*Capstone course evaluation system*

Product Design Specification

Version 3

04/16/2023

VERSION HISTORY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 1.0 | Cristina Powers  Abigail Noyes  Raad Bhuiyan  Bharath Palanisamy | *02/25/2023* | Cristina Powers  Abigail Noyes  Raad Bhuiyan  Bharath Palanisamy | *02/25/2023* | Initial Design Definition draft |
| 2.0 | Cristina Powers  Abigail Noyes  Raad Bhuiyan  Bharath Palanisamy | *03/04/2023* | Cristina Powers  Abigail Noyes  Raad Bhuiyan  Bharath Palanisamy | *03/04/2023* | Revision of Documents based on Notes. |
| 3.0 | Cristina Powers  Abigail Noyes  Raad Bhuiyan  Bharath Palanisamy | *04/16/2023* | Cristina Powers  Abigail Noyes  Raad Bhuiyan  Bharath Palanisamy | *04/16/2023* | Final draft of Design document. |

TABLE OF CONTENTS

[1 Introduction 5](#_Toc132545789)

[1.1 Purpose of The Product Design Specification Document 5](#_Toc132545790)

[2 General Overview and Design Guidelines/Approach 5](#_Toc132545791)

[2.1 Assumptions / Constraints / Standards 5](#_Toc132545792)

[2.1.1 Assumptions 5](#_Toc132545793)

[2.1.2 Constraints 5](#_Toc132545794)

[2.1.3 Standards 5](#_Toc132545795)

[3 Architecture Design 6](#_Toc132545796)

[3.1 Software Architecture 6](#_Toc132545797)

[3.2 Security Architecture 6](#_Toc132545798)

[3.3 Communication Architecture 8](#_Toc132545799)

[3.4 Performance 9](#_Toc132545800)

[4 System Design 10](#_Toc132545801)

[4.1 Use-Cases 10](#_Toc132545802)

[4.1.1 User Sign in 10](#_Toc132545803)

[4.1.2 User Sign Out 11](#_Toc132545804)

[4.1.3 User Set Password 12](#_Toc132545805)

[4.1.4 User Forgot Password 13](#_Toc132545806)

[4.1.5 Create Semester 14](#_Toc132545807)

[4.1.6 Create Section 15](#_Toc132545808)

[4.1.7 Add Professor 15](#_Toc132545809)

[4.1.8 Edit Professor 16](#_Toc132545810)

[4.1.9 Delete Professor 18](#_Toc132545811)

[4.1.10 Add Section 18](#_Toc132545812)

[4.1.11 Add GTAs 19](#_Toc132545813)

[4.1.12 Edit GTA 20](#_Toc132545814)

[4.1.13 Delete GTA 21](#_Toc132545815)

[4.1.14 Add Students 22](#_Toc132545816)

[4.1.15 Delete Student 23](#_Toc132545817)

[4.1.16 Create Groups 24](#_Toc132545818)

[4.1.17 Assign Students to a Group 24](#_Toc132545819)

[4.1.18 Edit Groups 25](#_Toc132545820)

[4.1.19 View Student 26](#_Toc132545821)

[4.1.20 Grade Student 27](#_Toc132545822)

[4.1.21 View Group 27](#_Toc132545823)

[4.1.22 Pick Group 28](#_Toc132545824)

[4.1.23 View Group 29](#_Toc132545825)

[4.1.24 Grade Weekly Reports 29](#_Toc132545826)

[4.1.25 Grade Assignments 30](#_Toc132545827)

[4.2 Use Case Diagram 32](#_Toc132545828)

[4.3 Sequence Diagram 35](#_Toc132545829)

[4.3.1 Professor Adds GTA Diagram 35](#_Toc132545830)

[4.3.2 User Set Password Diagram 36](#_Toc132545831)

[4.3.3 Admin Creates Semester Diagram 37](#_Toc132545832)

[4.3.4 GTA Grades Weekly Report Diagram 38](#_Toc132545833)

[4.3.5 Professor Creates Group Diagram 39](#_Toc132545834)

[4.4 Data Flow Diagram 40](#_Toc132545835)

[4.5 Database Design 41](#_Toc132545836)

[4.6 Class diagram 42](#_Toc132545837)

[4.7 Application Program Interfaces 42](#_Toc132545838)

[4.8 User Interface Design 43](#_Toc132545839)

[4.8.1 Sign in & Set Password 43](#_Toc132545840)

[4.8.2 Admin Interface 46](#_Toc132545841)

[4.8.3 Professor Interface 49](#_Toc132545842)

[4.8.4 GTA Interface 59](#_Toc132545843)

[Product Design Specification Approval 65](#_Toc132545844)

[Appendix A: References 66](#_Toc132545845)

[Appendix B: Key Terms 67](#_Toc132545846)

# Introduction

## Purpose of The Product Design Specification Document

The focus of the product design specification document aims to define and clarify the architecture and system design for the Capstone Course Evaluation System. This technical document is intended for individuals who seek to understand the details of the system's architecture and design, such as the client whose input and approval on features like the user interface are needed. It is also intended for the development team to create an appropriate architecture and system design that aligns with the system's objectives. It is highly technical due to containing numerous details, including designs, architectures, and diagrams. Ultimately, this document acts as a guide for the development team in creating the Capstone Course Evaluation System, ensuring that it is developed to meet its objectives effectively.

# General Overview and Design Guidelines/Approach

## Assumptions / Constraints / Standards

### Assumptions

It is assumed that during the development of the CCES web application, the administrator will possess the admin username and password. Throughout the development and implementation stages, it is imperative that Wayne State University's policies are adhered to. All Wayne State University policies shall be followed throughout the development and implementation stages of the application.

### Constraints

The development of the CCES is subject to several constraints, including budget, time, Wayne State policies, and technical limitations. The budget allocated for this project is currently zero dollars. Additionally, the project design is limited to a small development team. The team is composed of four undergraduate students with limited to no experience in the technology industry. The project is expected to be completed in 12 weeks from the start of the project. To ensure student privacy, the development team does not have access to any student or professor account data. Furthermore, the team may face limited internet access based on individual workplace circumstances.

### Standards

The CCES web application will be designed to comply with Wayne State University’s Privacy Policy and provide a user-friendly experience. The user interface is designed to be intuitive, easy to navigate, and accessible for all users. Furthermore, the CCES web application is designed to be simple, responsive, intuitive, and have clear and concise messaging. The design presents an uncluttered interface that makes using the application effortless while maintaining responsiveness throughout the web application.

# Architecture Design

## Software Architecture

The application uses the MVC (Model-View-Controller) pattern as its software architecture design. The architecture separates the different aspects of the application into distinct modules, which makes it easier to develop, test and maintain the application over time. The models act as a placeholder for all evaluation data, such as grades, students, and users. The controllers act as an intermediary between the model and the view. Controllers are responsible for user responses and requests by directing the flow of data through the web application. The views render the user interface of the CCES web application. For each user, an admin, professor, and GTA dashboard will be rendered. Each user dashboard will contain its own structures to render reports and evaluations.

Diagram

Description automatically generated

Figure 1

## Security Architecture

In order to ensure database security, access control mechanisms restrict user access to sensitive data. Password hashing fulfills password security, which prevents unauthorized access to sensitive information. Error handling and logging help to identify and address potential security breaches. Additionally, parameterized queries prevent SQL injection attacks. These measures ensure that the database is secure and protected against potential threats.

Security is first implemented at the point of user sign in. Users must be added to the application via administrator or professor to access any of the contents of the site; this ensures that unauthorized users will not be allowed into the application to manipulate or view students' grades.

User passwords will not show when the user is typing. The user's passwords will be saved in a hash format so that in the event of a security breach, hackers will not be able to understand the password, and the password will not be compromised and is protected in the database.

The implemented features in the web application that handle data, error handling, and input validation methods ensure that only accurate information is saved in the database. Figure 2 indicates that only authorized users can use the application to insert and view sensitive data from the database.

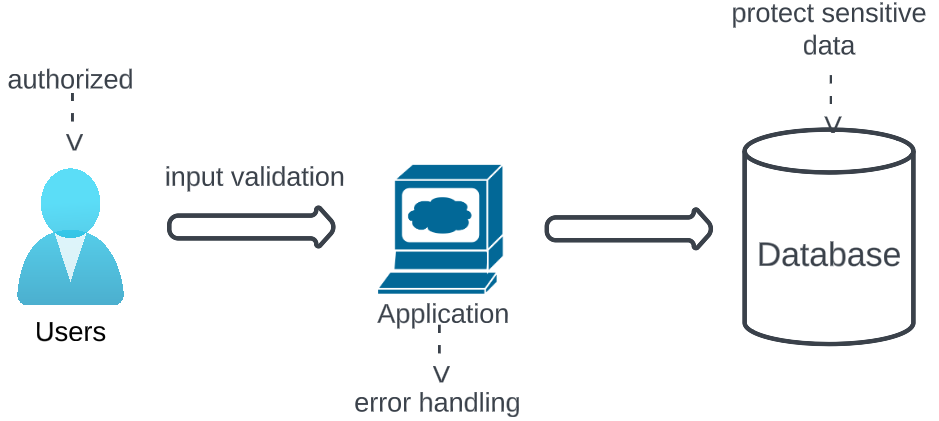


Figure 2

An email is sent to the user containing a URL that has two unique tokens in it; one to authenticate that this is the user the email was sent to and one that is going to check the database to correctly pinpoint the user when they go back to the website. Having two tokens avoid timing attacks by hackers. Figure 3 illustrates this architecture.

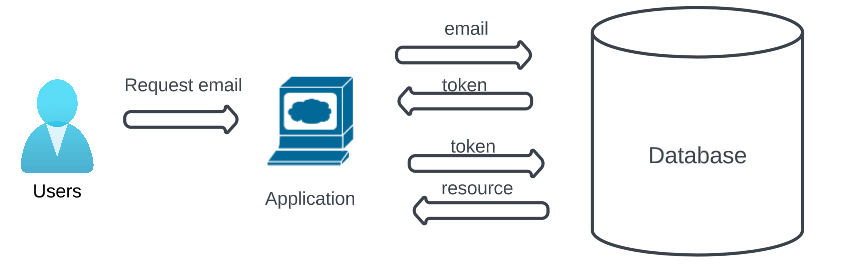


Figure 3

The application also includes a class that verifies whether the session has started. If a user attempts to access a page via a saved link, the application will redirect them to the sign-in page, prompting them to enter their credentials. The redirect ensures that only authorized users can access any page, which is critical because the grades and evaluations of the students are confidential.

In the “Dbh.class” file, a method called “query” addresses the concerns about possible SQL injection attacks. In order to prevent such attacks, it is recommended that developers avoid concatenating user input directly into SQL queries. Instead, the prepared statements in the “query” method can be used to pass parameters to a query, separate from the SQL statement. This allows the developers to write more secure code by separating the data from the code logic.

## Communication Architecture

The Capstone Course Evaluation System website utilizes a communication architecture that involves PHP scripts running on an Apache server, which interacts with the MySQL database. When a user accesses the website, the Apache server processes the PHP scripts that are responsible for handling the user's request. These scripts retrieve data from the MySQL database based on the user's request and generate the appropriate response, which is then sent back to the user's web browser. PHP scripts also utilize the GET and POST methods to send information by including input fields in HTML forms that users can fill out and submit. The PHP script then retrieves the submitted data through the $\_GET and $\_POST super global variables, respectively. SMTP (Simple Mail Transfer Protocol) can be used to send email messages from a PHP script. The script can utilize the built-in PHP mail function, which allows you to specify the sender, recipient, subject, and body of the email message. The function then sends the message to the SMTP server for delivery to the recipient's email address.

Diagram

Description automatically generated

Figure 4

Application

Description automatically generated with low confidence

Figure 5

## Performance

The performance of the Capstone Course Evaluation System considers responsiveness, scalability, security, reliability, and user experience. The CCES responsiveness maintains a quick user interaction experience, minimizing the time spent waiting for time lags or delays. The scalability of the web application handles a large volume of data without crashing or slowing down. For security, an implemented encrypted password with duo email authentication and authorization will protect the privacy of student evaluations and professor feedback. The user will experience an easy-to-use web application with a clear understanding of how to complete a student evaluation.

# System Design

## Use-Cases

### User Sign in

|  |  |
| --- | --- |
| ID | UC1 |
| Name | User Sign in |
| Actors | All Users |
| Description | When opening the application, the user will be able to sign in. They will first be prompted to enter an email and then a password. |
| Assumptions | The system assumes that the user’s email is saved in the database and the user has previously signed in once before to set up a password via email confirmation. |
| Trigger | The user clicks the “Submit” button. |
| Preconditions | The user’s email must already be in the application and the user must confirm their email address to complete a two-factor authentication process. |
| Normal Flow | 1. User enters in email. 2. User clicks “Next” button. 3. The screen reloads with in input box for the user to enter the password. 4. User enters their password. 5. User clicks “Submit” button. |
| Post Condition | The user is successfully authenticated and will redirect the user to their respective homepage based on their role of admin, professor, or GTA. |
| Alternative Flow | N/A |
| Exceptions | Error: Email is required  If the user tries to press next without entering anything, an error message will display, “Email is required.”  Error: Invalid Email  If the user enters an email that is not in a proper email format, an error message will display, “Invalid email.” The form will not submit until a valid email is entered.  Error: Email not in database  If the user enters an email that is not in our database, an error message will display, “Sorry that email is not in our database. Please seek out an admin.”  Error: Password is not valid  If the user enters the incorrect password and hits the submit button, the message “Sorry that password did not work. Try Again.” will display. The form will not submit until a valid password is entered. |

### User Sign Out

|  |  |
| --- | --- |
| ID | UC2 |
| Name | User Sign Out |
| Actors | All Users |
| Description | The user should be able to sign out of their account. The user can access the sign out button by going to the top left corner of the screen and opening the navigation. The user will be able to access this button on every screen after signing in. |
| Assumptions | The user has a valid account is signed into the application and the sign out button is clear and accessible to the user |
| Trigger | The user clicks the “Sign Out” button. |
| Preconditions | The user is currently on any page inside the application and they can locate the sign out button within the hamburger menu. |
| Normal Flow | 1. User clicks on hamburger menu to open the navigation menu. 2. User clicks the “Sign Out” button. |
| Post Condition | The sign in page will be displayed once user clicks the sign out button and access controls mechanisms will invalidate the users session. |
| Alternative Flow | The user closes the browser tab. |
| Exceptions | N/A |

### User Set Password

|  |  |
| --- | --- |
| ID | UC3 |
| Name | User Set Password |
| Actors | All Users |
| Description | The user should be able to set their password after entering their email and clicking “Next” from the sign in page. |
| Assumptions | The user’s email is in our database, the user has access to their email account, and the user is signing into the application for first time. |
| Trigger | The user enters their email and clicks “Next.” |
| Preconditions | The user has entered their email in the initial Sign in page and the application has relocated them to the Set Password page. |
| Normal Flow | 1. The user enters their email and clicks “Next.” 2. The screen reloads with the “Request Password” page. 3. The user enters their email again on the “Set Password” page. 4. An alert pops up with the message “Check your email!” 5. The application sends an email with a link to the set password page. 6. The user clicks on the URL or copy and pastes the URL into a new tab. 7. The application loads the “Set Password” page. 8. The user enters a new password and confirmation and clicks the “Set Password!” button. |
| Post Condition | The application will update the password in the database and redirect the user to the “Sign In” page, where the user can login with their newly set password. |
| Alternative Flow | N/A |
| Exceptions | Error: Email not in database  If the user enters an email that is not in our database, an error message will display, “Sorry that email is not in our database. Please seek out an admin.”  Error: Confirmation password does not match  If it is the users first time signing into the application, they will be asked to create a password and confirm it. If the user enters two different passwords, the application will display the error message “Sorry, those two passwords do not match. Please try again.” The application will keep asking for a password and a confirmation until they both match. |

### User Forgot Password

|  |  |
| --- | --- |
| ID | UC4 |
| Name | User Forgot Password |
| Actors | All Users |
| Description | The user should be able to reset their password from the sign in page. |
| Assumptions | The user’s email is in the database. |
| Trigger | The user clicks the “Forgot password?” link on the sign in page. |
| Preconditions | The user has previously created an account and has forgotten their password. |
| Normal Flow | 1. The user clicks the “Forgot password?” underneath the “Submit” button on the sign in page when prompted for a password. 2. The screen reloads with the “Request Password” page. 3. The user enters their email again on the “Set Password” page. 4. An alert pops up with the message “Check your email!” 5. The application sends an email with a link to the set password page. 6. The user clicks on the URL or copy and pastes the URL into a new tab. 7. The application loads the “Set Password” page. 8. The user enters a new password and confirmation and clicks the “Set Password!” button. |
| Post Condition | The user will receive an email under their registered email address with instructions on how to reset their password. The web application will update the password in the database and redirect the user to the “Sign In” page. |
| Alternative Flow | N/A |
| Exceptions | Error: Email not in database  If the user enters an email that is not in our database, an error message will display, “Sorry that email is not in our database. Please seek out an admin.”  Error: Confirmation password does not match  If it is the users first time signing into the application, they will be asked to create a password and confirm it. If the user enters two different passwords, the application will display the error message “Sorry, those two passwords do not match. Please try again.” The application will keep asking for a password and a confirmation until they both match. |

### Create Semester

|  |  |
| --- | --- |
| ID | UC5 |
| Name | Create Semester |
| Actors | Admin User |
| Description | The admin should be able to create a semester by going to the “Add Semester/Section” page from the navigation. |
| Assumptions | The admin is signed into the application. |
| Trigger | The admin clicks the “Add” button at the bottom of the form after inputting the information. |
| Preconditions | The user is logged in as an admin. The user is on the “Add Semester/Section” page. To add a semester, the user provides a semester date and semester name. |
| Normal Flow | 1. The admin clicks into the input box under the “Semester Name” label. 2. The admin types in a semester name in the form “Fall2020.” 3. The admin clicks into the input box next to the “Start Date” label. 4. The admin chooses a start date from the calendar or types the date in. 5. The admin clicks into the input box next to the “End Date” label. 6. The admin chooses an end date from the calendar or types the date in. 7. The admin clicks the “Add” button. |
| Post Condition | Upon successful completion of new semester, a new semester will be created and stored in the database with the name, start, and end date. The newly created data is stored in the Semester table and will now be available for viewing and use. The page will reload with the message “Semester added successfully.” |
| Alternative Flow | N/A |
| Exceptions | Error: Semester already created  The system will not allow an admin to create a semester that’s name is already taken and with the same start dates. |

### Create Section

|  |  |
| --- | --- |
| ID | UC6 |
| Name | Create Section |
| Actors | Admin user |
| Description | The admin should be able to create a section by going to the “Add Semester/Section” page from the navigation. On the page the admin will be able to create a section for a specific professor. |
| Assumptions | The admin is signed into the application. There should also be an active semester in the system. A professor has been added to the application by an admin. |
| Trigger | The admin clicks the “Add” button at the bottom of the form after inputting the information. |
| Preconditions | The admin is on the “Add Semester/Section” page. The admin must already have created an active semester and added professors in the system. To add a section, the user provides a section name, semester, and professor name. |
| Normal Flow | 1. The admin clicks into the input box under the “Section Name” label. 2. The admin types in a section name in the form “Section 101.” 3. The admin clicks the drop-down next to the “Semester” label. 4. The admin chooses a semester from the drop-down select. 5. The admin clicks the drop-down next to the “Professor” label. 6. The admin chooses a professor from the drop-down select. 7. The admin clicks the “Add” button. |
| Post Condition | The application will add the name, the semester ID, and the professors ID into the Section Table in the database. Once the data is loaded, the page will reload with the message “Section added successfully.” |
| Alternative Flow | N/A |
| Exceptions | Error: Section already created  The system will not allow an admin to create a section that’s name is already taken. |

### Add Professor

|  |  |
| --- | --- |
| ID | UC7 |
| Name | Add Professor |
| Actors | Admin user |
| Description | The admin should be able to add professors to the application by locating the "Add Professor” page from the navigation bar. On the page, the admin will be able to authorize a professor’s access to the application. |
| Assumptions | The admin is signed into the application and the professor being information added is accurate. |
| Trigger | After inputting the information, the admin clicks the “Add” button at the bottom of the form |
| Preconditions | The admin must be logged in and navigated to the “Add Professors” page, where the first name, last name and email fields are displayed on a form and available for input. |
| Normal Flow | 1. The admin clicks into the input box under the “First Name” label in the form. 2. The admin enters the first name of the professor. 3. The admin clicks into the input box under the “Last Name” label in the form. 4. The admin enters the last name of the professor. 5. The admin clicks into the input box under the “Email” label in the form. 6. The admin enters the Wayne State University email that belongs to the professor. 7. The admin clicks the “Add” button. |
| Post Condition | The first name, last name, and email have been inserted into the Users table in the database as well as the “role of the user” is set as “professor,” and the professor is marked as an active user.  The page will reload for the user with the message “Professor added successfully.” |
| Alternative Flow | N/A |
| Exceptions | Error: Professor is already in the system  The system will not allow an admin to add a professor to the application if their email is already in our database.  Error: Email is not a Wayne State University Email  The system will not allow an admin to add a professor to the application if the email entered is not in the Wayne State University access ID format followed by the “@wayne.edu.” domain. |

### Edit Professor

|  |  |
| --- | --- |
| ID | UC8 |
| Name | Edit Professor |
| Actors | Admin user |
| Description | The admin should be able to edit professors on the “Add Professors” page. |
| Assumptions | The user is signed into the application with admin credentials and there are already professors in the system. |
| Trigger | The admin clicks the “Update Data” button after making the necessary changes to the professor’s first name, last name, and/or email address. |
| Preconditions | The admin is on the “Add Professors” page and has clicked the “Edit” button to open a popup to edit the information for a specific professor. The admin has the accurate information available to update the professors information. |
| Normal Flow | 1. The admin clicks the “Edit” button next to the professor’s name. 2. A pop-up modal displays on the screen with the first name, last name, and email of that professor. 3. The admin clicks into the input box under the “First Name” label. 4. The admin makes the changes to the first name if they desire. 5. The admin clicks into the input box under the “Last Name” label. 6. The admin makes the changes to the last name if they desire. 7. The admin clicks into the input box under the “Email” label. 8. The admin makes the changes to the email if they desire. 9. The admin clicks the “Update Data” button. |
| Post Condition | If any changes are made to the first name, last name, and/or email address, the information will be updated in the Users table under that professor’s user ID.  The page will reload with the message, “Update success!” |
| Alternative Flow | N/A |
| Exceptions | Error: Blank inputs  When updating a professor, the admin must not leave any input fields blank. If any input fields are not filled out, an error message will display with the message, “All fields are required,” and will not submit. |

### Delete Professor

|  |  |
| --- | --- |
| ID | UC9 |
| Name | Delete Professor |
| Actors | Admin user |
| Description | The admin should be able to delete professors on the “Add Professors” page. |
| Assumptions | The admin is signed into the application and there are already professors in the system. |
| Trigger | The admin clicks the “Confirm Deletion” button clicking the “Delete” button. |
| Preconditions | The admin is on the “Add Professors” page and has clicked the “Delete.” |
| Normal Flow | 1. The admin clicks the “Delete” button next to the professor’s name. 2. A pop-up modal appears on the screen with the message “Do you want to delete this professor’s account access? You can add this professor again and all section data will still be available.” 3. The admin clicks the “Confirm Deletion” button at the bottom of the pop-up modal. |
| Post Condition | The application will not physically delete the user from the database but mark them as inactive in the Users table so that their history will still be able to be viewed by the admin.  The page will reload with the message, “Deletion success!” |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Add Section

|  |  |
| --- | --- |
| ID | UC10 |
| Name | Add Section |
| Actors | Professor user |
| Description | The professor should be able to add a section by going to the “Add Section” page from the navigation bar. |
| Assumptions | The professor is signed into the application. |
| Trigger | The professor clicks the “Add” button at the bottom of the form after entering the information. |
| Preconditions | The admin is on the “Add Section” page. |
| Normal Flow | 1. The professor clicks in the input box under the “Section Name” label. 2. The professor types in the name of the section in the form “Section 001.” 3. The professor clicks the “Add” button. |
| Post Condition | The application will add the section name, along with the semester ID and professor’s ID, into the Section table of the database.  The page will reload with the message, “Section added Successfully!” |
| Alternative Flow | N/A |
| Exceptions | Error: Section already in the semester  The system will not allow a professor to create a section with a name that has already been created and added to that semester. |

### Add GTAs

|  |  |
| --- | --- |
| ID | UC11 |
| Name | Add GTAs |
| Actors | Professor user |
| Description | The professor should be able to add GTAs to the application by locating the "Add GTAs” page from the navigation bar. |
| Assumptions | The professor is signed into the application. |
| Trigger | The professor clicks the “Add” button at the bottom of the form after inputting the information. |
| Preconditions | The professor is on the “Add GTAs” page. |
| Normal Flow | 1. The professor clicks into the input box under the “First Name” label in the form. 2. The professor enters the first name of the GTA. 3. The professor clicks into the input box under the “Last Name” label in the form. 4. The professor enters the last name of the GTA. 5. The professor clicks into the input box under the “Email” label in the form. 6. The professor enters the Wayne State University email that belongs to the GTA. 7. The professor clicks the “Add” button. |
| Post Condition | The first name, last name, and email are inserted into the Users table in the database as well as the “role of the user” is set as “GTA” and the GTA is marked as an active user.  The page will reload for the user with the message “GTA added successfully.” |
| Alternative Flow | N/A |
| Exceptions | Error: GTA is already in the system  The system will not allow an admin to add a professor to the application if their email is already in our database.  Error: Email is not a Wayne State University Email  The system will not allow an admin to add a professor to the application if the email entered is not in the Wayne State University access ID format followed by the “@wayne.edu.” domain. |

### Edit GTA

|  |  |
| --- | --- |
| ID | UC12 |
| Name | Edit GTA |
| Actors | Professor user |
| Description | The professor should be able to edit GTAs on the “Add GTAs” page. |
| Assumptions | The professor is signed into the application and there are already GTAs in the system. |
| Trigger | The professor clicks the “Update Data” button after making the necessary changes to the professor’s first name, last name, and/or email address. |
| Preconditions | The admin is on the “Add GTAs” page and has clicked the “Edit” button to open a popup to edit the information. |
| Normal Flow | 1. The professor clicks the “Edit” button next to the GTA’s name. 2. A pop-up modal displays on the screen with the first name, last name, and email of that GTA. 3. The professor clicks into the input box under the “First Name” label. 4. The professor makes the changes to the first name if they desire. 5. The professor clicks into the input box under the “Last Name” label. 6. The professor makes the changes to the last name if they desire. 7. The professor clicks into the input box under the “Email” label. 8. The professor makes the changes to the email if they desire. 9. The professor clicks the “Update Data” button. |
| Post Condition | If any changes are made to the first name, last name, and/or email address, the information will be updated in the Users table under that GTA’s user ID.  The page will reload with the message, “Update success!” |
| Alternative Flow | N/A |
| Exceptions | Error: Blank inputs  When updating a GTA, the professor must not leave any input fields blank. If any input fields are not filled out, an error message will display with the message, “All fields are required,” and will not submit. |

### Delete GTA

|  |  |
| --- | --- |
| ID | UC13 |
| Name | Delete GTA |
| Actors | Professor user |
| Description | The professor should be able to delete GTAs on the “Add GTAs” page. |
| Assumptions | The professor is signed into the application, and there are already GTAs in the system. |
| Trigger | The professor clicks the “Confirm Deletion” button after clicking the “Delete” button. |
| Preconditions | The professor is on the “Add GTAs” page and has clicked the “Delete.” |
| Normal Flow | 1. The professor clicks the “Delete” button next to the GTA’s name. 2. A pop-up modal appears on the screen with the message “Do you want to delete this GTA’s account access? You can add this GTA again and all section data will still be available.” 3. The professor clicks the “Confirm Deletion” button at the bottom of the pop-up modal. |
| Post Condition | The application will not physically delete the user from the database but mark them as inactive in the Users table so that their history will still be able to be viewed by the admin.  The page will reload with the message, “Deletion success!” |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Add Students

|  |  |
| --- | --- |
| ID | UC14 |
| Name | Add Students |
| Actors | Professor user |
| Description | The professor should be able to upload students into a specific section from the “Add Students” page. |
| Assumptions | The professor is signed into the application. |
| Trigger | The professor clicks the “Upload” button. |
| Preconditions | 1. The professor must have a file in (.CSV, .ODS, .XLSX, .XLS) form, with the student’s access ID in one column and the student’s name in the other. 2. The professor must already have created a section to add the students to. 3. The professor must be on the “Add Students” Page. |
| Normal Flow | 1. The professor clicks the “Choose File” button. 2. The professor selects the (.CSV, .ODS, .XLSX, .XLS) file they want and selects “Open.” 3. The professor chooses which section from the drop-down. 4. The professor clicks the “Upload” button. |
| Post Condition | The student’s name and access ID will be added to the Section table in the database, as well as the section ID.  The page will reload and display the message “Students added successfully.” |
| Alternative Flow | N/A |
| Exceptions | Error: File format  The system will check if the file uploaded is of a proper file type (.CSV, .ODS, .XLSX, .XLS). It will display the error message “Not an XLSX, XLS, CSV, or ODS file” and not submit the form.  Error: Access ID proper format  The system will check if the access IDs are in the correct Wayne State University format (two letters, followed by four numbers for a total of 6 characters). The error message will display “Non-Valid Access ID added. Please Reupload.”  Error: Duplicate students  The system will also check if access IDs are already in the database to stop duplicates. The system will add all the students except for the one that is a duplicate and give the error “[access ID] is already entered as a student. Cannot add existing students to the section.” |

### Delete Student

|  |  |
| --- | --- |
| ID | UC15 |
| Name | Delete Student |
| Actors | Professor user |
| Description | The professor should be able to delete a student by going to the “Add Students” page from the navigation bar. |
| Assumptions | The professor is signed into the application, and there are already students in the system. |
| Trigger | The professor clicks the “Confirm Deletion” button after clicking the “Delete” button. |
| Preconditions | The professor is on the “Add Students” page and has clicked the “Delete.” |
| Normal Flow | 1. The professor clicks the “Delete” button next to the student’s name. 2. A pop-up modal appears on the screen with the message “Do you want to delete this student from the section?” 3. The professor clicks the “Confirm Deletion” button at the bottom of the pop-up modal. |
| Post Condition | The application will not physically delete the user from the database but mark them as inactive in the “Students” table so that their history will still be able to be viewed by the admin.  The page will reload with the message, “Deletion success!” |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Create Groups

|  |  |
| --- | --- |
| ID | UC16 |
| Name | Create Groups |
| Actors | Professor user |
| Description | The professor should be able to create groups by going to the “Create Group” page. |
| Assumptions | The professor is signed into the application. |
| Trigger | The professor clicks the “Save Group(s)” button at the bottom of the form. |
| Preconditions | The professor is on the “Create Groups” page. The professor must type in a group name in the input box. If the professor wants to add multiple groups at one time, they can click the “Add More” button. |
| ­­­Normal Flow | 1. The professor clicks into the input box under the “Name” label. 2. The professor types in a group name. 3. The professor clicks the “Add More” button. 4. The professor clicks into the second input box under the “Name” label. 5. The professor types in a group name. 6. The professor repeats steps 3-5 as many times as he would like depending on the number of groups he would like to add. 7. The professor clicks the “Save Group(s)” button. |
| Post Condition | The group name will be added into the Group table in the database, as well as the corresponding section ID.  The page will reload with the message “Group added successfully.” |
| Alternative Flow | N/A |
| Exceptions | Error: Group already created  The system will not allow a professor to create a group that’s name is already taken. |

### Assign Students to a Group

|  |  |
| --- | --- |
| ID | UC17 |
| Name | Assign Students to a Group |
| Actors | Professor user |
| Description | The professor should be able to assign students to a group by going to the “Create Group” page from the navigation bar. |
| Assumptions | The professor has already added students to a section. |
| Trigger | The professor clicks the “Assign students” button after selecting the students they would like. |
| Preconditions | The professor is on the “Create Groups” page. |
| Normal Flow | 1. The professor clicks on the drop-down menu next to the “Group” label. 2. The professor selects the desired group. 3. The professor selects as many students as they desire in the students. 4. The professor clicks the “Assign students” button. |
| Post Condition | The student table is updated so that those students have a group ID with their corresponding group in the group table. |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Edit Groups

|  |  |
| --- | --- |
| ID | UC18 |
| Name | Edit Groups |
| Actors | Professor user |
| Description | The professor should be able to edit a group’s information by going to the “Edit Groups” button located on the navigation bar. |
| Assumptions | The professor has already created groups that are in the database and the GTAs have chosen their groups. |
| Trigger | The professor clicks the “Save” button after they make the necessary changes to the selected group. |
| Preconditions | The professor presses the edit button for the specific group row they would like to edit. Then modify the group name, members, and/or the GTA input box for the group. |
| Normal Flow | 1. The professor picks a group row listed on the page. 2. The professor presses the edit button for the group. 3. A pop-up form is opened for the professor, allowing for information in the group to be modified. 4. The professor will edit the specific input box that they would like to edit. 5. The professor clicks the “Save Changes” button. |
| Post Condition | The Group and Students table is updated so that those students have the relevant group ID that they are assigned as well as the Group table updates the group name and GTA ID assigned to the group in the table based on what was changed. |
| Alternative Flow | N/A |
| Exceptions | A professor inputs invalid or missing information, such as leaving the first name or last name fields blank or inputting an invalid email address that is not associate with Wayne state university, a validation exception will occur. |

### View Student

|  |  |
| --- | --- |
| ID | UC19 |
| Name | View Student |
| Actors | Professor user |
| Description | The professor should be able to view the student’s grades on the “Student dashboard” page. |
| Assumptions | The professor is signed into the application. The student information is already added by the professor and the student evaluations are added by the GTA. |
| Trigger | The professor will click on the student’s name on the professor’s “Home” page. |
| Preconditions | The professor has uploaded students and assigned them to a group ensuring the student information is correctly stored. The professor is on their home page. |
| Normal Flow | 1. The professor clicks the group name.   The professor clicks the student’s name. |
| Post Condition | The professor will be relocated to the student dashboard. The student that they selected information will be retrieved from the database. The professor will be able to see all the grades from the weekly reports, assignments, presentations, etc., that the GTA and himself have graded. |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Grade Student

|  |  |
| --- | --- |
| ID | UC20 |
| Name | Grade Student |
| Actors | Professor user |
| Description | The professor should be able to grade a student on their “Student dashboard” page. |
| Assumptions | The professor is signed into the application. |
| Trigger | The professor clicks the “Add” button after filling out the information in the “Add Grade” form. |
| Preconditions | The professor has clicked on a student’s name and is on their dashboard. |
| Normal Flow | 1. The professor clicks on the “Add Grade.” 2. The professor picks which assignment. 3. The professor gives the assignment a grade from the drop-down. 4. The professor will type a written evaluation into the form. 5. The professor will click the “Add.” button. |
| Post Condition | The information will be added to the “Grades” table in the database. It will also mark in the database that it was given by the professor. The written evaluation by the professor will be saved in a separate, connecting Grade Evaluation table.  The page will reload with the message “Grade added.” |
| Alternative Flow | N/A |
| Exceptions | Error: All inputs are required  If the professor tries to submit the form without giving a student a grade and/or evaluation, the error message “All fields in the form required” will display and will not submit until all fields have been inputted. |

### View Group

|  |  |
| --- | --- |
| ID | UC21 |
| Name | View Group |
| Actors | Professor user |
| Description | The professor should be able to view the group’s grades on the “Group dashboard” page. |
| Assumptions | The professor is signed into the application. |
| Trigger | The professor will click the “View Group” button on the student’s page. |
| Preconditions | The professor has clicked on a student’s name and is on their dashboard. |
| Normal Flow | 1. The professor clicks the “View Group” button. |
| Post Condition | The professor will be relocated to the group dashboard. The students in that group will be retrieved from the database. The professor will be able to see all the grades from the and click on the grade to see the corresponding evaluation with that grade. |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Pick Group

|  |  |
| --- | --- |
| ID | UC22 |
| Name | Pick Group |
| Actors | GTA user |
| Description | The GTA should be able to pick groups from the GTA “Home” page. |
| Assumptions | The GTA is signed into the application. |
| Trigger | The GTA clicks the “Add” button on the GTA “Home” page after selecting a group from the drop-down. |
| Preconditions | The GTA is on their “Home” page, and the professor has added at least one group to the section the GTA is assigned to. |
| Normal Flow | 1. The GTA clicks on the drop-down to open the menu. 2. The GTA selects a group name from the drop-down. 3. The GTA clicks the “Select” button. |
| Post Condition | The GTA’s ID, the group’s ID, and the section ID are all added to the GTA Assignment table in the database.  That group is now  The page will reload, and the database will retrieve the group and display it under the “Your Groups” title. |
| Alternative Flow | N/A |
| Exceptions | N/A |

### View Group

|  |  |
| --- | --- |
| ID | UC23 |
| Name | View Group |
| Actors | GTA user |
| Description | The GTA should be able to view any of the groups that they have been assigned to. |
| Assumptions | The GTA is signed into the application. |
| Trigger | The GTA clicks on a group name. |
| Preconditions | The GTA is on their “Home” page and has been assigned a group and |
| Normal Flow | 1. GTA clicks on the group name |
| Post Condition | The GTA will be relocated to the group’s dashboard. The database will automatically retrieve the students’ names that are in those groups. When the GTA clicks on the student’s name, the database will pull the grades from the database and display that student’s information on the screen. |
| Alternative Flow | N/A |
| Exceptions | N/A |

### Grade Weekly Reports

|  |  |
| --- | --- |
| ID | UC24 |
| Name | Grade Weekly Reports |
| Actors | GTA user |
| Description | The GTA can grade the weekly reports by going to the “Weekly Reports” page from the navigation bar. |
| Assumptions | The GTA is signed into the application and has groups assigned to them. |
| Trigger | The GTA clicks the “Submit” button after filling out the weekly report form. |
| Preconditions | The GTA is on the “Weekly Reports” page. |
| Normal Flow | 1. The GTA picks a group from the drop-down. 2. The GTA clicks the “Select” button. 3. The page populates with the students’ names and all of their weekly reports. 4. The week number and “Incomplete” buttons will display under the student’s name. 5. The GTA will click on “Incomplete” button for the popup form to pull up. 6. The GTA will select the “Submission” of the weekly report via a radio button. 7. The GTA will select the “Status” of the weekly report via a radio button. 8. The GTA will type their evaluation of the weekly report. 9. The GTA will click the “Submit” button. |
| Post Condition | The week, date submitted, the submission of the weekly report, the status of the weekly report, student ID, and group ID will be saved into the Weekly Reports table.  The written evaluation by the GTA will be saved in a separate, connecting Weekly Reports Evaluation table.  The page will reload, and the button will now say “Complete” for the weekly report that was just added. |
| Alternative Flow | N/A |
| Exceptions | Error: All inputs are required  If the GTA tries to fill out the evaluation by leaving any of the values blank, an error message will display, “All fields in the form are required,” and will not submit until all fields have been inputted. |

### Grade Assignments

|  |  |
| --- | --- |
| ID | UC25 |
| Name | Grade Assignments |
| Actors | GTA user |
| Description | The GTA should be able to grade any of the assignments by clicking on the assignment title in the navigation bar. |
| Assumptions | The GTA is signed into the application, and the GTA has groups assigned to them. |
| Trigger | The GTA clicks the “Submit” button after filling out the contents of the form. |
| Preconditions | The GTA is on the assignment they are trying to grade page. |
| Normal Flow | 1. The GTA selects a group from the drop-down. 2. The GTA selects the student from the drop-down. 3. The GTA selects the grade from the drop-down. 4. The GTA types in their written evaluation. 5. The GTA clicks the “Submit” button. |
| Post Condition | The assignment name, date submitted, group, student, and grade will save in the Grades table in the database. It will also mark the GTA as the person who gave the student that grade.  The written evaluation by the GTA will be saved in a separate, connecting Grade Evaluation table.  The page will reload with the message “Grade added for student successfully. |
| Alternative Flow | N/A |
| Exceptions | Error: All inputs are required  If the GTA tries to fill out the evaluation by leaving any of the values blank, an error message will display, “All fields in the form are required,” and will not submit until all fields have been inputted. |

## Use Case Diagram

Diagram

Description automatically generated

Figure 6: Use Case Diagram: Admin

Diagram

Description automatically generated

Figure 7: Use Case Diagram: Professor

Diagram

Description automatically generated

Figure 8: Use Case Diagram: GTA

## Sequence Diagram

### Professor Adds GTA Diagram

Figure 6 below displays the sequence diagram for the professor adding a GTA to the application. The sequence will take what the professor has submitted and if the inputs are valid will create a new GTA and assign the GTA to a specific section.

Diagram

Description automatically generated

Figure 9: Professor Adds GTA Diagram

### User Set Password Diagram

Figure 7 shows the sequence diagram for when a user is signing into the application for the first time and needs to set their password. The sequence starts after the user has entered in an email that is in the database and the application has redirected the user to the password request page.

Diagram

Description automatically generated

Figure 10: User Set Password Diagram

### Admin Creates Semester Diagram

Figure 8 shows the sequence diagram for when an admin creates a semester. The form will check to make sure there is an input. The semester data will be inserted into the database. A reply with a success message with display on the screen when the contents have successfully been saved into the database.

Diagram

Description automatically generated

Figure 11: Admin Creates a Semester Diagram

### GTA Grades Weekly Report Diagram

Figure 9 displays a sequence diagram for the GTA entering a weekly report for a student. The status of each weekly report will display on the “Weekly Reports” page so that the GTA can see the reports they have filled out and those they still have left to do.

Diagram

Description automatically generated

Figure 12: GTA Grades Weekly Report Diagram

### Professor Creates Group Diagram

Figure 10 shows the sequence diagram when the professor creates a group with the already uploaded students (ID: UC14) in a chosen section. The groups will be displayed on the “Add Groups” page and a success message will display when the group is successfully added into the database.

Diagram

Description automatically generated

Figure 13: Professor Create Group Diagram

## Data Flow Diagram

Diagram

Description automatically generated

Figure 14

Diagram

Description automatically generated

Figure 15

## Database Design

The MySQL database will keep all the data for this application. MySQL is a relational database management system that stores data in tables, where each table has columns and rows that represent the fields and records, respectively. The Figure 13 below shows the ER diagram for the database. Each box is a table with columns that make up the attributes or fields in that table. For instance, the "students" table will store all the students that have been added to our website along with all the relevant information about them. Each student will have their name and access IDs recorded, as well as information on what class they're in. The relationships between tables are established through foreign keys, which refer to the primary keys of other tables. This allows for efficient and organized storage and retrieval of data.

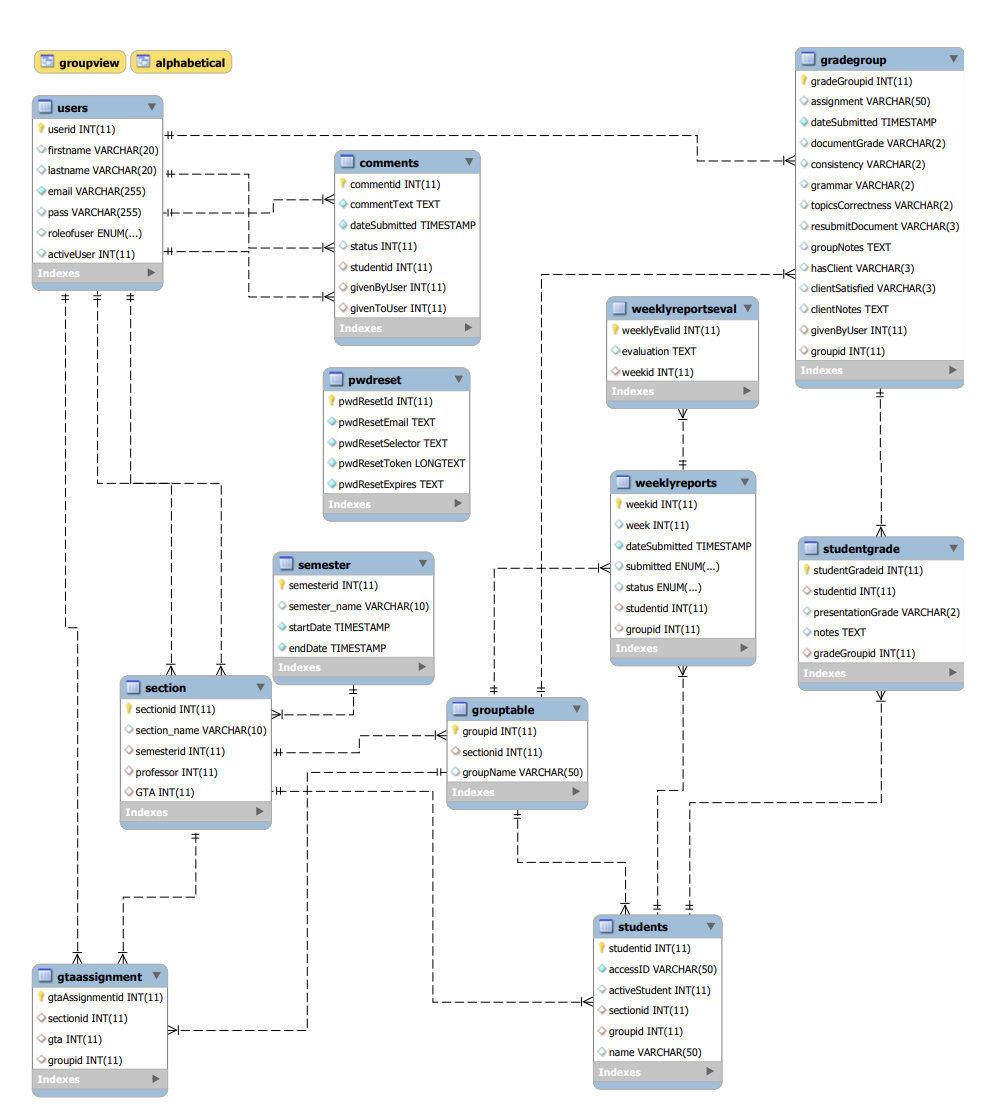
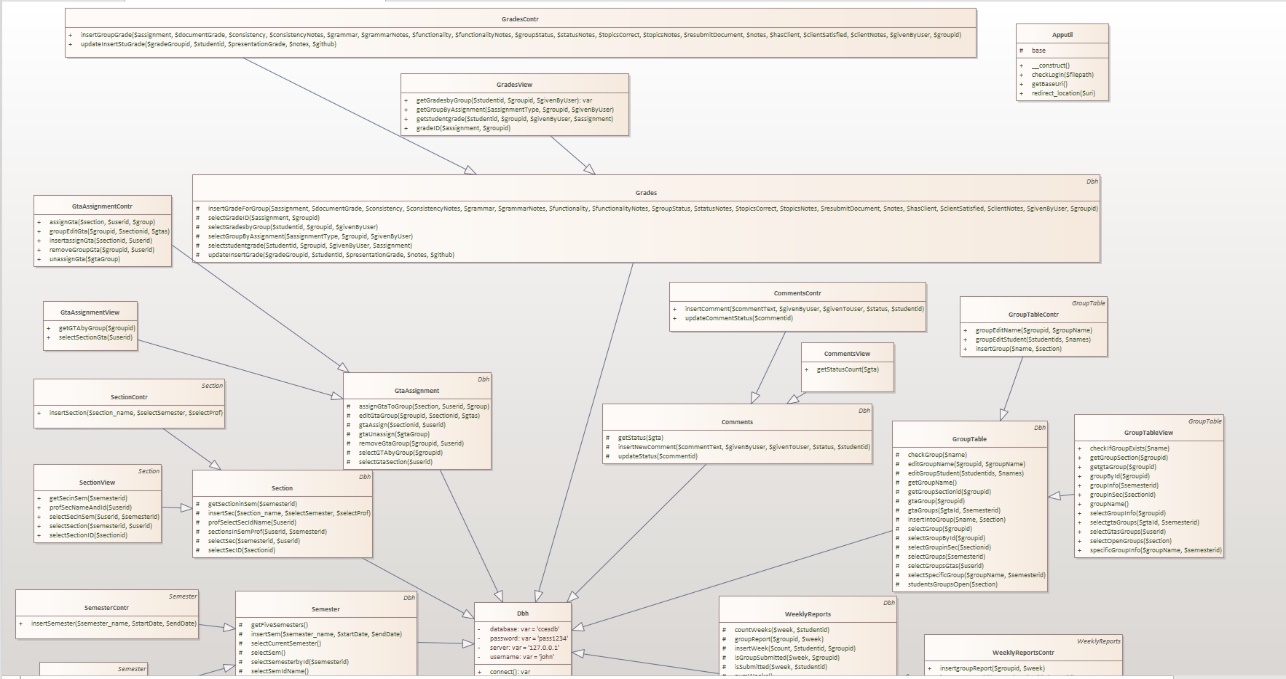


Figure 16: Database Design Diagram

## Class diagram



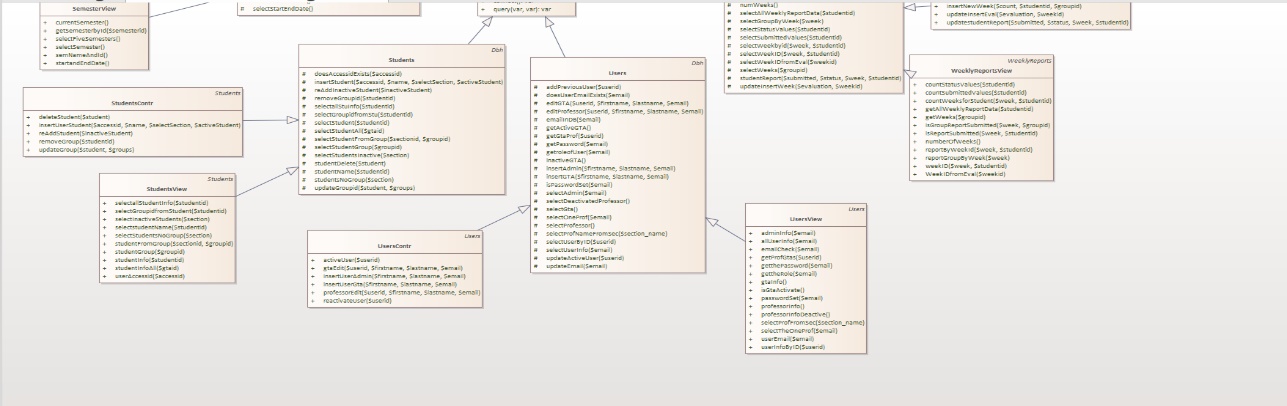


Figure 17: Class Diagram

## Application Program Interfaces

The Capstone Course Evaluation System currently does not use any APIs in the application. The structure of the software, however, is built so that functionalities can be taken out and added. The application could eventually utilize the Canvas API for the retrieval of users and students in the database.

## User Interface Design

Color is one of the most important aspects of the web application, utilizing the right colors and amount will make for a well-balanced interface design. The application uses the 60-30-10 Rule to create a simple, well-balanced color palette. The color palette for the application is given in Figure X below. The “Alice Blue” color makes up 60% of the website by being the background color and text when appropriate. The secondary color is “Dark Cyan,” used in the background of forms and the navigation bar. The accent color that makes up 10% of our website is the “Celeste” color used for buttons and other eye-catching necessities for the user. “Tea Rose (Red)” is also used, but less than 10%, to catch the user's attention with error messages.

Treemap chart

Description automatically generated

Figure 18: The colors used in the application.

### Sign in & Set Password

Graphical user interface, application

Description automatically generated

Figure 19

Graphical user interface, application

Description automatically generated

Figure 20

Graphical user interface, text, application

Description automatically generated

Figure 21

Graphical user interface, text, application, email

Description automatically generated

Figure 22

### Admin Interface

=Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 23

Graphical user interface, text, application, email

Description automatically generated

Figure

Graphical user interface, application

Description automatically generated

Figure 25

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 26

Graphical user interface, application, Word

Description automatically generated

Figure 27

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure

Graphical user interface, application

Description automatically generated

Figure 29: Navigation Bar for Admin

### Professor Interface

Graphical user interface, website

Description automatically generated

Figure 30: Before group button is clicked.

Graphical user interface, application, website

Description automatically generated

Figure 31: After group button is clicked.

Graphical user interface, text, application, chat or text message, email

Description automatically generated

Figure : Student page after clicking on a name button from Figure 31

Graphical user interface, text, application

Description automatically generated

Figure 33: Flag Button

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 34: After Clicking Flag Button

Graphical user interface, application

Description automatically generated

Figure 35

Graphical user interface, application, Word

Description automatically generated

Figure 36

Graphical user interface, text, application

Description automatically generated

Figure 37: Edit GTA Pop-Up

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Figure 38

Graphical user interface

Description automatically generated with medium confidence

Figure 39

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Figure 40

Graphical user interface, application

Description automatically generated

Figure

Graphical user interface, application

Description automatically generated

Figure

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Figure

Graphical user interface, text, application, email

Description automatically generated

Figure

Graphical user interface, text, application

Description automatically generated

Figure 47

### GTA Interface

Graphical user interface, website

Description automatically generated

Figure 48

Graphical user interface, text, application, email

Description automatically generated

Figure 49

Graphical user interface, text

Description automatically generated

Figure 50

Graphical user interface, application

Description automatically generated

Figure 51

Graphical user interface, application

Description automatically generated

Figure

A picture containing diagram

Description automatically generated

Figure

Graphical user interface

Description automatically generated

Figure 54

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure

Graphical user interface, application

Description automatically generated

Figure

Graphical user interface, application

Description automatically generated

Figure 57

# Approval

The undersigned acknowledge they have reviewed the Capstone Course Evaluation System Product Design Specification document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | CP | Date: | 2/25/2023 |
| Print Name: | Cristina Powers |  |  |
| Role: | Project Lead |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | RB | Date: | 2/25/2023 |
| Print Name: | Raad Bhuiyan |  |  |
| Role: | Backend |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | BP | Date: | 2/25/2023 |
| Print Name: | Bharath Palanisamy |  |  |
| Role: | Documentation lead/Front end |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | AN | Date: | 2/26/2023 |
| Print Name: | Abigail Noyes |  |  |
| Role: | Backend/QA/Documentation |  |  |

Appendix A: References

The following table summarizes the documents referenced in this document.

|  |  |  |
| --- | --- | --- |
| **Document Name and Version** | **Description** | **Location** |
| *How the 60-30-10 rule saved the day.* | UX Designer Ayobami Adelugba explains the 60–30–10 rule and how one can use proper color application to make a project more visual appealing | https://uxdesign.cc/how-the-60-30-10-rule-saved-the-day-934e1ee3fdd8 |

Appendix B: Key Terms

The following table provides definitions for terms relevant to this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| CCES | The abbreviation for the application name, Capstone Course Evaluation System. | |
| User | An individual who uses the Capstone Course Evaluation System | |
| Role | The type of user that can sign into the Capstone Course Evaluation System which are admin, professor, and GTA. | |
| Admin | A user of Capstone Course Evaluation System who is granted all access. | |
| GTA | This abbreviation for Graduate Teaching Assistant who use the Capstone Course Evaluation System | |
| Professor | The teacher assigned to a capstone course who uses the Capstone Course Evaluation System | |
| Semester | The term in which the specific class is taking place, ex: Fall 2023, Winter 2023, etc. | |
| Section | The specific class that the professor is going to teach for the semester. | |