SOFTWARE ENGINEERING LAB

B.Tech. II Year I Sem.

L	T	P	C
0	0	2	1

Prerequisites

- A course on “Programming for Problem Solving”.

Co-requisite

- A Course on “Software Engineering”.

Course Objectives:

- To have hands on experience in developing a software project by using various software engineering principles and methods in each of the phases of software development.

Course Outcomes:

- Ability to translate end-user requirements into system and software requirements
- Ability to generate a high-level design of the system from the software requirements
- Will have experience and/or awareness of testing problems and will be able to develop a simple testing report

List of Experiments

Do the following seven exercises for any two projects given in the list of sample projects or any other Projects:

1. Development of problem statements.
2. Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents.
3. Preparation of Software Configuration Management and Risk Management related documents.
4. Study and usage of any Design phase CASE tool
5. Performing the Design by using any Design phase CASE tools.
6. Develop test cases for unit testing and integration testing
7. Develop test cases for various white box and black box testing techniques.

Sample Projects:

1. Passport automation System
2. Book Bank
3. Online Exam Registration
4. Stock Maintenance System
5. Online course reservation system
6. E-ticketing
7. Software Personnel Management System
8. Credit Card Processing
9. E-book management System.
10. Recruitment system

TEXT BOOKS:

1. Software Engineering, A practitioner’s Approach- Roger S. Pressman, 6th edition, McGraw Hill International Edition.
2. Software Engineering- Sommerville, 7th edition, Pearson Education.
3. The unified modeling language user guide Grady Booch, James Rambaugh, Ivar Jacobson, Pearson Education.

REFERENCE BOOKS:

1. Software Engineering, an Engineering approach- James F. Peters, Witold Pedrycz, John Wiley.
2. Software Engineering principles and practice- Waman S Jawadekar, The McGraw-Hill

Task 1: Development of problem statements

***<u>Passport Automation System</u>*:**

Problem Statement: Passport Automation System is used in the effective dispatch of passport to all the eligible applicants after several rounds of processing and verification. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a lucid manner.

The core of the system is to get the online registration form (with details such as name, address etc.,) filled by the applicant whose testament is verified for its genuineness by the Passport Automation System with respect to the already existing information in the database. After the first round of verification done by the system, the information is in turn forwarded to the regional administrator's Ministry of External Affairs office. The application is then processed manually based on the report given by the system, and any forfeiting identified can make the applicant liable to penalty as per the law. The system forwards the necessary details to the police for its separate verification whose report is then presented to the administrator. After all the necessary criteria have been met, the original information is added to the database and the passport is sent to the applicant.

Here are a few issues with the manual system that could be addressed through the development of a Passport Automation System:

Manual Processing Inefficiencies:
The current passport issuance process relies heavily on manual data entry, paper-based forms, and lengthy queues at passport offices. This leads to delays, errors, and frustrated applicants.

Data Security and Privacy Concerns:
The manual handling of sensitive personal information during the passport application process raises concerns about data security and privacy breaches. Unauthorized access to personal data is a significant risk.

Inconsistent Application Processing:
Passport offices in different regions or countries may have varying procedures and processing times. This results in inconsistent experiences for applicants and a lack of centralized oversight.

Long Processing Times:
The existing process often requires applicants to wait for extended periods before receiving their passports. This delay can be problematic for individuals who need to travel urgently.

Lack of Application Tracking:
Applicants have limited visibility into the status of their passport applications. This lack of tracking capability makes it challenging for them to plan their travel arrangements.

Manual Verification Processes:
Verification of submitted documents and applicant information is time-consuming due to manual processes. This can lead to errors in verifying authenticity and identity.

Limited Accessibility:
Not all applicants can easily access physical passport offices due to geographical constraints. This limits access to passport services for remote or rural populations.

Paper-Based Documentation:
The reliance on physical documents creates challenges in terms of storage, retrieval, and sharing of information. There's a need for a more efficient digital document management system.

Lack of Integration with other Systems:
Existing passport systems might not be integrated with other government or immigration systems, leading to redundant data entry and inefficiencies.

Complex Documentation Requirements:
Applicants often struggle to understand and full fill the documentation requirements for passport applications, leading to rejections and repeated visits.

Inefficient Appointment Scheduling:
The current system might not offer efficient appointment scheduling mechanisms, causing long waiting times and dissatisfaction among applicants.

Manual Payment Processing:
Payment processes for passport fees might involve manual cash handling, leading to accounting discrepancies and potential corruption.

These problems/issues highlight the key challenges and inefficiencies that a Passport Automation System could address. The system aims to streamline processes, enhance security, improve applicant experiences, and provide efficient tracking and management of passport applications.

RECORD NOTES:

Task 2: Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents

1. <u>Software Requirement Specification (SRS) Document:</u>

The SRS document outlines the detailed requirements for the Passport Automation System. It defines the system's functionalities, constraints, and user expectations. The SRS document typically includes the following sections:

Introduction: Provides an overview of the document, its purpose, and the scope of the Passport Automation System.
System Overview: Describes the system's high-level architecture, components, and interactions with external systems.

Functional Requirements: Specifies detailed functionalities of the system. Each requirement should be clear, concise, and unambiguous. Use cases and user stories can be helpful here.

Non-Functional Requirements: Addresses system qualities like performance, scalability, security, usability, and reliability.

User Interfaces: Describes the user interfaces, including mockups or wireframes if available.

Data Requirements: Defines the data structures, databases, and data flow within the system.

System Architecture: Provides an overview of the system's technical architecture, including hardware and software components.

Dependencies: Lists external systems, libraries, or APIs that the system will depend on.

Constraints: Specifies any limitations or restrictions that the system must adhere to.

Assumptions: Documents any assumptions made during requirement gathering.

2. <u>Design Documents:</u>

Design documents elaborate on the technical architecture, system components, and implementation details of the Passport Automation System. These documents aid the development process. Common design documents include:

High-Level Design (HLD): Outlines the overall system architecture, component interactions, and major modules.

Low-Level Design (LLD): Provides in-depth details about each module, including data structures, algorithms, and interfaces.

Database Design: Describes the database schema, relationships, and data manipulation methods.

User Interface (UI) Design: Provides detailed information about the user interfaces, including layouts, styles, and user interactions.

API Design: Documents the APIs, endpoints, request-response formats, and authentication mechanisms.

3. <u>Testing Phase Related Documents</u>:

The testing phase requires a set of documents to ensure comprehensive testing and validation of the system:

Test Plan: Outlines the testing approach, test objectives, scope, testing strategies, and resources required for testing.

Test Cases: Provides detailed scenarios, inputs, expected outcomes, and procedures for testing various system functionalities.

Test Scripts/Automation: If applicable, provides automated scripts for executing test cases.

Test Data: Specifies the data needed to perform tests effectively.

Defect Reports: Documents any defects found during testing, including steps to reproduce and severity levels.

Traceability Matrix: Links test cases back to the specific requirements, ensuring all requirements are tested.

Test Summary/Report: Summarizes the testing process, results, and any open issues.

Remember that these documents should be continuously updated throughout the development process to reflect any changes or additions. The documents will serve as valuable references for developers, testers, and stakeholders, ensuring a well-organized and successful development and testing process for the Passport Automation System.

SOFTWARE REQUIREMENT SPECIFICATION (SRS) FOR PASSPORT AUTOMATION SYSTEM

PURPOSE

If the entire process of 'Issue of Passport' is done in a manual manner then it would takes several months for the passport to reach the applicant. Considering the fact that the number of applicants for passport is increasing every year, an Automated System becomes essential to meet the demand. So this system uses several programming and database techniques to elucidate the work involved in this process. As this is a matter of National Security, the system has been carefully verified and validated in order to satisfy it.

SCOPE

- The System provides an online interface to the user where they can fill in their personal details and submit the necessary documents (maybe by scanning).
- The authority concerned with the issue of passport can use thissystem to reduce his workload and process the application in aspeedy manner.
- Provide a communication platform between the applicant and theadministrator.
- Transfer of data between the Passport Issuing Authority and the Local Police for verification of applicant's information.

Users/Applicants will come to know their status of application and the date in which they must subject themselves for manual documentverification.

DEFINITIONS, ACRONYMS AND ABBREVIATIONS

- **Administrator**

  Refers to the super user who is the Central Authority with the privilege to manage the entire system. It can be any higher official in the Regional Passport Office of Ministry of External Affairs.
- **Applicant**

  One who wishes to obtain the Passport?
- **PAS**

  Refers to this Passport Automation System.
- **HTML**

  Markup Language used for creating web pages.

- **J2EE**

  Java 2 Enterprise Edition is a programming platform java platform for developing and running distributed java applications.
- **HTTP**

  Hyper Text Transfer Protocol.
- **TCP/IP**

  Transmission Control Protocol/Internet Protocol is the communication protocol used to connect hosts on the Internet.

TECHNOLOGIES TO BE USED

- HTML
- JSP
- JavaScript
- Java

TOOLS TO BE USED

- Eclipse IDE (Integrated Development Environment)
- Rational Rose tool (for developing UML Patterns)

OVERVIEW

SRS includes two sections overall description and specific requirements **Overall Description** will describe major role of the system components andinter- connections.

Specific Requirements will describe roles & functions of the actors.

OVERALL DESCRIPTION

PRODUCT PERSPECTIVE

The PAS acts as an interface between the 'applicant' and the 'administrator'. This system tries to make the interface as simple as possible and at the same time not risking the security of data stored in. This minimizes the time duration in which the user receives the passport.

SOFTWARE INTERFACE

- **Front End Client** - The applicant and Administrator online interface is built using JSP and HTML. The Administrator’s localinterface is built using Java.
- **Web Server** – Apache Tomcat application server (OracleCorporation).
- **Back End** – Oracle11g database.

HARDWARE INTERFACE

The server is directly connected to the client systems. The client systems have access to the database in the server.

SYSTEM FUNCTIONS

- Secure Registration of information by the Applicants.
- Schedule the applicants an appointment for manual verification oforiginal documents.
- Panel for Passport Application Status Display by the Administrator.
- SMS and Mail updates to the applicants by the administrator.
  Administrator can generate reports from the information and is the only authorized personnel to add the eligible application informationto the database.

USER CHARACTERISTICS

- **Applicant**
  These are the person who desires to obtain the passport and submit the information to the database.

- **Administrator**
  He has the certain privileges to add the passport status and to approve the issue of passport. He may contain a group of persons under him toverify the documents and give suggestion whether or not to approve the dispatch of passport.
- **Police**
  He is the person who upon receiving intimation from the PAS, perform a personal verification of the applicant and see if he has any criminal case against him before or at present. He has been vetoed with the power to decline an application by suggesting it to theAdministrator if he finds any discrepancy with the applicant. He communicates via this PAS.

CONSTRAINTS

- The applicants require a computer to submit their information.
- Although the security is given high importance, there is always achance of intrusion in the web world which requires constant monitoring.
  The user has to be careful while submitting the information. Much careis required.

ASSUMPTIONS AND DEPENDENCIES

- The Applicants and Administrator must have basic knowledge of computers and English Language.
- The applicants may be required to scan the documents and send.

RECORD NOTES

Task 3: Preparation of Software Configuration Management and Risk Management related documents.

Software Configuration Management (SCM) Document:

Introduction:
Brief overview of the document's purpose and scope.

Configuration Management Plan:
Outline the overall approach to configuration management.
Define roles and responsibilities of team members involved in SCM.
Specify the tools and resources to be used for configuration management.

Version Control:
Specify version control practices for source code, documents, and other artifacts.
Describe how branching, merging, and release management will be handled.

Configuration Identification:
Define naming conventions for components, files, and versions.
Explain how to uniquely identify and label each configuration item.

Configuration Control:
Detail the change management process, including how changes are requested, reviewed, and approved.
Describe the change tracking mechanism and how it's integrated with development activities.
Configuration Status Accounting:
Explain how configuration status will be tracked and reported.
Define the format and frequency of status reports.

Configuration Auditing:
Describe the process for conducting configuration audits to ensure compliance with standards and requirements.

Baseline Management:
Define how and when baselines will be established for different project phases.
Explain the process for managing changes after baselines are established.

Backup and Recovery:
Detail the backup and recovery strategy for configuration items and project data.

Release Management:

Describe the process for packaging, distributing, and deploying releases to different environments.

Risk Management Document:

Introduction:
Briefly explain the purpose and importance of risk management in the project.

Risk Management Plan:
Outline the overall approach to risk management throughout the project lifecycle.
Define roles and responsibilities of team members involved in risk management.

Risk Identification:
List and categorize potential risks related to the Passport Automation System project.
Include risks related to technical, organizational, and external factors.

Risk Assessment:
Evaluate and prioritize identified risks based on their impact and likelihood.
Create a risk matrix to visualize the severity of each risk.

Risk Mitigation Strategies:
Define strategies to mitigate high-priority risks.
Specify preventive and corrective actions for each risk.

Risk Monitoring and Control:
Explain how risks will be monitored throughout the project.
Describe the process for tracking risk triggers and implementing mitigation plans.

Risk Communication:
Define how risks and mitigation plans will be communicated to stakeholders.
Include a communication plan for addressing risks in a transparent manner.

Contingency Planning:
Detail contingency plans for handling risks that cannot be fully mitigated.

Lessons Learned:
Discuss how lessons learned from risk events will be documented and shared for future projects. Remember, both SCM and Risk Management documents should be tailored to your project's specific needs, and they should be reviewed and updated as the project progresses to ensure their continued relevance and effectiveness.

Task 4: Study and usage of any Design phase CASE tool

StarUML - Open Source Design Phase Case Tool

StarUML is an open source **Computer-aided software engineering (CASE)** tool that supports the UML (Unified Modeling Language) framework for system and software modeling. It is based on UML version 1.4, provides eleven different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages.

Website: https://staruml.io/

Version Tested: StarUML version 6.0.1, tested on Windows

Minimum System Requirements: Windows 2000, Windows XP, or higher; Microsoft Internet Explorer 5.0 or higher; 128 MB RAM (256MB recommended); 110 MB hard disc space (150MB space recommended)

License & Pricing: Open Source

Support: User mailing list

Installation

The installer follows the classic Windows install procedure without issues.

Documentation

The same help that could be browsed on the StarUML web site is available with the tool on your desktop. Documentation describes the concepts of tool but on high level vision. A more detailed documentation is available for the diagramming functions. Sample projects are provided with the tool and one of them contains the model of the tool itself, showing that the developers were able to eat their own dog food. Besides English, documentation exists in Korean, Japanese and Russian.

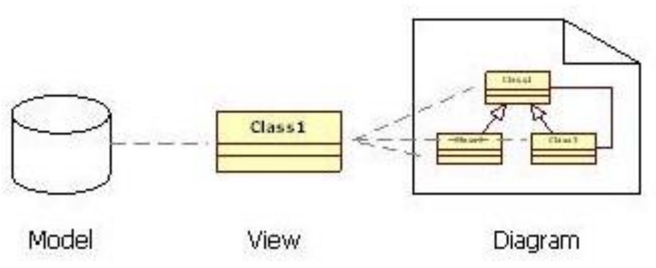
Configuration

Some general and diagram configurations options are available from the Tools/Option menu. You will find in this window also the configuration switches for the code generation. The interface is also very configurable as you can select what part of the tool you would like to view or not.

Features

When you start a new project, StarUML proposes which approach you want to use: 4+1 (Krutchen), Rational, UML components (from Cheesman and Daniels book), default or empty. Depending on the approach, profiles and/or frameworks may be included and loaded. If you don't follow a specific approach, the "empty" choice could be used. Although a project can be managed as one file, it may be convenient to divide it into many units and manage them separately if many developers are working on it together.

StarUML makes a clear conceptual distinction between models, views and diagrams. A Model is an element that contains information for a software model. A View is a visual expression of the information contained in a model, and a Diagram is a collection of view elements that represent the user's specific design thoughts.



StarUML supports the following diagram types

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- Statechart Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram
- Composite Structure Diagram

Task 5: Performing the Design by using any Design phase CASE tools

Ex no:1 **PASSPORT AUTOMATION SYSTEM**

AIM:

To create an automated system to perform the Passport Process.

(I) PROBLEM STATEMENT:

Passport Automation System is used in the effective dispatch of passport to all of the applicants. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a cogent manner. The core of the system is to get the online registration form (with details such as name, address etc.,) filled by the applicant whose testament is verified for its genuineness by the Passport Automation System with respect to the already existing information in the database.

(II)SOFTWARE REQUIREMENT SPECIFICATION:

2.1SOFTWARE INTERFACE

• **Front End Client -** The applicant and Administrator online interface is built using

JSP and HTML. The Administrators's local interface is built using Java.

•**Web Server -** Glassfish application server(Oracle Corporation).

•**Back End -** Oracle database.

2.2HARDWARE INTERFACE

The server is directly connected to the client systems. The client systems have access to the database in the server.

II) USECASE DIAGRAM :

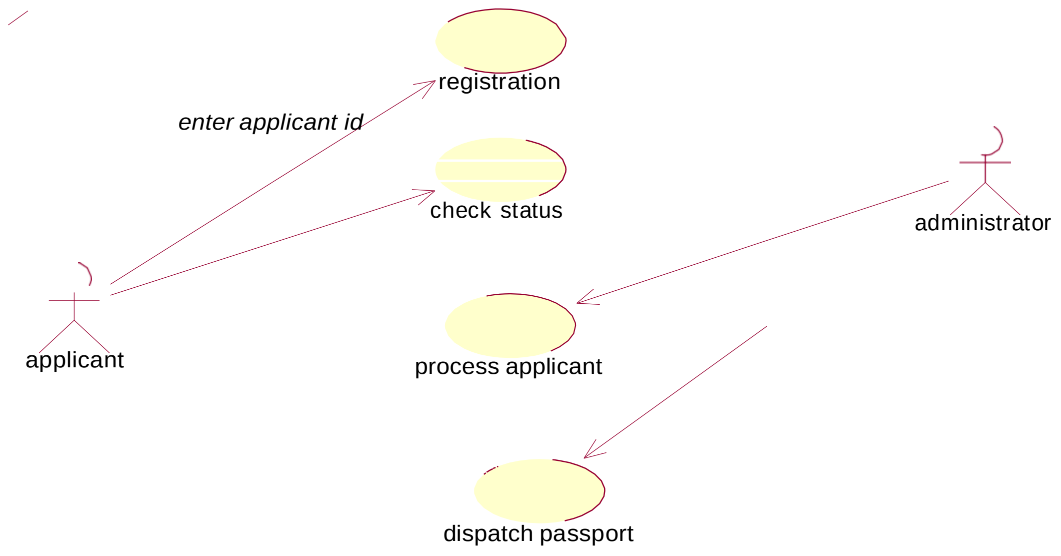


Fig.3. USECASE DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

(IV) ACTIVITY DIAGRAM:

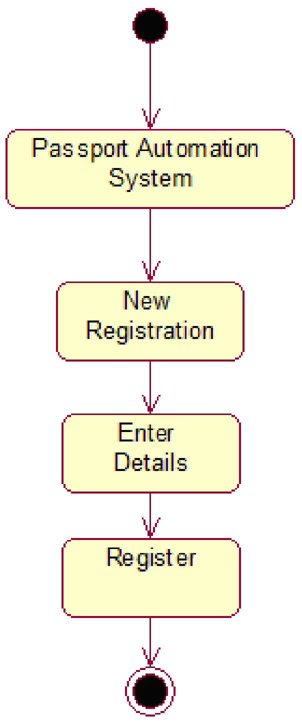


Fig.4.1. ACTIVITY DIAGRAM FOR REGISTER

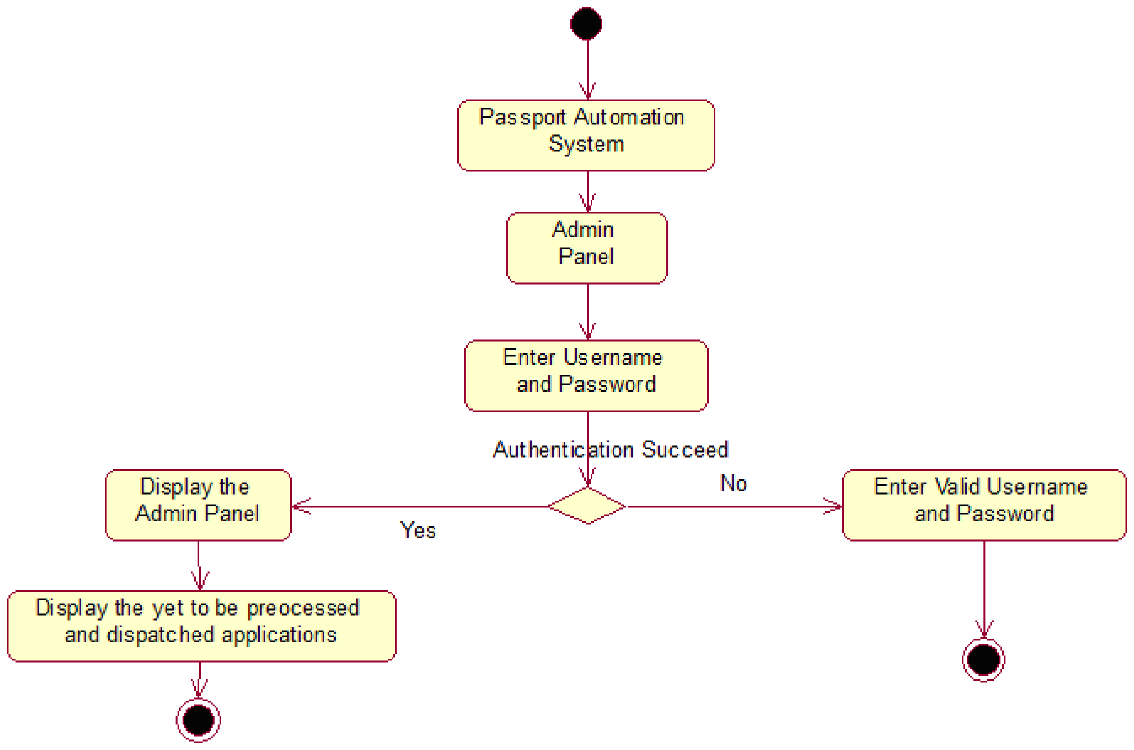

Passport Automation System
Admin Panel
Enter Username and Password
Authentication Succeed
No
Yes
Display the Admin Panel
Enter Valid Username and Password
Display the yet to be preocessed and dispatched applications

Fig.4.2. ACTIVITY DIAGRAM FOR ADMINISTRATION

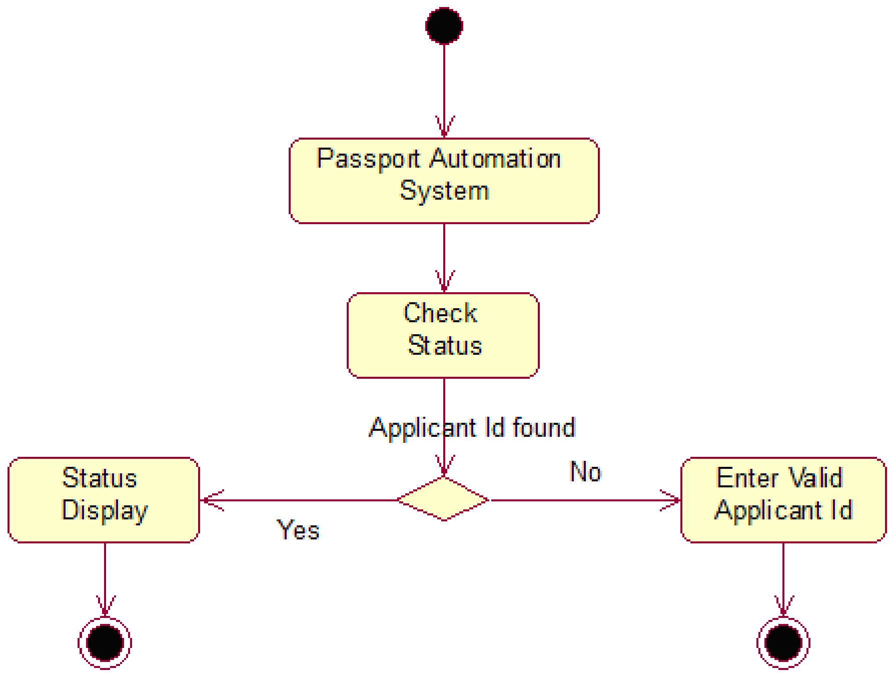


Fig.4.3. ACTIVITY DIAGRAM FOR CHECKING STATUS

(V) CLASS DIAGRAM:

The class diagram, also referred to as object modeling is the main static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.

The Passport Automation system class diagram consists of four classesPassport Automation System

1. New registration
2. Gender
3. Application Status
4. Admin authentication
5. Admin Panel

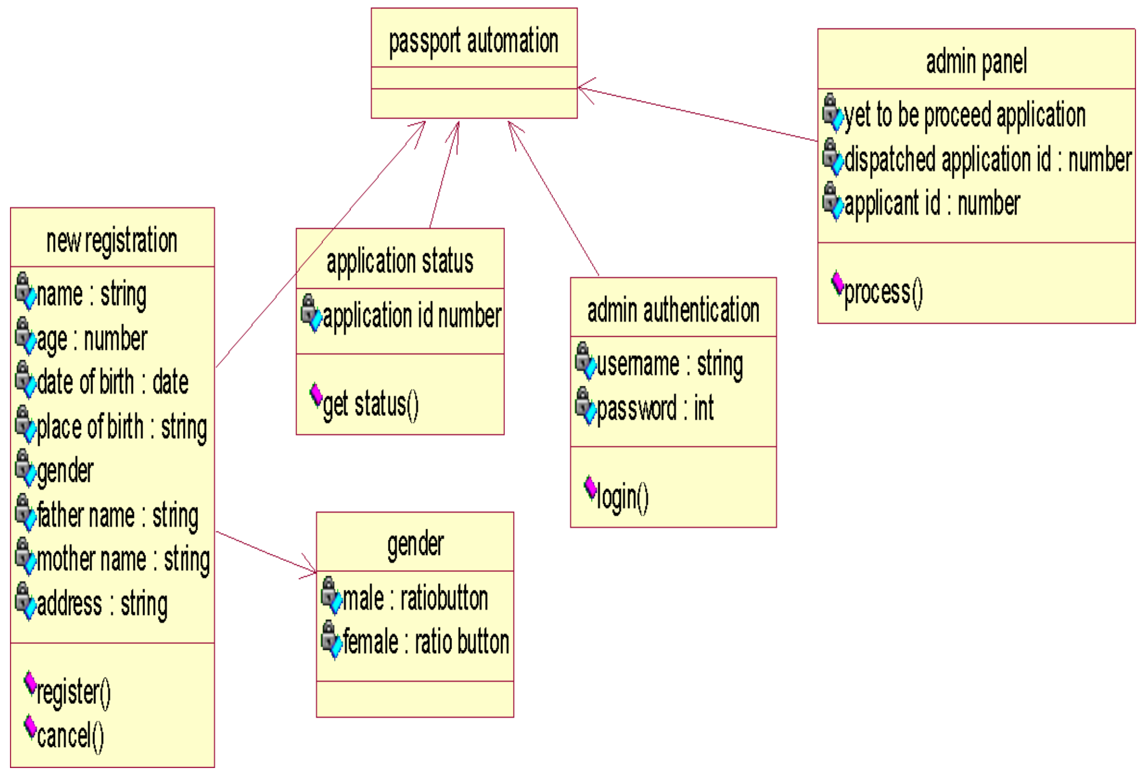


Fig.5. CLASS DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

(VI)) INTERACTION DIAGRAM:

A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the “form” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

The sequence diagram for each USE-CASE that exists when a user administrator, check status and new registration about passport automation system are given.

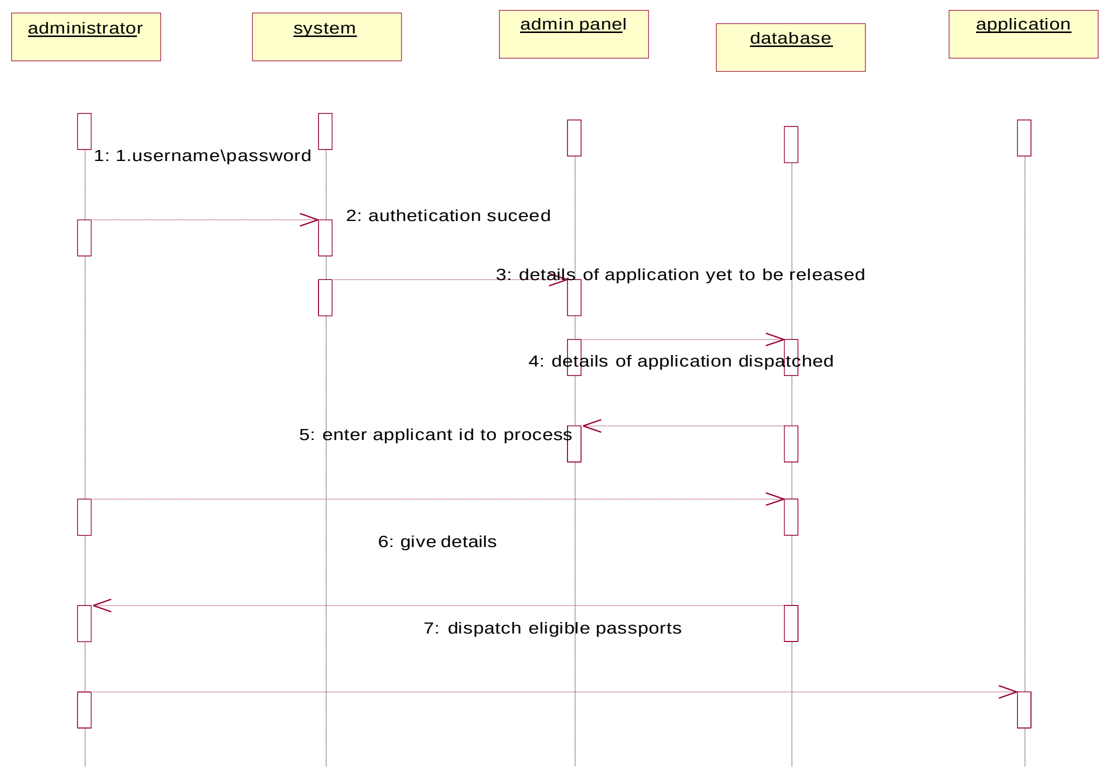

administrator
system
admin panel
database
application
1: 1.username\password
2: authetication suceed
3: details of application yet to be released
4: details of application dispatched
5: enter applicant id to process
6: give details
7: dispatch eligible passports

Fig.6.1.SEQUENCE DIAGRAM FOR ADMINISTRATOR

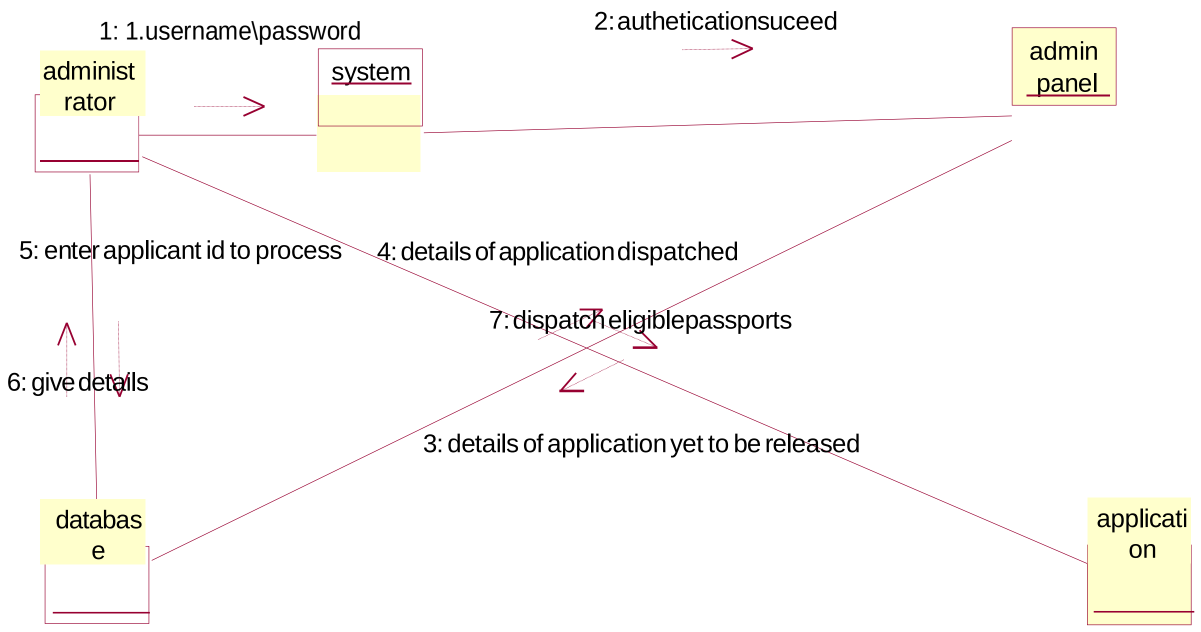

1: 1.username\password
2: autheticationsuceed
administrator
system
admin panel
5: enter applicant id to process
4: details of application dispatched
7: dispatch eligiblepassports
6: give details
3: details of application yet to be released
database
application

Fig.6.2.COLLABORATION DIAGRAM FOR ADMINISTRATOR

The diagrams show the process done by the administrator to the Passport Automation system. The applicant has to enter his details. The details entered are verified by the administrator and the applicant is approved if the details match then the passport is dispatch, otherwise an appropriate error message is displayed.

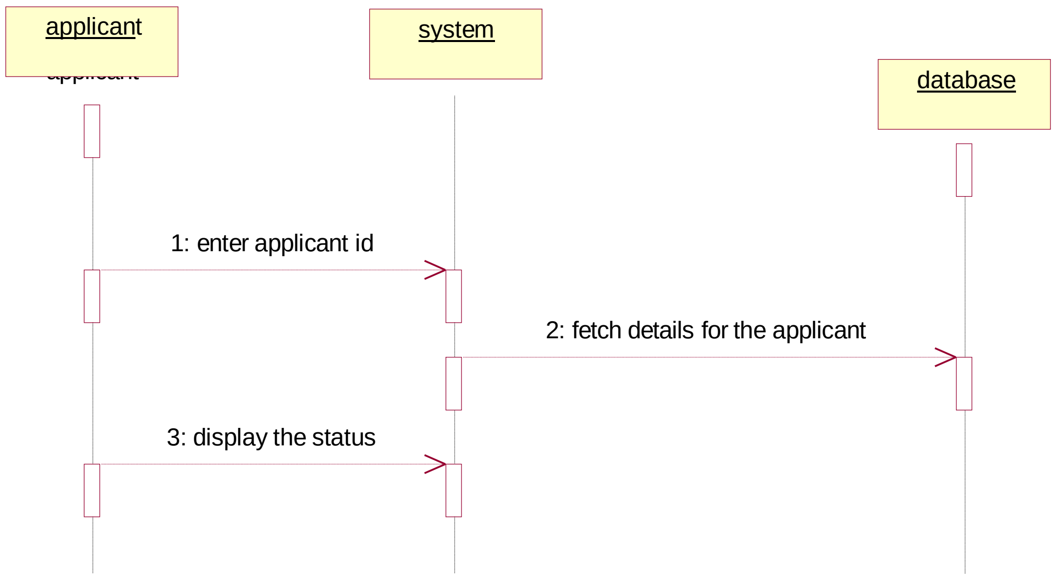


Fig.6.3.SEQUENCE DIAGRAM FOR CHECKING STATUS

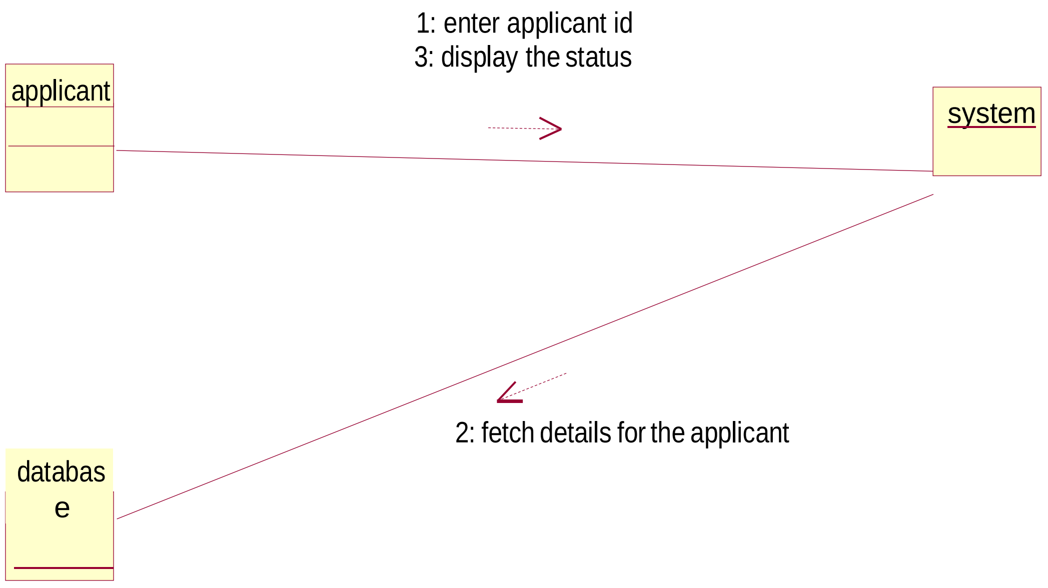


Fig.6.4.COLLABORATION DIAGRAM FOR CHECKING STATUS

The diagrams show the applicant enters his id and the system fetch the details from the database and display the status.

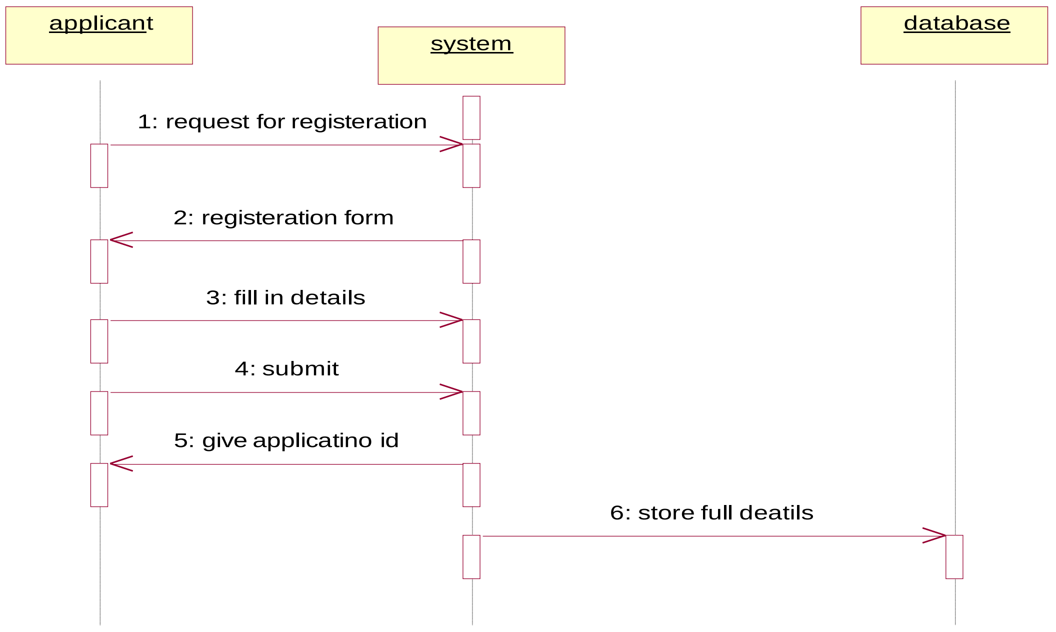


Fig.6.5.SEQUENCE DIAGRAM FOR NEW REGISTRATION

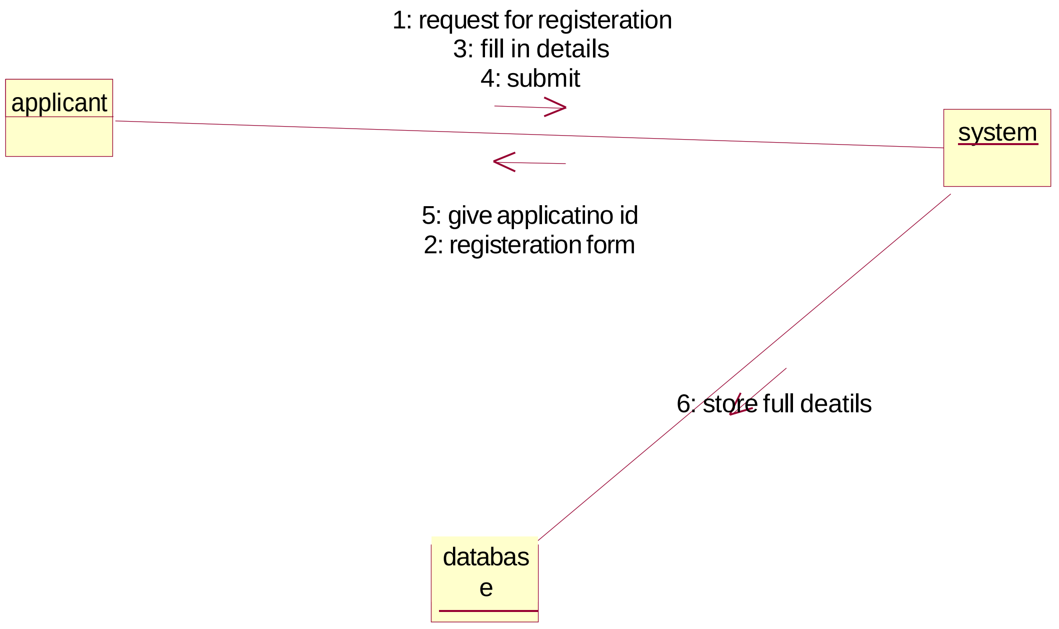
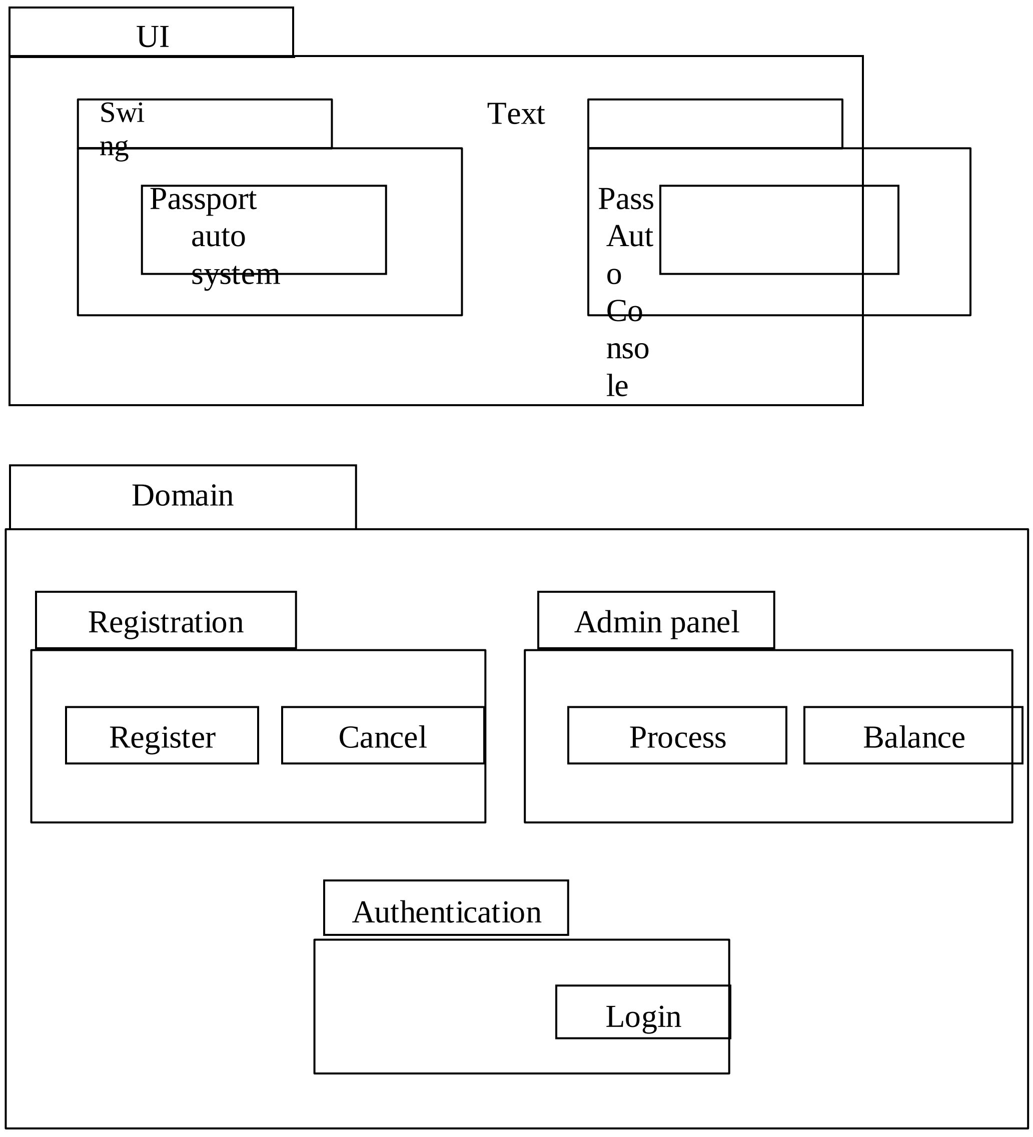
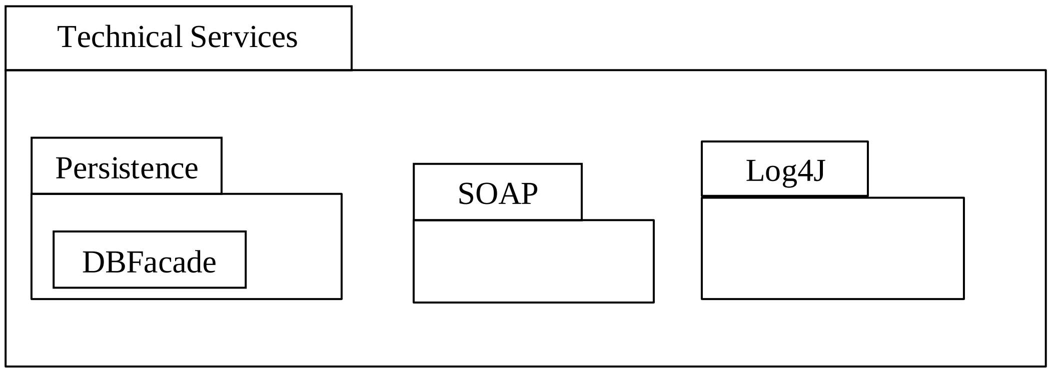


Fig.6.6.COLLABORATION DIAGRAM FOR NEW REGISTRATION

The diagrams show the applicant request the system for registration and the system provide the register form and applicant fill the form and submit and the system give the applicant id. The database stores the full details.

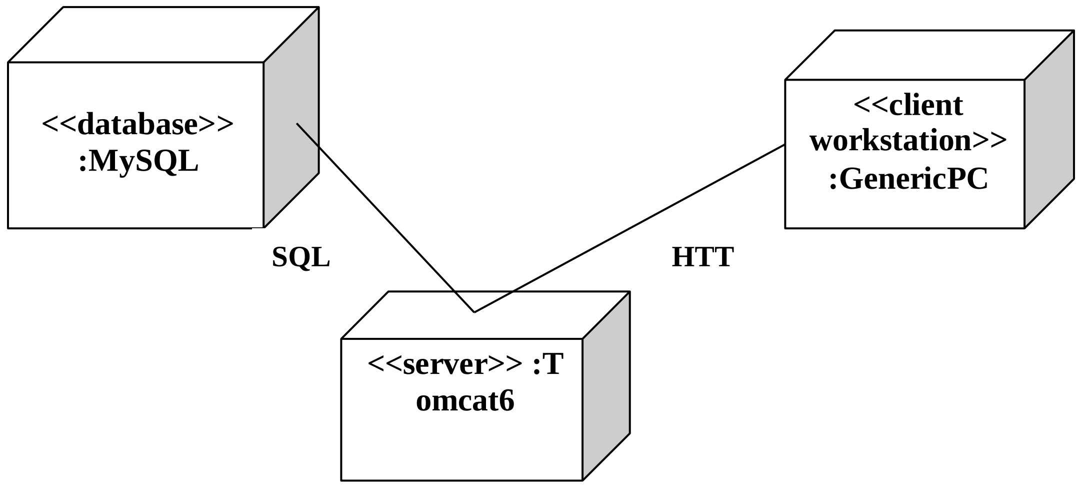
(VII) PARTIAL LAYERD LOGICAL ARCHITECTURE DIAGRAM




Technical Services
Persistence
DBFacade
SOAP
Log4J

(VIII) DEPLOYMENT DIAGRAM AND COMPONENT DIAGRAM

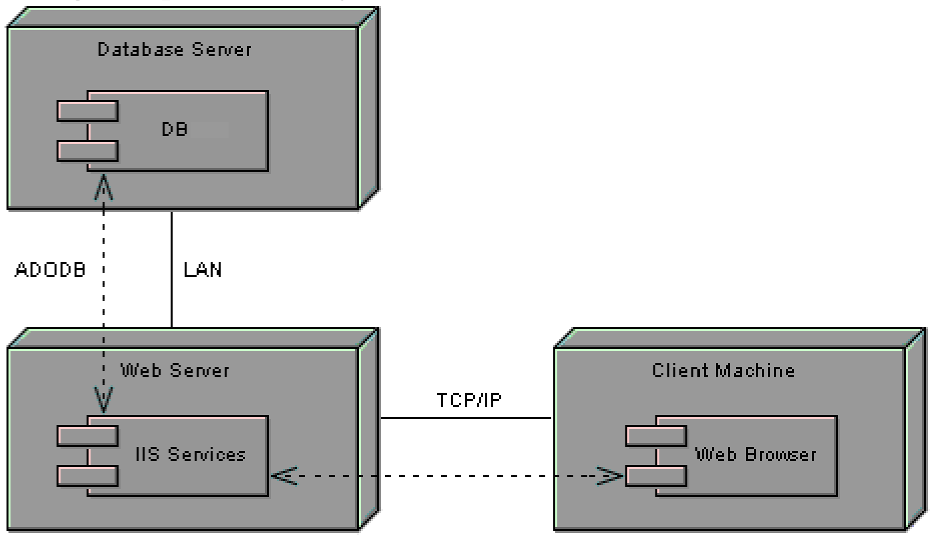
Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.



DEPLOYMENT DIAGRAM

COMPONENT DIAGRAM

Component diagrams are used to visualize the organization and relationship among components in system



RESULT:

Thus the mini project for passport automation system has been successfully executed and codes are generated.

Task 6: Develop test cases for unit testing and integration testing

Unit Testing Test Cases:

Test Case ID	Description	Input Data	Expected Output	
UT1	User Authentication - Invalid Credentials	Username: invalid, Password: invalid	Error message: "Invalid credentials"	
UT2	User Authentication - Valid Credentials	Username: valid, Password: valid	Successful login, user dashboard accessible	
UT3	Password Reset	Username: valid	Email sent with password reset instructions	
UT4	Applicant Information Storage	Valid applicant data	Data stored successfully in the database	
UT5	Special Characters in Applicant Information	Applicant name: John_Doe	System handles special characters correctly	
UT6	Document Verification - Valid Document	Valid document file	Document marked as verified in the system	
UT7	Document Verification - Invalid Document	Corrupted document file	Error message: "Invalid document file"	
UT8	Payment Processing - Fee Calculation	Applicant details for fee calculation	Correct fee calculated and displayed	
UT9	Payment Processing - Transaction Success	Valid payment details	Successful transaction, payment status updated	
UT10	Appointment Scheduling - Time Slot Selection	Applicant selects available time slot	Time slot reserved for the applicant	
UT11	Appointment Scheduling - Conflict Handling	Schedule appointment with an already booked slot	Error message: "Time slot not available"	

Integration Testing Test Cases:

Test Case ID	Description	Input Data	Expected Output	
IT1	User Authentication and Applicant Information	Valid login credentials and applicant details	Successful authentication and access to applicant data	
IT2	Document Verification and Applicant Information	Verified document linked to applicant profile	Document status updated in the applicant information	
IT3	Payment Processing and Appointment Scheduling	Successful payment and appointment selection	Appointment status updated after successful payment	
IT4	End-to-End Flow	User registration, document submission, payment, and appointment scheduling	Successful completion of the entire process	
IT5	Concurrency Testing	Multiple users accessing the system simultaneously	Data integrity maintained under concurrent access	

Task 7: Develop test cases for black box and white box testing

Black Box Testing Test Cases:

Test Case ID	Description	Input Data	Expected Output	
BB1	User Registration	Valid user details	Successful user registration	
BB2	Login Functionality	Valid login credentials	Successful login, access to user dashboard	
BB3	Password Reset	Valid email address	Password reset email sent	
BB4	Document Submission	Valid applicant details and documents	Documents successfully submitted	
BB5	Payment Processing	Valid payment details	Successful payment, transaction completed	
BB6	Appointment Scheduling	Available time slot selected	Appointment scheduled successfully	
BB7	Document Verification	Valid document file	Document marked as verified	
BB8	System Security	Unauthorized access attempt	Access denied, appropriate security message displayed	
BB9	Error Handling	Submitting incomplete form	Error message displayed, guiding user to complete form	
BB10	Application Performance	Simultaneous user access	System response time within acceptable limits	

White Box Testing Test Cases:

Test Case ID	Description	Input Data	Expected Output	
WB1	Code Coverage Analysis	Verify that all code paths are exercised	Achieve high code coverage percentage	
WB2	Unit Testing - User Authentication	Invalid login credentials	Verify appropriate error handling in authentication	
WB3	Unit Testing - Document Verification	Invalid document file	Ensure proper validation and error handling	
WB4	Unit Testing - Payment Processing	Invalid payment details	Verify error handling and transaction failure	
WB5	Unit Testing - Appointment Scheduling	Conflicting appointment schedules	Confirm conflict resolution and error messages	
WB6	Integration Testing - User Authentication	Valid login credentials	Ensure integration of authentication with other modules	

Test Case ID	Description	Input Data	Expected Output	
WB7	Integration Testing - Document Verification	Verified document linked to applicant profile	Validate integration between document and user modules	
WB8	Integration Testing - Payment Processing	Successful payment and appointment scheduling	Verify data consistency across payment and scheduling	
WB9	Security Testing	SQL injection attempt	Ensure system security measures prevent SQL injection	
WB10	Performance Testing	Simulate a large number of simultaneous users	Verify system performance and response time under load	

Task 1: Development of problem statements

1. **Filling the form manually to register in book bank**
   Have to fill the form manually, in order to register in the book bank. It may have any mistake if we register manually.
2. **Due date to submit the book**:
   To know the due date when to return the book in a book bank is difficult to know manually. It can be done by checking each and every person's details of books.
3. **Reserve the book manually is a problem:**
   In order to reserve the book manually is problematic. There is a big work to reserve the books by the candidates. They must not know the books availability to reserve the books.
4. **Fine Money calculation for the books taken by the candidate:**
   The fine money calculation may not be accurate when we calculate manually. The fine money may comes in case of books delay due date or in case of lost of books.
5. **To check books availability manually:**
   To check books availability manually is not easy. It can be done by checking the books each and every shelf. It may be done wrong by checking the books availability manually.
6. **To know the persons who took the book is difficult:**
   In order to find the candidates who took the books manually is difficult. It takes a lot of time and it may not be accurately to know if any of the details missed manually.
7. **Maintaining books details and candidates' details:**
   Maintaining all the details of books and along with the candidate details manually is difficult. In some cases, to retrieve the details of books or candidates is not easy.
8. **To know how many number of books each person took**:
   Manually, it is difficult to know the number of books which is taken by each candidate. Manually it may go wring if we missed any calculation or details maintenance.
9. **In order to renew the books once again in the book bank:**
   Manually, it is difficult to know the renew times and the renew date. Candidates can renew the books number of times and use it. This may become a problem for the demand of books in the book bank.

Task 2: Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents

1. <u>Software Requirement Specification (SRS) Document:</u>

The SRS document outlines the detailed requirements for the Book Bank System. It defines the system's functionalities, constraints, and user expectations. The SRS document typically includes the following sections:

Introduction: Provides an overview of the document, its purpose, and the scope of the Book Bank System.
System Overview: Describes the system's high-level architecture, components, and interactions with external systems.

Functional Requirements: Specifies detailed functionalities of the system. Each requirement should be clear, concise, and unambiguous. Use cases and user stories can be helpful here.

Non-Functional Requirements: Addresses system qualities like performance, scalability, security, usability, and reliability.

User Interfaces: Describes the user interfaces, including mockups or wireframes if available.

Data Requirements: Defines the data structures, databases, and data flow within the system.

System Architecture: Provides an overview of the system's technical architecture, including hardware and software components.

Dependencies: Lists external systems, libraries, or APIs that the system will depend on.

Constraints: Specifies any limitations or restrictions that the system must adhere to.

Assumptions: Documents any assumptions made during requirement gathering.

2. <u>Design Documents:</u>

Design documents elaborate on the technical architecture, system components, and implementation details of the Book Bank System. These documents aid the development process. Common design documents include:

High-Level Design (HLD): Outlines the overall system architecture, component interactions, and major modules.

Low-Level Design (LLD): Provides in-depth details about each module, including data structures, algorithms, and interfaces.

Database Design: Describes the database schema, relationships, and data manipulation methods.

User Interface (UI) Design: Provides detailed information about the user interfaces, including layouts, styles, and user interactions.

API Design: Documents the APIs, endpoints, request-response formats, and authentication mechanisms.

3. Testing Phase Related Documents:

The testing phase requires a set of documents to ensure comprehensive testing and validation of the system:

Test Plan: Outlines the testing approach, test objectives, scope, testing strategies, and resources required for testing.

Test Cases: Provides detailed scenarios, inputs, expected outcomes, and procedures for testing various system functionalities.

Test Scripts/Automation: If applicable, provides automated scripts for executing test cases.

Test Data: Specifies the data needed to perform tests effectively.

Defect Reports: Documents any defects found during testing, including steps to reproduce and severity levels.

Traceability Matrix: Links test cases back to the specific requirements, ensuring all requirements are tested.

Test Summary/Report: Summarizes the testing process, results, and any open issues.

Remember that these documents should be continuously updated throughout the development process to reflect any changes or additions. The documents will serve as valuable references for developers, testers, and stakeholders, ensuring a well-organized and successful development and testing process for the Book Bank System.

Ex. No.: **SOFTWARE REQUIREMENT SPECIFICATION**

Date:

Aim:

To write software requirement specification for book bank management system.

Software Requirement Specification:

1. Introduction:

1.1 Purpose:

The main objective of this document is to illustrate the requirements of the project Book Bank Management System. The document gives the detailed description of both functional and non functional requirements proposed by the client. The document is developed after a number of consultants with the client and considering the complete requirements specifications of the given project. The final product of the team will be meeting the requirements if this document.

1.2 Project Scope:

This system allows the book bank staff to maintain book records. With the help of this system, the staffs should be in a position to tell about the availability of books, books provided. Moreover this system reports about the actions held in book bank. This system makes the work easier.

1.3 Overview:

This system overall provides an easy solution to the staffs in book bank to keep track of members, stock of books and statistics.

2. General Description:

2.1 Product Description:

This book bank management system replaces the traditional, manual book bank management by which lot of paper work will be reduced. This system will provide a search functionality to facilitate the resources. This search will be based on various categories viz book name or the ISBN. Also advanced search features are provided in order to search various categories. These staff should have a computer to view the details of the members. These staff should able to see the number of books that the particular person has taken from the book bank, details of the book, returned or taken. This is the primary feature. Another feature is that the book bank staff is able to edit the details. There should be a person to look after longtime pending of books. E-mail should be sent to members those who failed to return the book for long time.

2.2 Product Features:

There are two different users who will be using this product:

Book bank staff who will be acting as administrator

The features that are available to the staff are:

- Can view the different categories of books available in the book bank.
- Can view the list of books available in each category.
- Can take the book returned from students.
- Add books and their information of the books to the database.

- Edit the information of the existing books in the database.
- Can check the report of the issued book.
- Can access all the accounts of the students `.

The features that are available to the members are:

- Can view the list of books available in each category
- Can view the different categories of books available in the book bank.
- Can own a ID by registering in the book bank system
- Can view the books issued to him/her.
- Can put a request for new book.
- Can view the history of books issued to him previously.
- Can search for particular book.

2.3 Design and implementation constraints:

The product is developed using VB. The backend data base for this SQL server. The product is accomplished with login facility so that specific function is available to specific member of book bank.

2.4 User Documentation:

The product includes user manual. The user manual will include product overview, complete configuration of the used software (such as SQL server), technical details, backup procedure and contact information which will include E-mail address. The database will be created in Microsoft SQL.

3. Functional Requirements:

This action gives the list of functional requirements which are applicable to the book bank management system.

3.1 Description:

Proposed Database is intended to store, retrieve, update and manipulate information related to university which includes

- Books availability
- Student details
- My account

The administrator can login and when the administrator logs into the book bank system, the system will check for validity of login. If the login and the password are valid, the response to this action is the administrator will be able to modify, view, add, delete and all other functions that can be performed on the database.

3.2 Technical Issues:

The system should be implemented using VB.

4. Interface Requirements:

This section describes how the software interfaces with the other software products or users for input or output.

4.1 User Interfaces:

Describes how this product interfaces with the user.

GUI:

Describes the graphical user interface if present. This section should include a set of screen dumps or mockups to illustrate user interface features.

4.1.1 Description:

The user interface must be customized by the administrator.

4.1.2 Criticality:

This issue is essential to the overall system. All the modules provided with the software must fit into the GUI and accomplish to the standard defined.

4.1.3 Technical issues:

In order to satisfy these requirements the design should be simple and the entire different interface should follow a standard template.

4.1.4 Risks:

To reduce the circumstances under which this requirements might not able to be satisfied, all the designers must have been developed websites previously and they must be aware of html restriction and uses browser implementation before starting the designing. In order to reduce the design team will be trained in basic html development and macromedia fireworks. This tool will be used instead of Photoshop.

4.1.5 Dependencies and other requirements:

All user interfaces should be able to interact with the user management module and a part of the interface must be dedicated to the login/login module.

4.2 Hard Interfaces:

A computer for users and another computer to hold all the details of books, members, etc.

4.3 Software interfaces:

Database	:	SQL server
Application	:	VB

5. Other Non functional Requirements:

5.1 Performance Requirements:

This system should work concurrently in multiple processors between the working hours in book bank. The system should support at least 10 users.

5.2 Safety Requirements:

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

5.3 Security Requirements:

The book provider staff must be provided with a password to log on to the system. Only he/she can be able to modify the data, delete, append, etc. All other users other than office staff only have the rights to retrieve the information about database.

5.4 Availability:

The system should be available during working hours.

5.5 Maintainability:

There should be a facility to add or delete or update members and books.

5.6 Reusability:

The same system will be used in every period of time.

6. Design Constraints:

The system should be designed within 6 months

Task 3: Preparation of Software Configuration Management and Risk Management related documents.

Software Configuration Management (SCM) Document:

Introduction:
Brief overview of the document's purpose and scope.

Configuration Management Plan:
Outline the overall approach to configuration management.
Define roles and responsibilities of team members involved in SCM.
Specify the tools and resources to be used for configuration management.

Version Control:
Specify version control practices for source code, documents, and other artifacts.
Describe how branching, merging, and release management will be handled.

Configuration Identification:
Define naming conventions for components, files, and versions.
Explain how to uniquely identify and label each configuration item.

Configuration Control:
Detail the change management process, including how changes are requested, reviewed, and approved.
Describe the change tracking mechanism and how it's integrated with development activities.
Configuration Status Accounting:
Explain how configuration status will be tracked and reported.
Define the format and frequency of status reports.

Configuration Auditing:
Describe the process for conducting configuration audits to ensure compliance with standards and requirements.

Baseline Management:
Define how and when baselines will be established for different project phases.
Explain the process for managing changes after baselines are established.

Backup and Recovery:
Detail the backup and recovery strategy for configuration items and project data.

Release Management:

Describe the process for packaging, distributing, and deploying releases to different environments.

Risk Management Document:

Introduction:
Briefly explain the purpose and importance of risk management in the project.

Risk Management Plan:
Outline the overall approach to risk management throughout the project lifecycle.
Define roles and responsibilities of team members involved in risk management.

Risk Identification:
List and categorize potential risks related to the Book Bank Automation System project. Include risks related to technical, organizational, and external factors.

Risk Assessment:
Evaluate and prioritize identified risks based on their impact and likelihood.
Create a risk matrix to visualize the severity of each risk.

Risk Mitigation Strategies:
Define strategies to mitigate high-priority risks.
Specify preventive and corrective actions for each risk.

Risk Monitoring and Control:
Explain how risks will be monitored throughout the project.
Describe the process for tracking risk triggers and implementing mitigation plans.

Risk Communication:
Define how risks and mitigation plans will be communicated to stakeholders.
Include a communication plan for addressing risks in a transparent manner.

Contingency Planning:
Detail contingency plans for handling risks that cannot be fully mitigated.

Lessons Learned:
Discuss how lessons learned from risk events will be documented and shared for future projects. Remember, both SCM and Risk Management documents should be tailored to your project's specific needs, and they should be reviewed and updated as the project progresses to ensure their continued relevance and effectiveness.

Task 4: Study and usage of any Design phase CASE tool

StarUML - Open Source Design Phase Case Tool

StarUML is an open source **Computer-aided software engineering (CASE)** tool that supports the UML (Unified Modeling Language) framework for system and software modeling. It is based on UML version 1.4, provides eleven different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages.

Website: https://staruml.io/

Version Tested: StarUML version 6.0.1, tested on Windows

Minimum System Requirements: Windows 2000, Windows XP, or higher; Microsoft Internet Explorer 5.0 or higher; 128 MB RAM (256MB recommended); 110 MB hard disc space (150MB space recommended)

License & Pricing: Open Source

Support: User mailing list

Installation

The installer follows the classic Windows install procedure without issues.

Documentation

The same help that could be browsed on the StarUML web site is available with the tool on your desktop. Documentation describes the concepts of tool but on high level vision. A more detailed documentation is available for the diagramming functions. Sample projects are provided with the tool and one of them contains the model of the tool itself, showing that the developers were able to eat their own dog food. Besides English, documentation exists in Korean, Japanese and Russian.

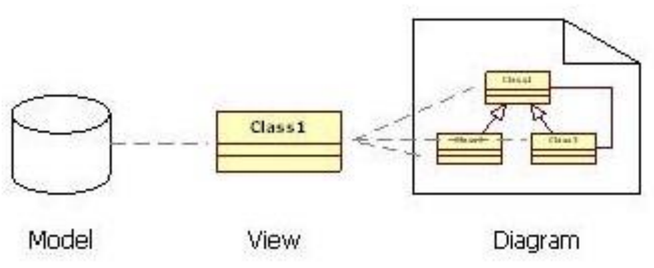
Configuration

Some general and diagram configurations options are available from the Tools/Option menu. You will find in this window also the configuration switches for the code generation. The interface is also very configurable as you can select what part of the tool you would like to view or not.

Features

When you start a new project, StarUML proposes which approach you want to use: 4+1 (Krutchen), Rational, UML components (from Cheesman and Daniels book), default or empty. Depending on the approach, profiles and/or frameworks may be included and loaded. If you don't follow a specific approach, the "empty" choice could be used. Although a project can be managed as one file, it may be convenient to divide it into many units and manage them separately if many developers are working on it together.

StarUML makes a clear conceptual distinction between models, views and diagrams. A Model is an element that contains information for a software model. A View is a visual expression of the information contained in a model, and a Diagram is a collection of view elements that represent the user's specific design thoughts.



StarUML supports the following diagram types

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- Statechart Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram
- Composite Structure Diagram

Task 5: Performing the Design by using any Design phase CASE tools

Ex no:2

BOOK BANK SYSTEM

Date:

AIM:

To create a system to perform book bank operation

(I) PROBLEM STATEMENT:

A Book Bank lends books and magazines to member, who is registered in the system. Also it handles the purchase of new titles for the Book Bank. Popular titles are brought into multiple copies. Old books and magazines are removed when they are out or date or poor in condition. A member can reserve a book or magazine that is not currently available in the book bank, so that when it is returned or purchased by the book bank, that person is notified. The book bank can easily create, replace and delete information about the tiles, members, loans and reservations from the system.

(II) SOFTWARE REQUIREMENTS SPECIFICATION:

2.1SOFTWARE INTERFACE

- **Front End Client -** The Student and Librarian online interface is built using JSP and HTML. The Librarians local interface is built using Java.
- **Web Server -** Glassfish application server (Oracle Corporation).
- **Back End -** Oracle database

2.2HARDWARE INTERFACE

The server is directly connected to the client systems. The client systems have access to the database in the server.

(III)USE-CASE DIAGRAM:

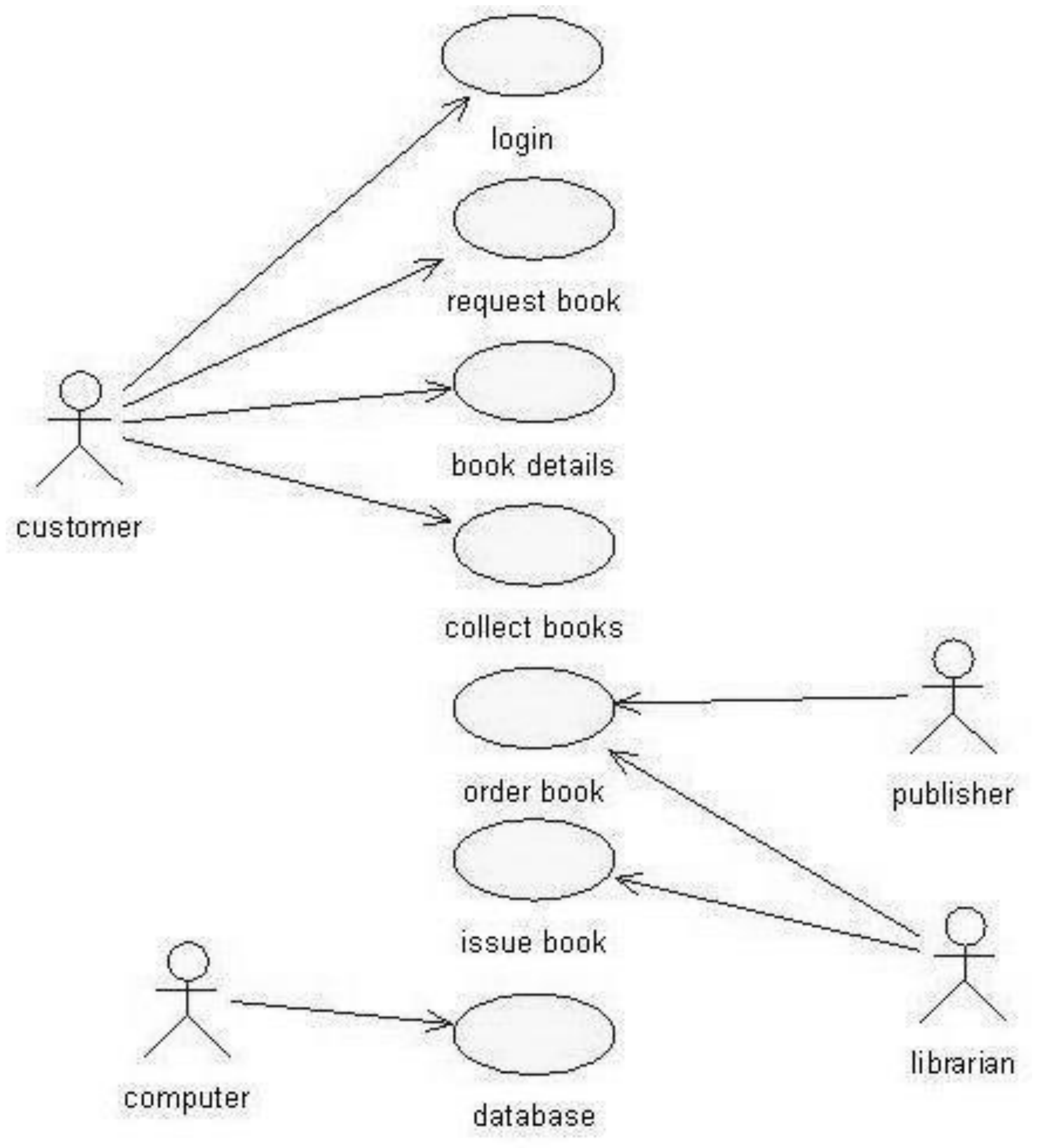


Fig 3. USE-CASE DIAGRAM FOR BOOK BANK SYSTEM

(IV) ACTIVITY DIAGRAM:

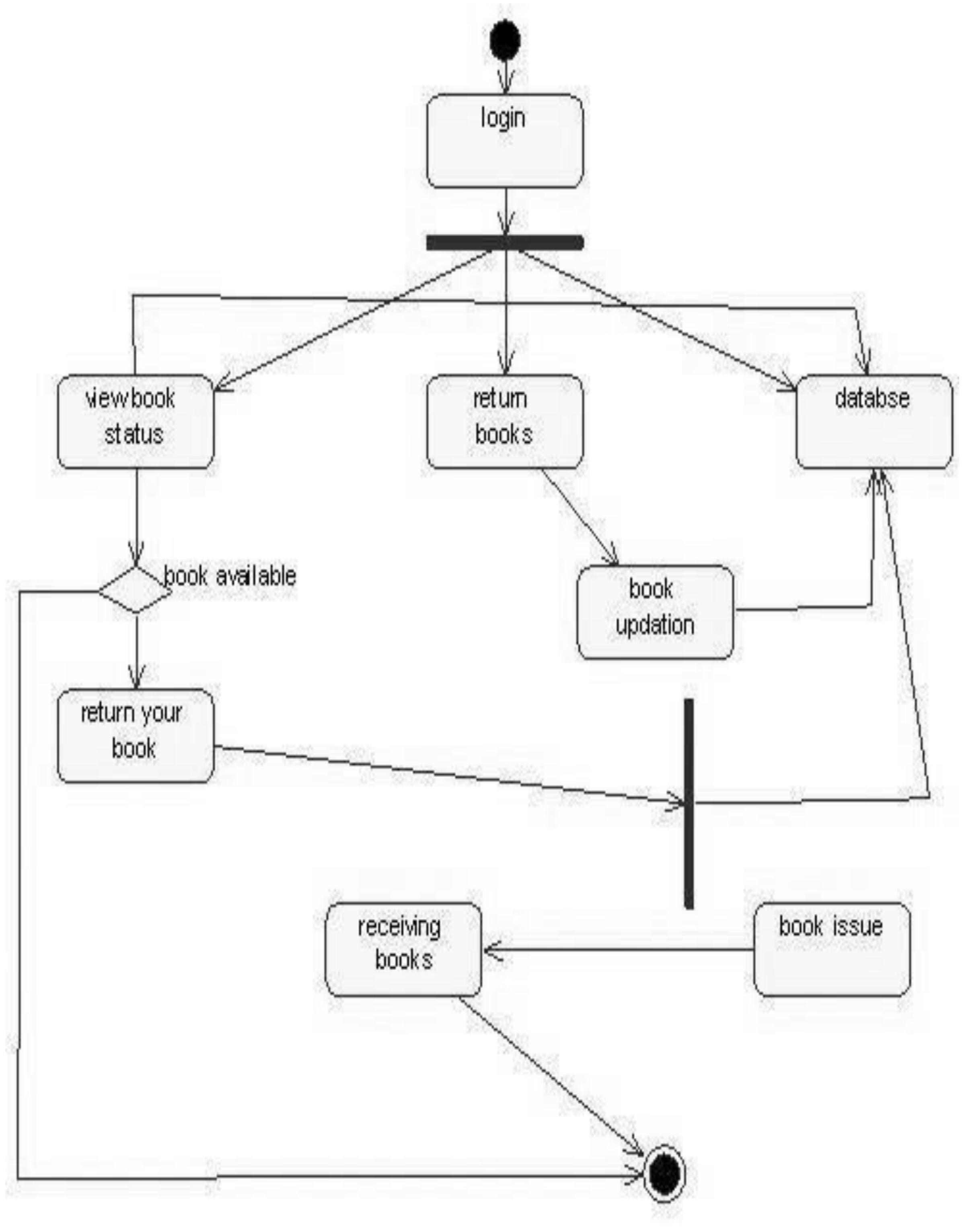


Fig.4. ACTIVITY DIAGRAM

(V) CLASS DIAGRAM:

The class diagram, also referred to as object modeling is the main static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.

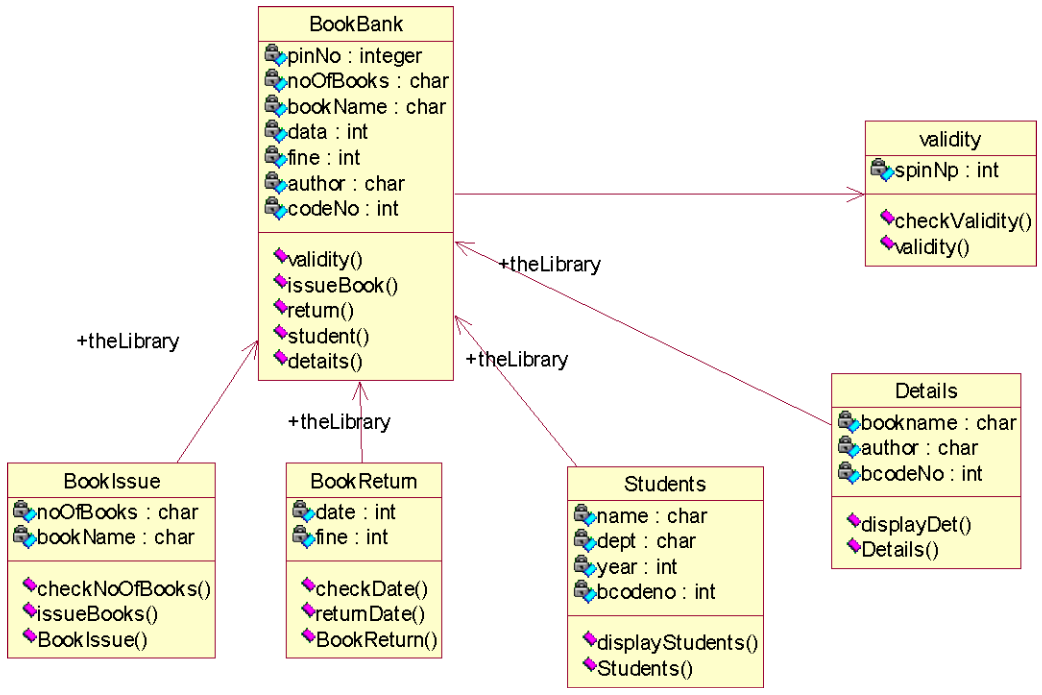


Fig.5. CLASS DIAGRAM FOR BOOK BANK SYSTEM

(VI)) SEQUENCE DIAGRAM:

A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the “form” object is requesting an operation be performed by the "to" object. The "to" object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

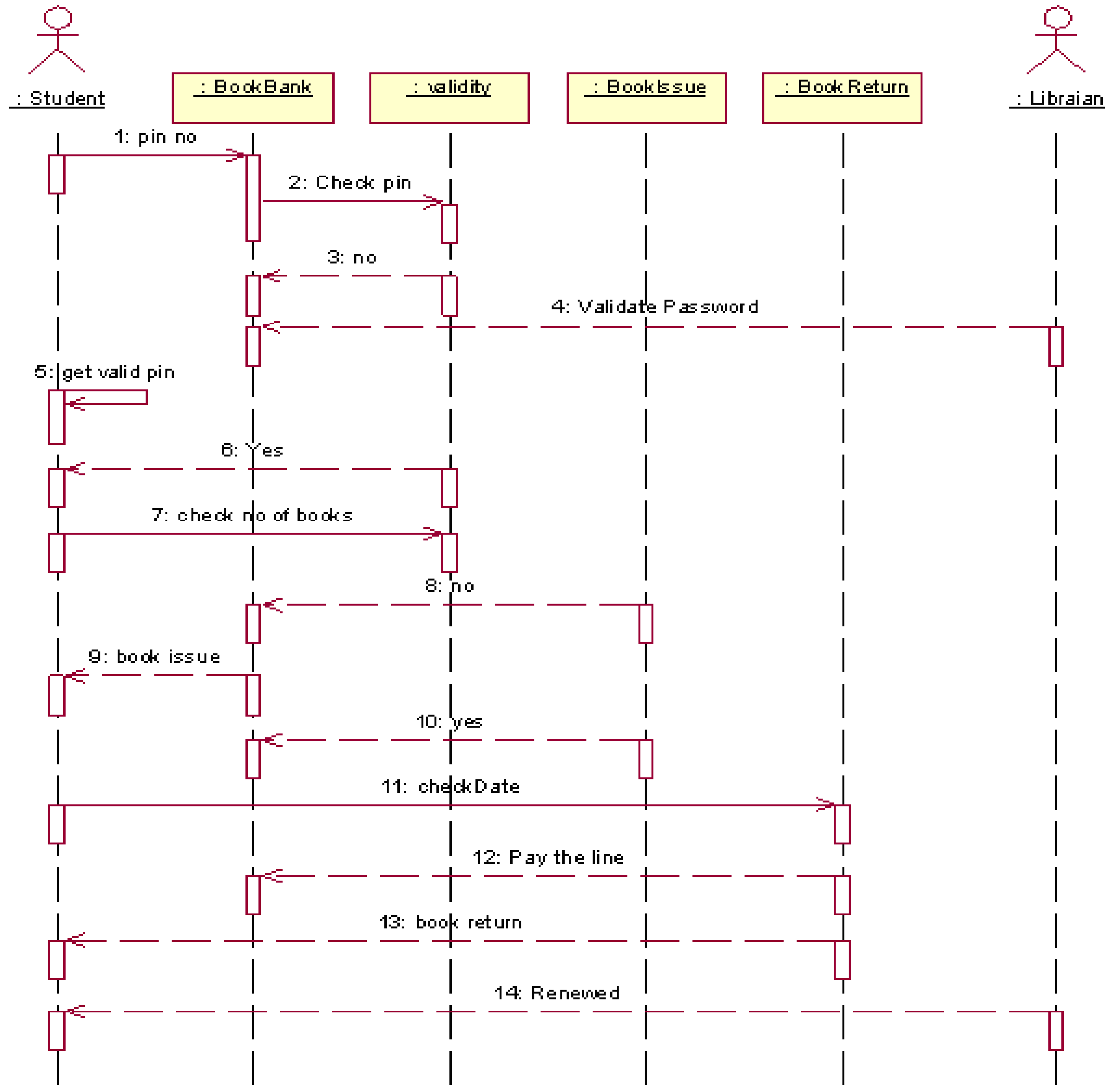


Fig. 6.1. SEQUENCE DIAGRAM FOR DEPOSIT PROCESS

The diagrams show the pin no is entered and check the pin .Get no and validate password check the condition based on condition book issue and return are done. Pay the online and renewed.

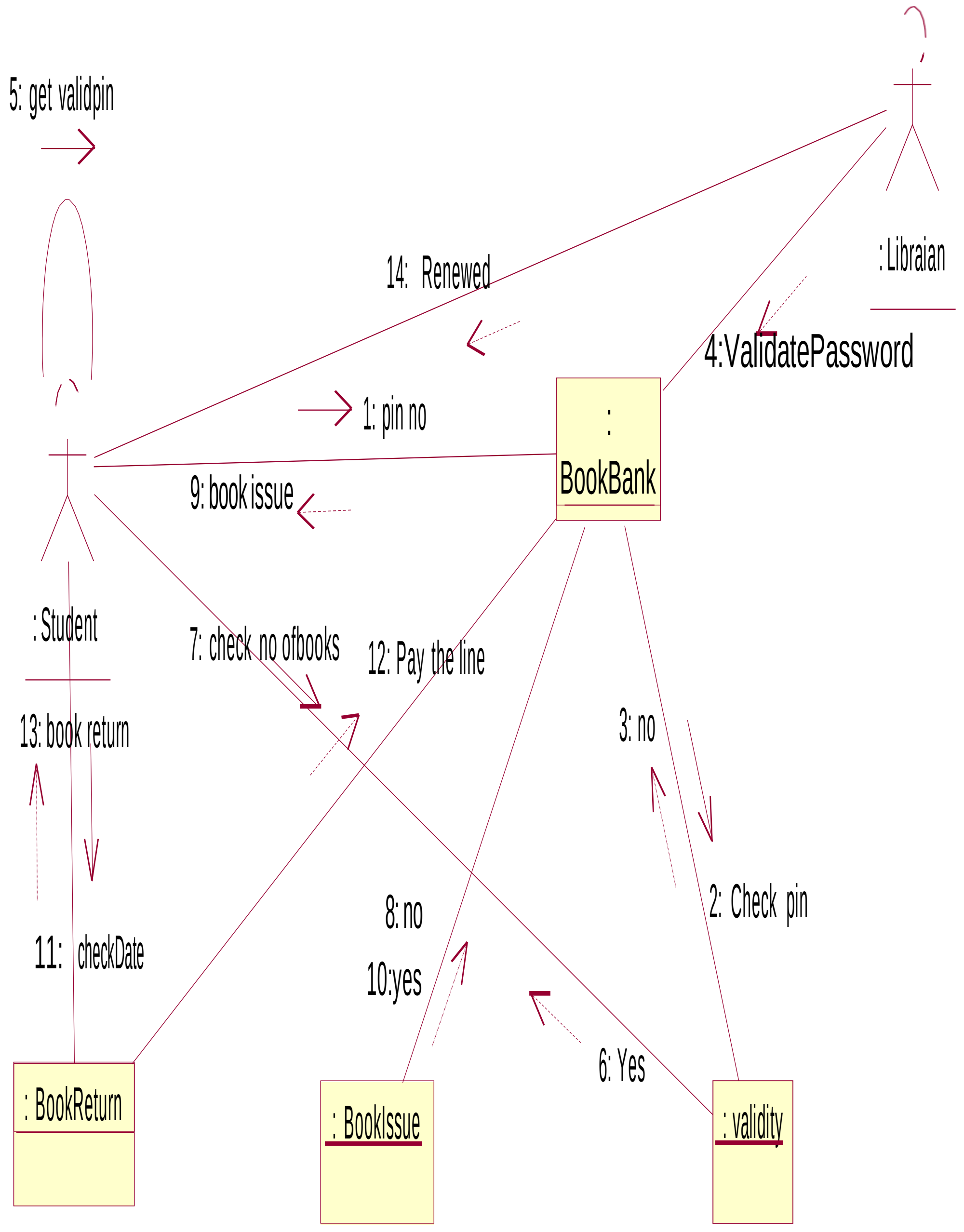
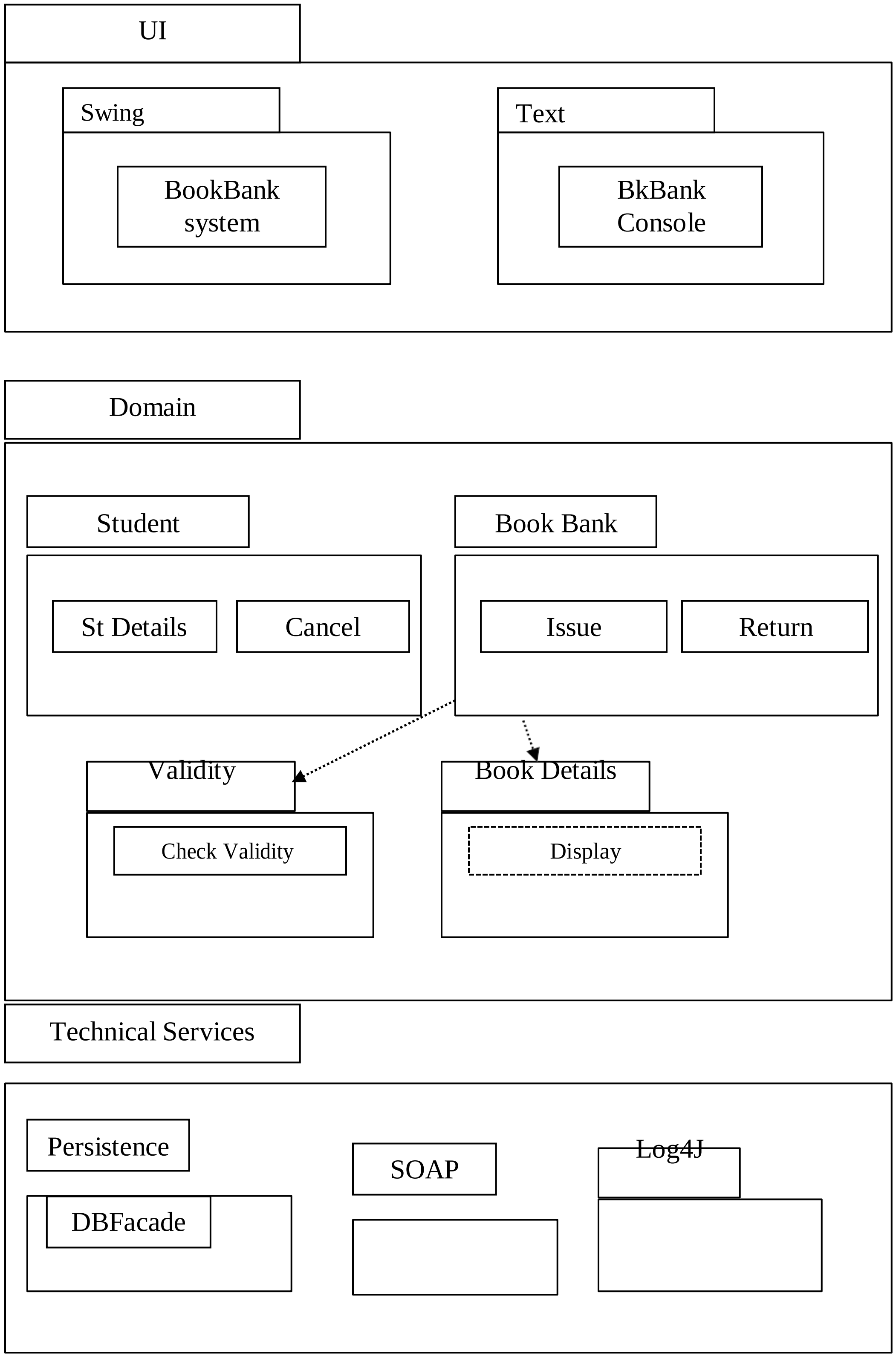


Fig. 6.2. COLLABORATION DIAGRAM FOR DEPOSIT PROCESS

(VII) PARTIAL LAYERD LOGICAL ARCHITECTURE DIAGRAM:



(VIII) DEPLOYMENT DIAGRAM AND COMPONENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

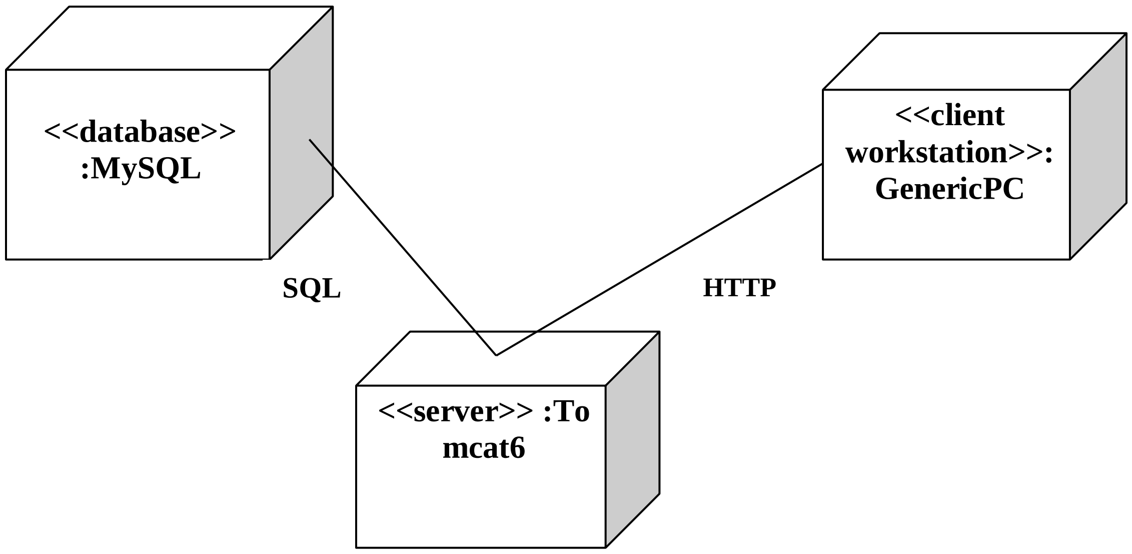


Fig.8.1.DEPLOYMENT DIAGRAM

COMPONENT DIAGRAM

Component diagrams are used to visualize the organization and relationships

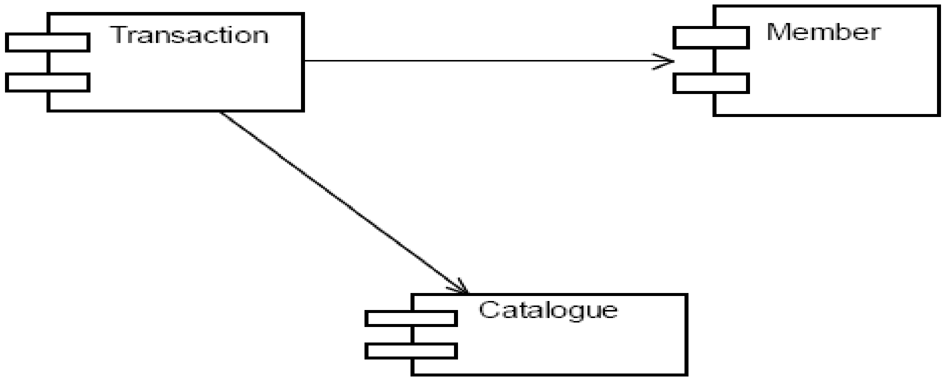


Fig.8.2.COMPONENT DIAGRAM

RESULT:

Thus the mini project for Book Bank System has been successfully executed and codes are generated.

Task 6: Develop test cases for unit testing and integration testing

Unit Testing Test Cases:

Test Case ID	Description	Input	Expected Output
UT001	Verify that a new book is added successfully	Book details: Title, Author, ISBN	Success message
UT002	Check if a book is deleted successfully	Book ID	Success message
UT003	Test if a book record is updated correctly	Book ID, Updated details	Success message
UT004	Ensure that searching for a book by title returns the correct result	Title	Matching book details
UT005	Validate that borrowing a book decreases its available quantity	Book ID, User ID	Success message
UT006	Verify that returning a book increases its available quantity	Book ID, User ID	Success message
UT007	Test if overdue books are correctly flagged	User ID, Due date	Overdue status

Integration Testing Test Cases:

Test Case ID	Description	Input	Expected Output	
IT001	Ensure that the user can borrow a book from the book bank	User ID, Book ID	Success message, Updated book availability	-
IT002	Check if the user can return a borrowed book	User ID, Book ID	Success message, Updated book availability	-
IT003	Verify that the system updates the user's record with borrowed books	User ID, Borrowed book details	Updated user record	-
IT004	Test if the system correctly updates the book's availability after a return	User ID, Returned book details	Updated book availability	-
IT005	Ensure that overdue books are handled correctly in the system	User ID, Overdue book details	Overdue status, Fine calculation	-
IT006	Validate that book search functionality integrates with the book database	Search query	Matching book details	-
IT007	Test if the system generates reports on borrowed books and fines	Report parameters	Report with accurate information	

Task 7: Develop test cases for black box and white box testing

Black Box Testing:

Test Case ID	Description	Input	Expected Output
BBTC01	Verify login functionality	Valid username and password	Successful login
BBTC02	Verify login functionality	Invalid username or password	Error message or unsuccessful login
BBTC03	Search for a book by title	Valid book title	Display relevant book details
BBTC04	Search for a book by title	Invalid book title	No results found message
BBTC05	Check book availability	Available book	Show book as available
BBTC06	Check book availability	Checked out book	Display checked out status
BBTC07	Add a book to the cart	Valid book ID	Book added to the cart
BBTC08	Add a book to the cart	Invalid book ID	Error message or unsuccessful addition
BBTC09	Check out a book	Book in the cart	Successful check-out
BBTC10	Check out a book	Empty cart	Error message or unsuccessful check-out
BBTC11	View transaction history	Valid user ID	Display list of previous transactions
BBTC12	View transaction history	Invalid user ID	Error message or no transactions found

White Box Testing:

Test Case ID	Description	Input	Expected Output
WBTC01	Verify database connection	N/A	Successful connection
WBTC02	Check for input validation	Invalid characters in book title	Proper error handling
WBTC03	Check for input validation	Negative book quantity	Proper error handling
WBTC04	Verify the integrity of data storage	Add a book to the database	Confirm data is stored correctly
WBTC05	Verify data retrieval	Retrieve book details by ID	Correct book details are fetched

Test Case ID	Description	Input	Expected Output
WBTC06	Verify business logic for book availability	Check availability of a checked-out book	Book status shows as unavailable
WBTC07	Test the cart functionality	Add and remove books from the cart	Cart is updated correctly
WBTC08	Verify the check-out process	Check out a book and update the database	Book status changes to checked out
WBTC09	Test session management	Log in and log out	Session state changes appropriately
WBTC10	Verify transaction history update	Complete a transaction	Transaction history is updated