URL of Application:

URL of GitHub Repository: <https://github.com/kawsar03/PCForum>

Username: username

Password: password

**PC Forum**

Outline: The application I built is a PC based forum. In this application we are able to view forum posts from others relating to the two forums: software and hardware. These two forums have a Title and Description where users enter their issues. You are also able to add your own issue to the application and see it displayed. You are able to Register, Login and Log out. You sign in using an email and it also asks for basic information such as First and Last name and also require a password. You are able to check out a list of free games which I have included from an API. There is also a List Users pages that shows a list of users within the database and also there is a delete user page in which you can delete a user from the database. On each page there is a search bar on the top left corner which can enable you to search for items within the forums and see if there is anything that matches. There is also a floating drop-down menu just below to redirect you.

**Architecture Diagram**

|  |  |  |
| --- | --- | --- |
| Presentation Tier (Front End) | Application Tier (Middleware) | Data Tier (Back end) |
| HTML, CSS and JavaScript | Node JS, Express, EJS, Sanitise, bcrypt, HTTPS | MySQL, this is used to create a database. API |
| HTML was used to in this application to help create the client faced program. HTML was used to build the majority of the structure of this web application. CSS was primarily used for styling this web application and also helped with navigation in the form of the drop-down menu. It was used to import backgrounds and style the way everything is laid out. JavaScript was used to manipulate data from databases and the API for the front end of the project. This can be seen especially for the API as it was raw data which we transformed to be displayed on the List of Games page. | These are required to communicate/ interact with the client/front end side via HTTP. These modules are required for certain security features to work. For example, I have used Sanitise to ensure scripts cannot overrun my application. BCrypt is also used to encrypt passwords to ensure they do not get hacked. This is part of password hashing. I used HTTPS to connect the API to my application. | Middleware can access the databases to actively store and use information. MySQL was used to create and use the database in which we store the forums and log in details. We also used an API which unlike the SQL database was retrieved from online. |

**User Facing Functionality**

In my application, we are straight away greeted with the Index page where we are able to branch into many different pages. Users can navigate through varied pages but will likely go to the forums either software or hardware. However, if they are not signed in, when trying to go into another webpage they are promptly redirected to the log in page. This is shown in the screenshot below via redirectLogin.

A computer screen with colorful text

Description automatically generated with medium confidence

Once inside a Forum page they are able to view posts and also add a post themselves by clicking the link at the bottom that will take them to another page called addsoftware or addhardware issue. By placing this page within the view forum page, the redirect feature will ensure that anyone trying to post on the forum is logged into an account. As well as this, on all pages on the top right corner of every page there is a search bar which users can use to search both databases for either forum or see if there is anything relating to what they need.

A computer screen shot of a program code

Description automatically generated

The screenshot above is taken from main.js and shows my get and post functions relating to the search bar. We see that it simultaneously searches both databases will return your results. This uses the UNION and LIKE features from SQL.

Another page they are able to navigate to is the Free Games page which you are able to locate from the Index or the drop-down menu. Once clicking this page, you are taken to a display of a list of games that have been imported from an API stating the name, genre and platform. This database was taken directly from an API and transformed as shown in the screenshot below.

A computer screen with text

Description automatically generated with medium confidence

As shown in the screenshot we took the API data and used the forEach method to iterate over the database and manipulate the data to be displayed in a certain way with the thumbnail, title, genre and platform. By using Bootstrap classes like col-md-4 I have ensured that each card takes up a certain, uniform amount of space as the rest.

Moreover, there is also a remove user page in which you are able to enter a username and remove it from the database. This is achieved by the SQL code shown below.

A screen shot of a computer code

Description automatically generated

We also see here each input field is sanitised to improve the applications security. This code deletes the row entirely from the userdetails table in the SQL database myPCForum.

Furthermore, there is also a navigation bar to help users move from one page to another. In the top right corner below any search bar there is a button called menu. Once hovering over this button, it displays a menu of other web pages to explore. The advantage of having a hover feature is that it does not take too much space up on the web page and is appealing for users and improves user experience.

**Security**

Security is paramount within this application, and various measures have been implemented to safeguard sensitive information against potential security threats. One crucial security measure implemented is password hashing, which encrypts user passwords, rendering them irreversible even if intercepted. This robust encryption ensures that hackers would face significant challenges attempting to revert hashed passwords back to their original form. This not only enhances the overall security of the system but also instils confidence in users, assuring them that their passwords are securely stored, in stark contrast to the vulnerabilities associated with storing passwords in plaintext.

Additionally, a proactive approach to input validation and sanitization has been adopted through the use of **req.sanitize** for input fields. This provides a robust defence against Cross-Site Scripting (XSS) attacks and SQL injection. By applying this measure, users are prevented from injecting malicious scripts into input fields, significantly reducing the risk of security breaches. Furthermore, the implementation of the **validator** library adds an extra layer of validation to input fields, ensuring that the data entered adheres to predefined standards.

For instance, minimum length requirements have been set to discourage empty input fields, and specific criteria, such as the inclusion of the '@' symbol and '.com' in email addresses, are enforced. Password complexity is also reinforced by requiring a minimum length of 8 characters and at least one numeric digit. This stringent approach not only strengthens the overall security posture but also promotes the creation of robust and less vulnerable user passwords.

Moreover, the application incorporates redirect functionality to mandate user authentication before attempting to post content on the forums. This strategic measure mitigates the risk of common trolls entering the web application and spamming forum pages with undesirable content. By requiring users to log in first, the application creates a secure environment and fosters a positive user experience for genuine users.

**API**

My API is straightforward to use as you just browse the data given on the web page that has been neatly lined up in a card format with photos. My API was taken from this website: https://rapidapi.com/digiwalls/api/free-to-play-games-database/details

Hardware

* ID (Primary Key)
* Title
* Issue

**Data Model – ER Diagram**



A long black line on a white background

Description automatically generated



Software

* ID (Primary Key)
* Title
* Issue

User Details

* username (Primary Key)
* first\_name
* last\_name
* email
* hashedPassword

The ER diagram above shows the relationship between the forums and the user database. There is a one-to-many relationship between the two forums and userdetails as one user can create many posts due to the redirect feature forcing you to be logged in to use the forums. There Is a many to many relationships between the forums considering they can share Title and Issue between the two.

**Special Techniques**

I embedded a YouTube video into my forum pages to help fix some common issues users may face. This can help stop forums being repetitively spammed with the same issue from different users. Moreover, I used iteration on my API database that I imported and formatted the data to look appealing for my users. When initially imported the API appeared as a block of text. As well as this, I used both Validator and HTML to ensure each entry into the register and log in pages are valid and contain certain grammar and adhere to the correct requirements.