Movie Theater

Software Requirements Specification

v1.0.0

5/26/2024

Group 3
Lexa Rao, James Ardilla, Eli Lopez

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

Revision History

Date	Description	Author	Comments
5/26/2024	v1.0.0	Lexa, James, Eli	Included basic tickets sales functionality, advisement capability, and, simple UI.

Document Approval

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

Signature	Printed Name	Title	Date
	Lexa, James, Eli	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 annotates are largely taken from the IEEE Guide to SRS).

1.1 Purpose

What is the purpose of this SRS and the (intended) audience for which it is written.

The purpose of this document is to document the ways that the Movie Theater ticketing system functions. It will document both the back and front end of the ticketing system and analyze the ways the user interacts with these components. It is intended to be used by us when creating the specifications from the information our contractor gave us. We will be using the Waterfall Method for the Development of the program.

1.2 Scope

This subsection should:

(1) Identify the software product(s) to be produced by name; for example, Host DBMS, Report Generator, etc

(2) Explain what the software product(s) will, and, if necessary, will not do

(3) Describe the application of the software being specified. As a portion of this, it should:

(a) Describe all relevant benefits, objectives, and goals as precisely as possible. For example, to say that one goal is to provide effective reporting capabilities is not as good as saying parameter-driven, user-definable reports with a 2 h turnaround and on-line entry of user parameters.

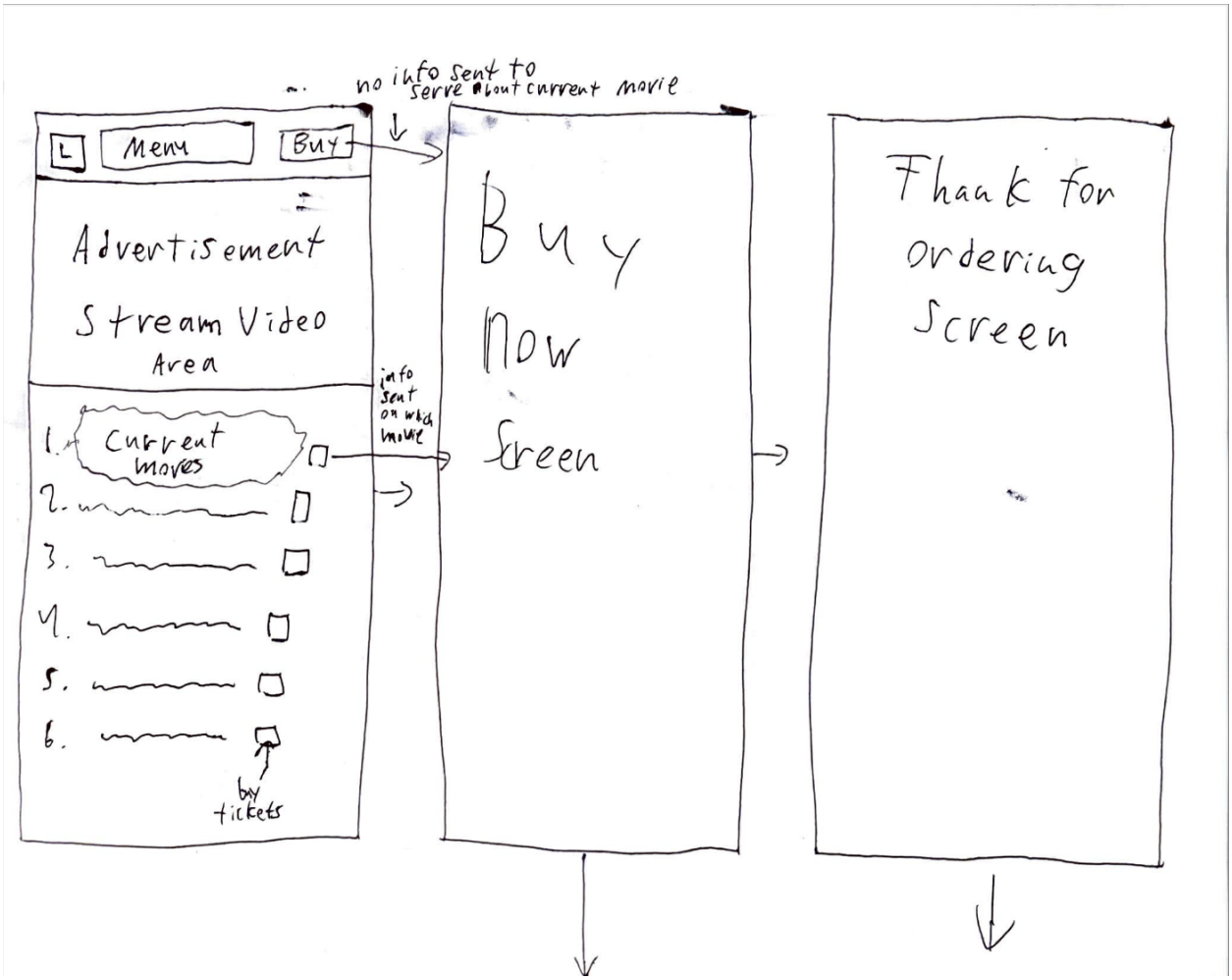
(b) Be consistent with similar statements in higher-level specifications (for example, the System Requirement Specification) , if they exist.What is the scope of this software product.

This project uses a client-server model to allow users to buy tickets from the vendor. To do this three separate software's will be designed that will work together to allow users to interact with the ticket vendor both in person and online. Ticket Server will be the software that runs on the server end. It will be responsible for a few different tasks that are split into subprograms. One task is indexing past and currently available movies including their descriptions. This means that it will add movie/movie descriptions to its currently playing library at the admin's request or allow for an admin to upload a new movie/movie description to it. No movies that are uploaded will be deleted by default when new movie/movie descriptions are uploaded. A second task the

servers will be responsible for is loading randomized clips of trailers that will be accessible to the client with a URL. The server will have a timer set for the length of the video and rotate the videos it is serving to the client every time this timer ends. A second URL will display the timer value in plain text. Both will be used by the clients to display the trailers to the user. A third functionality will be that the server will provide information about any movie currently playing to the client when it is requested. This will be searched either by the current playing index or the name of the movie.

The first of the two clients will be called Ticket Sales Anywhere. It will allow users to buy tickets from anywhere based on their requested order.

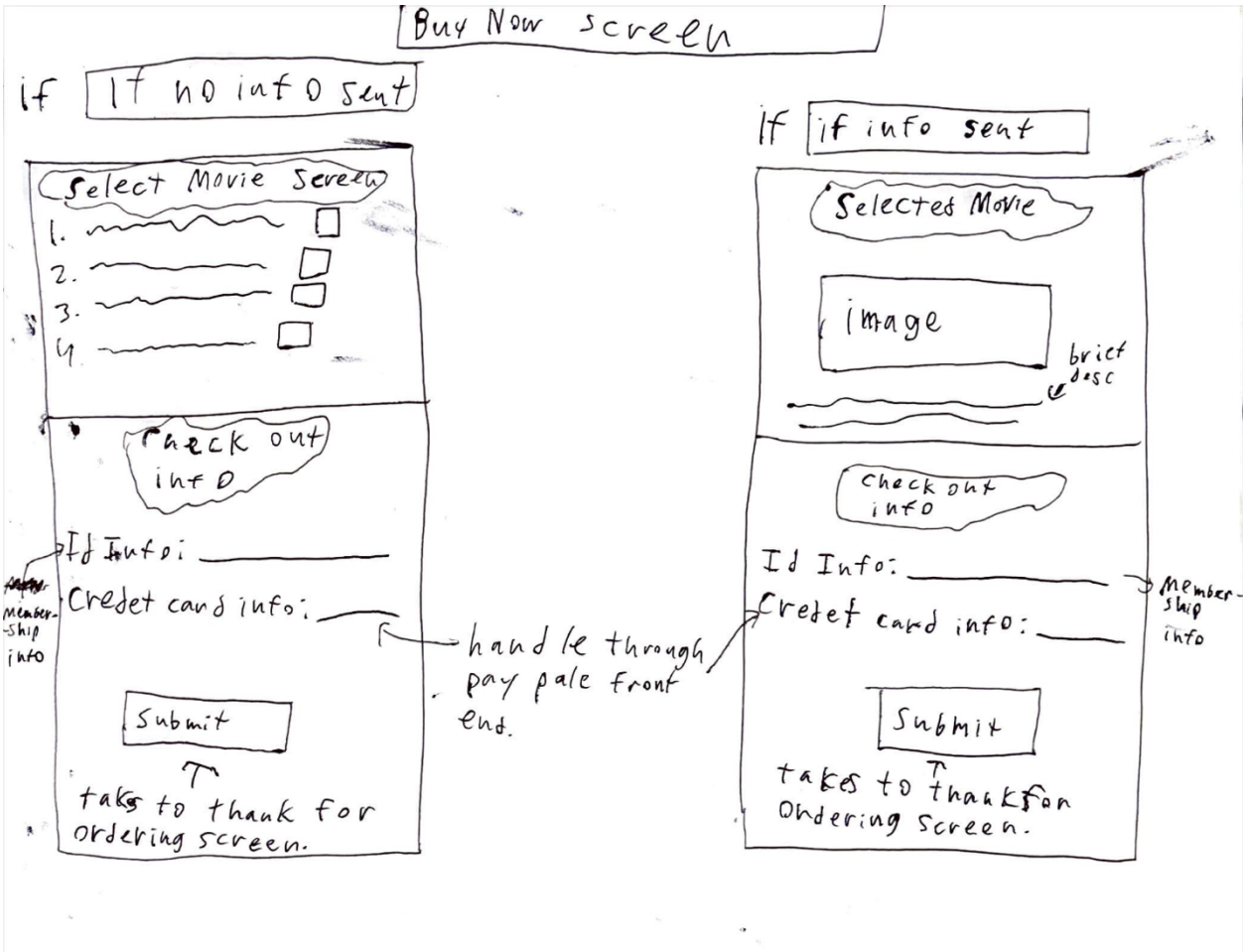
<Images are the rough draft of the storyboard>



This page allows people to see movies currently playing. It will display a current stream of movie trailers that the theatre is currently playing or movies that have been played in the past. It will also allow users to select from one of the current movies that are playing in theaters. Hovering over a title will bring up a trailer and description of the movie. If the user clicks buy

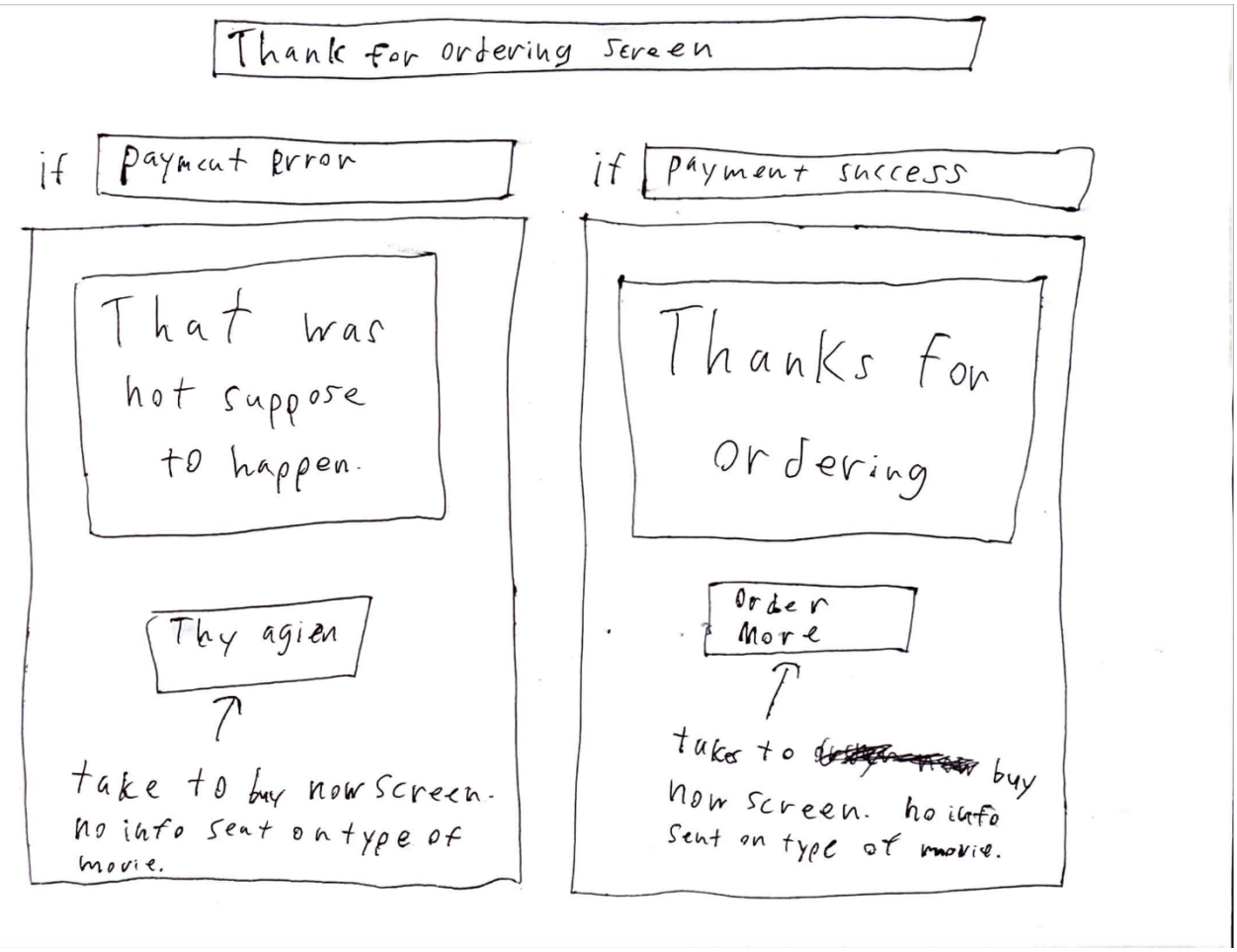
then they will get sent to the buy now screen without any info about the current movie they have selected in the website catch if not they will click on the movie they want and then get set to the buy now screen with the movie they wanted loaded into the website catch.

<Images are the rough draft of the storyboard>



This screen is responsible for letting a user pick out a ticket or tickets if they have not already. It then takes their membership ID for identification purposes and their credit card info using PayPal. When the submit button is hit the system will place this information from the form into the website catch and then go to the URL of the “thanks for ordering screen”.

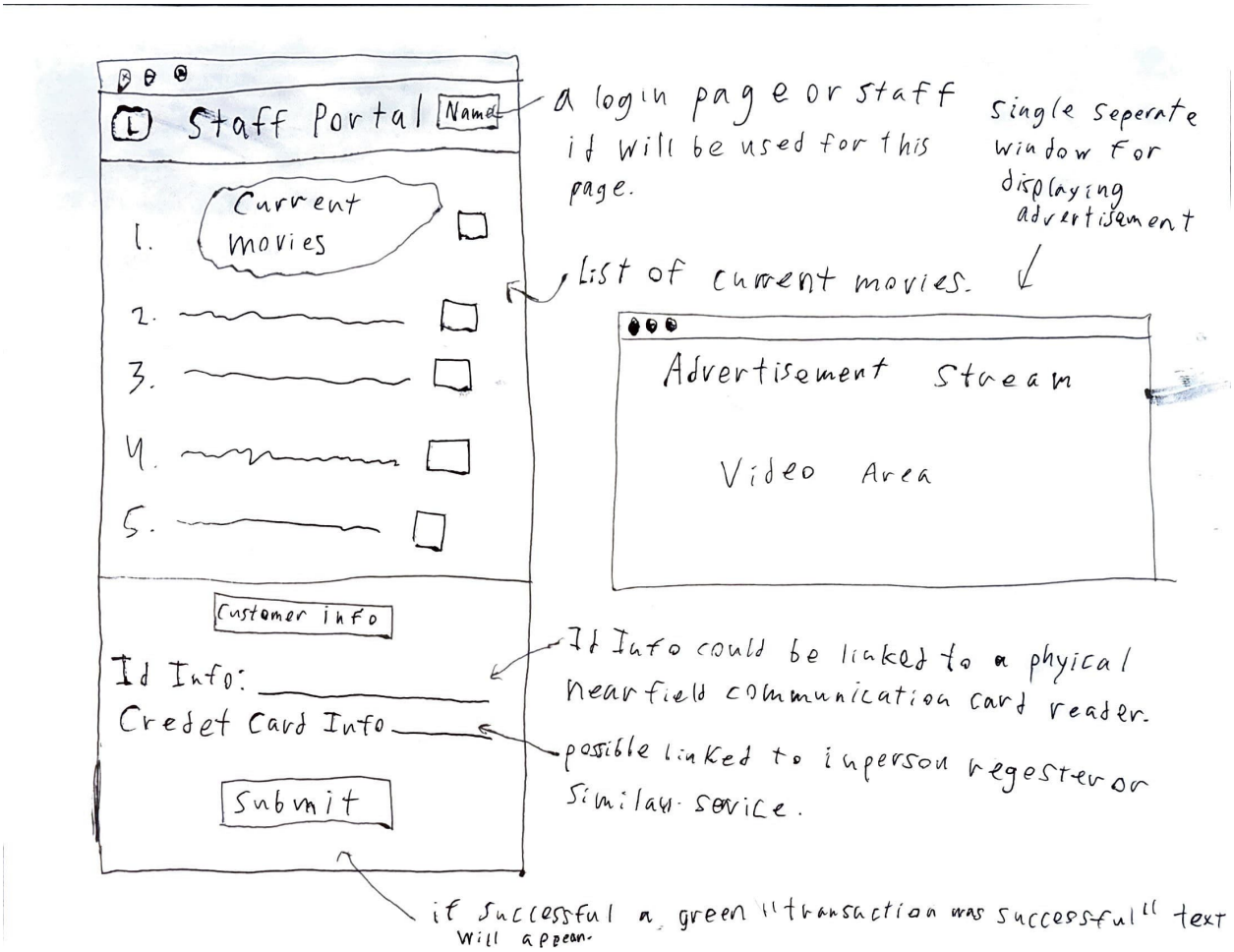
<Images are the rough draft of the storyboard>



This screen is responsible for displaying a loading screen and then posting information about the chosen movie, identification information, and PayPal / Payment information to the Ticket Server. This will then process the information. When done it will modify the html page so that it displays one of the two screens above. If successful it will then display an image with a caption over it saying thanks for ordering. It will then wipe the information from the local browser catch. If not, it will display an image with a caption over it saying that was not supposed to happen. In both situations, there will be a button at the bottom that redirects back to the Buy Now Screen. However, the text in the button may be different as you can see.

The second of the two clients will be a simple client made for staff members. The name of the program will be Ticket Sales Assistant. The web app will be written within JS nodes so it can run as an HTML and JavaScript-based app. It will have the following layout.

<Images are the rough draft of the storyboard>



The figure to the left is the staff portal screen. It will have a way for staff members to identify themselves. Login info will be provided with a popup page. The simple web-based app will display a list of current movies that are being played. Then a dialog box on the bottom will show the customer info entry screen. The two boxes for input which are respectively the ID info followed by a line and credit card info followed by a line will be represented in a form that will allow people to enter more info into it. Both could be linked to external services or peripherals such as a NEF communication card reader or a cash register. The submit button will upload the information to the Ticket Server. During the post and processing time, a small loading wheel will appear. After success, the server will change the HTML page so that a “success” text will be displayed, and if failed a small “failed, please try again” text will appear. If the logo is pressed a separate dialog page will appear that allows for the staff to input new movie information and trailers to the server.

The figure to the right is a small separate dialog box that will appear having an advertisement stream of movies that have been played in the past and are currently playing. These will be

taken from the video and time stamp function on the Ticket Server. The time stamp at the opening of the window will be used to tell how far into the video to play and when the video is complete so the page can reload without showing the user or view and fetch the next video from the URL described above.

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.

ID Info:

Information is relevant to a current user or customer that is unique enough to differentiate from other customers.

Credit Card Info:

The information that is needed for a financial transaction between the theater and the customer.

NEF Communication:

Near-field communication is used to transfer information over short distances.

Ticket Server:

Used for processing transactions, loading content about the movies, and managing user data.

Ticket Sales Anywhere:

Website for customers to order tickets from the theater online. The transactions are handled by the backend Ticket Server.

Ticket Sales Assistant:

Web app for displaying movie information and helping users to order tickets. Can interface with peripherals like a cash registrar and NEF communication reader to get information relevant to the transactions. The transactions are handled by the backend Ticket Server.

Advertisement Stream Video Area:

An area where past and present trailers and movies are played. This may be played in the background and is useful for keeping the customer engaged.

1.4 References

This subsection should:
(1) Provide a complete list of all documents referenced elsewhere in the SRS, or in a separate, specified document.
(2) Identify each document by title, report number - if applicable - date, and publishing organization.
(3) Specify the sources from which the references can be obtained.
This information may be provided by reference to an appendix or to another document.

This section will give a complete list of references documented somewhere else in the document.

- What Is MV3:
  - Title: What Is MV3
  - Report Number: N/A
  - Date: N/A
  - Publishing Organization: Chrome Developer

- IEEE Recommended Practice for Software Requirements Specifications:
  - Title: IEEE Recommended Practice for Software Requirements Specifications
  - Report Number: 830-1998
  - Date: N/A
  - Publishing Organization: IEEE Computer Society

- Paypal Client and Server API:
  - Title: Paypal Client and Server API
  - Report Number: N/A
  - Date: 04/24/2024
  - Publishing Organization: Paypal Developer

- Bitcoin RPC API
  - Title: Bitcoin RPC API
  - Report Number: N/A
  - Date: N/A
  - Publishing Organization: Bitcoin Developer

- Web server Clustering: High Availability Setup
  - Title: Web Server Clustering: High Availability Setup
  - Report Number: N/A
  - Date: 05/20/2024
  - Publishing Organization: Alibaba Cloud

- HTTP Requests
  - Title: HTTP Requests
  - Report Number: N/A
  - Date: N/A
  - Publishing Organization: Code Academy

- DevOps and Security Glossary Terms:
  - Title: DevOps and Security Glossary Terms
  - Report Number: N/A
  - Date: N/A
  - Publishing Organization: Sumo Logic

- The Ultimate Guide to Review Scraping in 2024
  - Title: The Ultimate Guide to Review Scraping in 2024
  - Report Number: N/A
  - Date: 01/27/2024
  - Publishing Organization: AIMultiple

- Access Control: Models and Methods
  - Title: Access Control: Models and Methods
  - Report Number: N/A
  - Date: N/A
  - Publishing Organization: Delinea

1.5 Overview

This subsection should:
(1) Describe what the rest of the SRS contains
(2) Explain how the SRS is organized.

The rest of this document will be responsible for documenting information about the product. This could include information such as what systems the product runs on, how the program

integrates with other systems at a lower level, and how will be using it. As well as other user interfaces not documented in the screen above. As well as documenting any software-based dependencies our program will have. It will also include the hardware, software, and communication interfaces, as well as the functional and non-functional requirements.

The second section will give a general description of the product. This will include the information on what the system is as mentioned above. The third section will go into the specification requirements of the program. As mentioned above it will split these requirements into several sections. We will not be filling the third section after 3.5.

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

This application is designed to operate within a cloud-based infrastructure of the Amazon Web Services (AWS) platform. Our decision to host most of the server-side content on AWS containers is driven by the need for scalability, reliability, and security AWS offers. Consequently, the application's performance and availability are linked to AWS's operational status and uptime.
We also are reliant upon the uptime and performance of credit card payment processing companies such as Visa and Mastercard. Downtime on these company's end would cause issues with our software.
We are also dependent on the uptime of the bitcoin network, which is a decentralized system that processes payments. Bitcoin payment processing times can vary wildly depending on the time of day as well as the volume of transactions at any given time.

2.2 Product Functions

This product has 4 main functions:

1) The user's ability to purchase tickets.
2) The client ability to enter an admin mode and perform various customer service actions.
3) The application will register and save user info either for customer use or for client purposes.
4) The application will use a queue system to handle influxes of users and limit purchases.

2.3 User Characteristics

There are two main user groups: customers and theater workers.

Theater workers can be given higher levels of access to the software, since they will need to access customer information to perform customer service work. It can be assumed that the theater will have done background checks on workers before allowing them to access this information.
Customers should be given a much more restrictive version of the software, since it is a public facing application on the open internet. We should take steps to ensure that the software is highly accessible and has minimal ambiguity. Accessibility for customers with disabilities such as vision or hearing impairment should be considered too.

2.4 General Constraints

Regulatory: Since this software accepts credit cards and bitcoin, we must adopt an appropriate logging policy to comply with local and federal laws. Bitcoin related purchases should be especially focused on due to the often-rapid changing legal status of cryptocurrencies. Additionally, we must handle sensitive user data appropriately and comply with state and federal data privacy laws, such as California's data privacy acts.

Reliability: The client has requested that the software handles at most 1000 people and has a queue system to control purchasing. We must ensure that we utilize appropriate load-balancing methods to manage this.

Audits: Similar to the above policy with CC and bitcoin we must provide appropriate logging for tax purposes in line with federal and state codes.

Safety: While not entirely in our control, we must be mindful of age-rating requirements for movies as well as content moderation for reviews. Another concern may be logging all chats with customer service representatives to document any inappropriate interactions.

2.5 Assumptions and Dependencies

The software is a web application, so we assume that modern browsers such as Firefox/Chrome/Safari will be able to support the software. If these browsers change support, we will change this SRS accordingly.
A known issue that may cause this change is Chrome's manifest v3, which will begin releasing to the public in mid to late 2024.

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 easily be 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

Do:

- Short error messages for easy customer support
- Visible purchasing UI, all buttons should be easy to find on their page
- Language selection should be offered upon first accessing the site and should be logged to users account
- Splash page should have users account info easily readible upon login ie points, past purchases etc

3.1.2 Hardware Interfaces

We should always ensure that our designed system does not pull too many resources from the clients' AWS instances. We should implement rate-limiting to prevent accidental DOS from users attempting to purchase tickets.

3.1.3 Software Interfaces

As of v1.0.0 we will require the following software:
- AWS
- JS Nodes (Server Application)
- Electron JS (Client Application)
- Paypal Client and Server API
- RPC API (Bitcoin)

3.1.4 Communications Interfaces

AWS staff handle most sever side communications interfaces, but we should still focus on ensuring that customers information is always transmitted between components in a secure manner with HTTPS and properly configured certifications.

3.2 Functional Requirements

This section describes specific features and functionalities for the movie ticketing system. The focus will be on the main functional requirements. For each requirement, the document will describe inputs, processing, outputs, and error handling where needed.

3.2.1 User Handling:

3.2.1.1 Introduction:

-The system will handle concurrent access by at least 1000 users.

3.2.1.2 Inputs:

- User requests for browsing, purchasing tickets, or accessing account information.

3.2.1.3 Processing:

- Implement load balancing and optimized database to deal with high traffic.

3.2.1.4 Outputs:

- Correct responses to different user interactions
- Queued responses when there are excess users.

3.2.1.5 Error Handling:

- Display error messages if the system is overloaded.
- Implement a queue system to manage excess users smoothly.

3.2.2 Web Browser Operation:

3.2.2.1 Introduction:

-The system will entirely operate within a web browser environment,

3.2.2.2 Inputs:

- URL requests from user's web browsers.

3.2.2.3 Processing:

- Create the UI
- Make sure the system is compatible with modern browsers (Chrome, Firefox, Safari, Edge).

3.2.2.4 Outputs:

- Interactive web pages for browsing, purchasing, and managing tickets.

3.2.2.5 Error Handling:

- If the user's browser is not compatible display error mssage

3.2.3 Bot Prevention:

3.2.3.1 Introduction:

- The system will block bots attempting to buy tickets.

3.2.3.2 Inputs:

- Ticket purchase requests from users.

3.2.3.3 Processing:

- Implement a verification system
- Monitor and analyze request patterns to identify and block bots.

3.2.3.4 Outputs:

- Blocked transaction attempts from identified bots.
- Successful transactions from legitimate users.

3.2.3.5 Error Handling:

- Message for users when verification system interaction is required.
- Log and alert administrators of suspected bot activity.

3.2.4 Database Interface:

3.2.4.1 Introduction:

- The system will interface with the database of showtimes and available tickets.

3.2.4.2 Inputs:

- Requests for showtimes, ticket availability, and user transactions.

3.2.4.3 Processing:

- Perform CRUD operations on the database.

3.2.4.4 Outputs:

- Updated info on showtimes and ticket availability.
- Confirmation of successful transactions.

3.2.4.5 Error Handling:

- Show error messages for issues when interacting with database.
- Log database errors for admin.

3.2.5 Ticket Purchase Limitations:

3.2.5.1 Introduction:

-The system will enforce ticket purchase limitations.

3.2.5.2 Inputs:

- User selected showtimes and ticket quantities.

3.2.5.3 Processing:

- Validate the max ticket quatinty of 20
- Confirm that purchase is 2 weeks prior to showtime or 10 minutes after showtime.

3.2.5.4 Outputs:

- Confirmation of ticket purchase.
- Error messages for exceeding ticket quantity limits or outside purchase times.

3.2.5.5 Error Handling:

- Notify users if they attempt to buy more than 20 tickets or purchase outside the allowed timeframe.

3.2.6 Administrative Mode:

3.2.6.1 Introduction:

- Admins will have more access than customers. In admin mode they will be able to modify movie information.

3.2.6.2 Inputs:

- Movie details such as descriptions or movie names.

3.2.6.3 Processing:

- Add new movies to the database and update the existing database.
- Validate input details

3.2.6.4 Outputs:

- Confirmation message for successful description update or movie addition.
- The movie will show the added movie or changed description.

3.2.6.5 Error Handling:

- Show error message for invalid inputs.
- Handle database issues

3.2.7 Customer Feedback System:

3.2.7.1 Introduction:

- The system will include a customer feedback system.

3.2.7.2 Inputs:

- Customer feedback submissions.

3.2.7.3 Processing:

- Store feedback in the database.
- Alert admin to new feedback entries.

3.2.7.4 Outputs:

- Confirmation messages for customers.
- Feedback reports for administrators.

3.2.7.5 Error Handling:

-Error messages for failed feedback submissions.

3.2.8 Discount Ticket Support

3.2.8.1 Introduction:

- The system will support various discount ticket types.

3.2.8.2 Inputs:

-Discount codes

3.2.8.3 Processing:

- Validate discount eligibility.
- Apply discounts to ticket purchases.

3.2.8.4 Outputs:

- Updated ticket pricing based on discounts.
- Confirmation of discounted purchases.

3.2.8.5 Error Handling:

- Notify users of invalid discount codes.

3.2.9 Online Review Scraping:

3.2.9.1 Introduction:

- The system will scrape online review sites to display reviews and critic quotes of movies.

3.2.9.2 Inputs:

- URLs of review sites and movie identifiers.

3.2.9.3 Processing:

- Extract review data.
- Update the database with new reviews and critic quotes.

3.2.9.4 Outputs:

- Display reviews and critic quotes on the movie detail pages.

3.2.9.5 Error Handling:

- Handle failures in scraping by notifying administrators.
- Show placeholder content if reviews cannot be fetched.

3.3 Use Cases:

3.3.1 Use Case #1:

Name: Purchase Movie Tickets
Actor: Customer

Flow of Events:

The customer accesses the movie theatre's website. The customer navigates to the main menu section to browse currently available movies. The customer selects a movie they wish to watch. The system displays movie details such as showtimes, available seats, and ticket prices. The customer selects the desired showtime and number of tickets. The system validates the ticket quantity and showtime eligibility. The customer proceeds to checkout. The system prompts the customer to log in or create an account if not already logged in. The customer enters payment information and completes the transaction. The system confirms the ticket purchase and provides a confirmation number.

Assumptions about Entry Conditions: The customer has access to a compatible web browser and an internet connection. The customer has selected a movie from the available options and has valid payment information.

3.3.2 Use Case #2:

Name: Admin Modify Movie Information
Actor: Admin

Flow of Events:

The admin accesses the admin portal of the movie theatre's website. The admin logs in using their credentials. The system presents a list of currently available movies. The admin selects the movie they want to modify. The system displays options to edit movie details such as descriptions, showtimes, or trailers. The admin makes the desired modifications. The system validates the changes and updates the database accordingly. The admin receives a confirmation message of successful modification.

Assumptions about Entry Conditions: The admin has appropriate permissions to access the admin portal. The admin is logged in and authenticated. The movie selected by the admin exists in the database.

3.3.3 Use Case #3:

Name: Submit Customer Feedback
Actor: Customer

Flow of Events:

The customer accesses the movie theatre's website. The customer navigates to the "Feedback" section. The system provides a form for the customer to submit feedback. The customer fills in the required fields such as name, email, and feedback message. The customer submits the feedback. The system validates the input and stores the feedback in the database. The system sends a confirmation email to the customer acknowledging receipt of feedback.

Assumptions about Entry Conditions: The customer has access to a compatible web browser and an internet connection. The feedback form is accessible and functional. The customer provides valid account information.

3.5 Non-Functional Requirements

The following section will state the non-functional requirements for the movie ticketing system.

3.5.1 UI:

- The user interface will be designed to be intuitive and user-friendly to make website navigation easier for inexperienced users.
- Response times for user interactions will be optimized for minimal wait times.

3.5.4 Security:

- The system will implement security measures to safeguard user data, payment information, and system integrity.
- User authentication will be enforced for all user interactions, with secure password storage in the database.
- Access controls will be implemented to ensure that only authorized users have access to sensitive functions.
- Payment transactions will comply with industry standards.
- Regular security audits and vulnerability audits will be conducted to identify and address potential security threats.

3.6 Inverse Requirements

The system cannot allow purchases more than two weeks in advance or 10 minutes after a showing.

3.7 Design Constraints

All parties that use this application are using a web-based version of the software, so we are constrained by web browser limitations.

3.8 Logical Database Requirements

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 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