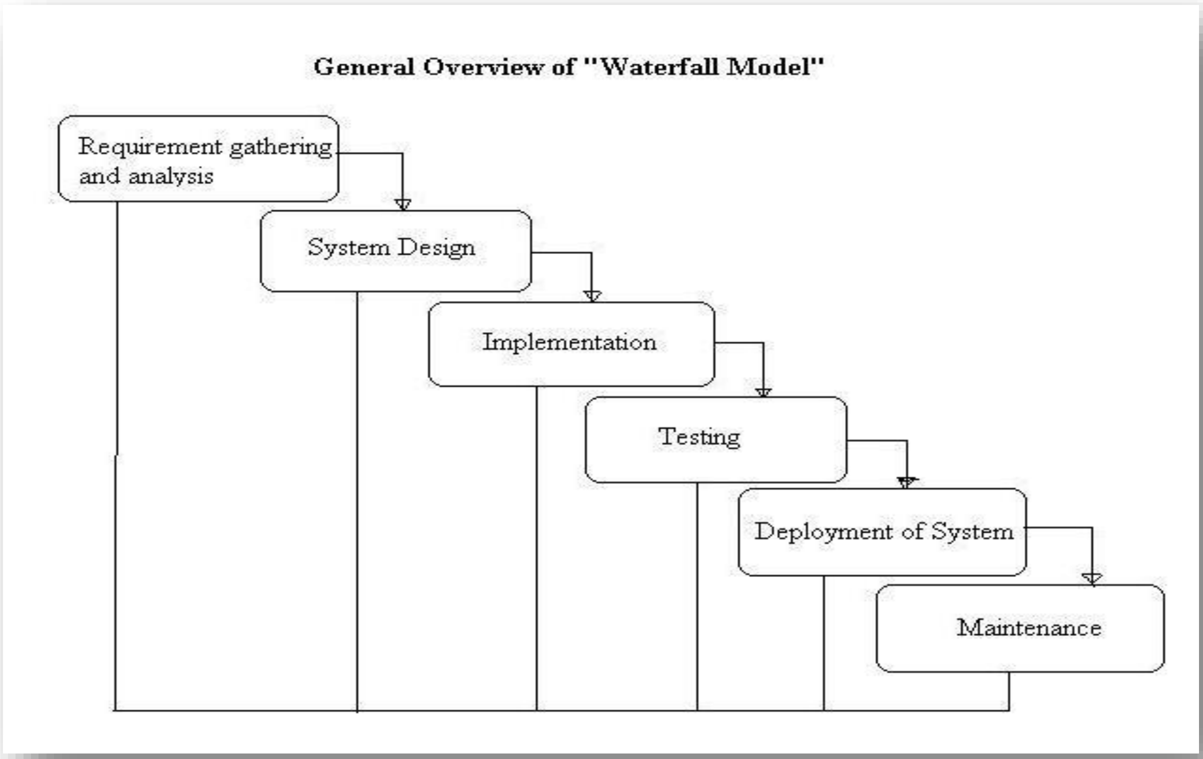
Practical no: 01

Aim: Application of at least two traditional process models.

1) Waterfall model:

- The classical waterfall model is the basic software development life cycle model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. However, it is very important because all the other software development life cycle models are based on the classical waterfall model.
- The waterfall model is useful in situations where the project requirements are well-defined and the project goals are clear. It is often used for large-scale projects with long timelines, where there is little room for error and the project stakeholders need to have a high level of confidence in the outcome.
- **Features of the SDLC Waterfall Model:**
    1. **Sequential Approach:** The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
    2. **Document-Driven:** The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
    3. **Quality Control:** The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.
    4. **Rigorous Planning:** The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

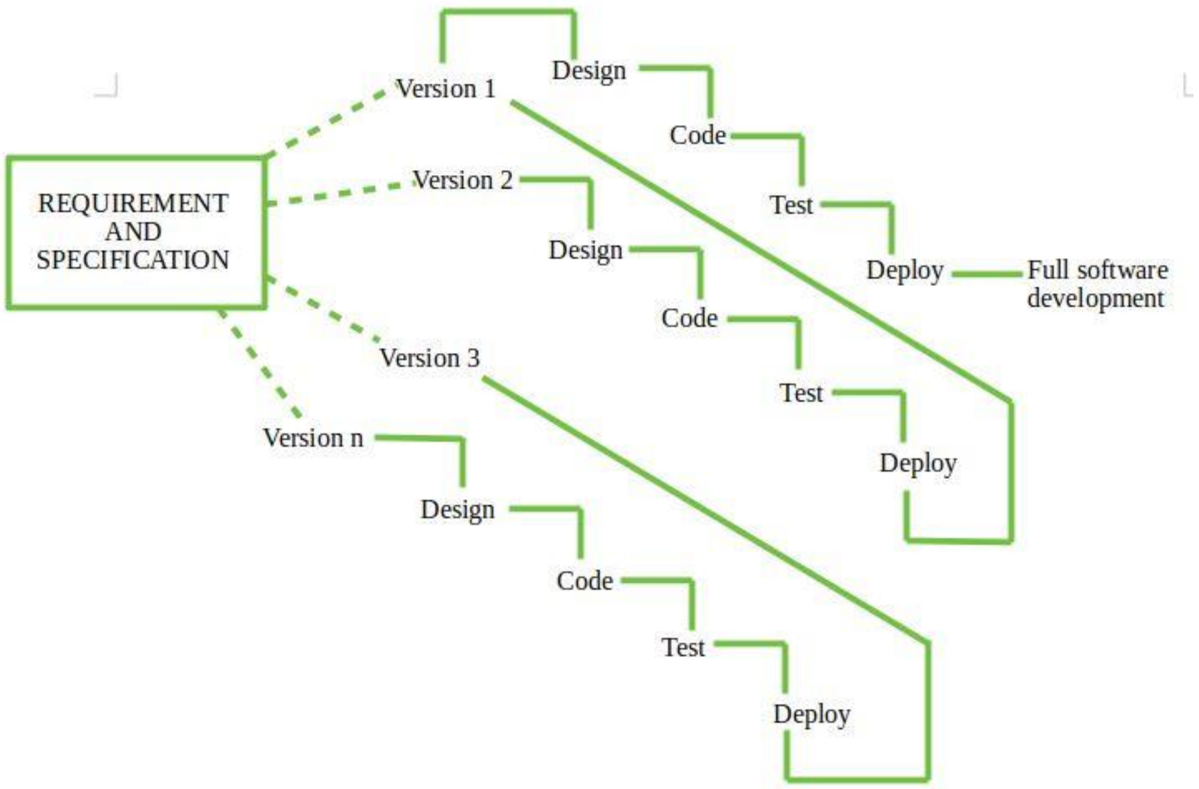

General Overview of "Waterfall Model"
Requirement gathering and analysis
System Design
Implementation
Testing
Deployment of System
Maintenance

- The Waterfall Model has six phases which are:
    1. **Requirements:** The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.
    2. **Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.
    3. **Development:** The Development phase include implementation involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.
    4. **Testing:** In the testing phase, the software is tested to ensure that it meets the requirements and is free from defects.
    5. **Deployment:** Once the software has been tested and approved, it is deployed to the production environment.
    6. **Maintenance:** The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

2) incremental model:

- A simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.
- **When to use the Incremental Process Model**
    1. Funding Schedule, Risk, Program Complexity, or need for early realization of benefits.
    2. When Requirements are known up-front.
    3. When Projects have lengthy development schedules.
    4. Projects with new Technology.
        - Error Reduction (core modules are used by the customer from the beginning of the phase and then these are tested thoroughly).
        - Uses divide and conquer for a breakdown of tasks.
        - Lowers initial delivery cost.
        - Incremental Resource Deployment.
    5. Requires good planning and design.
    6. The total cost is not lower.
    7. Well-defined module interfaces are required.
- **Characteristics of Incremental Process Model**
    1. System development is divided into several smaller projects.
    2. To create a final complete system, partial systems are constructed one after the other.

3. Priority requirements are addressed first.
4. The requirements for that increment are frozen once they are created.



- Phases of incremental model:

1. **Requirement analysis:** In Requirement Analysis At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer.
2. **Design & Development:** At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer. The Development Team first undertakes to develop core features (these do not need services from other features) of the system. Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.
3. **Deployment and Testing:** After Requirements gathering and specification, requirements are then split into several different versions starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site. in development and Testing the product is checked and tested for the actual process of the model.
4. **Implementation:** In implementation After the last version (version n), it is now deployed at the client site.

Practical No 02

Aim: Application of Agile process models.

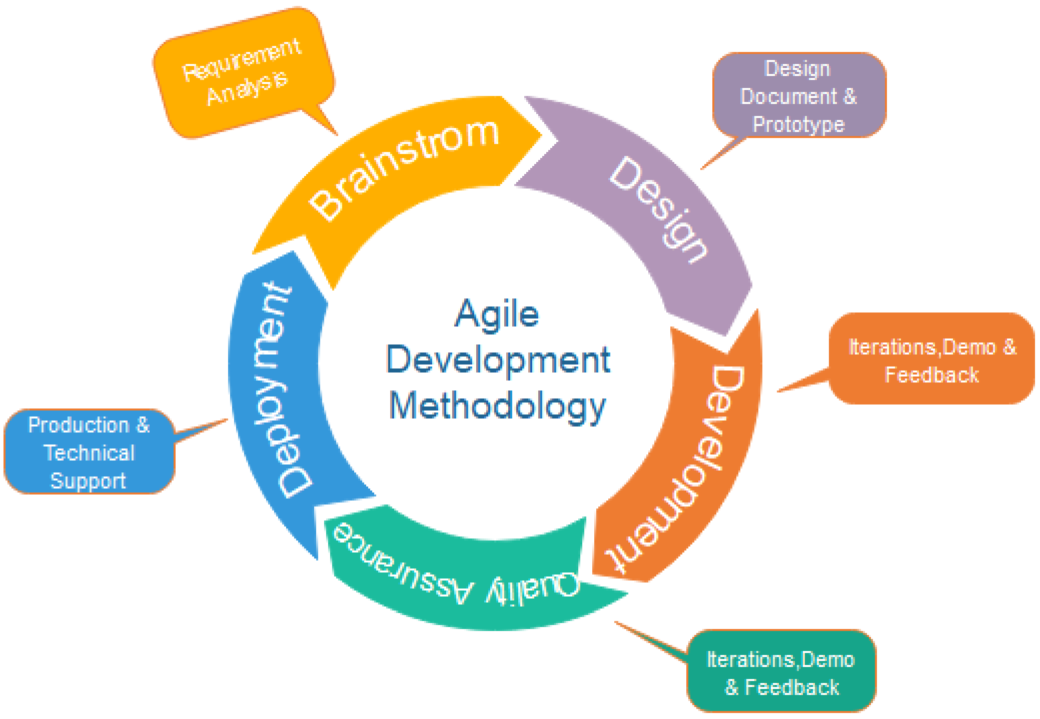

Requirement Analysis
Brainstrom
Design Document & Prototype
Design
Agile Development Methodology
Iterations,Demo & Feedback
Development
Production & Technical Support
Deployment
Quality Assurance
Iterations,Demo & Feedback

Fig. Agile Model

Agile process models are iterative and incremental approaches to software development that emphasize flexibility, customer collaboration, and rapid delivery of high-quality software. Here are some key aspects and applications of Agile process models:

Key Aspects of Agile Process Models

1. **Iterative Development**: Agile models focus on developing software in small, incremental releases or iterations. Each iteration includes planning, development, testing, and feedback.
2. **Customer Collaboration**: Agile promotes regular collaboration with customers and stakeholders to ensure the software meets their needs and expectations.
3. **Flexibility and Adaptability**: Agile methodologies embrace change, allowing teams to respond to new requirements or changes in project scope.
4. **Continuous Improvement**: Agile teams regularly reflect on their processes and practices to identify areas for improvement and implement changes in subsequent iterations.
5. **Cross-Functional Teams**: Agile teams are typically composed of cross-functional members who possess diverse skills, enabling them to handle various aspects of development collectively.

6. **Simplicity**: Agile promotes simplicity in design and implementation, focusing on delivering the most valuable features first.

Popular Agile Methodologies

1. **Scrum**:
    - **Roles**: Product Owner, Scrum Master, Development Team
    - **Artifacts**: Product Backlog, Sprint Backlog, Increment
    - **Events**: Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective
2. **Kanban**:
    - Visualizes work using a Kanban board
    - Limits work in progress (WIP) to improve flow
    - Emphasizes continuous delivery and improvement
3. **Extreme Programming (XP)**:
    - Practices include pair programming, test-driven development (TDD), continuous integration, and refactoring
    - Focuses on technical excellence and customer satisfaction
4. **Lean Software Development**:
    - Principles include eliminating waste, amplifying learning, and delivering as fast as possible
    - Emphasizes value creation and efficiency

Applications of Agile Process Models

1. **Software Development**: Agile is widely used in software development projects to deliver high-quality software that meets customer needs. It's suitable for projects with changing requirements and those requiring quick delivery.
2. **Project Management**: Agile methodologies can be applied to various project management scenarios, promoting transparency, collaboration, and adaptability.
3. **Product Development**: Agile can be used in product development to quickly iterate on prototypes, gather user feedback, and refine the product based on real-world usage.
4. **Marketing**: Agile marketing teams use iterative cycles to test and refine marketing strategies, ensuring they align with customer preferences and market trends.
5. **Education**: Agile practices can be applied in educational settings to develop curricula, create educational content, and manage classroom activities effectively.
6. **Healthcare**: Agile approaches can improve the development and implementation of healthcare solutions, ensuring they meet regulatory requirements and patient needs.

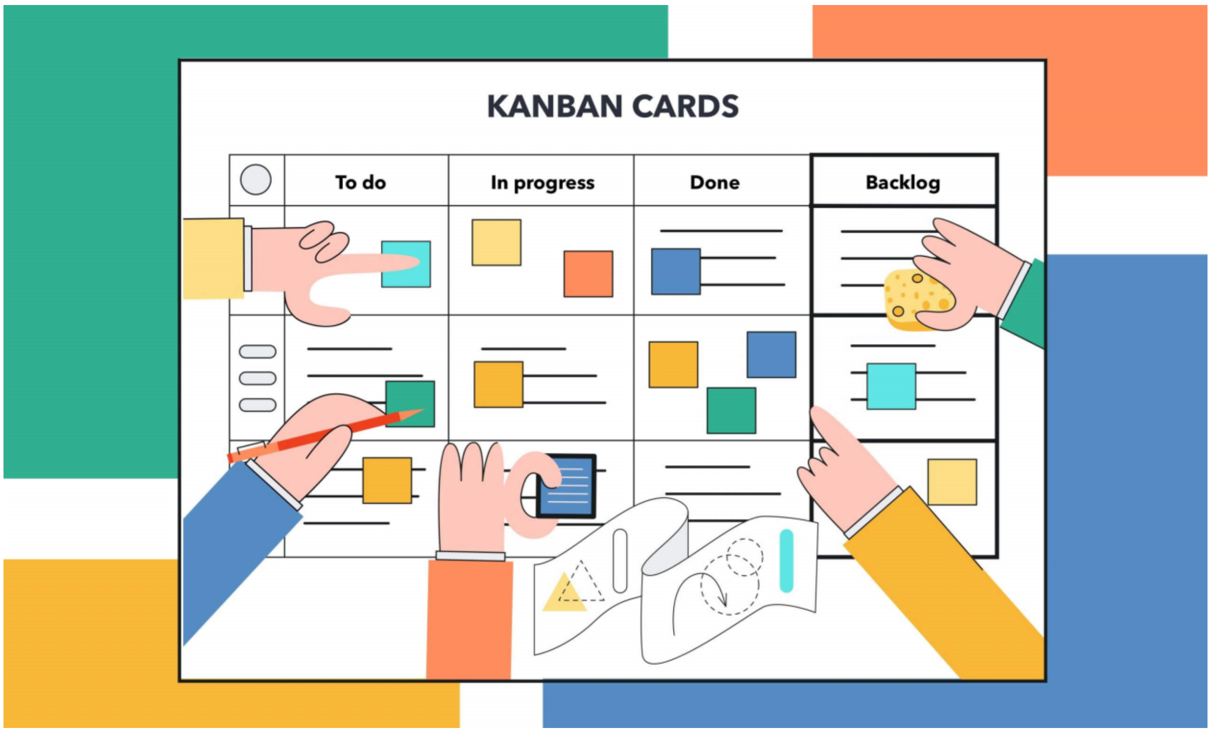
Implementing Agile in a Project

To implement Agile in a project, follow these steps:

1. **Adopt an Agile Methodology**: Choose a suitable Agile methodology (e.g., Scrum, Kanban, XP) based on the project requirements and team structure.
2. **Form an Agile Team**: Assemble a cross-functional team with the necessary skills and assign roles (e.g., Scrum Master, Product Owner).
3. **Define Project Vision and Goals**: Clearly articulate the project's vision, goals, and success criteria.
4. **Create a Product Backlog**: List and prioritize all the features, enhancements, and bug fixes required for the project.
5. **Plan Iterations**: Break the project into iterations (sprints) and plan the work to be completed in each iteration.
6. **Execute and Monitor**: Develop, test, and deliver increments of the software in each iteration. Conduct daily stand-up meetings to track progress and address any issues.
7. **Review and Reflect**: Hold sprint reviews to demonstrate the completed work and gather feedback. Conduct sprint retrospectives to identify improvements for the next iteration.
8. **Adapt and Improve**: Use feedback and retrospective insights to make continuous improvements to the process and product.

Kanban Board

Simple Kanban Board Setup



A simple Kanban board typically consists of three primary columns: "To Do," "In Progress," and "Done." Here's how you can set it up and use it effectively:

Components of a Simple Kanban Board

1. **Columns**:
   - **To Do**: Tasks that need to be started.
   - **In Progress**: Tasks that are currently being worked on.
   - **Done**: Tasks that have been completed.
2. **Cards**: Each card represents a task or work item and includes:
   - Task description
   - Assignee (optional)
   - Due date (optional)

Setting Up the Simple Kanban Board

1. **Define the Workflow**: Start with the basic three columns: "To Do," "In Progress," and "Done."
2. **Create Cards**: Add cards for each task in the "To Do" column. Ensure each card has a clear description of the task.
3. **Move Cards**: As work progresses, move cards from "To Do" to "In Progress," and finally to "Done" once the task is completed.

Example of a Simple Kanban Board

To Do	In Progress	Done
Task 1	Task 3	Task 2
Task 4		Task 5
Task 6		

Using the Simple Kanban Board

1. **Add Tasks**: At the beginning of the project or sprint, list all tasks in the "To Do" column.
2. **Start Work**: Team members pick tasks from the "To Do" column and move them to "In Progress" when they start working on them.
3. **Complete Tasks**: Once a task is finished, it is moved to the "Done" column.
4. **Review and Reflect**: Periodically review the board with the team to ensure tasks are progressing smoothly and discuss any blockers or issues.

Benefits of a Simple Kanban Board

1. **Easy to Use**: The straightforward structure makes it easy for any team to adopt and use effectively.
2. **Visual Clarity**: Provides a clear visual representation of task status and progress.
3. **Improved Focus**: Helps team members focus on one task at a time, reducing multitasking.
4. **Enhanced Collaboration**: Promotes transparency and better communication within the team.

Example Scenario: Simple Kanban Board for a Small Project

Project: Website Redesign

To Do	In Progress	Done
Create wireframes	Design homepage	Set up project repo
Write content for homepage		Create project plan
Set up hosting		

Software Requirements Specification

For

Library Management System

Prepared by

RUSHIKESH RAMDAS GARJE – 128
ARYAN SUNIL GUPTA – 130
SHREYASH SANTOSH GUPTA – 131

Guide Name :
MR. VILAS JADHAV

MGM'S COLLEGE OF ENGG. & TECH.
KAMOTHE , NAVI MUMBAI
UNIVERSITY OF MUMBAI.

2TH August, 2024

1. INTRODUCTION

With the increase in the number of readers, better management of libraries system is required. The Library management system focuses on improving the management of libraries in a city or town. "What If you can check whether a book is available in the library through your phone?" or "what if instead of having different library cards for different libraries you can just have one ?" or "you can reserve a book or issue a book from your phone sitting at your home!". The Integrated Library Management system provides you the ease of issuing, renewing, or reserving a book from an library within your town through your phone. The Integrated Library Management system is developed on the android platform which basically focuses on issuing, renewing and reserving a book.

1.1 PURPOSE

The purpose of the project is to maintain the details of books and library members of different libraries. The main purpose of this project is to maintain a easy circulation system between clients and the libraries, to issue books using single library card, also to search and reserve any book from different available libraries and to maintain details about the user (fine, address, phone number).Moreover, the user can check all these features from their home.

1.2 SCOPE

- ◉ Manually updating the library system into an android based application so that the user can know the details of the books available and maximum limit on borrowing from their computer and also through their phones.
- ◉ The ILM System provides information's like details of the books, insertion of new books, deletion of lost books, limitation on issuing books, fine on keeping a book more than one month from the issued date.
- ◉ Also user can provide feedback for adding some new books to the library.

1.3 Definition, Acronyms, Abbreviation:

- JAVA -> platform independence
- SQL -> Structured query Language
- DFD -> Data Flow Diagram
- CFD -> Context Flow Diagram
- ER -> Entity Relationship
- IDE -> Integrated Development Environment
- SRS -> Software Requirement Specification

2. OVERALL DESCRIPTION

2.1 PRODUCT PRESPECTIVE

The proposed Library Management System will take care of the current book detail at any point of time. The book issue, book return will update the current book details automatically so that user will get the update current book details.

2.2 SOFTWARE REQUIREMENT

- Front end:
    - Android developer tool
    - Advance java Back end:
    - MySQL

2.3 HARDWARE REQUIREMENT

- Android version 2.3 ginger bread(minimum, android user’s)
- 2GB ram
- 1.2 GHz processor
- Intel i5
- Windows 7/8/8.1/10

2.4.1 FUNCTIONAL REQUIREMENT

- **R.1:Register**
    - Description : First the user will have to register/sign up. There are two different type of users.
    - The library manager/head : The manager have to provide details about the name of library ,address, phone number, email id.
    - Regular person/student : The user have to provide details about his/her name of address, phone number, email id.

- **R.1.1: Sign up**
    - Input: Detail about the user as mentioned in the description.
    - Output: Confirmation of registration status and a membership number and password will be generated and mailed to the user.
    - Processing: All details will be checked and if any error are found then an error message is displayed else a membership number and password will be generated.

- **R.1.2 : Login**
    - Input: Enter the membership number and password provided.
    - Output : User will be able to use the features of software.

- **R.2 : Manage books by user.**

- **R.2.1 : Books issued.**
    - Description : List of books will be displaced along with data of return.

- **R.2.2 : Search**
    - Input : Enter the name of author's name of the books to be issued.
      Output : List of books related to the keyword.

- **R.2.3 : Issues book**
    - State : Searched the book user wants to issues.
    - Input : click the book user wants.
    - Output : conformation for book issue and apology for failure in issue.
    - Processing : if selected book is available then book will be issued else error will be displayed.

- **R.2.4 : Renew book**
    - State : Book is issued and is about to reach the date of return.
    - Input : Select the book to be renewed.
    - Output : conformation message.
    - Processing : If the issued book is already reserved by another user then error message will be send and if not then conformation message will be displayed.

- **R.2.5 : Return**
    - Input ; Return the book to the library.
    - Output : The issued list will be updated and the returned book will be listed out.

- **R.2.6 ; Reserve book**
    - Input ; Enter the details of the book.
    - Output : Book successfully reserved.
    - Description : If a book is issued by someone then the user can reserve it ,so that later the user can issue it.

- **R.2.6 Fine**
    - Input : check for the fines.
    - Output : Details about fines on different books issued by the user.
    - Processing : The fine will be calculated, if it crossed the date of return and the user did not renewed if then fine will be applied by Rs 10 per day.

- **R.3 Manage book by librarian**

- **R.3.1 Update details of books**

- **R.3.1.1 Add books**
    - Input : Enter the details of the books such as names ,author ,edition, quantity.
    - Output : confirmation of addition.

- **R.3.1.2 Remove books**
    - Input : Enter the name of the book and quantity of books.
    - Output : Update the list of the books available.

2.4.2 Non Functional Requirements

- **Usability Requirement**

The system shall allow the users to access the system from the phone using android application. The system uses a android application as an interface. Since all users are familiar with the general usage of mobile app, no special training is required. The system is user friendly which makes the system easy.

- **Availability Requirement**

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

- **Efficiency Requirement**

Mean Time to Repair (MTTR) **-** Even if the system fails, the system will be recovered back up within an hour or less.

- **Accuracy**

The system should accurately provide real time information taking into consideration various concurrency issues. The system shall provide 100% access reliability.

- **Performance Requirement**

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

- **Reliability Requirement**

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

2.5 USER CHARACTERSTICS

We have 3 levels of users :

- User module: In the user module, user will check the availability of the books.
  - Issue book
  - Reserve book
  - Return book
  - Fine details
- Library module:

- Add new book
- Remove books Update details of book

 Administration module:

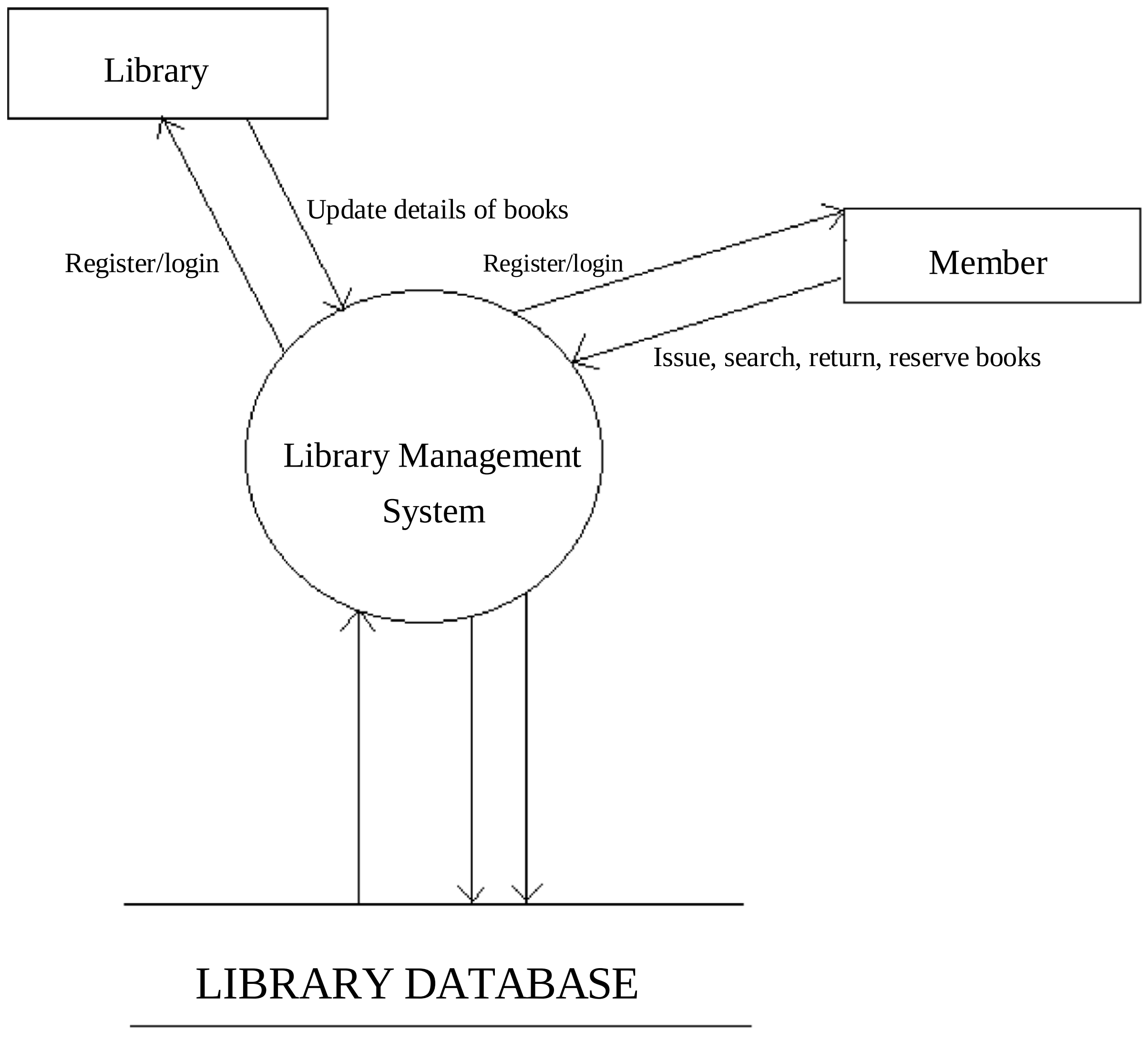
The following are the sub module in the administration module :

- Register user
- Entry book details
- Book issue

2.6 CONSTRAINTS

Any update regarding the book from the library is to be recorded to have update & correct values, and any fine on a member should be notified as soon as possible and should be correctly calculated.

2.7 FLOW DIAGRAM

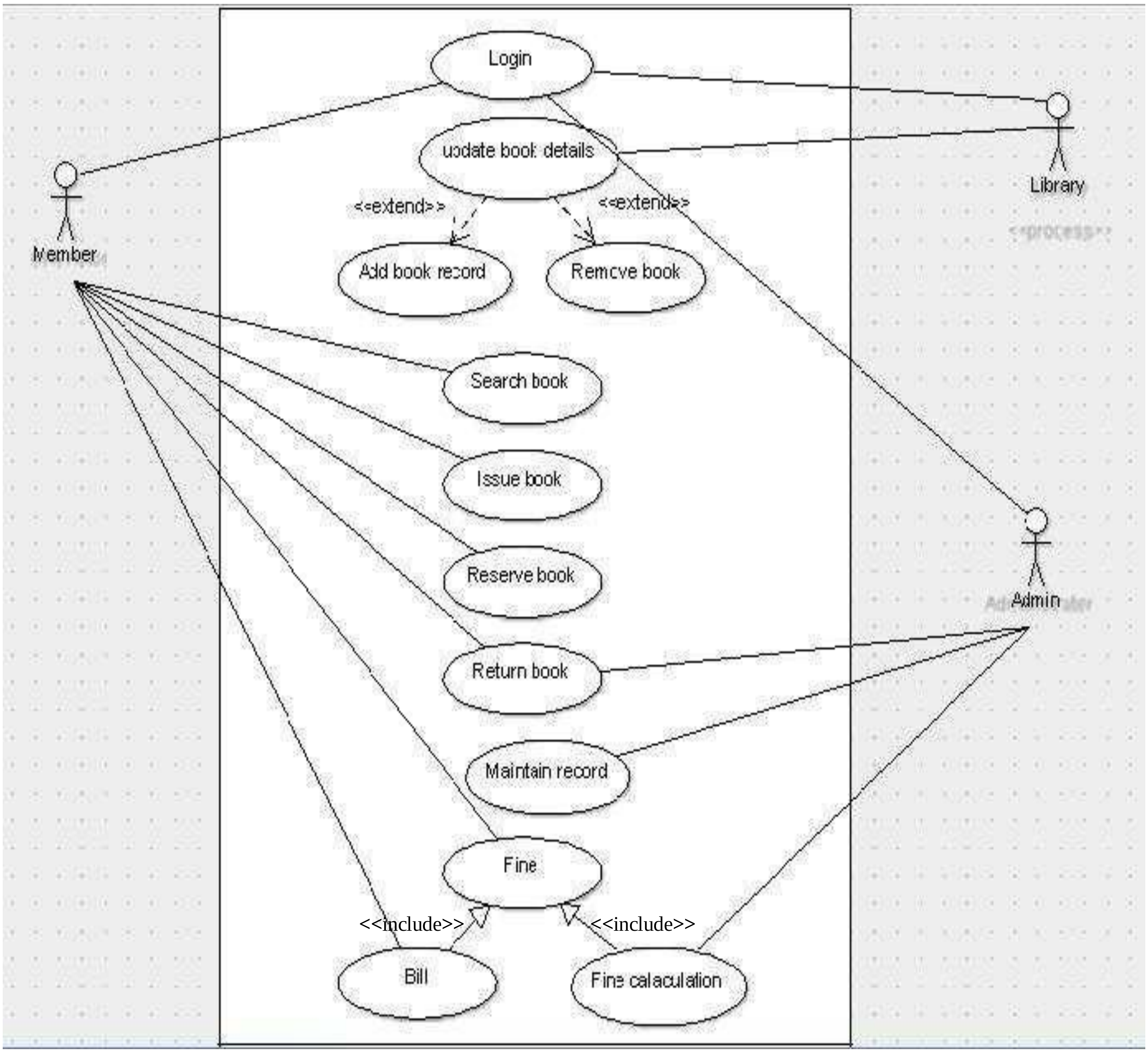


2.8 USE CASE MODEL DESCRIPTION:

Use Case selection	**Description**
Use Case name	Add student and library record
Level	Sub-Functional level
Primary actor	Student, Library
Stakeholders and interest	Student: wants to register into the system. Library: wants to register into the system and update book details. Administrator: responsible for the management of the transaction of fine and also login and register details.
Pre-condition	Students and Library have submitted their registration form.
Post-condition	Record for a student/library has been added.
Main success scenario	1 Student/Library opens the application to access the services of the LMS 2 Student/Library sign-up to get registered online. 3 He/She provides correct information and secret password. 4 He/She got registered.
Alternative flow	1 Student/Library opens the application in their phone 2 He/She tries to sign-up 3 He/She fails and receives an error 4 He/She will report an error and the error will be rectified as soon as possible.
Specific requirement	□ The response time for registration is 1 minute. □ The response time for login is 1 minute

Table 1: table for use case description.

2.9.1 USE CASE DIAGRAM


Login
update book details
<<extend>>
<<extend>>
Add book record
Remove book
Search book
Issue book
Reserve book
Return book
Maintain record
Fine
<<include>>
<<include>>
Bill
Fine calaculation
Member
Library
Admin

2.9.2 CLASS DIAGRAM

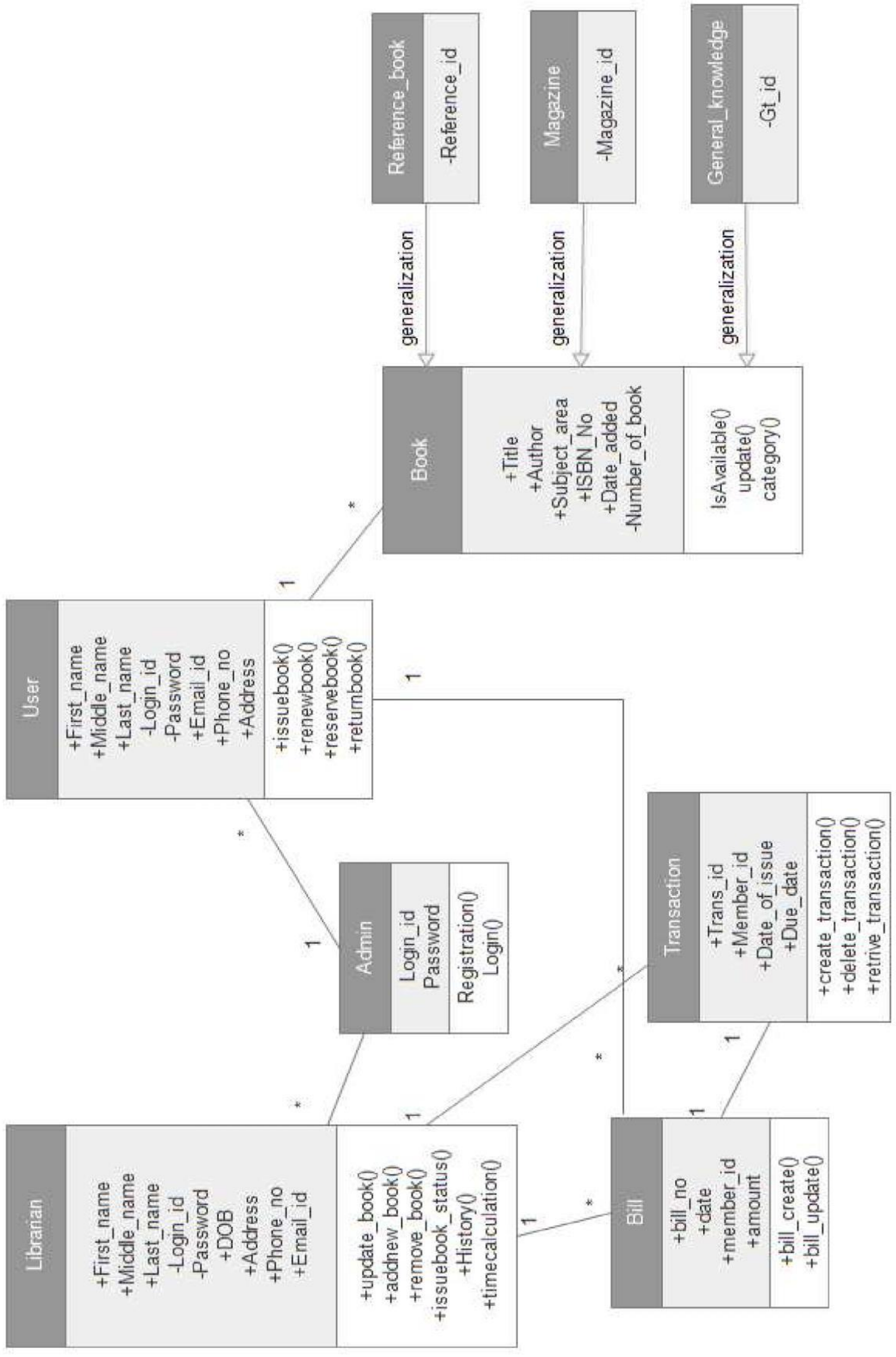


Fig 2. Class diagram

2.9.3 Sequential diagram

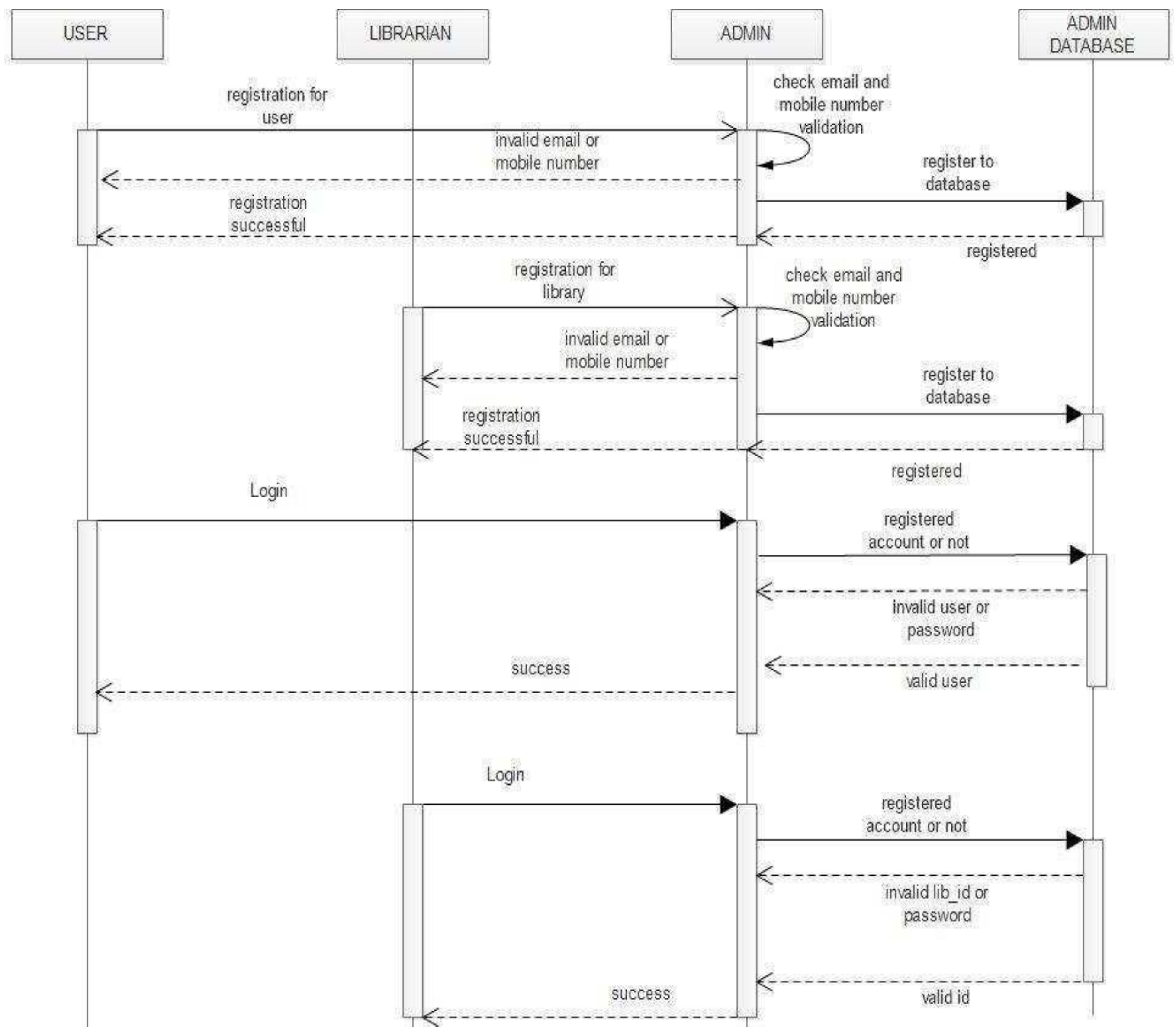


Fig 4(a). Register and login sequence diagram

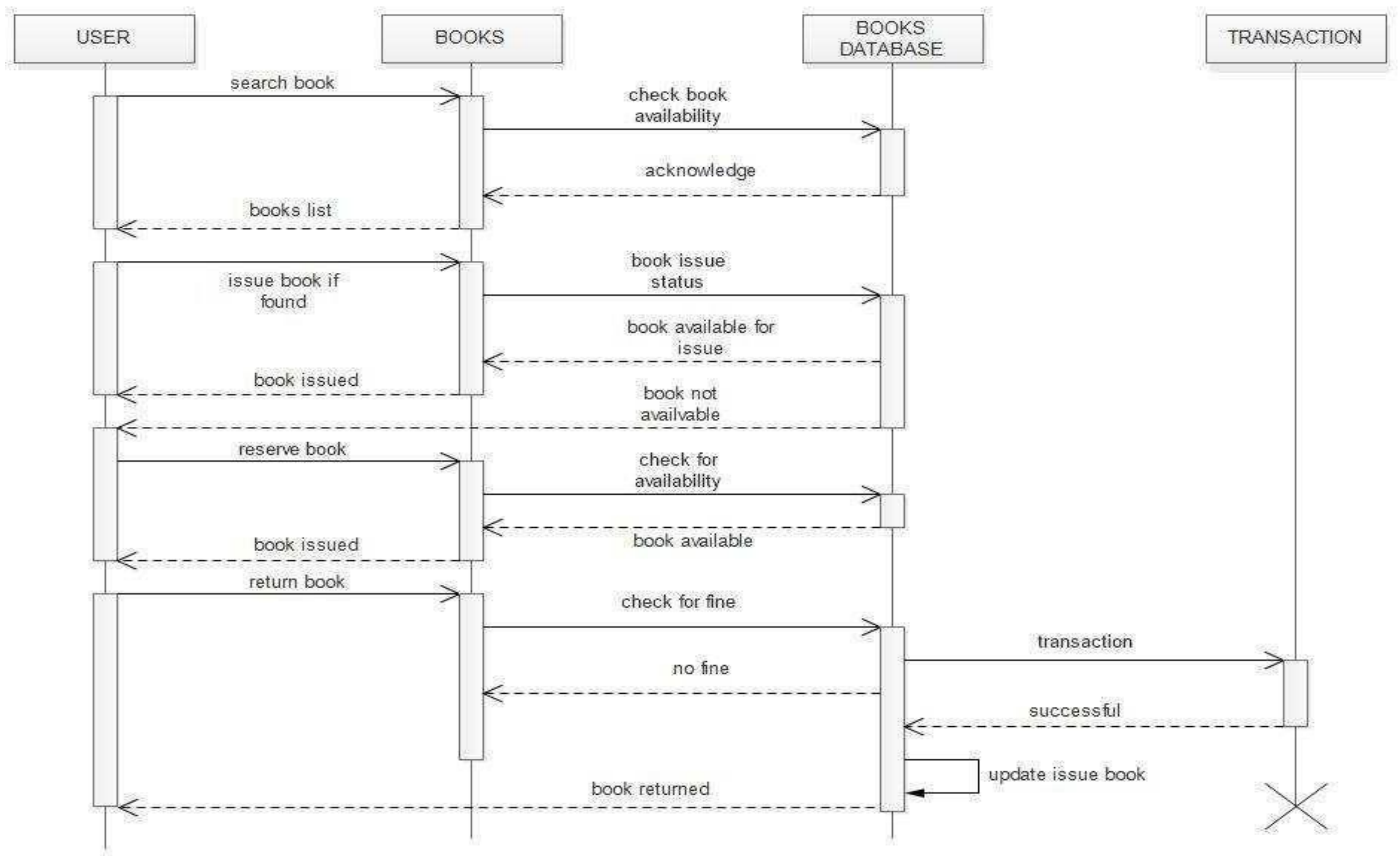


Fig 4(b) Services user could use

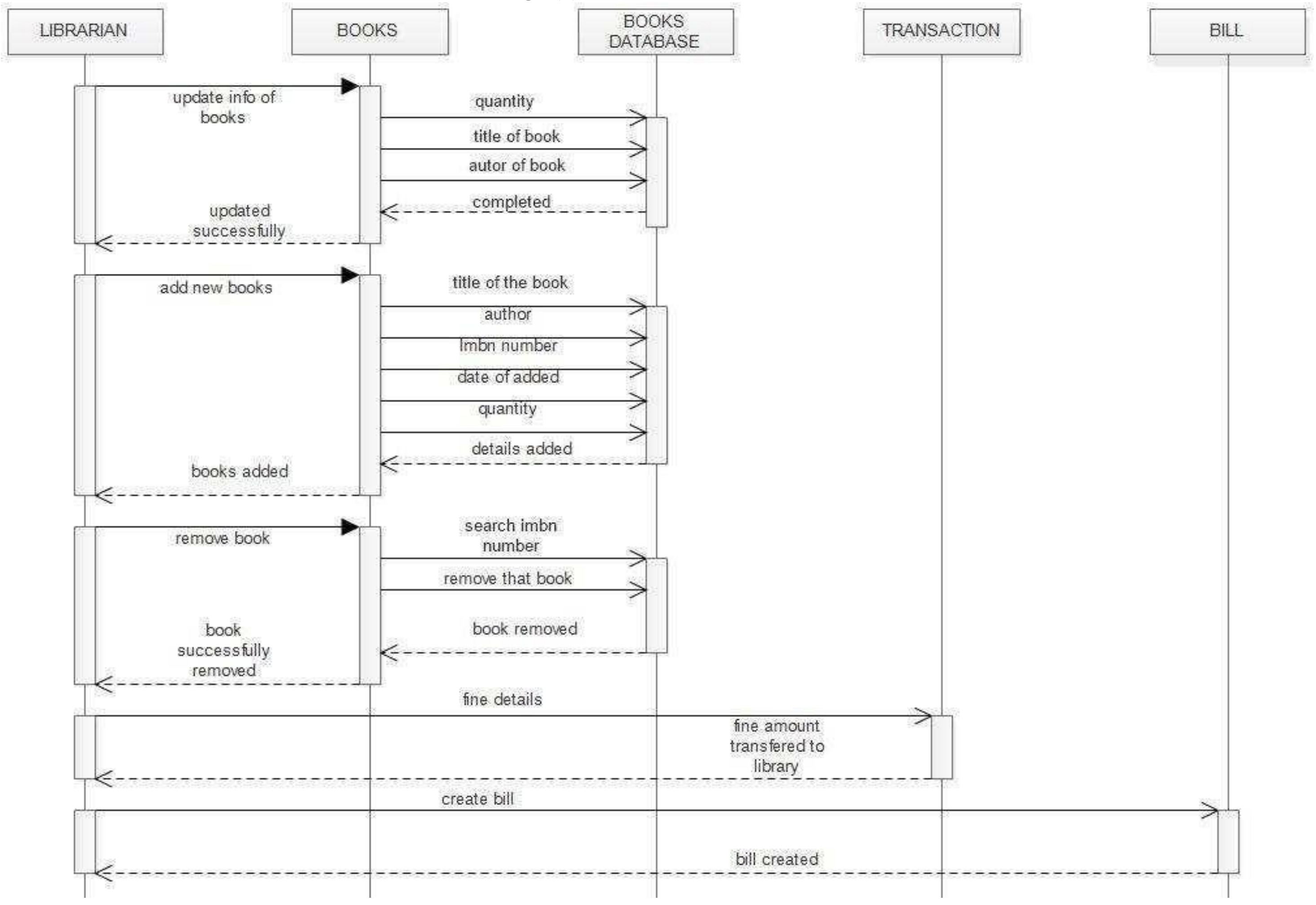


Fig 4(c) Services of a library

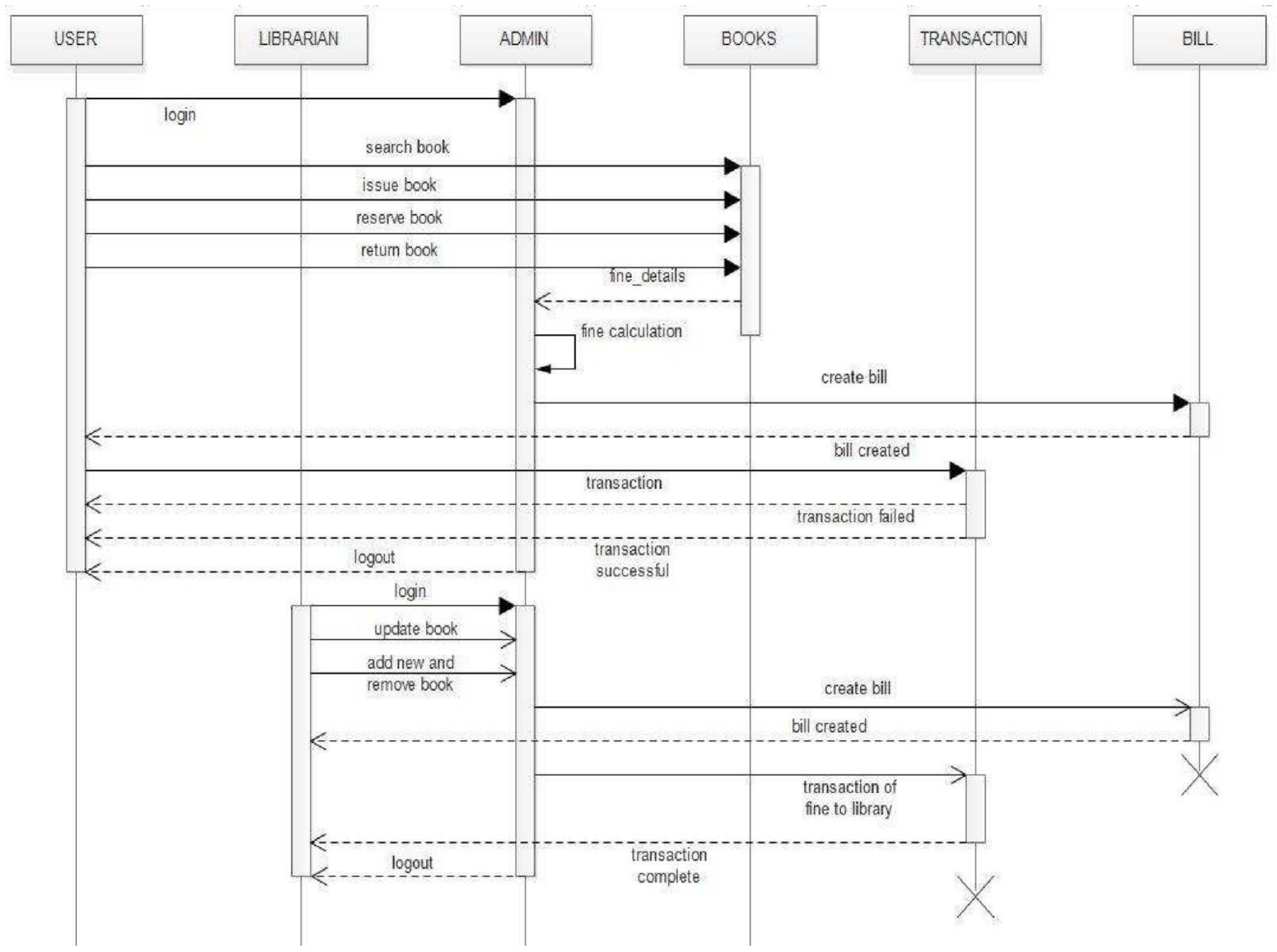
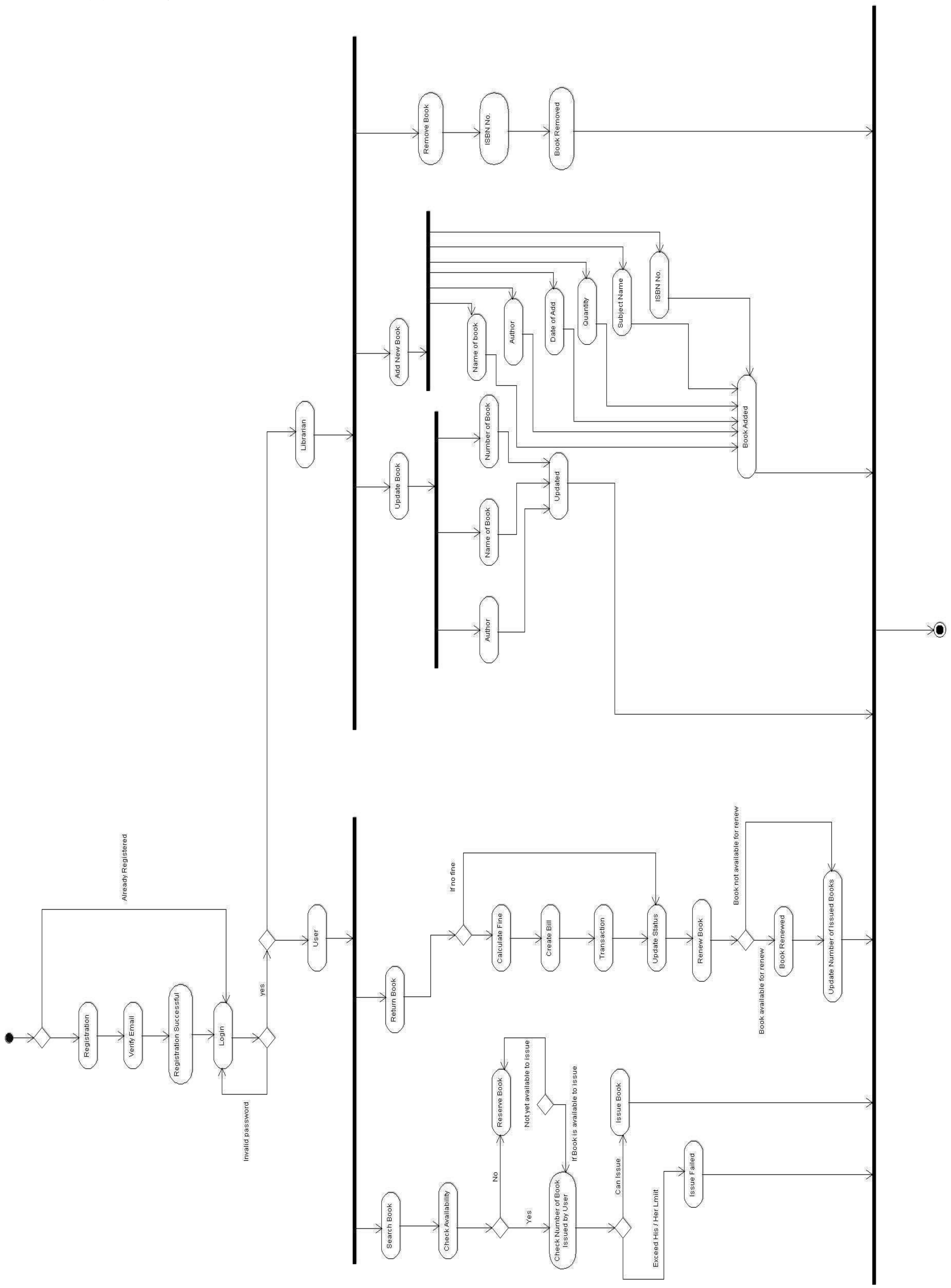
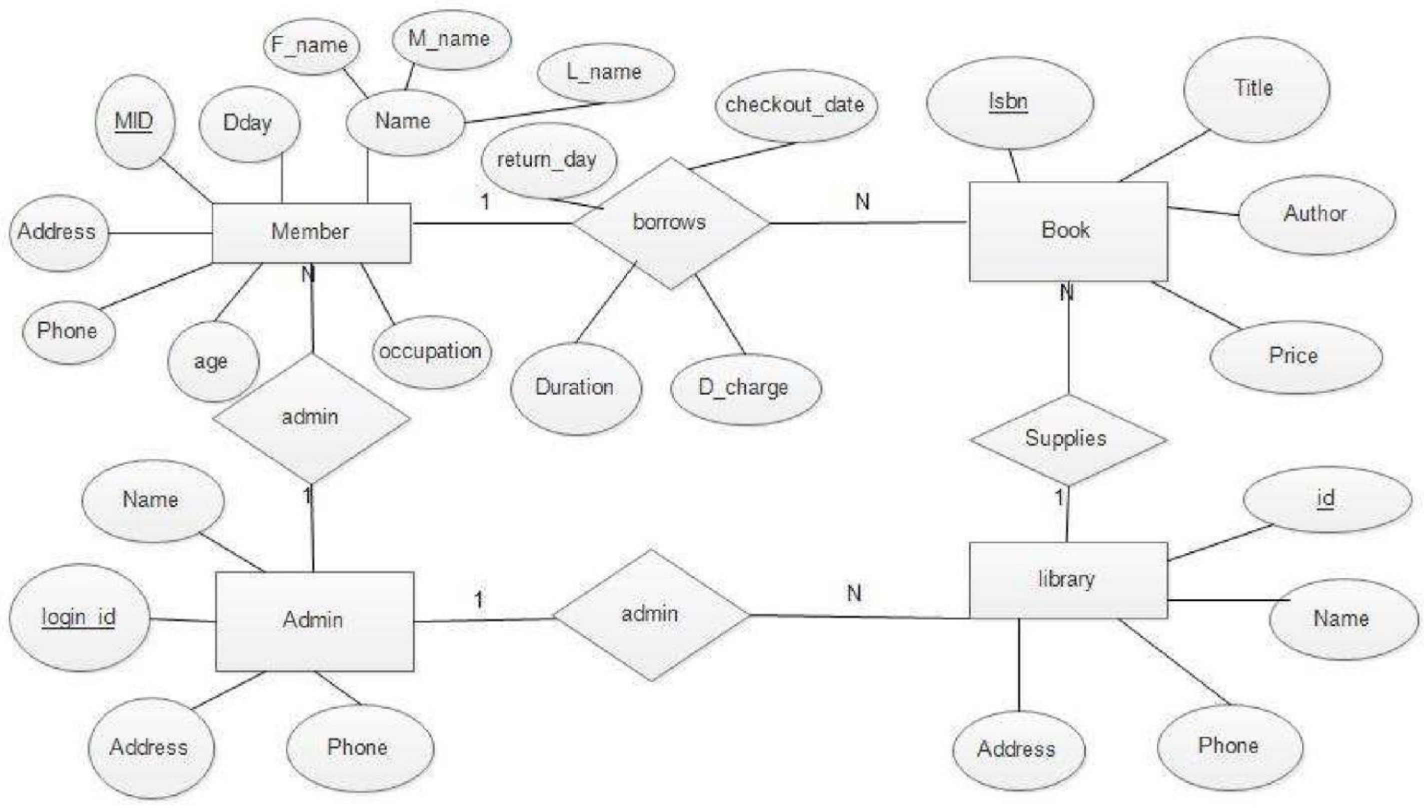


Fig 4(d) Full working of the system

2.9.4 ACTIVTY DIAGRAM



2.10 ER diagram



2.11 Assumptions and Dependencies

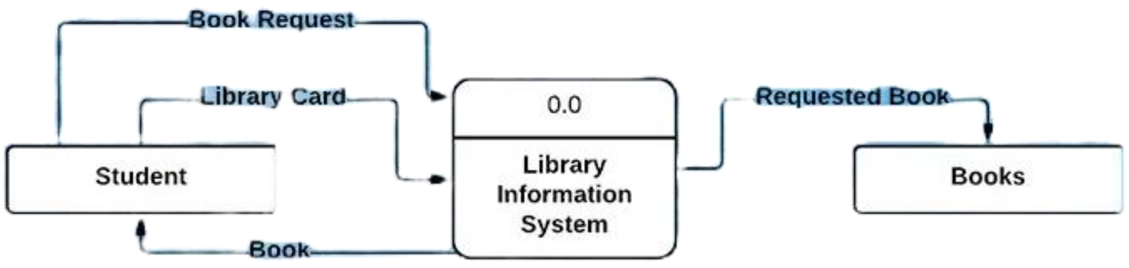
The product needs following third party applications for the development of the project:

- Android Studio (for development of android based applications)
- Netbeans
- Photoshop (for editing layouts, icons, buttons, etc)

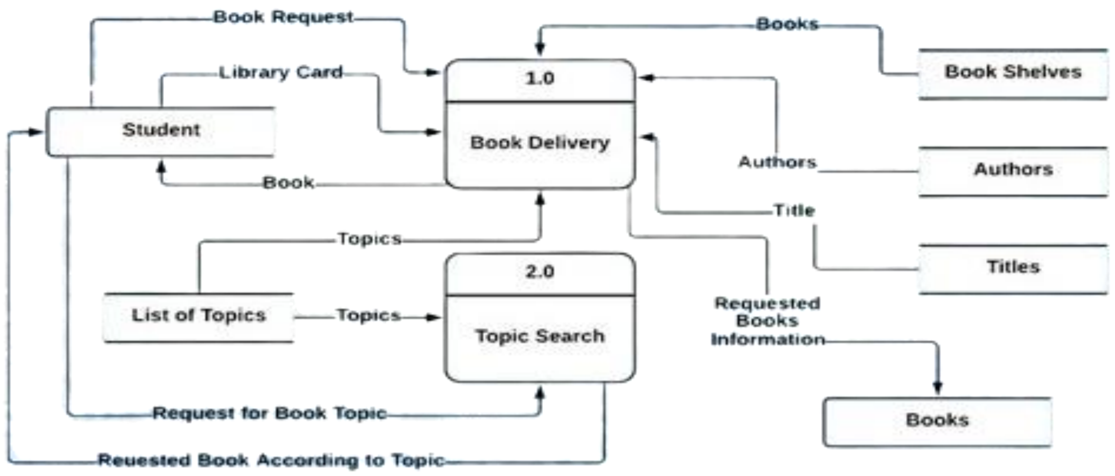
Practical No 04

Aim: Structured Data Flow Analysis(Library Management System)

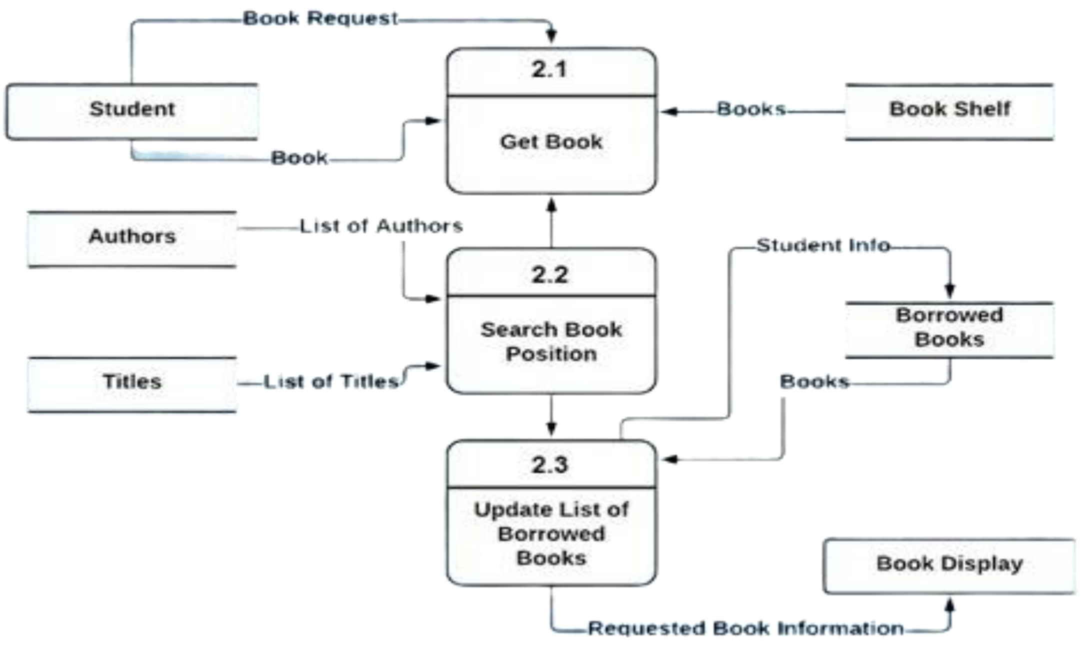
Level 0:



Level 1:



Level 2:



Book Request
Student
Book
2.1
Get Book
Books
Book Shelf
Authors
List of Authors
2.2
Search Book Position
Student Info
Borrowed Books
Titles
List of Titles
Books
2.3
Update List of Borrowed Books
Book Display
Requested Book Information

Practical No: 05

Aim: Use of Metrics to estimate the cost

cost estimate for a Library Management System, following the format you provided:

Estimated LOC:

- **User interface**: 2,500 LOC
- **Database management**: 3,700 LOC
- **Search functionality**: 2,800 LOC
- **Borrow/Return functionality**: 3,200 LOC
- **User management**: 2,100 LOC
- **Reports generation**: 3,500 LOC
- **Notification system**: 1,800 LOC
- **Security and Access control**: 2,250 LOC
- **Integration with external systems**: 1,500 LOC
- **Total estimated LOC**: 23,350 LOC

Average productivity based on historical data:

- **Productivity**: 620 LOC/pm
- **Cost per LOC**: Rs. 12.90/LOC
- **Cost per person-month (PP)**: Rs. 8,000 per month/PP

Cost and Effort Estimation:

- **Total estimated project cost**: Rs. 301,215 (23,350 x 12.90)
- **Estimated effort**: 38 person-months (23,350 / 620)

Function Points Estimation:

Given the functional units and their respective numbers:

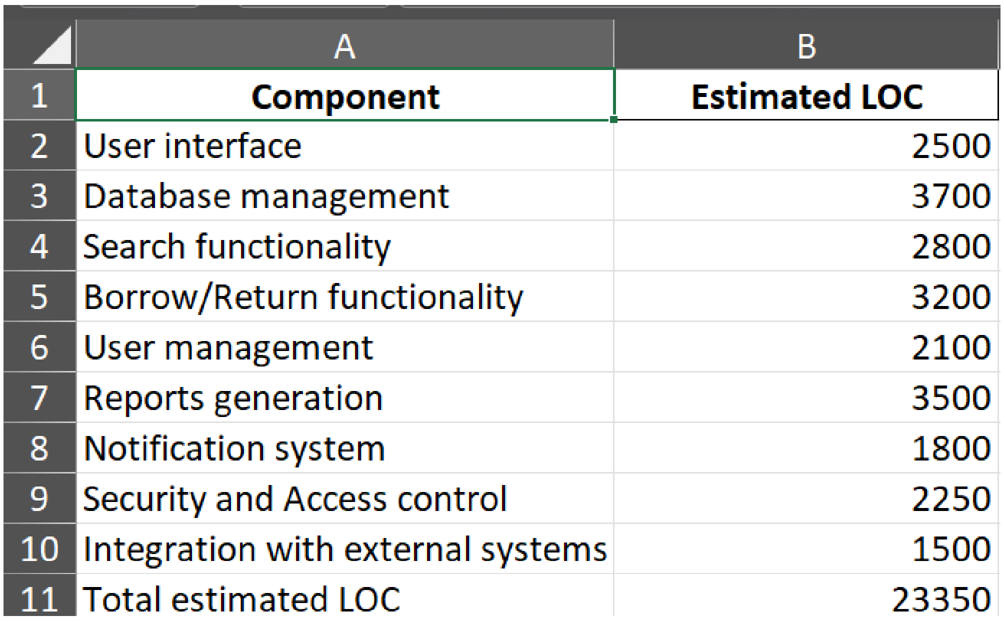
- **Number of user inputs**: 50
- **Number of user outputs**: 40
- **Number of user inquiries**: 35
- **Number of user files**: 6
- **Number of external interfaces**: 4

Assuming all complexity adjustment factors and weighing factors as average, the function points (FP) for the project are calculated as:

Function Points=(200+200+140+60+28)×[0.65+(0.01×(14×3))]\text{Function Points} = (200 + 200 + 140 + 60 + 28) \times \left[0.65 + \left(0.01 \times (14 \times 3)\right)\right]Function Points=(200+200+140+60+28)×[0.65+(0.01×(14×3))]
Function Points=628×(1.07)=672\text{Function Points} = 628 \times \left(1.07\right) = 672Function Points=628×(1.07)=672

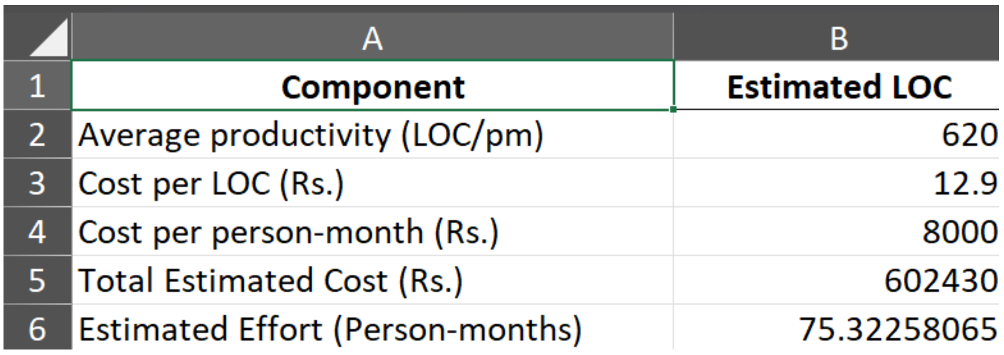
Thus, the function points for the Library Management System project are 672.

Cost Estimate



	A	B
1	**Component**	**Estimated LOC**
2	User interface	2500
3	Database management	3700
4	Search functionality	2800
5	Borrow/Return functionality	3200
6	User management	2100
7	Reports generation	3500
8	Notification system	1800
9	Security and Access control	2250
10	Integration with external systems	1500
11	Total estimated LOC	23350

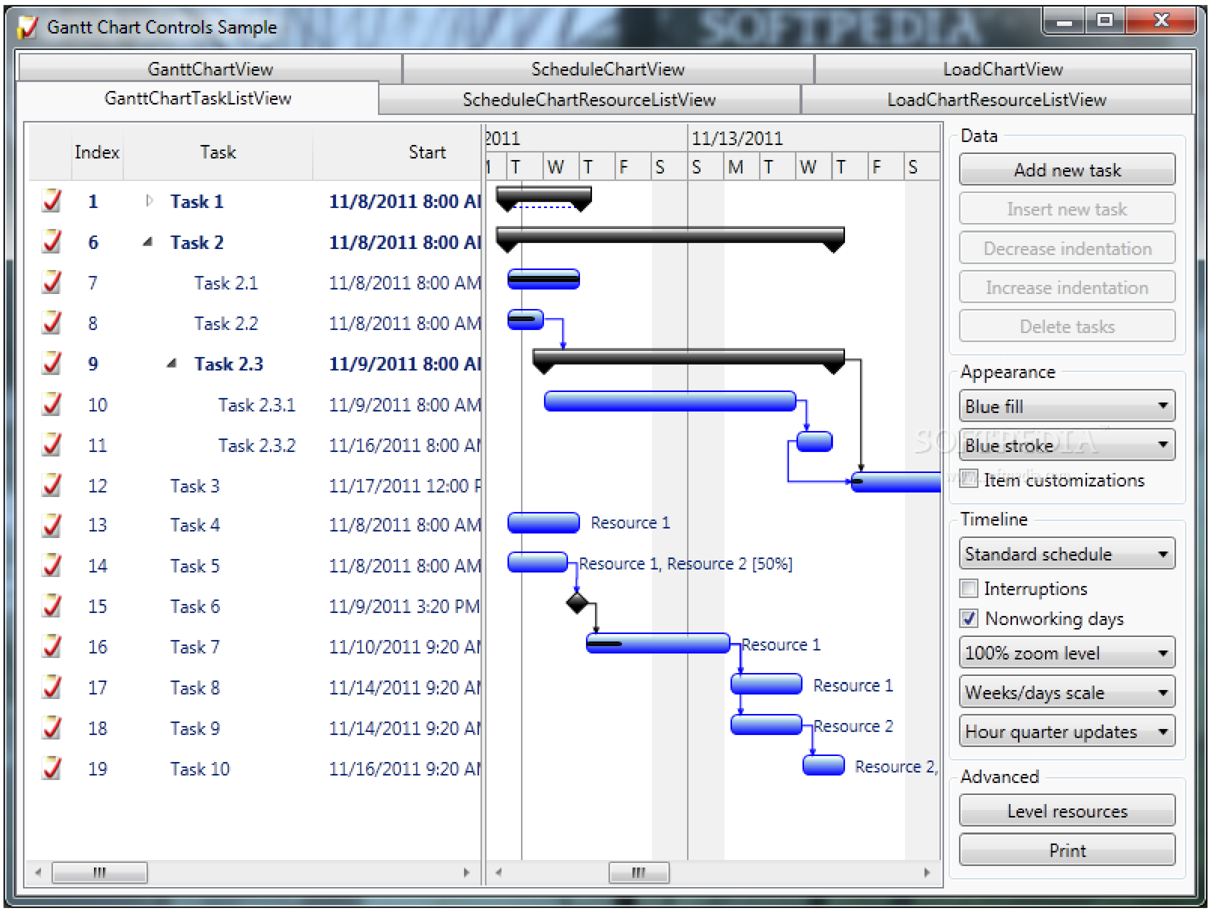
Summary



	A	B
1	**Component**	**Estimated LOC**
2	Average productivity (LOC/pm)	620
3	Cost per LOC (Rs.)	12.9
4	Cost per person-month (Rs.)	8000
5	Total Estimated Cost (Rs.)	602430
6	Estimated Effort (Person-months)	75.32258065

Practical No: 06

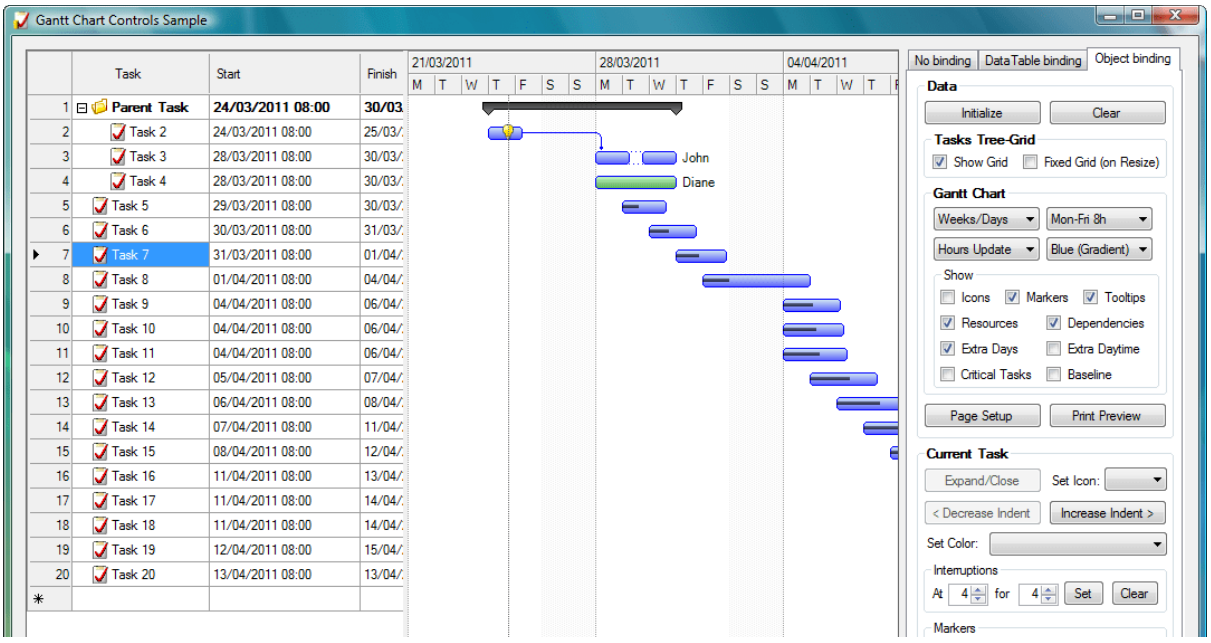
Aim: Scheduling and tracking of the project


Gantt Chart Controls Sample

Index	Task	Start
1	**Task 1**	**11/8/2011 8:00 AI**
6	**Task 2**	**11/8/2011 8:00 AI**
7	Task 2.1	11/8/2011 8:00 AM
8	Task 2.2	11/8/2011 8:00 AM
9	**Task 2.3**	**11/9/2011 8:00 AI**
10	Task 2.3.1	11/9/2011 8:00 AM
11	Task 2.3.2	11/16/2011 8:00 AI
12	Task 3	11/17/2011 12:00 F
13	Task 4	11/8/2011 8:00 AM
14	Task 5	11/8/2011 8:00 AM
15	Task 6	11/9/2011 3:20 PM
16	Task 7	11/10/2011 9:20 AI
17	Task 8	11/14/2011 9:20 AI
18	Task 9	11/14/2011 9:20 AI
19	Task 10	11/16/2011 9:20 AI

Appearance: Blue fill; Blue stroke; Item customizations

Timeline: Standard schedule; Interruptions; [✓ Nonworking days]; 100% zoom level; Weeks/days scale; Hour quarter updates


Gantt Chart Controls Sample

	Task	Start	Finish
1	**Parent Task**	**24/03/2011 08:00**	**30/03**
2	Task 2	24/03/2011 08:00	25/03/
3	Task 3	28/03/2011 08:00	30/03/
4	Task 4	28/03/2011 08:00	30/03/
5	Task 5	29/03/2011 08:00	30/03/
6	Task 6	30/03/2011 08:00	31/03/
7	Task 7	31/03/2011 08:00	01/04/
8	Task 8	01/04/2011 08:00	04/04/
9	Task 9	04/04/2011 08:00	06/04/
10	Task 10	04/04/2011 08:00	06/04/
11	Task 11	04/04/2011 08:00	06/04/
12	Task 12	05/04/2011 08:00	07/04/
13	Task 13	06/04/2011 08:00	08/04/
14	Task 14	07/04/2011 08:00	11/04/
15	Task 15	08/04/2011 08:00	12/04/
16	Task 16	11/04/2011 08:00	13/04/
17	Task 17	11/04/2011 08:00	14/04/
18	Task 18	11/04/2011 08:00	14/04/
19	Task 19	12/04/2011 08:00	15/04/
20	Task 20	13/04/2011 08:00	13/04/
*			

Tasks Tree-Grid: [✓ Show Grid] Fixed Grid (on Resize)

Gantt Chart: Weeks/Days; Mon-Fri 8h; Hours Update; Blue (Gradient)

Show: Icons [✓ Markers] [✓ Tooltips] [✓ Resources] [✓ Dependencies] [✓ Extra Days] Extra Daytime; Critical Tasks; Baseline

Interruptions: At 4 for 4

Practical No: 07

Aim: Write test cases for black box testing

Test Case 1: User Login

- **Test Case ID:** TC_LMS_001
- **Description:** Verify that registered users can log in with valid credentials.
- **Preconditions:** User account must be registered.
- **Test Steps:**
    1. Navigate to the login page.
    2. Enter a valid username.
    3. Enter a valid password.
    4. Click the "Login" button.
- **Expected Result:** User is successfully logged in and redirected to the dashboard.
- **Status: Pass**

Test Case 2: Add New Book

- **Test Case ID:** TC_LMS_002
- **Description:** Verify that a librarian can add a new book to the library database.
- **Preconditions:** Librarian must be logged in.
- **Test Steps:**
    1. Navigate to the "Add New Book" section.
    2. Enter the book title, author, ISBN, and other required details.
    3. Click the "Add Book" button.
- **Expected Result:** The book is added to the database, and a confirmation message is displayed.
- **Status: Pass**

Test Case 3: Search for a Book

- **Test Case ID:** TC_LMS_003
- **Description:** Verify that users can search for a book by title, author, or ISBN.
- **Preconditions:** Books must be present in the library database.
- **Test Steps:**
    1. Navigate to the search page.
    2. Enter the book title, author, or ISBN in the search field.
    3. Click the "Search" button.
- **Expected Result:** The search results display the correct book(s) based on the search criteria.
- **Status: Pass**

Test Case 4: Borrow a Book

- **Test Case ID:** TC_LMS_004
- **Description:** Verify that a user can borrow a book from the library.
- **Preconditions:**
    - The user must be logged in.
    - The book must be available for borrowing.
- **Test Steps:**

1. Search for a book.
2. Click on the "Borrow" button next to the book.
3. Confirm the borrow action.

- **Expected Result:** The book is marked as borrowed, and the user's borrowing history is updated.

- **Status: Pass**

Test Case 5: Return a Book

- **Test Case ID:** TC_LMS_005
- **Description:** Verify that a user can return a borrowed book.
- **Preconditions:**
    - The user must be logged in.
    - The book must be currently borrowed by the user.
- **Test Steps:**

1. Navigate to the "My Borrowed Books" section.
2. Click on the "Return" button next to the borrowed book.
3. Confirm the return action.

- **Expected Result:** The book is marked as returned, and it is available for borrowing again.
- **Status: Pass**

Test Case 6: Overdue Book Notification

- **Test Case ID:** TC_LMS_006
- **Description:** Verify that the system sends notifications for overdue books.
- **Preconditions:**
    - The user must have a book that is overdue.
    - The notification system should be functional.
- **Test Steps:**

1. Wait for the borrowed book to become overdue.
2. Check the notification system (email/SMS/in-app).

- **Expected Result:** The user receives a notification about the overdue book.
- **Status: Pass**

Test Case 7: Add New User

- **Test Case ID:** TC_LMS_007
- **Description:** Verify that an admin can add a new user to the system.
- **Preconditions:** Admin must be logged in.
- **Test Steps:**
    1. Navigate to the "Add New User" section.
    2. Enter user details like name, email, and user role.
    3. Click the "Add User" button.
- **Expected Result:** The user is successfully added to the system, and a confirmation message is displayed.
- **Status: Pass**

Test Case 8: Generate Report

- **Test Case ID:** TC_LMS_008
- **Description:** Verify that the librarian can generate reports on borrowed and available books.
- **Preconditions:** Librarian must be logged in.
- **Test Steps:**
    1. Navigate to the "Reports" section.
    2. Select the report type (e.g., borrowed books, available books).
    3. Click the "Generate Report" button.
- **Expected Result:** The system generates the report and displays it or allows downloading.
- **Status: Pass**

Test Case 9: Search Book Not Found

- **Test Case ID:** TC_LMS_009
- **Description:** Verify that the system handles cases where a searched book is not found.
- **Preconditions:** Search for a book that does not exist in the library database.
- **Test Steps:**
    1. Navigate to the search page.
    2. Enter a non-existing book title, author, or ISBN in the search field.
    3. Click the "Search" button.
- **Expected Result:** The system displays a message indicating that no matching book was found.
- **Status: Pass**

Test Case 10: User Logout

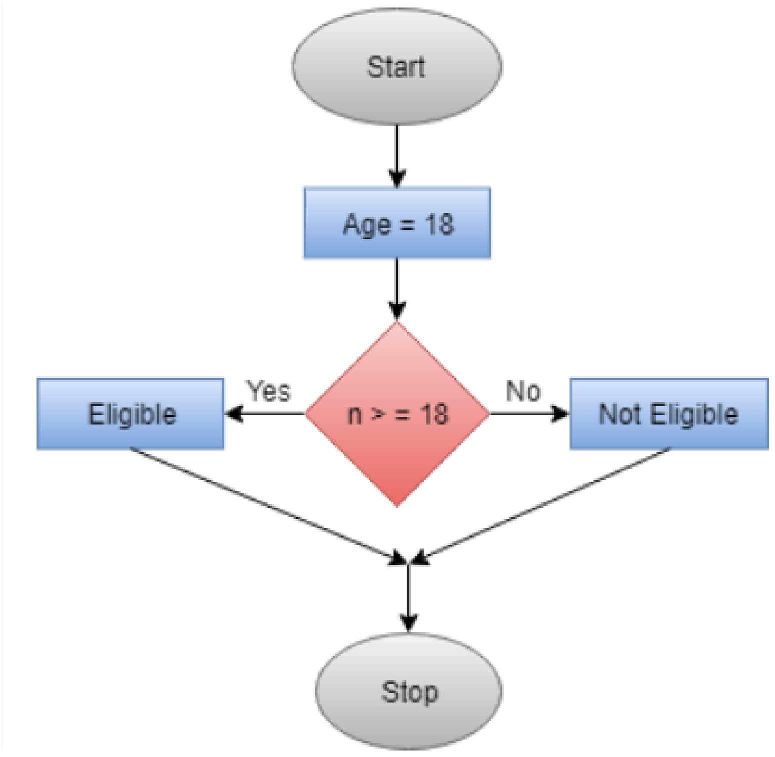
- **Test Case ID:** TC_LMS_010
- **Description:** Verify that a user can successfully log out of the system.
- **Preconditions:** User must be logged in.
- **Test Steps:**
    1. Click the "Logout" button.
- **Expected Result:** The user is logged out, and the login page is displayed.
- **Status: Pass**

Practical 8

Control Flow Testing

```
 public class VoteEligiblityAge{
  public static void main(String []args){
int n=45;
if(n>=18)
{
    System.out.println("You are eligible for voting");
 }  else
{
   System.out.println("You are not eligible for voting");
 }
}
}
```

Output



Practical no 9

RMMM Plan for Hospital Management System (HMS)

---

1. Introduction

Provide an overview of the HMS project and the importance of an RMMM plan. Mention how this plan will help to anticipate risks and handle them effectively.

Example:

The Hospital Management System (HMS) aims to streamline hospital operations, including patient records, billing, and staff management. This RMMM plan identifies potential risks, outlines mitigation strategies, and defines monitoring and response actions to minimize the impact of these risks on the project.

---

2. Risk Identification

List the potential risks for the HMS project. These could be technical, operational, or project-related risks. Here are a few examples:

Risk ID	Risk Description
1	Delays in integrating third-party APIs for medical data
2	Insufficient server capacity causing system downtime
3	Data security breaches of patient information
4	Unavailability of core team members
5	Regulatory non-compliance with healthcare standards

---

3. Risk Assessment

Assess the identified risks in terms of **likelihood** and **impact**. Use a risk matrix or similar tools to categorize them. Here's an example table to include:

Risk ID	Likelihood (Low/Medium/High)	Impact (Low/Medium/High)	Priority
1	Medium	High	High
2	High	High	High
3	Medium	High	High
4	Low	Medium	Medium

Risk ID	Likelihood (Low/Medium/High)	Impact (Low/Medium/High)	Priority
5	Medium	High	High

---

4. Risk Mitigation Strategies

For each identified risk, define strategies to mitigate or reduce its impact. Include detailed actions and steps.

Risk ID	Mitigation Strategy
1	Ensure early API integration testing with sandbox environments, set up alternative APIs as backup
2	Use cloud-based infrastructure for scalability, regular load testing
3	Implement encryption and access control policies, conduct regular security audits
4	Cross-train team members, have an HR contingency plan for hiring contractors if needed
5	Regularly review healthcare regulations, hire a compliance expert

---

5. Risk Monitoring

Explain how each risk will be monitored throughout the project lifecycle. Establish monitoring intervals and assign responsible teams.

Risk ID	Monitoring Method	Frequency	Responsible Team
1	API performance logs, testing	Weekly	Development Team
2	Server health checks, cloud usage reports	Daily	IT/DevOps Team
3	Security audits, log monitoring	Monthly	Security Team
4	Team member status meetings	Weekly	HR/Project Management
5	Compliance reviews, legal consultations	Quarterly	Compliance/Legal Team

---

6. Risk Management/Response Plan

Define what steps should be taken if a risk becomes an issue. Provide a concrete response plan for each risk.

Risk ID	Response Plan
1	If API integration fails, switch to backup API and inform stakeholders about timeline changes
2	Scale up server capacity through the cloud provider, and notify IT and users of downtime
3	In case of a breach, immediately isolate affected systems, notify authorities, and launch a full security investigation
4	Redistribute workload among cross-trained team members or hire external contractors
5	Pause implementation, update system to comply with new regulations, and conduct internal audits

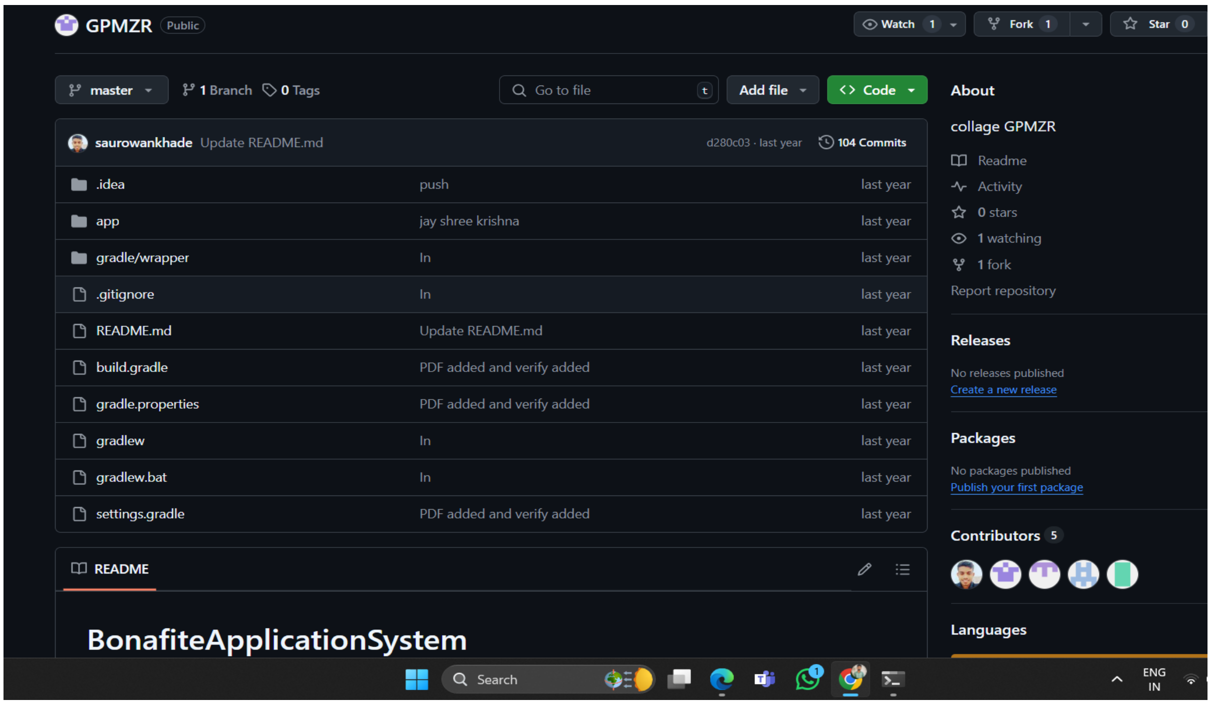
---

7. Review and Update Process

Mention how frequently the RMMM plan will be reviewed and updated to reflect new risks or changes in the project.

The RMMM plan will be reviewed every two months and updated based on the evolving project needs and external factors. Any significant risk event will trigger an immediate review.

Practical 10



GPMZR Public

Watch 1 · Fork 1 · Star 0

master · 1 Branch · 0 Tags · Go to file · Add file · Code

saurowankhade Update README.md · d280c03 · last year · 104 Commits

.idea	push	last year
app	jay shree krishna	last year
gradle/wrapper	In	last year
.gitignore	In	last year
README.md	Update README.md	last year
build.gradle	PDF added and verify added	last year
gradle.properties	PDF added and verify added	last year
gradlew	In	last year
gradlew.bat	In	last year
settings.gradle	PDF added and verify added	last year

README

BonafiteApplicationSystem

About

collage GPMZR

Readme
Activity
0 stars
1 watching
1 fork
Report repository

Releases

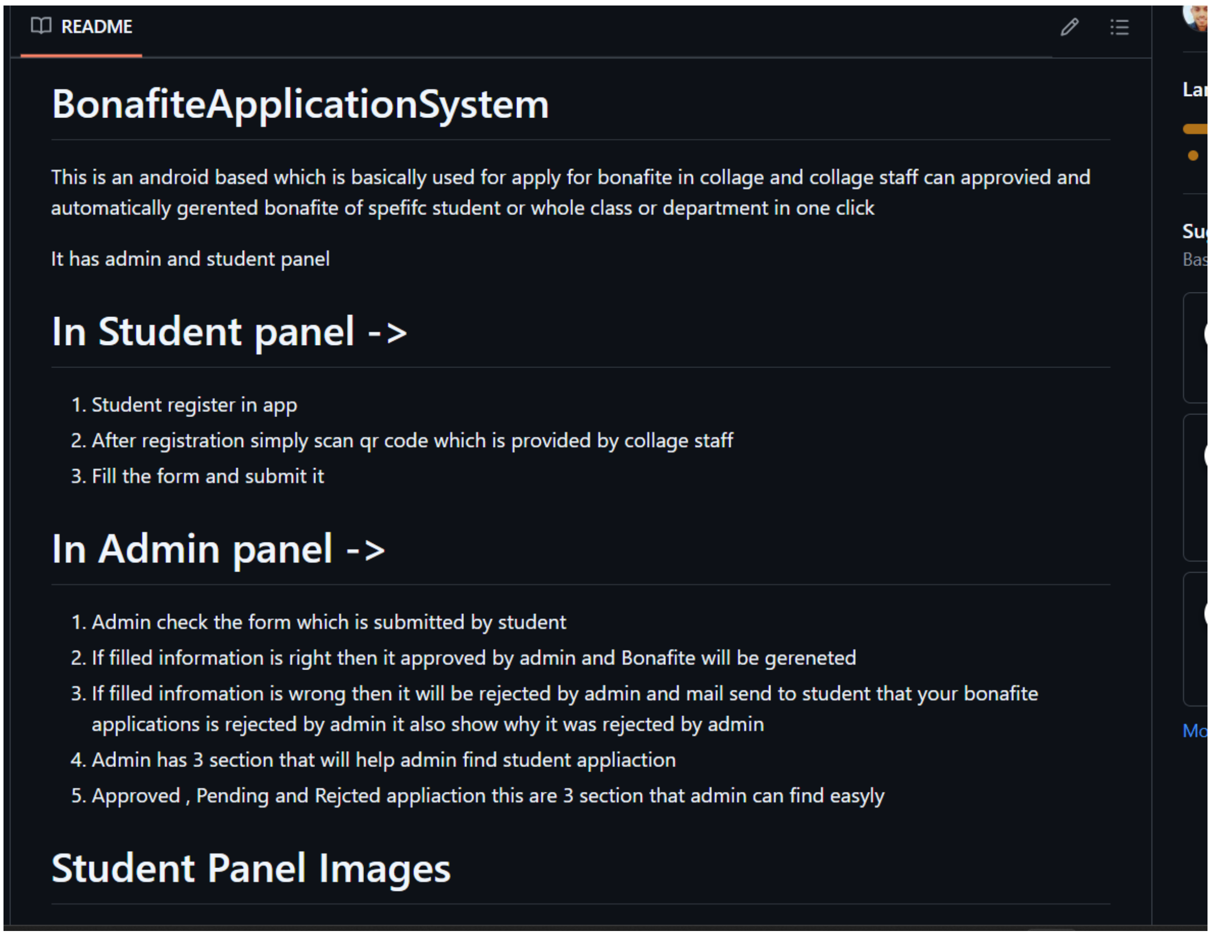
No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 5

Languages



README

BonafiteApplicationSystem

This is an android based which is basically used for apply for bonafite in collage and collage staff can approvied and automatically gerented bonafite of spefifc student or whole class or department in one click

It has admin and student panel

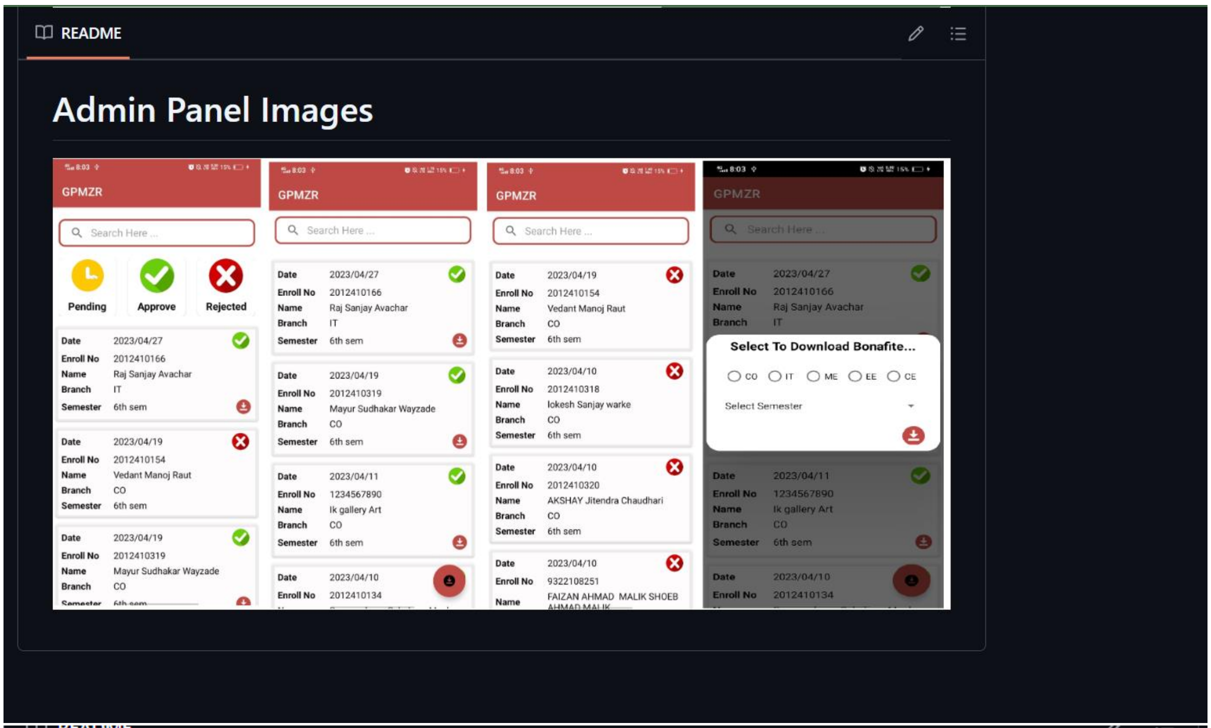
In Student panel ->

1. Student register in app
2. After registration simply scan qr code which is provided by collage staff
3. Fill the form and submit it

In Admin panel ->

1. Admin check the form which is submitted by student
2. If filled information is right then it approved by admin and Bonafite will be gereneted
3. If filled infromation is wrong then it will be rejected by admin and mail send to student that your bonafite applications is rejected by admin it also show why it was rejected by admin
4. Admin has 3 section that will help admin find student appliaction
5. Approved , Pending and Rejcted appliaction this are 3 section that admin can find easyly

Student Panel Images



Admin Panel Images

GPMZR

Search Here ...

Date	2023/04/27
Enroll No	2012410166
Name	Raj Sanjay Avachar
Branch	IT
Semester	6th sem

Date	2023/04/19
Enroll No	2012410154
Name	Vedant Manoj Raut
Branch	CO
Semester	6th sem

Date	2023/04/19
Enroll No	2012410319
Name	Mayur Sudhakar Wayzade
Branch	CO
Semester	6th sem

Date	2023/04/11
Enroll No	1234567890
Name	Ik gallery Art
Branch	CO
Semester	6th sem

Date	2023/04/10
Enroll No	2012410134

Date	2023/04/10
Enroll No	2012410318
Name	lokesh Sanjay warke
Branch	CO
Semester	6th sem

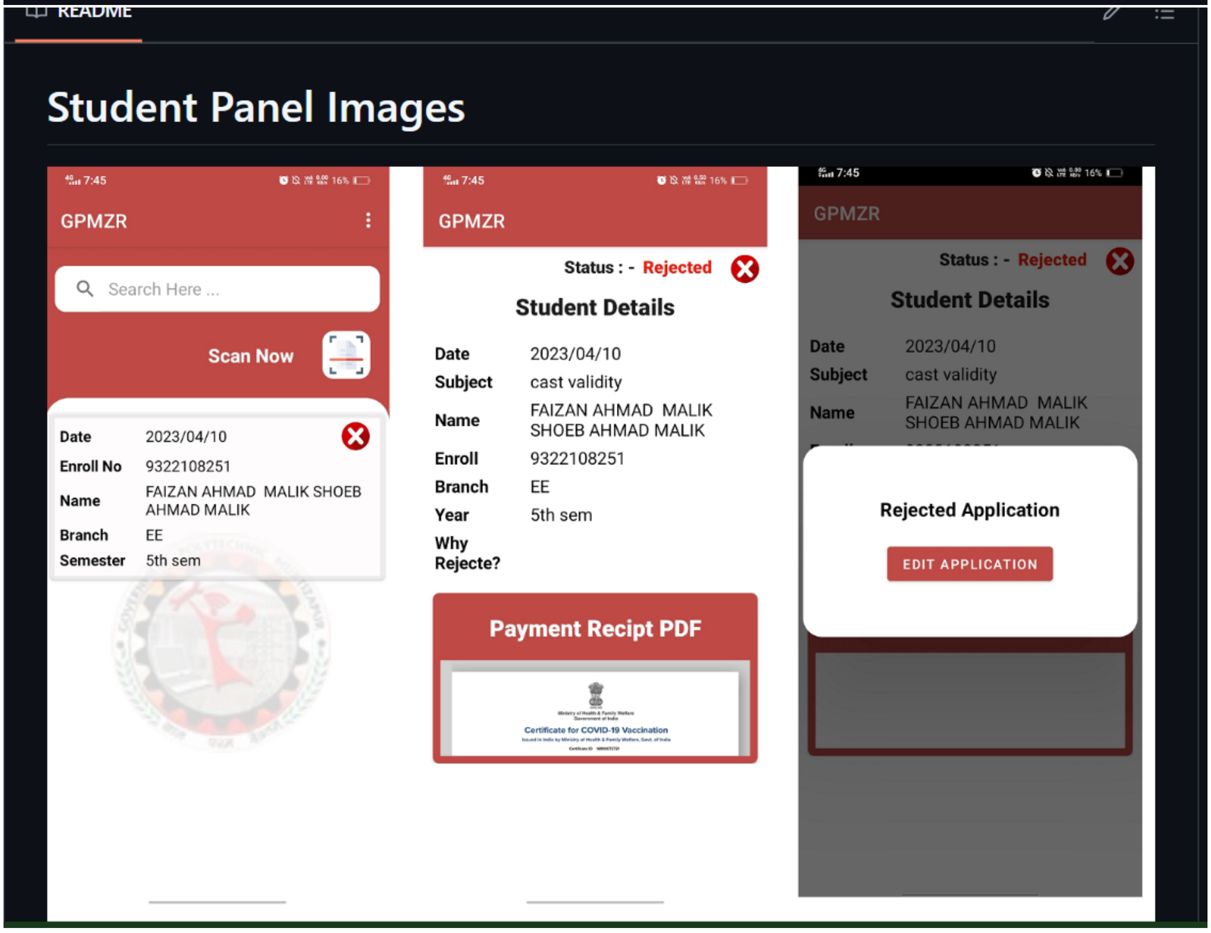
Date	2023/04/10
Enroll No	2012410320
Name	AKSHAY Jitendra Chaudhari
Branch	CO
Semester	6th sem

Date	2023/04/10
Enroll No	9322108251
Name	FAIZAN AHMAD MALIK SHOEB AHMAD MALIK

Select To Download Bonafite...

◯ CO ◯ IT ◯ ME ◯ EE ◯ CE

Select Semester



Student Panel Images

GPMZR

Search Here ...

Date	2023/04/10
Enroll No	9322108251
Name	FAIZAN AHMAD MALIK SHOEB AHMAD MALIK
Branch	EE
Semester	5th sem

Status : - Rejected

Student Details

Date	2023/04/10
Subject	cast validity
Name	FAIZAN AHMAD MALIK SHOEB AHMAD MALIK
Enroll	9322108251
Branch	EE
Year	5th sem
Why Rejecte?	

Payment Recipt PDF

Certificate for COVID-19 Vaccination

Rejected Application