Theatre Ticketing System

Software Requirements Specification

Version 4.0

03/27/2024

Group #18

Aryan Arora

Hunter Thomas

Krish Lodha

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Spring 2023

Revision History

Date	Description	Author	Comments
03/27/2024	Version 4.0	Aryan Arora	Fourth Revision
03/27/2024	Version 4.0	Hunter Thomas	Fourth Revision
03/27/2024	Version 4.0	Krish Lodha	Fourth Revision

Document Approval

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

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

Table of Contents

1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. The design of this document is to analyze and collect and report on the complete Theatre Ticketing System by stating the problem in detail. It will also define the high level features provided by the software. The specific requirements of the Theatre Ticketing System are defined in this document.

1.1 Purpose

The purpose of this document is to gather and design the system based on the requirements given by the consumers. We will also predict how this product should be used depending on the situation. This will allow us to get a better understanding of the needs and requirements of the project. We can outline concepts that will be added later, along with ideas for the project that will be used or discarded.

The purpose of this SRS document is to deliver an overview of the software we will develop. This document will highlight the project's main target audience, its user interface and the hardware needed along with the software parameter. It will also show the way our client, team and target audience see this project and the possible functions. In conclusion it will help any designers and developers to have the ability to assist in the development of the life cycle

1.2 Scope

The software will be able to allow the user to register for a new account. They will also be able to allow users to purchase tickets. The website will be compatible with operating systems such as windows and linux to ensure safety and diversity of the user environment. the software will

The benefits of this software include bot blocking, concurrent user administration, and provides administrative support. Our main objective is to create a website that is secure and meets all specified requirements The goal is to complete the prototype of this project within five months and using less that five hundred thousand that allows for convenient and easy purchasing of movie tickets. This SRS is aimed towards satisfying specific requirements that will improve the reliability and improve user experience.

1.3 Definitions, Acronyms, and Abbreviations.

MAC	*Macintosh*

MTBF	*Mean Time Between Failures*
SRS	*Software Requirements Specification*

1.4 References

-IEEE Recommended Practice for Software Requirements Specifications, 1998 Software Engineering Standards Committee
-How to Write a Software Requirements Specification (SRS Document) 2023
-E-Store Project Software Requirements Specification Version <4.0> 4/15/2007
-Your 2024 Guide to Writing a Software Requirements Specification – SRS Document 2024

1.5 Overview

This document's remaining sections include a general description describing the user characteristics and product functions. The general description of this project is in section 2 of this document. Section 3 includes the specific requirements for the system to function such as the user interface and hardware interfaces. Section 4 includes models and diagrams to support the information provided in this document.

2. General Description

This section provides an overview of the variables that affect the product and its requirements. It gives an overview of the product perspective, functions, user characteristics, constraints, and assumptions and dependencies.

2.1 Product Perspective

The Theater Ticketing System was designed to make the process of purchasing tickets for a variety of plays in San Diego more convenient and expedient.

2.2 Product Functions

Concurrent user administration, bot blocking, database interface, and discount support are all essential features. It also sets the purchase limit, determines showtime availability, and provides administrative support.
The system also allows individuals to save money, access a single database, and make the experience safe and simple.

2.3 User Characteristics

The majority of users of the system are individuals who purchase tickets online.

2.4 General Constraints

General constraints include preventing scalping and reselling, ensuring that each ticket is unique, making it simple to sell digital tickets, a $500,000 budget, support for up to 20 San Diego-area venues, only 150 regular seats and 75 deluxe seats available, the ability to purchase 20 tickets at once, only 6-8 shows available per day per movie, the ability to book tickets two weeks in advance, and the ability for customers to buy tickets 10 minutes after the movie begins.

2.5 Assumptions and Dependencies

Customers are anticipated to be able to create accounts, select seats, and use tickets just for San Diego movies. They must also have a wifi connection and basic experience of using a computer browser to purchase online tickets.
Customers can only receive refunds in person. The system's success is dependent on the availability and accuracy of movie-related data, such as showtimes, reviews, and critic quotes from third-party websites such as Rotten Tomatoes and IMDb.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The software will have both, command line and also the graphical user interface to enhance the users preferences. This will allow the user to interact through text based commands or visual elements.

3.1.2 Hardware Interfaces

The System will support browsers on mobile phones, tablets, PC’s, laptops, MACs to ensure that it is accessible to all.

3.1.3 Software Interfaces

The website will be compatible with operating systems like windows and linux to ensure a diverse user environment.

3.1.4 Communications Interfaces

It is also designed in such a way that the interface will include a number of communication interfaces like printers, fax machines which will be connected to the Wi-Fi to ensure network communications.

3.2 Functional Requirements

3.2.1 Scalability_of_the_system

3.2.1.1 Introduction: The system will support simultaneous access by at least 1000 users.

3.2.1.2 Inputs: Users can only use the website to access their accounts to purchase tickets as there is no application that they can download.
3.2.1.3 Processing: The system will process all the requests sent/received by the user through web technologies only.
3.2.1.4 Outputs: All the data will be shown through web pages.
3.2.1.5 Error Handling: The load on the system will be balanced and error messages will be shown to users during peak times.

3.2.2 User_interface

3.2.2.1 Introduction: The system will be accessible through a web browser only.
3.2.2.2 Inputs: Users can send/receive information through a web browser.
3.2.2.3 Processing: The system will process all the requests sent/received by the user through web technologies only.
3.2.2.4 Outputs: All the data will be shown through web pages.
3.2.2.5 Error Handling:Compatibility and alternative options for unsupported web browsers will be handled and the user will be then notified.

3.2.3 Prevention_of_bots

3.2.3.1 Introduction: Preventing bots from purchasing many tickets at once.
3.2.3.2 Inputs: Ticket purchasing requests.
3.2.3.3 Processing: A simple CAPTCHA or similar techniques will be used to separate humans from bots.
3.2.3.4 Outputs: Tickets will be given to verified humans.
3.2.3.5 Error Handling: There will be a limit on the number of times a CAPTCHA can be tried and also if it exceeds then that user will be locked out for some time.

3.2.4 Database_integration

3.2.4.1 Introduction: Web page with the showtimes and the seats that are available.
3.2.4.2 Inputs: Questions about the showtimes and the tickets.
3.2.4.3 Processing: Get and update the data from and to the database.
3.2.4.4 Outputs: The availability of show timings and the tickets will be shown to the users.
3.2.4.5 Error Handling: The handling of the errors of the database will be done.

3.2.5 Limit_of_purchasing_tickets

3.2.5.1 Introduction: There will be rules when a ticket or many tickets are purchased.
3.2.5.2 Inputs: Ticket Selection.
3.2.5.3 Processing: Verify purchase rules(20 Tickets max and availability window)
3.2.5.4 Outputs: Confirmation/Rejection of the purchase made by the user.
3.2.5.5 Error Handling: The system will notify the user of any violations or restrictions.

3.2.6 Administrator_mode

3.2.6.1 Introduction: A mode for the administrator of the system.
3.2.6.2 Inputs: Administrator’s login credentials will be asked.

3.2.6.3 Processing: Access to the System settings, user management and controls of the system will be provided.
3.2.6.4 Outputs: Controls and dashboard for the administrator.
3.2.6.5 Error Handling: An authentication and authorization issue will be handled if prompted.

3.2.7 User_feedback_system

3.2.7.1 Introduction: Collecting and managing the users feedback for the website.
3.2.7.2 Inputs: Feedbacks collected from users.
3.2.7.3 Processing: Store and carefully categorize the data received from the users.
3.2.7.4 Outputs: Feedback reports will be prepared and will be implemented.
3.2.7.5 Error Handling: The feedback will be validated.

3.2.8 Managing_the_discount

3.2.8.1 Introduction: Discounted tickets will be provided for some days or some people like students,military personnel, veterans.
3.2.8.2 Inputs: Discount selections during the purchase of the tickets.
3.2.8.3 Processing: Applying the discounts and also calculating the final price after the discount.
3.2.8.4 Outputs: The total price of the ticket/tickets will be provided.
3.2.8.5 Error Handling: If a certain coupon is expired or does not exist, the user will be prompted with an error message.

3.2.9 Collection_of_reviews_for_movies

3.2.9.1 Introduction: Collect different reviews from many online and reputable websites.
3.2.9.2 Inputs: The reviews will be automated in such a way that it will automatically be taken from websites.
3.2..3 Processing: The new data will be processed.
3.2.9.4 Outputs: The reviews will be displayed on the respectable movies.
3.2.9.5 Error Handling: All the changes on the source sites will be handled.

3.3 Use Cases

3.3.1 Use Case #1: New User Registration

- Objective: To allow users to register for a new account.
- Actors: Newly registered Users.
- Flow of Events: The users enter their information for their account(name,email and password). The system then validates their information and makes a new account for them to login into. After this, the system sends them a confirmation email to validate their information.
- Exception: If the information is invalid, the login fails and they have to try again.

3.3.2 Use Case #2: Purchasing of tickets

- Objective: The user purchases tickets on the website.

- Actors: Registered Users
- Flow of Events: The user logs in the website with their credentials and purchases tickets. The user enters their payment information on the website, and buys the tickets. The payment is then processed and issues the tickets if the payment information is valid. The user is sent an email/text with the details of the movie and the information of the seat.
- Exception: If the payment fails then the user is notified to try again or to cancel the transaction.

3.3.3 Use Case #3: Admin Login

- Objective: Allows the workers to login and access the the full system
- Actors: Workers
- Flow of Events: The workers login to the website with their worker credentials. They fix problems that crop up throughout the use of the system. They perform maintenance and upkeep of the system.
- Exception: If incorrect worker credentials are inserted it denies entry. Outdated credentials will be denied and notifies them that their credentials are incorrect.

3.4 Classes / Objects

3.4.1 Class: User_Account

3.4.1.1 Attributes

- UserID: A unique identifier for every user.
- Password: A password that the user ID sets.
- Name: The full name of the user.
- Email: The email ID of the user.
- Payment: Saved Payment information of the user.

3.4.1.2 Functions

- Register: This allows the user to register for new accounts.
- Login: Allows a user to get in their account.
- Forgot Password: This allows the user to reset their password using their email.

3.4.2 Class: Ticket

3.4.2.1: Attributes

- UserID: A unique identifier for the person who owns the ticket.
- TicketID: A unique identifier of the ticket.
- EventID: A unique identifier for the movie that is to be watched by the user.
- Price: The price of the ticket that is purchased by the user.

3.4.2.2: Functions

- Puchase_tickets: It allows the user to purchase tickets and also the seats(Regular/deluxe).
- Cancel: This allows the user to cancel their reservation and check how much refund they will receive.

3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

3.5.1 Performance: The system will process 95% of the transaction under 1 second which will ensure a smooth transition for all the users.

3.5.2 Reliability: A 30 day MTBF(mean time between failures) is expected, it makes the product more reliable as the website does not crash regularly.

3.5.3 Availability: Due to the high usage the system cannot run all day, so a 1 minute per day downtime is expected which provides access to all the users.

3.5.4 Security: A lot of safety and security measures will be taken to prevent cybersecurity attacks, bot buyers and various data breaches as the users privacy comes first. It will be up to the industry standards.

3.5.5 Maintainability: The system will have scheduled maintenance with a clear plan so it doesn't affect the users in a negative way.

3.5.6 Portability: The website will be portable as it will be available on windows, MACs, Linux and more to make sure that the users do not face any difficulties without exponential modifications to the system.

3.6 Inverse Requirements

- *The system will allow the users to buy tickets upto 10 minutes after the movie has started.*
- *The system will encrypt all the sensitive data of the users.*
- *The software will not be only on one platform so the users have no restrictions.*
- *Almost all the things like updates, backups and monitoring the system will be automated to ensure any human involvement is kept to the bare minimum.*

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitations, etc. that will impact this software project.

3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

3.9 Other Requirements

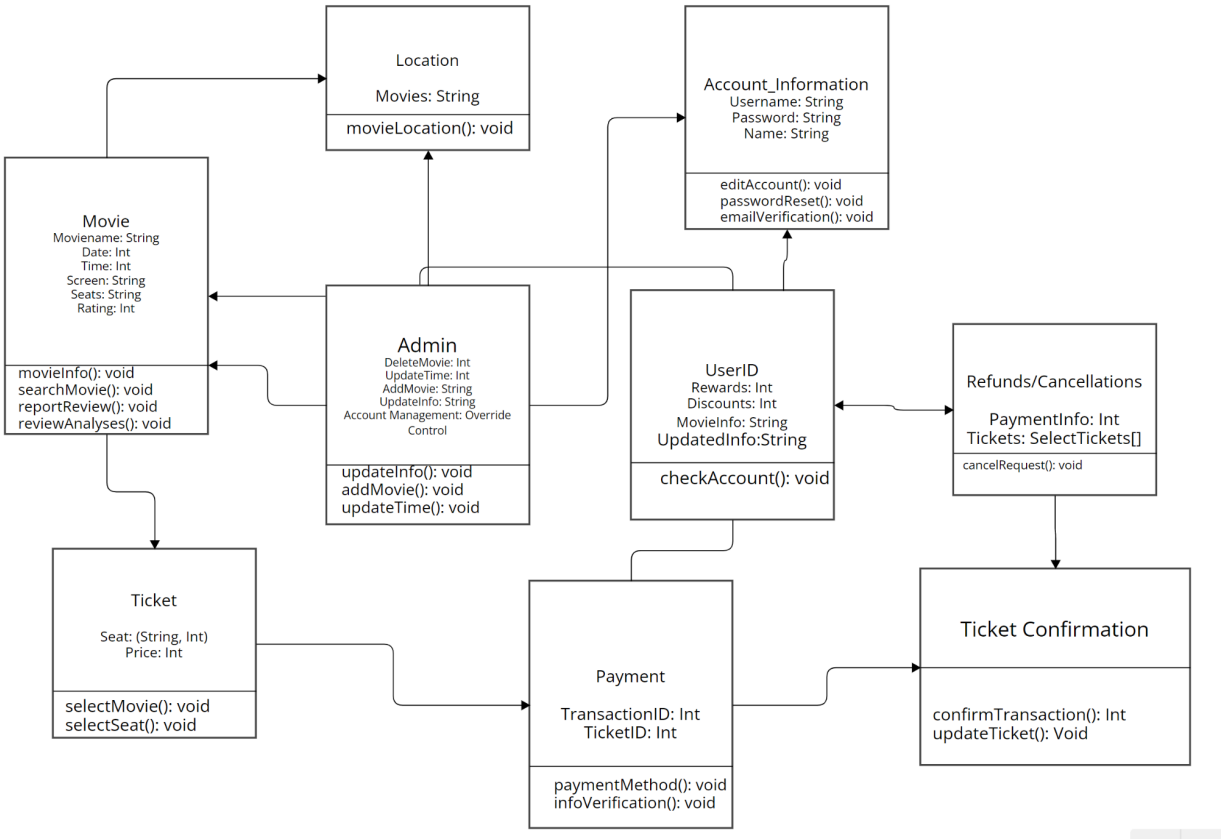
Catchall section for any additional requirements.

4. Software Design

4.1 System Description

The Software Requirement Specification (SRS) describes the overall design and analysis of the Theatre Ticketing System. The system was designed to be user-friendly, allowing for simple account creation and ticket purchase while still being compatible with a wide range of operating systems. The main features include bot blocking, concurrent user management, and extensive administrative support. The project's goal is to produce the prototype in five months on a budget of under $500,000. The Theatre Ticketing System, which prioritizes security and user experience, aims to improve the dependability and efficiency of ticket purchases for a wide range of audiences. This SRS acts as a roadmap for designers and engineers, outlining the project's goals, functionality, and ideal user interface.

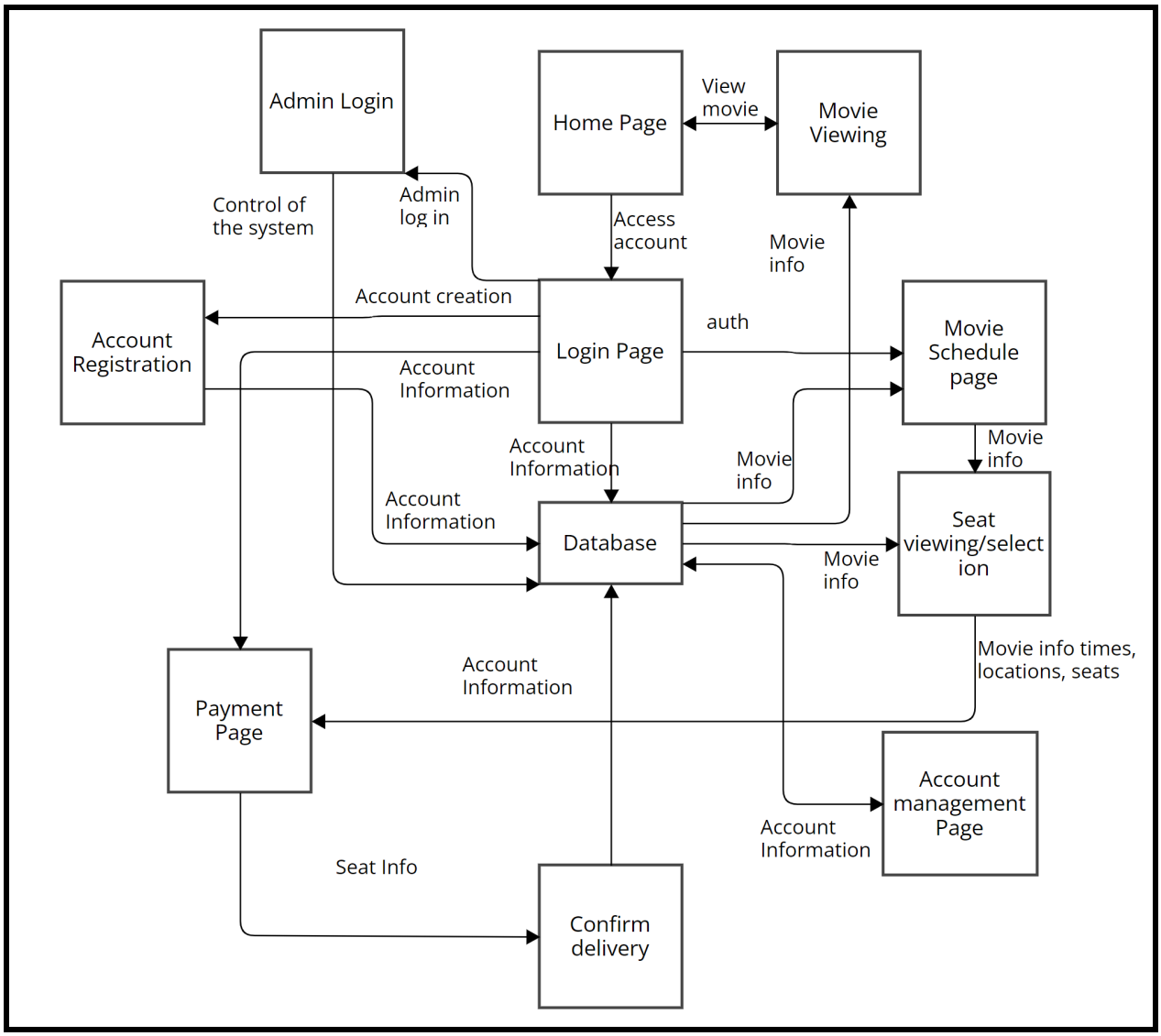
4.2 UML Diagram & Explanation:



Location: Holds a string attribute for the location of the movies(Location of the Theatre).
Movie: Holds details about the movie like the price,seats, rating of the movie, name, etc.
Ticket: This gives us many links to movies with plenty of attributes for seating, pricing, payment method to purchase tickets.

Account_Information: This section stores the user's name, userID, password and their email address.
Admin: THe admin has full control of the system. The admin has access to all the data and holds the power to override any function/attribute on the webpage.
UserID: Connects to AccountInfo and has integer attributes and for rewards earned by purchasing tickets. It also shows discounts, movie information and a string attribute for updated information.
Refunds/Cancellations: This Holds the information of the user and their payment information. As this function can be used by the user to cancel their tickets and possibly get refunds for the cancellations.
Payment: This section holds ticketIDs for conformations and also cancellations.
Confirmation: Linked to payment, this part includes methods to confirm your booked tickets or get a confirmation on the canceled tickets.

4.3 Software Architecture Overview



Home Page: This is the first page of our website that allows you to view movies and login.
Login Page: This page allows you to login into your account. If you do not have an account you can go to account registration.
Admin Login: This allows the staff to login into their account. It also enables them to have full control over the system.
Movie Viewing: This allows guests to view the movies we provide before they log in to see if they want to watch them.
Database: This allows us to store all of the data in our system. For example the movies, times, seats, account information and admin permissions. This is where the data is pulled from when it is needed.
Movie Schedule Page: This section allows the user to view available films, their showtimes and also their preferred location without logging in. Users can choose showtime which leads them to seat selection and then the purchase. This page gets and updates information from the database,

which helps the users to have the latest information about the movie and have an exceptionally good transition from selecting the movie to purchasing tickets.
Seat Viewing/Selection: This section allows the user to view available seats and select which ones they want to buy.
Account Registration: This is where the user can create their account with their information.
Payment Page: This is used to complete their transactions. After Selection a movie, number of seating and adding additional items/services the client is directed to the payment page. Here they enter their card details and/ or other payment details and complete their purchase.
Account Management Page: This allows the users to manage their account and their account only. After logging in users can access the page to update their personal information and also view/change their payment info, view transaction history and manage other services related to their account.
Confirm Delivery: This is the last step after the payment goes through. THis is going to be completely automated. The user will be notified by email/text along with a copy of their ticket and their Confirmation ID.

4.3.2 Description of classes

The User_Account class represents a system's user data and functionality. It serves as a central point for user account management, containing attributes such as a unique identifier (UserID) for user identification, a password for authentication (Password), the user's full name (Name), email address (Email), and saved payment information (Payment). The class provides essential services such as Register, which allows users to create new accounts; Login, which allows users to access their existing accounts; and Forgot Password, which allows users to reset their passwords using their email address. The Ticket class discusses event tickets and their characteristics. It includes information such as the ticket purchase price and the ticket holder's unique identification (UserID), the ticket itself (TicketID), the event or movie to be seen (EventID), and the ticket itself. The class has functions like Purchase_tickets, which allows consumers to purchase tickets and select between standard and luxury seats, and Cancel, which allows users to cancel bookings and view refund amounts.

4.3.3 Description of Attributes

The User_Account class includes features that define and distinguish users inside a system. Each user is assigned a unique identifier, known as a UserID, to ensure distinctiveness across the user base. Password, a user-defined password, serves as the authentication mechanism. The Name attribute stores the user's full name, allowing for comprehensive identification. Furthermore, the user's associated email address, known as Email, acts as a contact and account recovery point. The Payment property stores crucial information about the user's saved payment credentials, the Payment property contains crucial information about the user's saved payment credentials, such as the transaction ID, which allows for smooth transactions inside the system. In addition, the Ticket class is designed to represent event tickets and has unique properties that distinguish each ticket instance. The UserID feature assigns a unique identification to the individual who owns

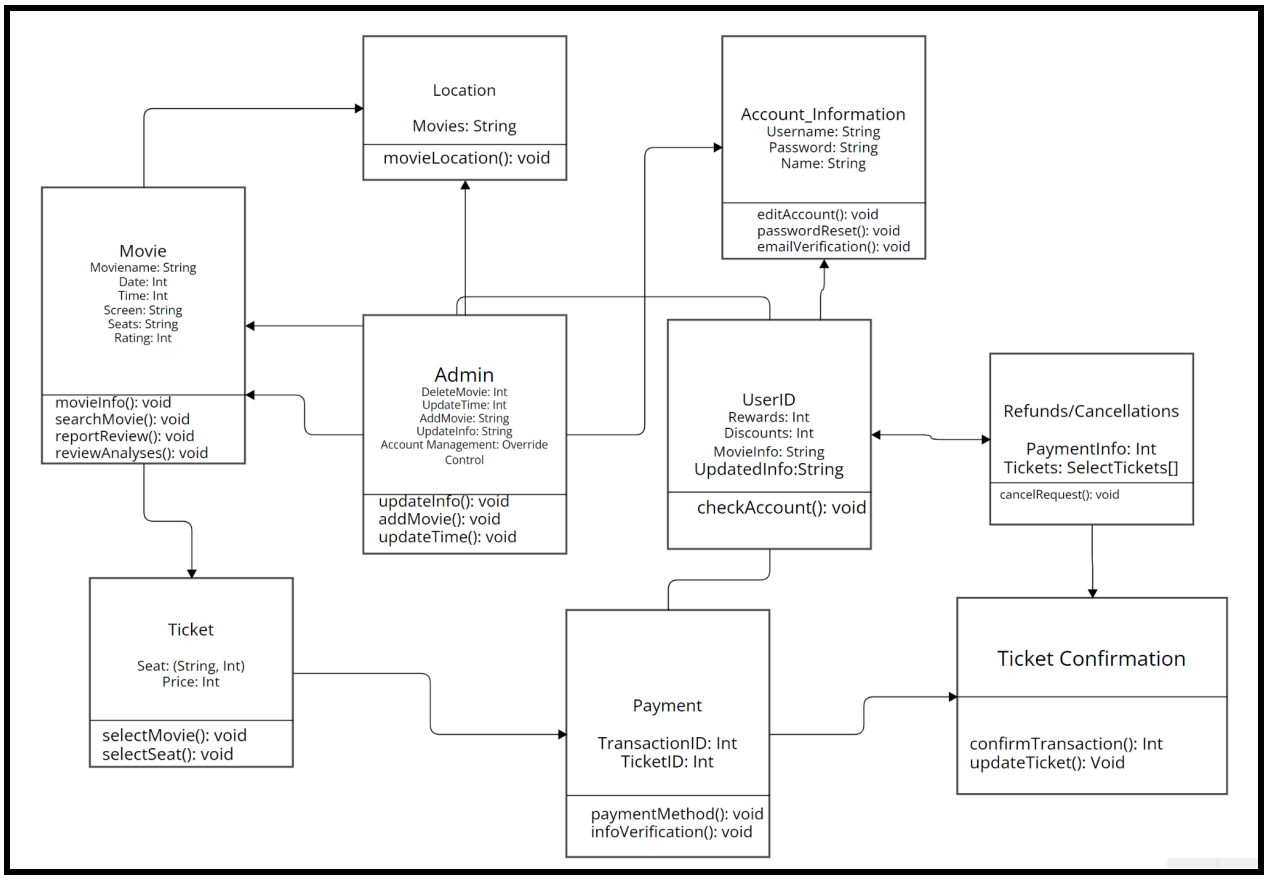
the ticket, ensuring traceability. Every ticket is assigned a unique identifier known as TicketID, which allows you to distinguish between different tickets. The EventID section contains a unique identification number associated with the event or movie for which the ticket was purchased. Finally, the Price field specifies the ticket's cost, rounding up the set of properties that identify each ticket in the system.

4.3.4 Description of operations

The User_Account class includes the activities that control how users interact with the system. Users can begin the account creation process and generate unique user credentials by clicking the Register button. Because it requires authentication with the user ID and password, the Login procedure acts as a gateway for users to access their accounts. If users forget their passwords, they can reset them by entering their email address into the Forgot Password service. This technology improves user account management and security by offering secure login and account recovery alternatives. On the other hand, activities in the Ticket class are designed exclusively for event ticket sales management. The Purchase_tickets action is a critical component that enables clients to purchase event tickets and select between regular and superior seating. On the other side, the Cancel function gives users control over ticketing transactions by allowing customers to cancel reservations while also displaying the estimated refund amount. Together, these actions enable a wide range of tasks, allowing the system to run quickly and easily for both event ticketing and user account management.

4.4 Development plan and timeline

Hunter Thomas has been charged with security management. In this position, he is in charge of supervising and implementing security measures to protect the organization's information, assets, and systems.
Krish Lodha is the test planner, which means he is in charge of organizing and coordinating all testing activities inside the project or business.
Aryan Arora is a developer, which means he designs, codes, tests, and maintains software programs.



This updated version of the UML diagram includes extra functionality that were inadvertently excluded from the prior version that was delivered with the diagram. This is because the UML diagram displayed here is an updated version of the one originally provided, and the work's tests have been completed. This is the cause of the problem. These functions were consolidated into one assignment following thorough testing of a variety of methods, including "updateInventory," "ProcessReturn," "ProcessTransaction," and "DisplayInventory," among others. This is directly tied to what occurred throughout the testing procedure.Regardless of whether you want to rephrase it, say it in another language, or restate it.

5 .1 Unit testing Management of account

Test 1: Registration of Account

- **Description:** This verifies that the user has efficiently created a new account by filling out their information without any errors.
- **Test Set/Vector:** A valid Last and First Name, Email Address and Password is entered by the user.
- **Expected Outcome:** Successful creation of a new account with valid information.
- **Pass/Fail Criteria:** Passes if all the information is valid and verified.

Test 2: Verification of Email

- **Description:** This ensures that the verification process works. The users need to click on the link sent to their email.
- **Test Set/Vector:**
    - Email verification is received.
    - Verification link clicked.
- **Expected Outcome:** Email is verified and account is activated.
- **Pass/Fail Criteria:** Passes when the user's email is verified.

Test 3: Password reset

- **Description:** This allows the user to reset their password using an encrypted link sent to their email address when requested.
- **Test Set/Vector:**
    - The user requests for a password reset link on their registered email address.
    - User receives the email for the password reset link.
    - User sets a new password using the link provided.
- **Expected Outcome:** A new password is set and can now login using their email and their new password.
- **Pass/Fail Criteria:** Passes when the user successfully resets their password and can access their account with the newly set password.

5.2 Unit testing Management of Ticket

Test 4: Process of ticket booking

- **Description:** This verifies that the interface is user friendly and efficient without any errors.
- **Test Set/Vector:**
    - The user selects the movie.
    - The user selects the seat and the type of seat(Regular/deluxe).
- **Expected Outcome:** Ticket is successfully booked after the following steps.
- **Pass/Fail Criteria:** Passes if the booking confirmation is received on their email.

Test 5: Ticket cancellation and refund.

- **Description:** The system checks which tickets are booked, and whether those tickets can be canceled/refunded.
- **Test Set/Vector:**
    - The user cancels the ticket within the permitted time period.
- **Expected Outcome:** The booking will be canceled and the refund amount will be processed to the original refund method.
- **Pass/Fail Criteria:** Passes when the user receives an email confirmation of the cancellation and/or refund of their booking.

Test 6:

- **Description:** Verify that customers are able to change their ticket bookings, such as the date, time, or seat.
- **Test Set/Vector:** Original information regarding the ticket. New information about the reservation you seek.
- **Expected Outcome:** The reservation has been modified to reflect new information.
- **Pass/Fail Criteria:** The test is successful if the modifications are confirmed and reflected in the booking.

Test 7:

- **Description:** Ensure that users may search for events or movies using keywords and filter the results by date, location, or genres.
- **Test Set/Vector:** Enter your search terms in the search form. Use various filters to narrow down the search results.
- **Expected Outcome:** The system displays events or videos that meet the search criteria and filters.
- **Pass/Fail Criteria:** The test passes if the search results correspond to the user's search keywords and filters.

5.3 Unit testing Management of Reviews and Ratings

Test 8:

- **Description:** Make sure that users can access and view ratings and reviews for a certain event or movie.
- **Test Set/Vector:** There are none (accessing publicly available reviews and ratings).
- **Expected Outcome:** Users can browse through ratings and reviews.
- **Pass/Fail Criteria:** The test passes if the reviews and ratings are easy to read.

Test 9:

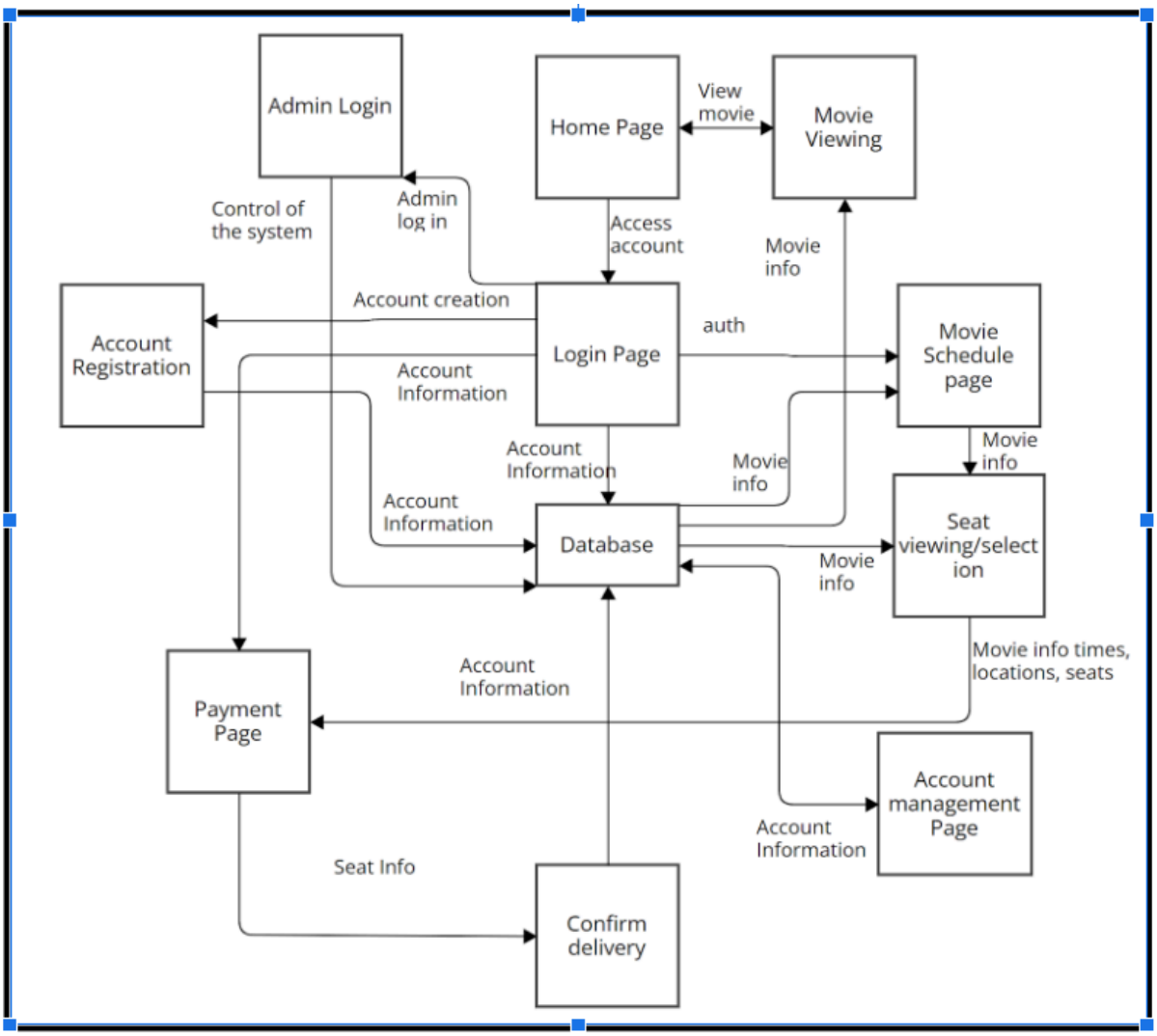
- **Description:** Allow people to report improper or objectionable content in reviews.
- **Test Set/Vector:** Selecting an improper review to read. Documenting the actions that were taken.
- **Expected Outcome:** The report was submitted and acknowledged to allow for moderation.
- **Pass/Fail Criteria:** If the system accepts the report for evaluation, the test is considered a success.

Test 10:

- **Description:** Ensure that the rating and review system is reliable and unbiased by conducting an inquiry into its reliability and impartiality.
- **Test Set/Vector:** A sample of reviews was analyzed to verify anti-manipulation procedures and assure consistency.
- **Expected Outcome:** The system presents an accurate depiction of user feedback with no indications of manipulation.
- **Pass/Fail Criteria:** If the reviews and ratings are authentic and objective, the test is deemed successful.

6.0 Architecture Design w/Data Mgmt.

6.1 Architectural Diagram:



6.2 General description:

The image shows a flowchart for a theater ticketing system. It includes options for paying purchases, watching movies, creating accounts, and signing in as an administrator. The system maintains accounts, schedules movies, and displays seat information. In addition, there is a library that saves and retrieves movie and account data. It communicates with several system components.

6.3 Data Management Strategy:

User and movie data are kept and organized in a single SQL relational database. This ensures that data utilized in operations such as ticketing and scheduling is correct and consistent.

6.3.1 Relational database (SQL): This sort of database is controlled using a structured query language, which arranges the data in a way that generates tables in a specified pattern.

6.3.2 Single Database: The system has a single, centralized database that stores all of its data. This simple solution reduces the hassle of maintaining synchronization across several databases.

6.3.3 Table structure: Although it is unclear from the image, tables are most likely used to manage multiple types of data, such as user accounts, movie information, seat assignments, and payment methods.

6.3.4 Foreign Key Relationships: These ensure referential integrity between tables. A foreign key in a user account table that connects individuals to payments in a transactions database is one example.

6.4 Trade Off Discussion:

6.4.1 Technology Choice (SQL vs. NoSQL): SQL databases feature well-defined relationships and ordered data, allowing for more advanced searches and data security. NoSQL may not be transactionally stable, but its flexibility and extension potential are worth considering.

6.4.2 Single Database vs. Multiple Databases: A single database makes system planning and maintenance easier, but it may become inefficient as the system grows. While having many systems can help with workload distribution, it makes it more difficult to maintain consistent and up-to-date data.

6.4.3 Table Organization: Organizing tables according to entities and their relationships improves data accuracy and query efficiency. However, careful preparation is required to prevent adding additional components that slow down the process.
- A NoSQL database can be used for system components that require high scalability, such as session storage; an alternative is to utilize a polyglot persistence technique, which stores various types of data in both SQL and NoSQL databases.

6.5 Summary:

The theater ticketing system uses relational databases with SQL capability to manage both relational and structured data. There is only one core database used, and the tables are carefully organized to ensure data consistency. This traditional method works well for difficult queries and maintaining data consistency, but as system requirements grow, scaling alternatives may need to be considered. While balancing complexity and performance, merging NoSQL databases or using multiple databases can assist meet a wide range of data storage needs.