ABC Theater's Movie Ticketing System

Software Requirements Specification

Version 4 (Design 2.0)

November 7, 2024

Group #6
Cameron Lee, Roger Dao, Ross Hackett,
Matthew Dominguez

Github Repository: https://github.com/papabeanstalk/G6-SRS

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Fall 2024

Revision History

Date	Description	Author	Comments
09/26/24	Version 1	Cameron Lee, Roger Dao, Ross Hackett, Matthew Dominguez	Added introduction, general description, specific requirements (excluding design constraints, logical database requirements, and elaboration on other requirements)
10/10/24	Version 2	Cameron Lee, Roger Dao, Ross Hackett, Matthew Dominguez	Addition of: 4.1 Software Architecture Overview (UML Diagram, UML Class Description) 4.2 Software Architecture Diagram (SWA Diagram, SWA Description) 4.3 Development plan and timeline
10/24/24	Version 3	Cameron Lee, Roger Dao, Ross Hackett, Matthew Dominguez	Addition of: 4.4 Test Plan Update of: 4.2 Software Architecture Overview: Added more components to UML Diagram such as the databases and a section for customer support.
11/7/24	Version 4: Design 2.0	Cameron Lee, Roger Dao, Ross Hackett, Matthew Dominguez	Addition of: Databases to Architectural diagram, in-depth data management policy (Data Management Strategy) Update of: 4.2 Software Architecture Diagram Update of design diagram with addition of databases, incl. explanation of modifications from initial design.

Document Approval

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

Signature	Printed Name	Title	Date

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to provide adequate understanding to software engineers and web designers of the functionality and features needed to be implemented for the product as well as a description of the intended user experience the product should deliver, such as navigation of the website, logging/signing in, payment and means of receiving tickets.

1.2 Scope

The product will be an online movie ticketing system that allows users to browse films showing at participating locations. Users should should be able to search by three methods

1. Browsing films sorted by popularity within a user selected location's selected radius.
2. Searching for specific films and all available showings within a user's selected radius.
3. Browse all films showing at a user selected theater.

Users should be able to purchase tickets via Paypal, Venmo, Affirm, and Shoppe, and receive tickets and payment confirmation via email used to login.

1.3 Definitions, Acronyms, and Abbreviations

There are no relevant keywords or uses of language contained in this document that should require defining.

1.4 Overview

The rest of the SRS contains general information on how the product should be organized and function followed by more specific details on how tasks the application should be able to perform and handle are meant to be structured and implemented. The SRS is organized as so to provide a general idea of how the finalized product is intended to look before delving into specific outlines for its individual characteristics.

2. General Description

Online ticket purchasing software (website) for ABC theater's 6 physical locations (they all exist within San Diego.)

The product will allow customers to conveniently remotely reserve seats and purchase tickets online from one of six ABC Movie locations.
Search function with ability to search for ABC Theater by location (zip code), search for a theater with showings of a particular movie by specifying the movie, search by time to view showings at theaters at that certain time, general movie availability showtimes and more..

Our website is designed keeping in mind the things needed to effectively purchase tickets in advance, in a rush, or to skip the wait.

Customers can register on ABC theater's website to save preferred location, payment methods, as well as view purchased tickets easily within the same site, expediting the theater booking process.
Tickets can be purchased as a guest or a user, customers will also receive via email a receipt for their order that includes their digital ticket that can be printed or serve as an entry ticket using a mobile device once scanned via QR code.
Each ticket is unique to prevent fraud.
Payment methods include most accepted credit/debit cards, and other popular payment app methods including Paypal, Venmo, Affirm, and Shoppe.
Customers can submit tickets to receive customer support for issues related to the website's functionality, user account credentials, order details, and order cancellation.
Customer support representatives can login using their credentials and review submitted tickets, cancel orders, issue refunds, or make adjustments to customer's purchases and/or registered account details.

2.1 Product Perspective

Independent and self-contained ticket purchasing website designed to be accessible on common mobile and desktop devices. The main goal is to provide convenience for patrons wanting to order tickets and reserve seating in advance, or patrons looking to skip the line by quickly and easily ordering tickets. Immediately after purchase, printable tickets including scannable QR codes are emailed to the customer, and also available on our website if the customer is signed in.

Considerations are made to handle increased traffic surrounding popular movie releases, and ticket limits are enforced to limit the act of reselling or scalping tickets. For ease of use for both patrons and staff in case of an outage, the ticketing system is meant to add on to an already complete in person system, and not halt operations in case of failure or during maintenance. Popular and verifiable digital payment methods (payment apps) will be accepted as well as commonly used credit/debit cards, the same as in our theaters.

The ticket purchasing system is meant to assist in the daily operations of ABC Theater's business and is designed to be cohesive with preexisting in person operations. Ease of use and identical function to kiosks and booths is a priority to cut down on additional training and limit burden on staff and maintenance personnel at theater locations to reduce overhead as well as to maintain stability. The simplicity of the design is to avoid limiting operations in case of outages, unique threats to the system, including online threats, server downtime, etc. (single point of failure)

2.2 Product Functions

Summary of the functions the software will perform:

To register as a user
Login using credentials of an account registered with ABC movies.

Search for movies by title. Will return locations, showtimes, etc.
Search of theaters by location (zip code)

Return list of movies playing at a theater location
Return list of available showtimes for a theater location

Reserve a ticket for purchase for 10 minutes.

Purchase tickets.

Purchase tickets as guest
Purchase tickets as registered user

Receive customer support for customer account assistance, payment and ticket assistance

Customer Support Representative access to: customer account information, management of payment processes (in the case that refunds or adjustments need to be made.)
Send ticket to email (to resend if there is a need to, will provide the purchased ticket(s) as scannable digital QR code and/or printable ticket)

2.3 User Characteristics

General characteristics of the eventual users of the product that will affect the specific requirements:

Users of the product: People with the desire and ability to purchase tickets online to skip the line in person, reserve tickets and seating ahead of time,
Education level: In or after high school (they are debit/credit card holders)
Experience: By this point they will have some or no experience with ordering online. Our site is simple to use and we maintain in person and over the phone sales taking this into account.
Technical expertise: By this point the user will have sufficient experience with using their personal computer or phone for similar purposes, if not, our site is simple to use and we maintain in person and over the phone sales taking this into account. (such as the customers who do not go online much, users of devices we don't maintain support for)

2.4 General Constraints

Reliability requirements: we have designed our system to be redundant with the functions of in person operations.

- We don't accept cryptocurrency.
- We send confirmation of purchase including the printable ticket & QR code to emails only, customers ordering via phone will have to pick up tickets in person with verification.
- We only list prices in USD as part of our design (our locations are in San Diego only)

Safety and security considerations:

- You need to use a browser, we don't have an app. There is too much overhead related to arbitrary phone operating systems updates involved in maintenance of an app. We choose to use browsers because of this.
- Adherence to protocol involving storage of personally identifying data, including information used in user account registration, payment methods, etc.

As a business, we want to maintain stability of our operations and maintaining expanded functionality of our website contradicts that.

2.5 Assumptions and Dependencies

Assumptions:
Designed to be operated through a desktop/laptop or phone's browser. As such, as long as the device is suitable for day to day use, it is (powerful enough to run youtube on a browser/app, access emails)
The user will encounter frustration if they are accustomed to ordering over the phone and receiving their purchased item in person
The employees will receive the necessary training on how to operate
Dependencies:
We are designing for personal devices x86 desktop/laptops and android/iOS. The user will be using Android, iOS, Windows, macOS, Linux on their personal device.
Infrastructure at our theater locations will require maintenance and support to support
Including training of employees to handle increased functionality, and instructions on what to do if online functions cannot be reached.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Homepage will have a clear display of available movie options, as well as a search function. Movie thumbnails can be clicked on to access more information including summary, showings, and location. Thumbnails can also be scrolled horizontally for browsing.

Search results can be filtered by rating and genre

Seat map will display when selecting seats and purchasing tickets.

3.1.2 Hardware Interfaces

Mobile version of the homepage can be interacted with touch. Scrolling feature is interactive with physical scrolling.

Movie times are accurate to the location and time of the Customer when accessing from Phone or desktop.

The website will be simple and require low data transfer from devices connected to the internet for fast, effortless, and easy use.

QR codes will be generated and can be physically scanned through phone screens at the appropriate cinema.

3.1.3 Software Interfaces

Connectivity to send SMS or Emails to customers for Receipts, tickets, or promotions.

Live update of tickets and seats available.

Up to date showtimes of movies of up to two weeks, and refreshes every week. Past times are removed from availability.

Third party payment integration with Paypal, Venmo, affirm, and Shoppe.

3.1.4 Communications Interfaces

Communication to API for showtimes.

Having stable security to handle private information between server and user.

Integration with the server to send SMS and emails immediately.

3.2 Functional Requirements

This section will outline the key features for

3.2.1 Ticket Purchasing

3.2.1.1 Introduction

This feature allows Customers to purchase tickets and seating from a selected showtime.

3.2.1.2 Inputs

- Select Movie
- Select Time
- Select Seating
- Choose payment option

3.2.1.3 Processing

1. The system retrieves available tickets and seating information
2. Customer selects time and seat
3. The system will retain this information temporarily
4. Customer completes purchasing prompts
5. Ticket is reserved with their information, and is removed from list of availability

3.2.1.4 Outputs

- Movie information
    - Showtimes
    - Location
    - Price
    - Summary
    - Available seats
- Ticket with Barcode/QR code
    - Delivered through Email and SMS
- Receipt for proof of purchase
    - Delivered through Email and SMS, and temporarily stored on server

3.2.1.5 Error Handling

1. If the Customer is inactive for 10 minutes, the temporary seat data is removed and the Customer must start again.

2. If the chosen seat is unavailable after purchase, the Customer is redirected to choose new seats.
3. If payment is unsuccessful, the Customer is redirected the cart to retry with another payment method

3.3 Use Cases

3.3.1 Use Case #1: Customer Movie Search

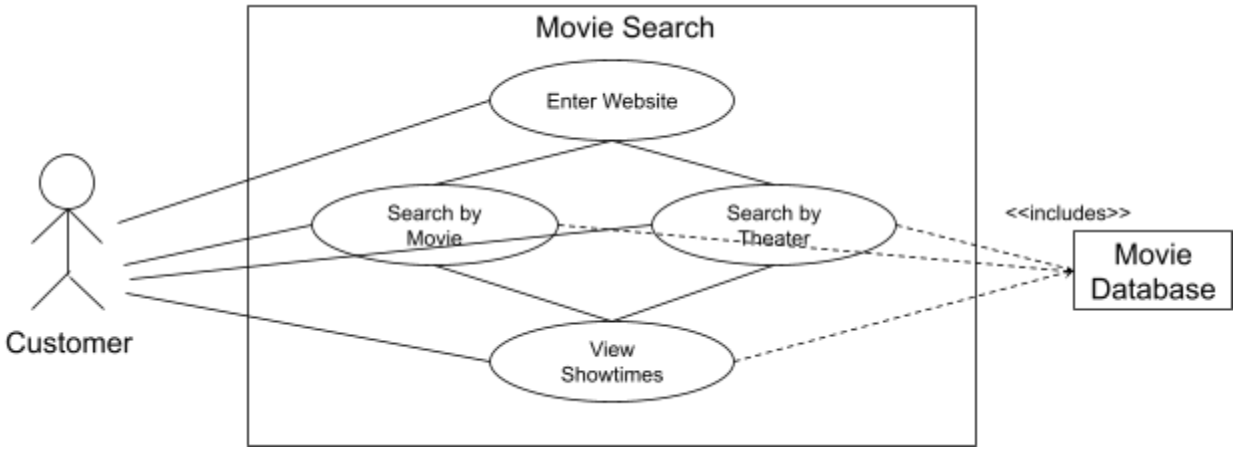
Actors: Customer

Flow of Events:

1. Customer accesses movie theater website from a browser.
2. Customer is greeted with a search bar along with current newly released movies.
3. If Customer searches by movie, the website will display a list of ABC Theater locations and their subsequent showtimes for the specified movie. Showtimes are categorized by Standard, Premium, and 3D theaters.
4. If Customer searches by theater then they will see a similar interface but with showtimes of all movies available at the selected location.
5. If no showtimes are available for the selected movie or theater the customer will be prompted with a message detailing so.
6. Clicking on a showtime will redirect the customer to the ticket purchasing case.

Entry Conditions:

1. Computing requirements: supported browser.



3.3.2 Use Case #2: Customer Login

Actors: Customer

Flow of Events:

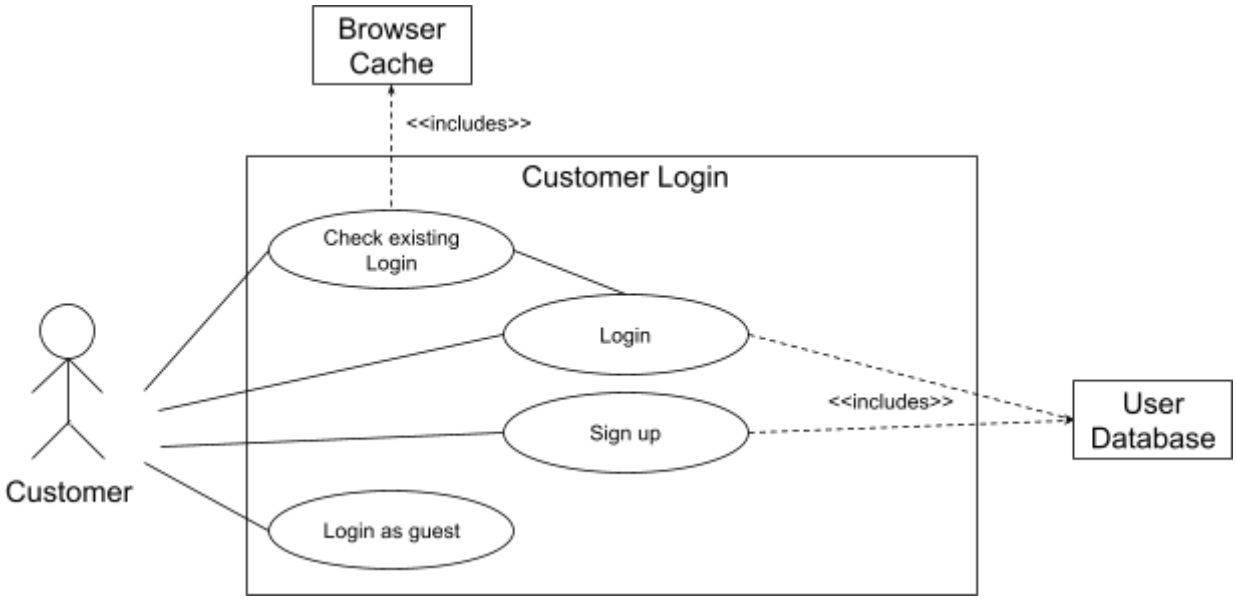
1. Customer attempts to purchase a ticket.
2. The website checks the browser cache for a stored login. If a login is found, the next events are skipped and the customer proceeds to ticket purchasing.
3. If no existing login is found, Customer is prompted with 3 options,
4. Customer selects either to login, create a new account, or continue as a guest.
   4.1. If continue as guest is selected, proceed with ticket purchasing.

4.2. If login is selected, Customer is prompted to enter a username and password.
4.3. If “create new" account is selected, Customer is prompted to enter a username, password, valid email, and valid phone number.

5. Website server queries database to validate the login or to check if an account already exists using the same username or email.
6. If login credentials are invalid, Customer is prompted to try again or reset their password.
7. Customer continues to purchase tickets.

Entry Conditions:

1. Customer attempts to purchase ticket.



3.3.3 Use Case #3: Customer Ticket Purchasing

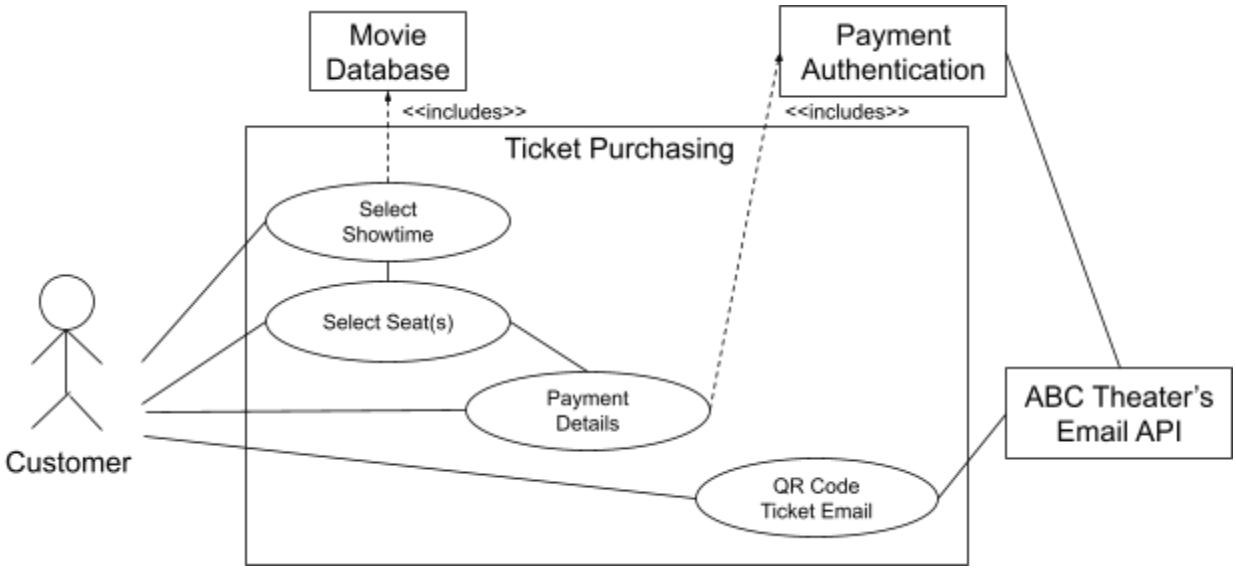
Actors: Customer, Third-Party Payment Processor (Apple, Google, Banks, etc)

Flow of Events:

1. After selecting showtime and logging in, the website accesses the database to see what seats are available for the selected showtime.
2. The customer can then select what seats they would like to purchase from a visual interface showing a top down view of the theater.
3. As the seats are selected, the tickets are added to the customer’s cart. There is no limit to how many tickets from the same theater a customer can select.
4. Once in the cart, the website tells the database to hold these tickets. A 10 minute countdown begins on the webpage. If the user does not checkout before the 10 minute timer ends, these tickets are made available to other customers, and the Customer is notified that their seat reservations have expired.
5. Selecting checkout redirects customers to a payment screen. Forms of payment accepted are Apple pay, Google pay, and all major credit cards.
6. Payment information is verified with a third-party payment processor. The payment processor sends a confirmation or denial of payment based on the card’s status.
7. Once verified, the customer will receive an email from ABC Theater’s with their tickets and QR codes for entry.

Entry Conditions:

1. Customer has selected a theater, movie, and showtime.
2. Customer has valid payment method to purchase tickets.



3.3.5 Use Case #4: Customer Support Contact

Actors: Customer, Customer Support Representative

Flow of Events:

1. A Customer may send a message to the ABC Theater's customer support team if they have any issues regarding the use of the website, a previous purchase, or any other technical problem.
2. From the home page of the website, Customer can scroll down to the bottom and select help
3. Customer will be shown a series of frequently asked questions and their answers.
4. If these don't help, a button for "Submit a Ticket" is available.
5. Once selected, Customer will be prompted to submit their name, phone number, email address, and write a message detailing their issue. An option to attach images is also available, with a file size limit of 350mb.
6. A Customer Support Representative will review the submission, and respond within 24-48 hours via phone call or email.
7. The Customer Support Representative is able to access customer account information as well as manage payment processes in the case that refunds or adjustments need to be made.

Entry Conditions:

1. Customer needs help with services provided by ABC Theaters.
2. Customer Support Representative is available to respond to tickets.

3.4 Classes / Objects

3.4.1 Class / Object #1: Customer

This class represents a customer interacting with the website

3.4.1.1 Attributes:

- **customerID**: An identification number used in the database for each customer.
- **username**: The customer's chosen username.
- **password**: Encrypted password for the user login.
- **email**: The customer's email address for contact and promotions.
- **phoneNumber**: The customer's phone number for notifications and promotions.

3.4.1.2 Functions:

- **searchMovie(movieTitle)**: Allows the customer to search for movies by title.
- **searchTheater(zipCode)**: Allows the customer to find theaters near them based on zip code.
- **login(username, password)**: Authenticates the customer based on credentials stored in the ABC Theater's user database.
- **holdSeat(showtimeID, seatNum)**: Reserve's seat for the user and begins a 10 minute countdown.
- **purchaseTicket(showtimeID, seatNum)**: Initiates the ticket purchasing portion.

References to functional requirements:

- Use Case 3.3.1: Customer Movie Search - utilizes searchMovie() and searchTheater().
- Use Case 3.3.2: Customer Login - needs all variables and login() function.
- Use Case 3.3.3: Customer Ticket Purchasing - uses holdSeat() and purchaseTicket().

3.4.2 Class / Object #2: Theater

3.4.2.1 Attributes:

- **theaterID**: Identification number used in database for each ABC Theater location
- **theaterName**: Name of the location used for searches.
- **address**: Street address of the theater.
- **showtimes**: A list of movies and showtimes playing at this theater.

3.4.2.2 Functions:

- **getShowtimes()**: Returns the available showtimes for this theater.
- **getMovies()**: Returns the list of movies playing at the theater.

References to functional requirements:

- Use Case 3.3.1: Customer Movie Search

3.4.3 Class / Object #3: Ticket

3.4.3.1 Attributes:

- **ticketID**: A unique code for each ticket.
- **movieTitle**: The title of the movie the ticket is for.
- **showtimeID**: The specific showtime for the movie.
- **seatNum**: The seat assigned to the ticket.
- **price**: Price of the ticket.
- **isPurchased**: A boolean for tracking if the ticket has been purchased or not.

3.4.3.2 Functions:

- **reserveTicket(showtimeID, seatNumber)**: Reserves the ticket for purchase.
- **purchaseTicket(paymentInfo)**: Processes the ticket purchase.
- **sendTicket(email)**: Sends ticket QR code to the ticket recipient.

References to functional requirements:

- Use Case 3.3.3: Customer Ticket Purchasing

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 transactions 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

- Response Time: Search results for movies and theaters should load within 3 seconds.
- Ticket Purchase Time: Payment confirmation and receiving the ticket via email should take no longer than 20 seconds after the customer submits their payment.
- Concurrent Users: The website should be able to handle up to 2,000 concurrent users browsing and searching for movies at the same time without experiencing performance impacts.
- Seat Availability Refreshes: Seat Availability should refresh constantly and be updated in real time to ensure that the same seat is not purchased by two different people.

3.5.2 Reliability

- Uptime: The system should guarantee a 99% uptime from 6am to midnight.

3.5.3 Availability

- 24/7 Access: The system should be available 24 hours 7 days a week, including holidays.
- Peak Traffic: The system should remain fully functional when experiencing high traffic, such as opening nights of popular movies.

3.5.4 Security

- Data Encryption: All sensitive customer data (login info, credit card details, and personal information) should be encrypted using industry-standard protocols (i.e. AES-256).
- Payment Security: Payment processing should comply with Payment Card Industry Data Security Standard to protect customers' financial data.

3.5.5 Maintainability

- Modular Code: The system should be designed with a modular codebase to ensure the individual components can be updated without impacting the rest of the system.
- Version Control: Use of version control during development should be utilized to ensure backups and collaboration in case of unexpected errors.
- Documentation: Documentation for all system components should be maintained and updated regularly by the developers.

3.5.6 Portability

- Cross-platform: The system should be accessible from various different desktop browsers (Chrome, Firefox, Safari, Edge) as well as mobile browsers on Android and iOS.

3.6 Inverse Requirements

- No Unauthorized Access: The system must not allow any customer or third party to access databases without proper authorization.
- No Slowdowns During High Traffic: The system must not experience impacted performance during peak usage.
- No Double Ticket Bookings: The system should not allow for two customers to book the same seat for the same showtime.
- No Unencrypted Data Storage: The system must not store sensitive customer info in any unencrypted formats.
- No Excessive Downtime: The system must not be unavailable for more than 1% of the time.
- No Exposure of Sensitive Information: The system must not display any sensitive customer information in plain view.

3.7 Design Constraints

Specify design constraints 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.

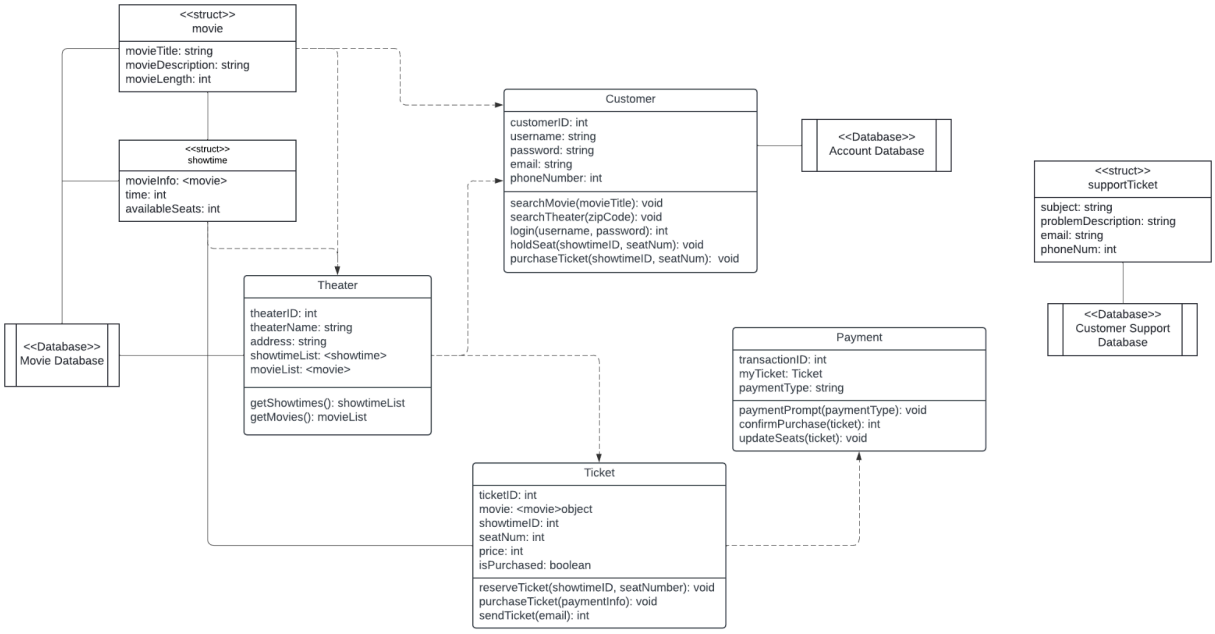
3.9 Other Requirements

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 UML Diagram (Updated from Version 2)



This is a UML Diagram for the ticketing system, visually representing the organization and interactions within the system. We can see how the movie, its show time, and theater create a purchasable ticket, as well as the relation to the customer. The process and flow of data from Customer to the system is highlighted, and will serve as a roadmap on how to develop the website.

Class Name: Theater

Purpose: This class represents a physical movie theater location.

Attributes:
theaterID: int - A unique identifier number used for every theater.
theaterName: string - The name of the specific theater.
address: string - Contains the address of the physical location
showtimeList: <showtime> - Stores a list of the available showtimes.
movieList: <movie> - Stores a list of movies showing at this theater

Methods:
getShowtimes(): showtimeList - Returns a list of the showtime structure.
getMovies(): movieList - Returns a list of the movie structure.

Description:
The Theater class manages the information about the physical locations as well as tracking the showtimes and available movies.

Class Name: movie

Purpose: Structure to hold the information about a particular movie

Attributes:
movieTitle: string - The title of the movie
movieDescription: string - The description of the movie
movieLength: int - The runtime of the movie

Description:
The movie structure stores the information of the movie to be used by other classes.

Class Name: showtime

Purpose: Structure to hold the information about a particular showtime

Attributes:
movieInfo: <movie> - A movie type structure
time: int - Time of the showing
availableSeats: Number of available seats

Description:
The movie structure stores the information of a specific showtime to be used by other classes.

Class Name: Customer

Purpose: This class represents a potential customer of ABC Theater's and has the functions available to the customer.

Attributes:
customerID: int - Unique identifier assigned to each user.
username: string - Username used for signing into the user account.
password: string - Password used for signing into the user account.
email: string - Email of customer to send receipts, tickets, and to store in account database.
phoneNumber: int - Phone number of customer to send SMS receipt, ticket QR code, and to store in account database.

Methods:
searchMovie(movieTitle): void - Takes in movie title to search and return Movie
searchTheater(zipCoder): void - Returns theater in local area after taking in zipCoder
login(username, password): void - takes Username and Password to access already saved user account data
purchase Ticket(showtimeID, seatNum): void: - Allows to purchase the correct ticket

Description:
This class demonstrates the basic interactions the customer can use while navigating the site.

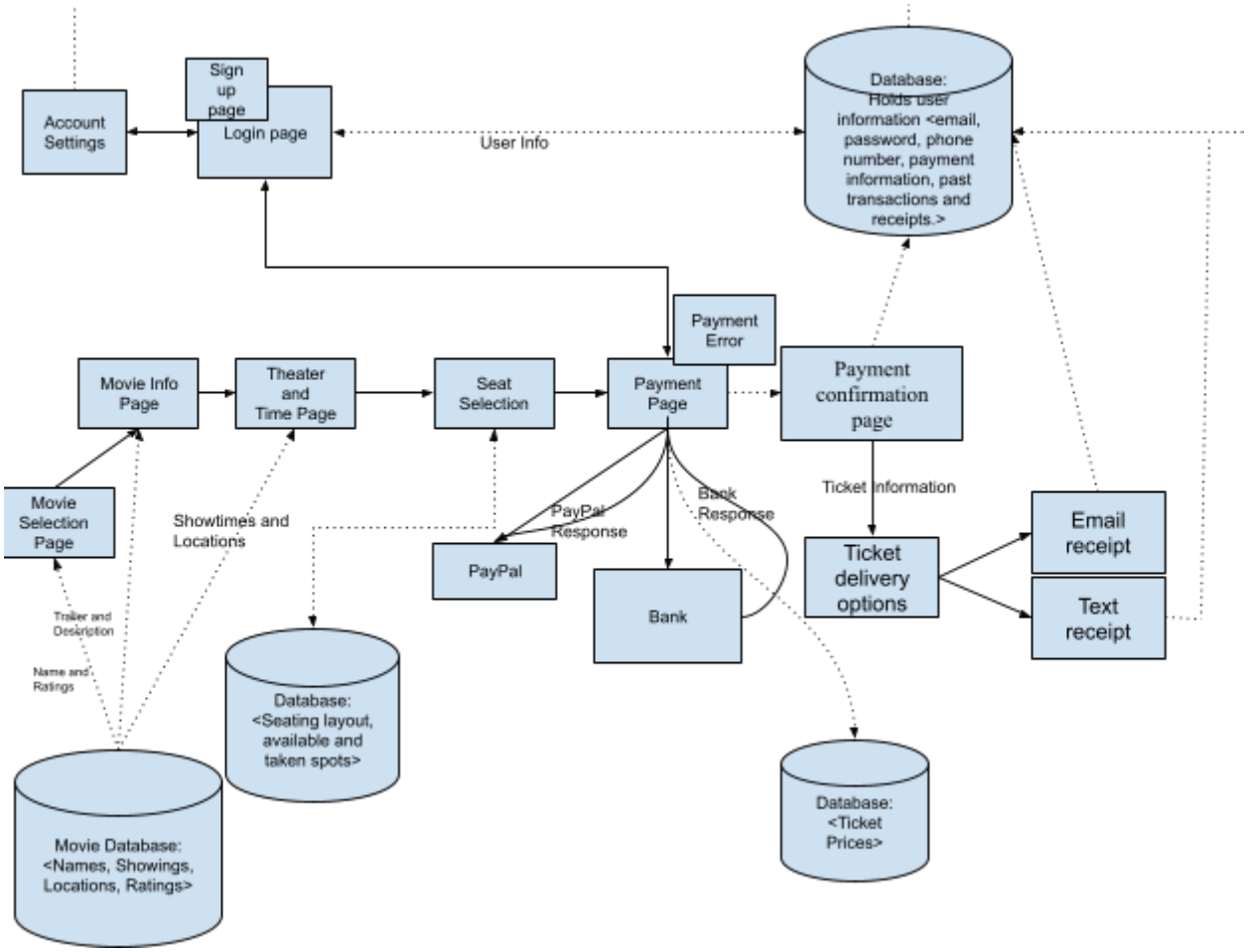
Class Name: Payment

Purpose: This class is used to allow the customer to make a payment
Attributes:
transactionID: int - Identifies the payment and transaction
myTicket: Ticket - Object that holds movie, showtime, and location data
paymentType: string - Method of payment

Methods:
paymentPrompt(paymentType): void - Takes paymentType object to determine appropriate prompts for payment.
confirmPurchase(ticket): int - Accepts ticket object to confirm the purchase with correct amount displayed. Will update the ticket with purchase if successful.
updateSeats(ticket): void - Updates the seating map with the dated from a confirmed ticket purchase

Description:
The class will represent a ticket being purchased. It will have information about payment method, ID, and amount.

4.2 Software Architecture Diagram (SWA) (updated from Version 2)



Description: SWA diagram for ABC Theater's movie ticketing system.
This diagram will show the system's key components and their interactions.

- Interaction between the user interface, databases, payment processing and external APIs involved in the ticket purchasing system.
- Inclusion of interactions between the customer side of the website and the system, such as accessing databases for various data.

- Login/Signup Page:
  - Connects to a User Database that stores data including usernames, passwords, and purchase history. includes integration with account settings, and payment functions
- Account Settings/Profile Page:

        - Users will be given options to alter usernames and login info that will be updated in the User Database
- Browsing and Selection:
    - Movie Selection Page: The user selects a movie from a Movie Database which stores available movies, times, and theater locations.
    - Movie Info Page: After selecting a movie from the list displayed on the selection page, relevant information such as reviews, cast, and description of the film from the Database.
    - Time and Theater Selection Page: After selecting the option to view screening options, the user chooses a specific time and theater. Accesses data from Movie Database holding movie names, showings, locations, and ratings.
    - Seat Selection Page: access data from database, which includes seat layout at the location, and availability of those seats.
- Payment Process:
    - Payment Page: This page collects payment details, and updates the associated database.
    - Confirmation Page: Once the payment is processed, the confirmation page displays the purchase details.
    - Payment Error Page: If an error with payment is encountered, the user will be alerted with a popup displaying whether or not the error encountered is on the user side.
- Ticket Delivery:
    - Ticket Delivery System: the system sends the ticket to the user's email or phone number via text based on user's selection, and updates the associated database.

4.3 Development Plan and Timeline

Cameron Lee (Software Developer) https://github.com/papabeanstalk

- Cameron will lead development of system software, using the most appropriate data structures for each feature. He will ensure that the site will run efficiently for the integrity and longevity of the product. Development will take 1 month per feature, following Roger's guidelines for architecture.

Roger Dao (Outline creation) https://github.com/rogerdao

- Roger will work on the structural organization of every software component. He will ensure that the outline is defined, and will serve as a guide for leading the project efficiently.

Ross Hackett (Tester) https://github.com/Bumple

- Ross will be leading and developing tests to ensure the system's integrity. His goal is to find possible solutions within the software and architecture using worst case scenarios. He will document and report issues found, as well as proof that the issue exists.

Matthew Dominguez (Web Developer) https://github.com/MatthewJD5

- Matthew will be creating and developing the Website to make sure it works with the client's needs. User Interface will have features requested and specified within this document adhering to the requirements of the client. Production will take 1 month for draft, and ongoing for any adjustments.

4.4 Test Plan

GitHub: https://github.com/papabeanstalk/G6-SRS

4.4.1 Introduction

The test plan verifies and validates the components of the ticketing system software and ensures that they are working as expected. The plan incorporates a combination of tests at the unit, functional, and system levels to thoroughly verify features of both core system functions and the overall functionality of the website for a customer. This section describes the design of the specific test cases, and expected results. The scope of our tests applies to all ABC Movie theater locations in the San Diego area.

Unit Testing - Includes testing of class methods and functions.
Functional Testing - Includes testing of interaction between separate classes and functions.
System Testing - Includes testing of entire systems to ensure end-to-end behavior is correct and the software system is functional.

4.4.2 Test Scope

The test plan will focus on the following core functions:

- Movie Search System (Back-end)
- Payment Processing (Back-End)
- Ticket Reservation (Front-End)
- Customer Registration and Login (Front-End)
- Customer Support Ticket Submission (Front-End)
- Ticket Purchasing Process (System Test)

4.4.3 Test Details

1. Movie Search System - Unit Test
    - Feature Tested: Functionality of movie search by title.
    - Priority: 1
    - Test Description: Back end test that ensures that users are able to search for movies based on their title by testing the searchMovie() module in the Customer Class. Prerequisites are an existing testing environment being set up that is connected to the movie database.
    - Test Steps:
        i. Enter the full name of a movie into the searchMovie() function
        ii. Enter a word that multiple movies start with into the searchMovie() function
        iii. Enter the name of a movie that does not exist into searchMovie()

- Expected Result: The movie being searched for in the first step should be returned. All the movies matching the search term in the second test should be returned as a list. No movies should be returned in the third search for a movie that does not exist.

2. Payment Processing - Unit Test
    - Feature Tested: Functionality of the payment processing
    - Priority: 0
    - Test Description: Back end test that ensures that the payment information is properly sent to the third party payment verification and that approval is received properly. Prerequisites are an existing testing environment with the functions for requested payment verification implemented.
    - Test Steps:
        i. Enter desired payment type into the paymentType string by calling paymentPrompt() in the Payment class.
        ii. Enter and submit the payment information.
        iii. Wait for the response from the entered payment verification.
    - Expected Result: If the entered payment information is deemed valid from the payment verification, a value of true should be returned. Otherwise, if the payment is invalid for whatever reason false will be returned.
3. Ticket Reservation - Unit Test
    - Feature Tested: Functionality of the system that stops users from selecting reserved tickets.
    - Priority: 0
    - Test Description: Front end test that makes sure that the temporary hold of tickets during a customer's purchasing process remains up to date and successfully prevents selling the same seat to multiple customers. Prerequisites are to have two devices on the movie ticket website with the same showtime selected.
    - Test Steps:
        i. On device A, select a group of seats to be purchased.
        ii. On device B, check the map of available seats, if able, try to select a seat that has already been selected on device A.
    - Expected Result: Device B should either have the seats selected by device A unselectable, or receive a notification saying the seat is not available when selected.
4. Customer Registration and Login - Functional Test
    - Feature Tested: Functionality of the customer account creation and login
    - Priority: 2
    - Test Description: Front end test that ensures that a customer is able to create a new account and checks whether that account is properly saved in the user database. Prerequisites are to have a browser open to the ticket website and to navigate to the account login.
    - Test Steps:
        i. Select "Create new account"

        ii. Enter Username, password, email, and phone number.
        iii. Verify the account from the email sent to the new account.
        iv. Select “Login”
        v. Enter the newly created username and password.
    - Expected Result: If the email entered is valid and the account is verified, the login should be accepted and the user’s information should be able to be accessed and stored. If the email is invalid, the account is not verified, or the information entered does not match a user in the database, an error should be returned when attempting to login.
5. Customer Support Ticket Submission - Functional Test
    - Feature Tested: Functionality of the customer support ticket submission system.
    - Priority: 3
    - Test Description: Front end test that ensures a user can navigate to the customer support portion of the website and submit a ticket that will be received by the customer support team. Prerequisites are a browser that has the website open as well as the backend being connected to a database used to store customer support tickets.
    - Test Steps:
        i. Scroll to bottom of the screen and select “help”
        ii. Scroll past the frequently asked questions and select “Submit Ticket”
        iii. Enter the subject line, the description of the problem, an email, and phone number.
        iv. Submit the newly entered ticket.
    - Expected Result: From the customer service’s database, a new ticket should appear accurately showing the information submitted by the user on the website.
6. Ticket Purchasing Process - System Test
    - Feature Tested: Functionality of the overall website and the process of purchasing a ticket from start to finish.
    - Priority: 0
    - Test Description: Front end test that verifies the entire system works as intended. Verifies if the process of purchasing a ticket is simple and intuitive, as well as whether or not a user can actually purchase and receive a ticket. Prerequisites are a browser opened to the website and a valid form of payment.
    - Test Steps:
        i. Navigate Website to the search bar
        ii. Search for a movie
        iii. Select a showtime
        iv. Select available seats
        v. Checkout tickets, enter valid payment information
    - Expected Result: Process should be intuitive and easy, seat selection should be accurate and up to date. The tickets purchased should be received in the entered email once the payment information has been verified. The process should take no

longer than five minutes if the desired movie and time is already known to the user.

4.5 Data Management Strategy

Our software system requires handling persistent and potentially sensitive data, requiring a safe and efficient strategy for data management. To meet these needs, we have chosen an SQL-based approach with a single database architecture. This choice prioritizes maintenance, reliability and data consistency. These aspects are all essential for managing customer information, movie data, and transaction records.

4.5.1 Rationale for Design Decisions:

- Single Database Approach
    - We opted for a single centralized database to simplify maintenance and provide a single point of management for data across all theater locations. This design minimizes redundancy and overhead and reduces the need for complex data synchronization across several separate databases.
    - A single database structure allows for a straightforward recovery process in the event of a system failure. Complex re-synchronization processes aren't necessary for a single database approach, thereby minimizing potential downtime.
- SQL Database:
    - We Chose SQL for its ACID (Atomicity, Consistency, Isolation, Durability) properties, that ensure that critical transactions, such as ticket purchases and customer account updates, are handled reliably and consistently.
    - SQL's well defined relationships between entities aligns well with our data requirements. SQL also supports complex queries, enabling efficient data retrieval and ensuring data integrity.
    - SQL databases are ideal for structured data with relational dependencies, which fits our application's data interactions since they are mostly predictable and follow established relationships.

4.5.3 Data Organization

- Logical Table Structure:
    - The database is organized into tables that include *Customer, Theater, Showtime, Movie, Ticket, Transaction, and CustomerSupport.* This structure keeps related data logically separated while minimizing redundancy.
        - Theater Table: Contains location-specific data about the separate theater locations.
        - Showtime and Movie tables: Manage details about available showtimes and the movies being shown.
        - Transaction table: Records each ticket sale with unique transaction IDs. Linking each sale to the customer and their selected seats.

        - CustomerSupport table: Manages details about tickets submitted by users to be reviewed by the customer support team.
- Data Segmentation:
    - We organize movie data separately from location-specific information to avoid redundancy and keep data logically separated.

4.5.4 Possible Alternatives and Tradeoffs

- Single vs Multiple Databases:
    - A single database is easy to maintain and sufficient for the scale of our theaters. However, as we expand, scaling a single database could lead to a higher load.
    - A multiple database architecture could distribute load across regional databases or location specific databases, improving performance and reducing latency.
    - Tradeoff: Initially, we are implementing a single database for simplicity and cost-effectiveness. Multiple databases introduce additional complexity and increase costs/maintenance demands.
- SQL:
    - SQL provides a reliable, structured solution that suits our current needs, avoiding the overhead of migrating to other systems and maintaining simplicity.

4.5.5. Future Considerations

- Our SQL-based, single database system meets our current operating requirements. ensuring a balance of cost-effectiveness, and straightforward management. We have also anticipated possible future scenarios
    - Regional Expansion: For future regional expansion, we may consider a distributed database model or regional database instances to manage location-specific data. We would explore options like partitioning by region or using database clusters to ensure consistent performance and scalability.
    - Enhance Scalability: To accommodate high transaction volumes or data growth, we could implement database partitioning or caching. This would allow us to continue benefiting from SQL's advantages while scaling to meet increased data demands.

In conclusion, we chose a single, SQL-based database for simplicity, ease of maintenance, and reliability. This approach minimizes operational complexity and is a cost-effective solution that meets our current needs. We have also identified potential enhancements for scalability and adaptability, ensuring our system remains prepared for future growth.

5. Change Management Process

We have chosen a single database design for ease of maintenance. In making this decision, we considered several alternatives and their tradeoffs:

1. Single Database vs. Multiple Databases

We opted for a single database because it minimizes overhead and simplifies maintenance. This choice also helps ensure that in case of a failure, normal operations are less likely to be disrupted.

2. SQL vs. NoSQL

   Our current SQL database meets all our operational needs, and we do not anticipate requiring NoSQL's expanded functionality. SQL also aligns well with our goal to keep the system easy to set up and maintain.

If we expand to new locations or regions, we may revisit this decision and consider options such as:

Adding a new database for those regions.

Building a new system capable of handling multi-regional data with additional personnel to manage the database.

Design Choices and Data Organization

Our data is organized logically to ensure that only necessary information is stored together. Key table contents include:

- Location-Specific Information
  - A movie database to store content unique to each location without redundancy.
- Sales Information by Location
  - Data on each location’s sales, organized by Product ID (PID).

Our design rationale is to keep complexity low by maintaining a single point of failure as much as possible, as increased complexity can quickly become unmanageable. We considered alternative designs, such as:

- Multiple databases.
- NoSQL databases.

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