Movie Ticketing

Software Requirements Specification

Version 1.0

May 27, 2024

Group 13

David Hernandez

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

Revision History

Date	Description	Author	Comments
May 27, 2014	Version 1.0	David Hernandez	Submitted for initial review.

Document Approval

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

Signature	Printed Name	Title	Date
	<Your Name>	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to outline the design and development requirements of the Movie Ticketing product. This document describes the customer needs, product functionality, and development requirements to evaluate development efforts, validation, and provides overall familiarization.

1.2 Scope

This product will be a Movie Ticket purchasing system. It will inform users of current movies playing in theaters and facilitate ticket purchasing. The site will allow for account creation, payment information storage, and ticket downloading for ease of access.

This product is designed to optimize the ticket purchasing workflow through a fluid and efficient user interface increasing ticket sales and customer loyalty.

1.3 Definitions, Acronyms, and Abbreviations

HTML – Hyper Text Markup Language
CSS – Cascading Stylized Sheets
AWS – Amazon Web Services
RDS – Relational Database Services
API – Application Programming Interface

1.4 References

1. IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

1.5 Overview

The following content elaborates on each software requirement component required for the software to be successfully implemented. This information involves product functionality, constraints, external and non-functional requirements, use cases, and analysis models. This information covers multiple disciplines, but each is crucial to the success of this system.

2. General Description

2.1 Product Perspective

The product allows for movie tickets to be purchased online.

2.2 Product Functions

The product is an online website that displays available theater movies and allows secure purchasing of tickets over the web. Users can create an account and save their information online.

2.3 User Characteristics

The product shall be operatable by people with a basic reading level and basic computer knowledge.

2.4 General Constraints

Constraints to consider are the user's available bandwidth. Slow internet providers or mobile device services in a poor coverage area may cause slow connections or loss of connection to database server.

2.5 Assumptions and Dependencies

It shall be assumed every time the page is accessed, users with accounts will be purchasing at least one movie ticket.

The system depends on Amazon Web Services for database handling, Square APIs for purchasing, and Google APIs for maps and location services.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Front-end: HTML, CSS
Backend: JavaScript, PHP

3.1.2 Hardware Interfaces

Windows
iOS
Internet Browsers with HTML & JavaScript interpreters

3.1.3 Software Interfaces

Operating System: Windows, iOS
Database: AWS
APIs: Google Maps Embed API, Square Payments API , and Square Customer API

3.1.4 Communications Interfaces

Project is supported on any browser with HTML and & JavaScript interpreters. Basic forms will be utilized to handle, save, and process data entry.

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 Purchasing of Tickets

3.2.1.1 Introduction
The user shall be able to purchase movie tickets.
3.2.1.2 Inputs
The product shall accept text and numerical values to obtain credit card information.
3.2.1.3 Processing
The product shall process credit card payment.
3.2.1.4 Outputs
The product shall output a confirmation email.
3.2.1.5 Error Handling
The product shall handle invalid credit card information.
The product shall not process payment due to insufficient funds.

3.2.2 Obtaining Tickets

3.2.2.1 Introduction
The product shall make the tickets accessible to the user.
3.2.2.2 Inputs
The user should be able to retrieve tickets with user’s account information.
3.2.2.3 Processing
The product provides the ticket in a format that is downloadable.
The product shall allow users to retrieve their tickets when requested.
3.2.2.4 Output
The product shall output the movie ticket as a pdf.

3.2.3 Movie Availability

3.2.3.1 Introduction
The product shall display which movies are available.
3.2.3.2 Inputs
The product shall accept a list of movies from the database and add them to the website.
3.2.3.3 Processing
The product shall notify users that all tickets for a movie have been purchased.
3.2.3.4 Outputs
The product shall show that the movie has been sold out.
3.2.3.5 Error Handling
The product shall notify the user if they are trying to purchase a ticket from an unavailable movie.

3.2.4 Display Seating

3.2.4.1 Introduction
The product shall display which seats are available.
3.2.4.2 Inputs
The product shall allow users to select which seats to pick.
3.2.4.3 Processing
The product shall assign the selected seat to the user.
3.2.4.4 Outputs
The product shall output the selected seat to the user’s ticket.
3.2.4.5 Error Handling
The product shall ensure that the user has selected seats.
The product shall ensure the user can only select available seats.

3.2.5 Location Selection

3.2.5.1 Introduction
The product shall allow users to select their desired theater location.
3.2.5.2 Inputs
The product shall accept the user’s address.
3.2.5.3 Processing
The product shall utilize the user’s address to search for nearby theaters.
3.2.5.4 Outputs
The product shall display theaters near the entered address.
3.2.5.5 Error Handling
The product shall validate addresses.
The product shall notify the user when there are no nearby theaters.

3.2.6 Profile Creation

3.2.6.1 Introduction
The product shall allow users to create an account.
3.2.6.2 Inputs
The product shall accept the users’ email address.
The product shall accept a unique username.
The product shall accept the users’ password.
The product shall ensure the users’ password length is greater than or equal to 8 characters.
3.2.6.3 Processing
The product shall save the entered information into a database.
3.2.6.4 Outputs
The product shall send an email to the user to confirm their account was created.
3.2.6.5 Error Handling
The product shall ensure the user has entered a valid email address.
The user shall ensure that the entered username is unique.
The product shall ensure that the password meets the minimum required length.

3.2.7 User Interface

3.2.7.1 Introduction
The product shall ensure easy navigation between pages and purchases.
3.2.7.2 Inputs
The product shall handle button clicks and text input.
3.2.7.3 Processing
The product shall process button clicks for navigating between pages and submitting information.
3.2.7.4 Outputs
The product shall output the user requested pages.
3.2.7.5 Error Handling
The product shall notify missing required entries.

3.2.8 Security

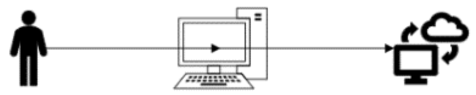
3.2.8.1 Introduction
The product shall encrypt user data.
The product shall ensure to sign out user after 2 minutes of inactivity.

3.2.9 Data Storage

3.2.1.1 Introduction
The product shall save all user data to a database.
3.2.1.2 Inputs
The product shall accept the users’ full name.
The product shall accept the users’ home address.
The product shall accept the users’ email address.
The product shall accept the users’ phone numbers.
3.2.1.3 Processing
The product shall process information from the online form and save it to the database.
3.2.1.5 Error Handling
The form shall only save valid information.

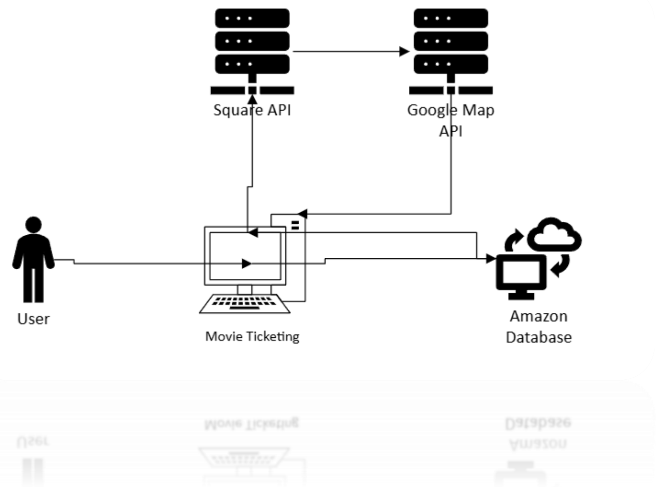
3.3 Use Cases

3.3.1 User creates account



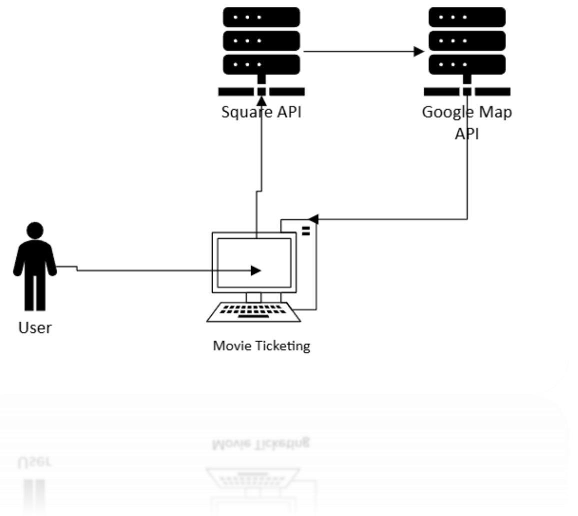
User connects to website and creates an account. Account information is saved to the database.

3.3.2 User purchases ticket as account holder



The user logs into their account, purchases a ticket for their selected theater. The ticket is then sent to them.

3.3.3 User Purchases ticket as guest



The user purchases a ticket from their selected theater. Account information will not be saved, and the user will have to re-enter all information on next purchase.

3.4 Classes / Objects

3.4.1 Movie Ticket

3.4.1.1 Movie Name
3.4.1.2 Ticket Number
3.4.1.3 Show times
3.4.1.4 Purchase Data
3.4.1.5 Expiration Date

3.4.2 User

3.4.2.1 Username
3.4.2.2 Password
3.4.2.3 First Name
3.4.2.4 Last Name
3.4.2.5 House Number
3.4.2.6 Street Name
3.4.2.7 City
3.4.2.8 State
3.4.2.9 Zip code

3.5 Non-Functional Requirements

3.5.1 Performance

The product shall ensure API calls execute only when required,
The product shall connect to the database only when required.

3.5.2 Reliability

The product shall ensure that transactions are performed in a timely manner.

3.5.3 Availability

The product shall be accessible by current internet browsers.
The product shall be available to all users with an internet connection.
The product shall be available 24/7.

3.5.4 Security

The database shall be an encrypted Amazon RDS DB instance utilizing industry standard AES-256 encryption algorithm.

3.5.5 Maintainability

The product's versioning shall be tracked to reduce conflicts and inconsistencies.
The product shall support regular updates.

3.5.6 Portability

The product shall operate on all operating systems with internet browsers.

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

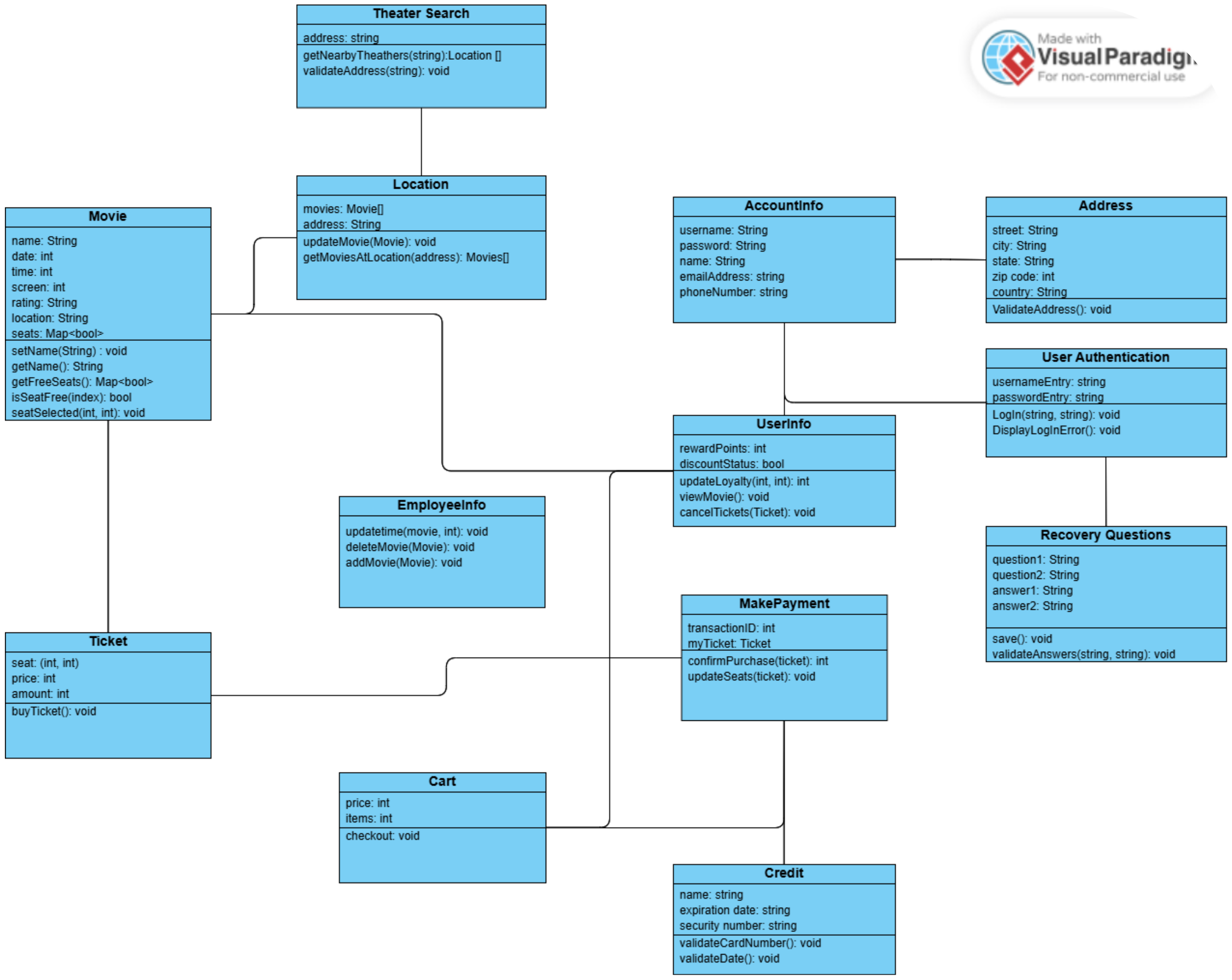
Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

3.9 Other Requirements

Catchall section for any additional requirements.

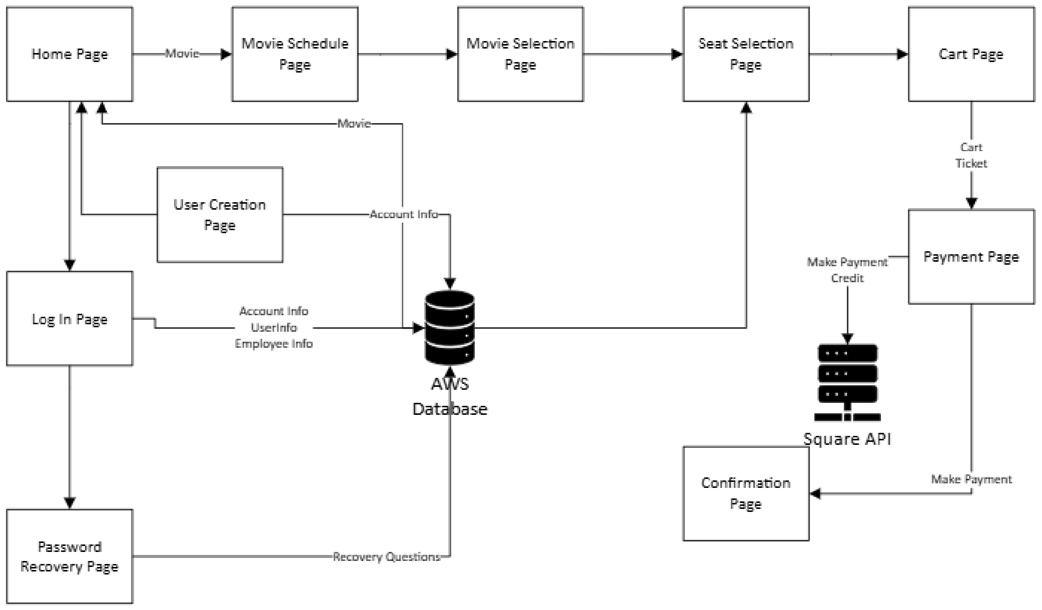
4. UML Diagram

Below is the visual representation of the classes and objects used within the Movie Ticket application. The UML Diagram aims to show the relationship between classes and identify class attributes. The operations displayed below the attributes illustrate the actions required to properly pass parameters and validate user inputs to ensure system functionalities.



5. Architectural Design

The following graphic represents the architectural design of the software. The purpose of the architectural design is to showcase the navigation and data transfer that occurs as the user navigates within the system. Class objects are passed and handled as dictated by user clicks and navigation.



6. Test Plan

6.1 Purpose

The test plan assesses the application's ability to perform as intended by verifying the product meets design specifications and user requirements.

6.2 Overview

The test plan is comprised of various test cases each evaluating a different part of the architecture. The following tests evaluate the application’s ability to perform efficiently. Tests focus on navigation, data validation, data storage, user creation and overall stability of the system. A copy of the test plan can be found at https://github.com/davidmht1/CS250-Repo.

6.3 Tests

6.3.1 Functional Test Case

Users can navigate to the home page.
Users can create an account
Users can log in to their account
Users can navigate to the selected movie’s schedule page
Users can navigate to movie show times page
Users can select seats
Users can select ticket quantities
Users can navigate to the cart
Users can pay with a credit card
Users receive payment confirmation
Users receive payment error
Users selected seats are assigned to seats
Users can search for theaters near them

6.3.2 Unit Test Cases

Test user account creation function
Test user log in function
Test seat update function
Test seat assignment function
Test seat availability function
Test return movie’s function
Test send confirmation function

6.3.3 Validation Test Cases

Validate user home address
Validate user date of birth
Validate user credit card information
Validate user password
Validate user email address
Validate user’s name
Validate users’ password
Notify user when inputs are blank

6.3.4 System Test Cases

Test application operates during high traffic hours
Test application can recover after network connection loss.
Test pages load under 2 minutes with a minimum internet connection speed of 10 Mbps.

6.3.5 Security Test Cases

User password shall not be displayed
Test database encryption
Test user is logged out after 2 min of inactivity

7. Data Management Strategy

7.1 Purpose

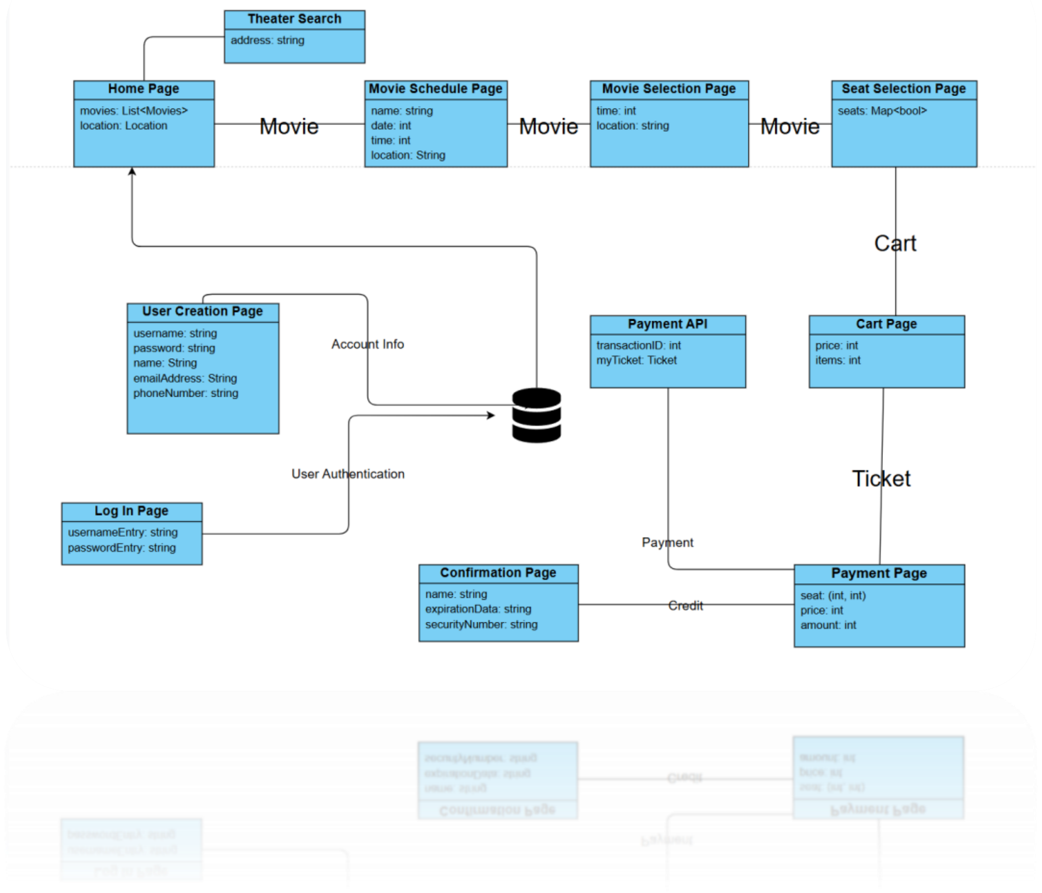
The diagram provides a simplified view of the overall application structure to assist with system comprehension, decision-making, and improve collaboration.

7.2 Overview

Individual application components are displayed. Database objects are outlined as they are passed between components such as pages or API. The entity components are then processed or modified according to the user's input and workflow.

7.3 Design Choices

The application utilizes a SQL Database to store movie, user and payment information. SQL is best suited for consistent data with well-defined relationships between tables and high transaction-based applications. NoSQL databases may be utilized as an alternative, however due to their inability to perform A.C.I.D transactions there is a risk that an unsuccessful transaction may corrupt table data.



8. 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.

8.1 Sequence Diagrams

8.2 State-Transition Diagrams (STD)

8.3 Data Flow Diagrams (DFD)

9. 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