Title – Accessibility

Context and Problem Statement – a user who is dyslexic, how might we make the system easier to use for those

Considered Options – use more diagrams and pictures rather than using a lot of words

Decision Outcome -

Consequences – for the users who do not have any issues with regards to accessibility will be lacking information from the system and see diagrams which may not be of use to them

Confirmation  
 Pros and Cons of the Options - One is better for the user who has accessibility issues and may not be beneficial for the user who will see a lot of diagrams rather they may want more info on the system.

Title - Security

Context and problem statement – how might we make the system secure with users who will be inputting their personal details and any payment details.

Considered options – one of the options for the payment details to be secure is to use an external payment system vendor. Another option is in the backend of the datbase, the admin can only see #### rather then them be able to see the details directly.

Decision outcome – when the users login, they will also need to do a 2-step authentication. The payment system that will be used is by an external vendor so that it also falls into line with GDPR and that the library will not be held responsible.

Consequences -

Confirmation

Pros and Cons of the Options – the positive for the extra authentication is so that the custmers details are more secure and thr account is less likely to be hacked. The negative for using an external vendor for payment systems, is say if a customer was overcharged or wanted a refund, it would take longer to get any of these things processed.

Title: Choosing MySQL over firebase

Context and Problem Statement - A key consideration is ensuring that sensitive user information, such as personal details and payment information, is securely managed. Users will add their personal details (name, address, etc.) and payment details during the checkout process. This will help us in following GDPR rules.

Considered Options

1. MySQL  
   MySQL is a relational database that supports structured storage of user data, including personal and payment details. It offers robust customization, with advanced access control and encryption options to secure sensitive data. MySQL can be hosted on-premise or with trusted cloud providers, giving full control over security settings.
2. Firebase  
   Firebase is a cloud-based NoSQL database known for real-time syncing and scalability. It's easy to set up and manage, with built-in features like user authentication and data storage. However, Firebase offers less control over security configurations, especially when handling sensitive payment information.

Decision Outcome - After evaluating the options, the decision is to implement a MySQL database into our system.

Consequences

1. Limited Control Over Security: Without Firebase, there's more control over encryption, access permissions, and database security configurations, which is crucial for protecting sensitive data like payment details.
2. Complex Data Relationships: MySQL's relational model better handles complex data relationships as an example users, books, transactions whereas Firebase's NoSQL model can be less efficient for such structured queries.

Confirmation

This decision was confirmed after discussions with the group members and deciding to go with MySQL.

Pros and Cons of the Options

MySQL

* Pro: MySQL provides advanced security features such as encryption, fine-grained access control, and the ability to fully manage security protocols.
* Con: MySQL requires more effort to configure, scale, and maintain, especially as the system grows.

Firebase

* Pro: Firebase is simple to set up and scales automatically, making it ideal for rapid development and handling large amounts of real-time data.
* Con: It offers less control over security configurations, which can be a concern when managing sensitive data like payment details.

Title: Framework Selection

Context and Problem Statement

We needed to select a framework for the library management system. The framework must be easy to use and secure so that it could be implemented and adapeted to the library management system needs.

Considered Options

1. CodeIgniter  
   CodeIgniter is a fast, lightweight PHP framework known for simplicity and speed. It provides essential features like database handling, routing, and security tools, and is ideal for smaller projects.
2. Laravel  
   Laravel is a more feature-rich PHP framework offering tools for advanced tasks like authentication, routing, and database management. It’s ideal for large projects but can be heavier and more complex than needed for this system.

Decision Outcome

CodeIgniter was selected for its simplicity, speed, and ease of use.

Consequences

* Performance: Its lightweight nature ensures the system will run efficiently without unnecessary complexity.
* Limited Features: While it’s simple, CodeIgniter has fewer built-in features compared to Laravel, which may require custom solutions for some complex tasks.

Confirmation

This decision was made based on the need for simplicity, quick development, and efficient performance.

Pros and Cons of the Options

CodeIgniter:

* Pro: Fast setup and development with essential security features.
* Con: Fewer built-in features, which might require extra work for complex needs.

Laravel:

* Pro: Rich in features, ideal for larger, more complex applications.
* Con: Can be too heavy for a simple library system and harder to maintain.

Certainly! Here’s a simple and straightforward ADR for **hashing passwords**:

**ADR: Password Hashing**

**Title: Password Storage**

**Context and Problem Statement**

The system needs to securely store user passwords. Storing passwords in plain text poses a significant security risk. We need to ensure that passwords are protected and cannot be easily accessible.

**Considered Options**

1. **Hashing Passwords**  
   Using a strong cryptographic hashing algorithm
2. **Storing Plain Text Passwords**  
   Storing passwords as plain text in the database

**Decision Outcome**

We decided to **hash passwords** This approach ensures that even if the database is compromised, the passwords cannot be directly retrieved.

**Consequences**

* **Security**: Passwords will be securely stored, making it much harder for attackers to retrieve them even if they gain access to the database.
* **Implementation**: We'll need to implement a password hashing mechanism and ensure all new and existing passwords are hashed properly.

**Pros and Cons of the Options**

**Hashing Passwords:**

* **Pro:** **Enhanced Security** – Passwords are stored in a hashed format, making it much harder for attackers to retrieve the original passwords, even if the database is compromised.
* **Con:** **Performance Overhead** – Hashing algorithms like bcrypt add some computational overhead, which may slightly affect performance during login, though it’s a necessary trade-off for security.

**Storing Plain Text Passwords:**

* **Pro:** **Fast Access** – No additional processing is needed to check passwords, leading to faster login times.
* **Con:** **Severe Security Risk** – Storing passwords in plain text makes them highly vulnerable to theft if the database is compromised.