|  | **MINISTRY OF EDUCATION AND TRAINING** |
| --- | --- |

| **FPT UNIVERSITY** |
| --- |
| Capstone Project Document |
| Library Professional System |

| **Group 5** | |
| --- | --- |
| **Group Members** | Truong Xuan Dung - Leader - SE160459  Vu Trung Hieu - Member - HE163586  Vu Ngoc Minh Quan - Member - HE160804  Vu Anh Tu - Member - HE161770  Nguyen Cong Hieu - Member - HE173333 |
| **Supervisor** | Le Phuong Chi |
| **Ext Supervisor** |  |
| **Capstone Project code** | LPS |

- Hanoi, October /2024 -

**Table of Contents**

[Acknowledgement 2](#_heading=h.30j0zll)

[Definition and Acronyms 2](#_heading=h.1fob9te)

[I. Project Introduction 3](#_heading=h.3znysh7)

[1. Overview 3](#_heading=h.2et92p0)

[2. Product Background 3](#_heading=h.1t3h5sf)

[3. Existing Systems 3](#_heading=h.4d34og8)

[4. Business Opportunity 3](#_heading=h.2s8eyo1)

[5. Software Product Vision 3](#_heading=h.17dp8vu)

[II. Project Management Plan 4](#_heading=h.26in1rg)

[1. Overview 4](#_heading=h.lnxbz9)

[2. Management Approach 5](#_heading=h.2jxsxqh)

[3. Project organisation 7](#_heading=h.1y810tw)

[4. Project Communications 8](#_heading=)

[5. Configuration Management 9](#_heading=h.2xcytpi)

[III. Software Requirement Specification 9](#_heading=h.qsh70q)

[1. Product Overview 9](#_heading=h.3as4poj)

[2. User Requirements 10](#_heading=h.1pxezwc)

[3. Functional Requirements 10](#_heading=h.49x2ik5)

[IV. Software Design Description 49](#_heading=h.1hmsyys)

[1. System Design 50](#_heading=h.41mghml)

[2. Database Design 52](#_heading=h.3fwokq0)

[V. Software Testing Documentation 61](#_heading=)

[1. Scope of Testing 61](#_heading=h.19c6y18)

[2. Test Strategy 62](#_heading=h.3tbugp1)

[4. Test Reports 64](#_heading=)

# Acknowledgement

*[Fill team’s acknowledgement here…]*

# Definition and Acronyms

*[Fill all the definitions, acronyms,… used within the document] in the table format as below]*

| **Acronym** | **Definition** |
| --- | --- |
| PWM | Psychology website |
| AWS | Amazon Web Services |
| BA | Business Analysis |
| BR | Business Rule |
| ERD | Entity Relationship Diagram |
| GUI | Graphical User Interface |
| PM | Project Manager |
| SDD | Software Design Description |
| SPMP | Software Project Management Plan |
| SRS | Software Requirement Specification |
| UAT | User Acceptance Test |
| UC | Use Case |
| API | Application Program Interface |

# I. Project Introduction

*[Provide final project introduction information follow the template as part II in the Report #1]*

## 1. Overview

### 1.1 Project Information

*Project name: Library Professional System*

*Project code: LPS*

*Group name:*

### 1.2 Project Team

*Project manager: Vu Trung Hieu*

*Software development: Vu Trung Hieu, Truong Xuan Dung, Nguyen Cong Hieu, Vu Anh Tu, Vu Ngoc Minh Quan*

*Graphic design: Group user*

*Business Analyst: Group user*

*Test: Whole group*

*Test user: Group user*

## 2. Product Background

*Library management systems have emerged to meet the needs of libraries to borrow books and keep up with the technological era. With the advancement of technology and increasingly busy lifestyles, consumers are turning to online systems to quickly get what they want and save time. The library app market has expanded, offering convenient personalised features, online payments, and real-time order tracking. This library business management system aims to provide a fast, convenient, and secure experience for users while helping users access more books.*

## 3. Existing Systems

Currently, there are also quite a few e-library systems such as Hanoi Library, FPTU Library. These systems also allow users to look up, borrow books and track order status. However, there are still functions that need improvement, especially in terms of convenience and speed such as automatic payment, book delivery methods, etc.

## 4. Business Opportunity

*The book borrowing system, although not yet very strong, has experienced growth, especially as more and more consumers seek the convenience of sitting at home using a network-connected device. This condition creates a significant business opportunity to fill the gaps in the market, such as providing a faster book borrowing service.*

## 5. Software Product Vision

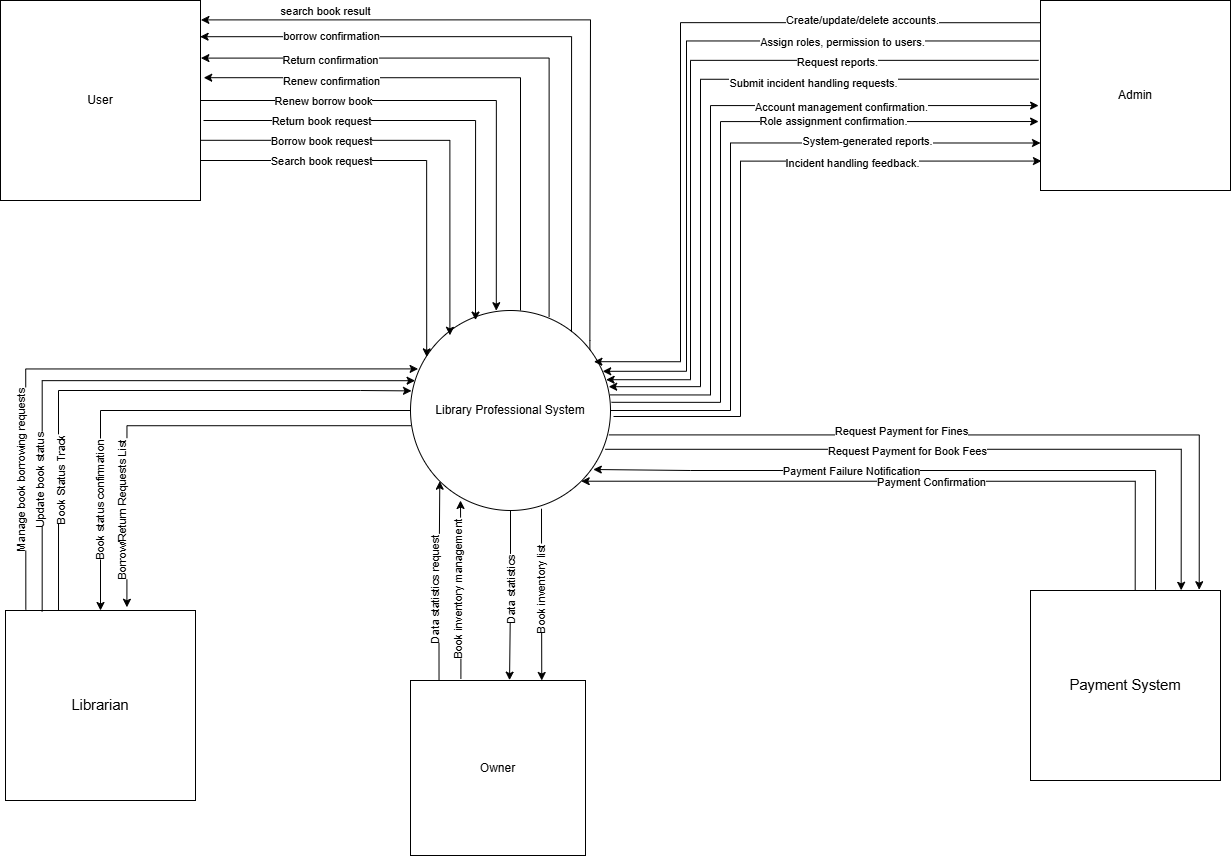
The Library Professional System envisions a world where libraries are seamlessly accessible, empowering diverse communities to learn, grow, and connect. By integrating intuitive, user-centered technology with advanced resource management, our system will transform the way patrons interact with library resources, making borrowing, research, and collaboration more efficient than ever. This platform will not only enhance the experience for library users but also streamline operations for staff, enabling them to focus more on community engagement and less on administrative tasks. Our balanced approach meets the needs of patrons, library professionals, and our organization, while staying aligned with current technological capabilities and resource constraints.

# II. Project Management Plan

*[Provide final project plan information follow the template as part II in the Report #2]*

## 1. Overview

### 1.1 Context Diagram



### 1.2 Scope & Estimation

[Create/Provide the list of software product following the table template as below. In this table, we categorize each software function into three levels of complexity (Simple, Medium, Complex) and estimate the total effort to complete each one in man-day]

### 1.3 Project Objectives

*[Provide the overall project objective description and then the specific target metrics of your project in term of quality, time, and cost (allocated effort distribution for project activities: requirement, design, coding, testing, project management, etc)]*

### 1.4 Project Risks

The table below contains the system’s project risks, each listed out with their description, impact on the project, likelihood of happening, and a response plan if the risk occurs.

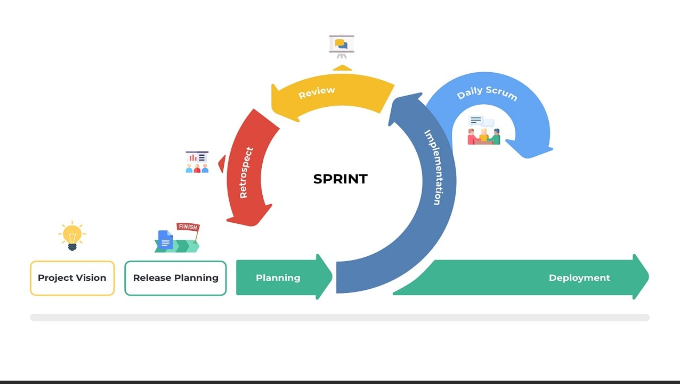
| **#** | **Risk Description** | **Impact** | **Possibility** | **Response Plans** |
| --- | --- | --- | --- | --- |
| 1 | Project management risks | Lead to schedule delays or weaken product quality. | Medium | Adjust schedules to ensure project completion on time. |
| 2 | Technology risks | Slow down development progress or cause errors that cannot be easily fixed | Medium | Conduct thorough testing before deployment, train employees to master new technology. |
| 3 | Quality risks | Reduces product reliability and increases repair costs. | Medium | Apply automated testing, continuous testing and enhance product quality management. |
| 4 | Requirements risk | Leads to expanding project scope or weakening product uniformity. | High | Establish a clear requirements validation process and regularly update it with stakeholders. |
| 5 | Security risks | Causes damage to the project's data and reputation. | High | Perform regular security audits, implement strong security measures, and train employees on information security. |

###### 

## 2. Management Approach

To manage and develop the library's book lending and returning system, the development team will apply the Scrum model, a flexible framework that allows for quick response to changing requirements from customers and users. Scrum helps ensure that the development process always focuses on the core values ​​of the system and completes the functionality through each sprint. Each sprint will allow the team to improve and develop additional features based on feedback from end users and stakeholders, helping the system to become more complete and suitable for the actual needs of the library.

### 2.1 Project Process

**

*The project process in Scrum will be divided into the following main steps:*

*Sprint Planning: At the beginning of each sprint, the team will identify the goals and functions that need to be developed or improved for the book lending and returning system. The work list will be selected from the product backlog, including features such as book list management, loan/return status tracking, overdue notifications, and membership card upgrade support.*

*Development and Implementation: During the sprint, members will develop and implement the planned features. The tasks are divided based on the roles and expertise of each member. For example: developing a user interface for borrowing/returning books, building an API to process data, and setting up automatic notifications for users.*

*Daily Standup: Every day, the team will have a short meeting (daily standup) to discuss progress, update difficulties encountered, and adjust the plan if necessary. This helps all members grasp the general situation and maintain flexibility, adapting to any changes in the development process.*

*Sprint Review: At the end of each sprint, the team will hold a review session to present newly completed features and collect feedback from stakeholders. This feedback will be added to the backlog for improvement in the following sprints.*

*Retrospective Meeting: After the sprint review, the team will conduct a retrospective to re-evaluate the work process, consider good points and issues that need improvement. This is an opportunity to optimise the work process and improve the team's efficiency in the following sprints.*

### 2.2 Training Plan

### 

| **Training Area** | **Participants** | **When, Duration** | **Waiver Criteria** |
| --- | --- | --- | --- |
| NestJS | All team member | Start at week 1, duration 1 days | Mandatory |
| NextJs | All team member | Start at week 1, duration 1 days | Mandatory |
| Git, Github | All team member | Start at week 1, duration 1 days | Mandatory |

## 3. Project organisation

| **Role** | **Responsibility** |
| --- | --- |
| Project Manager  (Dũng) | * Plan, develop work schedules, and coordinate communication between team members. * Ensure the team stays focused on the project's key goals. * Research and implement user experience (UX) design for the front-end of the system. |
| Business Analyst  (Tú) | * Analyse requirements. * Define project scope and create software requirements specification (SRS) documents. * Design entity relationship model (ERD) for the system. * Record and describe use cases in detail. |
| Technical Leader  (Quân) | * Define the overall system architecture based on the SRS document. * Configure the system and deploy the web to server. |
| Front-End / Back-End  Developer  (Hiếu CH/Hiếu/Quân) | * Participate in developing and coding product features, including both the front-end and back-end logic. * Collaborate with other team members to understand requirements and implement features based on the specification document (SRS). * Review and review other team members' source code to ensure code quality, compliance with technical standards and software development processes. * Participate in debugging the product after deployment. * Ensure seamless integration between the front-end and back-end, and assist in testing and deployment of the product. |
| Tester  (Team) | * Create a sample test document. * Define test strategy, create test cases * Implement test cases and log bugs. |

## 4. Project Communications

### 

| **Communication Item** | **Who/ Target** | **Purpose** | **When, Frequency** | **Type, Tool, Method(s)** |
| --- | --- | --- | --- | --- |
| Weekly meeting with mentor | All team members and mentor | Report progress  Discuss and solve problems | Every MON, WED | Offline meeting |
| | Team meeting | | --- | | All team members | Report tasks that  have done  Discuss issues,  resolve problems  Define the next task | Every TUE, FRI, SUN | Online meeting through Google Meet |
| Unscheduled meeting | All team members | Discuss and solve important issues | When there is  an important problem that needs to be solved quickly | Online meeting through Google Meet |

### 

## 5. Configuration Management

### 5.1 Document Management

*Document management tools: Google Drive will be used to store and share documents among members.*

*Document update process: Changes to documents will be made by the responsible member, and other members will be notified of important changes through a common communication channel.*

*Document tools: Microsoft Office 2019, Microsoft Excel 2019, Google Document, Google Sheet*

### 5.2 Source Code Management

*Source code management tools: Use Git and GitHub to manage and store source code.*

*Commit and merge process:*

*Each change will be committed with clear notes.*

*When completing a feature or fixing a bug, the source code will be merged into the develop branch and tested before merging into the main branch.*

### 6.3 Tools & Infrastructures

| **Category** | **Tools / Infrastructure** |
| --- | --- |
| **Technology** | NestJs, NextJs, Docker |
| **Database** | MongoDB |
| **IDEs/Editors** | Visual Studio Code |
| **Diagramming** | DrawIO |
| **Documentation** | Microsoft Office, Google Docs/Sheets/Slides |
| **Version Control** | Gitlab(Source Codes), Google Drive (Documents) |
| **Project management** | Jira (Schedule), GitHub(Tasks, Defects) |

## 

# III. Software Requirement Specification

*[Provide final software requirement specification follow the template as part II in the Report #3]*

## 1. Product Overview

*The Library Professional System is a software solution designed to manage various library processes efficiently. It allows users to search for, borrow, and return books while enabling librarians and book managers to handle book inventory and borrowing requests. The system also integrates with payment services to manage book fees and fines.*

*The system will primarily be used by Users (library members), Librarians, Book Managers, Admins, and external Payment Systems. Known constraints include dependency on accurate inventory data from book managers, proper role assignments by admins, and reliable integration with external payment systems for processing fees. It is assumed that all users will have access to the system via a web-based or local platform and that all components will function synchronously to avoid delays in book borrowing or payment processing.*

## 2. User Requirements

The **Library Professional System** interacts with several external entities, as depicted in the accompanying context diagram. The system serves as the core hub for interactions among **Users**, **Librarians**, **Book Managers**, **Admins**, and the **Payment System**.

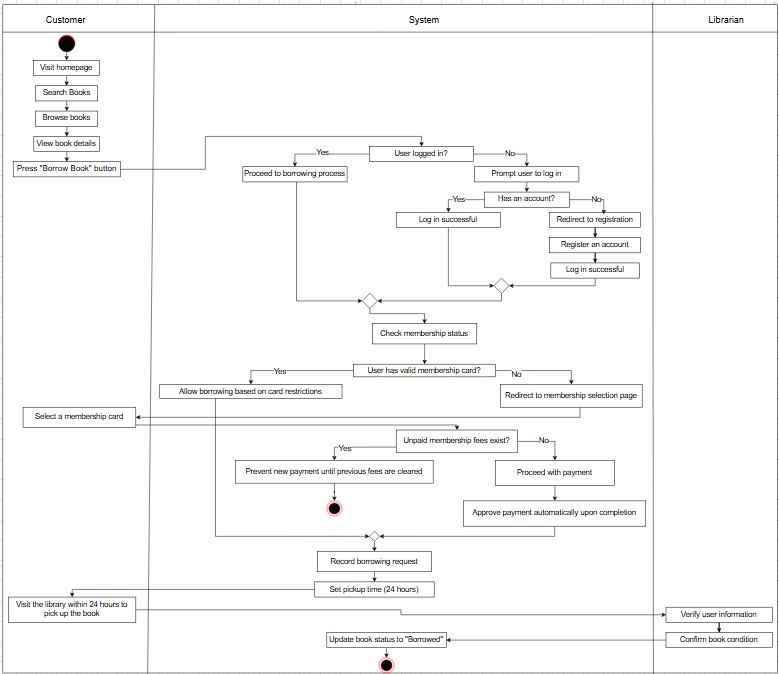
1. **Users** can search for books, place borrowing requests, renew borrowed books, and return books via the system.
2. **Librarians** manage these requests, update the status of books, and confirm the return and borrowing activities.
3. **Owners** maintain and update the book inventory, ensuring that the system is up to date with available books.
4. **Admins** handle account creation, role assignments, and monitor system incidents or generate reports.
5. The **Payment System** handles payments for book fines and fees, sending notifications of success or failure back to the system.

## 3. Functional Requirements

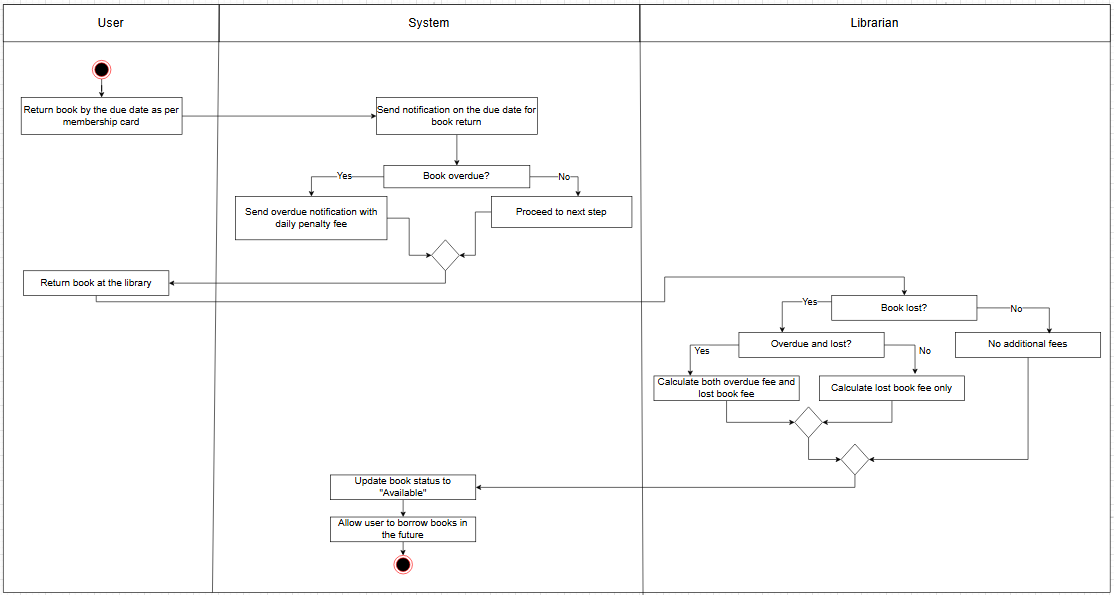
### 3.1 System Functional Overview

#### 3.1.1 Business flow

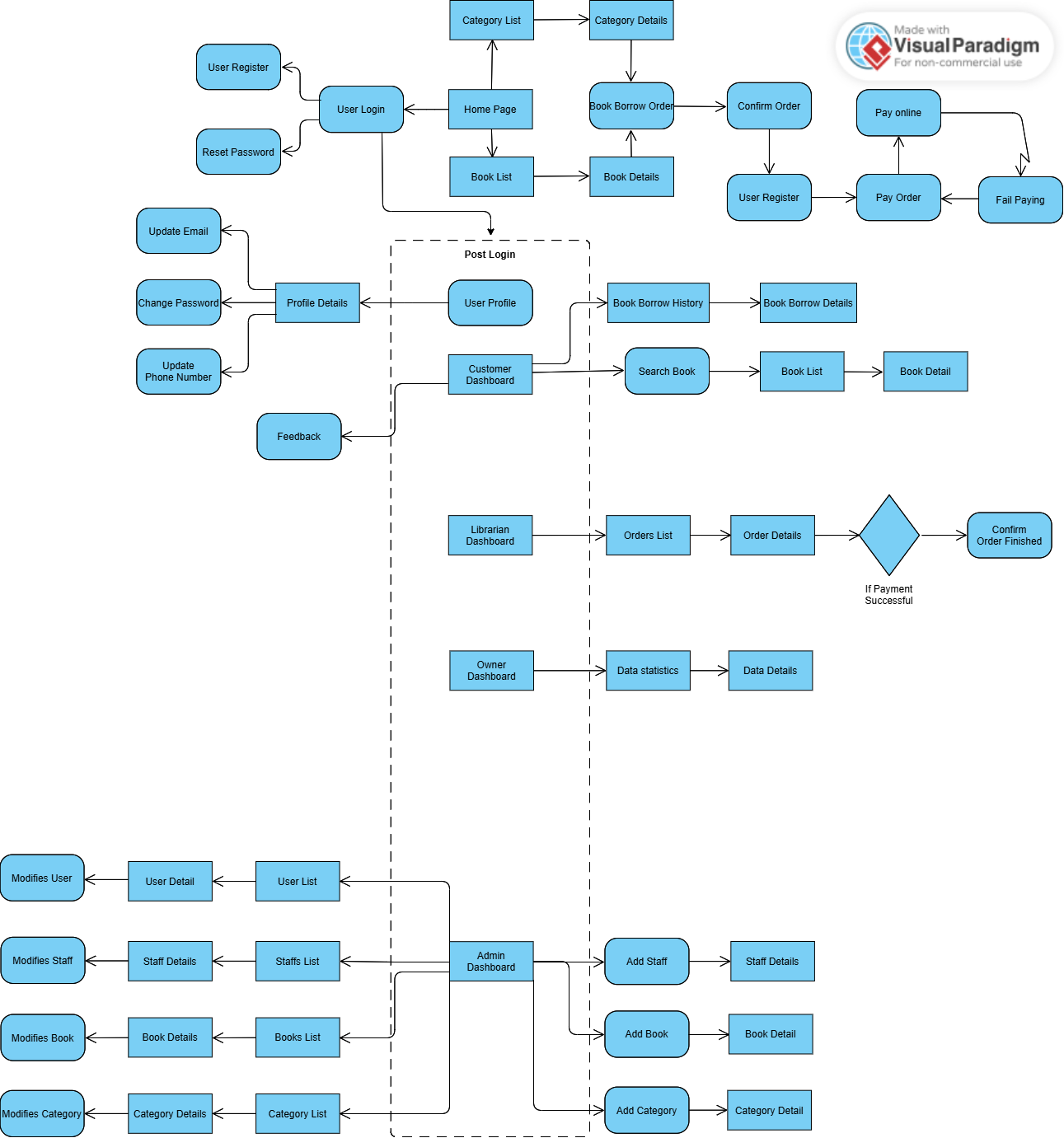
* Book borrowing stream:



* Book Return Stream:



#### 3.1.1 Screen flow



#### 3.1.2 Screen descriptions

| **#** | **Feature** | **Screen** | **Description** |
| --- | --- | --- | --- |
| 1 | View book catalogue | Homepage | The main page provides links to the book catalogue, search and manage loan applications. |
| 2 | User Management | User Register | New users can create an account by providing an email, username, and password. |
| 3 | User Management | User Login | Users can access the system using their registered account information. |
| 4 | User Management | Reset Password | Users may be provided with the option to reset their password via email. |
| 5 | User Management | User Profile | Allows displaying user information and allowing changing personal information |
| 6 | Category Management | Category List | Displays a list of available book categories |
| 7 | Category Management | Category Details | Displays details about the selected category, including a list of books in that category. |
| 8 | Book Management | Book List | Displays a list of books available in the library. |
| 9 | Book Details | Book Details | Provides book details like author, summary, availability. |
| 10 | Borrow | Book Borrow Order | The screen allows users to proceed with the loan order after selecting a book. |
| 11 | Payment | Pay Order | Users can proceed to pay the borrowing fee online after the book borrowing has been confirmed. |
| 12 | History | Book Borrow History | Displays a history of borrowed books, including date borrowed and return status. |
| 13 | Book Borrow Details | Book Borrow Details | Displays detailed information about a specific loan application. |
| 14 | Advanced book search | Advanced book search | Search for books in the library based on name, author, or category. |
| 15 | Librarian Dashboard | Librarian Dashboard | Librarian dashboard for managing book borrowing |
| 16 | Orders List | Orders List | List of pending or ongoing book borrow orders. |
| 17 | Order Details | Order Details | Displays borrowing order details, including borrower, and term |
| 18 | Owner Dashboard | Owner Dashboard | Dashboard to manage books, catalogues, and track library activities. |
| 19 | Admin Dashboard | Admin Dashboard | Admin centre, allows management of users, staff, books and reports. |

#### 

#### 3.1.3 System user role

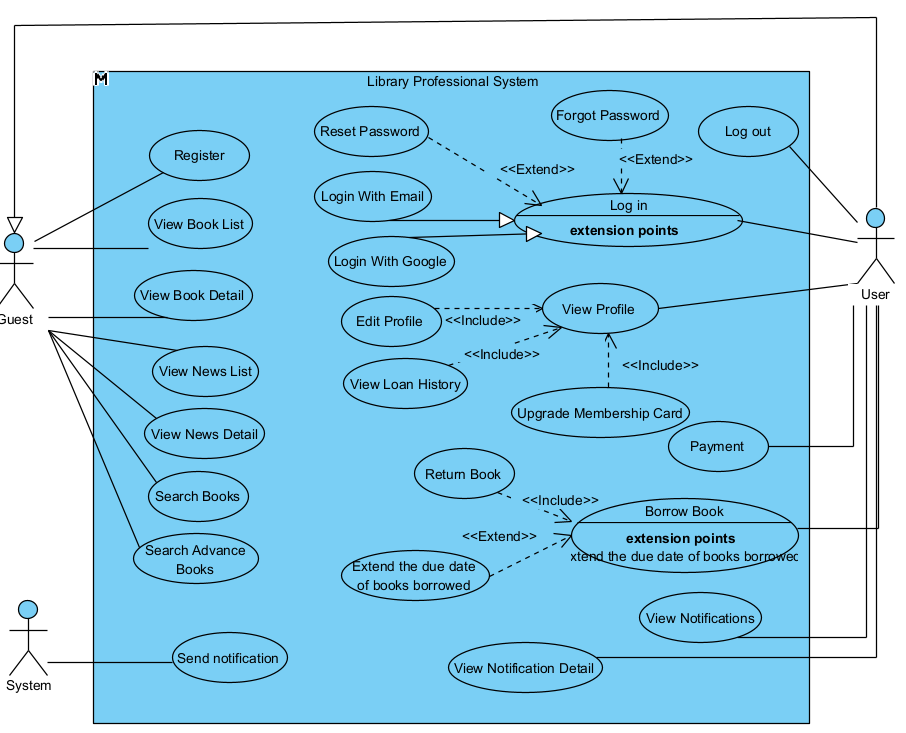
* User: Regular user of the library system. This role has the ability to search for books, borrow books, and pay related fees.
* Librarian: A librarian is a person who manages and operates activities related to borrowing and returning books in the library.
* Admin: This role has the ability to manage the entire system, manage users, reports, and authorization.
* Owner: The book owner is responsible for adding and managing book information in the library system. This role ensures that books are archived, updated, and available to users.

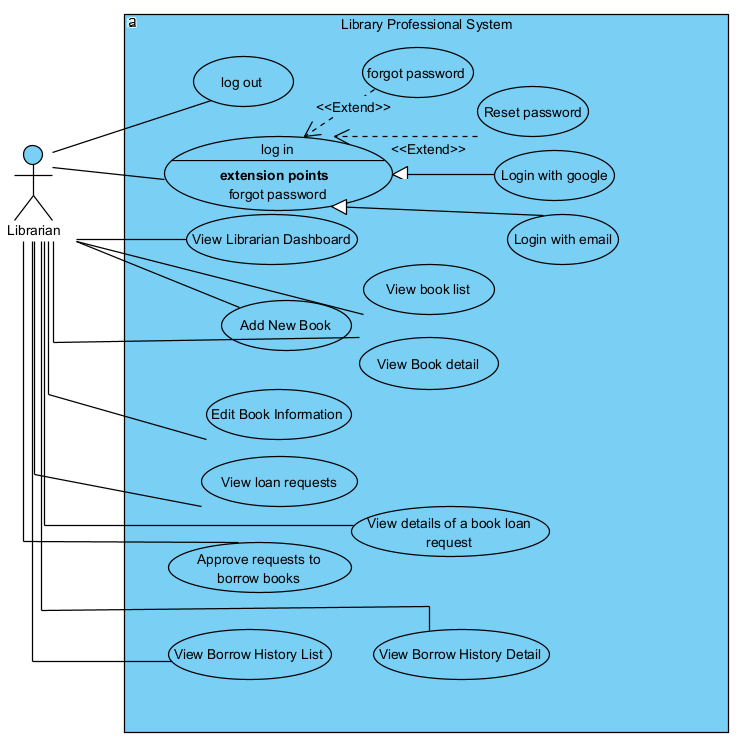
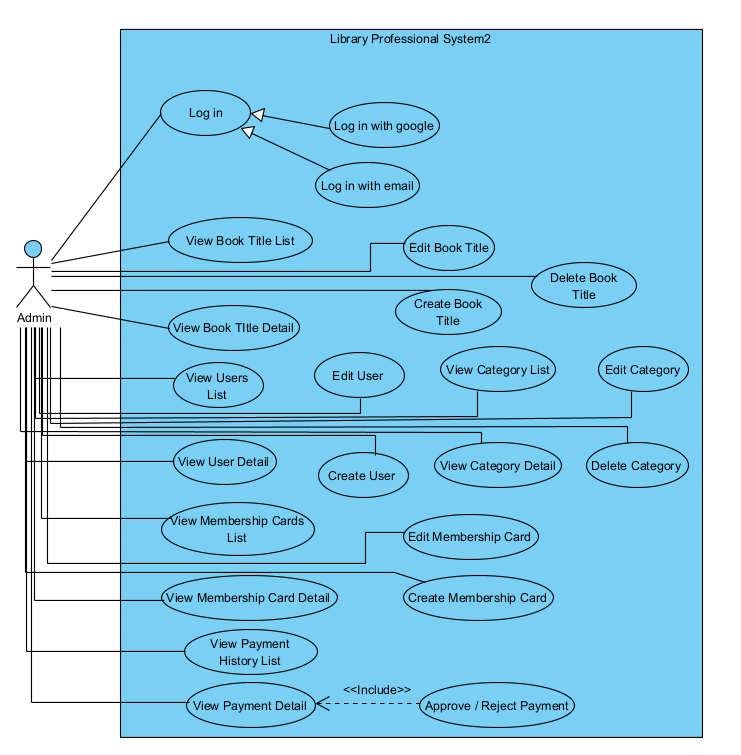
#### 3.1.4 Screen authorization

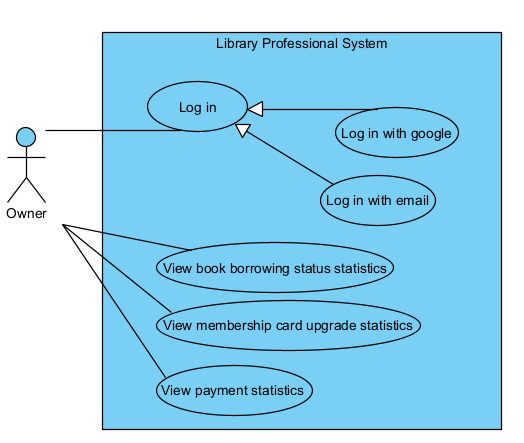
| **Screen** | **User** | **Librarian** | **Owner** | **Admin** |
| --- | --- | --- | --- | --- |
| Homepage | x | x | x | x |
| User Register | x |  |  |  |
| User Login | x | x | x | x |
| Reset Password | x |  |  |  |
| User Profile | x |  |  |  |
| Category List | x | x | x | x |
| Category Details | x |  | x | x |
| Book List | x | x | x | x |
| Book Details | x |  | x | x |
| Book Borrow Order | x | x |  | x |
| Pay Order | x |  |  | x |
| Book Borrow History | x | x |  |  |
| Book Borrow Details | x | x |  | x |
| Advanced book search | x |  |  | x |
| Librarian Dashboard |  | x |  |  |
| Orders List | x | x |  |  |
| Order Details | x | x |  |  |
| Owner Dashboard |  |  | x |  |
| Admin Dashboard |  |  |  | x |

#### 3.1.5 Use Case diagram

User:

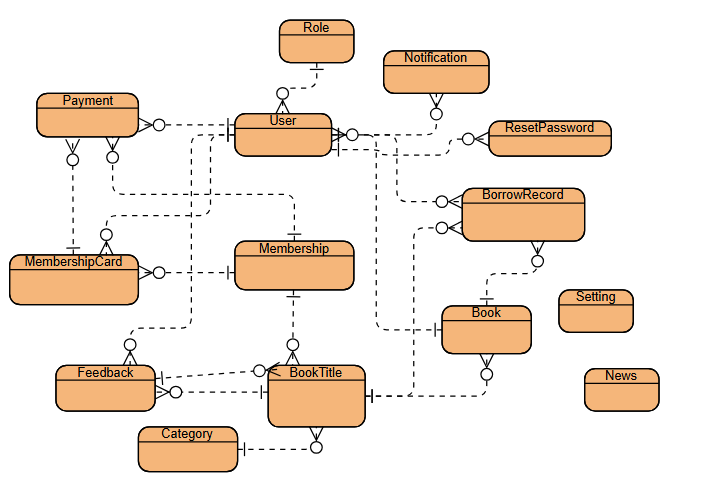


Librarian:  
  
Admin:  
  
Owner:



#### 3.1.6 ERD

*[Provide functionality overview of software system: screen flow, screen descriptions, system user roles, screen authorization, non-screen functions, ERD]*

**

### 3.2 User Management

*[Profile functional specification for the feature, with the details on each function]*

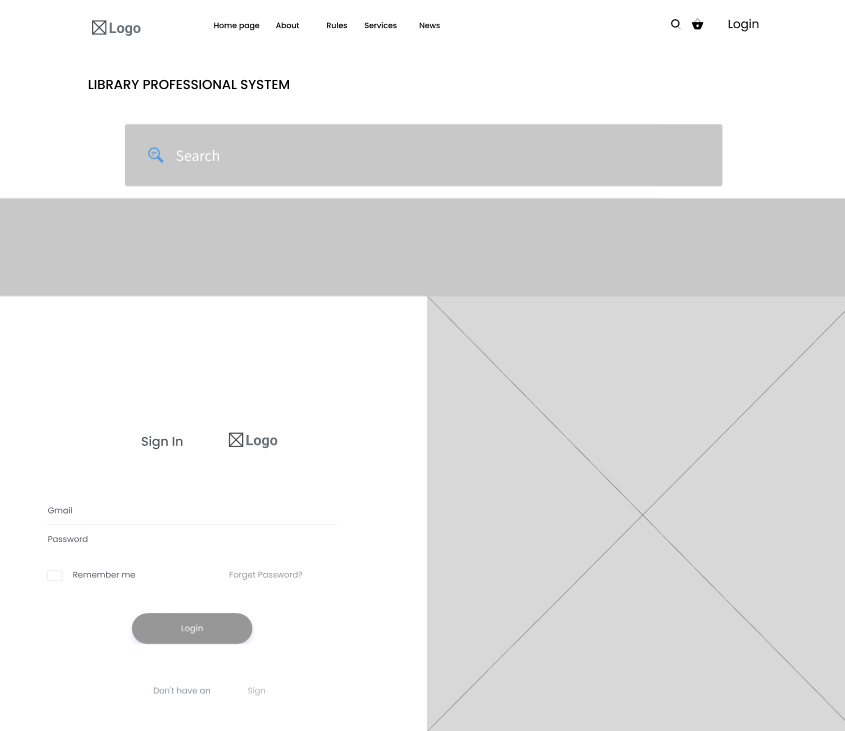
#### 3.2.1 User Login

**Function Trigger:**

* The user accesses the login screen through the "Login" button in the top navigation bar of the website.
* This function is triggered when the user needs to log into their personal account to access library-related services like borrowing books, managing reservations, or viewing account details.

**Function Description:**

* **Actors/Roles:** General users (library members) and administrators.
* **Purpose:**
* This function allows users to securely log into the library management system. It includes authentication using the user's email and password, with features like "Remember Me" to maintain login sessions and "Forget Password?" for password recovery.
* **Interface:**
  + Input fields include Email and Password.
  + Buttons include: "Login" for submitting credentials, "Forget Password?" for password recovery, and "Sign Up" for new user registration.
  + The "Remember Me" checkbox allows users to stay logged in on the current device.
* **Data Processing**: Upon submission, the system validates the provided email and password against the database. If the credentials are correct, the user is redirected to their personalised dashboard. If incorrect, the system displays an error message and limits the number of login attempts to prevent brute force attacks.
* **Screen Layout:** Mockup of the screen layout, as shown below for the User Login screen:

**

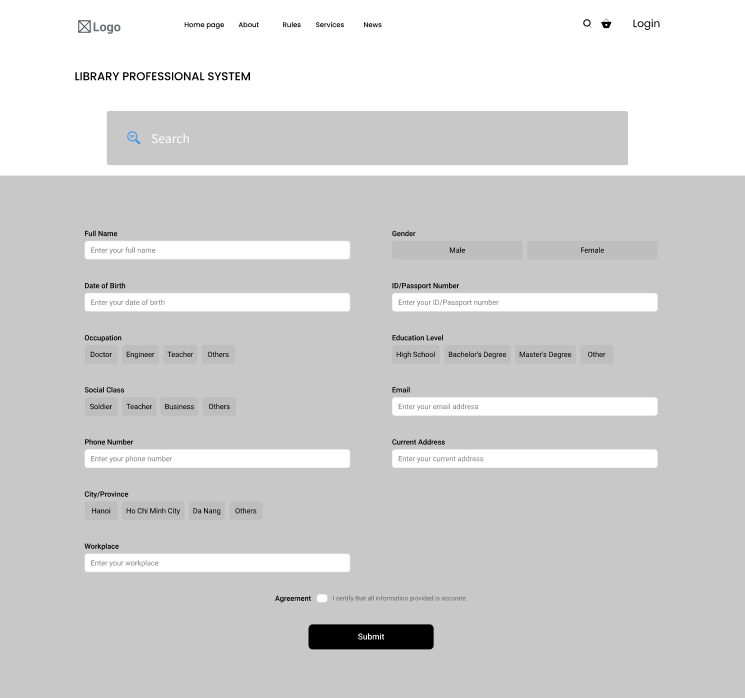
#### 3.2.2 User Registration

**Function Trigger:**

* The user accesses the registration screen by clicking the "Sign Up" button on the login screen or the homepage.
* This function is triggered when a new user wants to create an account to access the library management system.

**Function Description:**

* **Actors/Roles**: New users (library members) and administrators.
* **Purpose**:  
  This function allows new users to register for an account on the library management system by providing personal details like name, email, and password. Once registered, they can log into the system and access personalised features like borrowing books and managing reservations.
* **Interface**:
  + Input fields include: Full Name, Email, Password, Confirm Password.
  + Buttons include: "Register" to submit the details and create an account.
  + A "Cancel" button allows users to exit the registration process if needed.
* **Data Processing**:  
  Upon submission, the system validates the input fields to ensure that the email is unique, the passwords match, and all required fields are correctly filled. The data is then saved in the user database, and the user is prompted to log in with their new account credentials. If the email already exists, an error message is displayed.
* **Screen Layout**: Mockup of the screen layout, similar to the User Registration screen:



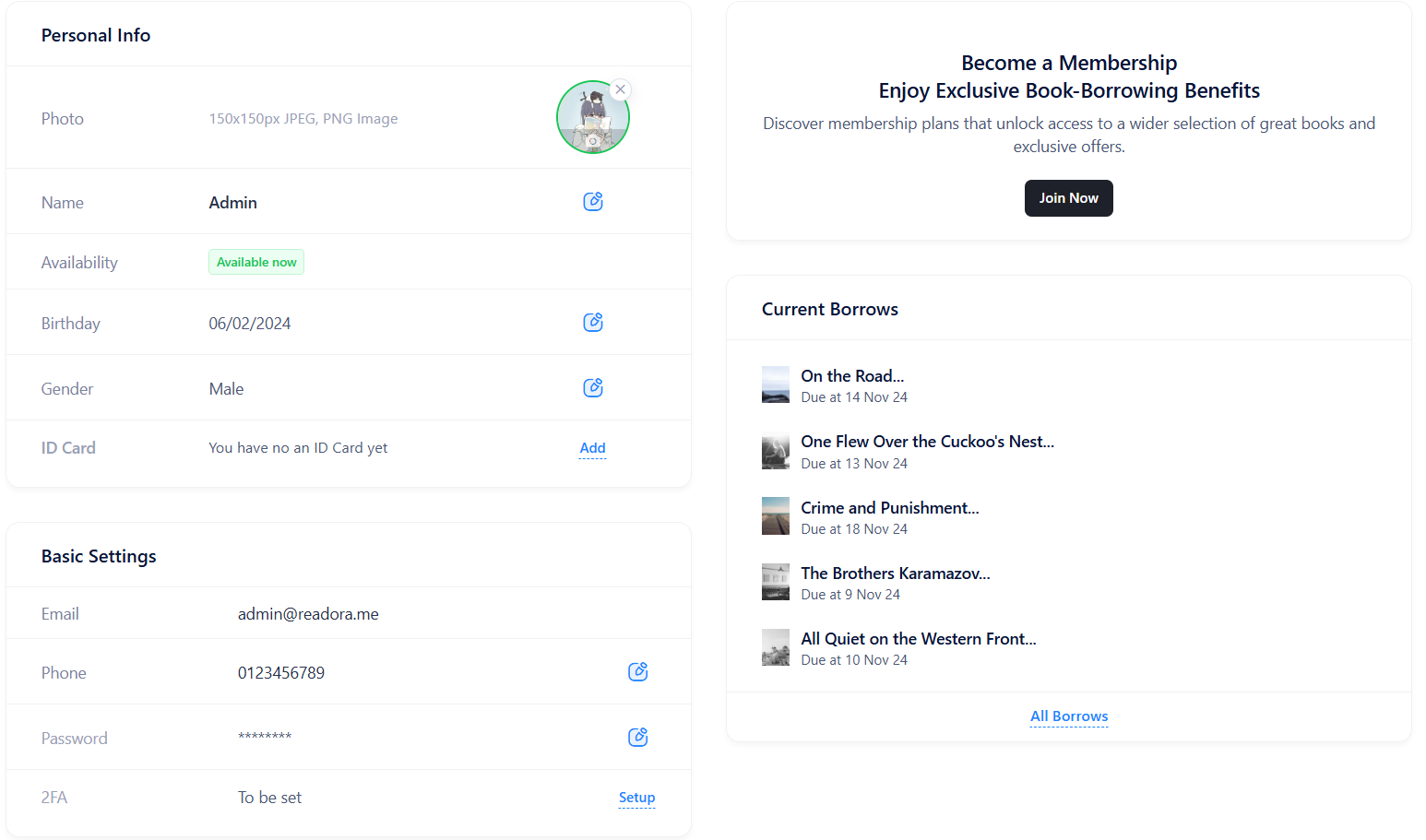
#### 3.2.3 View Profile

**Function Trigger:**

* This function is triggered when a user accesses their profile page from the Home Page. The system loads and displays the user’s personal information, account settings, and current borrowing information.

**Function Description:**

* **Actors/Roles:** Users
* **Purpose:** This feature allows users to view and manage their personal information, basic account settings, and borrowing history. It provides users with an overview of their profile, including membership options and current borrowed items.
* **Interface:**
  + **Personal Information Section**:
    - **Photo**: Displays the user’s profile picture with an option to update it.
    - **Name**: Shows the user’s name.
    - **Availability**: Displays the user's current availability status.
    - **Birthday**: Shows the user’s date of birth.
    - **Gender**: Displays the user’s gender.
    - **ID Card**: Shows the user's ID card status, with an option to add an ID if not already provided..
  + **Basic Settings Section**:
    - **Email**: Displays the user’s email address with an option to edit it.
    - **Phone**: Shows the user’s phone number with an option to edit it.
    - **Password**: Displays a masked password with an option to change it.
    - **2FA (Two-Factor Authentication)**: Shows the current 2FA status with a setup option if it’s not already enabled.
  + **Current Borrows Section:**
    - **Borrowed Books List:** Shows a list of books the user currently has on loan, including:
      * **Book Title**: The title of each borrowed book.
      * **Due Date**: The return due date for each borrowed book.
    - **All Borrows Link**: A link to view the complete borrowing history.
* **Data Processing**: The system retrieves the user's personal information, account settings, membership status, and current borrowing records from the database. It then displays this data in a structured format on the profile page. Updates to any personal information are validated and saved to the database.
* **Screen Layout:** Mockup of the screen layout, as shown below for the View User Profile screen:



#### 3.2.4 Forgot Password

**Function Trigger:**

* This function is triggered when a user clicks the "Forgot Password?" link on the login page. The system then guides the user through the password reset process by prompting them to enter their email address.

**Function Description:**

* **Actors/Roles:** Users, System.
* **Purpose:** This feature allows users who have forgotten their password to reset it securely. By providing their email, users can initiate a password reset request, enabling them to regain access to their account.
* **Interface:**
  + **Password Field with Forgot Password Link (Login Screen)**:
    - **Password Input**: An input field for users to enter their password when attempting to log in.
    - **Forgot Password Link**: A clickable link labeled "Forgot Password?" next to the password field. When clicked, it takes the user to the next screen to enter their email for password recovery.
  + **Email Entry for Password Reset**:
    - **Email Input Field**: A field where users can enter their registered email address to receive a password reset link.
    - **Continue Button**: A button labeled "Continue" that, when clicked, initiates the password reset process by sending an email to the provided address.
* **Data Processing**: When the user clicks "Forgot Password?" and enters their email, the system verifies if the email is registered. If valid, it sends a password reset link to the user's email. The link allows the user to securely reset their password. If the email is invalid or unregistered, the system displays an error message to prompt the user to check their input.
* **Screen Layout:** Mockup of the screen layout, as shown below for the **Forgot Password** screen:

#### 

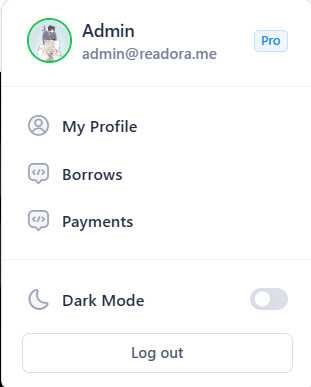
#### 3.2.5 Logout

**Function Trigger:**

* This function is triggered when a user clicks the "Log out" button within their profile menu. This action initiates the logout process, disconnecting the user from their current session.

**Function Description:**

* **Actors/Roles:** Users.
* **Purpose:** The purpose of this feature is to allow users to securely end their session, ensuring that their account and personal data remain protected when they leave the application.
* **Interface:**
  + **Profile Menu**: The profile menu includes the following elements:
    - **Profile Information**: Displays the user’s profile picture, name, email, and membership status (e.g., "Pro").
    - **Navigation Links**: Quick links to sections like:
      * **My Profile**: Takes the user to their profile page.
      * **Borrows**: Directs the user to their current borrowing information.
      * **Payments**: Directs the user to view payment details or manage membership.
  + **Dark Mode Toggle**: An option to switch between light and dark themes.
  + **Logout Button**: Located at the bottom, allowing the user to log out.
* **Data Processing**: When the "Log out" button is clicked, the system processes the request by ending the user’s current session. This may include clearing session cookies or tokens associated with the user's login credentials. Once logged out, the system redirects the user to the login page or home screen.
* **Screen Layout:** Mockup of the screen layout, as shown below for the **Log out**  screen:

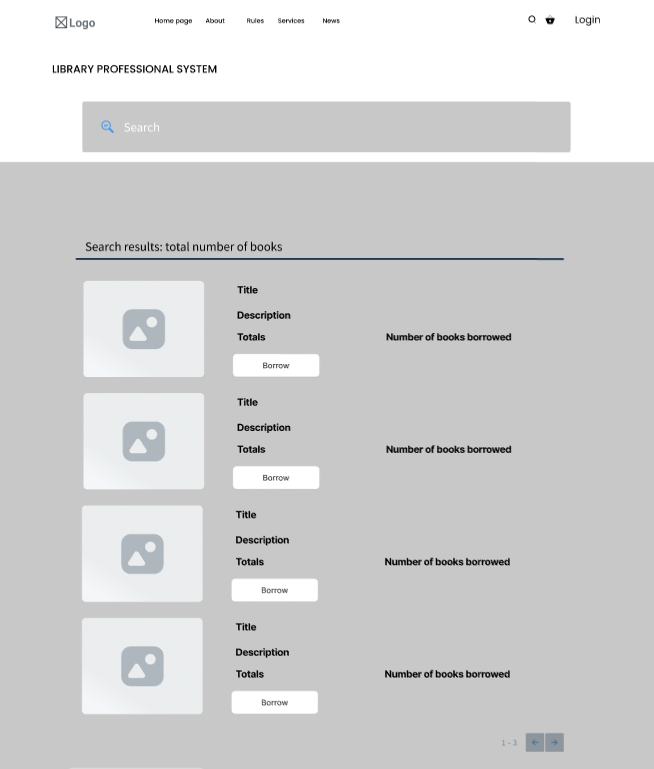


### 3.3 Book Management

#### 3.3.1 Search Book

**Function Trigger**:

* This function is triggered when a user enters a query into the search bar and presses the search button. Users can access the search functionality from the homepage or any other page with the search bar visible.
* It allows users to search for books based on various criteria such as title, author, or keyword.
* **Function Description**:
* **Actors/Roles**: General users (library members) and administrators.
* **Purpose**:  
  This function enables users to search the library catalogue for available books, view book details, and initiate a borrowing request if the book is available. It displays search results, including book title, description, total number of copies, and how many copies have been borrowed.
* **Interface**:
  + **Search Bar**: Located at the top of the page, allows users to input search queries (e.g., title, author).
  + **Search Results**:
    - For each book, the system displays:
    - Title
    - Description
    - Total number of copies available
    - Number of books already borrowed
    - Each book includes a "Borrow" button to initiate a borrow request.
  + **Pagination**: If the search results exceed a certain limit, pagination will be provided to navigate through pages of results.
* **Data Processing**:  
  When the user submits a search query, the system processes it by querying the database of available books. It returns a list of books matching the search criteria, along with the number of available copies and those already borrowed. The system checks the availability status to allow or disable the "Borrow" button.
* **Screen Layout**: Mockup of the screen layout, sample below is for the search results page:

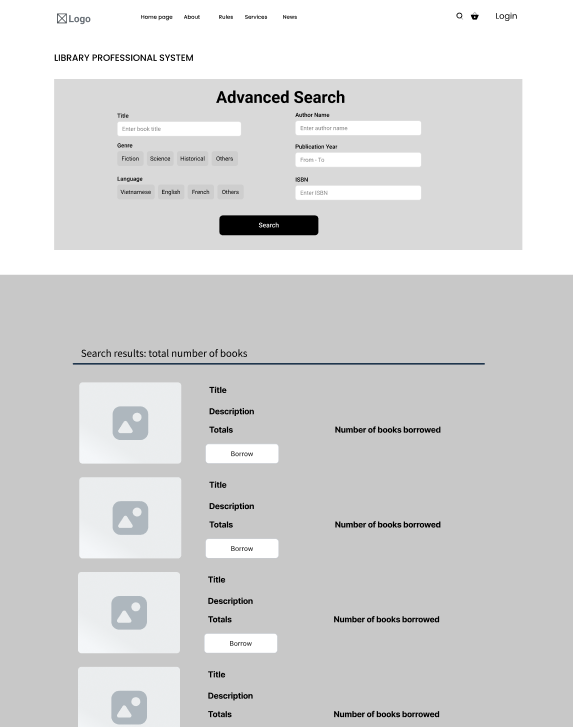


#### 

#### 3.3.2 Search Advanced Book

**Function Trigger**:

* The user can access the Advanced Search screen by selecting the "Advanced Search" option, either from the main navigation bar or directly from the homepage. This function is triggered when the user needs to search for books using specific criteria like title, author name, genre, language, publication year, or ISBN.
* **Function Description**:
* **Actors/Roles**: General users (library members) and administrators.
* **Purpose**:  
  The Advanced Search function allows users to conduct detailed searches within the library catalogue by entering or selecting from multiple fields, including Title, Author Name, Genre, Language, Publication Year, and ISBN. This provides users with more specific control over the search results, enabling efficient retrieval of desired books.
* **Interface**:
  + **Input Fields**:
    - Title: Text input for book title.
    - Author Name: Text input for author name.
    - Genre: Selection options (e.g., Fiction, Science, Historical).
    - Language: Selection options (e.g., Vietnamese, English, French).
    - Publication Year: A range input to specify the publication year (From - To).
    - ISBN: Text input for entering the book's ISBN.
  + **Search Button**: Initiates the search based on the provided inputs.
  + **Search Results**: Similar to basic search results, displaying book Title, Description, Total Copies, Number of Copies Borrowed, and a "Borrow" button for available books.
* **Data Processing**:
  + - Once the user submits the search query, the system processes the input fields by filtering the book catalogue based on the specified criteria. The results are displayed, allowing users to view the available books and borrow them if possible. The system ensures that all input criteria are considered, and results matching multiple criteria are prioritised.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Advanced Search** screen:

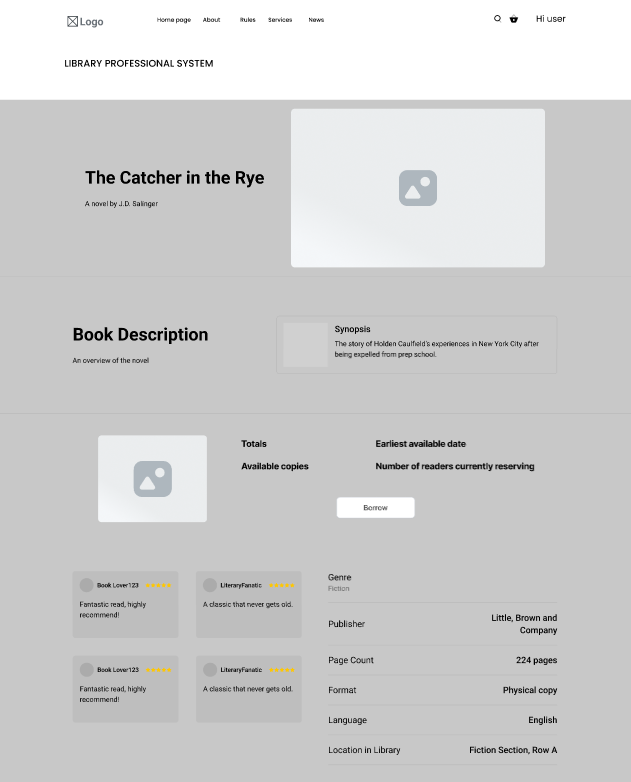


#### 3.3.3 View Book Details

**Function Trigger**: This function is triggered when a user clicks on a book title or cover image from the search results or any other section where books are listed. The system will then load the detailed view for the selected book.

**Function Description**:

* **Actors/Roles**: General users (library members) and administrators.
* **Purpose**:  
  This function allows users to view comprehensive information about a book, including its title, author, synopsis, availability, and reviews. Users can check how many copies are available, the earliest available date, and the number of readers currently reserving the book. The user can also request to borrow the book if it is available.
* **Interface**:
  + **Book Title and Author**: Displayed prominently at the top.
  + **Book Cover Image**: A placeholder image for the book cover.
  + **Synopsis Section**: Provides a brief description or summary of the book's content.
  + **Availability Information**:
    - Number of available copies.
    - The earliest available date for borrowing.
    - Number of users currently reserving the book.
  + **Borrow Button**: Enabled if copies are available for borrowing. If not, the button may be disabled or show a reservation option.
  + **User Reviews Section**: Displays recent reviews from users, including usernames, ratings, and comments.
  + **Additional Information**: Displays detailed metadata about the book, such as:
    - Genre
    - Publisher
    - Page count
    - Format (e.g., physical copy, e-book)
    - Language
    - Location in the library (section and row).
* **Data Processing**:  
  The system retrieves book details from the database when the user clicks on a book. It checks availability status in real time, showing the number of available copies and the number of users currently reserving it. Reviews and additional information (like genre and publisher) are also displayed. The "Borrow" button becomes clickable if the book is available for borrowing.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Book Detail View** screen:



#### 3.3.4 Create New Book

**Function Trigger:**

* This function is triggered when a library manager accesses the Create Book page from the manager dashboard. The page allows the manager to input details for a new book entry and submit it for approval.

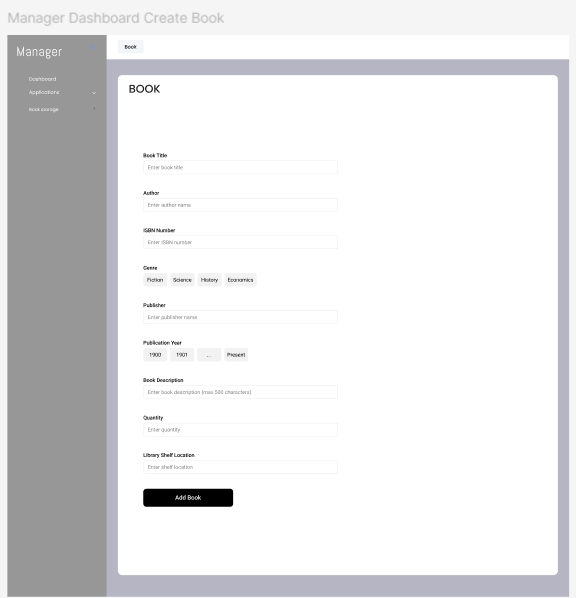
**Function Description**:

* **Actors/Roles**: Library managers, administrators.
* **Purpose**:  
  This function enables library managers to create new book entries for the library's inventory. Once the new book is added, it will be submitted for review by an administrator. The administrator must approve the entry before it is added to the library’s active catalog.
* **Interface**:
  + **Input Fields**:
    - **Book Title**: The title of the book.
    - **Author**: The author of the book.
    - **ISBN Number**: ISBN for the book.
    - **Genre**: The genre of the book, selected from options (e.g., Fiction, Science, History).
    - **Publisher**: The name of the publisher.
    - **Publication Year**: The year of publication, with a range selector for earlier years.
    - **Book Description**: A description or summary of the book (up to 500 characters).
    - **Quantity**: The total number of copies available.
    - **Library Shelf Location**: The physical location where the book will be stored in the library.
  + **Add Book Button**: A button to submit the book for creation. Once clicked, the book will not be immediately available but instead sent to the admin for approval.
* **Data Processing**:  
  Upon clicking "Add Book," the system validates all fields, checking for correct formats (e.g., ISBN number, valid year, and required fields). If validation is successful, the system saves the book entry in a pending state. A notification is sent to the administrator for approval. The administrator can approve, reject, or request changes for the book entry before it becomes available in the library's active inventory.

**Approval Workflow**:

* After the book is submitted, it will enter an "Awaiting Approval" status. The admin will receive a notification about the pending book and can:
  + Approve the book, adding it to the active library inventory.
  + Reject the book with reasons for rejection.
  + Request edits, sending the book back to the manager for further modifications.

**Screen Layout**:  
Mockup of the screen layout, as shown below for the **Create Book (Manager Dashboard)** page:



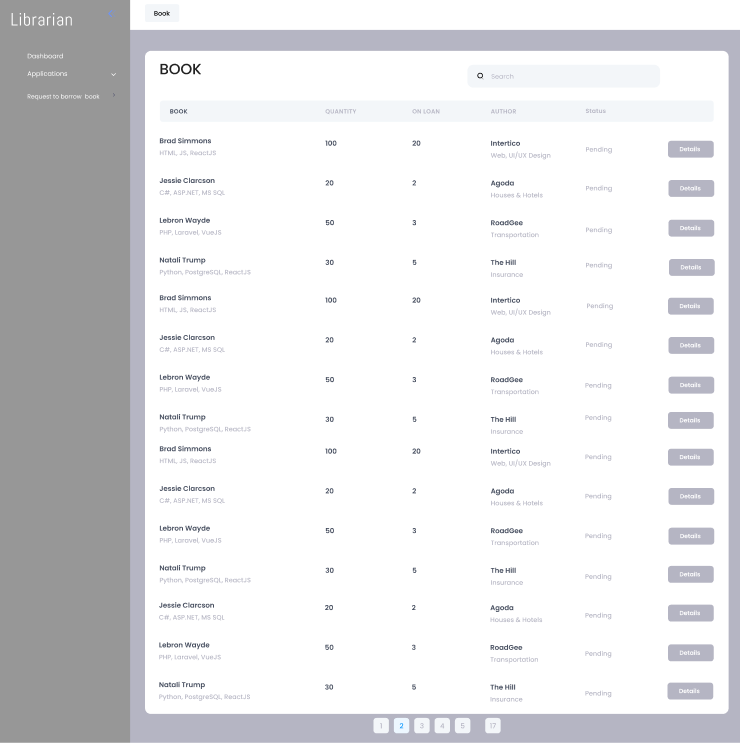
#### 3.3.5 Manage Book List

**Function Trigger**:

This function is triggered when the librarian accesses the **Book Inventory Management** page through their dashboard. The system loads all the books currently registered in the library system along with relevant details like quantity, copies on loan, author, and current request status.

**Function Description**:

* **Actors/Roles**: Librarians, administrators.
* **Purpose**:  
  This function enables librarians to manage the library's book inventory. It allows them to view detailed information about each book, including the total quantity, number of copies currently on loan, author details, and the status of any pending requests. Librarians can access further details for each book by clicking the "Details" button.
* **Interface**:
  + **Book List**:
    - Displays a list of all books in the library, including:
    - Book Title
    - Quantity: Total number of copies in the library.
    - On Loan: Number of copies currently borrowed by users.
    - Author: The author of the book.
    - Status: The current status of borrowing requests (e.g., Pending).
  + **Search Bar**: A search function located at the top of the page allows librarians to search for specific books by title or author.
  + **Details Button**: Each book has a "Details" button that, when clicked, takes the librarian to a page displaying detailed information about the book, including borrowing history, request management, and additional metadata.
  + **Pagination**: Pagination controls at the bottom of the page allow librarians to navigate through the book list if it exceeds one page.
* **Data Processing**:  
  The system loads book data from the library’s database and displays it in a table format. The system also tracks the number of books on loan and updates this information in real-time. When the librarian clicks "Details," the system fetches detailed information about that specific book, such as user borrowing history and pending requests.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Librarian Book Inventory Management** screen:



### 3.4 Book Borrowing Management

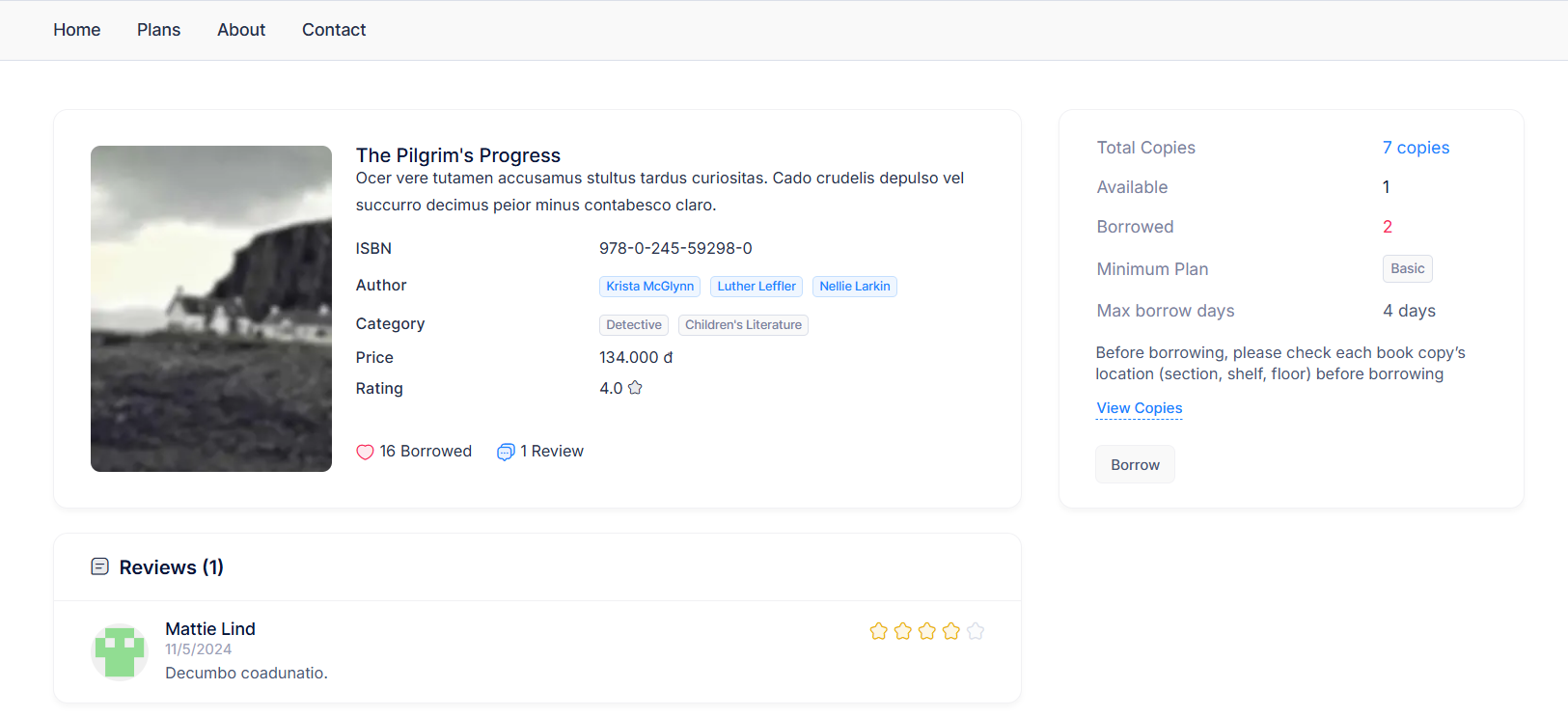
#### 3.4.1 Borrow Book

**Function Trigger**:

This function is triggered when a user accesses the "Borrow Book" section on their account dashboard. The system loads all available books in the library catalogue that the user can borrow, along with essential details like title, author, availability status, and borrowing limits based on the user's membership type.

**Function Description**:

* **Actors/Roles:** Users, System.
* **Purpose**:  
  This feature allows users to browse the library’s catalog and initiate borrowing requests for available books. Users can view key information about each book, including availability and author details. The "Borrow" function also provides a simple interface for requesting books to borrow, depending on the user's borrowing limits and membership status.
* **Interface**:
  + **Book Detail**:
    - **Book Title**: Title of the book.
    - **Author**: Name of the author(s).
    - **Description**: A brief synopsis or overview of the book's content.
    - **Genre/Category**: The book’s genre or category
    - **ISBN**: Displaying the book's ISBN for identification.
    - **Publisher and Published Date**: Information about the publisher and publication date
    - **Availability**: Indicates whether the book is available or currently on loan.
    - **Borrow Limit**: Shows the user's borrowing allowance based on their membership type (e.g., max books allowed at a time).
  + **Availability & Borrowing Information:**
    - **Total Copies:** Total copies in the library’s collection.
    - **Available Copies:** Number of copies currently available for borrowing.
    - **On Loan:** Number of copies currently borrowed by others.
    - **Borrow Limit:** Shows the user's current borrowing allowance based on their membership type.
    - **Expected Return Date:** If all copies are on loan, shows the earliest return date.
  + **Borrow Button:**Each book entry includes a "Borrow" button. When clicked, the system initiates a borrowing request for that book. If the request is successful, the book status updates, and the user is notified of the estimated pick-up time or next steps.
* **Data Processing**:  
  When the "Book Detail" page loads, the system fetches detailed information from the library’s database. If the user clicks the "Borrow" button, the system verifies the borrowing limits based on the user’s membership, logs the borrowing request, updates the availability, and notifies the user.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Borrow Book** screen:



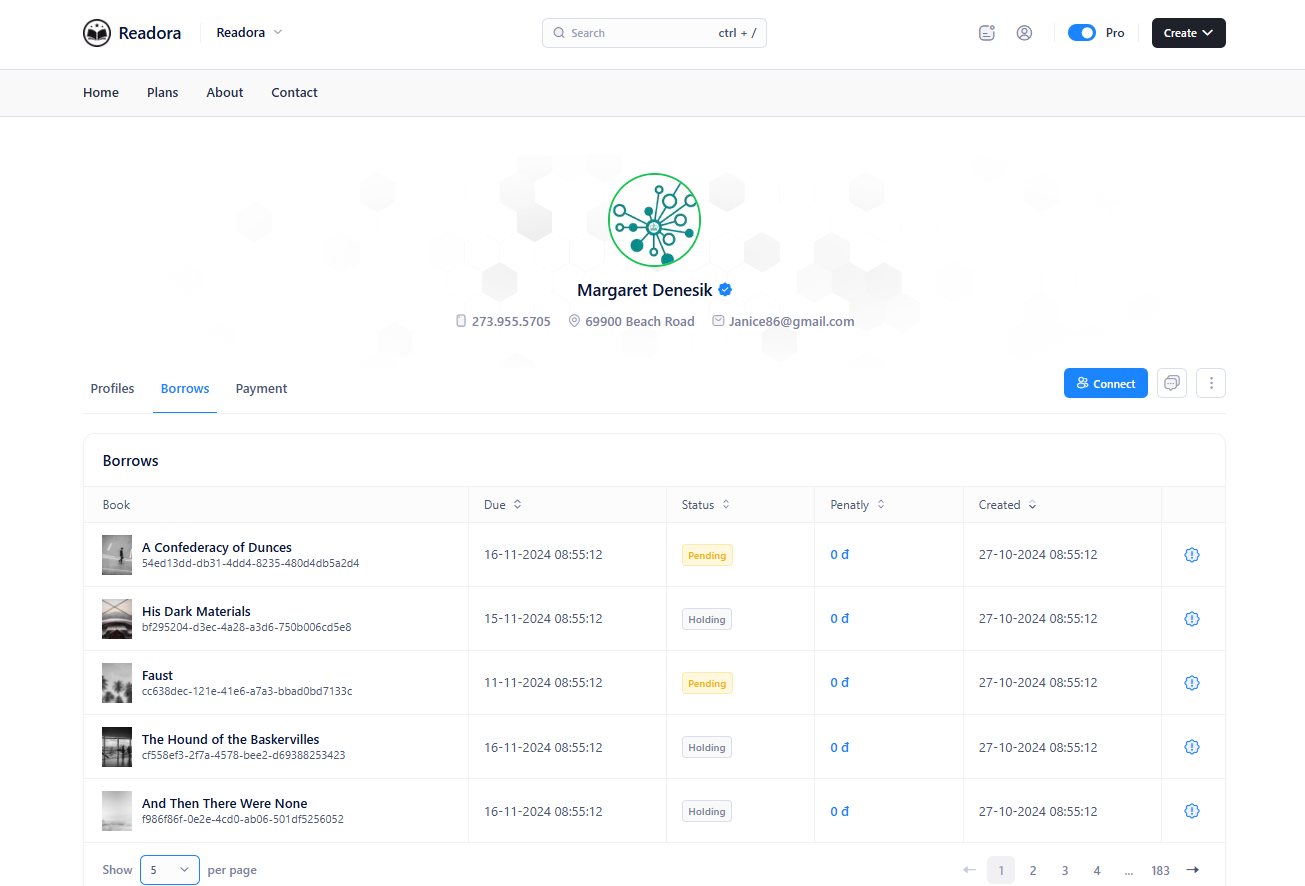
#### 3.4.2 View Borrowing History

**Function Trigger**:

This function is activated when the user accesses the "Borrows" tab on their profile page. The system loads and displays a list of all borrowing records for the user, including details such as the due date, current status, penalty, and creation date for each borrowing record.

**Function Description**:

* **Actors/Roles**: Users, System.
* **Purpose**:  
  This feature enables users to view their borrowing history, providing them with essential details on each borrowed book. It helps users keep track of due dates, check the current status of each borrowed item, and view any applicable penalties.
* **Interface**:
  + **Borrowing List:** Displays a list of all books borrowed by the user, showing the following details
    - **Book**: The title and cover image of the book.
    - **Due**: The due date by which the book should be returned.
    - **Status**: The current status of the borrowing record (e.g., Pending, Holding).
    - **Penalty**: Any applicable penalties for late returns, displayed as a numeric value.
    - **Created**: The date and time when the borrowing record was created.
  + **Navigation and Pagination:** Pagination controls at the bottom of the list allow users to navigate through their borrowing history if there are multiple pages of records.
* **Data Processing**:  
  The system retrieves borrowing data from the library’s database for the logged-in user. It displays this data in a structured table format, updating in real time to reflect the current status of each borrowing record, such as penalties for overdue items.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **View Borrowing History** screen:



#### 

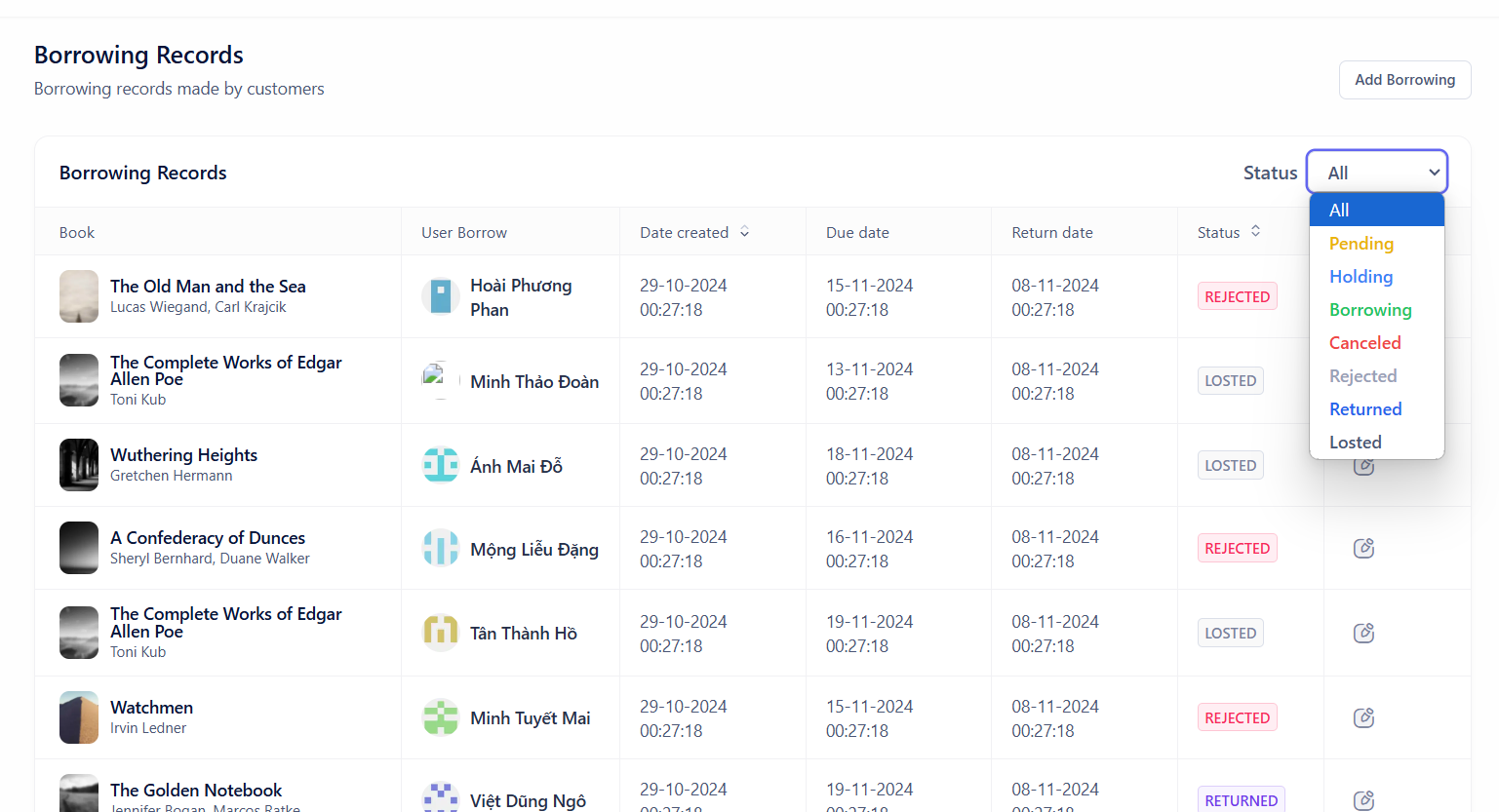
#### 3.4.2 View Borrowing History (Librarian)

**Function Trigger**:

This function is triggered when a librarian accesses the "Borrowing Records" section through their dashboard. The system loads and displays a list of all borrowing records, along with tools to manage and filter these records.

**Function Description**:

* **Actors/Roles**: Librarians
* **Purpose**:  
  This feature enables librarians to view, manage, and filter borrowing records. It allows librarians to monitor the status of borrowed books, check due dates, and manage the borrowing process efficiently.
* **Interface**:
  + **Borrowing Records Table**: Displays a detailed list of all borrowing records with the following columns:
    - **Book**: Title and cover image of the borrowed book.
    - **User Borrower**: Name and avatar of the user who borrowed the book.
    - **Date Created**: The date and time the borrowing record was created.
    - **Due Date**: The date by which the book is expected to be returned.
    - **Return Date**: The actual date the book was returned (if applicable).
    - **Status**: The current status of the borrowing record, with status options including:
      * **Pending**: Waiting for approval or further action.
      * **Holding**: The book is reserved for the user but not yet picked up.
      * **Borrowing**: The book is currently on loan to the user.
      * **Canceled**: The borrowing request was canceled.
      * **Rejected**: The borrowing request was rejected.
      * **Returned**: The book has been returned successfully.
      * **Losted**: The book is marked as lost.
  + **Status Filter Dropdown**: A dropdown menu that allows the librarian to filter borrowing records by status (e.g., All, Pending, Holding, Borrowing, Canceled, Rejected, Returned, Losted). This helps librarians focus on records with a specific status for more efficient management.
  + **Add Borrowing Button**: Located at the top right, this button allows librarians to manually create a new borrowing record, useful for cases where records need to be added directly by staff.
* **Data Processing**:  
  The system retrieves all borrowing records from the library’s database, displaying them in a structured table. When a librarian selects a filter option from the status dropdown, the system filters the records in real time to show only those with the selected status. Clicking "Add Borrowing" initiates a form where the librarian can input details for a new borrowing record.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **View Borrowing History (Librarian)** screen:



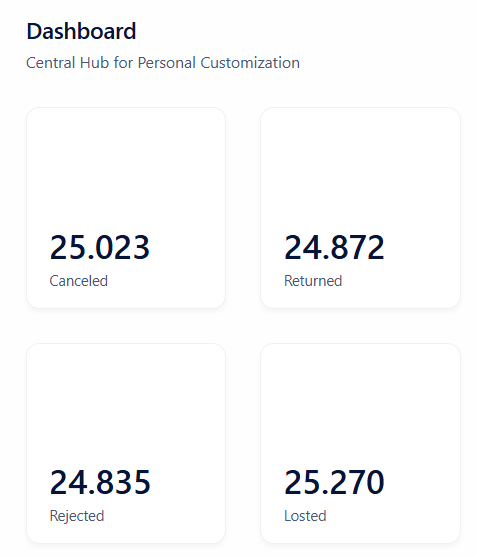
#### 3.4.3 Borrow Book Statistical

**Function Trigger**:

This function is triggered when the owner accesses the "Borrow Book Statistical" section from their dashboard. The system loads and displays key statistics related to the library's borrowing activities.

**Function Description**:

* **Actors/Roles**: Owner
* **Purpose**:  
  This feature provides the owner with a high-level overview of borrowing activity in the library, specifically focused on key metrics such as the number of canceled, returned, rejected, and lost borrow requests. It offers insights into user interactions with the borrowing system, highlighting trends and helping the owner identify areas for potential improvement.
* **Interface**:
  + **Statistical Summary Cards**: The dashboard displays four summary cards, each representing a different statistic related to book borrowing activities:
    - **Canceled**: Shows the total number of borrowing requests that were canceled.
    - **Returned**: Displays the total number of books that were successfully returned.
    - **Rejected**: Indicates the number of borrowing requests that were rejected.
    - **Losted**: Represents the number of books marked as lost.
* **Data Processing**:  
  The system aggregates data from the library's database to calculate the counts for each category (canceled, returned, rejected, and lost). These values are updated periodically or in real-time to reflect the latest statistics.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Borrow Book Statistical** screen:



### 3.5 Membership Card Management

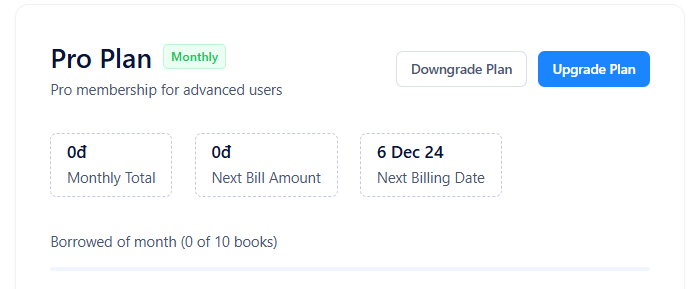
#### 3.5.1 View Membership Card Details

**Function Trigger**:

This function is triggered when a user accesses their "Membership Card" section in their profile. The system loads detailed information about the user's current membership card, including plan type, billing details, and borrowing allowance.

**Function Description**:

* **Actors/Roles**: Users, System.
* **Purpose**:  
  This feature allows users to view essential details about their membership card, including plan type, borrowing limits, billing cycle, and next billing date. It provides users with an overview of their membership status and options to upgrade or downgrade their membership.
* **Interface**:
  + **Membership Card Information**:
    - **Plan Type**: Indicates the current membership level (e.g., "Pro Plan") and billing frequency (e.g., "Monthly").
    - **Description**: Brief description, such as "Pro membership for advanced users."
    - **Upgrade/Downgrade Options**: Buttons for changing the membership plan, allowing users to upgrade or downgrade.
  + **Billing and Renewal Details**
    - **Monthly Total**: Shows the total duration of the current billing period, if applicable.
    - **Next Bill Amount**: Indicates the amount to be billed for the next cycle.
    - **Next Billing Date**: The date of the next billing cycle (e.g., "6 Dec 24")
  + **Borrowing Allowance**:
    - **Monthly Borrowing Limit**: Displays the user’s borrowing quota for the month and their current usage (e.g., "0 of 10 books borrowed this month").
    - **Borrowing Summary**: Helps users monitor how many books they can still borrow within the current month.
* **Data Processing**:  
  The system retrieves the user’s membership card data from the database, including plan type, billing amount, next billing date, and borrowing limits. If the user selects the "Upgrade" or "Downgrade" option, the system processes this request and updates their membership card accordingly.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **View Membership Card Details** screen:



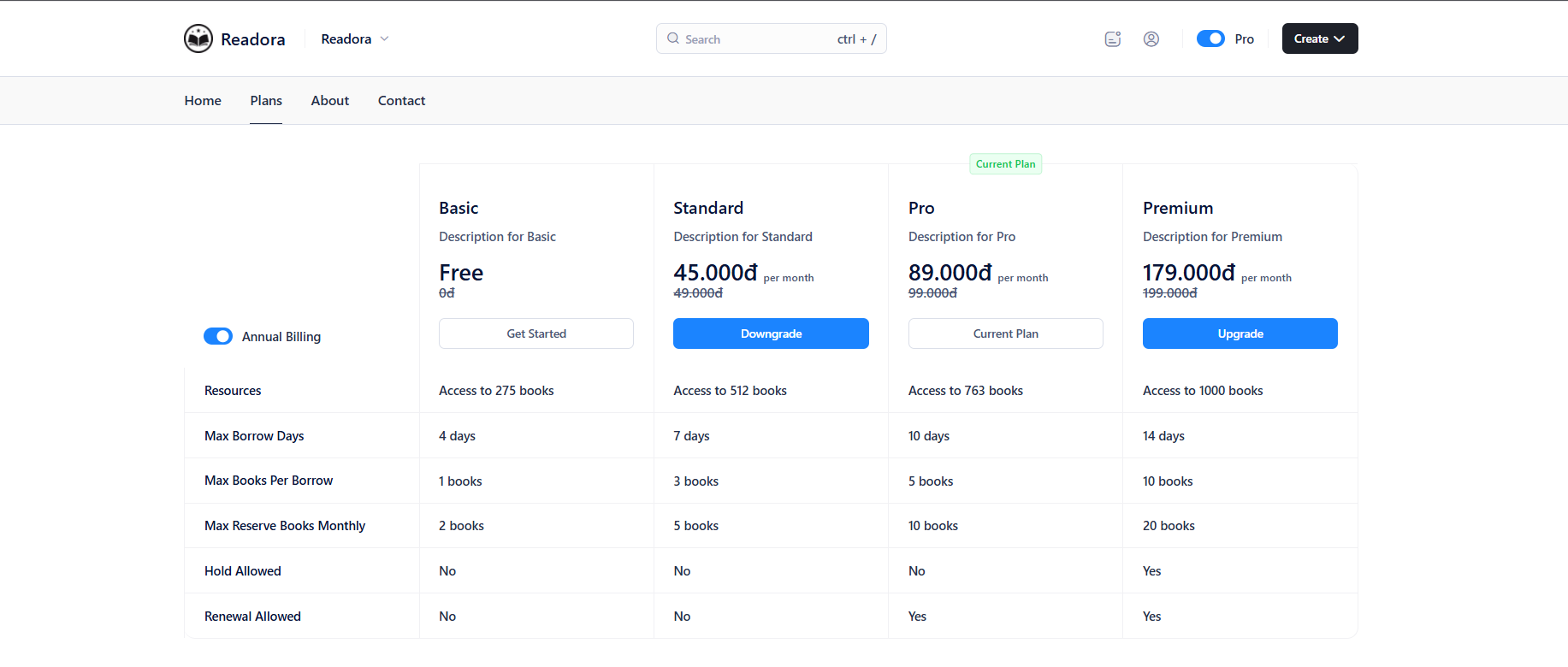
#### 3.5.2 Update Membership Card

**Function Trigger**:

This function is triggered when a user opens the "Membership" section on their profile and views the available membership plans. The system displays details about the user’s current membership plan and allows them to select and update their membership by upgrading or downgrading to other plans.

**Function Description**:

* **Actors/Roles**: Users, System.
* **Purpose**:  
  This feature enables users to update their membership card by selecting a new membership plan. It allows users to view and compare membership tiers, understand the benefits and limitations of each plan, and decide if they want to upgrade or downgrade their membership.
* **Interface**:
  + **Membership Plan Comparison:**
    - **Plan Name and Description**: Each plan has a name (e.g., Basic, Standard, Pro, Premium) and a short description that summarizes the target audience or key benefits of that plan.
    - **Pricing**: Monthly and, if applicable, annual subscription fees for each plan. Users can toggle between monthly and annual pricing options.
    - **Membership Benefits and Features**: Detailed information for each plan, including:
* **Access to Books**: Maximum number of books the user can access (e.g., Basic access to 175 books, Pro access to 748 books).
* **Max Borrow Days**: Maximum days a book can be borrowed under each plan.
* **Max Books per Borrow**: The number of books that can be borrowed simultaneously.
* **Max Reserved Books Monthly**: Number of books the user can reserve each month.
* **Hold Allowed**: Indicates whether the user is permitted to place holds on books.
* **Renewal Allowed**: Specifies if renewals for borrowed books are allowed.
  + **Current Plan Indicator**: The user’s current plan is clearly marked (e.g., with a "Current Plan" label) to distinguish it from other plans.
  + **Upgrade and Downgrade Options**:
    - **Upgrade Button**: Available next to higher-tier plans, allowing the user to upgrade to a more advanced plan.
    - **Downgrade Button**: Available next to lower-tier plans, allowing the user to downgrade if they want fewer features or a lower cost.
    - **Get Started Button**: For users currently on a trial or not subscribed, this button initiates a subscription to a selected plan.
  + **Billing Cycle Toggle**: Allows users to switch between monthly and annual billing cycles for each plan, letting them compare costs and choose their preferred payment frequency.
* **Data Processing**:  
  When a user selects a different membership plan (upgrade or downgrade), the system processes the change by updating the membership card information in the database. The system then adjusts borrowing limits, book access, reservation allowances, and billing settings according to the selected plan.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Update Membership Card** screen:



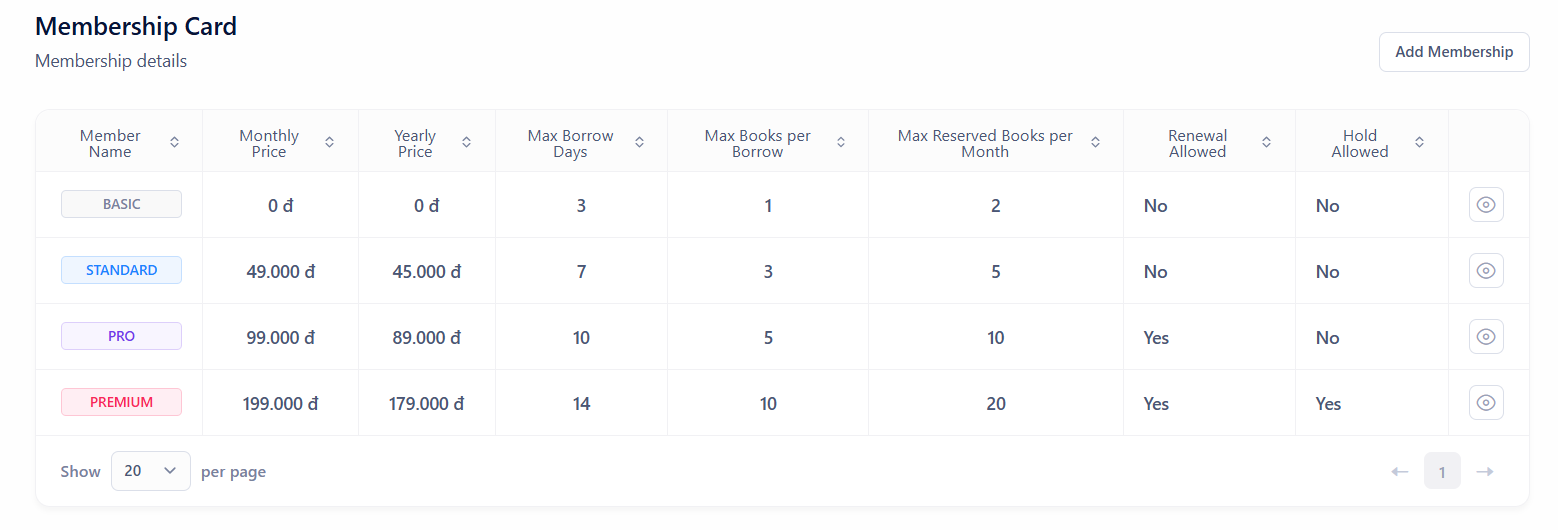
#### 3.5.3 Manage Membership Card List

**Function Trigger**:

This function is triggered when an admin accesses the "Membership Card List" section on their dashboard. The system loads all available membership plans, displaying relevant details for each plan in a table format.

**Function Description**:

* **Actors/Roles**: Administrators.
* **Purpose**:  
  This feature allows administrators to manage the library’s membership plans, including viewing all available membership tiers with specific benefits and pricing. Admins can add new plans, edit existing ones, or remove outdated plans, ensuring that the library offers appropriate membership options to users.
* **Interface**:
  + **Membership Card List Table**: Displays a table with details of each membership plan, including:
    - **Member Name**: The name of the membership plan (e.g., Basic, Standard, Pro, Premium).
    - **Monthly Price**: The monthly subscription cost for each plan.
    - **Yearly Price**: The annual subscription cost for each plan.
    - **Max Borrow Days**: The maximum number of days a user can borrow a book under each plan.
    - **Max Books per Borrow**: The maximum number of books a user can borrow at one time.
    - **Max Reserved Books per Month**: The maximum number of books a user can reserve each month.
    - **Renewal Allowed**: Indicates whether users on this plan are allowed to renew borrowed books.
    - **Hold Allowed**: Indicates whether users on this plan are allowed to place holds on books.
  + **Action Buttons**:
    - **Add Membership**: A button located at the top right that allows admins to add a new membership plan.
    - **Edit**: A button next to each membership plan for editing the plan’s details.
    - **Delete**: A button next to each membership plan that allows admins to remove the plan if it’s no longer offered.
  + **Pagination**: Pagination controls are located at the bottom of the table, allowing admins to navigate through multiple pages of membership plans if needed.
* **Data Processing**:  
  The system retrieves data for all membership plans from the database and displays it in a structured table format. When the admin clicks "Add Membership," "Edit," or "Delete," the system processes the request accordingly:
  + **Add**: Opens a form to enter details for a new membership plan.
  + **Edit**: Allows modifications to the selected plan’s details, saving updates to the database.
  + **Delete**: Prompts a confirmation dialog before removing the membership plan from the database.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Manage Membership Card List**  screen:



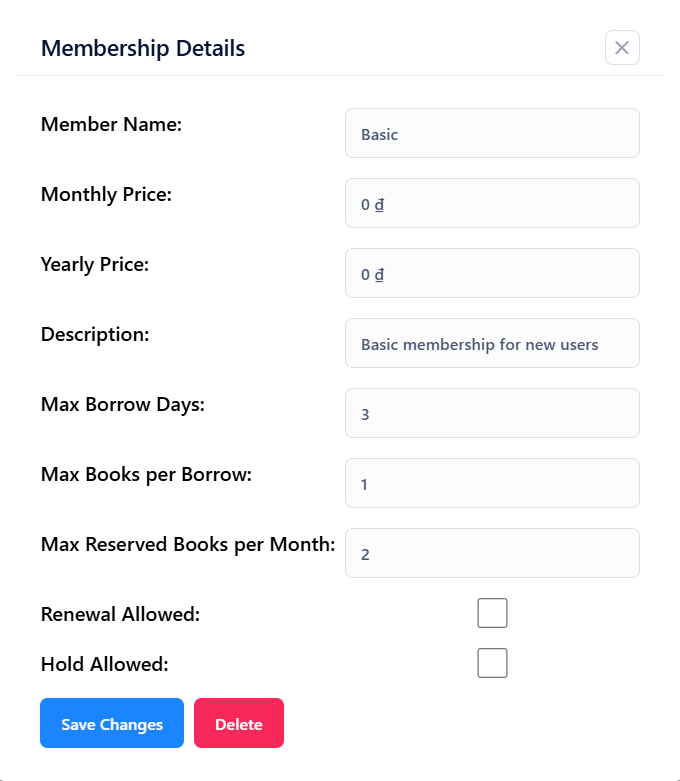
#### 3.5.4 View Detail / Update /Delete Membership Card

**Function Trigger**:

This function is triggered when an admin selects a membership plan from the membership management section to view its details. The system loads the selected membership card's details in an editable form, allowing the admin to make changes or delete the membership card.

**Function Description**:

* **Actors/Roles**: Administrators.
* **Purpose**:  
  This feature allows administrators to view, update, and delete membership card details. Admins can modify attributes such as pricing, borrowing limits, and permissions, or delete the membership card if it’s no longer needed.
* **Interface**:
  + **Membership Details Form**: The form displays detailed information about the selected membership card, including:
    - **Member Name**: The name of the membership plan (e.g., Basic).
    - **Monthly Price**: Editable field for the monthly price of the plan.
    - **Yearly Price**: Editable field for the yearly price of the plan.
    - **Description**: A brief description of the plan.
    - **Max Borrow Days**: The maximum number of days a book can be borrowed under this plan.
    - **Max Books per Borrow**: The maximum number of books a user can borrow at one time.
    - **Max Reserved Books per Month**: The maximum number of books a user can reserve each month.
    - **Renewal Allowed**: A checkbox indicating whether renewals are allowed for this plan.
    - **Hold Allowed**: A checkbox indicating whether holds are allowed for this plan.
  + **Action Buttons**:
    - **Save Changes**: A button to save any modifications made to the membership card details. When clicked, the system validates the input and updates the membership card in the database if valid.
    - **Delete**: A button to delete the membership card. When clicked, a confirmation prompt appears to ensure the admin wants to proceed with the deletion.
* **Data Processing**:  
  When the admin views the membership card, the system fetches its details from the database and displays them in the form. Upon clicking "Save Changes," the system validates the inputs and updates the membership information in the database. If the admin clicks "Delete," the system prompts for confirmation and, upon confirmation, removes the membership card from the database.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **View Detail / Update /Delete Membership Card**  screen:



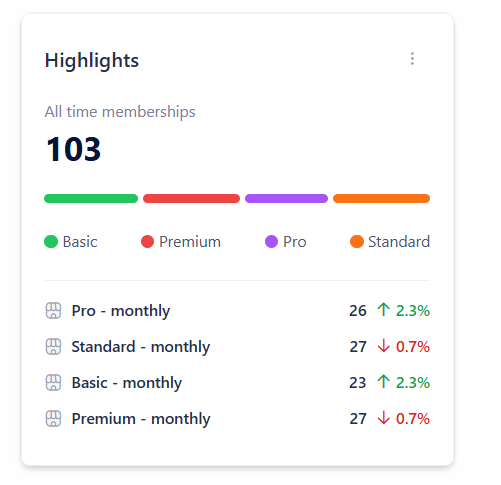
#### 3.5.5 Membership Card Statistical

**Function Trigger**:

This function is triggered when the owner accesses the "Membership Card Statistics" section on their dashboard. The system loads statistical data on all membership plans, providing insights into membership distribution and trends.

**Function Description**:

* **Actors/Roles**: Owner
* **Purpose**:  
  This feature provides the owner with a summary of membership plan usage, showing the total number of memberships and distribution across different membership types (e.g., Basic, Premium, Pro, Standard). It includes information on membership counts and percentage changes, helping the owner understand membership trends over time.
* **Interface**:
  + **Highlights Summary Card**:
    - **Total Memberships:** The total count of all-time memberships across all plans.
    - **Membership Distribution Bar:** A color-coded bar representing the proportion of each membership type:
      * **Basic:** Shown in green**.**
      * **Premium:** Shown in red.
      * **Pro:** Shown in purple.
      * **Standard:** Shown in orange.
  + **Detailed Membership Breakdown**:
    - **Membership Plan:** The name of the plan (e.g., Pro - monthly).
    - **Current Count:** The number of active memberships for that plan.
    - **Change Indicator:** A percentage indicating the growth or decline in memberships for each plan (e.g., 2.3% increase, 0.7% decrease), with an arrow pointing up or down to show the trend.
* **Data Processing**:  
  The system aggregates data on memberships from the database, calculating the total number of memberships, the distribution by type, and recent changes in membership counts. It then displays this data with percentage changes, calculated based on previous records.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Membership Card Statistical** screen:



### 3.6 Category Management

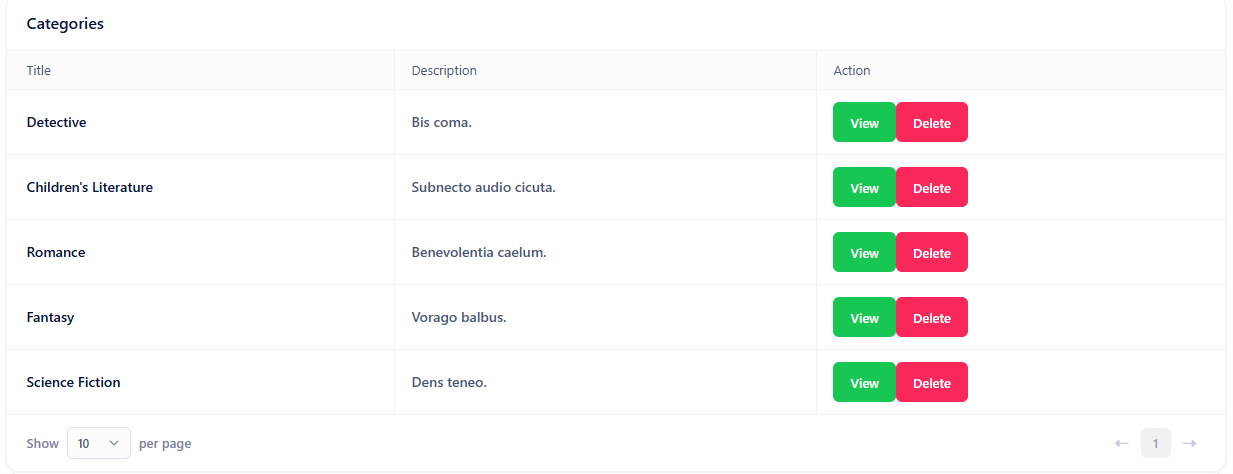
#### 3.6.1 View Category List

**Function Trigger**:

This function is triggered when an admin accesses the "Categories" section from their dashboard. The system loads all registered book categories in the library system, showing relevant details and allowing for management actions like viewing and deleting categories.

**Function Description**:

* **Actors/Roles**: administrators.
* **Purpose**:  
  This feature provides admins with tools to manage the library’s book categories. Admins can view an organised list of all categories, inspect detailed information, and delete categories as needed to keep the library's catalogue organised.
* **Interface**:
  + **Category List**:
    - **Title**: The name of the category (e.g., Detective, Children's Literature).
    - **Description**: A brief description of the category.
    - **Action Buttons**
      * **View**: Allows the admin to view detailed information about the selected category, such as books associated with it.
      * **Delete**: Enables the admin to delete a category that is no longer relevant in the library system. The system may ask for confirmation before processing the deletion.
  + **Pagination Controls**: Pagination controls at the bottom allow the admin to navigate through the category list if there are multiple pages.
* **Data Processing**:  
  The system retrieves category data from the database and displays it in a structured table. When the admin clicks "View," the system fetches detailed information about the selected category. If the admin clicks "Delete," the system verifies the action with a confirmation prompt and processes the deletion, removing the category from the database.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **View Category List** screen:



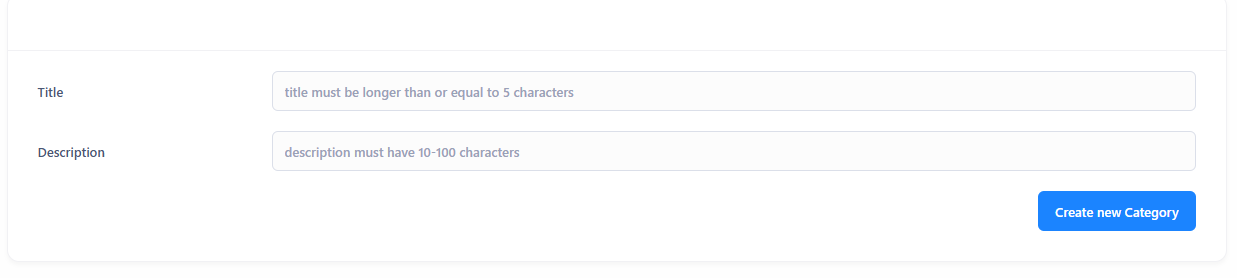
#### 3.6.2 Create New Category

**Function Trigger**:

This function is triggered when an admin accesses the "Create New Category" section in the Category Management area of their dashboard. The system provides a form to enter information for creating a new category.

**Function Description**:

* **Actors/Roles**: administrators.
* **Purpose**:  
  This feature enables administrators to add new categories to the library's collection, organising books more effectively by genre, subject, or other criteria. It allows for better management and categorization of the library’s inventory.
* **Interface**:
  + **Category Creation Form**:
    - **Title**: A text input field for the category name. Validation requires the title to be at least 5 characters long.
    - **Description**: A text input field for a brief description of the category. Validation requires the description to be between 10 and 100 characters in length.
  + **Create Button**: A "Create New Category" button at the bottom of the form. When clicked, the system validates the input fields and, if valid, submits the new category information to the database.
* **Data Processing**:  
  When the admin fills out the form and clicks "Create New Category," the system checks the inputs to ensure they meet the required character lengths. If validation passes, the system processes the request by adding the new category to the library database. If validation fails, the system displays an error message prompting the admin to correct the inputs.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Create New Category** screen:



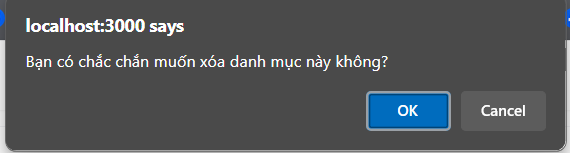
#### 3.6.3 Delete Category

**Function Trigger**:

This function is triggered when an admin clicks the "Delete" button next to a category in the Category Management section. Upon clicking, a confirmation dialog appears to ensure that the admin intends to proceed with the deletion.

**Function Description**:

* **Actors/Roles**: administrators.
* **Purpose**:  
  This feature allows administrators to remove a category from the library system. The confirmation dialog helps prevent accidental deletions by prompting the admin to confirm their action.
* **Interface**:
  + **Delete Confirmation Dialog**:
    - **Message**: "Bạn có chắc chắn muốn xóa danh mục này không?" (Are you sure you want to delete this category?).
    - **Buttons:**
* **OK**: Confirms the deletion. If clicked, the system proceeds with deleting the category.
* **Cancel**: Cancels the action, allowing the admin to keep the category without any changes.
* **Data Processing**:  
  When the admin clicks "Delete" for a category, the system triggers the confirmation dialog. If the admin confirms by clicking "OK," the system deletes the category from the database. If the admin clicks "Cancel," the deletion process is aborted, and no changes are made to the database.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Delete Category** screen:



### 3.7 Payment Management

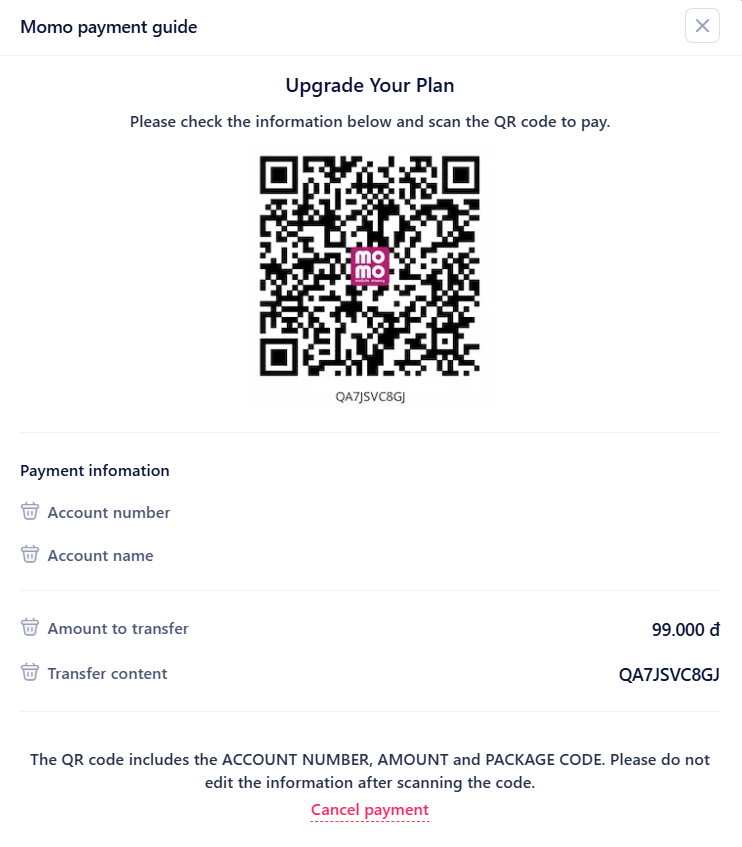
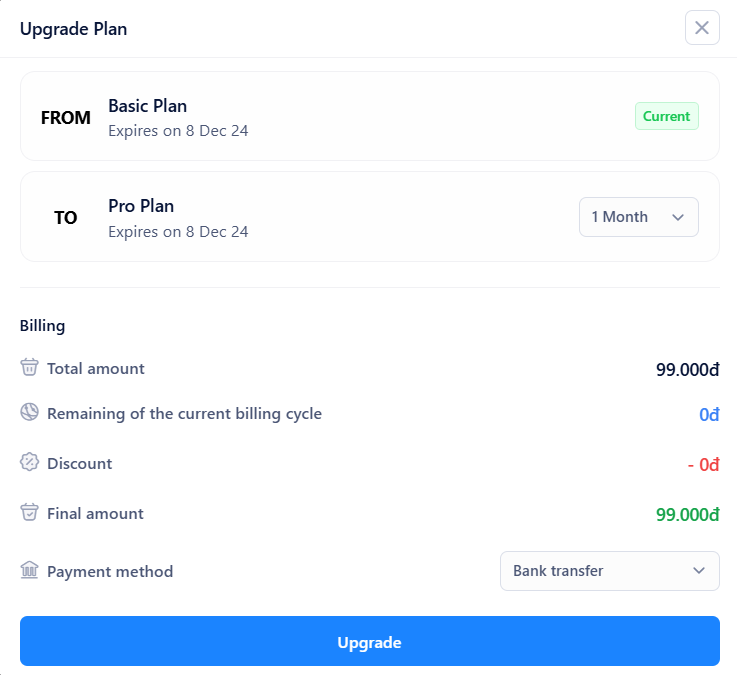
#### 3.7.1 Payment

**Function Trigger**:

This function is triggered after the user confirms an upgrade to a new membership plan and is ready to proceed with the payment. The system generates the necessary payment details and presents the user with instructions to complete the transaction.

**Function Description**:

* **Actors/Roles**: Users.
* **Purpose**:  
  This feature enables users to complete payment for a membership upgrade. It provides users with a clear summary of the billing details, along with a payment method (QR code) for convenient transaction processing through platforms like Momo.
* **Interface**:
  + **Billing Summary Screen**:
    - **From**: Displays the current membership plan and its expiration date.
    - **To**: Shows the new membership plan selected by the user for the upgrade.
    - **Billing Details**:
      * **Total Amount:** The full cost of the new membership plan.
      * **Remaining of the Current Billing Cycle:** Any remaining credits from the current billing period.
      * **Discount:** Any discounts applied to the upgrade.
      * **Final Amount:** The total amount the user needs to pay after adjustments
    - **Payment Method:** Displays the selected payment method, such as "Bank Transfer" or "Momo."
    - **Upgrade Button:** After reviewing, the user clicks "Upgrade" to proceed to the payment screen.
  + **Payment Screen (Momo QR Code Payment)**:
    - **QR Code:** A QR code is generated for the user to scan using the Momo app, allowing them to easily complete the payment.
    - **Payment Information:**
      * **Account Number:** The account number associated with the payment.
      * **Account Name:** The account holder’s name.
      * **Amount to Transfer:** The total amount the user needs to pay for the upgrade.
      * **Transfer Content:** A unique code that the user must include in the transaction details to identify their payment.
    - **Cancel Payment:** A "Cancel Payment" button allows the user to cancel the payment process if they choose not to proceed.
* **Data Processing**:  
  The system calculates the total payment required for the upgrade, taking into account any remaining balance and discounts. It then generates a QR code specific to the transaction, containing all necessary payment details. Once the user completes the payment, the system verifies the transaction and updates the user’s membership status.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Payment** screen:



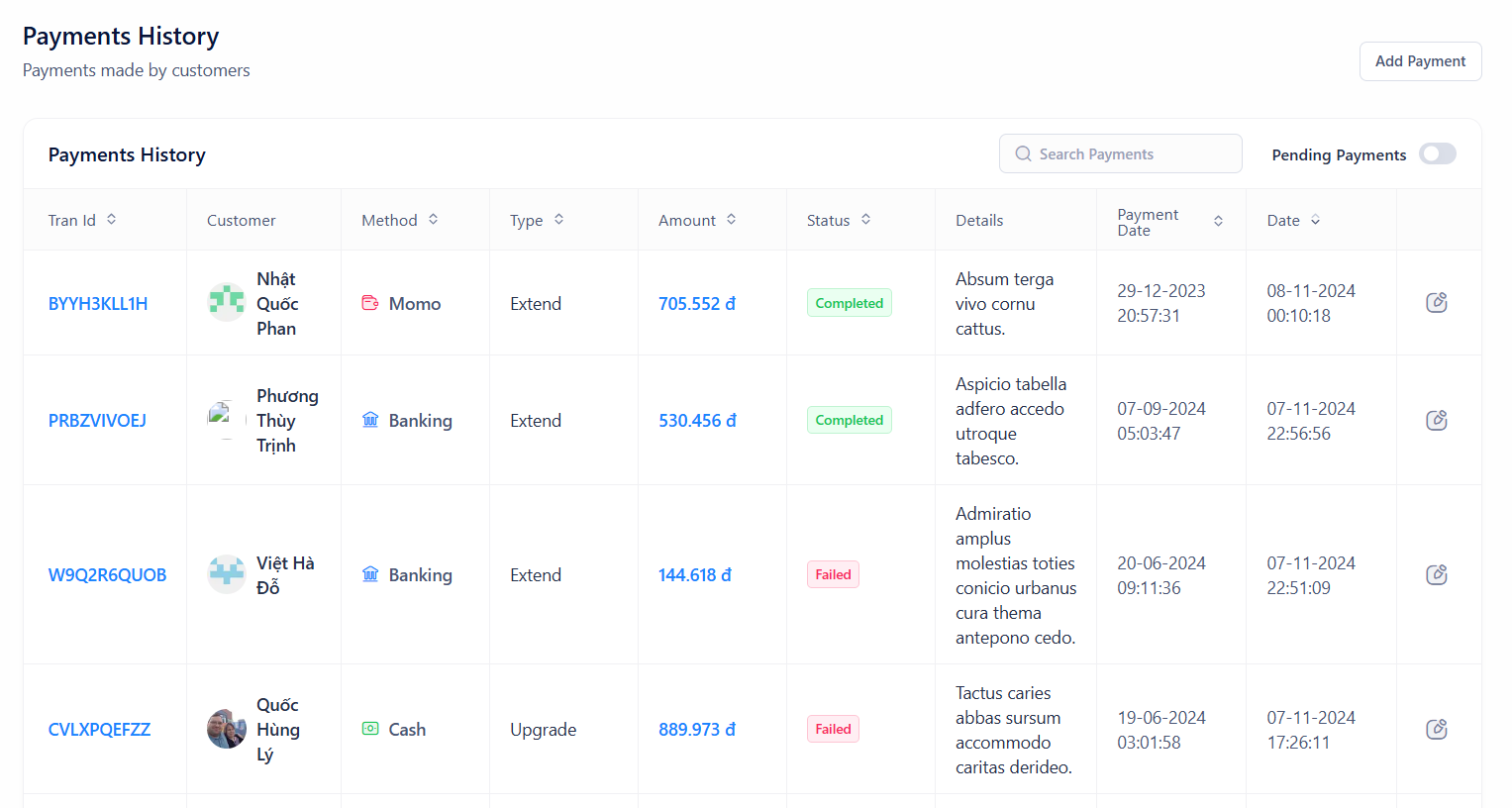
#### 3.7.2 View Payment History

**Function Trigger**:

This function is triggered when an admin accesses the "Payments History" section in their dashboard. The system loads and displays a detailed record of all payments made by customers, including both successful and failed transactions.

**Function Description**:

* **Actors/Roles**: Administrators.
* **Purpose**:  
  This feature allows admins to monitor and manage customer payments. It provides a comprehensive view of all payment transactions, including payment method, status, and transaction details. This information helps admins oversee financial activities related to memberships, upgrades, and other transactions.
* **Interface**:
  + **Payments History Table**:
    - **Tran ID**: A unique transaction ID for each payment, allowing easy reference.
    - **Customer**: The name and avatar of the customer who made the payment.
    - **Method**: The payment method used (e.g., Momo, Banking, Cash).
    - **Type**: The type of transaction, such as "Extend" or "Upgrade" of membership.
    - **Amount**: The payment amount.
    - **Status**: The current status of the payment, indicating if it’s "Completed" or "Failed."
    - **Details**: A brief description or note associated with the transaction.
    - **Payment Date**: The date and time when the payment was processed.
    - **Date**: The date and time of the transaction record.
  + **Search and Filter Options**:
    - **Search Payments**: A search bar at the top right, allowing the admin to search for specific transactions by transaction ID, customer name, or other criteria.
    - **Pending Payments Toggle**: A toggle switch to filter the list and display only pending payments, making it easier to manage incomplete or pending transactions.
* **Data Processing**:  
  The system retrieves payment records from the database and displays them in a structured format. Admins can use the search function to locate specific transactions or filter the list to show pending payments only. If an admin needs to add a payment manually, the system provides a form for entering transaction details, which then updates the database accordingly.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **View Payment History** screen:



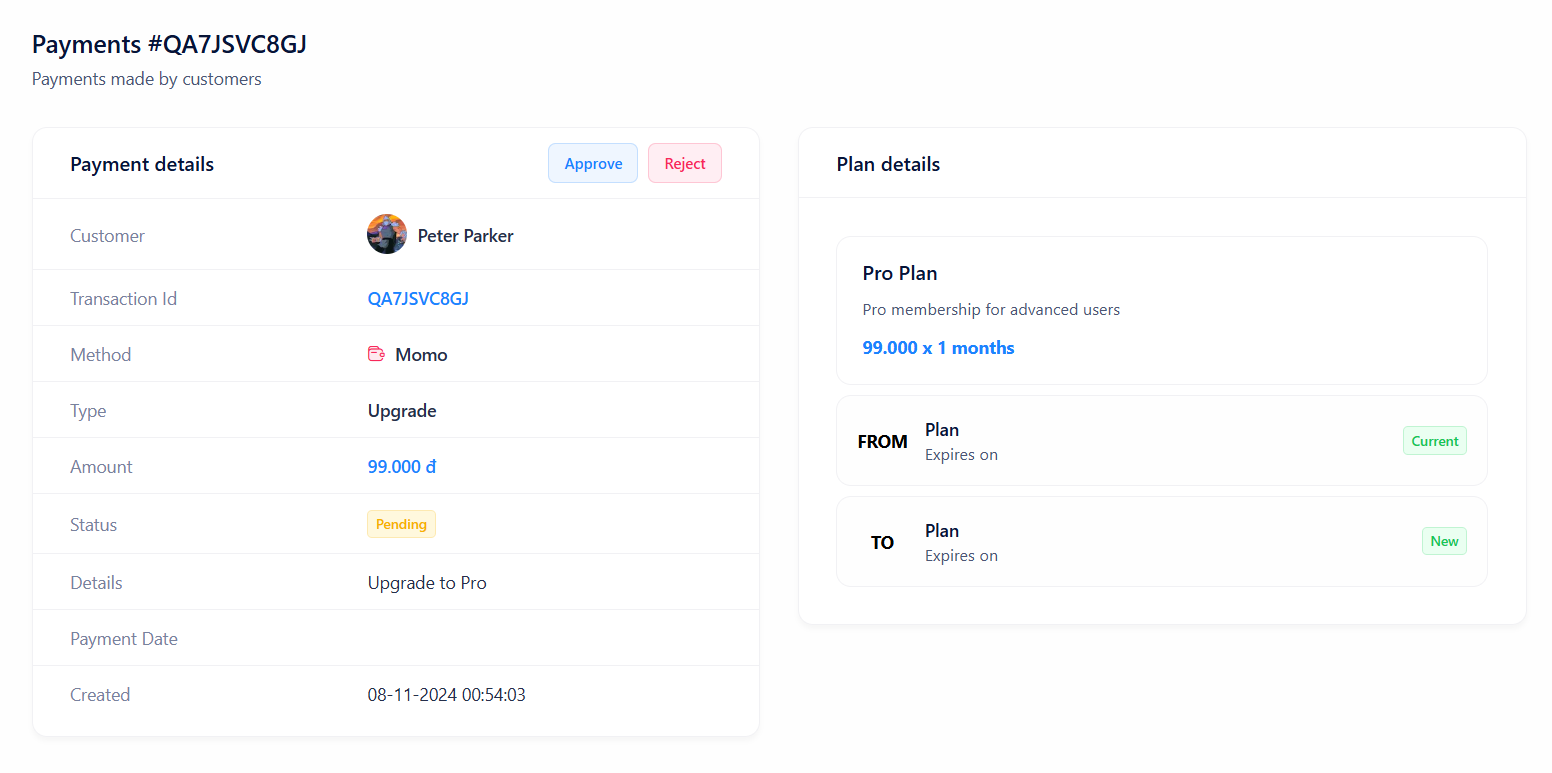
#### 3.7.3 Approve / Reject Payment

**Function Trigger**:

This function is triggered when an admin selects a specific payment from the payments history to review its details. The system displays the payment information and provides options to either approve or reject the payment.

**Function Description**:

* **Actors/Roles**: Administrators.
* **Purpose**:  
  This feature allows admins to review and manage pending payment transactions submitted by users. Admins can verify the payment details and decide to approve or reject the transaction based on the validity of the payment.
* **Interface**:
  + **Payment Details Section**:
    - **Customer:** Displays the customer’s name and avatar who made the payment.
    - **Transaction ID:** A unique identifier for the payment transaction.
    - Method: The payment method used (e.g., Momo).
    - **Type:** The type of transaction, such as "Upgrade" for upgrading the membership plan.
    - **Amount:** The amount paid by the customer.
    - **Status:** The current status of the payment, which is set to "Pending" initially.
    - **Details:** Any additional information regarding the payment, such as "Upgrade to Pro."
    - **Payment Date:** The date and time when the payment was processed.
    - **Created:** The date and time when the payment record was created in the system.
  + **Plan Details Section**:
    - **Plan Overview**: Displays details of the membership plan associated with the payment, including the name, description, and pricing.
    - **From**: Shows the current plan and its expiration date.
    - **To**: Displays the new plan and the future expiration date if the upgrade is approved.
  + **Action Buttons**:
    - **Approve**: A blue button that allows the admin to confirm and approve the payment, updating the user’s membership plan accordingly.
    - **Reject**: A red button that allows the admin to reject the payment if there is an issue with the transaction, such as an invalid payment or incorrect amount.
* **Data Processing**:  
  When the admin clicks "Approve," the system updates the payment status to "Completed," applies the membership upgrade, and adjusts the expiration date for the new plan. If the admin clicks "Reject," the system changes the payment status to "Rejected" and keeps the user’s current plan unchanged. Both actions are recorded in the system for auditing purposes.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Approve / Reject Payment** screen:



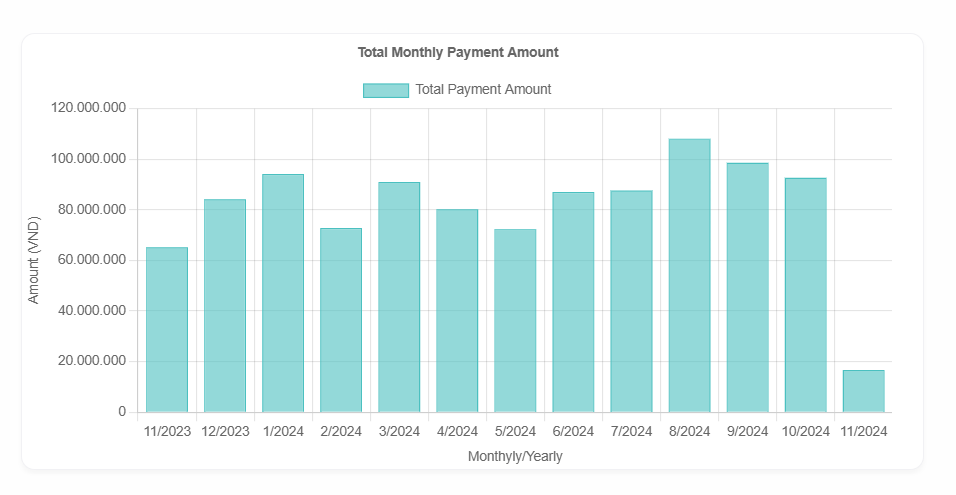
#### 3.7.4 Payment Statistical

**Function Trigger**:

This function is triggered when the owner accesses the "Payment Statistics" section in their dashboard. The system retrieves and displays a monthly summary of total payment amounts over a specified period.

**Function Description**:

* **Actors/Roles**: Owner.
* **Purpose**:  
  This feature provides the owner with a visual overview of the total payment amounts received each month. It helps the owner analyze revenue trends, assess financial performance, and identify monthly patterns in payments.
* **Interface**:
  + **Total Monthly Payment Amount Chart**:
    - **Chart Type**: Bar chart that displays the total payment amount for each month.
    - **X-Axis**: Represents the months and years (e.g., 11/2023, 12/2023, 1/2024).
    - **Y-Axis**: Represents the payment amount in Vietnamese Dong (VNĐ), scaled to show the range of monthly totals.
    - **Legend**: Includes a label "Total Payment Amount" with a color indicator matching the bars in the chart for easy reference.
* **Data Processing**:  
  The system aggregates payment data from the database, calculating the total amount received for each month. It then generates a bar chart to visually represent these monthly totals. This data may be refreshed periodically to reflect the latest payment transactions.
* **Screen Layout**: Mockup of the screen layout, as shown below for the **Payment Statistical** screen:

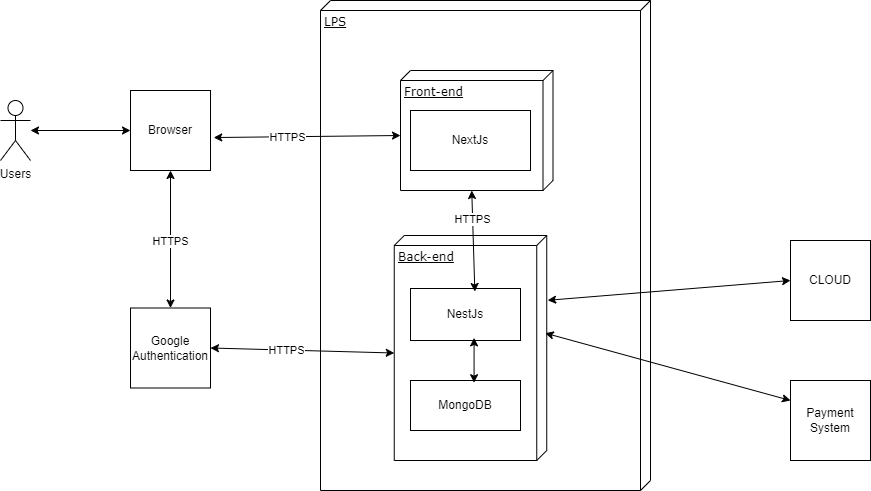


# IV. Software Design Description

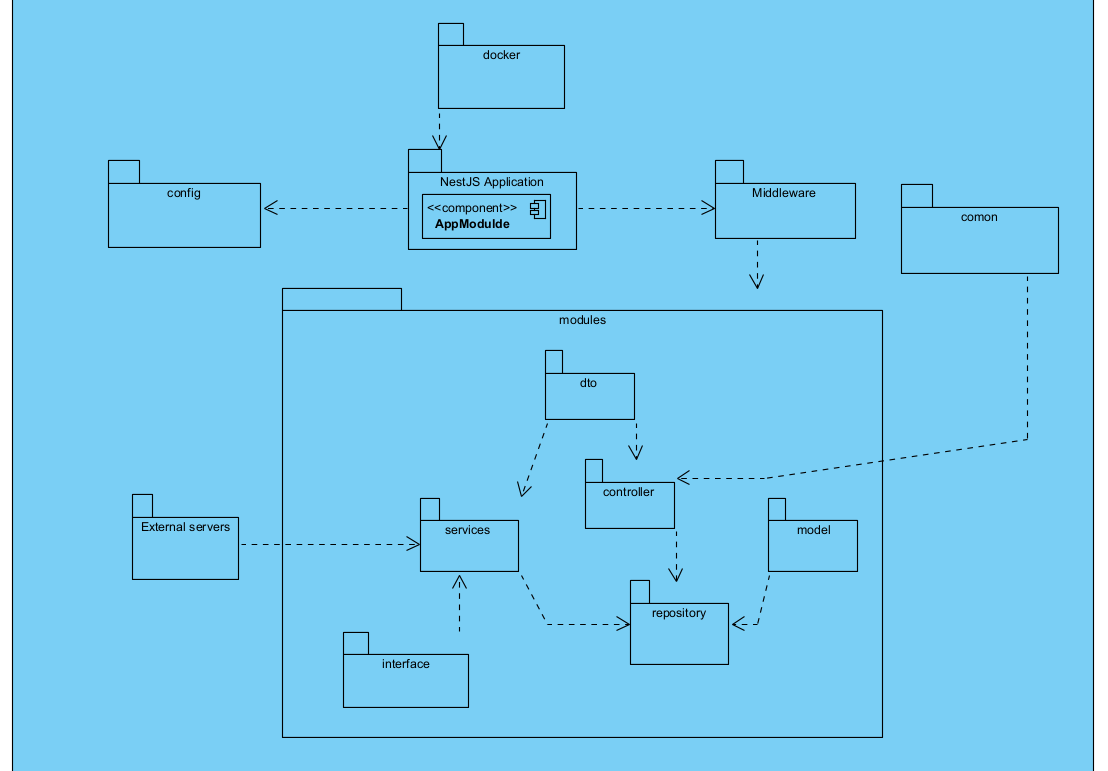
*[Provide final software design information follow the template as part II in the Report #4]*

## 1. System Design

### 1.1 System Architecture

**

### 1.2 Package Diagram



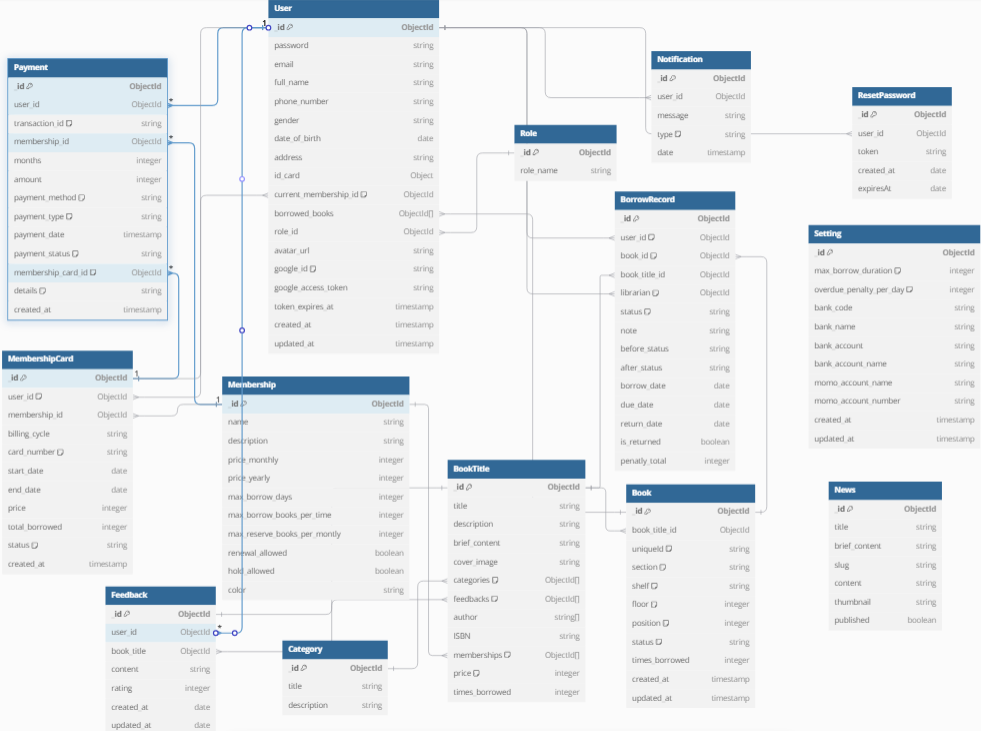
***Package Descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | **Docker & CI/CD**: | **Docker**: Packages the application into containers to ensure a consistent environment across servers.  **CI/CD Pipeline**: Automates the processes of testing, building, and deploying the application.   * **Build**: Generates an application build from the source code. * **Test**: Runs automated tests to ensure the quality of the source code. * **Deploy**: Deploys the application to the production or staging environment. |
| 02 | **NestJS Application**: | **AppModule**: The root module of the application, containing main configurations and initializing other components like controllers, services, and middleware.  **Middleware**: Handles tasks before requests reach the controllers, such as logging or security checks. |
| 03 | **Application** | **Controller:** Handles HTTP requests from users and calls services to execute business logic.  **Service:** Executes the business logic of the application, processes data, and interacts with repositories.  **Repository:** Retrieves and stores data from the database (MongoDB) as needed by the application.  **DTO (Data Transfer Object):** Used to transfer data between layers of the application, ensuring data validity and structure.  **Interface:** Defines data types and structures to ensure consistency across the components**.** |
| 04 | **Common** | **ExceptionFilter:** Handles errors during request processing, returning appropriate error responses to the user.  **AuthGuard:** Protects routes that require access control, ensuring users are authenticated and authorized to access resources. |
| 05 | **Config** | **AppConfig:** Holds general application settings such as port, environment, and security configurations.  **Database Config:** Manages database connection settings (MongoDB), providing necessary credentials and access info.  **MongoDB:** The database used to store the application's data. |
| 06 | **External Server:** | **Payment:** A third-party payment service (Payment Gateway), invoked by services within the application when processing financial transactions.  **Gmail:** Gmail service used to send email notifications, called from the Service when sending email alerts to users. |

## 

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | **Pages** | Contains all the routes/pages of the application (e.g., Home, About). Each file inside this directory represents a route based on its file name. |
| 02 | **Components** | Holds reusable UI components such as buttons, headers, and footers used across different pages. |
| 03 | **API** | Contains backend logic and API routes. Files in this folder correspond to server-side API routes handled by Next.js. |
| 04 | **StaticAssets** | Stores static files such as images, icons, and fonts that can be directly accessed via URL. |
| 05 | **Utils** | Contains utility functions that can be used throughout the project (e.g., date formatting, form validation). |
| 06 | **Styles** | **Contains global and module-based CSS/SCSS files for styling the application.** |

## 2. Database Design

**

### 2.1 User

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each user. |
| 2 | password | String | Required | Password for the user account. |
| 3 | email | String | Required | Email address of the user. |
| 4 | full\_name | String | Required | Full name of the user. |
| 5 | phone\_number | String | Required | Contact phone number of the user. |
| 6 | gender | String |  | Gender of the user (e.g., male, female, other). |
| 7 | date\_of\_birth | Date |  | Birthdate of the user. |
| 8 | address | String |  | Residential address of the user. |
| 9 | id\_card | ObjectId | Foreign Key | Reference to the user's ID card. |
| 10 | current\_membership\_id | ObjectId | Foreign Key | Reference to the user's current membership level. |
| 11 | borrowed\_books | ObjectId[] | Foreign Key | List of book IDs borrowed by the user. |
| 12 | role\_id | ObjectId | Foreign Key | Role assigned to the user (e.g., admin, member). |
| 13 | avartar\_url | String |  | URL to the user's profile or avatar image. |
| 14 | google\_id | String |  | ID for Google account integration. |
| 15 | google\_access\_token | String |  | Access token for Google account authorization. |
| 16 | token\_expires\_at | Datetime |  | Expiration time for the Google access token. |
| 17 | created\_at | Datetime |  | Timestamp for when the user account was created. |
| 18 | updated\_at | Datetime |  | Timestamp for the last update of the user record. |

### 

### 2.2 Role

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each role. |
| 2 | role\_name | String | Required | Name of each role. |

### 

### 2.3 Book

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each book. |
| 2 | book\_title\_id | ObjectId | Foreign Key | Reference to the title or information about the book. |
| 3 | uniqueId | String | Required | Unique identifier for the book (e.g., ISBN or library code). |
| 4 | section | Number | Required | Section where the book is located in the library. |
| 5 | shelf | String |  | Shelf where the book is stored within the section. |
| 6 | floor | Integer |  | Floor number where the book is located in the library. |
| 7 | position | Integer |  | Position on the shelf for the book's exact location. |
| 8 | status | String |  | Current status of the book (e.g., available, borrowed, reserved). |
| 9 | time\_borrowed | Integer |  | Duration (in days) the book has been borrowed. |
| 10 | created\_at | Datetime |  | Timestamp for when the book record was created. |
| 11 | updated\_at | Datetime |  | Timestamp for the last update of the book record. |

### 

### 2.4 Book Title

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each book title record. |
| 2 | title | String | Required | Title of the book, referencing the booking record if applicable. |
| 3 | description | String | Required | Detailed description of the book. |
| 4 | brief\_content | String | Required | Brief summary or content of the book. |
| 5 | cover\_image | String |  | URL to the cover image of the book. |
| 6 | categories | ObjectId[] | Foreign Key | List of category IDs associated with the book. |
| 7 | feedbacks | ObjectId[] | Foreign Key | List of feedback or review IDs related to the book. |
| 8 | author | String[] | Required | List of authors of the book. |
| 9 | ISBN | String | Required | International Standard Book Number (ISBN) of the book. |
| 10 | memberships | ObjectId[] | Foreign Key | List of membership IDs that have access to this book. |
| 11 | price | Integer |  | Price of the book. |
| 12 | times\_borrowed | Integer |  | Number of times this book has been borrowed. |

### 

### 2.5 Category

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each category record. |
| 2 | title | String | Required | Title of the category. |
| 3 | description | String | Required | Detailed description of the category. |

### 

### 2.6 BorrowRecord

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each borrowed record. |
| 2 | user\_id | ObjectId | Foreign Key | Reference to the user who borrowed the book. |
| 3 | book\_id | ObjectId | Foreign Key | Reference to the specific book being borrowed. |
| 4 | book\_title\_id | ObjectId | Foreign Key | Reference to the title or general information of the book. |
| 5 | librarian | ObjectId | Foreign Key | Reference to the librarian managing this borrow transaction. |
| 6 | status | String |  | Current status of the borrow record (e.g., pending, borrowed). |
| 7 | note | String |  | Additional notes related to the borrow transaction. |
| 8 | before\_status | String |  | Status of the borrow record before any recent change. |
| 9 | after\_status | String |  | Status of the borrow record after the recent change. |
| 10 | borrow\_date | Date | Required | Date when the book was borrowed. |
| 11 | due\_date | Date | Required | Due date for returning the book. |
| 12 | return\_date | Date | Required | Date when the book was actually returned. |
| 13 | is\_returned | Boolean | Required | Indicates if the book has been returned (true or false). |
| 14 | penalty\_total | Integer |  | Total penalty (if any) for late return or damages. |

### 

### 2.7 Payments

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each payment record. |
| 2 | user\_id | ObjectId | Foreign Key | Reference to the user making the payment. |
| 3 | transaction\_id | ObjectId | Foreign Key | Reference to the transaction related to the payment. |
| 4 | membership\_id | ObjectId | Foreign Key | Reference to the membership being paid for. |
| 5 | months | Integer | Required | Number of months the membership is being extended or paid for. |
| 6 | amount | Integer | Required | Total amount paid. |
| 7 | payment\_method | String | Required | Method of payment (e.g., credit card, cash, bank transfer). |
| 8 | payment\_type | String | Required | Type of payment (e.g., one-time, recurring). |
| 9 | payment\_date | Datetime | Required | Date and time of the payment. |
| 10 | payment\_status | String | Required | Status of the payment (e.g., completed, pending, failed). |
| 11 | membership\_card\_id | ObjectId | Foreign Key | Reference to the membership card used in this payment (if any). |
| 12 | details | String |  | Additional details or notes regarding the payment. |
| 13 | created\_at | Datetime | Required | Timestamp for when the payment record was created. |

### 

### 2.8 MembershipCard

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each membership card record. |
| 2 | user\_id | ObjectId | Foreign Key | Reference to the user associated with the membership card. |
| 3 | mebership\_id | ObjectId | Foreign Key | Reference to the membership type associated with this card. |
| 4 | billing\_cycle | String | Required | Billing cycle for the membership (e.g., monthly, annually). |
| 5 | card\_number | String |  | Unique card number assigned to the membership card. |
| 6 | start\_date | Date | Required | Start date of the membership. |
| 7 | end\_date | Date | Required | Expiration or end date of the membership. |
| 8 | price | Integer | Required | Cost of the membership for the specified billing cycle. |
| 9 | total\_borrowed | Integer | Required | Total number of items borrowed under this membership card. |
| 10 | status | String |  | Status of the membership (e.g., active, inactive, suspended). |
| 11 | created\_at | Datetime | Required | Timestamp for when the membership card record was created. |

### 

### 2.9 MemberShip

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each membership record. |
| 2 | name | String | Required | Name of the membership type. |
| 3 | description | String |  | Description of the membership benefits or details. |
| 4 | price\_monthly | Integer | Required | Monthly price of the membership. |
| 5 | price\_yearly | Integer | Required | Yearly price of the membership. |
| 6 | max\_borrow\_days | Integer | Required | Maximum number of days allowed for borrowing books under this membership. |
| 7 | max\_borrow\_books\_per\_time | Integer | Required | Maximum number of books that can be borrowed at one time. |
| 8 | max\_borrow\_books\_per\_monthly | Integer | Required | Maximum number of books that can be borrowed in a month. |
| 9 | renewal\_allowed | Boolean |  | Indicates if book renewals are allowed (true or false). |
| 10 | hold\_allowed | Boolean |  | Indicates if placing holds on books is allowed (true or false). |
| 11 | colour | Boolean |  | Indicates if colour printing is allowed (true or false). |

### 

### 2.10 Feedback

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each feedback record. |
| 2 | user\_id | ObjectId | Foreign Key | Reference to the user who made the feedback. |
| 3 | book\_title | ObjectId | Foreign Key | Reference to the book title that the feedback is about. |
| 4 | content | String | Required | Content of the feedback or review text. |
| 5 | rating | Integer |  | Rating given to the book, typically on a scale (e.g., 1-5). |
| 6 | created\_at | Date |  | Timestamp for when the feedback was created. |
| 7 | updated\_at | Date |  | Timestamp for when the feedback was last updated. |

### 

### 2.11 Notification

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each notification record. |
| 2 | user\_id | ObjectId | Foreign Key | Reference to the user who will receive the notification. |
| 3 | message | String | Required | Content or text of the notification message. |
| 4 | type | String | Required | Type or category of the notification (e.g., alert, reminder). |
| 5 | date | Datetime |  | Date and time the notification was created or sent. |

### 

### 2.12 Reset Password

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each reset password record. |
| 2 | user\_id | ObjectId | Foreign Key | Reference to the user who requested the password reset. |
| 3 | token | String | Required | The reset token used for verifying the password reset request. |
| 4 | created\_at | Date | Required | Date and time when the reset password request was created. |
| 5 | expireAt | Date | Required | Date and time when the reset token expires. |

### 

### 2.13 News

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each news record. |
| 2 | title | String | Required | Title of the news article. |
| 3 | brief\_content | String | Required | A brief content or summary of the news article. |
| 4 | slug | String | Required | URL-friendly version of the title (typically used for routing). |
| 5 | content | String | Required | Full content of the news article. |
| 6 | thumbnail | String |  | URL to the thumbnail image for the news article. |
| 7 | published | Boolean |  | Status indicating whether the news article is published or not. |

### 

### 2.14 Setting

| **No.** | **Attribute** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | **id** | ObjectId | Primary Key | Unique ID for each setting record. |
| 2 | max\_borrow\_duration | Integer | Required | Maximum duration (in days) allowed for borrowing items. |
| 3 | overdue\_penalty\_per\_day | Integer | Required | Penalty amount per day for overdue items. |
| 4 | bank\_code | String | Required | Code for the bank used for transactions. |
| 5 | bank\_name | String | Required | Name of the bank used for transactions. |
| 6 | bank\_account | String | Required | Account number in the bank for payments. |
| 7 | bank\_account\_name | String | Required | Name associated with the bank account. |
| 8 | momo\_account\_name | String | Required | Name associated with the MoMo account for transactions. |
| 9 | momo\_account\_number | String | Required | MoMo account number for payments. |
| 10 | created\_at | Datetime |  | Timestamp when the setting record was created. |
| 11 | updated\_at | Datetime |  | Timestamp when the setting record was last updated. |

# V. Software Testing Documentation

*[Provide final software testing information follow the template as part II in the Report #5]*

## 1. Scope of Testing

**System Under Test**: Book Borrowing and Return System

**Scope of Testing**: This testing scope includes all core functionalities for book borrowing, returning, and managing user memberships in the library system. Testing will focus on verifying that both functional and non-functional requirements are met, with special attention to user interaction, data integrity, and system responsiveness.

1. **Functional Features to be Tested**:
   1. User borrowing process (e.g., book availability check, borrowing limit enforcement)
   2. Book return process (due date, penalties for overdue returns)
   3. Membership upgrade options and restrictions
   4. Notification and reminders for overdue books
   5. Viewing borrowing history
   6. Payment processing and history viewing for overdue fees or membership renewal
2. **Non-Functional Requirements**:
   1. Performance: Ensure system responsiveness for high user volumes.
   2. Security: Data encryption and restricted access to user data.
   3. Usability: Intuitive design and accessible borrowing/return processes.
   4. Reliability: High system uptime and accurate transaction logs.

**Stages/Levels of Testing**:

1. **Unit Testing**:
   1. **In-charge**: Development team
   2. **Focus**: Verifying individual components, such as borrowing and return functions, membership checks, and notification triggers.
   3. **Input/Time**: Unit test data on book records, user profiles, and membership constraints.
   4. **Acceptance Criteria**: Each component must perform as expected under normal and edge cases without errors.
2. **Integration Testing**:
   1. **In-charge**: Development and QA team
   2. **Focus**: Ensuring components work together, such as the interaction between the user, book database, and notification system.
   3. **Input/Time**: Simulated user activities that involve book borrowing/return workflows.
   4. **Acceptance Criteria**: Integrated processes run smoothly, with all modules communicating accurately and data transferring correctly between components.
3. **System Testing**:
   1. **In-charge**: QA team
   2. **Focus**: Testing the full workflow for end-users, including book borrowing, returning, membership upgrades, and notifications.
   3. **Input/Time**: Realistic scenarios including various user roles (member, librarian) and complex borrowing patterns.
   4. **Acceptance Criteria**: The entire system meets the functional requirements, with a successful end-to-end borrowing and return process without issues.
4. **Constraints and Assumptions**:
5. **Constraints**:
   1. Limited access to real-time data due to testing in a controlled environment.
   2. Testing restricted to specific user roles (librarian and member).
6. **Assumptions**:
   1. Assumed stable network connectivity during performance testing.
   2. User and book databases contain sufficient mock data for all testing stages*.*

## 2. Test Strategy

### 2.1 Testing Types

**Unit Testing**

* **Objective**: Validate the correctness of individual components, such as methods for book borrowing, return calculations, and membership validations.
* **Technique**: Automated unit testing using assertions to validate input-output correctness, error handling, and boundary conditions for each function.
* **Completion Criteria**: All unit tests pass without errors, covering at least 90% of the core functions, with special focus on key functionalities like borrowing limits and overdue penalties.

**Integration Testing**

* **Objective**: Ensure that modules (e.g., user management, borrowing, and notification systems) interact correctly and maintain data integrity across the workflow.
* **Technique**: Test interactions between modules by simulating actions (e.g., borrowing a book, which triggers notifications and updates to the borrowing history).
* **Completion Criteria**: All integrations between components should work smoothly without data loss or inconsistencies. All integrated workflows complete without errors.

**System Testing**

* **Objective**: Verify that the complete system meets both functional and non-functional requirements by simulating real user scenarios.
* **Technique**: Execute end-to-end test scenarios based on typical user interactions, such as borrowing and returning books, handling overdue fees, and upgrading memberships.
* **Completion Criteria**: All user stories and workflows perform correctly, with each function behaving as expected. Performance, security, and usability requirements are also met.

**Performance Testing**

* **Objective**: Ensure the system performs efficiently under normal and peak load conditions, with acceptable response times.
* **Technique**: Conduct load tests with simulated users to gauge response times, and stress tests to identify performance limits under high transaction volumes.
* **Completion Criteria**: The system should meet or exceed defined performance benchmarks (e.g., <2 seconds response time for all main actions under normal load, and maintaining service quality under peak loads).

**User Acceptance Testing (UAT)**

* **Objective**: Verify that the system meets end-user needs and behaves as expected in real-life scenarios.
* **Technique**: Conduct testing sessions with representative end-users (e.g., librarians and library members) performing typical tasks such as borrowing, returning, and managing accounts.
* **Completion Criteria**: All user interactions yield expected outcomes with satisfactory usability. Users confirm that the system meets requirements and is ready for deployment.

**Regression Testing**

* **Objective**: Ensure that recent changes, fixes, or updates have not introduced new issues.
* **Technique**: Automated regression tests run on critical functionalities after updates, with focus on areas impacted by recent code changes.
* **Completion Criteria**: All critical paths and affected features pass regression tests without new errors.

### 2.2 Test Levels

**Unit Testing Level**

1. **Description**: Executed to validate the smallest testable parts of the application, typically at the function or method level.
2. **Test Types**: Unit Testing

**Integration Testing Level**

1. **Description**: Verifies that combined modules or services work together as expected.
2. **Test Types**: Integration Testing

**System Testing Level**

1. **Description**: Ensures that the entire system, with all integrated components, meets the defined requirements.
2. **Test Types**: System Testing, Performance Testing

**Acceptance Testing Level**

1. **Description**: Conducted to confirm that the system meets user requirements and is ready for production.
2. **Test Types**: User Acceptance Testing, Regression Testing (as needed)

## 4. Test Reports

| **No.** | **Deliverable Item** | **Description** |
| --- | --- | --- |
| 1 | Unit Test | [SE1738NJ\_G5\_Unit\_Test\_Report](https://docs.google.com/spreadsheets/d/175fNLT_XWjis6T5fnChQFOn74lJ8pd-gPLEYIc2z8q8/edit?usp=sharing) |
| 2 | Integration Test | [SE1738NJ\_G5\_Integration\_Test\_Report](https://docs.google.com/spreadsheets/d/1QHqDod2BfuPMi3BAD1H4qiBt96Z-hJz7xo23Z36rS30/edit?usp=sharing) |
| 3 | System Test | [SE1738NJ\_G5\_System\_Test\_Report](https://docs.google.com/spreadsheets/d/1ncKdAHIY49kHWI0huXQ3b0v3iOkw38SmjtvqgX6Whn0/edit?usp=sharing) |