



POLITECHNIKA POZNAŃSKA

DS5img

Edgar Benito Martín

Rosario Pavone

January 2025 / Technical Report

Table of Contents

| | | |
|-----------|------------------------------------|-----------|
| 1 | REPORT DATA | 2 |
| 2 | CHANGE CONTROL | 3 |
| 3 | DESCRIPTION | 4 |
| 4 | REQUIREMENTS | 4 |
| 5 | DEPENDENCIES AND TECH STACK | 7 |
| 6 | DATABASE STRUCTURE | 7 |
| 6.1 | Users table | 7 |
| 6.2 | Posts | 8 |
| 6.3 | Comments | 9 |
| 6.4 | Ratings | 10 |
| 6.5 | Categories | 10 |
| 7 | USE CASES | 11 |
| 12 | | |
| 8 | MOCKUPS | 13 |
| 9 | SECURITY ISSUES | 14 |
| 10 | PLANED DIVISION | 15 |



1 Report Data

Date: 16/12/2024

Destination Entity: Poznań University of Technology - Faculty of
Computing and Telecommunications

Title: PUTimg

Lecturer: Michał Apolinarski, Ph.D.

Author: Edgar Benito Martín
Rosario Pavone

2 Change Control

| Document Status | | | |
|------------------|------------------|--------------|-------------|
| Document Version | Publication date | Changes made | Affected By |
| 2.0 | 29/11/2024 | 19/01/2025 | - |

3 Description

This document outlines the design and functionality of an image application. The application features a login/register module, which ensures secure user authentication and provides access control to restricted areas within the platform. Registered users are granted the ability to upload posts and rate them. Additionally, the application offers functionality for non-registered users, allowing them to view posts on a public page, search for posts using various filters, and comment on the posts.

4 Requirements

1. Register

- **User Story:** As a visitor, I want to register an account so I can access features available only to registered users.
- **Acceptance Criteria:**
 - User can input email, username, and password.
 - User receives a success message upon successful registration.
 - User is prompted to verify their email.

2. Login

- **User Story:** As a registered user, I want to log into my account so I can access my profile and perform actions like uploading images and commenting.
- **Acceptance Criteria:**
 - User can log in using their email and password.
 - Error messages are shown for invalid credentials.
 - User is redirected to their profile/dashboard upon successful login.

3. Email Verification

- **User Story:** As a newly registered user, I want to verify my email address so I can ensure account security and activate my account.
- **Acceptance Criteria:**
 - An email with a verification link is sent after registration.
 - Clicking the verification link activates the account.
 - An error message is displayed for invalid or expired verification links.

4. Reset Password/Forgot Password

- **User Story:** As a user, I want to reset my password if I forget it so I can regain access to my account.
- **Acceptance Criteria:**
 - User can request a password reset by entering their registered email.
 - A password reset email with a link is sent to the user.
 - User can set a new password via the reset link.

- An error message is shown for expired or invalid reset links.

5. Logout

- **User Story:** As a user with an active session, I want to close my session.
- **Acceptance Criteria:**
 - User can log out from different pages.
 - The user is redirected to the main page as a visitor.

6. Navigate the Main Page

- **User Story 1:** As a visitor, I want to see the latest posts for discovery.
- **User Story 2:** As a user, I want to see the latest posts and rate them.
- **Acceptance Criteria:**
 - The latest posts of the application are displayed (showing title and description).
 - Registered users can rate the posts with a "like".
 - An option to view more details of the post is provided.

7. View Post Details

- **User Story:** As a visitor or registered user, I want to see more details of the post, such as the number of likes, comments, and category.
- **Acceptance Criteria:**
 - Detailed information for each post is displayed.
 - Additional options for interacting with the post are available.
 - A comment section is provided for users to engage.

8. View Posts from the Same Category

- **User Story:** As a visitor or registered user, I want to see all posts from the same category.
- **Acceptance Criteria:**
 - All posts from the same category are displayed.
 - An option to view more details of each post is provided.

9. Search for Images

- **User Story:** As a user or visitor, I want to search for images by keywords so I can find specific content easily.
- **Acceptance Criteria:**
 - Users and visitors can search by entering keywords.
 - Results include images that match the title or description.
 - Search results are displayed correctly.

10. Advanced Search for Images

- **User Story:** As a user or visitor, I want to search for images using various filters like date, category, or author.
- **Acceptance Criteria:**

- Users and visitors can fill out search filters, allowing for combined searches.
- Posts that match the search criteria are displayed.

11. Upload Images

- **User Story:** As a registered user, I want to upload images with a title, description, category, and date so I can share them with others.
- **Acceptance Criteria:**
 - Registered users can upload images along with metadata (title, description, category, and date).
 - Users can create a new category if the selected one does not exist.

12. Comment on Images

- **User Story:** As a user, I want to comment on images so I can engage with other users' content.
- **Acceptance Criteria:**
 - Registered users can comment on posts, and their comments are visible to other users.
 - Visitors can comment anonymously, and their comments are displayed as such.
 - Each comment displays the username, comment text, and timestamp.

13. Rate Images (Like)

- **User Story:** As a registered user, I want to like images to express my opinion about them.
- **Acceptance Criteria:**
 - Only registered users can rate images.
 - The total number of likes is displayed for each image.

14. Profile Management

- **User Story:** As a registered user, I want to edit my profile.
- **Acceptance Criteria:**
 - Profiles display uploaded images.
 - Users can update account details (e.g., password).

• 15. View Received Comments

- **User Story:** As a registered user, I want to view all the comments I have posted.
- **Acceptance Criteria:**
 - The system retrieves and displays all comments posted by the user.
 - All the comments of the user are listed.
 - A link to the detail view of the commented post is provided.

16. View Received Ratings

- **User Story:** As a registered user, I want to see all the likes I have received and from whom.
- **Acceptance Criteria:**
 - The system retrieves and displays all ratings received by the user's posts.
 - For each rating, the user who rated and the post they rated are displayed.

5 Dependencies and Tech Stack

- **Frontend Dependencies:**
 - **HTML/CSS:** Used for building a dynamic and intuitive, component-based user interface.
- **Backend Dependencies:**
 - **Java:** Utilized for server-side routing and handling HTTP requests.
 - **Bcrypt.js:** Employed for securely hashing and salting passwords prior to storage
 - **JSON Web Token (JWT):** Used for generating and managing secure authentication tokens.
 - **Http Sessions:** Utilized for managing user sessions and maintaining state across requests.
- **Database Dependencies:**
 - **MySQL:** A relational database used for structured data storage.

6 Database Structure

6.1 Users table

The Users Table stores essential user information, including credentials, email, account status, and token data for account activation or password reset. It ensures secure authentication and account management by utilizing hashed passwords and email verification tokens.

| Column | Data Type | Description |
|----------|--------------|--|
| id | INT | Primary Key, unique identifier for each user. |
| username | VARCHAR(50) | Unique username chosen by the user. |
| password | VARCHAR(100) | Securely stored password, likely hashed using algorithms such as bcrypt. |

| Column | Data Type | Description |
|------------------|-----------------------------|--|
| email | VARCHAR(100) | Unique email address for user communication and notifications. |
| creation_date | TIMESTAMP | Timestamp indicating when the user account was created. |
| last_modified | TIMESTAMP | Timestamp indicating the last modification to the user's account. |
| status | TINYINT(1) | Account status: 0 for inactive, 1 for active. |
| token | VARCHAR(255) | Token used for email verification or password reset. |
| token_type | ENUM('activation', 'reset') | Defines the token type: <code>activation</code> for account verification, <code>reset</code> for password reset. |
| token_expires_at | DATETIME | Date and time when the token expires. |

Processing:

- When a user registers, the password field is securely hashed before storage.
- During account activation or password reset, the token field stores a unique token generated for that specific action.
- The token_type distinguishes between activation tokens and reset tokens.
- The status field helps track if a user's account is active (1) or inactive (0).
- The creation_date and last_modified fields help keep track of the user account's lifecycle, while the last_modified timestamp updates automatically whenever the account is modified.
- The unique_email and unique_username constraints ensure that both the email and the username are unique within the system.

6.2 Posts

The Posts Table stores the details of the posts shared by registered users, including the post's content, author, category, and associated photo. It tracks user interactions such as the number of likes and supports the relationship with categories and users through foreign keys.

| Column | Data Type | Description |
|-----------|--------------|---|
| post_id | INT | Primary Key, unique identifier for each post. |
| title | VARCHAR(255) | Title of the post, which provides a brief description of the post's content. |
| content | TEXT | Content or body of the post, which can be any textual information. |
| author_id | INT | Foreign Key referencing <code>users(id)</code> , indicates the user who created the post. |

| Column | Data Type | Description |
|-------------|--------------|---|
| category_id | INT | Foreign Key referencing categories (category_id), defines the category of the post. |
| created_at | TIMESTAMP | Timestamp indicating when the post was created. |
| photo_path | VARCHAR(255) | Path to the uploaded photo associated with the post, if any. |
| likes | INT | Stores the number of likes the post has received, defaulting to 0. |

Processing:

- The author_id references the user who created the post, linking the post to the user table and ensuring the post is associated with a valid user.
- The category_id references the categories table, allowing posts to be organized into different categories.
- The created_at field automatically tracks the time when the post is created, helping in sorting or filtering posts by date.
- The photo_path stores the file path of an uploaded photo related to the post. If no photo is uploaded, this field can be NULL.
- The likes field tracks user engagement with the post and is incremented as users interact with it.
- Foreign key constraints ensure referential integrity between posts, users, and categories.

6.3 Comments

The Comments Table stores user comments on posts, including the content of the comment, the author's name, and the post being commented on. It supports user interaction with posts and helps in tracking the creation time of each comment.

| Column | Data Type | Description |
|-------------|--------------|---|
| comment_id | INT | Primary Key, unique identifier for each comment. |
| post_id | INT | Foreign Key referencing posts (post_id), links the comment to a specific post. |
| author_name | VARCHAR(100) | Name of the author of the comment (could be a registered user or an anonymous commenter). |
| content | TEXT | Content or body of the comment, which is the text input by the user. |
| created_at | TIMESTAMP | Timestamp indicating when the comment was created. |

Processing:

- The post_id references the posts(post_id) table, ensuring that comments are linked to a specific post.
- The author_name field captures the identity of the person commenting, which could be either the username of a registered user or a generic identifier for anonymous users.
- The content field holds the actual text of the comment submitted by the user.

- The `created_at` field automatically records the time when the comment is made, which is important for sorting or filtering comments by date.
- The foreign key constraint ensures referential integrity by preventing the deletion of posts that have associated comments.

6.4 Ratings

The Ratings Table stores user ratings for posts. Each rating is associated with a user and a post. Ratings are constrained to be between 1 and 5, and relevant foreign key relationships ensure data integrity with the posts and users tables.

| Column | Data Type | Description |
|----------------------------|--------------|---|
| <code>rating_id</code> | INT | Primary Key, unique identifier for each rating. |
| <code>rating</code> | INT | Rating score, constrained between 1 and 5. |
| <code>created_at</code> | TIMESTAMP | Timestamp indicating when the rating was created. |
| <code>post_owner_id</code> | INT | Foreign Key referencing the author ID of the post being rated (from <code>posts</code>). |
| <code>username</code> | VARCHAR(100) | Foreign Key referencing the username of the user who rated (from <code>users</code>). |

Processing:

- When a user rates a post, the rating value is stored in the table, associated with the `post_id` and the `username`.
- The `created_at` field is automatically set to the current timestamp, indicating when the rating was made.
- The rating field is validated to be between 1 and 5 using the `ratings_chk_1` constraint.
- Foreign key constraints ensure that:
 - A post's author ID exists in the posts table (through `post_owner_id`).
 - A user's username exists in the users table (through `username`).

6.5 Categories

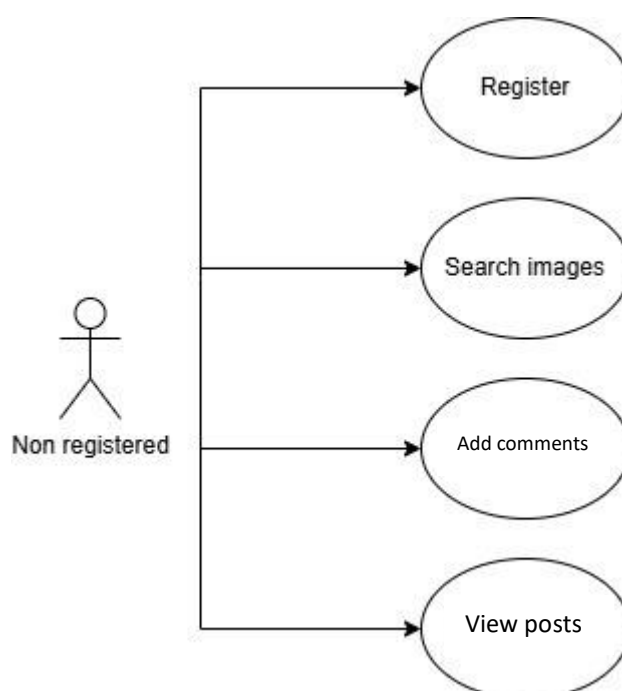
The Categories Table stores the available categories for posts. It helps in organizing posts and allows users to filter content by category.

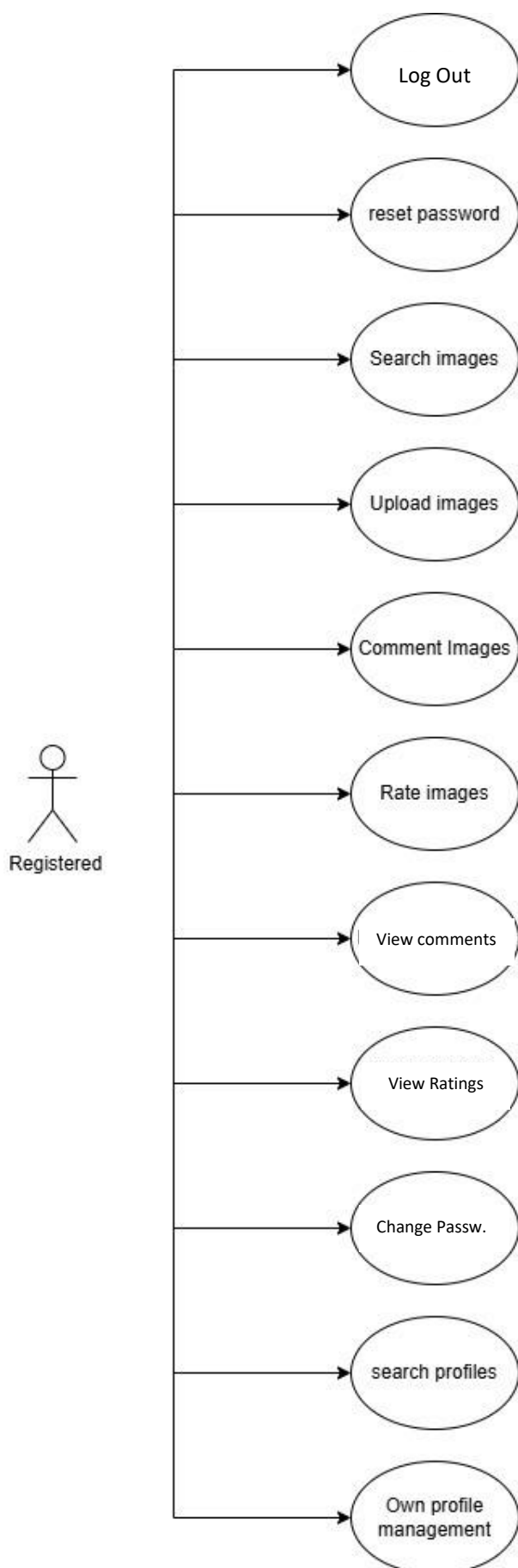
| Column | Data Type | Description |
|--------------------------|--------------|--|
| <code>category_id</code> | INT | Primary Key, unique identifier for each category. |
| <code>name</code> | VARCHAR(100) | Unique name of the category. Ensures that each category has a distinct name. |

Processing:

- The category_id serves as the unique identifier for each category and is automatically incremented with each new category added.
- The name field stores the category name, which is unique. The UNIQUE constraint ensures that there are no duplicate category names in the table.
- This table is linked to the posts table via the category_id, allowing posts to be categorized appropriately.

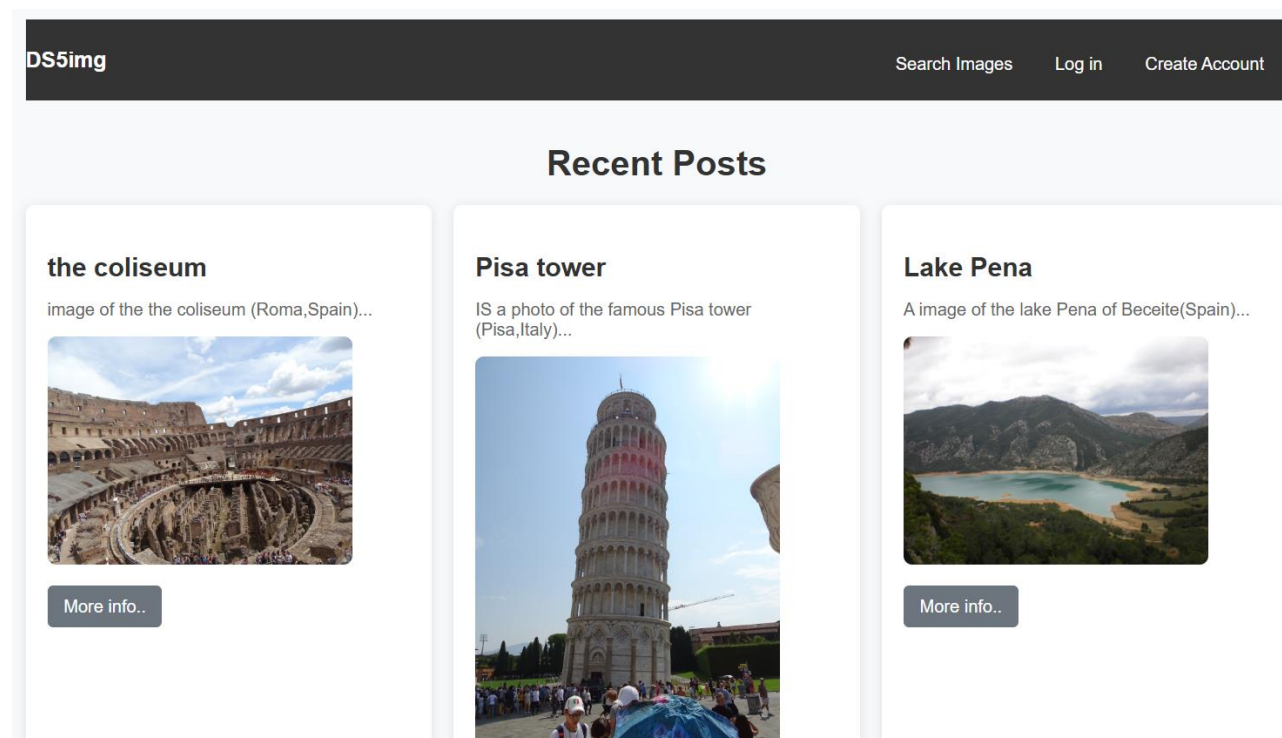
7 Use cases



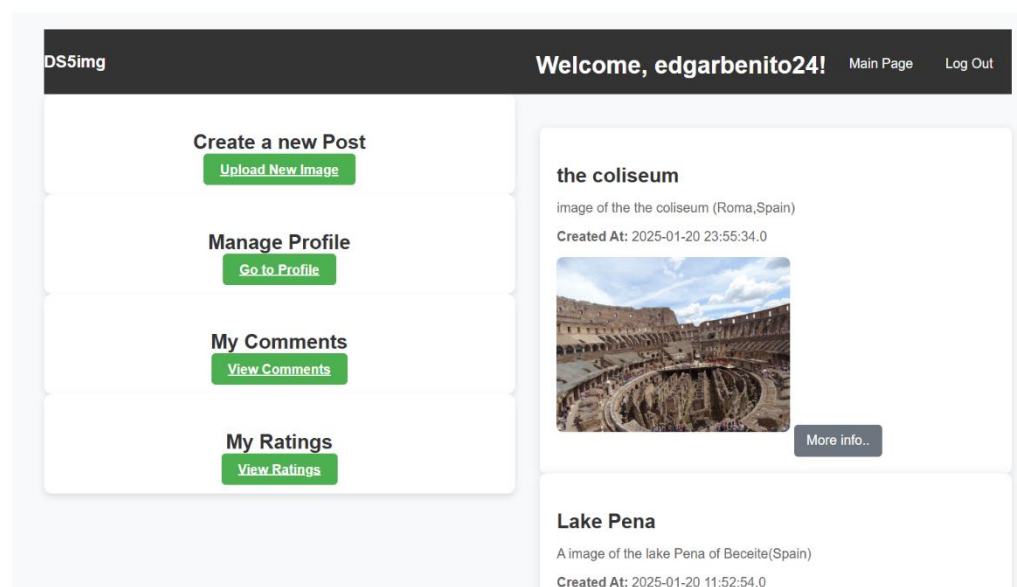


8 Mockups

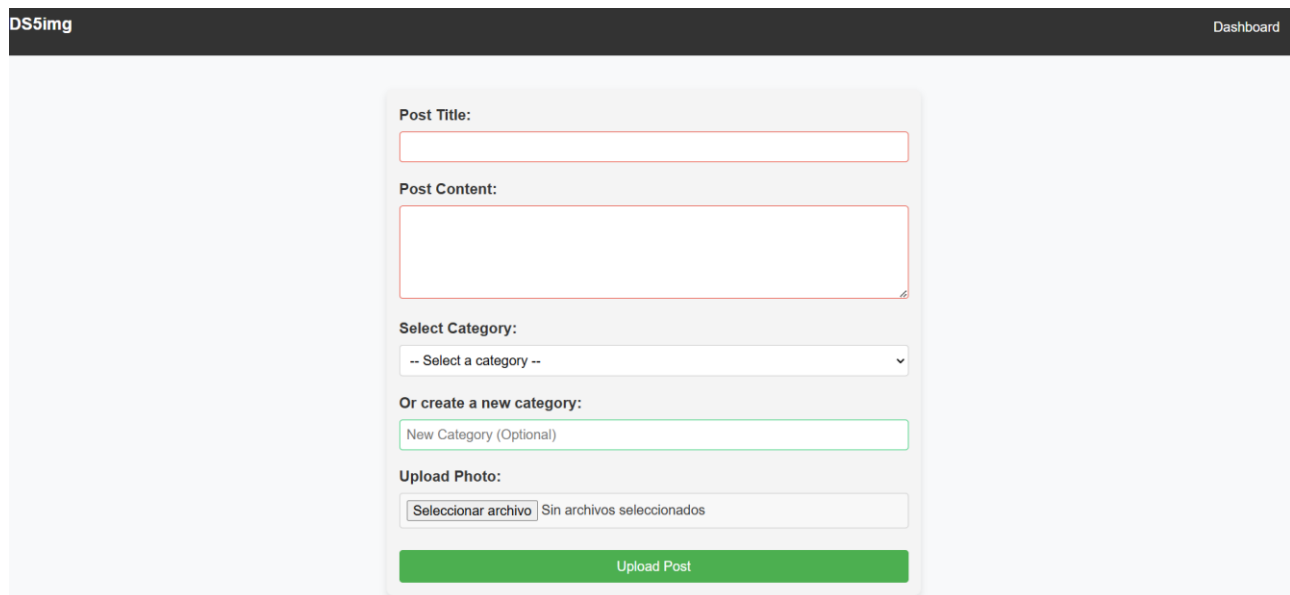
Main paige:



Upload photo:



Dashboard:



9 security issues

- **SQL Injection:**

Impact: Attackers can manipulate SQL queries, potentially gaining unauthorized access to or modifying the database. This can result in data leakage, loss of data, or complete system compromise.

Solution: Use prepared statements and parameterized queries to ensure user inputs are handled securely.

- **Cross-Site Scripting (XSS):**

Impact: Malicious users could inject scripts into the application that execute on other users' browsers. This can lead to stolen session cookies, redirected users, or exposure of sensitive information.

Solution: Sanitize and escape all user-generated content, especially in HTML, to prevent script execution.

- **Insecure Communication:**

Impact: If data is transmitted over unencrypted channels, attackers can intercept sensitive information like passwords, session tokens, or personal details through Man-in-the-Middle (MitM) attacks.

Solution: Use HTTPS with TLS/SSL to encrypt all communication between clients and the server to protect sensitive data.

- **Session Hijacking:**

Impact: Attackers can steal session tokens (e.g., from insecure cookies) and impersonate authenticated users, gaining unauthorized access to the system and performing actions on their behalf.

Solution: Use secure, HttpOnly cookies for session storage, and implement session expiration to reduce the risk of session hijacking.

- **Cross-Site Request Forgery (CSRF):**

Impact: Attackers could trick authenticated users into performing actions they did not intend, such as changing account settings or making unauthorized transactions.

Solution: Implement anti-CSRF tokens in all state-changing requests to verify the legitimacy of user actions and prevent CSRF attacks.

10 Planed division

| Task Division | Backend | Frontend | Documentation | Tests |
|-----------------------|---|-----------------|---------------|--------------------|
| Edgar Benito | Implementing Servlets and Database management | Developing CSS | Final Report | ApplicationTesting |
| Rosario Pavone | Database management | Developing HTML | Final Report | ApplicationTesting |