Company: AMC

Products: Ticketing Software, organizations

Software Requirements Specification Template

Software Engineering

The following annotated template shall be used to complete the Software Requirements Specification (SRS) assignment.

Template Usage:
Text contained within angle brackets ('<', '>') shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either 'Smart Home' or 'Sensor Network'.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

Ticketing System

Software Requirements Specification

v1.0.3
2/15/2024

Group 1
Khang Tonthat
Rafael Aguilar
Ethan Morales

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

Revision History

Date	Description	Author	Comments
<date>	<Version 1>	<Your Name>	<First Revision>
2/15/24	Version 1.0.0	Rafael Aguilar, Khang Tonthat, Ethan Morales	Created SRS Outline all the way up to Section 3.6
3/5/24	Version 1.0.1	Rafael Aguilar, Khang Tonthat, Ethan Morales	Completed the remaining Section 3.7 and the entirety of Section 4 of SRS
3/14/24	Version 1.0.2	Rafael Aguilar, Khang Tonthat, Ethan Morales	Added section 6 into SRS, outlining Test Validation, Estimation, and Cases that are to be used in our development.
3/28/24	Version 1.0.3	Rafael Aguilar, Khang Tonthat, Ethan Morales	Added section 7 into SRS, adding Data diagrams regarding SQL and management into the system.

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. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document. (Note: the following subsection annotations are largely taken from the IEEE Guide to SRS).
The introduction of this Software Requirements Specification

1.1 Purpose

The purpose of this SRS document is to collect all ideas that our group has come up with which will serve to define our software requirements. As well as building and predicting various concepts that may help or become an obstacle in the process of building the software. This document also illustrates the various requisites and prerequisites that the stakeholders may specify.

1.2 Scope

This product will produce a ticketing website system in which customers can buy tickets for movies available in theaters. Providing movie options for customers in need of buying tickets. As well as providing a selection of movie theaters nearby for the customer's best accommodation. And it will be well protected from malicious outside sources which may harm customers.

This product will not have any malicious hardware which may put the customer's personal information at risk. It will not have a messy layout that may be difficult and confusing to the customer. It will run smoothly software wise to make it as fast as possible for the customer to buy tickets.

The goal is to provide a more streamlined and convenient way for user to enjoy there film without delay

Modern look, easy to use website with various services which will facilitate users into buying tickets.

Secure broadband, virus-free server that secures users information into an unbreakable information vault.

Provide a comfortable experience regarding signing up for rewards, searching for selected movies, and securing important information in a secure server.

1.3 Definitions, Acronyms, and Abbreviations

Server: Manages access to a centralized service in a network.
User: Customers trying to purchase any tickets we are offering.
Software: Necessary components in order to build and create the backend compartments of the website.
Hardware: The necessary components in which the website should be able to operate.
User interface: Interface in which customers will interact with various services that are provided on the website.
Functional Requirements: Requirements necessary for product to properly function in order
Broadband: high-capacity transmission technique that can carry multiple signals and data types simultaneously over a communication medium.
Data Privacy: Protection of personal and sensitive information from unauthorized accessed from the website.

1.4 References

IEEE Computer Society. (1998). IEEE Recommended Practice for Software Requirements Specifications. STD 830-1993.
Document used to understand the practice of basic software requirements specifications in order to create our own.

NATO Software Engineering. October 11th, 1968. NATO Science Committee.
Document used in order to understand the concept of a working team constructing an SRS.

Software Requirements Specification Document With Example. May 8th, 2024. Ravi Bandakkanavar.
Used to understand the basics of important elements of an SRS like use cases as well as requirement specifications.

1.5 Overview

The rest of this document contains:
General description of the product.

Specific requirements that are needed for the product to be successfully developed.

Analysis models providing examples of the finished products.

Change management process identifies the process and changes towards the main SRS.

2. General Description

This document describes many factors about problems this product may have when it comes to the building of it. As well as solutions for any of these problems. It contains a list of various solutions that users and stakeholders will have. It as well shows the requirements from various stakeholders. It furthers our knowledge and understanding of the specific requirements a stakeholder may need and want for the ticketing system. As well as a brief description of major features.

2.1 Product Perspective

The ticketing system is a component of a larger system that involves multiple interfaces in order to make it function. Products such as Ticketmaster, AMC Theaters, and Regal Cinema operate using similar systems and encounter multiple of the same functions as well as constraints due to the nature of handing out and managing tickets on a large scale.

Our product should be easily integrable onto multiple platforms or third party services, whether it be through the use of API's or compatibility that allows users to vendor their own tickets depending on the event.

Through these platforms, they should be able to seamlessly link to a payment gateway to easily vend the ticket without much hassle by making our software compatible with modern financial systems

We'll also provide data analytics for our users who are vending tickets, giving them information on how many tickets have been bought, sections/seats that have been taken, revenue, etc.

2.2 Product Functions

The functions of the software will be able to perform a ticketing based where clients can come to the website and buy tickets for films they would like to view. Its functions should include scheduling, ticketing, availability, and a pop of details in regard to the film to entice the clients to buy the tickets.

Links to other software such as Instagram, Twitter, Facebook, TikTok, and YouTube to promote clients to buy tickets.

2.3 User Characteristics

- Elderly (60+)
- Military/ Service Members/ Veterans
- Under 18 (Teens)
- Children (2-12)
- V.I.P / Movie Members

2.4 General Constraints

- Scalability (Handling large volumes of traffic since there are peak periods of time due to selling tickets for popular movies)
- Cookies
- Allocated memory set for the website
- Data Privacy (Handling User Data when They Input Private Information to Purchase Tickets or Logging into Accounts)
- General Performance (We want our website to be modern and sleek, but not unresponsive so that our site runs on all devices to maximize users)
- Location based (Different laws may affect what is available and allowed)

2.5 Assumptions and Dependencies

- Platform Compatibility (MacOS, Windows, etc.)
-
- Internet Connectivity: User is assumed to have internet connectivity in order to access the product.
- User authentication: User is assumed to have valid credentials.
- User Training: User is assumed to be familiar with website structure.
- Data Accuracy: Data by users is assumed to be accurate and valid.

Dependencies:

- Web Server: Website depends on a functional web server in order to work properly.
- Network Infrastructure: Functionality depends on well developed network infrastructure.
- Security Software: Security depends on the functionality of security measures like detection systems.
- Payment Paywalls: Purchasing tickets depends on payment paywalls for secure transactions.
- Compatibility of Browsers: Whole system depends on compatibility with various web browsers.

3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- *Correct*
- *Traceable (both forward and backward to prior/future artifacts)*
- *Unambiguous*
- *Verifiable (i.e., testable)*
- *Prioritized (with respect to importance and/or stability)*
- *Complete*

- *Consistent*
- *Uniquely identifiable (usually via numbering like 3.4.5.6)*

Attention should be paid to the carefully organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

3.1 External Interface Requirements

3.1.1 User Interfaces

- The system shall display a variety of movie options for the customer.
- The system shall display a brief summary of the movie that the customer selects.
- The system shall display options of seating in the form of rows.
- Include itinerary displaying movie time and dates.
- Should display the length of the movie as well as rating and when it was first released.
- Should display a search bar for customer needs.
- Should include a sidebar with various options ranging from rewards, finding a theater/movie.
- Should display the amount of tickets a customer may want and the age range for each ticket.
- Should display the total amount of money the customer owes.
- After each purchase, a QR code should be displayed to indicate the customer's tickets.
- Should display an option regarding whether a customer may want to sign up and become a member,
- Should display a sign-in button for customers with existing accounts.

3.1.2 Hardware Interfaces

- Software should be compatible with hardware configurations
    - Configurations include: Personal Computers, Laptops, Smartphones/Tablets
- Should also support machinery such as printers so that users can print out tickets if they want them physically rather than digitally
- Should also support machinery such as scanners so that if users use a barcode or a QR scanner, their ticket will allow them access to their event

3.1.3 Software Interfaces

- Will be accessed through most web browsers, so should be compatible with multiple types of them
    - Chromium: Google Chrome, Microsoft Edge, Opera
    - Gecko: Mozilla Firefox
    - WebKit: Apple's Safari
- Users will access the website from multiple types of devices, so it's important that the website can run on many types of operating systems
    - MacOS
    - Windows
    - iOS

    - Android
- Must be integrable with outside software that helps authorize and process payment methods
    - PayPal
    - Stripe
    - Shopify

3.1.4 Communications Interfaces

- Websites should use HTTPS to help secure communication between the user and our website servers.
- Website should also be capable of sending emails for tasks such as user login confirmation, ticket purchase confirmation, account updates, and special announcements

3.2 Functional Requirements

3.2.1 <Online Ticket Purchasing>

3.2.1.1 Introduction
The system should allow users to browse available movies, see when they start showing, view available and taken seats, and finally make their secure online payments to purchase tickets.
3.2.1.2 Inputs
User inputs will include movie selection, preferred showtime, seat choices, and their payment information.
3.2.1.3 Processing
The system will check seat availability to prevent duplicate tickets, calculate the total cost, and process the payment securely through a payment processor
3.2.1.4 Outputs
If a user completes their purchase, the system will send them a confirmation email with either a unique QR or Barcode for their ticket(s). The website will update to take into account the tickets purchased by removing their seating from being available.
3.2.1.5 Error Handling
In the case that the user fails to purchase their ticket because their payment method declines, or the seat they desired is taken during the payment process, they will be offered a refund for any money that may have been already processed, and an error message will be displayed.

3.2.2 <Member Registration>

3.2.2.1 Introduction
Guest users are given the option to create an account when they first open our website, allowing them to register through our online ticketing system. Through the creation of an account, the user is given specific perks such as special announcements, viewing purchase history, and having their preferences saved.
3.2.2.2 Inputs
User Inputs will include the option to create an account, the input of their email, phone number, and password.
3.2.2.3 Processing

The system will check to see if their email or phone number is already linked to an account, then will create a unique user ID and register the user after sending them a confirmation email with a 30 minute window.

3.2.2.4 Outputs

Through the creation of the account, the user is given access to specific perks that are catered to help their purchasing experience and record their purchases

3.2.2.5 Error Handling

In the case that the user's email or phone number is already registered, an error message is output telling the user that their account has failed to be created since they may already have an account registered, barring the creation of the account and the generation of a confirmation email.

In the case that the user fails to confirm their information through the confirmation email after 30 minutes, the system will send a reminder email asking the user to generate a new confirmation email so that they may start using their account. If they choose to ignore the reminder, their account will be deactivated until confirmation has been received.

3.3 Use Cases

3.3.1 Use Case #1: Online Ticket Purchase

3.3.1.1 Actors

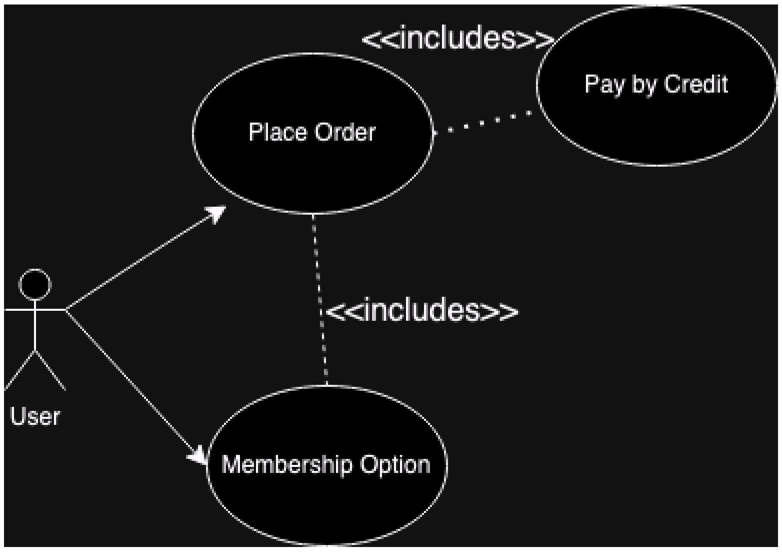
- Guest User
- Registered User

3.3.1.2 Purpose

Guest users can browse available movies, select showtimes, choose seats, and make online payments for ticket purchase. Registered users can log in to access their saved information.

3.3.1.3 Scenarios

1. Guest/Registered user selects a movie, seats, and showtime.
3. The system validates seat availability and calculates the total cost.
4. User is provided the option to log into or create an account to making ticket purchasing easier
5. User is then sent to view payment processor in order to validate payment
6. User is given the option to choose their payment method (Debit, Credit)
6. Email containing ticket information is sent and the user's account is updated



3.3.2 Use Case #2: Member Registration

3.3.2.1 Actors

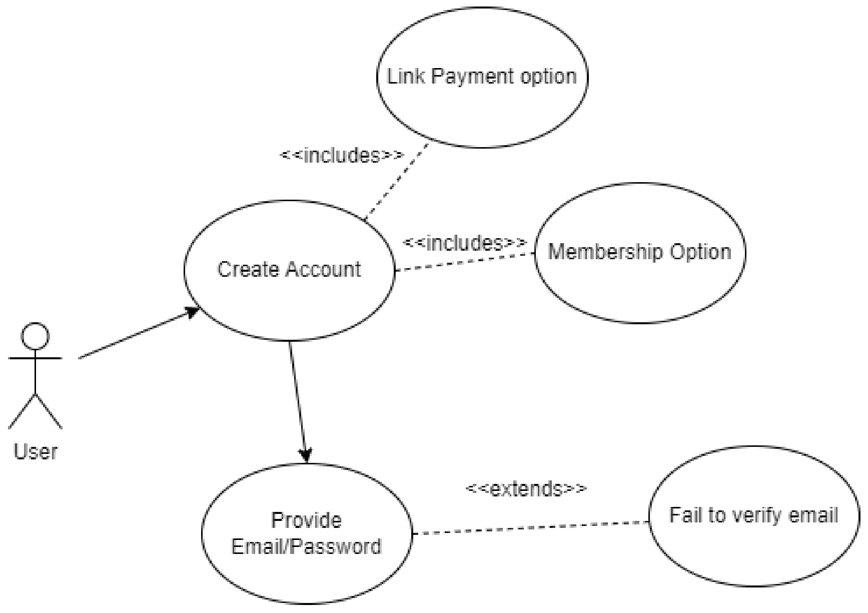
- Guest User

3.3.2.2 Purpose

Guest users can create an account by registering with the online ticketing system. Registration allows users to access additional features such as saving preferences, viewing purchase history, and receiving personalized recommendations.

3.3.2.3 Scenarios

1. Guest user navigates to the registration page.
2. System displays a registration form with fields such as: Name, Email/Phone number, Password, Address
3. Guest user fills in the required information.
4. the System validates the entered information for completeness and accuracy.
5. If validation is successful, the system generates a unique user ID and registers the user.
6. System sends a confirmation email to the provided email address.
7. User clicks the confirmation link in the email to verify their account.
8. Once verified, the user gains access to additional features and can log in to the system.

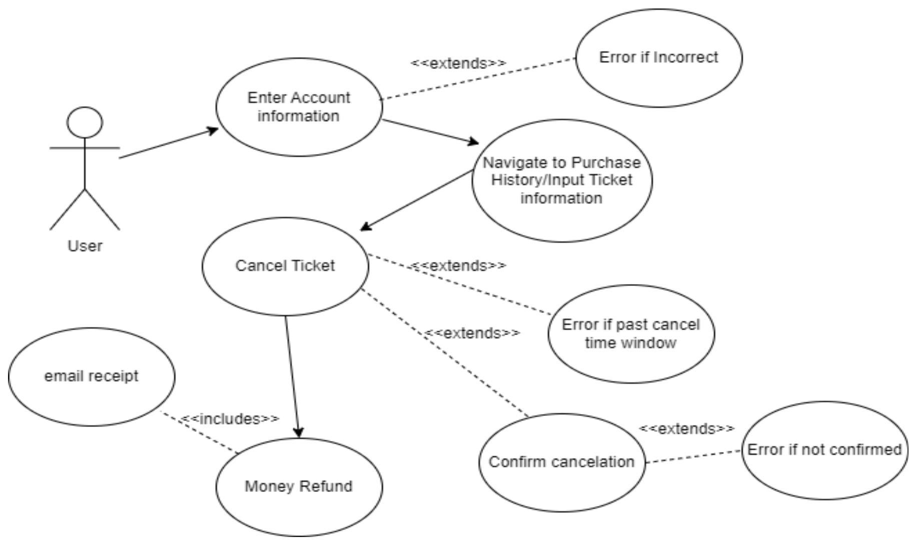


3.3.3 Use Case #3: Ticket Cancellation
3.3.3.1 Actors

- Registered User

3.3.3.2 Description
Users are allowed to cancel any purchased tickets as long as it is before their showing, giving them a full refund of their purchase.
3.3.3.3 Scenarios

1. User logs into system
2. User navigates to their purchase history
3. User is allowed to cancel their purchase as long as it is before their showing
4. System asks for confirmation in their cancellation
5. System will process their cancellation, update the user account, and give them their refund in due time.



3.4 Classes / Objects

3.4.1 <Class / Object #1>: User Account

3.4.1.1 Attributes

- User ID (int): Unique Identification for each user account
- Email (string): Email address associated with the account, will be used in the case of a username
- Password (string): Encrypted password meant to serve as securing user information
- Phone number (int): User's phone number that serves as a second platform for messages other than email

3.4.1.2 Functions

- updateProfile(): Allows user to update their information, such as adding phone number
- changePassword(): Allows user to change their password
- changeEmail(): Allows user to change their email address

3.4.2 <Class / Object #2>: Ticket

3.4.2.1 Attributes

- Ticket ID (int): Unique identifier for each ticket
- MovieTitle(string): Title of movie that is corresponded to the ticket purchased
- Showtime(int): Date and Time of showing
- SeatNumber(string): Assigned seat corresponded to the ticket purchased
- Price(float): Ticket cost
- Status(bool): Indicated whether the ticket has been purchased or still available.

3.4.2.2 Functions

- generateQRCode(): Generates code corresponding to the ticket ID that allows you to scan your ticket at check-in.
- cancelTicket(): Allows the user to cancel their ticket, updating its status back to being available and prompting a refund.

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

- Scalability: The system should support a minimum of 5000 users without affecting the performance of the system.
- Response Time: The system should be able to respond to a user within a maximum of 1.5 seconds.

3.5.2 Reliability

- Tolerance regarding any faults: The system should still continue to function regarding any data loss or small faults throughout the service until the problem is solved.

- Recovery: The system should be able to document various data in order to prevent any data loss from a fault.

3.5.3 Availability

- Time Available: The system should be able to operate within all times around the world. Should be operating at all times.
- World Wide Usage: The system should be able to be used only locally.

3.5.4 Security

- Authentication: Users should be able to authenticate by using secure methods. The proper authentication should be included for users to authenticate themselves.
- Data Encryption: The system should encrypt any private and sensitive information that a user may enter in order to secure and avoid any data cracks by unauthorized users.
- Access Control

3.5.5 Maintainability

- Software Maintainability: This system should have the tools and personnel in order to maintain a quality assurance over the system.
- Documentation: Documentation should be provided for the architecture of the system in case of any faults in the system.
- Hardware Maintainability: The system should function properly even in case of various hardware failures in the absence of the customer.

3.5.6 Portability

- Training Requirements: This system should be designed to accommodate the user for as less minimal training required for the use of this system.

3.6 Inverse Requirements

3.6.1 Duplicate Accounts

- Users shouldn't be able to create an account with an email that has already been registered.
- Inverse Requirement: The system must enforce uniqueness for email addresses as a user tries to create an account, barring them from creation if they input an already registered email.

3.6.2 Duplicate Tickets

- Users shouldn't be able to purchase a ticket that has already been purchased by another user
- Inverse Requirement: The system will track ticket status, only allowing the purchasing of a specific ticket as long as its status still deems it available, and then turning its status off once availability of the ticket is gone.

3.6.3 Unauthorized Ticket Cancellation

- Users shouldn't be allowed to cancel tickets without proper authorization and confirmation

- Inverse Requirement: User will have to log into their account in order to access their purchase history, and will implement multiple prompts to confirm whether the user intends to refund their ticket, such as re-entering their login information.

3.7 Design Constraints

- Mobile web browser support
- Scalability (Handling large volumes of traffic since there are peak periods of time due to selling tickets for popular movies Cookies
- Allocated memory set for the website
- Data Privacy (Handling User Data when They Input Private Information to Purchase Tickets or Logging into Accounts)
- General Performance (We want our website to be modern and sleek, but not unresponsive so that our site runs on all devices to maximize users)
- Location based (Different laws may affect what is available and allowed)

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. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

Member Github Accounts:

Ethan Morales: https://github.com/ethanmorales10

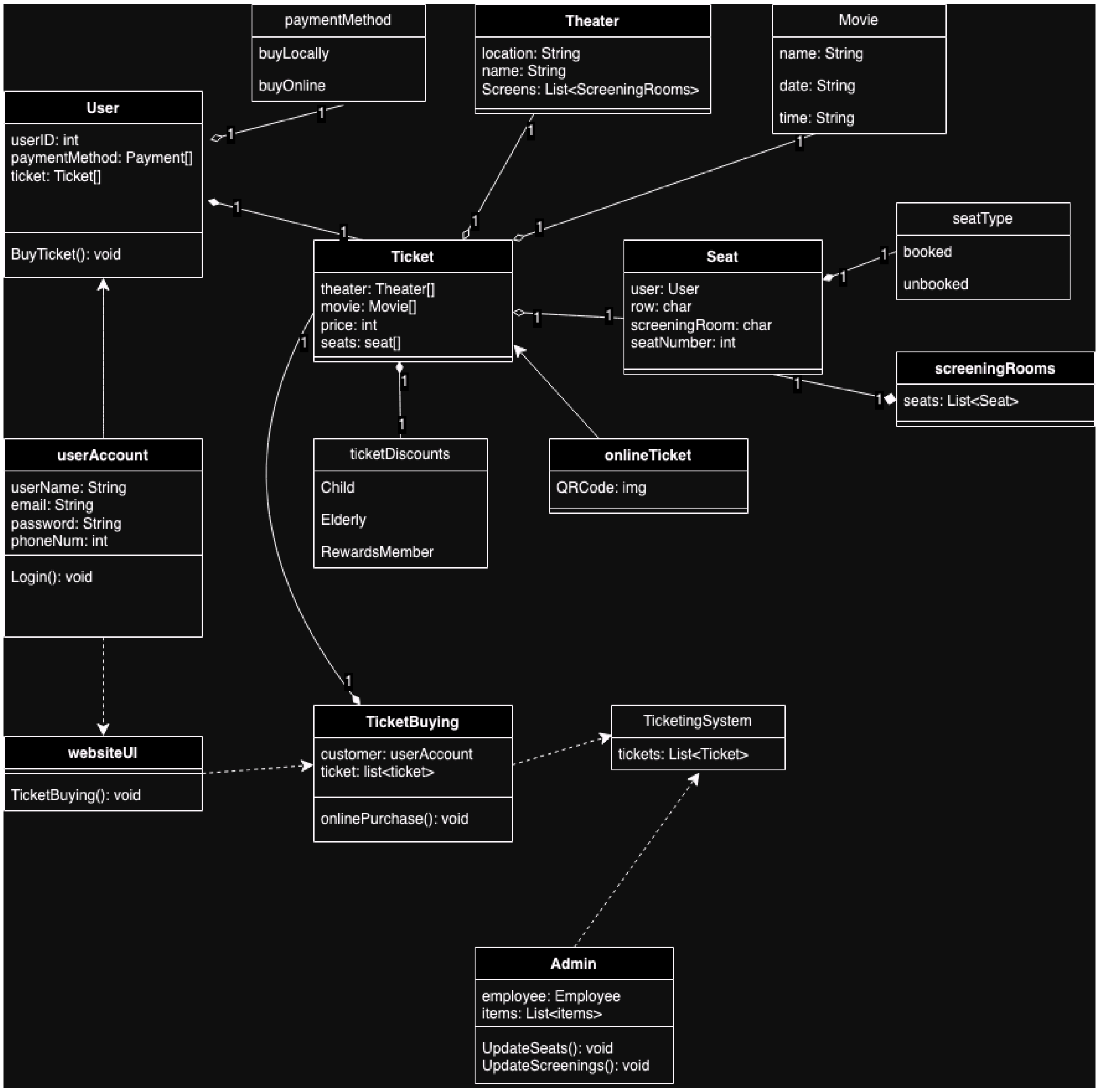
Khang Tonthat: https://github.com/khato7624

Rafael Aguilar: https://github.com/ralphaguilar

Github Repository URL:

https://github.com/ethanmorales10/CS250-Group1-Ticketing-Systemhttps://github.com/ethanmorales10/CS250-Group1-Ticketing-System

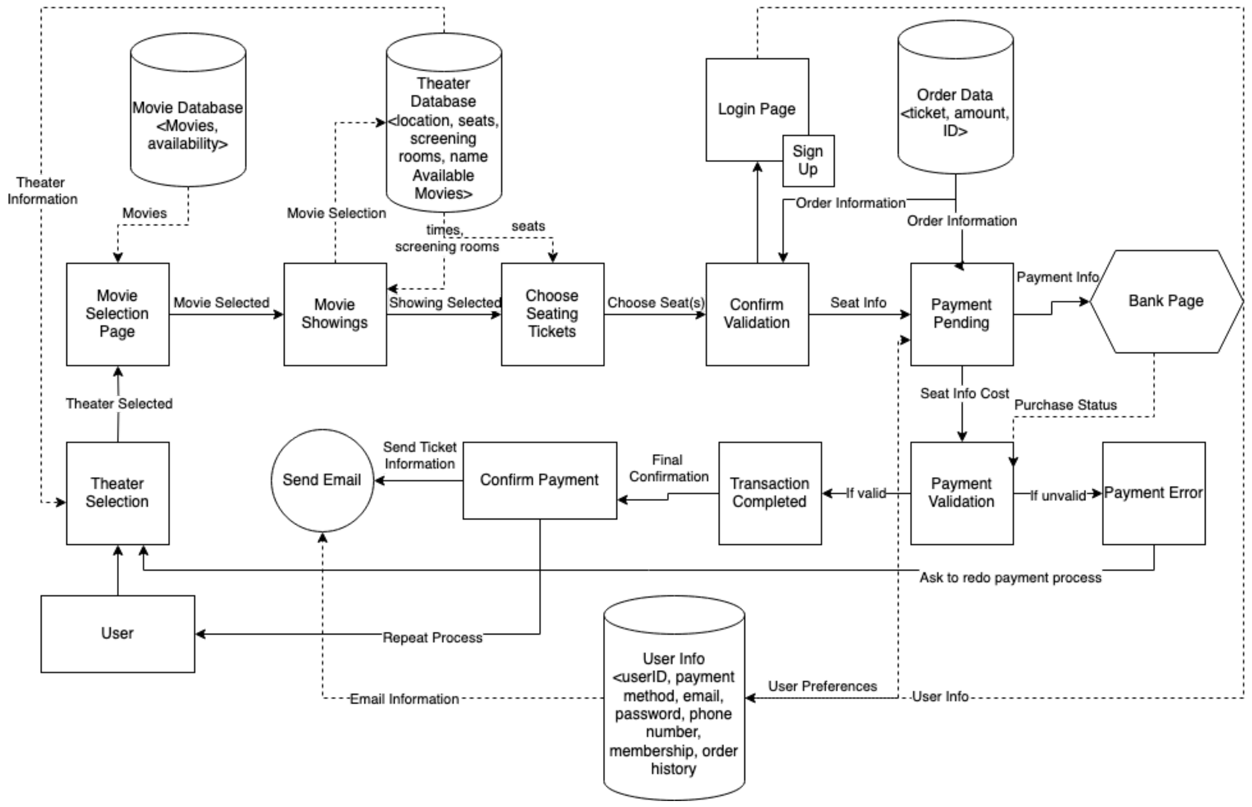
4.1 UML Class Diagram



- The UML Diagram is supposed to show the process the moment a user inserts their card at whichever theater that our ticketing system happens to be in place.
- The user only has the option to buy a ticket, along with providing their user account, how many tickets they need, and whether they want to pay for their tickets physically or digitally.
- When moving into the website UI, they are only able to move onto the ticket buying function.
- The user provides their account info and what ticket(s) they are planning to purchase. This prompts the ticketing system to select a ticket for them to have

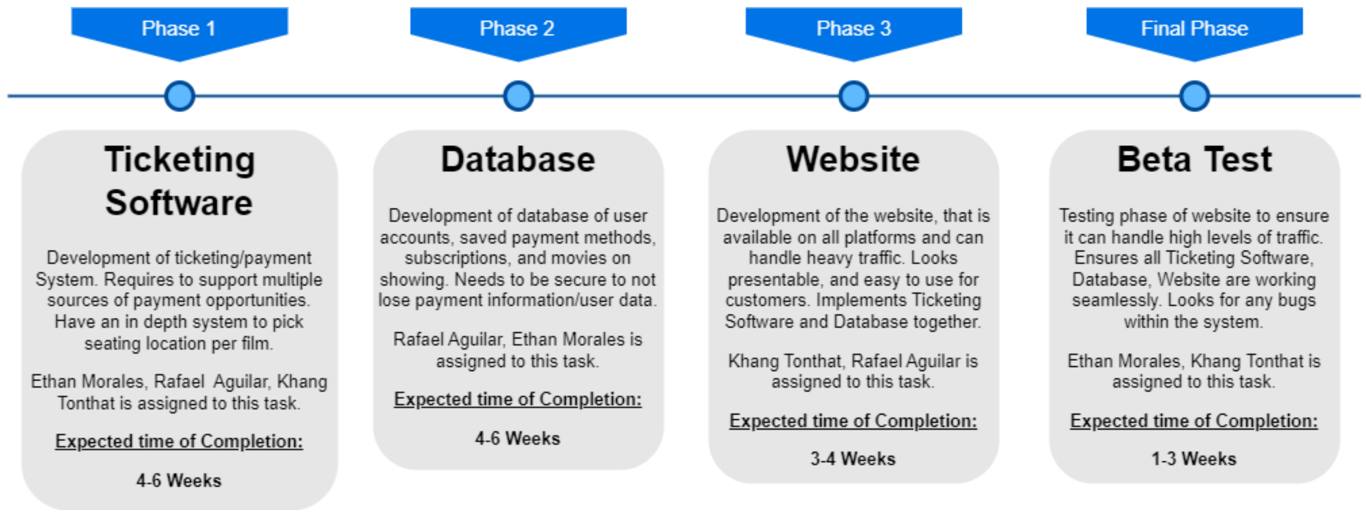
- The ticket determines the theater, movie, seats, and digital formatting depending on the user input, then should be outputted when the user confirms their online purchase.

4.2 SWA Diagram



- This graph is meant to show how data is stored and moved across multiple storage that are allocated for specific types of data.
- When the user first goes through the site, they are asked to select a theater that's provided by the theater database.
- They are then prompted to choose a movie and a showing depending on what's provided from the movie and theater database, along with whichever seats provided by the theater database.
- After confirming all of this, they are prompted to make their payment. When going through validation, they are told to log in (optional) if they have their info stored in the user information database.
- Order Data is then allocated to track the ticket purchase, providing order information throughout the purchase validation.
- While the purchase is pending, the user is prompted towards their bank page to provide their payment information, which will determine the final purchase status.
- When the user reaches the final stage of payment validation, it will determine whether the payment had gone completely through. If the purchase failed(unvalid), it will generate a payment error and prompt the user to restart the payment process.
- If the payment has gone through validation, it will deem that it is completed, then will send an email with their ticket information, and restart the process of ticket purchasing.

4.3 Timeline


Phase 1
Phase 2
Phase 3
Final Phase
Ticketing Software
Development of ticketing/payment System. Requires to support multiple sources of payment opportunities. Have an in depth system to pick seating location per film.
Ethan Morales, Rafael Aguilar, Khang Tonthat is assigned to this task.
Expected time of Completion:
4-6 Weeks
Database
Development of database of user accounts, saved payment methods, subscriptions, and movies on showing. Needs to be secure to not lose payment information/user data.
Rafael Aguilar, Ethan Morales is assigned to this task.
Expected time of Completion:
4-6 Weeks
Website
Development of the website, that is available on all platforms and can handle heavy traffic. Looks presentable, and easy to use for customers. Implements Ticketing Software and Database together.
Khang Tonthat, Rafael Aguilar is assigned to this task.
Expected time of Completion:
3-4 Weeks
Beta Test
Testing phase of website to ensure it can handle high levels of traffic. Ensures all Ticketing Software, Database, Website are working seamlessly. Looks for any bugs within the system.
Ethan Morales, Khang Tonthat is assigned to this task.
Expected time of Completion:
1-3 Weeks

5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

6. Test Plan

The validation process ensures that the ticketing system meets the specified requirements and functions as expected. It involves organizing, scheduling, and assigning responsibilities for testing activities.

6.1 Test Validation

6.1.1 Overview

a. **Organization:**
   Testing activities will be organized by our admin team, with support from our development and operations teams, in order to ensure that our design is free of bugs
b. **Tasks and Schedules:**
   Testing will be divided into three categories based on our test cases seen below in 6.2: Unit, System, and Functional Testing. All of these categories are there to ensure that our system works well from the server side, whenever we interact with our databases, and when a user/admin executes a command that modifies the system.
c. **Responsibilities:**
   Admin team will be responsible for test case creation, execution, and reporting. Developers will help resolve these issues through patches.
d. **Tools, Techniques, Methods:**
   Testing will be conducted in controlled environments separated from outside users, where we will conduct scenarios such as web interface testing and manual testing to determine the status of the system.

6.1.2 Processes

a. **Management:**
Management is to be moderated by the administrator head, who will ensure that testing meets basic requirements and that our methods are beneficial to the development of the system.
b. **Acquisition:**
Testing tools and resources will be acquired based on the type of testing being performed and the real scenarios
c. **Supply:**
Test environments and datasets will be supplied by the administrator team to help simulate real-world scenarios that might lead to bugs.
d. **Development:**
Test cases will be developed based on requirements outlined or are provided by our work within the SRS document and will be updated as development steadily progresses.
e. **Operation:**
Testing will be conducted in the environments that the admin team will provide, including development and production environments, to help us see how our system responds to test cases under certain modifications.
f. **Maintenance:**
Development and the frequent use of test cases will be utilized as our system maintenance, as it will make sure that our system is up-to-date and completely functional.

6.1.3 Reporting Requirements

- *Test Execution results, including pass/fail status and potential defects, will be stored in our Test Cases (6.3)*
- *Executions that yield a failed status will be reported to the entire admin and development team, as these are important areas of improvement and will be needing immediate attention.*

6.1.4 Administrative Requirements

- *Test Environments and resources will be managed by the admin team to ensure that these environments are always available for use*

6.1.5 Documentation Requirements

- *Test cases, test plans, and test reports will be documented and stored in our remote repository for future reference in case of failures or complications.*
- *Documentation is to be updated for every version provided, and will reflect the changes provided in each version along with any new functionalities.*

6.1.6 Resource Requirements

- *Resources required for testing will involve hardware used by our teams, software to maintain the testing environments, testing tools such as our cases, personnel, etc.*
- *Resources are to be managed to fit within our budget, with us only using the appropriate amount of resources to save money and time.*

6.1.7 Completion Criteria

- *Testing will be considered complete once all test cases have been completed with a pass, and the system has been deemed suitable for users.*
- *System requirements include that it be stable, reliable, and functional with the minimum amount of bugs that have been outlined within the SRS.*

6.2 Test Estimation

6.2.1 Estimating Test Required

- *Estimated tests required for the testing process were about 10 which covered important functions between the system and the development of the system.*

6.2.2 Estimated Test Development Time

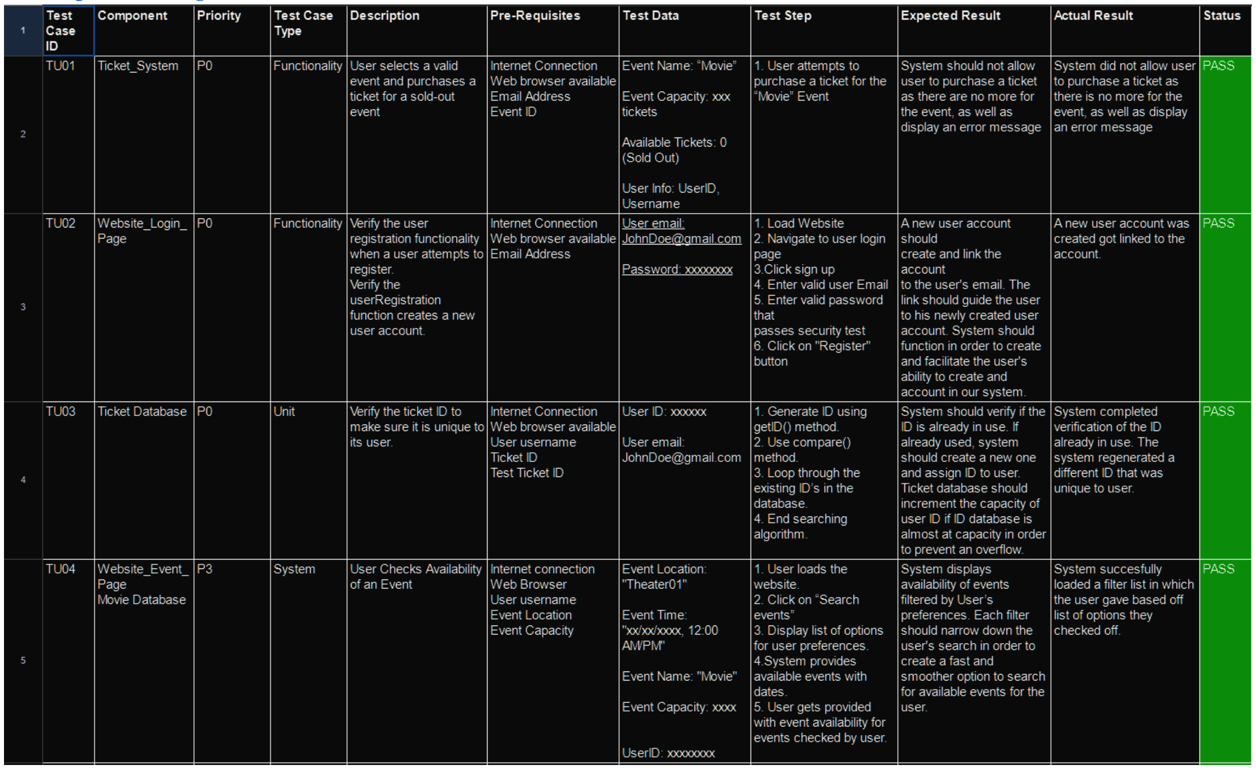
- *Estimated test development time for testing was around 580 hours in preparation for testing various test cases which were crucial for the development of the ticketing system.*

6.2.3 Estimated Test Execution Time

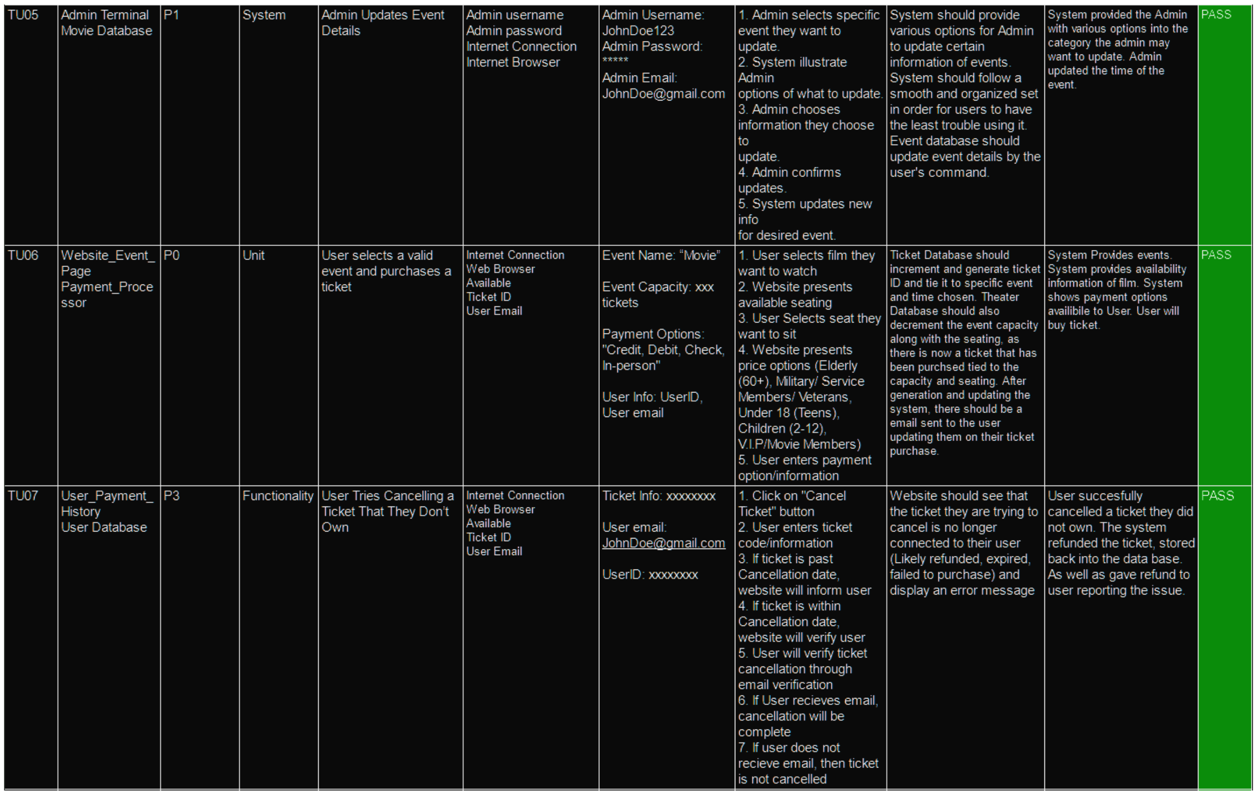
- *Estimated Test Execution time will be estimated to be around 250 hours.*

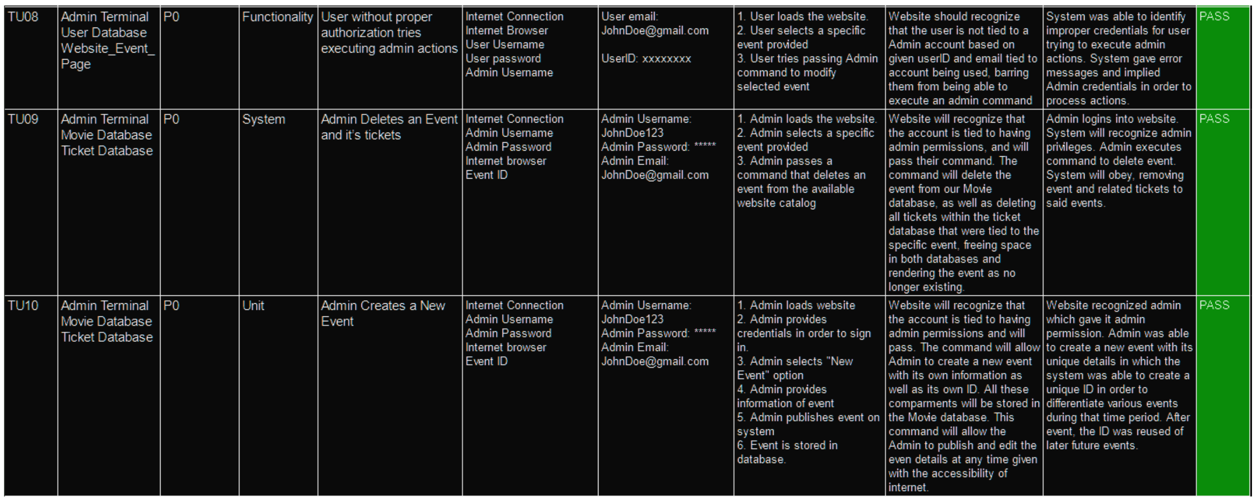
6.3 Test Cases

Please refer to the spreadsheet for our test cases:
https://docs.google.com/spreadsheets/d/1ZXC66__BeCExv0t9JtCvz591pQ4cqt-FMQx3fahwfE4/edit?usp=sharing



1	Test Case ID	Component	Priority	Test Case Type	Description	Pre-Requisites	Test Data	Test Step	Expected Result	Actual Result	Status
2	TU01	Ticket_System	P0	Functionality	User selects a valid event and purchases a ticket for a sold-out event	Internet Connection Web browser available Email Address Event ID	Event Name: “Movie” Event Capacity: xxx tickets Available Tickets: 0 (Sold Out) User Info: UserID, Username	1. User attempts to purchase a ticket for the “Movie” Event	System should not allow user to purchase a ticket as there are no more for the event, as well as display an error message	System did not allow user to purchase a ticket as there is no more for the event, as well as display an error message	PASS
3	TU02	Website_Login_Page	P0	Functionality	Verify the user registration functionality when a user attempts to register. Verify the userRegistration function creates a new user account.	Internet Connection Web browser available Email Address	User email: JohnDoe@gmail.com Password: xxxxxxxx	1. Load Website 2. Navigate to user login page 3.Click sign up 4. Enter valid user Email 5. Enter valid password that passes security test 6. Click on "Register" button	A new user account should create and link the account to the user's email. The link should guide the user to his newly created user account. System should function in order to create and facilitate the user's ability to create and account in our system.	A new user account was created got linked to the account.	PASS
4	TU03	Ticket Database	P0	Unit	Verify the ticket ID to make sure it is unique to its user.	Internet Connection Web browser available User username Ticket ID Test Ticket ID	User ID: xxxxxx User email: JohnDoe@gmail.com	1. Generate ID using getID() method. 2. Use compare() method. 3. Loop through the existing ID's in the database. 4. End searching algorithm.	System should verify if the ID is already in use. If already used, system should create a new one and assign ID to user. Ticket database should increment the capacity of user ID if ID database is almost at capacity in order to prevent an overflow.	System completed verification of the ID already in use. The system regenerated a different ID that was unique to user.	PASS
5	TU04	Website_Event_Page Movie Database	P3	System	User Checks Availability of an Event	Internet connection Web Browser User username Event Location Event Capacity	Event Location: "Theater01" Event Time: "xx/xx/xxxx, 12:00 AM/PM" Event Name: "Movie" Event Capacity: xxxx UserID: xxxxxxxx	1. User loads the website. 2. Click on “Search events” 3. Display list of options for user preferences. 4.System provides available events with dates. 5. User gets provided with event availability for events checked by user.	System displays availability of events filtered by User’s preferences. Each filter should narrow down the user's search in order to create a fast and smoother option to search for available events for the user.	System succesfully loaded a filter list in which the user gave based off list of options they checked off.	PASS





TU05	Admin Terminal Movie Database	P1	System	Admin Updates Event Details	Admin username Admin password Internet Connection Internet Browser	Admin Username: JohnDoe123 Admin Password: ***** Admin Email: JohnDoe@gmail.com	1. Admin selects specific event they want to update. 2. System illustrate Admin options of what to update. 3. Admin chooses information they choose to update. 4. Admin confirms updates. 5. System updates new info for desired event.	System should provide various options for Admin to update certain information of events. System should follow a smooth and organized set in order for users to have the least trouble using it. Event database should update event details by the user's command.	System provided the Admin with various options into the category the admin may want to update. Admin updated the time of the event.	PASS
TU06	Website_Event_Page Payment_Processor	P0	Unit	User selects a valid event and purchases a ticket	Internet Connection Web Browser Available Ticket ID User Email	Event Name: "Movie" Event Capacity: xxx tickets Payment Options: "Credit, Debit, Check, In-person" User Info: UserID, User email	1. User selects film they want to watch 2. Website presents available seating 3. User Selects seat they want to sit 4. Website presents price options (Elderly (60+), Military/ Service Members/ Veterans, Under 18 (Teens), Children (2-12), V.I.P/Movie Members) 5. User enters payment option/information	Ticket Database should increment and generate ticket ID and tie it to specific event and time chosen. Theater Database should also decrement the event capacity along with the seating, as there is now a ticket that has been purchsed tied to the capacity and seating. After generation and updating the system, there should be a email sent to the user updating them on their ticket purchase.	System Provides events. System provides availability information of film. System shows payment options availibile to User. User will buy ticket.	PASS
TU07	User_Payment_History User Database	P3	Functionality	User Tries Cancelling a Ticket That They Don't Own	Internet Connection Web Browser Available Ticket ID User Email	Ticket Info: xxxxxxxx User email: JohnDoe@gmail.com UserID: xxxxxxxx	1. Click on "Cancel Ticket" button 2. User enters ticket code/information 3. If ticket is past Cancellation date, website will inform user 4. If ticket is within Cancellation date, website will verify user 5. User will verify ticket cancellation through email verification 6. If User recieves email, cancellation will be complete 7. If user does not recieve email, then ticket is not cancelled	Website should see that the ticket they are trying to cancel is no longer connected to their user (Likely refunded, expired, failed to purchase) and display an error message	User succesfully cancelled a ticket they did not own. The system refunded the ticket, stored back into the data base. As well as gave refund to user reporting the issue.	PASS
TU08	Admin Terminal User Database Website_Event_Page	P0	Functionality	User without proper authorization tries executing admin actions	Internet Connection Internet Browser User Username User password Admin Username	User email: JohnDoe@gmail.com UserID: xxxxxxxx	1. User loads the website. 2. User selects a specific event provided 3. User tries passing Admin command to modify selected event	Website should recognize that the user is not tied to a Admin account based on given userID and email tied to account being used, barring them from being able to execute an admin command	System was able to identify improper credentials for user trying to execute admin actions. System gave error messages and implied Admin credentials in order to process actions.	PASS
TU09	Admin Terminal Movie Database Ticket Database	P0	System	Admin Deletes an Event and it's tickets	Internet Connection Admin Username Admin Password Internet browser Event ID	Admin Username: JohnDoe123 Admin Password: ***** Admin Email: JohnDoe@gmail.com	1. Admin loads the website. 2. Admin selects a specific event provided 3. Admin passes a command that deletes an event from the available website catalog	Website will recognize that the account is tied to having admin permissions, and will pass their command. The command will delete the event from our Movie database, as well as deleting all tickets within the ticket database that were tied to the specific event, freeing space in both databases and rendering the event as no longer existing.	Admin logins into website. System will recognize admin privileges. Admin executes command to delete event. System will obey, removing event and related tickets to said events.	PASS
TU10	Admin Terminal Movie Database Ticket Database	P0	Unit	Admin Creates a New Event	Internet Connection Admin Username Admin Password Internet browser Event ID	Admin Username: JohnDoe123 Admin Password: ***** Admin Email: JohnDoe@gmail.com	1. Admin loads website 2. Admin provides credentials in order to sign in. 3. Admin selects "New Event" option 4. Admin provides information of event 5. Admin publishes event on system 6. Event is stored in database.	Website will recognize that the account is tied to having admin permissions and will pass. The command will allow Admin to create a new event with its own information as well as its own ID. All these comparments will be stored in the Movie database. This command will allow the Admin to publish and edit the even details at any time given with the accessibility of internet.	Website recognized admin which gave it admin permission. Admin was able to create a new event with its unique details in which the system was able to create a unique ID in order to differentiate various events during that time period. After event, the ID was reused of later future events.	PASS

7. Data Management

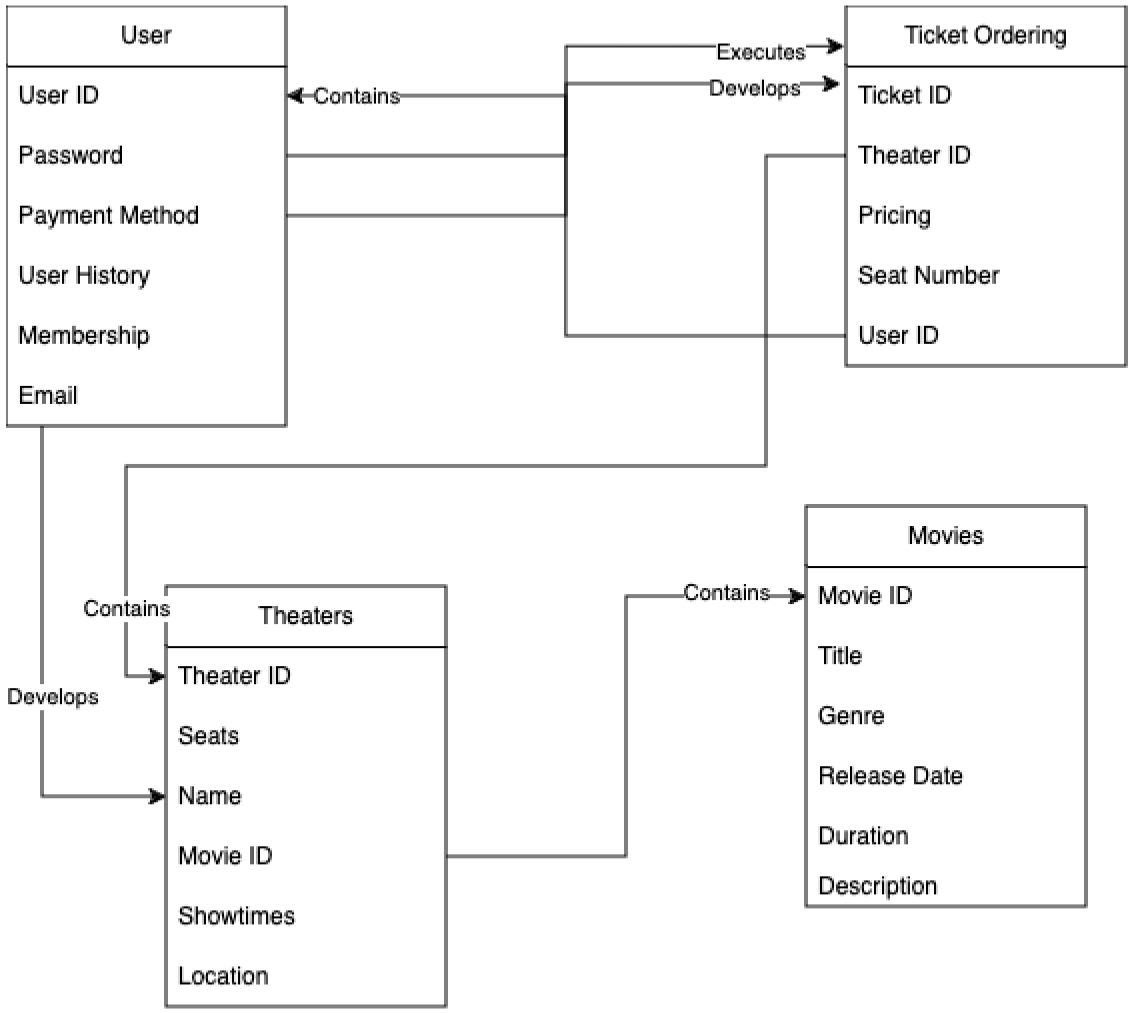
Data management is a necessary part in the process of making sure that our ticketing system is up to par and will be able to handle large volumes of data that is given to it due to having to manage data such as available theaters, ticket volume, movie availability, seating availability, and user profiles.

7.1 SQL Database

7.1.1 SQL Over non-SQL Discussion

We made the decision to use a SQL structure over a non-SQL structure due to multiple factors such as the databases we plan to use along with the necessary requirements that we will have to handle money transactions between the users and the ticketing system. When compared to a non-SQL, a SQL structure is catered towards clear and structured databases with easily differentiable data, which fits our guidelines as our data can easily be organized in tables and be made sense of. SQL is also much more preferred if we are to execute money transactions as it follows ACID guidelines, making it much more capable of handling tasks such as cancelling, booking, and purchasing tickets.

7.1.2 SQL Database Design w/ Entity Relationship



7.1.3 SQL Data Dictionary

Users:

- UserID (Primary Key): Unique identifier for each user.
  Data Type: int
- Email: Email address of the user.
  Data Type: varchar
  Range: 255
- Password: password for user authentication.

Data Type: varchar
Range: 255
- Payment Method: User's chosen payment method when purchasing a ticket
  Data Type: varchar
  Range: 255
- User History: User table containing all tracked transactions linked to the account
  Data Type: List<varchar>
  Range: 255
- Membership: User membership that determines whether they are participating in a rewards program with the ticketing system
  Data Type: varchar
  Range: 255

Movies:
- MovieID (Primary Key): Unique identifier for each movie.
  Data Type: int
- Title: Title of the movie.
  Data Type: varchar
  Range: 255
- Description: Brief description of the movie.
  Data Type: varchar
  Range: 255
- Genre: Category or genre of the movie.
  Data Type: varchar
  Range: 100
- Release Date: Date when the movie was released.
  Data Type: varchar (varchar to date)
  Range: 50
- Duration: Duration of the movie in minutes.
  Data Type: int

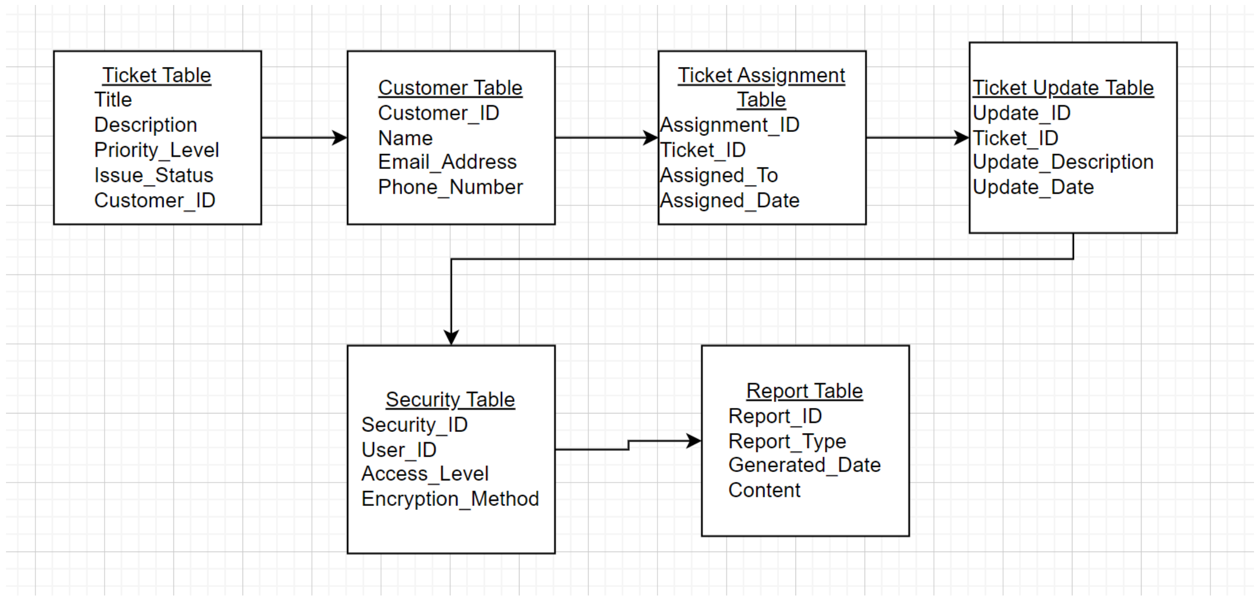
Theaters:
- TheaterID (Primary Key): Unique identifier for each theater.
  Data Type: int
- Name: Name of the theater.
  Data Type: varchar
  Range: 255
- Location: Location or address of the theater.
  Data Type: varchar
  Range: 255
- Seats: Maximum seating capacity of the theater.
  Data Type: int
- Showtimes: Info related to time and date of a movie
  Data Type: varchar (varchar to Date/Time)

Range: 255
- Movie ID (Foreign Key): ID referencing Movies table
  Data Type: int

Ticket Ordering:
- Ticket ID (Primary Key): Unique identifier for each ticket.
  Data Type: int
- Theater ID (Foreign Key): ID referencing the Theater table.
  Data Type: int
- User ID (Foreign Key): ID referencing the Users table.
  Data Type: int
- Seat Number: Seating assigned number related to seating that is at the showtime of ticket purchase
  Data Type: int
- Pricing: Total price of purchasing ticket(s)
  Data Type: int

7.2 Database Organization

7.2.1 Database Diagram



7.2.2 Database Descriptors

- Ticket Creation: Users shall create tickets for reporting issues and flaws they may experience. With each ticket having their own specific identifier in order for better organization.
- Ticket Updates: Users shall be able to receive various updates regarding their status of the issues they reported to the company. Databases may send information towards various clients using their specific Identifier for quicker service.
- Ticket Assignment: Database shall be able to store and assign various tickets regarding the customer's needs. All information should be stored in the system database in which every piece of data needs a specific identifier in order for quicker search.
- Security: Databases should widthold various security measures in order to keep the database safe from outside sources. Security should have various encryptions in which sensitive information regarding customers and databases shall be stored. Access to sensitive data shall be restricted based on the user's roles and permissions.
- Scalability: System should be scalable in order to pair with various outside sources like sending emails to clients. The system's architecture should provide accommodation for customers in order to provide comfort.
-Reports: System should provide reports for users regarding ticket availability, ticket volume and customer satisfaction ratings in order for improving the system with any visible flaws that the system may have.

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

A.1 Appendix 1

A.2 Appendix 2