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.

<Encore Ticket Solution>

Software Requirements Specification

<Version 0.1>

<02/01/24>

<Group #>
<Nicolas Brizuela, Ian Carscadden, Christopher Anderson >

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

Revision History

Date	Description	Author	Comments
<01/02/24>	<Version 0.1>	<Chris Anderson>	<First Revision>
		Ian Carscad…	
		Nico Brizuela	

Document Approval

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

Signature	Printed Name	Title	Date
	<Chris Anderson, Ian Carscadden, Nico Brizuela>	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

The Encore Ticketing Service is meant to provide a fluid and cohesive process for a user/customer to browse movie titles, select times and dates, reserve seats in showrooms, purchase food/drinks, and pay for all the mentioned products above. Furthermore, the website should be a clear path and easy to navigate. All initial steps in the process can be unsecured until private details are required. Accounts should be an option to create and offer a premium version of the website for paying members for exclusive deals, perks, and events.

1.1 Purpose

The purpose of this project document is to provide a ticketing service to manage, reserve, monitor ticket sales, and track customer interactions within the Theatre Ticketing App.

1.2 Scope

(1)

Partnered Theater Databases

(2)

The app is not responsible for the content quality for each showing, event logistics such as parking and transportation, security for physical events, licensing and legal compliances based on event showing.

(3)

The goals and objectives for this software system are as follows. Achieve a notable reduction in average time taken to complete a ticket transaction, earn 95% customer satisfaction, create a report system that notifies users of alerts in a timely manner, secure transactions through data encryption and network security, customize event pages and achieve a high utilization rate for custom features by event organizers.

The scope of this project is to create a software service that includes the development, maintenance and implementation of a ticketing system. This product is designed to be robust and includes, but is not limited to, the following specifications, event management, ticket transactions, user profile data retrieval, seating management, notification alerts, sale security, customization, accessibility, global communication networking and integration between other platforms.

1.3 Definitions, Acronyms, and Abbreviations

ETS - Encore Ticketing Service
Embedding - Uploading content from the app onto the internet
Traffic - Total amount of users on the app at any given time
Website Optimization - Implementation used to improve the site's ability to convert customers to the service
Resolution - The number of pixels that can be shown on a devices display
User Interface (UI) - All the parts of the app that the user can manipulate and interact with

User Experience (UX) - The range of emotions, attitudes, and ease-of-use a person has when using the service
Back End - The server side of a software system that allows software to communicate
Front End - The visual aspects of a website or app
Cache - Temporary storage for data to reduce the time it takes to retrieve information
Encryption - The process of converting data into secured code to protect from unauthorized access
Firewall - A security system that monitors incoming and outgoing network traffic
Scalability - How a system handles an increasing number in workload without compromising performance
Tokenization - The process of of replacing data with a identifier, a token
Repository - Storage location where control files and code are stored
Pipeline - Term used to describe where the output of one element is the input to the next one
Git - A version control system used for source code management
Application Programming Interface (API) - Set of rules that allows different software applications to interact with one another
SMTP - Simple Mail Transfer Protocol

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.

1.5 Overview

This SRS is organized and should be read in the order of each subsection. All information explained in each subsection will be vital and come up again in others. The SRS is written to introduce, describe and then attack the problem at hand in that order.

2. General Description

The vision of Encore Ticketing Service is to create a fast, simple, but engaging ticketing service for use by anyone who is in the market for movie tickets. In addition to this, ETS creates an engaging environment by providing users with movie ratings, reviews, trailers. The Encore Ticketing System uses integration with certain APIs to gain information about movie schedules, ticketability, seating availability, as well as movie reviews and ratings for users to see. In addition, integration with API for payment processing will allow users to make secure payments within the application when purchasing. For all of the data on our site to be accurate and updated timely, we will be integrating with an API directly provided by each movie theater.

2.1 Product Perspective

All services required for website:
Software:
- *Server to reach users*
- *UI for users to interact with*
- *API to connect to banking services and servers to reference availabilities*

Hardware:
- *Storage servers to hold users, movies, and open dates*

- *On-site computers for customers to buy tickets on-site*

2.2 Product Functions

The Encore Ticketing Service will navigate through web pages to reserve movie tickets. It will also hold the login information of users who decide to make an account, otherwise, a temporary guest client will be created to purchase tickets and additional products and sent to the email they provide. Another function of the website is the ability to get seat, date, and time availability. It will also connect to charge their bank accounts with the billing information they provide. It will email the purchased movie tickets to the user. It will connect to banks through API services to verify banking information and verify balance is adequate for purchasing tickets, if the user's bank balance is not sufficient the charge will be rejected, and inform the user they can not buy tickets. It will also be manageable by an administrator so customers can be refunded, change dates/times, or edit screenings and showtimes.

2.3 User Characteristics

The users intended for the website are basic level education such as elementary level of comprehension. The technical expertise is minimal, users should be able to understand how a webpage works and how to navigate a website. Users are expected to understand how to input personal information and fill in prompts.

2.4 General Constraints

Hardware Constraints:

To ensure the optimal functioning of a ticketing service software system tailored for a movie theater, specific hardware constraints must be met. The system should be compatible with both iOS and Android platforms, accommodating various devices such as tablets, mobile devices, and personal computers. Seamless operation is essential across different network environments, including 3G, 4G, 5G, and wifi, to provide users with flexibility in accessing the ticketing application. Maintaining a consistent resolution per device is crucial for a unified and visually appealing user experience. Additionally, compatibility with location-based services, such as GPS, is necessary to enhance features like seat selection and personalized recommendations, offering a comprehensive and efficient ticketing solution.

Software Constraints:

The movie theater ticketing software system needs to follow specific software rules to work well. It's important to use full-stack languages for complete app development, so it can work smoothly on different platforms. Developing with React framework makes the user interface and experience better, creating a responsive and dynamic ticketing app. Adding third-party APIs is important to access more features and external data sources, making the system more useful. It's crucial to have strong security measures to protect user data and transactions, making the ticketing platform safe and trustworthy. Following license agreements and App Store requirements is important to meet industry standards and guidelines. Also, considering browser capabilities is necessary to make sure the app works well on different web browsers for a wide range of users.

Environmental Constraints:

The ticketing app should seamlessly adapt to changing networking conditions, handle geographical extremes for consistent performance, and exhibit cultural sensitivity when used internationally.

Budget Constraints:

Effectively managing budget constraints is crucial for the movie theater ticketing app's development and sustainability. This includes allocating resources for technology frameworks, defining scalability thresholds, anticipating maintenance costs, budgeting for marketing promotions, managing third-party integration expenses, addressing compliance requirements, and allocating resources for training and record-keeping. Balancing these considerations is essential to meet objectives and ensure financial sustainability.

2.5 Assumptions and Dependencies

If the operating system on the hardware is unsupported by our software, the application will show a dialog box saying that the OS is unsupported and a different OS is needed to open. In a related case, if the browser protocol is unsupported, a dialog box will communicate a new preferred browser to run the application. If the app is loaded onto unsupported hardware the application will not open and show a dialog box communicating with the hardware that is not supported to run the application. If the application is out of date, the application will not open and require a software update before opening. If an API is not compatible with the application, the information provided by the API will be shown as unavailable and will have to be fixed or worked around by the software team.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

This product should intake a user's input and simultaneously give it appropriate feedback. There should be keyword navigation when typing to help speed up the process and recommend movies properly for the customer. For each movie or play showing there should be an image followed with a small amount of text describing the event. Transcripts and captions for any videos to help those who may be hard of hearing.

3.1.2 Hardware Interfaces

The ticketing software system must be compatible with a range of Operating Systems. Examples of such are Windows, macOS and Linux. For the systems hardware and low level computation the app must be capable to work with 32 and 64 bit architecture systems. The display and resolution must fit the minimum 1024 x 1024 pixel requirements and adapt when necessary. The system must have printing functionality if a user requests a physical copy of their ticket. Along with all of the previously listed, the devious should come off as user friendly.

Integration with AWS will aid in lifting some of the heavy lifting done by the system. Lastly, the device must be able to support multiple devices.

3.1.3 Software Interfaces

The system shall allow external integration from third party applications. Data transfers need to be at a high level to help with debugging. The server being up to date is a top priority with the success of the app. Along with this it must not crash. Messaging systems need anonymous options so some sort of encryption should be used to preserve identity when necessary. Passwords, usernames and more should be stored away from any particular person's access. This can be completed with two factor authentication. File transfer should be secure along with a max file size limit. Certain files can only be accepted for viewing, such as JPG and PNG. Version control systems like Git needed to be used for collaborations between developers. This can also help with branching or merging with another company.

3.1.4 Communications Interfaces

For the large range of communication with the system, email must be obtainable. The email must comply with the SMTP. There needs to be a call center for users to interact with another voice rather than strictly a machine. Alongside this a chatbot can be used to help clear minor doubts within the app. Text size regulation can help readability, nothing too small or too large. Font fits in this category. From a security measure standpoint, there needs to be API endpoints that can only be directed and accessed through API keys. There should be no risk of data breaches as API keys will not be stored directly in source code. All server chats need to be logged and recorded. If a user wants to communicate with another device, one must understand the responsibility in that. Integration with foreign servers alongside their payment gateways will be necessary to complete transactions. A notification center should be created to keep users engaged and on track to their event.

3.2 Functional Requirements

3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

The application will need to handle up to 1000 people at a time on varying web browsers so that users can connect to the application from the preferred browsing domain. Users will need to be verified to be human to limit the flow of people and ensure no bots are trying to buy mass amounts of tickets. It must be connected to a database containing movie titles, showtimes, and theater names. Users are also limited to buying a maximum of 20 tickets at once and are given 5 minutes to go through, choose seats, showtime, theater, and input purchase information so that idle users are removed and ensure users aren’t taking up space in the available user count. It will also support an administrator mode where a supervisor can manage movies, customer tickets, and account information.

3.2.1.2 Inputs

Users will input movie theaters into a search menu to find nearby theaters. They will also be able to browse movies at each theater and select preferred times and purchase food along with

their movie tickets. The user will be able to select the quantity of movies before inputting financial information. If a user is an admin or supervisor, they will have system-specific tokens to access an alternative version of the webpage to manage movies, visuals of the page, or access account information.

3.2.1.3 Processing

All requests will be processed by the task-specific API, such as handling banking transactions, requesting the availability of movies and showtimes, or checking user account information.

3.2.1.4 Outputs

The output will be shown to the user on the webpage for them to select the movie, showtime, and theater.

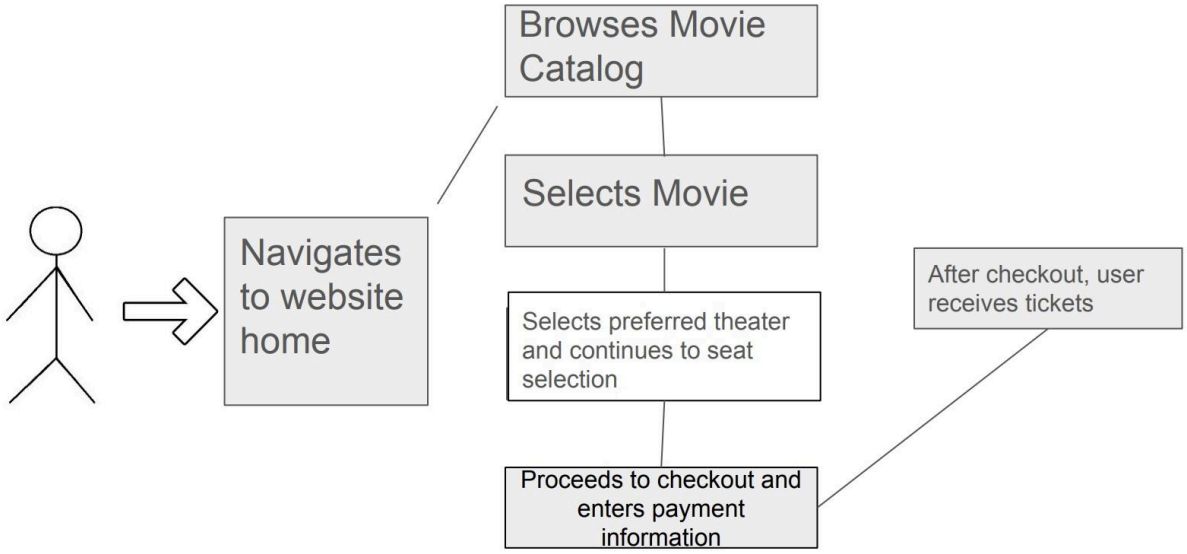
3.2.1.5 Error Handling

If when a user tries to complete a purchase but when the final availability check is done the item is no longer available, an error message will be displayed saying, “We could not complete your request - item is no longer available.” If a user is attempting to navigate to a certain page in the website and the server will not respond with the page, a message will be displayed saying “We could not complete your request” and the users will stay on the current page they are on. If reviews on a movie are not being scraped in correctly from the web a message will be displayed where the reviews would normally be that will say “Could not load reviews.” If a certain API for any process inside of the website cannot be reached to read data, text will be displayed that says “Could not load information” for anywhere that the data from the certain uncommunicative API is needed.

3.3 Use Cases

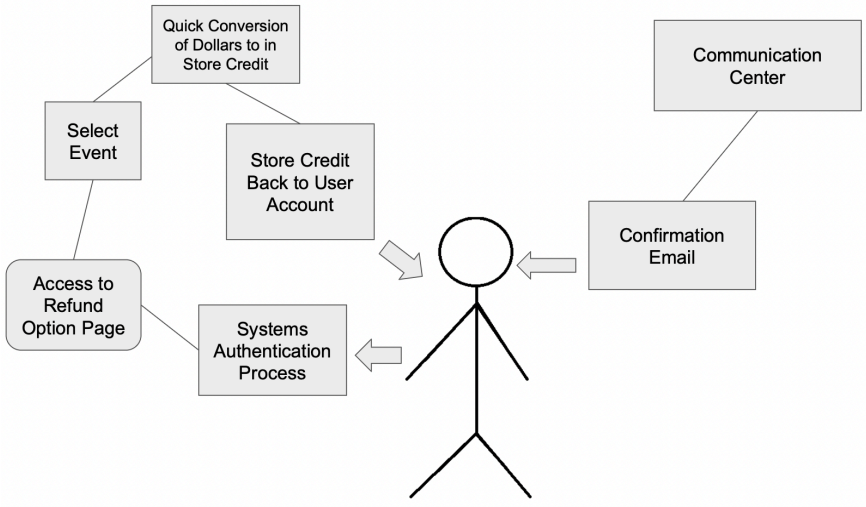
3.3.1 Use Case #1

User wants to go see a movie. They browse the movie catalog and click on a movie they are interested in. They select their theater and press a button to get tickets. The user chooses their seat and continues the checkout process. The user inputs their payment information and continues with the payment. The payment gets processed and the user receives an email with their tickets.



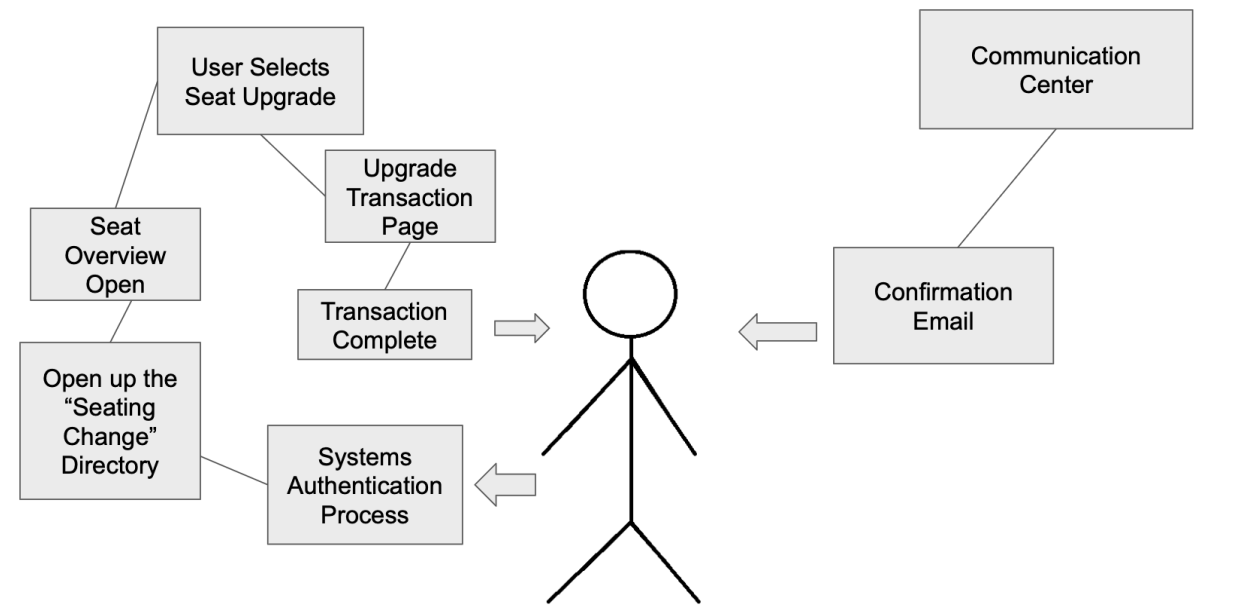
3.3.2 Use Case #2

User wants a refund on a ticket they recently purchased. There first needs to be an authentication to prove who the user is. Then, navigating through the app to the specific refund section they can request a refund. Immediately after the request a message is sent to the software system to initiate a credit deposit, worth the amount purchased, back into the user's account. Shortly after, an email notification follow up is sent to the user to confirm the transaction.



3.3.3 Use Case #3

A user logs into the app. They then navigate to the seating change section. The user decides to upgrade their seat. The User is brought to an overview of the available seating options with price. They pay the upgrade fee. The user receives an update from the notification center



3.4 Classes / Objects

3.4.1 <Class / Object #1>

Theater Class
3.4.1.1 Attributes:

The theater class will have the name of the theater, Movie subclass, and the availability of seats in the theater. It will also have what food is served at the theater. The location of the theater will be theater-specific.

3.4.1.2 Functions:

The theater class will have a function to request a bank transaction from the user. Another function will be to get a Movie class based on the name. Another function will be to get a list of food served.

<Reference to functional requirements and/or use cases>

3.4.2 <Class / Object #2>

Movie Class

Attributes:
The movie class will have the name of the movie, the ratings of the movie from different rating providers (IMDB, Rotten Tomatoes), movie reviews, movie trailer, movie description. The movie class will also contain an identification number for easy communication with API's.

Functions:
The movie class will have functions that return certain attributes of the class when they are requested. For example a function to get title, get description, get reviews, get ID num, etc. These functions will return what is contained in the attribute to the point that requests it. There will also be functions for changing a specific attribute of a movie. For example a method to set title, set description, set trailer link etc.

3.5 Non-Functional Requirements

3.5.1 Performance

Response Time on Transactions: Purchases should be completed within a fraction of a second and processed to give customers their orders quickly and effectively.
Response Time on Logins: Users should be able to login within a fraction of second and be able to handle 1000 users at one time.
Capacity of Users: Over a million users should be able to be stored in a database with usernames and passwords.
Capacity of Movies: Over 10,000 movies should be stored on a database and hold 1000 theaters that play these movies.

3.5.2 Reliability

The platform should be up and running constantly. Downtime should be limited to one minute per day of the year. Platform slowdown should be handled upon releases of anticipated movies and account for large platform traffic with slowdown messages, waistline, and place users in a queue.

3.5.3 Availability

The system should be available at all times except for scheduled maintenance that will be performed during non-peak hours to minimize the impact on users using the platform. The system will be backed up daily as to be a fall back in case a failure happens with the database for the website.

3.5.4 Security

The user data that gets transmitted over the internet will be encrypted using the industry standard encryption protocols. Authorization per user will also be employed to ensure that users can only access functions that are permitted for them.

3.5.5 Maintainability

The program should be assembled in a modular fashion to enable certain pieces of the program to undergo maintenance without disrupting the entire system. To ensure easy updates there should be documentation for all API's that are used as well as thorough comments in the code.

3.5.6 Portability

The platform will be able to be responsive, dynamic to the device being used, and accessible across all up to date web browsers and mobile devices. The platform will be laid out corresponding to the device it is being accessed from such as a desktop, laptop, tablet, or mobile cellular smart device.

3.6 Inverse Requirements

The platform should adapt to each device the layout should adapt to the device the platform is being accessed at.

The platform should be responsive and not be unusable on each device and be fast and fluid for users to buy and browse movies on the platform.

Complete 3.7 - 4 for 2nd assignment.

3.7 Design Constraints

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

Some common design constraints that the app must follow the Payment Card Industry Security Standard, comply with data protection laws, adhere to company specific coding standards, and integrate with company databases. The software system must also fit UX standards such as screen resolution adaptation and adhere to UX design principles. If the app parties with any outside resources there should be reservations for the third-party advisors. Pay close attention to budget constraints along to ensure overspending does not occur. Lastly, network limitations can factor to poor performance. An engineer must have a thorough understanding about the bandwidth around their service.

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.

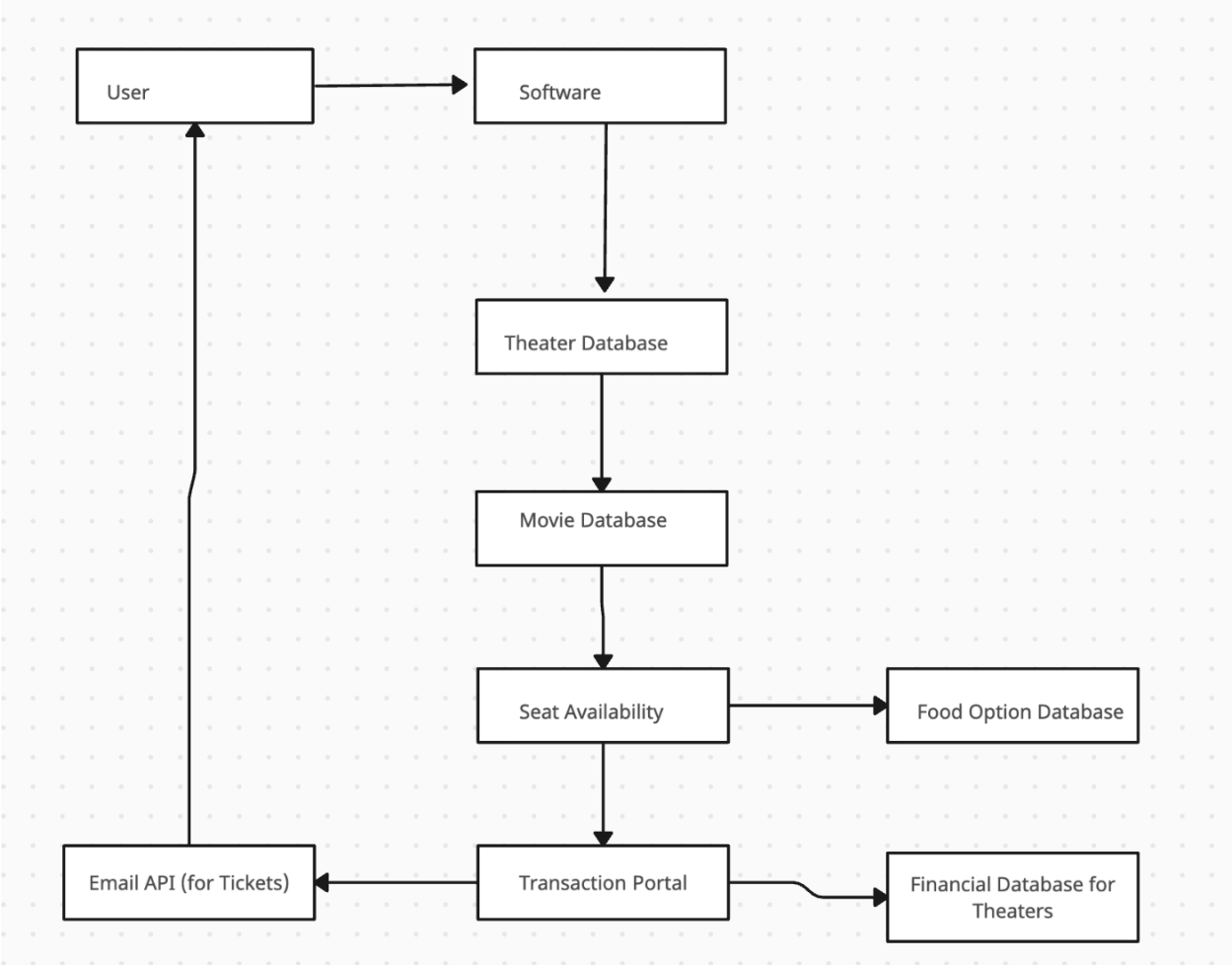
4. Analysis Models

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.

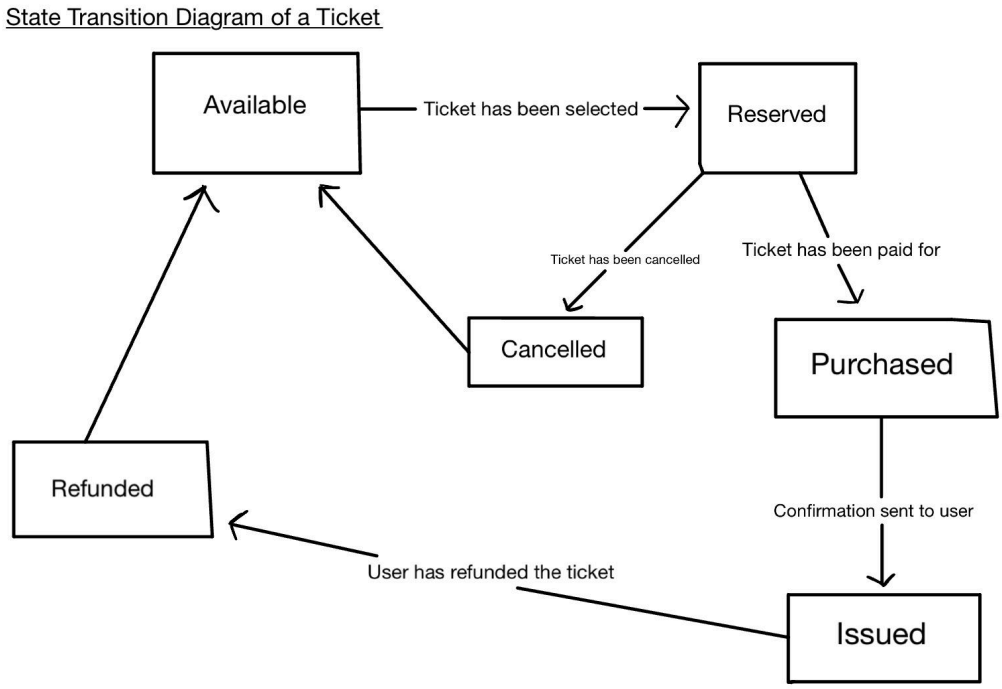
4.1 Sequence Diagrams

Functionality: The software will provide a user with a medium to browse theaters and the movies each theater offers. Along with the ability to browse showtimes, seating availability, and food options at each theater. The

External Interfaces: The software will be accessible through up-to-date and supported browsers such as Chrome, Firefox, Edge, and Internet Explorer. The browser will be connected via a server to access a database of available theaters in the user's area, along with the database of movies in the theater they choose. The user will also be connected to a secure server to provide billing information to complete the transaction of tickets for the movie they select.


User
Software
Theater Database
Movie Database
Seat Availability
Food Option Database
Email API (for Tickets)
Transaction Portal
Financial Database for Theaters

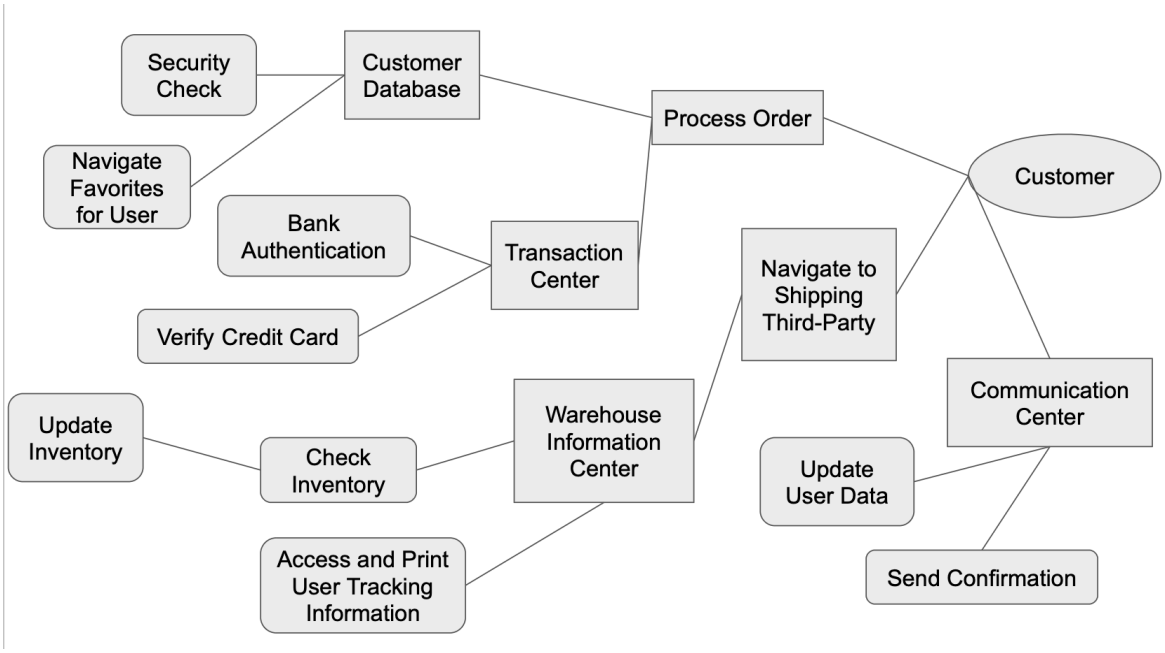
4.2 State-Transition Diagrams (STD / (SWA))



The State Transition Diagram of the ticket describes the certain states that the ticket undergoes when a user is in the process of purchasing, canceling, or refunding a ticket. We start out with the

ticket being available to anyone for purchase. When a user selects the ticket, the ticket's state is now reserved, meaning the ticket is reserved currently by whoever has selected it and it is not currently available to another user. After the ticket is reserved its state will either change to purchased or canceled, depending on the user's course of action. If the user decides to back out of the purchase the ticket's state will be changed to canceled, then back to available. If the user decides to purchase the ticket, after the user's payment details are confirmed the money is received, the ticket's state will be changed to purchased, then once the confirmation is sent to the user via email, the ticket's state will be changed to issued. If the user chooses to refund the ticket, its state will be changed to refunded, then back to available.

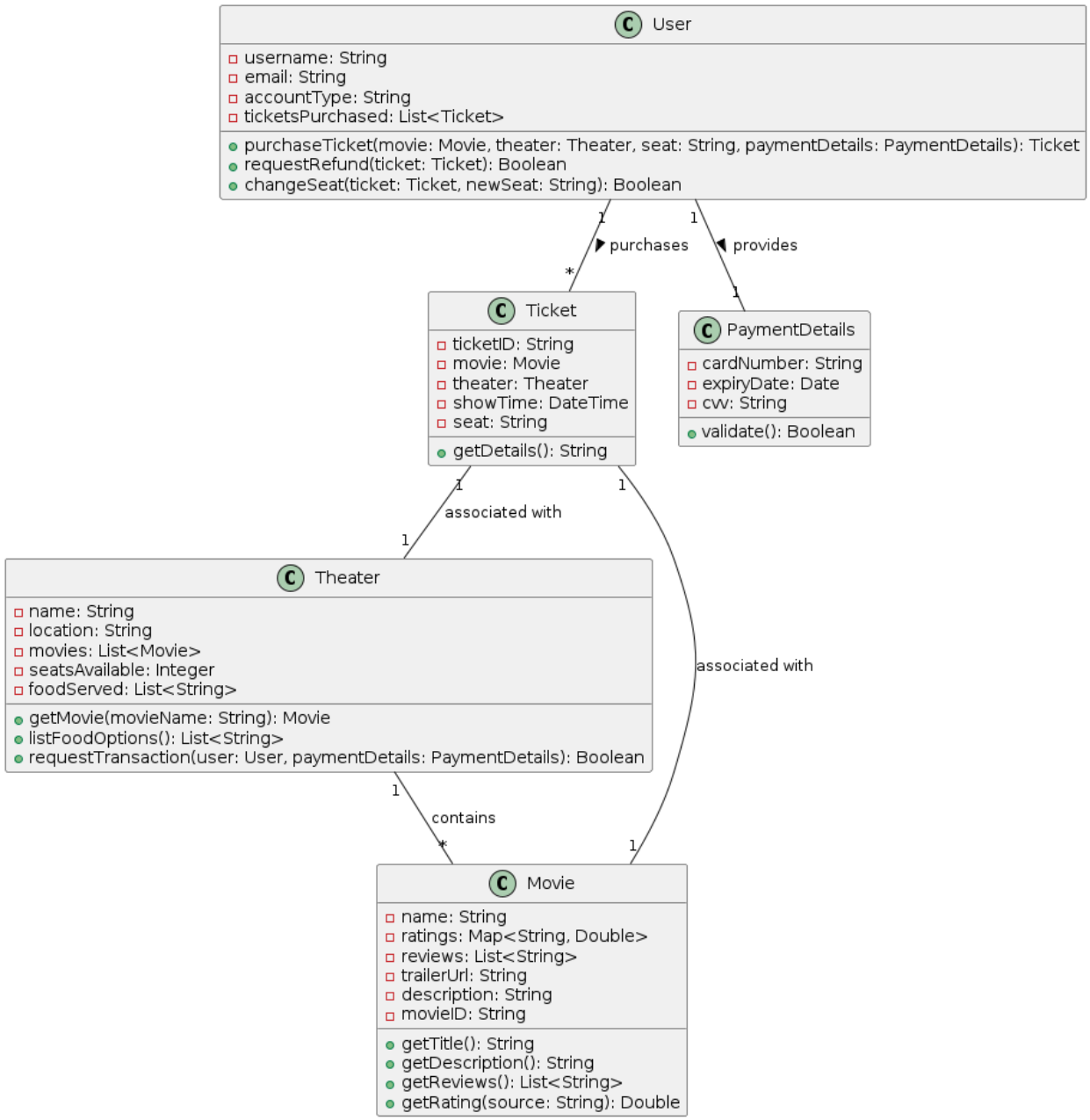
4.3 Data Flow Diagrams (DFD)



The data flow diagram for Encore Ticketing Service depicts the center of most major data transfers. It all starts with the customer. Their information should be protected through each stage. The customer order processing, shipping and contact are the three priorities he or she will face first. Each step holds a larger group of sub-steps that are purposely hidden from the user. This is to not overload the user with information. The processing of the order must connect the transaction point which later branches into bank transparency and final purchases. The order

processing also must touch the users database which helps the app remember and collect the customer's product information securely. The shipping aspect of the transaction must work alongside a third party shipping company. Here the shipping company passes along the user's location and purchase history to award the correct product to the user. Then the company must update both their product and the app's information. The last of the three sections is the communication center. Here a speedy update of the users data and any confirmation or follow up messages should be sent to the customer to ensure they are aware of each step of the process explained above.

4.4 UML Class Diagram



The UML Class Diagram for the Encore Ticketing Service provides a structured overview of the system's architecture, highlighting the relationships and interactions between its core components: Theater, Movie, User, Ticket, and PaymentDetails. In this diagram, Theaters contain multiple Movies and offer various food options, alongside managing seat availability. Movies are characterized by attributes such as name, ratings, reviews, and a unique ID, facilitating user engagement and selection processes. Users, differentiated by account types, can purchase Tickets, request refunds, or change seats, indicating a dynamic interaction with the system based on their needs. Tickets serve as a bridge between Users, Movies, and Theaters, encapsulating details like showtime and seating. PaymentDetails are crucial for transaction validation, securing the financial interactions within the service. This diagram effectively maps

out the system's functionality, from movie selection to the final ticket purchase, underlining the service's aim to provide a seamless ticketing experience.

stop here 🙂

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