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.

DEL Ticketing System

Software Requirements Specification

<1.0>

<2/1/2024>

<Group 9>
<Diego Sandoval, Edmund Yin, Liam Colburn>

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

Revision History

Date	Description	Author	Comments
<date>	<Version 1>	<Your Name>	<First Revision>
		Liam Colburn	
		Diego Sandoval	
		Edmund Yin	

Test_Plan_DEL_Ticketing.xlsx

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

This Software Requirement Specification (SRS) provides a document containing a complete overview of the entire SRS by defining ***DEL Ticketing's*** *purpose, scope, definitions, acronyms, abbreviations, and an overall overview of the SRS. The goal of this document is to provide an in-depth analysis of the final* ***DEL Ticketing*** *software system. A list of the detailed specifications that are defined by the programmers, with the purpose of meeting customer requirements and extra wants, are provided in this document.*

1.1 Purpose

This SRS serves as a guide to better understand the main functions of our ticketing system. This document will

The purpose of DEL Ticketing is to provide a ticketing system that not only can be used in the theater industry, but has applications in the sports, and music industries, etc… DEL Ticketing's software is programmed and designed to provide an easy and efficient to use user interface when it comes to purchasing tickets. With functions past just your average ticketing buying system, our software includes other benefits that provide our clients with a more satisfactory experience.

1.2 Scope

Some software products which will be necessary for this system are Host DBMS, Report Generator, Payment Gateway Integration, Authentication and Authorization, etc.

DBMS stands for Database Management System, and is responsible for creating, organizing, and managing databases. It also provides users access to these databases and efficient retrieval of information.

Report Generator is a software tool which is responsible for creating reports based on user parameters and preferences. This allows for the generation of specified reports which ensures user satisfaction.

Payment Gateway Integration is the software API provided by payment gateways, such as PayPal, Stripe, Square, and Apple Pay. This is necessary as it facilitates transactions which are made through the system.

Authentication and Authorization tools include OAuth, OpenID Connect, or JSON Web Tokens (JWT) for secure user login, and authorization mechanisms to control access to different system features.

The scope of the DEL Ticketing System will mainly be orientated towards online sales, specifically those which involve anything related to tickets, whether it be movie tickets, concert tickets, or even sports tickets. The software products will allow for efficient and secure transactions between the system and the customer. We hope to be able to retrieve unique tickets from different databases and send them out to the user within 3 minutes in order to ensure efficiency. We hope to implement the majority of Payment methods securely to achieve a 95% transaction success rate.

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.

API	*Application Programming Interface:* A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.
CRM	*Customer Relationship Management:* The combination of practices, strategies and technologies that companies use to manage and analyze customer interactions and data throughout the customer lifecycle. The goal is to improve customer service relationships and assist with customer retention and drive sales growth.
UI	*User interface:* The means by which the user and a computer system interact.
OAuth	*Open Authorization: An open standard for access delegation, commonly used as a way for internet users to grant websites or applications access to their information on other websites but without giving them the passwords.*
SES	*Simple Email Services: An email platform that provides an easy, cost-effective way for you to send and receive email using your own email addresses and domains.*
DBMS	*Database Management System: DBMS are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.*
PCI DSS	*Payment Card Industry Data Security Standard: An information security standard used to handle credit cards from major card brands. The standard is administered by the Payment Card Industry Security Standards Council, and its use is mandated by the card brands.*

1.4 References

- *Software Life-Cycle Model*

1.5 Overview

The resulting sections of the Software Requirement Specification (SRS) document consist of how our product compares to others, how the product precisely operates, how we accommodate for the user, design limits we may encounter, and different assumptions in regards to the software. Furthermore, in part 3 we discuss the customer's requirements, external interface requirements for our product, along with specific features that will emphasize the ability of our product while utilizing use cases, inverse requirements, design constraints such as hardware limitations, details on how our database will operate, and any other additional requirements. Additionally, part 4 covers different diagrams to really help others visualize our project through the use of analysis models, sequence diagrams, data flow diagrams, and state-transition diagrams. Lastly, in part 5 we discuss how the SRS will be updated when need be and how that functions.

2. General Description

This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.

2.1 Product Perspective

Our project is one that contains many innovative ways to search for your favorite movies, see what movies are trending, look at ratings, purchase tickets quickly, recommend seating options, and much more. While other products of the same kind complete similar tasks, ours does it in a way which is much more efficient for the user, allowing one to find or access what they are looking for in a much quicker manner in comparison to other ticketing products of the same kind, which may only allow you to, for example, find a movie only through looking up it's exact name.

2.2 Product Functions

2.2.1 User Authentication

- *Users will be allowed to securely create accounts in our system, and log into their accounts by putting in their emails and passwords. The passwords must meet a specific requirement for safety purposes. And if the user puts in misinformation, the software won't allow them to log in.*

2.2.2 Searching

- *Users will be allowed to search for movies from a search tab. Our software will offer the option to search based on different tags such as movie name, genre, main actors, and ratings.*

2.2.3 Purchasing Tickets

- *Users will be allowed to purchase tickets online via the virtual "cart". Once clicking on a movie, users will see a list of available seats where they will have the option to choose how many seats they want, and where they want to sit. After choosing tickets based on these, they will automatically be put into their cart, where the tickets will be waiting for purchase. Once the user clicks into the cart, they will be able to put in their credit/debit card information which will be processed by a third party app.*

2.2.4 Order Confirmation and Ticket Retrieval

- *Once a user payment has been processed, an email will automatically get sent out to the email in which the account was authorized under. This email will provide the user with a uniquely generated number alongside a barcode generated based on the order number. This can be shown at the movie theater to get into the movie.*

2.2.5 DEL Ticketing Premier Member

- *Users have the opportunity to purchase a Premier Membership. This premier membership comes with exclusive deals and benefits. A few examples of this are a point reward system that can be traded in for free snacks or tickets in the future.*

2.2.6 Two-Factor authentication

- *Users have the opportunity to sign up for two-factor authentication that provides extra security of their account. This will be done by emailing the email provided to confirm that the person attempting to log in, is indeed the true account holder.*

2.3 User Characteristics

Some of the characteristics of our project available to the user will be:

- *Quick and easy ticket purchasing*
- *Quick and easy ticket purchase canceling*
- *Recommended seating arrangements*
- *Most current popular snacks to consume during the movie*
- *Filtering through movies by genre, actor(s), most popular - least popular*
- *Search bar to look for movies, actor(s), genres*
- *User's own personal list of favorite movies with the rating they gave it (optional inclusion of a message describing their thoughts)*

2.4 General Constraints

Since we are designing a ticketing system to support all types of tickets, we must ensure that we comply with all restrictions of each type of ticket. For example, some companies will request that only 1 ticket may be purchased per person, while others do not have that constraint.

Considering that the DEL ticketing system will be supporting all types of tickets, the system must be highly available and reliable to prevent downtime during critical ticket sales periods to ensure a positive user experience. Constraints related to fault tolerance, redundancy, and disaster recovery planning may influence architectural decisions to enhance system resilience.

2.5 Assumptions and Dependencies

Seeing that we are offering every single type of ticket available, we will also have to ensure that we are up to date with the various venues. Different venues have distinct qualities and limits that can affect ticketing operations. Seating capacity, layout, accessibility, and technical infrastructure (e.g., ticket scanning systems) should all be factored into the criteria to guarantee consistent ticketing experiences across several venues.

Event organizers may need extensive reporting and analytics tools to monitor ticket sales, income, attendance numbers, and customer demographics. WIth this in mind, we may need to implement reporting dashboards, data visualization tools, and export options.

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 carefully organize the requirements presented in this section so that they may be 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 user interface should be extremely user-friendly and easy to navigate. It should also adapt to different screen sizes, such as monitors, phones, iPads, etc. Users should be able to search and browse different events as well as view their own dashboard to see events they are interested in or have purchased tickets to.

3.1.2 Hardware Interfaces

The DEL Ticketing System should implement and be compatible with commonly used ticketing hardware interfaces. This includes the construction of barcodes and QR codes to validate the tickets. Tickets should also be able to be converted to a printable form in case users want a physical ticket/receipt.

3.1.3 Software Interfaces

The DEL Ticketing System should implement external software interfaces to ensure seamless exchange of data between servers and users. Other interfaces include payment gateways to enable secure and efficient payment options for ticket purchasing.

3.1.4 Communications Interfaces

The DEL Ticketing System should implement quick and secure communication interfaces between the system and the users, as well as between the system and event organizers. This will primarily *be done through email services as notifications and confirmations will be sent to the user for new events, confirmation of purchase, and updates on event statuses.*

3.2 Functional Requirements

3.2.1 Purchasing Tickets / Reserving Seats

3.2.1.1 Introduction

- *The DEL Ticketing System will allow users to search and browse through available events, choose specific seats, and pay for tickets. After this, confirmation emails will be sent to the user.*

3.2.1.2 Inputs

- *Information input by user, such as their name and contact information (to receive emails / messages)*
- *Ability to select preferred movie*
- *Select open seats (and see which ones are taken)*
- *Ticket payment (credit card, apple pay, debit card, etc)*

3.2.1.3 Processing

- *The system will first only allow you to select available seats, and gray out taken seats. Upon selecting the seat, and adding it to your cart, the system will reserve your seat for 5 minutes before making it available again. The system will then process your payment through secure and efficient payment gateways. It will then generate a unique barcode/QR code tied to your reserved seat.*
- *The system will not allow users to purchase more than 20 tickets for a single event*
- *The system will not allow users to buy tickets more than 2 weeks in advanced, or longer than 10 minutes into the movie*

3.2.1.4 Outputs

- *The system's primary output will be through emails as it will be sending confirmation emails after purchase of a ticket. You will also receive emails for future events (if you decide to opt-in) and any updates that occur.*
- *You will also receive a printable version of the ticket, if you desire a physical copy.*

3.2.1.5 Error Handling

- *If the user comes across a problem viewing any of the pages, purchasing tickets, viewing seats, or paying; system will display a message emphasizing that the system may be currently unavailable*
- *If there's too many users on the webpage at once, system will display message saying how many other users are ahead of them in terms of priority to enter the webpage*

3.2.2 Premium Ticket Account

Ability to purchase a premium membership, which upgrades a user's account and allows them to access more features and promotional deals

- Can purchase a monthly, 6 month, or yearly subscription
- Membership allows you to connect with other premium members and see other users' preferred movie genres, favorite movies, and allows you to write messages about specific movies that others could be able to see

3.2.2 Customer Feedback

3.2.2.1 Introduction

The DEL Ticketing System customer feedback system allows for users to see what others have said about the system, while also being able to contact help from real people for any reason they may have in order to provide a more user-friendly experience

3.2.2.2 Inputs

- Users will be given a list of recommended issues that they may have encountered, otherwise they can click on “Other” and provide a description of what their problem is
- Users can provide a rating of their experience for others to see, in order to encourage users to stay on the site and know that they will receive appropriate help

3.2.2.3 Processing

Our system holds onto feedback from the users safely, ensuring that only administrators can access it initially.

3.2.2.4 Outputs

Users will get a message sent from our system that notifies them that their feedback has been received and that we will respond promptly

3.2.2.5 Error Handling

If there are any problems with inputting feedback, users will be prompted to fill in the appropriate needed information, and if there’s connection issues, they will be prompted to reload their webpage or that our systems are temporarily unavailable.

3.2.3 Administrator Mode

3.2.3.1 Introduction

- The system will allow Administrators to be able to log in to make direct changes on the webpage, or add messages directly to users when specific functions aren’t working properly. Overall this feature will allow us to manage the system more effectively

3.2.3.2 User Management

- Admins will be allowed to edit, add and delete user accounts
- Admins will be allowed to manually change user registration information

3.2.3.3 Content Management

- Admins with be allowed to edit, add, and delete events on the webpage

3.2.3.4 Backup and Disaster Recovery

- The Admin will be able to schedule and manage system backups
- The admin will have access to tools for disaster recovery
- Backup storage options and retention policies should be configured by the Admin.

3.3 Use Cases

3.3.1 User Registration

- The guest selects the “Sign Up” button.
- The system presents the registration form.
- The guest enters the required information (email and password).
- The System validates the information.
- Once validation is complete, the account is added into the system database.

3.3.2 Booking Tickets

- The user selects the desired movie.
- the system displays the event details and ticket options.
- The user selects the desired ticket(s).
- The system ensures it is not more than 20 in quantity.
- Once confirmed the system calculates and displays the total price.
- The user will confirm the booking by purchasing the tickets.
- The system with process hte payment.
- The system emails the user with an order confirmation once the payment is processed.

3.3.3 Searching For Movies

- The user will enter search criteria. (Movie name, actors, genre, etc…)
- The System displays all movies that fit the search criteria in a list.
- The User can then select their desired event to begin booking their ticket.

3.3.4 Viewing Order History

- The user can navigate to hte “Order History” section
- The system retrieves nad displays all previous orders
- The user can select desired past orders to view order details
- The system presents all order details (Movie name, date, time, quantity, and price)

3.4 Classes / Objects

3.4.1 Tickets

3.4.1.1 Attributes

- Ticket Number: The uniquely generated number of each ticket
- Event Id: Identifies the movie associated with the ticket
- User ID: The ID of the user that purchased the ticket
- Ticket Type: Specifies the type of ticket (Child, Adult, Senior)
- Ticket Price: The Price of each ticket
- Ticket Status: The status of the ticket (Available, purchased)

3.4.1.2 Functions

- GenerateTicketID(): Generates the unique number that identifies a ticket
- SetEventID(eventID): Assigns the ticket to a specific move.
- SetUserID(userID): Assigns the ticket to a specific user
- SetTicketType(ticketType): Assigns a category of ticket to the ticket
- SetTicketPrice(ticketPrice): Sets the price to the ticket
- GetTicketStatus(): Retrieves the status of the ticket
- SetTicketStatus(newStatus): Updates the status of a ticket

References: 3.3.2

3.4.2 User Account

3.4.2.1 Attributes

- UserID: Unique Identifier for each user
- Email: Email identifier of each account.
- Password: Password associated with the Account.

- Phone Number: Phone number used as a 2FA method.
- User Status: The type of account the user has (Regular Account, Premium Account)
- Previous Order: List of Previous Orders on past events under the account
- Current Orders: List of orders for events that haven't yet happened

3.4.2.2 Functions

- GenerateUserID(): Generates a unique userID for each user
- SetEmailAddre]ess(email): Sets the email address of the user's account
- SetPassword(password): Sets and encrypts the password for the user's account (will be decrypted when checking login credentials)
- GetAccountStatus: (Retrieves the status of the account)
- UpdateAccountStatus(newStatus): Updates the status of the user in the event of upgrade/downgrade in their account type.
- GetPreviousOrders(PreviousOrders): Retrieves and displays a list of all past orders to be accessed by the user
- GetCurrentOrders(CurrentOrders): Retrieves and displays a list of current orders to be accessed by the user
-

References: 3.3.1

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 DEL Ticketing system will be based on its webpage. The webpage would be accessed through a web browser.

- The time that this product will take to load will depend entirely on the strength of the internet connection that the user has
- Loading times will also depend on the media from where the product was accessed.

3.5.2 Reliability

- The system should have a fault tolerance of at least 99.95%, ensuring minimal service disruptions in case of hardware or software failures.
- Data integrity must be maintained with a maximum data loss of 0.05% in case of system failure.

3.5.3 Availability

- The system will be up for 24 hours a day, 7 days a week. With a planned downtime at the start of every month for around 1 hour at the most for maintenance purposes.
- Any unplanned downtime, will result in immediate notice of all users that have a profile with Del Ticketing.

3.5.4 Security

- We will ensure security and privacy of user data through encryption using the AES-256 encryption standard.
- Authorization and authentication features will be implemented as well through protocols such as OAuth.
- Passwords will be expected to include letters, numbers, as well as special characters to secure user accounts.

3.5.5 Maintainability

- Code maintainability index will be upheld to a standard of above 85. This will make sure that changes and adjustments can be made easily.
- Documentation of the code will also be comprehensive and updated frequently, to ensure readability of understanding of complicated sections of code.

3.5.6 Portability

- DEL Ticketing will be available via a web page, making it very easy for users to access nearly anywhere they want, so long as they have access to the internet.
- Deployment

3.6 Inverse Requirements

The system should facilitate event management, ticket sales, and attendee communication for scheduled events. Inverse Requirement: The system should handle event cancellations or postponements quickly by providing timely notifications to affected customers and processing refunds efficiently.

–

STOP HERE FOR NOW

–

3.7 Design Constraints

3.7.1 Hardware Limitations
The software's design must accommodate for possible hardware limitations. Such as memory constraints, processing power, and network bandwidth.
3.7.2 Scalability Requirements
Our design must be able to handle and increase in volume in active users, transactions and other data all at once without it significantly impacting our systems performance.
3.7.3 Budget and Resource Constraints
Our system must not overstep the budgets set to develop the system, throughout the development lifecycle.
3.7.4 Security Constraints
Our System must be able to uphold security requirements through; data encryption, 2-Factor Authentication, password requirements, and other administrative capabilities

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.

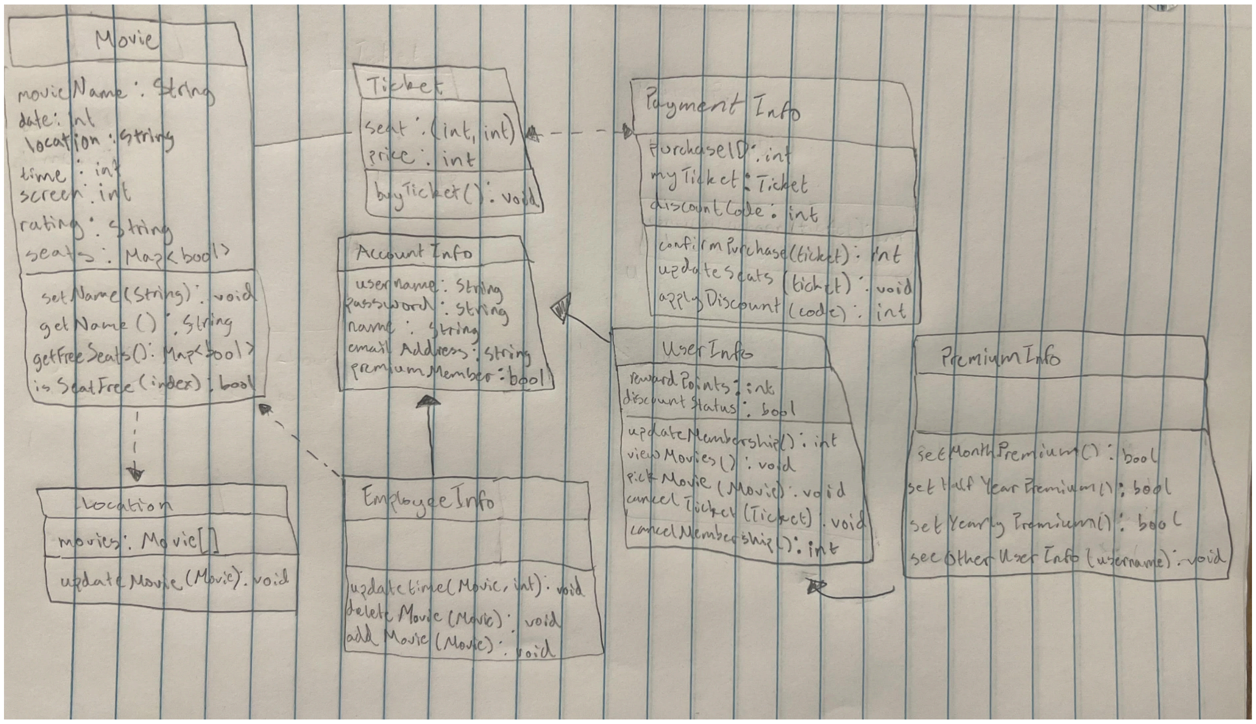
3.9 Other Requirements

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.

4.1 UML Diagram

- 

Description:

- *The "Movie" class represents a movie showing at a particular location, date, and time. It stores information such as the movie name, date, time, screen number, rating, location, and the seating arrangement. The seating arrangement is represented using a map where each seat index is associated with a boolean value indicating whether the seat is occupied or vacant. The class provides methods to set and retrieve the movie name, get information about the free seats, and check whether a specific seat is free. These operations facilitate managing and querying information related to the movie and its seating arrangement.*
- *The "Ticket" class represents a ticket for a movie showing. It holds information about the seat number, row number, and the price of the ticket. The seat information is stored as a*

tuple (int, int) where the first integer represents the row number and the second integer represents the seat number within that row.

- *The "Account Info" class holds all the information relevant to the user. It ensures that the user is able to access their account through their email and password, as well as check whether or not they are a premium member.*
- *The "UserInfo" class represents information and actions related to a user's account within a movie ticketing or loyalty program system. It holds details such as the user's reward points and discount status, along with operations to manage their loyalty program, view available movies, select a movie, and cancel a ticket.*
- *The "PaymentInfo" class manages information and actions related to the payment and purchase of tickets within a movie ticketing system. It includes attributes to track purchase and transaction IDs, as well as the purchased ticket itself.*
- *The "PremiumInfo" class manages premium membership-related functionalities within a movie ticketing or subscription-based service system. It provides operations to set premium membership for different durations and to view information about other users.*

Attributes:

- *movies: Movie[]*
  - *Represents the array of movies available at certain locations, or upon search*
- *movieName: String*
  - *Represents the name of the movie.*
- *date: int*
  - *Represents the date of the movie showing.*
- *time: int*
  - *Represents the time of the movie showing.*
- *screen: int*
  - *Represents the screen number where the movie is being shown.*
- *rating: String*
  - *Represents the rating of the movie (e.g., PG, PG-13, R).*
- *location: String*
  - *Represents the location where the movie is being shown.*
- *seats: Map<bool>*
  - *Represents the seating arrangement for the movie, where the keys represent seat indices, and the values represent whether the seat is occupied (true) or vacant (false).*
- *seat: (int, int)*
  - *Represents the seat number and the corresponding row number for the ticket.*
- *price: int*
  - *Represents the price of the ticket.*
- *purchaseID: int*
  - *Represents the unique number generated for each ticket upon a purchase made*
- *myTicket: ticket*
  - *Represents any purchased ticket that is about to be purchased*

- *discountCode: int*
  - *Represents the unique integer that users can enter to get a discounted ticket price*
- *rewardPoints: int*
  - *Represents the amount of points gained, points are racked up upon purchases*
- *username: String*
  - *Represents the unique username attached to a specific account*
- *password: String*
  - *Represents the string that clients make to secure their account*
- *name: String*
  - *Represents the name of the account holder*
- *emailAddress: String*
  - *Represents the email address that is tied to the account upon sign up*
- *premiumMember: bool*
  - *Represents whether or not the user is a premium member*
- *rewardPoints: int*
  - *Represents the reward points accumulated by the user.*
- *discountStatus: bool*
  - *Represents whether the user has a discount status (true if the user has discount status, false otherwise).*
- *purchaseID: int*
  - *Represents the unique identifier for the purchase.*
- *transactionID: int*
  - *Represents the unique identifier for the transaction associated with the purchase.*
- *myTicket: Ticket*
  - *Represents the ticket purchased by the user.*
  -

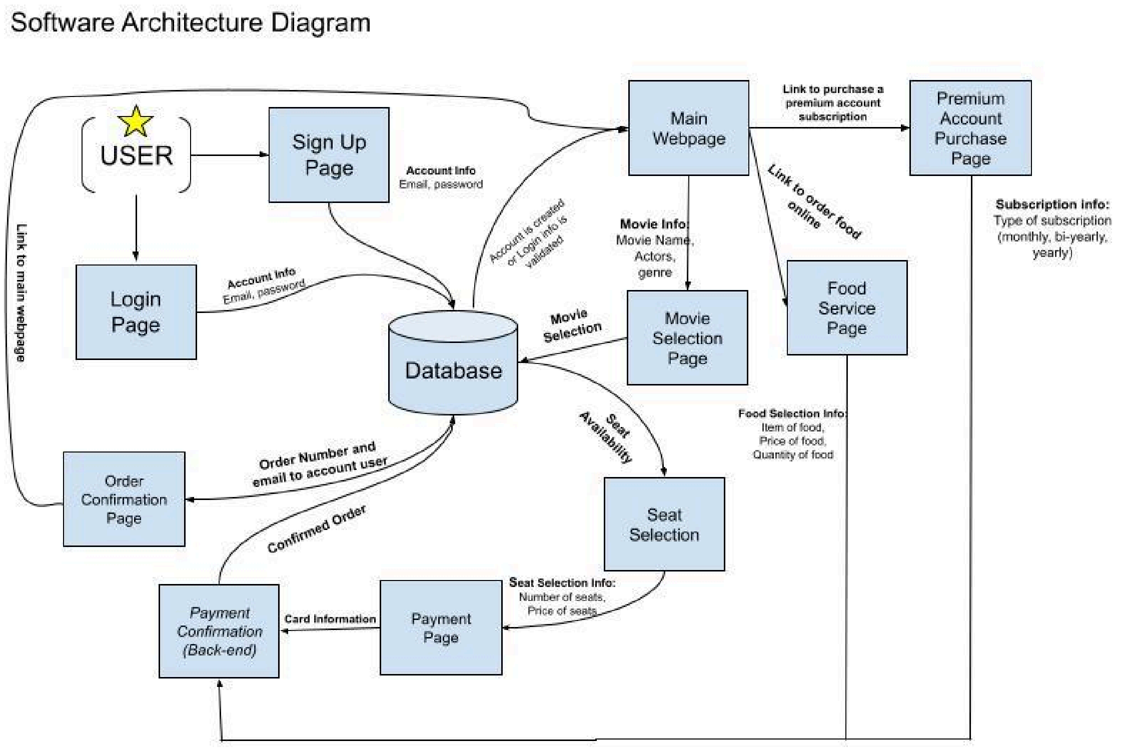
Operations:

- *setName(String): void*
  - *Sets the name of the movie to the provided string.*
- *getName(): String*
  - *Retrieves the name of the movie.*
- *getFreeSeats(): Map<bool>*
  - *Retrieves a map indicating the availability of seats, where the keys represent seat indices, and the values represent whether the seat is free (true) or occupied (false).*
- *isSeatFree(index): bool*
  - *Checks whether the seat at the specified index is free. Returns true if the seat is vacant; otherwise, returns false.*

- *buyTicket(): void*
    - *Represents the action of purchasing the ticket.*
- *setMonthlyPremium(): bool*
    - *Confirms that the user is buying the monthly subscription*
- *setHalfYearPremium(): bool*
    - *Confirms that the user is buying the six-month subscription*
- *setYearlyPremium(): bool*
    - *Confirms that the user is buying the yearly subscription*
- *confirmPurchase(ticket): int*
    - *returns the price of the ticket*
- *updateSeats(Ticket, Movie): void*
    - *Confirms the ticket availability as unavailable from the movie*
- *applyDiscount(discount): int*
    - *returns the discounted amount based on the discount code inputted*
- *updateLoyalty(int, int): int*
    - *Updates the loyalty program of the user with the provided parameters (e.g., adding points, updating discount status) and returns the updated reward points.*
- *viewMovies(): void*
    - *Displays available movies to the user.*
- *pickMovie(Movie): void*
    - *Allows the user to select a movie from the available options by passing the chosen Movie object.*
- *cancelTicket(Ticket): void*
    - *Cancels a previously booked ticket by passing the corresponding Ticket object.*
- *confirmPurchase(ticket): int*
    - *Confirms the purchase of the provided ticket and returns the purchase ID.*
- *updateSeats(ticket): void*
    - *Updates the seating arrangement based on the purchased ticket.*
- *applyDiscount(code): int*
    - *Applies a discount specified by the provided code and returns the updated purchase amount.*
- *setMonthPremium(): bool*
    - *Sets the premium membership for one month and returns a boolean value indicating whether the operation was successful.*
- *setHalfYearPremium(): bool*
    - *Sets the premium membership for six months and returns a boolean value indicating whether the operation was successful.*
- *setYearlyPremium(): bool*
    - *Sets the premium membership for one year and returns a boolean value indicating whether the operation was successful.*

- *seeOtherUserInfo(username): void*
    - *Allows viewing information about other users by providing their username.*

4.2 SWA Diagram


Software Architecture Diagram
USER
Sign Up Page
Account Info Email, password
Main Webpage
Link to purchase a premium account subscription
Premium Account Purchase Page
Subscription info: Type of subscription (monthly, bi-yearly, yearly)
Link to order food online
Account is created or Login info is validated
Movie Info: Movie Name, Actors, genre
Login Page
Account Info Email, password
Link to main webpage
Database
Movie Selection
Movie Selection Page
Food Service Page
Food Selection Info: Item of food, Price of food, Quantity of food
Seat Availability
Order Number and email to account user
Order Confirmation Page
Confirmed Order
Seat Selection
Seat Selection Info: Number of seats, Price of seats
Payment Confirmation (Back-end)
Card Information
Payment Page

Description: This Software Architecture Diagram provides an easier-to-follow guide on the flow of the system through the point of view of a client using our software. It shows the structural component of the webpage. Covering how our system interacts from software to hardware to create an easy user interface, which ultimately leads to a good user experience. It covers the key components of DEL Ticketing such as the following; web page functionality, SQL database functionality, login/signup page functionality, and payment processing. Users will open our webpage and be automatically led to the homepage, here users will have the option to either sign into their account, or sign up to make an account. Regardless of whether they sign up or sign in, the software takes their account details, and either creates it by putting it into the SQL database, or confirming information by accessing the database. Once an account is confirmed, users are automatically sent to the webpage. From here they can do three things; go towards the Movie Selection page, Food Service page, or purchase a premium membership which allows for many exclusive benefits. When a movie is selected, that information will be stored in the SQL database, leading to the user being able to choose where they are seated. Once chosen, they are directed to a payment page, then must confirm their payment, and that confirmed order will lead them to be submitted to the SQL database as well. In return, users will be forwarded to the order confirmation page. When continuing on from the Food Service page and the Premium Account purchase page, users will be directed to the payment confirmation page, where the same process will occur as before.

UPDATED DESIGN SPECIFICATION (3/14):

1. Movie Selection Page - This is done by adding a movie name, main actors, genre, etc… to the search bar. From there the software accesses the database to retrieve the movie of choice, and retrieves seat selection options which is then shown on screen.
    a. From the Seat selection Page, the user will choose a specified number of seats, which that information alongside the price of the tickets is then transferred to the payment page.
2. Food Service Page - The user has the option to click on the online food ordering service button to be able to purchase and select food that will be ready for them by the time they get to the theater, from this page, the items and prices of specific food is transferred to the payment page for purchasing.
3. Premium Account Page - Users will be subject to our advertisements of a premium account, this page gives the users a description of the benefits of a premium account, and users can then choose a monthly, bi-yearly or yearly account. The subscription chose is then sent to the payment page for purchasing

On the Payment Page, any information transferred there will be put into an API, and users will be able to put in their crest card information. Once a user purchased their items, whether it's food, tickets, or a premium account, the system will use the API to complete and confirm the purchase. Once a purchase is verified the users are guided to an order confirmation page, then the only place from there is a link to the main webpage. That concludes the full ride-through of the software.

4.3 Development Plan and Timeline

Diego Sandoval: UML Diagram
Edmund Yin: Description of UML diagram
Liam Colburn: SWA Diagram and Description

4.4 Test Plan Document & Test Cases

LINK TO TEST PLAN DOCUMENT

LINK TO TEST CASES

5. Change Management Process

First, only an exclusive and select group of people will be able to make changes. These people may include project managers, testers, developers, and major stakeholders. They must submit a request for any changes and this will further be reviewed by our project management team. The requested changes will be evaluated to see if such changes will be necessary. Once approved, our development team will implement the changes based on urgency and significance of the request. Next, documentation will be written down in order to keep track of what changes were made and when. Finally version control will be updated and the system will be reset to ensure that the changes take place.

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.

6. Data Management Strategy

Our data management strategy primarily involves SQL databases for their robustness and relational capabilities. We've opted for a single centralized database to streamline data access and ensure consistency across the platform, reducing the complexity of data synchronization and maintenance. The database is logically split into tables for users, events, tickets, feedback, and administrative functions, facilitating efficient data retrieval and manipulation. While a single database simplifies management, it may pose scalability challenges in the long term, though this is mitigated by careful schema design and optimization. Additionally, regular backups and disaster recovery protocols are implemented to safeguard against data loss. Overall, our decision balances simplicity and robustness to ensure effective data management for our ticketing system.

6.1 Justifications and Tradeoffs

- *6.1.1: Single Database vs. Multiple Databases:*
    - *Chosen Approach: The decision to use a single database was made to simplify data management, reduce overhead in maintenance, and ensure data consistency across the system.*
    - *Tradeoffs: While a single database simplifies management, it may lead to scalability challenges in the long run. However, with proper optimization and scaling techniques, these challenges can be mitigated.*

- *6.1.2: SQL vs. NoSQL:*
    - *Chosen Approach (SQL): SQL databases were chosen for their strong consistency, ACID compliance, and familiarity among developers.*
    - *Tradeoffs: SQL databases provide a structured and reliable data model but may require more upfront schema planning. NoSQL databases offer flexibility but may sacrifice some transactional guarantees and consistency.*

- *6.1.3: Data Security:*
    - *Chosen Approach: Strong encryption and access control mechanisms were chosen to ensure data confidentiality and integrity.*
    - *Tradeoffs: Implementing robust security measures may add complexity to the system and incur performance overhead. However, the tradeoff is necessary to protect sensitive user information and maintain trust in the platform.*

6.3 Quality Assurance

- *Chose Approach: MySQL is a renowned database that is well known for its reliability and stability that has high capabilities to handle high-traffic websites. Ensuring top quality data management for the web page, while also having high levels of scalability as DEL Ticketing grows*

6.5 Privacy and Security

- *6.5.1: Encryption of Payment Data*
  Implement encryption protocols to secure payment information (Card numbers, CVV code, expiration dates)
- *6.5.2: Fraud Detection and Prevention*
  Implement Fraud detection algorithms to identify fraudulent activities write

6.6 Scalability

- *6.6.1: Vertical Scalability:Initially, vertical scaling will be employed to accommodate growing data and user loads by upgrading server resources such as CPU, RAM, and storage.*
- *6.6.2: Horizontal Scalability: As the system grows, horizontal scaling techniques such as sharding or partitioning may be considered to distribute data across multiple servers for improved performance and scalability.*

A.1 Appendix 1

A.2 Appendix 2