**Event Booking and Management System**

**A PROJECT REPORT**

**for**

**Mini Project (KCA353)**

**Session (2024-25)**

**Submitted by**

**Ishu**

**2300290140079**

**Kishan Batra**

**2300290140090**

**Deepraj Singh**

**2300290140051**

**Mayank Saini**

**2300290140101**

**Submitted in partial fulfilment of the**

**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**Mr. R.N. Panda**

### Associate Professor



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**Dec. 2024**

**CERTIFICATE**

Certified that **Ishu 2300290140079, Deepraj Singh 2300290140051, Kishan Batra 2300290140090, and Mayank Saini 2300290140101** has carried out the project work having “**Event Booking & Management System**” (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU)(formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other Institution.

**Mr. R.N. Panda Dr. Arun Kumar Tripathi**

**Associate Professor Head**

**Department of Computer Applications Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

**Event Booking & Management System**

**Ishu**

**Kishan Batra**

**Deepraj Singh**

**Mayank Saini**

**ABSTRACT**

The Event Booking and Management System is a full-stack web application designed to make event planning and management easier. It was created using the MERN (MongoDB, Express, React, and Node.js) stack. It gives clients and event planners a single location to address the issues of manual inefficiencies, inadequate coordination, and restricted accessibility that occur with conventional event management procedures.

Customers may browse and book events in real-time using the system's user-friendly interface, while organizers can easily establish, administer, and keep an eye on events. User authentication, event classification, dynamic search filters, safe payment integration, and real-time notifications are some of the main features. While MongoDB offers a scalable NoSQL database solution for storing user and event data, the backend, which is powered by Express.js and Node.js, guarantees effective handling of requests. The user experience is responsive and engaging thanks to the React.js-based frontend.

The system is thoroughly tested, including unit, integration, and performance testing, to guarantee dependability. It can handle up to 500 concurrent users with little delay, demonstrating its support for large concurrency. Scalability and availability are guaranteed when deployment takes place on cloud platforms such as AWS or Heroku.

By decreasing manual errors, increasing user happiness, and providing a solid, scalable solution for future integration with AI-based suggestions and third-party services, this system improves event booking and management efficiency. It has the potential to revolutionize event management by making it easier to use, more efficient, and more successful.

**ACKNOWLEDGEMENTS**

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. R.N. Panda** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Ishu**

**Kishan Batra**

**Deepraj Singh**

**Mayank Saini**

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Certificate | | ii |
|  | Abstract | | iii |
|  | Acknowledgements | | iv |
|  | Table of Contents | | v |
|  | List of Tables | | vii |
|  | List of Figures | | viii |
| 1 | Introduction | | 1-8 |
|  | 1.1 | Overview | 1 |
|  | 1.2 | Problem Statement | 3 |
|  | 1.3 | Objectives | 4 |
|  | 1.4 | Scope | 5 |
|  | 1.5 | Features | 6 |
|  | 1.6 | Hardware Requirements | 7 |
|  | 1.7 | Software Requirements | 8 |
| 2 | Feasibility Study | | 9-13 |
|  | 2.1 | Market Feasibility Study | 9 |
|  | 2.2 | Technical Feasibility Study | 10 |
|  | 2.3 | Financial Feasibility Study | 11 |
|  | 2.4 | Operation Feasibility Study | 12 |
|  | 2.5 | Legal Feasibility Study | 13 |
| 3 | Software Requirement Specification | | 14-19 |
|  | 3.1 Functionalities | | 14 |
|  | 3.2 User and Characteristics | | 15 |
|  | 3.3 Features of Project | | 16 |
|  | 3.4 Features of Admin | | 17 |
|  | 3.5 Features of Organizer | | 18 |
|  | 3.7 Features of Attendee | | 19 |
| 4 | Software Requirement | | 20-31 |
|  | 4.1 Functional Requirement | | 20 |
|  | 4.2 Non- Functional Requirement | | 22 |
|  | 4.3 Design Goal | | 24 |
|  | 4.4 Conclusion | | 25 |
| 5 | System Design | | 32-38 |
|  | 5.1 Architectural Design | | 32 |
|  | 5.2 Interface Design | | 33 |
|  | 5.3 Detailed Design | | 37 |
|  | 5.4 Conclusion | | 38 |
| 6 | Architecture | | 39-41 |
|  | 6.1 System Architecture | | 39 |
|  | 6.2 Technical Architecture | | 40 |
|  | 6.3 Database Architecture | | 41 |
| 7 | References | | 42-44 |
| 8 | Bibliography | | 45 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Name of Table** | **Page** |
| 1.1 | Overview of Project | 2 |
| 1.2 | Hardware Requirements | 7 |
| 1.3 | Software Requirements | 8 |
| 2.1 | Types of Feasibility | 13 |
| 3.1 | User types and their Characteristics | 15 |
| 3.2 | Features of Attendee | 19 |
| 4.1 | Functional Requirements | 21 |
| 4.2 | Non-Functional Requirements | 23 |
| 6.1 | System Architecture | 39 |
| 6.2 | Database Architecture | 41 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Figure No.** | **Name of Figure** | **Page No.** | |
| 4.1 | Use Case Diagram | | 26 | |
| 4.2 | ER Diagram | | 27 | |
| 4.3 | Class Diagram | | 28 | |
| 4.4 | Level 0 DFD | | 29 | |
| 4.5 | Level 1 DFD | | 30 | |
| 4.6 | Level 2 DFD | | 31 | |
| 5.1 | Sign Up Page | | 34 | |
| 5.2 | Sign in Page | | 34 | |
| 5.3 | Home Page | | 35 | |
| 5.4 | Callendar | | 35 | |
| 5.5 | Payment Page | | 36 | |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview**

The Event Booking and Management System (EBMS) is a dynamic web application designed to streamline the process of organizing and attending events. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js). This system provides a robust platform that caters to both event organizers and attendees, ensuring a seamless experience from event creation to ticket booking.

For organizers, the EBMS simplifies event management by allowing them to create and manage events with ease. They can input essential details such as date, time, location, capacity, and ticket pricing, significantly reducing the likelihood of errors associated with traditional manual processes.

Attendees benefit from an intuitive interface that enables easy navigation through various events. They can search for events based on their interests and securely book tickets online, eliminating the need for physical transactions. Real-time notifications keep users informed about updates or changes to their booked events.

Additionally, the system fosters community engagement by allowing attendees to leave reviews and ratings for events they have attended. Overall, the EBMS is a comprehensive solution that enhances user satisfaction while addressing the challenges of event management in today’s digital landscape.

**Overview of the Project**

|  |  |
| --- | --- |
| **Aspect** | **Description** |
| Project Name | Event Booking and Management System (EBMS) |
| Purpose | To streamline the process of organizing and attending events. |
| Technology Stack | MERN Stack (MongoDB, Express.js, React.js, Node.js) |
| Target Users | Event organizers, attendees, and administrators. |

Table 1.1 Overview of the Project

* 1. **Problem Statement**

The traditional methods of event management often lead to various challenges that hinder efficiency and user satisfaction. Key issues include:

* **Inefficient Planning:** Manual processes can lead to errors and miscommunication.
* **Limited Accessibility:** Users often struggle to find relevant events due to scattered information.
* **Complex Booking Processes:** Complicated ticketing Systems can deter potential attendees.
* **Lack of Real-Time Updates:** Users may miss important changes or notifications regarding events.
* **Bad User Experience:** Non-intuitive interface can frustrate users and reduce engagement.
  1. **Objective**

The primary objectives of Event Booking & Management System are as follows:

* To provide an intuitive platform for event organizers to create and manage events.
* To enable attendees to easily search for and book events.
* To facilities secure online payment processing for ticket purchases.
* To implement user authentication for personalized experiences.
* To allow real-time notifications for event updates.
* To support multiple event categories for diverse user interests.
* To ensure mobile responsiveness for accessibility across devices.
* To create an admin dashboard for effective oversight of events.
* To gather user feedback to improve future iterations of the system.
* To maintain high security standards for user data protection.
* To foster community engagement through event reviews and ratings.
* To provide technical support resources for users encountering issues.
  1. **Scope**

The scope of the Event Booking and Management System includes:

* Event Creation: Organizers can create detailed event listings with essential information.
* User Registration: The system supports user accounts with role-based access.
* Ticket Booking: Attendees can book tickets securely online.
* Payment Integration: The application integrates with payment gateways for transactions.
* Event Discovery: Users can browse events based on categories or dates.
* Notifications: Real-time alerts keep users informed about their events.
* Feedback Mechanism: Users can leave reviews to enhance community engagement.
* Admin Functionality: Admins can manage users and oversee all events.
  1. **Features**

The Event Booking & Management System includes several key features:

* + 1. **User-Friendly Interface:** Designed for ease of use across all devices.
    2. **Event Management Tools:** Organizers can create, edit, or delete events effortlessly.
    3. **Search Functionality:** Attendees can filter events by category or date.
    4. **Secure Payment Processing:** Integrated payment gateways ensure safe transactions.
    5. **User Reviews:** Attendees can provide feedback on their experiences at events.
    6. **Role-Based Access Control:** Different functionalities based on user roles (organizer, and attendees).
    7. **Admin Dashboard:** A centralized interface for managing users and events effectively.
  1. **Hardware Requirements**

To run the Event Booking and Management System efficiently, the following hardware requirements are recommended:

* Processor: Dual-core processor (Intel i3 or equivalent)
* Ram: Minimum 8 GB
* Storage: At least 256 GB SSD
* Network: Stable internet connection (broadband recommended)

These specifications ensure optimal performance during development and usage.

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum Requirements** | **Recommended Requirements** |
| **Processor** | 4 cores, 2.0 GHz | 8 cores, 3.0 GHZ |
| **RAM** | 8 GB | 16 GB or more |
| **Storage** | 100 GB SSD for system disk  100 GB HDD for data disk | 256 GB SSD for system disk  500 GB HDD or larger for data storage |
| **Network Card** | 1 Gbps | 10 Gbps |
| **Operating System** | Windows Server 2016 or higher  Linux (Ubuntu, CentOS) | Windows Server 2019 or higher  Latest stable version of Linux |

Table 1.2 Hardware Requirements

* 1. **Software Requirements**

The software requirements for the Event Booking and Management System include:

* Operating System: Windows 10 or later / macOS / Linux
* Node.js: Version 14.x or later
* MongoDB: Version 4.x or later
* React.js: Version 17.x or later
* Express.js: Version 4.x or later

Additionally, development tools such as Visual Studio Code or any preferred code editor should be installed to facilitate coding.

|  |  |
| --- | --- |
| **Software Component** | **Version/Requirement** |
| Operating System | Window Server 2016 or higher / Linux (Ubuntu/CentOS) |
| Database | MongoDB (latest stable version) |
| Web Server | Apache HTTP Server (if using Linux)  IIS (Internet Information Services) |
| Programming Language | Node.js (Latest stable version)  React.js (Latest stable version)  Express.js (Latest stable version) |
| Frameworks/Libraries | Bootstrap for responsive design  jQuery for DOM manipulation |
| Security Software | SSL Certificate for secure transactions |

Table 1.3 Software Requirements

**CHAPTER 2**

**FEASIBILTY STUDY**

**2.1 Market Feasibility Study**

A market feasibility study assesses the demand for your Event Booking and Management System within the target market. This study involves analysing the total addressable market (TAM), identifying target customers, understanding competition, and evaluating market trends. For an EBMS, it is essential to determine whether there is sufficient demand for such a platform among event organizers and attendees.

Key components of this study include:

* **Target Audience Analysis:** Identifying who will use the system. It is event organizers, attendees, or both.
* **Competitive Analysis:** Evaluating existing solutions in the market to understand their strengths and weaknesses.
* **Market Trends:** Investigating current trends in event management and online booking to gauge future growth potential.

By understanding these factors, stakeholders can ascertain whether there is available market for the EBMS and how it can position itself effectively against competitors.

**2.2 Technical Feasibility Study**

The technical feasibility study evaluates whether the necessary technology and resources are available to develop and implement the EBMS. This includes assessing hardware, software, and technical expertise needed for successful execution.

Key aspects include:

* **Technology Requirements:** Identifying the specific technologies required to build the application, such as databases (MongoDB), server frameworks (Node.js), and front-end libraries (React.js).
* **Infrastructure Assessment:** Evaluating whether existing infrastructure can support the new system or id upgrades are necessary.
* **Technical Expertise:** Assessing whether the development team has the skills required to implement the project effectively.

This study ensures that all technical components align with project goals and that any potential technical challenges are identified early in the process.

**2.3 Financial Feasibility Study**

A financial feasibility study analyzes the costs associated with developing and operating the EBMS compared to its projected revenues. This assessment is vital for determining whether the project is economically viable.

Key components include:

* **Cost Analysis:** Estimating development costs, operational expenses, and ongoing maintenance costs.
* **Revenue Projections:** Forecasting potential income from ticket sales, subscription fees, or advertising.
* **Return on Investment (ROI):** Evaluating whether the expected financial returns justify the initial investment.

By conducting this analysis, stakeholders can make informed decisions about funding and resource allocation for the EBMS.

**2.4 Operational Feasibility Study**

An operational feasibility study examines whether your organization has the capacity to implement and sustain the EBMS. This involves analysing existing processes, human resources, and organizational culture.

Key factors include:

* **Staffing Requirements:** Determining if additional personnel are needed for system maintenance or customer support.
* **Process Integration:** Assessing how well the new system can integrate with current operational workflows.
* **User Training Needs:** Identifying training requirements for staff and users to ensure smooth adoption of the system.

This study helps ensure that operational capabilities align with project goals and that any potential disruptions to existing operational are minimized.

**2.5 Legal Feasibility Study**

A legal feasibility study identifies any legal issues that may affect your project. For an EBMS, this includes compliance with laws related to data protection, licensing requirements, and intellectual property rights.

Key considerations include:

* **Data Privacy Regulations:** Ensuring compliances with laws such as GDPR (General Data Protection Regulation) or CCPA (Central Consumer Protection Regulation) regarding user data handling.
* **Licensing Requirements:** Verifying that all necessary licenses for software use are obtained.
* **Zoning Laws:** If applicable, checking local regulations regarding business operations in specific areas.

This study ensures that your project adheres to all legal requirements, reducing risks associated with non-compliance.

|  |  |
| --- | --- |
| **Types of Feasibility** | **Key Findings** |
| Market Feasibility | Sufficient demand identified among target users for an integrated event management system. |
| Technical Feasibility | The technology stack is suitable for building a scalable and maintainable application. |
| Financial Feasibility | Project costs are justified by projected revenue from ticket sales and subscriptions. |
| Operational Feasibility | Existing processes can be integrated with minimal disruption to current operations. |
| Legal Feasibility | Compliance with data protection regulations is feasible with appropriate measures in place. |

Table 2.1 Types of Feasibility

**CHAPTER 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Functionalities**

The functionalities of the Event Booking and Management System (EBMS) encompass a wide range of operations that facilitate event management for organizers, attendees, and administrators. At its core, the system allows event organizers to create and manage events efficiently. This includes functionalities such as setting event details (date, time, location), managing ticket types and pricing, and tracking ticket sales. Organizers can also edit or cancel events as needed.

For attendees, the EBMS provides functionalities that enable easy browsing of available events. Users can search for events based on various criteria, such as date, type, or location. The booking process is streamlined through an intuitive interface that allows users to select tickets, enter personal information, and make secure payments. Additionally, attendees can receive notifications about their bookings and updates related to the events they are interested in.

The system also includes administrative functionalities that allow for user management, reporting, and oversight of all events. Admins can monitor ticket sales, manage user accounts, and ensure compliance with data protection regulations. Overall, the functionalities of the EBMS are designed to enhance user experience while ensuring efficient event management.

**3.2 User and Characteristics**

The EBMS will cater to three primary user groups: event organizers, attendees, and administrators. Each user group has distinct characteristics and requirements that influence the design and functionality of the system.

Event Organizers are typically individuals or organizations responsible for planning and executing events. They require a robust set of tools to create event listings, manage registrations, track sales, and communicate with attendees. Their characteristics include a need for efficiency in managing multiple events simultaneously and the ability to analyse attendance data to improve future events.

Attendees are individuals looking to participate in various events. They seek a user-friendly interface that allows them to easily search for events, book tickets, and receive timely notifications about their bookings. Their characteristics include a preference for convenience in the booking process and access to information about event details.

Administrators are responsible for overseeing the entire system's operation. They require access to comprehensive reporting tools to monitor system performance and user activity. Their characteristics include a focus on data security, user management capabilities, and ensuring compliance with legal requirements.

|  |  |
| --- | --- |
| User Type | Characteristics |
| Event Organizers | Require tools for efficient event management; need analytics on attendance; focus on usability. |
| Attendees | Seek a user-friendly interface; prefer convenience in booking; value timely notifications. |
| Administrators | Seek a user-friendly interface; prefer convenience in booking; value timely notifications. |

Table 3.1 User Types and their characteristics

**3.3 Features of Project**

The EBMS is equipped with several key features that enhance its functionality and usability:

* **Event Creation**: Organizers can create detailed event listings with essential information.
* **User Registration**: Both organizers and attendees can create accounts for personalized experiences.
* **Search Functionality**: Attendees can filter events by date, location, or category.
* **Secure Payment Processing**: The system integrates with payment gateways for safe transactions.
* **Real-Time Notifications**: Users receive alerts regarding event updates or changes.
* **Feedback Mechanism**: Attendees can leave reviews and ratings for events.
* **Admin Dashboard**: A centralized interface for managing users and monitoring events.
* **Mobile Responsiveness**: The application is designed to function seamlessly on various devices.

These features collectively ensure that the EBMS meets the needs of all users while providing a comprehensive solution for event management.

**3.4 Features of Admin**

The administrative features of the EBMS are crucial for maintaining control over the platform's operations:

* **User Management**: Admins can create, edit, or delete user accounts as necessary.
* **Event Oversight**: Administrators have access to all event listings and can manage them effectively.
* **Reporting Tools**: The system provides analytics on ticket sales, user engagement, and overall performance.
* **Access Control**: Admins can set permissions for different user roles within the system.
* **Data Security Management**: Ensuring compliance with data protection regulations is a priority.
* **Communication Tools**: Admins can send bulk notifications or updates to users regarding events.
* **Issue Resolution**: The admin panel includes tools for addressing user inquiries or problems.

These features empower administrators to ensure smooth operation while maintaining high standards of service delivery.

* 1. **Features of Organizer**

Event organizers have access to specialized features tailored to their needs:

* **Event Creation Wizard**: A guided process helps organizers set up new events quickly.
* **Ticket Management System**: Organizers can create multiple ticket types with different pricing structures.
* **Sales Tracking Tools**: Real-time analytics provide insights into ticket sales performance.
* **Communication Tools**: Organizers can send updates or messages directly to attendees.
* **Calendar Integration**: Syncing with external calendars helps manage scheduling effectively.
* **Customizable Event Pages**: Organizers can personalize event pages with branding elements.
* **Reporting Features**: Detailed reports on attendee demographics and preferences help optimize future events.

These features empower organizers to manage their events efficiently while enhancing attendee engagement.

* 1. **Features of Attendee**

Attendees benefit from a range of features designed to enhance their experience:

* **User-Friendly Interface**: A simple layout allows easy navigation through available events.
* **Event Discovery Tools**: Attendees can search for events based on various filters.
* **Secure Ticket Booking**: The booking process is straightforward and secure.
* **Booking History Access**: Users can view their past bookings for reference.
* **Real-Time Notifications**: Attendees receive updates about their bookings directly through the platform.
* **Review System**: Users can provide feedback on their experiences at events they attended.
* **Profile Management**: Attendees can manage their personal information within their accounts.

These features ensure that attendees have a seamless experience when interacting with the EBMS.

|  |  |
| --- | --- |
| Feature | Description |
| Event Creation | Organizers can create detailed event listings with essential information. |
| User Registration | Both organizers and attendees can create accounts for personalized experiences. |
| Search Functionality | Attendees can filter events by date, location, or category easily. |
| Secure Payment Processing | Integration with payment gateways ensures safe transactions during ticket booking. |
| Admin Dashboard | A centralized interface for managing users and monitoring all events efficiently. |

Table 3.2 Features of Attendee

**CHAPTER 4**

**SOFTWARE REQUIREMENT**

**4.1 Functional Requirement**

Functional requirements define the specific behaviours and functionalities that the Event Booking and Management System (EBMS) must exhibit. These requirements are essential for ensuring that the system meets user needs and performs its intended tasks effectively. For the EBMS, key functional requirements include:

* **User Authentication**: The system must allow users to create accounts, log in, and manage their profiles securely. This includes features for password recovery and account verification.
* **Event Management**: Organizers should be able to create, edit, and delete events. This includes specifying event details such as date, time, location, capacity, and ticket pricing.
* **Ticket Booking**: Attendees must have the ability to search for events based on various criteria (e.g., date, type, location) and book tickets online. The system should handle different ticket types and pricing structures.
* **Payment Processing**: The EBMS must integrate with secure payment gateways to facilitate online transactions. Users should receive confirmations of their purchases via email.
* **Admin Dashboard**: Administrators need a centralized interface to manage users, monitor event activity, generate reports, and ensure compliance with regulations.

These functional requirements ensure that the EBMS provides a comprehensive solution for managing events efficiently while enhancing user experience.

|  |  |
| --- | --- |
| Requirement | Description |
| User Authentication | Users can create accounts, log in, and manage profiles securely. |
| Event Management | Organizers can create, edit, and delete events with necessary details. |
| Ticket Booking | Attendees can search for events and book tickets online. |
| Payment Processing | Integration with secure payment gateways for online transactions. |
| Admin Dashboard | Centralized interface for managing users and monitoring event activity. |

Table 4.1 Functional Requirements

**4.2 Non-Functional Requirement**

Non-functional requirements specify the quality attributes of the EBMS that are crucial for its overall performance and user satisfaction. These requirements focus on how the system performs its functions rather than what functions it performs. Key non-functional requirements include:

* **Usability**: The system should have an intuitive user interface that allows users to navigate easily without extensive training. A responsive design is essential for accessibility across devices.
* **Performance**: The EBMS must handle a high volume of concurrent users without significant delays. Load times should be optimized to ensure a smooth user experience during peak usage times.
* **Scalability**: The architecture of the system should support future growth, allowing for an increase in users, events, and data without compromising performance.
* **Security**: User data must be protected through robust security measures, including data encryption during transmission and secure storage practices. Compliance with data protection regulations (e.g., GDPR) is mandatory.
* **Availability**: The system should maintain high availability with minimal downtime. Scheduled maintenance should be communicated to users to minimize disruption.
* **Backup and Recovery**: Regular data backups must be implemented to prevent data loss. The system should have a recovery plan in place to restore functionality quickly in case of failures.

These non-functional requirements are essential for ensuring that the EBMS operates effectively and provides a reliable service to users.

|  |  |
| --- | --- |
| Requirement | Description |
| Usability | Intuitive user interface for easy navigation across devices. |
| Performance | System must handle high traffic volumes with minimal delays. |
| Scalability | Architecture should support future growth in users and events without performance degradation. |
| Security | Robust measures to protect user data and ensure compliance with regulations. |
| Availability | High availability with minimal downtime during operation. |
| Backup and Recovery | Regular data backups and a recovery plan in place to prevent data loss. |

Table 4.2 Non-Functional Requirements

**4.3 Design Goal**

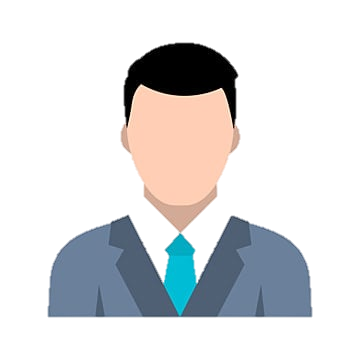
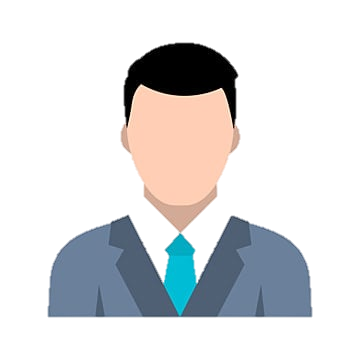
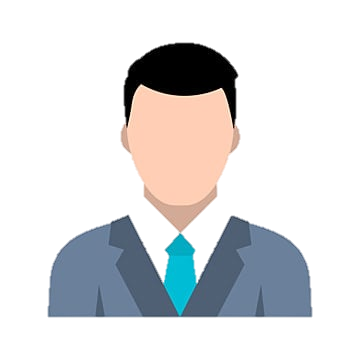
The design goals of the EBMS guide the overall architecture and user interface of the application. These goals are critical for achieving both functional and non-functional requirements while ensuring a positive user experience:

* **Modularity**: The system architecture should be modular, allowing for easy updates and maintenance of individual components without affecting the entire system.
* **User-Centric Design**: The interface should prioritize user experience by being intuitive and visually appealing. User feedback should be incorporated into design iterations to enhance usability.
* **Integration Capabilities**: The EBMS should support integration with third-party services such as payment gateways, email marketing tools, and calendar applications (e.g., Google Calendar) to streamline operations.
* **Responsive Design**: The application must be fully responsive to ensure compatibility across various devices, including desktops, tablets, and smartphones.
* **Performance Optimization**: Design choices should focus on optimizing performance through efficient coding practices, minimizing load times, and ensuring quick response times during interactions.
* **Accessibility Compliance**: The design must adhere to accessibility standards (e.g., WCAG) to ensure that all users, including those with disabilities, can effectively use the platform.

By adhering to these design goals, the EBMS aims to create a robust application that meets user needs while maintaining high standards of performance and usability.

**4.4 Conclusion**

In summary, the Software Requirement Specification for the Event Booking and Management System outlines critical functional and non-functional requirements alongside design goals essential for successful implementation. By focusing on these aspects, stakeholders can ensure that the system not only meets user expectations but also operates efficiently in a competitive market environment. This comprehensive approach will facilitate effective event management while enhancing overall user satisfaction within the platform.

**Use Case Diagram**

Admin

User

Event Organizer

Event Booking & Management System

Fig 4.1 Use Case Diagram

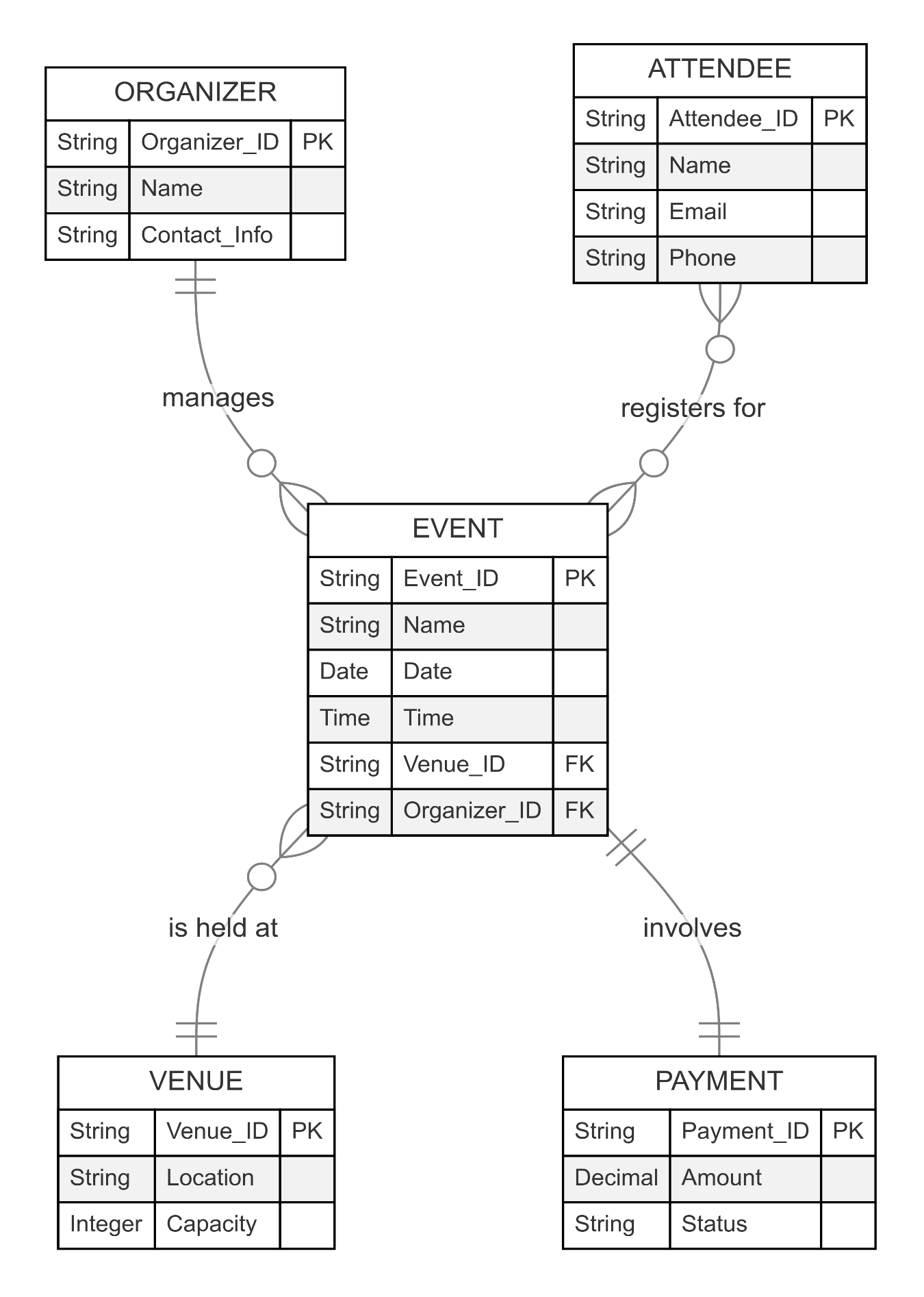
**ER Diagram**

Fig 4.2 ER Diagram

**Class Diagram**

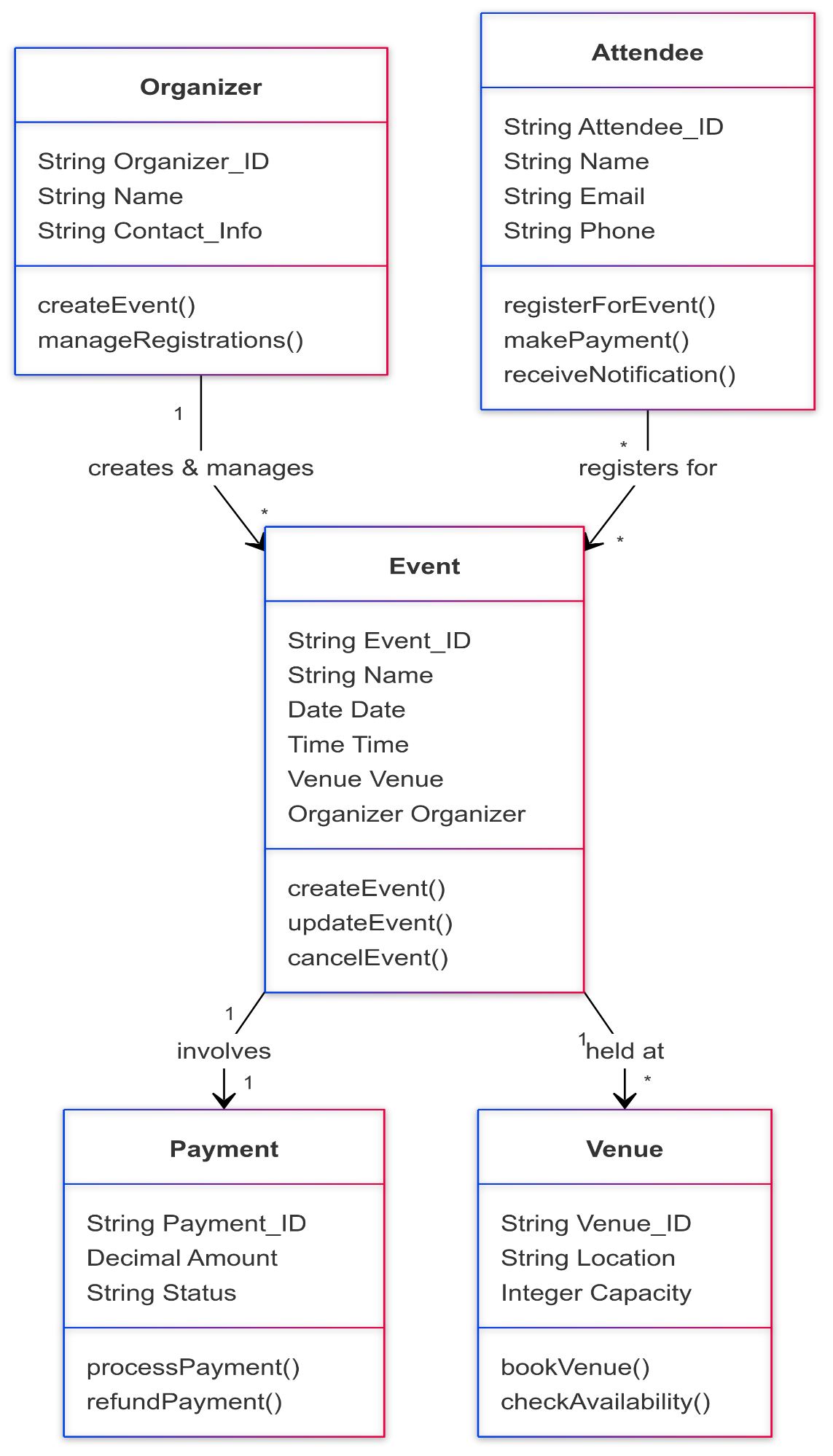
****

Fig 4.3 Class Diagram

**Level 0 DFD**

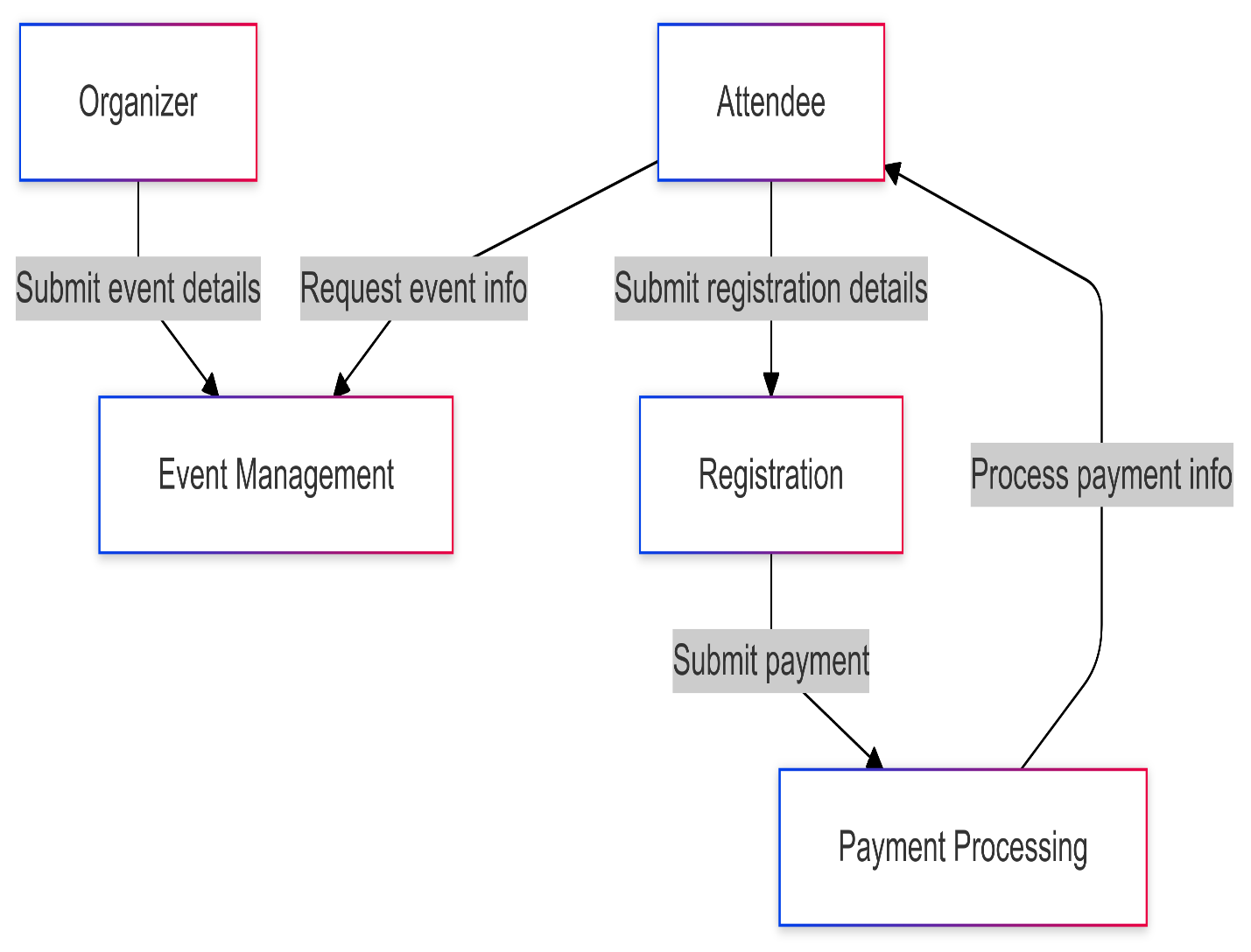
****

Fig 4.4 Level 0 Data Flow Diagram

**Level 1 DFD**

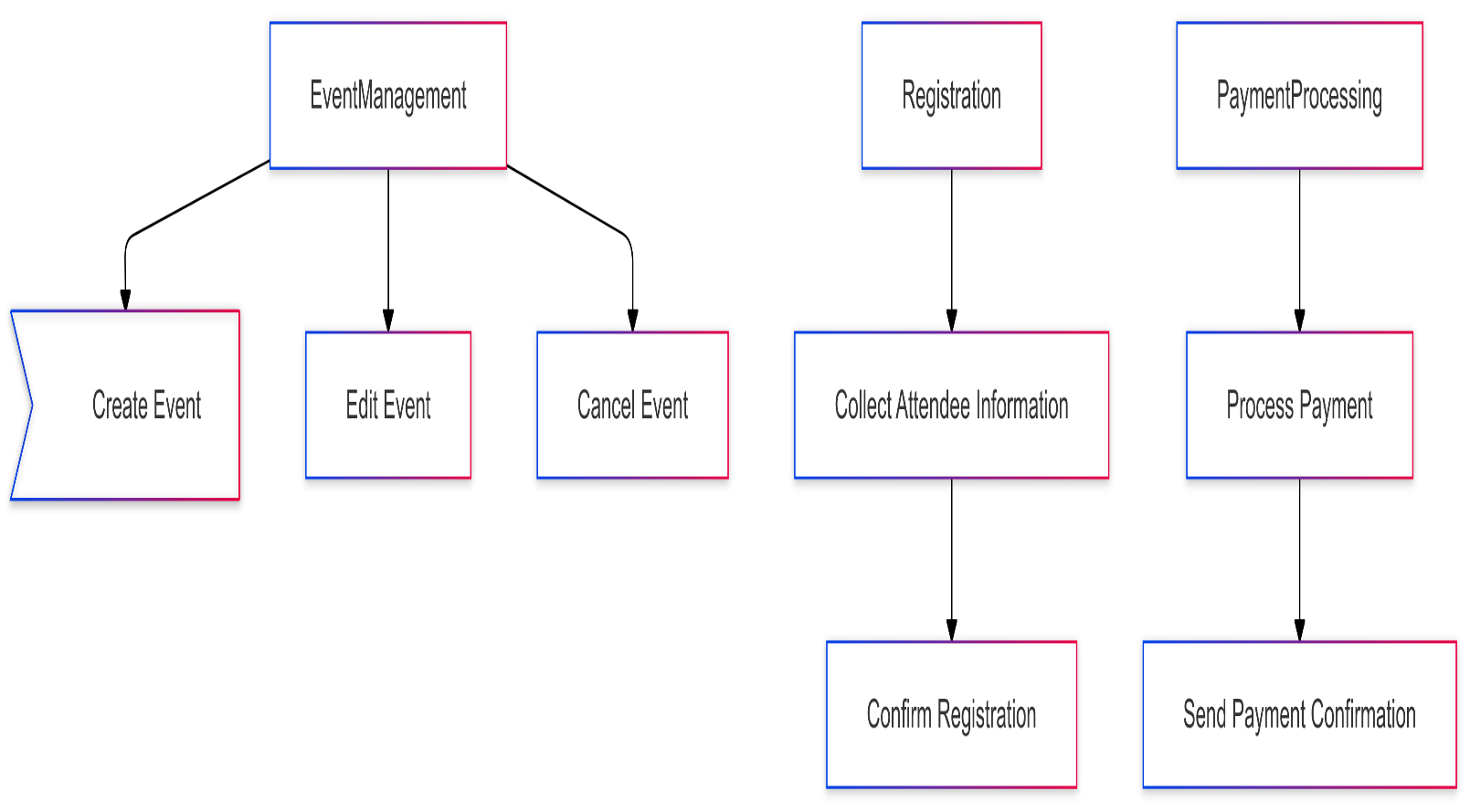
****

Fig 4.5 Level 1 Data Flow Diagram

**Level 2 DFD**

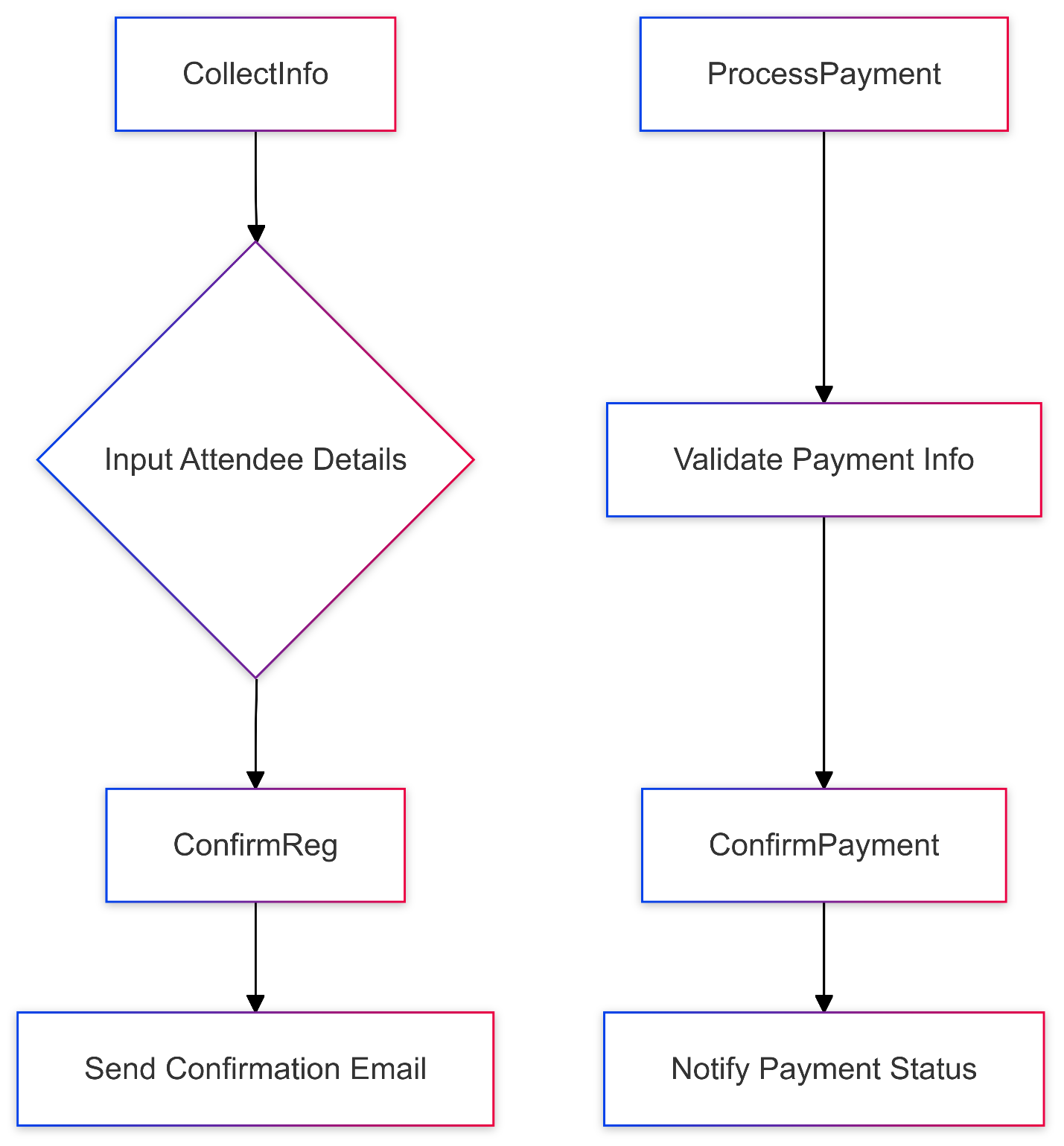
****

Fig 4.5 Level 2 Data Flow Diagram

**CHAPTER 5**

**SYSTEM DESIGN**

The design phase of the software development life cycle (SDLC) is crucial for transforming requirements into a structured blueprint that guides the development process. For an Event Booking and Management System (EBMS), this phase can be broken down into three primary types of design: **Architectural Design**, **Interface Design**, and **Detailed Design**. Each type plays a vital role in ensuring that the system is built efficiently and meets user expectations.

**5.1 Architectural Design**

Architectural design serves as the foundation of the EBMS, outlining the overall structure and high-level components of the system. This phase involves defining how different modules of the application will interact with each other and with external systems. For instance, in an EBMS, key components may include user authentication, event management, ticket booking, and payment processing modules.

The architectural design also specifies the technology stack to be used, such as the MERN stack (MongoDB, Express.js, React.js, Node.js). This choice influences performance, scalability, and maintainability. The design must consider how data flows between components and how they communicate through APIs. Additionally, it addresses concerns such as security protocols and data storage solutions to protect sensitive user information.

A well-defined architectural design helps in identifying potential risks early in the development process. It allows for modular development, where individual components can be developed and tested independently before integration. This modularity not only simplifies debugging but also enhances collaboration among development teams.

**5.2 Interface Design**

Interface design focuses on creating an intuitive and user-friendly experience for both event organizers and attendees using the EBMS. This includes designing both the graphical user interface (GUI) and the user experience (UX). The goal is to ensure that users can navigate the system easily and accomplish their tasks without confusion.

During this phase, wireframes and prototypes are often created to visualize how different screens will look and function. For example, the event listing page should allow attendees to filter events by date or category easily. Similarly, organizers need a straightforward interface to create and manage events efficiently.

Usability testing is a critical aspect of interface design. By gathering feedback from potential users during the prototyping stage, designers can make necessary adjustments before final implementation. This iterative process ensures that the final product meets user needs and preferences while maintaining aesthetic appeal.

Moreover, accessibility considerations are essential in interface design to accommodate users with disabilities. Implementing features like keyboard navigation and screen reader compatibility ensures inclusivity, making the EBMS accessible to a broader audience.

**A screenshot of a computer

Description automatically generated**

Fig 5.1 Sign up Page

**A screenshot of a login screen

Description automatically generated**

Fig 5.1 Sign in Page

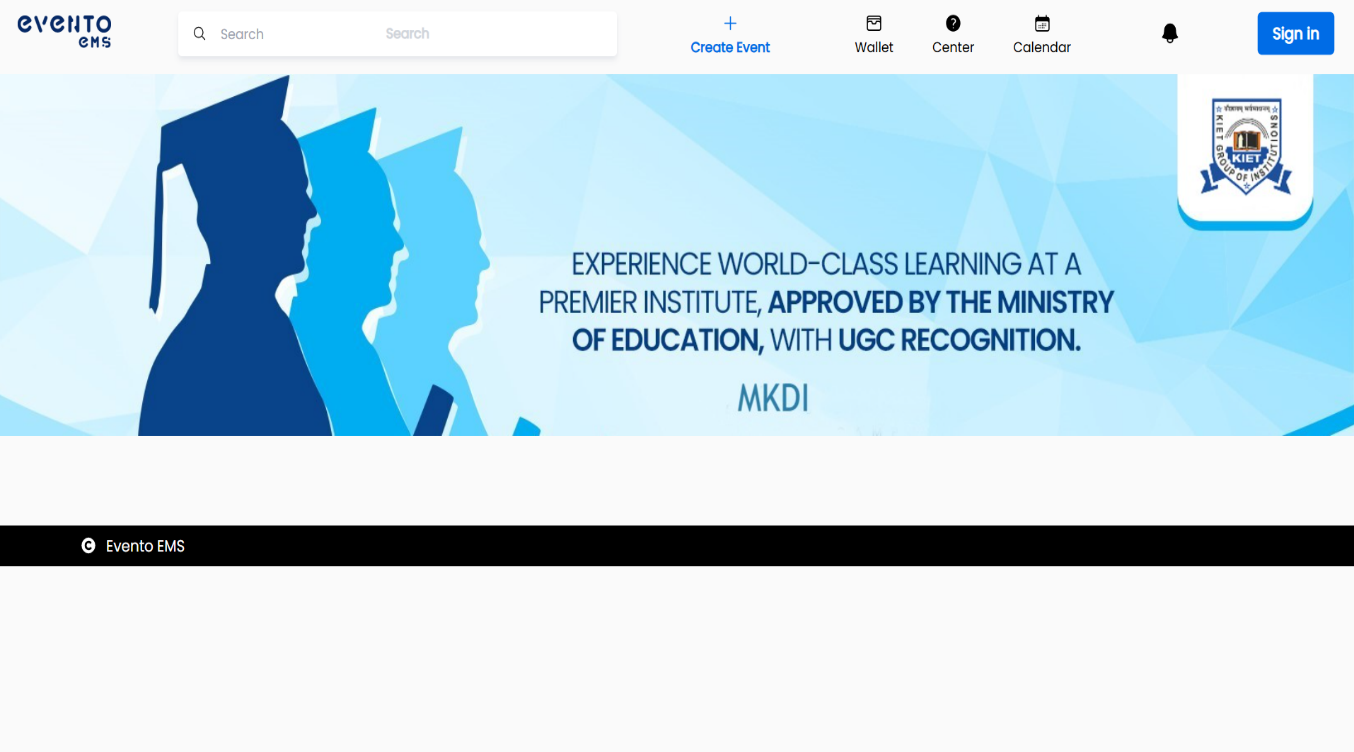
****

Fig 5.3 Home Page

**A screenshot of a calendar

Description automatically generated**

Fig 5.4 Callendar

**A screenshot of a computer

Description automatically generated**

Fig 5.5 Payment Page

**5.3 Detailed Design**

The detailed design phase translates high-level architectural and interface designs into specific implementation details that developers will use during coding. This includes defining database schemas, specifying data types, outlining algorithms for business logic, and creating detailed specifications for each module.

For the EBMS, detailed design might involve creating database tables for users, events, tickets, and transactions. Each table would have defined relationships to ensure data integrity and efficient querying. Additionally, this phase includes outlining API endpoints that will facilitate communication between front-end components (built with React.js) and back-end services (using Node.js).

This level of detail is crucial for developers as it provides clear guidelines on how to implement each feature of the system accurately. It reduces ambiguity by specifying how different parts of the application should behave under various conditions.

Furthermore, detailed design should also consider performance optimization techniques such as caching strategies or load balancing methods to ensure that the EBMS can handle high traffic volumes during peak booking times.

**5.4 Design Conclusion**

In summary, the design phase encompasses architectural design, interface design, and detailed design each contributing uniquely to building a robust Event Booking and Management System. By focusing on these aspects during the design phase, developers can create a system that is not only functional but also user-friendly and scalable. This comprehensive approach ensures that all requirements are met effectively while laying a solid foundation for successful implementation in subsequent phases of development.

**CHAPTER 6**

**ARCHITECTURE**

**6.1 System Architecture**

The architecture of the Event Booking and Management System (EBMS) is designed to provide a robust framework that supports various functionalities required for managing events efficiently. The system architecture can be categorized into three primary layers: Presentation Layer, Business Logic Layer, and Data Access Layer.

The Presentation Layer is responsible for the user interface, where users interact with the system. This layer is built using React.js, providing a responsive and dynamic user experience. It allows event organizers, attendees, and administrators to access the system's features seamlessly. The design prioritizes usability, ensuring that users can easily navigate through event listings, booking forms, and account management options.

The Business Logic Layer handles all the core functionalities of the EBMS. This layer is implemented using Node.js and Express.js, which manage the application’s routing, API endpoints, and business rules. Here, the system processes requests from the presentation layer, performs necessary computations, and interacts with the data access layer to retrieve or store information. This separation of concerns enhances maintainability and scalability, allowing developers to update business logic without affecting the user interface.

Finally, the Data Access Layer interacts with the database (MongoDB) to manage data storage and retrieval. This layer abstracts database operations from the business logic, providing a clean interface for data manipulation. It ensures that all user data, event details, and transaction records are securely stored and efficiently accessed.

|  |  |  |
| --- | --- | --- |
| Layer | Description | Technologies Used |
| Presentation Layer | User interface for interaction with the system. | React.js |
| Business Logic Layer | Handles core functionalities and business rules. | Node.js, Express.js |
| Data Access Layer | Manages data storage and retrieval operations. | MongoDB |

Table 6.1 System Architecture

**6.2 Technical Architecture**

The technical architecture of the EBMS outlines how various technologies are integrated to support its functionalities. The system employs a microservices architecture that allows for independent deployment of different components such as user management, event management, and payment processing.

Each microservice communicates over RESTful APIs, enabling scalability and flexibility in development. For instance, the payment processing service can be updated or replaced without impacting other services. This architecture also supports load balancing, ensuring that multiple requests can be handled simultaneously without performance degradation.

To enhance security, the technical architecture incorporates authentication mechanisms such as JSON Web Tokens (JWT) for secure user sessions. Data encryption is implemented for sensitive information during transmission between the client and server. Additionally, regular security audits are conducted to identify vulnerabilities and ensure compliance with data protection regulations.

Furthermore, cloud services may be utilized for hosting the application to provide scalability and reliability. Services like AWS or Azure can offer resources on-demand based on traffic fluctuations, ensuring that users have uninterrupted access to the EBMS.

**6.3 Database Architecture**

The database architecture of the EBMS is critical for managing all data related to users, events, bookings, and transactions. A centralized MongoDB database is employed to store structured and unstructured data efficiently.

The Database schema consists of several key collection: **Users, Events, Tickets, Payment,** and **Feedback**. Each collection is designed with specific fields to capture essential information. For example, the Users collection includes fields such as user ID, name, email address, role (admin, organizer, attendee), and hashed passwords for security purpose.

The Events collection contains details about each event such as event ID, name, description, date/time, location, capacity, and ticket pricing. The Tickets collection links attendees to their respective events by storing ticket IDs along with associated user IDs and event IDs.

|  |  |
| --- | --- |
| Collection | Key Fields |
| Users | userID, name, email, role (admin, organizer, attendee), hashed password |
| Events | eventID, name, description, date/time, location, capacity, ticket pricing |
| Tickets | ticketID, userID (attendee), eventID |
| Payments | paymentID, userID, ticketID, amount |

To optimize performance and ensure quick access to frequently queried data, indexes are created on critical fields such as event date and user email addresses. This design choice enhances query efficiency while maintaining data integrity through proper relationships between collections.

Table 6.2 Database Architecture

**CHAPTER 7**

**REFERENCES**

* Acharya, Kamal. "STUDENT INFORMATION MANAGEMENT SYSTEM." Authorea Preprints (2023).
* Acharya, Kamal. "Library Management System." Available at SSRN4807104 (2019). ACHARYA, KAMAL, et al. "LIBRARY MANAGEMENT SYSTEM." (2019).
* Acharya, Kamal. "Online bus reservation system project report." Authorea Preprints (2024).
* Acharya, Kamal. "Online bus reservation system project report." (2024).
* Acharya, Kamal. “Online Bus Reservation System.” SSRN ElectroNIC ASIA Journal (2024)
* Acharya, Kamal. “Student Information Management System Project.” SSRN ElectroNIC ASIA Journal (2024): n. pag.
* Acharya, Kamal. “ATTENDANCE MANAGEMENT SYSTEM.” International Research Journal of Modernization in Engineering Technology and Science (2023)
* Acharya, Kamal. “College Information Management System.” SSRN ElectroNIC ASIA Journal (2024)
* Acharya, Kamal, Attendance Management System Project (April 28, 2024). Available at SSRN: https://ssrn.com/abstract=4810251 or http://dx.doi.org/10.2139/ssrn.4810251
* Acharya, Kamal, Online Food Order System (May 2, 2024). Available at SSRN: https://ssrn.com/abstract=4814732 or http://dx.doi.org/10.2139/ssrn.4814732
* Acharya, Kamal, University management system project. (May 1, 2024). Available at SSRN: https://ssrn.com/abstract=4814103 or http://dx.doi.org/10.2139/ssrn.4814103
* Acharya, Kamal, Online banking management system. (May 1, 2024). Available at SSRN: https://ssrn.com/abstract=4813597 or http://dx.doi.org/10.2139/ssrn.4813597
* Acharya, Kamal, Online Job Portal Management System (May 5, 2024). Available at SSRN: https://ssrn.com/abstract=4817534 or http://dx.doi.org/10.2139/ssrn.4817534
* Acharya, Kamal, Employee leave management system. (May 7, 2024). Available at SSRN: https://ssrn.com/abstract=4819626 or http://dx.doi.org/10.2139/ssrn.4819626
* Acharya, Kamal, Online electricity billing project report. (May 7, 2024). Available at SSRN: https://ssrn.com/abstract=4819630 or http://dx.doi.org/10.2139/ssrn.4819630
* Acharya, Kamal, POLICY MANAGEMENT SYSTEM PROJECT REPORT. (December 10, 2023). Available at SSRN: https://ssrn.com/abstract=4831694 or http://dx.doi.org/10.2139/ssrn.4831694
* Acharya, Kamal, Online job placement system project report. (January 10, 2023). Available at SSRN: https://ssrn.com/abstract=4831638 or http://dx.doi.org/10.2139/ssrn.4831638
* Acharya, Kamal, Software testing for project report. (May 16, 2023). Available at SSRN: https://ssrn.com/abstract=4831028 or http://dx.doi.org/10.2139/ssrn.4831028
* Acharya, Kamal, ONLINE CRIME REPORTING SYSTEM PROJECT. (August 10, 2022). Available at SSRN: https://ssrn.com/abstract=4831015 or http://dx.doi.org/10.2139/ssrn.4831015
* Acharya, Kamal, Burber ordering system project report. (October 10, 2022). Available at SSRN: https://ssrn.com/abstract=4832704 or http://dx.doi.org/10.2139/ssrn.4832704
* Acharya, Kamal, Teachers Record Management System Project Report (December 10, 2019). Available at SSRN: https://ssrn.com/abstract=4835238 or http://dx.doi.org/10.2139/ssrn.4835238
* Acharya, Kamal, Online book store management system project report. (February 10, 2020). Available at SSRN: https://ssrn.com/abstract=4835277 or http://dx.doi.org/10.2139/ssrn.4835277
* Acharya, Kamal, Paint shop management system project report. (January 10, 2019). Available at SSRN: https://ssrn.com/abstract=4835441 or http://dx.doi.org/10.2139/ssrn.4835441
* Acharya, Kamal, Supermarket billing system project report. (August 10, 2021). Available at SSRN: https://ssrn.com/abstract=4835474 or http://dx.doi.org/10.2139/ssrn.4835474
* Acharya, Kamal, Online texi booking system project report. (March 10, 2022). Available at SSRN: https://ssrn.com/abstract=4837729 or http://dx.doi.org/10.2139/ssrn.4837729
* Acharya, Kamal, Online car servicing system project report. (March 10, 2023). Available at SSRN: https://ssrn.com/abstract=4837832 or http://dx.doi.org/10.2139/ssrn.4837832
* Acharya, Kamal, School management system project report. (July 10, 2021). Available at SSRN: https://ssrn.com/abstract=4837837 or http://dx.doi.org/10.2139/ssrn.4837837
* Acharya, Kamal, Furniture Showroom Management System Project Report (March 21, 2021). Available at SSRN: https://ssrn.com/abstract=4839422 or http://dx.doi.org/10.2139/ssrn.4839422

**BIBLIOGRAPHY**

1. Visual Basic - Jamew Foxall

2. Fundamentals Of Software Engineering - Rajib Mall

3. Software Engineering - K.K. Aggarwal & Yogesh Singh

4. Visual Basic Projects - Tony Martin, Dominic Selly

5. Visual Basic Professional Projects - Kuljit Kaur, Pooja Bembey

6. Using SQL-SERVER 2000, Microsoft publication

7. SQL-SERVER 2000 BPB publication