<Theater Ticketing System>

Software Requirements Specification

<Version 1.1>

<10/05/2023>

<Group 10>
<Adam Graves>
<Leo Sullivan>
<Mark Truong>

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>
9/17	V1.0	Mark Truong	Requirements Specification
9/17	V1.0	Adam Graves	Requirements Specification
9/17	V1.0	Leo Sullivan	Requirements Specification
10/5	V1.1	Mark Truong	Software Design Specification
10/5	V1.1	Adam Graves	Software Design Specification
10/5	V1.1	Leo Sullivan	Software Design Specification

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	
	Cole Simpson	TA, CS 250	

Table of Contents

1. Introduction

The introduction of this document is to articulately define and describe the Software Requirements Specification (SRS). This will provide not only an overview of the SRS but will go into detail of the purpose, scope, definitions, acronyms, abbreviations and references that are needed to be understood within the SRS. The goal of the document is to analyze and give insight to build Sullivan's Theater Ticketing System (STTS).

1.1 Purpose

The purpose of this document (SRS) is to implement and design the functionality needed to run a Theater Ticketing System software in the browser and the requirements to meet the consumers needs. It will give a detailed overview of the product itself and how the audience sees its functionality as a whole.

1.2 Scope

The purpose of Sullivan's Theater Ticketing System is to streamline theater ticket management and develop a user-friendly and convenient application for customers who are looking to purchase theater tickets. This system will use a non-relational database to handle ticket management and the reservation process. Our infrastructure will have a robust database server that supports numerous significant theater venues across many locations, accommodating an array of theater events. Our primary objective is to develop a seamless user experience along with competitive ticket pricing to cater to the customer.

1.3 Definitions, Acronyms, and Abbreviations

- (STTS) Sullivan's Theater Ticketing System
- (SRS) Software Requirements Specification
- (IMAX) Image Maximum
- (MERN) MongoDB, Express, React, Node

1.4 References

https://www.mongodb.com/docs/relational-migrator/diagrams/understand-diagrams/#std-label-rm-understand-diagrams

1.5 Overview

The product STTS will be a user-friendly web-based movie theater ticket system and its' SRS following this overview shall go into detail of the general description of the system as a whole, user characteristics, functional requirements and more.

2. General Description

2.1 Product Perspective

Sullivan's Theater Ticketing System incorporates a database system that holds the following data:

- **Theater Event Details:**
  It covers information for theater events, the event's date and venue, show date and time, available seats, and pricing.

- **Customer Profiles:**
  It maintains records of customers, their personal information such as name, address, and contact numbers. These details serve in the case for emergency situations and communication purposes.

- **Ticketing Reservations:**
  It covers data for ticket reservations, customer specifics, reservation codes, event identifiers, booking dates, and the theater performance date.

2.2 Product Functions

This product is a web service that allows users to browse, search for, and purchase movie tickets at participating theaters.

Functions include:

1. Display of a comprehensive list of theaters and currently playing movies, alongside movie information such as movie titles, showtimes, screening formats, genres, ratings, trailers, and release dates.
2. Offer information regarding theaters, including movies screened, addresses, contact information, and available facilities.
3. Seat selection and ticket purchase through interactive seating charts. Available and unavailable seats will be shown to users.
4. Pricing and accompanying discounts for seats and screenings, including promotions, matinee, child, senior, and military discounts.

2.3 User Characteristics

Users of Sullivan's Theater Ticketing System can be categorized into two groups: Customers and Employees, each with different privileges and functionalities to their roles in the system. Customers have standard access to the system, focused on ticket reservations and other related functions. Employees have elevated access, where they can manage both customer-related functions and theater event management. Managers have administrative access where they significantly modify the system such as adding venues and voiding tickets.

- **Customers Functions:**

    - Make a new Reservation
    - Confirmation for reservations
    - Cancellation for reservations
    - View ticketing event itinerary
    - Account creation(Sign in)
    - Password Reset
    - Movie research by Actor, certain rooms(IMAX), genre based
    - View history of movies seen from account
    - Contact support
    - Ability to leave customer feedback
    - Print Tickets or Email/Digital Wallet Tickets

- **Employees Functions:**
    - Retrieve Customer Reservations:
        - fetch customer list who booked tickets for theater event
    - Retrieve Theater Event Schedule:
        - schedule of theater events, showtimes, locations, available seats
    - Retrieve On-Time/Delayed Events:
        - list of theater events based on time or delayed status
    - Calculate Total Sales:
        - calculate total revenue from ticket sales for a theater event

- **Administrative Functions (Manager):**
    - Add/Delete Theater Event
    - Void/Refund Ticket
    - Add New Theater Venue
        - can add new theater venues to database
    - Update Ticket pricing
    - Reviews feedback activity
    - Manage Event Seating
        - manage seating arrangements for different theater events
    - Update Event Details
        - title, date, time, descriptions

It is crucial to have a clear distinction between customers, employees, and managers, ensuring each group has the necessary functions to facilitate their roles

2.4.1 Operating Environment

The operating environment for Sullivan's Theater Ticketing System is listed below

- Distributed database
    - data storage, retrieval, and synchronization between database nodes
- client/server system

- Operation system: Any browser
- Database: MongoDB
    - data storage, indexing, and queries
- Development platform: Typescript/Express.JS/React.JS/Node.JS
    - constraints: influences languages, frameworks, and tools in development process

2.4.2 General Constraints

1. Global, Fragmentation, and Allocation Schema
    a. structure of database: data partitioning, allocation of data sharding to different chunks
2. MongoDB Commands:
    a. Database commands for queries and applications will be provided, determining how data is retrieved, modified, and managed in the distributed database
3. Response Generation for Global Queries:
    a. sends queries to relevant shards based on global query, collects results from each shard and aggregates them forming a global response
4. Implement the database using a Central Database Management system
    a. constraints affect database design, query optimization, data storage, and management

With these constraints we understand the limitations and requirements of the design and implementation of STTS. We have designed a distributed database

2.5 Assumptions and Dependencies

Let us assume this is a distributed theater ticketing system used in the following application

- Ticket Printing and Delivery: If physical tickets are given, the ticketing system can generate and deliver printed tickets as required. If not, this impacts customer experience
- Event Capacity Limits: The ticketing system follows venue capacity limits and regulations. Any changes in capacity limits will affect ticket sales and event attendance

These assumptions and dependencies depend on performance metrics on the website and will affect the quality of the user experience. Monitoring and optimization is important to maintain a high user experience.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

- GUI
  The user interface for this program consists of custom developed web pages. There is a homepage where users can search through the movie database with user-chosen criteria.

There is a separate, optional user login page for users that have or wish to create an account. Lastly, is a cart and accompanying checkout page to finalize purchases.

- API
  User accounts will connect with two company database API's. One is for the webpage when users log into, will connect with a movie database of partnered theaters showcasing times, actors, and theater rooms(IMAX). The second one will be for when the user enters their account information to confirm with the backend that they entered the correct information to log in.
- Diagnostics
  There is a help desk AI to help users with basic questions. In addition, there is a help desk support team to escalate user issues.

3.1.2 Hardware Interfaces

There are cloud servers that host the webpage and associated databases. User end web page access will be through user hardware.

3.1.3 Software Interfaces

The web page will access our cloud servers and associated databases. User login will have the web API communicate with our cloud servers. This process is built into the user interface.

3.1.4 Communications Interfaces

The operating system will handle communication between web pages and associated shared databases.

3.2 Functional Requirements

3.2.1 Home Page

The home page allows users to search for theaters and movies through a search bar. The home page will also have a row of the most recently released movies, a featured movie row for sponsored movies, and a row for theaters near users that allow location access.

3.2.2 Account Creation

Users shall have the option of creating an account associated with an email. The user account must store billing information and have the option of receiving promotional emails.

3.2.3 Search Function

Users will be able to search for movies or theaters. When searching for theaters, users can search by city, state, or zip code. When searching for movies, users can search by movie name or genre. User searches return the relevant result of a movie or theater. The search will comb the database for either a movie or theater that best matches user input search criteria. For both movies and theaters, a row of dates starting from the current date through the latest date of movie releases for that theater is shown. An additional row to filter movies will include the all, IMAX, standard, and Dolby criteria is available. Movies in the chosen category will be displayed in a list with the movie poster, rating, movie length, closed captioning availability, and showtimes. Participating theaters in order of shortest distance will be listed below.

3.2.4 Movie Synopsis
Each movie page will have a short synopsis of the movie.

3.2.5 Trailers
If a movie is chosen, the movie poster and trailer is shown. The trailer will be embedded in the webpage with autoplay disabled.

3.2.6 Reviews
Aggregated professional reviews will be available for each movie alongside user submitted reviews. A short blurb for professional reviews will be available on site with a link to the original review. User submitted reviews will have a 500 word limit and a rating from 0.5 to 5 stars.

3.2.7 Purchase Tickets
Users after selecting a movie and purchasing their tickets will be sent a confirmation email that includes the itinerary and receipt that includes the price of the type of ticket that was purchased. This will also be processed and stored in our customer database to update the user's movie history. Users must only be allowed to buy 20 tickets max at a time. In addition to this, tickets are only available for purchase 2 weeks prior to showtime and 10 minutes after showtime starts.

3.2.8 Purchase Confirmation
Users will have the option to print emailed ticket(s), scan them via QR code in the order confirmation emails, or save the QR code to a digital wallet.

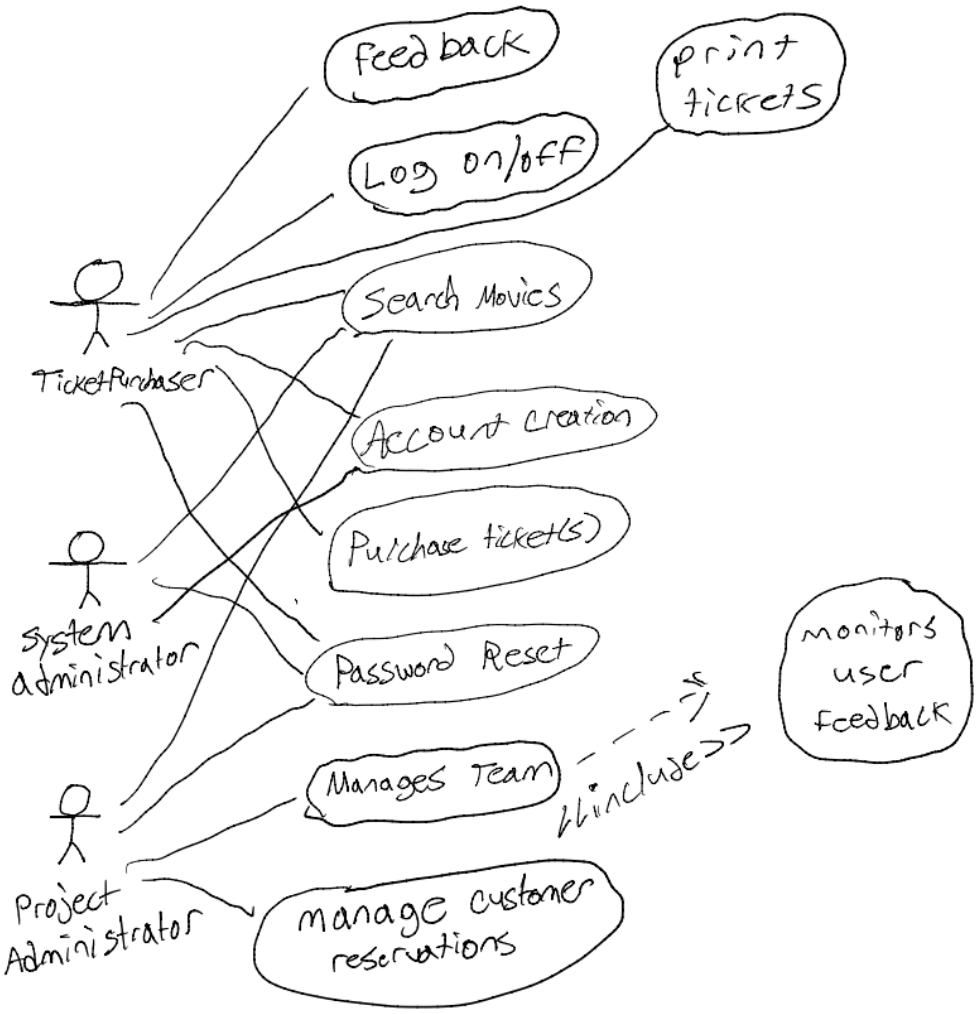
3.2.9 User Feedback
Users will have the option to leave feedback on site functionality and user experience.

3.2.10 Refund Purchase
Users will be allowed to refund tickets for full purchase price up to and until one hour before showing. Users that wish to change their order will need to cancel their order before placing a new order.

3.2.11 Error Handling
In the event the user enters their account information incorrectly, they will be prompted a message of how many attempts they have left and will see a link to reset their password via their personal email. If the user purchases the wrong ticket(s), they will have the option to contact our customer service to void and refund their ticket(s) altogether barring they have not already scanned the tickets at the theater.

3.3 Use Cases

The use cases for the following will have three actors/roles named TicketPurchaser, SystemAdministrator and ProjectAdministrator. Listed below are 10 ideal use cases that will diagram a basic flow that the TicketPurchaser, System Administrator and Project Administrator shall go through when accessing our ticketing theater system.



Name of Use Case: Ticket Purchasing
Actor: TicketPurchaser

3.3.1 Use Case #1

Account Creation

Actor(s): TicketPurchaser

Flow of events:

- TicketPurchaser connects to the Sullivan's Theater Ticketing System (STTS) web server via account creation.

- TicketPurchaser is prompted to create an account via email and password. System sends a confirmation email when the account is created.

- STTS server checks if TicketPurchaser entered correct account name and password, process authentication and if necessary an error warning if information was entered incorrectly.

3.3.2 Use Case #2

Search Movies

Actor(s): TicketPurchaser, SystemAdministrator, ProjectAdministrator

Flow of events:

- TicketPurchaser, SystemAdministrator and ProjectAdministrator searches movies from a list of options and their respective itineraries.

- TicketPurchaser will be able to select what type of room (IMAX).

3.3.2 Use Case #3

Purchase Ticket(s)

Actor(s): TicketPurchaser

Flow of events:

- TicketPurchaser selects how many ticket(s) they would like to purchase.

3.3.2 Use Case #4

Log on/off

Actor(s): TicketPurchaser

Flow of events:

- TicketPurchaser when logged into their account must have the option to log out and in.

3.3.2 Use Case #5

Print Ticket(s)

Actor(s): TicketPurchaser

Flow of events:

- When a purchase is complete, STTS server prompts TicketPurchaser if they would like to print the tickets, scan them via QR code on the email confirmation or save the QR code in a digital wallet.

3.3.2 Use Case #6
Password Reset

Actor(s): TicketPurchaser, SystemAdministrator, ProjectAdministrator

Flow of events:

- System Administrator and Project Administrator shall have access to customer itinerary and information if TicketPurchaser contacts with questions or needs assistance with troubleshooting such as a password reset.
- TicketPurchaser shall have the option to click a password reset button before logging into their account.

3.3.2 Use Case #7
Manages Team

Actor(s): ProjectAdministrator

Flow of events:

- ProjectAdministrator shall have access to customer feedback activity and system administrator activity to assist any escalated inquiries.

3.3.2 Use Case #8
Manage Customer Reservations

Actor(s): ProjectAdministrator

Flow of events:

- ProjectAdministrator shall have the option to edit customer reservations upon escalated inquiries.

3.3.2 Use Case #9
Feedback

Actor(s): TicketPurchaser

Flow of events:

- TicketPurchaser shall have the option to leave feedback on the system or contact customer support.

3.3.2 Use Case #10
Monitors Users Feedback

Actor(s): ProjectAdministrator

Flow of events:

- Project Administrator shall have access to customer feedback activity and system administrator activity to assist any escalated inquiries.

Flow of events:

- TicketPurchaser connects to the Sullivan's Theater Ticketing System (STTS) web server via account creation.

- TicketPurchaser is prompted to create an account via email and password. System sends a confirmation email when the account is created.

- STTS server checks if TicketPurchaser entered correct account name and password, process authentication and if necessary an error warning if information was entered incorrectly.

- TicketPurchaser searches movies from a list of options and their respective itineraries.

- TicketPurchaser selects how many ticket(s) they would like to purchase.

- When a purchase is complete, STTS server prompts TicketPurchaser if they would like to print the tickets, scan them via QR code on the email confirmation or save the QR code in a digital wallet.

- TicketPurchaser shall have the option to leave feedback on the system or contact customer support.

- System Administrator shall have access to customer itinerary and information if TicketPurchaser contacts with questions or needs assistance with troubleshooting such as a password reset.

- Project Administrator shall have access to customer feedback activity and system administrator activity to assist any escalated inquiries.

- TicketPurchaser logs out.

Entry Conditions:

1. TicketPurchaser must have proper authentication credentials to access their account through the STTS server to obtain movie itineraries and ticket purchase options.

2. Computing requirements: supported browser since this is a web platform and not an app based platform.

3.5 Non-Functional Requirements

Some Quality Attributes
Availability
Interoperability
Modifiability
Performance
Security
Testability
Usability

Other quality attributes include portability, scalability, deployability, mobility, safety.

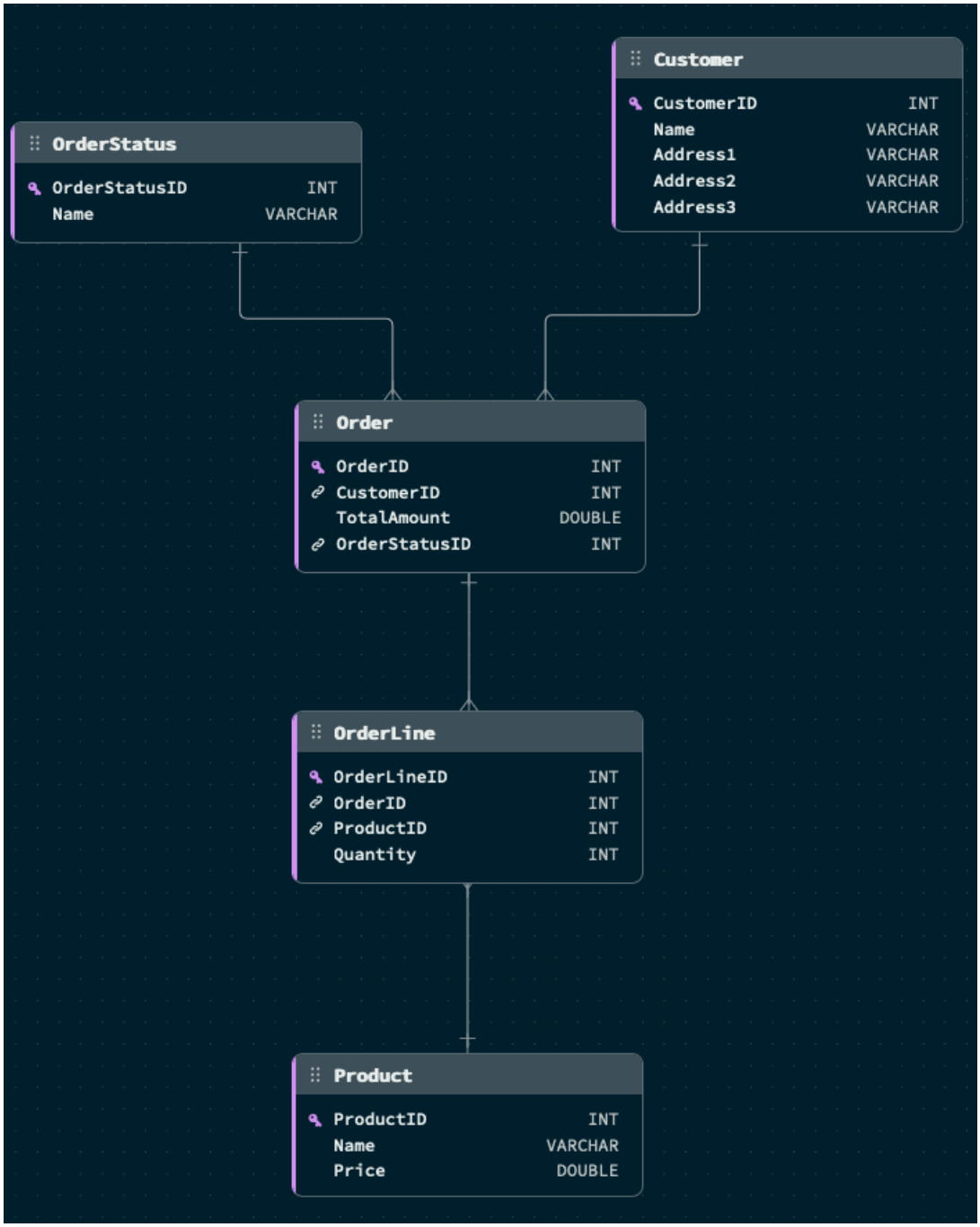
3.5.1 Performance

The steps to perform the implementation of the theater ticketing database are listed below

A) Entity Relationship Diagram

The Entity Relationship Diagram represents the logical structure of a database in a visual manner. This analysis is used to organize data as relational data in MongoDB.

- **Entities:** real-world items in an application
- **Properties/Attributes:** specify properties of entity and relationships
- **Relationships:** connects entities and represent dependencies



3.5.2 Reliability

Reliability is a core attribute of this system. Backups of the databases that hold user accounts and orders within the past five years will be made regularly. This is to ensure minimal data loss and ease of restoration in the case of server outages or cyber attacks. This backup system will be routinely tested by IT for reliability.

3.5.3 Availability

Adequate servers are available and used to host web services that guarantee 99.99% uptime. Additional backup servers are available for service in the event of an unforeseen outage.

3.5.4 Security

For Sullivan's Theater Ticketing System, MongoDB has enterprise-grade security that simplifies managing and deploying databases. For example, one can only access the database at a certain IP address. In addition to protecting the system it must have block bots to prevent them from buying tickets in bulk to high-demand movies.

3.5.5 Maintainability

The web application must support updates for compatibility with the latest web technologies, bug fixes, and security patches. For example, we use the technology stack MERN, which we would watch out for updates in the Express.JS and Node.JS middleware, routing, and packages. Also, there are numerous updates for the React.JS framework used to build user interfaces. Lastly, MongoDB updates from time to time on the commands and the api-key used to connect the codebase with the database.

3.5.6 Portability

The system is designed for any web browser use. Users can access this software from any device that supports web browsing, including mobile devices.

3.5.7 Interoperability

The degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context.

Syntactic interoperability – the ability to exchange data.

Semantic interoperability – the ability to correctly
interpret the data being exchanged.

3.5.8 Modifiability

Modifiability is about change, and our interest in it centers on the cost and risk of making changes.

3.5.9 Testability

Refers to the ease with which software can be made to demonstrate its faults through (typically execution-based) testing.

Refers to the probability, assuming that the software has at least one fault, that it will fail on its next test execution.

Industry estimates indicate that between 30 and 50 percent (or in some cases, even more) of the cost of developing well-engineered systems is taken up by testing.

3.5.10 Usability

Concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides.

Usability comprises
- Learning system features.
- Using a system efficiently.
- Minimizing the impact of errors.
- Adapting the system to user needs.
- Increasing confidence and satisfaction.

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 Sequence Diagrams

4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

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.

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

ASSIGNMENT 2 SOFTWARE DESIGN SPECIFICATIONS

6.0 Software Design Specification

6.1 Brief overview of system

The main objective of the software design specifications of the system Sullivan's Theater Ticketing System (STTS) is to ensure that the final version of this software product meets the requirements of the customers wanting to access our user-friendly system. The promised and detailed functions work as expected and most importantly are reliable and easy to use. Below are further details of the following:

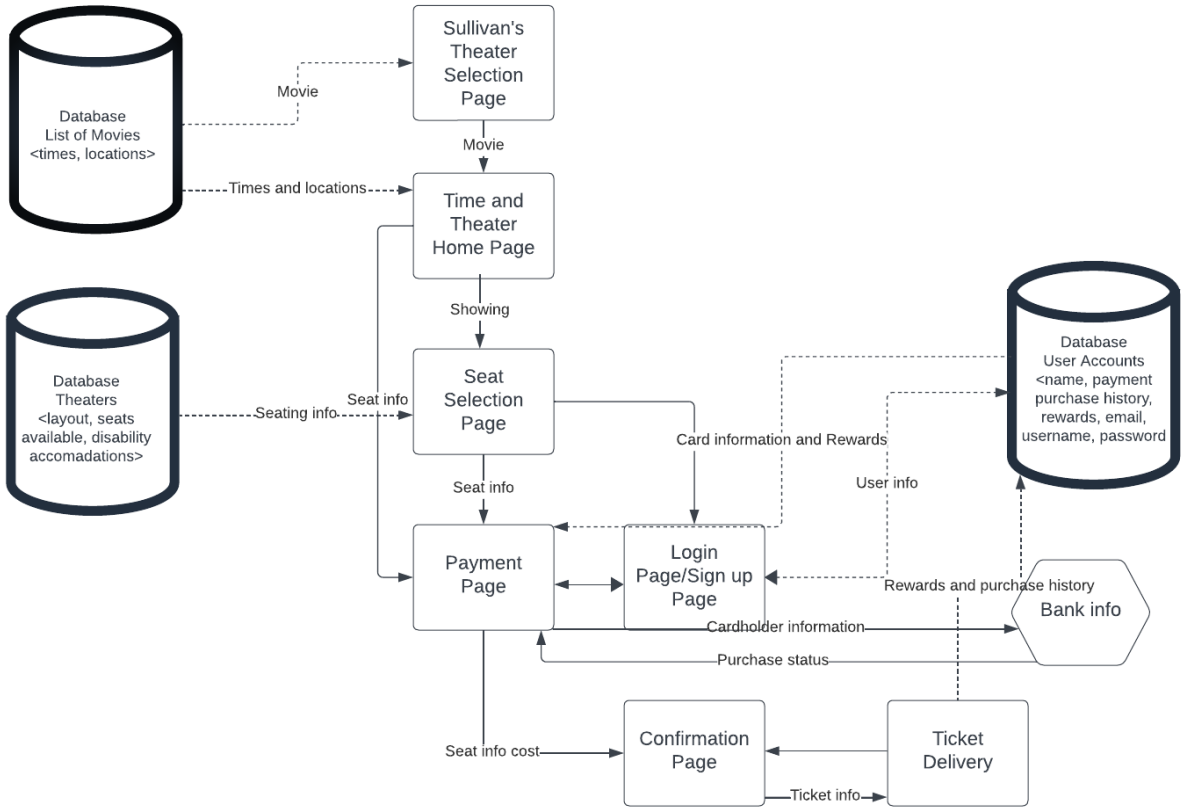
- **Software Architecture Overview**
- **Architectural diagram**
- **UML Class Diagram**
- **Description of Classes**
- **Description of attributes**
- **Description of operations**
- **Development plan and timeline**
- **Partitioning of tasks**
- **Team member responsibilities**

This will conclude the software design specification and should contain enough information to enact and enable the product to be produced.

7.0 Software Architecture Overview

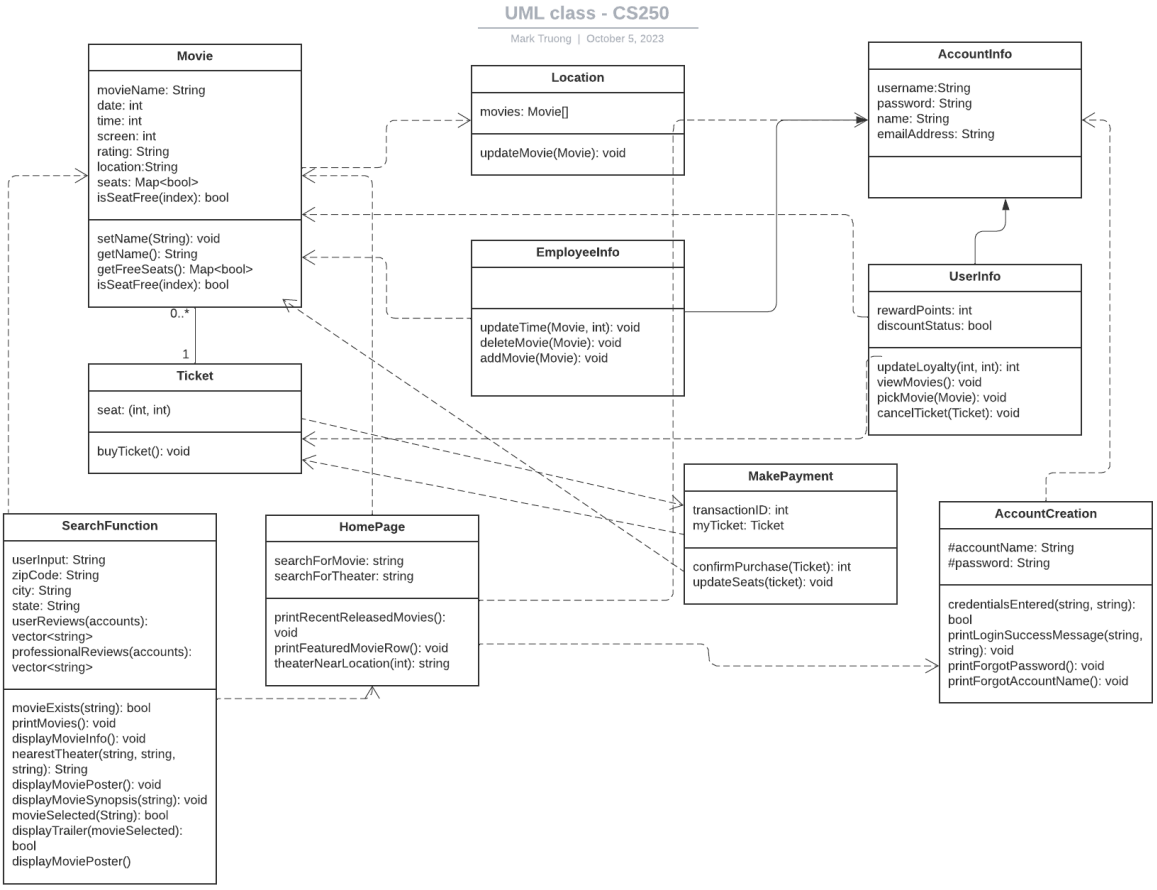
The Software Architecture of the Sullivan Theater Ticketing System (STTS) will define a high-level structure and organization of the software system as a whole. It will outline our function components and the way the components use each other's functionality. In addition to this, we will establish the way control is managed between the components as well.

7.1 Architectural Diagram of All Major Components (SWA)



This is the SWA diagram explaining the architecture of the software.

7.2 UML Class Diagram



This is a diagram of classes containing functions and variables.

7.3 Description of Classes

Classes

- **Home Page**
- **Account Creation**

This is the UML diagram outlining the classes, variables, and functions.

- **Search Function**
- **Movie Synopsis**
- **Trailers**
- **Reviews**
- **Purchase Tickets**

- **Purchase Confirmation**
- **User Feedback**
- **Refund Purchase**

7.4 Description of Attributes

Attributes related to the classes functions

- **Home Page attributes**
    - **String searchForMovie;**
    - **String searchForTheater;**
    - **Void printRecentReleasedMovies();**
    - **Void printFeaturedMovieRow();**
    - **String theaterNearLocation(int zipCode);**
- **Account Creation attributes**
    - **Protected: string accountName;**
    - **Protected: string password;**
    - **bool credentialsEntered(string accountName, string password);**
    - **Void printLoginSuccessMessage(string accountName, string password);**
    - **Void printForgotPassword();**
    - **Void printForgotAccountName();**
- **Search Function attributes**
    - **String userInput;**
    - **Bool movieExists(string userInput);**
    - **void printMovies();**
    - **String zipCode;**
    - **String city;**
    - **String state;**
    - **void displayMovieInfo();**

    - **String nearestTheater(string zipCode, string city, string state);**
    - **void displayMoviePoster();**
- **Movie Synopsis attributes**
    - **void displayMovieSynopsis(string movieTitle);**
- **Movie Trailers attributes**
    - **bool movieSelected(string movieTitle);**
    - **bool displayTrailer(movieSelected);**
    - **displayMoviePoster();**
- **Movie Reviews attributes**
    - **vector<string> userReviews(accounts);**
    - **vector<string> professionalReviews(accounts);**
    - **void displayReviews(string movieTitle);**
    - **bool wordCountLimit(userReviews->getWordCount());**
    - **void displayStarRating(int stars);**
- **Purchase Ticket(s) attributes**
    - **bool purchaseButtonActivated(string accountName);**
    - **void printTicketsPDF(string movieDetails);**
    - **int numberOfTickets(int tickets);**
    - **string displayMovieHistory(string accountName);**
- **Purchase Confirmation attributes**
    - **bool purchaseComplete(string accountName);**
    - **Void displayConfirmation(bool purchaseComplete);**
    - **void displayOptionsToPrint();**
- **User feedback attributes**
    - **vector<string> feedback(accounts);**
    - **string addFeedback(string feedback);**

- **Refund Purchase attributes**
    - **bool refundButtonActivated();**
    - **void displayCustomerServiceInfo();**

7.5 Description of Operations

- **Operations related to the functions inside the classes**
    - **Home Page**
        - **Display page will communicate with the server to display the home page correctly, showcasing the featured movies and the list of options necessary to create an account and search for movies based on their zip code.**
    - **Account Creation**
        - **Creating an account will communicate with the database servers and return a boolean value from the functions confirming their account is unique and will send a confirmation email.**
    - **Search Function**
        - **From the home page, search function will communicate with the server and return movies based on the user input as well as their location if entered.**
    - **Movie Synopsis**
        - **After selecting a movie from search result, function will communicate with the server and display the movie information and showcase local theaters.**
    - **Trailers**
        - **After selecting a movie, a box will pop up and autoplay the trailer for the movie selected.**
    - **Reviews**
        - **After selecting a movie, if users wish they will be able to scroll to the bottom of the page to view feedback from other users their experience with the movie.**
    - **Purchase Tickets**

        - **When a user selects purchase tickets, they will be prompted of how many tickets they would like based on a selection that goes up to 20. If they need more, they will be prompted to contact customer service.**

    - **Purchase Confirmation**

        - **Server will communicate with the purchase confirmation class and the functions to return that their purchase went through and prompt the users the options to print their tickets.**

    - **User Feedback**

        - **Users will have the option to leave feedback only after they purchase tickets to a movie they have listed in the history of their account.**

    - **Refund Purchase**

        - **Server will communicate with purchase class if refund is requested and the user will be prompted to contact customer service.**

8.0 Development Plan and Timeline

1. **Project Establishment (Week 1 - 2)**
    - **Communicate with clients and define requirements.**
    - **Define scope and objectives.**
    - **Establish project teams and roles.**
2. **Gather Requirements (Week 3)**
    - **Document functional and non-functional requirements.**
    - **Build use cases.**
3. **Design Architecture (Week 4-6)**
    - **Decide on front-end, back-end, and database stacks.**
    - **Create prototypes.**
    - **Design user interface and experience.**
4. **Front-end Development (Week 7 - 10)**
    - **Start on HTML, JavaScript, and CSS.**
    - **Build primary pages (home page, payment page, browsing and search results, etc).**
    - **Design user authentication.**
5. **Back-end Development (Week 11 - 14)**
    - **Use Node.js to develop server side API.**
    - **Design management of user accounts and payment.**
    - **Establish database security.**
6. **Database Development (Week 15 - 16)**

    - **Create databases and accompanying access functions.**
    - **Create backup measures.**
    - **Test and implement query performance.**
7. **Test Phase (Week 17 - 20)**
    - **Test servers and service under expected loads.**
    - **Test UI and UX for usability.**
    - **Unit testing and integration.**
    - **Identify and resolve major bugs.**
8. **Security (Week 21 - 22)**
    - **Penetration testing.**
    - **Carry out security measures to protect transaction information and user data.**
    - **Establish compliance with data regulations.**
9. **Deployment (Week 23 - 24)**
    - **Deploy to production.**
    - **Establish error tracking and monitoring.**
10. **Launch and Maintenance (Week 25+)**
    - **Launch software system.**
    - **Receive and identify user feedback.**
    - **Establish updates based on user feedback.**
    - **Monitor system performance.**

8.1 Partitioning of Tasks

Adam Graves

- **Documented brief system overview**
- **Documented software architecture overview**

Leo Sullivan

- **Documented development plan and timelines**
- **Partitioned tasks**
- **Outlined team member responsibilities**

Mark Truong

- **Created architectural diagram of all major components (SWA diagram)**
- **Created UML class diagram**

8.2 Team Member Responsibilities

Product manager, engineering manager, software architect, software developers, designers, testers, team/tech leads

Product Manager - Ensure the software meets user needs and matches the vision of stakeholders. Communicates with stakeholders and gathers feedback from customers to make informed decisions regarding product direction. Facilitates requirements gathering while communicating with cross-functional teams.

Engineering Manager - Leads and manages software engineers by guiding work direction and providing technical expertise. Help plan the project alongside product managers and stakeholders to clearly define project goals and timelines. Oversee code review to maintain coding standards and best practices are followed.

Software Architect - Primary planner for the software system, designing the system so that it meets both business and technical requirements. Makes technical decisions regarding selection of technologies, frameworks, and other components while taking into consideration maintainability, scalability, performance, and security. Writes comprehensive documentation outlining architecture and other design decisions. Collaborates with development teams, project managers, and stakeholders to ensure the product vision is put into practice.

Software Developer - Main builders of the software system. Software developers write the code using the chosen language and framework on the system as chosen by the software architects. They test code and fix bugs, integrate modules and other components, and document their work. Once a product is deployed, they maintain the system and deploy code into production.

Software Designer - Designs the user interface (UI) and experience (UX) through wireframing and prototyping interfaces. Takes into account how user-friendly the interactive features are while maintaining business goals. Ensures visual aspects are aesthetically pleasing and clearly indicate function and purpose. Tests for usability with users and uses feedback to improve design.

Software Tester - Tests software to assure quality and performs as expected. Creates test methods to test cases through scripts. Creates and configures testing environment to replicate environment software till be used in. Utilizes regression testing to ensure that new code does not impact old code functionality beyond what is expected. Tests for security issues and vulnerabilities.

ASSIGNMENT 2 SOFTWARE DESIGN SPECIFICATIONS