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 - The system will incorporate more visual aids (e.g., diagrams, infographics, icons) to replace or supplement text-heavy content, making it easier for dyslexic users to naviate and understand the system.

Consequences

* **Positive Consequences:** Dyslexic users or users with reading challenges will find it easier to comprehend the information. They can rely on visuals to guide them through the system, leading to a more inclusive and user-friendly experience for this group.
* **Negative Consequences:** Users without any accessibility needs may find the increase in diagrams distracting. For these users, the visuals may not be helpful and could create confusion, especially if they prefer more detailed, text-based information.

Confirmation: The decision to use more diagrams and pictures is aimed at improving accessibility for users with dyslexia, but it needs to be balanced carefully to avoid alienating other users who prefer detailed text-based information

Pros and Cons of the Options –

Pros:

**Improved Accessibility for Dyslexic Users:** Visual aids like diagrams and pictures are generally easier to process for dyslexic users compared to large blocks of text. This reduces cognitive load and improves their ability to navigate and use the system effectively.

**Broader Inclusivity:** Helps ensure the system is more accessible to a wider range of users, not just those with dyslexia, but potentially other cognitive disabilities as well.

Cons:

**Potential Confusion for Non-Dyslexic Users:** Users who do not need accessibility features may find the increase in diagrams overwhelming or unnecessary. They may prefer detailed, textual explanations and could find the visual-heavy content insufficient.

**Dilution of Information:** Important textual content might be overlooked or reduced in favor of visuals, leading to the loss of some nuanced or detailed information for users who prefer text.

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

**Option 1:** Use an external payment service provider to process payments securely. This approach ensures that payment information is handled by a third party with specialized security measures, reducing the risk of data breaches on the library’s end.

* **Option 2:** Implement a system where, in the backend database, administrators can only view masked payment details (e.g., ####) instead of the actual payment information. This approach helps limit the exposure of sensitive data to internal users.

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 – Although an external payment vendor provides strong security, it can introduce delays in processing customer requests, such as refunds or disputes. Customers may experience longer wait times in these situations.

Confirmation: This decision to implement enhanced security measures by using a external vendor

Pros and Cons of the Options –

**Pros:**

1. **Improved Security for Users:** The addition of two-step authentication significantly reduces the risk of account breaches, protecting user information from unauthorized access.
2. **Compliance with GDPR:** Using an external payment provider ensures that sensitive financial data is handled by specialists, aligning with GDPR regulations and relieving the library from direct responsibility for payment security.

**Cons:**

1. **Dependency on External Vendors:** Relying on a third-party payment provider may introduce delays in resolving customer service issues, such as refunds or payment disputes, potentially leading to slower response times for users.
2. **User Inconvenience:** The two-step authentication process, while improving security, may add an extra step for users during login, potentially causing slight inconvenience for those who prioritize ease of access.

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.

**Title**: Using MVC

**Context and Problem Statement**:  
The Library Management System (LMS) needs a clear structure to manage users, books, and loans. The challenge is to organize the code in a way that makes it easy to maintain and scale as new features are added.

**Considered Options**:

1. **MVC (Model-View-Controller)**  
   The MVC design separates the system into three components:
   * **Model**: Handles data
   * **View**: Displays information to users
   * **Controller**: Manages user input and updates the Model/View. This separation makes the system easier to maintain and scale.
2. **Monolithic Architecture**  
   All components are in a single codebase. It’s simple to set up but harder to maintain as the system grows.

**Decision Outcome**:  
We decided to use **MVC** because it provides a clear structure that makes the system easier to maintain and expand.

**Consequences**:

* **Better Testing**: We can test each part (Model, View, Controller) separately.
* **Scalability**: MVC will allow us to add new features without affecting other parts of the system.

**Confirmation**:  
The team agreed that MVC is the best option for organizing the LMS, ensuring long-term maintainability.

**Pros and Cons**:

* **MVC**
  + **Pro**: Clear structure, easier to maintain and scale.
  + **Con**: Initial setup is more complex.
* **Monolithic**
  + **Pro**: Simple to start with.
  + **Con**: Hard to scale and maintain as the system grows.