INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD

DEPARTMENT OF COMPUTER SCIENCE AND IT

PROJECT

COURSE: OBJECT-ORIENTED ANALYSIS AND DESIGN (OOAD)

COURSE CODE: SE 321

PROFESSOR: MR ABID JAMEEL

SUBMISSION DATE: 1ST JULY 2025

SUBMITTED BY:

- ***WAHAB EJAZ (5033-FOC/BSCS/F23)***
- ***SAAD KHALEEQ (5049-FOC/BSCS/F23)***

EVENT MANAGEMENT SYSTEM

PROJECT REPORT

Abstract

The Event Management System (EMS) is designed to automate and streamline the process of organizing, booking, and managing events. It serves as a platform for customers to book events and venues, while allowing managers to oversee event details and send notifications. This project applies Object-Oriented Analysis and Design (OOAD) principles to model the system's structure and behavior using UML tools. The EMS features include user management, event creation, booking handling, venue allocation, and notification delivery. The system ensures organized event planning, eliminates manual processes, and increases efficiency and accuracy. Through this project, we demonstrate how a real-world system can be modeled using OOAD concepts and diagrams such as Use Case, Class, Activity, Sequence, and Data Flow Diagrams.

Table of Contents

List of Figures:

1. Introduction

Event planning is a common and essential task in various domains like academia, business, and social functions. The Event Management System (EMS) provides a software solution to handle this process in an efficient, structured way.

1.1 Goals and Scope:

- Enable customers to browse and book events.
- Allow managers to create and manage events and venues.
- Notify users about booking confirmations or updates.
- Provide an object-oriented model of the EMS.

1.2 Technologies Used:

- C++ or Java for implementation
- UML tools like StarUML, Lucidchart, or Draw.io for diagrams

2. Problem Statement

Managing events manually is time-consuming and error-prone. Customers face issues booking venues, managers struggle to track bookings, and communication gaps often occur.

The EMS addresses the following problems:

- Difficulty in managing event logistics manually.
- Lack of centralized booking and notification system.
- Inefficient communication between users and managers.

This system is needed to digitize the workflow, provide a user-friendly interface, and reduce event handling overhead.

3. UML Diagrams

3.1 Use Case Diagram

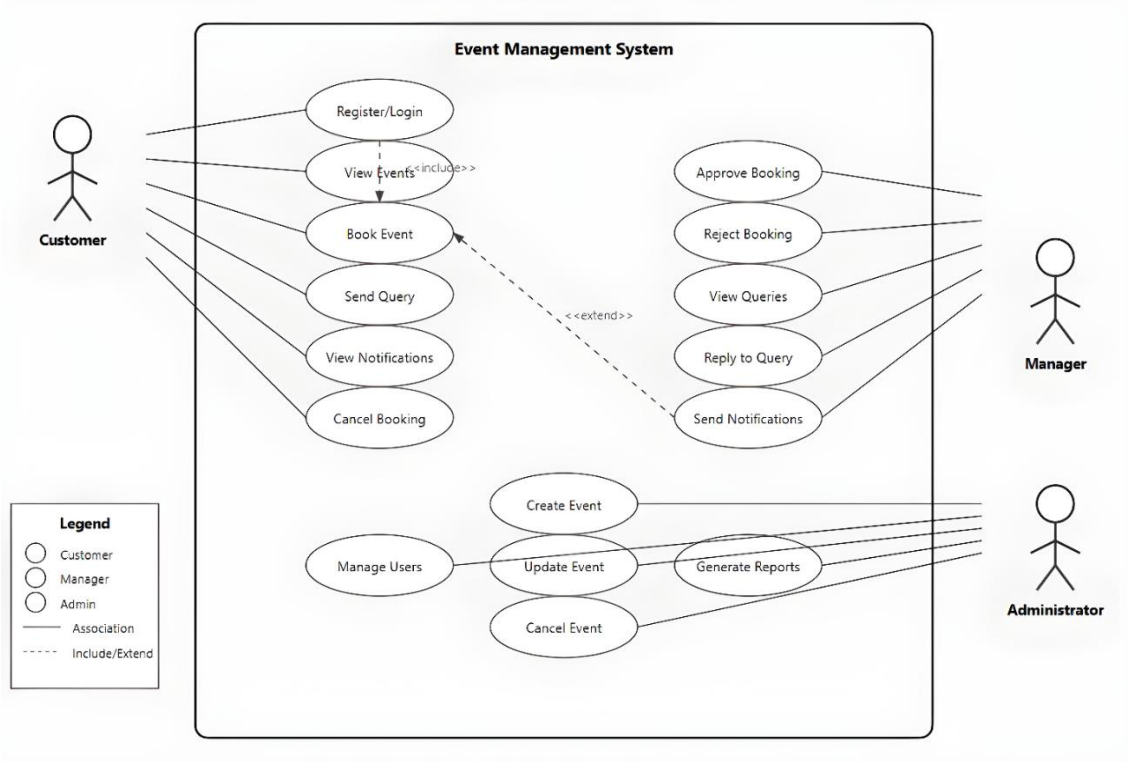

Figure 1: Use case Diagram

Explanation:

This diagram represents the functional interaction between the system and the actors (Customer and Manager). Main use cases include: Book Event, Create Event, Manage Venue, and Send Notification.

Customer Features

- Register/Login.
- Browse and filter events.
- Book events (if venue is available).
- Cancel bookings.
- Send queries and view notifications.

Manager Features

- Create, update, or cancel events.
- Approve/reject bookings.
- Reply to customer queries.
- Generate reports (e.g., booking trends).

3.2. Class Diagram

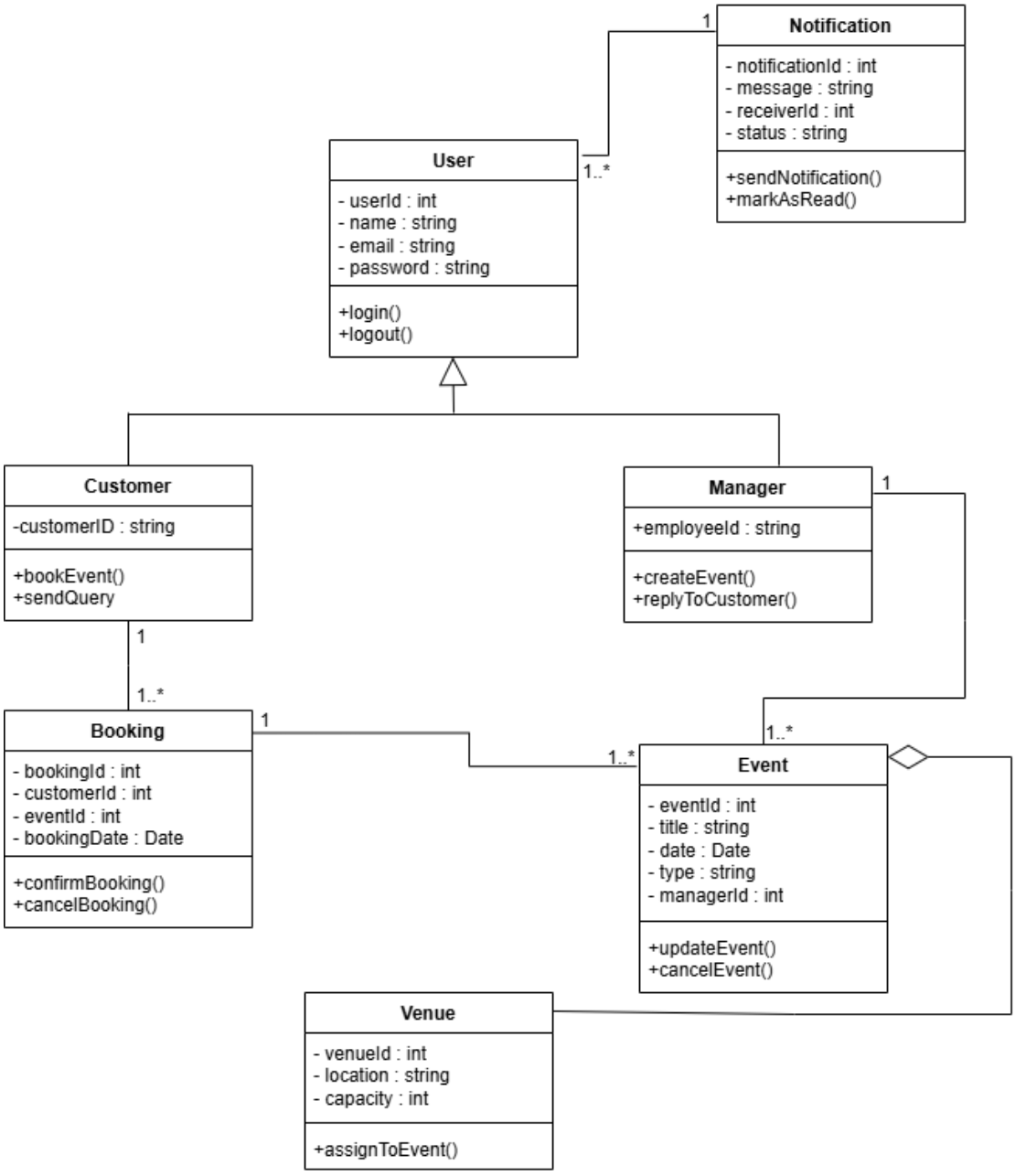

Figure 2: Class Diagram

Explanation:

This diagram shows the core classes (User, Customer, Manager, Event, Booking,

Venue, Notification) and their relationships such as inheritance, associations, and multiplicities.

Key classes

- User (attributes: userId, name, email; methods: login(), logout())
- Customer and Manager (inherit from User)
- Event (attributes: eventId, title, date, capacity; methods: updateDetails())
- Booking (attributes: bookingId, status; methods: confirm(), cancel())
- Venue (attributes: venueId, location, capacity)
- Notification (attributes: messageId, timestamp; methods: send())

3.3 Activity Diagram

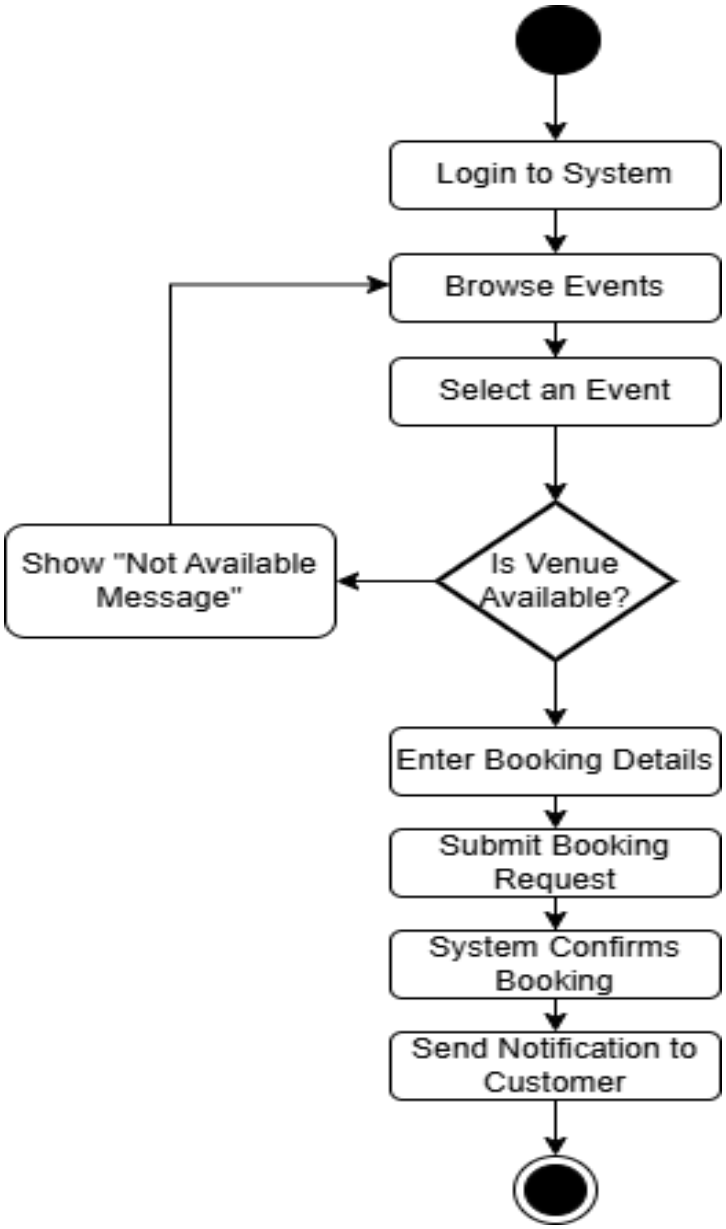

Figure 3: Customer Activity Diagram

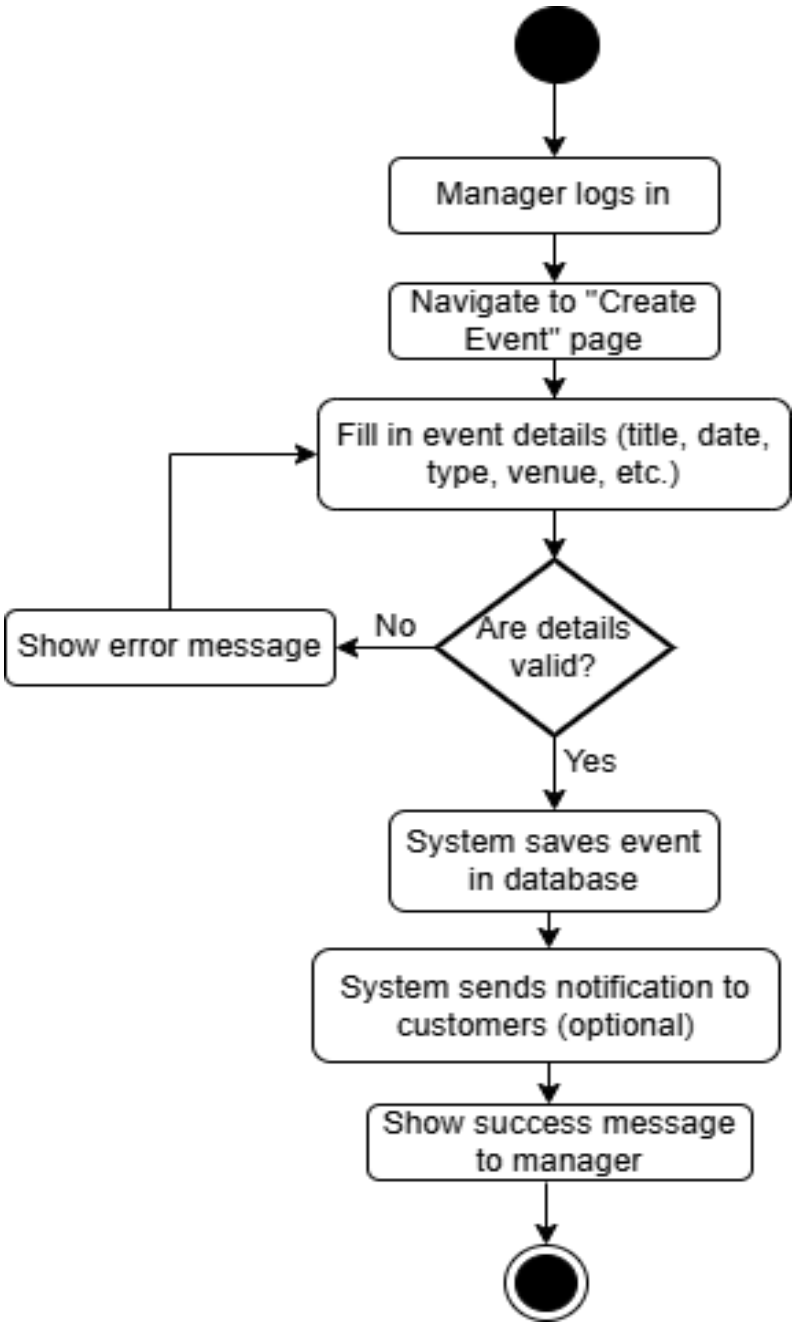

Figure 4: Manager Activity Diagram

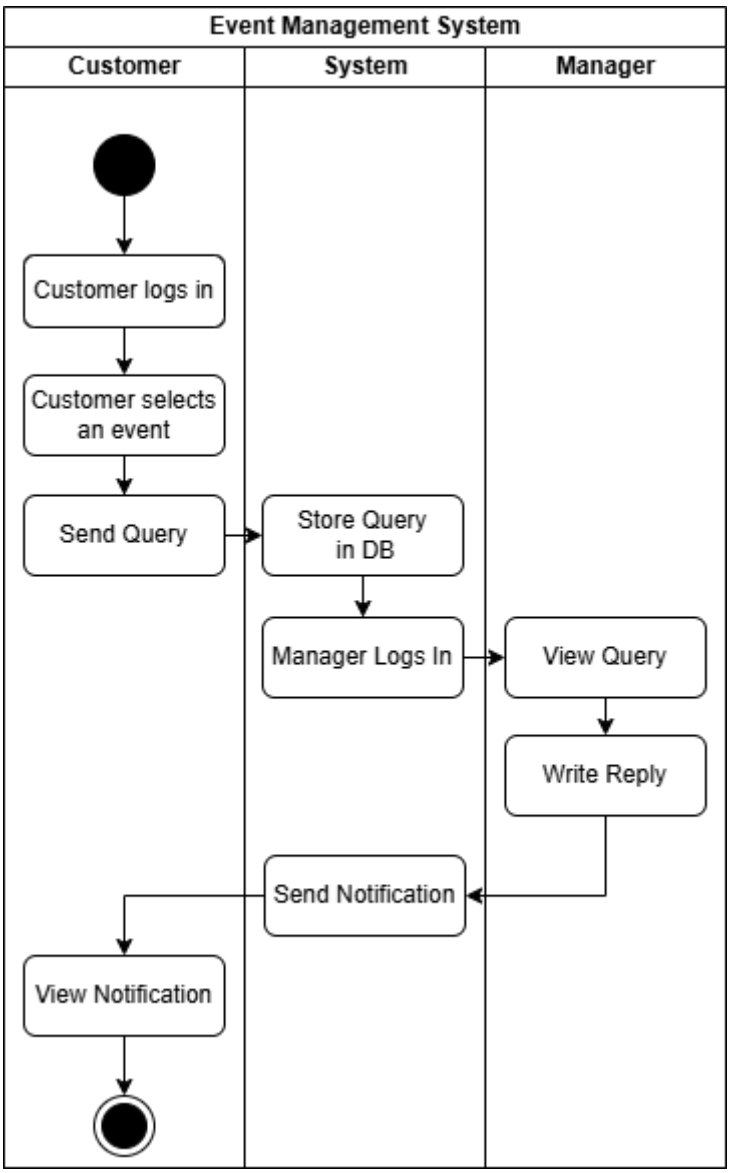

Figure 5: EMS Full System Activity Diagram

Explanation:

Demonstrates the workflow for event booking or event creation. Starts from user login and ends at successful event booking/creation.

Example workflow

- Customer selects event
- System checks availability

- Booking is created or declined
- Notification is sent

3.4 Sequence Diagram

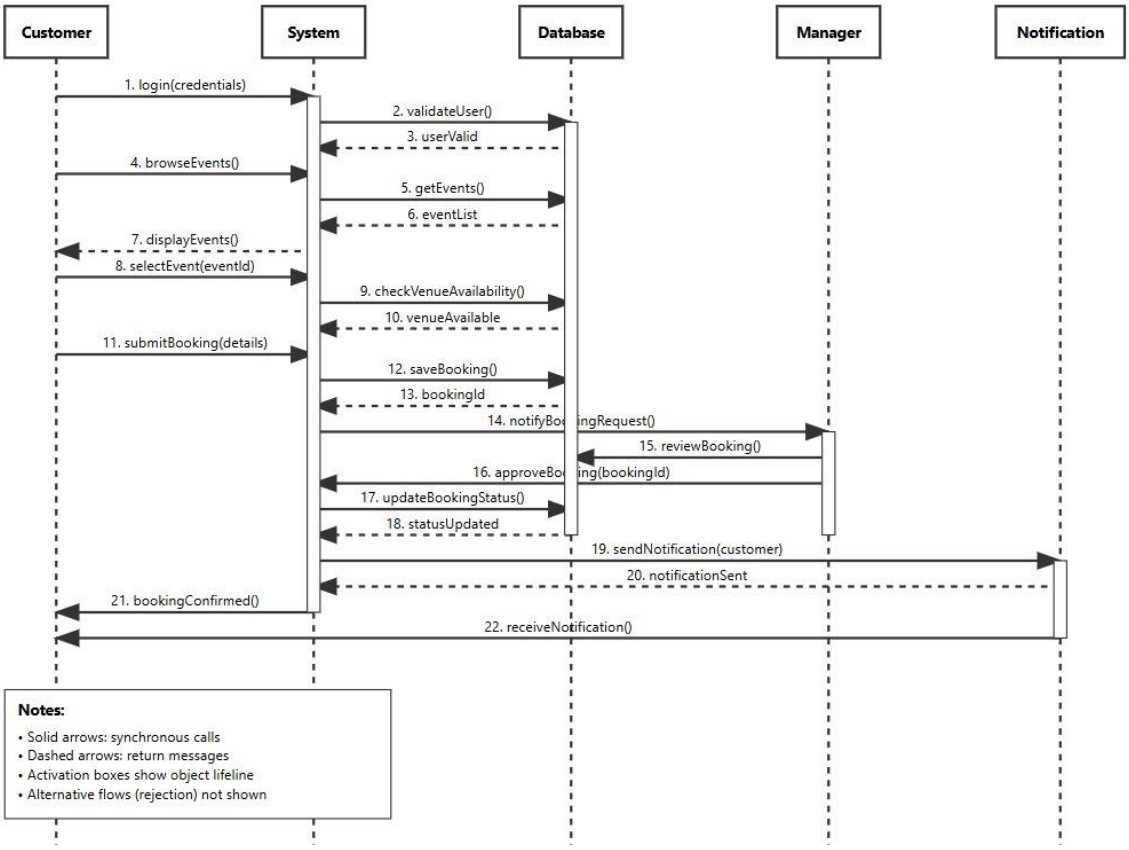

Figure 6: Sequence Diagram

Explanation:

Illustrates the sequence of interactions between Customer, Booking system, and Manager for an event booking process.

Illustration of booking sequence

1. Customer → System: submitBookingRequest()
2. System → Event: checkAvailability()
3. Event → System: availabilityStatus
4. System → Booking: createBooking()

5. System → Notification: sendConfirmation()
6. Notification → Customer: deliverMessage()

4. Data Flow Diagrams (DFD)

4.1 Level 0 – Context Diagram

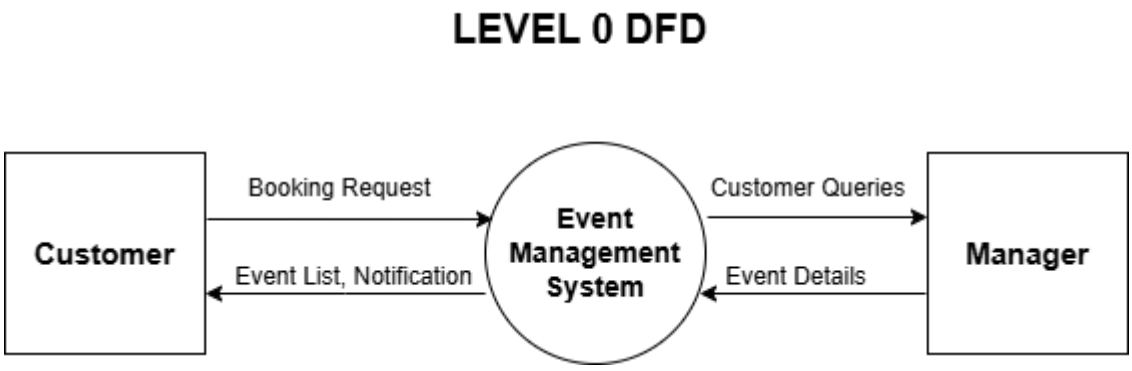

Figure 7: Level 0 DFD

Explanation:
Shows the entire EMS as a single process interacting with external entities like Customer and Manager

4.2 Level 1 – Process Breakdown

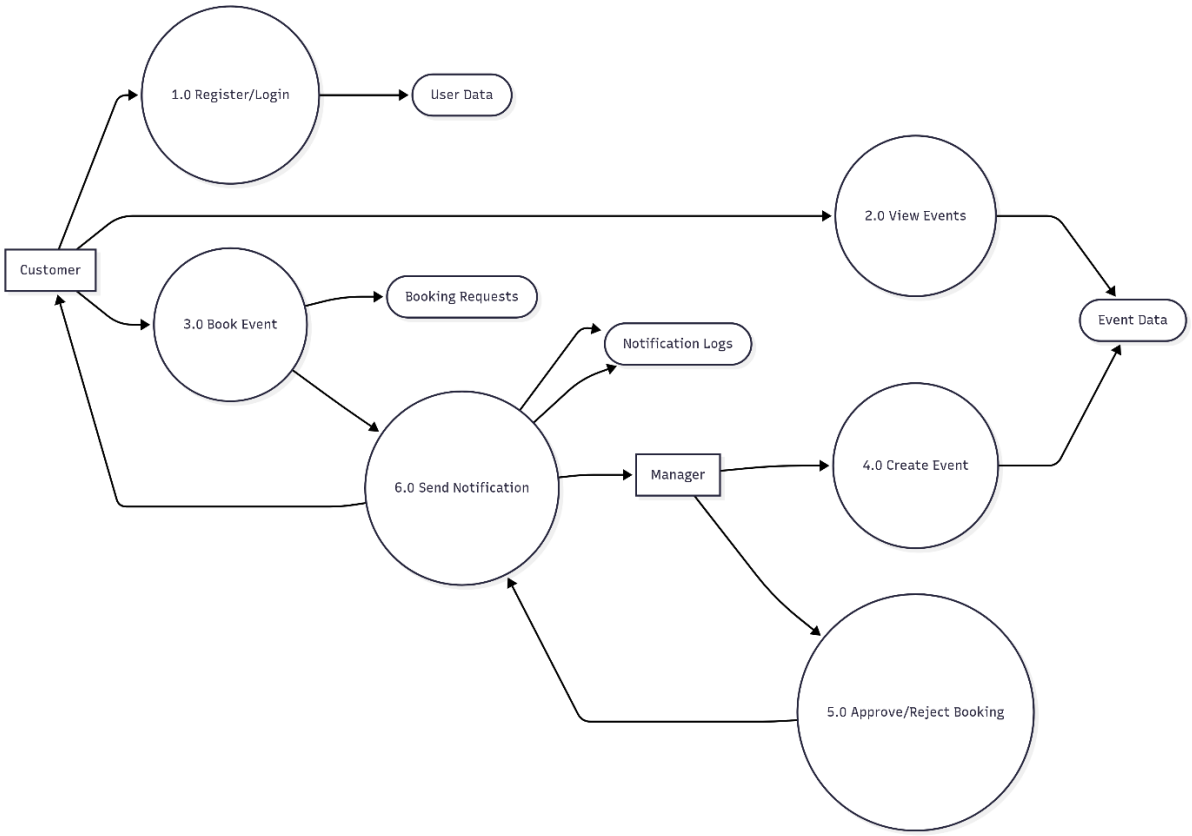

Figure 8: Level 1 DFD

Explanation:

Breaks down EMS into key processes like Booking Management, Event Management, Venue Management, and Notification.

5. System Design

5.1 Architecture Overview:

- Layered Architecture (Presentation, Business Logic, Data Layer)
- Interface for each user role (Customer/Manager)
- Class-based modules for handling Booking, Event, Venue

5.2 Interfaces:

- Login Page
- Customer Dashboard (View Events, Book Venue)

- Manager Dashboard (Create Event, Assign Venue)

6. Conclusion

This project gave us practical experience in applying OOAD principles to a real-world system. By modeling the Event Management System through class design, workflows, and data flow diagrams, we learned how to convert abstract user requirements into visual, object-oriented models. Some challenges faced included modeling real-life interactions accurately and ensuring proper relationships between classes, but these were overcome through iterative design and team discussion.

7. References

1. Object-Oriented Analysis and Design with Applications – Grady Booch
2. UML Distilled – Martin Fowler
3. https://www.lucidchart.com/
4. https://draw.io/
5. Lecture Notes & OOAD Course Material – IIUI