# Interface Designs

Sitemap / Hierarchy Diagram

Home Page

Login Page

About Us

Student Signed In

Dashboard

Tasks

Profile

Teacher Signed In

Dashboard

Task Setter

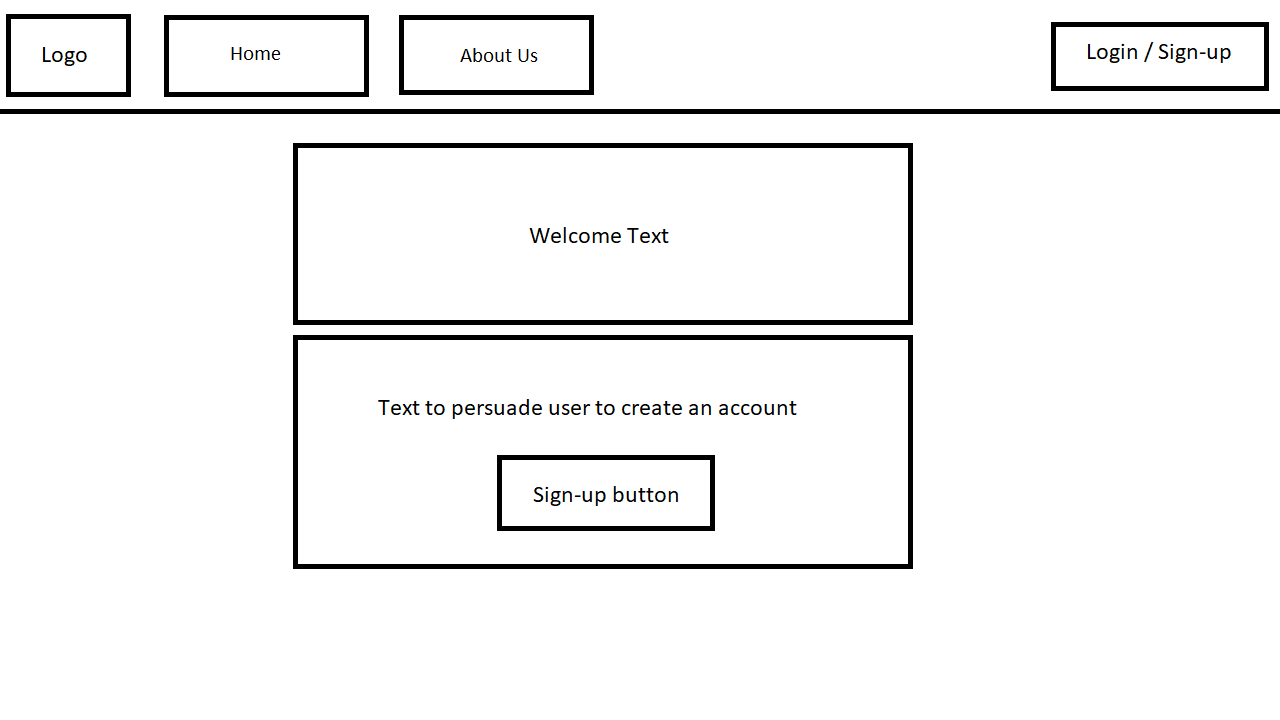
Manage Students

Profile

Feedback

Points / Level

## Not Logged in home page



Page Specification:

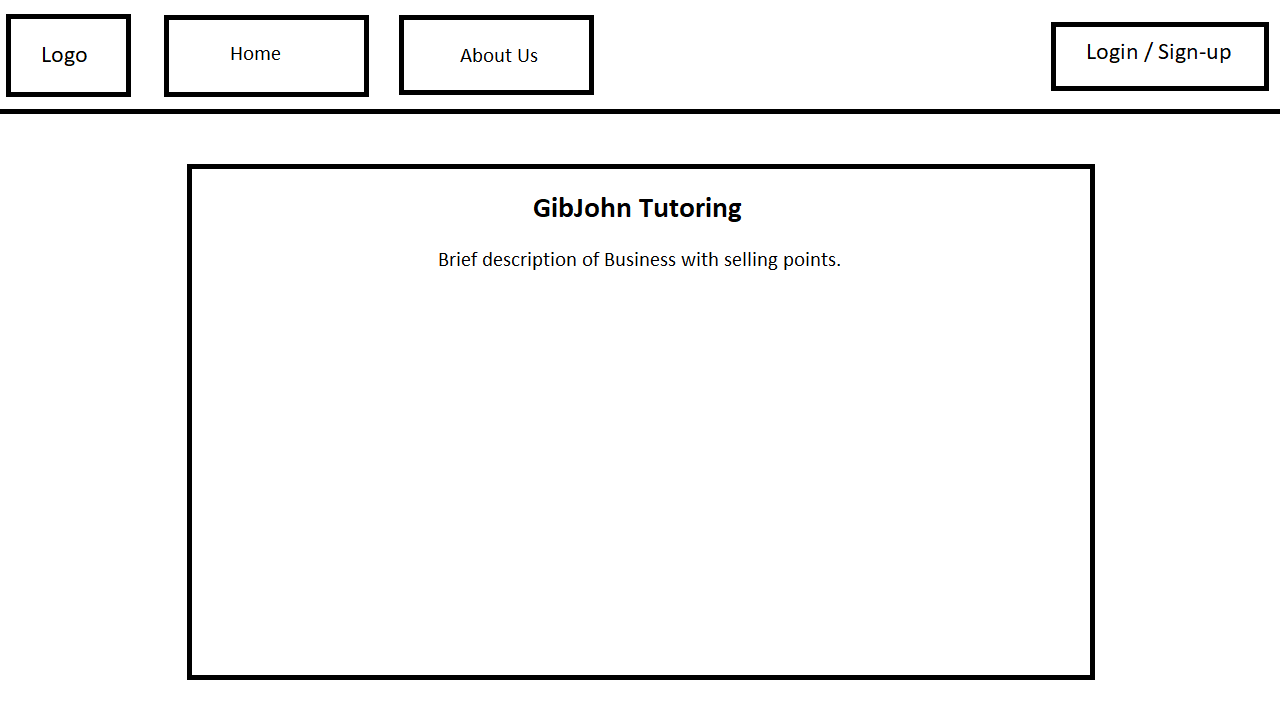
User Interface:

* Smooth transitions
* Use readable and matching colours
* Different boxes of information

Elements:

* Navbar with links to different pages and profile information
* Welcome text
* Sign-up button

## About Page



Page Specification:

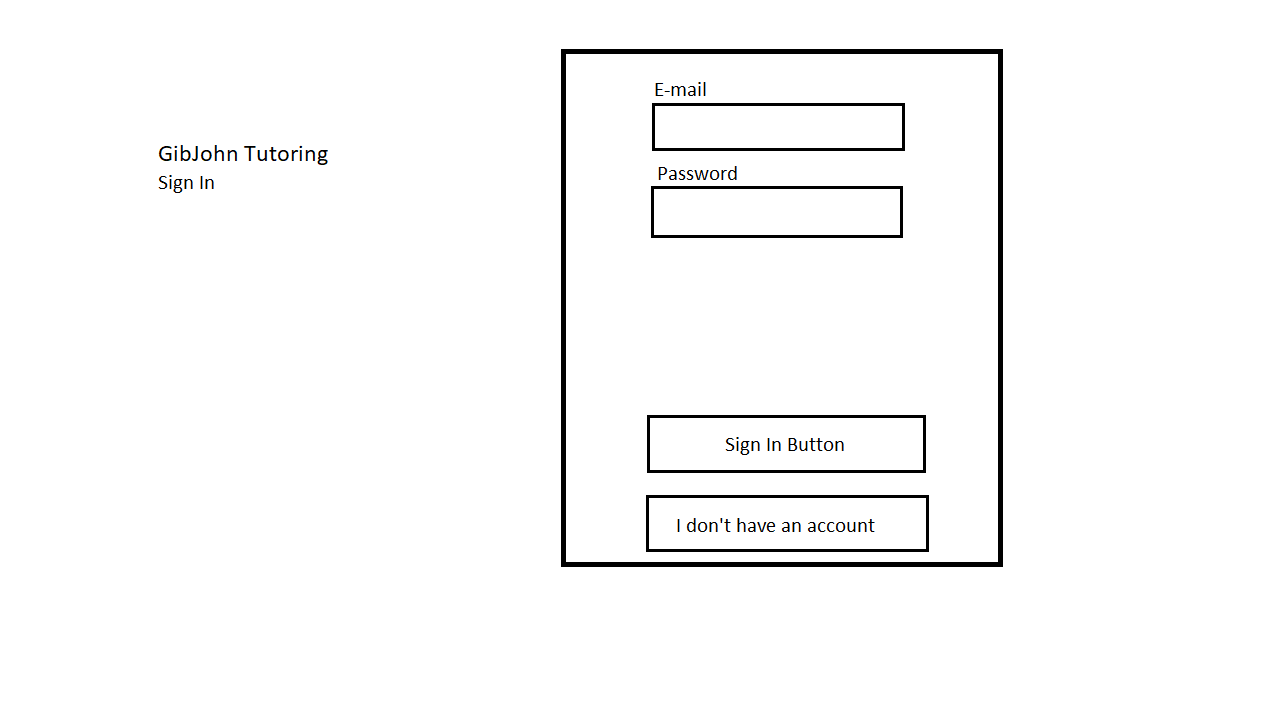
User Interface:

* Smooth transitions
* Use readable and matching colours
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Description of business

## Login Page



Page Specification:

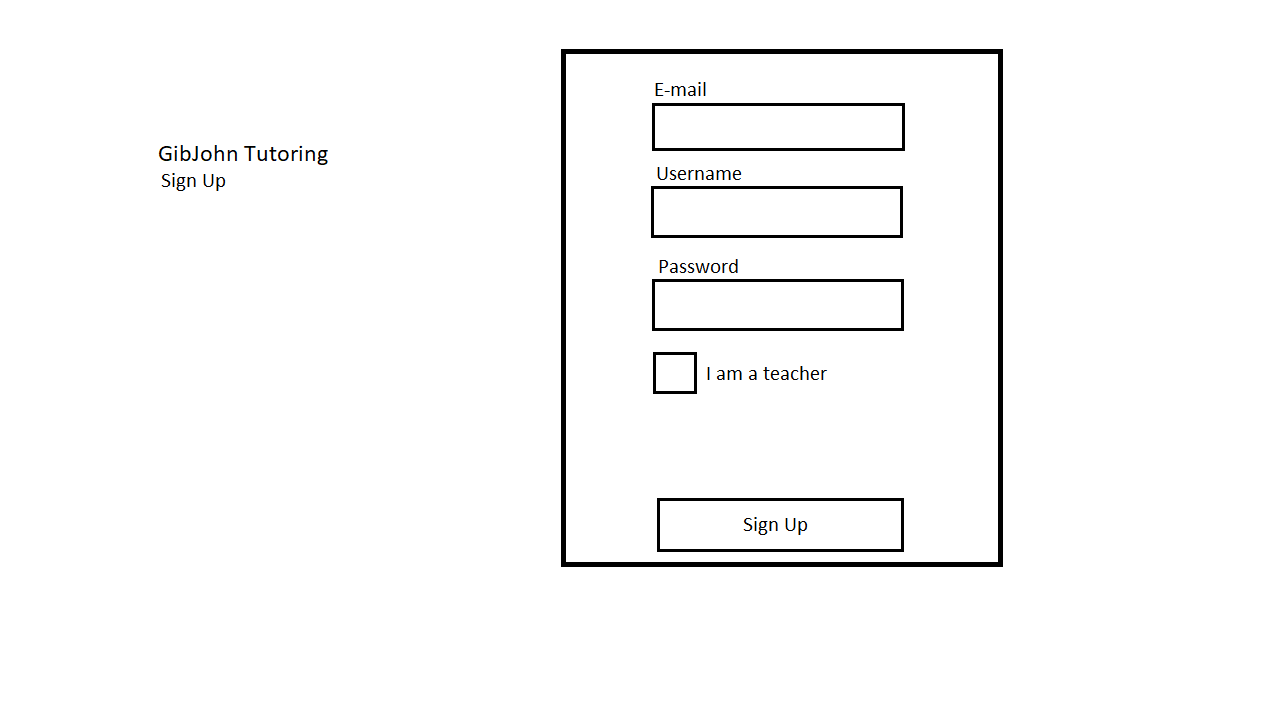
User Interface:

* Smooth transitions
* Use readable and matching colours
* Tooltips using bootstrap to make the page more accessible

Elements:

* E-mail text box
* Password text box
* Remember me checkbox
* Sign in button
* Sign up button

## Sign Up Page



Page Specification:

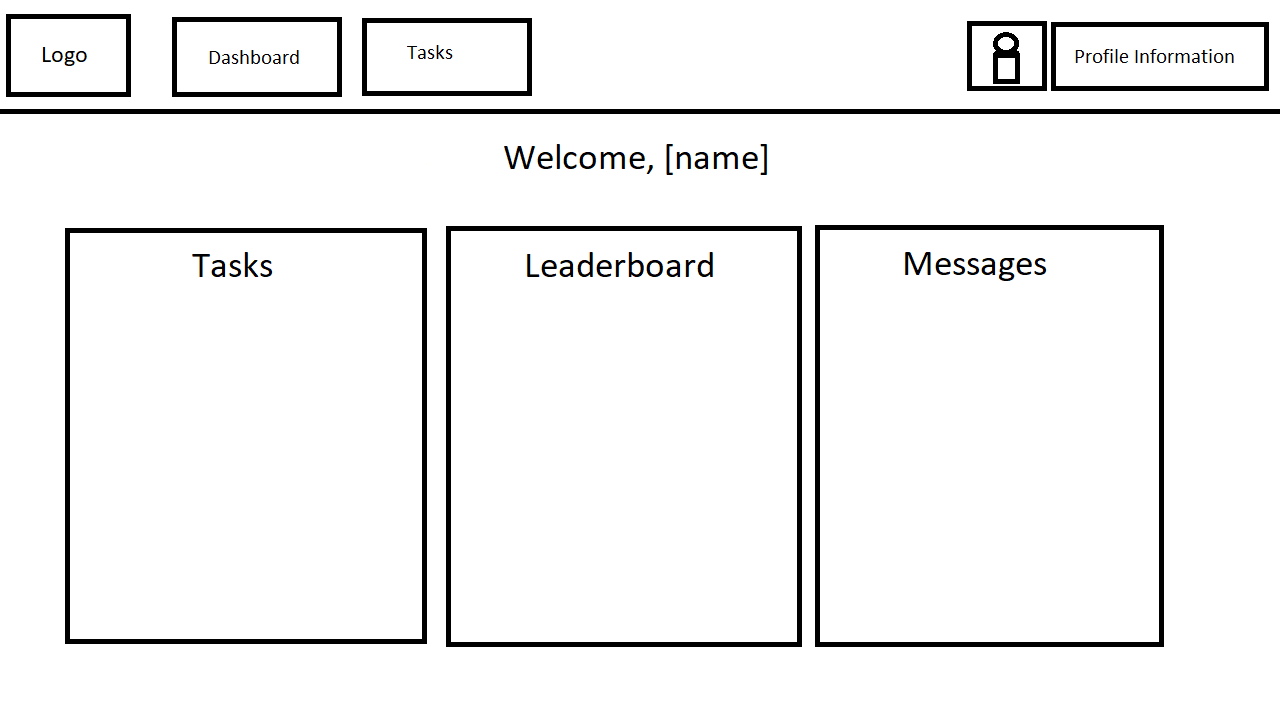
User Interface:

* Smooth transitions
* Use readable and matching colours
* Tooltips using bootstrap to make the page more accessible

Elements:

* E-mail text box
* Full Name
* Password text box
* Teacher checkbox
* Sign up button

## Student Dashboard



Page Specification:

User Interface:

* Smooth transitions
* Use readable and matching colours
* Different boxes of information
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Box to display set tasks
* Box to display position on leader board
* Box to display message inbox
* Customised welcome text
* Modal pop-ups when items are clicked

## Student Tasks Page

A picture containing diagram

Description automatically generated

Page Specification:

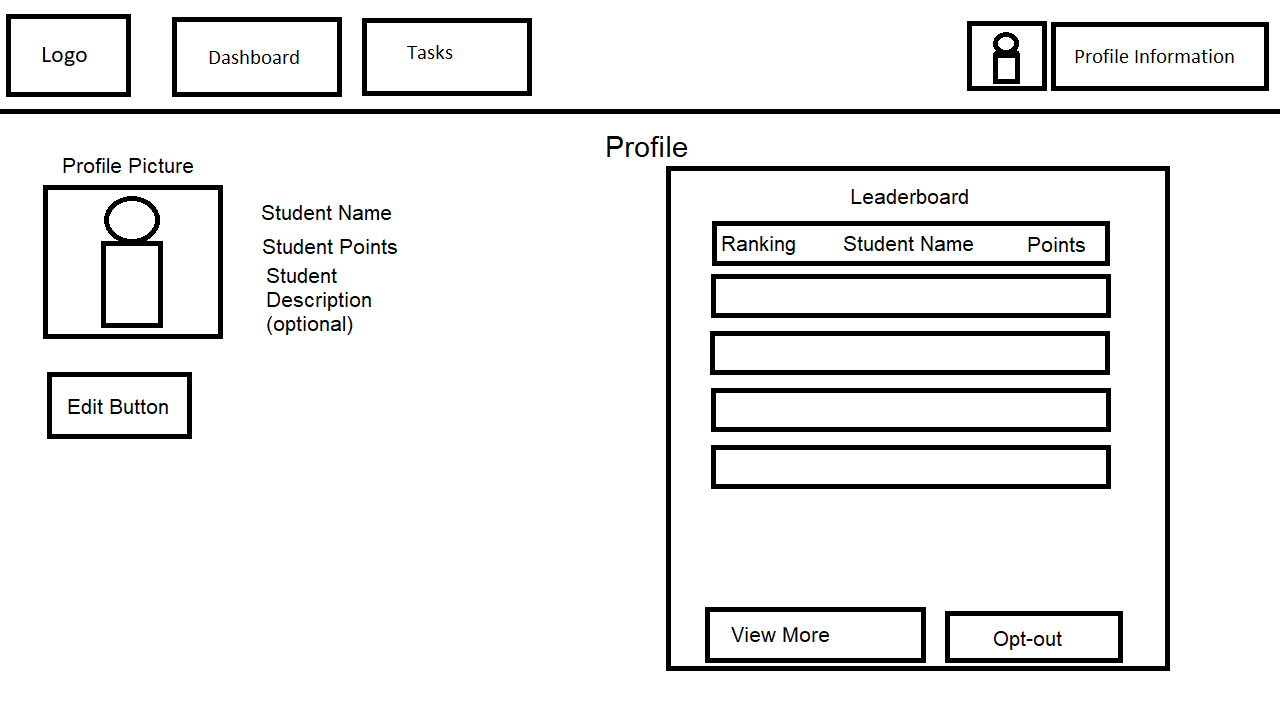
User Interface:

* Smooth transitions
* Use readable and matching colours
* Different boxes of information
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Search bar for tasks
* Task entries presented in order of score
* Teacher feedback box
* Pinned and Homework icons
* Modal pop-up when task clicked

## Student Profile Page



Page Specification:

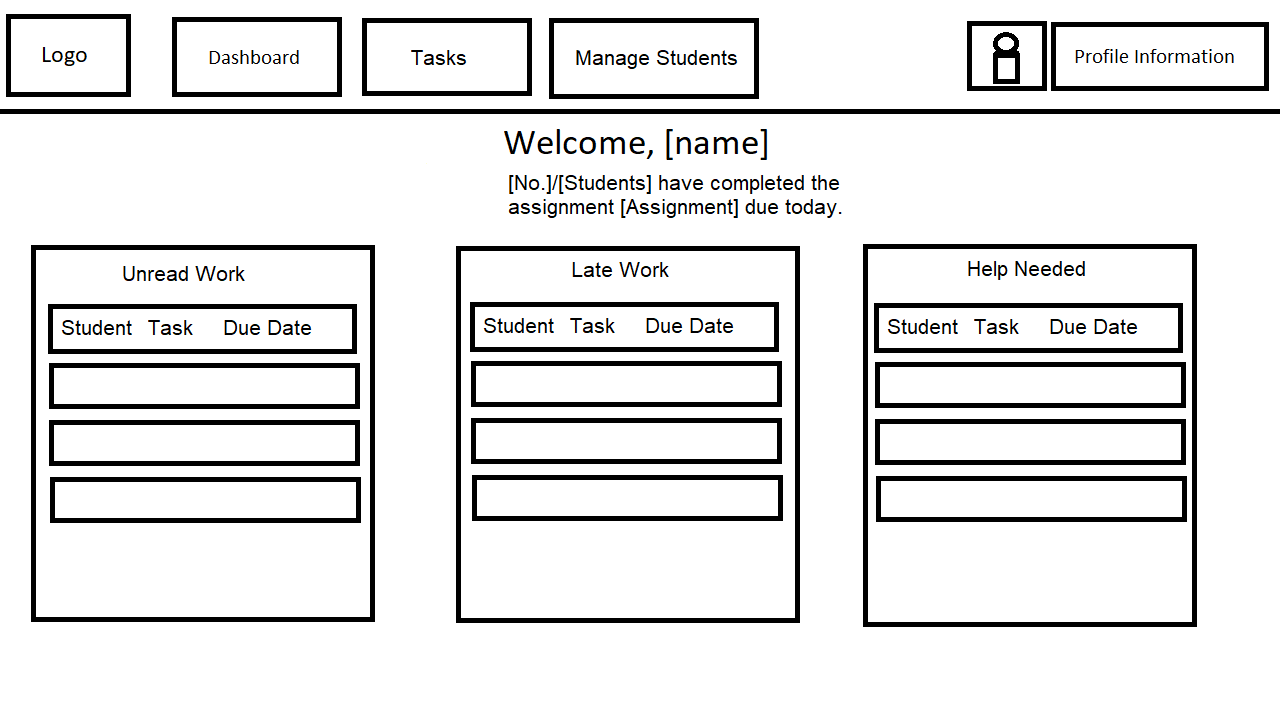
User Interface:

* Smooth transitions
* Use readable and matching colours
* Different boxes of information
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Leader board box
* Leader board entries in order of ranking (first entry is student)
* Profile Information
* Edit Profile Button
* Modal pop-up when leader board entry is clicked

## Teacher Dashboard



Page Specification:

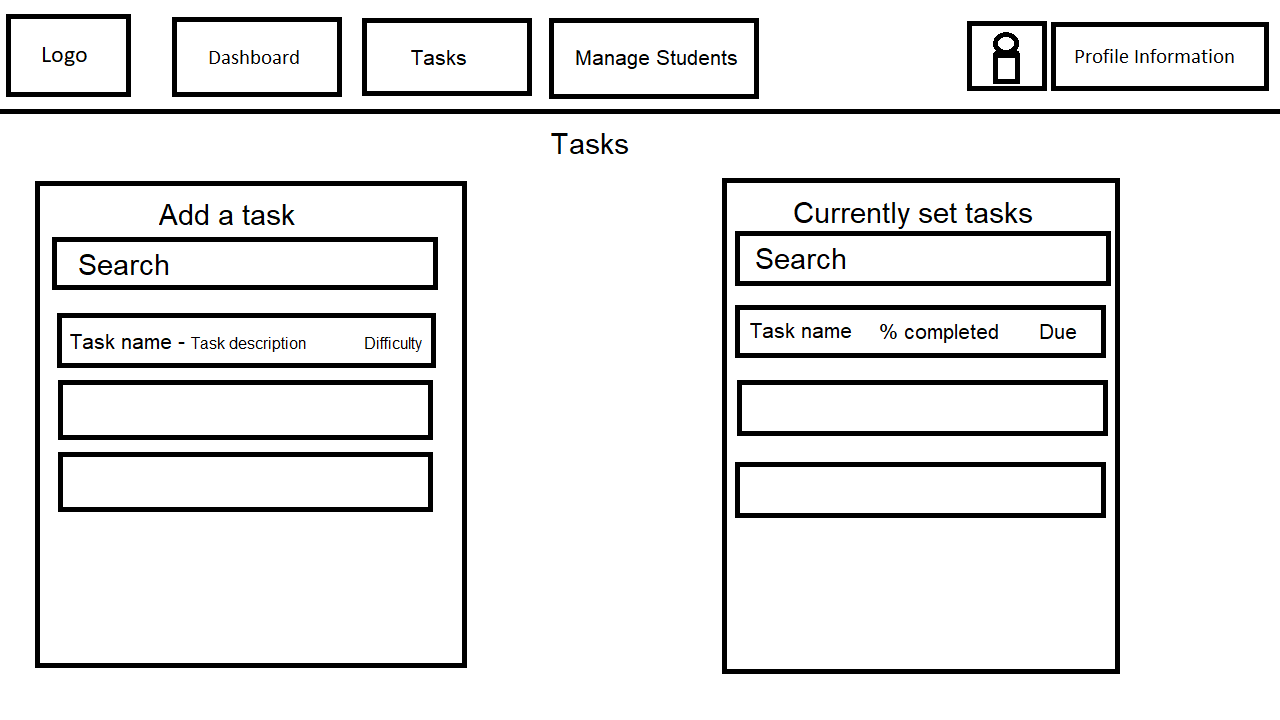
User Interface:

* Smooth transitions
* Use readable and matching colours
* Different boxes of information
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Box to display set tasks
* Box to display position on leader board
* Box to display message inbox
* Customised welcome text
* Modal pop-ups when items are clicked

## Teacher Tasks Page



Page Specification:

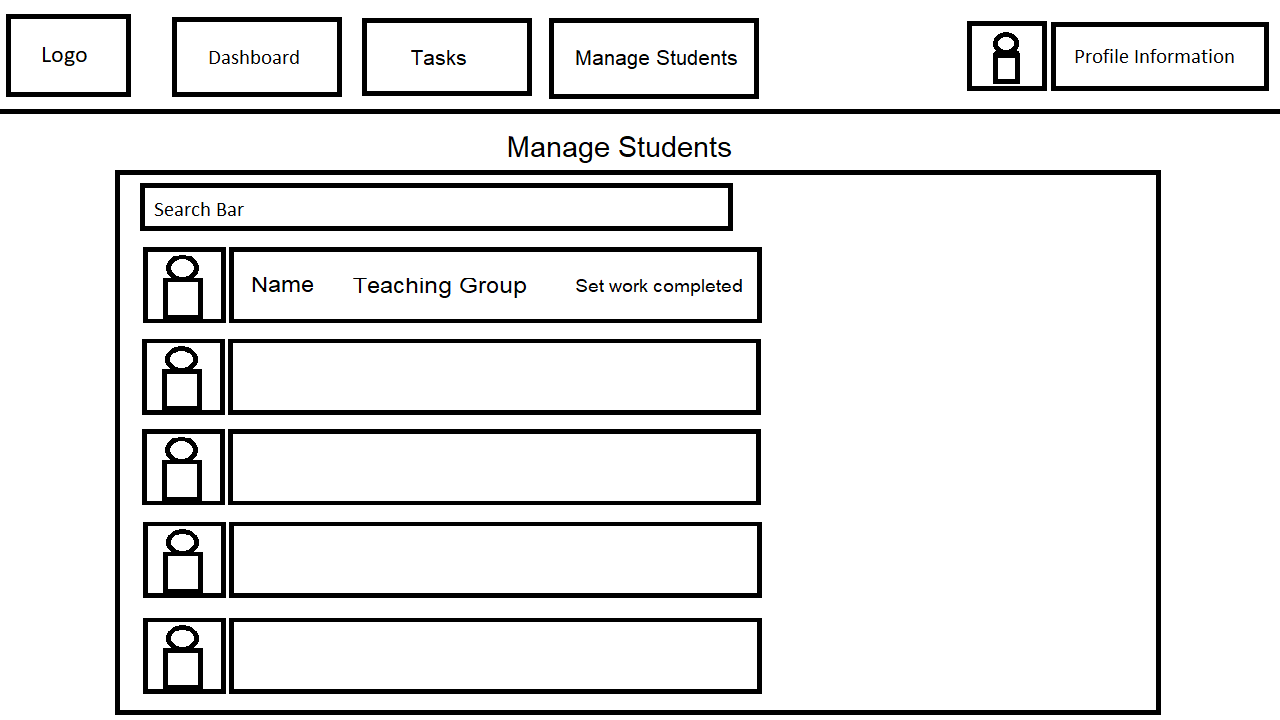
User Interface:

* Smooth transitions
* Use readable and matching colours
* Different boxes of information
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Box to add tasks to teaching groups or students
* Box to display tasks that have already been set
* Search bars
* Modal pop-up when task clicked

## Teacher Manage Students Page



Page Specification:

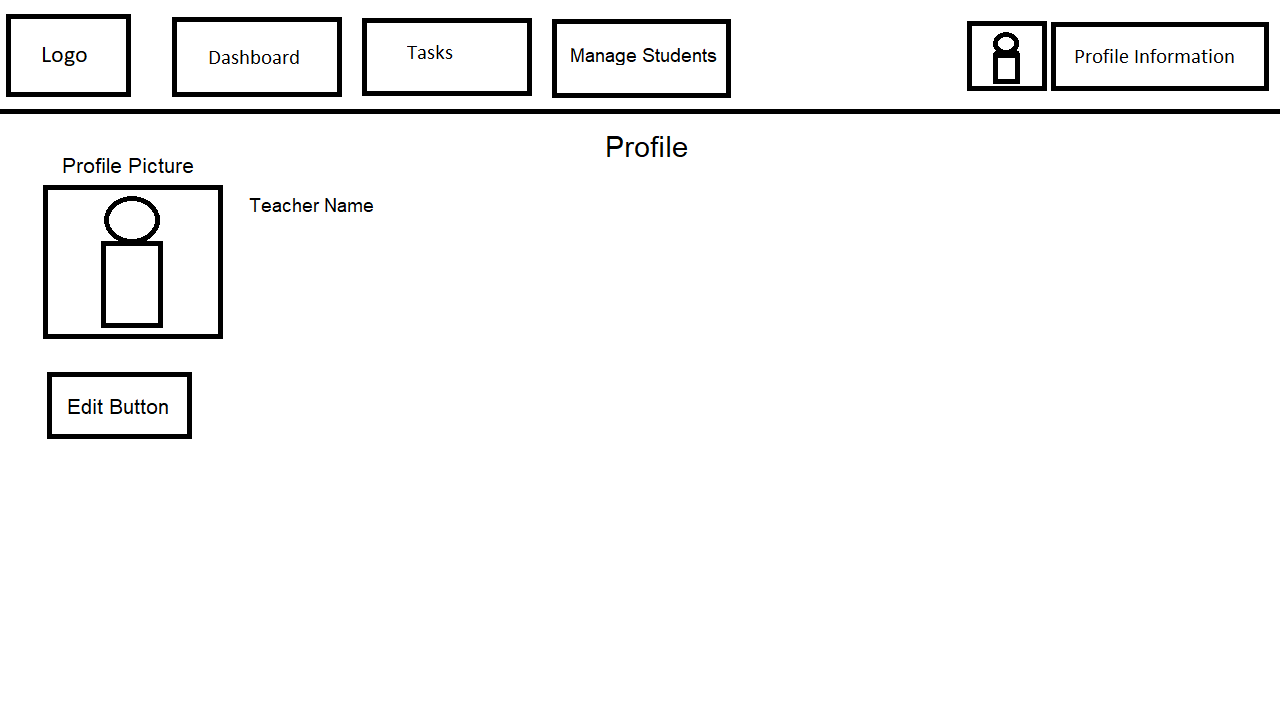
User Interface:

* Smooth transitions
* Use readable and matching colours
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Search bar to find students or teaching groups
* Modal pop-up when student clicked

## Teacher Profile Page



Page Specification:

User Interface:

* Smooth transitions
* Use readable and matching colours
* Tooltips using bootstrap to make the page more accessible

Elements:

* Navbar with links to different pages and profile information
* Profile Information
* Edit Profile Button

# Front End Requirements

* Interface must be simple and organised
* Interface must be as helpful and accessible to the end-user as possible
* Must have a consistent colour scheme throughout the site
* Website must be designed in a way that encourages the user to read from the top down

(F-Pattern)

* Make use of user-friendly icons where possible
* Search bars must be easy to use
* Must allow easy navigation around the site using the navigation bar

# Back End Requirements

* Properly set-up relational database
* Use of templates
* Hash passwords

# Algorithms

## Login in pseudocode

#### Function Main ():

Set AttemptCounter to 0 //Maybe in database so the counter persists

Set AttemptInvalidTimer to 0 //Timer until password can be attempted again

//Should also go in the database

Display Login Page

If Login Button is Pressed, then:

If AttemptInvalidTimer is 0:

Set AttemptInvalidTimer to 0

Do Validation(Email)

If returned True, then:

Do Login(Email, Password)

If returned no error, then:

Add UserLoggedIn to the Session

Display dashboard page

Else if password error:

Display Login page with error: “The Password entered is Invalid.”

Increment AttemptCounter by 1

If AttemptCounter is greater than 3, then

Set AttemptInvalidTimer to currentTime() + 300 //5 Minutes

Else if user not found error:

Display Login page with error: “This User does not exist, please create an account.”

EndIf

Else:

Display Login page with error: “The E-mail Address entered is Invalid.”

EndIf

Else If AttemptInvalidTimer is less than currentTime():

Set AttemptInvalidTimer to 0

Set AttemptCounter to 0

EndIf

EndIf

End

#### Function Validation (Email):

Email Valid = True

If length of email is less than 6

set Email Valid to False

EndIf

If email contains any special characters (Excluding @ and .)

Set Email Valid to False

EndIf

End

#### Function Login (Email, Password):

User = Get user from database using Email.

If User exists (not null or empty), then:

Hash = Make hash from password

Password Match = Compare New Hash with Database Hash (User)

If Password Matches, then:

UserLoggedIn = User

Else:

Return password error

EndIf

Else:

Return user not found error

EndIf

End

### Testing

|  |  |  |
| --- | --- | --- |
| Test Number | Test Data | Expected Outcome |
| 1 | Email = “invalid()\*&@?.com” | Error message: “The E-mail address entered is invalid.” |
| 2 | Password = “” or a password that is too long | Error message: “The password entered is invalid.” |
| 3 | Email = “[invalidperson@email.com](mailto:invalidperson@email.com)” (A user that doesn’t exist) | Error message: “This User does not exist, please create an account.” |
| 4 | Email = “[validperson@email.com](mailto:validperson@email.com)”  (A user that is valid)  And  Password = “validPassword”  (The correct password for the user) | Login succeeds, Page redirects to dashboard. |

Diagram

Description automatically generated

## Display task view modal pseudocode

#### Function Main ()

Display Page Without Modal

If Modal Button Pressed:

TaskID = Get Task ID from button

Open connection to SQL Database

If Database Connection has an Error, then:

Display error modal with text “Internal Server Error”

return

EndIf

TaskData = Request row from database using Task ID

Close connection to SQL Database

If TaskData is Empty or an Error, then:

Display error modal with text “The selected task is invalid”

return

EndIf

Do SetTaskModal(TaskData)

Display Task Modal with new data

End

#### Function SetTaskModal (TaskData)

Set Template variable TaskName using TaskData

Set Template variable TaskSubject using TaskData

Set Template variable TaskDescription using TaskData

Set Template variable TaskDuration using TaskData

Set Template variable TaskSetBy using TaskData

End

### Testing

|  |  |  |
| --- | --- | --- |
| Test Number | Test Data | Expected Outcome |
| 1 | TaskID == -1 or any number that does not correlate to a task | Error modal: “The selected task is invalid” |
| 2 | TaskName == “” | Task name in modal = “No Name” |
| 3 | TaskSubject == “” | Task subject in modal = “No Subject” |
| 4 | TaskDescription == “” | Task description in modal = “No Description” |
| 5 | TaskDuration == any invalid number | Task duration in modal = “Invalid Duration” |
| 6 | TaskSetBy == “” | Task Set By in modal = “Unknown” |
| 7 | Database offline | Error modal: “Internal Server Error” |

## Display Task Cards

//Create a dynamic array

//Query the database for all set tasks for the student ID

//Place all returned rows into the dynamic array

//Create a for loop that increments through the array and creates HTML for each entry

//(could use Laravel syntax, in end product could send the array to the view and loop from there instead)

Function GenerateTaskList(ThisStudentID):

Set TaskArray to empty array

Try:

Connect to database

Query SetTasks table for all rows where StudentID is equal to ThisStudentID

Place query results in TaskArray

For each entry in TaskArray, do:

Create a TaskCard template

Set CurrentSetTask to the current entry of TaskArray

Query Tasks table using the TaskID from CurrentSetTask

Set CurrentTask to the result of the query

Set the template value {{TaskCardName}} to the value of TaskName in CurrentTask

Set the template value {{TaskCardDescription}} to the value of TaskDescription in CurrentTask

Set the template value {{TaskCardScore}} using a function CalculateScore(TaskDifficulty in CurrentTask)

Set the template value {{TaskCardDue}} to the value of SetTaskDate in CurrentSetTask

EndFor

Catch Database Errors:

Display error message "Server error, please try again later."

//Could have an admin debug mode where error codes could be displayed.

Return

# Test Plan for the Solution

Below is a test plan containing both white box and black box tests. White box testing is the process of testing a software solution with the underlying code in mind. In a white box test the tester will be testing each part of code so that they can write down where an error occurs. This makes the code easy to fix. Black box testing on the other hand is the process of testing a software solution without knowing the underlying source code. Black box testing better reflects the experience of the average end-user while white box testing is better for testing against malicious activity and covers different known attack vectors.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Test Type | Test Name | Test Purpose | Test Data | Expected Outcome |
| 1 | Unit Test | Login | To see if a user can log in | Correct Email and Password entered | Should log in successfully and redirect to the dashboard |
| 2 | Unit Test | Registration | To see if a student or teacher can create an account | A valid email, password, and full name | User should be created successfully and redirect to the login screen |
| 3 | Unit Test | Modals | To see if modals can be created and displayed using jQuery AJAX, PHP and the BladeOne template engine | Click on a task entry to open the modal | Modal should be created with all correct data and be sent to the client using jQuery’s $.ajax() method |
| 4 | Interface Test | Navigation Bar | To see if a user can navigate between pages smoothly | Click on a link to a different page | Current page should fade out and new page should fade in |
| 5 | Integration Test | Login System (Login, Register, Logout) | To see if a user can log in, log out and register with the service | Create an account and log in, then, log out | User should be able to create, log in and log out of their account without error |
| 6 | Unit Test | Task Setting | To see if a teacher can assign a task to a teaching group | Open the Task page as a teacher and assign a task to a teaching group | All students in the teaching group should be able to see the set task along with details about it |
| 7 | Unit Test | Login | To see if the incorrect password system works properly | Enter an invalid password 4 times | User should be unable to try again for 5 minutes. |

# Data Requirements for the Login system

Below is a list of some of the data requirements involved in creating the login system. They are listed in a data dictionary as a list of variables with their data types, a description of the variable and a reason for the variables use.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Data Type | Reason |
| $email | Stores the users email address | String | This variable is needed to search the database to find the user with this email |
| $emailValid | Used to store whether the email is valid. | Boolean | This variable is used in the email verification process to check if the current email is valid. |
| $password | Stores the unhashed password of the user | String | In development I should just be using the $\_POST global variable but for the sake of representing the password I will use this |
| $passwordHash | Stores the hashed password of the user | String | This variable stores the password hash of the user so it can be compared to the correlating password hash from the database |
| $User | An object that contains methods to log a user in and out, as well as information about the user when logged in. | Object | Creating a user object means that I can abstract model from controller, this means that I will not have to write database code outside of objects like this |
| $\_SESSION | The session global variable | Array | Stores values in the session. Data in this variable will persist until the browser is restarted or until the session is explicitly closed. Used to store the ‘User’ Object when logged in |
| $attemptInvalidTimer | The time at which a password can be attempted again. | Integer | Allows a password attempt cooldown timer to be implemented |
| $attemptCounter | Counts the amount of invalid password attempts | Integer | Is incremented every incorrect attempt, when the value is greater than 3, the timer will be started. |

# Data Requirements for Task View Modal

Here is another data dictionary for the Task View Modal. Modals will be stored as templates in a separate folder and will be created using a PHP script and called with a POST request from the client side.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Data Type | Reason |
| $taskId | Stores the ID of the current task from the client. | Integer | The ID is stored so that the task can be looked up in the database |
| $taskData | Stores each entry of the task modal | Array | It can be pulled directly from the database |
| $conn | Object for the SQL connection | Object (PDO) | A connection to the database is needed to pull and modify data on it. |
| {{TaskName}} | The template variable for the task name | String | Templates allow the separation of view and controller. This can make code cleaner and simpler as there is no PHP in the actual task view modal file.  This variable is used to change the task name in the template |
| {{TaskSubject}} | The template variable for the task subject | String | This variable is needed to change the task subject in the template |
| {{TaskDescription}} | The template variable for the task description | String | This variable is needed to change the task description in the template |
| {{TaskDuration}} | The template variable for the task duration | String | This variable is needed to change the task duration in the template |
| {{TaskSetBy}} | The template variable for the name of the teacher that set the task | String | This variable is needed to change the task duration in the template |

# Entity Relationship Diagram / Database Design

Below is an Entity Relationship Diagram of the database that will help develop the final solution.

Diagram

Description automatically generated

## Data Types

A picture containing diagram

Description automatically generated

# Data Flow Diagrams

## Login System

## View a student’s set tasks

