Topic :- EXPERIMENT 7 Date : __/__/____

AIM:

To create the use case template.

THEORY:

A use case template is a structured format used to describe the functional behaviour of a system from a user's point of view. It defines how actors interact with the system to achieve a specific function or outcome.

Uses:

1. Requirement analysis: Helps to understand user requirements clearly.
2. System design: Assists in defining system behaviour flow.
3. Documentation: Provides a structured way to describe functional requirements.
4. Testing: Used as the basis of creating test cases and validating user interaction.
5. Communication: Acts as bridge between stakeholder and developer.

Good Write

Teacher's Sign. .....

EXPERIMENT 7

USE CASE TEMPLATE – EVENT TICKET BOOKING SYSTEM

USE CASE ELEMENT	DESCRIPTION
1. Brief Description	This use case describes the process through which a user (customer) browses available events, selects seats, makes payment, and receives an electronic ticket. The system validates the payment, confirms booking, and updates the ticket inventory in real time.
2. Actors	• **User (Customer):** Initiates the ticket booking process. • **Admin:** Manages event data, booking information, and system records. • **Payment Gateway:** Handles secure online payments. • **System (Event Ticket Booking System):** Facilitates the entire booking workflow.
3. Flow of events	
3.1 Basic Flow	1. The user logs into the system. 2. The user browses available events. 3. The system displays event details and proceeds to checkout. 4. The user selects the desired seats and proceeds to checkout. 5. The system redirects to the payment gateway. 6. The user makes payment through a secure gateway. 7. The system verifies the payment confirm booking. 8. The system generates and sends an e-ticket to the user. 9. The ticket inventory is updated automatically. 10. The admin can view booking details and analytics.
3.2 Alternate Flows	1. **Failed Payment:** If payment fails, the system notifies the user and cancels the booking. 2. **Seats Already Booked:** If selected seats are already booked, the system prompts the user to choose other seats. 3. **Event Cancelled:** If the event is cancelled, the system issues an automatic refund and sends a notification.

4. Special Requirements	• Secure payment integration. • Real time seat availability updates. • Responsive web interface. • Reliable backend. • Cloud hosting and database management.
5. Pre-Conditions	• User has access to the internet and the ticket booking platform. • Events and seats are available in the database. • Payment gateway service is active.
6. Post-Conditions	• User receives a valid e-ticket. • Booking data and payment status are updated. • Admin and organizer can track sales and bookings.
7. Extension Points	• Integration with loyalty or coupon systems. • SMS/Email/WhatsApp notifications. • Feedback collection after event.

Topic :- EXPERIMENT 3 Date : __/__/____

Aim:

To create the SRS document of the project.

THEORY:

A SRS document is a formal document that defines function and non functional requirements of software system. It describes what the system is expected to do and the constraints under which it operates.

Uses:

1. Define the scope and boundaries of the system.
2. Acts as a contract between stakeholders and developers.
3. Helps in estimating cost, time and resources for the project.
4. Serves as a base for designing ~~that~~ test case and validation.
5. Aid in future maintainance and enhancement.

EXPERIMENT 8

SOFTWARE REQUIREMENTS SPECIFICATION (SRS) - EVENT TICKET BOOKING SYSTEM

1. Introduction

1.1 Purpose

This SRS defines the functional and non-functional requirements for developing the Event Ticket Booking System (ETBS), an online platform that automates the process of browsing, booking, and managing event tickets. It provides customers with real-time access to events and allows event organizers to efficiently manage event details and sales.

1.2 Scope

The ETBS enables users to:

- Search and view upcoming events.
- Select seats and make secure online payments.
- Receive electronic tickets instantly.

Admins can:

- Add and manage event listings.
- Track bookings and analyze sales.

The system aims to enhance user convenience, reduce errors, and provide real-time updates on ticket availability.

2. Overall Description

2.1 Product Perspective

The system is a web-based application operating in a client-server environment. It integrates:

- Customer Module – Event search, booking, payment.
- Admin Module – Event management and reporting.

Data Stores:

- D1: Event Info
- D2: Booking Info
- D3: Payment Info

2.2 Product Functions

1. Search Events: Retrieve event lists from the database.
2. Select Tickets: Choose events, seats, and ticket categories.
3. Make Payment: Process and validate online payments.
4. Generate Confirmation: Issue e-tickets and send confirmation.

2.3 User Characteristics

Customer: End user booking tickets online.

Admin: Manages events and monitors bookings.

2.4 Constraints

Requires internet connection.

Must comply with PCI-DSS for payment security.

Compatible with major browsers.

2.5 Assumptions

Third-party payment gateway available.

Email/SMS services active for notifications.

3. Specific Requirements

3.1 Functional Requirements

ID	Requirement Description
FR-1	System shall allow users to search and view events.
FR-2	System shall allow users to select and book tickets.
FR-3	System shall securely process online payments.
FR-4	System shall generate and send electronic booking confirmations.
FR-5	Admin shall be able to manage event details and monitor bookings.

3.2 Non-Functional Requirements

Category	Requirement Description
Performance	Should support 500+ concurrent users.
Security	Use SSL/TLS encryption and secure payment gateways.
Usability	Intuitive, mobile-friendly interface.
Reliability	Maintain uptime of 99.5%.
Scalability	Support addition of new events and organizers.

3.3 Interface Requirements

- UI: Web-based dashboard for users and admins.
- Software Interfaces: Payment Gateway API, Email/SMS API, MySQL/PostgreSQL DB.
- Communication: HTTPS protocol for secure data exchange.

4. Change Management

4.1 Process

Any change in requirements shall follow:

1. Submission of Change Request (CR).
2. Evaluation by the project and QA teams.
3. Approval by Project Manager or Client.
4. Documentation and implementation in next release.

4.2 Version Control

Each update shall include version number, author, date, and summary of changes.

5. Document Approvals

Role	Name/Designation	Signature	Date
Project Manager			
Business Analyst			
Lead Developer			
QA Lead			
Client Representative			

6. Supporting Information

6.1 Related Documents

- Level 1 DFD Diagram
- Database Schema Document
- UI Wireframe Document

Topic EXPERIMENT - 9 Date 18/11/25

AIM:

To draw the structural view diagram for the system: Class diagram and Object diagram.

THEORY:

A class diagram represents the static structure of a system. It shows classes, their attributes, methods and the relationships between them.
An object diagram shows a snapshot of the system at runtime using actual instances of classes. It illustrates real data and relationships occuring in a specific scenario.

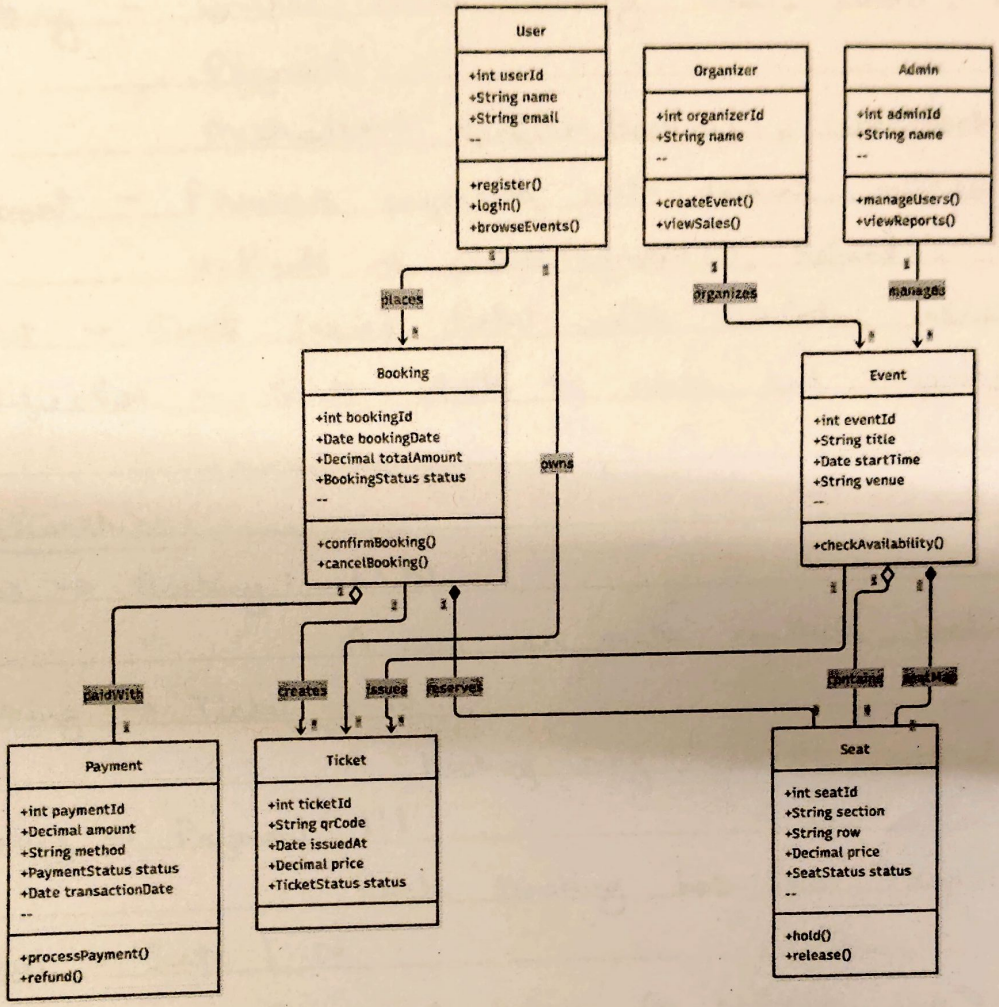
Class Diagram of the Project:

Key classes:

- User - holds userID, name, email.
  Actions → register(), login(), browseEvents().
- Organizer - Creates and manages events
  Methods → createEvent(), viewSales().
- Admin - Handles platform level tasks.
  Methods → manageUsers(), viewReports().
- Event - Stores event details (title, venue, timings).
  Method → checkAvailability().
- Seat - Represents event seating with sections, row, price.
  Methods → hold(), release().

SAARTHI™

Class Diagram



Topic ..... Date .....

- Booking – Central class linking User, Event, Seat, Ticket, Payment.
  Methods → confirmBooking(), cancelBooking().
- Payment – Processes payments with amount, method, status.
  Methods → processPayment(), refund().
- Ticket – Final issued ticket with QRCode, issuedAt, status.
- Notification – Sends alerts to users and organizers.

Relationships:

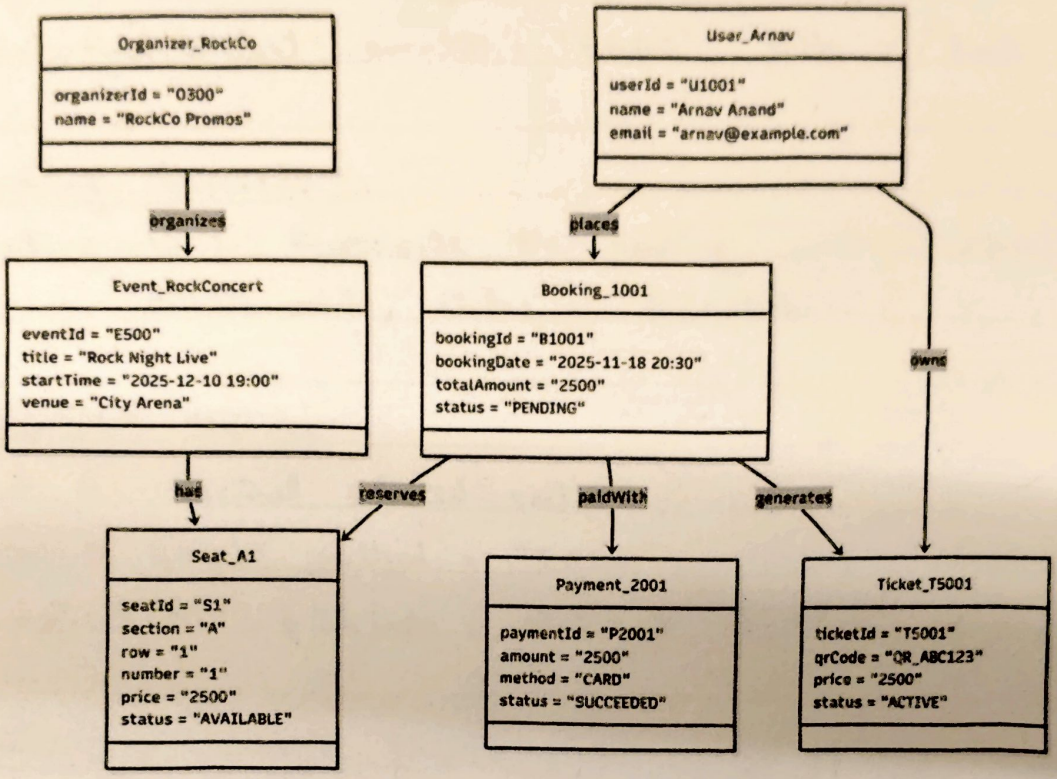
- User → Booking: 1:N
  A user can make multiple bookings.
- Booking → Ticket: 1:N
  A booking may generate several tickets.
- Booking → Payment: 1:1
  Each booking has one payment.
- Event → Seat: 1:N
  One seat belong to multiple events.
- Booking → Seat: Booking reserves seat.
- User → Ticket: Users own the tickets issued.
- Organizer/Admin → Event: Organizes and manages the events.
- Event → Ticket: Tickets are linked to specific events.
- Notification → User/Organizer: System sends updates.

Object Diagram of the Project:

1. Runtime Objects:

- User_AS: userID: "U1001", name = "AS".

Object Diagram



Topic ..... Date .....

- Organizer – Rocko : Organizer managing the event.
- Event – Rock Concert : eventID = "E500", title = "Rock Night Live"

2. Booking Scenario :
   - Booking – 1001 : Represents the booking with bookingDate, totalAmount, status = "PENDING".

3. Associated Stances :
   - Seat – A1 : Actual selected seat.
   - Payment – 2001 : method = "CARD", status = "SUCCEEDED"
   - Ticket – 5001 : & QR Code, status = "ACTIVE"

4. Relationship between Objects :
   - User – AS → Booking : User places the booking.
   - Organizer – Rocko → Event – Rock Concert : Organizer manages the event.
   - Event – Rock Concert → Seat – A1 : Seat belongs to the event.
   - Booking – 1001 → Seat – A1 : Seat reserved in the booking.
   - Booking – 1001 → Payment – 2001 : Payment linked to booking.
   - Booking – 1001 → Ticket – 5001 : Ticket generated from booking.
   - User – AS → Ticket – 5001 : User owns the ticket.

Topic EXPERIMENT - 10 Date 18/11/25

AIM:

To draw the behavorial view diagram : State chart diagram and Activity diagram.

THEORY:

A state chart diagram shows how an object changes states in response to events, focusing on the lifecycle of a single entity.
An activity diagram models workflows, operations and the sequence of actions in a system, showing decision points, parallel flows and conditional activities.

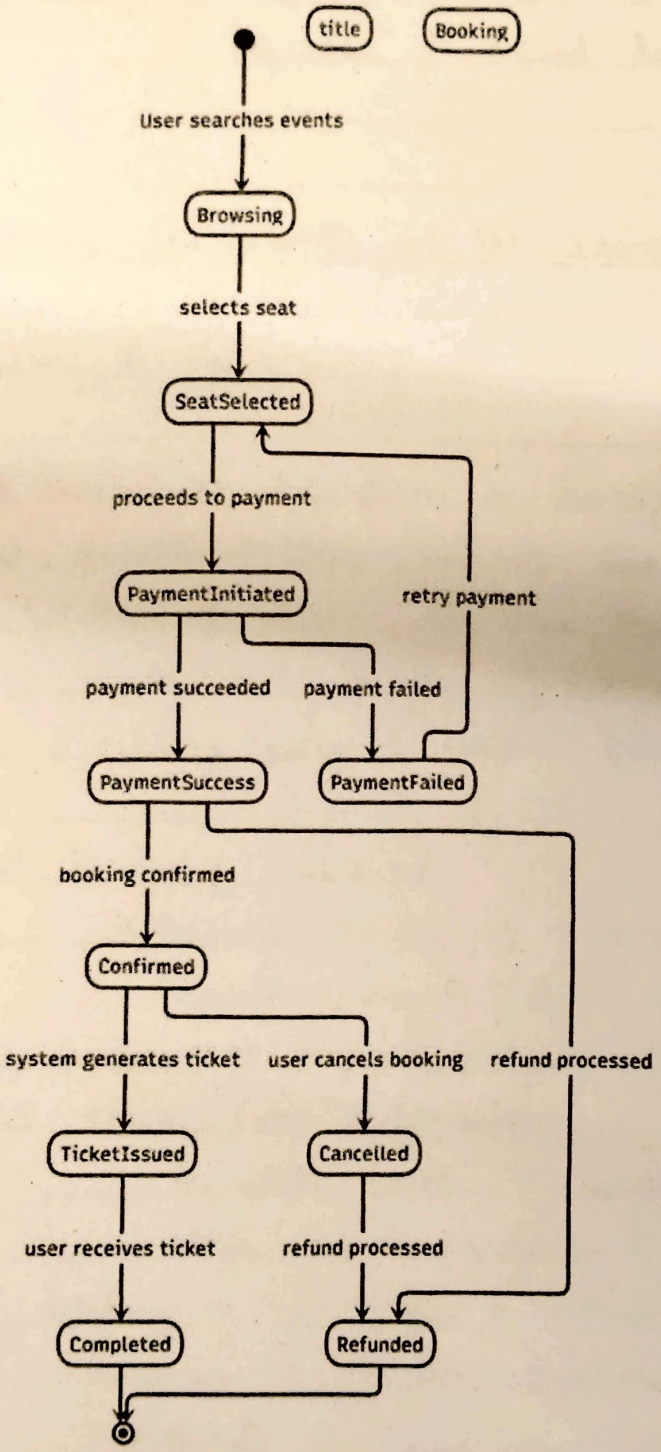
State Chart Diagram of the Project:

States & Transitions:

1. Initial State - User starts browsing events → Browsing state
2. Seat Selection - User selects a seat → Seat Selected state
3. Payment Processing - Payment Initiated → two outcomes:
   - Payment Success → move to booking confirmation.
   - Payment Failed → user may retry.
4. Booking Confirmation - Successful Payment → Confirmed state
5. Ticket Generation / Cancellation - Ticket Issued → ticket generated
   Cancelled → user cancels → refund

SAARTHI™

State Chart Diagram



Topic ..... Date .....

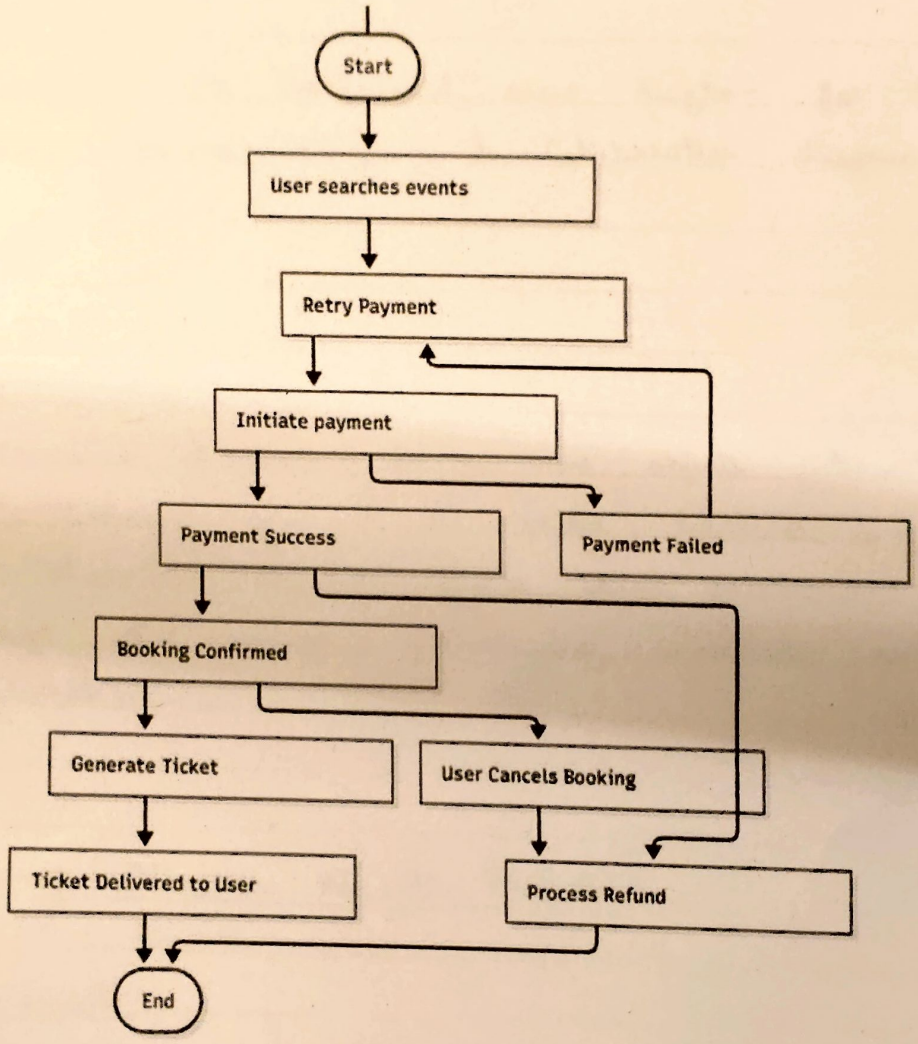
6. Final states ← Completed → Ticket delivered
Refunded → Refund done, booking cancelled.

ACTIVITY DIAGRAM of the Project:

Activities & Flow:

1. Start Node – Initiation of booking process.
2. Search Events – User browses available events.
3. Seat Selection & Payment Initiation – User selects events/ seats and starts payment.
   - Retry payment: Handles failed attempts; user can retry or exit.
4. Payment Processing Outcomes ←
   - Payment Success → Proceed to booking confirmation
   - Payment Failed → Redirect to retry payment.
5. Booking Confirmed ← Payment Validated → Booking confirmed.
6. Parallel Flows Post Confirmation:
   - Generate Ticket → system issues e-ticket.
   - User cancels booking → triggers refund process.
7. Ticket delivery – E-ticket is delivered to user's email/app.
8. Refund Processing – Handles cancellation or failed payment refunds.
9. End Node – Marks completion of either ticket delivery or refund process.

Activity Diagram



Topic EXPERIMENT - 11 Date 18/11/25

AIM:

To perform the behavioral view diagram for the suggested system: Sequence diagram & Collaboration diagram.

THEORY:

A sequence diagram shows how objects interact over time, focusing on message order, lifelines and interactions. A collaboration diagram shows object interactions via message exchange, emphasizing connections and collaborations.

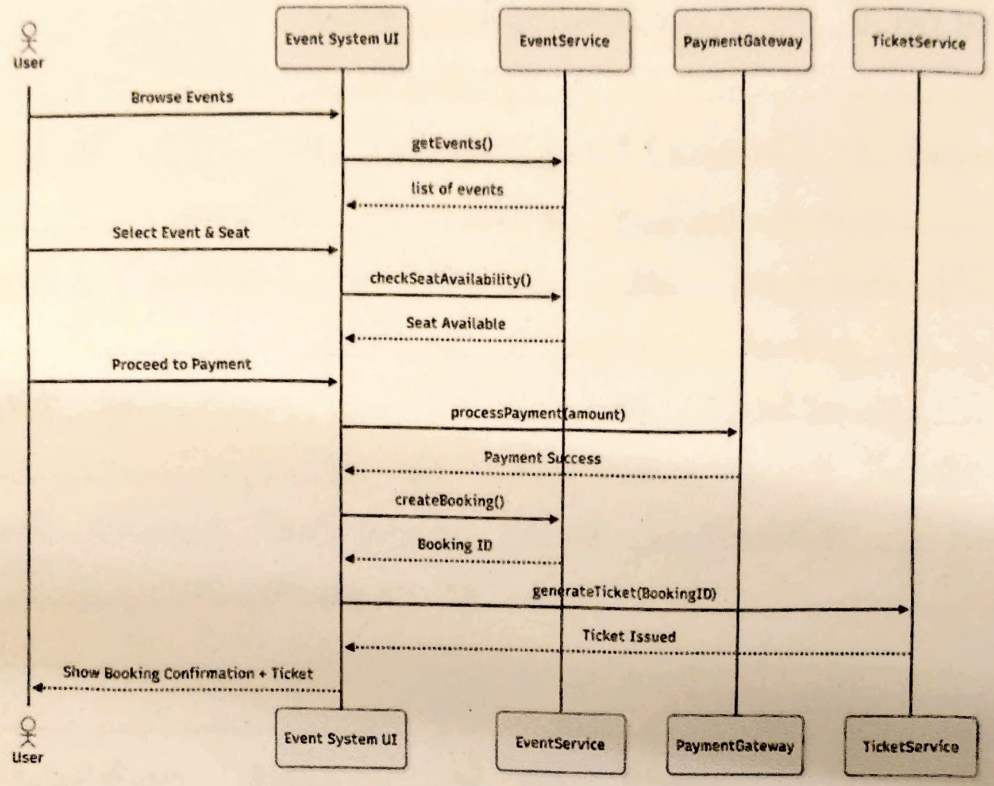
Sequence Diagram of the Project:

- Participants:
- User - Initiates booking activities.
- Event System UI - Front end interface.
- EventService - Backend handling event data, seat checking and bookings.
- Payment Gateway - External payment processor.
- TicketService - Generates and issues tickets.

Flow of Interactions:

1. Browsing Events: User requests events → UI calls getEvents() → EventService returns event list.
2. Selecting Event & Seat: User selects event/seat → UI calls

Sequence Diagram



Topic ..... Date .....

checkSeatAvailability() → EventService confirms availability.

3. Payment: UI calls processPayment(amount) → PaymentGateway processes → returns Payment Success.
4. Booking Creation: EventService calls createBooking() → stores booking → returns Booking ID.
5. Ticket Generation: EventService calls TicketService generateTicket(BookingID) → tickets issued.
6. Final Display: EventService sends confirmation + ticket → UI displays to user.

Collaboration Diagram of the Project:

Key Objects:

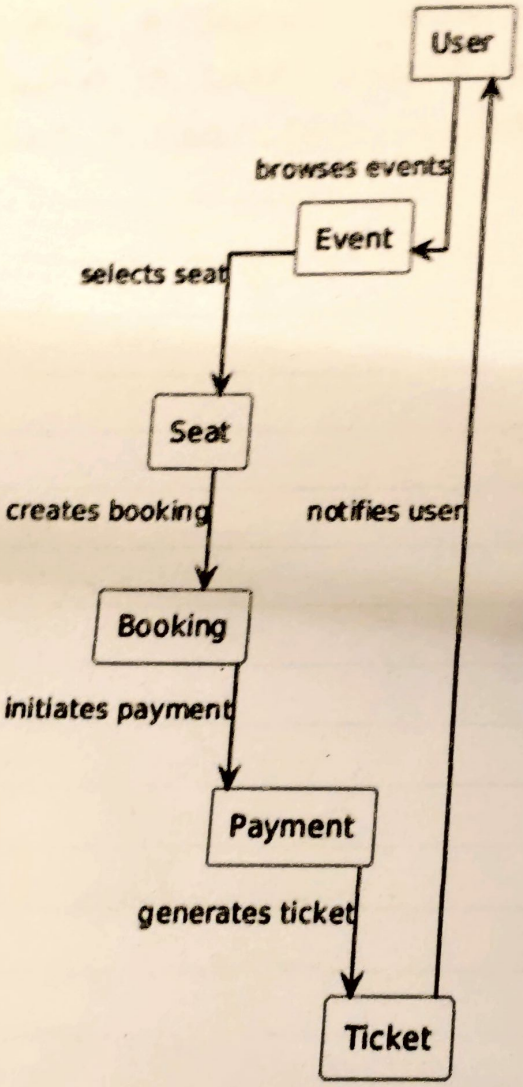
- User
- Event
- Seat
- Booking
- Payment
- Ticket

Interaction Flow:

1. User → Event: Browses available events → receives details
2. User → Seat: Selects preferred seat → system verified and reserved temporarily.
3. Seat → Booking: Passes seat info → booking object creates booking with user, seat, price, event details.

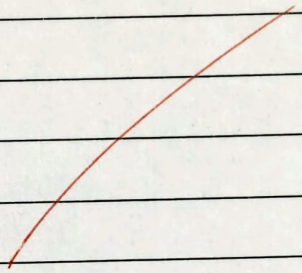
SAARTHI™

Collaboration Diagram



Topic ..... Date .....

4. Booking → Payment: Initiates secure payment process.
5. Payment → Ticket: Payment success triggers ticket generation.
6. Ticket → User: Sends e-ticket/confirmation to the user.



Topic EXPERIMENT – 12 Date 18/11/25

AIM:

To perform the implementation view diagram : Component diagram of the system.

THEORY:

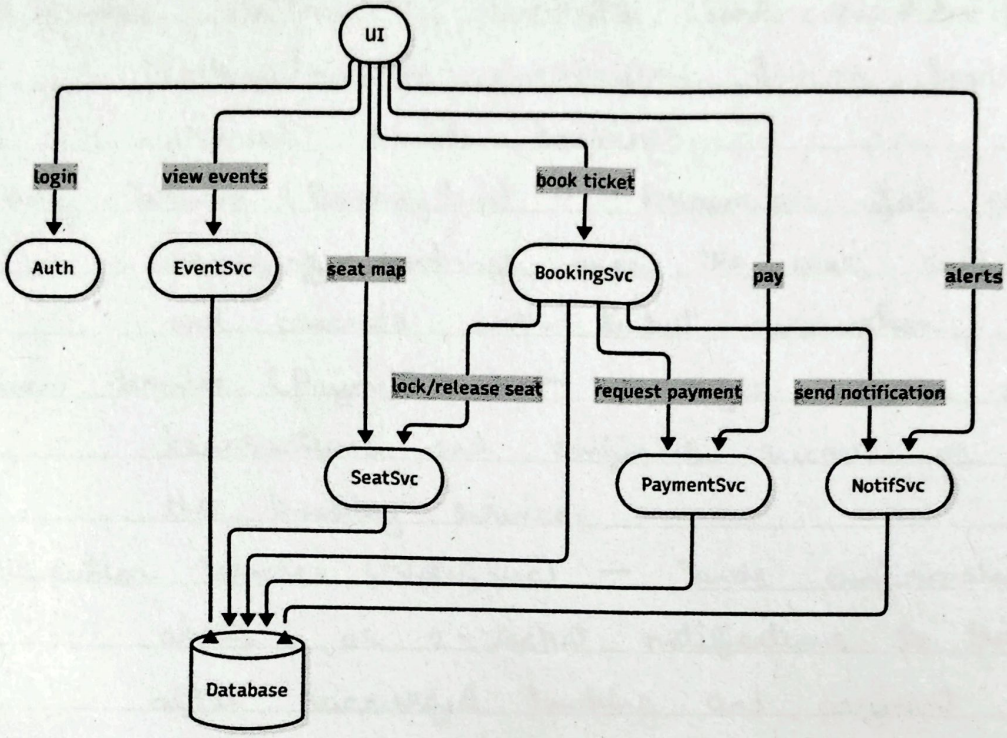
A component diagram illustrates the physical architecture of the software system by showing how different components are organized and interact with each other. It highlights the structural relationships between subsystems and the interfaces they expose. Each component represents a deployable, replaceable and reusable part of the software such as modules, services or databases.

Component Diagram of the Project:

Components:

- User Interface – Acts as the front end where the user interacts with the system to log in, browse events, view seat maps and book tickets.
- Auth Service (Auth) – Handles user authentication such as login and access permissions.
- Event Services (Event Svc) – Manages event data including event listings, venues and schedules. Provides information to the UI when the user browses events.

Component Diagram



Topic ..... Date .....

- Seat Service (SeatSvc) ← Controls seat availability and locking/release mechanism during booking to prevent double bookings.
- Booking Service (BookingSvc) – Responsible for creating and managing bookings once the user selects a seat and proceeds with ticket reservation.
- Payment Service (PaymentSvc) – Manages secure payment transactions and confirms success or failure to the booking service.
- Notification Service (NotifSvc) – Sends confirmation messages, alerts, or e-ticket notifications to the user after successful booking and payment.
- Database: Central storage for all system data including users, events, seats, bookings and transaction records. All services interact with the database to fetch or update records.

Flow Explanation:

1. The user interacts with the UI to log in, browse events and book tickets.
2. The auth component validates user credentials.
3. The EventSvc provides event details and updates UI.
4. When booking is initiated, the SeatSvc locks/releases seats to ensure availability.
5. BookingSvc creates a booking entry and coordinates the overall process.
6. PaymentSvc processes the payment request securely.
7. After a successful transaction, NotifSvc sends alerts/

Topic ..... Date .....

notifications to the user.

8. All key components constantly and retrieve information from the Database.