A LAB MANUAL

on

SOFTWARE ENGINEERING LAB (U23CD5L1)

B.E. III Year I Semester

Dept. of CSE – Data Science

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY
(UGC Autonomous)
Approved by AICTE | Affiliated to Osmania University | Estd.2003
Accredited by **NBA**, Accredited '**A**' Grade by NAAC
(Academic Year: 2025– 2026)

PREPARED BY

Mr. Md Anjar Ahsan
Mr. Asif Ahmed
Mrs.Y.Shakuntala

INDEX

Vision of the Department:

To develop the world's next generation Data Scientist by offering top-notch education with cutting edge technologies.

Mission of the Department:

DM1: Providing students to understand the principles of Data Science with hands-on experience.

DM2: Offering students to analyze the data through effective teaching learning methods and cutting edge technologies in multi disciplinary fields.

DM3: Preparing students for R&D, Industrial design, entrepreneurship and employment.

DM4: Encouraging students with interaction and Industry Institute partnership through various organizations.

Note: DM: Department Mission

Head of the Department

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

B.E. Computer Science and Engineering (Data Science) Program Educational Objectives (PEOs):

PEO1	Students will have strong foundation In Basic Science, Mathematics, Statistics, Computer Science and Allied engineering.
PEO2	Students will develop Data Science applications at advanced level for employability in IT industry especially in niche technologies.
PEO3	Students will be provided with strong foundation in Data Science and its applications for their carrier as Data Scientist/Data Engineers.

Head of the Department

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

Approved by AICTE | Affiliated to Osmania University | Estd.2003.

Department of Computer Science and Engineering (Data Science)

B.E. Computer Science and Engineering (Data Science) Program Outcomes (POs):

Engineering Graduates will be able to:

S. No.	Program Outcomes (POs):
1.	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2.	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3.	**Design/Development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4.	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5.	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6.	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7.	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8.	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9.	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10.	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11.	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12.	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

Head of the Department

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

Approved by AICTE | Affiliated to Osmania University | Estd.2003.

Department of Computer Science and Engineering (Data Science)

B.E. Computer Science and Engineering (Data Science) Program Specific Outcomes (PSO's):

PSO1	**Professional Skills:** The ability to apply knowledge of Data Science in real-time software project development using open-source and commercial programming environment to deliver quality software product for the organization's success.
PSO2	**Problem-Solving Skills:** The ability to acquaint with the current trends in industrial/research areas and there by innovate through novel solutions to the existing problems.

Head of the Department

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

Course Outcomes: U23CD5L1 Software Engineering Lab

Student will able to

CO. No.	**Description**	**Bloom's Taxonomy Level**
C56.1	Recall the basic principles of software Engineering and UML modeling.	**BTL1**
C56.2	Interpret and explain use-case diagram, class diagram and activity diagram for software requirements.	**BTL3**
C56.3	Construct various UML diagram (Use Case, Class, Sequesnce,Collaboration,activity,state,component,deployment)	**BTL4**
C56.4	Analyze software requirement to select appropriate modeling techniques and tools.	**BTL5**
C56.5	Evaluate the consistency and correctness of UML diagrams with respect to given requirements.	**BTL6**

Note: Bloom's Taxonomy Levels

BTL1-Remember	**BTL2-Understand**	**BTL3 - Apply**
BTL4-Analyze	**BT5 - Evaluate**	**BTL6 - Create**

Course Articulation Matrix:

Mapping of Course Outcomes (CO) with Program Outcomes (PO) and Program Specific Outcomes (PSO’s):

Course Title: Software Engineering Lab	**Course code:** U23CD5L1
Sem: V	**Academic Year:** 2025-26
Name of the Faculty: Mr. Md Anjar Ahsan	

Course Outcomes (CO)	**Program Outcomes (PO)**												**Program Specific Outcomes (PSO’s)**
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO 1**	**PSO 2**
C56.1	3	2	-	-	-	-	-	-	1	-	-	3	2
C56.2	2	3	2	-	-	-	-	-	2	2	1	3	2
C56.3	2	2	3	-	3	-	-	-	1	1	1	3	3
C56.4	-	3	2	3	3	-	-	-	2	2	2	3	3
C56.5	2	3	2	3	2	-	-	-	2	1	2	3	2
Avg	2.25	2.60	2.25	3.00	2.33	-	-	-	1.40	1.50	1.50	3.00	2.40

- **Level 1**: Low correlation
- **Level 2**: Medium correlation
- **Level 3**: High correlation

POs Defined:

- **PO1**: Engineering knowledge
- **PO3**: Design/Development of solutions
- **PO5**: Modern tool usage
- **PO7**: Environment and sustainability
- **PO9**: Individual and teamwork
- **PO11**: Project management and finance
- **PO2**: Problem analysis
- **PO4**: Investigation of complex problems
- **PO6**: The engineer and society
- **PO8**: Ethics
- **PO10**: Communication
- **PO12**: Life-long learning

PSOs Defined:

- **PSO1**: Professional Skills
- **PSO2**: Problem-Solving Skills

Signature of the Faculty

Course Code	Course Title						Core/Elective
U23CD5L1	**ARTIFICIAL INTELLIGENCE LAB**						**CORE**
Prerequisite	**Contact Hours Per Week**				**CIE**	**SEE**	**Credits**
OS	**L**	**T**	**D**	**P**			
	-	**-**	**-**	**3**	**25**	**50**	**1.5**

Course Objectives

Develop ability to

1.To identify Project Scope, Objectives and infrastructure
2. To understand software engineering methodologies for project development.
3. To capture the requirements specification for an intended software system
4. To draw the UML diagrams for the given specification 5. To map the design properly to code

Course Outcomes

At the end of the course, student would be able to

1. Recall the basic principles of Software Engineering and UML modeling.
2. Interpret and explain use-case diagrams, class diagrams, and activity diagrams for software requirements.
3. Construct various UML diagrams (use case, class, sequence, collaboration, activity, state, component, deployment).
4. Analyze software requirements to select appropriate modeling techniques and tools.
5. Evaluate the consistency and correctness of UML diagrams with respect to given requirements.

List of Experiments

1. Understanding UML diagrams
2. Understanding the software to implement UML diagrams
3. Design the Use Case Diagram highlighting student and librarian interactions.
4. Draw the Class and Object Diagrams for the system structure and a sample book issue transaction.
5. Create the Sequence Diagram and Collaboration Diagram for the book issue scenario.
6. Create State chart and Activity Diagrams showing workflows for ATM transaction and LMS book issue process.
7. Develop the Component and Deployment Diagram indicating physical deployment on hardware infrastructure.
8. Design an UML diagrams for Exam Registration Process
9. Design an UML diagrams for credit card Processing

Text Books:

1. Software Engineering: A Practitioner's Approach | 9th Edition
2. The Software Engineer's Guidebook: Navigating Senior, Tech Lead, and Staff Engineer Positions at Tech Companies and Startups (Greyscale Indian Edition)

Suggested Readings:

1. Software Engineering with UML
2.Object Oriented Software Engineering: Using Uml Patterns And Java, 3rd Edn

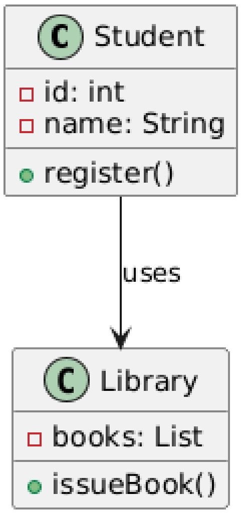
Experiment No. 01

Q. Understanding UML diagrams

Aim:
To understand the different types of UML diagrams and their classifications.
Theory:
UML diagrams are classified into structural and behavioral diagrams. Structural diagrams model the static aspects of the system while behavioral diagrams model the dynamic behavior.

Procedure:
1. List the 14 UML diagrams.
2. Categorize them into structural and behavioral.
3. Use PlantUML to draw at least one of each.

Diagram:-



This is a **UML (Unified Modeling Language) class diagram** showing the relationship between two classes: Student and Library.

Class: Student

- **Attributes:**
  - id: int — An integer representing the student's ID.
  - name: String — A string representing the student's name.
- **Method:**
  - register() — A method that likely handles student registration.

Class: Library

- **Attributes:**

  - books: List — A list containing the books available in the library.
- **Method:**
  - issueBook() — A method that probably issues a book to a student.

Relationship:

- **Dependency ("uses" relationship):**
  - The Student class **uses** the Library class.
  - This means a Student object can interact with or call methods from the Library class, such as issueBook().

Experiment No. 02

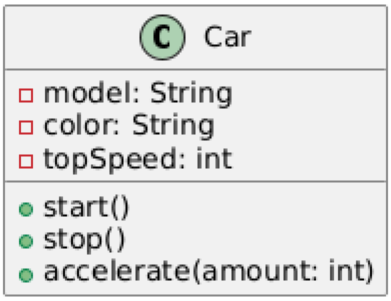
Q. *Understanding the software to implement UML diagrams.*

Aim:
To get hands-on experience with UML tools (PlantUML preferred).

Theory:
PlantUML allows text-based modeling that integrates easily with version control and automation.

Procedure:
1. Visit https://plantuml.com or open VS Code with PlantUML.
2. Create a `car.puml` file.
3. Define class and generate diagram.

Diagram



This is a **UML class diagram** representing a class named **Car**.

Class: Car

Attributes (Properties):

- model: String — Stores the model of the car (e.g., "Toyota Corolla").
- color: String — Stores the color of the car (e.g., "Red").
- topSpeed: int — Stores the top speed of the car in integer format (e.g., 220 km/h).

Methods (Functions/Behaviors):

- start() — A method to start the car.
- stop() — A method to stop the car.
- accelerate(amount: int) — A method to accelerate the car by a specified amount (in integer value)

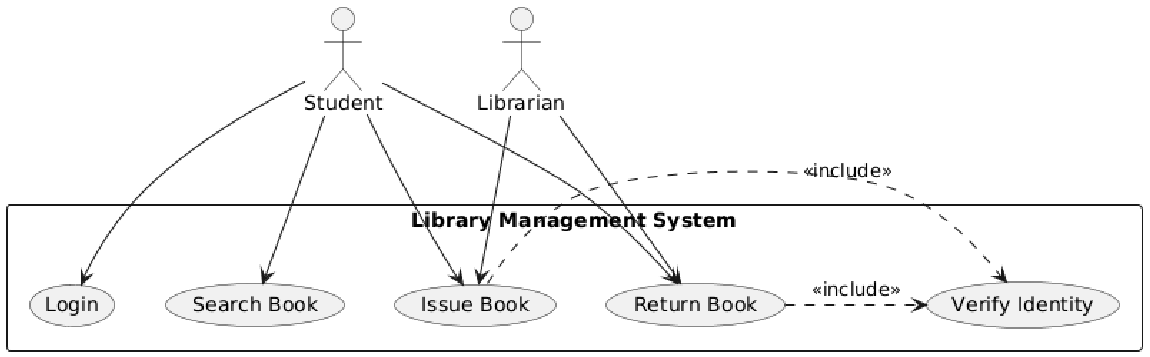
Experiment No. 03:

Q. Design the Use Case Diagram highlighting student and librarian interactions.

Aim:
To model interactions in a Library Management System using Use Case Diagram.
Theory:
Use Case diagrams capture functional requirements by showing interactions between actors and the system.

Procedure:
1. Identify actors: Student, Librarian
2. Identify use cases: Login, Search Book, Issue Book, Return Book
3. Use <<include>> and <<extend>> where appropriate.

Diagram :-



This is a **Use Case Diagram** for a **Library Management System**. It visually shows how different **actors** (users) interact with the system through various **use cases** (functions or services).

□ Actors:

1. **Student** — A user who interacts with the system to search, issue, or return books.
2. **Librarian** — A user who manages book transactions and also verifies identities.

□ Use Cases (Ellipses):

1. **Login** — Allows a user to log in.
2. **Search Book** — Lets students search for books in the system.
3. **Issue Book** — Students or librarians can issue a book.
4. **Return Book** — Handles the return of books.
5. **Verify Identity** — A supporting function to confirm user identity.

🔄 Relationships:

- **Solid lines** (→): Direct interaction between actor and use case.
- **«include» relationships** (dashed arrows with label):
  - Issue Book **includes** Verify Identity
  - Return Book **includes** Verify Identity

This means that verifying a user's identity is a necessary step whenever someone tries to issue or return a book.

Experiment No. 04

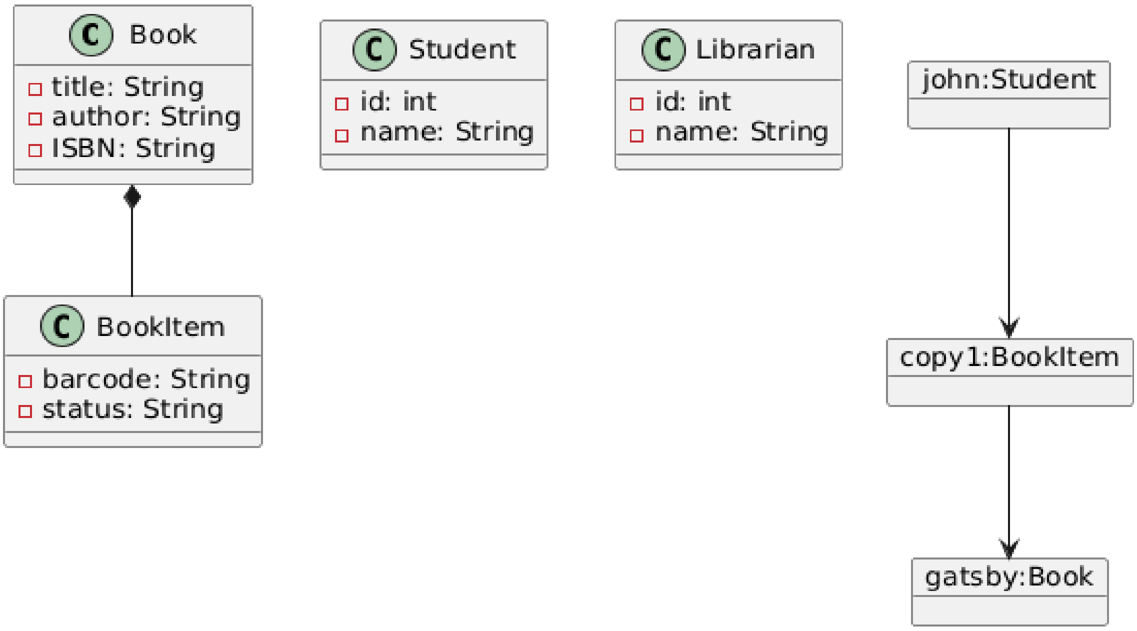
Q. Draw the Class and Object Diagrams for the system structure and a sample book issue transaction.

Aim:
To model the structure of a Library Management System using Class and Object Diagrams.

Theory:
Class Diagrams show static structure including attributes, methods, and relationships. Object Diagrams show instances at a given time.

Procedure:
1. Identify classes: Book, Student, Librarian.
2. Define relationships like composition and inheritance.
3. Create an object diagram for a book being issued.

Sample PlantUML Code:



This is a **Class and Object Diagram** for a **Library Management System** showing how classes relate and how real-world objects (instances) are connected.

Class Diagram (Top Section)

It defines the **structure** of entities in the system:

1. Book

- title: String — Title of the book.

- author: String — Author's name.
- ISBN: String — Unique book identifier.

2. BookItem

- barcode: String — Unique barcode for a physical copy.
- status: String — Current status (e.g., Available, Issued).

▼ Relationship:

- A **BookItem** is a **specific physical copy** of a Book.
  (Denoted by the black diamond — a **composition** relationship: a book can have many book items, and a BookItem cannot exist without a Book.)

3. Student

- id: int
- name: String

4. Librarian

- id: int
- name: String

🔍 Object Diagram (Bottom Section)

It shows specific **instances** (real-world examples) of the classes:

- john:Student

An object (instance) of class Student.

- copy1:BookItem

An instance of BookItem.

- gatsby:Book

An instance of Book.

Relationships between instances:

- john has borrowed copy1, which is a copy of gatsby.
- It visually represents the **borrowing relationship** — a student is linked to a book item, and that item is linked to a specific book.

Department of Computer Science and Engineering (Data Science)

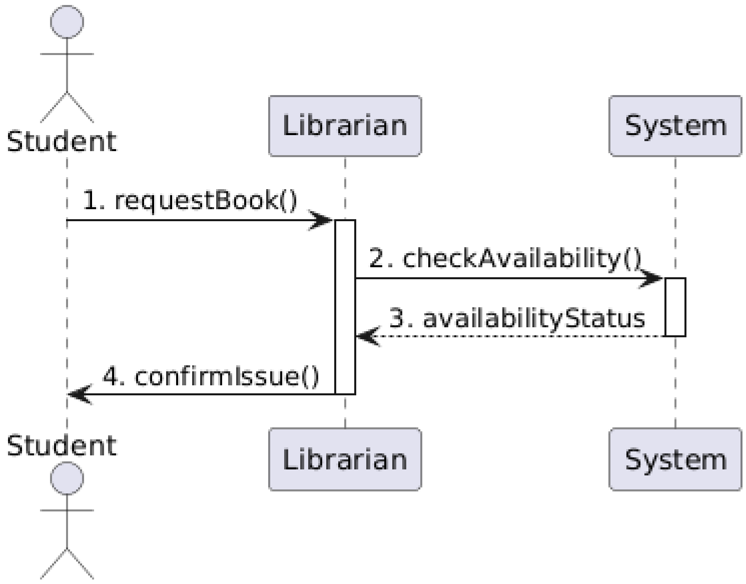
Experiment No. 05

Q. Create the Sequence Diagram and Collaboration Diagram for the book issue scenario.

Aim:
To represent object interactions during a book issue process.
Theory:
Sequence Diagrams show the time-ordered flow of messages. Communication Diagrams highlight object relationships and message sequence.

Procedure:
1. Identify participants: Student, Librarian, System.
2. Show method calls and responses using arrows and lifelines.
3. Use numbered messages in communication diagram.

Diagram :-



This is a **UML Sequence Diagram** that models the interaction between a **Student**, a **Librarian**, and a **System** for issuing a book in a **Library Management System**.

□♂□ Actors/Objects Involved:

1. **Student** – Initiates the request to borrow a book.
2. **Librarian** – Facilitates the transaction.
3. **System** – Checks the availability of the book.

🔄 Flow of Messages (Steps):

1. **requestBook()**
   - The **Student** sends a request to the **Librarian** to issue a book.
2. **checkAvailability()**
   - The **Librarian** queries the **System** to check if the requested book is available.
3. **availabilityStatus**
   - The **System** responds to the **Librarian** with the book's availability status (e.g., Available / Not Available).
   - This is shown as a **dashed line**, indicating a **return message**.
4. **confirmIssue()**
   - The **Librarian** confirms to the **Student** whether the book can be issued or not based on the availability.

Experiment No. 06

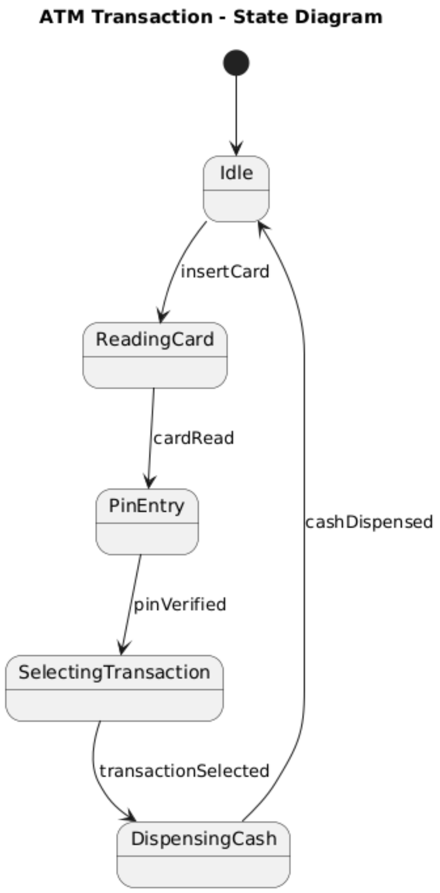
Q. Create State chart and Activity Diagrams showing workflows for ATM transaction and LMS book issue process.

Aim:
To model the workflows and states using Activity and State Chart diagrams.

Theory:
State diagrams model the lifecycle of an object. Activity diagrams show the control flow from one activity to another.

Procedure:
1. Define states for ATM (Idle, ReadingCard, PinEntry, etc).
2. For LMS, define steps in book issue process.
3. Use decisions, fork/join where needed.

Diagram :-



Activity Diagram – Library Book Issue Flow

This is a **State Diagram** for an **ATM Transaction**, illustrating how the ATM transitions between different states during a user session.

□ Start State

- The black dot at the top marks the **initial state** of the ATM.

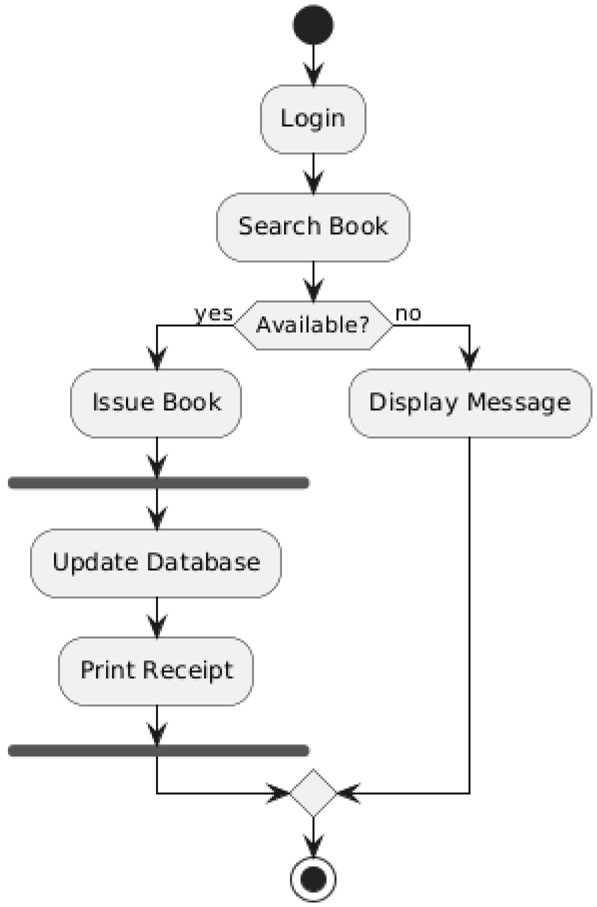
□ States & Transitions:

1. **Idle**
   - Initial resting state of the ATM.
   - **Event:** insertCard → transitions to ReadingCard.
2. **ReadingCard**
   - ATM reads data from the inserted card.
   - **Event:** cardRead → transitions to PinEntry.
3. **PinEntry**
   - User is prompted to enter their PIN.
   - **Event:** pinVerified → transitions to SelectingTransaction.
4. **SelectingTransaction**
   - User selects the type of transaction (e.g., withdraw cash).
   - **Event:** transactionSelected → transitions to DispensingCash.
5. **DispensingCash**
   - ATM dispenses the cash to the user.
   - **Event:** cashDispensed → returns to Idle state.

🔁 Cycle Complete

- After dispensing cash, the ATM goes back to the **Idle** state, ready for the next user.

Library Book Issue - Activity Diagram



This is an **Activity Diagram** representing the **Library Book Issue** process. It models the flow of activities from logging in to either issuing a book or handling its unavailability.

🔄 Activity Flow Explanation:

1. **Start (Black Circle):**
   - The process begins.
2. **Login:**
   - The user logs into the library system.
3. **Search Book:**
   - The user searches for the desired book.
4. **Decision: Available?**
   - A decision is made based on book availability:
     - **Yes** → Proceed to **Issue Book**.
     - **No** → Go to **Display Message** (e.g., "Book not available").

□ If Book is Available:

5. **Issue Book:**
   - Book is marked for issue.
6. **Update Database:**
   - The system records that the book has been issued (updates inventory and student record).
7. **Print Receipt:**
   - A receipt is printed for the user.

🛑 If Book is Not Available:

- **Display Message:**
  - The system shows a message that the book isn't available.

🔚 Merge and End:

- After either path, the flows merge and the process ends (marked by the bullseye symbol).

✨ Highlights:

- ☑ Kept diagrams isolated to avoid syntax conflicts.
- ☑ Added title for easy understanding and annotation.
- ☑ Used fork and if/else properly for parallel and conditional actions.

Need a visual flow between these two systems, like simulating an integrated kiosk that handles both cash and book issues? I could sketch a hybrid diagram or write a short narrative to explain it. Just say the word!

Experiment No. 07

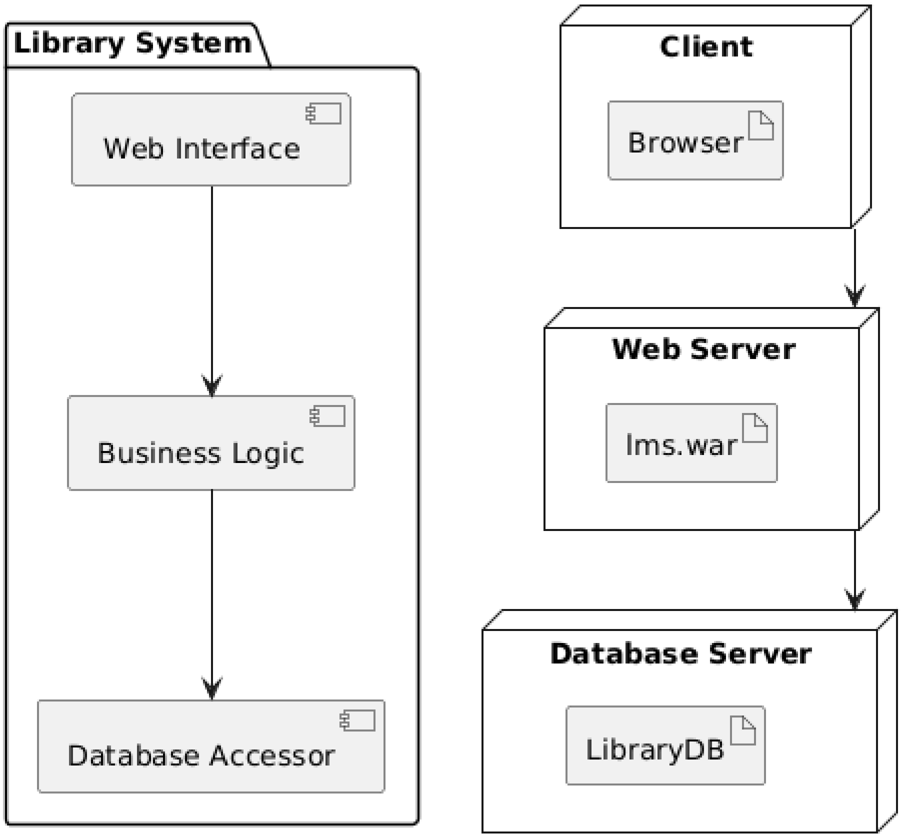
Q. Develop the Component and Deployment Diagram indicating physical deployment on hardware infrastructure.

Aim:
To design component and deployment views for LMS.

Theory:
Component diagrams show the modular parts of the system. Deployment diagrams show where the software components are physically deployed.

Procedure:
1. Define logical components like UI, DatabaseAccessor.
2. Define nodes like WebServer, DBServer.
3. Show mappings and communication.

Diagram :-



This is a **Deployment Diagram** for a **Library Management System**, showing how different software components are deployed across hardware nodes (servers and clients) and how they interact.

□ System Overview

The system is composed of three main layers:

1. **Client (User Interface Layer)**
2. **Web Server (Application Layer)**
3. **Database Server (Data Layer)**

🔍 Detailed Explanation

✅ *Client Node*

- **Component:** Browser
- **Role:** The end user (e.g., student, librarian) uses a web browser to interact with the Library System via the web interface.

✅ *Web Server Node*

- **Component:** lms.war
- **Role:** Hosts the **web application** (packaged as a .war file).
- It receives HTTP requests from the client and processes them using the **Library System** components.

✅ *Library System Components (on Web Server)*

Inside the .war file (i.e., part of the deployed application), we have:

1. **Web Interface**
   - Manages UI logic and handles user interactions (e.g., search books, login).
2. **Business Logic**
   - Contains the core functionality: validates actions, processes rules, and manages workflows (e.g., issuing books).
3. **Database Accessor**
   - Communicates with the database to fetch/store data (e.g., book availability, student records).

✅ *Database Server Node*

- **Component:** LibraryDB

- **Role:** Stores all persistent data related to the system (e.g., books, users, transactions).

🔄 Flow of Interaction

1. User interacts via the **Browser** (Client).
2. The **Web Server** runs the application (lms.war), handling requests through:
   - Web Interface → Business Logic → Database Accessor
3. The **Database Accessor** interacts with **LibraryDB** on the **Database Server**.
4. The processed response flows back up to the browser.

Experiment No. 08

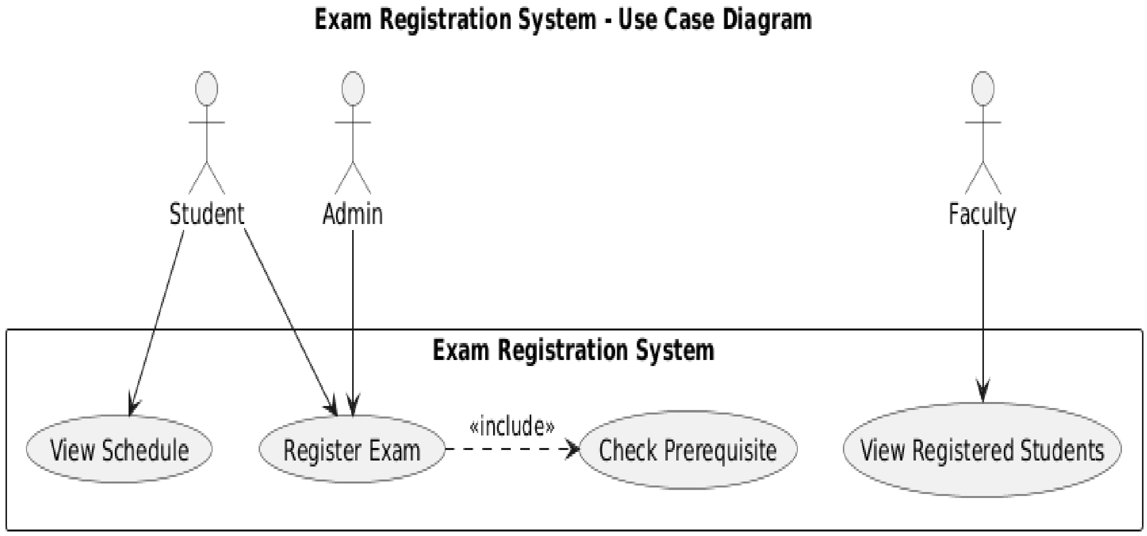
Q. Design an UML diagrams for Exam Registration Process.

Aim:
To model an online exam registration system using UML diagrams.
Theory:
Use Case, Class, and Sequence diagrams will represent user interactions and internal flow.

Procedure:
1. Actors: Student, Admin, Faculty.
2. Model registration, schedule viewing, conflict checking.
3. Include backend validation.

Diagrams (PlantUML):

Use Case Diagram



This is a **Use Case Diagram** for an **Exam Registration System**. It visually represents the **interactions between different actors** (users) and the **system's functionalities** (use cases).

□♂□ Actors (External Users):

1. **Student**
2. **Admin**
3. **Faculty**

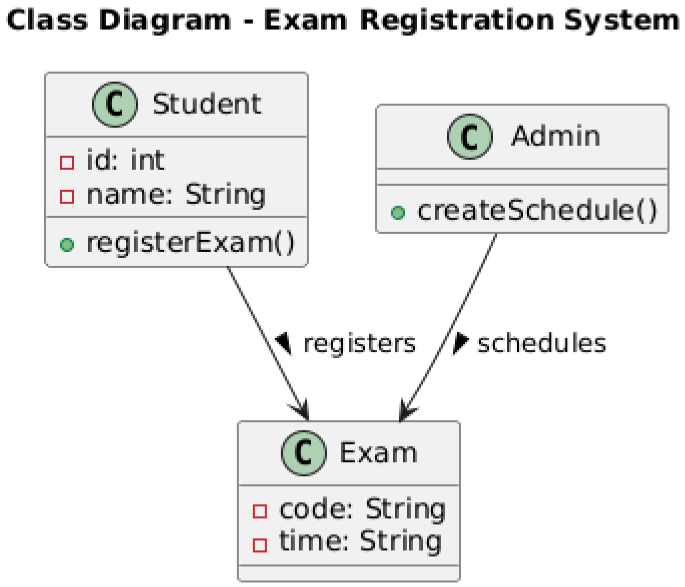
□ Use Cases (Functionalities):

1. **View Schedule**
   - **Actor:** Student
   - Allows students to check the exam schedule.
2. **Register Exam**
   - **Actors:** Student, Admin
   - Enables users to register a student for an exam.
3. **Check Prerequisite**
   - **Included Use Case** (dashed arrow labeled «include»)
   - Automatically performed as part of **Register Exam** to verify if the student meets eligibility (e.g., completed required courses).
   - This is **not performed directly** by any user.
4. **View Registered Students**
   - **Actor:** Faculty
   - Allows faculty to see which students have registered for which exams.

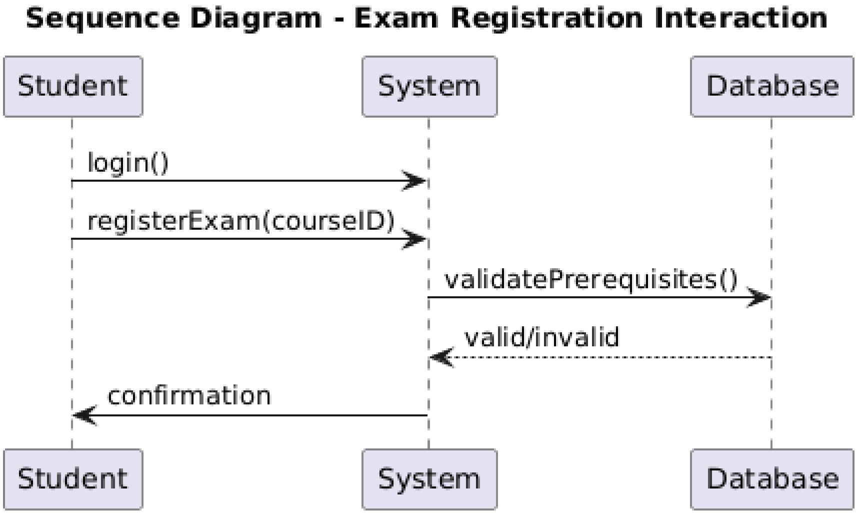
🔁 «include» Relationship:

- The **Register Exam** use case **includes** the **Check Prerequisite** use case.
- This means **checking prerequisites is a mandatory sub-process** of registering for an exam.

Class Diagram:-



Sequence diagram :-



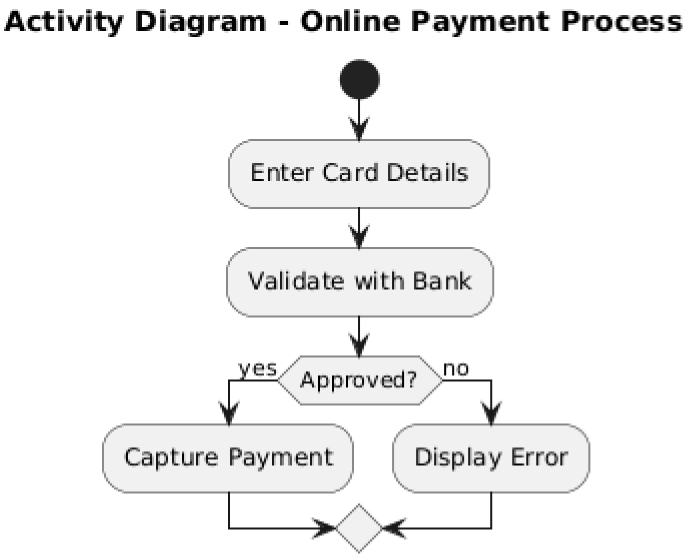
Experiment No. 09

Q. Design an UML diagrams for credit card Processing.

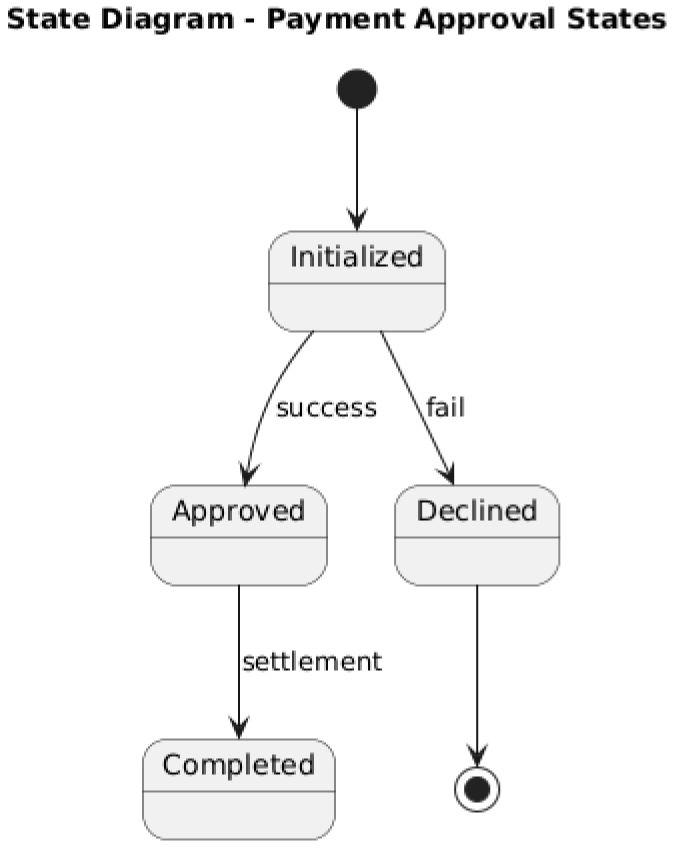
Aim:
To model workflow and transaction states of a credit card processing system.
Theory:
Activity diagrams show transaction steps; state diagrams model status of the transaction object.

Procedure:
1. Identify workflow steps: request, auth, capture.
2. Show transaction object states: initialized, approved, declined.
3. Model in UML.

Sample PlantUML Code:



Activity Diagram - Online Payment Process


State Diagram - Payment Approval States
Initialized
success
fail
Approved
Declined
settlement
Completed