

**TICKET EMPORIUM: A TICKET SELLING SYSTEM**

**PETER MACHARIA MUNGAI**

**220520**

Project Submitted for partial fulfillment of the requirements for the award of Diploma in Full Stack Software development to the department of web development at the Institute of Software Technologies

**©2023**

# 

# DECLARATION AND APPROVAL

## **Declaration**

I hereby declare that this project documentation is a result of my original work and efforts, and to the best of my knowledge, it has not been submitted elsewhere for any other assessment or award.

**Student’s Name: PETER MACHARIA MUNGAI**

**Registration Number: 220520**

**Signature: .....................................................**

## **Approval**

This project documentation has been submitted with my approval as the assigned supervisor.

**Assigned supervisor:** **MR. SAMUEL KIRAGU**

**Signature: .....................................................**

ABSTRACT

The Ticket Emporium project aims to create a user-friendly ticket selling website that caters to both event organizers and ticket buyers. This documentation presents a comprehensive overview of the project, starting with an introduction to the industry and existing challenges. The project's objectives, scope, and justification are outlined, emphasizing the need for a more efficient and accessible ticket selling platform. The development methodology and key software methodologies are discussed, followed by an analysis of similar existing systems and their gaps. The evaluative report also presents a critical examination of the project's pros and cons. Additionally, the system analysis and requirement modeling, system design, and implementation are addressed, including a focus on the conceptual and physical models. Lastly, the limitations encountered during the project's execution are highlighted, along with recommendations for future improvements. The documentation is supported by references and an appendix containing the user manual and code samples.

TABLE OF CONTENTS

[DECLARATION AND APPROVAL ii](#_Toc148433705)

[**Declaration** ii](#_Toc148433706)

[**Approval** ii](#_Toc148433707)

[ABSTRACT iii](#_Toc148433708)

[TABLE OF CONTENTS iv](#_Toc148433709)

[LIST OF TABLES vii](#_Toc148433710)

[LIST OF FIGURES/ILLUSTRATIONS viii](#_Toc148433711)

[CHAPTER 1: Introduction and Project Overview 1](#_Toc148433712)

[**1.1 Introduction:** 1](#_Toc148433713)

[**1.2 Background of the Study:** 1](#_Toc148433714)

[**1.3 Problem Statement(s):** 2](#_Toc148433715)

[**1.4 Objectives:** 2](#_Toc148433716)

[**1.5 Scope of the Study:** 2](#_Toc148433717)

[**1.6 Justification:** 3](#_Toc148433718)

[**1.7 Budget and Resources:** 3](#_Toc148433719)

[**1.8 Project Risk and Mitigation:** 3](#_Toc148433720)

[**1.8.1 Project Risks** 4](#_Toc148433721)

[**1.8.2 Mitigation Strategies** 4](#_Toc148433722)

[**1.8.3 Conclusion** 5](#_Toc148433723)

[**1.9 Project Schedule:** 5](#_Toc148433724)

[CHAPTER 2: Evaluative Analysis and Comparative Study 7](#_Toc148433725)

[**2.1 Evaluative Report** 7](#_Toc148433726)

[**2.2 Use Case Diagram** 7](#_Toc148433727)

[**2.3 Abstract Project or Existing Similar Projects** 8](#_Toc148433728)

[**2.4 Analysis of Existing Similar Projects** 8](#_Toc148433729)

[**2.5 Pros and Cons of Ticket Emporium** 10](#_Toc148433730)

[**Pros:** 10](#_Toc148433731)

[**Cons:** 10](#_Toc148433732)

[CHAPTER 3: Project Methodology and Development Process 11](#_Toc148433733)

[**3.1 Project Methodology** 11](#_Toc148433734)

[**Planning Phase:** 12](#_Toc148433735)

[**Design Phase:** 12](#_Toc148433736)

[**Implementation Phase:** 12](#_Toc148433737)

[**Testing Phase:** 12](#_Toc148433738)

[**Deployment Phase:** 12](#_Toc148433739)

[**Maintenance Phase:** 12](#_Toc148433740)

[CHAPTER 4: System Analysis and Requirement Modeling 13](#_Toc148433741)

[**4.1 Description of Current Systems and Problems** 13](#_Toc148433742)

[**4.2 Requirement Modeling for Each System Module** 13](#_Toc148433743)

[**4.2.1 User Management Module** 13](#_Toc148433744)

[**4.2.2 Event Management Module** 15](#_Toc148433745)

[**4.2.3 Ticket Listing Module** 16](#_Toc148433746)

[**4.2.4 Event Listing Module** 17](#_Toc148433747)

[**4.2.5 Ticket Purchase Module** 18](#_Toc148433748)

[**4.2.6 Admin Management Module** 19](#_Toc148433749)

[CHAPTER 5: System Design and Database Architecture 23](#_Toc148433750)

[**5.1 Conceptual Model** 23](#_Toc148433751)

[**5.2 Physical Model** 24](#_Toc148433752)

[CHAPTER 6: System Implementation 27](#_Toc148433753)

[**6.1 Tools and Technologies** 27](#_Toc148433754)

[**Programming Languages:** 27](#_Toc148433755)

[**Frameworks and Libraries:** 27](#_Toc148433756)

[**Database Management System:** 27](#_Toc148433757)

[**Integrated Development Environment (IDE):** 27](#_Toc148433758)

[**Version Control:** 27](#_Toc148433759)

[**6.2 Sample Input/Output Screenshots** 27](#_Toc148433760)

[**1. User Registration Form:** 28](#_Toc148433761)

[**2. Seller Dashboard:** 28](#_Toc148433762)

[CHAPTER 7: Limitations and Recommendations 30](#_Toc148433763)

[**7.1 Limitations** 30](#_Toc148433764)

[**7.2 Recommendations** 30](#_Toc148433765)

[**7.3 Conclusion.** 31](#_Toc148433766)

[REFERENCES 32](#_Toc148433767)

[APPENDIX 33](#_Toc148433768)

[**User Manual for Ticket Emporium - Online Ticket Selling Platform (for ticket buyers)** 33](#_Toc148433769)

[**Code Sample: User Authentication API Endpoints** 36](#_Toc148433770)

# LIST OF TABLES

[Table 1 Table showing the project budget 3](#_Toc141434396)

# LIST OF FIGURES/ILLUSTRATIONS

[Figure 1 Gantt chart showing the different stages of Ticket Emporium’s development and the corresponding timeline for each stage 6](#_Toc147910133)

[Figure 2 Use Case Diagram illustrating interactions between the actors and the system 8](#_Toc147910134)

[Figure 3 Waterfall Model of System Development 11](#_Toc147910135)

[Figure 4 Flowchart illustrating the user registration and authentication process 15](#_Toc147910136)

[Figure 5 Flowchart illustrating the event creation process by the event organiser 16](#_Toc147910137)

[Figure 6 Flowchart illustrating the ticket purchase process 19](#_Toc147910138)

[Figure 7 Entity-Relationship Diagram (ERD) representing the conceptual model of the Ticket Emporium system. 23](#_Toc147910139)

[Figure 8 Screenshot of the user registration form 28](#_Toc147910140)

[Figure 9 Screenshot of the seller dashboard 28](#_Toc147910019)

# CHAPTER 1: Introduction and Project Overview

## **1.1 Introduction:**

This document serves as an introduction to the Ticket Emporium project, outlining the existing challenges in the ticket selling industry and the need for a comprehensive solution. The Ticket Emporium project aims to address the inefficiencies and limitations of the current ticket selling process. The project is centered on the development of an online ticket selling platform that is easy to use, reliable, and secure.

With the Ticket Emporium website, customers will no longer need to physically visit ticket outlets to purchase tickets, thereby saving time and enhancing convenience. It will also help organizers streamline their ticketing process and reach a wider audience. With the ability to purchase tickets online, event organizers can attract customers from anywhere in the world, as opposed to relying solely on physical ticket outlets or phone sales. This increases the potential for ticket sales and revenue.

## **1.2 Background of the Study:**

Ticket sales have always been a critical aspect of event management. The conventional methods of selling tickets, such as physical ticket outlets and phone sales, are now seen as outdated and inefficient. Therefore, the need for a more convenient, accessible, and user-friendly ticket-selling platform has become increasingly important.

To address this need, I have developed an online ticket selling website called Ticket Emporium through which, customers can purchase tickets from the comfort of their homes, without the need to visit physical ticket outlets. This online platform is convenient, fast, and secure, offering a range of payment options that customers can choose from.

This website also provides a user-friendly interface for event organizers and ticket sellers, making it easy to list events and sell tickets. With the website's robust back-end system, event organizers can keep track of ticket sales, monitor event attendance, and generate detailed reports on ticket sales.

This project aims to develop an efficient and effective online ticket-selling platform that can compete with traditional methods of selling tickets. The website's design and features aim to address the limitations of traditional ticket-selling methods while providing an intuitive and straightforward user experience.

## **1.3 Problem Statement(s):**

The current method of ticket sales has several limitations and inefficiencies, including:

* Limited accessibility - customers are limited to purchasing tickets at physical ticket outlets, which can be inconvenient for those who are not located close to these outlets.
* Long wait times - customers may have to wait in long lines to purchase tickets, which can be time-consuming and frustrating.
* Lack of convenience - the current method of ticket sales does not provide customers with the ability to purchase tickets online, which is becoming increasingly popular and convenient.

## **1.4 Objectives:**

* To investigate the current method of ticket sales used by the event management company.
* To develop an online ticket selling website that will provide customers with a convenient and efficient way to purchase tickets.
* To analyze the effectiveness of the online ticket selling website in terms of customer satisfaction and sales revenue.

## **1.5 Scope of the Study:**

The scope of the Ticket Emporium project is to develop an online ticket selling website that will provide a platform for customers worldwide to purchase event tickets with ease. The website will feature a user-friendly interface and a secure payment gateway for customers to make transactions safely. The project will not include the development of a physical ticket outlet system. This decision was made based on the project's scope, which focuses on developing an online ticket selling website. The geographical scope of the project is global, allowing customers from any part of the world to purchase tickets.

## **1.6 Justification:**

The Ticket Emporium project presents a solution to the challenge of the outdated and inefficient methods of ticket sales used in the event industry. By creating an online ticket selling platform, the project offers a timely and innovative solution that will provide customers with a more convenient and efficient way to purchase tickets. The advantages of this project are numerous, including improved customer satisfaction and increased sales revenue for event organizers. The project is also in line with current trends towards increased digitization and online accessibility.

## **1.7 Budget and Resources:**

To build the Ticket Emporium website, certain hardware and software resources will be required. The hardware resources include a computer and other necessary equipment, while the software resources include a web development platform and a secure payment gateway. In addition, funds will be allocated for website development and other miscellaneous expenses that may arise during the project. The proposed budget for these resources is;

Table Table showing the project budget

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Items | Specifications | Quantity | Unit Price (KSH) | Total Cost (KSH) |
| Third-Party Software/Libraries | Integration and licensing fees | 1 | 500 | 500 |
| TOTAL |  |  |  | 500 |

## **1.8 Project Risk and Mitigation:**

The success of any project depends on identifying potential risks and implementing effective mitigation strategies to address them. This section outlines the key project risks for the Ticket Emporium online ticket selling platform and proposes mitigation measures to ensure its smooth development and implementation.

### **1.8.1 Project Risks**

The following are the key project risks for Ticket Emporium:

***Technical risks:*** These risks include inadequate scalability and integration challenges. If the system is not scalable, it will not be able to handle increased traffic and user demands. If there are integration challenges, the system will not be able to seamlessly exchange data with payment gateways, email services, and other third-party systems.

**Schedule risks:** These risks include delays in development and resource constraints. If the project is delayed, it will not be completed on time or within budget. If there are resource constraints, the project may not be able to be completed with the desired features or quality.

***Budget risks***: These risks include cost overruns. If the project costs more than expected, it may not be completed or may not be completed with the desired features or quality.

**Security risks**: These risks include data breaches and vulnerabilities in third-party libraries. If there is a data breach, sensitive user data could be compromised. If there are vulnerabilities in third-party libraries, these vulnerabilities could be exploited by attackers.

### **1.8.2 Mitigation Strategies**

The following mitigation strategies have been proposed to address the key project risks for Ticket Emporium:

**Technical risks:** Thorough scalability testing will be conducted during development to ensure the system can handle increased traffic and user demands. Cloud-based hosting will be implemented to accommodate future growth. Comprehensive integration testing will be performed with payment gateways, email services, and other third-party systems to verify seamless data exchange and functionality.

**Schedule risks:** A detailed project schedule with clear milestones and deadlines will be established. Regular progress monitoring and frequent status meetings will be conducted to address any potential delays promptly. Resources will be allocated effectively to ensure the availability of skilled developers, designers, and testers to meet project demands. Plans will be made for contingencies in case of unexpected resource shortages.

**Budget risks:** A detailed budget plan with accurate cost estimates for development, infrastructure, and other project components will be created. Strict budget controls will be implemented and expenditures will be regularly reviewed to prevent overruns.

**Security risks:** Robust security measures will be implemented, including encryption, firewalls, and regular security audits. This will help to protect sensitive user data and prevent unauthorized access. A thorough security assessment of all third-party libraries used in the project will be conducted. These libraries will be regularly updated and patched to address any identified vulnerabilities.

### **1.8.3 Conclusion**

By proactively identifying potential risks and implementing effective mitigation measures, the Ticket Emporium project can navigate challenges successfully and achieve its objectives. Risk management is an ongoing process that will be closely integrated into every aspect of the project's development and implementation to ensure its ultimate success.

## **1.9 Project Schedule:**

The Ticket Emporium project will follow the Waterfall methodology for software development. The project is expected to take approximately 4-5 months to complete, depending on the complexity of the website and the availability of resources. The Waterfall methodology provides a structured approach that emphasizes planning and sequential execution, with each stage being completed before moving on to the next. This approach ensures that each stage is completed thoroughly before moving on to the next, reducing the risk of errors and ensuring that the project stays on schedule. The following Gantt chart represents the different stages of Ticket Emporium’s development and the corresponding timeline for each stage.

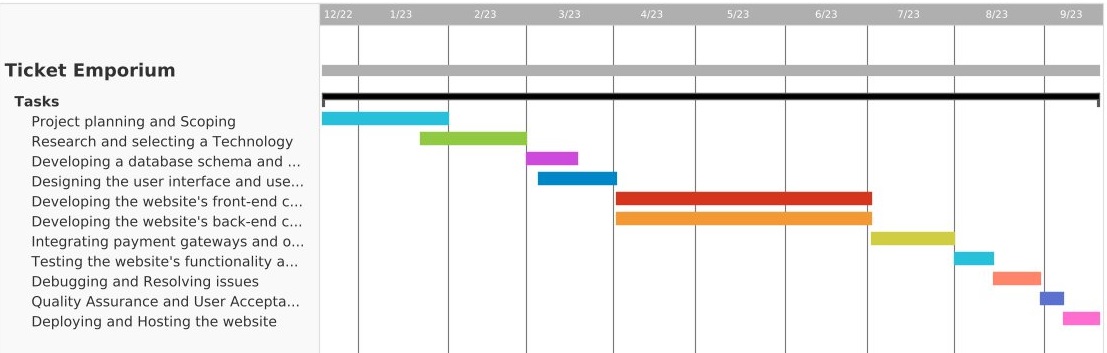


Figure Gantt chart showing the different stages of Ticket Emporium’s development and the corresponding timeline for each stage

# CHAPTER 2: Evaluative Analysis and Comparative Study

## **2.1 Evaluative Report**

Ticket Emporium is a ticket selling website that is designed to allow users to purchase event tickets online. The system is designed to be easy to use, reliable, and efficient. The platform provides a seamless user experience that allows users to quickly find and purchase tickets to their desired events.

The system features a simple and user-friendly interface that allows users to navigate and use the website with ease. It offers a range of payment options and provides a secure platform for transactions. The website also offers customer support and assistance to help users resolve any issues they may face.

## **2.2 Use Case Diagram**

A use case diagram provides a visual representation of the interactions between actors and the system. In the context of Ticket Emporium, the following use case diagram illustrates the interactions between the actors and the system.



Figure Use Case Diagram illustrating interactions between the actors and the system

## **2.3 Abstract Project or Existing Similar Projects**

While the Ticket Emporium project is my original project, there are several existing projects with similar functionalities and features in the market. A review of these related projects will be conducted to identify potential areas for improvement and to ensure that the Ticket Emporium website is competitive in the online ticket selling market.

## **2.4 Analysis of Existing Similar Projects**

There are various ticket selling websites in the market. However, I analyzed three related works that are similar to Ticket Emporium in the ticket selling industry to identify their strengths, weaknesses, and how Ticket Emporium aims to solve any gaps in the current systems.

**TicketSasa**

TicketSasa is a platform where users can purchase or advertise events and event tickets. However, one notable limitation of TicketSasa is the absence of a centralized location for visitors to browse and discover events through their categories and locations. This lack of event categorization and organization can have a negative impact on customers, as they may find it challenging to explore a wide range of events beyond their immediate location. Although TicketSasa does offer the option to view events based on the user's country, it limits the overall visibility and accessibility of events across the platform.

**Patahapa**

Patahapa is a ticket selling platform that allows majors on Sports, Movies, Training, Conferences and even Music Festival events ticketing. Patahapa has faced criticism for its high fees, performance issues and its handling of ticket sales during peak demand periods.

**My ticket**

Mtickets is an online marketplace for buying and selling tickets to various events. It offers a range of features. The primary payment method is through M-pesa My ticket’s fees can be high, and it faces criticism for its handling of customer data.

Comparing Ticket Emporium to these systems, it aims to provide a more user-friendly experience with a strong emphasis on customer support and assistance. Ticket Emporium acknowledges the limitations observed in TicketSasa, such as the absence of a centralized location for browsing and discovering events by categories and locations. In response, Ticket Emporium seeks to provide a comprehensive event categorization and organization system to enhance the customer's ability to explore a wide range of events beyond their immediate location.

Similarly, considering the criticism faced by Patahapa for its high fees and performance issues during peak demand periods, Ticket Emporium is committed to offering transparent pricing with no hidden fees. It also aims to optimize its platform's performance to ensure smooth ticket sales even during high-demand periods.

Furthermore, in contrast to the concerns raised about My Ticket's handling of customer data, Ticket Emporium places a strong emphasis on data security and privacy to ensure a trustworthy and reliable ticket-selling environment.

Overall, Ticket Emporium strives to provide a superior ticket-selling platform by prioritizing user satisfaction, transparent pricing, efficient performance, and robust data security measures. By implementing innovative features like an inventory tracking system to prevent ticket overselling, Ticket Emporium aims to offer a seamless and enjoyable experience for both buyers and sellers.

## **2.5 Pros and Cons of Ticket Emporium**

### **Pros:**

* User-friendly interface.
* Multiple payment options.
* Transparent pricing.
* Secure platform.
* Customer support and assistance.
* Inventory tracking system.

### **Cons:**

* Potential challenges in gaining user trust as a new platform.
* Limited selection of events compared to larger ticket selling platforms.

# CHAPTER 3: Project Methodology and Development Process

## **3.1 Project Methodology**

This chapter outlines the waterfall model used for the development of ‘Ticket Emporium’. The waterfall model is a linear and sequential approach to software development that follows a predefined set of phases. The following figure shows a representation of the processes in the waterfall model.

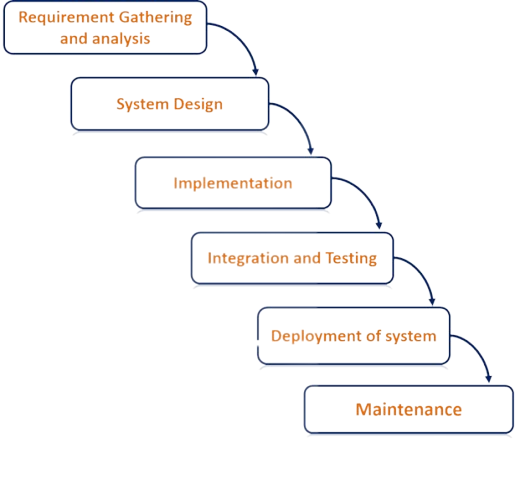


Figure Waterfall Model of System Development

**Planning Phase:** During the planning phase, the requirements for the website were gathered, and a clear scope was defined. A project plan was created, including a timeline with specific milestones and deadlines.

**Design Phase:** In the design phase, the website's architecture and design were defined based on the requirements gathered during the planning phase. The design phase involved creating wireframes, mockups, and prototypes to ensure that the website's design met the client's needs.

**Implementation Phase:** In the implementation phase, the website was developed based on the design specifications. The code was written, tested, and debugged to ensure that the website worked as intended.

**Testing Phase:** In the testing phase, the website was tested to ensure that it met the requirements defined in the planning phase. Various types of testing, such as unit testing, integration testing, and system testing, were conducted to detect and fix any defects.

**Deployment Phase:** In the deployment phase, the website was released to the public. The website was deployed on a web server, and the necessary configurations were made to ensure that the website was accessible to users.

**Maintenance Phase:** During the maintenance phase, the website is being monitored and maintained to ensure that it continues to function correctly. Any necessary updates or bug fixes are being implemented as needed. This phase is ongoing to ensure that the website remains fully functional and up to date.

# CHAPTER 4: System Analysis and Requirement Modeling

## **4.1 Description of Current Systems and Problems**

The current ticket selling systems available in the industry often suffer from issues such as limited user-friendliness, lack of scalability, inefficient inventory management, and inadequate tracking of sales and revenue. These problems result in long wait times, confusion for customers, poor user experience, lost revenue opportunities, and difficulties in managing ticket inventory effectively. Additionally, these systems lack integration with various payment methods, causing further complications in the purchasing process. Therefore, it is crucial to develop a new system design that addresses these issues comprehensively.

## **4.2 Requirement Modeling for Each System Module**

To ensure the successful development and implementation of the Ticket Emporium system, it is crucial to model the requirements for each module comprehensively. In this section, we will utilize various tools, such as flowcharts, to depict the requirements for each module in detail.

### **4.2.1 User Management Module**

The User Management module is responsible for handling user registration, authentication, and account management functionalities. The requirements for this module may include:

**User Registration:**

* Users should be able to create accounts with a unique username and password.
* The system must validate the uniqueness of usernames during registration to prevent duplicates.

**Authentication:**

* Registered users must be able to log in with their credentials.
* The system should securely store and verify user passwords using encryption.

**Account Management:**

* Users should be able to update their profile information, such as email, name, and contact details.
* The system must provide password reset functionality in case users forget their passwords.

**Role-Based Access:**

* The system should support different user roles, such as regular users and administrators.
* Certain functionalities, like user deletion or content management, should be restricted to administrators.

**User Profile:**

* Each user should have a profile page displaying their basic information and account activity.
* Users should be able to view and manage their activities, such as past orders or event registrations.

**Security Measures:**

* The system must implement proper security measures to protect user data and prevent unauthorized access.
* Passwords should be securely hashed and stored in the database.

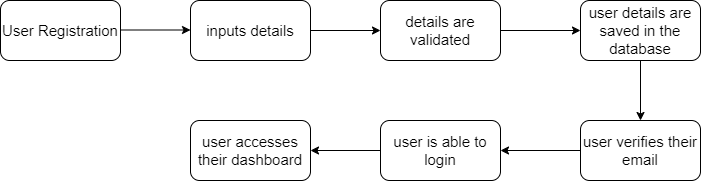


Figure Flowchart illustrating the user registration and authentication process

### **4.2.2 Event Management Module**

The Event Management module is responsible for creating and managing events on the Ticket Emporium platform. The requirements for this module may include:

**Event Creation:**

• Organizers should be able to create new events with details such as event name, date, location, and description.

• Each event must have at least one associated ticket listing.

**Event Updates:**

• Organizers should be able to update event information as needed (e.g., changes in event schedule or location).

• Users who have purchased tickets to an event should be notified of any updates.

**Event Visibility:**

• Events should be displayed with relevant details, and users should be able to browse and search for events.

• The system should support event categorization to improve discoverability.

**Event Registration:**

• Users should be able to view and register for events they are interested in attending.

• The system should generate a confirmation for successful event registration.

**Event Analytics:**

• The system should provide event organizers with analytics, such as ticket sales, attendance, and user demographics.



Figure Flowchart illustrating the event creation process by the event organiser

### **4.2.3 Ticket Listing Module**

The Ticket Listing module is responsible for managing event tickets listed for sale by sellers. The requirements for this module are:

**Ticket Creation:**

• Sellers should be able to create new ticket listings with event details, ticket price, and quantity available.

• Ticket listings must be associated with the corresponding event.

**Ticket Management:**

• Sellers should be able to edit or delete their ticket listings as needed.

• The system should prevent sellers from editing or deleting tickets once they are sold out.

**Ticket Display:**

• Ticket listings should be displayed with event information, ticket availability, and price.

• Users should be able to filter and search for specific ticket listings.

**Ticket Inventory:**

• The system must track ticket quantities and ensure accurate inventory management.

• Sellers should be notified when ticket quantities are low.

### **4.2.4 Event Listing Module**

The Event Listing module is responsible for managing and displaying a comprehensive list of available events on the Ticket Emporium platform. The requirements for this module may include:

**Event Listing:**

* The system should showcase a user-friendly and visually appealing event listing page that provides detailed information about each event, including event name, date, location, ticket prices, and event description.
* Users should be able to filter events based on categories, locations, dates, and other relevant criteria to easily find events of interest.
* Events should be displayed with clear calls-to-action, allowing users to view event details, select tickets, and proceed to checkout seamlessly.

**Event Search:**

* The system must support a robust event search functionality, enabling users to quickly find specific events by entering keywords, event names, or other relevant search queries.
* Search results should be presented in a clear and organized manner, with relevant events displayed based on user search criteria.

**Event Details:**

* Each event listing should have a dedicated event details page, providing comprehensive information about the event, such as event schedule, venue details, event organizers, and any additional relevant details.
* Users should be able to view event images and read event reviews to make informed decisions about attending the event.

**Ticket Availability:**

* The system should dynamically display the availability of tickets for each event, indicating the number of available tickets and any remaining tickets in real-time.
* Users should be able to view seating charts for events that offer reserved seating options, helping them make ticket selections based on their seating preferences.

**Event Recommendations:**

* The system should incorporate a recommendation engine that suggests related or similar events to users based on their past interactions, purchase history, and preferences.
* Recommendations should be prominently displayed on the event listing page to encourage users to explore additional events of interest.

### **4.2.5 Ticket Purchase Module**

The Ticket Purchase module is responsible for handling the ticket purchasing process on the Ticket Emporium platform. The requirements for this module may include:

**Ticket Selection:**

• Users should be able to select specific ticket listings and add them to their cart for purchase.

• The system should provide a clear view of available ticket options and prices.

**Checkout Process:**

• Users should be guided through a seamless checkout process to complete their ticket purchase.

• The system should collect necessary information, such as user details and payment information.

**Order Confirmation:**

• Users should receive an order confirmation with details of their ticket purchase.

• The system should also provide users with a unique order ID for reference.

**Guest Checkout:**

• Optional feature: The system can offer a guest checkout option for users who do not wish to create an account.

**Order History:**

• Registered users should be able to view their order history, including past ticket purchases and order status.

C:\Users\admin\Desktop\user_registration_flow_chart.png

Figure Flowchart illustrating the ticket purchase process

### **4.2.6 Admin Management Module**

The Admin Management module in the Ticket Emporium system plays a crucial role in overseeing and managing various aspects of the platform. Admins are granted special privileges to monitor and maintain the system's integrity while ensuring a seamless user experience. With this module in place, the Ticket Emporium system empowers admins to efficiently oversee and maintain the platform's operations. By providing necessary tools and privileges, admins can ensure a secure and user-friendly environment for both buyers and sellers. The requirements for the Admin Management module are as follows:

**User Roles and Privileges:**

* The system must define different user roles, with "Admin" being one of them. Admins should have unique privileges to access and manage administrative functionalities.
* Admins should be able to view and update user roles to grant or revoke specific privileges as needed.

**Event Management:**

* + Admins should have the ability to add, edit, or remove events from the system. This includes specifying event details such as title, date, time, location, and ticket pricing.
  + The system should allow admins to categorize events for better organization and user browsing.

**Content Management:**

* Admins should have control over the content displayed on the website, such as announcements, news, or event promotions.
* They should be able to add, edit, or remove content to keep the platform up-to-date and relevant.

**Order Management:**

* Admins should be able to view all orders placed by users and have access to order details, including ticket quantities, payment status, and customer information.
* They should have the authority to manage and resolve any issues related to orders or ticket purchases.

**User Management:**

* Admins should be able to view and manage user accounts, including user details, activities, and account status.
* They should have the ability to suspend or terminate user accounts in case of violations or misuse.

**Reports and Analytics:**

* Admins should have access to various reports and analytics to track the platform's performance, user engagement, and revenue.
* The system should generate reports on ticket sales, popular events, and user activity to help admins make data-driven decisions.

**System Configuration:**

* Admins should be able to configure system settings, such as email notifications, security parameters, and website appearance.
* They should have the authority to enable or disable certain features or functionalities based on the system's requirements.

**Security Measures:**

* The Admin Management module must implement robust security measures to prevent unauthorized access and data breaches.
* Admin accounts should have multi-factor authentication and strong password requirements to ensure secure access.

**Audit Trail:**

* The system should maintain an audit trail of admin actions and activities to track changes made within the system.
* The audit trail helps in monitoring admin activities and identifying any unauthorized modifications.

By modeling the requirements for each module, we can ensure a clear understanding of the system's functionalities and the necessary components to be developed. The flowcharts serve as a reference for the development and assist in proper implementation.

# CHAPTER 5: System Design and Database Architecture

## **5.1 Conceptual Model**

The conceptual model of the Ticket Emporium system provides a high-level overview of the entities, relationships, and attributes involved in the ticket selling process. It helps to establish a clear understanding of the system's structure and functionality. The following Entity-Relationship Diagram (ERD) represents the conceptual model of the Ticket Emporium system:

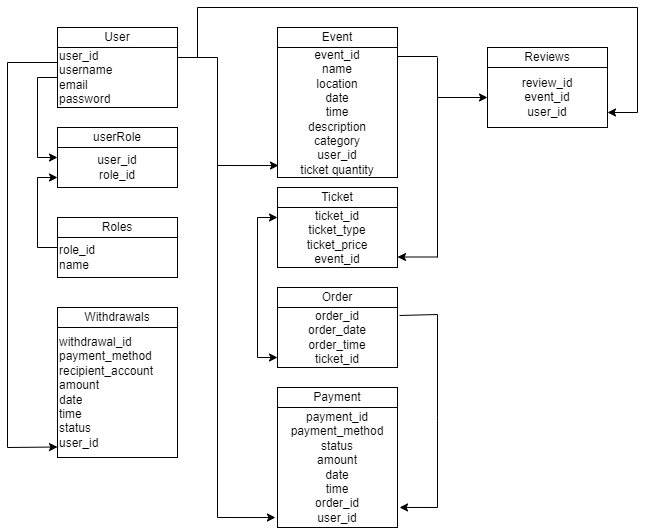


Figure Entity-Relationship Diagram (ERD) representing the conceptual model of the Ticket Emporium system.

The conceptual model showcases the main entities. The relationships between these entities are defined, including associations such as event-seller, buyer-payment, and others. The attributes for each entity are also identified, capturing relevant information like user details, event information, ticket quantities, payment details, and more.

## **5.2 Physical Model**

The physical model of the Ticket Emporium system outlines the implementation details for various entities, including users, roles, userRoles, events, tickets, orders, reviews, payments, and withdrawals. It describes how these conceptual entities, relationships, and attributes will be translated into physical tables in the database, considering primary keys, foreign keys, unique keys, and data types. The physical database design ensures the efficient storage and retrieval of data. The following is an overview of the physical tables in the Ticket Emporium system:

1. Users Table

- user\_id (Primary Key, Auto-increment)

- username (String)

- email (String)

- password (String)

2. Roles Table

- role\_id (Primary Key, Auto-increment)

- role\_name (String)

3. UserRoles Table

- user\_id (Foreign Key)

- role\_id (Foreign Key)

4. Events Table

- event\_id (Primary Key, Auto-increment)

- event\_name (String)

- event\_date (Date)

- event\_location (String)

- event\_description (Text)

5. Tickets Table

- ticket\_id (Primary Key, Auto-increment)

- event\_id (Foreign Key)

- ticket\_name (String)

- ticket\_price (Decimal)

- ticket\_quantity (Integer)

6. Orders Table

- order\_id (Primary Key, Auto-increment)

- user\_id (Foreign Key)

- order\_date (Date)

- total\_amount (Decimal)

7. Reviews Table

- review\_id (Primary Key, Auto-increment)

- user\_id (Foreign Key)

- event\_id (Foreign Key)

- review\_text (Text)

- review\_rating (Integer)

8. Payments Table

- payment\_id (Primary Key, Auto-increment)

- user\_id (Foreign Key)

- order\_id (Foreign Key)

- payment\_amount (Decimal)

- payment\_date (Date)

9. Withdrawals Table

- withdrawal\_id (Primary Key, Auto-increment)

- user\_id (Foreign Key)

- withdrawal\_amount (Decimal)

- withdrawal\_date (Date)

The physical model ensures that the Ticket Emporium system's data is well-structured and organized, facilitating effective data storage and retrieval for seamless system functionality.

# CHAPTER 6: System Implementation

## **6.1 Tools and Technologies**

The implementation of the Ticket Emporium system relies on a combination of tools and technologies to develop a robust and user-friendly ticket selling platform. The following tools were utilized during the system implementation:

**Programming Languages:** The system was primaly developed using JavaScript in combination with some of it’s frameworks/libraries. These include: VueJS, Express.

**Frameworks and Libraries:** To enhance development efficiency and streamline code organization, popular frameworks and libraries were employed. These include VueJS for the frontend, Express.js for the backend, and Sequelize as the ORM (Object-Relational Mapping) tool.

**Database Management System:** The Ticket Emporium system utilizes a relational database management system, specifically MySQL, to store and manage data effectively.

**Integrated Development Environment (IDE):** An IDE, specifically Visual Studio Code, was used for coding and development tasks. The IDE offers advanced features, debugging capabilities, and code editor extensions to enhance productivity.

**Version Control:** Git, a distributed version control system, was employed to track changes and maintain code integrity throughout the development process. GitHub was utilized as a remote repository for code hosting and version control management.

## **6.2 Sample Input/Output Screenshots**

To provide a glimpse of the Ticket Emporium system's user interface and functionality, here are some sample input/output screenshots:

### **1. User Registration Form:**

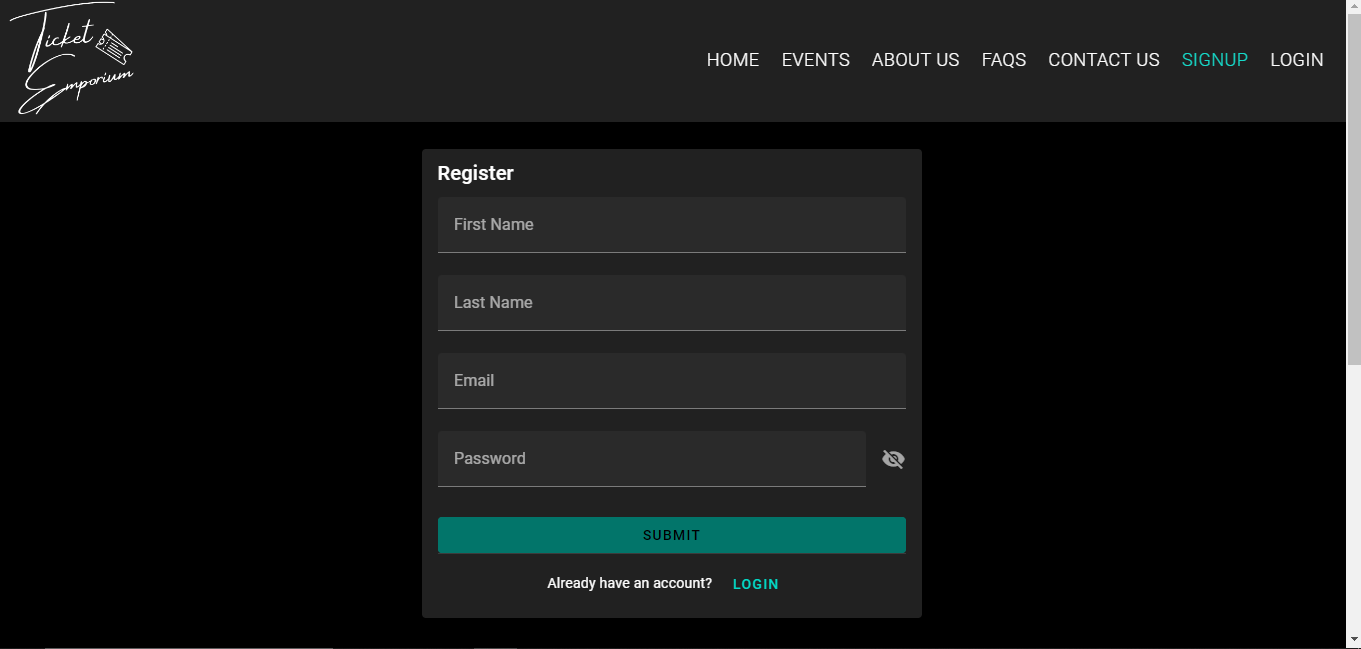


Figure Screenshot of the user registration form

### **2. Seller Dashboard:**

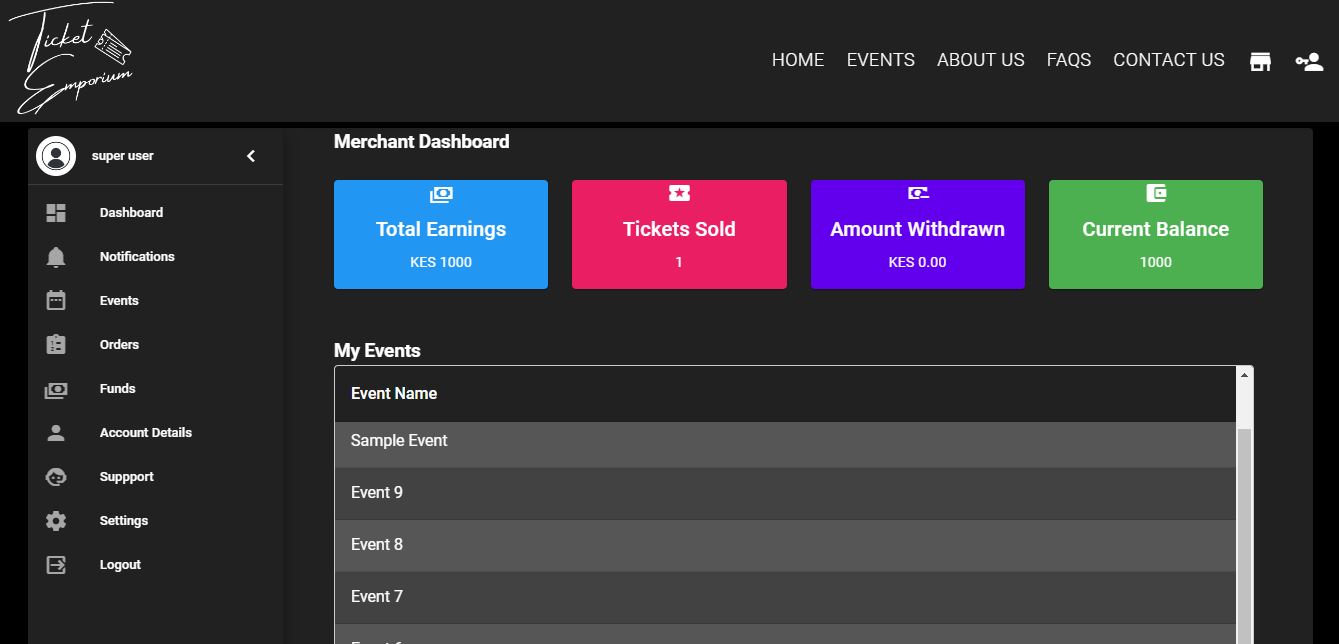


Figure 9 Screenshot of the seller dashboard

These sample screenshots showcase the user interface components and system interactions that users can expect when using the Ticket Emporium platform.

**Note:** The screenshots provided here are for illustrative purposes only and may not represent the complete user interface or functionality of the Ticket Emporium system.

The implementation phase of the Ticket Emporium system involved the utilization of appropriate tools, technologies, and frameworks to bring the envisioned ticket selling platform to life. By leveraging modern development practices and user-centric design principles, the system aims to deliver a seamless and satisfying user experience for buyers and sellers alike.

# CHAPTER 7: Limitations and Recommendations

This chapter highlights the limitations encountered during the development of the Ticket Emporium website, as well as recommendations for further improvements in the system.

## **7.1 Limitations**

During the course of developing the Ticket Emporium website, the following limitations were encountered:

**Time constraints:** The project had a tight deadline, which made it challenging to implement all the desired features and conduct thorough testing.

**Budget limitations:** There were financial constraints that limited the purchase of certain resources and tools that could have been beneficial to the project.

**Technical challenges:** There were instances of technical difficulties encountered during the development process, such as compatibility issues with certain software tools.

## **7.2 Recommendations**

In order to improve the Ticket Emporium website and address some of the limitations encountered during the development process, the following recommendations are suggested:

**Enhanced User Experience:** Conduct user testing and gather feedback to identify areas of improvement in terms of usability and user interface. Incorporate user feedback to make the system more intuitive, user-friendly, and visually appealing.

**Performance Optimization:** Continuously monitor and optimize the system's performance by identifying and addressing any bottlenecks or areas where the system may experience slow response times. This could involve optimizing database queries, improving code efficiency, or utilizing caching mechanisms.

**Scalability and Security:** As the user base and data volume grow, ensure that the system is designed to handle increased traffic and can scale effectively. Implement robust security measures to safeguard user information and protect against potential threats such as data breaches or unauthorized access.

**Integration with Payment Gateways:** Explore the integration of additional payment gateways to provide users with a wider range of payment options, accommodating different preferences and ensuring smooth transactions.

**Continuous Maintenance and Updates:** Establish a maintenance plan to address any bugs, security vulnerabilities, or performance issues that may arise. Regularly update the system to incorporate new features, address user feedback, and stay up-to-date with evolving technologies and industry standards.

**Address technical challenges:** To mitigate technical challenges, it is recommended to conduct regular system checks and ensure that all software tools are compatible with each other.

**Enhanced security measures:** To ensure the safety of users' personal and financial information, it is recommended to implement additional security measures, such as two-factor authentication and encryption.

## **7.3 Conclusion.**

By acknowledging the limitations and challenges faced during the development process and adopting the recommended strategies, Ticket Emporium can continue to enhance its system and provide a better user experience. Emphasizing user-centric design, performance optimization, scalability, and security will contribute to the system's success and competitiveness in the ticket selling industry. The implementation of these recommendations will ensure that Ticket Emporium remains a reliable, efficient, and customer-oriented platform for both event organizers and ticket buyers.

# REFERENCES

Nyabuto, John. “List of All Sites for Purchasing Event Tickets Online in Kenya.” *Victor Matara*,

23 Jan. 2020, <https://victormatara.com/list-of-event-ticketing-companies-in-kenya/>

OpenAI.. *ChatGPT*,

2023, [https://chat.openai.com](https://platform.openai.com/)

# APPENDIX

## **User Manual for Ticket Emporium - Online Ticket Selling Platform (for ticket buyers)**

Welcome to Ticket Emporium, your one-stop destination for purchasing event tickets conveniently online. This user manual aims to provide a comprehensive guide on how to use the Ticket Emporium platform effectively, ensuring a seamless ticket buying experience. Let's get started!

**Covered Areas:**

Creating an Account

* + Registration
  + Login

Browsing Events

* Event Categories
* Search Functionality

Purchasing Tickets

* Selecting an Event
* Ticket Options and Pricing
* Adding Tickets to Cart
* Proceeding to Checkout
* Payment Options

Managing Your Account

* Updating Profile Information
* Viewing Order History
* Resetting Password

**1. Creating an Account**

**Registration:**

To begin using Ticket Emporium, you need to create an account. Click on the "Sign Up" button on the homepage and provide the required information, including your name, email address, and a secure password. Ensure the password meets the security requirements for your account's safety.

**Login:**

For returning users, simply click on the "Login" button on the homepage and enter your registered email and password. Click "Login," and you will be redirected to your account dashboard.

**2. Browsing Events**

**Event Categories:**

On the homepage, you'll find a list of event categories, such as Sports, Concerts, Movies, Conferences, and more. Click on a category of interest to view the events available in that category.

**Search Functionality:**

Looking for a specific event? Utilize the search bar at the top of the page to find events by name, location, or date. Our advanced search feature will help you quickly find your desired event.

**3. Purchasing Tickets**

**Selecting an Event:**

Once you've found an event of interest, click on it to view event details, including date, time, location, and ticket options.

**Ticket Options and Pricing:**

Choose the ticket type that suits your needs, such as General Admission or VIP. The ticket prices will be displayed next to each option.

**Adding Tickets to Cart:**

Select the number of tickets you wish to purchase and click the "Add to Cart" button. You can add tickets from multiple events to your cart.

**Proceeding to Checkout:**

When you've added all your desired tickets, click on the shopping cart icon at the top right corner of the page to proceed to checkout.

**Payment Options:**

Ticket Emporium offers various secure payment options. Choose your preferred payment method and follow the instructions to complete the transaction.

**4. Managing Your Account**

**Updating Profile Information:**

Need to change details? Navigate to the "Account Settings" section under the user icon to update your profile information.

**Viewing Order History:**

To keep track of your past ticket purchases, click on "Order History" under the user icon to view a list of all your previous transactions.

**Resetting Password:**

If you forget your password, click on the "Forgot Password" link on the login page. Follow the instructions to reset your password and regain access to your account.

## **Code Sample: User Authentication API Endpoints**

In the code sample below, we will illustrate the backend implementation of both the User Login and User Signup API endpoints for the Ticket Emporium system. These endpoints handle user authentication and registration, allowing users to securely log in and create new accounts on the platform.

import jwt from 'jsonwebtoken';

import bcrypt from 'bcrypt';

import db from "../models/index.js";

import dotenv from 'dotenv'; // Import dotenv to load environment variables

dotenv.config(); // Load environment variables from .env file

const User = db.User;

const Role = db.Role;

const UserRole = db.UserRole;

const signup = async (req, res, next) => {

  const { firstname, lastname, email, password } = req.body;

  try {

    // Check if the email already exists in the database

    const existingUser = await User.findOne({ where: { email: email } });

    if (existingUser) {

      return res.status(400).json({ message: 'This Email is already registered. Please use a different email.' });

    }

    // If the email is not found, proceed to create a new user

    const hash = await bcrypt.hash(password, 10);

    const user = await User.create({ firstName: firstname, lastName: lastname, email: email, password: hash });

    // Create a new user role for the buyer

    const buyerRole = await Role.findOne({ where: { role: 'buyer' } });

    if (buyerRole) {

      await user.addRole(buyerRole); // Assuming you have a method like addRole to associate the user with a role

    }

    // Generate a new JWT token

    const token = jwt.sign({ id: user.id }, process.env.SECRET\_KEY, { expiresIn: '1hr' });

    // Save the token in the database

    await user.update({ verificationToken: token });

    // Respond with the token to the user

    res.json({ token });

  } catch (error) {

    next(error);

  }

};

const login = async (req, res, next) => {

  const { email, password } = req.body;

  try {

    const user = await User.findOne({ where: { email: email } });

    if (!user) {

      return res.status(401).json({ message: "This email is not registered" });

    }

    // Check if the user is verified

    if (!user.verified) {

      return res.status(401).json({ message: "Your Email is not verified. Please verify your email to log in." });

    }

    // Check if the user's status is 'deactivated'

    if (user.status == 'deactivated') {

      return res.status(401).json({ message: "Your account is deactivated. Please contact support for recovery." });

    }

    // Check if the user's status is 'deactivated'

    if (user.status == 'blocked') {

      return res.status(401).json({ message: "Your account is blocked. Please contact support for recovery." });

    }

    const result = await bcrypt.compare(password, user.password);

    if (result) {

      // Now, retrieve the user's role IDs from the user\_roles table

      const userRoles = await UserRole.findAll({ where: { user\_id: user.id } });

      // Extract an array of role IDs from the userRoles

      const roleIds = userRoles.map(userRole => userRole.role\_id);

      const token = jwt.sign({ id: user.id }, 'secret\_jwt', { expiresIn: '1hr' });

      // Include token, user information along with the role IDs in the response

      res.json({ token: token, user: user, roleIds: roleIds });

    } else {

      res.status(401).json({ message: "Invalid email or password" });

    }

  } catch (error) {

    console.error('Login error:', error);

    next(error);

  }

};

**Explanation:**

In this code, we first import various modules and dependencies required for user authentication and security. Notably, we import jsonwebtoken (jwt) for handling JSON Web Tokens (JWT) and bcrypt for securely hashing passwords. Additionally, we import database models and environment variables using Sequelize and dotenv.

**User Signup API Endpoint:**

1. Create the User Signup API Endpoint:

We define a POST endpoint at the /signup route to handle user registration. This endpoint expects the user's first name, last name, email, and password as part of the request body.

2. Check for Existing Users:

We query the database to check if a user with the given email already exists. If an existing user is found, we return a 400 Bad Request status code with a message indicating that the email is already registered.

3. Hash the Password:

To ensure the security of user passwords, we use the bcrypt library to hash the password before storing it in the database. The hashed password is then saved along with the user's first name, last name, and email.

4.Save the User to the Database:

We create a new User instance with the provided user information and save it to the database using the create method.

5. Create a User Role (e.g., Buyer):

We create a new user role, such as "buyer," and associate it with the user.

6. Generate and Return a JSON Web Token (JWT):

Upon successful registration, we generate a JSON Web Token (JWT) using the jsonwebtoken library. The token includes the user's ID as its payload and is signed with a secret key. It is set to expire in 1 hour for security reasons.

7. Return the JWT in the Response:

Finally, we send the generated JWT in the response as a JSON object. The client can store this token and use it to access protected routes within the application.

8. Handle Errors:

We use a try...catch block to handle any errors that may occur during the authentication and registration processes. If an error occurs, we pass it to the next middleware to handle the error appropriately.

**User Login API Endpoint:**

Create the User Login API Endpoint:

We define a POST endpoint at the /login route to handle user login. This endpoint expects the user's email and password as part of the request body.

Find the User in the Database:

We use the User model to query the database and find the user with the provided email. If the user is not found, we return a 401 Unauthorized status code with a message indicating invalid credentials.

Verify User Status and Email Verification:

After finding the user, we check various conditions, such as whether the user's email is verified and whether the user's status is "deactivated" or "blocked." Depending on the user's status, we respond with the appropriate messages.

Verify the Password:

If the user is found and has a valid status, we use the bcrypt library to compare the provided password with the hashed password stored in the database. If the passwords match, the user is authenticated.

Generate and Return a JSON Web Token (JWT):

Upon successful authentication, we generate a JSON Web Token (JWT) using the jsonwebtoken library. The token includes the user's ID as its payload and is signed with a secret key. It is set to expire in 1 hour for security reasons.

Return the JWT in the Response:

Finally, we send the generated JWT in the response as a JSON object. The client can store this token and use it to access protected routes within the application.

Handle Errors:

As with the signup endpoint, we use a try...catch block to handle errors during the login process and pass them to the next middleware to handle the error appropriately.

This code demonstrates the implementation of both the User Signup and User Login API endpoints, which are crucial parts of the authentication and security features in your application. The use of JWTs, password hashing, and secure database queries helps ensure user data remains safe and secure.