WhiteBoard

by Team: DROP DATABASE

SLACK CHANNEL: [https://csc483w18.slack.com](https://csc483w18.slack.com/)

Final Document

For: Dr. Mark Allison

On: 23 January 2018

Tyler Elton

Zach Nelson

Jason Moehlman

Brandon Krug

Gary Landrum

**Table of Contents**

**Abstract………………………………………………………………………….. 4**

*1.1. Purpose of System………………………………………………………….4*

*1.2. Scope of System…………………………………………………………….4*

*1.3. Development Methodology…………………………………………………5*

*1.4. Definitions, Acronyms, and Abbreviations………………………………..5*

*1.5. Overview of Document……………………………………………………. 5*

**2. Current System……………………………………………………………… 5**

**3. Project Plan………………………………………………………………….. 5**

*3.1. Project Organization………………………………………………………. 6*

*3.2. Software and Hardware Requirements…………………………………...7*

*3.3. Work Breakdown…………………………………………………………… 7*

**4. Requirements of System……………………………………………………9**

*4.1. Functional and Nonfunctional Requirements……………………………..9*

*4.2. Identify Personas…………………………………………………………10*

*4.3. Use Case Diagram…………………………………………………………..11*

*4.4. Requirements Analysis……………………………………………………...11*

**5. Software Architecture………………………………………………………..12**

*5.1. Overview………………………………………………………………………12*

*5.2. Subsystem Decomposition………………………………………………….12*

*5.4. Persistent Data Management………………………………………………12*

**6. Object Design…………………………………………………………………12**

*6.1. Overview……………………………………………………………..............12*

*6.2. Object Interaction ……………………………………………………………12*

*6.3. Detailed Class Design………………………………………………………13*

**7. Testing Process………………………………………………………………13**

**8. Glossary………………………………………………………………………..13**

**9. Appendix……………………………………………………………………….13**

*9.1. Appendix A – Gantt Chart…………………………………………………..13*

*9.2. Appendix B – Use Cases …………………………………………………...13*

*9.3. Appendix C – User Interface Designs……………………………………..13*

9.5. Appendix E – Class Interfaces for Implemented Subsystems………*14*

# 

# 

# 

# 

# 

# **Abstract**

The WhiteBoard system is designed to allow student access to course materials online regardless of location and/or time. Students will have access to each of their registered courses for the semester. Allowable tasks will range from students being able to find instructor 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 if they should choose to, give access to other students for collaborative efforts in note-taking during the scheduled period of instruction. This will enhance the student learning capabilities. The instructor(s) and tutor(s) will also be able to access these notes to provide clarity on topics the student is unsure of. This makes it possible for the instructor and/or tutor to determine whether the student is learning the prescribed techniques correctly.

**1. Introduction**

## **1.1. Purpose of System**

The purpose of WhiteBoard 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 have been posted by the grader or instructor to being able to chat and share collaborative notes with other students in the class.

## **1.2. Scope of System**

The scope of the system will be restricted to students, tutors, graders, instructor(s) for the course, and 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 WhiteBoard 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 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 students, looks over their class schedule, and guides them in reaching their intended degree goal.

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 remainder of this document consists of our project plan, requirements of the system, software architecture, object design, and testing process. Each applicable chapter is broken into sections for easier review.

# **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. All machines need interconnectivity between each other and require internet access.

Software: *PHP 7.0+ (using the Laravel 5.5 framework), Composer, NodeJS/NPM, MySQL*

## 

## **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 days | 4(M2) |
| 6 | Develop Core Functionality | Divide project into subsystems, identify objects, complete design document. | 21 days | 5(D2) |
| 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 tutor
  + 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 any course to troubleshoot any issue that may arise

Nonfunctional Requirements:

* The system must provide secure and reliable logon 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**

1. Tony Stark - *administrator*
2. Dr. Bruce Banner - *instructor*
3. Jennifer Walters - *grader*
4. Steve Rogers - *tutor*
5. Anna Marie - *student*

Each persona reflects the roles that the system will use to differentiate the permission levels of each user.

## 

## 

## 

## 

## 

## 

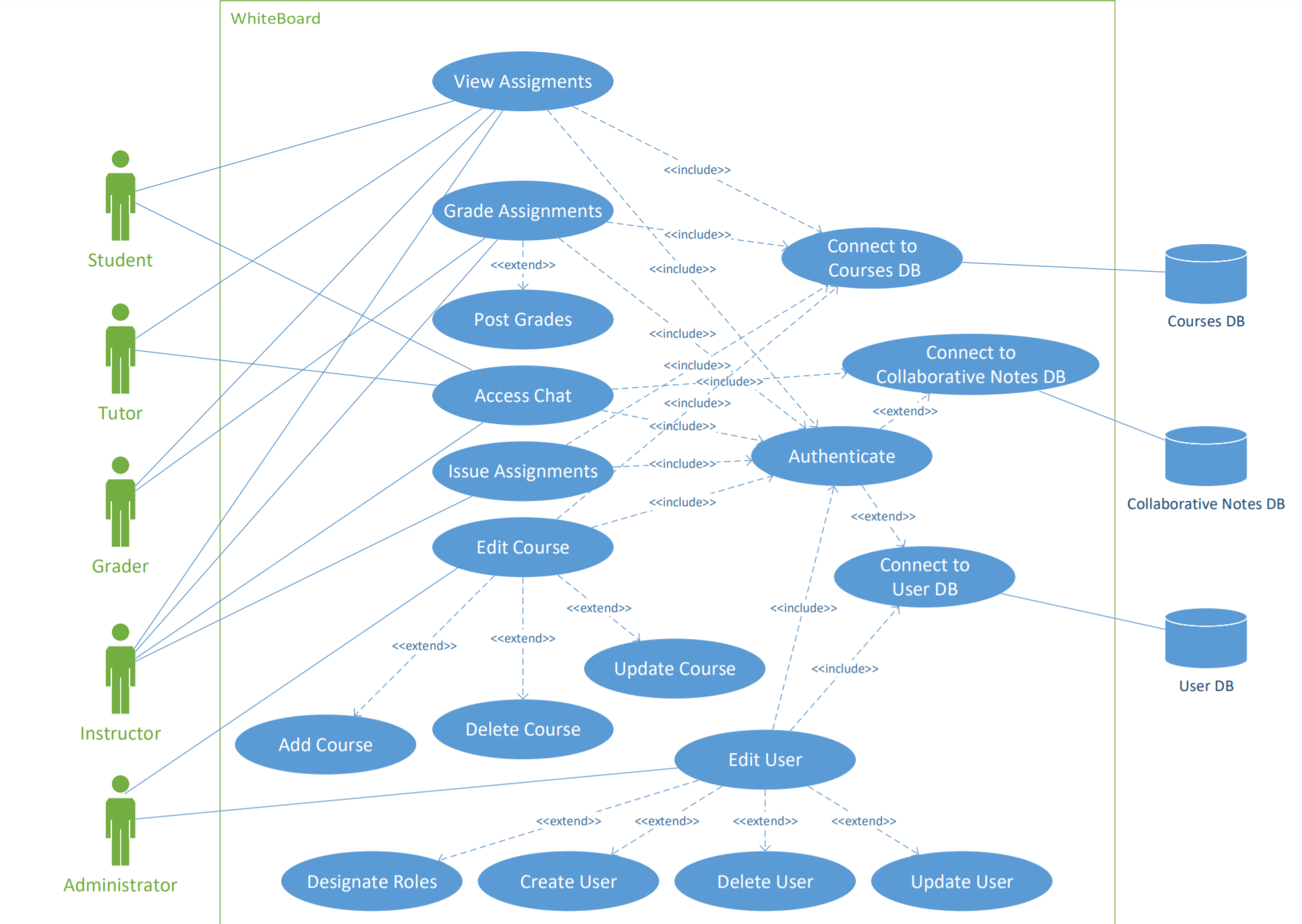
## 

## 

## 

## **4.3. Use Case Diagram**

The next figure depicts the interaction between the actors and the WhiteBoard system. A description for each actor follows.



*Figure 1: Use Case Diagram*

## **4.4. Requirements Analysis**

After identifying client requirements, we, as system designers, are in 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 that will meet end-user needs. This will ensure the attainment of customer 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 proven, and highly utilized three-tier architecture was chosen for our project. The selection of a three-tier architecture will allow our team to effectively divide the work during our implementation phase. Team member area of expertise will dictate which part of the system they’ll work with; this way our team takes full advantage of our multidisciplinary aspects.

*Figure 2: Schedule Maker Package Diagram*

## **5.2. Subsystem Decomposition**

## **5.4. Persistent Data Management**

# **6. Object Design**

This chapter will build on the architecture and design.

## **6.1. Overview**

## **6.2. Object Interaction**

*Figure 3: Sequence Diagram for Use Case Team2\_Login*

*Figure 4: Sequence Diagram for Use Case Team2\_ManageSchedule*

## **6.3. Detailed Class Design**

# **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 B – Use Cases**

## **9.3. Appendix C – User Interface Designs**

*Figure 6: Login Screen*

## **9.5. 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)

{}