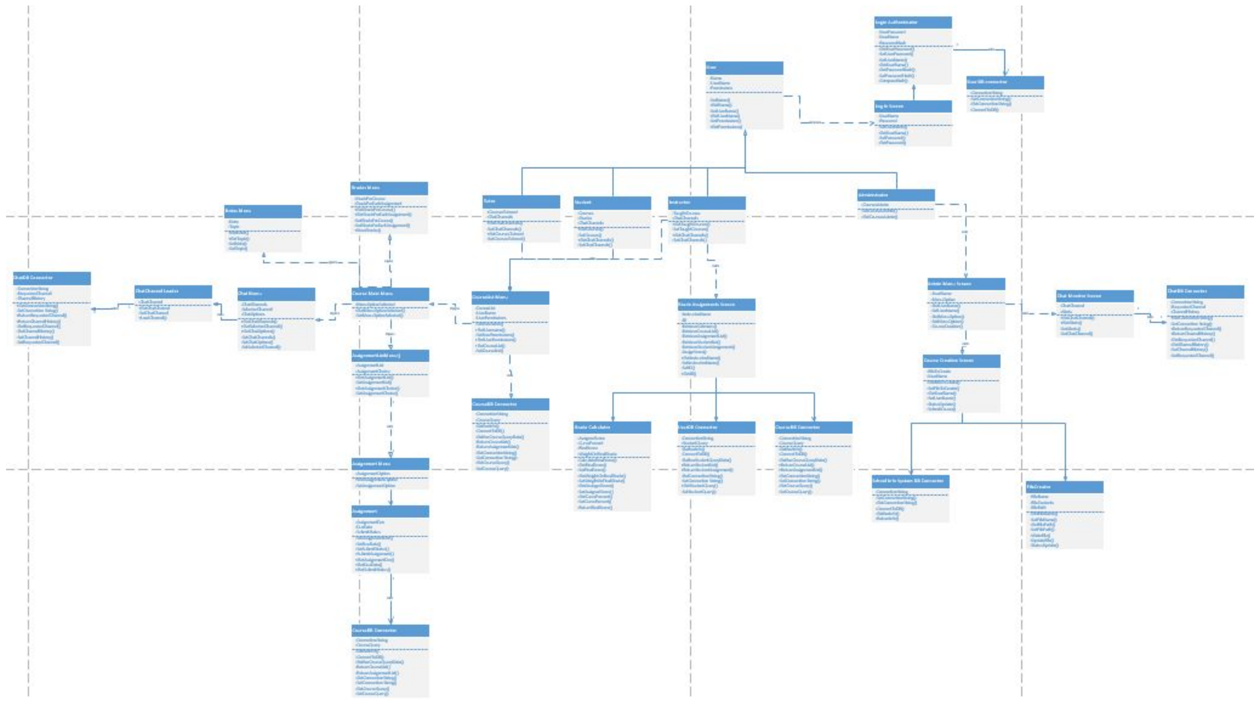


Figure 4: Detailed Class Diagram for the Overall System

7. Testing Process

7.1. User Experience Tests

7.2. Systems Tests

7.3 Subsystems Tests

8. Glossary

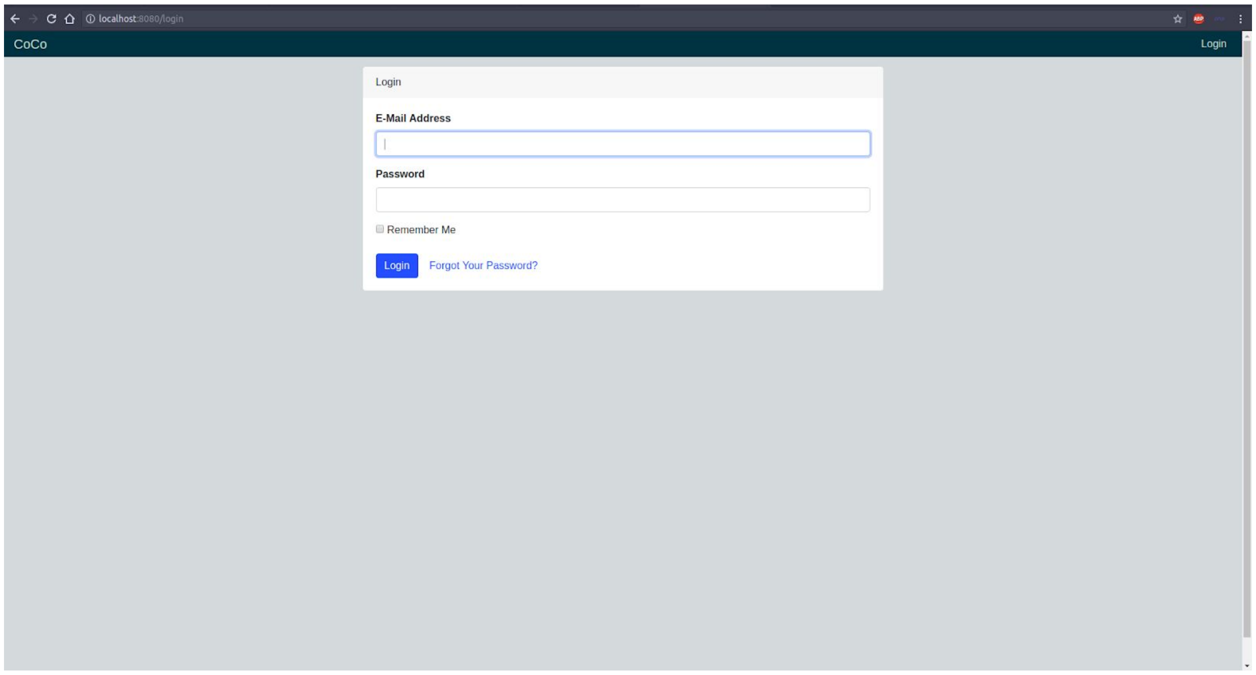
9. Appendix

9.1. Appendix A – Gantt Chart

Figure 5: Gantt Chart

9.2. Appendix C – User Interface Designs

<u>NOTE:</u> All designs are *in progress* and are subject to change.



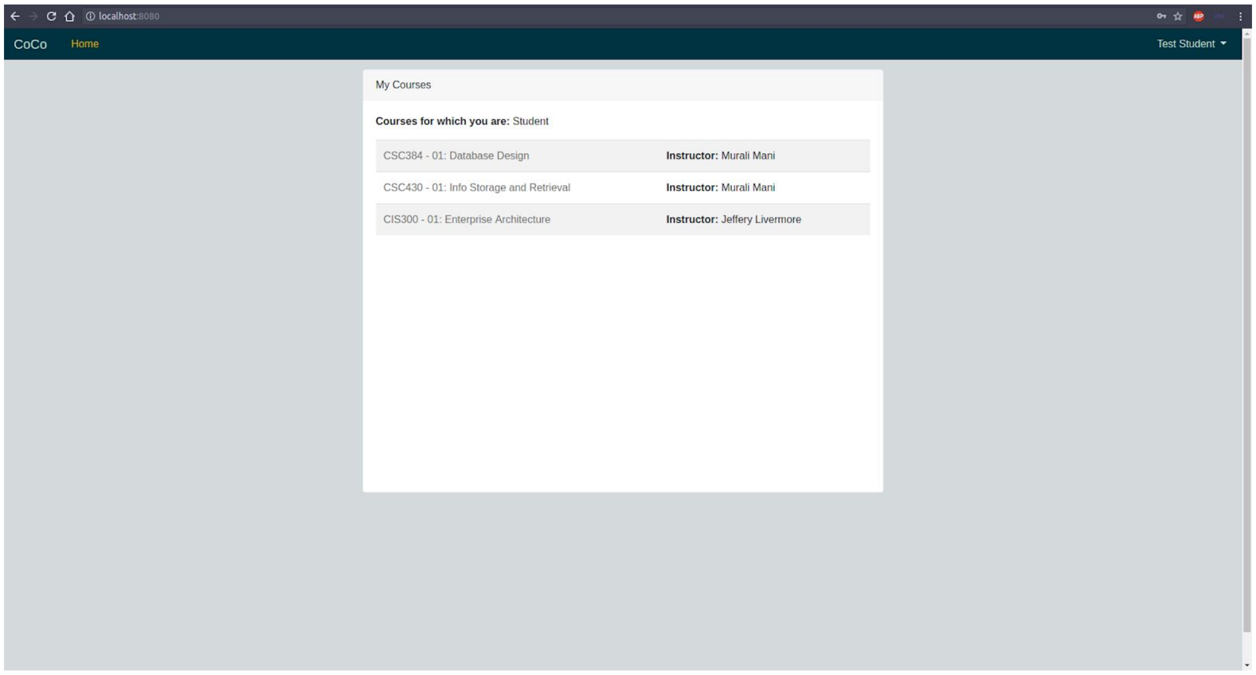
Login

E-Mail Address

Password

Remember Me

Figure 1: Login Screen

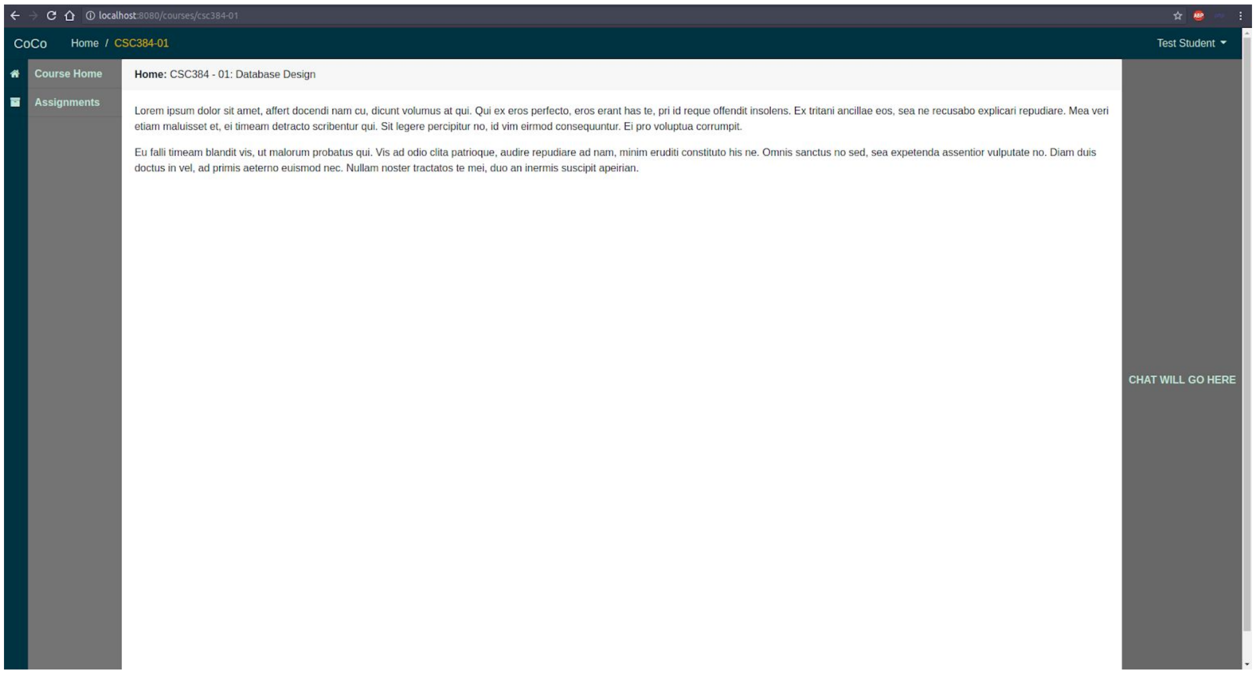


My Courses

Courses for which you are: Student

CSC384 - 01: Database Design	**Instructor:** Murali Mani
CSC430 - 01: Info Storage and Retrieval	**Instructor:** Murali Mani
CIS300 - 01: Enterprise Architecture	**Instructor:** Jeffery Livermore

Figure 2: Homepage (after logging in)

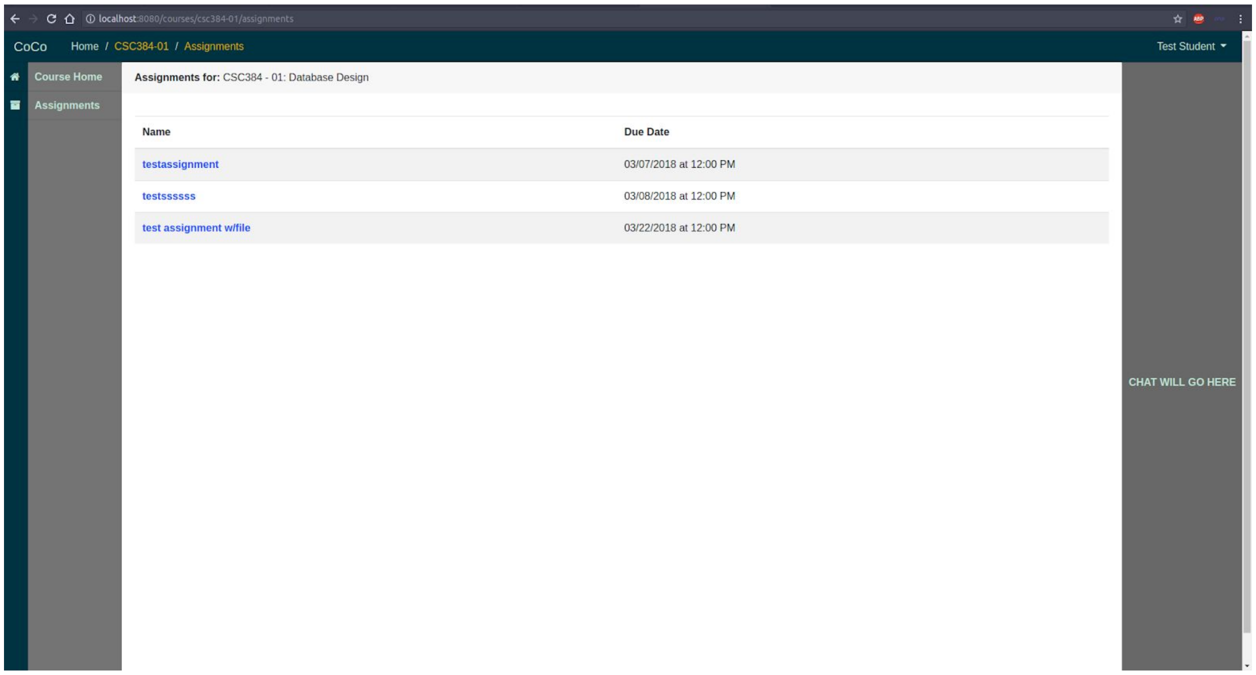


Home: CSC384 - 01: Database Design

Lorem ipsum dolor sit amet, affert docendi nam cu, dicunt volumus at qui. Qui ex eros perfecto, eros erant has te, pri id reque offendit insolens. Ex tritani ancillae eos, sea ne recusabo explicari repudiare. Mea veri etiam maluisset et, ei timeam detracto scribentur qui. Sit legere percipitur no, id vim eirmod consequuntur. Ei pro voluptua corrumpit.

Eu falli timeam blandit vis, ut malorum probatus qui. Vis ad odio clita patrioque, audire repudiare ad nam, minim eruditi constituto his ne. Omnis sanctus no sed, sea expetenda assentior vulputate no. Diam duis doctus in vel, ad primis aeterno euismod nec. Nullam noster tractatos te mei, duo an inermis suscipit apeirian.

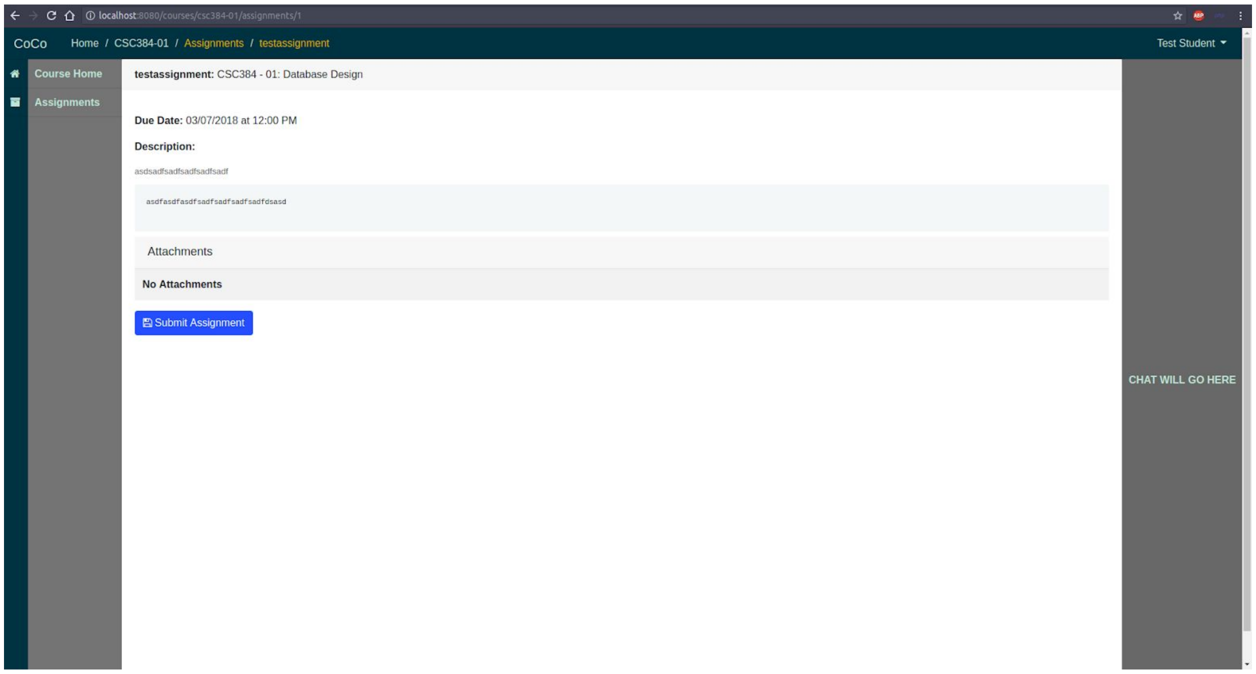
Figure 3: Course Homepage


localhost:8080/courses/csc384-01/assignments
CoCo
Home / CSC384-01 / Assignments
Test Student
Course Home
Assignments
CHAT WILL GO HERE

Assignments for: CSC384 - 01: Database Design

Name	Due Date
testassignment	03/07/2018 at 12:00 PM
testssssss	03/08/2018 at 12:00 PM
test assignment w/file	03/22/2018 at 12:00 PM

Figure 4: Course Assignment Listing Page


localhost:8080/courses/csc384-01/assignments/1
CoCo
Home / CSC384-01 / Assignments / testassignment
Test Student
Course Home
Assignments
Submit Assignment
CHAT WILL GO HERE

testassignment: CSC384 - 01: Database Design

Due Date: 03/07/2018 at 12:00 PM

Description:

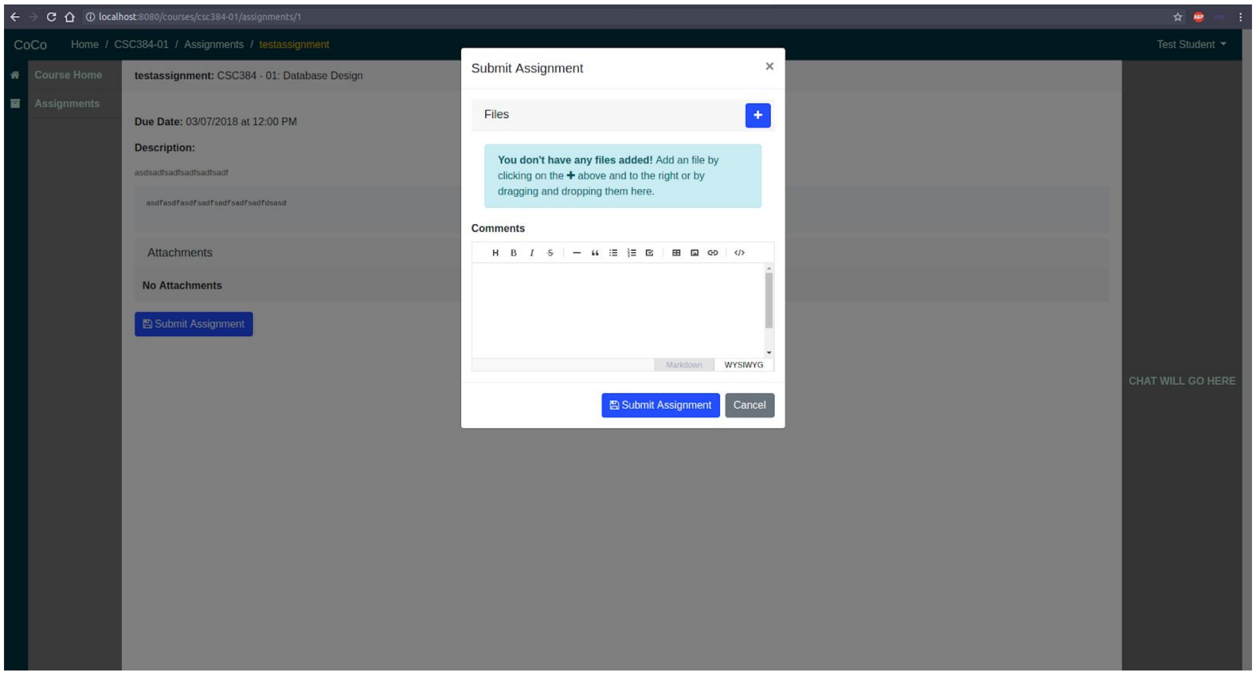
asdsadfsadfsadfsadfsadf

```
asdfasdfasdfsadfsadfsadfsadfdsasd
```

Attachments

No Attachments

Figure 5: Course Assignment Show Page



You don't have any files added! Add an file by clicking on the + above and to the right or by dragging and dropping them here.

Figure 6: Course Assignment Submit Modal

9.3. Appendix E – Class Interfaces for Implemented Subsystems

Interface Subsystem

Browser.java

```
package Interface;
public class Browser
{
public void enterData()
{}

public void validateData()
{}

public void submitData()
{}

public void displaySchedules()
{}

public void createScheduleLink()
{}

public void displayPage()
{}
public void viewSavedScheduleLink()
{}

public void checkSemesterBalanceLink()
{}

public void displayBalance()
{}

public void enterLoginInfo()
{}

public void clicksLogOut()
{}
}
```

Application Logic Subsystem

Authentication.java

```
package ApplicationLogic;
public class Authentication
{
public void createSession()
{}

public void closeSession(String sessionID)
{}
}
```

FormatPage.java

```
package ApplicationLogic;
import Storage.Schedule;
public class FormatPage
{
public void createPage()
{}

public void createPage(Schedule schedule[])
{}

public void buildPage()
{}

public void requestPage(String pageName)
{}
}
```

LoginOption.java

```
package ApplicationLogic;
public class LoginOption
{
private String pantherID;
private String password;

public LoginOption(String pantherID, String password)
{}

public String getPantherID()
{}

public void setPantherID(String pantherID)
{}

public String getPassword()
{}
public void setPassword(String password)
{}
}
```

ScheduleMakerController.java

```
package ApplicationLogic;
public class ScheduleMakerController {

private boolean conflict(Collection c)
{}

private boolean sizeIsOne(Collection c)
{}
private void findSchedule(Collection<Collection<ClassDetails>> c,           int cursor, Collection<Schedule>
schedules)
{}

public Collection<Schedule> createSchedule(ScheduleOptions schOpt)
{}

public Collection<Schedule> createSchedule(String term,           Collection<String> courses, String cmp,
String SPDays)
{}
```