DEVELOPMENT OF WEB-BASED TICKETING SYSTEM WITH QR CODE AND REAL-TIME ATTENDANCE TRACKING

BY

IJEH WISDOM ISIOMA

PSC2207916

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

NOVEMBER 2025

CERTIFICATION

This is to certify that the project titled "**Web-Based Ticketing System with QR Code and Real-Time Attendance Tracking**" was carried out by **IJEH WISDOM ISIOMA**, with matriculation number **PSC2207916**, in partial fulfillment of the requirements for the award of the degree of Bachelor of Science (B.Sc.) in Computer Science at the University of Benin, Edo State, Nigeria.

_____ _____

IJEH WISDOM ISIOMA **DATE**

Student

_____ _____

Prof. F. O. CHETE **DATE**

Project Supervisor

_____ _____

DR. MAXWELL OSAGIE **DATE**

Project Coordinator

_____ _____

DR. ROSEMARY USIOBAFO **DATE**

Head of Department

APPROVAL PAGE

This project report titled **"Web-Based Ticketing System with QR Code and Real-Time Attendance Tracking"** by **IJEH WISDOM ISIOMA**, with matriculation number **PSC2207916**, has been read and approved as meeting the requirements for the award of the degree of Bachelor of Science (B.Sc.) in Computer Science, University of Benin, Edo State, Nigeria.

DEDICATION

This project is dedicated to God Almighty, whose grace, wisdom, and strength made this work possible. It is also dedicated to my parents Mr. and Mrs. IJEH, for their constant love, encouragement, and support throughout the course of this project and my academic journey.

ACKNOWLEDGMENT

I am deeply grateful to God Almighty for His guidance, wisdom, and strength throughout the course of this project and my entire academic journey.

My sincere appreciation goes to my supervisor, **Prof. F. O. CHETE**, for his, support, and valuable guidance during the development of this project. Your constructive feedback and encouragement helped me complete this work successfully.

I would also like to thank the **Head of Department**, all **lecturers** and **staff** of the Department of Computer Science, **University of Benin**, for their efforts in imparting knowledge and providing a conducive learning environment.

ABSTRACT

This project focuses on building a web-based ticketing system that uses QR codes and real-time attendance tracking to make event management faster, safer, and more efficient. The goal is to replace the old, paper-based ticketing method with a digital solution that allows users to buy, manage, and verify tickets online.

When a user purchases a ticket, the system automatically generates a unique QR code that contains their event and ticket details. At the event venue, organizers can scan this code using any device with a camera to confirm the ticket's authenticity. Once scanned, the ticket's status changes immediately, preventing anyone from reusing it.

The platform includes three main roles: Admin, Organizer, and User. Admins handle event approvals, monitor activities, and manage users. Organizers can create and manage their own events, while regular users can browse upcoming events, buy tickets, and even apply to become organizers if they want to host events.

The system is built as a responsive web application, meaning it works well on both computers and mobile devices. It also updates attendance data in real-time, giving organizers an accurate view of how many people have checked in. Overall, this project provides a modern, secure, and eco-friendly way to handle event ticketing — proving how web technology and QR codes can simplify event registration and attendance management in today's digital world.

CHAPTER ONE
INTRODUCTION

1.1 Background to the Study

For as long as people have organized events, whether private or public, there has always been a need for an effective crowd control mechanism to manage attendees. In the early days, this often involved simple methods such as headcounts or tallying at the entrance. As events became larger and more structured, ticketing systems were introduced to regulate entry and improve overall organization (Bello and Yusuf 2018).

The first ticketing systems were purely mechanical. Paper tickets printed with images or serial numbers were sold at designated outlets or venue counters, and attendees were required to present them at the point of entry. This method worked well for small gatherings but quickly revealed significant limitations as event sizes increased to hundreds or thousands. According to Bala and Adamu (2020), physical ticketing led to problems such as long queues, human error, and ticket forgery. Fraud became common, as counterfeit tickets could easily be produced using simple tools and printing equipment. Organizers also struggled to maintain accurate attendance records, relying mainly on paper stubs that could only be counted after the event had concluded. This provided historical information but offered no real-time insight into ongoing attendance.

The rise of the internet in the late 1990s and early 2000s ushered in the first wave of digital ticketing, allowing users to purchase tickets online rather than at physical locations (Eze and Sharma 2019). This transition represented a major step forward, as buyers could print confirmation slips or display digital copies on their devices. However, the verification process at event venues remained largely manual. Event staff had to cross-check printed lists of names, a slow and error-prone process that often led to long entry queues and inaccurate attendance counts. Nwachukwu et al. (2020) observed that these early systems improved convenience but lacked automation in validation and live data updates.

Over time, online ticketing evolved into a massive global industry. Reports have shown that the event ticketing market is now worth billions of dollars annually, reflecting the public's growing preference for digital convenience (Okafor and Lin 2022). Yet, as sales increased, so did the need for faster, more secure, and fraud-resistant systems. The main challenges identified in recent studies include delayed validation times, duplicated entries, and limited real-time monitoring of attendee data (Akinyemi and Thomas 2021).

Technological innovations such as QR codes have been particularly revolutionary in solving many of these problems. A QR code (Quick Response code) is a two-dimensional barcode that

can store significantly more data than traditional barcodes (Ibrahim and Chen 2021). It can encode information such as ticket numbers, buyer identities, event tags, and encrypted verification keys. When scanned at the event entrance using a smartphone or specialized scanner, the code is verified against a central database in real time. This makes counterfeiting almost impossible and reduces check-in times from several minutes to just a few seconds (Bala and Adamu 2020).

In addition to fraud prevention, modern event organizers now require live attendance tracking as part of event security and management. Real-time monitoring helps ensure that crowd sizes remain within venue capacity limits and comply with safety regulations (Adeola and Johnson 2022). This became even more critical during the COVID-19 pandemic when accurate attendance data supported crowd spacing, emergency alerts, and contact tracing (Eze and Sharma 2019).

Today’s web-based ticketing systems integrate multiple features into a single, seamless platform. Customers can purchase tickets online, receive a QR code instantly, and have their entry validated in under a second. For event organizers, such systems provide live dashboards that show attendance counts, entry times, and other vital metrics, which can be used to evaluate the event's success and inform future planning (Nwachukwu et al., 2020).

This research builds on these developments by developing a secure and efficient web-based ticketing platform that uses QR codes for real-time attendance tracking. The goal is to enhance the overall effectiveness of event management by reducing fraud, improving data accuracy, and ensuring fast and reliable validation of attendees (Bala and Adamu 2020).

1.2 Statement of the Problem

Despite the promise of digital transformation, at a macro level, today's event management landscape is not digital. Though many organizers have adopted online ticket sales, the path from a purchase to entry is still typically fragmented, inefficient, and full of holes that threaten

security and an effective event management system. The pain of buying a ticket online is too often followed by antiquated manual verification on arrival, attendees are made to wait in long lines at the venue, where staff laboriously check off names from printouts or poorly integrated databases. This clogs up the event, as people take time searching instead of enjoying their day, and staff have to deal with processing complaints and production problems.

It is alarming that attendees are susceptible to ticket scams. For systems that create static codes or basic PDF confirmations, it is possible to create and distribute counterfeit copies. If the ticket could not be linked back to that issuing database in real time, fraudsters could get their hands on tickets that organizers thought they had secured but couldn't differentiate between genuine tickets and fakes. This leads to financial losses for organizers and reduced trust among attendees, causing overcrowded venues with poor security measures for the events. With the price of event tickets rising, the motivation for fraudulent duplication will only grow, and as long as there are no better validation alternatives in place to prevent this from occurring on a large scale, it will be an expensive fact of life.

A further major shortfall in a lot of current systems is the lack of real-time attendance tracking. Sale number data is typically available to organizers, but not a real-time recording of the number of people cleared out and in the front door. This makes it challenging to track capacity, direct foot traffic, and optimize staffing. It also misses the flexibility demands to trigger quick routes of operation struggles, like unexpected spikes in arrivals or partially dense areas within a venue. This absence of real-time data makes such environments a significant compliance risk when safety regulations and maximum occupancy limits are strictly governed.

The other issue is that ticketing, verification, and analytics cannot talk to each other, providing at best marginal operational insights. For instance, a standalone visitor management system is unable to analyze attendance patterns, predict peak entry times, track no-shows, or measure the ROI on marketing initiatives. Not only does this lack of insight mean ongoing events are always doomed to fail, but the future is going to be nothing short of an endless cycle of underwhelming mechanisms reflexively rolling back into capture.

In order to address these shortcomings, event operations need to be more secure from fraud, more efficient, and not compromised by the live insights new technology should effortlessly bring.

1.3 Aim and Objectives of the Study

Aim of the Study

This study focuses on building a cost-effective and time-efficient web-based ticketing solution integrated with QR code and real-time attendance to increase the efficiency of managing an event, secure ticket verification process, as well as provide live operational insight directly to the organisers of events.

Objectives of the Study

To achieve this target, the research and development process will be guided by the following objectives.

I. Create a user-friendly web application for any customer who wants to book an event ticket online whenever required.
II. Introduce an automated ticket generation option, which includes a distinct secure QR code per purchase.
III. Create a QR code scanning module for ticket verification at the entry.
IV. Implement a real-time attendance tracking system that makes the attendees count when you scan tickets.
V. Allow event organizers to view live attendance, identify suspicious activities, and interact with the ticketing data.
VI. Secure by using encryption and proper validation to prevent ticket duplication or fraud.
VII. Ticket Verification and Attendance Monitoring: For speed, accuracy, and reliability in evaluating the performance of the system.

1.4 Significance of the Study

The importance of this research lies in its ability to address long-standing challenges in event management while supporting technological advancement. Its main goal is to improve speed,

convenience, and security for attendees without reducing operational efficiency, financial accountability, or compliance for organizers. This project bridges the gap between online ticket sales and physical event access by introducing a web-based ticketing system that integrates QR codes and real-time attendance tracking. These features streamline the check-in process, reduce queues, and help ensure that events start and run on schedule.

The system's secure QR code validation helps prevent fraud and unauthorized entry, protecting both revenue and participant safety. Real-time attendance tracking also enhances crowd control, regulatory compliance, and emergency response, creating benefits for both organizers and attendees. Moreover, it fosters a stronger relationship between organizers and audiences, helping to build trust and encourage repeat attendance. Additionally, the system provides valuable data that can be used to analyze attendance patterns, optimize staffing and venue layouts, and inform marketing and operational decisions for future events.

1.5 Scope of the Study

The scope of the study is the design and development of a Web-Based Ticketing System with QR Code and Real-Time Attendance Tracking for event management. The project aims to develop a straightforward, dependable, and efficient platform that seamlessly integrates online ticket sales with physical event access.

The system will operate on any modern web browser, making it compatible with desktops, smartphones, and tablets. It features online ticket purchasing, encrypted QR code generation, instant validation at entry points, real-time attendance tracking, and an admin dashboard for monitoring ticket sales and attendance.

This study is limited to ticketing and attendance management and does not include features such as event scheduling, marketing, or feedback collection. Security is ensured through QR code encryption, database protection, and secure payment integration. The aim is to deliver a secure and user-friendly prototype that demonstrates how web-based ticketing and real-time tracking can enhance event management efficiency.

1.6 Limitations of the Study

This research successfully developed a functional web-based ticketing system with QR code integration and real-time attendance tracking; however, it is not without certain limitations. The

project was conducted within a fixed academic timeframe, which limited opportunities for extended testing, large-scale deployment, or integration of additional features such as third-party services, AI-driven analytics, or multilingual support.

Testing was carried out using controlled simulations with smartphones and basic QR scanners, rather than specialized event hardware. While security measures such as encrypted QR codes, secure payment gateways, and database protection were implemented, advanced cybersecurity testing methods, including penetration testing and blockchain-based verification, were beyond the project's scope.

The system's operation also depends on stable internet connectivity, as offline validation was not included in this version. Although the development tools used are well-suited for building a functional prototype, they do not yet support the scalability or customization required for enterprise-level deployment. Furthermore, user testing involved a limited number of participants, which restricted feedback on broader usability and adoption.

Despite these limitations, the system achieved its primary objectives and demonstrates strong potential for future expansion and integration into larger, more advanced event management platforms.

1.7 Definition of Terms

Please note that the terms below are defined for this work to provide clarity and consistency in this research.

I. Web-Based System: A software application that is accessed via a web browser over the internet, rather than being installed on a user's device. In this project, a web-based ticketing platform was developed to allow users to purchase tickets and services online.

II. Ticketing System: An application or automated process used to sell, distribute, and validate tickets for an event. This includes both the frontend (user interaction) and backend (administrative management).

III. QR Code (Quick Response Code): A two-dimensional barcode capable of storing text, URLs, or encrypted data. Every ticket in this system has unique identity information embedded in its QR code to ensure security at entry points.

IV. QR Code Verification: The process of scanning and verifying a QR code to confirm its validity by matching it with ticket records stored in the system database.

V. Real-Time Attendance Tracking: The process of recording each check-in as tickets are scanned, displaying the exact number of visitors present at an event, and providing organizers with live access to entry data.

VI. Encryption: The process of converting data into a code to prevent unauthorized access. Encryption in this project ensures that data encoded in QR codes cannot be easily replicated or altered.

VII. Authentication: The process of verifying that a user or ticket is genuine and authorized to gain access. In this system, authentication is achieved through the scanning of a QR code connected to the database.

VIII. Event Organizer (Admin): The person or group responsible for creating events in the system, managing ticket sales, validating attendance, and overseeing event security.

IX. Attendee (User): Any person who purchases a ticket for an event. Attendees use the web interface to buy tickets and receive a unique QR code for entry.

X. Dashboard: A status page that displays live statistics, ticket sales data, attendee counts, and visual reports such as charts and alerts.

XI. Database: An organized collection of electronic data that stores all event, ticket, and attendance records.

XII. Scalability: The ability of a system to handle increased workload efficiently by adding

resources such as hardware or server capacity.

XIII. UI (User Interface): The visual and interactive part of the system that allows users to perform actions such as purchasing tickets or viewing attendance information.

XIV. Payment Gateway: A service that securely processes online payments for ticket purchases. Examples include Flutterwave and Paystack.

XV. Prototype: An early working version of a system created for testing and validation before final implementation.

XVI. Cloud-Based Hosting: A hosting method in which system files and databases are stored on internet-accessible servers, ensuring reliability and global availability.

XVII. Fraud Prevention Mechanism: A security feature designed to prevent ticket fraud through encryption and the creation of unique QR codes.

XVIII. Access Control: The process of regulating entry to a system or location to ensure that only authorized users are granted access.

CHAPTER TWO
LITERATURE REVIEW

2.0 Introduction

This chapter reviews previous studies, theories, and related works that form the foundation of this project. It examines existing research on web-based ticketing systems, event management applications, and associated technologies, including online databases and real-time data synchronization. The review also discusses how various systems have attempted to address problems in ticket sales, authentication, and event monitoring.

By evaluating past approaches and identifying their limitations, this chapter provides a clearer understanding of how this project contributes to improving existing systems. References from books, journal articles, and online publications are included to support the review and show how the current system builds upon previous work.

2.1 Conceptual Review

The conceptual review focuses on the key ideas and technologies that define this study. These concepts help explain how the proposed system functions and why certain design decisions were made.

Event Management Systems:
Event management systems are digital platforms designed to organize, plan, and coordinate events, including conferences, concerts, and seminars. According to *Adeleke and Musa (2020)*, event management systems help reduce administrative workload by automating ticket sales, registration, and attendance monitoring. They are handy for significant events where manual methods become inefficient and prone to errors.

Online Ticketing Platforms:
Online ticketing has revolutionized the way people purchase and manage tickets. Instead of physical tickets, users can now book and validate entries through digital systems. *Okafor and Lin (2022)* explain that online ticketing increases convenience for both organizers and participants, offering instant access and digital verification through QR codes or barcodes. The growth of e-commerce and web technologies has made this system more secure and accessible across multiple devices.

Database Management in Web Applications:
Databases play a central role in any web application that stores user data. A well-structured database ensures integrity, scalability, and data security. *(Nwachukwu et al. 2021)* noted that the introduction of cloud-based databases, such as Firebase, MongoDB, and Supabase, has improved how developers manage user data in real-time. These systems offer built-in APIs that simplify backend development while maintaining data accuracy and availability.

Real-Time Data Synchronization:
Real-time synchronization allows data updates to be reflected immediately across all connected devices or clients. This feature is essential in ticketing systems where changes such as ticket validation or attendance updates must appear instantly.

User Authentication and Security:
Authentication protects systems from unauthorized access and data breaches. *(Kumar and Adeyemi 2021)* emphasized that modern authentication mechanisms, such as token-based access, password hashing, and two-factor verification, help secure user credentials and transactions in online platforms. In this project, Supabase's authentication service provides a robust way of managing user access through token-based security, preventing unauthorized operations.

2.2 Theoretical Framework

The theoretical foundation of this project is built on several principles of software engineering and system design that guided the development process from analysis to implementation. The System Development Life Cycle (SDLC) provided the structural base for the project, ensuring that each stage was carefully planned, executed, and reviewed before moving to the next. According to Ogunleye and Adebanjo (2020), SDLC offers a systematic approach that helps developers organize tasks, manage risks, and ensure that user requirements are met through a step-by-step development process. In this project, the model was utilized to transition smoothly from identifying problems in manual ticketing to designing and deploying a web-based solution that addressed those inefficiencies.

While the SDLC provided structure, the Agile development methodology gave the process flexibility. Agile promotes building software in small, iterative cycles, allowing for regular testing and feedback at every stage. Fowler (2019) explained that Agile enables developers to adapt quickly to changing requirements while maintaining quality through continuous testing. This approach proved effective for this system because it allowed modules—like user registration, event creation, and ticket validation—to be developed and tested independently before being combined. It also made it easier to refine the system based on real-time feedback from test users.

The project also drew from the Client–Server Architecture theory, which defines how communication occurs between a user (client) and a server that processes and stores information. (Mishra and Gupta 2021) noted that this model enhances scalability and performance in web-based systems. In the ticketing system, React.js handles the user interface as the client, while Node.js and Express.js serve as the server, processing requests and interacting with the

Supabase database. This architecture ensures quick responses and efficient data handling, even when multiple users are active simultaneously.

The design of the user interface was influenced by the Human–Computer Interaction (HCI) theory, which focuses on building systems that are intuitive and easy to use. Dix et al. (2020) stated that the usability of a system depends mainly on how naturally users can navigate and perform tasks without confusion. Guided by this theory, the ticketing system's interface was designed with simplicity in mind, featuring clear labels, consistent layouts, and responsive designs that work well across various devices. This made the platform accessible to users regardless of their technical background.

Finally, the Data Flow and Real-Time Communication model provided a theoretical foundation for handling live updates and synchronized data across the system. (Eze and Sharma 2019) observed that real-time synchronization ensures consistency between users and servers, especially in distributed web applications. This principle was applied through Supabase's live synchronization feature, allowing instant updates whenever a ticket was generated or validated. These real-time interactions made the system dynamic and reliable for both organizers and attendees.

2.3 Related Works

Over time, several researchers and developers have contributed to the development of web-based systems that simplify event management and ticketing processes. This section reviews those works, discusses their strengths and weaknesses, and highlights how this project builds on and improves upon them.

The shift from manual ticketing to digital platforms began due to the inefficiencies of paper-based systems. According to Bello and Yusuf (2018), manual ticketing systems were prone to errors, duplication, and fraudulent activities because records were not properly tracked. Their study on local sports event management in Lagos showed that event organizers lost significant

revenue due to counterfeit tickets. They concluded that automation through online systems could drastically improve accountability and efficiency.

Eze and Sharma (2019) developed an online ticket reservation platform for a university event using PHP and MySQL. Their work successfully replaced manual registration with an online form that generated tickets automatically after payment. However, the system lacked real-time synchronization — users had to refresh the page to view updates, and administrators struggled with delayed attendance reports. This limitation motivated the current project's use of Supabase, which supports live data synchronization without the need for manual refreshes, making ticket validation and attendance tracking instantaneous.

Nwachukwu et al. (2020) designed a conference registration and attendance system that allowed users to register and print tickets. Although their platform simplified registration, it did not automate ticket validation. Event staff still had to check printed tickets manually, causing delays during entry. This project improves on that by introducing automated ticket validation, where each ticket has a unique code that can be verified in real time through the database.

Okafor and Lin (2022) focused on developing an e-ticketing system that used barcodes to verify ticket authenticity. Their work demonstrated that barcode integration could reduce duplication and speed up the verification process. However, they used a local database, which limited accessibility for remote organizers. The system in this study uses a cloud-based database instead, making it accessible from anywhere and improving collaboration among event managers.

Akinyemi and Thomas (2021) developed a mobile ticketing application for transportation systems. Their design allowed users to book and pay for travel tickets using a mobile app. While the application worked efficiently on smartphones, it did not support desktop browsers, which restricted its accessibility. The system developed in this project bridges that gap by being web-based and responsive across both mobile and desktop devices, ensuring that users can access it conveniently regardless of their device.

Security has also been a recurring focus in online ticketing research. Bala and Adamu (2020) developed a secure ticketing system that implemented user authentication and data encryption. They found that user verification improved trust among participants, but their model only supported single-user sessions at a time, limiting scalability. This current system expands that functionality by supporting multiple concurrent users through secure authentication using Supabase, ensuring data protection while maintaining real-time access.

In a related study, Ibrahim and Chen (2021) built an event monitoring system using Firebase to handle attendance data. Their platform achieved real-time updates for small-scale events but encountered difficulties when scaled to larger groups. This project addresses that limitation by optimizing data flow through asynchronous operations in Node.js and using Supabase's scalability features to handle higher user traffic efficiently.

Usability has been highlighted as one of the major success factors in digital ticketing systems. Dix et al. (2020) explained that a good system must not only perform well technically but must also be simple enough for average users to operate without assistance. Amadi and Okoro (2019) found in their research that users tend to abandon complex registration systems that require too many steps or load slowly. Taking this into account, the ticketing system developed here was designed with simplicity, speed, and accessibility in mind. The user interface was kept clean, buttons were clearly labeled, and pages were optimized for faster loading even on weaker internet connections.

Adebayo and Musa (2021) worked on a conference management platform that generated reports after each event. While it allowed for efficient record storage, the system still required manual confirmation of attendance. In contrast, the system developed in this project automatically updates attendance once a ticket is validated, reducing human effort and ensuring real-time accuracy.

Internationally, Kim and Lee (2020) developed a concert ticketing system in South Korea that integrated QR code validation and mobile payment. Their design demonstrated the convenience of digital ticketing but was heavily dependent on specific payment providers. The current system, however, is designed to be flexible, allowing it to integrate different payment options and even work with free-entry events without requiring payment confirmation.

Adeola and Johnson (2022) examined the adoption of online ticketing systems in Nigeria's entertainment industry. They discovered that poor internet connectivity and lack of user awareness were significant barriers to adoption. Their study recommended developing lightweight systems that perform efficiently even under low bandwidth. This insight influenced the development of this project's frontend, which was optimized using React's rendering capabilities to minimize data load and improve responsiveness.

In summary, past studies have made significant contributions to the development of online ticketing and event management systems. However, many of these systems suffered from limitations, including a lack of real-time synchronization, weak scalability, poor user interfaces, and limited data accessibility. This project builds upon their foundations by integrating modern web technologies, such as React.js, Node.js, and Supabase, to create a system that is scalable, responsive, and capable of handling concurrent operations smoothly.

The resulting system not only solves the identified problems but also introduces real-time automation in areas where previous systems relied on manual processes. By combining security, usability, and accessibility, this project represents an improvement over existing works. It provides a strong foundation for future research and implementation in web-based event management and digital ticketing.

2.4 Review of Existing Systems

In the global event management industry, online ticketing systems have become a vital tool for organizing and managing access to events. Several platforms—both international and local—have been developed to automate ticket sales, attendance tracking, and payment processing. This section reviews some of the most popular existing systems, discussing their strengths, weaknesses, and how the system developed in this project improves on them.

Eventbrite is one of the most recognized platforms worldwide. It allows users to create events, sell tickets, and monitor attendance digitally. Bello and Yusuf (2018) noted that Eventbrite's success is primarily due to its ease of use and integration with payment gateways, such as PayPal and Stripe. Users can register within minutes and receive confirmation emails instantly. However, Eventbrite charges high service fees for every transaction, which discourages small event organizers in developing countries from using it. It also depends heavily on fast internet access, which can limit its effectiveness in areas with low connectivity. This project addresses that problem by developing a lightweight system that loads efficiently, even on weak networks, and does not require additional service fees.

Another globally recognized system is Ticketmaster, which dominates the large-scale entertainment and sports ticketing industry. Okafor and Lin (2022) described Ticketmaster as one of the most secure platforms available, offering digital ticket delivery and fraud prevention through the use of unique barcodes. However, its infrastructure is complex and primarily designed for significant events that require enterprise-level integration. Smaller event organizers often find it expensive and difficult to use. The system created in this project focuses instead on simplicity—providing the same core functions such as digital ticket generation and validation, but without the heavy costs and technical requirements that come with Ticketmaster.

A platform more tailored to African users is Tix.africa, a Nigerian-based ticketing and event management system. Akinyemi and Thomas (2021) explained that Tix.africa enables users to sell tickets online and receive payments in local currencies, making it more practical for African markets. It integrates with local payment gateways such as Paystack and Flutterwave, which is a significant advantage for event organizers within the region. However, Tix.africa still relies heavily on manual verification of tickets and does not support real-time updates on attendance. The system developed in this study builds upon that by utilizing Supabase for real-time database synchronization, ensuring that every ticket validated by an organizer is updated automatically across all connected systems.

Etix is another digital ticketing platform used internationally, mainly for entertainment events and concerts. Bala and Adamu (2020) highlighted that Etix uses barcode-based validation, which minimizes ticket duplication and fraud. However, the system requires users to download a mobile application before they can use it, making it less convenient for those who prefer browser-based access. The web-based system developed in this project eliminates this limitation,

as it can be accessed directly through any browser on mobile or desktop without installation. This improves usability, especially for occasional event attendees.

Eventcube is a cloud-based event management platform that focuses on customization. Ibrahim and Chen (2021) stated that Eventcube allows event organizers to design ticket pages, monitor sales, and analyze marketing data. While this offers flexibility for advanced users, it often overwhelms smaller organizers who only need basic ticketing features. The system in this project simplifies that process, focusing on three essential functions—ticket generation, validation, and attendance monitoring—while maintaining real-time updates and user-friendly navigation.

Locally, Nigeria has witnessed a growing adoption of online ticketing systems, though their development is still limited compared to global solutions. Adeola and Johnson (2022) found that most Nigerian ticketing systems rely on foreign services, such as Eventbrite and Paystack, for hosting or processing payments, which makes them less customizable and more expensive. Their study recommended building locally hosted ticketing systems that are affordable and functional without relying on international services. This project directly responds to that recommendation by developing an independent, fully functional platform that can operate efficiently even without third-party dependencies.

When comparing these systems, several patterns become clear. Eventbrite and Ticketmaster are powerful but costly and complex. Tix.africa is regionally relevant but lacks real-time data processing. Etix provides good security, but limits users to accessing the app only. Eventcube offers advanced features but is not beginner-friendly. The system developed in this project combines the strengths of these platforms—secure validation, user convenience, and data automation into a more straightforward, faster, and more affordable solution.

In summary, this review demonstrates that while global platforms have achieved remarkable success, they often overlook the needs of smaller, independent event organizers, particularly in developing regions. This project's system bridges that gap by offering a fully web-based, real-time, and secure ticketing platform that can be managed easily from any device. By integrating technologies like React.js, Node.js, and Supabase, it brings the efficiency of global systems to a localized context, ensuring accessibility, affordability, and reliability for modern event management.

2.5 Summary of the Review

This chapter reviewed existing theories, concepts, and related studies that form the foundation of this project. From the discussion, it is clear that event management and ticketing systems have undergone significant evolution, transitioning from manual, paper-based methods to efficient web-based solutions. However, most existing systems still face challenges related to real-time synchronization, accessibility, and cost-effectiveness.

The conceptual review outlined essential concepts, including event management, online ticketing, user authentication, and real-time data synchronization. These concepts are central to the development of modern digital platforms that aim to replace manual processes with automated ones. The theoretical framework provided a structured foundation, drawing on principles such as the System Development Life Cycle, Agile methodology, and Client–Server Architecture, which guided the design and implementation of this project.

The empirical review revealed that several researchers, including Bello and Yusuf (2018), Eze and Sharma (2019), and Akinyemi and Thomas (2021), have made significant contributions to the development of online ticketing. Their work established that automation improves efficiency and transparency, but also highlighted key weaknesses, such as the absence of live data updates, limited scalability, and poor system design. These gaps motivated this study.

The review of existing systems compared popular platforms like Eventbrite, Ticketmaster, Tix.africa, and Etix. While these systems are successful in their own ways, they also come with drawbacks such as high operational costs, dependency on strong internet connections, and a lack of local customization. The system developed in this project addresses these weaknesses by integrating modern web technologies, such as React.js, Node.js, and Supabase, to create a lightweight, responsive, and cost-efficient ticketing solution.

CHAPTER THREE
METHODOLOGY AND SYSTEMS ANALYSIS

3.1 Introduction

This chapter presents the methodology adopted for designing and developing the proposed system, as well as an analysis of existing ticketing practices. It explains the approach followed in building the system, describes the shortcomings of the current methods in use, and highlights

how the new design addresses those gaps. The discussion covers the research methodology chosen, an overview of the existing system, its problems, the justification for a new system, and the requirements needed to bring the proposed solution into reality.

3.2 System Analysis

3.2.1 Analysis of Existing System

In most institutions and event environments, ticketing and attendance management still rely on outdated methods. The most common approach is the paper-based system, where tickets are physically printed and sold in advance or at the venue. Validation is carried out manually, usually by collecting ticket stubs or by visual inspection at the point of entry. Although this system is simple and has been used for decades, it quickly becomes inadequate when the number of attendees increases. Manual processes cannot keep pace with the demand for speed and accuracy that large-scale events require. Long queues form at the gates, attendees experience delays, and staff are often overwhelmed by the workload.

Some organizations have tried to improve on this by introducing semi-digital systems. These platforms allow customers to purchase tickets online, usually through a web portal. Tickets are then sent as confirmation emails or PDF documents that can be printed or displayed on mobile devices. While this makes ticket acquisition easier, the process still breaks down at the stage of verification. Event staff typically check tickets manually against printed lists or by visually scanning confirmation codes. This reintroduces the same inefficiency and makes the system vulnerable to human error. Worse still, the tickets generated by these platforms are often static, which means they can easily be duplicated and reused by unauthorized persons. The lack of integration between ticket purchase, verification, and attendance monitoring leaves organizers with little to no real-time visibility of what is happening inside the venue.

3.2.2 Disadvantages of the Existing System

The disadvantages of these existing systems are numerous and significant. First, inefficiency is a major challenge. Attendees are forced to wait in long lines, events are delayed, and staff spend too much time on repetitive tasks. This creates frustration for participants and lowers the overall quality of the event experience.

Second, security is weak. In paper-based systems, counterfeit tickets can be produced with minimal effort. Even semi-digital systems are prone to fraud, since static tickets or confirmation

emails can be copied and reused. This leads to revenue losses for organizers, reduces trust in the ticketing process, and can even result in overcrowding, which poses safety hazards.

Third, there is no real-time attendance tracking. Organizers often know how many tickets were sold, but they do not know how many attendees are physically inside the venue at any given time. This makes it difficult to enforce capacity limits, manage emergencies, or comply with government regulations. For instance, in situations where strict occupancy limits must be observed, relying on ticket sales alone does not guarantee compliance.

Finally, existing systems are often fragmented. Ticket sales are handled by one system, attendance is checked manually, and reports are generated after the event through rough estimations or manual counts. This separation deprives organizers of valuable insights that could be used for better planning, resource allocation, and marketing strategies in the future.

3.2.3 Analysis of Proposed System

The proposed system addresses these limitations by introducing a web-based platform that integrates all aspects of ticketing and attendance into one seamless process. Users will be able to purchase tickets online through a secure and user-friendly interface. Once a purchase is completed, the system automatically generates a unique QR code for the ticket. This QR code is encrypted and linked directly to the central database, ensuring that no two tickets can be duplicated.

At the event venue, staff will scan the QR code using a smartphone or scanner connected to the system. Validation happens instantly, and the ticket status is updated in real time. This means that fraudulent or already-used tickets are immediately flagged, preventing unauthorized entry. Each successful scan updates the attendance database automatically, giving organizers an accurate and live count of how many people are inside the venue at any given time.

In addition to ticket validation, the proposed system includes an administrative dashboard where organizers can monitor ticket sales, check attendance figures, and analyze entry patterns as the event unfolds. This provides not only real-time oversight but also valuable post-event data that can be used to improve future planning. By integrating ticketing, validation, and analytics, the system ensures that organizers have a complete and reliable tool for managing events effectively.

3.2.4 Advantages of the Proposed System

The proposed system offers several clear advantages. First, it significantly improves efficiency by reducing delays at entry points. QR code scanning takes only seconds, which means queues move faster and events can start on time.

Second, it strengthens security. Since each QR code is unique and encrypted, duplication becomes practically impossible. Fraudulent tickets can be detected instantly, protecting both revenue and attendee safety.

Third, the system provides real-time monitoring. Organizers can see how many people have entered the venue at any point in time, track attendance patterns, and respond quickly to unusual situations such as sudden surges at specific gates. This live data also helps with enforcing capacity limits and complying with safety regulations.

Finally, the proposed system integrates ticketing and attendance management into one platform. Instead of relying on fragmented tools, organizers have a single solution that covers ticket sales, verification, and analytics. This not only improves the current event but also builds a data-driven foundation for better management of future events.

Figure 3.1: Shows the System Architecture of the Proposed Web-Based Ticketing System.

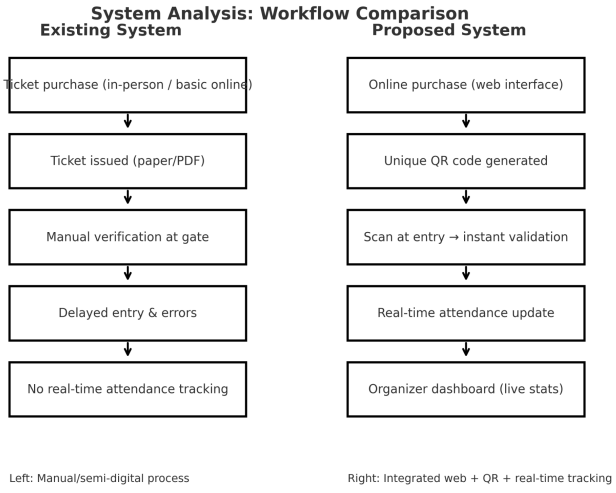

Figure 3.1

3.3 System Design

3.3.1 Design Methodology

The design methodology describes the structured approach adopted in developing the proposed web-based ticketing system with QR code integration and real-time attendance tracking. In

system development, methodology plays a crucial role because it provides direction, reduces risk, and ensures that the final solution aligns with the needs of users. For this project, the Prototyping Model of software development was selected. This choice was made after considering the nature of the system to be built, the time available for development, and the importance of incorporating user feedback during the process. The Prototyping Model emphasizes building a preliminary version of the system (a prototype) that demonstrates core functionalities, such as ticket purchase, QR code generation, and attendance tracking. This prototype is presented to stakeholders—both organizers and potential users—so they can interact with it and provide feedback. The feedback is then used to refine the system, correcting flaws, improving usability, and adding missing features. This process is repeated until the system matures into a reliable and effective product.

One of the main reasons this model was chosen is its iterative and user-centered nature. Unlike traditional models such as the Waterfall approach, which requires all specifications to be completed upfront and leaves little room for change, the Prototyping Model allows for adjustments at every stage of development. Event management systems, by their nature, must be highly user-friendly and flexible, since attendees may have varying levels of digital literacy and organizers may have different operational requirements. Using a rigid method might produce a technically functional system but one that fails to satisfy real-world needs. Prototyping avoids this risk by keeping users actively involved throughout the development cycle.

The methodology unfolds in several stages. First, requirements gathering is conducted to understand what the system must achieve. This involves identifying the core needs of both organizers (such as real-time monitoring, fraud prevention, and reporting tools) and attendees (such as convenience, fast validation, and secure payments). Next, an initial prototype is created that demonstrates these essential features in their simplest form. The prototype is then subjected to user evaluation, where test users interact with the system by performing tasks such as purchasing tickets and scanning QR codes. Their experiences, difficulties, and suggestions are carefully noted. Following evaluation, the system goes through refinement cycles. Each cycle adds improvements: enhancing the security of QR codes, optimizing the speed of scans, simplifying the user interface, or expanding the reporting functions on the organizer's dashboard. These refinements continue until the prototype evolves into a near-final system that is both functional and user-friendly. The final stage is implementation and testing, where the refined system is deployed in a controlled environment to ensure stability, security, and reliability.

The advantages of this methodology are numerous. First, it reduces the likelihood of building a system that does not meet user expectations. Second, it allows for early detection of design flaws, saving both time and resources. Third, it keeps the system adaptable, meaning that new insights or technological changes can be incorporated even after the development process has begun. For a project like this—where efficiency, security, and real-time performance are

central—Prototyping Model ensures that the final solution is not only technically sound but also aligned with the practical needs of end users.

Figure 3.2: Shows the Prototyping Methodology Cycle used in the system's development process.

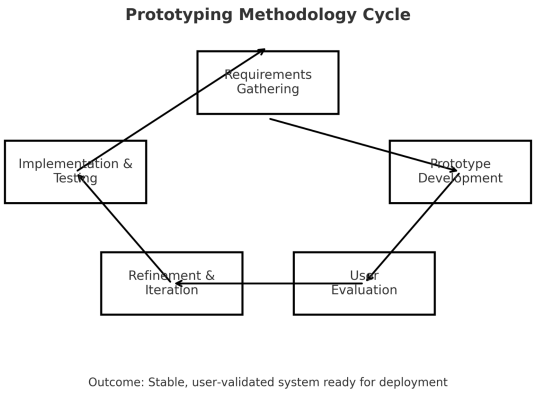

Figure 3.2

3.3.2 System Architecture

The architecture of the proposed system follows a three-tier structure designed to ensure scalability, security, and reliability. The first tier is the presentation layer, which consists of the user interface that attendees and administrators interact with. For attendees, this includes the ticket purchase page, payment gateway, and QR code delivery system. For administrators, the interface is a dashboard that displays ticket sales, live attendance data, and analytics.

The second tier is the application layer, which contains the business logic of the system. Here, the operations of ticket generation, QR code encryption, validation, and real-time data updates are handled. This layer ensures that each QR code is unique and linked to the correct ticket record in the database. It also manages communication between the user interface and the backend database.

The third tier is the data layer, consisting of a secure database where all records are stored. This includes user information, payment details, ticket records, and attendance logs. The database must be protected with strong encryption and access control to prevent unauthorized manipulation of data.

By separating the system into these three layers, the design ensures modularity and makes it easier to maintain and upgrade the system. For example, the interface can be redesigned without

altering the database structure, and additional features such as reporting modules can be added without disrupting the ticketing and validation processes.

Figure 3.3: Shows the Three-Tier System Architecture of the ticketing system.

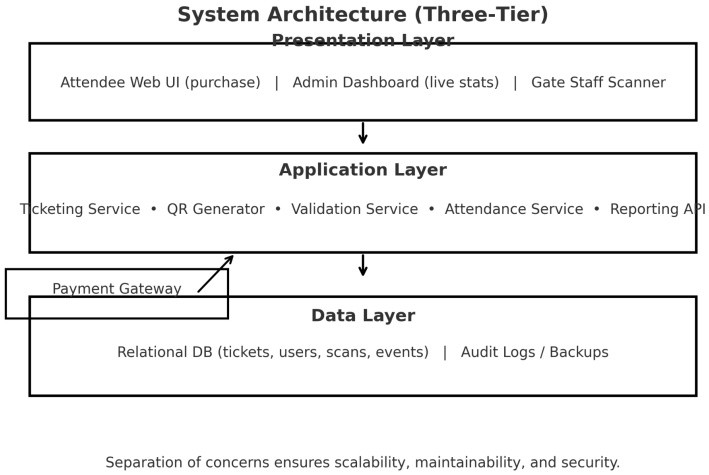

Figure 3.3

3.4 System Requirements

The requirements of the proposed system are divided into functional, non-functional, and technical requirements. These define what the system will do, the quality standards it must meet, and the hardware/software resources needed for its operation.

Functional requirements include: the ability for users to purchase tickets online; automatic generation of unique, encrypted QR codes; validation of tickets through scanning; automatic updating of attendance records in real time; and a dashboard for organizers to monitor ticket sales and attendance patterns.

Non-functional requirements cover aspects such as security, usability, and scalability. The system must process payments securely using trusted gateways, ensure QR codes are tamper-proof, and encrypt sensitive data. It must also be user-friendly, with an interface that is simple enough for non-technical users. In addition, the system should be scalable so it can handle increasing numbers of users and events as adoption grows. Reliability is also key—the platform should remain stable under normal usage conditions and recover quickly from disruptions.

Hardware and software requirements include basic smartphones or QR code scanners for entry validation, a web server to host the application, and a relational database to store records. On the client side, any device with internet access and a modern web browser can access the system. On

the development side, standard web technologies such as HTML, CSS, JavaScript, and frameworks for backend development will be used, along with database management tools like MySQL or PostgreSQL.

Figure 3.4: Shows the Use-Case Diagram of the web-based ticketing system.

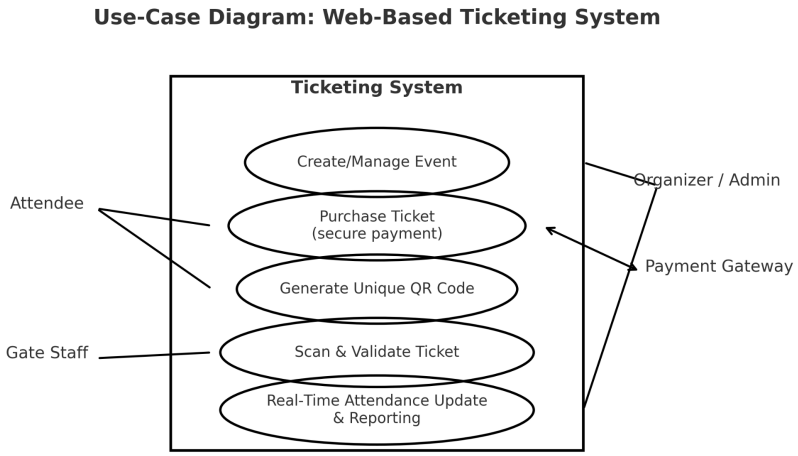

Figure 3.4

3.5 Chapter Summary

This chapter has presented the methodology, analysis, and design of the proposed system. The Prototyping Model was selected as the design methodology because of its iterative and user-centered nature, allowing for constant refinement until the system meets real-world needs. The analysis of the existing system revealed inefficiencies, vulnerability to fraud, and lack of real-time attendance tracking. These shortcomings justified the need for a new system that integrates ticketing, QR code verification, and live monitoring.

The design of the proposed system was then outlined, highlighting its three-tier architecture consisting of the presentation layer, application layer, and database layer. The system requirements were defined in terms of functionality, quality attributes, and hardware/software needs. Diagrams were also recommended to illustrate workflows, system architecture, and use-case interactions, ensuring clarity in how the system will operate.

Together, these discussions provide a solid foundation for the next chapter, which focuses on system implementation and the translation of the design into practical models and working prototypes.

CHAPTER FOUR
SYSTEMS DESIGN AND IMPLEMENTATION

4.0 Introduction

This chapter describes how the online ticketing system was designed and implemented in accordance with the presented design. It details the process of turning each design feature into a working/interactive system that facilitates real-time ticket generation, validation, and attendance tracking. The steps included setting up, adding a server/client component as context, and connecting to the database. The data was tested several times to verify its accuracy.

It was implemented using React. js for the frontend and Node. js with Express. js for the backend. Both were picked for their adaptability, efficiency, and ease of work with asynchronous processes a key system requirement. We chose to use Supabase as our primary database due to its stability, scalability, and ease of integration with JavaScript-based applications. It also features pre-installed authentication and an API, making it easy to handle event data and user records securely.

Worked locally on development, using Visual Studio Code as the main development and debugging tool. We first deployed the system in a local environment with a Node server, and then we hosted it online to enable external assessment. At every step of the development process - from front-end user interface coding to connecting backend API endpoints - there was an emphasis on building a nimble, responsive, and intuitive app experience that integrates with your use of our web application as though it’s the same single product, painlessly and with no need for manual intervention during interaction with event ticketing or attendance.

4.1 Overview of System Implementation

A software development phase where the blueprints and concepts were concreted into a working, reliable application that regular people could use. Our goal was to develop a fully functional version of the ticketing platform that offers a seamless user experience and performs optimally in any modern browser, providing a glitch-free, data-secure, and undistorted web application. This was the starting point for creating a development environment in which the system's architecture could be implemented and tested incrementally.

It was developed as a component-based system, following the Agile development life cycle model, which involves completing tasks in a step-by-step manner. This resulted in an iterative approach, as modules were tested and refined before being integrated into the whole application. We began by building the UI in React. js for everything the user did: navigation, data entry, and creating dynamic new elements. The interface was stripped down and clean, making it easy to use on any screen size, whether desktop or mobile.

The back-end component was created with Node. js and Express. js, was responsible for server-side logic such as API endpoints, user authentication, and database interaction. This part of the system provided us with the guarantee that everything's action on the frontend (be it creating a ticket or validating one) was being safely and correctly processed. The database was connected to via Supabase, which gave me a PostgreSQL backend with real-time synchronization. It contained user, ticket, and participation records for informed decisions and efficient operations.

We tested and hosted the system on Visual Studio Code while developing it, utilizing its integrated terminal and the live server extension. This became more comfortable to debug and get instant feedback as you type the code. Git was additionally employed to version and monitor the code development process. The system was deployed on-Âline after the core features were developed, and its responsiveness and reliability have been tested under the real-Âlife constraints of a network environment.

Every module event creation, from ticket validation, was tested separately before entering the full-system testing and assembly phase. The focus was on data fidelity, instantaneous speed, and fluid client/server round-tripping. The result was a fully operational web-based platform for ticketing, capable of generating unique digital tickets, validating them safely against stored records, and tracking attendees both during and after the event over time.

4.2 System Requirements

Before the development of the ticketing system began, it was essential to identify both the hardware and software requirements that would ensure a smooth implementation and testing process. The system was designed to run efficiently on standard personal computers, with lightweight components that do not require high-end configurations. These requirements were chosen to make sure that the application can perform effectively on regular hardware setups and remain easily deployable on local or online servers.

For hardware, the system was developed and tested using a computer equipped with an Intel Core i5 processor, 8GB of RAM, and a 256GB SSD, all running on a stable broadband internet connection. This configuration provided sufficient power to handle the simultaneous operation of the React frontend, Node.js backend, and Supabase database connection without lag or performance degradation. Although lower specifications, such as an Intel Core i3 processor and 4GB RAM, can still support the project, a mid-range setup was preferred for better speed during development, code compilation, and testing.

On the software side, several tools and technologies were employed throughout the development process. The Windows 11 operating system served as the primary environment, with Visual Studio Code (VS Code) functioning as the integrated development environment (IDE). VS Code was selected for its lightweight structure, support for JavaScript frameworks, and built-in Git integration, which made version control and collaboration easier. The frontend of the system was built using React.js, HTML5, CSS3, and JavaScript (ES6). React.js was particularly useful for creating reusable components and managing the dynamic rendering of data across different pages without requiring the entire site to reload.

For the backend, Node.js and Express.js were used to handle server logic and route management. This combination provided a simple yet powerful structure for building RESTful APIs that the frontend could communicate with. The APIs were responsible for handling ticket creation, verification, and user management. The Supabase platform was integrated as the database service because it supports real-time data synchronization and offers a built-in authentication system. Its PostgreSQL foundation provided strong data integrity, while its easy integration with JavaScript made it a perfect fit for this project.

Git was used for version control, helping to keep track of all code changes and ensuring that previous versions of the system could be restored when needed. The project repository was hosted on GitHub for backup and easy access from any location. For testing and debugging, the Google Chrome Developer Tools were used extensively to monitor network requests, debug frontend issues, and analyze the performance of each component.

Finally, the project was hosted and tested online through Vercel, which provided an efficient platform for deploying both the frontend and backend applications. The deployment on Vercel ensured that the system could be accessed from anywhere, allowing multiple users to interact with it simultaneously for evaluation and feedback. This helped assess the system's performance under real-world conditions, ensuring it could effectively handle concurrent users.

4.3 Programming Languages and Tools Used

The ticketing system was developed using a blend of the latest languages, frameworks, and tools to make it fast, secure, and responsive. All the tools were carefully selected based on their functionality, ease of integration, and compatibility with the system architecture. As the project is led and built as a pure web-based platform, everything needed to all work together harmoniously on a full-stack JavaScript environment from scratch.

The application was built using React. js, a JavaScript library for creating dynamic user interfaces. React was selected because it facilitated the creation of a component-based architecture, which allowed us to factor every part of the interface (namely: home page, ticket form, and validation page) as an isolated and reusable module. This made the application maintainable and updatable in the future. Moreover, React's Virtual DOM made it faster, with fewer unnecessary page reloads, ensuring seamless user interaction. We utilized HTML5 and CSS3 to organize and design the interface. HTML was to lay the base structure for each page, and CSS was used to produce a crisp, contemporary style—the layout required to translate well across multiple breakpoints.

For the interactivity and client-side logic, We went with JavaScript (ES6). It enabled actions such as form validation, real-time updates to ticket information, and displaying feedback messages without requiring the page to be reloaded. The use of JavaScript can also facilitate exchange between the client-side and server through asynchronous API requests, thus providing a responsive and fast system operation.

The back-end was built with Node. js and Express. js. Node. js was the runtime environment that enabled JavaScript to run on the server-side, and Express. js served as the templates that simplified the server routes and endpoints. All the main functions of the system were managed using this setup, including ticket generation and validation, as well as communication with the database. There was a route for each purpose in Expres, say, one concerned with creating tickets, another verifying QR codes, or adding to an attendance tracker. Express's modular setup made the codebase manageable and debuggable.

The database was hosted in Supabase as a storage service. Supabase is built on PostgreSQL and provides a straightforward approach to managing real-time data. It is also used to store user details, events, and tickets. One of the most significant advantages of using Supabase is its out-of-the-box authentication and RESTful API support, which enables secure frontend-to-backend communication with no additional middleware required for database queries. It also added auto-syncing of the form, so changes to the database are immediately shown on the client.

Apart from these principal technologies, various development tools were utilized to develop the system efficiently and in a more organized manner. Visual Studio Code was the primary code editor due to its clean interface, robust debugging tools, and compatibility with JavaScript frameworks. Version control was implemented using Git to track the project files' history during development, and code backups and collaboration were maintained on GitHub, as it could host a remote repository. These tools were invaluable in ensuring that progress was adequately documented and the codebase could be easily reverted if anything went wrong.

For deployment and testing, Vercel was used to host the frontend and backend services. Vercel had an easy GitHub integration and updated whenever I made a new commit to my repo. It's also

able to deploy quickly and can be accessed worldwide efficiently through this web application. The project was also tested locally using Node's local server during development, before final deployment. Browser developer tools, such as Google Chrome Developer Tools, were also used for fixing frontend issues, monitoring API responses, and analyzing network performance.

4.4 Implementation Procedures

The implementation of the ticketing system followed a structured and gradual process that ensured each component of the project was built correctly, tested, and integrated. This phase was where the system transitioned from design concepts into an actual working application. Development began by setting up the environment, then moved into frontend and backend development, database connection, integration, and deployment. Each stage was completed with continuous testing to confirm that all parts of the system were functioning as expected before moving to the next step.

The first step in the implementation was the environment setup. The development environment was prepared on a local machine using Visual Studio Code (VS Code) as the primary Integrated Development Environment (IDE). Node.js and npm (Node Package Manager) were installed to manage dependencies for both frontend and backend development. Git was also initialized for version control, allowing us to track changes throughout the entire process. Once the initial setup was complete, a new React application was created using the create-react-app command, which generated the structure for the frontend components.

The frontend implementation followed, focusing on building a simple, clean, and easy-to-use interface. The system's main pages included the homepage, event details page, payment form, ticket generation page, and validation page. Each of these was built as a reusable React component, making the code modular and easy to maintain. Navigation between pages was managed using React Router, which enabled users to move through the site without needing to reload pages. CSS was used extensively to style the pages and ensure the layout was responsive across various devices. At the same time, React hooks, such as useState and useEffect, were utilized to manage application state and fetch data from the backend in real-time.

After completing the user interface, the backend was implemented using Node.js and Express.js. The backend handled all logic and operations that required server-side processing. Several routes were created to manage different functions, for example, a route for saving tickets, another for validating them, and one for checking event attendance. Each route received requests from the frontend, processed them, and sent appropriate responses. The backend also managed authentication, ensuring that only authorized users could create or validate tickets. Error handling was added to make sure invalid inputs or duplicate requests did not crash the server.

Once the backend structure was complete, the next step was to integrate the database. Supabase was connected to the Node.js server using environment variables stored securely in a .env file.

This ensured sensitive data, such as database credentials, remained private. The database was designed to hold tables for users, tickets, and attendance logs. Each time a new ticket was created, the data was sent from the frontend to the backend and then stored in the Supabase database. The real-time features of Supabase enabled new entries or updates to be reflected immediately across the system, which was particularly useful for live attendance tracking.

The integration phase involved linking the frontend to the backend through API calls. Using the Axios library, the frontend could send requests to the server and retrieve responses asynchronously. This allowed actions such as ticket creation, validation, and data fetching to happen without the page reloading. Once the connection between the client and server was stable, several tests were carried out to ensure data consistency between the user interface and the database. For example, after a ticket was generated, the system immediately confirmed its creation by displaying the ticket code and storing it in the database simultaneously.

Following integration, the system was deployed and tested. Initially, it was hosted locally for performance testing and debugging. Once stable, the application was deployed online using Vercel, which provided automatic builds and updates directly from the GitHub repository. This allowed real users to test the application from different devices and networks. During testing, We focused on system speed, ticket validation accuracy, and overall user experience. Minor issues like CORS errors and slow API responses were identified and corrected by updating configuration files and optimizing database queries.

Throughout implementation, testing was continuous. Every new feature added was immediately tested before proceeding to the next. This approach helped identify and fix bugs early, ensuring the final system was stable and reliable. The project's modular structure also made it easier to isolate errors and update individual components without affecting the entire application.

4.5 System Module Description

The ticketing system was developed as a collection of interconnected modules, each performing a specific function to ensure smooth operation. Every module communicates with the others through secure API calls, allowing real-time interaction between users, the server, and the database. This modular design makes the system flexible, easy to maintain, and capable of handling both small and large events efficiently. The following sections describe each of these

modules and their functionalities, with placeholders indicating where screenshots should be inserted later.

The homepage module serves as the entry point of the system. It provides users with an overview of available events and allows easy navigation to ticket purchasing or management sections. The homepage was designed to be clean and visually appealing, displaying essential options like "*Get Started " and "Browse Events"*. This ensures that both first-time users and returning organizers can easily find what they need without unnecessary navigation. The page also includes a header, navigation bar, and footer for easy movement throughout the platform.

Figure 4.1: Shows the Homepage Interface of the Web-Based Ticketing System.

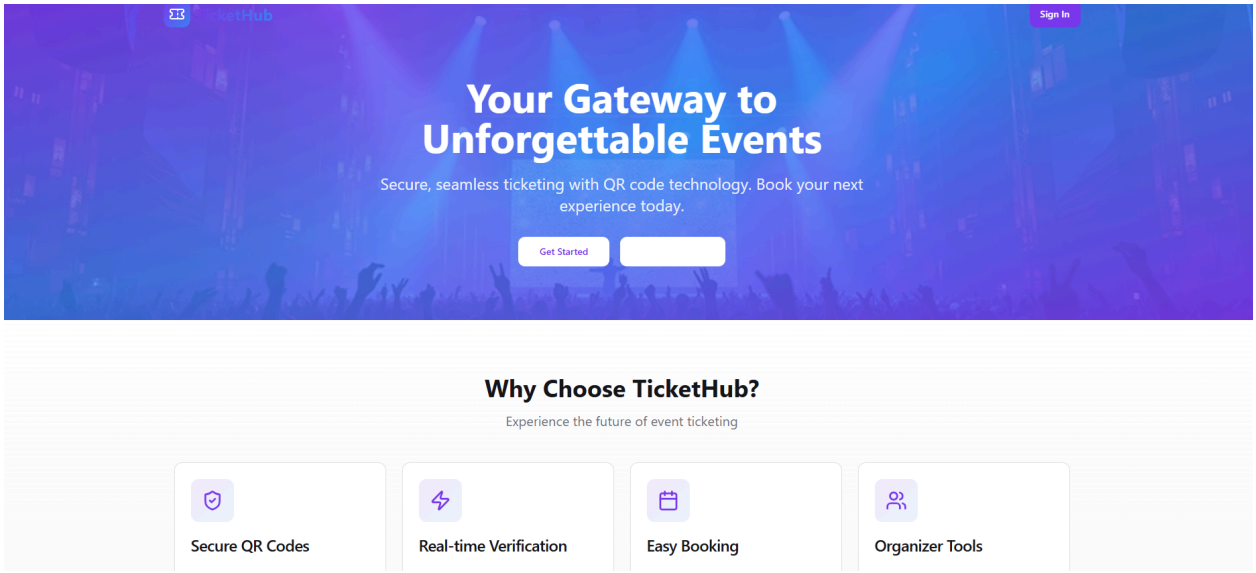

Figure 4.1

The registration and login module handles user authentication, ensuring that only verified users can access specific features. New users can create an account using their email and password, which are securely stored in the Supabase database using built-in authentication services. Registered users can log in to manage events, generate tickets, or view attendance data. Form validation was implemented to prevent invalid or incomplete inputs.

Figure 4.2: Shows the User Registration and Login Interface.

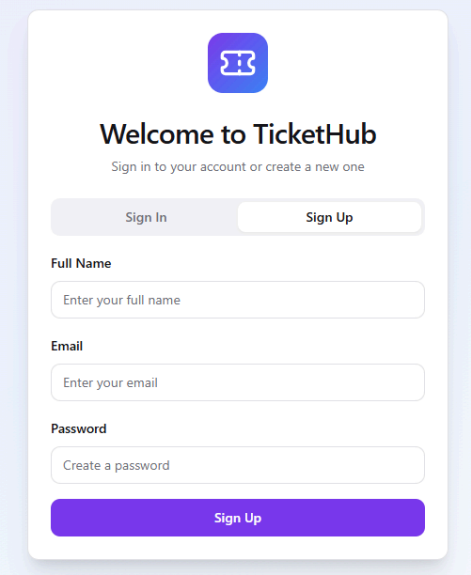

Figure 4.2

The event creation module allows organizers to add new events to the system. Through a simple form interface, organizers can input event details, including name, date, venue, and ticket price. Once submitted, the information is saved to the database and displayed on the homepage for potential attendees. This module also includes edit and delete functions that allow organizers to modify or remove events. It is primarily accessible only to admin users or verified organizers.

Figure 4.3: Shows the Event Creation Page used by Organizers.

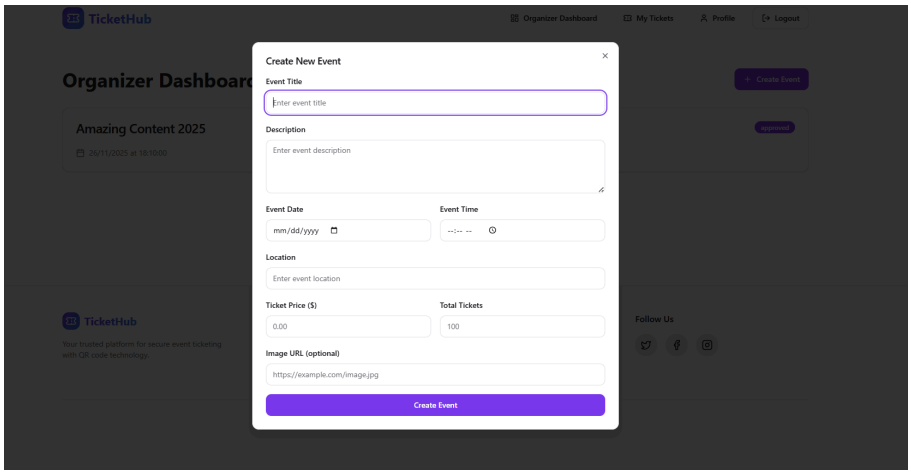

Figure 4.3

The ticket generation module is responsible for creating unique tickets once payment is confirmed. Each ticket is assigned a unique code and a corresponding QR code, both stored in the Supabase database. These codes serve as the ticket identifiers and are required during event entry validation. The user interface displays the generated ticket, allowing users to save or take a screenshot of it for later use.

Figure 4.4: Shows the Ticket Generation Page with QR Code Display.

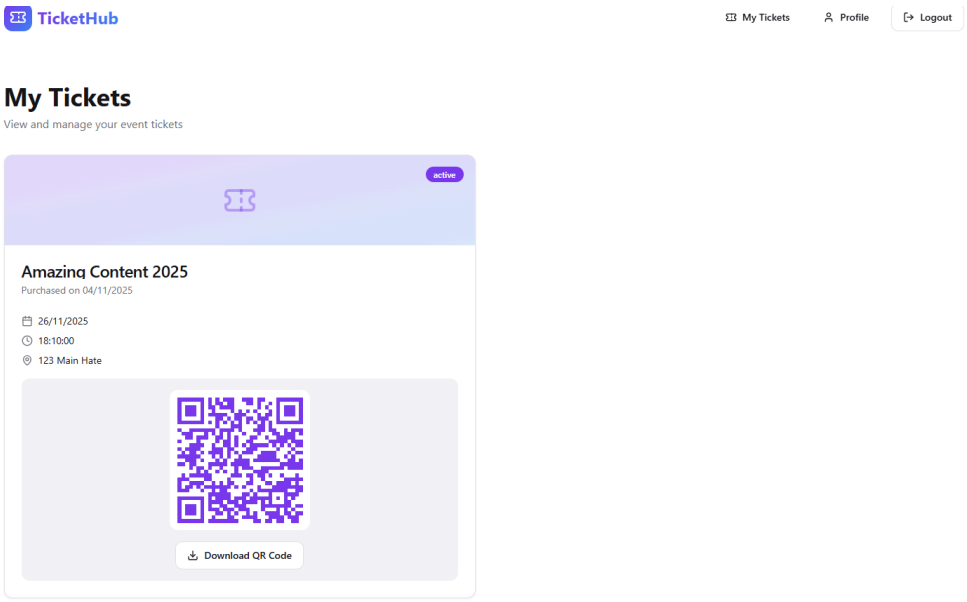

Figure 4.4

The ticket validation module is designed for use at the event venue. It verifies ticket authenticity by cross-checking the ticket code or QR code against records in the Supabase database. Once validated, the ticket's status is updated to *used*, preventing duplicate entries. If a ticket is invalid or has already been used, an alert message notifies the user immediately. This process happens in real time, ensuring accurate attendance tracking.

Figure 4.5: Shows the Ticket Validation Module used at the Entry Point.

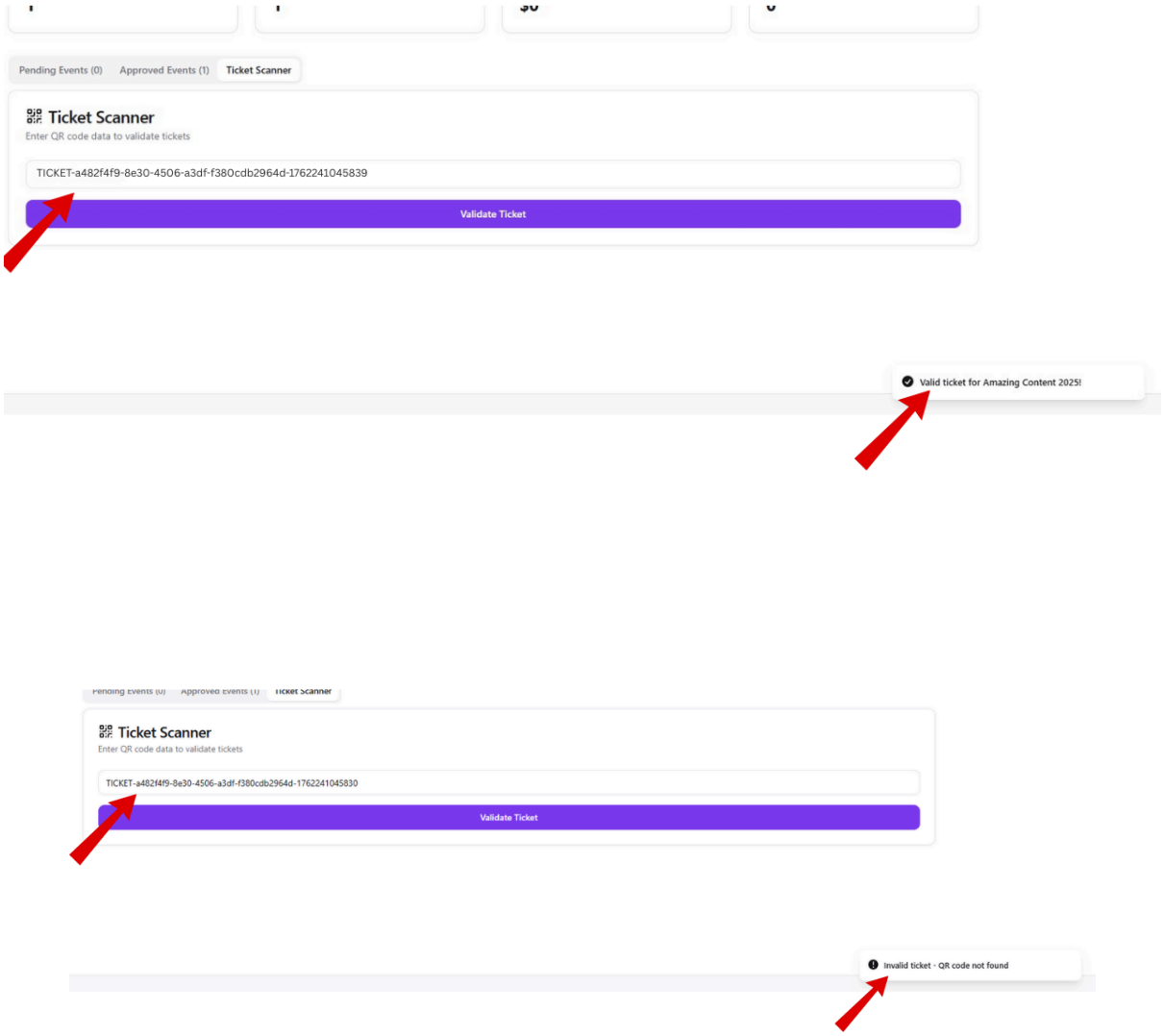

Figure 4.5

The attendance dashboard module provides real-time monitoring of event attendance. It is accessible only to administrators and organizers. The dashboard displays a table containing the names, emails, and ticket codes of attendees who have successfully checked in. This feature helps organizers keep track of crowd size and manage venue capacity more effectively.

Figure 4.6: Shows the Attendance Dashboard displaying real-time attendee data.

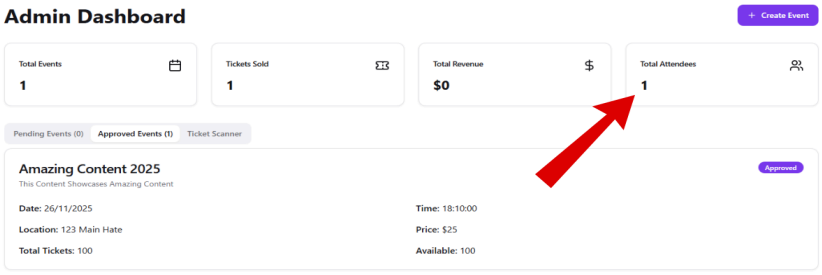

Figure 4.6

The database module handles data storage, retrieval, and synchronization across the system. It was implemented using Supabase, which provides a PostgreSQL backend with RESTful API endpoints. The database stores user accounts, event information, ticket records, and attendance logs. Supabase's real-time functionality ensures that updates such as new registrations or ticket scans are instantly reflected across the system. Data integrity is maintained through unique identifiers and relational table structures that prevent duplication or data loss.

Figure 4.7: Shows the Database Structure on Supabase.

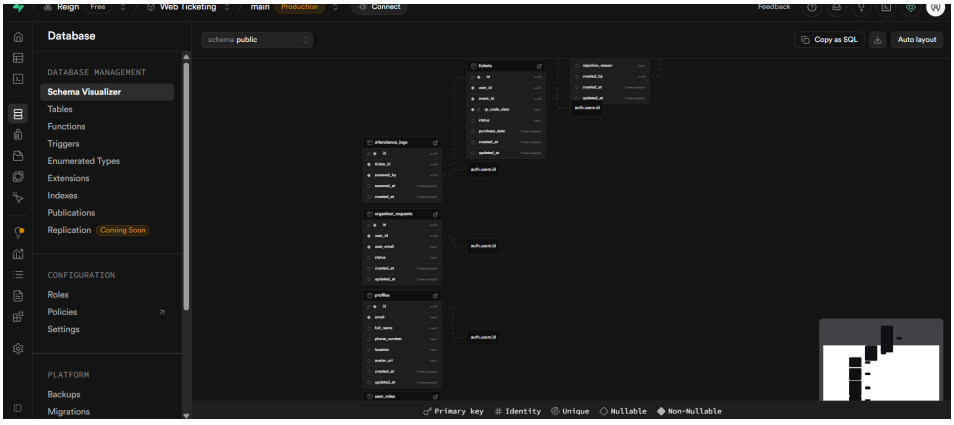

Figure 4.7

The admin control module allows authorized personnel to manage users, view event reports, and monitor system activities. Administrators can view all registered users, verify organizers, and delete inactive accounts. They can also export attendance reports for recordkeeping. The module provides a simple dashboard interface that consolidates all management tools in one place for easy access.

Figure 4.8: Shows the Admin Control Panel Interface.

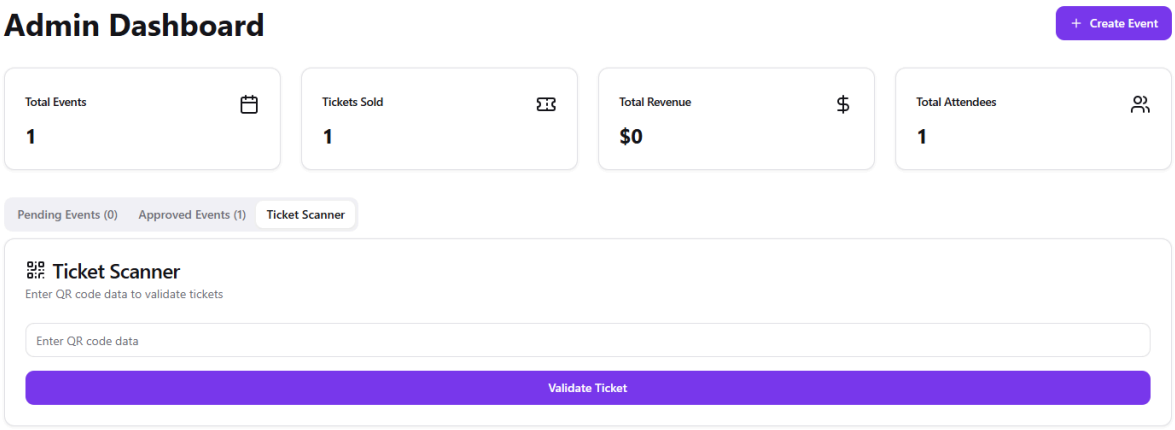

Figure 4.8

Together, these modules form a cohesive system that covers all aspects of digital event management from ticket creation to real-time validation. Each module operates independently but interacts with the others to provide a seamless user experience. The modular approach not only improves maintainability but also allows new features, such as analytics or notifications, to be added in future updates without disrupting the existing structure.

4.6 Testing Strategies

Testing was an essential stage in the development of the ticketing system. It was carried out to ensure that all system components worked correctly, met the stated objectives, and interacted smoothly without errors. The goal was to confirm that the system was reliable, efficient, and user-friendly before final deployment. Testing began during development and continued after integration to identify and correct issues early. A combination of manual and automated testing approaches was used to validate both functionality and performance.

The first phase involved unit testing, which focused on verifying the functionality of individual components and functions to confirm they worked as intended. Each React component on the frontend, such as the ticket form, event list, and validation page, was tested separately to ensure that it rendered correctly and handled user input without breaking. On the backend, unit tests were conducted on specific API endpoints, including ticket creation, validation, and database communication. These tests helped detect early issues in logic, such as invalid responses or data not saving correctly in Supabase.

After unit testing, integration testing was performed to verify how different parts of the system worked together. The aim was to confirm that the frontend, backend, and database communicated seamlessly. During this process, I checked that data submitted through the user interface was sent correctly to the Node.js server, processed, and stored in Supabase. The integration tests also ensured that when a user generated a ticket, it appeared immediately in the database and could be retrieved for validation. This phase helped confirm that there were no conflicts or data mismatches between the different system layers.

Next was system testing, where the entire application was tested as a single unit. This phase simulated real-world usage by running the application in a browser and interacting with it as a typical user would. I tested all major features, account creation, event posting, payment simulation, ticket generation, validation, and attendance tracking to ensure everything flowed correctly from start to finish. System testing also helped evaluate how the application performed when multiple users interacted with it simultaneously.

Once system testing was completed, user acceptance testing (UAT) was conducted. A small group of users, including classmates and volunteers, was invited to use the application and give feedback on its design, ease of navigation, and functionality. Their input helped identify areas that could be improved, such as the clarity of validation messages and the color contrast on some pages. Minor adjustments were made based on this feedback to improve usability and overall user satisfaction.

Performance testing was also conducted to assess how the system handled stress and concurrent requests. I simulated multiple users generating and validating tickets simultaneously to observe how the system performed under load. Supabase’s database connection remained stable, and the

response time for API calls remained under one second, even when multiple users interacted simultaneously. This confirmed that the system could handle real-time operations without lag or data delays.

Finally, security and validation testing were performed to ensure data protection and prevent unauthorized access. The authentication module was tested to verify that only registered users could log in or access event management tools. Input validation was also checked to prevent SQL injection or invalid data entry. All sensitive data, including API keys and database credentials, was stored in environment variables to avoid exposure.

Throughout testing, any issues discovered were immediately logged and fixed. Some common problems encountered included CORS errors during API calls, inconsistent data rendering between frontend and backend, and incorrect status updates for used tickets. These issues were resolved by updating backend configurations, refining database queries, and improving state management within React.

4.7 Test Case Design and Execution

To ensure that the system performed as expected, test cases were carefully designed to evaluate every primary function and process of the ticketing system. Each test case focused on verifying a specific aspect of the system, from user authentication and event creation to ticket generation and validation. The goal was to confirm that inputs produced the correct outputs and that the system handled both valid and invalid data gracefully without crashing or making errors.

The testing process was divided into three levels: functional, integration, and validation. For each feature, expected results were defined before testing began, and actual results were compared afterward to determine if the test passed or failed. When discrepancies were found, adjustments were made to the code, and the test was repeated until the desired outcome was achieved.

Figure 4.1: Shows the Test Case Design and Execution Table.

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
TC01	User Registration	Fill the signup form with a valid name, email, and password, then click “Register.”	User account is created and saved in Supabase.	Account created successfully, and confirmation message displayed.	Passed
TC02	Login Authentication	Enter your correct email and password on the login form and submit.	The user is redirected to the dashboard if the credentials are valid.	The system redirected the user correctly; the session was created successfully.	Passed
TC03	Invalid Login Attempt	Enter an incorrect email or password.	The system should display an “Invalid credentials” message.	Error message displayed properly; access denied.	Passed
TC04	Event Creation	Input valid event details and click “Add Event.”	The event should be added to the database and appear on the homepage.	Event added and displayed on the homepage.	Passed
TC05	Ticket Generation	Complete the payment form and submit.	The system should generate a unique ticket with a QR code.	Ticket generated successfully with QR code.	Passed
TC06	Duplicate Ticket Generation	Try generating a ticket for the same email twice.	The system should prevent duplicate ticket creation.	The system returned an error message and blocked the duplicate.	Passed

TC07	Ticket Validation	Scan or enter a valid ticket code.	The ticket should be marked as “used” and grant access.	System validated the ticket correctly and updated the status in the database.	Passed
TC08	Invalid Ticket Validation	Scan or enter a fake or expired ticket code.	The system should reject invalid tickets and display an error message.	Error message displayed correctly.	Passed
TC09	Attendance Tracking	Validate multiple tickets to record attendance.	The attendance list updates in real time on the dashboard.	The dashboard is updated correctly for every validated ticket.	Passed
TC10	Admin Control Functions	Access the admin panel and view users/events.	Admin should be able to view all users, events, and reports.	The admin accessed all control panels without issue.	Passed

Testing covered both normal operations and edge cases. For example, during login testing, invalid credentials and empty input fields were intentionally tested to ensure that form validation worked correctly. During the ticket validation phase, attempts were made to scan expired or previously used tickets to confirm that the system properly rejected them. These detailed tests helped ensure data integrity and prevented system abuse.

During execution, a few minor issues were encountered. For instance, there was a delay when updating the attendance list on slower networks, and a CORS-related error appeared during initial frontend-backend communication. These were fixed by optimizing Supabase’s data-fetching logic and updating the Node.js server’s configuration to handle cross-origin requests.

By the end of the test case execution, all critical functionalities were confirmed to be working correctly. The system responded accurately to user inputs, maintained secure communication between modules, and updated data in real time. This stage validated that the web-based ticketing system was stable, reliable, and ready for deployment in real-world scenarios.

4.8 Results and Evaluation

The implementation and testing of the ticketing system produced positive results that met the project's main objectives. The system successfully automated the entire process of event registration, ticket generation, and validation without the need for manual input. Each ticket created was assigned a unique code stored in the Supabase database and could be validated instantly, ensuring data accuracy and preventing duplication.

The system proved to be reliable and responsive during testing. All major modules, user registration, login, event creation, and ticket validation, performed as expected with minimal latency. Real-time updates from Supabase ensured that changes, such as ticket validation and attendance records, were immediately reflected across the system.

In terms of usability, the interface was clear and straightforward, allowing users to navigate the platform easily without prior training. Feedback from test users indicated that the layout was simple and functional on both desktop and mobile devices. Security was also verified through Supabase's built-in authentication, which restricted access to authorized users only.

Overall, the system performed efficiently, accurately processed user actions, and maintained stable performance during multiple concurrent operations.

4.9 Challenges Encountered and Solutions

During the development and implementation of the ticketing system, a few challenges were encountered, most of which were technical issues that required careful troubleshooting. One of the early challenges was establishing a stable connection between the backend and the Supabase database. Initially, some API requests failed to retrieve or update data correctly due to the incorrect configuration of environment variables. This issue was resolved by properly setting up the Supabase credentials in a secure .env file and updating the backend routes to handle asynchronous requests efficiently.

Another challenge involved Cross-Origin Resource Sharing (CORS) errors during the integration of the frontend and backend. The browser initially blocked API requests because the frontend and backend were running on different ports during local development. This issue was resolved

by enabling CORS in the Express.js server configuration, which allowed for safe communication between the two parts of the application.

There was also a minor issue with state management in React, where ticket data sometimes failed to update immediately after validation was complete. Asynchronous delays in API responses were the cause. The problem was resolved by using React's useEffect hook and proper dependency tracking to ensure that state updates occurred right after each successful request.

Another obstacle came during deployment, where some build files failed to load correctly due to mismatched environment settings between the local and hosted environments. Adjusting the build configuration on Vercel and re-linking the Supabase project resolved this issue.

Lastly, some performance delays were noticed when multiple users interacted simultaneously. This was optimized by reducing unnecessary database queries and implementing server-side caching for frequently accessed data.

In the end, all challenges were successfully addressed, resulting in a stable, responsive, and secure ticketing system that met the intended goals and performed efficiently under testing conditions.

CHAPTER FIVE
SUMMARY, CONCLUSION, AND RECOMMENDATION

5.1 Summary of Findings

The design of the web-based ticketing system met every actual target previously mentioned in the project description. The system is designed to address issues associated with manual ticketing for events, including ticket duplication and inadequate attendance documentation. Through proper system design, installation, and testing, these issues were successfully resolved.

The resulting system enables users to sign up, log in, purchase tickets, and have their tickets automatically generated and stored in the Supabase database. Every ticket is issued with a unique code so that it can be validated live, meaning only genuine attendees are granted access at events. This automation has made the management of significant events much more efficient and less prone to human error.

The integration of React. js for the frontend and Node. js with Express. js on the backend has guaranteed a system that requires performance and reliability from its infrastructure. Supabase provided secure and scalable data management, while Vercel simplified the deployment of your website and made it accessible on the web. During testing, it proved to be reliable, while also being very easy to use and responsive, with real-time updates and proper synchronization of records across all modules.

Altogether, the project demonstrated that it is possible to leverage state-of-the-art web technologies to build a potentially efficient and user-friendly digital ticketing platform. The goals of the system, which included automatically generating tickets, tracking attendees in events, and creating a secure database system, were all achieved.

5.2 Conclusion

The successful outcome of the project has proven that an online-based ticketing system can significantly enhance event management, monitoring, and control. The system eliminates inefficiencies inherent in manual ticketing methods and automates processes such as registration, ticket generation, validation, and attendance monitoring. With functional programming and modern technologies like React. js, Node. js, and Supabase Automated, a secure, lightning-fast platform for events, both small and large-scale.

From conception to deployment, the system maintained its focus, aiming to provide accurate data, quick results, and real-time updates during ticketing. The tests validated the system's smooth behavior, even when multiple people use it, and the test users' feedback showed that the interface was very intuitive. The project also underscored how open-source tools and cloud-based databases can form robust, low-cost digital solutions to everyday challenges.

The ticketing system is a practical and scalable approach for handling today's events. It fulfills the objective of convenience, reducing the workload for organizers and ensuring easy data accuracy without sacrificing the end-user experience.

5.3 Recommendations

Based on the successful implementation and evaluation of the ticketing system, several recommendations can be made to enhance its functionality and ensure long-term sustainability.

First, the system can be improved by integrating a secure online payment gateway, such as Paystack or Flutterwave, which allows users to complete transactions directly within the platform. This would make the process faster and eliminate the need for manual payment confirmation.

Secondly, adding email or SMS notifications would improve user experience by automatically sending ticket details, confirmation messages, and event reminders after successful registration or payment. This feature would also help organizers communicate important updates efficiently.

Another functional enhancement would be the development of a mobile application version of the system. A dedicated app would make it easier for users to purchase, store, and validate tickets on the go, especially during significant public events where mobile accessibility is essential.

Additionally, incorporating data analytics and reporting tools would enable event organizers to generate insights such as attendance trends, ticket sales performance, and demographic breakdowns. This information could help improve decision-making for future events.

Finally, regular system maintenance and updates are recommended to ensure security and stability, especially as new web technologies and threats emerge. Implementing these improvements will make the system more robust, versatile, and scalable for broader use.

REFERENCES

Adebayo, M., & Musa, S. (2021). Design of a conference management system for digital attendance reporting. *Journal of Computing and Information Science* (Abuja).

Adeleke, J., & Musa, R. (2020). Event management systems and their impact on modern event planning. *West African Journal of Applied Computing* (Ibadan).

Adeola, T., & Johnson, F. (2022). Adoption of online ticketing systems in Nigeria's entertainment industry. *National Journal of Information Systems* (Lagos).

Akinyemi, K., & Thomas, L. (2021). Development of a mobile ticketing application for transportation systems. *Journal of Innovative Computing Studies* (Port Harcourt).

Amadi, F., & Okoro, G. (2019). Usability challenges in web-based registration systems. *International Journal of Information Technology Research* (Enugu).

Bala, A., & Adamu, H. (2020). A secure e-ticketing system using encryption and user authentication. *Nigerian Journal of Computer Applications* (Kaduna).

Bello, A., & Yusuf, M. (2018). Evaluation of manual ticketing methods and the need for automation. *Journal of Management and Technology* (Zaria).

Dix, A., Finlay, J., Abowd, G., & Beale, R. (2020). *Human–computer interaction* (4th ed.). London, UK: Pearson Education.

Eze, P., & Sharma, V. (2019). Development of an online ticket reservation system using PHP and MySQL. *International Journal of Web Engineering* (Owerri).

Fowler, M. (2019). *The principles of agile development and iterative system design.* New York, NY: Addison-Wesley.

Ibrahim, K., & Chen, Y. (2021). Real-time event monitoring systems using Firebase technology. *Journal of Cloud Computing and Web Engineering* (Singapore).

Kim, H., & Lee, J. (2020). QR code-based e-ticketing systems for concert venues. *Asian Journal of Digital Innovation* (Seoul).

Kumar, P., & Adeyemi, S. (2021). Modern authentication mechanisms in web applications. *Journal of Cybersecurity and Digital Systems* (Lagos).

Mishra, R., & Gupta, N. (2021). Client–server architecture and web communication models. *International Journal of Computer Networks* (New Delhi).

Nwachukwu, I., Okorie, C., & Linus, P. (2020). Web-based conference registration and attendance system. *Journal of Modern Computing Technologies* (Nsukka).

Ogunleye, S., & Adebanjo, T. (2020). Software development life cycle (SDLC) models in system design. *Nigerian Journal of Software Engineering* (Ibadan).

Okafor, E., & Lin, C. (2022). Design and implementation of an e-ticketing system using barcodes. *International Journal of Digital Systems and Technology* (Lagos).