https://github.com/MZULALI/CS250Group6

ENIGMA TICKET

Software Requirements Specification

Version 1.2

February 15, 2024

Group # 6

Aybars Seyrek https://github.com/AybarsSeyrek

Brayden Eleccion https://github.com/beleccion

Patrick Onyemah https://github.com/Patrickdaboss

Muhammad Zulali https://github.com/MZULALI

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph,D.
Spring 2024

Revision History

Date	Description	Author	Comments
<2/15/2024>	<Version 1>	<Group 6>	<First Revision>
<2/29/2024>	<Version 2>	<Group 6>	<Second Revision>
<3/14/2024>	<Version 3>	<Group 6>	<Third Revision>

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	<Group 6>	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirement Specification (SRS) document is to provide a comprehensive description of the functionalities, features, and operational constraints of the Enigma Ticket platform. This document is intended to serve as a foundational blueprint for the development team, outlining the technical and user experience requirements for building the platform. Additionally, it acts as a point of reference for stakeholders, including project managers, designers, developers, and quality assurance teams, to ensure alignment with project goals and user expectations. The SRS is also designed to facilitate clear communication and understanding among all parties involved, minimizing ambiguities and setting the stage for the successful development and deployment of Enigma Ticket.

1.2 Scope

The software product to be developed, referred to herein as Enigma Ticket, is an online movie ticketing platform designed to provide a seamless, intuitive, and accessible way for users to discover movies, view showtimes, and purchase tickets through a web browser.

Enigma Ticket will enable users to create personal accounts, search for movies using various filters, view detailed information about movies (including showtimes, theater locations, and seat availability), and securely purchase tickets. The platform will also feature a bot detection system to prevent bulk buying and scalping, offer promotional discounts, and integrate user reviews and ratings. Conversely, it will not support offline functionalities or the sale of merchandise unrelated to movie-going.

1.3 Definitions, Acronyms, and Abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

1-Definitions:

Ticket: A document or electronic record serving as proof of purchase to access the service.

User: Any individual interacting with the ticket selling app including buyers, sellers and administrators.

Event: A scheduled occasion for which tickets are sold.

Transaction: The process of buying or selling tickets within the app which includes payment and confirmation.

Inventory: The collection of available tickets for sale within the app.

Venue: The physical location where the event takes place.

Confirmation: Notification or acknowledgment of a successful transaction which was sent to the buyer and seller via document or electronic record.

Administrator: A user with special privileges responsible for managing the app which includes adding events, moderating listings, and resolving disputes.

Acronyms:

SRS: Software Requirements Specification
API: Application Programming Interface
UI: User Interface
UX: User Experience
DB: Database
MVP: Minimum Viable Product
FAQ: Frequently Asked Questions
CRM: Customer Relationship Management
HTTPS: Hypertext Transfer Protocol Secure

Abbreviations:

App: Application
Admin: Administrator
FAQ: Frequently Asked Questions
GA: General Admission
ETA: Estimated Time of Arrival
VAT: Value Added Tax

1.4 References

This section lists all documents and resources referenced throughout the Enigma Ticket SRS. Each reference includes the document's title, report number (if applicable), publication date, and publishing organization. For detailed access information, refer to Appendix A or the specified document.

1. **IEEE Guide to Software Requirements Specifications** (Std 830-1998)
   Title: IEEE Recommended Practice for Software Requirements Specifications
   Report Number: IEEE Std 830-1998
   Date: 1998
   Publishing Organization: Institute of Electrical and Electronics Engineers (IEEE)
   Source: IEEE Xplore Digital Library

1.5 Overview

(1)Enigma Ticket aims to set a new standard for online movie ticketing systems. With a focus on user experience, the system will incorporate advanced features such as personalized movie recommendations, social sharing capabilities, and interactive seat maps. The platform will be designed to support high traffic volumes, ensuring reliability and speed during peak times. Through Enigma Ticket, the company will not only streamline the ticket purchasing process but also enhance the overall movie-going experience, fostering customer loyalty and driving revenue growth.

(2)This introduction sets the stage for the detailed software requirements and specifications that will follow, providing a blueprint for the development of Enigma Ticket. The subsequent

sections will delve into specific requirements, including system features, functional and non-functional requirements, and the system architecture needed to realize this vision.

2. General Description

Enigma Ticket is designed as a standalone, comprehensive ticketing solution that integrates seamlessly with existing movie theater systems. Utilizing cloud-based technology, it ensures real-time synchronization of showtimes, seating availability, and ticket sales across multiple platforms. Its architecture is built for scalability, allowing for the accommodation of both small independent cinemas and large multiplex chains. The system incorporates data encryption and secure payment gateways to protect user information and transactions, setting a new standard for safety and reliability in the movie ticketing industry.

2.1 Product Perspective

Enigma Ticket is positioned uniquely in the market by offering a blend of features that are specifically tailored to enhance the movie-going experience. Unlike traditional ticketing platforms, which primarily focus on the transactional aspect of ticket sales, Enigma Ticket integrates additional functionalities such as real-time showtime updates, bot detection mechanisms to prevent bulk buying and scalping, and user reviews and ratings aggregation from reputable online sources..

2.2 Product Functions

This product is designed to revolutionize the movie-going experience with its array of streamlined and user-centric functions. At its core, the platform offers robust user account management, enabling personalized movie recommendations and seamless account handling. Users can easily browse and discover movies due to comprehensive listings, advanced search capabilities, and intuitive filtering options. The ticket reservation and purchasing process is straightforward, supporting a variety of payment methods and adhering to purchase limits to prevent bulk buying. Security is paramount, with sophisticated bot detection and prevention measures in place to ensure a fair purchasing environment.

2.3 User Characteristics

The product platform is designed for a broad spectrum of users, ranging from tech-savvy movie enthusiasts looking for a quick and easy way to purchase tickets, to less technologically inclined individuals who require a simple, straightforward user interface. The platform must also cater to security-conscious users, demanding high standards of data protection for their personal and financial information. Accessibility is a key concern, ensuring the system is navigable and usable for individuals with disabilities, making use of assistive technologies.

2.4 General Constraints

The development of the Enigma Ticket platform is influenced by a range of general constraints that limit design and implementation choices, including technological limitations requiring compatibility across all major web browsers, stringent adherence to security and data protection standards like GDPR and PCI DSS, and compliance with accessibility guidelines such as WCAG. Integration with third-party services necessitates flexibility in design to accommodate various APIs, while scalability and performance requirements dictate the architecture and hosting strategy.

2.5 Assumptions and Dependencies

(1) It is assumed that users will have access to an internet connection and an email account for registration.
(2) The system's performance is dependent on the integration with third-party services for payment processing and social media sharing.

3. Specific Requirements.

3.1 External Interface Requirements

3.1.1 User Interfaces-

The user interface being used is a form based user interface. this is a type os web interface and it is best used for inputting information, creating profiles and filling out surveys

3.1.2 Hardware Interfaces-

Radio Frequency Identification readers, mobile phones and ticket printers are the most user friendly and efficient hardware interfaces to use

3.1.3 Software Interfaces-

The software system being used is linux. linux is being used because it is a web based operating system and it is notable for its security and open source nature

3.1.4 Communications Interfaces-

The form based interface will be used

3.2 Functional Requirements

This section outlines the specific functionalities of the Enigma Ticket software, detailing how each requirement contributes to the overall operation and user experience of the movie ticketing website. The requirements are presented with a focus on their implementation, including inputs, processing steps, outputs, and error handling mechanisms.

3.2.1 User Load Handlining

3.2.1.1 Introduction

The Enigma Ticket platform is designed to efficiently manage high traffic volumes, specifically to support at least 1000 users concurrently without degradation of performance or user experience. This capability is critical for handling peak times, such as when tickets for highly anticipated movie releases become available.

3.2.1.2 Inputs

- **User Requests**: Includes page loads, movie search queries, ticket booking requests, and payment processing.
- **System Monitoring** Data: Real-time data on server load, response times, and resource utilization.

3.2.1.3 Processing

- **Load Balancing**: Distributes incoming user requests evenly across multiple servers to prevent any single server from becoming a bottleneck.
- **Resource Allocation**: Dynamically allocates resources based on demand, ensuring that server capacity scales up or down as required.
- **Session Management**: Efficiently manages user sessions to ensure that data consistency is maintained even as users are distributed across servers.
- **Caching**: Frequently accessed data, such as movie showtimes and ticket availability, is cached to reduce database access times and improve response speed for user queries.
- **Database Optimization**: Utilizes optimized database queries and indexes to speed up data retrieval and handling, ensuring quick processing of ticket bookings and user searches.

3.2.1.4 Outputs

- **User Feedback**: Immediate confirmation of actions taken, such as successful ticket bookings or informative error messages if an action cannot be completed.
- **System Status Updates**: Real-time updates on system performance and capacity, accessible by administrators for monitoring and management purposes.

3.2.1.5 Error Handling

- **Overload Prevention**: In case of demand exceeding the system's capacity to handle 1000 concurrent users effectively, a queue system is introduced to manage access, ensuring users are gradually let into the system to prevent crashes or severe slowdowns.
- **Failover Mechanisms**: In the event of a server failure, traffic is automatically rerouted to standby servers to maintain system availability without impacting user experience.
- **User Notification**: Users are informed of any delays or issues with processing their requests, including estimated wait times if a queuing system is activated.

- **Error Logging and Analysis**: All errors and system anomalies are logged for post-event analysis to identify root causes and guide system improvements to prevent future occurrences.

3.2.2 Web Browser Compatibility

3.2.2.1 Introduction

The Enigma Ticket platform is designed to be accessible exclusively through web browsers, ensuring wide accessibility without the need for users to download and install a standalone application. This approach leverages the ubiquity of web browsers on various devices, including desktops, laptops, tablets, and smartphones, to provide a consistent and convenient user experience.

3.2.2.2 Inputs

- **User Device Information**: The type of device, operating system, and web browser version are identified to optimize the presentation and functionality of the website.
- **User Actions**: Actions performed by the user within the web browser, such as navigating through the website, selecting movie showtimes, and completing ticket purchases

3.2.2.3 Processing

- **Responsive Design**: The website employs a responsive design, automatically adjusting its layout and content to suit the screen size and resolution of the user's device.
- **Cross-Browser Compatibility**: Ensures that the website functions correctly across a wide range of web browsers and versions through thorough testing and code adjustments.
- **Client-Side Processing**: Utilizes client-side scripting to enable interactive features such as form validation and dynamic content updates without needing to reload the page.
- **Server-Side Processing**: Backend servers handle data processing, such as database interactions for retrieving movie showtimes and processing ticket transactions, ensuring that sensitive operations are securely managed.

3.2.2.4 Outputs

- **Dynamic Web Pages**: Users are presented with web pages that dynamically update based on their interactions, providing a smooth and interactive experience.
- **Transaction Confirmations**: Upon completing a ticket purchase, users receive immediate confirmation directly within the web browser, including details of their purchase and a digital ticket when applicable.

3.2.2.5 Error Handling

- **Compatibility Notices**: Users attempting to access the website with an unsupported browser or disabled JavaScript are presented with notices advising them of the compatibility issue and suggesting solutions.
- **Graceful Degradation**: For features that may not be supported in older browsers, the website provides fallback options or degrades gracefully, ensuring that core functionalities remain accessible.
- **Input Validation Feedback**: The website provides real-time feedback on user input, highlighting errors (e.g., invalid payment information) within the web browser to prevent submission of incorrect data.
- **Session Timeouts**: Handles session timeouts with clear notifications to the user, allowing them to safely and securely log back in to continue their session without losing progress.

.

3.3 Use Cases

3.3.1 Manage User Account

3.3.1.1 Register/Create Account

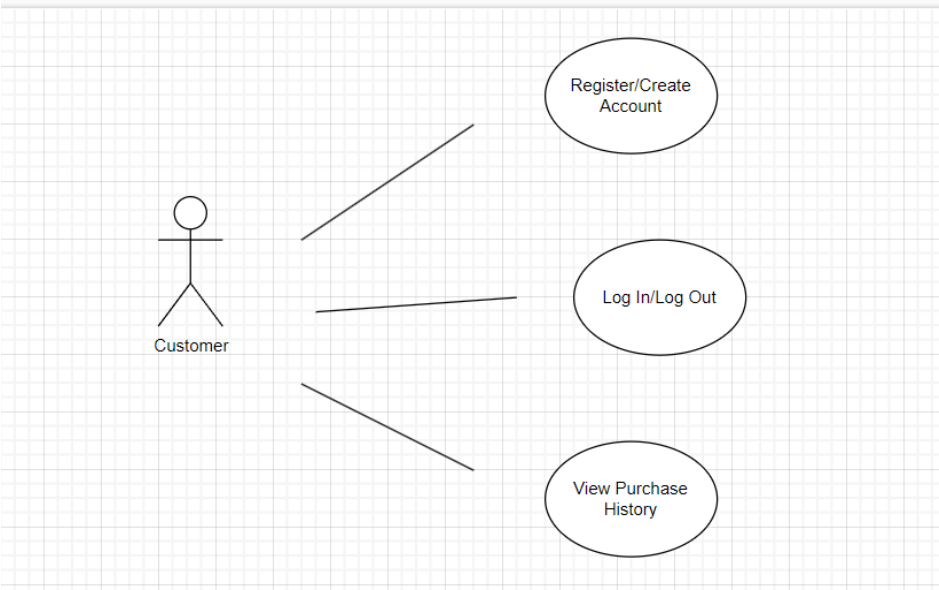
- Actor: New Customer
- Flow Of Events: Users can create a new account by providing required information. If information is valid, the system will create a new account
- Entry Conditions: User does not have an existing account

3.3.1.2 Log in/out

- Actor: Customer
- Flow of Events: User can log into their account if they provide the correct required credentials or they can log out.
- Entry Conditions: User has an existing account to log in to, or user is already logged in to log out.

3.3.1.3 View Purchase History

- Actor: Customer
- Flow Of Events: User navigates into their account to check their purchase history. This section lists out their tickets they have purchased. This will display information such as movie title, seat number, show time, and purchase date.



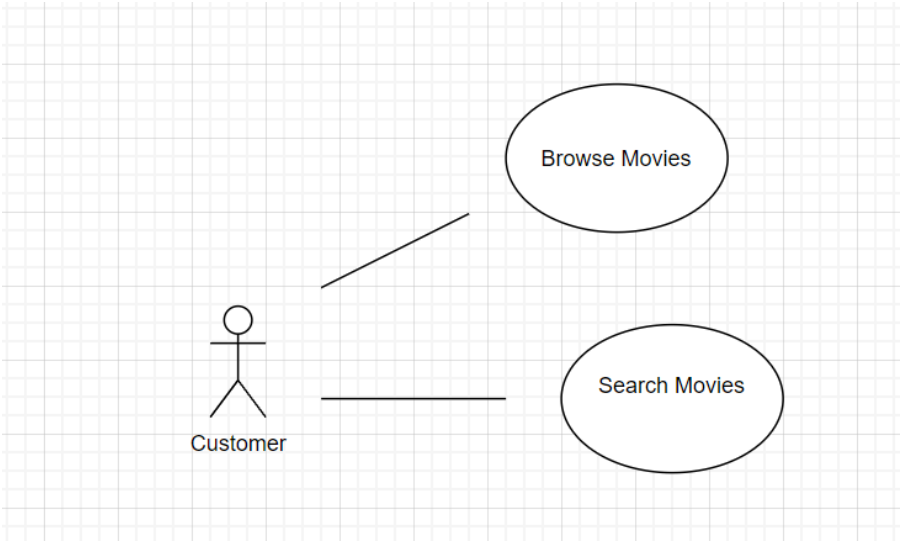
3.3.2 Browse Movie/ Search

3.3.2.1 Browse Movies

- Actor: Customer
- Flow Of Events: User views information about movies, including summaries, actors, reviews, and ratings
- Entry Conditions: User has selected a Movie

3.3.2.2 Search for Movies

- Actor: Customer
- Flow Of Events: Users can search for movies based on various criteria, such as genre or ratings.



3.3.3 Purchase Tickets

3.3.3.1 Select Movie and Showtime

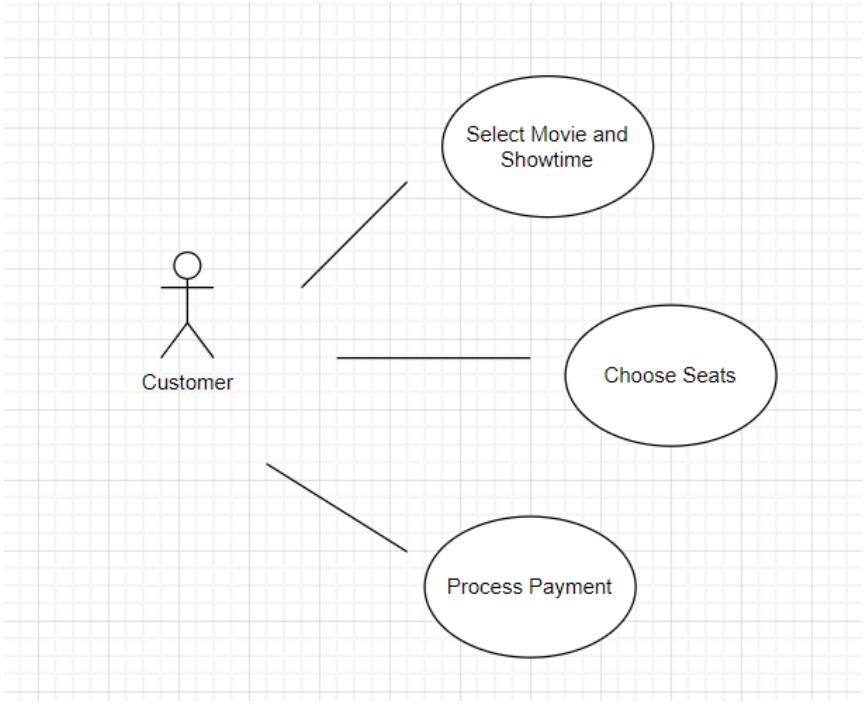
- Actor: Customer
- Flow of Events: User selects a movie and a showtime.
- Entry Condition: None

3.3.3.2 Choose Seats

- Actor Customer: User chooses their seats from the available selection.
- Entry Condition: User has selected a movie and a showtime.

3.3.3.3 Process Payment

- Actor: Customer
- Flow of Events: User enters their required payment details to purchase a ticket. Virtual Ticket is then added to the customer's account.
- Entry Condition: User has selected their preferred seating.



…

3.4 Classes / Objects

3.4.1 <User>

3.4.1.1 Attributes

- userID
- Username
- Password
- Email
- PurchaseHistory

3.4.1.2 Functions

- Register()
- Login()
- ViewPurchaseHistory()

3.4.2 <Movie>

3.4.2.1 Attributes

- Title
- Director
- ShowTime
- Genre

3.4.2.2 Functions

- addMovie()
- listShowTime()

3.4.3 <Ticket>

3.4.3.1 Attributes

- TicketID
- Showtime
- userID
- SeatNumber
- Price

3.4.3.2 Functions

- Purchase()

3.5 Non-Functional Requirements

Non-functional requirements for a ticketing website may encompass various attributes that contribute to the overall performance, reliability, security and usability of the system.

3.5.1 Performance

-Response Time: 90% of webpage loads shall complete within 2 seconds.

-Transaction Processing Time: 90% of ticket transactions shall be processed in less than 3 seconds.
-Scalability: The system shall support up to 10,000 concurrent users during peak hours without significant degradation in performance.

-Throughput: The system shall handle a minimum of 1000 ticket transactions per minute.

3.5.2 Reliability

-System Uptime: The system shall have a downtime of no more than 30 seconds per day for maintenance purposes.
-Mean Time Between Failures (MTBF): The system shall have an MTBF value of greater than 30 days.

-Fault Tolerance: The system shall continue to operate with minimal disruption in the event of hardware or software failures.

3.5.3 Availability

-only one device per user at a time
-it needs to handle at least 1000 people at once
-it needs to have language support for English, spanish and swedish
-

3.5.4 Security

-Data Encryption: All sensitive data, including user credentials and payment information, shall be encrypted using industry-standard encryption algorithms.
-Access Control: Only authorized personnel shall have access to administrative functions.
-Compliance: The system shall comply with relevant data protection regulations such as GDPR or HIPAA.

3.5.5 Maintainability

-Code Maintainability: The codebase shall follow best practices and coding standards to facilitate ease of maintenance and future enhancements.
-Documentation: Comprehensive documentation shall be provided for system architecture, APIs, and codebase to aid in maintenance and troubleshooting.

3.5.6 Portability

-The website shall be compatible with multiple operating systems, including Windows, macOS, and Linux.
-The website shall be deployable on various web server platforms such as Apache, Nginx, and Microsoft IIS
.-The system shall support multiple database management systems (DBMS) such as MySQL, PostgreSQL, and MongoDB.
-The codebase shall be written using cross-platform programming languages and frameworks such as Python, Node.js, or Java.
-The website shall support multiple languages and locales, allowing users to interact with the system in their preferred language.

3.6 Inverse Requirements

Inverse requirements are specifications that describe what a system should not do or features it should not have. These are as crucial as regular requirements because they help to refine the scope of the project and avoid unnecessary functionalities or behaviors that might hinder the user experience or system performance.

-No Intrusive Ads: The website shall not include pop-up ads or any other form of intrusive advertising that disrupts the user experience.
-No Compromising User Security for Convenience: The website shall not compromise user security for the sake of convenience, such as storing sensitive information in plaintext or using weak encryption methods.

-No Data Retention Beyond Legal Requirements: The website shall not retain user data beyond the period required by law or specified in the website's privacy policy.

3.7 Design Constraints.

Design constraints are limitations or restrictions that influence the design and development of the software project. These constraints may arise from various sources such as industry standards, company policies, hardware limitations or external dependencies.

1. **Industry Standards and Regulations:**

Compliance with Payment Card Industry Data Security Standard requirements for handling payment information securely.

Adherence to General Data Protection Regulation or other data protection laws to ensure user privacy and data security.

2. **Company Policies and Guidelines:**

Adherence to company branding guidelines for consistent visual design and user experience across the website.

Integration with existing authentication and authorization systems as per company security policies.

3. **Hardware and Infrastructure Limitations:**

Compatibility with existing hardware infrastructure, ensuring that the software can run efficiently within the available hardware resources.

Compatibility with legacy systems or technologies that are still in use within the organization.

4. **Technology Stack Constraints:**

Utilization of specific programming languages, frameworks, or libraries mandated by the organization's technology standards.

Integration with third-party APIs or services for functionalities such as payment processing, email notifications, or geolocation services.

5. **Performance and Scalability Requirements:**

Consideration of performance limitations imposed by the hosting environment, such as bandwidth restrictions or server capacity.

Scalability constraints, including limitations on the maximum number of concurrent users or transactions that the system must support.

6. **User Accessibility Requirements:**

Compliance with accessibility standards such as Web Content Accessibility Guidelines to ensure the website is usable by individuals with disabilities.

Compatibility with assistive technologies such as screen readers or voice recognition software.

7. **Budget and Resource Constraints:**

Adherence to budgetary constraints for development, deployment, and maintenance of the software project.
Availability of human resources, including developers, testers and support personnel, may impact project timelines and deliverables.

8. **Legal and Regulatory Constraints:**

Compliance with intellectual property laws, licensing agreements, and copyright restrictions for third-party assets used in the software project.
Restrictions on the use of certain technologies or algorithms due to patents or legal agreements.

3.8 Logical Database Requirements

1. **Data Models**: Utilize relational database models for user data, ticket transactions, and movie information to ensure relationships are well-defined and data integrity is maintained.
2. **Data Formats**: Standardize data formats for dates, times, currency, and personal information to ensure consistency across the database. Use UTF-8 encoding to support internationalization.
3. **Storage Capabilities**: The database must support scalable storage solutions to accommodate growth in data volume from user registrations, ticket sales, and movie listings.
4. **Data Retention**: Implement data retention policies that comply with legal requirements and best practices for user data and transaction history. This includes purging obsolete data and archiving important transactional information for the required period.
5. **Data Integrity**: Enforce data integrity through the use of primary keys, foreign keys, and constraints to prevent data duplication and ensure accuracy across related tables.
6. **Security**: Secure stored data using encryption at rest and access controls to protect sensitive information such as user credentials and payment information.
7. **Backup and Recovery**: Regularly backup the database to prevent data loss and ensure a recovery plan is in place for restoring data in case of a system failure.

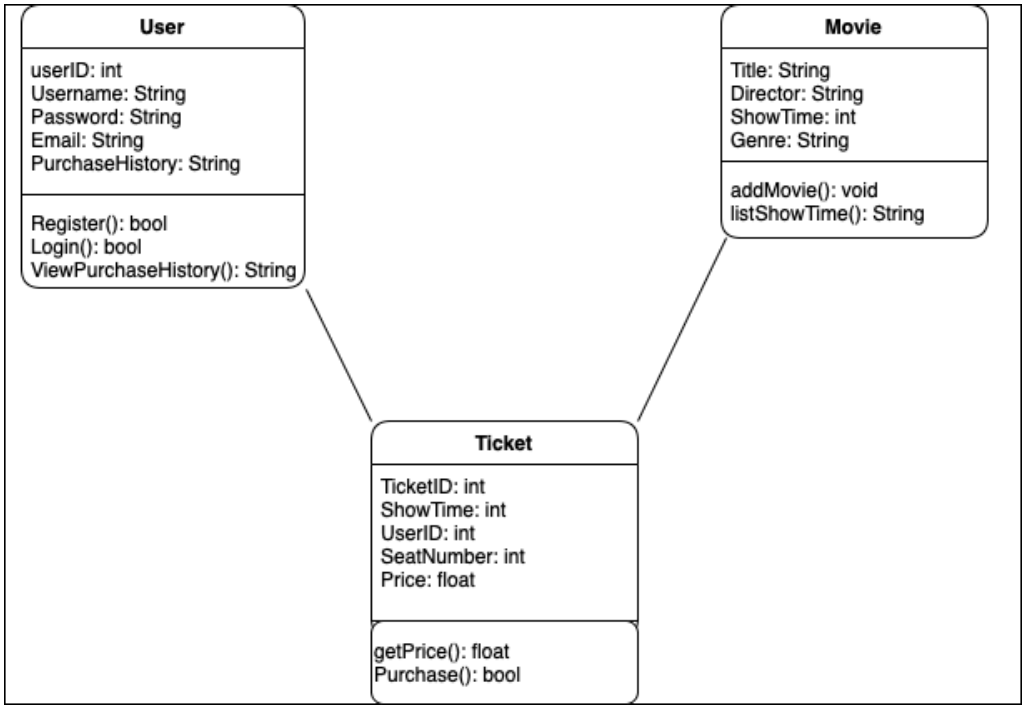
3.9 Other Requirements

1. **Interoperability**: The system must ensure compatibility and seamless integration with external systems and APIs, such as payment gateways, social media platforms for login and sharing, and movie databases for fetching movie details.
2. **Localization and Internationalization**: The platform should be designed to support multiple languages and regions, enabling easy localization of content, currency, and date formats to enhance user experience in different geographical markets.
3. **Regulatory Compliance**: The platform must comply with all relevant local, national, and international laws and regulations affecting online ticket sales, digital commerce, and user data protection. This includes compliance with copyright laws for movie titles and descriptions, as well as tax regulations related to online sales.

4. **Accessibility**: The platform should adhere to best practices and standards for digital accessibility, ensuring that all users, including those with disabilities, can navigate, understand, and interact with the website effectively.

4. Software Design Specification

4.1 UML Diagram



4.1.1 UML Diagram Description

4.1.1.1 User Class

- The user class represents the user holds useful information in the attributes that identifies the user.
- Attributes:
    - username: holds the user's sign in username as a string
    - name: holds the actual name of the user
    - password: holds the user's password used to authenticate sign in
    - emailAddress: hold's user's email address as a String
    - userID: holds the ID of the account as an int
    - purchaseHistory

- User Class Functions:

- Register() Function allows a new user to create a new account, must have a unique username and unique email address
- Login() Function allows a returning user with existing credentials to log in if username and password match
- ViewPurchaseHistory() allows a user to view purchases of tickets

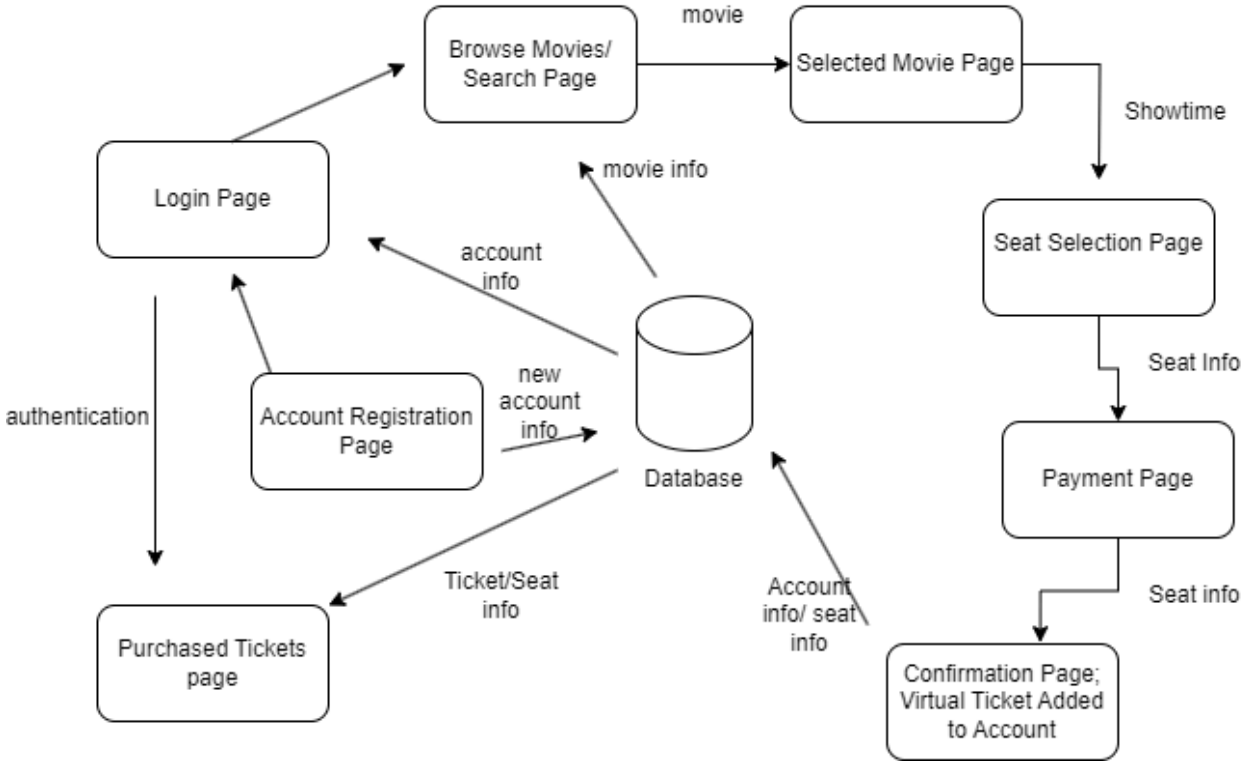
4.1.1.2 Ticket Class

- The Ticket class acts as a blueprint to create ticket objects that hold important information for ticket details. IT also holds the purchase function
- Attributes:
  - TicketID: Unique identifier for the ticket.
  - Showtime: DateTime indicating when the show is scheduled.
  - userID: Reference to the user who owns the ticket.
  - SeatNumber: Designation of the seat assigned to the ticket.
  - Price: Cost of the ticket.
- Functions:
  - Purchase(): Initiates the ticket buying process.

4.1.1.5 Movie Class

- The Movie Class gives holds relevant information to create a movie object.
- Attributes:
  - Title: The name of the movie. String datatype
  - Director: The person who directs the film. String datatype
  - ShowTime: Schedule times when the movie will be shown. int datatype
  - Genre: Category of the movie (e.g., action, comedy). String datatype
  -
- Functions:
  - addMovie(): Registers a new movie to the system.
  - listShowTime(): Displays the showtimes for movies.

4.2 SWA Diagram



4.2.1 SWA Diagram Description

4.2.1.1 Login Page

- Users start their interaction here. If they are returning users, they will enter their credentials to authenticate and proceed to the Browse/Search Page.
- For new users, there is a link to the Account Registration Page where they can create a new account.

4.2.1.2 Account Registration Page

- Here, new users submit their account information, which is then stored in the Database. After registration, they proceed to the Login Page for authentication.

4.2.1.3 Authentication

- This is a process that verifies the user's credentials. If the authentication is successful, the user gains access to the Browse/Search Page. If not, they remain on the Login Page.

4.2.1.4 Browse Movies/Search Page

- Authenticated users arrive at this page to browse movies or perform searches. When they select a movie, they are taken to the Selected Movie Page.
- This page pulls movie information from the Database to display to the user.

4.2.1.5 Selected Movie Page

- Displays detailed information about the user's selected movie, including showtimes. When a showtime is selected, the user is taken to the Seat Selection Page.

4.2.1.6 Seat Selection Page

- Users can view and select available seats. Seat information is retrieved from and updated in the Database. Once a seat is selected, the user proceeds to the Payment Page.

4.2.1.7 Payment Page

- Users provide payment details to purchase the ticket. The seat information is used here to finalize the purchase. After payment is processed, the user is directed to the Confirmation Page.

4.2.1.8 Confirmation Page

- The user receives confirmation of their purchase, and a virtual ticket is added to their account. This information, along with the seat info, is stored in the Database.

4.2.1.9 Purchased Tickets Page

- This page allows users to view their purchased tickets. The information is retrieved from the Database, which contains ticket/seat information linked to the user's account.

4.2.1.10Database

- Central to the workflow, the Database stores and manages all account info, movie info, ticket/seat info, and processes new account information. It interacts with nearly all components of the system, providing a persistent data storage solution.

4.3 Development Plan and Timeline

The project is scheduled for a 10-month development cycle, divided into four main phases

Planning and Design (Months 1-2): This phase focuses on finalizing the project scope, detailed SRS documentation, and initial design mock-ups.
Development (Months 3-6): The core development work, including front-end and back-end coding, database setup, and initial testing.
Testing and Refinement (Months 7-8): Dedicated to thorough testing, bug fixing, and system optimizations to ensure reliability and performance.
Deployment and Launch (Month 9): Final preparations for going live, including final testing, deployment to production servers, and launch.
Post-Launch Support (Month 10): Initial post-launch support period for monitoring system performance and addressing any emerging issues.

4.3.1 Partitioning of Tasks

Tasks are divided among specialized teams to leverage expertise and ensure focused progress

- **Design Team**: Responsible for UI/UX design, mock-ups, and design specifications.
- **Development Team**: Split into Front-end and Back-end sub-teams to handle respective components.
- **QA Team**: Conducts testing, including unit tests, integration tests, and system tests.
- **DevOps Team**: Manages deployment, continuous integration, and continuous delivery processes.

4.3.2 Team Member Responsibilities

- **Project Manager**: Oversees project progress, ensures milestones are met, and facilitates communication between teams.
- **Design Lead**: Coordinates the design team's efforts, ensuring designs meet user experience goals and technical requirements.
- **Front-end Lead**: Manages the front-end development team, ensuring the implementation matches design specs and SRS requirements.
- **Back-end Lead**: Oversees the back-end development team, responsible for database management, server-side logic, and API integration.
- **QA Lead**: Coordinates testing efforts to identify bugs, ensure functionality aligns with requirements, and maintain overall system quality.
- **DevOps Lead**: Ensures smooth deployment processes, manages server infrastructure, and implements efficient CI/CD pipelines.

5. Verification Test Plan

5.1 Scope and Objectives

The Verification Test Plan for the Enigma Ticket system aims to ensure that the developed software meets the requirements specified in this Software Requirements Specification (SRS) document. The primary objectives of the testing process are to verify the functionality, performance, security, and compatibility of all modules, including user registration, authentication, movie search, seat selection, payment processing, and purchase history. The test plan will validate that these modules work seamlessly together to provide a smooth and reliable user experience.

5.2 Testing Approach

The testing approach for the Enigma Ticket system will involve a combination of manual and automated testing techniques to comprehensively validate the system's functionality, usability, and reliability. The following testing levels will be employed:

5.2.1 Unit Testing

Unit testing will be conducted to verify individual components and modules in isolation, ensuring they perform as expected based on their specific requirements. Developers will be responsible for writing and executing unit tests for their respective modules.

5.2.2 Integration Testing

Integration testing will be performed to validate the interaction and data flow between different modules, ensuring they work together seamlessly. This testing level will focus on the integration points and the exchange of data between the user interface, backend services, and database.

5.2.3 System Testing

System testing will be carried out to assess the end-to-end functionality of the Enigma Ticket system, simulating real-world scenarios and user interactions. This testing level will validate the system's behavior as a whole, including user flows, error handling, and boundary conditions.

5.3 Test Environment

A dedicated test environment will be set up to closely mimic the production environment in terms of hardware, software, and network configurations. This test environment will be isolated from the development and production environments to avoid any unintended impacts. Test data will be maintained in a version-controlled manner, covering various scenarios, including valid and invalid inputs, edge cases, and boundary values.

5.4 Test Cases

Detailed test cases will be designed and documented based on the requirements specified in this SRS document, covering all the identified features and functionalities. Test cases will be prioritized based on their criticality, focusing on the core functionalities and high-risk areas. Each test case will include clear steps, expected results, and acceptance criteria to determine the success or failure of the test. Both positive and negative scenarios will be covered, including error handling and boundary conditions. Test cases will be organized into logical test suites based on their functional areas or modules to facilitate efficient test execution and tracking.

5.5 Test Execution

Test cases will be assigned to the designated testing team members, ensuring an even distribution of workload and expertise. The execution of test cases will follow the defined test plan, and the actual results will be documented, including any discrepancies between the expected and actual outcomes. Failed test cases will be retested after the identified issues are resolved to ensure the effectiveness of the fixes.

5.6 Defect Management

A defect tracking system will be established to record, prioritize, and manage identified issues throughout the testing process. Defects will be categorized based on their severity and impact on the system's functionality and user experience. The testing team will collaborate with the development team to investigate and resolve defects, ensuring timely communication and status updates. The resolution of defects will be verified through retesting and regression testing to ensure the fixes do not introduce new issues.

5.7 Test Reporting

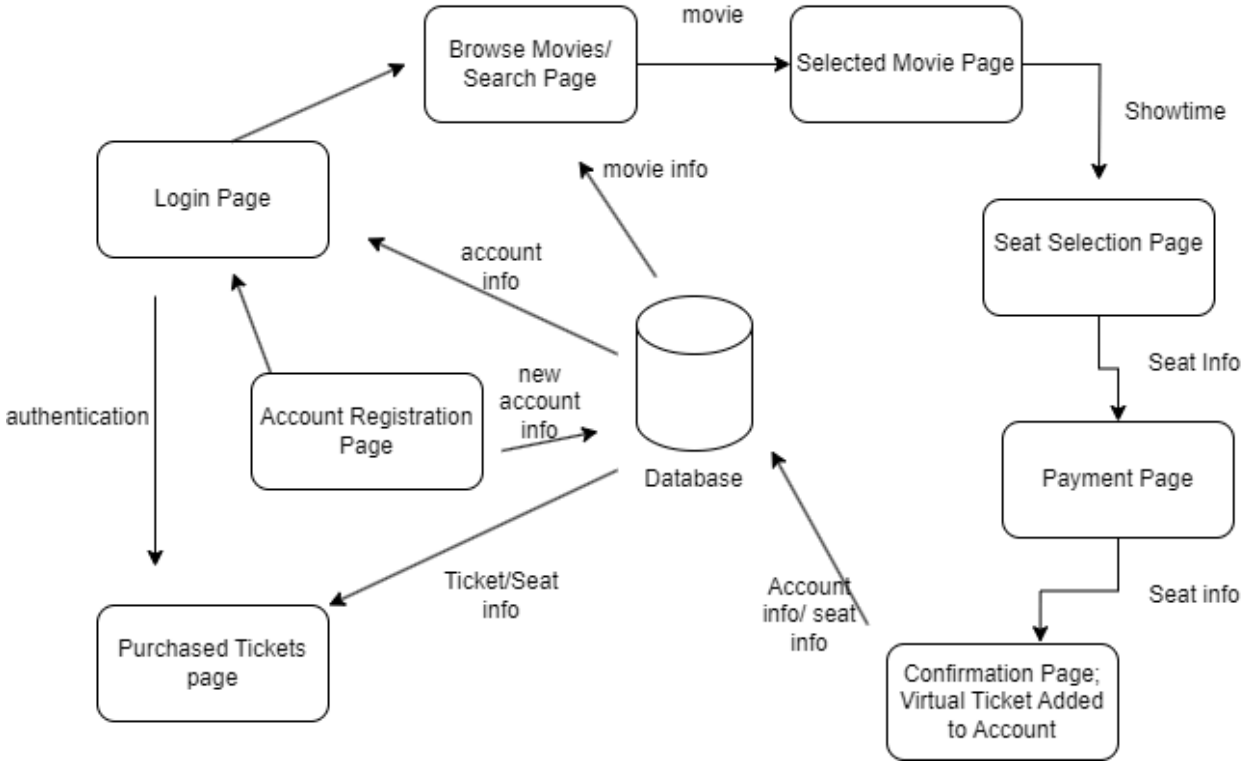
Comprehensive test coverage metrics will be maintained, including the number of test cases executed, passed, failed, and blocked. Regular status reports will be provided to customers, highlighting the progress of testing activities, identified issues, and key metrics. Detailed test reports will be generated upon the completion of each testing phase, summarizing the overall results, defect trends, and recommendations for improvement. Analysis will be conducted to identify lessons learned and suggest process enhancements for future testing efforts.

6. Software Architecture and Data Management

The Enigma Ticket system follows a three-tier architecture, with the Presentation Tier handling user interactions, the Application Tier processing business logic, and the Data Tier storing and managing data using PostgreSQL, a robust and scalable relational database management system (RDBMS). The data management strategy involves a normalized schema split into four logical databases (User, Movie, Ticket, and Payment) to improve organization, security, and scalability. Security measures such as password hashing, secure communication protocols, access control, and regular backups are implemented.

6.1 Software Architecture Diagram

[Architectural Diagram] –



This architectural diagram represents the most recent design of the Enigma Ticket system and reflects the data management strategy. The diagram illustrates the three-tier architecture, consisting of the Presentation Tier (Web Browser), Application Tier (Web Server and Application Server), and Data Tier (Database Server). The arrows indicate the flow of data and communication between the tiers.

6.2 Data Management Strategy

6.2.1 Database Choice

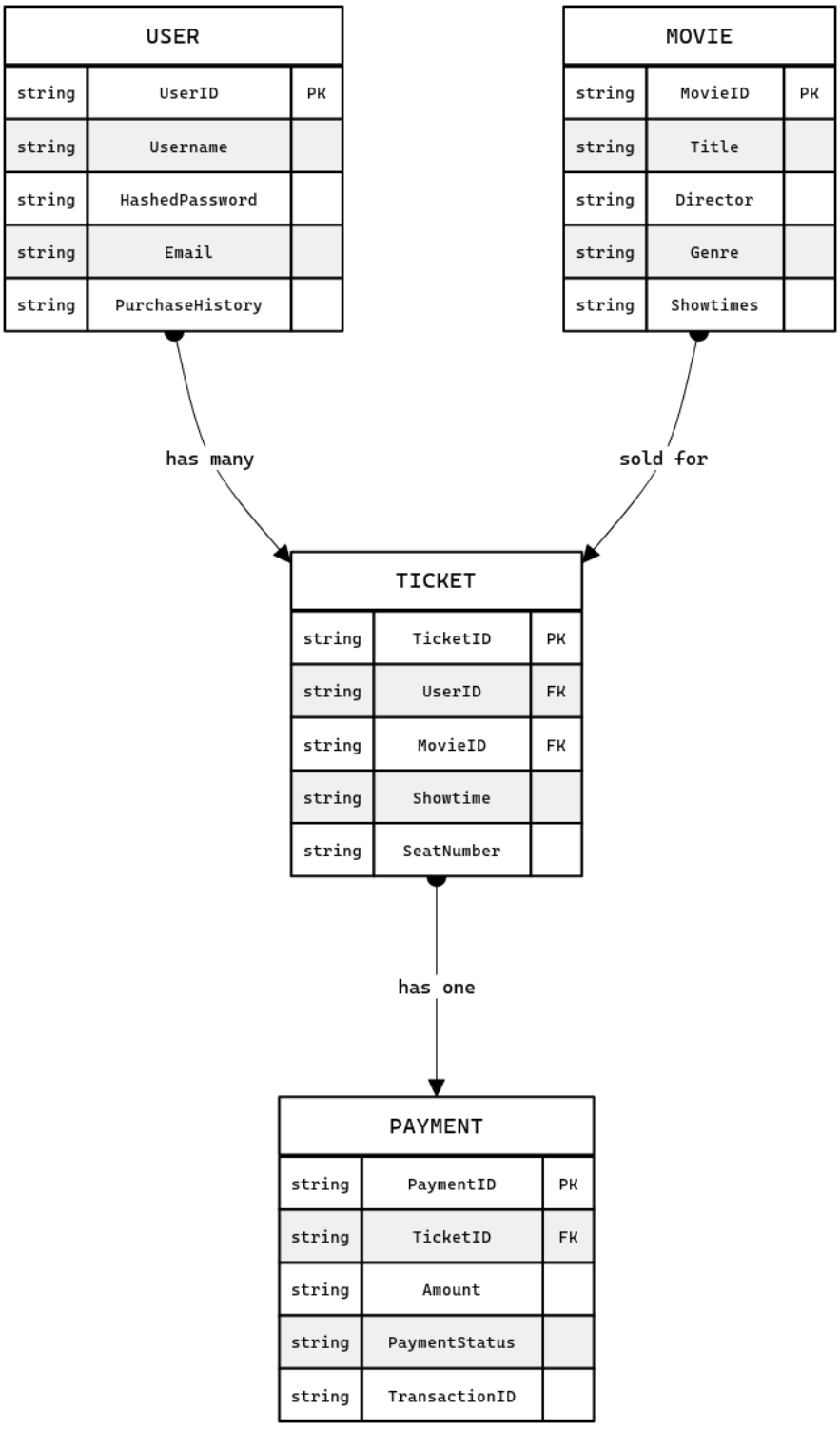
The Enigma Ticket system will utilize a relational database management system (RDBMS), specifically PostgreSQL, to store and manage its persistent and sensitive data. PostgreSQL was chosen for its robustness, scalability, and strong support for data integrity and security.

6.2.2 Database Design

The database will be designed using a normalized schema to ensure data consistency and minimize redundancy. The system will consist of four main databases:

- User Database: Stores user account information, including username, hashed password, email, and purchase history.
- Movie Database: Contains details about movies, such as title, director, genre, and showtimes.
- Ticket Database: Manages ticket reservations, linking users, movies, showtimes, and seat information.
- Payment Database: Records payment transactions, including user details, ticket information, and payment status.

[Entity-Relationship Diagram (ERD)] –



The ERD above illustrates the relationships between the entities in the databases. The User entity is connected to the Ticket entity, representing the tickets purchased by each user. The Movie entity is related to the Ticket entity, indicating the movie associated with each ticket. The

Payment entity is linked to the Ticket entity, tracking the payment details for each ticket purchase.

6.2.3 Data Security

To ensure the security of sensitive user and payment information, the following measures will be implemented:

- User passwords will be hashed and salted before storing them in the User Database.
- Secure communication protocols (HTTPS) will be used for all data transmission between the Application Tier and the Data Tier.
- Access to the databases will be restricted to authorized personnel only, with proper authentication and role-based access control.
- Regular database backups will be performed to prevent data loss and enable quick recovery in case of any failures.

6.2.4 Alternative Considerations

While the chosen RDBMS (PostgreSQL) and the four-database design provide a solid foundation for the Enigma Ticket system, alternative approaches were considered:

- Single Database: Instead of splitting the data into four separate databases, a single database could have been used to store all the information. However, separating the data into logical databases improves data organization, security, and scalability. It allows for better access control, backup strategies, and potential horizontal scaling of individual databases based on specific requirements.
- NoSQL Databases: NoSQL databases, such as MongoDB or Cassandra, were considered for their scalability and flexibility in handling unstructured data. However, given the structured nature of the ticket booking system and the need for strong data consistency and integrity, an RDBMS was deemed more suitable. RDBMSs provide ACID (Atomicity, Consistency, Isolation, Durability) properties, which are crucial for ensuring data reliability and transactional integrity.