Chapter 1: Introduction

1.1 Introduction

The "Popcorn" Online Movie Ticket Booking website aims to provide users with a seamless and convenient platform to purchase movie tickets. This chapter introduces the project, outlining its objectives and the key features that will be incorporated into the system.

1.2 System Request

The system request for "Popcorn" emphasizes the need for an efficient online movie ticket booking system. It highlights the benefits the system will bring to users, including the ability to easily browse and purchase tickets, manage user profiles, and provide valuable feedback through reviews.

1.3 Plan/Methodology

The development plan and methodology for "Popcorn" involve leveraging technologies such as PHP, JavaScript, Laravel, MySQL, and Tailwind CSS. This chapter discusses the overall approach to be taken, including the development lifecycle, key milestones, and the iterative process of designing and implementing the system.

Chapter 2: Requirement Study

The requirement study phase delves into the detailed analysis of the system's functional and non-functional requirements. This chapter outlines the specific needs and expectations of the stakeholders, providing a foundation for the subsequent development phases.

**2.1 Functional Requirements**

2.1.1 User Module:

* **User Registration:**
  + New users can register with valid credentials.
* **User Login:**
* Registered users can log in using their credentials.
* **Ticket Purchase:**
* Users can browse movie details, select a hall, and securely purchase tickets.

2.1.2 Hall Management Module

* **Manager Approval:**
* Admin can approve or reject hall manager requests.
* **Hall Management:**
* Hall managers can oversee movie details, trailers, pricing, and reviews for their respective halls.

2.1.3 Admin Module

* **Dashboard:**
* Admin has an interactive dashboard for system-wide management.
* **Hall Management:**
* Admin can manage all halls, approve/reject manager requests, and oversee system operations.

**2.2 Non-functional Requirements**

2.2.1 Performance

* The system should handle a minimum of 1000 concurrent users.
* Response time for user interactions should be less than 2 seconds.

2.2.2 Security

* User data should be securely stored and transmitted.
* Payment transactions must be encrypted using SSL.
* Admin access is protected with secure authentication mechanisms.

2.2.3 Compatibility

* The system should be compatible with popular web browsers (Chrome, Firefox, Safari).

2.2.4 Usability

* The user interface should be intuitive and easy to navigate.
* The website must be responsive to various devices.

Chapter 3: Design Diagrams

**3.1 Design Diagrams**

3.1.1 User Interface Mockups

This section presents the visual representation of the "Popcorn" user interface. Included are mockups illustrating the layout and design of key pages such as the home screen, movie details page, user profile, and checkout process.

3.1.2 System Architecture Diagram

The system architecture diagram provides an overview of the components and their interactions within the "Popcorn" Online Movie Ticket Booking system. It outlines the flow of data between the user interface, server, database, and external services such as the Stripe payment gateway.

**3.2 Database Plan**

3.2.1 Entity-Relationship (ER) Diagram

The ER diagram depicts the relationships between entities in the "Popcorn" database. Entities such as User, Hall, Movie, Transaction, and Review are illustrated along with their attributes and connections.

3.2.2 Enhanced Entity-Relationship (EER) Diagram

Building upon the ER diagram, the EER diagram incorporates additional details such as specialization, generalization, and other relationships that enhance the understanding of the data model.

3.2.3 Database Schema

The database schema outlines the structure of the database tables, their fields, and relationships. This section provides a detailed description of each table, specifying primary and foreign keys, data types, and any constraints.

3.2.3.1 User Table

Fields: UserID (PK), Username, Password, Email, ...

3.2.3.2 Hall Table

Fields: HallID (PK), HallName, ManagerID (FK), ...

3.2.3.3 Movie Table

Fields: MovieID (PK), Title, Description, Price, ...

3.2.3.4 Transaction Table

Fields: TransactionID (PK), UserID (FK), HallID (FK), Amount, ...

3.2.3.5 Review Table

Fields: ReviewID (PK), UserID (FK), HallID (FK), Rating, Comment, ...

3.3 System Flow Diagram

The system flow diagram illustrates the step-by-step flow of activities within the "Popcorn" system. It provides a high-level overview of user interactions, data processing, and the flow of information between different components.

Chapter 4: Implementation Details

**4.1 Pseudocode**

4.1.1 User Authentication (PHP)

-Code\*\*\*

4.1.2 Ticket Purchase (Laravel Controller)

-Code\*\*\*

**4.2 Code Complexity**

The code complexity of the "Popcorn" system is moderate. It involves CRUD operations, user authentication, and payment processing. The use of Laravel framework simplifies many aspects of development, reducing overall complexity.

**4.3 Screenshots**

4.3.1 Home Screen

-Home(Picture)

Figure 4.3.1: The home screen displays a list of available movies and provides a user-friendly interface for navigation.

4.3.2 User Profile

-User Profile(Picture)

Figure 4.3.2: The user profile page allows users to manage their personal information, view transaction history, and submit reviews.

**4.4 System Architecture Image**

4.4.1 System Architecture

-System Architecture(Picture)

Figure 4.4.1: The system architecture illustrates the flow of data between the user interface, server, database, and external services.

**4.5 Description and Leveling**

**4.5.1 User Authentication**

4.5.1.1 Description

The user authentication process ensures that registered users can securely log in to the "Popcorn" system.

4.5.1.2 Leveling

**Level 1:**

* Validate input parameters.
* Query the database to check user credentials.

**Level 2:**

* Check if the user exists and credentials are valid.
* Return authentication status.

**4.5.2 Ticket Purchase**

4.5.2.1 Description

The ticket purchase process allows users to select a movie, choose the quantity of tickets, and securely complete the transaction.

4.5.2.2 Leveling

**Level 1:**

* Validate input parameters.
* Calculate the total amount based on ticket price and quantity.

**Level 2:**

* Create a transaction record in the database.
* Process payment using the Stripe gateway.
* Return transaction details.

Chapter 5: Testing Results

**5.1 Black Box Testing**

5.1.1 User Authentication

**Test Case 1: Valid Login**

* **Input**: Valid username and password.
* **Expected Output**: Successful login.
* **Result**: Pass

**Test Case 2: Invalid Login**

* **Input**: Invalid username or password.
* **Expected Output**: Authentication failure.
* **Result**: Pass

5.1.2 Ticket Purchase

**Test Case 3: Successful Ticket Purchase**

* **Input**: Valid movie selection, quantity, and payment.
* **Expected Output**: Successful transaction.
* **Result**: Pass

**Test Case 4: Insufficient Funds**

* **Input**: Valid movie selection and quantity, insufficient funds.
* **Expected Output**: Transaction failure.
* **Result**: Pass

**5.2 White Box Testing**

5.2.1 User Authentication

**Test Case 5: SQL Injection**

* **Input**: Malicious input attempting SQL injection.
* **Expected Output**: Prevent SQL injection and return authentication failure.
* **Result**: Pass

5.2.2 Ticket Purchase

**Test Case 6: Transaction Calculation**

* **Input**: Valid movie selection and quantity.
* **Expected Output**: Correct calculation of the total amount.
* **Result**: Pass

**Test Case 7: Database Integrity**

* **Input**: Valid ticket purchase.
* **Expected Output**: Accurate recording of transaction details in the database.
* **Result**: Pass

**5.3 Performance Testing**

5.3.1 Concurrent Users

**Test Case 8: Simultaneous Ticket Purchase**

* **Input**: Multiple users attempting to purchase tickets simultaneously.
* **Expected Output**: System can handle concurrent transactions without significant performance degradation.
* **Result**: Pass

**5.4 System Integration Testing**

5.4.1 Payment Gateway Integration

**Test Case 9: Successful Stripe Integration**

* Input: Payment processing using the Stripe gateway.
* Expected Output: Successful completion of transactions.
* Result: Pass

5.5 User Acceptance Testing (UAT)

5.5.1 General User Experience

**Test Case 10: User-Friendly Interface**

* Input: User navigates through the website.
* Expected Output: Intuitive and user-friendly experience.
* Result: Pass

**5.6 Summary of Results**

The testing results indicate that the "Popcorn" Online Movie Ticket Booking system has successfully passed various black box, white box, performance, system integration, and user acceptance tests. The system demonstrates robustness in handling different scenarios, ensuring a secure and user-friendly experience.

Chapter 6: User Manual

**6.1 Introduction**

Welcome to the "Popcorn" Online Movie Ticket Booking System! This user manual provides comprehensive guidance on using the system efficiently. Whether you are a user looking to purchase movie tickets, a hall manager managing your hall, or an admin overseeing the entire system, this manual will assist you in navigating through the features seamlessly.

**6.2 Table of Contents**

**User Section**

* Registering an Account
* Logging In
* Browsing Movies
* Purchasing Tickets
* Managing User Profile
* Submitting Reviews

**Hall Manager Section**

* Requesting Manager Access
* Managing Hall Information
* Overseeing Hall Operations

**Admin Section**

* Admin Dashboard
* Managing Halls
* Approving Hall Manager Requests

**6.3 User Section**

6.3.1 Registering an Account

To access the "Popcorn" system, you need to register an account. Follow these steps:

* Click on the "Sign Up" or "Register" button on the homepage.
* Fill in the required information, including your username, email, and password.
* Click "Submit" to complete the registration.

6.3.2 Logging In

If you already have an account, follow these steps to log in:

* Click on the "Log In" button on the homepage.
* Enter your username and password.
* Click "Log In" to access your account.

6.3.3 Browsing Movies

* Navigate to the homepage.
* Browse through the list of available movies.
* Click on a movie to view details, including trailers and pricing.

6.3.4 Purchasing Tickets

* Select a movie.
* Choose the number of tickets and click "Purchase."
* Follow the on-screen instructions to complete the transaction.

6.3.5 Managing User Profile

* Go to the user profile section.
* Update your personal information and view transaction history.

6.3.6 Submitting Reviews

* After purchasing tickets, go to the user profile section.
* Find the "Submit Review" option.
* Provide a rating and optional comments for the hall.

**6.4 Hall Manager Section**

6.4.1 Requesting Manager Access

* Log in to your user account.
* Navigate to the admin-managed halls section.
* Click on "Request Manager Access" for the desired hall.

6.4.2 Managing Hall Information

* Log in to the system as a hall manager.
* Access the hall management dashboard.
* Update movie details, pricing, and hall-specific information.

6.4.3 Overseeing Hall Operations

* Monitor ticket sales and transaction history.
* Respond to user reviews and feedback.

6.5 Admin Section

6.5.1 Admin Dashboard

* Access the admin dashboard to view system-wide statistics.
* Manage halls, approve/reject manager requests, and oversee operations.

6.5.2 Managing Halls

* View a list of all halls in the system.
* Edit hall information, including managers and operational details.

6.5.3 Approving Hall Manager Requests

* Access the admin dashboard.
* Review pending hall manager requests.
* Approve or reject requests based on the provided information.

**6.6 Conclusion**

Congratulations! You have successfully navigated the "Popcorn" Online Movie Ticket Booking System. For any additional assistance, refer to the help section or contact our support team.

Chapter 7: Deployment

**7.1 Deployment Documentation**

7.1.1 Overview

The deployment process for the "Popcorn" Online Movie Ticket Booking System involves transferring the developed application from the development environment to a live production server. This documentation outlines the steps and considerations for a successful deployment.

7.1.2 Deployment Plan

Step 1: Pre-Deployment Preparation

* **Backup**: Ensure a backup of the entire application and database is created for recovery purposes.
* **Environment Check**: Verify that the production server meets the system requirements for hosting the application.

Step 2: Code Repository

* **Clone Repository**: Clone the latest version of the application from the version control system (e.g., Git).
* **Update Dependencies**: Run composer install to ensure all required dependencies are up to date.

Step 3: Configuration

* **Environment Variables**: Configure environment variables for the production environment, including database connection details, API keys, and other sensitive information.
* **File Permissions**: Set appropriate file permissions for security.

Step 4: Database Migration

* **Run Migrations**: Execute database migrations to create necessary tables and structures.
* **Seed Database (Optional)**: Populate the database with initial data if needed.

Step 5: Web Server Configuration

* **Virtual Host**: Set up a virtual host on the web server (e.g., Apache or Nginx) to point to the application's public directory.
* **SSL Configuration**: If applicable, configure SSL for secure communication.

Step 6: Security Measures

* **Firewall Settings**: Configure server firewall rules to restrict access.
* **Update Software**: Ensure all server software, including the operating system and web server, is up to date with the latest security patches.

Step 7: Testing

* **Health Check**: Perform a health check to ensure the application is running smoothly.
* **Browser Testing**: Conduct cross-browser testing to verify compatibility.

Step 8: Monitoring

* **Logging**: Set up logging mechanisms to monitor errors and application activities.
* **Performance Monitoring**: Implement tools for performance monitoring and issue tracking.

Step 9: Final Checks

* **Backup (Again)**: Take a final backup of the deployed application and database.
* **Domain Configuration**: Confirm that the domain is correctly configured to point to the application.

Step 10: Deployment

* **Deploy Application**: Transfer the application files to the production server.
* **Restart Services**: Restart the web server and any relevant services to apply changes.

7.1.3 Post-Deployment

* **Monitoring**: Continuously monitor the application for performance, security, and potential issues.
* **Scalability**: Plan for future scalability and consider load balancing if needed.

**7.2 Domain**

7.2.1 Domain Registration

* **Choose a Domain Name**: Select a relevant and memorable domain name.
* **Register Domain**: Use a domain registrar service to register the chosen domain.

7.2.2 Domain Configuration

* **DNS Configuration**: Set up Domain Name System (DNS) records to point the domain to the IP address of the production server.
* **SSL Certificate**: If using HTTPS, configure and install an SSL certificate for the domain.

**7.3 Conclusion**

The deployment of the "Popcorn" Online Movie Ticket Booking System involves careful planning, configuration, and testing to ensure a smooth transition from the development environment to the live production server. Continuous monitoring and updates are essential for maintaining the system's performance and security.