

EVENT FINDER

Project report submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

Submitted by

ANNE CARMEL

(2338M0108)

Under the Supervision and Guidance of

Dr.K.YEMUNARANE M.Sc., M.Phil., M.Sc(App.Psyc), Ph.D.,

ASSISTANT PROFESSOR



DEPARTMENT OF COMPUTER SCIENCE

KGiSL INSTITUTE OF INFORMATION MANAGEMENT

Approved by AICTE, New Delhi and Affiliated to Bharathiar University

ISO 9001:2015 Certified Institution

KGiSL Campus, Saravanampatti, Coimbatore – 641 035

MARCH 2025

DECLARATION

I hereby declare that this project work titled “**EVENT FINDER**”, submitted to the Department of Computer Science, KGiSL Institute of Information Management, Saravanampatti, Coimbatore is a record of original work done by me under the supervision and guidance of **Dr. K. Yemunarane M.Sc.,M.Phil.,M.Sc(App.Psyc.),Ph.D.**, Assistant Professor and that this project work has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or similar title to any candidate of any University.

PLACE:

DATE:

(Anne Carmel)

CERTIFICATE

This is to certify that the project work titled **EVENT FINDER** submitted to Bharathiar University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a record of original work done by **ANNE CAMEL, 2338M0108** under my supervision and guidance and that this project work has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or similar title to any candidate of any University.

SIGNATURE OF THE GUIDE

SIGNATURE OF THE HOD

(College Seal)

Submitted for the Viva-Voce Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINE

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deep sense of gratitude to all who helped me directly or indirectly during this project work.

I express my sincere thanks to **Dr. Ashok Bakthavathsalam, B.E, M.S., Ph.D.,** Managing Director, for giving me an opportunity to do this course of study and to undertake this project work.

I take this opportunity to convey my sincere thanks to **Dr. T. S. Prabhu M.Sc., M.Phil., Ph.D,** Director, KGiSL Institute of Information Management for his moral support throughout the course.

I have great pleasure in acknowledging my thanks to **Mr. Alwin Pinakas James M.Sc., M.Phil., (Ph.D).,** Head, Department of Computer Science, KGiSL Institute of Information Management for his encouragement and help throughout the course.

I would like to thank my supervisor, **Dr.K.YEMUNARANE, M.Sc., M.Phil., M.Sc (App.Psyc), Ph.D.,** Assistant Professor, Department of Computer Science, KGiSL Institute of Information Management for being a great mentor and the best adviser I could ever have. Her advise, encouragement and critics are source of innovative ideas, inspiration and causes behind the successful completion of this dissertation.

Last but not the least, I express my thanks to my parents and friends who have kindly provided necessary support for successful completion of the project.

- **Anne Carmel**

CONTENTS

S.No	TITLE	PG.NO
1	INTRODUCTION	01
	1.1 Salient Features	01
2	SYSTEM STUDY AND ANALYSIS	03
	2.1 Problem Statement	03
	2.2 Existing System	04
	2.2.1 Drawback	05
	2.3 Proposed System	07
	2.3.1 Advantages	07
	2.4 Feasibility Analysis	09
	2.4.1 Economic Feasibility	09
	2.4.2 Technical Feasibility	10
	2.4.3 Operational Feasibility	10
3	DEVELOPMENT ENVIRONMENT	12
	3.1 Hardware Requirements	12
	3.2 Software Requirements	12
	3.3 Programming Environment	13
	3.3.1 JavaScript	13
	3.3.2 NodeJS	13
	3.3.3 PostgreSQL	13
4	SYSTEM DESIGN	19
	4.1 Module Description	19
	4.2 Input Design	22
	4.3 Output Design	28
	4.4 Database Design	30
	4.5 Data Flow Diagram (DFD)	36
5	SYSTEM TESTING AND IMPLEMENTATION	41
	5.1 System Testing	
	5.1.1 Unit Testing	41

5.1.2	Integration Testing	41
5.1.3	Functional Testing	42
5.1.4	User Interface (UI) Testing	42
5.1.5	Performance Testing	43
5.1.6	Security Testing	43
5.1.7	Acceptance Testing	43
5.1.8	Test Cases	44
5.2	IMPLEMENTATION AND MAINTENANCE	47
5.2.1	Implementation	47
(1)	Implementation Strategies	48
(2)	Steps in Implementation	48
5.2.2	Maintenance	49
(1)	Types of Maintenance	50
(2)	Maintenance Activities	50
(3)	Future Enhancements	51
6	CONCLUSION & RECOMMENDATIONS	53
6.1	Conclusions	53
6.2	Future Enhancement	54
	BIBLIOGRAPHY AND REFERENCES	57
	APPENDICES	58

ABSTRACT

In today's digital era, event management platforms are essential for connecting people with events of interest. The proposed, "Event Finder", aims to build a comprehensive full-stack platform for users to discover and participate in events happening around the globe, ranging from concerts and sports meets to smaller, niche gatherings like BTS Army fan meets or political rallies. The platform caters to both event organizers and participants, providing an intuitive interface for event discovery, booking, and navigation. The system is designed to offer a seamless experience to users by allowing them to browse a wide range of events based on their preferences, location, and interests. Organizers can post, update, and manage events, ensuring accurate details such as venue, time, event description, and category (paid or free). For free events, users can freely access event details and navigate to the location, whereas paid events require booking and payment via the platform.

This system is integrated with real-time map navigation, providing users with an automated route to the event location upon booking or choosing to attend. To ensure the credibility of event organizers and provide a secure environment for participants, the platform implements a verification process. For paid events, organizers must provide valid identification and authorization proof, ensuring that only legitimate events are posted. In contrast, free events only require basic information from the organizer. This mechanism ensures that users can trust the platform for both free and paid event discovery. The platform leverages modern technologies, including React for the frontend, Node.js and Express for the backend, and MongoDB for the database. Additionally, it uses Google Maps API for location services and navigation, which can be inherited to any vehicle infotainment system that the user choose, making the platform user-friendly and highly interactive. A key feature of the platform is its dynamic event display system, where users can view upcoming events, get detailed information by hovering over or clicking on an event, and book tickets directly through the platform. For events that are no longer available or have concluded, the system applies a fading effect & disappears, to indicate that the event has ended. This ensures that the user interface is both informative and visually appealing. As the user interacts with the platform, notifications are provided for events that are booked or about to occur. A timer will count down the days until the event begins, offering users reminders and updates. When users approach the event date, the map route will automatically open to guide them to the venue, adding convenience and enhancing the overall user experience.

CHAPTER 1

INTRODUCTION

In today's fast-paced world, staying informed about local events can be challenging. Whether it's concerts, sports matches, conferences, or cultural festivals, people often miss out on events due to a lack of timely updates and proper event discovery platforms. The Event Finder application is designed to bridge this gap by providing a real-time event discovery system that helps users find and attend events effortlessly.

This system integrates Google Maps for seamless navigation, real-time updates, event booking, and an interactive calendar to track past and upcoming events. With a user-friendly interface and AI-powered recommendations, the application personalizes event suggestions based on user preferences. Additionally, event organizers can efficiently manage listings, reach a wider audience, and boost attendance through the platform.

1.1 SALIENT FEATURES

The "Event Finder" is a web and mobile application designed to help users discover ongoing and upcoming local events. It provides live updates, event navigation through Google Maps, booking features, and a calendar to track past and scheduled events.

The Event Finder application is designed to enhance the event discovery experience by providing users with real-time updates, seamless navigation, and a user-friendly booking system. It ensures that users can easily locate ongoing events, receive live updates, and navigate effortlessly using Google Maps. The platform also offers a robust event management system for organizers, enabling them to reach a wider audience. Below are the key features of the application:

1. Real-time Event Discovery:

- Displays ongoing events with live updates until the event ends.
- Shows upcoming events with details like location, time, and organizer info.

2. Seamless Google Maps Integration:

Google Maps preview appears when the cursor nears an event listing.

Clicking on an event starts real-time navigation in Google Maps.

3. Event Booking System:

- Users can book event tickets directly through the application.
- Generates booking confirmations and maintains booking history.

4. Interactive Event Calendar:

- Displays scheduled events with reminders.
- Tracks previously attended events for user reference.

5. Live Event Timer:

- Displays a countdown for ongoing events.
- Sends alerts when an event is nearing its end.

6. User-Friendly Interface:

- Intuitive design for easy event search and booking.
- Responsive layout for mobile and web compatibility.

7. Event Organizer Dashboard:

- Enables event hosts to create, manage, and update events.
- Provides insights on attendee engagement and ticket sales.

8. Push Notifications & Alerts:

- Sends reminders for upcoming events.
- Notifies users about ticket availability and event updates.

9. Personalized Event Recommendations:

- Uses AI-driven algorithms to suggest events based on user preferences.
- Allows users to follow favorite event organizers.

10. Secure Payment Gateway:

- Supports multiple payment methods for seamless transactions.
- Ensures secure and encrypted payments for bookings.

The proposed application not only simplifies event discovery but also enhances user engagement with its intuitive features. By integrating real-time updates, seamless navigation, and an efficient booking system, the platform ensures a smooth experience for both attendees and organizers. The inclusion of AI-driven recommendations, push notifications, and a secure payment gateway further adds to its reliability and usability.

With its user-friendly interface and robust event management capabilities, It serves as an all-in-one solution for discovering, attending, and organizing events. Whether users are looking for entertainment, networking opportunities, or professional gatherings, this platform provides a hassle-free and enjoyable experience.

CHAPTER 2

SYSTEM STUDY AND ANALYSIS

2.1 PROBLEM STATEMENT

In the modern digital landscape, staying updated with ongoing and upcoming events is crucial for individuals looking to engage in social, professional, and entertainment activities. However, finding relevant events in real-time remains a major challenge due to the lack of a centralized, user-friendly platform. Existing methods such as social media promotions, newspaper advertisements, and word-of-mouth recommendations are often scattered and inefficient, making event discovery time-consuming and unreliable.

One of the primary challenges users face is the difficulty in accessing live event updates. Many event-goers miss out on important gatherings due to the unavailability of real-time information. Additionally, even when users find an event, they often struggle with navigation, as existing platforms do not provide seamless integration with mapping services. This results in inconvenience, confusion, and missed opportunities.

Another issue lies in event booking and management. Many event discovery platforms either lack a direct ticket booking feature or redirect users to external websites, leading to a fragmented and frustrating experience. Furthermore, attendees often do not have a structured way to track past events they attended or set reminders for future events they plan to join. Without a proper scheduling system, users may forget important dates or struggle to manage multiple event commitments efficiently.

On the other hand, event organizers face significant difficulties in reaching the right audience and maintaining engagement. Traditional promotion methods often fail to attract sufficient attendees, while existing event listing platforms lack interactive features to boost visibility and provide organizers with insights into user engagement. Without real-time updates and notifications, organizers cannot effectively communicate last-minute changes or updates to attendees.

The lack of AI-driven personalization further complicates the event discovery experience. Users have different interests and preferences, but most platforms do not offer intelligent recommendations based on individual behavior. As a result, attendees may miss

out on events that align with their hobbies, professions, or social interests simply because they are unaware of them.

The **Event Finder** application is designed to address these challenges by providing a **comprehensive, real-time event discovery and management platform**. It offers **live updates, Google Maps integration, direct booking features, an interactive calendar, and a secure payment system**. Additionally, AI-driven recommendations help users discover events tailored to their interests, while push notifications ensure they stay updated. By bridging the gap between event attendees and organizers, **Event Finder** enhances accessibility, engagement, and convenience, creating a seamless and efficient event discovery experience for all users.

2.2 EXISTING SYSTEM

Currently, event discovery and management rely on multiple fragmented platforms, including social media, event listing websites, and traditional advertising methods. While these solutions help users find events, they come with several limitations that make the process inefficient and less engaging.

Many users depend on **social media platforms** such as Facebook, Instagram, and Twitter to learn about upcoming events. However, these platforms do not provide a structured way to browse events based on location, category, or user preference. Additionally, posts and promotions often get lost in the overwhelming amount of content, making event discovery inconsistent and unreliable.

Event listing websites like Eventbrite and Meetup offer a more organized approach but come with their own drawbacks. These platforms typically focus on event registration and ticket sales but lack real-time updates, personalized recommendations, and integrated navigation features. Moreover, users often have to manually search for events, and there is no seamless way to track previously attended or scheduled events.

Traditional advertising methods, such as posters, banners, newspaper ads, and word-of-mouth, are still used by many event organizers. However, these methods are outdated and have a very limited reach. They fail to provide live updates, direct navigation assistance, or a simple way for users to engage with events.

Another major issue with existing systems is the **lack of real-time updates**. Users may not receive notifications about event changes, such as time adjustments or venue modifications, leading to confusion and missed opportunities. Additionally, most existing platforms do not provide a dedicated countdown timer for ongoing events, making it difficult for attendees to keep track of event durations.

Furthermore, **event booking systems are often disconnected** from event discovery platforms. Users may have to visit separate websites to purchase tickets, leading to a cumbersome experience. Many platforms do not offer a **secure payment gateway**, increasing the risk of fraudulent transactions and reducing user trust.

Overall, the existing systems for event discovery and management are **scattered, inefficient, and lack integration**. Users struggle to find real-time information, navigate to event locations, and book tickets seamlessly. Organizers, on the other hand, find it challenging to reach their target audience effectively.

2.2.1 DRAWBACK

Despite the availability of multiple platforms for event discovery, the existing systems suffer from several limitations that hinder user experience and engagement. These drawbacks make it difficult for both event attendees and organizers to efficiently interact and manage events.

1. Lack of Real-Time Updates:

- Most existing platforms do not provide live updates about events, such as time changes, cancellations, or venue modifications.
- Users may miss out on important information due to the absence of push notifications.

2. Limited Event Discovery Options:

- Social media platforms rely on random posts and ads, making it difficult for users to discover events based on their interests or location.
- Event listing websites require manual searches, and there is no AI-driven recommendation system to suggest relevant events.

3. No Integrated Navigation System:

- Users have to manually copy event locations and search for them in navigation apps like Google Maps.

- There is no seamless integration that allows one-click navigation directly from the event listing.

4. Disjointed Booking Process:

- Many platforms redirect users to external sites for ticket purchases, leading to a fragmented and frustrating experience.
- Some platforms lack secure payment gateways, increasing the risk of fraudulent transactions.

5. Absence of a Centralized Calendar System:

- Users do not have an efficient way to track previously attended events or set reminders for future ones.
- There is no built-in event countdown timer to notify attendees about ongoing events.

6. Limited Engagement for Event Organizers:

- Organizers struggle to reach their target audience effectively due to inefficient promotion methods.
- There is no interactive dashboard for managing bookings, tracking attendee engagement, or analyzing event performance.

7. No AI-Driven Personalization:

- Existing platforms do not use AI to suggest events based on user preferences, past event attendance, or browsing history.
- Users may miss out on events that align with their interests simply because they are unaware of them.

8. Scattered and Inefficient Communication:

- There is no streamlined communication between event organizers and attendees for queries, updates, or announcements.
- Users have to rely on social media comments or separate emails, making the process slow and ineffective.

The drawbacks of the existing system highlight the need for a more integrated, intelligent, and user-friendly event discovery platform. The Event Finder application aims to overcome these challenges by offering real-time updates, seamless navigation, an interactive calendar, AI-driven recommendations, and a secure booking system. These improvements will enhance user experience, increase event participation, and provide better management tools for organizers.

2.3 PROPOSED SYSTEM

To overcome the limitations of the existing event discovery and management systems, the Event Finder application introduces a smart, real-time, and user-friendly solution that enhances the event experience for both attendees and organizers. The proposed system integrates live updates, seamless navigation, AI-driven recommendations, and an efficient booking system to provide a comprehensive platform for event discovery and management.

The Event Finder application ensures that users can easily discover ongoing and upcoming events, receive real-time notifications, and navigate effortlessly using Google Maps integration. With a built-in event calendar, users can track their attended events, schedule future ones, and receive reminders. Additionally, the system acts as a booking agent, allowing users to reserve tickets securely through an integrated payment gateway. The platform also benefits event organizers by offering an interactive dashboard to create, update, and monitor event performance.

2.3.1 ADVANTAGES

The Event Finder application provides numerous benefits over the existing event discovery and management platforms. By integrating real-time updates, seamless navigation, AI-driven recommendations, and a user-friendly booking system, the proposed system ensures a smooth and efficient experience for both users and event organizers.

Key Advantages:

1. Enhanced Event Discovery:

- Users can easily find ongoing and upcoming events without manually searching multiple platforms.
- AI-powered recommendations suggest events based on user interests and past attendance.

2. Real-Time Event Updates & Notifications:

- Users receive live updates about event changes, cancellations, or venue modifications.

- Push notifications ensure that attendees never miss an important event update.
- User can alter what they get updated on live, according to their interests.

3. Seamless Google Maps Integration:

- Eliminates the hassle of manually searching for event locations.
- Users can preview the location when hovering over an event and start navigation with a single click.

4. Efficient Event Booking System:

- Users can book event tickets directly within the app, avoiding third-party redirections.
- Integrated secure payment gateway ensures fast and safe transactions.

5. User-Friendly Event Calendar:

- Helps users keep track of attended events, upcoming events, and important schedules.
- Sends reminders and alerts for scheduled events.

6. Live Countdown Timer for Ongoing Events:

- Displays a real-time countdown for current events.
- Keeps users informed about event duration and end times.

7. Better Event Management for Organizers:

- Organizers can easily create, update, and manage events in real time.
- The interactive dashboard provides insights into ticket sales, user engagement, and event success.

8. AI-Driven Personalization:

- Users get customized event recommendations based on their location, interests, and activity.

- Reduces the time spent searching for relevant events.

9. Push Notifications & Alerts:

- Ensures users are notified about upcoming events, booking confirmations, and last-minute changes.
- Helps event organizers keep attendees engaged with timely updates.

10. Secure & Reliable Transactions.

- Integrated **secure payment gateway** ensures **safe and encrypted transactions**.
- Supports multiple payment methods for user convenience.

2.4 FEASIBILITY ANALYSIS

The **Feasibility Analysis** is conducted to assess the practicality of implementing the Event Finder application by evaluating its technical, operational, and financial aspects. This ensures that the project is viable and can be developed within the available resources, time, and budget.

2.4.1 Economic Feasibility

The project is cost-effective as it primarily relies on open-source technologies and cloud-based services.

- **Development Costs:** Minimal, as open-source frameworks (React, Node.js, Django) reduce licensing fees.
- **Hosting & Maintenance:** Cloud-based solutions like **AWS, Firebase, or Google Cloud** ensure **low-cost, scalable deployment**.
- **Revenue Generation:** The platform can generate income through:
 - **Event listing fees** for organizers.
 - **Commission on ticket sales** via the integrated booking system.
 - **Premium event promotions** for better visibility.
 - **Advertisements and sponsorships** from event partners.

Since the initial development costs are low and multiple revenue streams exist, the project is economically viable and **sustainable in the long run**.

2.4.2 Technical Feasibility

The **Event Finder** system is technically feasible as it uses **widely adopted technologies** and APIs to deliver its core functionalities.

- Developed using **React.js** for an interactive and responsive user interface.
- Uses **Node.js** as the backend to ensure efficient handling of real-time data.
- Integrated with **Google Maps API** for seamless navigation.
- Secure payment gateways (Razorpay, PayPal, Stripe) ensure smooth and encrypted transactions.
- Hosted on **AWS**, ensuring scalability and high performance.
- AI-based recommendation engine enhances user experience by suggesting relevant events.

Since all the required technologies are **well-documented and widely supported**, implementation is **technically feasible** with minimal risk.

2.4.3 Operational Feasibility

The system is designed to be **user-friendly** and provides a seamless experience for both event attendees and organizers.

- **For Users:**
 - Easy event discovery with **AI-driven recommendations**.
 - One-click navigation and **real-time updates** for ongoing events.
 - Secure event booking and transaction history.
- **For Event Organizers:**
 - Simple **event creation and management** dashboard.
 - Real-time engagement tracking and ticket sales analytics.
 - Instant notifications and alerts for better audience reach.

With an **intuitive UI/UX** and **minimal learning curve**, the system ensures high usability and easy adoption.

CHAPTER 3

DEVELOPMENT ENVIRONMENT

3.1 HARDWARE REQUIREMENTS

CPU	: Intel Core i3
RAM	: 4 GB
Hard Disk	: 500 GB
Mouse	: Logitech Mouse
Keyboard	: 104 Keys
Motherboard	: Intel
Speed	: 3.3 GHz
Floppy Disk Drive	: 2 MB

3.2 SOFTWARE REQUIREMENTS

Operating System	: Windows 10/11, macOS, or Linux (Ubuntu 20.04+)
Programming Languages	: JavaScript
Frontend Framework	: Next.js
Backend Framework	: Next.js
Database	: PostgreSQL
API Integration	: Google Maps API, Payment Gateway API
Development Tools	: Visual Studio Code, Git, Postman, Selenium
Web Server	: Apache
Cloud Hosting	: AWS

3.3 PROGRAMMING ENVIRONMENT

The **Event Finder** application is developed in a structured programming environment that ensures efficiency, scalability, and maintainability. The selection of technologies is based on their robustness, performance, and compatibility with modern web development standards. The following are the key components of the programming environment:

Operating System

The application is primarily developed and tested on:

- **Windows 10** – Ensures compatibility with widely used development tools and frameworks.

Front-End Development

The front-end is designed to provide a seamless and responsive user experience using:

- **TypeScript** – A statically typed superset of JavaScript that enhances code maintainability and error detection.
- **Next.js** – A powerful React-based framework that offers server-side rendering (SSR) and static site generation (SSG) for optimized performance.

Back-End Development

The back-end handles business logic, API requests, and database interactions using:

- **Next.js** – A full-stack framework that supports both client-side and server-side operations, simplifying development with built-in API routes.

Database Management

For efficient data storage and retrieval, the application utilizes:

- **PostgreSQL** – A powerful, open-source relational database known for its scalability, security, and advanced data-handling capabilities.

Integrated Development Environment (IDE)

Developers use a feature-rich code editor for efficient coding and debugging:

- **Visual Studio Code** – A lightweight, fast, and extensible editor with built-in support for TypeScript and Next.js.

Version Control System

To track code changes and collaborate efficiently, the following version control tools are used:

- **Git** – A distributed version control system for managing source code.
- **GitHub** – A cloud-based repository hosting service that enables collaboration and project management.

Package Management

To streamline dependency management and automate installations, the following package manager is used:

- **npm (Node Package Manager)** – Handles JavaScript dependencies required for Next.js and other libraries.

Server Environment

The back-end services run on a robust server-side environment:

- **Node.js Runtime Environment** – Allows execution of JavaScript code outside the browser, making it suitable for server-side applications.

API Integrations

The application integrates external APIs to enhance functionality:

- **Google Maps API** – Provides interactive mapping and navigation features.
- **Payment Gateway API** – Facilitates secure online transactions for event bookings.

3.3.1 Brief Note on TypeScript and Next JS

Frontend Language: TypeScript

TypeScript is a statically typed superset of JavaScript that enhances code quality and maintainability. It helps in catching errors during development, improving productivity, and making large-scale applications easier to manage. Since TypeScript is compiled to JavaScript, it runs efficiently in web browsers.

Frontend Framework: Next.js

Next.js is a React-based framework that enables server-side rendering (SSR), static site generation (SSG), and client-side rendering (CSR). It improves performance, SEO, and scalability while offering built-in API routes, making it a popular choice for modern web applications.

3.3.2 Brief Note on Next.js

Backend Language & Framework: Next.js

Next.js is also used for the backend, leveraging its API routes feature to handle server-side logic, database interactions, and authentication. This eliminates the need for a separate backend framework, making development more streamlined and efficient.

Database: PostgreSQL

PostgreSQL is a relational database management system (RDBMS) known for its scalability, security, and support for complex queries. It is widely used for web applications that require efficient data handling and reliability.

3.3.3 Brief Note on PostgreSQL

PostgreSQL is a powerful, open-source relational database management system (RDBMS) known for its scalability, reliability, and advanced data-handling capabilities. It follows the ACID (Atomicity, Consistency, Isolation, Durability) principles, ensuring data integrity and consistency.

Key Features of PostgreSQL:

- **High Performance:** Supports indexing, partitioning, and query optimization for faster data retrieval.
- **Extensibility:** Allows custom functions, stored procedures, and integration with multiple programming languages.
- **Security:** Provides robust authentication, encryption, and access control mechanisms.
- **Scalability:** Can handle large datasets efficiently, making it suitable for enterprise applications.
- **Concurrency Control:** Uses Multi-Version Concurrency Control (MVCC) to handle multiple transactions without conflicts.

Database Tables in the System

Event Finder application uses a well-structured relational database with PostgreSQL to manage event details, user information, bookings, and real-time updates. Below are the key database tables:

1. Users Table

Stores user details, including attendees and event organizers.

Column Name	Data Type	Description
user_id	SERIAL (PK)	Unique identifier for each user
name	VARCHAR(255)	User's full name
email	VARCHAR(255)	User's email address (unique)
password	TEXT	Hashed password for authentication
phone_number	VARCHAR(20)	Contact number
user_role	VARCHAR(50)	Role (Attendee/Organizer)
created_at	TIMESTAMP	Account creation timestamp

2. Events Table

Stores details of all events listed in the system.

Column Name	Data Type	Description
event_id	SERIAL (PK)	Unique identifier for each event
event_name	VARCHAR(255)	Name of the event
description	TEXT	Event details and description
location	VARCHAR(255)	Event location
latitude	DECIMAL(10,8)	Latitude for mapping
longitude	DECIMAL(11,8)	Longitude for mapping
start_time	TIMESTAMP	Event start time
end_time	TIMESTAMP	Event end time
organizer_id	INTEGER (FK)	References users(user_id)
status	VARCHAR(50)	Event status (Ongoing/Upcoming/Ended)

3. Bookings Table

Tracks event ticket bookings made by users.

Column Name	Data Type	Description
booking_id	SERIAL (PK)	Unique identifier for each booking
user_id	INTEGER (FK)	References users(user_id)
event_id	INTEGER (FK)	References events(event_id)
booking_date	TIMESTAMP	Date and time of booking
payment_status	VARCHAR(50)	Payment status (Paid/Pending)

4. Event Updates Table

Stores live updates related to events.

Column Name	Data Type	Description
update_id	SERIAL (PK)	Unique identifier for each update
event_id	INTEGER (FK)	References events(event_id)
update_text	TEXT	Update message
created_at	TIMESTAMP	Timestamp of update

5. Notifications Table

Stores system-generated notifications for users.

Column Name	Data Type	Description
notification_id	SERIAL (PK)	Unique identifier for each notification
user_id	INTEGER (FK)	References users(user_id)
message	TEXT	Notification content
status	VARCHAR(50)	Read/Unread
created_at	TIMESTAMP	Time of notification

CHAPTER 4

SYSTEM DESIGN

4.1 MODULE DESCRIPTION

The Event Finder application is structured into several functional modules, each serving a specific purpose to ensure a smooth and efficient event discovery and management experience for users and event organizers. These modules work together to provide real-time event updates, seamless navigation, secure booking, and user-friendly interactions. Below is a detailed description of each module: **User Authentication Module**

1. User Authentication Module

The **User Authentication Module** is responsible for handling all user-related functionalities, including registration, authentication, and profile management. Users can sign up as attendees or event organizers, with different access levels and privileges.

- **User Registration & Login:** New users can create an account using their email, phone number, or social media login. Existing users can log in using their credentials.
- **User Profile Management:** Users can edit their profiles, update contact details, and set event preferences.
- **Role-based Access Control:** The system differentiates between attendees and event organizers, granting different permissions to each.
- **Secure Authentication:** Passwords are securely stored using encryption, and multi-factor authentication (MFA) can be implemented for added security.

2. Event Discovery Module

The **Event Discovery Module** is designed to help users find events based on location, category, and preferences. This module ensures that users can quickly access relevant event details without hassle.

- **Real-time Event Listings:** Displays ongoing and upcoming events with live updates.
- **Advanced Search & Filters:** Users can filter events based on date, location, category (music, sports, tech, etc.), and ticket availability.

- **Event Details Page:** Clicking on an event opens a detailed page with descriptions, timings, venue details, and booking options.
- **Personalized Recommendations:** Uses AI-driven algorithms to suggest events based on user interests and past interactions.

3. Google Maps Integration Module

This module enables seamless navigation and event location tracking using the **Google Maps API**. It enhances the user experience by providing real-time directions and a location preview.

- **Interactive Map Display:** When a user hovers over an event, a small map preview appears.
- **Click to Navigate:** Clicking on an event's location starts real-time navigation in Google Maps.
- **Geo-location Based Event Discovery:** Users can view nearby events based on their current location.
- **Traffic & Route Suggestions:** Provides traffic updates and suggests the best route to the event venue.

4. Event Booking Module

The **Event Booking Module** allows users to reserve tickets for events directly through the application. This module ensures a smooth and secure ticketing experience.

- **Easy Ticket Booking:** Users can select ticket types, quantities, and complete their bookings.
- **Secure Payment Gateway:** Supports multiple payment options (credit/debit cards, UPI, wallets).
- **E-Ticket Generation:** After booking, users receive a digital ticket with a unique QR code.
- **Booking History & Management:** Users can view past bookings and cancel tickets if needed.

5. Event Calendar & Tracking Module

This module helps users keep track of their event schedules, providing reminders and logs of past events attended.

- **Scheduled Event Calendar:** Displays all upcoming events in a calendar view.
- **Reminders & Alerts:** Sends notifications about upcoming events.
- **Event Attendance History:** Users can review past events they have attended.
- **Countdown Timer:** A real-time countdown for ongoing events is displayed on the dashboard.

6. Live Event Updates Module

The **Live Event Updates Module** allows organizers to send real-time updates, ensuring that attendees stay informed about any changes or announcements.

- **Live Event Feed:** Users can view updates such as event schedule changes, guest appearances, or venue modifications.
- **Push Notifications:** Sends instant alerts about important updates.
- **Organizer Announcements:** Organizers can post messages, images, and videos for attendees.
- **Engagement Features:** Users can comment, like, or share event updates.

7. Notification & Alert Module

This module is responsible for keeping users engaged and informed about their events, bookings, and personalized recommendations.

- **Event Reminders:** Sends reminders for upcoming and ongoing events.
- **Booking Confirmation Alerts:** Notifies users about successful bookings and cancellations.
- **Personalized Suggestions:** Users receive recommendations based on their interests.
- **Emergency Notifications:** In case of event cancellations or venue changes, immediate alerts are sent.

8. Event Organizer Dashboard Module

The **Event Organizer Dashboard** is a dedicated module for event organizers to create, manage, and monitor their events efficiently.

- **Event Creation & Management:** Organizers can add event details, upload images, and set ticket prices.
- **Attendee Tracking:** Organizers can view the list of attendees and booking details.
- **Event Performance Insights:** Provides analytics on ticket sales, user engagement, and revenue.
- **Post-event Feedback Collection:** Organizers can collect reviews and ratings from attendees.

4.2 INPUT DESIGN

Input design is a crucial component of system development that determines how users interact with the system. A well-designed input system ensures efficiency, accuracy, and ease of use while minimizing user errors. The Event Finder application is designed with an intuitive input system that enables users to register, search for events, book tickets, and manage event listings seamlessly.

This section describes the various input mechanisms implemented in the Event Finder system, detailing the types of inputs, validation techniques, and their impact on overall usability.

Objectives of Input Design

1. Ensure Accuracy and Data Integrity

- Inputs are validated to prevent incorrect, incomplete, or duplicate entries.
- Uses data constraints such as required fields, character limits, and format validations (e.g., email, phone number).
- Auto-suggestions and dropdowns minimize user errors while entering event details.

2. Enhance User Experience

- Provides a **user-friendly interface** with structured input fields for easy navigation.
- Uses interactive elements like date pickers, dropdowns, and auto-fill options to streamline data entry.
- Minimizes the number of steps needed for actions like event booking and registration.

3. Improve Efficiency and Speed

- Implements **real-time validation** to alert users about errors instantly.
- Reduces the time taken for data entry using features like **Google Maps auto-complete** and autofill options.
- Uses smart filters for quick event searches and ticket bookings.

4. Ensure Security and Authentication

- Prevents unauthorized access through **secure login and registration** mechanisms.
- Implements **CAPTCHA verification** to reduce bot interactions.
- Uses **encrypted payment input fields** for secure financial transactions.

5. Support Multiple Input Methods

- Allows input via **keyboard, mouse, or touch** for better accessibility.
- Supports file uploads (e.g., profile pictures, event banners) with size and format restrictions.
- Integrates with APIs like **Google Maps** to reduce manual data entry.

6. Provide Error Handling and Feedback

- Displays **clear error messages** when users enter incorrect or missing data.
- Uses **confirmation prompts** before finalizing actions like booking or event submission.
- Implements tooltips and placeholder texts to guide users through form fields.

7. Optimize for Different Devices

- Ensures a **responsive design** for seamless input on desktops, tablets, and smartphones.
- Uses adaptive input methods based on the device (e.g., mobile keyboards for number entry).
- Provides a consistent user experience across different platforms.

8. Ensure Compatibility with Backend Processing

- Ensures that input data formats are compatible with the database structure.
- Validates that entered information aligns with business rules (e.g., event timings must be future-dated).
- Optimizes data submission processes to prevent overload or duplicate entries.

Types of Input

1. User Registration & Login Input

The first interaction a user has with the Event Finder application is through the registration and login system. This ensures that only authenticated users can access the platform, thereby improving security and personalization.

Input Fields

Users must provide the following details to register:

- Full Name: (Text Input) – Accepts only alphabets and spaces.
- Email ID: (Email Validation) – Ensures a valid email format for communication and login.
- Phone Number: (Numeric with Country Code) – Validates numbers according to international formats.
- Password: (Masked Input with Strength Indicator) – Requires at least one uppercase letter, one number, and one special character.
- Confirm Password: (Validation against Password) – Ensures password confirmation matches the original input.

- Role Selection: (Radio Button – Attendee / Organizer) – Determines user permissions.
- Profile Picture Upload: (File Input) – Accepts image files (JPEG, PNG) for profile setup.

Validation Techniques

- Email Validation: Uses regex to ensure valid email formats.
- Password Strength Check: Ensures strong passwords to enhance security.
- Mandatory Fields: Users cannot proceed without filling all required fields.
- CAPTCHA Verification: Protects against bot registrations.

2. Event Creation & Management Input (For Organizers)

Event organizers need to provide detailed information about their events. The system is designed to make event creation simple while ensuring complete and accurate data input.

Input Fields

- Event Title: (Text Input) – Name of the event.
- Event Category: (Dropdown – Music, Sports, Business, etc.) – Classifies events for easy filtering.
- Event Date & Time: (Date & Time Picker) – Selects an event schedule.
- Event Venue: (Text Input with Google Maps Auto-Complete) – Fetches precise location details.
- Event Description: (Rich Text Editor) – Allows formatted text input.
- Ticket Pricing: (Numeric Input with Currency Selector) – Defines ticket cost per entry.
- Ticket Availability: (Numeric Input) – Sets the maximum number of tickets.
- Upload Event Banner: (File Upload – JPG, PNG, WebP) – Enhances event visibility.

Validation Techniques

- Auto-suggestions for Locations: Google Maps API suggests valid event venues.
- Date Restriction: Prevents event scheduling in the past.
- Mandatory Fields: Ensures that all required data is provided.
- Price Check: Ensures ticket prices are positive numbers.

3. Event Search & Navigation Input

Users can search for events based on various filters, allowing for a personalized experience.

Input Fields

- Search Box: (Text Input with Auto-suggestions) – Allows keyword-based event searches.
- Filter by Category: (Checkbox / Dropdown) – Sorts events by type.
- Filter by Date: (Date Picker) – Finds events within a specific timeframe.
- Filter by Location: (Google Maps Location Input) – Displays events near the selected location.
- Sort By: (Dropdown – Popularity, Date, Price) – Orders results based on user preference.

Validation Techniques

- Keyword Verification: Ensures relevant search terms.
- Multiple Filters Handling: Allows simultaneous filtering for better results.
- Date Validation: Ensures users cannot select past dates when filtering future events.

4. Event Booking Input

The Event Finder application allows users to book event tickets with ease. The booking system ensures accuracy and prevents overbooking.

Input Fields

- Event Name: (Auto-filled) – The system auto-fills event details.
- Number of Tickets: (Numeric Input with Limit) – Users can choose tickets based on availability.
- Payment Method: (Radio Button – Credit Card, UPI, PayPal) – Provides multiple payment options.
- Card Details: (Text & Numeric Input) – Users enter payment card details securely.
- Billing Address: (Text Input with Auto-complete) – Provides precise location entry.

Validation Techniques

- Availability Check: Ensures tickets are not oversold.
- Secure Payment Processing: Encrypts financial details for security.
- Mandatory Billing Address: Ensures valid transaction records.

5. User Feedback & Review Input

The system encourages users to provide feedback and rate events they attended.

Input Fields

- Rating: (Star Rating System – 1 to 5) – Allows users to rate events.
- Review Description: (Text Input with Word Limit) – Ensures concise reviews.
- Upload Media: (Optional File Upload – Images/Videos) – Allows sharing event moments.

Validation Techniques

- AI Moderation: Detects offensive language.
- Word Count Limit: Ensures reviews remain relevant.
- User Authentication: Allows only verified attendees to submit feedback.

6. Admin Input Panel

The administrator manages system-wide settings, user access, and event approvals.

Input Fields

- User Management: (Dropdown - Approve, Block, Delete) - Admin can control user actions.
- Event Approvals: (Checkbox Selection) - Admin can approve or reject event listings.
- System Logs: (Auto-generated) - Displays system activities for security monitoring.

Validation Techniques

- Admin Authentication: Ensures secure system access.
- Action Logging: Tracks administrative actions for transparency.

4.3 OUTPUT DESIGN

The output design ensures that the system provides **meaningful, well-structured, and real-time** information to users. The output includes live event updates, booking confirmations, reports, and notifications for various system activities. The structured presentation of data improves **user experience, decision-making, and system efficiency**.

Objectives of Output Design

- **To present accurate and relevant event information** in a structured format.
- **To ensure clarity and readability** of displayed data for users, organizers, and admins.
- **To provide real-time responses** to user actions such as booking confirmations and payments.

Types of Output

User Outputs (Attendees)

- **Event Booking Confirmation:** Displays event details, location, date/time, and booking status.
- **Upcoming Events List:** Shows personalized event recommendations based on user preferences.
- **Event Updates:** Real-time notifications on changes in event status, venue, or timing.
- **Payment Receipt:** Generates a digital invoice after a successful payment.

Organizer Outputs

- **Event Dashboard:** Provides an overview of scheduled, ongoing, and past events.
- **Attendee List:** Displays registered attendees for an event with contact details.
- **Revenue Reports:** Shows earnings from ticket sales and payment processing status.
- **Event Performance Analytics:** Provides insights on ticket sales, user engagement, and popular event categories.

Admin Outputs

- **User Reports:** Lists registered users (attendees and organizers) along with login activity.
- **Event Moderation Logs:** Displays flagged events, reported issues, and event status updates.
- **System Revenue Reports:** Generates financial reports on bookings, payments, and earnings.
- **Security Logs:** Monitors login attempts, failed transactions, and account access records.

Output Formats

- **On-Screen Displays:** Live event updates, dashboards for attendees, organizers, and admins.
- **Printable Reports:** Export reports in **PDF, Excel, or CSV** formats for record-keeping.

- **Email/SMS Notifications:** Automated confirmations and reminders for event bookings and updates.

By implementing an **efficient output design**, the system ensures that all stakeholders receive the necessary information in a structured, **timely, and user-friendly manner**.

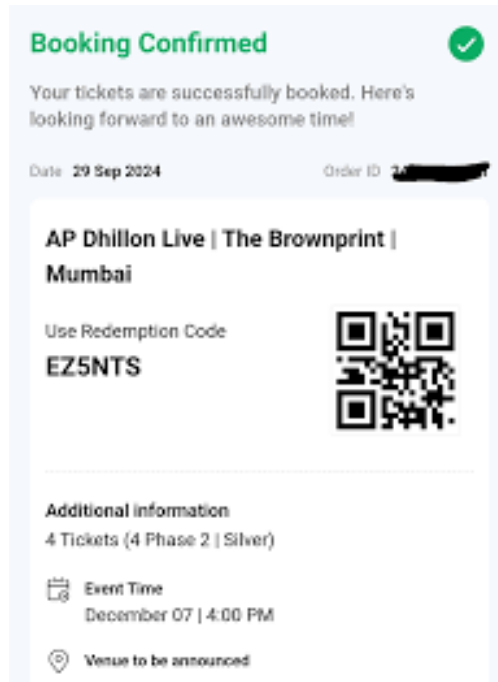


Figure – 01 Ticket Confirmation

4.4 DATABASE DESIGN

Database design is a crucial aspect of the **Event Finder** application as it ensures efficient data storage, retrieval, and management. A well-structured database enables seamless interaction between users and the system, ensuring that event details, bookings, user information, and transactions are stored securely and accessed efficiently.

The database design follows a **relational model** using **PostgreSQL**, which provides **data integrity, security, and scalability**. The tables are designed to handle various aspects of the system, including user profiles, event details, booking transactions, and real-time updates.

Schema Design

Schema design plays a vital role in defining how data is structured, stored, and accessed efficiently in the **Event Finder** application. The database follows a **relational**

model using **PostgreSQL** and is structured to ensure data integrity, minimize redundancy, and optimize query performance.

The database schema consists of the following key entities:

- **Users:** Stores user-related information (attendees, organizers, admins).
- **Events:** Stores event details, including location and status.
- **Bookings:** Manages event ticket reservations.
- **Payments:** Tracks transaction details for booked events.
- **Event Updates:** Stores live updates for ongoing events.

Tables and Relationships

1. Users Table

Stores user details, including authentication and roles.

Column Name	Data Type	Constraints	Description
user_id	SERIAL	PRIMARY KEY	Unique ID for each user
name	VARCHAR(255)	NOT NULL	Full name of the user
email	VARCHAR(255)	UNIQUE, NOT NULL	Email for login and notifications
password	TEXT	NOT NULL	Encrypted password for authentication
phone	VARCHAR(15)	UNIQUE, NOT NULL	Contact number
role	VARCHAR(50)	CHECK(role IN ('attendee', 'organizer', 'admin'))	Defines the role of the user

2. Events Table

Stores details of all listed events.

Column Name	Data Type	Constraints	Description
event_id	SERIAL	PRIMARY KEY	Unique event ID
event_name	VARCHAR(255)	NOT NULL	Name of the event
description	TEXT	NOT NULL	Event details
date_time	TIMESTAMP	NOT NULL	Date and time of the event
location	VARCHAR(255)	NOT NULL	Address of the event
latitude	DECIMAL(9,6)	NOT NULL	Geo-coordinate for map integration
longitude	DECIMAL(9,6)	NOT NULL	Geo-coordinate for map integration
organizer_id	INT	FOREIGN KEY REFERENCES Users(user_id)	Event organizer
total_seats	INT	NOT NULL	Number of seats available
status	VARCHAR(50)	CHECK(status IN ('upcoming', 'ongoing', 'completed'))	Event status

3. Bookings Table

Manages event ticket reservations.

Column Name	Data Type	Constraints	Description
booking_id	SERIAL	PRIMARY KEY	Unique booking ID
user_id	INT	FOREIGN KEY REFERENCES Users(user_id)	User making the booking
event_id	INT	FOREIGN KEY REFERENCES Events(event_id)	Event booked
booking_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of booking
status	VARCHAR(50)	CHECK(status IN ('confirmed', 'cancelled'))	Booking status

4. Payments Table

Handles financial transactions for event bookings.

Column Name	Data Type	Constraints	Description
payment_id	SERIAL	PRIMARY KEY	Unique payment ID
booking_id	INT	FOREIGN KEY REFERENCES Bookings(booking_id)	Associated booking ID
user_id	INT	FOREIGN KEY REFERENCES Users(user_id)	User who made the payment
amount	DECIMAL(10,2)	NOT NULL	Amount paid
payment_status	VARCHAR(50)	CHECK(payment_status IN ('successful', 'pending', 'failed'))	Transaction status
payment_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Payment timestamp

5. Event Updates Table

Stores live updates for ongoing events.

Column Name	Data Type	Constraints	Description
update_id	SERIAL	PRIMARY KEY	Unique update ID
event_id	INT	FOREIGN KEY REFERENCES Events(event_id)	Associated event
update_text	TEXT	NOT NULL	Live update content
update_time	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Time of the update

Normalization and Optimization

To enhance performance and prevent data redundancy, the database schema follows **normalization principles**:

Normalization Levels Applied:

First Normal Form (1NF):

- Eliminated duplicate columns and ensured atomic data storage.
- Each event has a unique ID, and no multi-valued attributes exist.

Second Normal Form (2NF):

- Removed partial dependencies by ensuring all non-key attributes depend entirely on the primary key.
- The **Bookings** table references both **Users** and **Events** instead of storing redundant user details.

Third Normal Form (3NF):

- Removed transitive dependencies.
- The **Payments** table stores payment details separately, preventing unnecessary dependencies on booking data.

Security Measures

To ensure **data integrity, user privacy, and security**, the following measures are implemented:

1. Authentication & Authorization:

- **Password Hashing:** Securely stores user passwords using **bcrypt** encryption.
- **Role-Based Access Control (RBAC):** Restricts access to event management features based on user roles (admin, organizer, attendee).

2. Data Protection & Encryption:

- **SSL/TLS Encryption:** Ensures secure data transmission over the network.
- **AES Encryption:** Protects sensitive user details like payment data.

3. SQL Injection Prevention:

- **Prepared Statements & Parameterized Queries:** Prevents injection attacks by ensuring user inputs are properly sanitized.

4. API Security:

- **Token-Based Authentication:** Uses **JWT (JSON Web Token)** for secure API access.
- **Rate Limiting & API Throttling:** Limits the number of requests per user to prevent abuse.

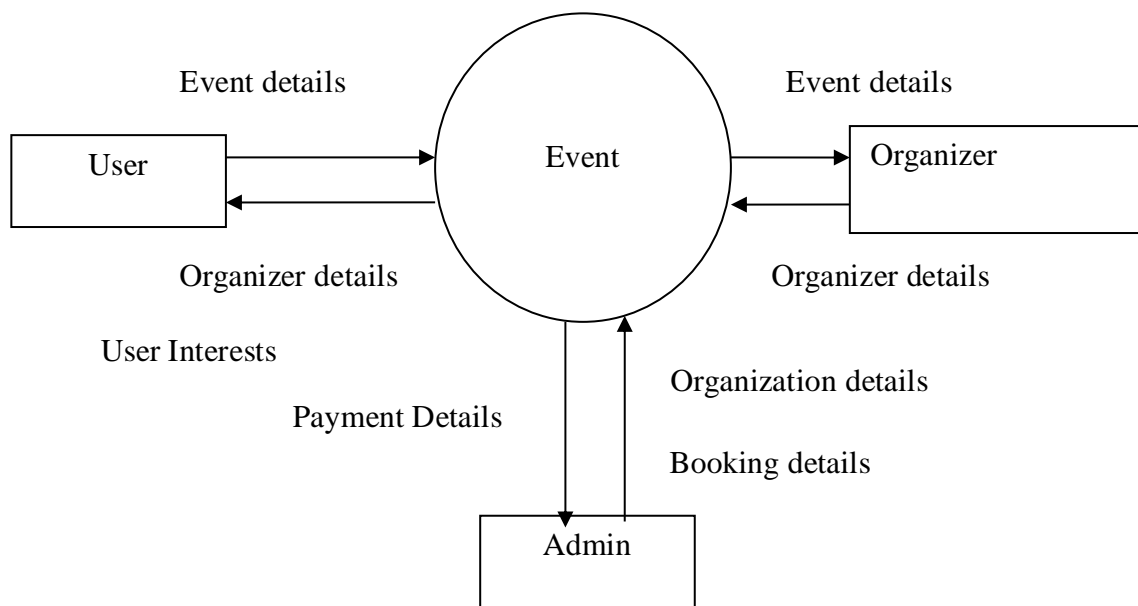
5. Backup & Disaster Recovery:

- **Automated Daily Backups:** Ensures data recovery in case of failure.
- **Multi-Region Data Storage:** Stores backups in different locations for reliability.

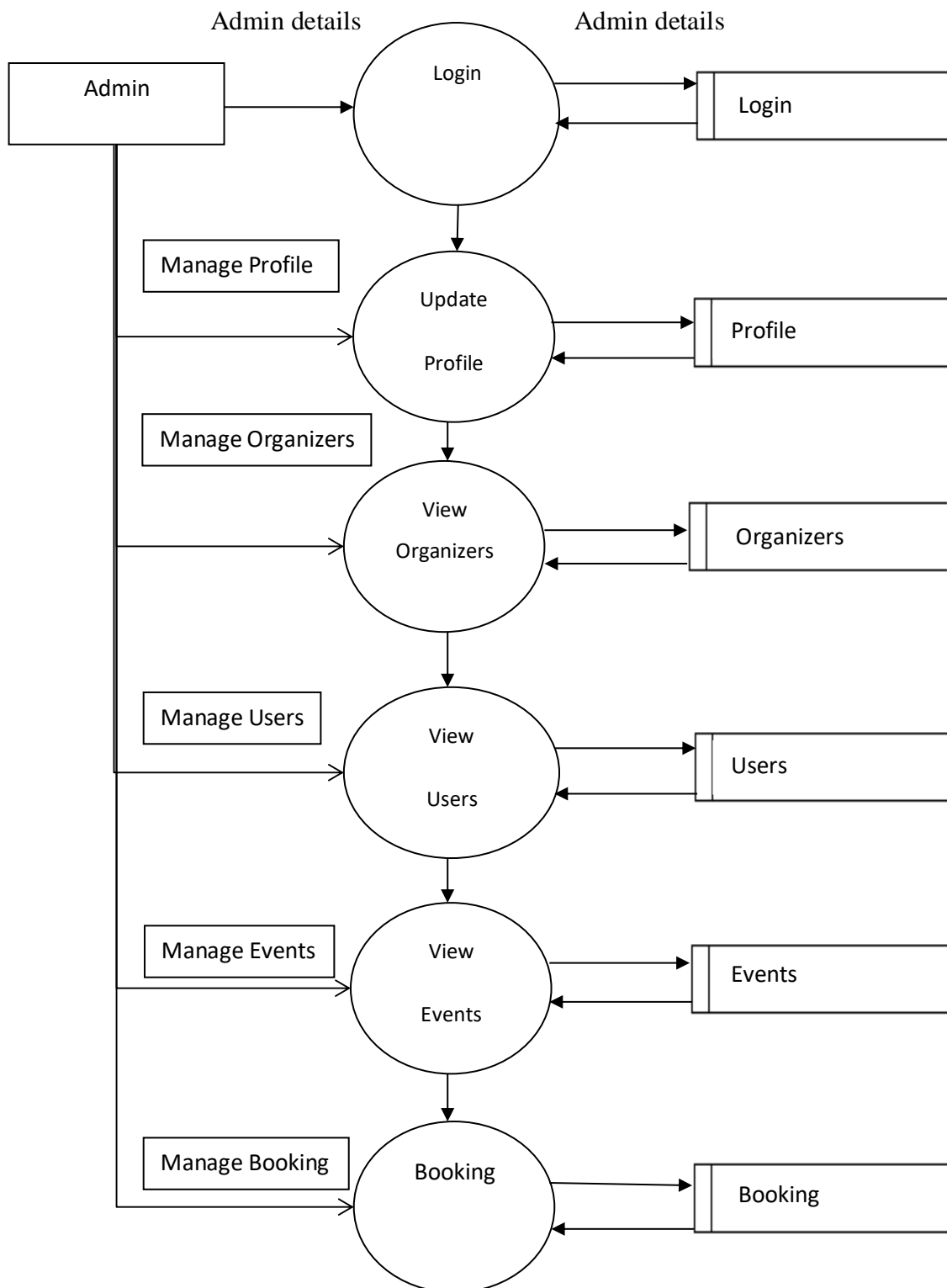
4.5 DATA FLOW DIAGRAM (DFD)

LEVEL 0

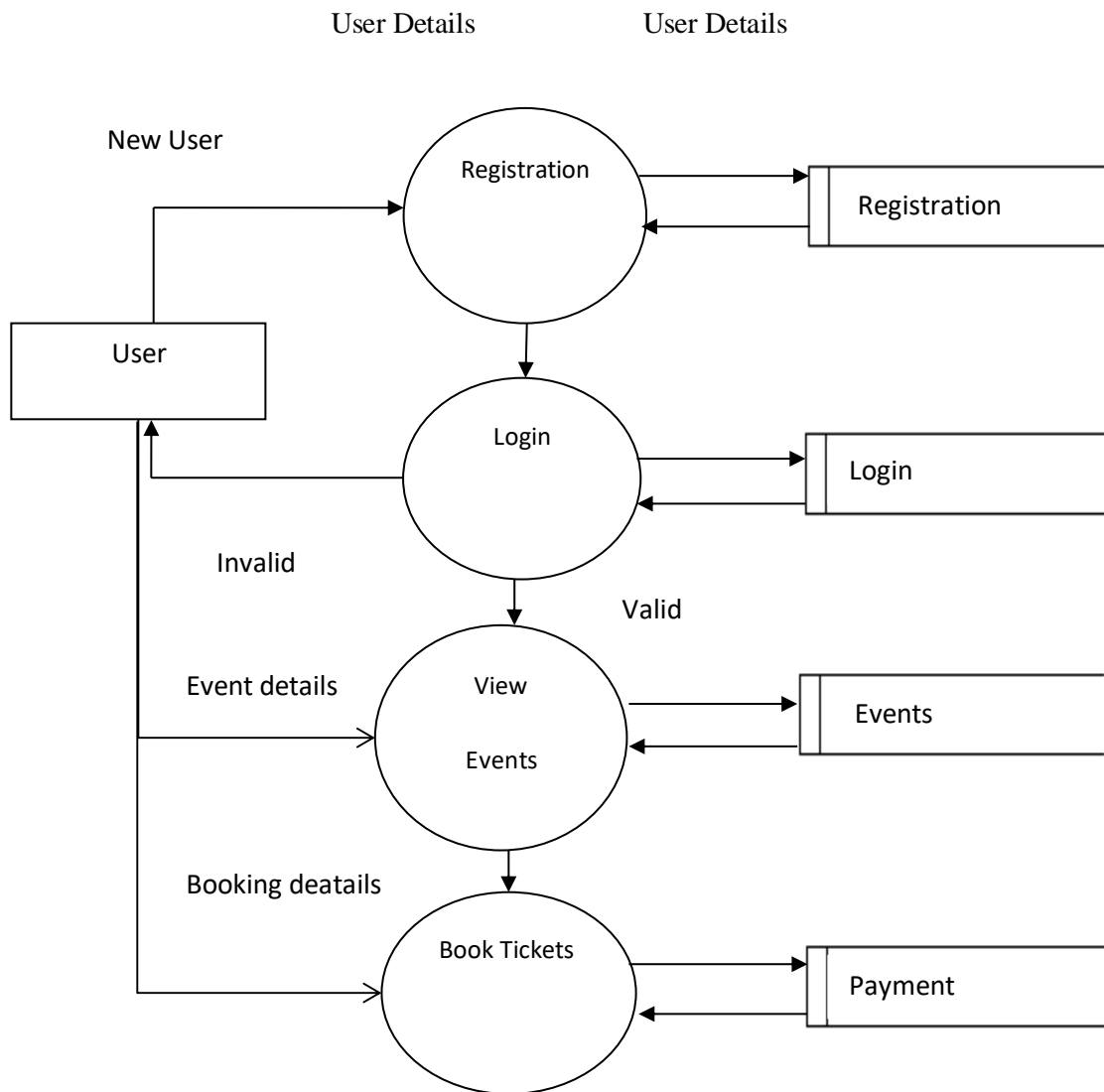
Interaction or Context Level diagram



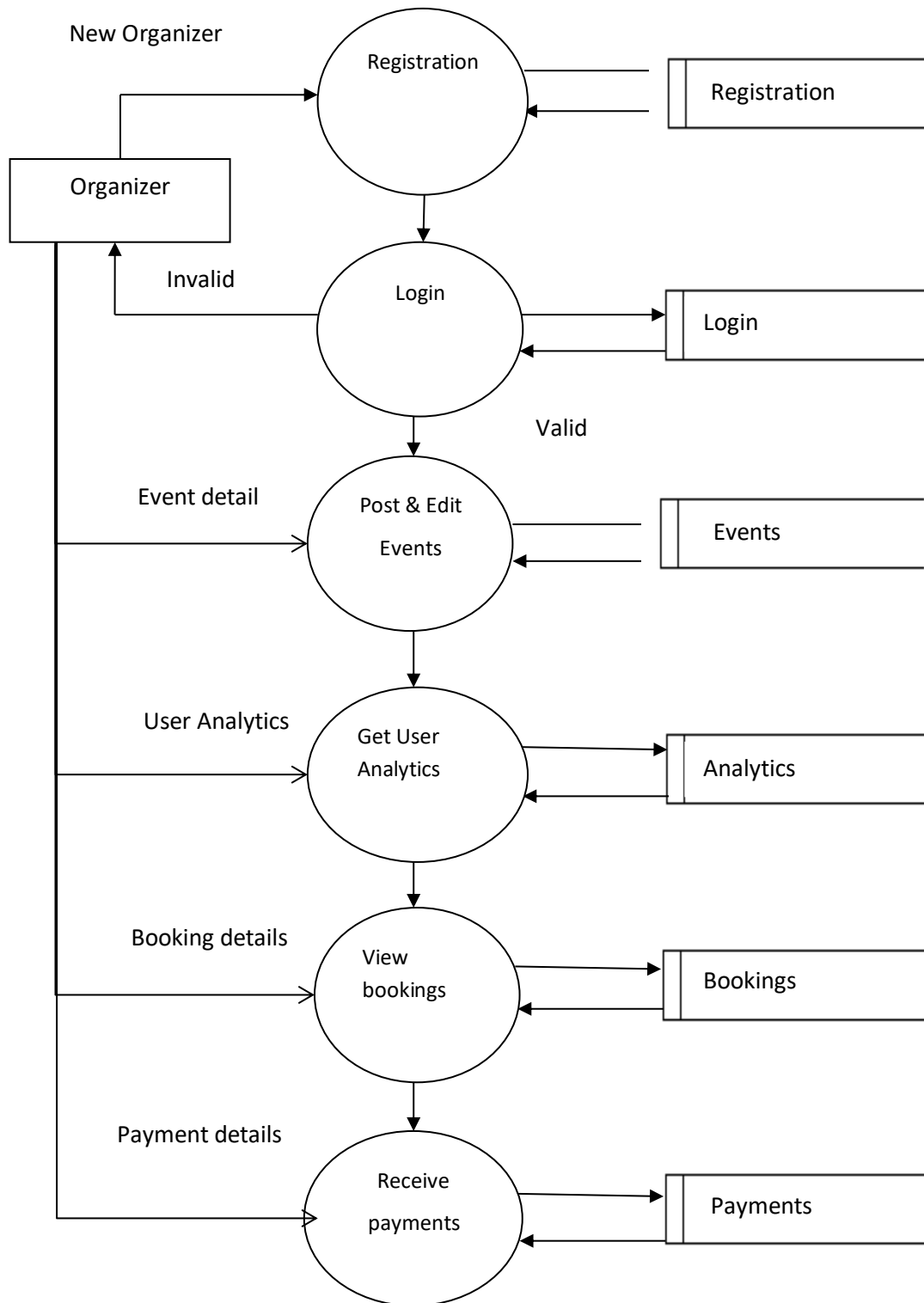
Level 1



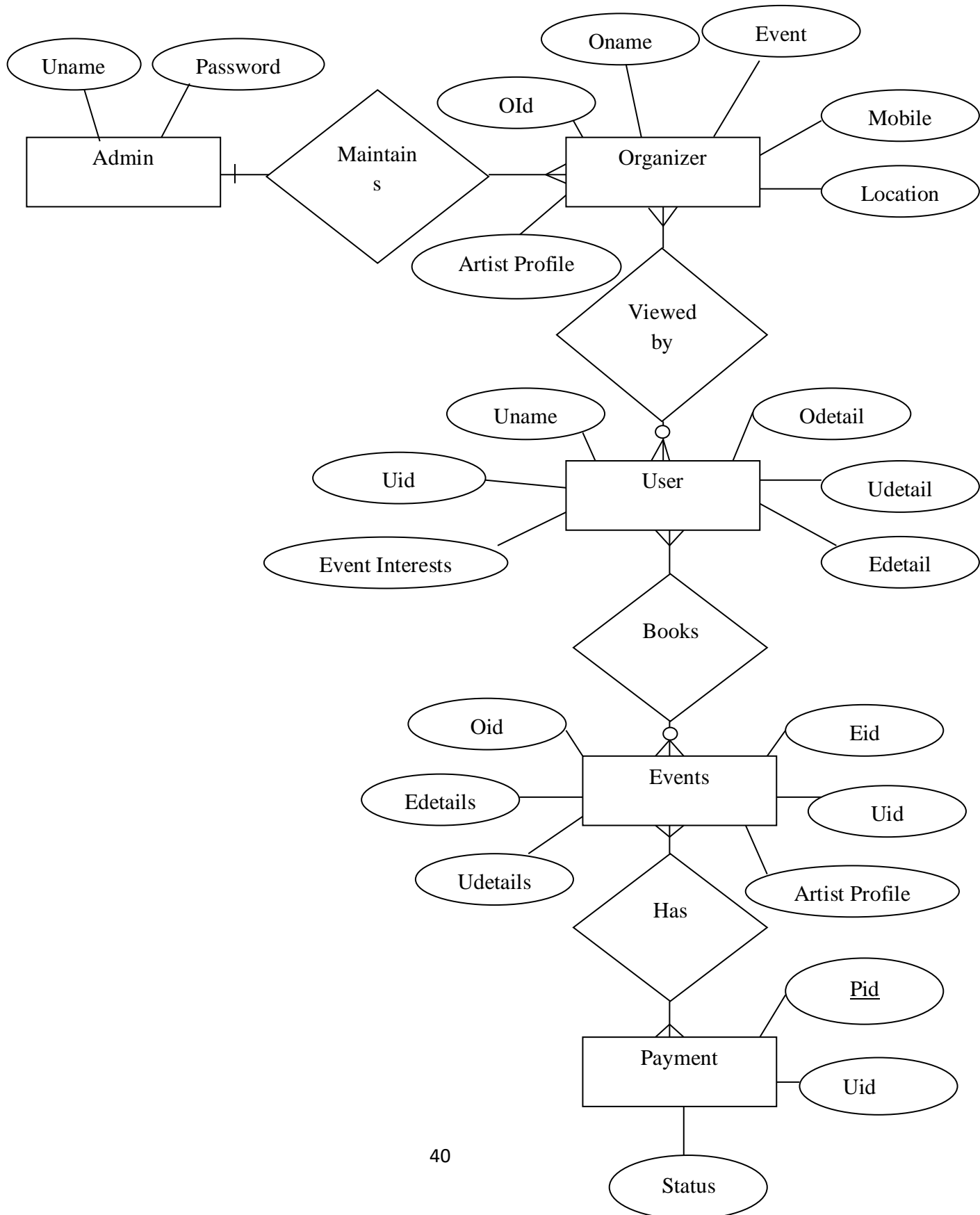
Level 2



Level 3



ER DIAGRAM



CHAPTER 5

SYSTEM TESTING AND IMPLEMENTATION

System testing and implementation are critical phases in the software development lifecycle to ensure that the "Event Finder" application functions correctly, meets user expectations, and operates seamlessly across different environments. The system undergoes rigorous testing to identify and resolve any defects before deployment. Implementation involves deploying the system for real-world use, ensuring smooth integration and functionality.

5.1 SYSTEM TESTING

5.1.1 Unit Testing

Unit testing is performed to check the functionality of individual components of the system.

Objective: Validate the correctness of core functions, like event search, user authentication, and ticket booking.

Approach: Each function or method is tested in isolation using frameworks like Jest or Mocha for **Next.js backend and frontend**.

Example Use Case:

- Testing the **event filtering function** to ensure that selecting "Music Concerts" only returns events tagged under music.
- Ensuring that entering incorrect credentials results in a **"Invalid username or password"** message.

5.1.2 INTEGRATION TESTING

Integration testing ensures that various system components (frontend, backend, and third-party services) work together properly.

Objective: Validate that different modules (event booking, user authentication, Google Maps, etc.) communicate seamlessly.

Approach: Tests focus on API responses, data synchronization, and session management.

Example Use Case:

- Ensuring that when a **user books an event**, the system updates the database and triggers a **confirmation email/SMS**.
- Checking whether clicking on an event's **location preview** correctly **opens Google Maps with navigation enabled**.

5.1.3 FUNCTIONAL TESTING

Functional testing ensures that every feature performs as per the project requirements.

Objective: Confirm that all user actions (searching, booking, navigating, etc.) yield correct outputs.

Approach: A set of test cases is created based on real-world user interactions.

Example Use Case:

- A user searches for a nearby sports event, applies filters (e.g., "free entry"), and books a ticket.
- The system should verify the availability, reserve a ticket, and display a "Booking Confirmed" message.
- If the event is fully booked, the system should show "No slots available" instead of proceeding with payment.

5.1.4 USER INTERFACE (UI) TESTING

UI testing ensures that the **Event Finder's** interface is visually consistent, responsive, and user-friendly.

Objective: Verify that UI components (buttons, dropdowns, event cards) function correctly and display information properly.

Approach: The application is tested on **different screen sizes (mobile, tablet, desktop)** and browsers.

Example Use Case:

- Ensuring that **hovering over an event** correctly **displays a Google Maps preview**.

- Checking that event **details and images load properly** without any distortion on all devices.

5.1.5 PERFORMANCE TESTING

Performance testing measures how well the system handles different levels of traffic and event data.

Objective: Ensure that the system remains stable and fast under high user loads.

Approach: Simulating thousands of concurrent users and event searches.

Example Use Case:

- Testing how the system **responds when 10,000 users** are searching for events simultaneously.
- Checking if the **live event updates** refresh without lag under heavy usage.

5.1.6 SECURITY TESTING

Security testing ensures that user data, payments, and event details are **protected against cyber threats**.

Objective: Identify and fix security loopholes to prevent **data breaches, unauthorized access, and fraudulent activities**.

Approach: The system is tested for vulnerabilities using **penetration testing tools and ethical hacking methods**.

Example Use Case:

- Testing whether a **user can access another user's booking details** by modifying the URL.
- Ensuring **SSL encryption is implemented for payment transactions**.

5.1.7 ACCEPTANCE TESTING

Acceptance testing checks whether the **Event Finder** application meets all business and user requirements before deployment.

Objective: Ensure that the final product is **ready for launch** and meets stakeholder expectations.

Approach: Real users (event organizers, attendees) test the system and provide feedback.

Example Use Case:

- An **event organizer posts an event**, users book tickets, and the system successfully generates booking confirmation.
- Testing the **calendar feature** to verify if past and upcoming events are displayed correctly.

5.1.8 TEST CASES

Test cases provide structured test scenarios with predefined inputs and expected outputs to validate system behavior. Below are key test cases executed for the system:

1. User Authentication Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
UA01	Login with valid credentials	1. Open login page 2. Enter valid email & password 3. Click on "Login"	User is redirected to the homepage	Pass/Fail
UA02	Login with invalid credentials	1. Open login page 2. Enter incorrect password 3. Click on "Login"	Error message: "Invalid username or password"	Pass/Fail
UA03	Password reset functionality	1. Click on "Forgot Password" 2. Enter registered email 3. Click on "Reset Password"	Password reset email is sent	Pass/Fail
UA04	Register a new user	1. Open registration page 2. Enter valid details 3. Click on "Register"	User account is created successfully	Pass/Fail
UA05	Login with Google authentication	1. Click "Login with Google" 2. Choose Google account	User is logged in via Google	Pass/Fail

2. Event Search & Filtering Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
ES01	Search for an event by keyword	1. Open search bar 2. Type event name (e.g., "Music Concert") 3. Press Enter	Relevant events are displayed	Pass/Fail
ES02	Apply category filter	1. Select "Sports" from category filter 2. Click "Apply"	Only sports events are shown	Pass/Fail
ES03	Search for a non-existent event	1. Search for "xyz123" 2. Press Enter	Message: "No events found"	Pass/Fail
ES04	Filter events by date	1. Select a date from the calendar filter 2. Click "Search"	Events matching the selected date appear	Pass/Fail

3. Event Booking Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
EB01	Book a ticket for an event	1. Click on an event 2. Click "Book Now" 3. Enter payment details 4. Confirm booking	Booking confirmation is displayed and email is sent	Pass/Fail
EB02	Attempt booking when seats are full	1. Select a fully booked event 2. Click "Book Now"	Message: "No seats available"	Pass/Fail
EB03	Cancel a booked ticket	1. Navigate to "My Bookings" 2. Click "Cancel Ticket" 3. Confirm cancellation	Booking is canceled, and refund process starts	Pass/Fail
EB04	Payment failure handling	1. Enter incorrect payment details 2. Click "Pay"	Message: "Payment failed. Please try again"	Pass/Fail

4. Google Maps & Navigation Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
GM01	Open event location on Google Maps	1. Click on event location 2. Click "Open in Maps"	Google Maps opens with the correct location	Pass/Fail
GM02	Get directions to the event	1. Click "Get Directions" 2. Select current location	Google Maps shows directions	Pass/Fail
GM03	Search for nearby events	1. Click "Nearby Events" 2. Allow location access	Events near user's location are displayed	Pass/Fail

5. Notification & Email Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
NT01	Receive booking confirmation email	1. Book a ticket for an event	Email with booking details is received	Pass/Fail
NT02	Receive event reminder notification	1. Book an event happening tomorrow 2. Wait for the reminder time	Notification is received 24 hours before the event	Pass/Fail
NT03	Receive cancellation confirmation email	1. Cancel a booked ticket	Email confirming cancellation is received	Pass/Fail

6. Performance & Load Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
PT01	Handle 10,000 concurrent users	1. Simulate 10,000 users searching events at once	System does not crash, and response time is under 2s	Pass/Fail
PT02	Handle 1000 event bookings in 1 minute	1. Simulate 1000 users booking events simultaneously	System processes all bookings correctly	Pass/Fail

7. Security Test Cases

Test Case ID	Test Case Description	Test Steps	Expected Result	Status
ST01	Attempt SQL injection in search bar	1. Enter DROP TABLE users;-- in search bar	System prevents SQL injection	Pass/Fail
ST02	Attempt unauthorized access	1. Try accessing /admin without login	User is redirected to login page	Pass/Fail
ST03	Test data encryption	1. Inspect network traffic for payment details	Payment data is encrypted using HTTPS	Pass/Fail

5.2 IMPLEMENTATION AND MAINTENANCE

The **Implementation and Maintenance** phase is crucial in ensuring the successful deployment and long-term sustainability of the **Event Finder System**. This phase involves deploying the system in a live environment, training users, monitoring performance, fixing issues, and upgrading the system as needed.

5.2.1 Implementation

Implementation is the process of making the system available for use. This involves **installing software, configuring servers, migrating data, and training users**. The

implementation strategy depends on the project requirements and may include **direct cutover, parallel running, phased implementation, or pilot implementation.**

1 Implementation Strategies

1. Direct Cutover:

- The old system is completely replaced by the new system at once.
- Suitable for small-scale systems with minimal risk.
- High-risk if issues arise, as there is no backup system.

2. Parallel Implementation:

- Both the old and new systems run simultaneously for a specific period.
- Reduces risk but increases operational costs and workload.
- Ensures smooth transition while users adapt to the new system.

3. Phased Implementation:

- The system is implemented in stages (e.g., module by module).
- Allows gradual transition and easier troubleshooting.
- Suitable for large systems with multiple components.

4. Pilot Implementation:

- The system is deployed to a small user group before full rollout.
- Used for testing the system in a controlled environment.
- Ensures that major issues are identified and fixed before complete deployment.

2 Steps in Implementation

1. Server Setup & Hosting

- Configuring cloud or on-premise servers (AWS, Azure, DigitalOcean, etc.).
- Deploying the database (PostgreSQL) and backend (Next.js).
- Setting up domain and SSL security for secure transactions.

2. Database Migration & Integration

- Transferring existing event data, user accounts, and previous bookings.
- Ensuring database integrity and optimizing queries for performance.

3. Frontend Deployment

- Deploying the **Next.js frontend** on a cloud platform (e.g., Vercel, Netlify).
- Ensuring UI responsiveness and compatibility across devices.

4. Testing in Live Environment

- Performing **beta testing** with a small group of users.
- Checking system performance, security, and user experience.
- Resolving any issues before full-scale deployment.

5. User Training & Documentation

- Providing **training sessions, manuals, and FAQs** for users.
- Conducting **webinars, tutorials, and live demos** for event organizers and customers.

6. Go-Live & Post-Implementation Support

- Official launch of the system for all users.
- 24/7 technical support for handling issues during the initial period.

5.2.2 Maintenance

System maintenance ensures that the **Event Finder System** remains operational, secure, and updated with the latest technology. Maintenance activities include **bug fixes, feature enhancements, performance optimizations, and security updates**.

1 Types of Maintenance

1. Corrective Maintenance:

- Fixing bugs, errors, and security vulnerabilities reported by users.
- Example: Resolving **login failures** or **incorrect event details** in the system.

2. Adaptive Maintenance:

- Updating the system to comply with **new technology or legal requirements**.
- Example: Adding **GDPR compliance** features for better data privacy.

3. Perfective Maintenance:

- Enhancing **performance, UI design, and user experience**.
- Example: Improving **search algorithms** for better event recommendations.

4. Preventive Maintenance:

- Proactive monitoring and issue detection before failures occur.
- Example: Running **automated health checks** to detect **database overloads**.

2 Maintenance Activities

1. Regular Software Updates

- Ensuring **Next.js, PostgreSQL, and APIs** are up-to-date.
- Updating **third-party libraries and dependencies** to prevent vulnerabilities.

2. Performance Optimization

- Identifying and fixing slow database queries.
- Improving **server response time and load balancing**.

3. Security Monitoring & Threat Detection

- Running **penetration tests** to identify security loopholes.

- Implementing **firewalls, encryption, and multi-factor authentication (MFA)**.

4. **Backup & Disaster Recovery**

- Automating **daily database backups** to prevent data loss.
- Implementing **disaster recovery plans** to restore services in case of system failure.

5. **User Feedback & Feature Enhancements**

- Gathering user feedback through **surveys and support tickets**.
- Implementing **new features based on user suggestions**.
- Example: Adding **AI-powered event recommendations** based on user preferences.

6. **Log Analysis & Error Tracking**

- Using **monitoring tools like New Relic, Datadog, or ELK Stack** to track errors.
- Setting up **automated alerts** for system failures.

3 **Future Enhancements**

1. **AI-powered Personalized Recommendations**

- Using machine learning algorithms to suggest events based on user interests.

2. **Blockchain-based Ticketing System**

- Implementing **NFT-based tickets** for fraud prevention.

3. **Voice Assistant Integration**

- Allowing users to search events using voice commands via **Google Assistant or Siri**.

4. Augmented Reality (AR) for Virtual Event Previews

- Providing users with AR-based previews of event venues before booking.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The **Event Finder System** successfully addresses the challenges faced by users in discovering and managing events efficiently. By leveraging modern web technologies such as **Next.js for the frontend, PostgreSQL for the database, and advanced API integrations**, the system provides a **seamless, secure, and user-friendly** experience for event organizers and attendees.

The implementation of **real-time updates, personalized recommendations, and an intuitive user interface** enhances user engagement, making event discovery and management **faster and more efficient**. Security measures, including **data encryption, authentication mechanisms, and regular system updates**, ensure that user information remains protected.

Through rigorous **testing methodologies**, including **unit testing, integration testing, and user acceptance testing (UAT)**, the system has been validated for reliability and performance. The structured **maintenance plan** ensures continuous improvements, incorporating **user feedback and emerging technologies** to enhance system capabilities further.

In conclusion, the **Event Finder System** provides a **robust and scalable** solution that simplifies the process of **event organization, ticketing, and participation**. With **future enhancements** such as **AI-powered event recommendations, blockchain-based ticketing, and AR-based venue previews**, the system is well-positioned to adapt to evolving user needs and industry trends.

6.2 FUTURE ENHANCEMENT

The **Event Finder System** has been designed with scalability and adaptability in mind. As technology advances and user expectations evolve, several **future enhancements** can be integrated to further improve the platform's efficiency, security, and user engagement. Below are some of the key areas for future development:

1. AI-Powered Event Recommendations

- Implementing **machine learning algorithms** to analyze user behavior and preferences.
- Providing **personalized event suggestions** based on location, past attendance, and interests.
- Enhancing event discovery with **predictive analytics** to suggest trending and popular events.

2. Augmented Reality (AR) for Venue Previews

- Allowing users to explore event venues using **AR technology** before attending.
- Providing **360-degree virtual tours** of event locations for better decision-making.
- Enabling **interactive venue mapping** for easy navigation inside large event spaces.

3. Blockchain-Based Ticketing System

- Ensuring **secure, tamper-proof, and transparent** ticketing using **blockchain technology**.
- Reducing **ticket fraud and unauthorized resales** by enabling smart contract-based transactions.
- Introducing **NFT-based event tickets** for exclusive access and digital collectibles.

4. Voice Assistant and Chatbot Integration

- Implementing AI-powered **voice assistants** and chatbots for **quick event searches and booking assistance**.
- Enabling users to inquire about events using **natural language processing (NLP)**.
- Providing **24/7 customer support** for event-related queries and troubleshooting.

5. Enhanced Security and Biometric Authentication

- Introducing **fingerprint and facial recognition** for secure logins and event check-ins.
- Enhancing **multi-factor authentication (MFA)** for user accounts to prevent unauthorized access.
- Implementing **end-to-end encryption** to protect user data and transaction details.

6. Integration with Wearable Devices

- Developing **smartwatch and fitness band integration** for event notifications and reminders.
- Enabling **contactless event check-ins** using NFC-enabled smartwatches.
- Tracking user activity at fitness or sports events using **real-time health monitoring**.

7. Gamification and Loyalty Programs

- Introducing **reward-based engagement** where users earn points for attending events.
- Implementing **leaderboards, badges, and exclusive discounts** for frequent attendees.
- Encouraging **social sharing** of event experiences with incentives.

8. Multi-Language and Localization Support

- Adding **multi-language support** to cater to users from diverse linguistic backgrounds.
- Providing **localized event recommendations** based on regional interests and trends.
- Offering **currency conversion for ticket payments** to support international users.

9. Integration with Smart Home and IoT Devices

- Enabling **smart home assistants (Google Home, Alexa)** to provide event notifications.
- Allowing users to **control event-related settings** (e.g., reminders, ticket scanning) via IoT devices.
- Enhancing accessibility with **voice-controlled commands** for users with disabilities.

10. Social Media and Community Features

- Enabling **live event sharing** on social media platforms for real-time engagement.
- Providing **community forums and discussion groups** for event-related interactions.
- Integrating **event-based challenges and contests** to boost user participation.

CHAPTER 7

BIBLIOGRAPHY AND REFERENCES

BIBLIOGRAPHY

Books:

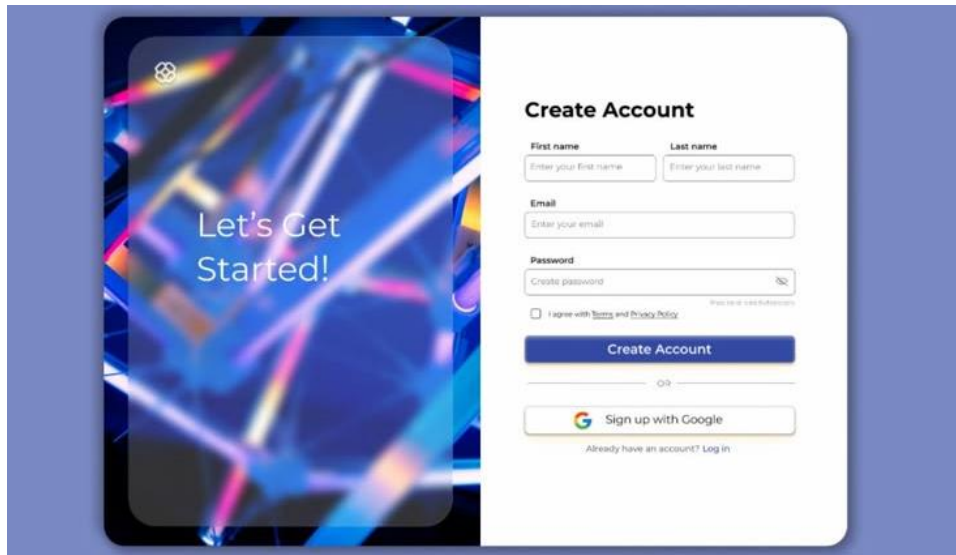
- **Pressman, R. S., & Maxim, B. R.** (2019). *Software Engineering: A Practitioner's Approach (9th Edition)*. McGraw-Hill Education.
- **Sommerville, I.** (2015). *Software Engineering (10th Edition)*. Pearson Education.
- **Kurose, J. F., & Ross, K. W.** (2020). *Computer Networking: A Top-Down Approach (8th Edition)*. Pearson Education.
- **Elmasri, R., & Navathe, S. B.** (2015). *Fundamentals of Database Systems (7th Edition)*. Pearson.
- **Nielsen, J.** (2012). *Usability Engineering*. Morgan Kaufmann.

REFERENCES

- **Google Maps API Documentation** – <https://developers.google.com/maps/documentation>
- **Next.js Documentation** – <https://nextjs.org/docs>
- **PostgreSQL Documentation** – <https://www.postgresql.org/docs/>
- **TypeScript Official Documentation** – <https://www.typescriptlang.org/docs/>
- **React.js Documentation** – <https://reactjs.org/docs/getting-started.html>
- **Payment Gateway API Documentation** – (e.g., PayPal, Stripe)
<https://developer.paypal.com/docs/api/overview/>
- **IEEE Research Papers on Event Management Systems and AI-powered Recommendations**
- **Stack Overflow Discussions for Troubleshooting and Implementation Issues**
- **GitHub Repositories for Open-Source Event Management Projects**

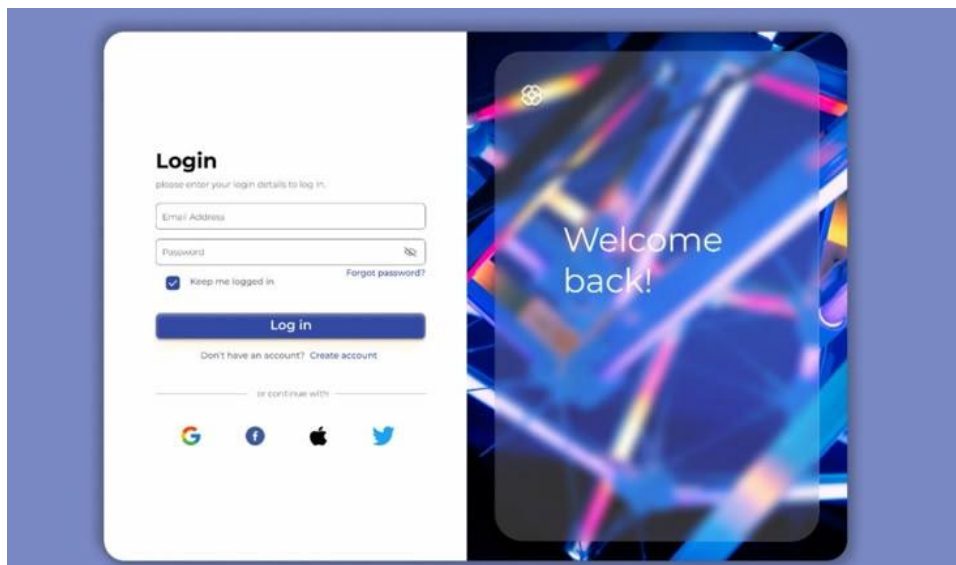
APPENDICES

A. Reports



The image shows a user registration interface. On the left is a decorative panel with a blue and purple abstract background and the text "Let's Get Started!". On the right is a white "Create Account" form. The form includes input fields for "First name", "Last name", "Email", and "Password". Below the password field is a checkbox for "I agree with Terms and Privacy Policy" and a link to "View our privacy policy". A blue "Create Account" button is positioned below the form. At the bottom, there is a "Sign up with Google" button and a link for "Already have an account? Log in".

Figure 2 – User Registration



The image shows a login interface. On the left is a white "Login" form. It includes input fields for "Email Address" and "Password", a "Keep me logged in" checkbox, and a "Forgot password?" link. A blue "Log in" button is at the bottom of the form. Below the button is a link for "Don't have an account? Create account". At the very bottom, there is a section for "or continue with" followed by social media icons for Google, Facebook, Apple, and Twitter. On the right is a decorative panel with a blue and purple abstract background and the text "Welcome back!".

Figure 3 – Login page

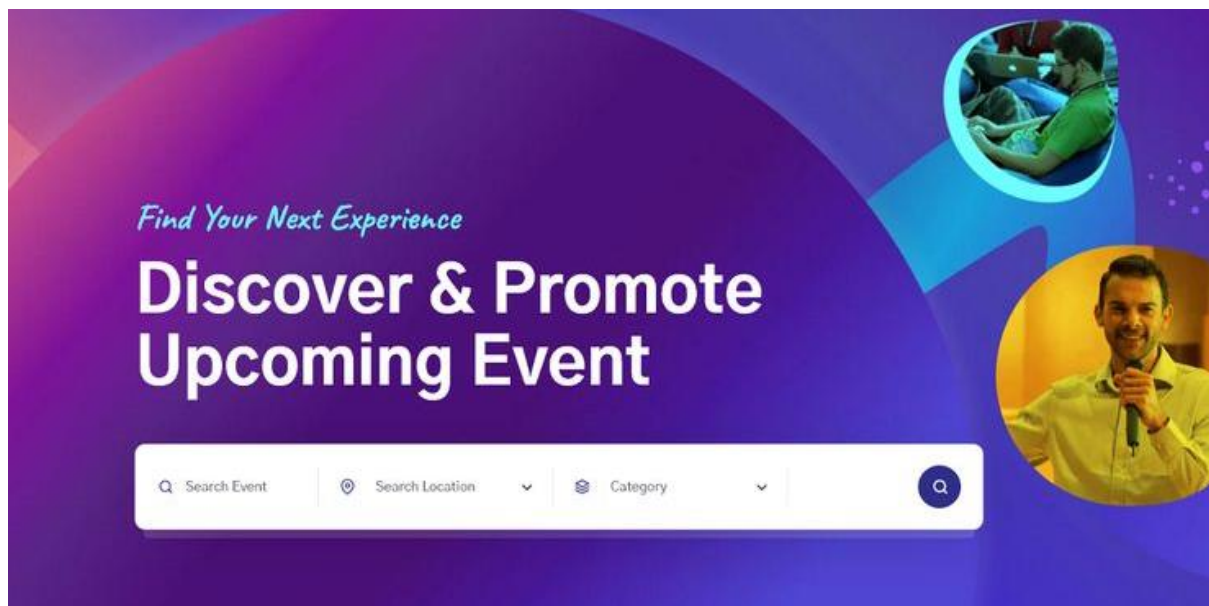


Figure 4 – Home page

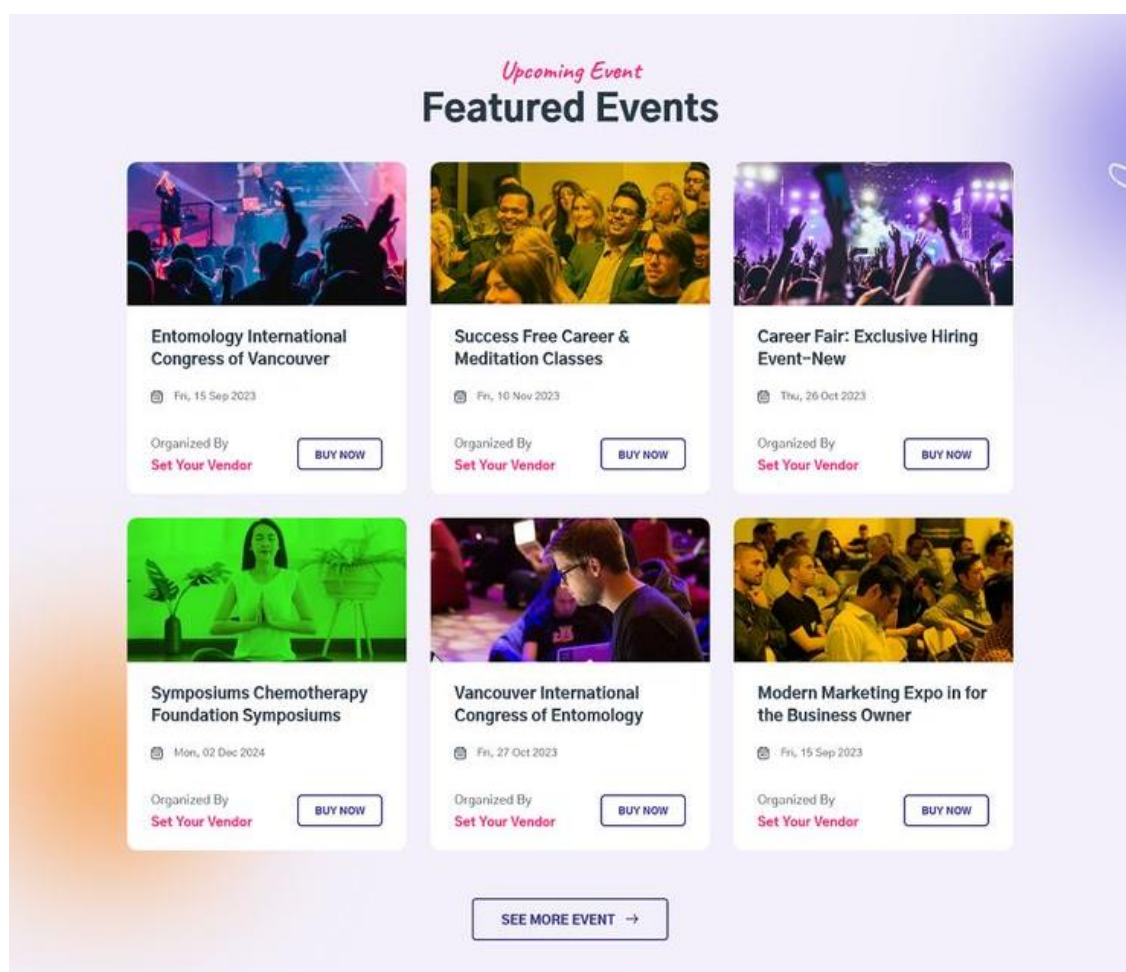


Figure 5 – Event Page



Figure 6 – Timer/Schedule

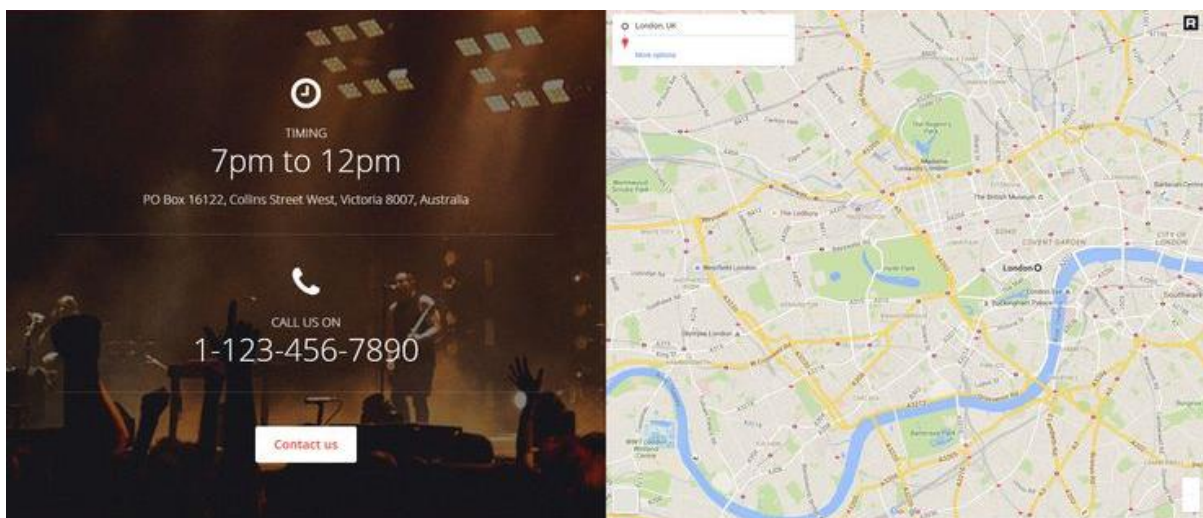



Figure 7 – Google Map

Create Event

Event title

Category

Description



Drag photo here

SVG, PNG, JPG

Select from computer

Event location or Online

Start Date: 12/13/2023 4:56 PM

End Date: 12/13/2023 4:56 PM

Price

Free Ticket ☐

URL

Create Event


Figure 8 – Event creation Page

Update Event

GitHub Universe 2024

Category

Universe 23 is about AI, security, and the developer experience. about how to spark innovation, stay in the flow, optimize collaboration, and prevent vulnerabilities with AI-powered security.



Online

Start Date: 12/19/2024 12:25 PM

End Date: 12/26/2024 12:25 PM

\$ 999

Free Ticket ☐

<https://jsmastery.pro>

Update Event

Figure 9 – Event update Page

61

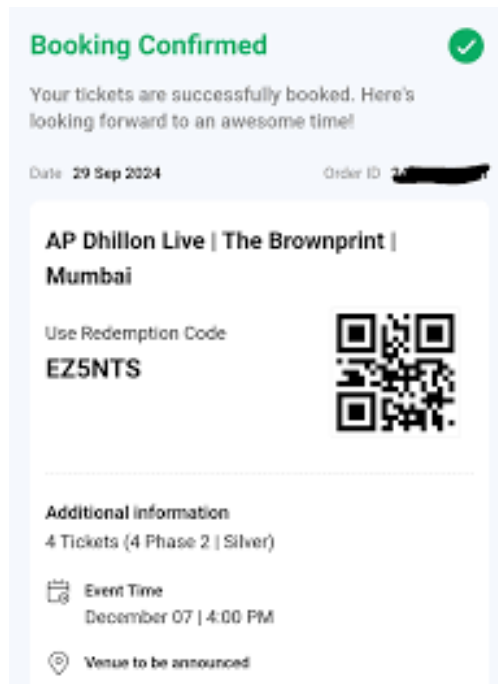


Figure 10 – Registration Form

B. Sample Code

```
// Import required modules

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");


const app = express();

app.use(express.json()); // Middleware to parse JSON requests

app.use(cors()); // Enable Cross-Origin Resource Sharing


// Connect to MongoDB

mongoose.connect("mongodb://localhost:27017/eventfinder", {

  useNewUrlParser: true,
```



```
    useUnifiedTopology: true,  
  })  
  
  .then(() => console.log("MongoDB Connected"))  
  
  .catch(err => console.error("MongoDB Connection Failed:", err));
```

```
// Event Schema Definition
```

```
const eventSchema = new mongoose.Schema({  
  
  title: { type: String, required: true },  
  
  description: { type: String, required: true },  
  
  date: { type: Date, required: true },  
  
  location: { type: String, required: true },  
  
  category: { type: String, required: true },  
  
  price: { type: Number, default: 0 },  
  
  isPaid: { type: Boolean, default: false },  
  
  organizer: { type: String, required: true },  
  
  createdAt: { type: Date, default: Date.now }  
  
});
```

```
// Event Model
```

```
const Event = mongoose.model("Event", eventSchema);
```

```
// Route: Create a New Event
```

```
app.post("/api/events", async (req, res) => {
```

```

    try {

        const newEvent = new Event(req.body);

        await newEvent.save();

        res.status(201).json({ message: "Event created successfully", event: newEvent });

    } catch (error) {

        res.status(500).json({ error: "Failed to create event", details: error.message });

    }

});

```

// Route: Get All Events

```

app.get("/api/events", async (req, res) => {

    try {

        const events = await Event.find();

        res.status(200).json(events);

    } catch (error) {

        res.status(500).json({ error: "Failed to fetch events", details: error.message });

    }

});

```

// Route: Get Single Event by ID

```

app.get("/api/events/:id", async (req, res) => {

    try {

        const event = await Event.findById(req.params.id);

```

```

    if (!event) return res.status(404).json({ error: "Event not found" });

    res.status(200).json(event);

  } catch (error) {

    res.status(500).json({ error: "Error fetching event", details: error.message });

  }

});

// Route: Update an Event by ID

app.put("/api/events/:id", async (req, res) => {

  try {

    const updatedEvent = await Event.findByIdAndUpdate(req.params.id, req.body, { new:
true });

    if (!updatedEvent) return res.status(404).json({ error: "Event not found" });

    res.status(200).json({ message: "Event updated successfully", event: updatedEvent });

  } catch (error) {

    res.status(500).json({ error: "Failed to update event", details: error.message });

  }

});

// Route: Delete an Event by ID

app.delete("/api/events/:id", async (req, res) => {

  try {

    const deletedEvent = await Event.findByIdAndDelete(req.params.id);

```

```
    if (!deletedEvent) return res.status(404).json({ error: "Event not found" });

    res.status(200).json({ message: "Event deleted successfully" });

  } catch (error) {

    res.status(500).json({ error: "Failed to delete event", details: error.message });

  }

});
```

```
// Start the Server
```

```
const PORT = process.env.PORT || 5000;
```

```
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```