BOOKBYTES

Your Trusted Book Recommendation System

SECJ2154 –OBJECT ORIENTED PROGRAMMING

Book Recommendation System

Presented by: Group 9

Supervise by : Madam Lizawati

Group Member

Puan Kang Wei

A23CS5043

Mohamad Rezqi

B23CS0006

Abdul Mateen

B23CS0002

Faiq Mustaqim

B23CS3001

Overview

- Introduction
- Objectives
- User Journey
- UML Diagram
- Flowcharts
- OOP Concepts

Introduction

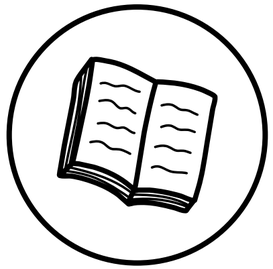
General concepts

- The BOOKBYTES Book Recommendation System is a platform designed to provide personalized book recommendations based on user preference such as genre, author, and title.

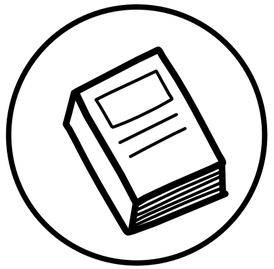
Target User

- Undergraduate students (UG), Postgraduate students (PG), and other users with limited access.

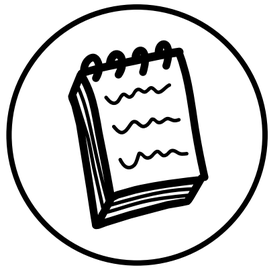
Project Objectives



To provide personalized book recommendations to users based on their preferences.



To enhance the user experience



increasing user engagement and satisfaction.

User Journey

Scenario

A user is blur with the book resources. He recently heard about an online book recommendation system and decided to give it a try.

UG/PG Student

Upon completing his/her profile, the system presents with a list of initial book recommendations based on her preferences

For recommendation system, the user finds a steady stream of books that match her tastes and expands her literary horizons. The system helps her stay engaged with her love of reading and provides a platform to share her thoughts with like-minded readers.

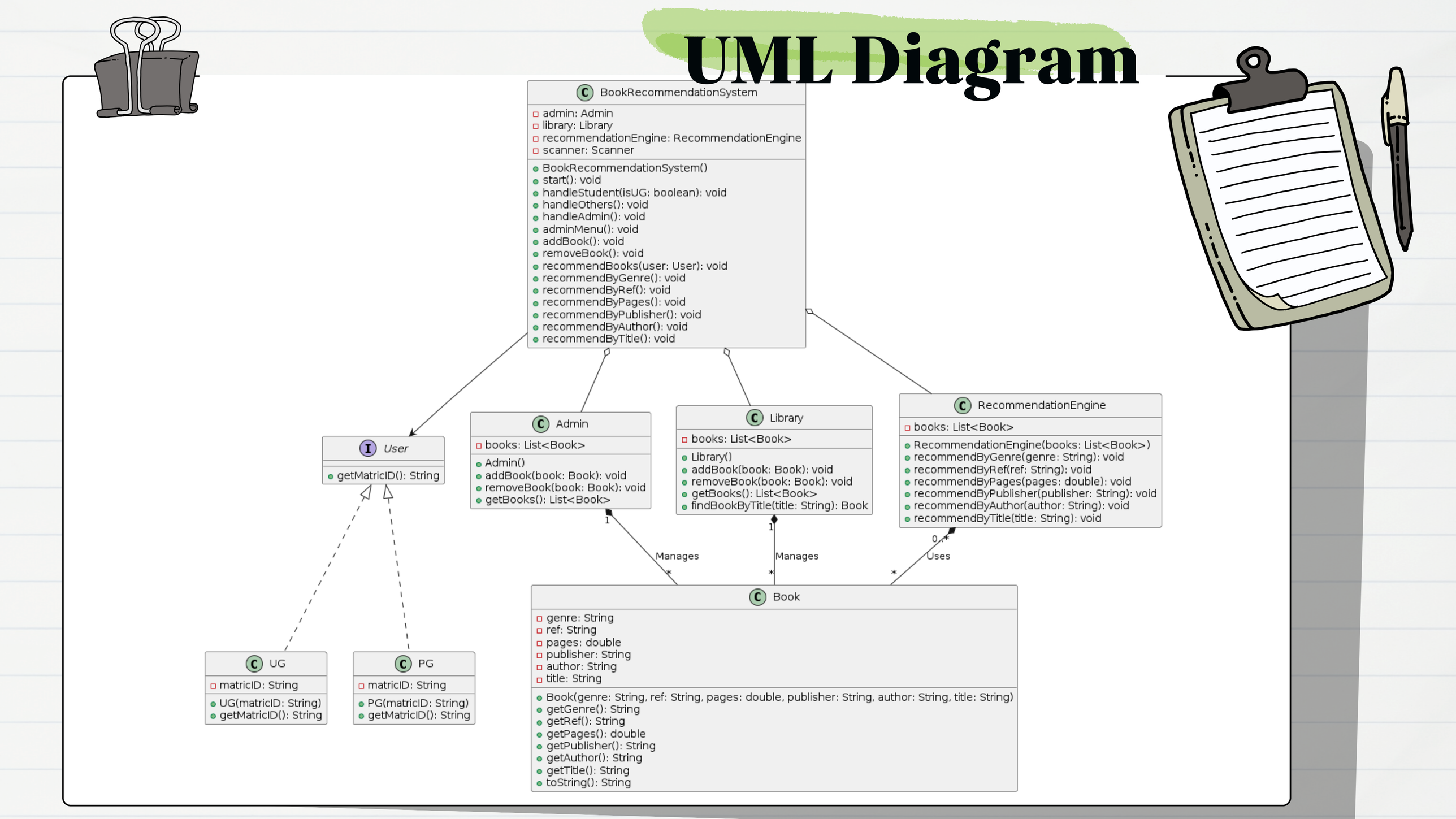
Admin

The system initialize the admin details. after in the admin dashboard, he checks for new book releases and uploads metadata and descriptions

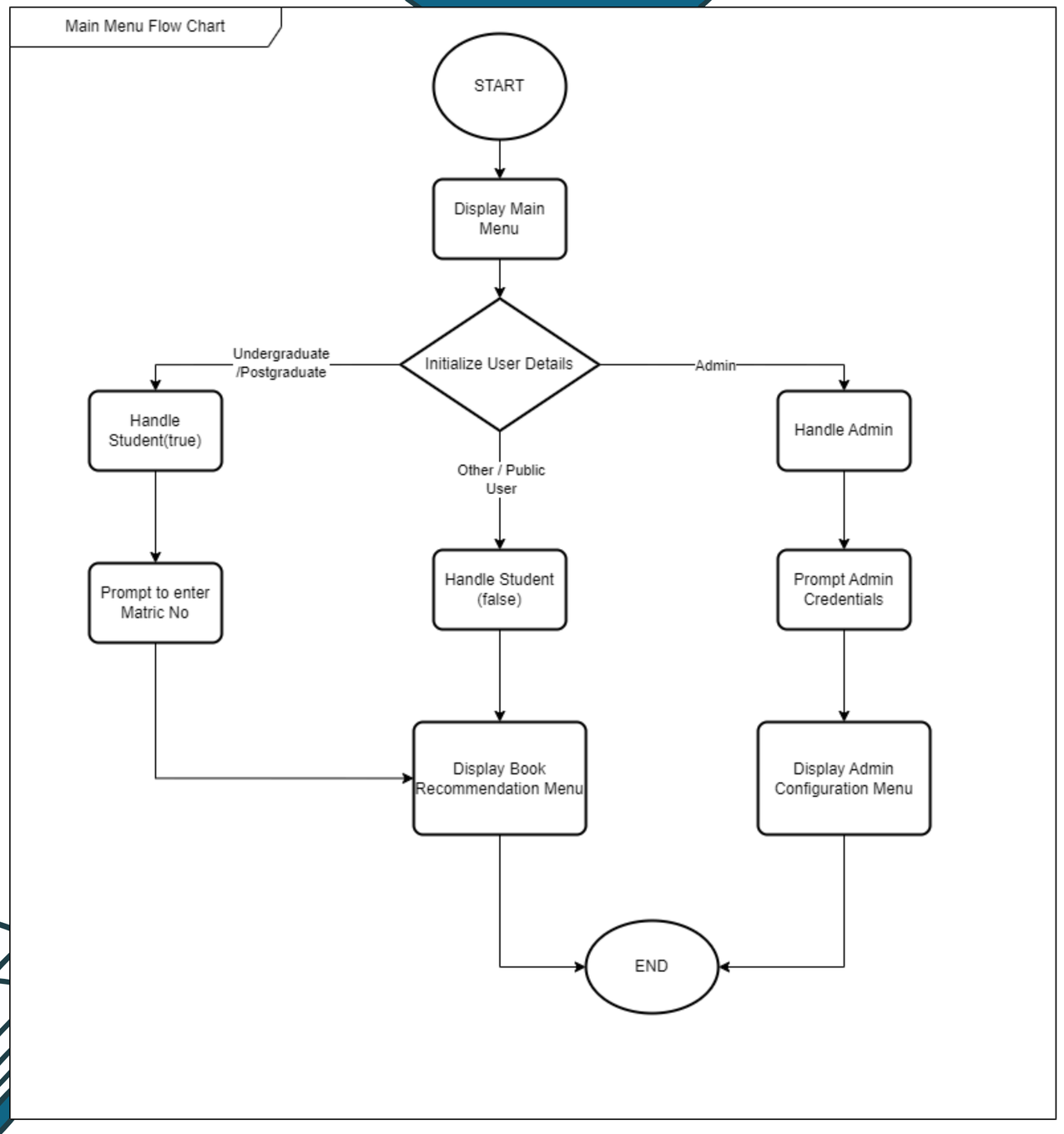
Adding Books

Removing Books

UML Diagram


BookRecommendationSystem
admin: Admin
library: Library
recommendationEngine: RecommendationEngine
scanner: Scanner
BookRecommendationSystem()
start(): void
handleStudent(isUG: boolean): void
handleOthers(): void
handleAdmin(): void
adminMenu(): void
addBook(): void
removeBook(): void
recommendBooks(user: User): void
recommendByGenre(): void
recommendByRef(): void
recommendByPages(): void
recommendByPublisher(): void
recommendByAuthor(): void
recommendByTitle(): void
User
getMatricID(): String
Admin
books: List<Book>
Admin()
addBook(book: Book): void
removeBook(book: Book): void
getBooks(): List<Book>
Library
books: List<Book>
Library()
addBook(book: Book): void
removeBook(book: Book): void
getBooks(): List<Book>
findBookByTitle(title: String): Book
RecommendationEngine
books: List<Book>
RecommendationEngine(books: List<Book>)
recommendByGenre(genre: String): void
recommendByRef(ref: String): void
recommendByPages(pages: double): void
recommendByPublisher(publisher: String): void
recommendByAuthor(author: String): void
recommendByTitle(title: String): void
1
Manages
*
1
Manages
*
0..*
Uses
*
UG
matricID: String
UG(matricID: String)
getMatricID(): String
PG
matricID: String
PG(matricID: String)
getMatricID(): String
Book
genre: String
ref: String
pages: double
publisher: String
author: String
title: String
Book(genre: String, ref: String, pages: double, publisher: String, author: String, title: String)
getGenre(): String
getRef(): String
getPages(): double
getPublisher(): String
getAuthor(): String
getTitle(): String
toString(): String

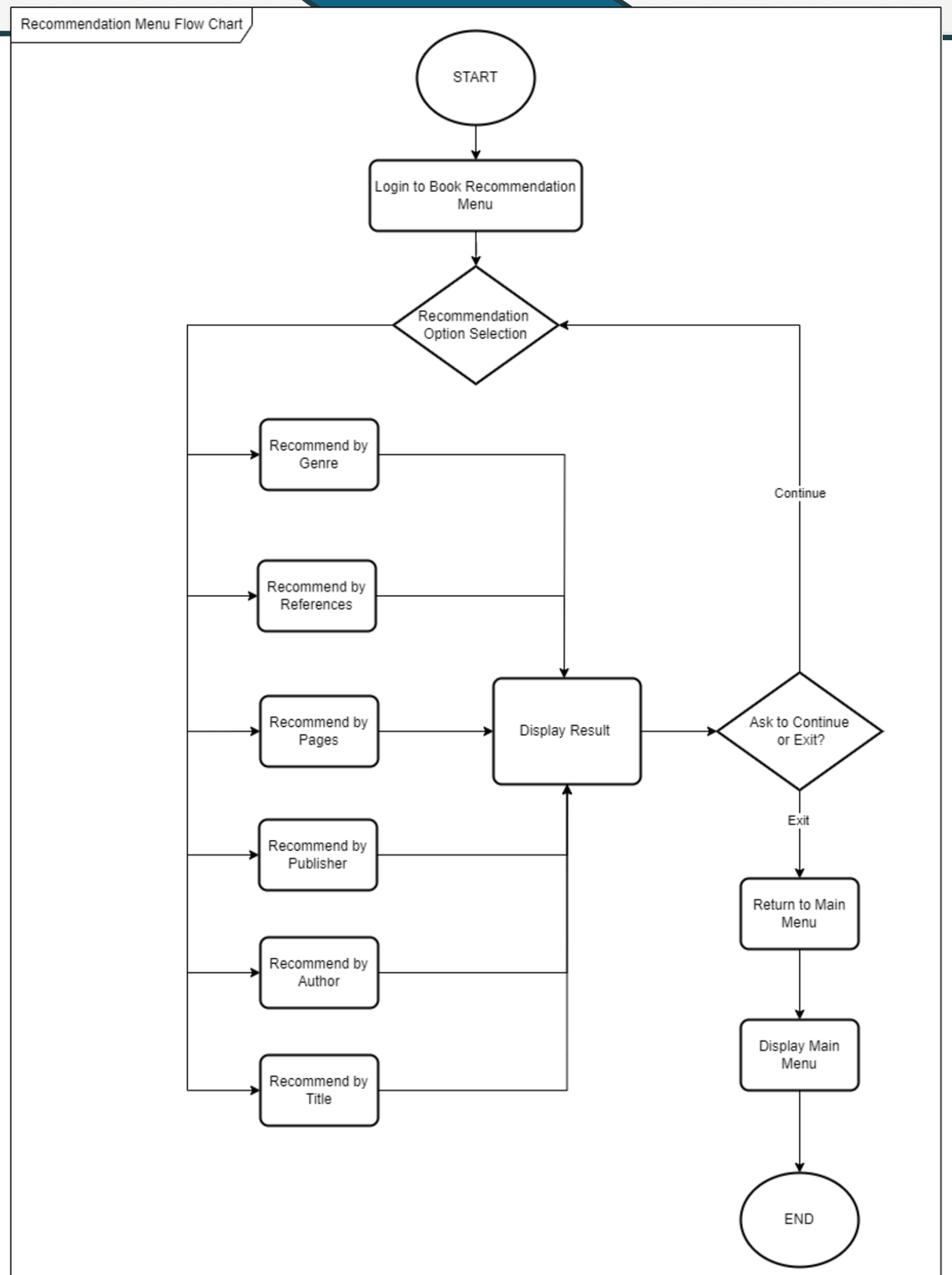
Flow Chart



Main Menu

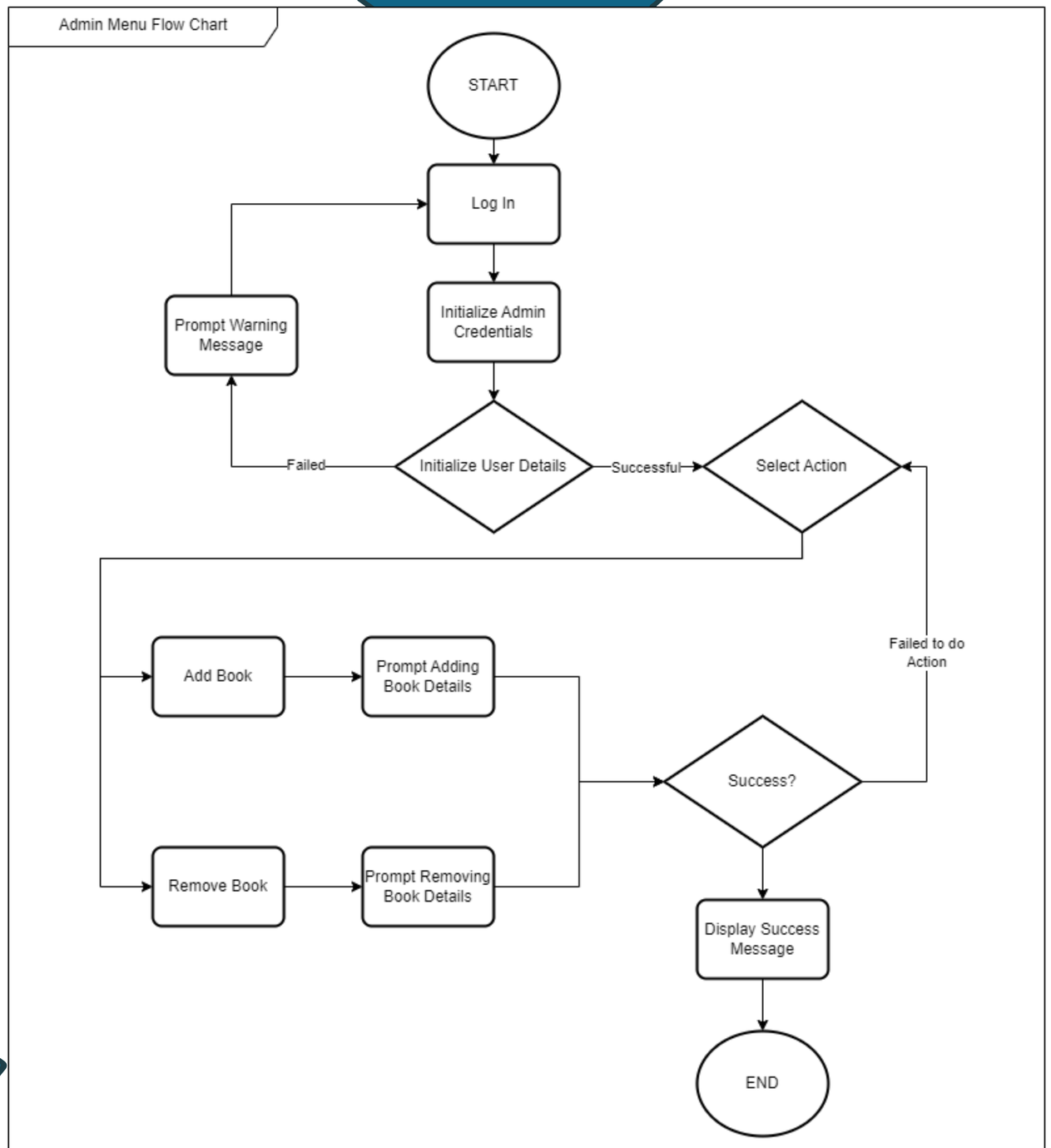
Flow Chart

Recommendation Menu



Flow Chart

Admin Menu



OOP Concepts

Encapsulation and Data Hiding

Wrapping data and its methods into a single object. This helps by keeping the data safe from unintended access and modification.

Association

Association explains how different classes or objects are related to one another and in the way they interact.

Inheritance

Inheritance creates a hierarchy of classes that share common attributes and behaviours

Exception Handling

Exception handling within the system is crucial for handling unexpected errors or inputs by the user when interacting with the systems

Encapsulation

```
class Book {
    // Encapsulation: Private fields to hide data from outside access
    private String genre;
    private String ref;
    private double pages;
```

```
    // Getter methods to provide controlled access to private fields
    public String getGenre() {
        return genre;
    }

    public String getRef() {
        return ref;
    }

    public double getPages() {
        return pages;
    }
```

Association

```java
class Admin {
    // Composition: Admin 'has a' list of books. Admin's lifecycle
does not depend on books
    private List<Book> books;

    public Admin() {
        this.books = new ArrayList<>();
    }
```

```java
class Library {
    // Composition: Library 'has a' list of books. Library's
lifecycle does not depend on books
    private List<Book> books;

    public Library() {
        this.books = new ArrayList<>();
    }
```

Inheritance

```java
interface User {
    public String getMatricID();
}

// UG class uses inheritance to implement User interface
class UG implements User {
    private String matricID;

    public UG(String matricID) {
        this.matricID = matricID;
    }

    public String getMatricID() {
        return matricID;
    }
}
```

Exception Handling

```
    public void start() {
        while (true) {
            try {
                System.out.println("Welcome to the Main Menu
ystem\n");
```

```
            }
        } catch (Exception e) {
            // Exception handling: catching any unexpected exceptions to prevent crash
            System.out.println("An error occurred: " + e.getMessage());
            scanner.next(); // Clear invalid input
        }
    }
```

Thank's For Listening