DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CME 2210
Object Oriented Analysis and Design

Library Management System

by
2022510013 YASAMIN VALISHARIATPANAHI
2022510046 SARA MAHYANBAKHSHAYESH
2020510121 GÜLNAZ HİLMİOĞLU
2022510011 KASRA SOLTANI

CHAPTER ONE

INTRODUCTION

1.1 Overview

In today's digital age, libraries face the challenge of adapting to modern technologies while maintaining their core mission of providing access to knowledge and fostering a love for reading and learning. Manual systems for library management are often inefficient and prone to errors, hindering the ability of libraries to effectively serve their patrons. To address these challenges, the development of a Library Management System (LMS) presents a viable solution.

In this report, we introduce our library system management, which aims to streamline access to resources and enhance user experience within the library setting. Our system caters to administrators, staff, and members, providing a seamless platform for book management and user interactions.

1.2 Purpose of the LMS Project

By exploring the intricacies of this Library Management System, this report aims to elucidate its significance in modernizing library operations, enhancing user experience, and optimizing resource utilization. Through an in-depth examination of its design principles, features, and implementation, readers will gain valuable insights into the potential benefits of adopting such systems in libraries of varying scales.

1.3 Scope of the LMS Project

The scope of the Library Management System project encompasses the development of software tailored specifically for library management purposes. Key functionalities include:

- Book Management: Adding, removing, updating, searching, and viewing book information, as well as organizing books by genre.

- Member Management: Adding, removing, updating, searching, and viewing member information, managing membership statuses, and tracking borrowing history.
- Staff Management: Adding, removing, updating, searching, and viewing staff information, roles, and permissions within the library, including salary calculation.
- Loan Tracking: Automating the process of loaning books to members, including due date management and fine calculation for late returns.
- Authentication Mechanisms: Implementing secure authentication for both library members and staff to access the system according to their roles and permissions.
- Data Persistence: Storing library data, including book information, member details, staff information, and transaction history, in a centralized database for easy access and management.
- Account Management: Allowing users to manage their account information, including passwords and contact details.

1.4 Definitions, Acronyms, and Abbreviations

- LMS: Library Management System, a software solution aimed at streamlining library operations.
- Admin: Administrator, possesses full control over all aspects of the system.
- Staff: Library personnel with restricted access, typically limited to viewing information rather than actively managing it.
- Members: Registered users of the system, able to borrow books and access various features.

1.5 Additional Tools

In addition to the Library Management System, other tools can enhance library operations. Lucidchart is a powerful application for designing Unified Modeling Language (UML) diagrams, aiding in the visualization of system architectures and workflows. For project management, ClickUp offers a comprehensive platform for organizing tasks,

tracking progress, and collaborating with team members, ensuring efficient project execution. Additionally, the use of TestNG facilitates efficient testing of software components, ensuring robustness and reliability.

1.6 References

- [1] Java Programming Language: Oracle Corporation. (n.d.). https://www.oracle.com/java/
- [2] Eclipse IDE: The Eclipse Foundation. (n.d.). https://www.eclipse.org/
- [3] Lucidchart: Lucid Software Inc. (n.d.). https://www.lucidchart.com/
- [4] ClickUp: ClickUp Inc. (n.d.). https://clickup.com/
- [5] Books Dataset CSV File Of Total 271380 Books: Bagchi, Saurabh. (n.d.). https://www.kaggle.com/datasets/saurabhbagchi/books-dataset?resource=download

CHAPTER TWO

REQUIREMENTS

In this chapter, we will delve into the specific requirements of the library management system project. The scope and functionalities of the system, along with the user requirements and their respective solutions, will be outlined.

Specific Requirements

2.1 External Interfaces

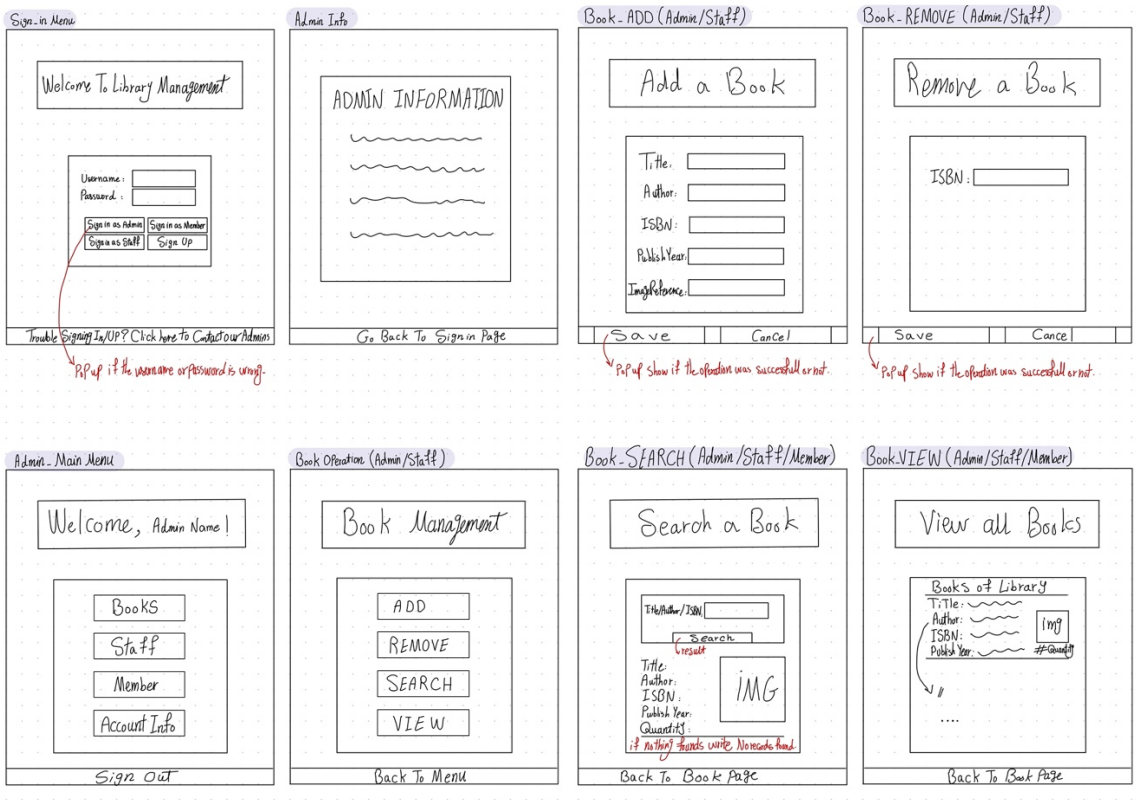
The external interfaces of the Library Management System (LMS) project are essential components that define how users interact with the system. By supporting both console-based interaction and a graphical user interface (GUI) implemented using Swing and AWT libraries, the LMS ensures versatility and accessibility for users with varying preferences and technical expertise.

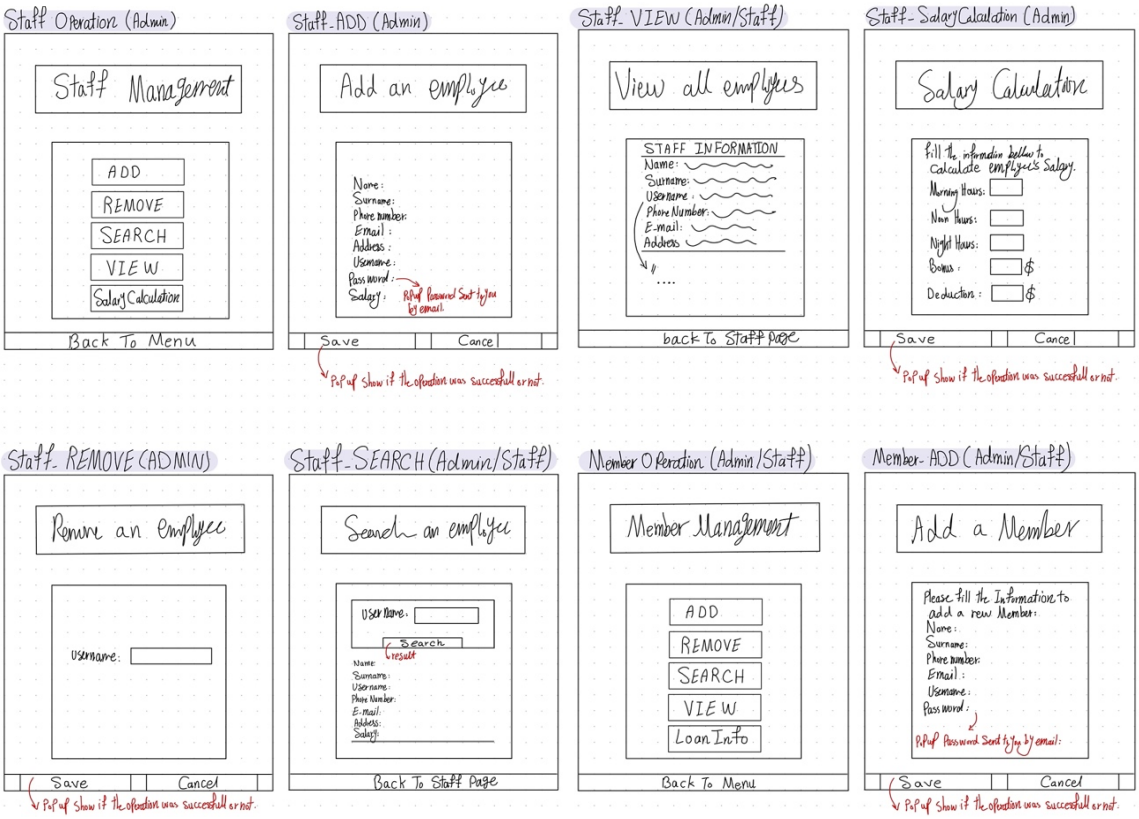
- Console Interface
  - Users interact with the system through a command-line interface, entering commands and data via the console.
  - The console interface provides a straightforward method for users to input commands and receive system responses.
  - Commands are entered in text format, with the system parsing and interpreting the input to perform the appropriate actions.

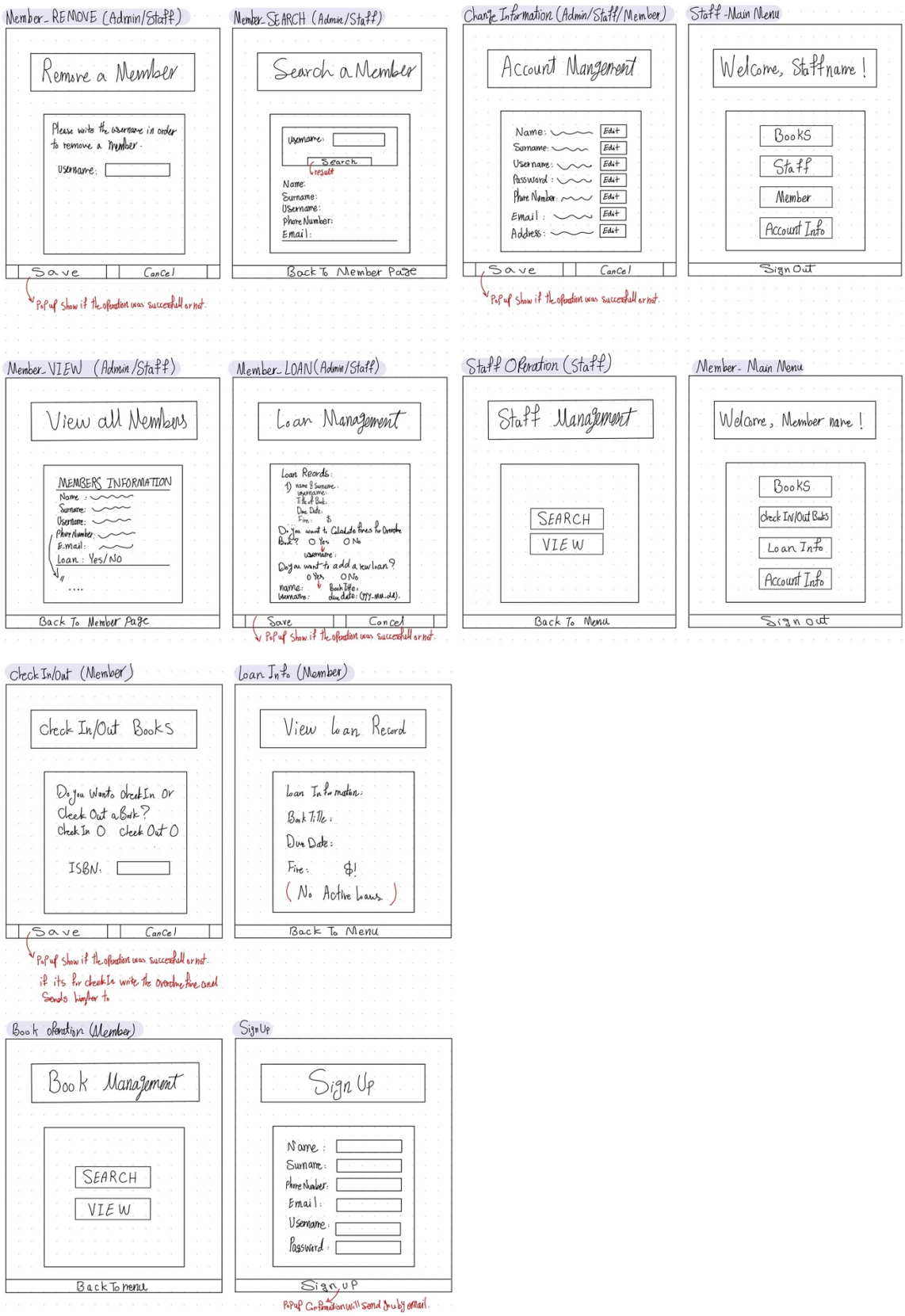
- Graphical User Interface (GUI)

- The GUI enhances user experience by providing a visually appealing and intuitive interface for interacting with the system.
- Implemented using Swing and AWT libraries, the GUI offers features such as buttons, text fields, and menus for user interaction.
- Users can perform tasks such as adding, removing, and updating book and member information through the GUI.
- Visual elements such as images and icons enhance the aesthetics and usability of the interface.

- Mock-up Prototype
  - A mock-up prototype of the GUI has been developed to illustrate the appearance and layout of the interface.
  - The mock-up serves as a visual representation of the final product, showcasing the design and functionality of the system.
  - While the mock-up provides an initial glimpse of the GUI, the actual implementation may vary in appearance and features.




Staff Operation (Admin)
Staff Management
ADD
REMOVE
SEARCH
VIEW
Salary Calculation
Back To Menu
Staff_ADD (Admin)
Add an employee
Name :
Surname :
Phone number:
Email :
Address :
Username :
Password :
Salary :
Pop up Password Sent to you by email.
Save
Cancel
Pop up Show if the operation was succesfull or not.
Staff_VIEW (Admin/Staff)
View all employees
STAFF INFORMATION
Name:
Surname:
Username :
Phone Number:
E-mail:
Address
....
back To Staff Page
Staff_SalaryCalculation (Admin)
Salary Calculation
Fill the information bellow to Calculate employee's Salary.
Morning Hours:
Noon Hours:
Night Hours:
Bonus :
$
Deductions :
$
Save
Cancel
Pop up Show if the operation was succesfull or not.
Staff_REMOVE (ADMIN)
Remove an employee
Username:
Save
Cancel
Pop up Show if the operation was succesfull or not.
Staff_SEARCH (Admin/Staff)
Search an employee
Username:
Search
result
Name:
Surname:
Username:
Phone Number:
E-mail:
Address:
Salary:
Back To Staff Page
Member Operation (Admin/Staff)
Member Management
ADD
REMOVE
SEARCH
VIEW
Loan Info
Back To Menu
Member_ADD (Admin/Staff)
Add a Member
Please fill the Information to add a new Member:
Name :
Surname:
Phone number:
Email :
Username :
Password :
Pop up Password Sent to you by email:
Save
Cancel
Pop up Show if the operation was succesfull or not.


Member_REMOVE (Admin/Staff)
Remove a Member
Please write the username in order to remove a Member.
Username:
Save
Cancel
Pop up show if the operation was succesfull or not.
Member_SEARCH (Admin/Staff)
Search a Member
Username:
Search
result
Name:
Surname:
Username:
Phone Number:
Email:
Back To Member Page
Change Information (Admin/Staff/Member)
Account Mangement
Name: Edit
Surname: Edit
Username: Edit
Password: Edit
Phone Number: Edit
Email: Edit
Address: Edit
Save
Cancel
Pop up show if the operation was succesfull or not.
Staff -Main Menu
Welcome, Staffname!
Books
Staff
Member
Account Info
Sign Out
Member_VIEW (Admin/Staff)
View all Members
MEMBERS INFORMATION
Name:
Surname:
Username:
Phone Number:
E-mail:
Loan: Yes/NO
Back To Member Page
Member_LOAN(Admin/Staff)
Loan Management
Loan Records:
Do you want to add a new loan?
Yes
No
Save
Cancel
Pop up show if the operation was succesfull or not.
Staff Operation (Staff)
Staff Management
SEARCH
VIEW
Back To Menu
Member- Main Menu
Welcome, Member name!
Books
check In/Out Books
Loan Info
Account Info
Sign out
check In/Out (Member)
check In/Out Books
Do you Want to check In or Check Out a Book?
Check In Check Out
ISBN:
Save
Cancel
Pop up show if the operation was succesfull or not.
if its for check In write the overdue fine cost
Loan Info (Member)
View Loan Record
Loan Information:
Book Title:
Due Date:
Fine: $!
(No Active Loans)
Back To Menu
Book operation (Member)
Book Management
SEARCH
VIEW
Back To menu
Sign Up
Sign Up
Name:
Surname:
Phone Number:
Email:
Username:
Password:
Sign up
Pop up Confirmation will send you by email.

2.2 Functions

❖ Library Management System

- Main
  - Description: Main method for the Library Management System.
  - Method: main(String[] args)
  - Parameters: args - An array of strings containing command-line arguments.
  - Return Type: void
  - Throws: IOException

- Login
  - Description: Method for handling the login process in the console.
  - Method: login()
  - Parameters: None
  - Return Type: int

- SignIn SignUp Menu
  - Description: Method for displaying the sign-in/sign-up menu in the GUI.
  - Method: signIn_UpMenu()
  - Parameters: None
  - Return Type: void

- Show Admin Information
  - Description: Method for displaying the admin information page in the GUI.
  - Method: showAdminInformation()
  - Parameters: None
  - Return Type: void

- Sign Up
  - Description: Method for handling the sign-up process in the GUI.
  - Method: signUp ()
  - Parameters: None
  - Return Type: void

- Account Management
  - Description: Method for managing the account information for admin or staff in the GUI.
  - Method: accountManagement ()
  - Parameters: None
  - Return Type: void

- Account Management Member
  - Description: Method for managing the account information for Member in the GUI
  - Method: accountManagement _Member()
  - Parameters: None
  - Return Type: void

- Admin Staff Menu
  - Description: Method for showing the Admin or Staff menu in the GUI
  - Method: Admin_StaffMenu()
  - Parameters: None
  - Return Type: void

- Member Menu
  - Description: Method for showing the Member menu in the GUI
  - Method: MemberMenu()
  - Parameters: None

- Return Type: void

- Book Management
  - Description: Method for book operations in the GUI
  - Method: bookManagement()
  - Parameters: None
  - Return Type: void
  -
- Staff Management
  - Description: Method for staff operations in the GUI
  - Method: staffManagement()
  - Parameters: None
  - Return Type: void

- Member Management
  - Description: Method for member operations in the GUI
  - Method: memberManagement()
  - Parameters: None
  - Return Type: void

- View Books
  - Description: Method for showing all the books in the GUI
  - Method: viewBooks()
  - Parameters: None
  - Return Type: void

- View Staff
  - Description: Method for showing all the staffs in the GUI
  - Method: viewStaff()
  - Parameters: None
  - Return Type: void

- View Members
  - Description: Method for showing all the members in the GUI
  - Method: viewMembers()
  - Parameters: None
  - Return Type: void

- View Member Loans
  - Description: Method for showing a member's loan in the GUI
  - Method: viewMemberLoan()
  - Parameters: None
  - Return Type: void

- Add Book
  - Description: Method for adding a book in the GUI
  - Method: addBook()
  - Parameters: None
  - Return Type: void

- Add Staff
  - Description: Method for adding a staff in the GUI
  - Method: addStaff()
  - Parameters: None
  - Return Type: void

- Add Member
  - Description: Method for adding a member in the GUI
  - Method: addMember()
  - Parameters: None
  - Return Type: void

- Salary Calculation Staff
  - Description: Method for calculating staff salary in the GUI
  - Method: salaryCalculationstaff ()
  - Parameters: None
  - Return Type: void

- Remove Book
  - Description: Method for removing a book in the GUI
  - Method: removeBook()
  - Parameters: None
  - Return Type: void

- Remove Staff
  - Description: Method for removing a staff in the GUI
  - Method: removeStaff()
  - Parameters: None
  - Return Type: void

- Remove Member
  - Description: Method for removing a member in the GUI
  - Method: removeMember()
  - Parameters: None
  - Return Type: void

- Search Book
  - Description: Method for search a book in the GUI
  - Method: searchBook()
  - Parameters: None
  - Return Type: void

- Search Staff

- Description: Method for search a staff in the GUI
- Method: searchStaff()
- Parameters: None
- Return Type: void

- Search Member
  - Description: Method for search a member in the GUI
  - Method: searchMember()
  - Parameters: None
  - Return Type: void

- Check In/Out Books
  - Description: Method for checking in or checking out a book in the GUI
  - Method: checkInOutNooks()
  - Parameters: None
  - Return Type: void

- Loan Management
  - Description: Method for loan Operations in the GUI
  - Method: loanManagement ()
  - Parameters: None
  - Return Type: void

❖ Person

- Get ID
  - Description: Users can retrieve the ID of a person.
  - Method: getID()
  - Parameters: None
  - Return Type: int

- Set ID
  - Description: Users can set the ID of a person.
  - Method: setID(int ID)
  - Parameters: ID - The ID to set for the person
  - Return Type: Void

- Get Name
  - Description: Users can retrieve the name of a person.
  - Method: getName()
  - Parameters: None
  - Return Type: String

- Set Name
  - Description: Users can set the name of a person.
  - Method: setName(String name)
  - Parameters: name - The name to set for the person
  - Return Type: Void

- Get Surname
  - Description: Users can retrieve the surname of a person.
  - Method: getSurname()
  - Parameters: None
  - Return Type: String

- Set Surname
  - Description: Users can set the surname of a person.
  - Method: setSurname(String surname)
  - Parameters: surname - The surname to set for the person
  - Return Type: Void

- Get Phone Number

- Description: Users can retrieve the phone number of a person.
- Method: getPhoneNumber()
- Parameters: None
- Return Type: String

- Set Phone Number
  - Description: Users can set the phone number of a person.
  - Method: setPhoneNumber(String phoneNumber)
  - Parameters: phoneNumber - The phone number to set for the person
  - Return Type: Void

- Get Address
  - Description: Users can retrieve the address of a person.
  - Method: getAddress()
  - Parameters: None
  - Return Type: String

- Set Address
  - Description: Users can set the address of a person.
  - Method: setAddress(String address)
  - Parameters: address - The address to set for the person
  - Return Type: Void

- Get Email
  - Description: Users can retrieve the email of a person.
  - Method: getEmail()
  - Parameters: None
  - Return Type: String

- Set Email
  - Description: Users can set the email of a person.

- Method: setEmail(String email)
- Parameters: email - The email to set for the person
- Return Type: Void

- Get Username
  - Description: Users can retrieve the username of a person.
  - Method: getUsername()
  - Parameters: None
  - Return Type: String

- Set Username
  - Description: Users can set the username of a person.
  - Method: setUsername(String username)
  - Parameters: username - The username to set for the person
  - Return Type: Void

- Get Password
  - Description: Users can retrieve the password of a person.
  - Method: getPassword()
  - Parameters: None
  - Return Type: String

- Set Password
  - Description: Users can set the password of a person.
  - Method: setPassword(String password)
  - Parameters: password - The password to set for the person
  - Return Type: Void

- Set Password
  - Description: Users can display information about a person.
  - Method: setPassword(String password)

- Parameters: None
- Return Type: Void

❖ Admin

- Get Salary
  - Description: Users can retrieve the salary of the admin.
  - Method: getSalary()
  - Parameters: None
  - Return Type: double

- Set Salary
  - Description: Users can set the salary of the admin.
  - Method: setSalary(double salary)
  - Parameters: salary The new salary of the admin
  - Return Type: Void

- Get Admins
  - Description: Users can retrieve the list of admins.
  - Method: getAdmins()
  - Parameters: None
  - Return Type: ArrayList<Admin>

- Display Info
  - Description: Users can display information about the admin.
  - Method: displayInfo()
  - Parameters: None
  - Return Type: Void

- Get All Admin Info
  - Description: Users can get information about all admins.
  - Method: getAllAdminsInfo(ArrayList<Admin> admins)

- Parameters: admins - The list of admins
- Return Type: String

- Get All Admins Info GUI
  - Description: Users can get GUI-friendly information about all admins.
  - Method: getAllAdminsInfo_GUI(ArrayList<Admin> admins)
  - Parameters: admins - The list of admins
  - Return Type: String

- Get All Admins Information
  - Description: Users can get information about all admins.
  - Method: getAllAdminsInformation()
  - Parameters: None
  - Return Type: String

- Search Admin By Username
  - Description: Users can search for an admin by username.
  - Method: searchAdminByUsername(String username)
  - Parameters: The username of the admin to search for
  - Return Type: Admin

- Update Admin Information
  - Description: Users can update information of an admin.
  - Method: updateAdminInformation(String username, String newName, String newSurname, String newPhoneNumber, String newAddress, String newEmail, String newPassword)
  - Parameters:
    - username - The username of the admin to update
    - newName - The new name of the admin
    - newSurname - The new surname of the admin
    - newPhoneNumber - The new phone number of the admin
    - newAddress - The new address of the admin

        - newEmail - The new email of the admin
        - newPassword - The new password of the admin
    - Return Type: boolean

- Update Admin Information CSV
    - Description: Users can update information of an admin from CSV.
    - Method: updateAdminInformation_CSV(String username, String newName, String newSurname, String newPhoneNumber, String newAddress, String newEmail, String newPassword)
    - Parameters:
        - username - The username of the admin to update
        - newName - The new name of the admin
        - newSurname - The new surname of the admin
        - newPhoneNumber - The new phone number of the admin
        - newAddress - The new address of the admin
        - newEmail - The new email of the admin
        - newPassword - The new password of the admin
    - Return Type: boolean

- Read From CSV
    - Description: Users can read admin information from CSV file.
    - Method: readfromcsv()
    - Parameters: None
    - Return Type: void
    - Throws: IOException

- Authenticate
    - Description: Users can authenticate an admin with given username and password.
    - Method: authenticate(String username, String password)
    - Parameters:
        - username - The username of the admin

        - password - The new password of the admin
    - Return Type: boolean

- Admin Menu
    - Description: Users can display the admin menu.
    - Method: adminMenu()
    - Parameters: None
    - Return Type: void
    - Throws: IOException

❖ Staff

- Get Salary
    - Description: Get the salary of the staff member.
    - Method: getSalary()
    - Parameters: None
    - Return Type: double

- Set Salary
    - Description: Set the salary of the staff member.
    - Method: setSalary(double salary)
    - Parameters: salary - The salary to set
    - Return Type: Void

- Calculate Salary
    - Description: Calculate the total salary of the staff member including bonuses and deductions
    - Method: calculateSalary(int morningHours, int noonHours, int nightHours, double bonusAmount, double deductionAmount)
    - Parameters:
        - morningHours - The number of hours worked in the morning shift
        - noonHours - The number of hours worked in the noon shift

        - nightHours - The number of hours worked in the night shift
        - bonusAmount - The amount of bonus to be added to the salary
        - deductionAmount - The amount of deduction to be subtracted from the salary
    - Return Type: double
- Authenticate
    - Description: Authenticate the user with the provided username and password.
    - Method: authenticate(String username, String password)
    - Parameters:
        - username - The username of the staff member trying to authenticate
        - password - The password of the staff member trying to authenticate
    - Return Type: boolean

- Staff Menu
    - Description: Display the menu options available to the staff member.
    - Method: staffMenu()
    - Parameters: None
    - Return Type: Void
    - Throws: IOException

❖ Member

- Get Loans
    - Description: Get the list of loans associated with the member.
    - Method: getLoans()
    - Parameters: None
    - Return Type: ArrayList<Loan>

- Authenticate

- Description: Authenticate the user with the provided username and password.
- Method: authenticate(String username, String password)
- Parameters:
    - username - The username of the user trying to authenticate
    - password - The password of the user trying to authenticate
- Return Type: boolean

- Member Menu
    - Description: Display the menu options available to the member.
    -
    - Method: memberMenu()
    - Parameters: None
    - Return Type: Void

❖ Book

- Get Title
    - Description: Get the title of the book.
    - Method: getTitle()
    - Parameters: None
    - Return Type: String

- Set Title
    - Description: Set the title of the book.
    - Method: setTitle(String title)
    - Parameters: title - The title to set
    - Return Type: Void

- Get Author
    - Description: Get the author of the book.
    - Method: getAuthor()

- Parameters: None
- Return Type: String

- Set Author
  - Description: Set the author of the book.
  - Method: setAuthor(String author)
  - Parameters: author - The author to set
  - Return Type: Void

- Get ISBN
  - Description: Get the ISBN of the book.
  - Method: getISBN()
  - Parameters: None
  - Return Type: String

- Set ISBN
  - Description: Set the ISBN of the book.
  - Method: setISBN(String ISBN)
  - Parameters: ISBN - The ISBN to set
  - Return Type: Void

- Get Publish Year
  - Description: Get the publish year of the book.
  - Method: getPublishYear()
  - Parameters: None
  - Return Type: String

- Set Publish Year
  - Description: Set the publish year of the book.
  - Method: setPublishYear(String publishYear)
  - Parameters: publishYear - The publish year to set

- Return Type: Void

- Get Total Quantity
  - Description: Get the total quantity of the book.
  - Method: getTotalQuantity()
  - Parameters: None
  - Return Type: int

- Set Total Quantity
  - Description: Set the total quantity of the book.
  - Method: setTotalQuantity(int totalQuantity)
  - Parameters: totalQuantity - The total quantity to set
  - Return Type: Void

- Get IMG
  - Description: Get the image URL of the book.
  - Method: getIMG()
  - Parameters: None
  - Return Type: String

- Set IMG
  - Description: Set the image URL of the book.
  - Method: setIMG(String img)
  - Parameters: img - The image URL to set
  - Return Type: Void

- Decrease Total Quantity
  - Description: Decrease the total quantity of the book by the specified amount.
  - Method: decreaseTotalQuantity(int amount)

- Parameters: amount - The amount by which to decrease the total quantity
- Return Type: Void
- Throws: IllegalArgumentException

- Increase Total Quantity
  - Description: Increase the total quantity of the book by the specified amount.
  - Method: increaseTotalQuantity(int amount)
  - Parameters: amount - The amount by which to increase the total quantity
  - Return Type: Void
  - Throws: IllegalArgumentException

❖ Loan

- Get Book
  - Description: Gets the book associated with the loan.
  - Method: getBook()
  - Parameters: None
  - Return Type: Book

- Set Book
  - Description: Sets the book associated with the loan.
  - Method: setBook(Book book)
  - Parameters: book - The book to set.
  - Return Type: Void

- Get Member
  - Description: Gets the member associated with the loan.
  - Method: getMember()
  - Parameters: None

- Return Type: Member

- Set Member
  - Description: Sets the member associated with the loan.
  - Method: setMember(Member member)
  - Parameters: member - The member to set.
  - Return Type: Void
- Get Due Date
  - Description: Gets the due date of the loan.
  - Method: getDueDate()
  - Parameters: None
  - Return Type: Date

- Set Due Date
  - Description: Sets the due date of the loan.
  - Method: setDueDate(Date dueDate)
  - Parameters: dueDate - The due date to set.
  - Return Type: Void

- Get Fine
  - Description: Gets the fine associated with the loan.
  - Method: getFine()
  - Parameters: None
  - Return Type: double

- Set Fine
  - Description: Sets the fine associated with the loan.
  - Method: setFine(double fine)
  - Parameters: fine - The fine to set.
  - Return Type: Void

- Calculate Fine
  - Description: Calculates the fine for the loan based on the due date and the current date.
  - Method: calculateFine()
  - Parameters: None
  - Return Type: double

- Get Date
  - Description: Gets the formatted date of the loan for GUI printing.
  - Method: getDate()
  - Parameters: None
  - Return Type: String

❖ Library

- Read Operations
  - Description: Reads data from CSV files and performs necessary operations.
  - Method: readoperations()
  - Parameters: None
  - Return Type: void
  - Throws: IOException

- Get Current ID Update book
  - Description: Retrieves the current IDs and updates book information if necessary.
  - Method: getCurrentID_Updatebook()
  - Parameters: None
  - Return Type: void
  - Throws: IOException

- Add Book
  - Description: Adds a new book to the library system.
  - Method: addBook()
  - Parameters: None
  - Return Type: void

- Add Book GUI
  - Description: Adds a new book to the library system with provided information.
  - Method: addBook_GUI(String title, String author, String ISBN, String publishYear, String img)
  - Parameters:
    - title - The title of the book.
    - author - The author of the book.
    - ISBN - The ISBN of the book.
    - publishYear - The publishing year of the book.
    - img - The image reference of the book.
  - Return Type: boolean

- Remove Book
  - Description: Removes a book from the library system.
  - Method: removeBook()
  - Parameters: None
  - Return Type: void
  - Throws: IOException

- Remove Book By ISBN GUI
  - Description: Removes a book from the library system by ISBN.
  - Method: removeBookByISBN_GUI(String ISBN)
  - Parameters: ISBN - The ISBN of the book to be removed.
  - Return Type: boolean

- Search Book
  - Description: Searches for a book in the library system.
  - Method: searchBook()
  - Parameters: None
  - Return Type: void
  - Throws: IOException

- Search Book GUI
  - Description: Searches for a book in the library system by the given search key for GUI purposes.
  - Method: searchBook_GUI(String searchKey)
  - Parameters: searchKey - The title, author, or ISBN of the book to search for.
  - Return Type: boolean
  - Throws: IOException

- View Found Book
  - Description: Searches for a book in the library system by the given search key.
  - Method: viewFoundBook(String searchKey)
  - Parameters: searchKey - The title, author, or ISBN of the book to search for.
  - Return Type: String

- Print Books
  - Description: Prints details of all books in the library system.
  - Method: printBooks()
  - Parameters: None
  - Return Type: void

- Print Books GUI

- Description: Prints details of all books in the library system for GUI display.
- Method: printBooks_GUI()
- Parameters: None
- Return Type: String

o Add Staff
- Description: Adds a new staff member to the library system.
- Method: addStaff()
- Parameters: None
- Return Type: void

o Add Staff with Details
- Description: Adds a new staff member to the library system with provided information.
- Method: addStaff(String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary)
- Parameters: Parameters as described in the method signature.
- Return Type: void

o Add Staff GUI
- Description: Adds a new staff member to the library system with provided information.
- Method: addStaff_GUI(String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary)
- Parameters: Parameters as described in the method signature.
- Return Type: boolean

o Remove Staff
- Description: Removes a staff member from the library system.

- Method: removeStaff()
- Parameters: None
- Return Type: void

- Remove Staff GUI
  - Description: Removes a staff member from the library system by username.
  - Method: removeStaffByUsername_GUI(String username)
  - Parameters: username - The username of the staff member to be removed.
  - Return Type: boolean

- Search Staff
  - Description: Searches for a staff member in the library system.
  - Method: searchStaff()
  - Parameters: None
  - Return Type: void
  - Throws: IOException

- Search Staff by Username
  - Description: Searches for a staff member in the library system by username.
  - Method: searchStaffByUsername(String searchKey)
  - Parameters: searchKey - The username of the staff member to search for.
  - Return Type: Staff
  - Throws: IOException

- Search Staff by Username GUI
  - Description: Searches for a staff member in the library system by username, name, or email for GUI purposes.
  - Method: searchStaffByUsername_GUI(String searchKey)

- Parameters: searchKey - The username, name, or email of the staff member to search for.
- Return Type: boolean

- View Found Staff
  - Description: Displays information of the found staff member.
  - Method: viewFoundStaff(String searchKey)
  - Parameters: searchKey - The username of the staff member to find.
  - Return Type: String

- Print Staff
  - Description: Prints details of all staff members in the library system.
  - Method: printStaff()
  - Parameters: None
  - Return Type: void

- Print Staff GUI
  - Description: Prints details of all staff members in the library system for GUI display.
  - Method: printStaff_GUI()
  - Parameters: None
  - Return Type: String

- Add to Array Staff
  - Description: Updates staff member information and writes changes to CSV.
  - Method: addToArrayStaff(String username, String newName, String newSurname, String newPhoneNumber, String newAddress, String newEmail, String newUsername, String newPassword)
  - Parameters: Parameters as described in the method signature.
  - Return Type: String
  - Throws: IOException

- Staff Calculate Salary
  - Description: Calculates and updates the salary of a staff member and writes changes to CSV.
  - Method: StaffCalculateSalary(String username, int morningHours, int noonHours, int nightHours, double bonusAmount, double deductionAmount)
  - Parameters: Parameters as described in the method signature.
  - Return Type: boolean

- Add Member
  - Description: Adds a new member with user input.
  - Method: addMember()
  - Parameters: None
  - Return Type: void

- Add Member with Details
  - Description: Adds a new member with specified details.
  - Method: addMember(String name, String surname, String phoneNumber, String address, String email, String username, String password)
  - Parameters: Parameters as described in the method signature.
  - Return Type: void

- Add Member GUI
  - Description: Adds a new member with specified details and returns true if added successfully.
  - Method: addMember_GUI(String name, String surname, String phoneNumber, String email, String username, String password)
  - Parameters: Parameters as described in the method signature.
  - Return Type: boolean

- Remove Member
  - Description: Removes a member by username.
  - Method: removeMember()
  - Parameters: None
  - Return Type: boolean

- Remove Member by Username GUI
  - Description: Removes a member by username and returns true if removed successfully.
  - Method: removeMemberByUsername_GUI(String username)
  - Parameters: username - The username of the member to remove.
  - Return Type: boolean

- Search Member
  - Description: Searches for a member by name or surname.
  - Method: searchMember()
  - Parameters: username - The username of the member to remove.
  - Return Type: void
  - Throws: IOException

- Search Member by Username
  - Description: Searches for a member by username.
  - Method: searchMemberByUsername(String searchKey)
  - Parameters: searchKey - The username to search for.
  - Return Type: Member
  - Throws: IOException

- Search Member by Name and Surname
  - Description: Searches for a member by name and surname.
  - Method: searchMemberBy_NameandSurname(String name, String surname)

- Parameters:
    - name - The name of the member.
    - surname - The surname of the member.
- Return Type: Member
- Throws: IOException

o Search Member by Username GUI
- Description: Searches for a member by username, name, or email for GUI purposes.
- Method: searchMemberByUsername_GUI(String searchKey)
- Parameters: searchKey - The username, name, or email to search for.
- Return Type: boolean

o View Found Member
- Description: Displays information of the found member.
- Method: viewFoundMember(String searchKey)
- Parameters: searchKey - The username of the member to find.
- Return Type: String

o Print Member
- Description: Prints details of all members.
- Method: printMembers()
- Parameters: None
- Return Type: void

o Print Member GUI
- Description: Returns a formatted string containing details of all members.
- Method: printMembers_GUI()
- Parameters: None
- Return Type: String

- Add to Array Member
  - Description: Updates member information with the specified details.
  - Method: addToArrayMember(String username, String newName, String newSurname, String newPhoneNumber, String newEmail, String newUsername, String newPassword)
  - Parameters: Parameters as described in the method signature.
  - Return Type: void
  - Throws: IOException

- View Loans
  - Description: View all loan records and optionally calculate fines for overdue books.
  - Method: viewLoans()
  - Parameters: None
  - Return Type: void

- View Member Loans
  - Description: View all loans associated with a specific member.
  - Method: viewMemberLoans(String username)
  - Parameters: username - The username of the member.
  - Return Type: void

- View Loans GUI
  - Description: View all loans and return a formatted string.
  - Method: viewLoans_GUI(String username)
  - Parameters: None
  - Return Type: String

- View Member Loans GUI
  - Description: View all loans associated with a specific member and return a formatted string.

- Method: viewMemberLoans_GUI(String username)
- Parameters: username - The username of the member.
- Return Type: String

o Delete Loans
  - Description: Delete loan records associated with a specific member and book.
  - Method: deleteLoans(String username, String ISBN)
  - Parameters:
    - username - The username of the member.
    - ISBN - The ISBN of the book.
  - Return Type: void
  - Throws: IOException

o Fine Loan Search
  - Description: Search Fine of loan records for a specific member's username
  - Method: fineLoanSearch(String username)
  - Parameters: username - The username of the member.
  - Return Type: String

o Check In Book
  - Description: Check if a member has any overdue books and calculate the total fine.
  - Method: checkInBook(String username, String ISBN)
  - Parameters:
    - username - The username of the member.
    - ISBN - The ISBN of the book.
  - Return Type: double
  - Throws: IOException

o Check In Book
  - Description: Check out a book for a member.

- Method: CheckOutBook(String username, String ISBN)
- Parameters:
    - username - The username of the member.
    - ISBN - The ISBN of the book.
- Return Type: boolean
- Throws: IOException

o Process Book Transaction

- Description: Process a book transaction (check-in or check-out) for a member.
- Method: processBookTransaction(String username, boolean checkOut)
- Parameters:
    - username - The username of the member.
    - checkOut - True if checking out a book, false if checking in.Return Type: boolean
- Throws: IOException

❖ CSV

o Add Book

- Description: Adds a book to the CSV records.
- Method: addBook(Book book)
- Parameters: book - The book to add
- Return Type: Void

o Add Staff

- Description: Adds a staff member to the CSV records.
- Method: addStaff(Staff staff)
- Parameters: staff - The staff member to add
- Return Type: Void

o Add Member

- Description: Adds a member to the CSV records.

- Method: addMember(Member member)
- Parameters: member - The member to add
- Return Type: Void

o Add Loan
- Description: Adds a loan record to the CSV records.
- Method: addLoan(Loan loan)
- Parameters: loan - The loan record to add
- Return Type: Void

o Delete Loan
- Description: Deletes a loan record from the CSV records.
- Method: deleteLoan(Loan loan)
- Parameters: loan - The loan record to delete
- Return Type: Void

o Get Books
- Description: Retrieves the list of books.
- Method: getBooks()
- Parameters: None
- Return Type: List<Book>

o Get Staff
- Description: Retrieves the list of staff members.
- Method: getStaff()
- Parameters: None
- Return Type: List<Staff>

o Get Members
- Description: Retrieves the list of members.
- Method: getMembers()

- Parameters: None
- Return Type: List<Member>

- Get Loan Records
  - Description: Retrieves the list of loan records.
  - Method: getLoanRecords()
  - Parameters: None
  - Return Type: List<Loan>

❖ CSV_Reader

- Read Book
  - Description: Reads book information from a CSV file and adds them to the records.
  - Method: readBook(String filePath)
  - Parameters: filePath - The path to the book CSV file
  - Return Type: Void
  - Throws: IOException

- Read Staff
  - Description: Reads staff information from a CSV file and adds them to the records.
  - Method: readStaff(String filePath)
  - Parameters: filePath - The path to the staff CSV file
  - Return Type: Void
  - Throws: IOException

- Read Member
  - Description: Reads member information from a CSV file and adds them to the records.
  - Method: readMember(String filePath)
  - Parameters: filePath - The path to the member CSV file

- Return Type: Void
- Throws: IOException

- Read Loan Records
  - Description: Reads loan records from a CSV file.
  - Method: readLoanRecords(String fileName)
  - Parameters: fileName - The name of the loan record CSV file
  - Return Type: String[]
  - Throws: IOException

- ID Found
  - Description: Checks for the last ID found in a CSV file for staff or member.
  - Method: IDFound(String filePath)
  - Parameters: filePath - The path to the CSV file
  - Return Type: int
  - Throws: IOException

❖ CSV_Writer

- Write Book
  - Description: Writes book information to a CSV file.
  - Method: writeBook(String filename, String title, String author, String ISBN, String publishYear, String img, String publisher, String img2, String img3)
  - Parameters:
    - filename - The name of the file to write to.
    - title - The title of the book.
    - author - The author of the book.
    - ISBN - The ISBN of the book.
    - publishYear - The publish year of the book.
    - img - The image of the book.

        - publisher - The publisher of the book.
        - img2 - The second image of the book.
        - img3 - The third image of the book.
    - Return Type: Void

- Write Staff
    - Description: Writes staff information to a CSV file.
    - Method: writeStaff(String filename, int ID, String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary, String $)
    - Parameters: Parameters as described in the method signature.
    - Return Type: Void

- Write Member
    - Description: Writes member information to a CSV file.
    - Method: writeMember(String filename, int ID, String name, String surname, String phoneNumber, String email, String username, String password)
    - Parameters: Parameters as described in the method signature.
    - Return Type: Void

- Update Staff
    - Description: Updates staff information in a CSV file.
    - Method: updateStaff(String filename, String username, String newName, String newSurname, String newPhoneNumber, String newEmail, String newAddress, String newUsername, String newPassword, double newSalary)
    - Parameters: Parameters as described in the method signature.
    - Return Type: Void

- Update Member
    - Description: Updates member information in a CSV file.

- Method: updateMember(String filename, String username, String newName, String newSurname, String newPhoneNumber, String newEmail, String newUsername, String newPassword)
- Parameters: Parameters as described in the method signature.
- Return Type: Void

o Write or Update Loan Records
- Description: Writes or updates loan records in a CSV file.
- Method: writeOrUpdateLoanRecords(String filename, List<Loan> loans)
- Parameters:
  - filename - The name of the file to write to or update.
  - loan - The list of loan records to write or update.
- Return Type: Void

o Write Loan Record
- Description: Writes a new loan record to a CSV file.
- Method: writeLoanRecord(String filename, String memberName, String memberSurname, String bookName, String dueDate, double fine, java.util.Date date)
- Parameters: Parameters as described in the method signature.
- Return Type: Void

o Remove Entry From CSV
- Description: Removes an entry from a CSV file.
- Method: removeEntryFromCSV(String filename, String valueToRemove)
- Parameters: Parameters as described in the method signature.
- Return Type: Void

o Remove Loan Entry
- Description: Removes a loan entry from a CSV file.

- Method: remove_CSV_loan(String filename, String name, String surname, String bookname)
- Parameters: Parameters as described in the method signature.
- Return Type: Void

2.3 Performance Requirements

Performance requirements for the Library Management System are essential to ensure efficient operation and responsiveness. These requirements focus on key aspects such as system speed, scalability, and reliability. The following performance requirements are identified for the system:

- Response Time
  - The system should respond to user interactions within a maximum of 2 seconds under normal load conditions.
  - GUI components should render smoothly without noticeable lag or delay.

- Scalability
  - The system should be capable of handling a large volume of concurrent users without significant degradation in performance.
  - As the number of books, members, and transactions increases, the system should scale seamlessly to accommodate the growth.

- Reliability
  - The system should operate reliably without frequent crashes or downtime.
  - Database transactions should be ACID-compliant to ensure data integrity and consistency.

- Database Performance
  - Database queries should execute efficiently, with response times kept to a minimum.

- Indexing and optimization techniques should be employed to improve query performance and reduce latency.

- Error Handling
  - The system should gracefully handle errors and exceptions to prevent data loss or corruption.
  - Error messages should be informative and user-friendly, guiding users on how to resolve issues effectively.

- Resource Utilization
  - The system should efficiently utilize system resources such as CPU, memory, and disk space.
  - Resource usage should be monitored and optimized to prevent bottlenecks and ensure optimal performance.

2.4 Logical Database/File System Requirements

The Library Management System relies on a combination of CSV files to store data related to books, staff, members, loans, and administrators. These files serve as the logical database for the system, providing a structured format for storing and retrieving information. The following requirements outline the specifications for the logical database and file system:

- Data Organization
  - Each CSV file corresponds to a specific entity in the system, such as books, staff, members, loans, and administrators.
  - Data within each CSV file is organized into rows and columns, with each row representing a single record and each column representing a field or attribute.

- File Naming Convention

- CSV files follow a consistent naming convention to facilitate easy identification and management.
- File names should be descriptive and indicative of the type of data they contain, such as "books.csv," "staff.csv," "members.csv," "loans.csv," and "admins.csv."

- Data Integrity
  - CSV files should maintain data integrity by ensuring that each record is complete and accurate.
  - Data validation techniques should be employed to prevent inconsistencies and errors in the data.

- Data Storage
  - CSV files are stored in a designated directory or folder within the system's file system.
  - File paths are configured within the system to facilitate seamless access to the CSV files.

- Backup and Recovery
  - Regular backups of CSV files are performed to prevent data loss in the event of system failure or corruption.
  - Backup files are stored in a secure location and can be easily restored when needed.

- Security
  - Access to CSV files is restricted to authorized users to maintain data confidentiality and integrity.
  - File permissions are configured to ensure that only authorized personnel can read from and write to the CSV files.

- Performance
  - CSV files should be optimized for efficient read and write operations to minimize latency and improve system performance.

- Indexing and caching mechanisms may be employed to enhance data retrieval speed and responsiveness.

- Scalability
  - The system should be able to handle a growing volume of data stored in CSV files without compromising performance or reliability.
  - File organization and storage strategies should be designed to accommodate scalability requirements effectively.

2.5 Design Constraints

Design constraints for the Library Management System impose certain limitations and considerations that influence its design and implementation. These constraints include:

- Technological Constraints
  - The system must be developed using Java programming language and Eclipse IDE.
  - Limited support for advanced features and libraries due to platform restrictions.

- Hardware Limitations
  - System performance may be impacted by the hardware specifications of the host machine, such as CPU, memory, and disk space.
  - Compatibility with legacy hardware may restrict the use of resource-intensive features.

- Software Dependencies
  - Dependencies on third-party libraries and frameworks, such as Swing and AWT for GUI development, impose constraints on system architecture and design choices.

- Compatibility issues with future versions of software dependencies may require updates and maintenance.

- Regulatory Compliance
  - Compliance with data protection regulations, such as GDPR or HIPAA, imposes constraints on data handling, storage, and security measures.
  - Legal requirements and industry standards must be adhered to ensure regulatory compliance.

- Time Constraints
  - Project deadlines and timelines impose constraints on the development and delivery of the system.
  - Iterative development and agile methodologies may be employed to meet project milestones and deliverables.

- User Requirements
  - System design must align with user requirements and expectations, limiting flexibility in design choices.
  - User feedback and usability testing may influence design decisions and feature prioritization.

- Integration Constraints
  - Integration with existing systems and databases may impose constraints on data formats, protocols, and compatibility.
  - Interoperability with external systems and APIs must be considered during system design and implementation.

2.6 Software System Quality Attributes

Software system quality attributes define the characteristics and qualities of the Library Management System that contribute to its overall performance, reliability, and usability. These attributes include:

- Reliability
  - The system should operate reliably under normal and exceptional conditions, minimizing the occurrence of failures or errors.
  - Reliable data storage and transaction processing ensure the integrity and consistency of library records.

- Performance
  - The system should perform efficiently, responding promptly to user interactions and queries.
  - Optimized data retrieval and processing speed enhance user experience and system responsiveness.

- Usability
  - The system should be user-friendly and intuitive, allowing users to navigate and perform tasks with ease.
  - Clear and intuitive interfaces, along with informative feedback, enhance usability and user satisfaction.

- Scalability
  - The system should be scalable, capable of accommodating growth in the number of users, books, and transactions.
  - Scalable architecture and efficient resource utilization ensure that the system can handle increasing loads without degradation in performance.

- Security
  - The system should prioritize data security, protecting sensitive information such as user credentials and personal data.

- Robust authentication mechanisms and encryption techniques safeguard data against unauthorized access and breaches.

- Maintainability
  - The system should be maintainable, allowing for easy updates, modifications, and enhancements.
  - Modular design, clear documentation, standardized coding practices, and TestNG for efficient testing facilitate system maintenance and troubleshooting.

- Flexibility
  - The system should be flexible, capable of adapting to changing requirements and environments.
  - Configurable settings and customizable features enable users to tailor the system to their specific needs and preferences.

- Portability
  - The system should be portable, able to run on different platforms and environments.
  - Cross-platform compatibility and adherence to industry standards ensure that the system can be deployed across various operating systems and hardware configurations.

- Interoperability
  - The system should be interoperable, able to integrate seamlessly with external systems and services.
  - Standardized data formats and communication protocols enable interoperability with other library systems and third-party applications.

2.6 Object Oriented Models

Object-oriented models provide a structured representation of the system's entities, their attributes, and relationships. These models help visualize the system's architecture and behavior, facilitating design and implementation. The following sections detail the analysis class model (static model) and analysis collaborations (dynamic model) for the Library Management System:

- 2.7.1 Analysis Class Model (Static Model)

The analysis class model represents the static structure of the Library Management System, depicting the classes, attributes, and relationships among entities within the system. The classes identified in the analysis class model serve as blueprints for creating objects and encapsulating behavior and data. Key classes and their attributes include:

- Book
  - Attributes: title, author, ISBN, publishYear, quantity, availability, img
  - Methods: getters/setters of the attributes, decreaseTotalQuantity, IncreaseTotalQuantity

- Admin
  - Attributes: ID, name, surname, phone number, adress, email, username, password, salary
  - Methods: Authenticate, manageBooks, manageMembers, manageStaff, calculateSalary, manageLoan, accountManagement

- Staff
  - Attributes: ID, name, surname, phone number, address, email, username, password, salary
  - Methods: Authenticate, manageBooks, manageMembers, manageLoan, accountManagement,

- Member
    - Attributes: ID, name, surname, phone number, email, username, password
    - Methods: Authenticate, Register, accountManagement, viewBooks, SearchBooks, checkInOutBooks, viewLoans

- Loan
    - Attributes: Loan ID, book ISBN, member username, loan date, due date, return date, status
    - Methods: Issue, return, renew, calculateFine

- 2.7.2 Analysis Collaborations (Dynamic Model)

The analysis collaborations model depicts the dynamic interactions and collaborations among objects within the Library Management System. It illustrates the flow of messages and actions between objects during system operation. Key collaborations include:

- Book Checkout
    - Member requests to borrow a book
    - System checks book availability
    - If available, system issues a loan and updates book status
    - If not available, system notifies member of unavailability

- Book Return
    - Member returns a borrowed book
    - System updates book availability and loan status
    - System calculates any applicable fines for late returns

- User Authentication
    - User enters username and password
    - System verifies credentials
    - If authentication is successful, user gains access to system functionalities

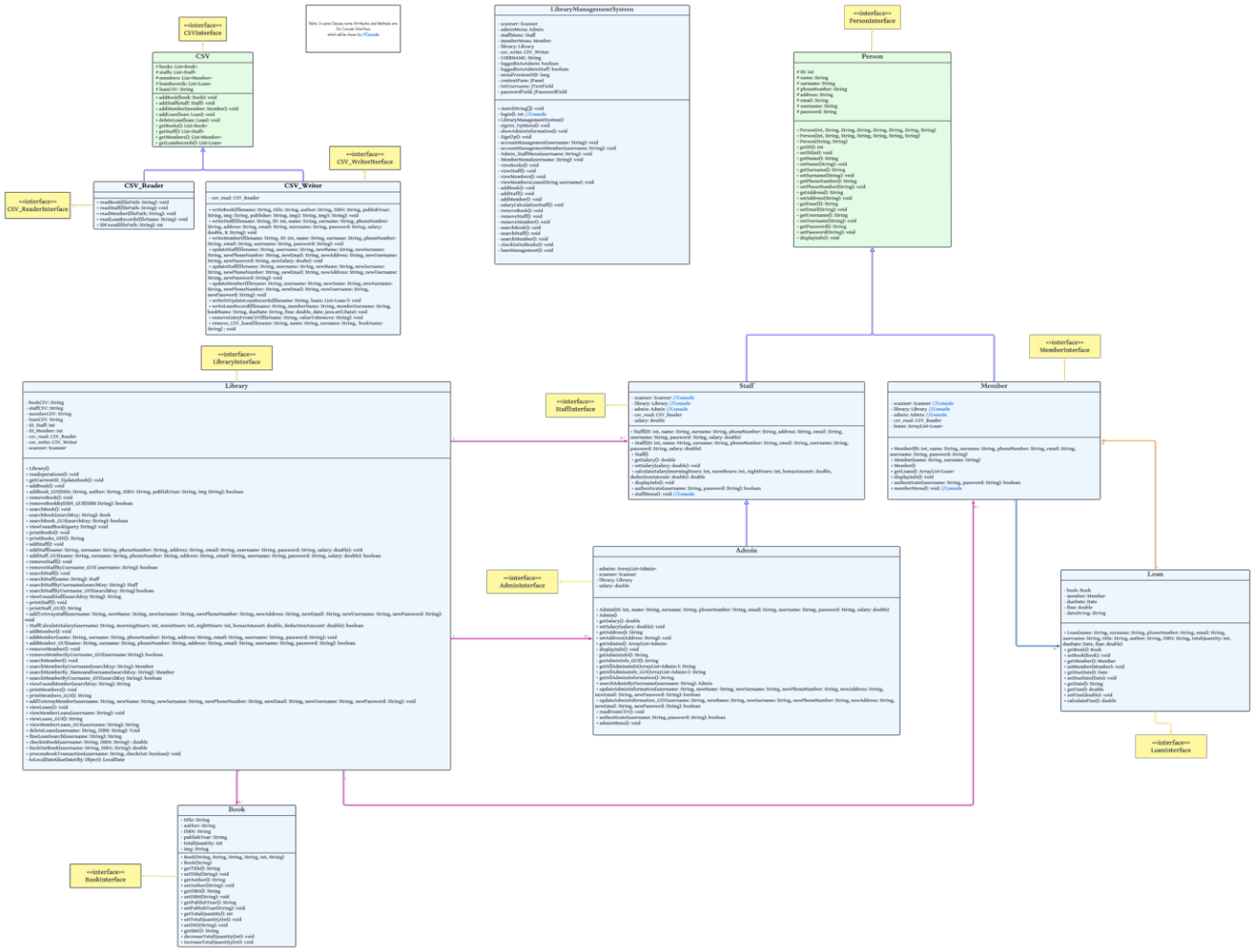
- If authentication fails, system denies access and prompts user to retry

- Admin Operations
  - Admin performs various management operations, such as adding/removing books, managing members and staff, and calculating salary and loan management
  - System validates admin credentials and permissions before executing operations

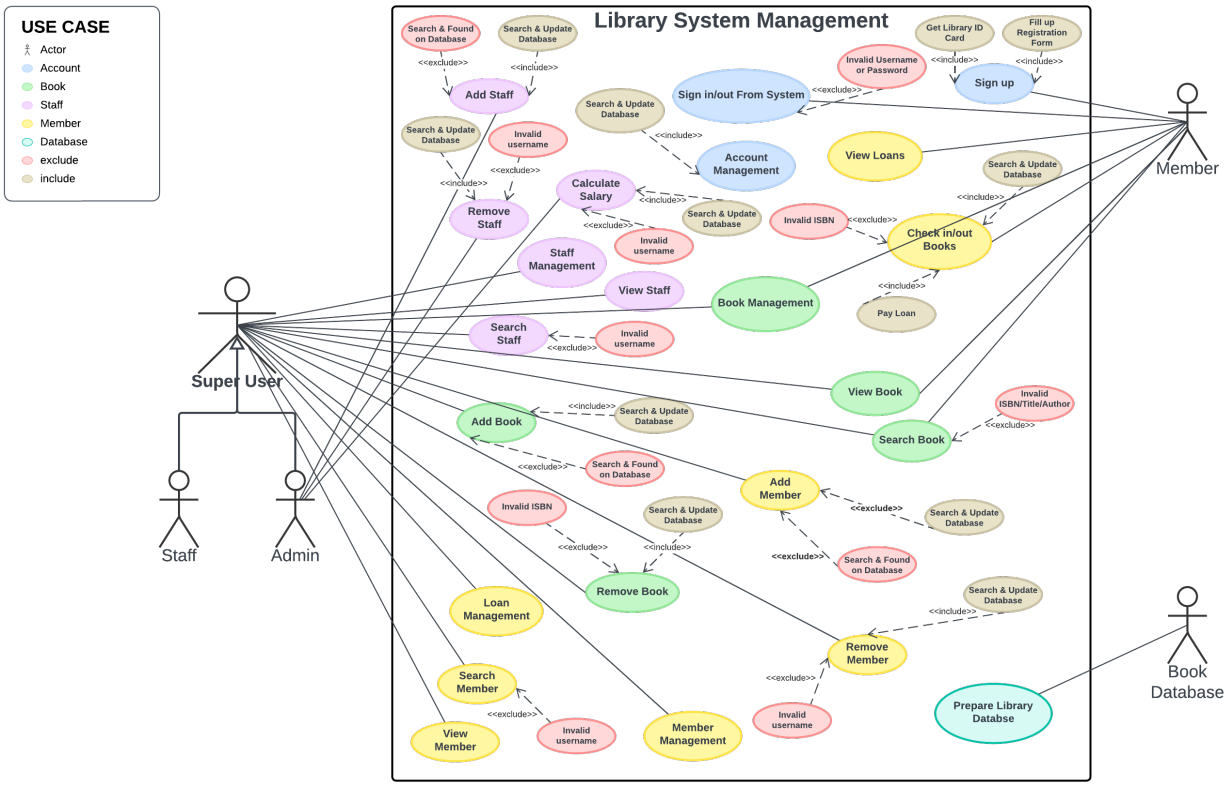
CHAPTER THREE

UML DIAGRAMS

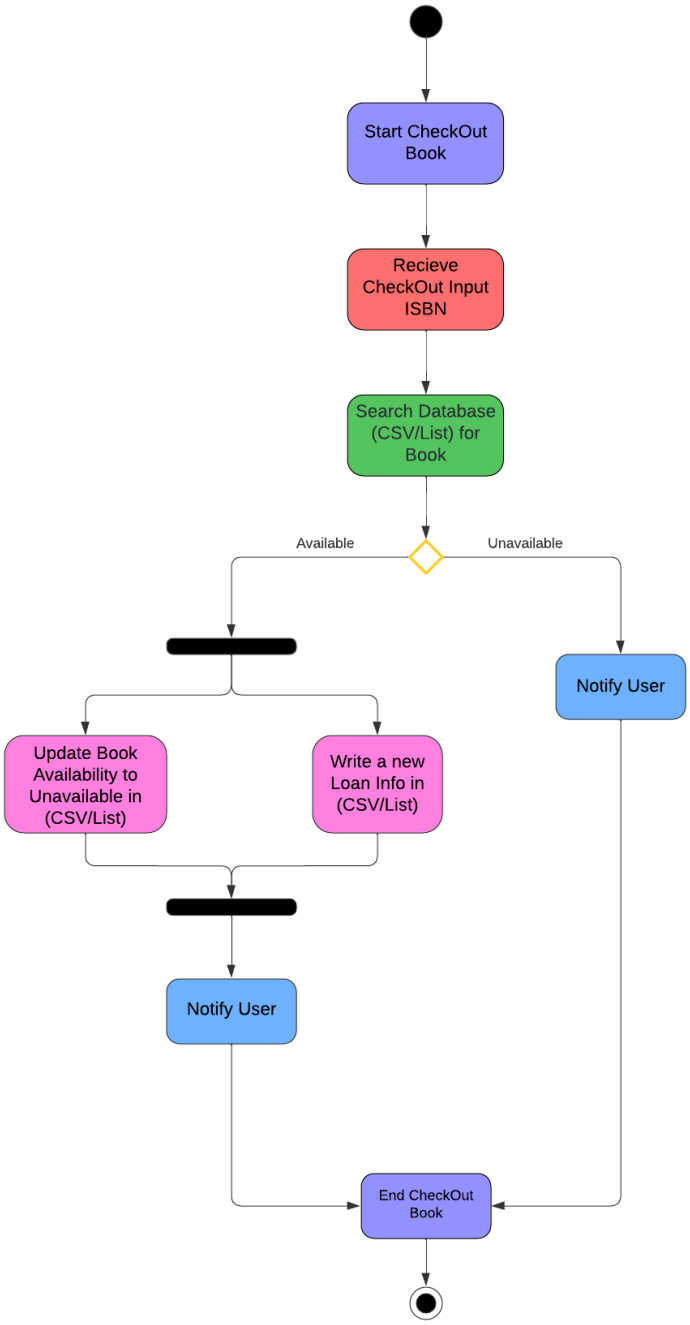
- Class Diagram:
    - The class diagram for the library management project depicts the various entities and their relationships within the system. The "Admin" class represents the administrative personnel responsible for overseeing the library operations, while the "Book" class encapsulates information about the library's collection. The "CSVReader" class facilitates the reading of data from CSV files, aiding in data management. The "Library" class serves as the central hub for managing books, members, and staff. The "Loan" class manages the loaning process of books to members. The "Member" class represents individuals registered with the library, while the "Staff" class encapsulates information about library staff members. Lastly, the "Person" class serves as a superclass, containing common attributes and methods shared by members and staff.



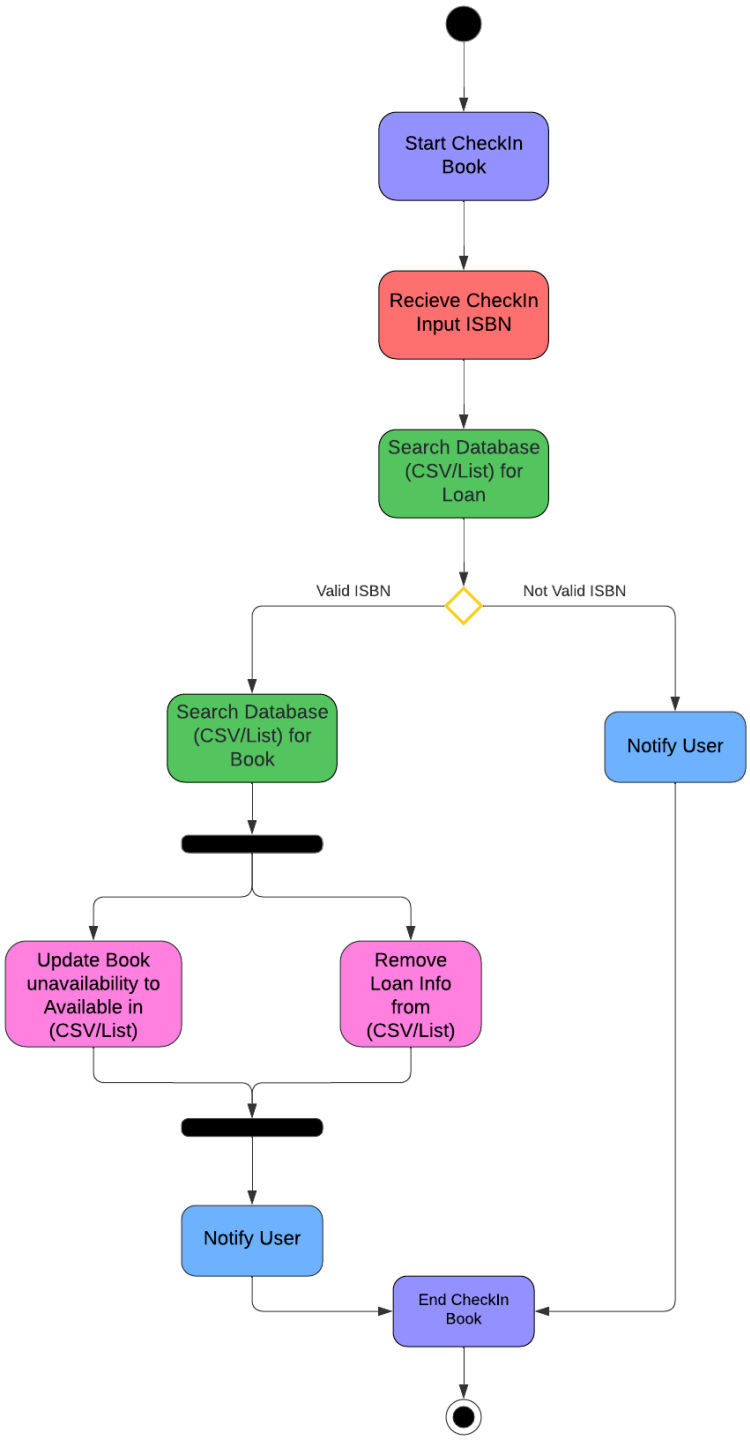
- Use Case Diagram:
    - The library management system is depicted through a use case diagram showcasing interactions between actors like "Admin," "Member," and "Staff" and system functionalities such as "Add Book," "Remove Book," "Register Member," "Issue Book," "Return Book," and "Generate Report." These use cases encapsulate core functionalities allowing administrators to manage inventory, register new members, facilitate book loans, and generate reports for analysis. Moreover, the system supports efficient book lending operations and staff and member management, enabling users to add, remove, search, and view books, staff, and members. Administrators can calculate staff salaries, while users manage book loans, including extensions, returns, and terminations, with automated fine calculation for overdue books and check-in/out functionalities. This comprehensive feature set empowers library staff to manage resources effectively, ensuring smooth operations and enhanced user experience.



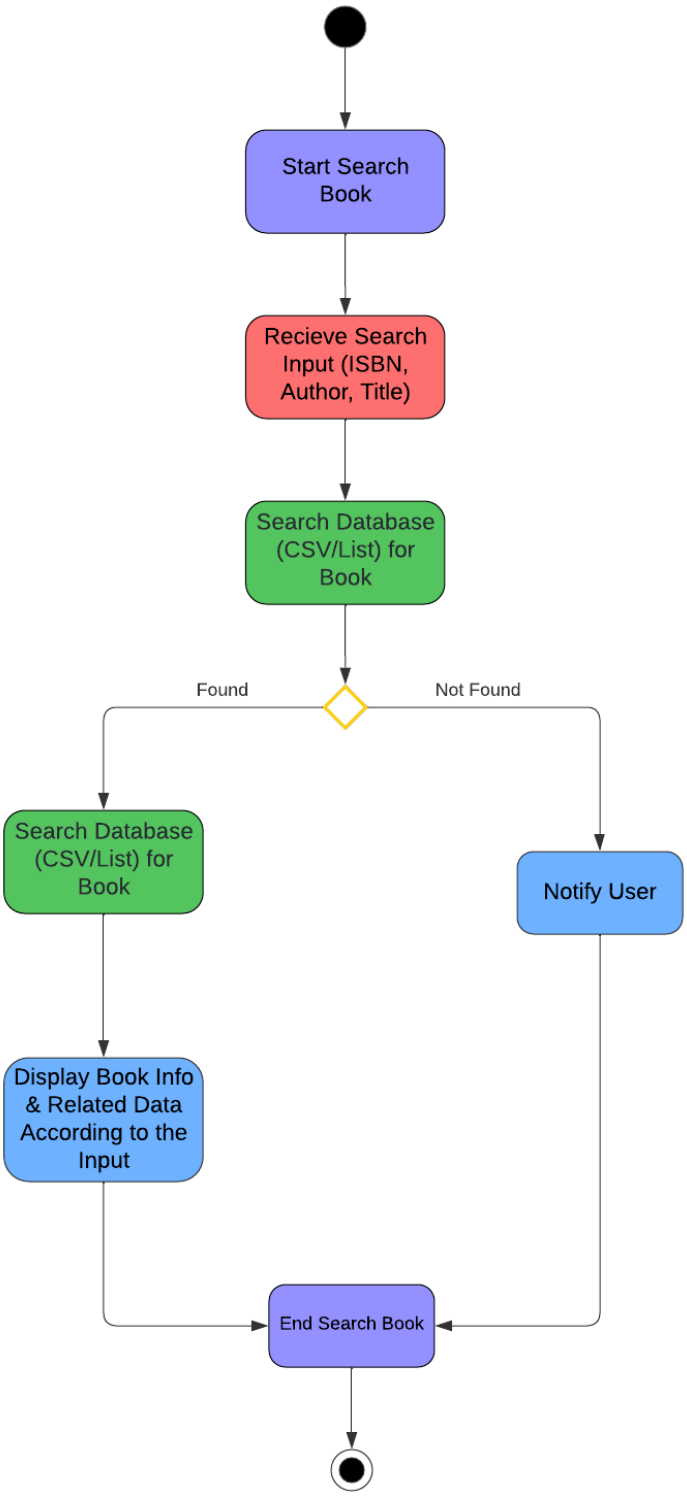
- Activity Diagram:
  - CheckOut Book
    - Firstly, the user provides the ISBN. Then, the system searches for the book in the database. If the book is not available, the user is notified. If the book is available, the parallel process begins: it records the client's information in the loan CSV and list, and updates the book's status to make it unavailable. Finally, the operation concludes.



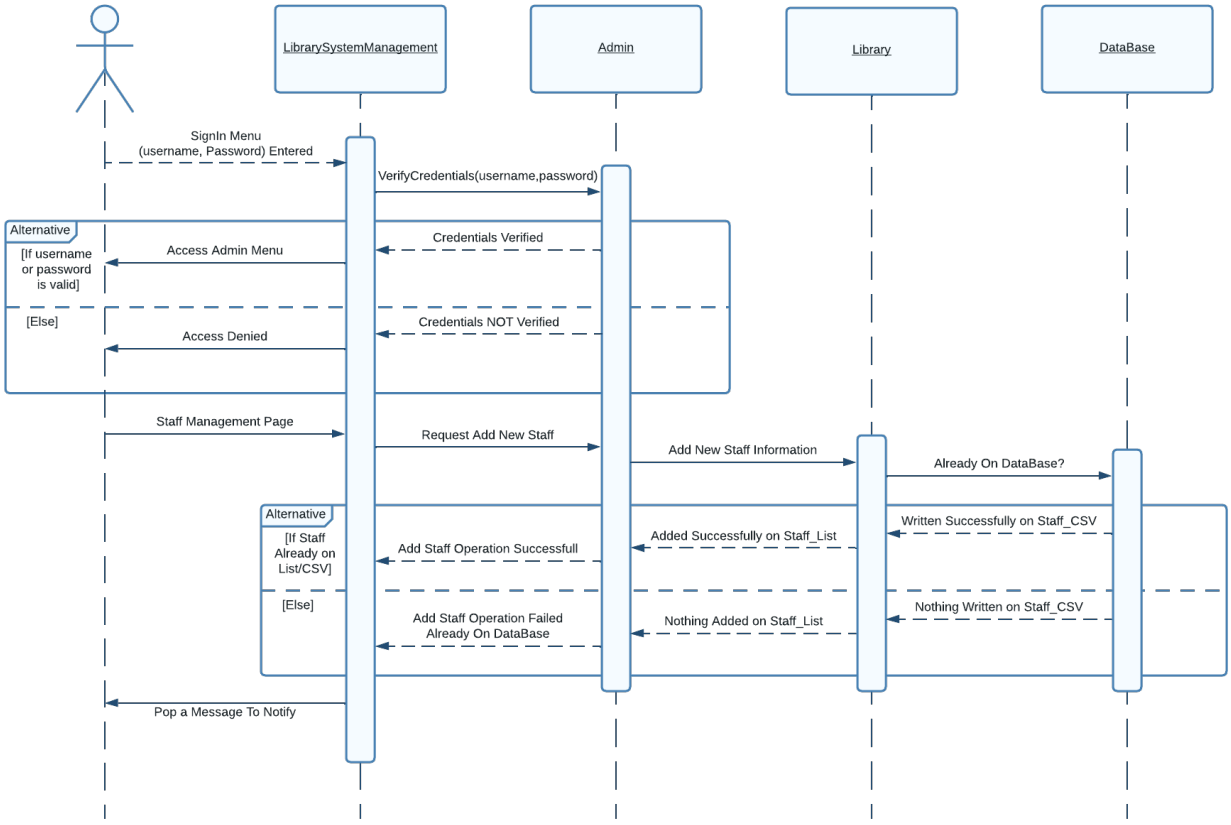
- CheckIn Book
  - Firstly, we obtain the ISBN from the user. Then, we check whether it is valid or not. If it is not valid, we notify the user. If it is valid, we search for the book in the CSV and list it. Simultaneously, we remove the user from the loan CSV and loan list, and update the book's availability. Once these parallel tasks are completed, we return the user's successful check-in, and all operations conclude.



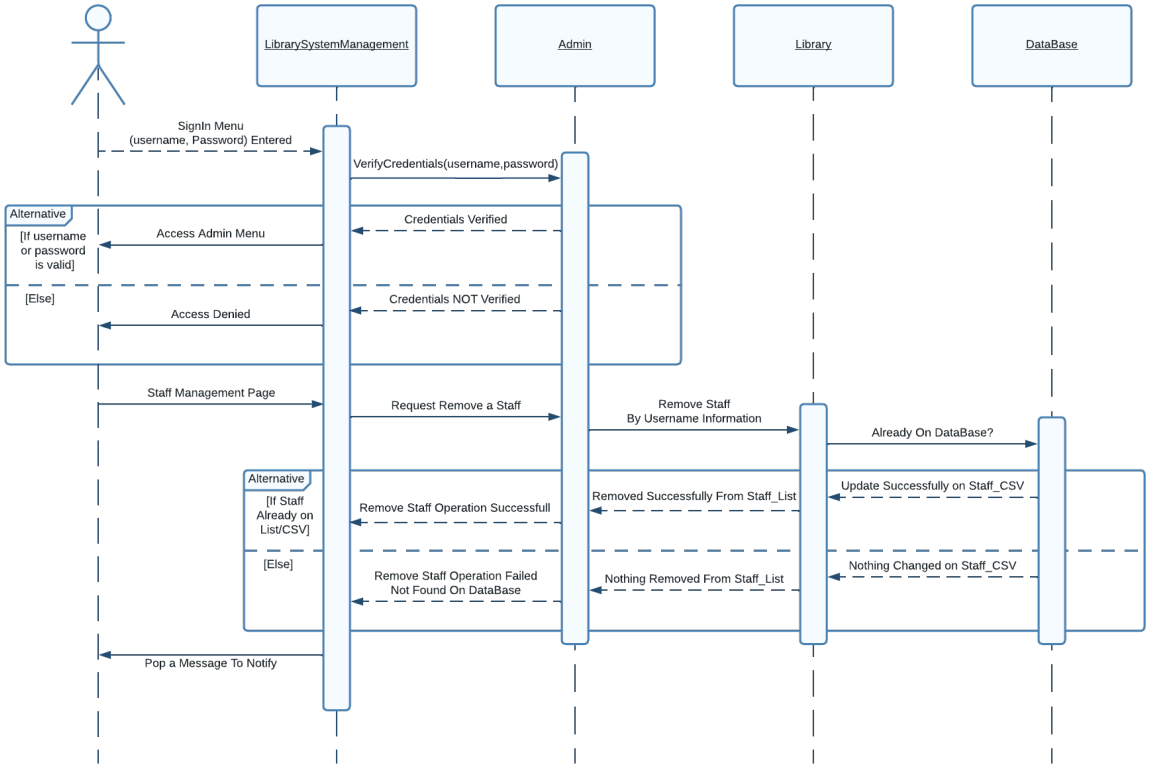
- Search Book
  - Initially, we gather information from the user: ISBN, author, and title. Then, we check if the book exists in our list and database. If we couldn't find it, we notify the user. If we do find it, we display the information of the searched book.



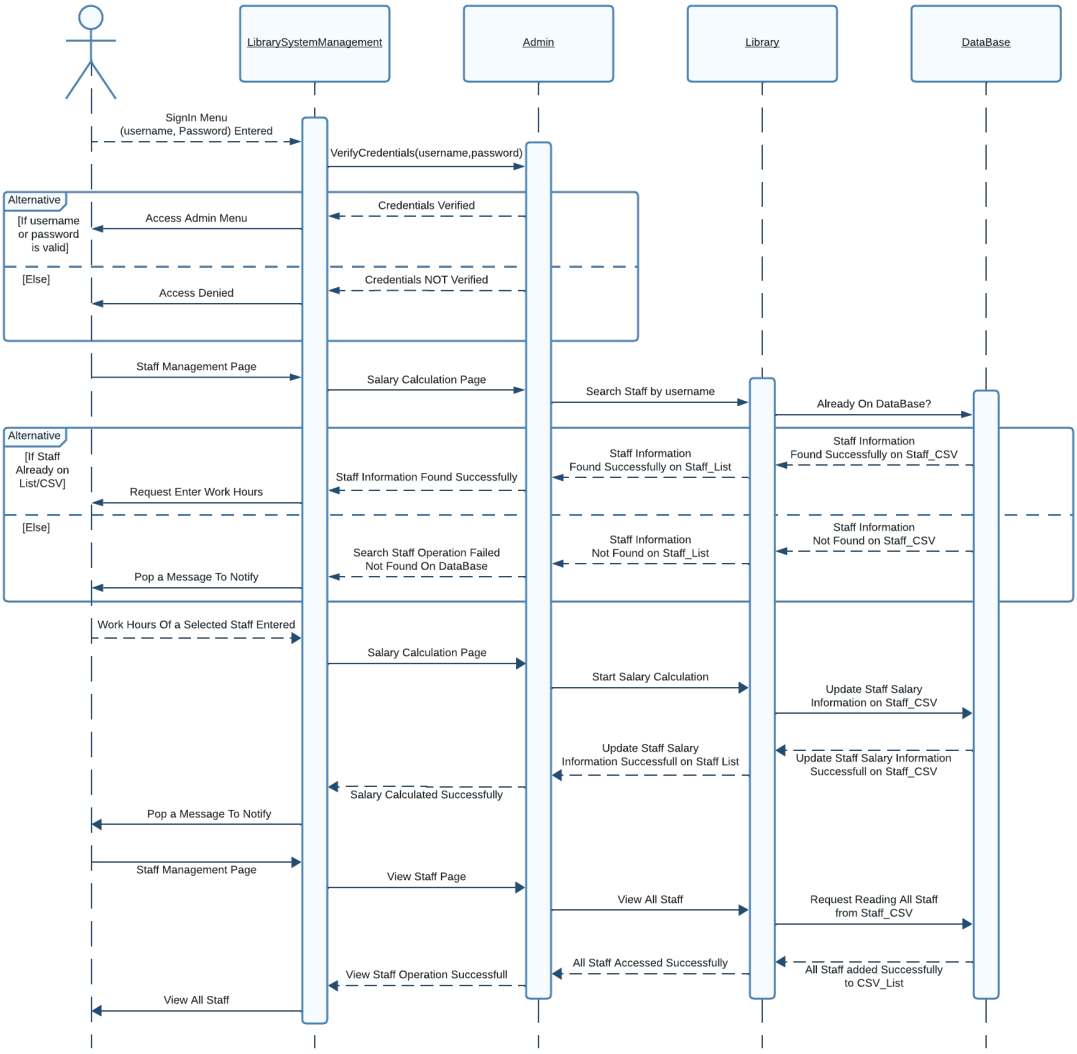
- Sequence Diagram
  - Add Staff Operation
    - In the sequence diagram, the process starts with the Admin entering their username and password to access the Admin page. The system verifies these credentials; if the verification is successful, the Admin gains access to the Admin Menu page, otherwise, an "Access Denied" message is displayed. Once accessing the Admin Menu, the Admin selects "Staff Management" to initiate the addition of a new staff member. The Admin inputs the required staff information and submits the addition request. The system then checks if the staff member already exists; if so, it displays a "Staff Exists" message. If no such staff member exists, the system adds the new staff and confirms the action with an "Added Successfully" message, completing the process.



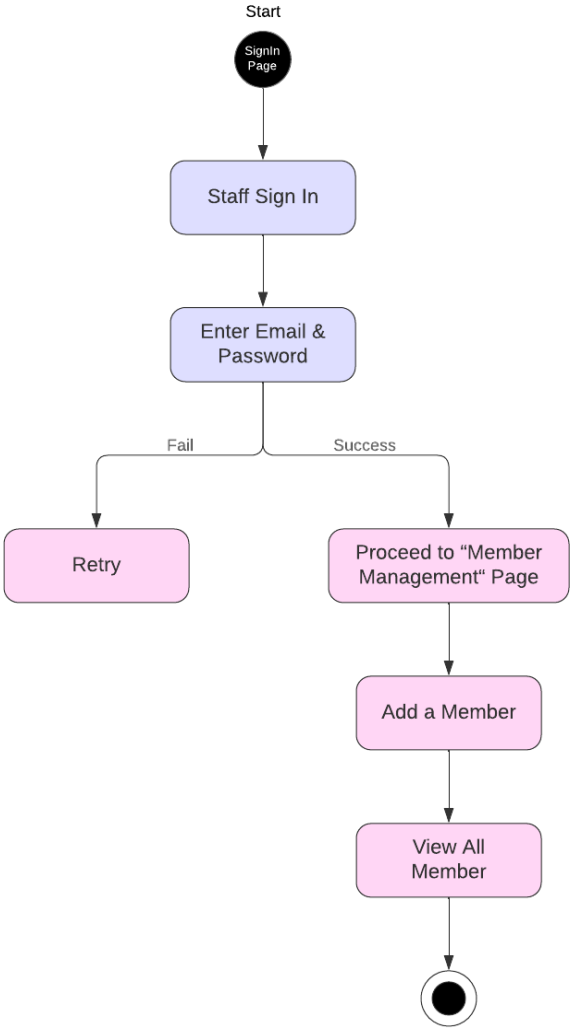
- Remove Staff Operation
  - In this diagram, we aim to explain the process by which an Administrator can access the Admin page and perform staff removal operations. Initially, the user, identified as the Admin, enters their username and password. The system verifies these credentials; if verified, the Admin gains access to the Admin Menu page. If not, an "Access Denied" message is returned. Upon accessing the Admin menu, user select the "Staff Management" option to remove a staff member, then enters the staff member's username, prompting the system to check if this staff member exists in the database or not. If the staff member does not exist in the database, the system returns the message "Failed to Remove" and nothing changes in Staff_CSV. If the staff member exists, they are removed from the database, and the system returns "Removed Successfully" which Staff_CSV will be updated.



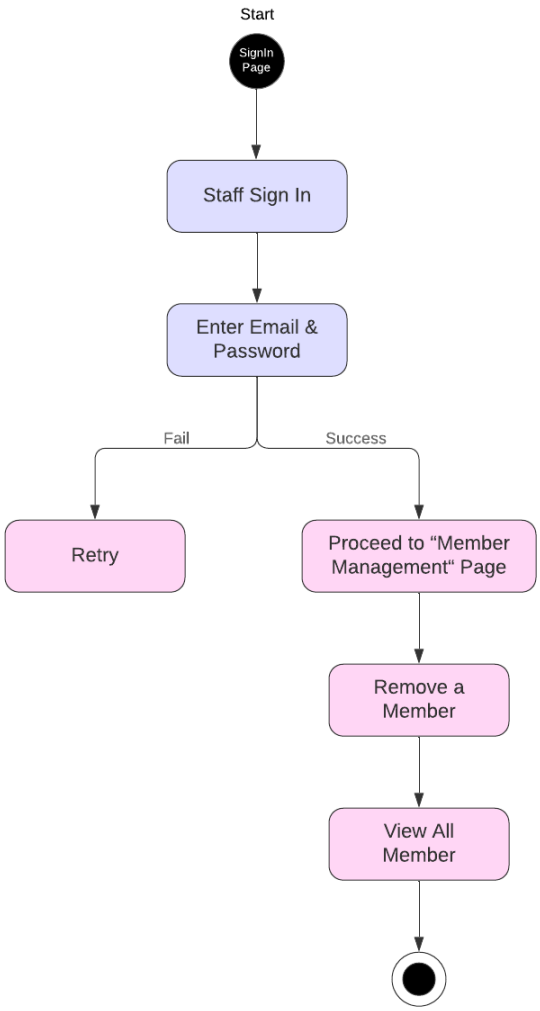
- Salary Calculation of a Staff Then View All Staff To see Changes
  - In this diagram, we explore the functionality of the Admin panel that facilitates the management of staff-related operations. Initially, the user, identified as the Admin, logs in by entering their username and password. Upon successful verification of these credentials, the Admin gains access to the Admin Menu page; if not, an "Access Denied" message is issued. Within the Admin Menu, the Admin can choose to calculate salaries by selecting "Staff Management." The system then prompts the Admin to enter the staff member's username to check for their existence in the database. If the staff member exists, the system confirms "Staff Exists"; if not, it returns "Staff Does Not Exist." Once a staff member is confirmed, the Admin enters the number of work hours. The system then calculates the salary based on the entered hours and updates this information in both the form list and the CSV file. Another key operation available to the Admin is "View Staff." When selected, the system displays a list of all staff members along with any recently updated information.



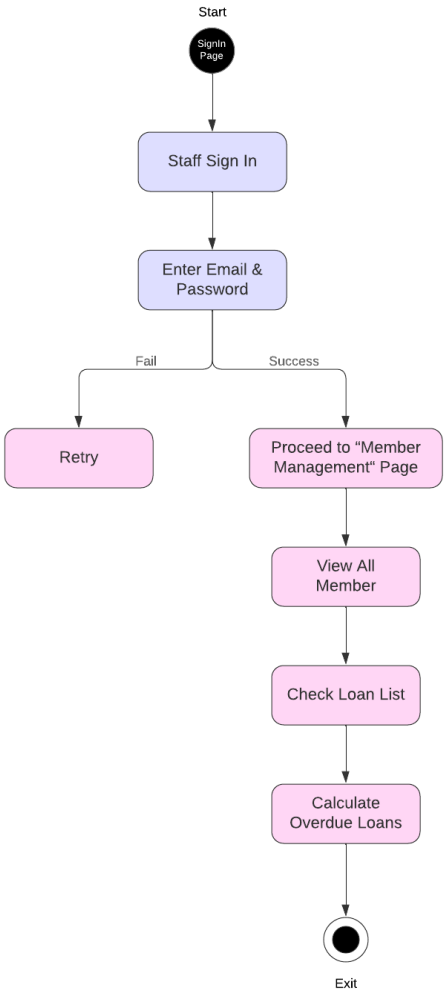
- State Diagram
    - Adding a Member with Staff Account
        - In this state diagram, the user is assumed to be a staff of the library. The process begins when the sign-in page is displayed, prompting the staff to enter their username and password. The system then checks these credentials. If the user is not found, the system does not proceed further and returns an error message indicating the user was not found. Conversely, if the credentials are valid and the staff member is recognized, the user is directed to the main menu page. From here, the staff member can navigate to "Member Management," where they have the options to "Add Member" and "View All Members." Selecting "View All Members" displays a comprehensive list of all members, completing the process. This diagram outlines the workflow from sign-in to member management, illustrating both the navigation and possible outcomes based on the staff member's actions.



- Remove a Member with Staff Account
  - In this state diagram, the user is assumed to be a staff of library. The process begins with the sign-in page, where the staff is prompted to enter their username and password. Upon entering their credentials, the system checks for validity. If the credentials do not match and the user is not found, the system halts further action and returns an error message indicating the user was not found. If the credentials are verified successfully and the staff member is recognized, the user advances to the main menu page. Here, the staff can navigate to "Member Management," where they now have the options to "Delete Member" and "View All Members." Selecting "Delete Member" allows the staff member to remove an existing member from the system. Following this action, they can also choose "View All Members" to confirm the current list of members, ensuring the deletion was successful. This diagram illustrates the workflow from sign-in to member management, specifically focusing on the deletion of members and confirming the outcomes through the member viewing option.



- Managing Member’a Loan with Staff Account
  - In this state diagram, the user is assumed to be a staff of library. The process initiates when the sign-in page appears, prompting the staff to enter their username and password. After the credentials are entered, the system verifies them. If no matching user is found, the system remains inactive and returns an error message indicating that the user was not found. However, if the staff member is successfully authenticated, they are directed to the main menu page. From this menu, the staff selects "Member Management," where they choose the "View All Members" option to display a list of all members. Subsequently, the diagram illustrates that the staff then proceeds to check the "Loan List" for these members. The final step involves the system calculating any overdue loans associated with these members, completing the workflow as detailed in the state diagram. This sequence effectively maps out the staff's navigation through the system, from authentication to complex functions like managing member details and loan calculations.



CHAPTER FOUR

IMPLEMENTATION

❖ CSV Class

This Java class serves as a base class for handling CSV (Comma-Separated Values) operations related to managing books, staff, members, and loan records. This class provides a foundation for CSV handling in a library management system, offering methods for manipulating different entities and interacting with CSV files.

- **Attributes:**
  - books, staffs, members, loanRecords: These are lists to store instances of Book, Staff, Member, and Loan respectively.
  - loanCSV: A string representing the filename of the CSV file for loan records.
- **Methods:**
  - addBook(Book book): Adds a Book object to the books list.
  - addStaff(Staff staff): Adds a Staff object to the staffs list.
  - addMember(Member member): Adds a Member object to the members list.
  - addLoan(Loan loan): Adds a Loan object to the loanRecords list.
  - deleteLoan(Loan loan): Removes a Loan object from the loanRecords list.
  - Getters:
    - getBooks(): Returns the list of books.
    - getStaff(): Returns the list of staff.
    - getMembers(): Returns the list of members.
    - getLoanRecords(): Returns the list of loan records.
- **Inheritance:**
  - The class implements CSVInterface, indicating it must provide implementations for methods declared in that interface.
- **Additional Notes:**
  - This class is abstract, meaning it cannot be instantiated directly but can be extended by other classes.

- It provides basic CRUD (Create, Read, Update, Delete) operations for managing books, staff, members, and loan records.
- The implementation of methods like adding, deleting, and getting records suggests that this class serves as a data repository for CSV-based storage of library-related entities.

❖ CSV_Reader class

This class extends the previously defined CSV class and implements the CSV_ReaderInterface. It provides methods for reading data from CSV files and populating the lists of books, staff, and members. This class encapsulates the functionality for reading different types of data from CSV files and is crucial for initializing the system with existing data during startup or importing data from external sources.

➢ **Methods:**

- readBook(String filepath): Reads book data from a CSV file specified by the filepath. It reads each line, splits it into fields using ";" as a delimiter, creates a new Book object from the extracted data, and adds it to the books list using the addBook() method inherited from the CSV class.
- readStaff(String filepath): Reads staff data from a CSV file specified by the filepath. Similar to readBook(), it parses each line, creates a new Staff object, and adds it to the staffs list.
- readMember(String filepath): Reads member data from a CSV file specified by the filepath. It parses each line, creates a new Member object, and adds it to the members list.
- readLoanRecords(String fileName): Reads loan record data from a CSV file specified by the fileName. It stores each line of the file in an array and returns it. This method doesn't directly populate the loanRecords list; rather, it's used for retrieving loan records as strings.
- IDFound(String filePath): Reads the last ID from a CSV file specified by the filePath. It's used for checking the last ID used in staff or member records, presumably for generating unique IDs for new records.

➢ **Additional Notes:**

- All methods handle file I/O operations using BufferedReader to read the contents of CSV files line by line.
- The data from each line is split into fields using ";" as the delimiter.
- The readLoanRecords() method is designed to read loan records specifically and returns an array of strings representing each line of the loan record CSV file.
- The IDFound() method appears to be a utility method for extracting the last used ID from a staff or member CSV file, presumably to generate new unique IDs for future records.

❖ CSV_Writer

The CSV_Writer class is responsible for writing and updating data in CSV files. These methods handle file operations such as reading, writing, updating, and removing entries from CSV files, as well as managing the corresponding data structures in memory.

➢ **Methods:**

- writeBook: Appends a new row to the book CSV file with the provided book details.
- writeStaff: Appends a new row to the staff CSV file with the provided staff details.
- writeMember: Appends a new row to the member CSV file with the provided member details.
- updateStaff: Updates the staff information in both the CSV file and the staff list.
- updateMember: Updates the member information in both the CSV file and the members list.
- writeOrUpdateLoanRecords (overloaded): Writes or updates loan records in the loan CSV file.
- writeLoanRecord: Writes a new loan record to the loan CSV file.
- removeEntryFromCSV: Removes an entry from the specified CSV file based on the given value.
- remove_CSV_loan: Removes a loan entry from the loan CSV file based on the specified member name, surname, and book name.

➢ **Additional Notes:**

- File Operations: The class uses various file operations such as reading from and writing to CSV files. It utilizes buffered readers and writers for efficient reading and writing of large files.
- Error Handling: Error handling is implemented using try-catch blocks to handle any potential IOExceptions that may occur during file operations. Proper error messages are printed to the console to aid in debugging.
- Data Validation: While the class performs operations on CSV files, it assumes that the input data is properly formatted. Further validation could be added to ensure that the data conforms to expected formats and constraints.
- Concurrency: The class doesn't currently address issues related to concurrent access to the CSV files. If multiple threads or processes attempt to read from or write to the same file simultaneously, synchronization mechanisms may be needed to prevent data corruption.
- Performance Considerations: Depending on the size of the CSV files and the frequency of operations, performance considerations such as buffering, file locking, and batch processing could be further optimized for improved efficiency.
- Data Integrity: Care should be taken to ensure data integrity, especially when performing updates or deletions. It's essential to maintain consistency between the CSV files and any in-memory data structures to avoid discrepancies.
- Documentation: Adding comprehensive comments and documentation to the class and its methods would improve code readability and maintainability, making it easier for other developers to understand and extend the functionality.

❖ Book Class

The Book class is a representation of a book entity within a library management system.

➢ **Attributes:**

- title: Represents the title of the book.
- author: Represents the author of the book.
- ISBN: Represents the International Standard Book Number (ISBN) of the book.
- publishYear: Represents the year the book was published.

- totalQuantity: Represents the total quantity of copies available for this book.
- img: Represents the image associated with the book.

➢ **Constructors:**

- Book(String title, String author, String ISBN, String publishYear, int totalQuantity, String img): Initializes a book object with all its attributes.
- Book(String title): Initializes a book object with only the title provided. This is useful for creating book objects with minimal information.

➢ **Methods (Setters/Getters):**

- getTitle(), getAuthor(), getISBN(), getPublishYear(), getTotalQuantity(), getIMG(): Retrieve the values of the corresponding attributes.
- setTitle(String title), setAuthor(String author), setISBN(String ISBN), setPublishYear(String publishYear), setTotalQuantity(int totalQuantity), setIMG(String img): Set the values of the corresponding attributes.
- decreaseTotalQuantity(int amount): Decreases the total quantity of the book by the specified amount.
- increaseTotalQuantity(int amount): Increases the total quantity of the book by the specified amount.

➢ **Additional Notes:**

- The totalQuantity attribute is initialized with a default value of 1 if not specified explicitly.
- Methods like decreaseTotalQuantity and increaseTotalQuantity ensure that the total quantity is adjusted properly, handling cases where the specified amount is non-negative and ensuring that the total quantity doesn't go below zero.

❖ Loan Class

The Loan class represents a loan record within a library management system. the Loan class encapsulates the information related to a loan record and provides methods for retrieving and updating this information, as well as calculating fines for late returns.

➢ **Attributes:**

- book: Represents the book being loaned.
- member: Represents the member who is borrowing the book.

- dueDate: Represents the due date for returning the book.
- fine: Represents the fine incurred for late return of the book.
- dateString: Represents the due date formatted as a string.

➢ **Constructor:**

- Loan(String name, String surname, String phoneNumber, String email, String username,String title, String author, String ISBN, int totalQuantity, Date dueDate, double fine): Initializes a loan object with the necessary information about the member, book, due date, and fine. It also formats the due date as a string.

➢ **Methods (Setters/Getters):**

- getBook(), getMember(), getDueDate(), getDate(), getFine(): Retrieve the values of the corresponding attributes.
- setBook(Book book), setMember(Member member), setDueDate(Date dueDate), setFine(double fine): Set the values of the corresponding attributes.

➢ **Additional Notes:**

- The calculateFine() method calculates the fine incurred for late return of the book based on a predefined fine rate per day. It checks the current date against the due date and calculates the difference in days. If the book is returned before or on the due date, no fine is incurred.

❖ Person Class

The Person class forms the foundation for representing individuals within the library management system. It encapsulates essential information and behaviors applicable across various roles such as staff members and library patrons.

➢ **Attributes:**

- ID: This attribute stores a unique identification number assigned to each person within the system. It serves as a primary means of distinguishing individuals.
- Name: Represents the given name of the person.
- Surname: Denotes the family name or surname of the person.
- Phone Number: Stores the contact telephone number of the person, facilitating communication.

- Address: Represents the physical address where the person resides or can be reached.
- Email: Stores the email address of the person, enabling electronic communication.
- Username: Represents the unique username used for authentication purposes, often required for accessing the library management system.
- Password: Stores the password associated with the username, serving as a credential for secure access to the system.

➢ **Constructors:**

- Person(int ID, String name, String surname, String phoneNumber, String address, String email, String username, String password): This constructor initializes a Person object with comprehensive details, including ID, name, contact information, and authentication credentials.
- Person(int ID, String name, String surname, String phoneNumber, String email, String username, String password): Similar to the previous constructor, but excludes the address field, catering to scenarios where the address might not be available or necessary.
- Person(String name, String surname): Provides a simpler constructor for cases where only the name and surname of the person are known or required.

➢ **Methods (Setters/Getters):**

- The Person class includes a comprehensive set of getter and setter methods for accessing and modifying the attribute values. These methods enable interaction with the Person object and ensure encapsulation and controlled access to its data.

➢ **Abstract Method:**

- displayInfo(): This abstract method serves as a blueprint for displaying information about a person. Since the exact format and content of the information display may vary depending on the context or role of the person (e.g., staff member, library patron), it is left abstract for implementation by subclasses.

❖ Admin Class

The Admin class represents administrators within the library management system, who typically have elevated privileges and responsibilities compared to regular staff members.

- **Attributes:**
  - Admins List: This attribute stores a list of all admin objects, allowing for easy management and access to administrative accounts within the system.
  - Salary: Represents the salary of the admin, reflecting their compensation for their administrative duties and responsibilities.
  - Inherited Attributes: Inherits attributes such as ID, name, surname, phone number, address, email, username, and password from the Person class. These attributes provide basic identification and contact information for the admin.
- **Constructors:**
  - Admin(int ID, String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary): Initializes an Admin object with detailed information, including ID, contact details, authentication credentials, and salary.
  - Admin(): Default constructor that initializes an Admin object with default or empty values. This constructor is useful for creating placeholder admin objects or initializing the admin list.
- **Methods (Setters/Getters):**
  - getSalary() / setSalary(double salary): Allows retrieving and updating the salary of the admin. Salary information is crucial for payroll management and financial planning.
  - getAdmins(): Returns the list of all admin objects stored in the system. This method facilitates accessing and managing admin accounts.
  - Inherited Methods: Inherits getter and setter methods for the inherited attributes from the Person class, enabling access to and modification of basic information about the admin.
- **Additional Methods:**
  - displayInfo(): An inherited abstract method from the Person class, serving as a blueprint for displaying information about the admin. Since the format and content of the information display may vary, this method is left abstract for implementation by subclasses.

- getAdminInfo(): Returns formatted information about a specific admin, including their ID, name, surname, phone number, and email. This method is useful for displaying admin details in a readable format.
- getAllAdminsInfo(ArrayList<Admin> admins): Constructs a string containing information about all admins in the system. It iterates through the list of admins, retrieves their information using the getAdminInfo() method, and appends it to the string.
- getAllAdminsInfo_GUI(ArrayList<Admin> admins): Similar to getAllAdminsInfo(), but formats the admin information in a way suitable for a graphical user interface (GUI). It constructs a string with admin details, including ID, name, surname, phone number, and email, formatted for GUI display.
- getAllAdminsInformation(): Constructs a string containing information about all admins in the system, formatted for display. It iterates through the list of admins, retrieves their information using the getAdminInfo() method, and appends it to the string.
- searchAdminByUsername(String username): Searches for an admin with the specified username in the admin list. If found, it returns the admin object; otherwise, it returns null. This method facilitates admin authentication and access control.
- updateAdminInformation(String username, String newName, String newSurname, String newPhoneNumber, String newEmail, String newAddress, String newPassword): Updates the information of an admin identified by their username. It modifies the admin's details such as name, surname, contact information, and password.
- updateAdminInformation_CSV(String username, String newName, String newSurname, String newPhoneNumber, String newEmail, String newAddress, String newPassword): Updates the information of an admin stored in a CSV file. It reads the CSV file, locates the admin with the specified username, updates their details, and rewrites the CSV file with the updated information.
- readFromCSV(): Reads admin information from a CSV file and populates the admin list with admin objects. It reads each line of the CSV file, extracts admin details, and creates admin objects using the retrieved information.

- authenticate(String username, String password): Verifies the credentials (username and password) provided by an admin for authentication. It checks whether the provided credentials match those stored for the admin in the system.

❖ Staff Class

The Staff class represents personnel who work within the library, handling various administrative tasks and assisting patrons.

➢ **Attributes:**
  - Salary: This attribute stores the salary of the staff member, reflecting their compensation for their role and responsibilities within the library.
  - Inherited Attributes: The Staff class inherits attributes such as ID, name, surname, phone number, address, email, username, and password from the Person class. These attributes provide basic identification and contact information for the staff member.

➢ **Constructors:**
  - Staff(int ID, String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary): Initializes a Staff object with detailed information, including ID, contact details, authentication credentials, and salary.
  - Staff(int ID, String name, String surname, String phoneNumber, String email, String username, String password, double salary): A constructor variant that excludes the address field, suitable for cases where the address is not required or available.
  - Staff(): Provides a default constructor that initializes a Staff object with default or empty values. This constructor can be useful for initializing placeholder staff objects.

➢ **Methods (Setters/Getters):**
  - getSalary() / setSalary(double salary): These methods allow retrieving and updating the salary of the staff member. Salary information is essential for payroll management and financial planning.
  - Inherited Methods: The Staff class inherits getter and setter methods for the inherited attributes from the Person class. These methods enable accessing and

modifying the basic information of the staff member, such as their name, contact details, and authentication credentials.

- **Additional Methods:**
  - calculateSalary(int morningHours, int noonHours, int nightHours, double bonusAmount, double deductionAmount): This method calculates the total salary of the staff member based on the number of hours worked during different shifts (morning, noon, night), along with any bonuses or deductions. It provides a comprehensive overview of the staff member's compensation.
  - displayInfo(): An abstract method inherited from the Person class, which serves as a blueprint for displaying information about the staff member. Since the exact format and content of the information display may vary depending on the context or requirements, it is left abstract for implementation by subclasses.
- **Authentication Method:**
  - authenticate(String username, String password): This method allows verifying the credentials (username and password) provided by a staff member for authentication purposes. It checks whether the provided credentials match those stored for the staff member in the system.

❖ Member Class

The Member class represents library members who can borrow books from the library.

- **Attributes:**
  - Loans: An ArrayList storing loan objects associated with the member. Each loan represents a book borrowed by the member.
  - Inherited Attributes: Inherits attributes such as ID, name, surname, phone number, email, username, and password from the Person class. These attributes provide basic identification and contact information for the member.
- **Constructors:**
  - Member(int ID, String name, String surname, String phoneNumber, String email, String username, String password): Initializes a Member object with detailed information, including ID, contact details, and authentication credentials.

- Member(String name, String surname): Initializes a Member object with only a name and surname. This constructor provides flexibility for creating member objects with minimal information.
- Member(): Default constructor that initializes a Member object with default or empty values. This constructor is useful for creating placeholder member objects or initializing the loans list.

➢ **Methods (Setters/Getters):**

- getLoans(): Returns the list of loan objects associated with the member. This method facilitates accessing and managing the books borrowed by the member.
- displayInfo(): Displays information about the member, including their name, surname, contact details, and borrowed books. If the member has no loans, it indicates that no books are borrowed. This method provides a concise overview of the member's details and borrowed books.
- authenticate(String username, String password): Verifies the credentials (username and password) provided by a member for authentication. It checks whether the provided credentials match those stored for the member in the system.

➢ **Additional Methods:**

- memberMenu(): Represents a menu interface for members to interact with the library system. It guides members through various options such as searching for books, viewing available books, managing loans, updating information, and accessing admin information. This method allows members to perform common tasks within the library system conveniently.

❖ Library Class

The book management section of the Library class handles various operations related to books, such as adding, removing, searching, and displaying book information.

➢ **Attributes:**

- bookCSV, staffCSV, memberCSV, loanCSV: File paths for CSV files containing information about books, staff, members, and loans respectively.
- ID_Staff, ID_Member: Static variables tracking the current ID for staff and members, used for assigning unique IDs to new staff and members.
- csv_read, csv_write: Instances of the CSV_Reader and CSV_Writer classes respectively, used for reading from and writing to CSV files.
- scanner: A Scanner object for user input.

➢ **Constructor:**

- Library(): Initializes the library by reading data from CSV files, updating current IDs, and performing necessary setup operations.

➢ **Methods:**

- readoperations(): Reads data from CSV files for books, members, staff, and loans. Converts loan data to loan objects and adds them to the system.
- getCurrentID_Updatebook(): Updates the current ID for staff and members based on existing records. Decreases the total quantity of books borrowed by members.
- addBook(): Allows staff to add a new book to the library by entering details such as title, author, ISBN, publish year, and image reference. Writes the book information to the CSV file and adds the book to the system.
- addBook_GUI(String title, String author, String ISBN, String publishYear, String img): Allows adding a new book to the library through a graphical user interface (GUI). Validates input parameters and adds the book to the system if valid.
- removeBook(): Allows staff to remove a book from the library by entering the ISBN of the book. Removes the book from the list of books and updates the CSV file accordingly.
- removeBookByISBN_GUI(String ISBN): Allows removing a book from the library through a GUI by providing the ISBN of the book. Removes the book from the system if found.
- searchBook(): Allows users to search for a book by entering its title, author, or ISBN. Displays details of the found book if available.
- searchBook(String searchKey): Searches for a book based on the provided search key (title, author, or ISBN) and returns the first matching book object if found.

- searchBook_GUI(String searchKey): Allows searching for a book through a GUI by providing the search key. Returns true if a matching book is found, false otherwise.
- viewFoundBook(String query): Displays details of the found book based on the provided search query (title, author, or ISBN).
- printBooks(): Displays details of all books in the library, including title, author, ISBN, publish year, total quantity, and image reference.
- printBooks_GUI(): Displays details of all books in the library through a GUI. Returns a string containing book information formatted for display.

The staff management section of the Library class handles various operations related to staff members, such as adding, removing, searching, and displaying staff information.

- **Attributes:**
  - staffCSV: File path for the CSV file containing information about staff members.
  - ID_Staff: Static variable tracking the current ID for staff members, used for assigning unique IDs to new staff.
  - csv_read, csv_write: Instances of the CSV_Reader and CSV_Writer classes respectively, used for reading from and writing to the staff CSV file.
  - scanner: A Scanner object for user input.
- **Methods:**
  - addStaff(): Allows adding a new staff member to the library by entering details such as name, surname, phone number, email, address, username, password, and salary. Writes the staff information to the CSV file and adds the staff member to the system.
  - addStaff(String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary): Allows adding a new staff member to the library using provided parameters. Similar to the previous method but accepts parameters directly.
  - addStaff_GUI(String name, String surname, String phoneNumber, String address, String email, String username, String password, double salary): Allows

adding a new staff member through a graphical user interface (GUI). Validates input parameters and adds the staff member to the system if valid.

- removeStaff(): Allows removing a staff member from the library by entering the username of the staff member. Removes the staff member from the list of staff members and updates the CSV file accordingly.
- removeStaffByUsername_GUI(String username): Allows removing a staff member from the library through a GUI by providing the username. Removes the staff member from the system if found.
- searchStaff(): Allows searching for a staff member by entering their name or surname. Displays details of the found staff member if available.
- searchStaffByUsername(String searchKey): Searches for a staff member based on the provided username and returns the matching staff object if found.
- searchStaffByUsername_GUI(String searchKey): Allows searching for a staff member through a GUI by providing the username. Returns true if a matching staff member is found, false otherwise.
- viewFoundStaff(String searchKey): Displays details of the found staff member based on the provided search key (username).
- printStaff(): Displays details of all staff members in the library, including their ID, name, surname, phone number, address, email, username, password, and salary.
- printStaff_GUI(): Displays details of all staff members in the library through a GUI. Returns a string containing staff information formatted for display.
- addToArrayStaff(String username, String newName, String newSurname, String newPhoneNumber, String newAddress, String newEmail, String newUsername, String newPassword): Updates the details of a staff member in the system based on the provided parameters.
- StaffCalculateSalary(String username, int morningHours, int noonHours, int nightHours, double bonusAmount, double deductionAmount): Calculates and updates the salary of a staff member based on the working hours, bonus amount, and deduction amount provided.

The member management section of the Library class handles various operations related to library members, such as adding, removing, searching, and displaying member information.

- **Attributes:**
  - memberCSV: File path for the CSV file containing information about library members.
  - ID_Member: Static variable tracking the current ID for members, used for assigning unique IDs to new members.
  - csv_read, csv_write: Instances of the CSV_Reader and CSV_Writer classes respectively, used for reading from and writing to the member CSV file.
  - scanner: A Scanner object for user input.
- **Methods:**
  - addMember(): Allows adding a new member to the library by entering details such as name, surname, phone number, email, username, and password. Writes the member information to the CSV file and adds the member to the system.
  - addMember(String name, String surname, String phoneNumber, String address, String email, String username, String password): Allows adding a new member to the library using provided parameters. Similar to the previous method but accepts parameters directly.
  - addMember_GUI(String name, String surname, String phoneNumber, String email, String username, String password): Allows adding a new member through a graphical user interface (GUI). Validates input parameters and adds the member to the system if valid.
  - removeMember(): Allows removing a member from the library by entering the username of the member. Removes the member from the list of members and updates the CSV file accordingly.
  - removeMemberByUsername_GUI(String username): Allows removing a member from the library through a GUI by providing the username. Removes the member from the system if found.
  - searchMember(): Allows searching for a member by entering their name or surname. Displays details of the found member if available.

- searchMemberByUsername(String searchKey): Searches for a member based on the provided username and returns the matching member object if found.
- searchMemberBy_NameandSurname(String name,String Surname): Searches for a member based on the provided name and surname and returns the matching member object if found.
- searchMemberByUsername_GUI(String searchKey): Allows searching for a member through a GUI by providing the username. Returns true if a matching member is found, false otherwise.
- viewFoundMember(String searchKey): Displays details of the found member based on the provided search key (username).
- printMembers(): Displays details of all members in the library, including their ID, name, surname, phone number, email, username, and password.
- printMembers_GUI(): Displays details of all members in the library through a GUI. Returns a string containing member information formatted for display.
- addToArrayMember(String username, String newName, String newSurname, String newPhoneNumber, String newEmail, String newUsername, String newPassword): Updates the details of a member in the system based on the provided parameters.

The loan management section of the Library class handles operations related to book loans, including viewing loan records, checking out books, checking in books, calculating fines, and processing book transactions. These methods provide comprehensive functionality for managing book loans, including viewing loan records, calculating fines, checking in and checking out books, and processing book transactions, both through textual input and graphical user interfaces.

- **Attributes:**
  - loanCSV: File path for the CSV file containing loan records.
  - csv_read, csv_write: Instances of the CSV_Reader and CSV_Writer classes respectively, used for reading from and writing to the loan CSV file.
  - scanner: A Scanner object for user input.
- **Methods:**

- viewLoans(): Displays loan records including member details, book details, due dates, and fines. Also provides options to calculate fines for overdue books and add new loans.
- viewMemberLoans(String username): Displays loan records for a specific member identified by their username.
- viewLoans_GUI(): Displays loan records through a graphical user interface (GUI). Returns a formatted string containing loan information.
- viewMemberLoans_GUI(String username): Displays loan records for a specific member through a GUI. Returns a formatted string containing loan information.
- deleteLoans(String username, String ISBN): Deletes a loan record based on the username of the member and the ISBN of the book. Updates book quantity and removes the loan record from the CSV file and the list of loans.
- fineLoanSearch(String username): Calculates and displays fines for a specific member's overdue books. Returns a string containing fine information.
- checkInBook(String username, String ISBN): Checks in a book for a specific member identified by their username and ISBN of the book. Returns the calculated fine for the checked-in book.
- CheckOutBook(String username, String ISBN): Checks out a book for a specific member identified by their username and ISBN of the book. Returns true if the checkout was successful, false otherwise.
- processBookTransaction(String username, boolean checkOut): Processes a book transaction, either checking out or checking in a book for a member. Prompts the user for the ISBN of the book and performs the corresponding action.
- toLocalDate(Object dueDateObj): Helper method to convert due date to a LocalDate object.

❖ Library Management System Class

➢ Sign-In Page: The Library Management System Sign-In Page serves as the gateway for users to access the library's digital services based on their roles: administrator, staff, or member. Designed with both functionality and aesthetics in mind, the sign-

in page provides an intuitive interface for users to authenticate their identities and gain access to the system's features.

- **Key Components:**
  - Background Image: The sign-in page features a visually appealing background image that sets the tone for the user experience, creating an inviting atmosphere.
  - Welcome Message: A prominent welcome message greets users at the top of the page, reinforcing the system's purpose and inviting users to proceed with signing in.
  - Input Fields: Clear and labeled input fields for username and password allow users to enter their credentials securely.
  - Sign-In Buttons: Distinct sign-in buttons for administrators, staff, and members provide users with a clear path to access their respective functionalities within the system.
  - Sign-Up Option: A sign-up button offers new users the opportunity to create accounts, facilitating user onboarding and account creation processes.
  - Contact Admin Button: For users experiencing sign-in or account-related issues, a dedicated button provides direct access to contact administrators for assistance, ensuring prompt resolution of user concerns.
  - Error Handling: The system includes error handling mechanisms to inform users of incorrect credential inputs, guiding them to correct any mistakes and facilitating successful sign-in attempts.
  - User Authentication: Behind the scenes, the sign-in page utilizes role-based authentication methods to verify user credentials and grant appropriate access privileges based on user roles.

- **Purpose:** The primary objective of the Library Management System Sign-In Page is to streamline the user authentication process and provide users with secure access to the system's features based on their roles. By offering a user-friendly interface and robust authentication mechanisms, the sign-in page enhances user experience and contributes to the efficient operation of the library management system.

- **Conclusion:** With its blend of functionality, usability, and visual appeal, the Library Management System Sign-In Page serves as the entry point for users to engage with the library's digital services, facilitating seamless access to resources and administrative functionalities tailored to each user's role.

➢ Admin Information and Sign-Up Page: The Admin Information and Sign-Up Page is a crucial component of the Library Management System, offering users the ability to access administrative information and register as new members. It combines functionality with user-friendly design elements to streamline the process of acquiring information and creating new accounts.

- **Admin Information Display:**
  - Upon accessing the page, users are greeted with an interface displaying relevant administrative information.
  - The page presents a clear and organized layout, ensuring easy comprehension and navigation.
  - An aesthetically pleasing background sets the tone for the user experience, enhancing visual appeal.
- **Key Features:**
  - Admin Information Section:
    - ♦ Users can access essential administrative details, such as contact information and other relevant data.
    - ♦ The information is presented in a structured format within a panel, enhancing readability and accessibility.
  - Sign-Up Section:
    - ♦ New users are provided with a sign-up form to create their accounts seamlessly.
    - ♦ The form includes fields for essential user information, such as name, surname, phone number, email, username, and password.
    - ♦ Clear labeling and intuitive design elements make the sign-up process straightforward and user-friendly.
- **Interactive Elements:**

- Buttons for navigating between different sections, such as returning to the main menu or accessing admin information, ensure smooth user interaction.
- Cursor changes upon hovering over interactive elements, providing visual feedback to users.

– **Error Handling:**
- Robust error handling mechanisms notify users of any issues during the sign-up process.
- Error messages guide users on how to rectify their inputs, enhancing user experience and reducing frustration.

– **Confirmation Feedback:**
- Upon successful sign-up, users receive confirmation feedback, informing them of the completion of the process.
- Additional instructions, such as notification of a confirmation email, ensure users are informed about the next steps.

– **User Experience Enhancement:**
- The Admin Information and Sign-Up Page prioritizes user experience, offering a seamless and intuitive interface.
- Clear instructions, visually appealing design elements, and efficient navigation options contribute to a positive user experience.

– **Conclusion:** The Admin Information and Sign-Up Page serves as a vital gateway for users to access administrative details and create new accounts within the Library Management System. By combining functionality with user-friendly design principles, the page enhances user experience and facilitates efficient interaction with the system.

➢ Change Information for Admin and Staff: The "Change Information" feature in the Library Management System allows administrators and staff members to update their personal details conveniently. This functionality ensures that user information remains accurate and up-to-date within the system.

– **Interface Design:**
- Upon accessing the feature, users are greeted with a visually appealing interface, making it easy to navigate and interact with the system.

- The interface includes clear labels and text fields for each piece of user information, ensuring clarity and ease of use.

– **Personalized Experience:**
  - Depending on whether the user is logged in as an admin or staff member, the interface adapts to display relevant information and options.
  - For administrators, the interface reflects admin-specific details and functionalities, while staff members see options tailored to their roles.

– **Account Information Display:**
  - Users can view their current account information, such as name, surname, phone number, email, address, username, and password, conveniently organized within the interface.
  - Text fields are pre-filled with existing information, allowing users to easily identify and edit the details they wish to change.

– **Updating Information:**
  - Users can make changes to their account information by modifying the text fields as needed.
  - Upon making changes, users have the option to save their updates by clicking the "Save Changes" button. The system verifies that all required fields are filled before proceeding with the update.

– **Error Handling:**
  - The system provides error messages if any required fields are left blank, prompting users to complete all necessary information before saving changes.
  - Additionally, if users attempt to change their username or password, the system logs them out for security purposes and prompts them to sign in again.

– **Navigation and Sign-Out:**
  - Users have the option to navigate back to the main menu or sign out of their account directly from the interface.
  - The "Back to Menu" button redirects users to the main menu, while the "Sign Out" button logs users out of their account and returns them to the sign-in/sign-up page.

- **Conclusion:** The "Change Information" feature enhances user experience within the Library Management System by providing a convenient and secure method for administrators and staff members to update their personal details. With its intuitive interface design and robust functionality, the feature ensures that user information remains accurate and accessible at all times.

➢ Change Information for Members: The "Change Information" feature for members in the Library Management System allows users to update their personal details conveniently.

- **Interface Design:**
  - Upon accessing the feature, members are greeted with a visually appealing interface, featuring a background image that enhances the user experience.
  - The interface includes clear labels and text fields for each piece of user information, ensuring clarity and ease of use.
- **Account Information Display:**
  - Members can view their current account information, such as name, surname, phone number, email, username, and password, conveniently organized within the interface.
  - Text fields are pre-filled with existing information, allowing members to easily identify and edit the details they wish to change.
- **Updating Information:**
  - Members can make changes to their account information by modifying the text fields as needed.
  - Upon making changes, members have the option to save their updates by clicking the "Save Changes" button. The system verifies that all required fields are filled before proceeding with the update.
- **Error Handling:**
  - The system provides error messages if any required fields are left blank, prompting members to complete all necessary information before saving changes.
- **Navigation and Sign-Out:**

- Members have the option to navigate back to the main menu or sign out of their account directly from the interface.
- The "Back to Menu" button redirects members to the main menu, while the "Sign Out" button logs them out of their account and returns them to the sign-in/sign-up page.

– **Conclusion:** The "Change Information" feature enhances user experience within the Library Management System by providing a convenient and user-friendly method for members to update their personal details. With its intuitive interface design and robust functionality, the feature ensures that member information remains accurate and accessible at all times.

➢ Admin/Staff Menu: The Admin/Staff Menu provides access to various management functions based on the user's role.

– **Interface Design:**
  - The menu features a welcoming message addressed to the user, providing a personalized touch.
  - A background image adds visual appeal to the interface, with different images displayed based on whether the user is logged in as an admin or staff member.

– **Menu Options:**
  - Book Management: Allows users to manage books within the library system. Clicking this button redirects users to the book management interface.
  - Staff Management: For admins only, this option enables the management of staff members. Clicking this button redirects admins to the staff management interface.
  - Member Management: For admins only, this option facilitates the management of library members. Clicking this button redirects admins to the member management interface.
  - Account Management: Allows users to manage their own account information. Clicking this button redirects users to the account management interface, where they can update their details.

- Sign Out: Logs the user out of the system and returns them to the sign-in/sign-up page.

– **Member Menu:**
  - The Member Menu is tailored specifically for library members, providing access to functionalities relevant to their needs. Here's an overview:

– **Interface Design:**
  - Similar to the Admin/Staff Menu, the Member Menu displays a personalized welcome message for the user.
  - A background image enhances the visual appeal of the interface, creating a cohesive and engaging user experience.

– **Menu Options:**
  - Books: Allows members to browse and manage books available in the library system.
  - Check In/Out Books: Provides functionality for members to check books in or out of the library.
  - Loan Information: Enables members to view information about their current loans and borrowing history.
  - Account Management: Allows members to update their account information, such as name, contact details, and password.
  - Sign Out: Logs the member out of the system and returns them to the sign-in/sign-up page.

– **Conclusion:** Both menus offer intuitive navigation and access to essential functionalities, tailored to the specific roles and needs of users within the library management system. The visually appealing interface, coupled with clear menu options, enhances user experience and promotes efficient interaction with the system.

➢ Book/Staff/Member Management: They provide interfaces for administrators, staff, and members to perform various tasks related to managing books, staff and members within the library.

– **bookManagement():** This method orchestrates the book management interface, offering a range of functionalities tailored to the user's role:

- For Admin/Staff:
  - ♦ Add Book: Enables the addition of new books to the library inventory.
  - ♦ Remove Book: Allows the removal of existing books from the library inventory.
  - ♦ Search Book: Facilitates searching for books based on different criteria.
  - ♦ View Books: Provides a comprehensive view of all books available in the library.
  - ♦ Back to Main Menu: Allows users to return to the main menu for further navigation.
  - ♦ Sign Out: Enables users to log out of their accounts securely.
- For Member:
  - ♦ Search Book: Allows members to search for specific books within the library.
  - ♦ View Books: Provides members with a list of available books they can browse.
  - ♦ Back to Main Menu: Allows members to return to the main menu for additional options.
  - ♦ Sign Out: Enables members to securely log out of their accounts.

– **staffManagement():** This method oversees the staff management interface, offering functionalities for administrators and staff members:

- For Admin/Staff:
  - ♦ Add Staff: Allows administrators to add new staff members to the system.
  - ♦ Remove Staff: Permits the removal of staff members from the system.
  - ♦ Search Staff: Facilitates searching for staff members based on various criteria.
  - ♦ View Staff: Provides a comprehensive view of all staff members registered in the system.
  - ♦ Salary Calculation: Enables administrators to calculate salaries for staff members.
  - ♦ Back to Main Menu: Allows users to return to the main menu for further navigation.

- Sign Out: Enables users to log out of their accounts securely.

- For Member:
  - Search Staff: Allows members to search for staff members within the system.
  - View Staff: Provides members with a list of staff members and their details.
  - Back to Main Menu: Allows members to return to the main menu for additional options.
  - Sign Out: Enables members to securely log out of their accounts.

– **memberManagement():** handles the interface for managing members in the library system, catering to both administrators and staff. It provides functionalities such as adding, removing, searching, and viewing member information.

- **Background Image:** Depending on whether the user is logged in as an admin or staff, the method sets a corresponding background image (Admin.JPG or Staff.JPG).
- **Welcome Label:** Displays a prominent "Member Management" title at the top of the interface.
- **Buttons:**
  - Add Member: Allows administrators and staff to add new members to the system.
  - Remove Member: Permits the removal of existing members from the system.
  - Search Member: Enables searching for specific members based on various criteria.
  - View Member: Provides a comprehensive view of all members registered in the system.
  - Loan Information: Redirects to the loan management interface for handling member loan information.
  - Back to Main Menu: Returns users to the main menu for further navigation.
  - Sign Out: Logs users out of their accounts securely.

- **Action Listeners:** Each button is equipped with an action listener to perform the corresponding functionality when clicked.

➢ Book/Staff/Member view:

– **viewBooks():** presents an interface for viewing available books in the library system.

- **Background Image:** Displays a background image (view.JPG) for the interface.
- **Welcome Label**: Presents a title "View Books" at the top of the interface.
- **Book Information Panel:** Displays book information retrieved from the library system. It includes details such as book titles, authors, and availability.
- **Buttons:**
  - ♦ Back to Book Management: Redirects users to the book management interface for further actions.
  - ♦ Sign Out: Logs users out of their accounts securely.

– **viewStaff():** allows administrators and staff to view information about staff members within the library system.

- **Clear Existing Content:** Removes any existing content from the content pane to prepare for the new interface.
- **Background Image:** Sets a background image for the interface, creating a visually appealing backdrop.
- **Welcome Label:** Displays a welcoming "View Staff" title at the top of the interface, providing clear context for the user.
- **Staff Information Panel:** Presents staff information retrieved from the library system, including details such as staff names, roles, and other relevant data.
- **Buttons:**
  - ♦ Back to Staff Management: Allows users to navigate back to the staff management interface for further actions.
  - ♦ Sign Out: Logs users out of their accounts securely.

- **viewMembers():** facilitates the viewing of member information within the library system.
  - **Clear Existing Content:** Clears any existing content from the content pane to ensure a clean slate.
  - **Background Image:** Sets a background image for the interface, maintaining consistency with the system's visual design.
  - **Welcome Label:** Presents a welcoming "View Members" title at the top of the interface, providing clear direction to the user.
  - **Member Information Panel:** Displays member information retrieved from the library system, including details such as member names, contact information, and other relevant data.
  - **Buttons:**
    - Back to Member Management: Allows users to return to the member management interface for further interactions.
    - Sign Out: Provides a secure option for users to log out of their accounts.
- **viewMembersLoan():** users can see their loan records, while the addBook() method allows administrators and staff to add new books to the library.
- **viewMemberLoans(String username):**
  - **Clear Existing Content:** Clears any existing content from the content pane to prepare for the new interface.
  - **Background Image:** Sets a background image for the interface, providing visual appeal.
  - **Welcome Label:** Displays a welcoming "Your Loan Records" title at the top of the interface to provide clear context for the user.
  - **Loan Information Panel:** Displays loan records retrieved from the library system for the specific member.
  - **Buttons:**
    - Back to Member Menu: Allows users to navigate back to the member menu for further actions.
    - Sign Out: Logs users out of their accounts securely.

- Add Book/Staff/Member: The addBook(), addStaff() and addMember() methods facilitate the addition of book, staff and members, respectively, to the library management system.
  - **addBook():**
    - **Clear Existing Content:** Removes any existing content from the content pane to ensure a clean slate.
    - **Background Image:** Sets a background image for the interface to maintain consistency with the system's visual design.
    - **Welcome Label:** Presents a welcoming "Add Book" title at the top of the interface to provide clear direction to the user.
    - **Add Panel:** Provides fields for users to input book information such as title, author, ISBN, publish year, and image reference.
    - **Save Button:** Allows users to add the book to the library by capturing the information entered in the fields.
    - **Buttons:**
      - Back to Book Management: Enables users to return to the book management interface for further interactions.
      - Sign Out: Provides a secure option for users to log out of their accounts.
  - **addStaff():**
    - **Clear Existing Content:** Clears any existing content from the content pane to prepare for the new interface.
    - **Background Image:** Sets a background image for the interface, maintaining consistency with the system's visual design.
    - **Welcome Label:** Displays a welcoming "Add Staff" title at the top of the interface to provide clear direction to the user.
    - **Add Panel:** Provides fields for entering staff information such as name, surname, phone number, email, address, username, password, and salary.
    - **Save Button:** Allows users to add the staff member to the system by capturing the information entered in the fields.
    - **Buttons:**
      - Back to Staff Management: Enables users to return to the staff management interface for further interactions.

- Sign Out: Provides a secure option for users to log out of their accounts.

- **addMember():**
  - **Clear Existing Content:** Removes any existing content from the content pane to ensure a clean slate.
  - **Background Image:** Sets a background image for the interface to maintain consistency with the system's visual design.
  - **Welcome Label:** Presents a welcoming "Add Member" title at the top of the interface to provide clear direction to the user.
  - **Add Panel:** Provides fields for entering member information such as name, surname, phone number, email, username, and password.
  - **Save Button:** Allows users to add the member to the system by capturing the information entered in the fields.
  - **Buttons:**
    - Back to Member Management: Enables users to return to the member management interface for further interactions.
    - Sign Out: Provides a secure option for users to log out of their accounts.

- salaryCalculationStaff(): provides a user interface for staff salary calculation within the library management system.
  - **Clear Existing Content:** Clears any existing content from the content pane to prepare for the new interface.
  - **Background Image**: Sets a background image for the interface, maintaining consistency with the system's visual design.
  - **Welcome Label:** Displays a welcoming "Staff Salary Calculation" title at the top of the interface to provide clear direction to the user.
  - **Calculation Panel:** Provides fields for entering staff salary calculation details, including staff username, morning, noon, and night hours, bonus amount, and deduction amount.
  - **Save Button:** Allows users to save the salary calculation by capturing the information entered in the fields and invoking the StaffCalculateSalary() method from the library.
  - **Buttons:**

- Back to Admin Menu: Enables users to return to the admin menu for further interactions.
- Sign Out: Provides a secure option for users to log out of their accounts.

➢ remove Book/Staff/Member: provide functionalities to remove books, staff and members, respectively, from the library management system.

– **removeBook():**

- **Interface Setup:**
  - ♦ Clears the existing content pane to prepare for the new interface.
  - ♦ Sets a background image and adds a welcome label for the book removal section.
  - ♦ Creates a light transparent background for the instruction and input fields.
- **Input Field:**
  - ♦ Displays a label instructing the user to enter the book's ISBN to be removed.
  - ♦ Provides a text field for users to input the ISBN of the book they wish to remove.
- **Remove Button:**
  - ♦ Allows users to initiate the removal process.
  - ♦ Upon clicking the button, the system attempts to remove the book using the provided ISBN.
  - ♦ Displays a success or error message based on the outcome of the removal operation.
- **Navigation Buttons:**
  - ♦ Includes buttons to navigate back to the book management section or sign out of the system.

– **removeStaff():**

- **Interface Setup:** sets up the interface for staff removal.
- **Input Field:**
  - ♦ Displays a label instructing the user to enter the staff member's username to be removed.

- ♦ Provides a text field for users to input the username of the staff member they wish to remove.

- **Remove Button:**
  - ♦ Allows users to initiate the removal process.
  - ♦ Upon clicking the button, the system attempts to remove the staff member using the provided username.
  - ♦ Displays a success or error message based on the outcome of the removal operation.
- **Navigation Buttons:**
  - ♦ Includes buttons to navigate back to the staff management section or sign out of the system.

– **removeMember():**

- **Interface Setup:**
  - ♦ Clears the existing content pane to prepare for the new interface.
  - ♦ Sets a background image and adds a welcome label for the member removal section.
  - ♦ Creates a light transparent background for the instruction and input fields.
- **Input Field:**
  - ♦ Displays a label instructing the user to enter the member's username to be removed.
  - ♦ Provides a text field for users to input the username of the member they wish to remove.
- **Remove Button:**
  - ♦ Allows users to initiate the removal process.
  - ♦ Upon clicking the button, the system attempts to remove the member using the provided username.
  - ♦ Displays a success or error message based on the outcome of the removal operation.
- **Navigation Buttons:**
  - ♦ Includes buttons to navigate back to the member management section or sign out of the system.

- search Book/Staff/Member: provide functionalities to seach books, staff and members, respectively, from the library management system.
  - **searchBook():**
    - **Interface Setup:**
      - Clears the existing content pane to prepare for the new interface.
      - Sets a background image and adds a welcome label for the book search section.
      - Creates a light transparent background for the instruction and input fields.
    - **Input Field:**
      - Displays a label instructing the user to enter the book title, author, or ISBN to search for.
      - Provides a text field for users to input their search query.
    - **Search Button:**
      - Allows users to initiate the search operation.
      - Upon clicking the button, the system searches for the book based on the provided query.
      - Displays a success message if the book is found, along with its information.
      - If the book is not found, it displays an error message.
    - **Navigation Buttons:**
      - Includes buttons to navigate back to the book management section or sign out of the system.
  - **searchStaff():**
    - **Interface Setup:**
      - Clears the existing content pane and sets up a new background.
      - Adds a welcome label for the staff search section.
      - Creates a light transparent background for the search panel.
    - **Input Field:**
      - Provides a text field for users to input the staff username they want to search for.
    - **Search Button:**

- Allows users to initiate the search operation.
- Displays a success message if the staff is found, along with their information.
- If the staff is not found, it displays an error message.

- **Navigation Buttons:**
  - Includes buttons to navigate back to the staff management section or sign out of the system.

– **searchMember():**

- **Interface Setup:**
  - Clears the existing content pane and sets up a new background.
  - Adds a welcome label for the member search section.
  - Creates a light transparent background for the search panel.
- **Input Field:**
  - Provides a text field for users to input the member username they want to search for.
- **Search Button:**
  - Allows users to initiate the search operation.
  - Displays a success message if the member is found, along with their information.
  - If the member is not found, it displays an error message.
- **Navigation Buttons:**
  - Includes buttons to navigate back to the member management section or sign out of the system.

➢ Check in/out Book and Loan Management: provide functionalities to Check in books, Check out Books and Managing the members Loan with Admin/Staff authorization respectively, from the library management system.

– **checkInOutBooks() :**

- **Interface Setup:**
  - Clears the existing content pane and sets up a new background.
  - Adds a welcome label for the check-in/out books section.

- Creates a light transparent background for the check-in/out panel.

- **Input Fields:**
  - Provides radio buttons for users to select whether they want to check in or check out a book.
  - Includes a text field for users to input the ISBN of the book.
- **Done Button:**
  - Initiates the check-in/out operation based on user selection.
  - Displays appropriate success or error messages.
- **Navigation Buttons:**
  - Includes buttons to navigate back to the book management section or sign out of the system.

– **loanManagement() Method:**

- **Interface Setup:**
  - Clears the existing content pane and sets up a new background.
  - Adds a welcome label for the loan management section.
  - Creates a light transparent background for the loan records panel.
- **Loan Information Display:**
  - Retrieves loan information for the member and displays it in a text area.
- **Fine Calculation:**
  - Provides radio buttons for users to select whether they want to calculate fines for overdue books.
  - Includes a text field for users to input the username for fine calculation.
  - Initiates the fine calculation operation and displays the result.
- **Navigation Buttons:**
  - Includes buttons to navigate back to the member management section or sign out of the system.

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

- **Conclusion**

In this report, we introduced a comprehensive Library Management System (LMS) designed to address the evolving needs of libraries in the digital era. The LMS offers a range of functionalities aimed at streamlining library operations, enhancing user experience, and optimizing resource utilization.

Through the exploration of the system's design principles, features, and implementation, we have highlighted its significance in modernizing library services while staying true to the core mission of providing access to knowledge and fostering a love for reading and learning. The LMS caters to administrators, staff, and members, providing a seamless platform for book management and user interactions.

- **Key Findings**

Our examination of the LMS revealed several key findings:

1. The system facilitates efficient book management, including adding, removing, updating, and searching for book information, thus enabling librarians to maintain an organized collection.
2. Member management functionalities allow for the effective tracking of membership details, borrowing history, and membership statuses, enhancing user experience and administrative efficiency.
3. Automation of loan tracking processes, including due date management and fine calculation, ensures smooth transactions and promotes accountability among library members.
4. Implementation of secure authentication mechanisms ensures the integrity and confidentiality of library data, protecting sensitive information from unauthorized access.

- **Future Works**

While the current iteration of the LMS represents a significant advancement in library management technology, there are several avenues for future research and development:

1. **Enhanced User Interfaces**: Continuously improving the user interfaces of the LMS to enhance usability and accessibility for both administrators and library patrons.

2. **Integration with External Systems**: Exploring opportunities to integrate the LMS with external systems, such as online cataloging platforms and digital lending services, to expand its functionality and reach.
3. **Advanced Analytics**: Incorporating data analytics capabilities into the LMS to provide insights into library usage patterns, popular genres, and borrowing trends, aiding in collection development and resource allocation.
4. **Mobile Applications**: Developing mobile applications for the LMS to enable on-the-go access to library services, catering to the needs of an increasingly mobile user base.
5. **Collaborative Features**: Implementing collaborative features, such as discussion forums and recommendation systems, to foster community engagement and knowledge sharing among library patrons.
6. **Reflective Practices**: Incorporating reflective practices into the project lifecycle to ensure continuous improvement. This includes careful team selection to optimize collaboration and learning from iterative cycles of analysis, design, and implementation. By adopting a more structured approach to project management, we can mitigate the risk of scope creep and enhance the overall quality of the system.

- **Contributions**

The development of the LMS represents a significant contribution to the field of library science and information technology. By leveraging modern technologies and design principles, the system offers a robust solution for modernizing library operations and enhancing user experiences.

- **Limitations**

Despite its potential, the development of the LMS faced some constraints:

- Time and resource limitations may have restricted the breadth of features.
- Testing and feedback were primarily limited to a specific user group.
- Integration with existing library infrastructures might pose challenges.

While acknowledging these constraints, the LMS remains poised to transform library services and knowledge access.