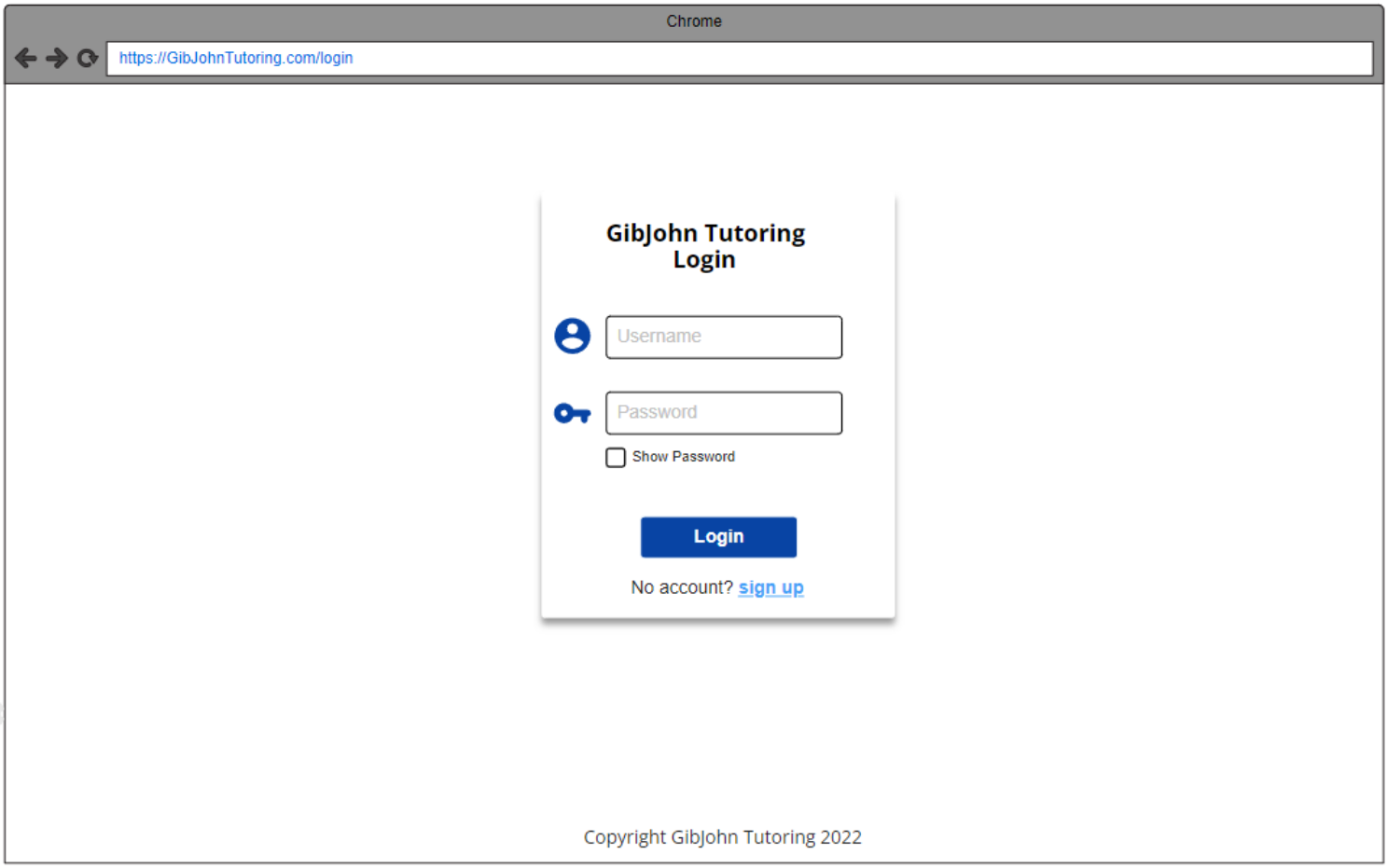
Task 1 B – Design Documents – Nathan Hannah

Interface designs

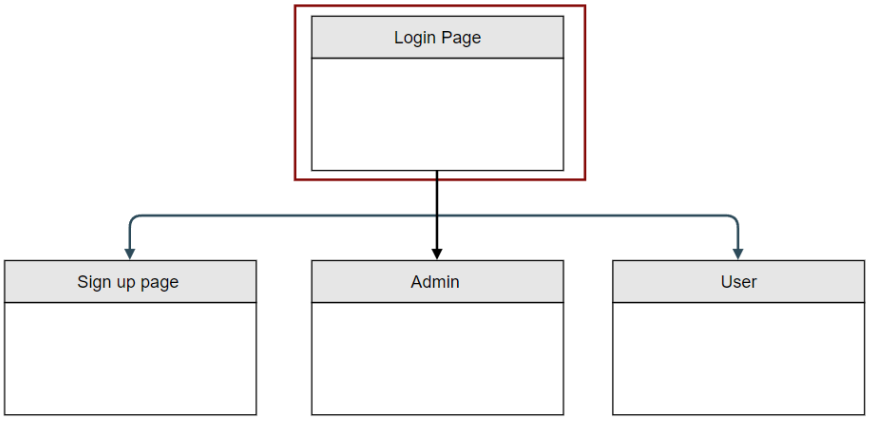
Below are the interface designs for the prototype solution.



Page Flow summary – Login Page

The login page will be the first page a user views, all pages will redirect to this if a user is not logged in. This page will allow controlled access to the system as it prevents unauthorised access.

Once authenticated as user or admin they will be redirected to the homepage for their role.



Login Page

Elements

- Username input field
- Password input field
- Login button
- Link to signup page
- Footer with copyright notice

Features

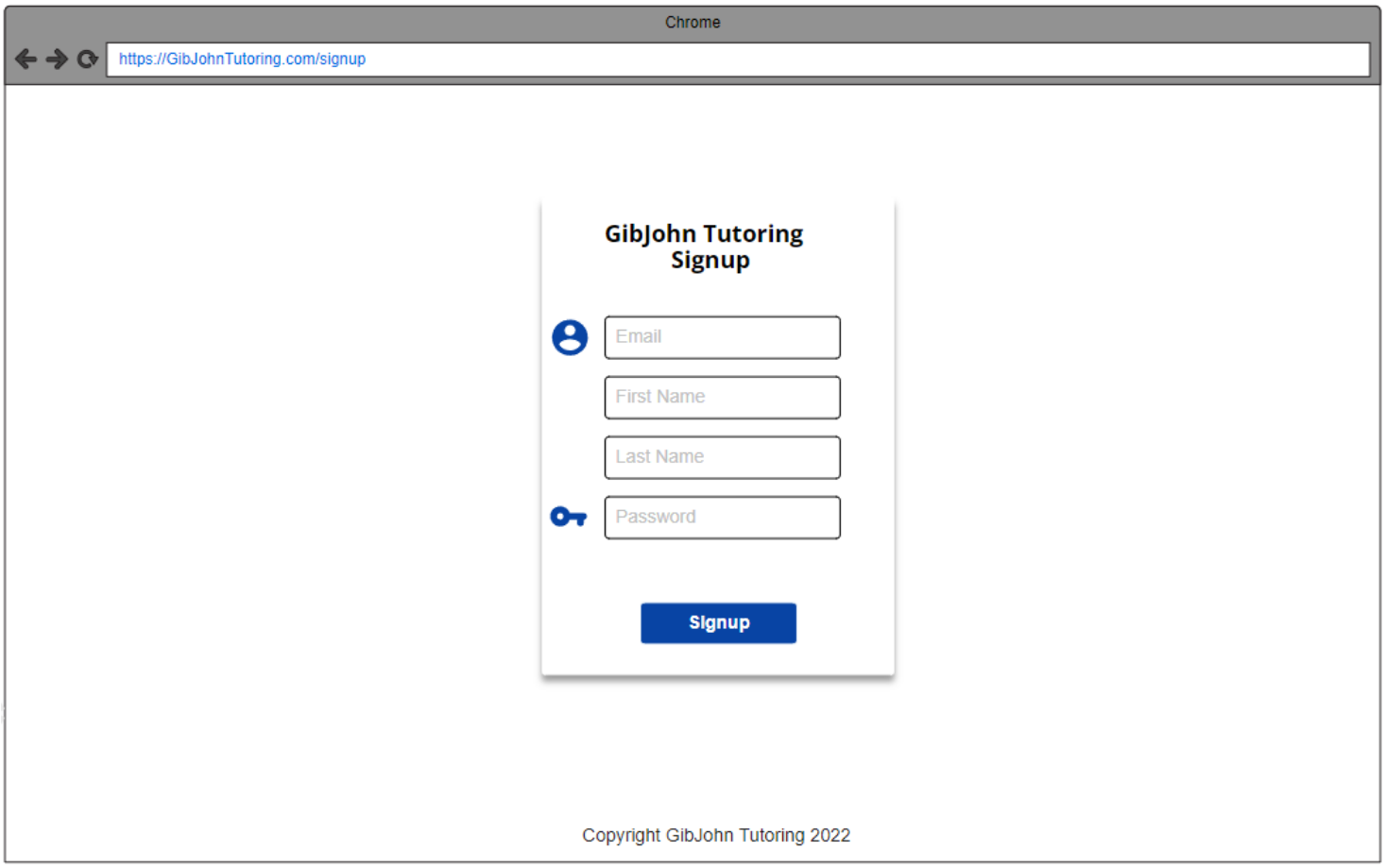
- Show password toggle
- Link to signup page

Fonts

- Open Sans

Icons

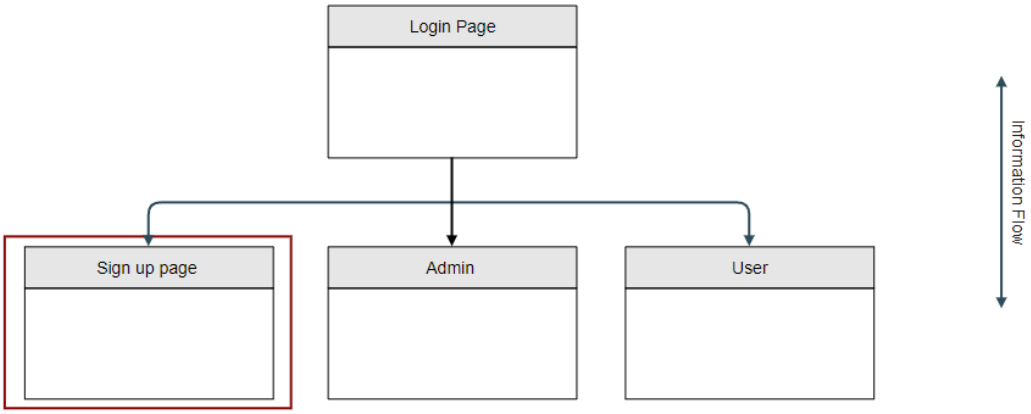
- Username and password icons



Page Flow summary – Signup Page

The signup page can be accessed by anyone; the user will access the login page if they do not have a user account they can be directed to this page.

Once signed up the user can access the user resources.



Signup Page

Elements

- Email input field
- Password input field
- Signup button
- Footer with copyright notice

Features

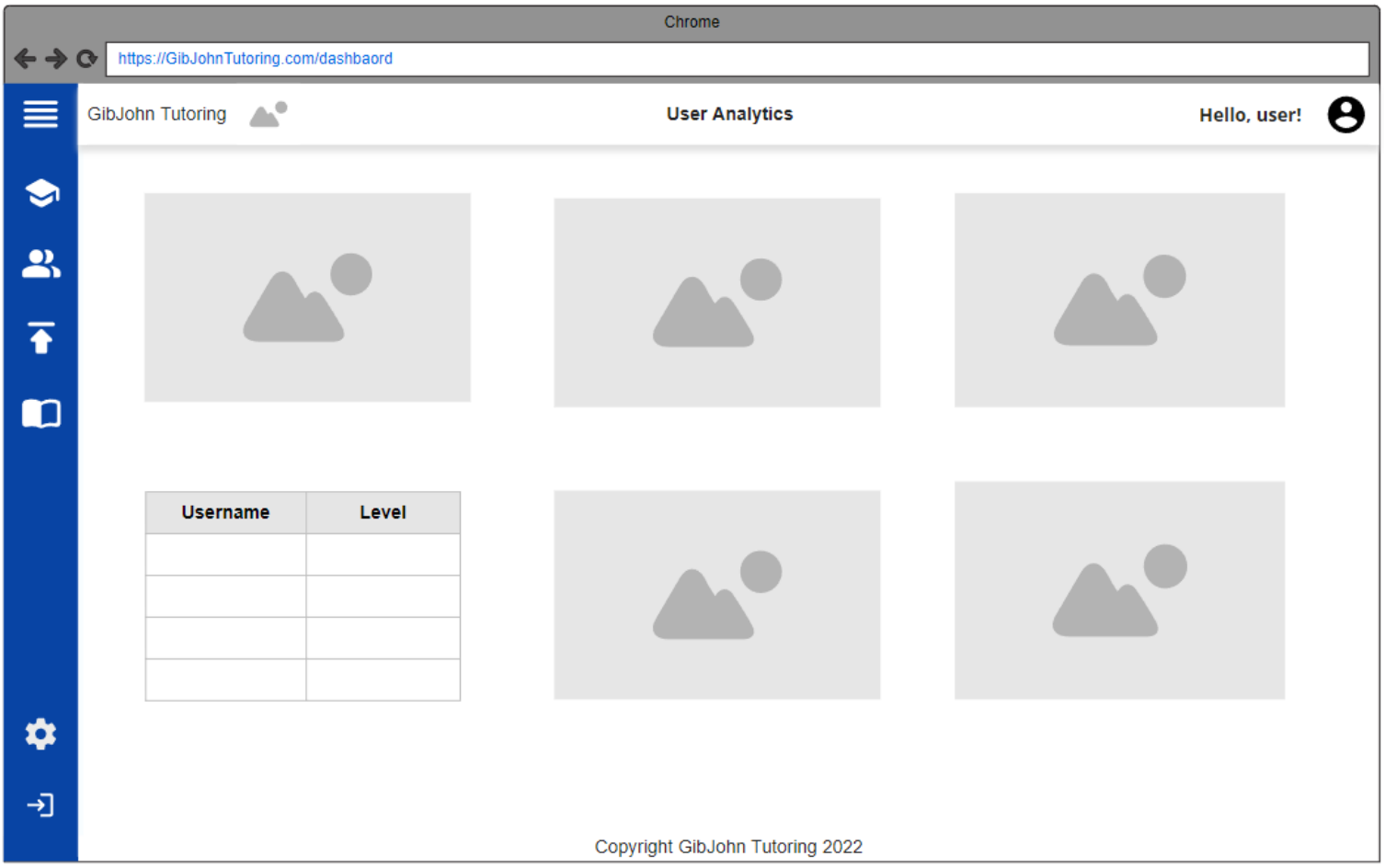
- Allows users to enter details to sign up as a user

Fonts

- Open Sans

Icons

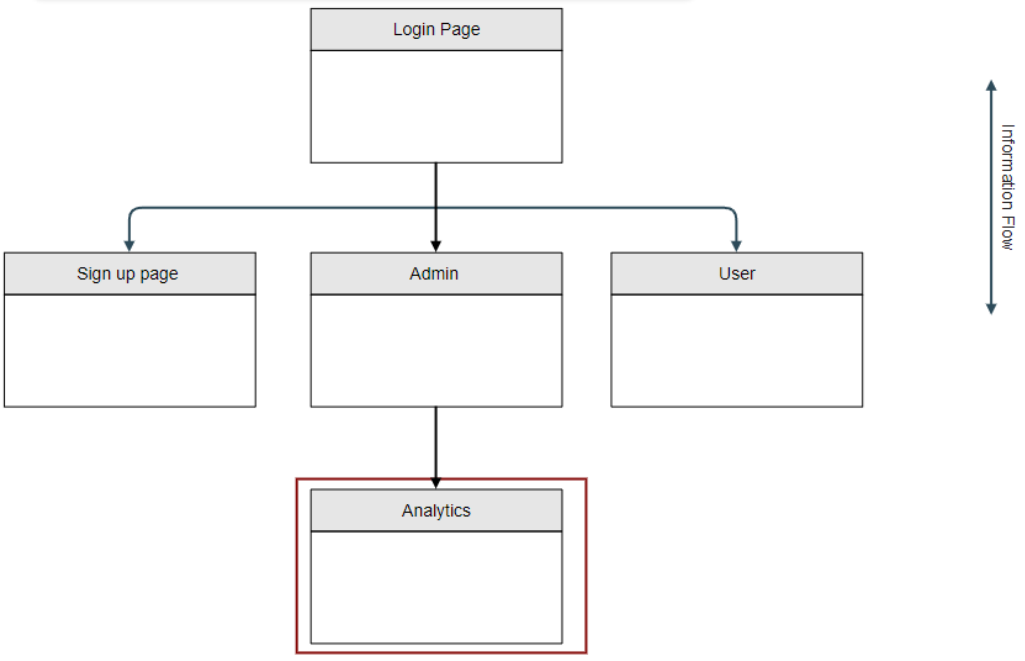
- Username and password icons



Page Flow summary – Analytics Page

Only a signed in admin can access the page, if they are not logged in they will be redirected to the login page.

This page is the first page an admin will see when they access the system.



Admin - Analytics page

Elements

- Side navigation bar to access all pages required
- Data table for user statistics
- Relevant Images and graphs
- Footer with copyright notice

Features

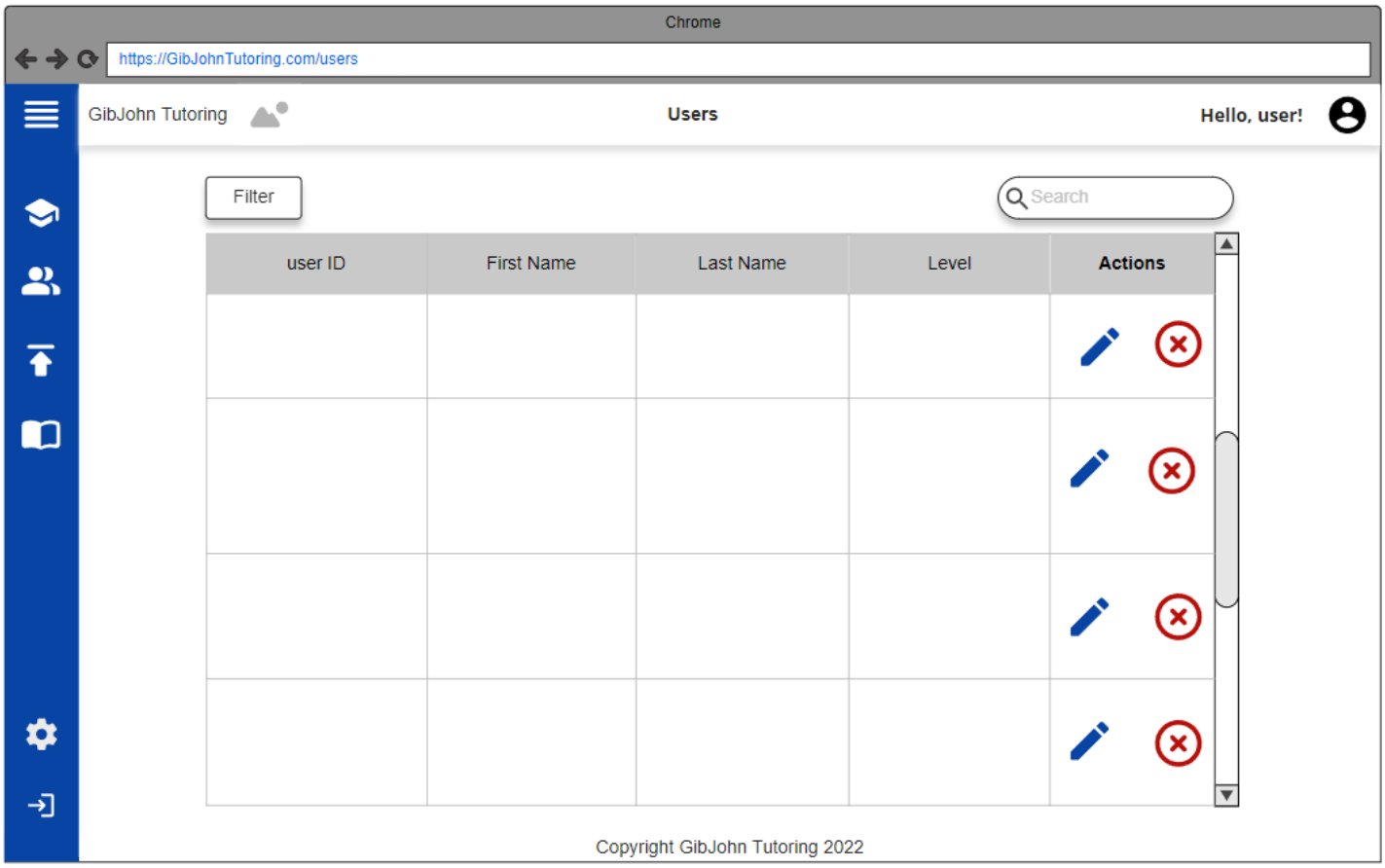
- Allows admins to view all statistics and information about users accessing the system

Fonts

- Open Sans

Icons

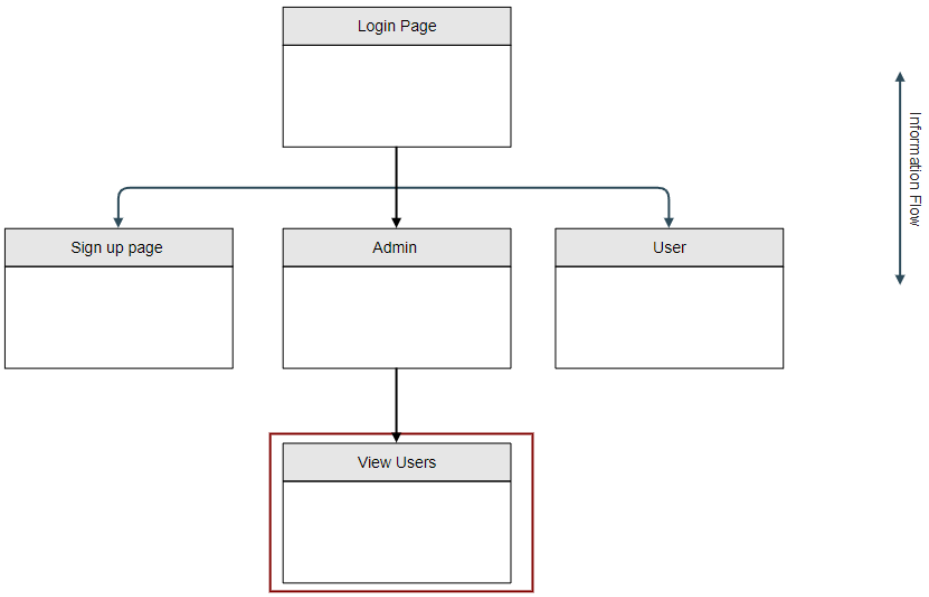
- Navigation bar icons
- Logo Icon



Page Flow summary – Admin - Users page

Only a signed in admin can access the page, if they are not logged in they will be redirected to the login page.

This page can be accessed through the navigation bar.



Admin - Users page

Elements

- Data table to display user information
- Navigation sidebar
- Buttons to edit and delete users

Features

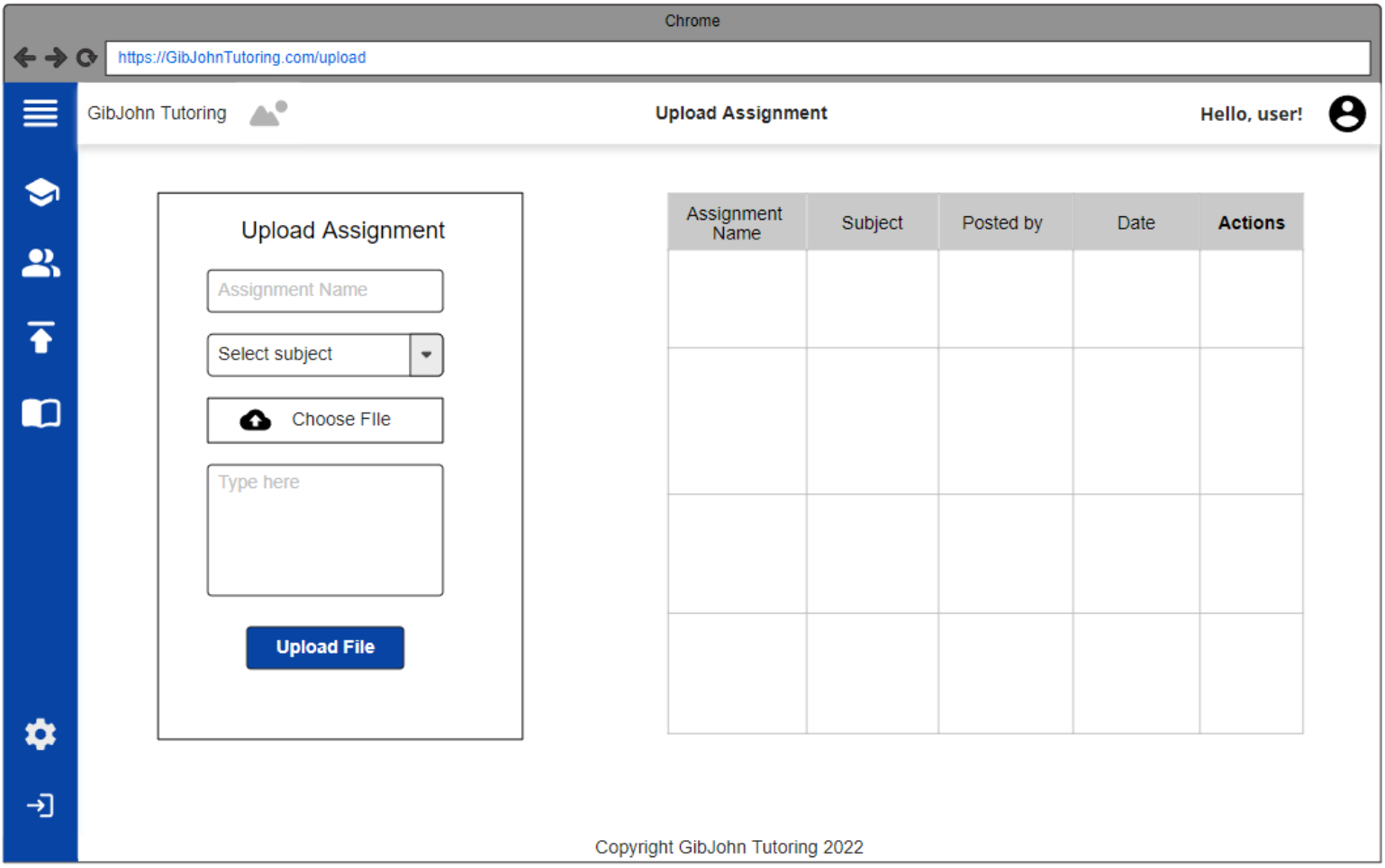
- Allows admins to view all users
- Ability to filter users by column
- A search feature to find specific users

Fonts

- Open Sans

Icons

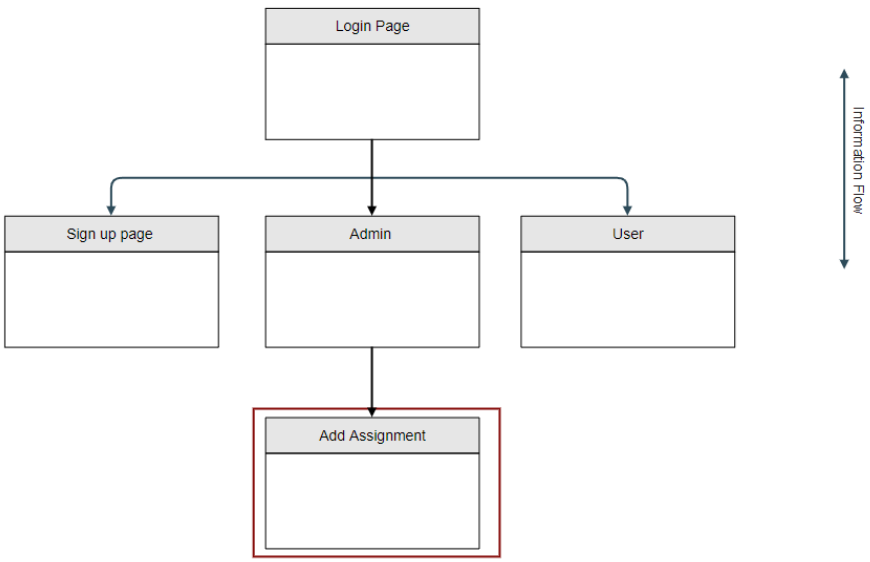
- Navigation bar icons
- Logo Icon



Page Flow summary – Upload assignments

Only a signed in admin can access the page, if they are not logged in they will be redirected to the login page.

This page can be accessed through the navigation bar.



Admin – Upload assignments

Elements

- Data table to view all previous uploaded assignments
- Text area to input assignment description
- Selection input to give the assignment a specific subject

Features

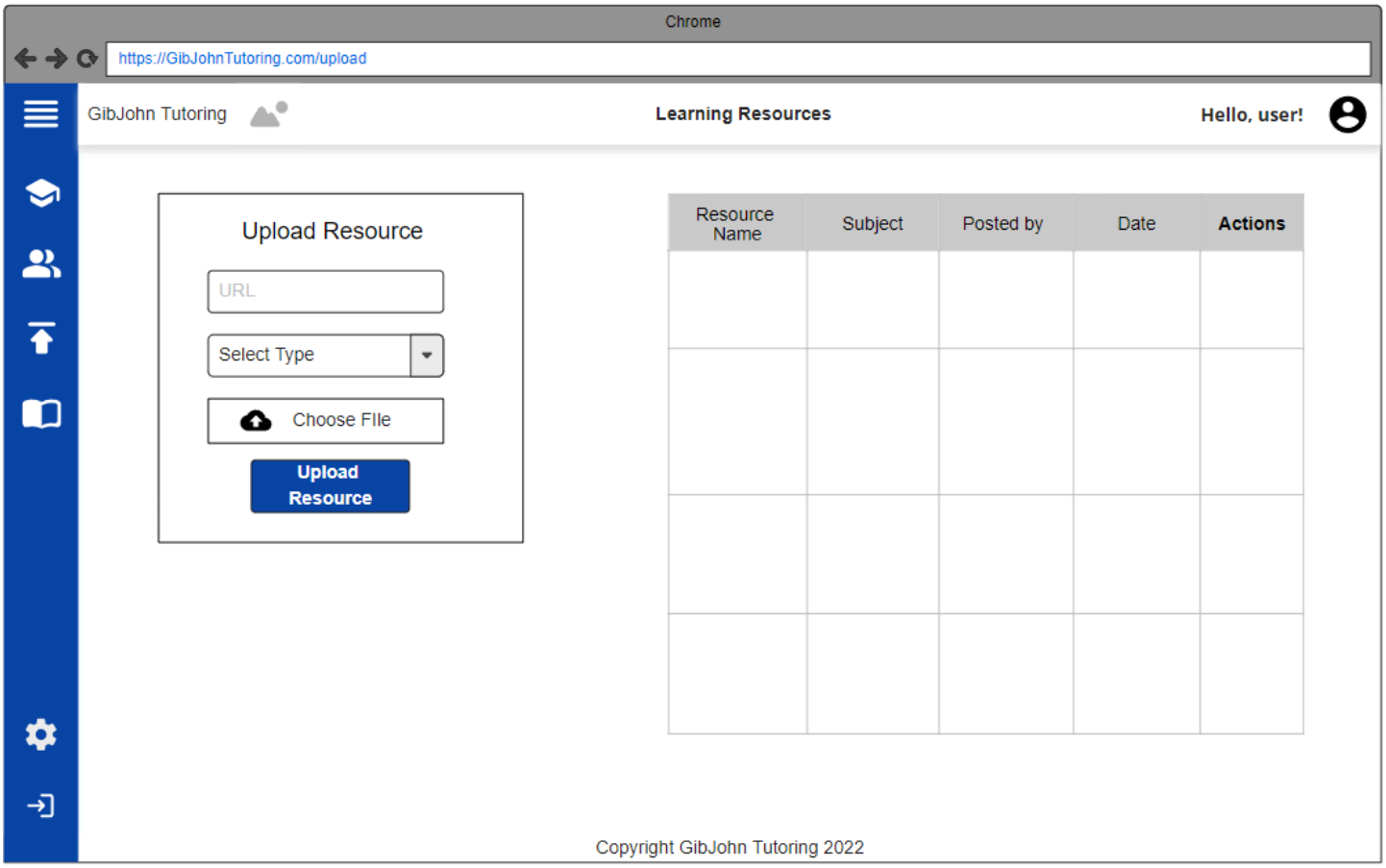
- Allows admins to view previous assignments uploaded
- Allows admins to upload files to users in several formats

Fonts

- Open Sans

Icons

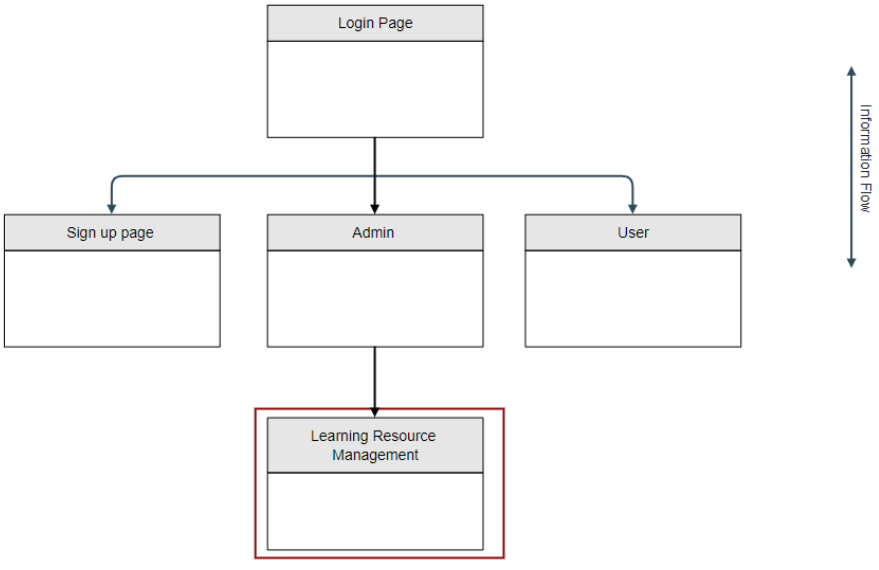
- Navigation bar icons
- Logo Icon



Page Flow summary – Learning resources

Only a signed in admin can access the page, if they are not logged in they will be redirected to the login page.

This page can be accessed through the navigation bar.



Admin – Learning Resources

Elements

- Data table for information display
- Button to submit data
- Selection box to select type of file
- Text box to paste a URL

Features

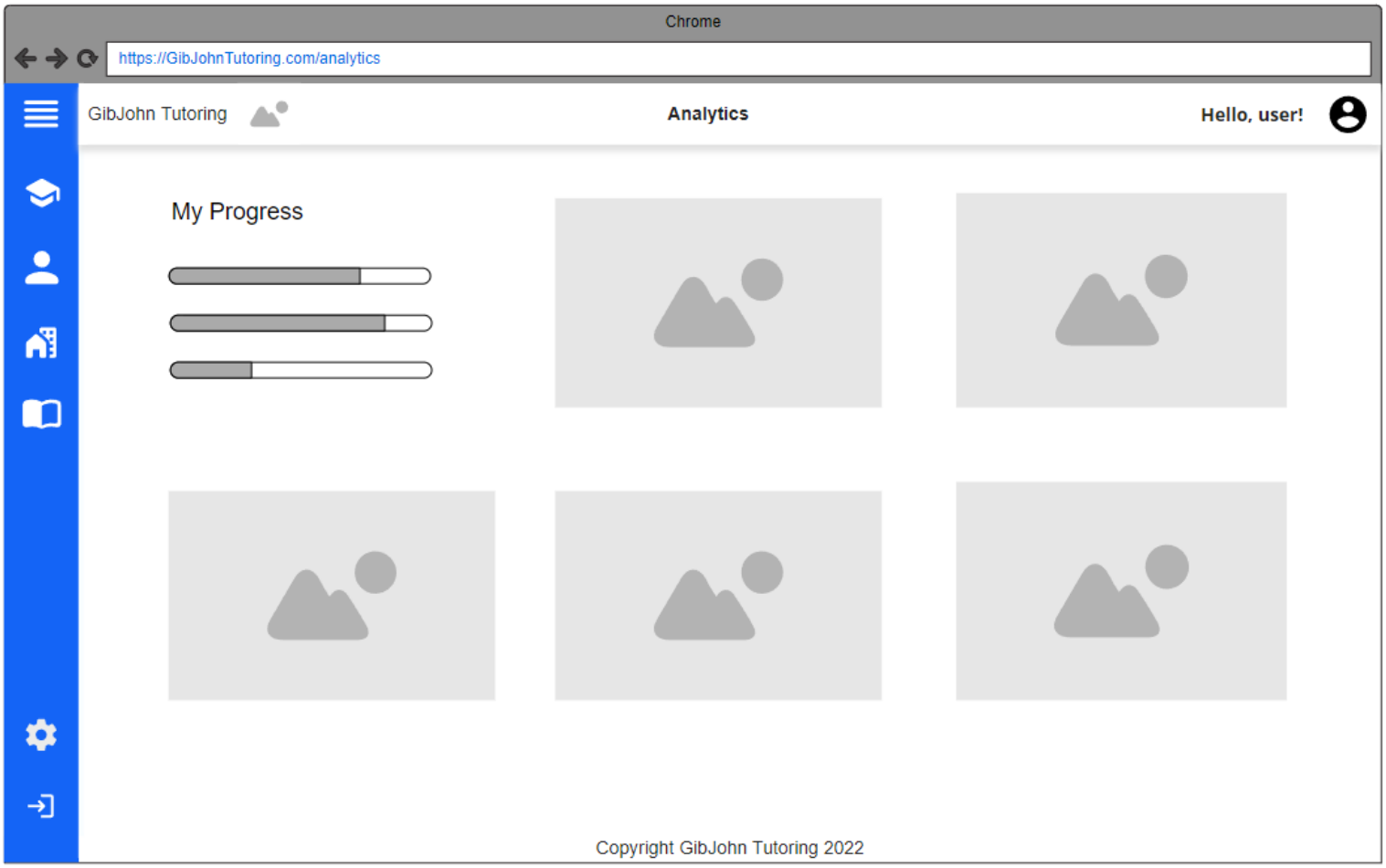
- Allows admins to upload resources through file upload or by URL
- Allows admins to view previously uploaded learning resources

Fonts

- Open Sans

Icons

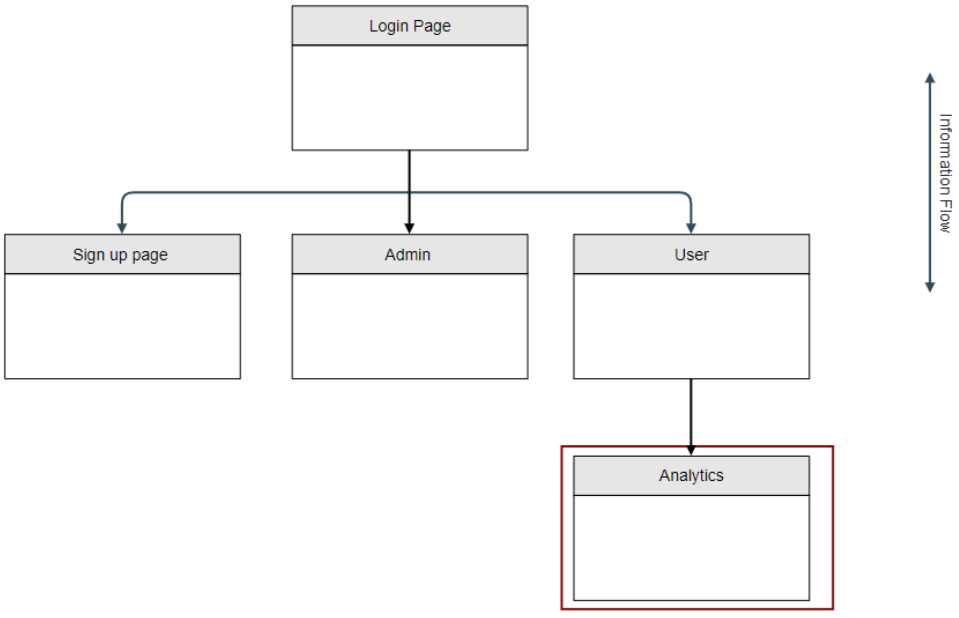
- Navigation bar icons
- Logo Icon
- Upload file icon



Page Flow summary – Analytics

Only a signed in user can access the page, if they are not logged in they will be redirected to the login page.

This page is the page that a user will see as they access the system, giving them overall information



Users – Analytics

Elements

- Progress bar to show user progress
- Images and graphs to give information to the user

Features

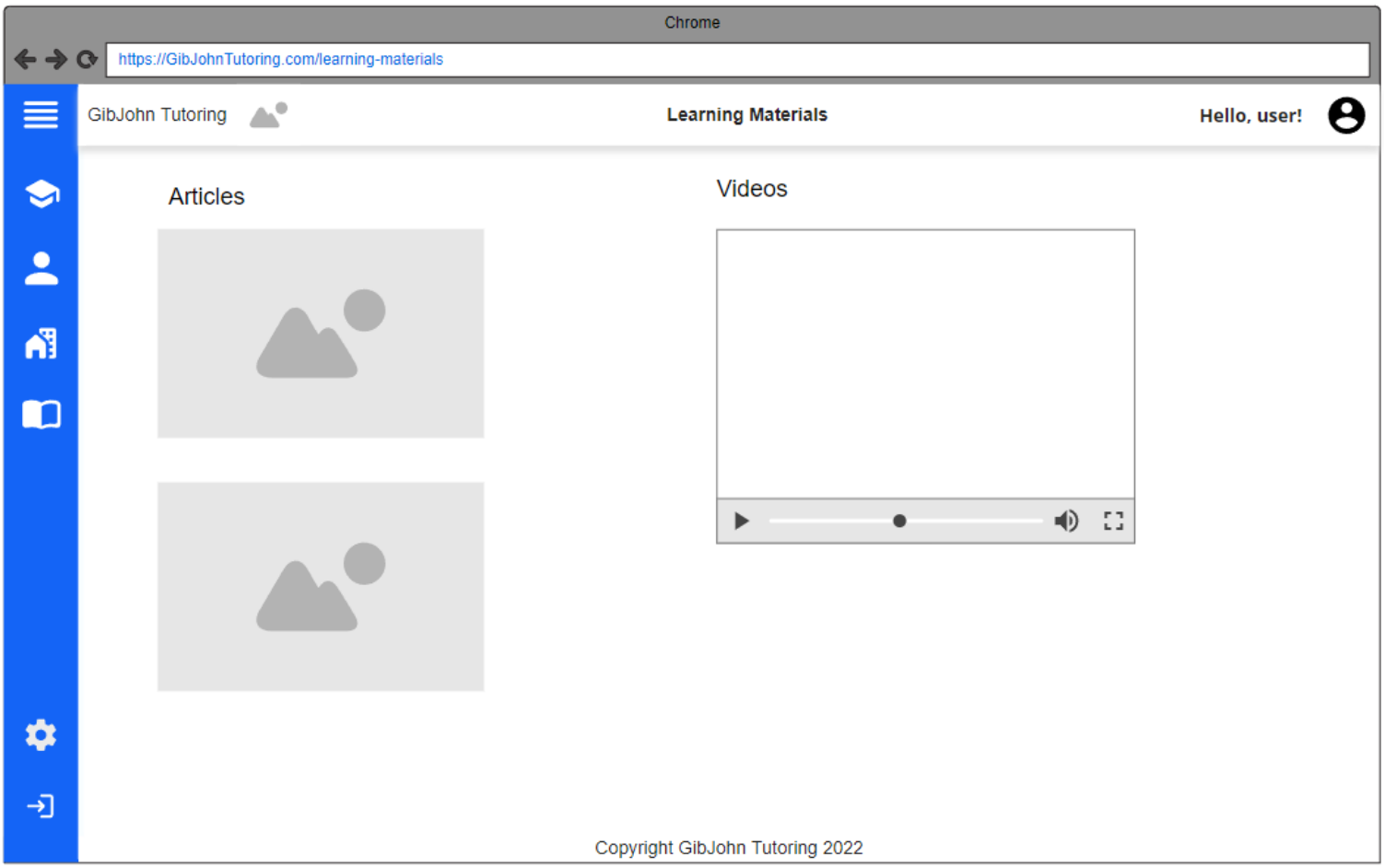
- Allows users to view all statistics and information about them self and compared to others

Fonts

- Open Sans

Icons

- Navigation bar icons
- Logo Icon

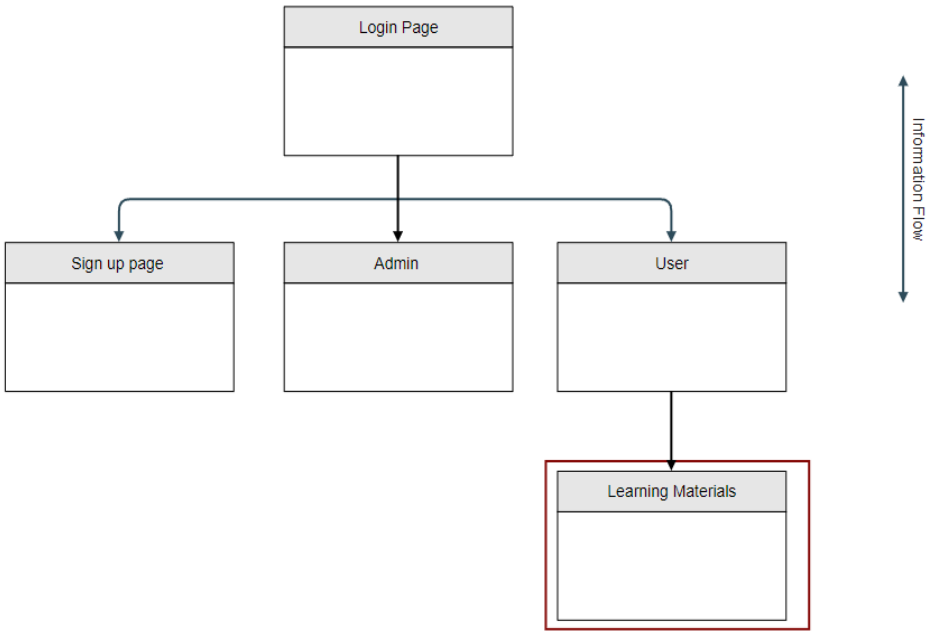


Page Flow summary – Learning Materials

Only a signed in user can access the page, if they are not logged in they will be redirected to the login page.

This page can be accessed through the navigation bar.

This page contains external links to other sites to provide users with useful information.



Users – Learning materials

Elements

- Video player
- Text
- Images

Features

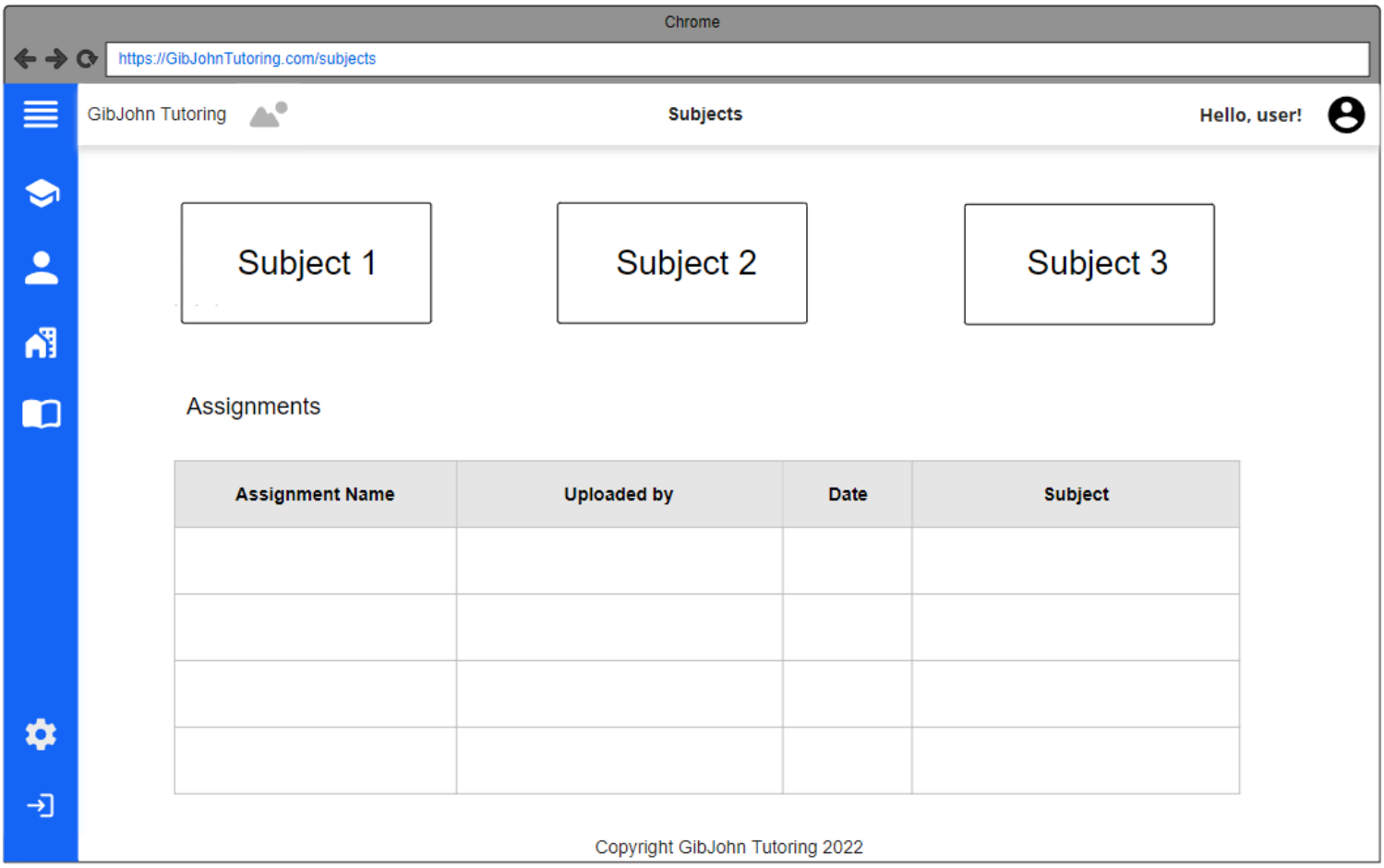
- Shows admin uploaded material
- Shows external links and videos to useful articles relevant to education

Fonts

- Open Sans

Icons

- Navigation bar icons
- Logo Icon

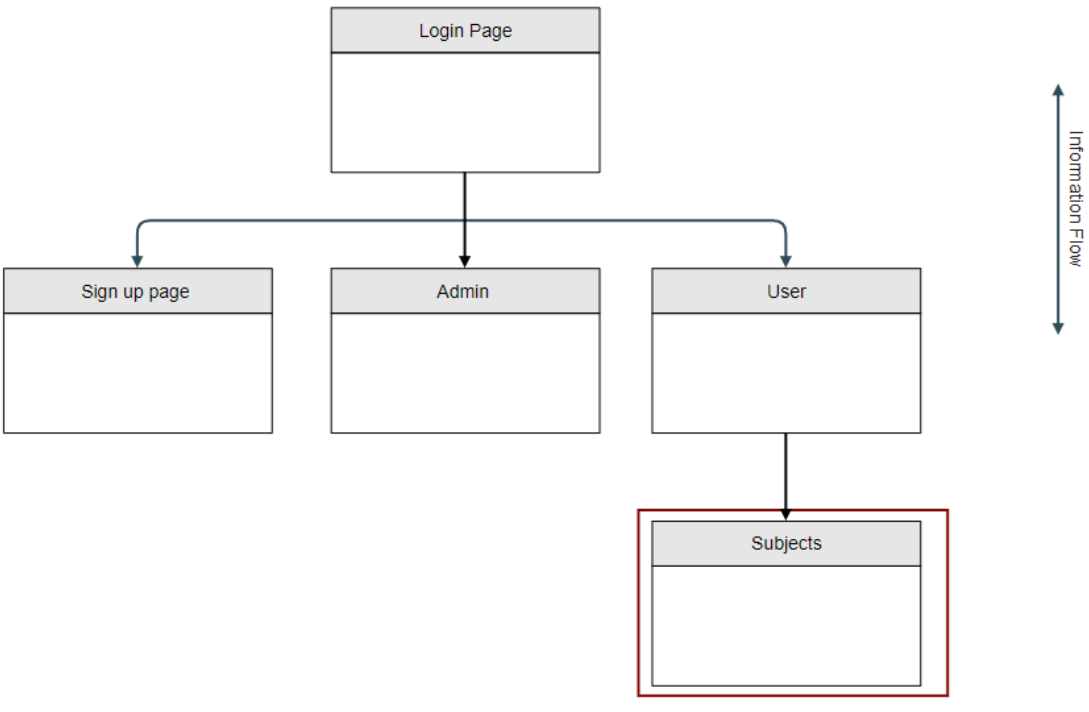


Page Flow summary –Subjects

Only a signed in user can access the page, if they are not logged in they will be redirected to the login page.

This page can be accessed through the navigation bar.

Some data in the data table may provide links to other external sites



Admin – Upload assignment

Elements

- Data table
- Buttons to select subject area

Features

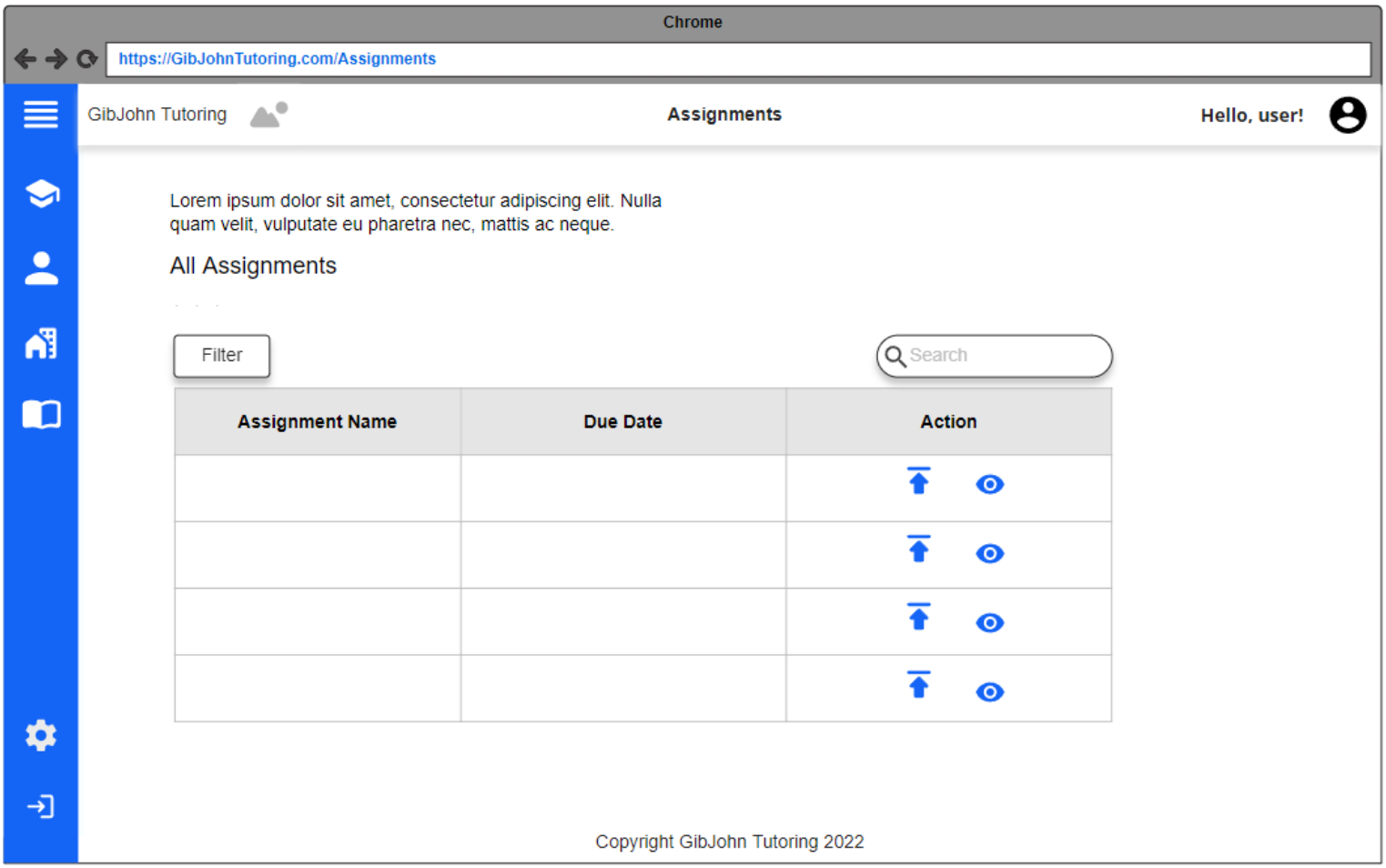
- Allows users to filter files based on subjects
- Buttons with subject area will provide information

Fonts

- Open Sans

Icons

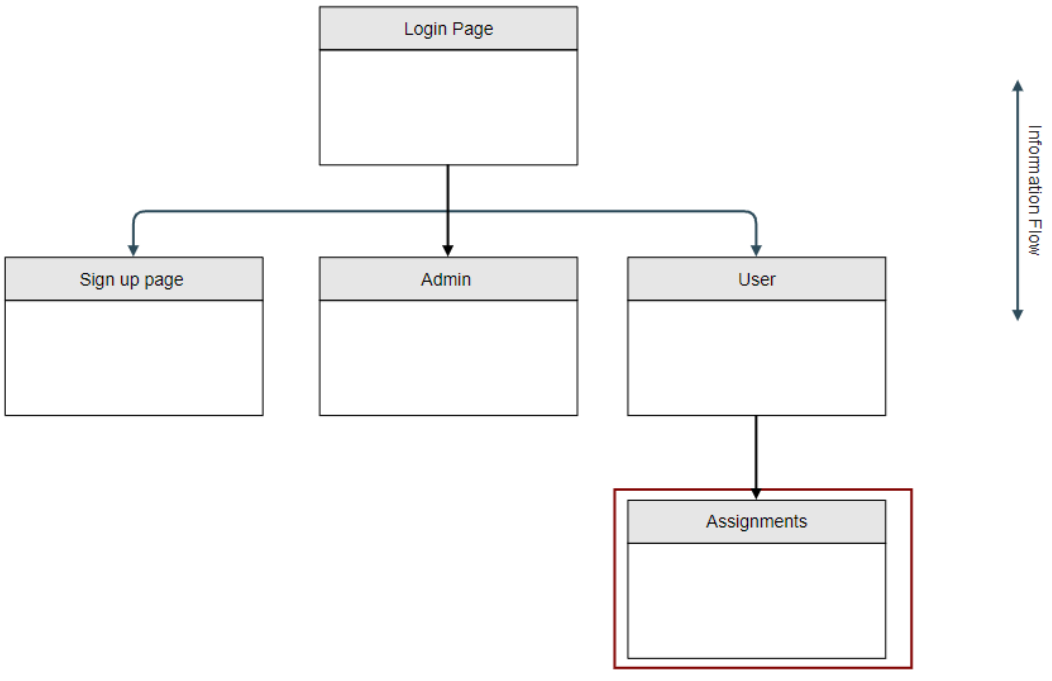
- Navigation bar icons
- Logo Icon



Page Flow summary – Assignments

Only a signed in user can access the page, if they are not logged in they will be redirected to the login page.

This page can be accessed through the navigation bar.



Admin – Upload assignment

Elements

- Buttons to upload and view details
- Data table to store information

Features

- Allows admins to view all assignments uploaded
- Admins can search through assignments

Fonts

- Open Sans

Icons

- Navigation bar icons
- Logo Icon
- Action buttons

Prototype Design Colour Scheme

Colours used in the designs for the prototype solution, these are still to be concluded within the design stage.

Colour	Hex	Location
	#ffffff	Main website body and icons
	#0844a4	Admin sidebar and buttons
	#1464f6	User sidebar and buttons

User interface

Below outline the differences between the admin UI and the user UI

Admin interface	User interface
Interface only accessible to authorised admins	Accessible to users that are logged in, this can be achieved by signup
Admin specific icons are used, such as multiple users and upload assignments	User specific icons are used with a single user icon and work icon
The colour has a dark shade of blue (#0844a4)	Colour has a light shade of blue (#1464f6)

Considerations

Accessibility

- Contrast is used throughout to make text more accessible to read
- The text size is adjustable so that a user that is visually impaired can view the contents without any issues
- Standards are put into place such as red meaning danger, seen with the delete buttons
- Image and videos will have alt tags will be included so they can be read by screen readers

Usability

- Whitespace is used to make the page more easily understandable and simplistic
- Icons are used as well as text to ensure users can navigate the system
- A side navigation bar allows users to easily navigate through systems webpages
- On each page the username of the user logged in can be viewed, this makes it easy to identify who is logged into the system at all times.

Front-end (user-facing)

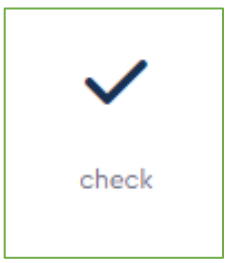
- A webpage delivered to the user with HTML, CSS and Bootstrap
- A GUI that is easy to navigate for the user, allowing them to access all features required
- Display all the required features that have been requested by the client
- A link to the backend of the system to send and receive data to and from the database
-

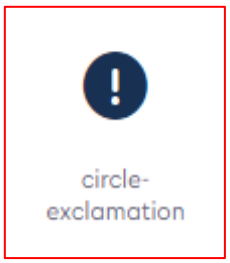
Back-end

- SQL database to store and retrieve data such as user information and credentials
- Server to respond to client requests so that users can use the features within the system and see the webpages
- Link to the front end to respond to requests and send data
- Read/write ability so that users can update settings and upload files
- Efficient way of retrieving and storing data

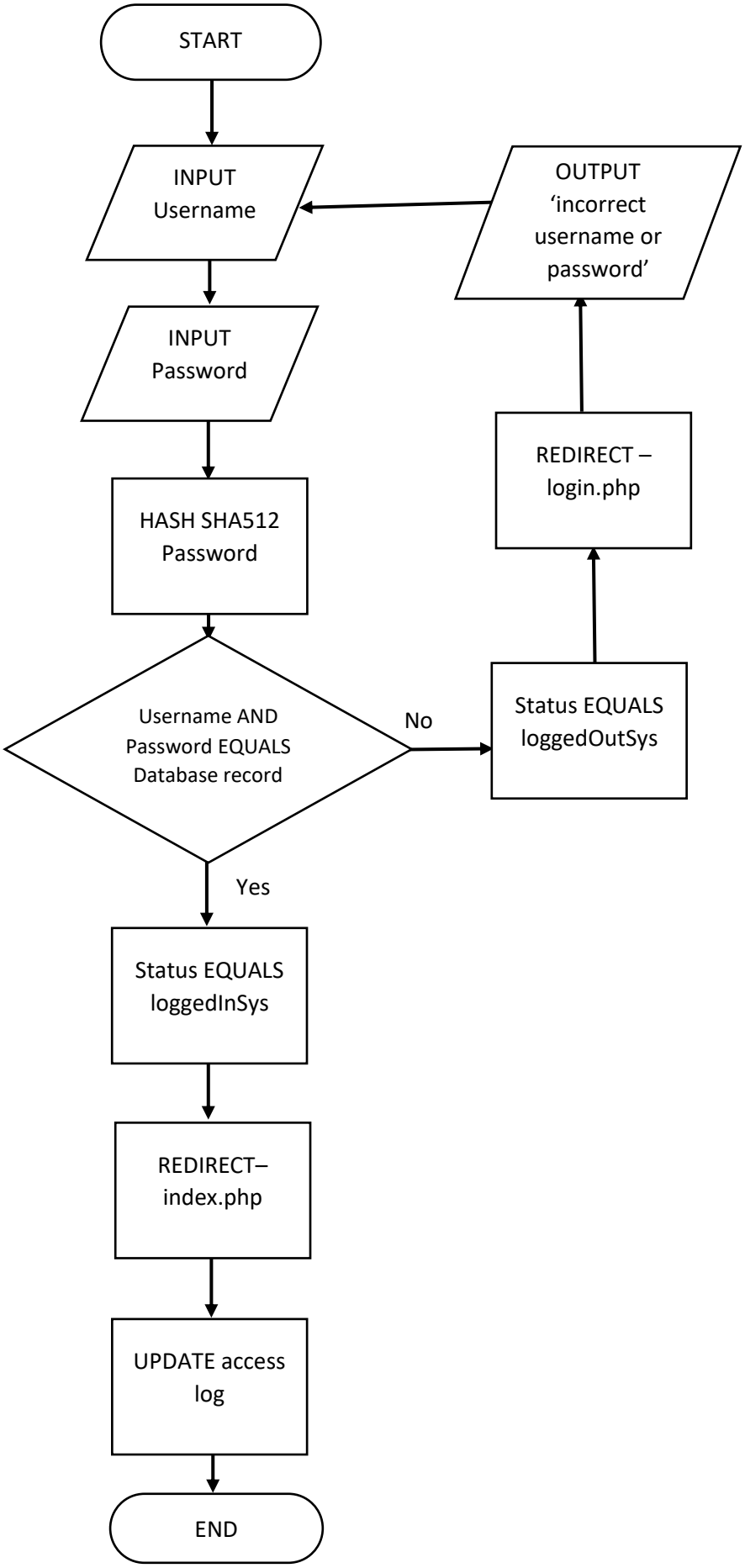
From the designs of the prototype, the solution allows users to access all the required features and requirements mentioned in the proposal of the solution. I have made it so that there is a consistent layout throughout with a sidebar that allows users to easily access different pages. Icons are also used to make identifying certain features easier, this ultimately adds to the ease of use of the system, so a user finds the system easy to navigate.

The design also uses certain standards and usability features that allow a user to easily use the system without prior knowledge of the system this is seen with the use of icons and easily identifiable function. The system will also use messages for users in an understandable format, this gives feedback and guidance to the user. Examples would include error messages containing the required data a user has to enter when they input incorrect data and success messages to confirm that an action has been complete for example on sign in. This allows users from different technical background to further understand system functions and wat each action the take has resulted in. Alongside this I will use icons to further demonstrate success and danger with universal symbols such as those below: *(source https://fontawesome.com/icons)*





Flowchart – Login



Flowchart – Login

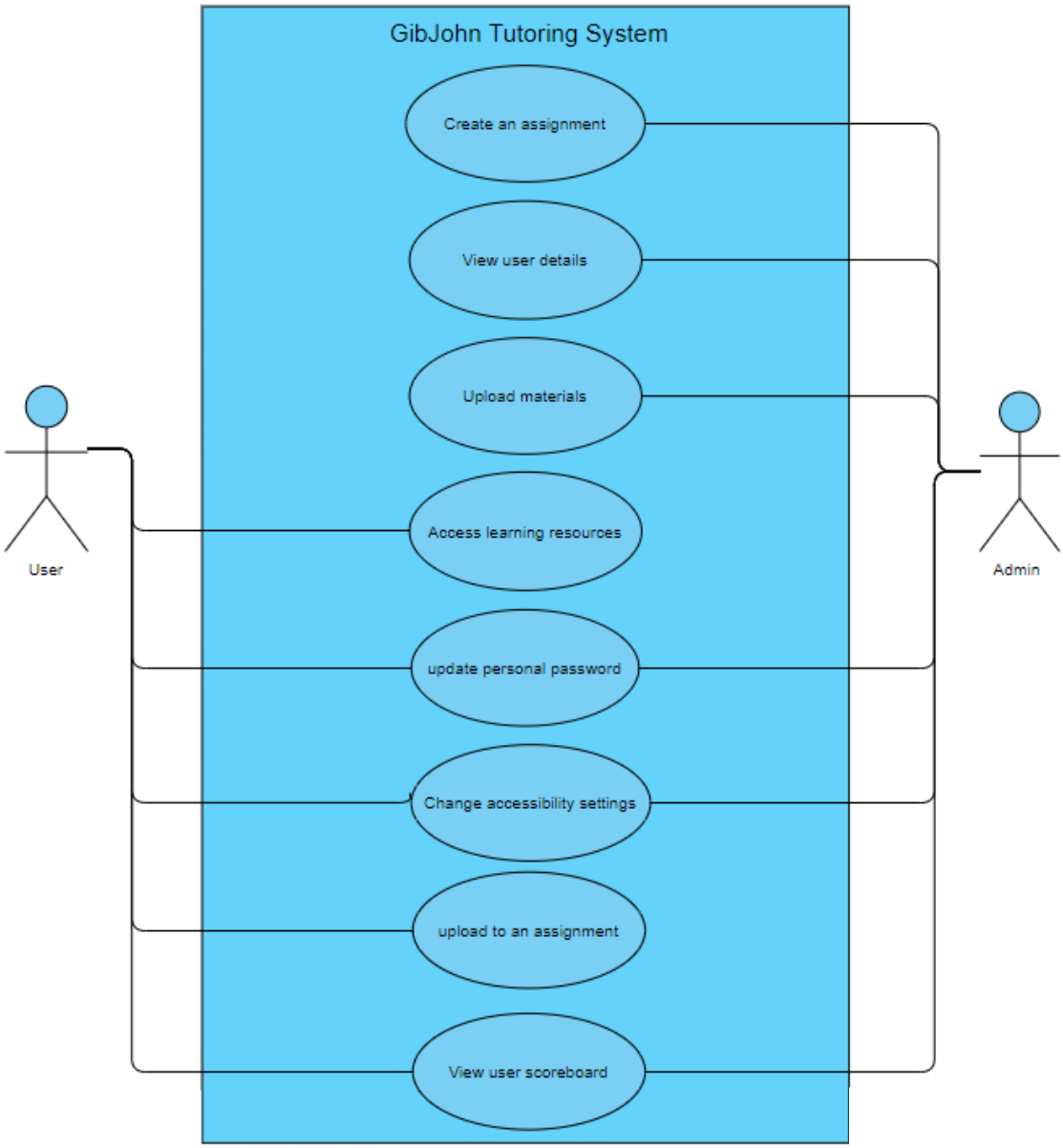
This flowchart shows how the user will login to the system. The system will take a username and password and compare what the user has entered to database records, if the user has entered the correct details the system will direct them to the homepage, if not then the user will be sent back to the login screen with an error message.

Justification

The login system is important to prevent unauthorised access to the system. This helps with access control as users and admins have different access rights. The passwords are encrypted using SHA512 to ensure they are stored securely.

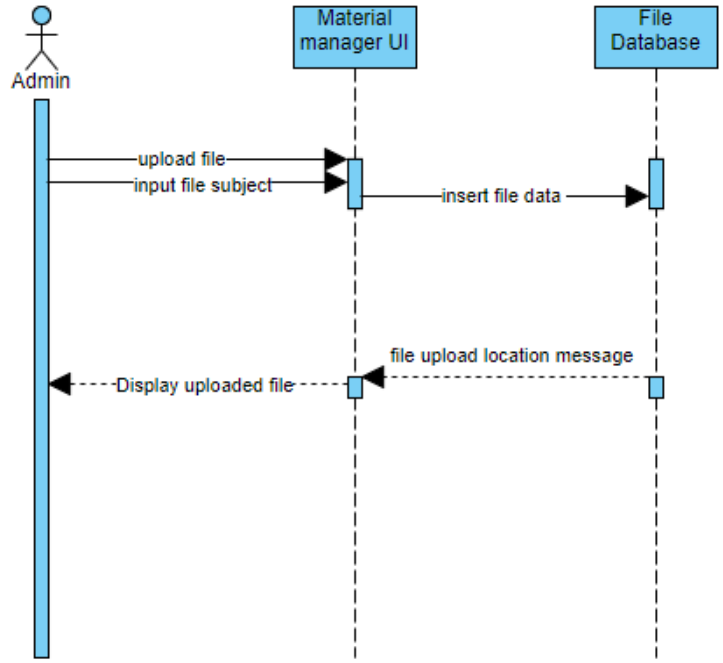
Use case diagram – GibJohn Tutoring

This is a use case diagram created for the system, this demonstrates the roles actors have with the system. This diagram shows the features and functionalities accessible to the different types of users that interact with the system. This is a visual way that further demonstrates the key parts of the system and how each user can interact, allowing a clear view to access restriction and consideration when developing the solution.



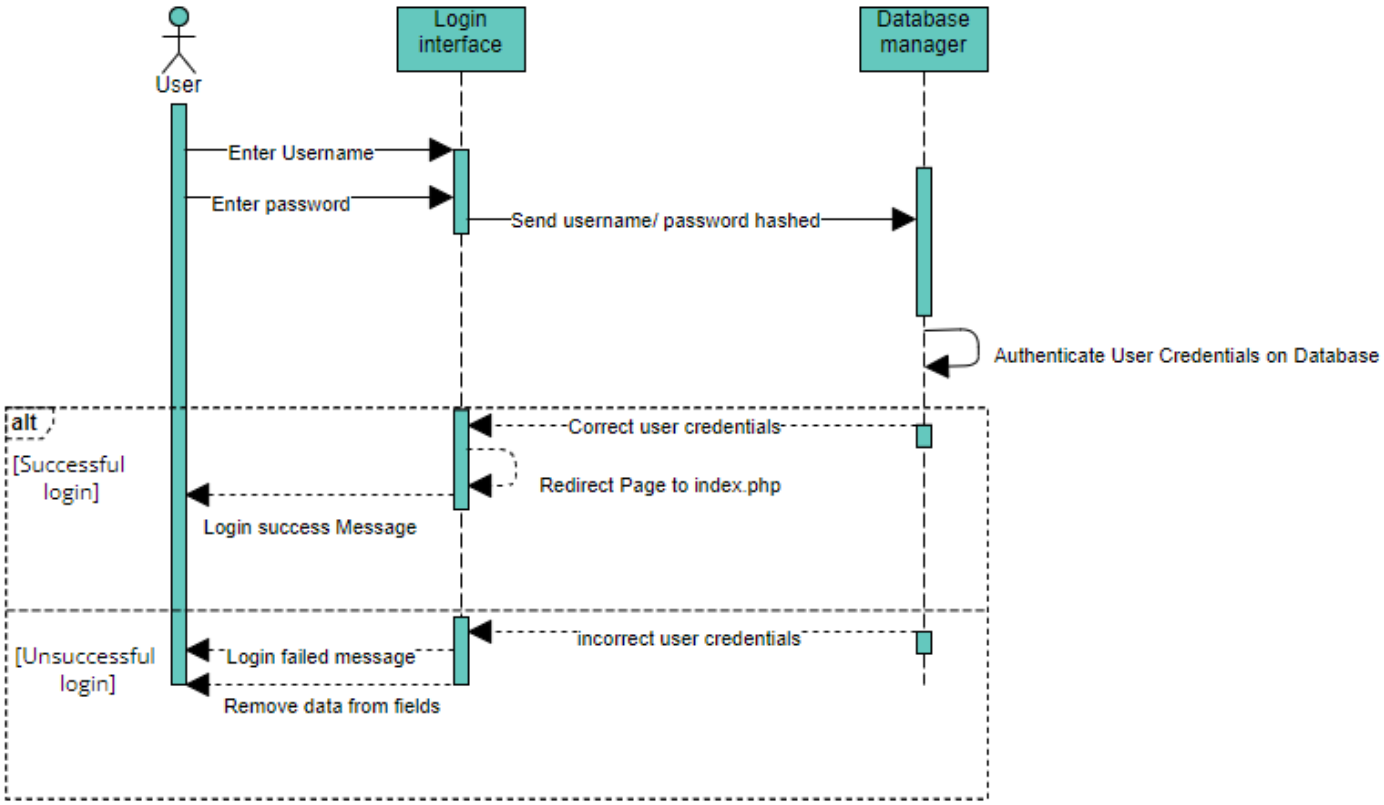
Sequence Diagram – File Database

Below is a sequence diagram I have created which shows the interaction between the user and the File database showing the flow of information and data in sequential order of the system.



Sequence diagram – Login System

Below is a diagram I have created which outlines the interaction between the user and the system when they are required to log into the system, there are two alternative scenarios for when the user have provided the correct and incorrect credentials to login.



Pseudocode – Authenticate user

```
// takes posted data from user login form
username = validateInputs (POST username)      // calls validate inputs for user entered username
password = validateInputs (POST password)      // calls validate inputs for user entered password
hashedPassword = HASH SHA512 (password)  // hashes password with PHP built in function

checkUser = checkCreds (username, hashedPassword)
IF checkUser EQUALS  'success' THEN
        SESSION status = 'userLoggedIn'          // status is logged in to access pages
        SESSION message = 'Login Successful'     // success message stored in session to be output to
the user
        REDIRECT location = index.php
ELSE
        SESSION status = 'loggedOut' // prevents users from accessing restricted webpages
        SESSION message = 'Username or Password Incorrect'
        REDIRECT location = login.php // Takes the user back to the login screen
ENDIF

// Function – validate and sanitises user inputs
FUNCTION validateInputs (data)
        HTML SPECIAL CHARS (data)      // built in PHP function
        TRIM (data)                    // built in PHP function
        return data
END FUNCTION

// Function – checks the credentials with username and hashed password against database entries
to determine if the user is authorised
FUNCTION checkCreds (username, hashedPassword)
        // Database credentials
        dbName      =  'localhost'
```

Validating and sanitising inputs within a solution is important to ensure the system is robust, this prevents users form entering data which can affect the system, an example would be an SQL style attack on the input fields.

```
        dbUser        = 'root
        dbPassword  = ' password'
        database      = users
        TRY
                PDO = connect to users with dbUser, dbPassword  // connect to database with creds
                SQL = 'SELECT name, email FROM users WHERE  userName = (username) AND password = (hashedPassword) '  // sql to retrieve data with matching username and password
                statement = EXECUTE connection WITH PDO SQL
        CATCH
                SESSION errorMessage = PDO connection error  // catches and saves error to session variable

        IF statement EQUALS 1 THEN     // when one entry is found
                return 'success'
        ELSE                           // no entry is found
                return 'failed'
END FUNCTION
```

The solution uses PDO and SQL to retrieve credentials from the database, these are checked against the POSTed inputs a user provides to check if they match, if they match a user is logged in and redirected to the homepage otherwise the user will not have access to the system and they will be redirected to the login screen. This is an important part of the system as it controls access to different functions and feature within the system.

When using the SQL, I have considered selecting the individual columns required compared to a * which selects all columns, this is important especially if scalability is considered as it would take more resources and time to retrieve the data required.

Pseudocode – Score Management

```
userId = POST id // get user id
score = POST score // get user score

// Retrieves database credentials (dbUser, dbName, dbPassword, database) from another file
```

```
INCLUDE 'database-connection.php'


updateUserScore = updateScore (userId, score, dbUser, dbName, dbPassword, database )
IF updateUserScore EQUALS  'failed' THEN          // Error handling - failed update, outputs error to the user
        SESSION errorMessage = 'Failed to update score'
        REDIRECT LOCATION index.php  // takes user back to previous screen
ELSE
        checkUserLevel = checkLevel(name)       // calls function to check the user level
ENDIF


// Function – update user score in the database
FUNCTION updateScore(userId, score, dbUser, dbName, dbPassword, database)
        TRY
                PDO = connect to users with dbUser, dbPassword
                SQL = 'UPDATE users SET userScore = (score) WHERE  id = (userId) ' // updates user score where  the user id matches, this is the primary key for each value
                statement = EXECUTE connection WITH PDO SQL  // Execute statement
        CATCH
                SESSION errorMesssage = PDO connection error // if fails, catches error and stores into a session variable
                return 'failed'


        IF statement EQUALS True THEN           // if successful returns success
                return 'success'
        ELSE                            // if fails to update return failed for error handling
                return 'failed'
END FUNCTION


// Function checks the  users level to output to the user
FUNCTION checkLevel (userId, dbUser, dbName, dbPassword, database)
```

```
        TRY // Try to connect to database
                PDO = connect to users with dbUser, dbPassword
                SQL = 'SELECT userScore FROM users WHERE  id = (userId) ' // gets the up to date score from the user
                connection = EXECUTE connection WITH PDO SQL
        CATCH
                OUTPUT PDO connection error
                return 'failed'
                EXIT

        score = connection // gets value from database and stores into score variable
        / / compares user score to find user level
        IF score < 100 THEN
                level = 0           // level 0 If score below 100
        ELIF score >= 100 AND score < 200 THEN
                level = 1 // level 1 if between 100-200
        ELIF score >= 200 AND score <400 THEN
                level = 2 // level 2 if between 200-400
        ELSE
                level = 3 // any larger score is level 3
        SESSION userLevel = level
END FUNCTION
```

This Pseudocode shows the logic for the score system, a user will receive a score based on an assignment or task, once this data is received it will be input to the database to update the current user score. The user level is then calculated based on the score of the user with predefined levels at certain scores.

The justification for retrieving the score from the database allows user to use multiple devices at once, if a user on one device scores points the database will record this and the score will be updated successfully. This ensures that the system allows multiple logins without the users score being affected.

Pseudocode – Check Logged in Status

```
// checks if user is logged in to access the page

PROCEDURE checkLoggedIn()

        IF SESSION status NOT EQUAL  'userLoggedIn' THEN       // if not logged in give error
        message

                SESSION errorMessage = 'You do not have access to this page'
                REDIRECT Location =  login.php  // redirect user to login screen to prevent access

                EXIT

        ELSE

                OUTPUT 'Welcome Back'          // Outputs to logged in user

                SESSION errorMessage = ''      // Clears error message

        END IF

END PROCEDURE
```

This procedure will be used on all webpages within the system that require a user to be logged in to view, on the user login a session variable will be created to contain if a user is logged, on each page this will be added to check the user has this variable, on a user logout the session variable is destroyed and a user will no longer have access to these pages.

When developing the solution this can be added to the code as a PHP include, this allows the code to be added to each page from one source, ensuring that the code is 'DRY' (Do not repeat yourself) making the solution more easily maintainable throughout.

Data requirements

Below are the key variables used throughout the solution to store data required for the main system processes to function.

Variable name	Data Type	Example data	Reason
(session) status	String	'userLoggedIn'	This variable is used to check the status of the user to access the system
$password	String [SHA512 Encrypted]	[SHA512 encrypted password]	Checks a user is using the correct details when signing in to access the system
$username	String	'nathan@gmail.com'	Used to store the user entered username to check if it is valid, this is the users email

$firstName	String	‘Nathan’	This is used to store the first name of a user when changing or outputting the users name
$lastName	String	‘Hannah’	This is used to store the last name of a user when changing or outputting the users name
$email	String	‘nathan@gmail.com’	This is used to store and the email of a user, this is used when outputting the users email from the database
$userId	Integer	3	Used when deleting or editing user information, the user ID stores the id of a user.
$dbPassword	String	‘password’	Stores the password for database access
$dbUsername	String	‘root’	Stores the username for database access
$database	String	‘customers’	Stores the specific database that is required
$sql	String	‘SELECT id, name, email FROM customers’	Stores the SQL statement for CRUD operations
(session) userLevel	Integer	2	Stores the user level in a session variable so that the user level can be displayed throughout the system to the user
$level	Integer	1	Stores the levels that can be reached by the user
$userScore	integer	75	Used to output the score from an assignment or task to the user
(session) errorMessage	String	‘Username or Password incorrect’	Stores an error message for the user which can be output
(session) successMessage	String	‘Record deleted successfully’	Stores the success message for successful operations a user takes
$count	Integer	10	Used for counting rows that are within the

			database to be output to the user such as assignments
$statement	Boolean	True	Used to store the reply from the PDO request

There are several considerations that have to be made when using data within a solution, it is important that measures are in place to ensure data adheres to CIA/AIC (Confidentiality, Integrity and Availability) and legislation, within the solution I aim to implement the following:

- Variable naming standards, within the solution it is important that files and variables follow the same standards, this allows better code maintainability. Within the solution I will be using camelCase for variable names this is one of many standards used for both PHP and JavaScript within industry. I will ensure that the naming conventions are consistent throughout with names that are relevant and descriptive.

- Information will be sent through POST methods, this method does not send data through the URL, which makes it more secure as it prevents malicious actors from clearly viewing data sent to and from the server.

- Built in functions such as htmlspecialcharacters() will be implemented on all inputs retrieved form the user to sanitise data passed to the server, this assists in preventing attacks such as SQL injection.

- Passwords stored, sent and received will be encrypted using hashing techniques and password salt, within the solution SHA512 will be used to implement this technique.

- Session variables are used throughout, they are used to a store temporary values in a secure way, these will be used within the solution to store variables that provide user information or user input settings such as user preferences. I have considered alternative storage such as the use of cookies, however session variables are server side and can be easily implemented into PHP while cookies are on the client side which can be easily modified.

Database Data requirements

These are the data requirements for the database used to store information and data that is required to perform CRUD (Create Read Update Delete) operations within the system.

User Details

This database table will be used to store information on the users of the system, this is personal information such as name and email therefore it is important that this data is kept secure an only each individual can access their own data.

Users	
Field Name	**Datatype**
User ID (PRIMARY)	AUTO INCREMENT (INT (20))
First name	VARCHAR(64)
Last name	VARCHAR(64)
email	VARCHAR(255)
Password	VARCHAR(255)
Level	INT(255)
Score	INT(255)

Admin Details

This table is used to store administrator details, these are details which must be kept secure as admins can access and edit all other user data.

Admins	
Field Name	**Datatype**
Admin ID (PRIMARY)	AUTO INCREMENT(INT (20))
First name	VARCHAR(64)
Last name	VARCHAR(64)
email	VARCHAR(255)
Password	VARCHAR(255)

Uploaded Material

This table holds information on the materials uploaded by admins onto the system, this could be different files and formats which provide users with learning materials.

Materials	
Field	Datatype
ID (PRIMARY)	AUTO INCREMENT(INT (20))
File name	VARCHAR(64)
File Type	VARCHAR(64)
Location	VARCHAR(255)
Uploaded by	VARCHAR(255)
Date	DATETIME(23)

Assignments

This database table will hold assignments uploaded by admins, this allows users to view assignments that are required to be completed.

Assignments	
Field	**Datatype**
ID (PRIMARY)	AUTO INCREMENT(INT (20))
Name	VARCHAR(64)
Description	VARCHAR(64)
Due date	VARCHAR(255)
Uploaded by	VARCHAR(255)
File location	VARCHAR(255)
Date	DATETIME(23)

Uploaded work

This table is used to store user uploaded work in response to assignments, a user will be able to upload new entries into this table.

User Uploaded	
Field	**Datatype**
ID (PRIMARY)	AUTO INCREMENT(INT (20))
Assignment ID	VARCHAR(64)

Uploaded by	VARCHAR(255)
File location	VARCHAR(255)
Date uploaded	DATETIME(23)

User Scoreboard

The scoreboard is used to track user progress, this can be accessed by both admins and users to see a user's score. A user will be able to view other users score to track where they are in comparison.

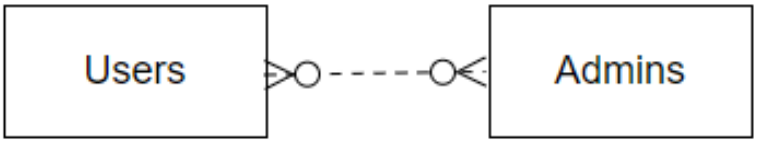
Scoreboard	
Field	**Datatype**
ID (PRIMARY)	AUTO INCREMENT(INT (20))
Username	VARCHAR(64)
Score	VARCHAR(255)
Level	VARCHAR(255)

ERD

One to many – One user to zero or many assignments

One to many – One Scoreboard to zero or many users

Many to many – zero or many users to zero or many admins



Approach to testing

Testing will be conducted throughout the development process to ensure the solution works in the way it is intended from the set requirements and user needs. The testing ensures that errors or bugs within the code are identified. When these are identified and resolved it ensure the code is overall more robust. Testing is also vital for ensuring security is implemented within the solution as vulnerabilities may be identified during this process that malicious actors could exploit.

There are different ways of testing the solution these include:

White box testing- This look sat testing the system and its functionalities whilst looking into how the internal of the system work and what functions and features are within the system, it can also look into more depth on how data is retrieved and stored.

Unit testing – This looks at testing a component of a system as its own unit, unit testing is done on key functionalities that are separated into their own unit to be tested, this allows developers to focus testing into one component of the solution at a time.

Integration testing – This type of test is done to ensure that all the separate components can work together, integration testing take the different units and tests the compatibility between them, this could look at thig such as passed variables and data types shared from one part of the system to another

System testing – this looks at testing the system as a whole, that it functions as it should with all the components of the system, typically this test is done once all the components have been developed to ensure the system works as intended to the requirements.

When developing the solution these tests will be used throughout to ensure the system meets the client's needs. These strategies particularly unit testing is important to be used throughout, this makes it so that the development processes more efficient in the way errors are identified, if testing was not done throughout it would be more challenging and take more time to identify issue which could have been easily identified and fixes with iterative testing methods.

Purpose

The purpose of testing the solution is to ensure that errors, data storing and retrieving and system functions are looked into in depth which will allow the system to become successful in terms of a solution that functions as it should per the requirements. As error are identified it allows the develop to produce fixes and changes to the code which allows the code to be more robust, the wider the range of tests included assist in identifying more errors that could occur, this is particularly important for data entry to test if a user would be able to break the system in some way.

User score test plan

This test plan shows the tests that should be carried out in the development stage for the user score system.

Index	Component to be tested	Test Data	Test Type	Expected Outcome	Prerequisites and dependencies
1	Update user score	Score = 60	Normal Valid	The score is updated on the database	The assignment has been submitted to be graded
2	Update user score	Score = -1	Invalid Extreme	The score is rejected as it is a negative value	
3	Update user score	Score = 10000000000	Invalid Extreme	The system should output that the maximum score is 100	
4	Scoreboard	‘nathan’ ‘100’	Normal Valid	The data should be output to the user	The data is stored and retrieved from the database

Login Test plan

Below shows the test intended to be carried out for the login system within the solution.

Index	Component to be tested	Test Data	Test Type	Expected Outcome	Prerequisites and dependencies
1	Check login status	$_SESSION status = ‘userLoggedIn’	Normal valid	The system should redirect the user to the homepage as they have already logged into the system	The user would have already logged into the system to get this status, therefore they are not required to log in again
2	Check login status	$_SESSION status = ‘’	Normal Valid	The system should allow the user to access only the login page as they are not logged in	The user has not already logged into the system
3	Login username input field	‘nathanhannah’	Normal Valid	This should be accepted by the system	The user should access the login page
4	Login username input field	‘ ’	Invalid erroneous	The system should request the user enter a valid username	
5	Login username Field	‘1234’	Invalid erroneous	The system should request the user enter a valid username	
6	Login password field	‘password’	Normal Valid	The system should accept this to be checked	
7	Login password field	‘ ’	Invalid Erroneous	The system should output a message	

				‘enter a password’	
8	Toggle password	[checkbox active]	Normal Valid	On checkbox the system show the users password in plain text	
9	Check login Credentials	‘nathan’ ‘letmein’	Normal valid	The system show accept these credentials as valid and redirect the user to the homepage	Both the username and password should be stored in the database to authenticate the user

Upload learning material test plan

These are the tests that should be conducted when testing and developing the file upload system for the solution.

Index	Component to be tested	Test Data	Test Type	Expected Outcome	Prerequisites and dependencies
1	File name input	‘material1’	Normal Valid	The system should accept this as a file name	
2	File name input	‘material1’	Invalid	The system should not accept this as there is already a file with that name	Test index 1 was carried out
3	File name input	‘ ’	Invalid erroneous	The system should output ‘enter a file name’	

4	File description input	‘This is learning material’	Normal Valid	The system should accept this as a file description	
5	File description input	‘ ’	Invalid erroneous	The system should reject this and output ‘enter a valid file description’	
6	File subject selection	[no selection]	Invalid erroneous	The system should output ‘please select the subject for this material’	
7	File subject selection	‘computer science’	Normal Valid	The system should accept this data	This is one of the subjects that is selectable
8	File upload	[no upload]	Invalid	Output ‘Please upload a file’	
9	File upload	Img/png/pdf/docx	Normal Valid	Upload the file to the database output ‘File upload successful’	All other data is entered and valid
10	File upload	.php / .js	Invalid	The system should reject this file type and output ‘You cannot upload this file type’	