Movie Theatre Ticketer

Software Requirements Specification

Version 5

4/10/2025

Group 9
Nicolas Mendoza
Alex Franco
Jackson Frankel

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

Revision History

Date	Description	Author	Comments
2/19/2025	Version 1	Nicolas Mendoza	Finished Ch. 1
2/19/2025	Version 2	Alex Franco	Finished Ch. 2 and 3
2/19/2025	Version 2	Jackson Frankel	Finished Ch. 2 and 3
2/19/2025	Version 2	Nicolas Mendoza	Finished Ch. 2 and 3
3/5/2025	Version 3	Alex Franco	Finished and added SDS
3/5/2025	Version 3	Nicolas Mendoza	Finished and added SDS
3/19/2025	Version 4	Nicolas Mendoza	Added test cases
3/20/2025	Version 4	Alex Franco	Updated document
3/20/2025	Version 4	Jackson Frankel	Updated document & Test cases plan
4/10/2025	Version 5	Nicolas Mendoza	Added architectural diagram and data management strategy

Document Approval

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

Signature	Printed Name	Title	Date
	Nicolas Mendoza	Software Eng.	2/19/2025
	Dr. Gus Hanna	Instructor, CS 250	
	Alex Franco	Software Eng.	2/19/2025
	Jackson Frankel	Software Eng.	2/19/2025
	Alex Franco	Software Eng.	3/5/2025
	Jackson Frankel	Software Eng.	3/5/2025
	Nicolas Mendoza	Software Eng.	3/5/2025
	Alex Franco	Software Eng.	3/20/2025
	Jackson Frankel	Software Eng.	3/20/2025
	Nicolas Mendoza	Software Eng.	3/20/2025
	Nicolas Mendoza	Software Eng.	4/10/2025
	Alex Franco	Software Eng	4/10/2025
	Jackson Frankel	Software Eng	4/10/2025

Table of Contents

1. Introduction

The introduction of the Software Requirements Specification(SRS)[1] provides an overview of the SRS along with the purpose, scope, definitions, acronyms, abbreviations, references, and an overview. The purpose will describe exactly what this SRS document is trying to achieve. The scope will help introduce the software products that are to be used in development, what they will and will not do, as well as a description of the product. The next section is meant to define any acronyms or terms from throughout the document. The references will contain any other texts mentioned in the SRS that need to be cited. Finally the overview will go over the rest of the SRS as well as describe how it will be organized. This document will also heavily rely on the IEEE guide to SRS[1], see References section. Overall the detailed requirements of the Movie Theatre Ticketer(MTT)[2] system will be contained in this document.

1.1 Purpose

The purpose of this document is to provide a specific goal and description of the product the software customers want. Along with appealing to the customers this document also wishes to help the software suppliers and software designers understand exactly what needs to be designed as well as how they should go about doing it. This SRS hopes to be a clear guide for the clients, developers, and managers to help alleviate any misunderstandings as well as streamline efficiency to result in an amazing finished product as timely as can be done.

1.2 Scope

The MTT is an application designed to allow customers to book, purchase, and manage their movie tickets online. The application will provide an efficient, user-friendly, and secure method for customers to look for movie listings, select times to watch, reserve their seats, and complete payments online with credit card, paypal, zelle, venmo, and bitcoin. The application will allow users to create an account in order to gain loyalty points, save purchase history, and save payment and personal information. The system will also be able to have language support for English, Spanish, Chinese, Vietnamese, Tagalog, and Swedish as they are some of the most popular languages in the US. The MTT will also be accessible via a desktop or mobile device in order to maximize convenience. What our system will NOT include is handling of external transportation or parking nor access to streaming services or downloads through the application. These specifications will be mentioned more in depth later in the SRS.

1.3 Definitions, Acronyms, and Abbreviations

1. Software Requirements Specification (SRS)
2. Movie Theatre Ticketer (MTT)

1.4 References

1. IEEE Recommended Practice for Software Requirements Specifications, Developed by the Institute of Electrical and Electronics Engineers, June 25th, 1998

1.5 Overview

The rest of the SRS is split into four sections with appendices at the end. Section two will cover a general description of the product including its functions, characteristics, as well as general constraints. Section 3 will provide specific requirements for the application. This includes external interfaces that will be required, functional requirements, two use cases, classes and objects, non-functional requirements, inverse requirements, some design constraints, database requirements, and any other requirements needed. Then the fourth section lists the analysis models that were used in developing any requirements contained in the SRS. Specifically there will be three kinds of models shown, sequence diagrams, data flow diagrams, and state-transition diagrams. Finally the fifth section will cover the change management process which is how exactly the SRS will be updated when needed as requirements change, as well as some details on by who and how they can be changed. There will also be appendices at the end of the SRS which will provide any additional and helpful information about the process and product.

2. General Description

This section of the SRS will describe general factors that will affect the product and its requirements. It will not list specific, concrete requirements but rather describe them to make them easier to understand in section 3.

2.1 Product Perspective

The SRS will be on par with other online movie ticket vendors such as Fandango and Atom Tickets. Similar to those sites we will have a way for the user to create an account to save their personal information or purchase tickets as a guest. Users will get emailed their tickets and will just have to present the tickets at the door of the theater. The MTT will list movies for the user to pick from and after a selection will prompt the user to select their seat and purchase tickets.

2.2 Product Functions

The SRS will be able handle payments of tickets, will have support for the following 6 languages: English, Spanish, Chinese, Vietnamese, Tagalog and Swedish, will convert local currency when purchasing tickets, will keep systemwide daily logs of tickets purchased, allow users to create accounts to store their personal information, accept a variety of payment methods

such as credit card and paypal, will allow users to select and reserve tickets for a specific showing, and allow users to refund or exchange tickets.

2.3 User Characteristics

The User characteristics of the Movie Ticketing system described in this SRS are divided into 3 categories, Guest User, User, and Employee. The difference between the three categories is their status within the company. Users in this ticketing system are customers and therefore do not have similar access to Employees. Unlike Guest Users, Users have information already set up within the ticketing system and can therefore be prompted differently. In this SRS Guest Users do not have this type of information stored within the system and must be prompted differently likewise. Employees are separated from guests in their authority and access in the company. Employees are managers of the website and of the customers and must have the authority to edit or access information and bookings as needed.

2.4 General Constraints

This subsection of the SRS will provide a general description of any items that will limit the developers options for designing the system:

a) Regulator policies;
b) Hardware limitations;
c) Interfaces to other applications;
d) Parallel operation;
e) Audit functions;
f) Control Functions;
g) Higher-order language requirements;
h) Signal handshake protocols;
i) Reliability requirements;
j) Criticality of the application;
k) Safety and security considerations

2.5 Assumptions and Dependencies

The system according to this SRS and its goals will depend upon the hardware of the customer's device whether it be handheld, or grounded. The system will depend upon a contractual agreement between the company and an internet service provider. This system will assume a website location and address has been developed and claimed by the company. The system will depend and assume the customer has direct access to the internet through the customers own means. The system will depend and assume on the employees and customer service access to the internet of their own accord or through the company. The system depends on and assumes the program has access to servers which can be used by the software. The system assumes there are third party payment connectors with valid and working payment systems that the system can access. The system assumes that movie listings, schedules, and pricing will be provided by theaters or a centralized database with timely updates. The system depends on external email and SMS services for sending ticket confirmations and notifications.The system assumes that users have devices capable of running modern web applications and supporting necessary security protocols. The system assumes compliance with data privacy regulations and industry standards

for online transactions. The system assumes that users access the website through a supported web browser. The system assumes that a customer support team will be available to handle disputes, refunds, or account issues.

3. Specific Requirements

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

Each requirement in this section should be:

- *Correct*
- *Traceable (both forward and backward to prior/future artifacts)*
- *Unambiguous*
- *Verifiable (i.e., testable)*
- *Prioritized (with respect to importance and/or stability)*
- *Complete*
- *Consistent*
- *Uniquely identifiable (usually via numbering like 3.4.5.6)*

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

3.1 External Interface Requirements

3.1.1 User Interfaces

This System will provide a uniform and well blended look.
This System will have a readable and easily navigable menu.
This System will have pictures and text for each film or event.
This System will incorporate the use of icons and toolbars.

3.1.2 Hardware Interfaces

Since the application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g. Modem, WAN – LAN, Ethernet Cross-Cable.

3.1.3 Software Interfaces

1. The checkout system shall communicate with the Configurator to identify all the available components to configure the product.
2. The checkout system shall communicate with the content manager to get the product specifications, offerings and promotions.
3. The checkout system shall communicate with multiple third party payment systems to identify available payment methods , validate the payments and process payment.
4. The checkout system shall communicate with a customer support system to provide support.
5. The checkout system shall communicate with Sales system for order management.
6. The checkout system shall communicate with an sms system to text and email customers.
7. The checkout system shall communicate with external Tax system to calculate tax.
8. The checkout system shall communicate with export regulation system to validate export regulations.
9. The system shall be partnered with a security system to ensure safe payments.
10. The system shall work with an AI detection system to prevent overbooking and fake sales.
11. The booking and content system will work with theatre systems directly to edit, add, delete, and change times or content on their locations interface.

3.1.4 Communications Interfaces

The system will utilize electronic forms to communicate with customers for booking reservations and customer support.

The system will send confirmation emails for ticket purchases, refunds, and account-related notifications.

The system may send customers SMS alerts for booking confirmations, reminders, or urgent updates.

The system will communicate with third-party payment processors and movie databases via secure API calls.

Users can submit support requests through an integrated help desk or email-based ticketing system.

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 <Search for movie>

3.2.1.1 The System shall allow searching for a specific movie
3.2.1.2 The system shall allow the user to select a movie
3.2.1.3 The system shall take the user to a new page after a movie is selected
3.2.1.4 The system shall notify the user about any problems

3.2.2 <Select movie>

3.2.1.1 The system shall display all movies available to be seen
3.2.1.2 The system shall allow the user to select a movie
3.2.1.3 The system shall take the user to a new page after a movie is selected
3.2.1.4 The system shall notify the user about any problems

3.2.3 <Select seating>

3.2.1.1 The system shall will display all available seats
3.2.1.2 The system shall allow the user to select up to 20 seats
3.2.1.3 The system shall take the user to a new page after a movie is selected
3.2.1.4 The system shall produce an error message if seat is already taken

3.2.4 <Purchase tickets>

3.2.1.1 The system shall allow the user to purchase tickets
3.2.1.2 The system shall take the users payment method to purchase the tickets
3.2.1.3 The processing will differ depending on payment type
3.2.1.4 The system will display a confirmation if successful
3.2.1.5 The system shall display an error if payment does not succeed

3.2.5 <Create account>

3.2.1.1 The system shall allow the user to create an account
3.2.1.2 The system will take in the users personal information
3.2.1.3 The system shall send the information to a server to be saved
3.2.1.4 The system shall display a confirmation and keep user signed in if successful
3.2.1.5 The system shall display an error if account could not be created

3.2.6 <Checkout as guest>

3.2.1.1 The system shall allow the user to purchase tickets without an account
3.2.1.2 The system shall take the user's name, email, and payment method
3.2.1.3 The system shall store the information for a period of time after the movie showing
3.2.1.4 The system shall display a confirmation if tickets were successfully purchased
3.2.1.5 The system shall display an error if there is an error

3.2.7 <Allow refunds or exchanges>

3.2.1.1 The system shall display an tickets eligible for refunds or exchanges
3.2.1.2 The system shall allow the user to select any tickets available for refunds or exchanges
3.2.1.3 The system shall will a confirmation if the ticket was successfully refunded or exchanged
3.2.1.4 The system shall display an error if ticket could not be refunded or exchanged

3.2.8 <Offer rewards and promotions>

3.2.1.1 The system shall display all available promotions to the user

3.2.1.2 The system shall allow the user to select any available promotion

3.3 Use Cases

This section will outline multiple use cases, basically a user's interaction with the application accomplishing a certain goal.

3.3.1 Use Case #1, Normal Use Case, Trying to buy a ticket

The user will enter the application and search for their movie by navigating to the search bar. After finding their desired movie they will click the sign in button and put in their username and password to log into their account. The user will then continue by clicking the chosen movie to select a showing time and theatre they wish to watch at. They will see the availability and choose the number of available seats they want. Then they will be able to apply any rewards or gift cards to their purchase. Finally they will pay with their chosen payment method(credit card, paypal, zelle, venmo, or bitcoin) and get an email confirmation containing their QR code of their tickets as well as their new amount of loyalty rewards points.

3.3.2 Use Case #2, Using Loyalty Rewards, Trying to buy tickets with reward points

The user will have received an email after their last purchase notifying them that they have earned enough loyalty points to cover a free ticket. The user will then click a link in the email that will take them to the application sign in page where they will enter their username and password. They will then navigate to the movie they want to watch by searching for it. After finding it they will select their desired showtime and theatre. Then they will choose their seat from the available ones and move onto the payment section. This will be the section where they can apply their loyalty rewards which will make the ticket be free. Finally they will confirm their payment and receive an email confirmation containing their QR code ticket and new value of their loyalty rewards points.

3.3.3 Use Case #3, Returning a Ticket, Returning a ticket by logging into their account

The user has their email confirmation of their ticket but realizes they are actually busy that day and decide they want to return a ticket. Thankfully they have an account so they navigate to the application and click the sign in button. They then are prompted to enter their username and password to enter their account. After signing in they navigate to their profile where they can see the tickets they have purchased. They go to their most recent ticket which they wish to return and click the ticket. There is then an option on the ticket to return the ticket. They decide they want to return it and click the button. They are then prompted if they want to be refunded or want loyalty points in equal value to their ticket. They decide they want to be refunded on their credit card so they select the refund option. Their payment information is saved so they just select confirm and they then receive the money refunded into their account as well as a confirmation email stating the amount refunded.

3.4 Classes / Objects

3.4.1 Users

The customers who want to buy tickets using our application.

3.4.1.1 Attributes
The username of their account.
The password of their account.
The profile picture of their account.
Loyalty point amount.
Payment information.
Personal information.

3.4.1.2 Functions
Searching and looking for movies on application.
Viewing seats and prices on application.
Purchasing seats with different payment methods.
Refunding tickets for loyalty points or money.
Receiving confirmation emails containing QR tickets for movies as shown in use cases.

3.4.2 Administrator

The workers at the theatre who have authorization to edit the system.

3.4.2.1 Attributes
The username of their account.
The password of their account.
Theatre they belong to.
Employee level.

3.4.2.2 Functions
Editing prices for their theatre tickets.
Editing seat availability for movies.
Editing movies that are being shown in their theatre
Allowed to issue refunds to any user

3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

3.5.1 Performance

a) The system should be quick to respond to user input; <1 second
b) The system shall be able to handle many transactions quickly; 1000 per second
c) The system shall be able to be upgraded to handle more users and increasingly heavier traffic
d) System downtime may not exceed 1 minute per day unless there is emergency or scheduled maintenance

3.5.2 Reliability

a) The system should process payments clearly and efficiently.
b) The system should provide payment confirmation and booking confirmation.
c) The system should provide ticket access and any information regarding an order within the website.
d) The system should display ticket data and order data to employees.
e) The system shall provide storage of all databases on redundant computers with automatic switchover.
f) The system shall provide for replication of databases to off-site storage locations.

3.5.3 Availability

a) The system should be available to anyone with an internet connection and a browser

3.5.4 Security

a) The system shall perform authentication when a user tries to sign in
b) The system shall encrypt user data to prevent security breaches

3.5.5 Maintainability

a) The system must be edited with new data for movie listings or events daily.
b) The system must be updated per order or edit for each booking.
c) The system must be updated on new Interface designs or system bugs.

3.5.6 Portability

a) The system shall be available on mobile devices and on computer

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

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

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

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.

6. Software Design Specification

Nicolas Mendoza, Alex Franco, Jackson Frankel

6.1 System Description

The Movie Theatre Ticketer (MTT) is an online movie ticketing system designed to provide customers with a seamless and convenient way to browse movie listings, select showtimes, reserve seats, and purchase tickets for upcoming screenings. The system will be accessible via both desktop and mobile devices, allowing users to make reservations from anywhere with an internet connection.

The primary goal of the MTT system is to improve the movie ticket purchasing process by reducing wait times at theaters, enhancing the booking experience, and offering users flexible payment options. Users will be able to create accounts, track their purchase history, and earn loyalty rewards for future ticket purchases. The system will also offer guest checkout, allowing customers to complete transactions without an account.

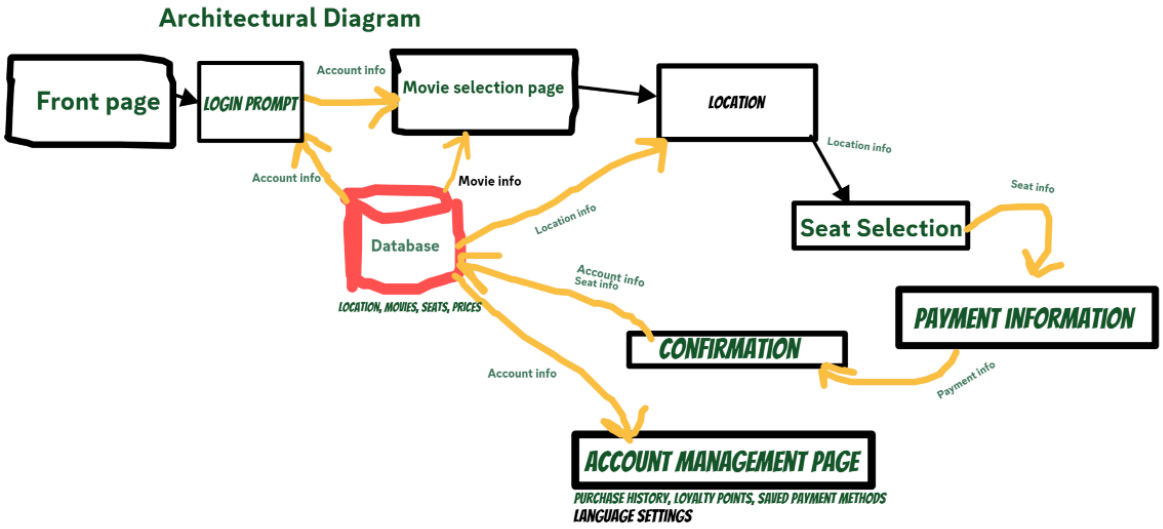
To ensure accessibility, the MTT system will provide multi-language support in English, Spanish, Chinese, Vietnamese, Tagalog, and Swedish. Security is a primary concern, and all payment transactions will be processed securely using industry-standard encryption. The system will integrate with third-party services for secure payments, email notifications, and movie schedule updates.

The MTT system will also include an administrative dashboard that allows movie theater employees to manage movie schedules, seat availability, ticket pricing, and refunds. Administrators will have access to booking data and system reports to optimize theater operations.

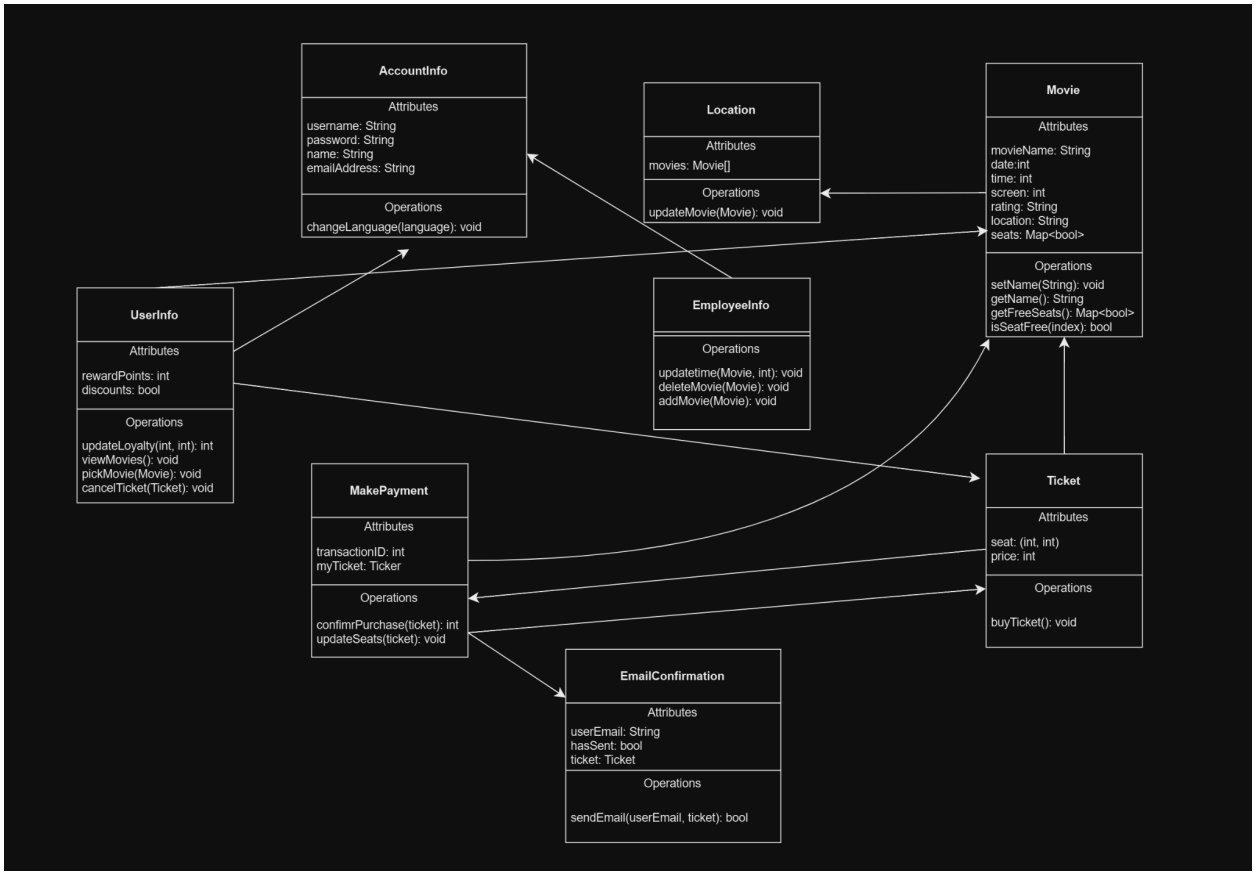
The following subsections will further describe the system's software architecture, design, and development plan.

6.2 Software Architecture Overview

6.2.1 Architectural Diagram



6.2.2 UML Class Diagram



6.2.3 Description of classes

- Movie will have movieName, the date and time of each showing, the location, and seats available as variables. 6.2.5 will list all of the functions
- AccountInfo will have username, password, name and email address as variables. 6.2.5 will list all of the functions
- UserInfo class will have rewardPoints and discountStatus as variables. 6.2.5 will list all of the functions
- MakePayment will have transactionID and myTicket as variables. 6.2.5 will list all of the functions
- Ticket will have seat and price as variables. 6.2.5 will list all of the functions
- EmployeeInfo will have no variables
- Location will have movies. 6.2.5 will list all of the functions
- EmailConfirmation will have userEmail, hasSent, and ticket as variables. 6.2.5 will list all of the functions

6.2.4 Description of attributes

Movie: Movie name and location will be a string, date and time will be an integer.
AccountInfo: All of them will be strings
UserInfo: rewardPoints will be an integer and discountStatus will be a boolean
MakePayment: transactionID will be an integer and myTicket will be a Ticket
Ticket: both will be integers

Location: movies will be Movie[]
EmailConfirmation: userEmail will be a string, hasSent a boolean, and ticket will be a Ticket

6.2.5 Description of operation

Movie Class
setName(String name): Sets the name of the movie.
getName() → String: Returns the name of the movie.
getFreeSeats() → Map<bool>: Returns a mapping of available seats.
isSeatFree(int index) → bool: Returns a boolean indicating whether the seat at the specified index is available.
Location Class
updateMovie(Movie movie): Updates the movie object with any changes that may occur, including modifications to its attributes.
Ticket Class
buyTicket(): Uses the MakePayment class to finalize the ticket purchasing process and reserves the selected seat.
EmployeeInfo Class
updateTime(Movie movie, int newTime): Updates the movie's scheduled time to the provided integer value.
deleteMovie(Movie movie): Removes the movie from the system, making it unavailable for ticket purchases.
addMovie(Movie movie): Adds a new movie to the system, making it available on the main page for selection.
AccountInfo Class
changeLanguage(String language): Allows the user to update their preferred language settings.

6.3 Development Plan and Timeline

This section outlines the division of tasks, responsibilities, and timeline for the development of the Movie Theatre Ticketer (MTT) system. The workload is evenly distributed among the three team members: Jackson Frankel, Nicolas Mendoza, and Alex Franco.

6.3.1 Partitioning of tasks

Task	Assigned Team Member(s)	Description
Software Architecture Design	Jackson	Create the **architectural diagram** outlining the system’s major components.
UML Class Diagram & Documentation	Alex	Design the **UML Class Diagram** and provide descriptions of each class, attribute, and method.

Development Environment Setup	Nico	Set up the **GitHub repository**, define **coding standards**, and configure the project environment.
User Management System	Jackson	Implement user-related functionality, including **account creation, login, and loyalty points tracking**.
Movie Listing & Selection	Alex	Implement **movie searching, filtering, and selection** features.
Seat Reservation System	Nico	Develop the **seat selection and availability tracking** feature.
Ticket Purchase & Payment System	Jackson	Implement the **payment processing system** and integrate third-party payment services.
Refunds & Exchanges System	Alex	Develop the **refund and exchange process** for tickets.
Notifications System (Email & SMS)	Nico	Implement the **email and SMS notification system** for confirmations and updates.
Administrator Dashboard	Jackson	Create the **admin panel** for managing movies, refunds, and user accounts.
Testing & Debugging	All Members	Perform **unit testing, integration testing, and debugging** to ensure system stability.
Final Documentation & Report	All Members	Compile the **final project report**, including all previous assignments.

6.3.2 Development timeline

Week	Milestone	Tasks Completed	Responsible Members

Week 1	**Project Setup**	GitHub repository created, environment configured	Nico
Week 2	**System Architecture**	Architectural diagram finalized	Jackson
Week 3	**UML Class Diagram**	UML diagram completed and documented	Alex
Week 4-5	**User & Movie Systems**	User management, movie browsing, and selection implemented	Jackson, Alex
Week 6	**Seat Reservation System**	Seat selection and availability tracking completed	Nico
Week 7-8	**Payment & Ticketing**	Payment integration and ticket purchase system finalized	Jackson
Week 9	**Refunds & Exchanges**	Refund and ticket exchange system implemented	Alex
Week 10	**Notifications System**	Email and SMS confirmations integrated	Nico
Week 11	**Admin Dashboard**	Administrator features implemented	Jackson
Week 12	**Testing & Debugging**	System testing and debugging	All Members
Week 13	**Final Report Submission**	Compile and submit final project report	All Members

7. Test Plan

7.1 Verification and Validation Tests

Purpose

The purpose of this test plan is to ensure that the Movie Theatre Ticketer (MTT) system meets the functional and non-functional requirements defined in the Software Requirements Specification (SRS). Testing will help verify the correctness, usability, and robustness of the system through unit, functional, and system-level test cases.

Scope of Testing

This test plan covers the major features of the MTT system, including:

- User account creation and login
- Movie search and selection
- Seat reservation
- Ticket purchasing using multiple payment options
- Refunds and exchanges
- Guest checkout
- Loyalty rewards
- Email confirmation system

Test Strategy

We used a **three-level testing approach**:

Unit Testing

Tests individual functions or methods, such as buyTicket(), getFreeSeats(), and updateMovie().

Functional Testing

Verifies whether major features (e.g., "Search Movie", "Purchase Ticket") behave according to the requirements.

System Testing

Tests the entire flow from movie selection to payment confirmation including edge cases like payment failure.

Test Design and Mapping to Requirements

Each test case is tied to one or more requirements from Section 3.2 of the SRS. We tested the following:

Feature	Requirement	Test Focus
Search Movie	3.2.1	Input validity, navigation to detail page
Select Seating	3.2.3	Seat availability logic
Purchase Ticket	3.2.4	Payment processing and error handling

Create Account	3.2.5	Data validation and confirmation
Guest Checkout	3.2.6	Temporary data handling
Refunds	3.2.7	Eligibility and flow
Rewards	3.2.8	Points calculation and usage
Admin Edits	SDS 6.2.5	Employee privilege functions

Test Environment

- **Frontend:** Web browser (Chrome, Firefox, Safari)
- **Backend:** Local development server or test hosting environment
- **Tools:** GitHub for version control, Excel for test case tracking
- **Dependencies:** Email service API, Payment service API

Test Cases Overview

The test cases are stored in the attached TestCaseSamples.xlsx file.

Coverage Summary:

Test Type	# of Cases
Unit Tests	4 (e.g., buyTicket(), changeLanguage())
Functional Tests	4 (e.g., Search, Checkout, Refund, Rewards)
System Tests	2 (e.g., Full booking + reward, Guest checkout with invalid payment)

Responsibilities

- **Alex Franco** – Refund system, Class Diagram
- **Nicolas Mendoza** – Email confirmation system, Seat selection
- **Jackson Frankel** – User system, Full payment flow
  All members contributed test cases and pushed commits to GitHub.

Summary
The test plan for the Movie Theatre Ticketer (MTT) system outlines the verification and validation process to ensure functionality, reliability, and security. The plan covers critical features such as user account creation, movie search and selection, seat reservation, ticket purchasing, refunds, guest checkout, loyalty rewards, and email confirmations. The test cases are categorized into unit, functional, and system

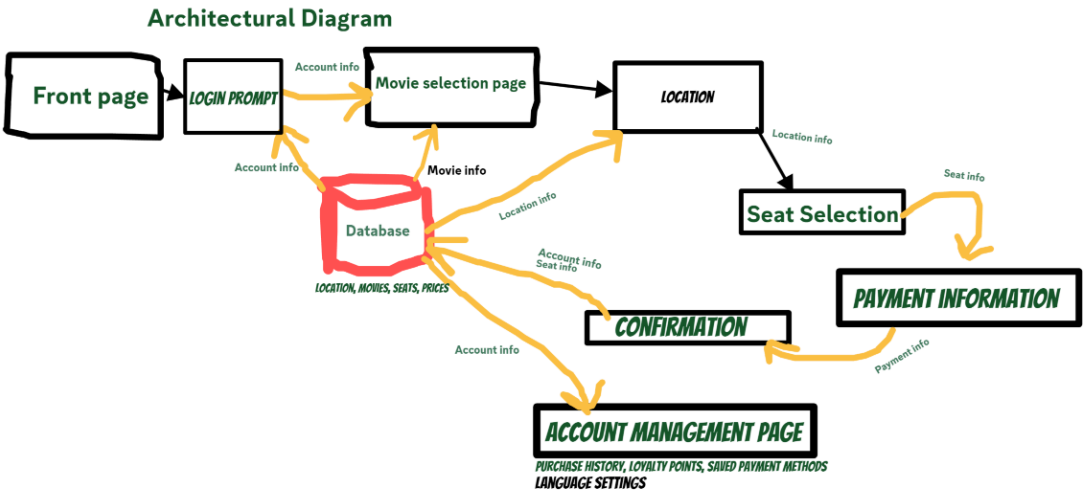
tests, with a focus on testing individual components, feature-level interactions, and full system workflows. The test set includes specific test vectors targeting expected behaviors, input validation, error handling, and performance. Modifications to the system's design diagram may be used to indicate the scope and targets of these tests, covering potential failures like invalid inputs, system crashes, and payment errors. Testing will be conducted in a controlled environment using a web browser-based front-end and a backend server, incorporating third-party integrations such as payment gateways and email services. All test cases and results will be documented in TestCaseSamples.xlsx and stored in the GitHub repository for validation and grading.

7.2 Sample Test Cases

Sample Test Cases here

8. Architecture Design

8.1 Architecture Diagram



8.2 Data Management Strategy

8.2.1 Why a SQL database

We will store our data on an SQL based database due to how straightforward and effective the syntax is, the ability to have a variety of data types, and the capacity for sophisticated queries. Additionally we will split our database into two, an account database and a movie database. This allows us to minimize the amount of databases as well as efficiently and quickly access our data

for accounts or movies. This also keeps the accounts separate from the movies in order to ensure more security as well.

8.2.2 Account database

We will store all of our customers' personal information in one database. The user's email, username, password, rewards points, and payment methods.

8.2.3 Movie database

We will have all the movie information stored in another database. The movie titles, runtimes, and locations.

8.2.4 Alternative data management strategies

In contrast to our primary SQL-based database approach, a NoSQL-based database could provide greater horizontal scalability. This might be more suitable in scenarios requiring high-volume data handling across distributed systems. However, NoSQL solutions often require additional infrastructure, increased power consumption, and more complex maintenance due to the use of multiple distributed nodes.

Alternatively, a cloud-based data management solution could be used, offering enhanced scalability, accessibility, and reduced need for on-site maintenance. This approach would improve flexibility for customers by allowing real-time data access and updates across various platforms.

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