Moviicket

Software Requirements Specification

Version: Software Design

2/29/2024

Group 4
Shelley Lam
Adelina Martinez
Amilcar Galvez

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

Revision History

Date	Description	Author	Comments
2/1/2024	Gathering Requirements	Group 4	Complete sections 1 - 3.2.2
2/8/2024	Use Cases and Completion	Group 4	Complete sections 3.3 - 3.6
2/15/2024	Final Revision 1	Group 4	Revise all sections
2/22/2024	Software Design	Group 4	Complete section 3.7
2/29/2024	Software Design	Group 4	Design Specifications

Document Approval

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

Signature	Printed Name	Title	Date
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

This document provides and contains the list of information needed to write software requirements for Moviicket as specified by the client. It describes the necessary contents and qualities of a good Software Requirement Specification (SRS) and provides an overview of the complete SRS outline.

1.1 Purpose

The purpose of this document is for the developers to have an idea of what the client wants when programming and building the Moviicket website. This guide is designed to specify all elements and requirements of the software to be developed.

1.2 Scope

The software products of this document should be produced as follows:

(1) Website Host to have the website up and running for the clients
(2) User Login system with option for signing up and creating an account or proceeding as a guest
(3) Payment options during the purchasing stage of a ticket
(4) Search bar functionalities by movie name
(5) Movie page overview containing a small summary along with ratings and reviews

In this document, the finished product will result in a software system that supports and builds the movie ticketing website called Moviicket. The site will allow users to purchase tickets of their choice with limitations and other policies when it comes to purchasing.

1.3 Definitions, Acronyms, and Abbreviations

MID- Movie Information Database
CID- Customer Information Database
GU- General User
CU- Customer User
AU- Administrative User

1.4 References

[1] AMC Entertainment Holdings, Inc., *AMC Theatres,* Kansas City, Missouri, 1920.
[2] CINEMARK, *Cinemark Theatres,* Plano, Texas, February 6, 1984.
[3] Fandango Media, LLC, *FANDANGO,* Beverly Hills, California, April 27, 2000.
[4] Regal Entertainment Group, *REGAL,* Knoxville, Tennessee, July 30, 1924.

1.5 Overview

The remainder of this document is organized as follows:

(1) Section 2 describes the general description and perspective of the products to be developed.
(2) Section 3 provides the external interface requirements used to make the interface.
(3) Section 4 contains the analysis models during the testing phase before release.
(4) Section 5 provides the list of changes and adjustments made over the course of the website being developed.
(5) Section Appendix provides additional information containing conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

2. General Description

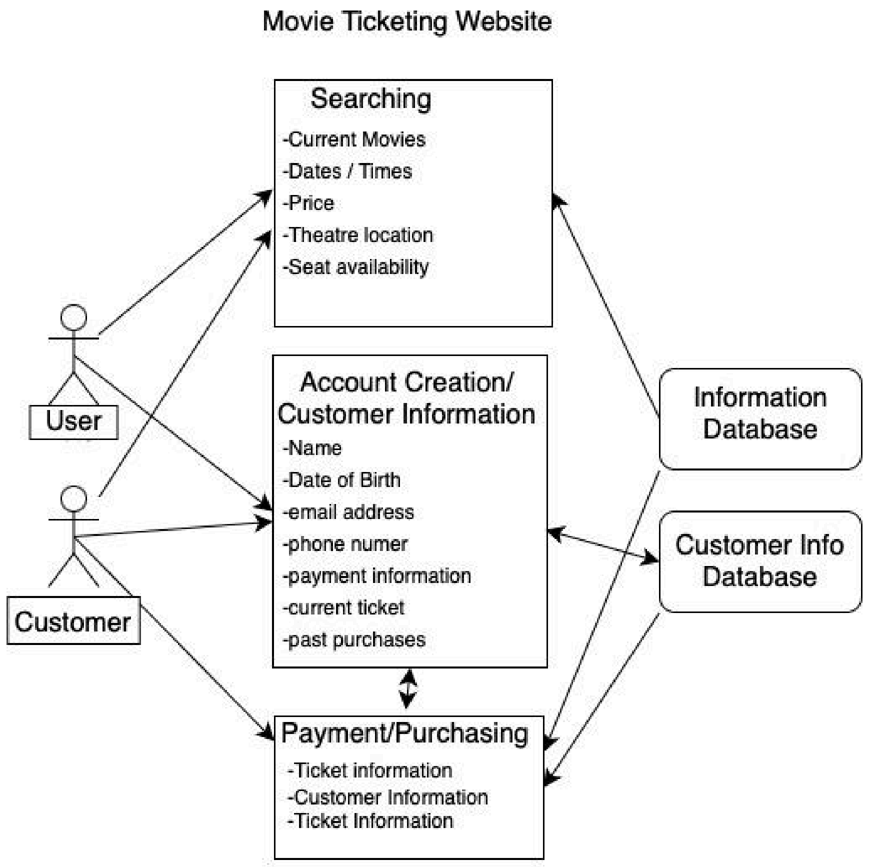
2.1 Product Perspective

Moviicket is a system that includes the following functions and features.

- **Movie Information/ Searching**
  The system will use a database of current movie information including movies playing in theaters, relevant dates and times, ticket prices, seat availability, and theater location. Searching and viewing will be allowed for all users of the website.
- **Customer Information/ Account Creation**
  To make a purchase, users will create an account. Customer information will include a first and last name, date of birth, email address, password, phone number, payment information, current tickets, and past ticket purchases. This information will be saved for later purchasing use and communication with customers about ticket information such as confirmation via text or email.
- **Payment/ Purchasing**
  Before creating an account users will be allowed to select a movie ticket with the desired movie, seat number, theater, showing date, and time. Payment will be by credit/debit card. Confirmation of purchase will be through email.

2.2 Product Functions

The major features and functions of Moviicket are shown below in a general environment diagram example. Not shown is the Administrator user who has access to all functions including the Movie Information Database, and the Customer Information Database.



2.3 User Characteristics

Users will be anyone who is viewing the website, of type; general user, customer who has created an account, and administrative user. The user should be familiar with how to use a search engine and purchase items online.

GU (GENERAL USER) Abilities:
- Search for available movies
- View ticket information
- Select tickets
- Create account

CU (CUSTOMER USER) Abilities:
- Search for available movies
- View ticket information
- Select tickets
- Delete account
- Purchase tickets
- View current/ past purchased tickets
- Change, edit, and delete account and account information

AU (ADMINISTRATIVE USER) Abilities

- Search for available movies
- View ticket information
- Select tickets
- Delete account
- Purchase tickets
- View current/ past purchased tickets
- Change, edit, and delete account and account information
- Search for and view all or any accounts using the Customer Information Database
- Delete accounts from the Customer Information Database
- Edit and update information in the Movie Information Database

2.4 General Constraints

Constraints include access to an easy-to-use functional interface via a Mac or Windows machine with access to general search engine browsers such as Chrome, and Safari, to view websites. The website requires at a bare minimum adequate hardware to access web browsers and a base requirement of wifi to be able to run and load the website.

2.5 Assumptions and Dependencies

Assume the URL and website domain are already purchased and created. Assume the user has access to the system on any general desktop browser and hardware system with adequate wifi connection and memory. Assume the Movie Information Database is current and regularly updated by Administrative Users to show the most relevant data with only valid dates and times, updated availability, and accurate prices will be shown. Assume Customer Information Database and customer accounts are safe and accessible. Assume the user will not request refunds, tickets are one-time purchases only. Assume only valid payment methods and card numbers are accepted and processed.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

An interface that appeals to the youth. It should show pictures of the movies as a way to appeal to the customers and also allow the user to search for a movie by their name. The catalog of movies also are listed based on movies available in the present day in which they were searched. If a movie was available earlier in the day but isn't anymore it's taken out of the catalog. The UI also filters the movies based on days, rotten tomato scores, and movie ratings. Once a customer clicks on the movies they can select an available time of the movie and it shows the seats available in a blueprint-type style. In this screen, you can see the distinct types of seats such as disabled support seatings, where they are proportional to the screen, whether the seats are available or not, and exclusive seatings. Users will also be able to create accounts. When purchasing tickets users will be taken to another webpage, where they will confirm they are human and pay for the tickets. Once the purchase goes through users will be given digital copies of their tickets.

3.1.2 Hardware Interfaces

Since we are building a website accessed by the most easily accessible and popular web browsers such as Chrome or Safari, all forms of hardware must have adequate RAM, CPU, and GPU capabilities. The hardware must have access to a stable internet connection preferably through a modem connected to an ethernet cable.

3.1.3 Software Interfaces

Operating System: macOS and Windows
Database: MySQL
Language: Javascript for frontend, Python for backend

3.1.4 Communications Interfaces

The website should be supported by all web browsers. When the website is down for a form of maintenance a separate host will take over on a separate server to reduce downtime.

3.2 Functional Requirements

3.2.0 General Functional Requirements:

Needs to support administrator mode.
Handle at least 1500 users at once.
Needs a customer feedback system.
It needs to be able to be run in a web browser.
Interface with the database of showtimes and tickets available.

3.2.0.b Non-Functional Requirements:

Clean and secure UI.

3.2.1 Seat Selection

Allow a user to pick from available seats; once seats are taken, they no longer appear available in the system. Also, the website should be able to limit the number of seats a single user can purchase up to 10, and prevent bots from purchasing tickets for high in-demand movies.

3.2.1.1 Introduction

Without the ability to purchase a ticket a movie ticketing website would be non-functional. To try and prevent customer dissatisfaction we would limit purchasing tickets by a single user and a CAPTCHA test to combat bots.

3.2.1.2 Inputs

Inputs would be users selecting the seats they want up to a maximum of 10.

3.2.1.3 Processing

For the system to allow a seat to be purchasable it must first check if that seat/seats are available and are less than equal to 10 total seats. After the user will be taken to a different webpage to buy the tickets where they will first confirm they're human.

3.2.1.4 Outputs
The output would be the purchase transaction giving the user a digital ticket for their seat/seats. Then those same seats should be made unavailable in the system so nobody else could have an option to purchase them.

3.2.1.5 Error Handling
In the case of an error in the system in which a user somehow manages to purchase an unavailable seat, we would give the customer an option of either picking new seats or getting a refund. This would then be bug-fixed as quickly as possible. As for users being allowed to purchase more than 10 tickets, this would be found out quickly as there will be constant tests checking for this issue but in case it's not found the website will temporarily be shut down and fix the issues as quickly as possible.

3.2.2 Customer Benefits
For a membership holder, allow priority when trying to purchase seats a set time before a movie release and give member discounts for snacks and drinks at the theater.

3.2.2.1 Introduction
This feature gives users a reason to become membership owners by providing benefits. A membership owner would have the ability to reserve seats from 2 weeks to 1 week before showtime

3.2.2.2 Inputs
Similar to the last functional requirement the user selects the seats they want.

3.2.2.3 Processing
When processing the transaction the system checks if the money was successfully sent, if the seats aren't already reserved, and if the user is a membership holder. If any of these cases fail the process fails.

3.2.2.4 Outputs
The output would be the digital ticket or tickets in case every requirement listed in the processing section is filled. If any of them fail the user will be prompted with the requirement they failed and the solution. Ex: Not a membership holder: wait until a normal user can get seats or become a membership owner. Seats already reserved: Look for different seats. Money wasn't transferred: Retry transaction.

3.2.2.5 Error Handling
In the case a user who isn't a membership holder gets the benefits that come with one, we would disable the website until the bug is fixed. The user will be refunded and those sets will be made available again. To prevent this we will have engineers and hackers test for these cases so hopefully it never happens.

3.3 Use Cases

3.3.1 Browse and Search: Valid Search

Actor: GU/CU

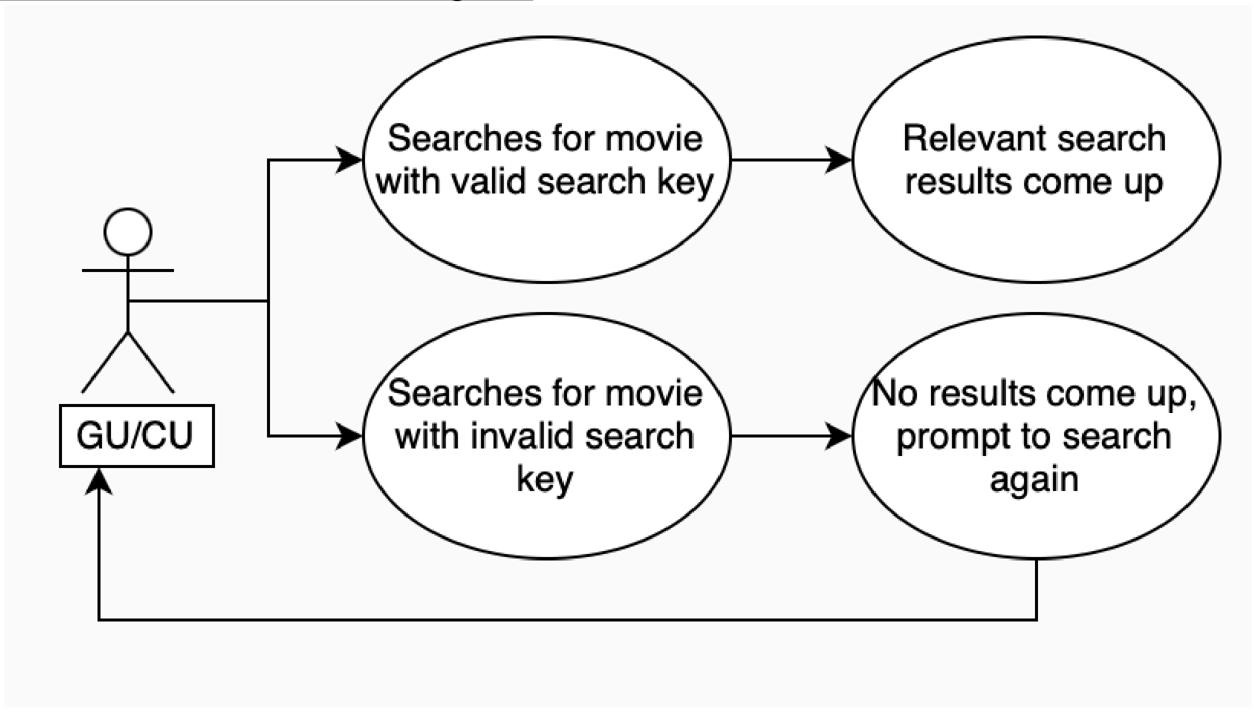
1. The user browses and searches for a movie using a valid search key. Assume that a valid search key consists of the Movie title, director name, or an alphabetically ordered search result.
2. Relevant search results come up and the user can view ticket information for each movie.

3.3.2 Browse and Search: Invalid Search

Actor: GU/CU

1. The user browses and searches for a movie using an invalid search key.
2. No results show up and the user is prompted to try searching again.

Browse and Search Use Case Diagram:


Searches for movie with valid search key
Relevant search results come up
GU/CU
Searches for movie with invalid search key
No results come up, prompt to search again

3.3.3 Account Creation: Valid Info

Actor: GU

1. The user selects the create account option
2. They enter a valid email, password, name, phone number, date of birth, and payment information.

3. The account is successfully created, and the GU becomes a CU and can now purchase tickets, view their current/past tickets, edit their account information, or delete their account.

3.3.4 Account Creation: Invalid Info

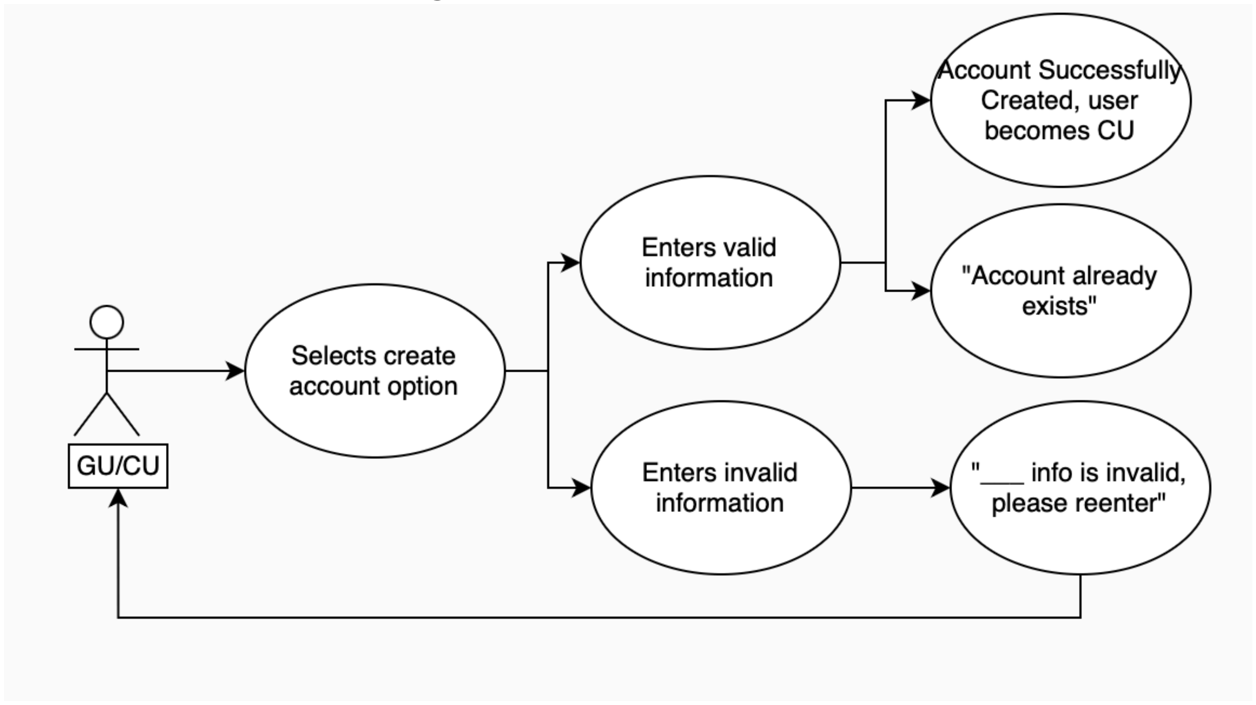
Actor: GU/CU

1. The user selects the create account option
2. They enter an invalid email, (or) password, (or) name, (or) phone number, (or) date of birth, and (or) payment information.
3. The account is not successfully created and the user is prompted to renter the invalid information.
4. If the information is valid, create an account.
5. If information is invalid, back to step 3.

3.3.5 Account Creation: Account Already Exists

A CU selects the create account option, they enter a valid email, password, name, phone number, date of birth, and payment information. The account already exists and the account is not created again.

Account Creation Use Case Diagram:


Account Successfully Created, user becomes CU
Enters valid information
"Account already exists"
Selects create account option
GU/CU
Enters invalid information
"___ info is invalid, please reenter"

3.3.6 Account Login: Valid Info

Actor: CU

1. User will be prompted to log in with their information credentials.

2. If valid, they will proceed where they left off.

3.3.7 Account Login: Invalid Info

Actor: CU

1. User will be prompted to log in with their information credentials.
2. If invalid, they will be told the information is invalid.
3. They will be prompted to re-enter their information until they have logged in.

3.3.8 Account Login: No Existing Account

Actor: GU

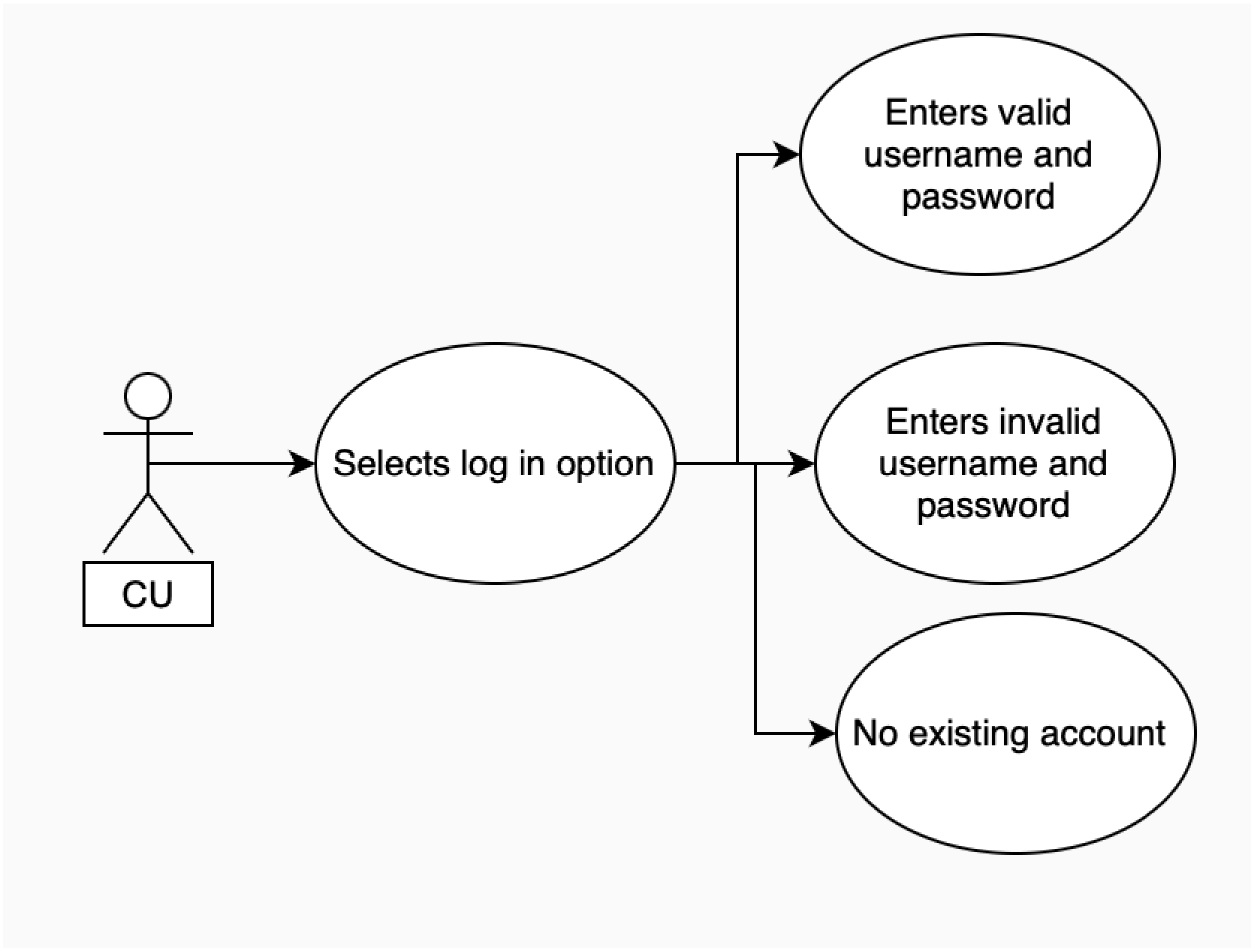
1. User will be prompted to log in with their information credentials.
2. If invalid because the account has not been created or in the system, they will be told that the information doesn't exist and asked if they want to sign up.

3.3.9 Account Editing and Deletion

Actor: CU

1. User deletes account or edits account information

Account Login Use Case Diagram:


CU
Selects log in option
Enters valid username and password
Enters invalid username and password
No existing account

3.3.10 Ticket Selection: Seat Available

Actor: GU/CU

1. The user will be prompted to select seats when a movie has been successfully found and selected.
2. If selected seats are available, the User will be allowed to select the open and available seats with a maximum of five seats.
3. If the user is satisfied with the seats selected, they can proceed to the Check Out to purchase selected seats.

3.3.11 Ticket Selection: Seat Not Available

Actor: GU/CU

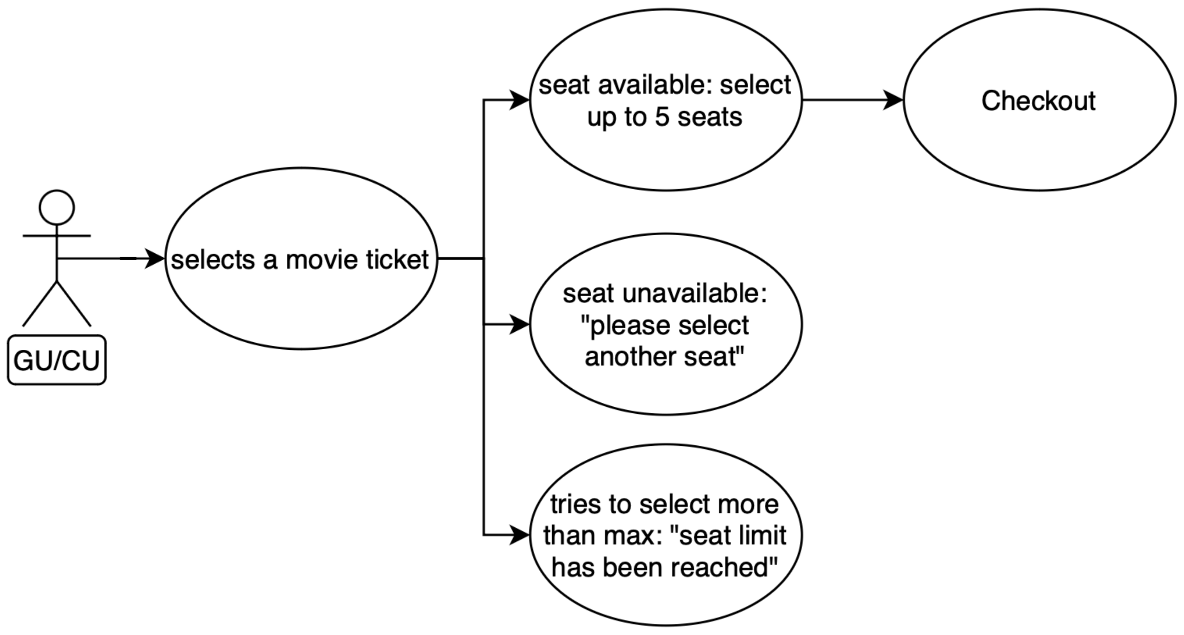
1. The user will be prompted to select seats when a movie has been successfully found and selected.
2. If selected seats are unavailable, the User will be not allowed to select the taken and unavailable seats.
3. The user is instead asked to select another seat until they are satisfied to proceed with Check Out.

3.3.12 Ticket Selection: Too Many Seats Selected

Actor: GU/CU

1. The user will be prompted to select seats when a movie has been successfully found and selected.
2. If selected seats have reached the maximum limit of five selected, they will not be allowed to select any further seats.
3. If they try to add another seat with five already selected, they will be told the limit of seats has been reached already.

<u>Ticket Selection Use Case Diagram:</u>


GU/CU
selects a movie ticket
seat available: select up to 5 seats
Checkout
seat unavailable: "please select another seat"
tries to select more than max: "seat limit has been reached"

3.3.13 Ticket Purchase: Not Logged In Yet

Actor: GU

1. After selecting desired seats, if the user is not logged in yet, they will be prompted to log into their account or to sign up and make an account.
2. Once they have logged in from either choice, they are then able to proceed with ticket purchase.

3.3.14 Ticket Purchase: No Account Yet

Actor: GU

1. After selecting desired seats, if the user is not logged in yet, they will be prompted to log into their account or to sign up and make an account.

2. If they decide to log into a non-existent account, they will be told that account has not been made.
3. If they decide to sign up and make an account, they will then be able to proceed with ticket purchase.

3.3.15 Ticket Purchase: Logged in and Valid Payment

Actor: CU, AU

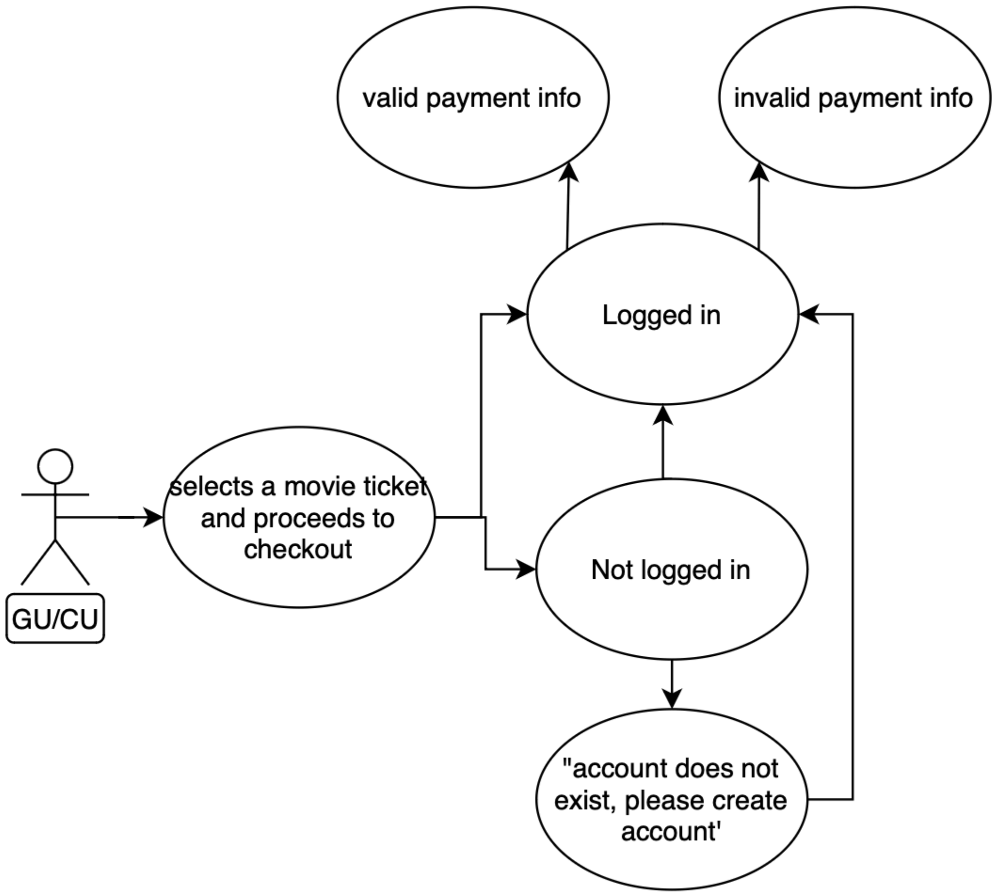
1. CU will be prompted with a page to enter their payment information with choices of payment such as credit card, debit card, or PayPal.
2. If any of the information from one of the choices is valid, the purchase will go through and CU will be told their purchase is successful. A confirmation email will be sent to both the AU with a copy of their purchase and ticket information.

3.3.16 Ticket Purchase: Logged in and Invalid Payment

Actor: CU

1. User will be prompted with a page to enter their payment information with choices of payment such as credit card, debit card, or PayPal.
2. If any of the information from one of the choices is invalid, the purchase will not go through and they will be told their payment is invalid.
3. The user will be prompted to re-enter their information until the purchase has been completed.

<u>Ticket Purchase Use Case Diagram:</u>


valid payment info
invalid payment info
Logged in
selects a movie ticket and proceeds to checkout
Not logged in
GU/CU
"account does not exist, please create account'

3.4 Classes / Objects

3.4.1 TicketingSystem

3.4.1.1 Attributes:

- ticket: String
- user: String
- isUserLoggedIn: boolean

3.4.1.2 Functions

- purchaseTicket(ticket, user, isUserLoggedIn)

3.4.2 PaymentSystem

3.4.2.1 Attributes:

- payment_info_fullname: String

- payment_info_card: Integer

3.4.2.2 Functions:
- payCC(payment_info_fullname, payment_info_card)
- payDC(payment_info_fullname, payment_info_card)
- payPayPal(payment_info_fullname, payment_info_card)

3.4.3 WebsiteSystem
3.4.3.1 Attributes:
- userLimit: Integer
- siteDown: boolean

3.4.2.2 Functions:
- checkWebsiteStatus(siteDown)
- checkCurrentUserCount(userLimit)

3.4.4 TicketPurchased
3.4.4.1 Attributes:
- ticket: String
- movieName: String
- dateOfPurchase: String
- user: String
- userEmail: String

3.4.4.2 Functions:
- sendUserCopyOfPurchase(user, userEmail, ticket, movieName, dateOfPurchase)

3.4.5 UserLoginSystem
3.4.5.1 Attributes:
- user: String
- password: String

3.4.5.2 Functions:
- userInfoLogin(user, password)

3.5 Non-Functional Requirements

3.5.1 Performance

- The front page supporting a maximum of 10,000 users per hour should provide an 8-second or less response time when on a browser, regardless of the device used to access the site.
- Movie listings should be displayed within 5 seconds for approximately 90% of the searches in normal conditions.
- The system should refresh and display updated movies within 10 seconds of receiving new data and information for viewers.

3.5.2 Reliability

- The website should perform without failure or downtime in 95% of use cases during every month.
- All transactions should be processed with 100% accuracy in information, and the system should ensure the safety of customers' data at all times.

3.5.3 Availability

- The website should be available to users 99% of the time every month, providing service during all 24 hours of the day.
- The system should perform consistently across different devices and operating systems with a high-reliability rate unless the user experiences issues on their end.
- The purchasing process must be uninterrupted 99.99% of the time under normal network conditions.

3.5.4 Security

Since we will hold credit card information, security will be a top priority. When collecting credit card information we will store it in separate database storage. The use of cryptography will also be in place in case of potential security breaches. As for account information, we will prompt the user with a valid password and provide equal importance as it will also hold credit card information. Limit access between parts of the program.

3.5.5 Maintainability

The webpage must be kept up to date to be compatible with internet web browsers. This would also include bug fixes and potential security updates which would be found through constant monitoring and vulnerability evaluation.

3.5.6 Portability

The use of Python was chosen for this project and apart from its many benefits it's also excellent portability which supports portable GUIs to Unix, Windows, and MacOS. Making most of the code portable to other host machines and operating systems.

3.6 Inverse Requirements

- User should not be able to create an account with invalid account info for example email, phone number, date of birth, etc.
- User should not search or enter invalid search keys.
- User should not be able to make another account under the same email, phone number, or username.
- User should provide valid payment information.
- The system should not allow the user to purchase over the limit of five tickets.

- User’s payment of choice should have viable funds needed for purchase.
- System should not allow over 10,000 users on-site at the same time.
- Users cannot access the site unless they have wifi.

3.7 Design Constraints

Time constraint: deadline, clients want the project to be completed at a firm deadline. Financial constraints: manpower, the limit of human resources causes a limit of testing done in the project which could lead to bugs in the website and in worst cases security breaches. The scope of the website could also be affected as a limit of people working on it combined with a time constraint could lead to a loss of features for the website to satisfy the time constraint. The budget could also lead to technical constraints, one of which is the limit of bandwidth which stated previously 10,000 users at a time, at this point network congestion will occur and users will experience serious lag and loading times. Lack of employee knowledge in front end development. Employees aren’t as knowledgeable in doing front end development which will result in slower work being done.

3.8 Logical Database Requirements

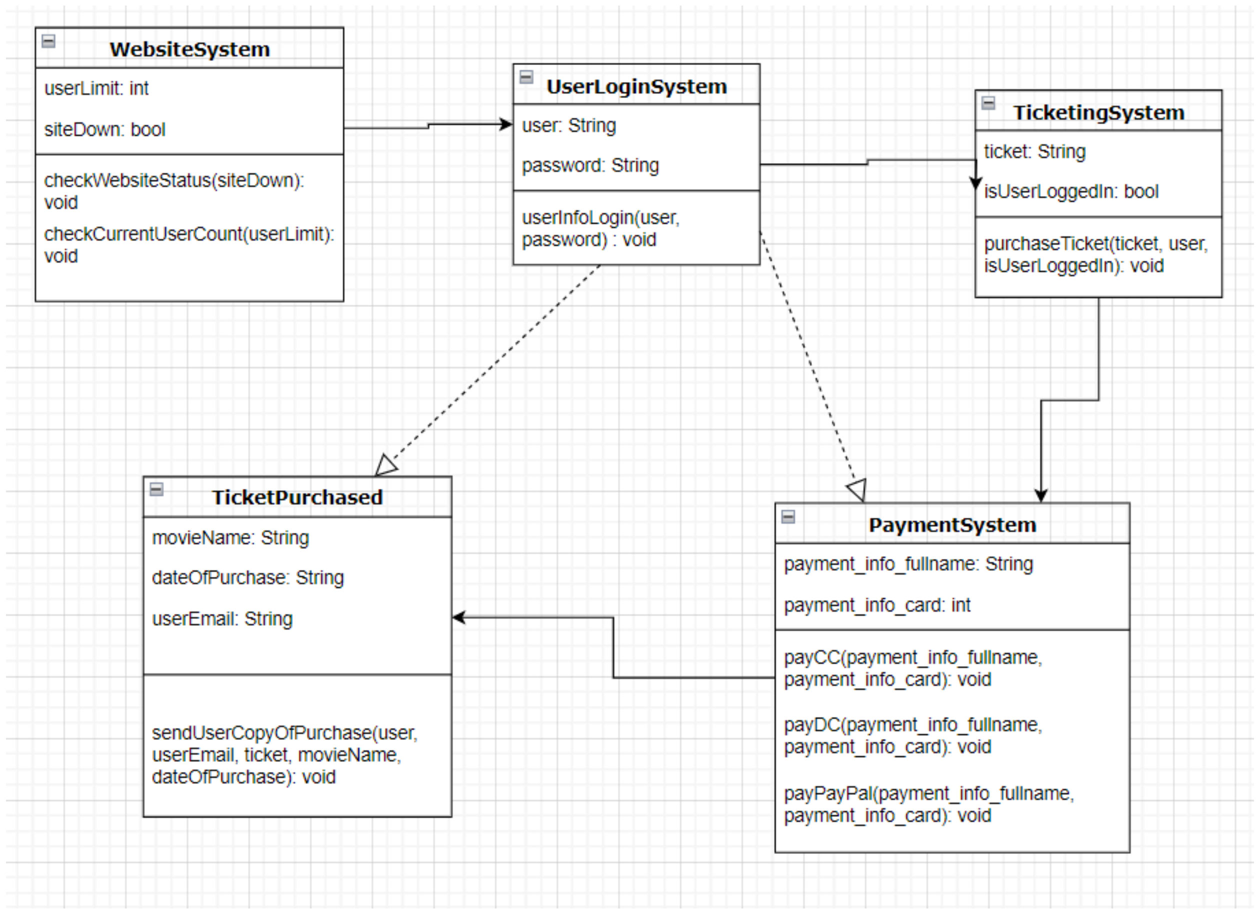
Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc

3.9 Other Requirements

Catchall section for any additional requirements.

4. Analysis Models

4.1 UML Class Diagram


WebsiteSystem
userLimit: int
siteDown: bool
checkWebsiteStatus(siteDown): void
checkCurrentUserCount(userLimit): void
UserLoginSystem
user: String
password: String
userInfoLogin(user, password) : void
TicketingSystem
ticket: String
isUserLoggedIn: bool
purchaseTicket(ticket, user, isUserLoggedIn): void
TicketPurchased
movieName: String
dateOfPurchase: String
userEmail: String
sendUserCopyOfPurchase(user, userEmail, ticket, movieName, dateOfPurchase): void
PaymentSystem
payment_info_fullname: String
payment_info_card: int
payCC(payment_info_fullname, payment_info_card): void
payDC(payment_info_fullname, payment_info_card): void
payPayPal(payment_info_fullname, payment_info_card): void

WebsiteSystem: A system that checks if the current site is up and running before any further processes such as logging in, viewing tickets, or even purchasing tickets can be performed. If requirements are met, the system will then direct the number of users allowed on the site.

- **Attributes:**
  - **userLimit: int**- stores the number of current users on the website.
  - **siteDown: bool** - stores true/false if the site is currently down or not.
- **Functions:**
  - **checkWebsiteStatus(siteDown): void** - uses the parameter of siteDown to check before users are able to access the site, log in, view tickets, or even purchase tickets.
  - **checkCurrentUserCount(userLimit): void** - uses the parameter of userLimit to see if the threshold of maximum users on the site is met before allowing any further users onto the site.

UserLoginSystem: A system that allows the user to log in by existing credentials.

- **Attributes:**
  - **user: String** - stores the user's designated username that they have created and chosen for themselves.

    - **password: String** - stores the user's designated password that they have created and chosen for themselves.
- **Functions:**
    - **userInfoLogin(user, password): void** - checks whether the user is in the matching database with the parameters, allowing them to successfully log in if there is a match.

TicketingSystem: A reliable system that allows the user to go through the ticketing process of a movie of their choice.

- **Attributes:**
    - **ticket: String** - stores the ticket's unique ID combination consisting of integers and characters.
    - **user: String** - stores the user's designated username that they have created and chosen for themselves.
    - **isUserLoggedIn: bool** - checks whether the user is currently logged in or not within the system.
- **Operations:**
    - **purchaseTicket(ticket, user, isUserLoggedIn): void** - a function that uses the parameters to proceed through the purchasing stage.

PaymentSystem: The following stage and system after the TicketingSystem procedure where the user will go through the process of filling out their private information to purchase tickets of choice.

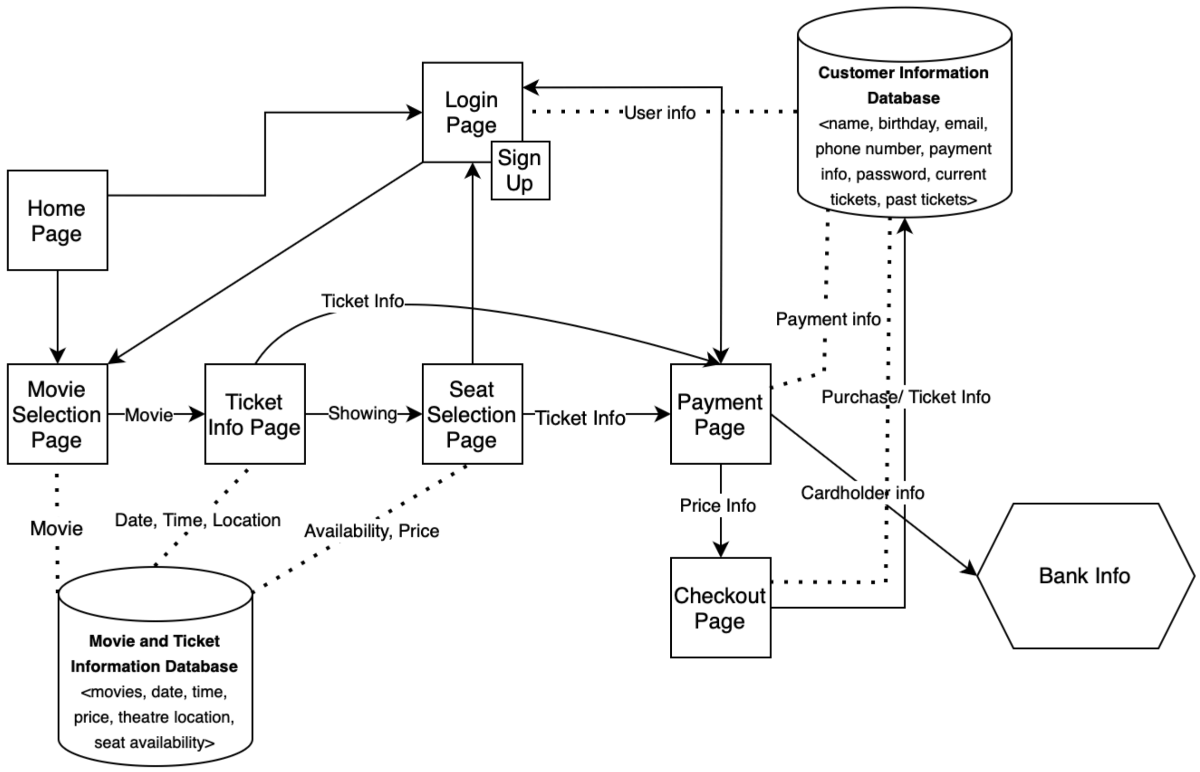
- **Attributes:**
    - **payment_info_fullname: String** - stores the user's full legal name, consisting of first and last name.
    - **payment_info_card: int**- stores the user's card number whether they use a credit card, debit card, Paypal, etc.
- **Functions:**
    - **payCC(payment_info_fullname, payment_info_card): void** - a function where the user will proceed by paying with a credit card using the parameters storing their full legal name and their credit card information.
    - **payDC(payment_info_fullname, payment_info_card): void** - a function where the user will proceed by paying with a debit card using the parameters storing their full legal name and their debit card information.
    - **payPayPal(payment_info_fullname, payment_info_card): void** - a function where the user will proceed by paying with a Paypal using the parameters storing their full legal name and their Paypal information.

TicketPurchased: The final stage of the user's experience after successfully logging in, picking tickets of their choice, and purchasing said tickets where they will receive a copy of what they have purchased in full details.

- **Attributes:**
    - **ticket: String** - stores the ticket's unique ID combination consisting of integers and characters.
    - **movieName: String** - stores the movie's title and name.
    - **dateOfPurchase: String** - stores the date the ticket was processed and purchased.

    - **user: String** - stores the user's designated username that they have created and chosen for themselves.
    - **userEmail: String** - stores the user's designated email that they have chosen for receiving the details of their purchase.
- **Functions:**
    - **sendUserCopyOfPurchase(user, userEmail, ticket, movieName, dateOfPurchase): void** - a function that uses the following parameters to send the details of the purchase of tickets directly to the user using their email address they've chosen. Said details will contain the ticket's unique ID, the movie of the ticket(s), and the date that the purchase was made successfully.

4.2 Software Architecture Diagram



Movie and Ticket Information Database: The Movie and Ticket Information Database holds information such as movies, date, time, price, theater location, and seat availability.

Customer Information Database: The Customer Information Database holds information such as user name, date of birth, phone number, payment info, password, current tickets, and past tickets.

Home Page: The Home Page serves as the main welcome and landing for the website. It holds general information about the website and its purpose. Page will hold the most visual interest.

From the Home Page the user can choose to login or to search for a movie. If they choose to login they will be taken to the Login Page. If they choose to search for a movie they will be taken to the Movie Selection Page.

Login Page: The Login Page allows the user to enter their username, which is their valid email, and password. From the Login Page the user can go back to the Movie Selection Page and continue to search for a movie. From the Login Page they can also edit their account information and view past and current tickets. This information is directly related to the Customer Information Database.

Sign Up: Embedded into the Login Page is the Sign Up option, it allows users to create an account by entering their account information including name, date of birth, phone number, username, and password. This information is directly related to the Customer Information Database.

Movie Selection Page: The Movie Selection Page allows the user to search for movies and select one. Movie information is directly related to the Movie and Ticket Information Database. Once the user selects a movie they will be taken to the Ticket Information Page.

Ticket Information Page: Based on the movie selected from the Movie Selection Page, the Ticket Information Page allows the user to view further information about the available tickets. Date, Time, and Location information is directly related to the Movie and Ticket Information Database. This information will also be displayed on the Payment Page and Checkout Page. Once the user selects a showing they will be taken to the Seat Selection Page.

Seat Selection Page: Based on the showing selected from the Ticket Information Page, the Seat Selection Page allows the user to view further information about their seats including price and availability. Availability and Price information is directly related to the Movie and Ticket Information Database. This information will also be displayed on the Payment Page and Checkout Page. Once the user selects a ticket they will be taken to the Payment Page.

Payment Page: Based on the Ticket selected from the Seat Selection Page, the Payment Page allows the user to enter payment information such as card number, ccv, name, expiration, etc. If the user is not already logged in they will be redirected to the login page before they are allowed to continue. Payment Information is directly related to the Customer Information Database. The Payment Information is also directly sent to the Bank Information to be verified. Once the user submits valid payment information they will be taken to the Checkout Page.

Checkout Page: Based on the price and ticket info selected from prior pages, the Checkout Page is a final purchase verification page that allows the user to either confirm or cancel their purchase. If they confirm their purchase, their purchase and ticket information will be sent to the Customer Information Database. The user will receive a confirmation email about their purchase and ticket information.

4.3 Development Plan and Timeline

Partitioning of tasks and responsibilities:

Phase 1: Planning and Requirements (1.5 weeks)

Employee 1:

- Task: Gather funds and figure out budgeting for project
- Responsibility: formulate and present budget plan to other members of team, as well as client of the project

Employee 2:

- Task: Contact client and regularly meet with them for project details
- Responsibility: file and report list of details and requirement client wants for the website

Employee 3:

- Task: Set up the coding project for group members, assigning them the parts they will do
- Responsibility: Partitioning roles to programmers as well as deadlines and requirements using client's information

Phase 2: Design and Prototype (2 weeks)

Employee 1:

- Task: Purchase and afford the tools needed for the development of the project
- Responsibility: Make sure the tools needed stay within the budget

Employee 2:

- Task: Gather data and information on current or upcoming movies
- Responsibility: Consistently keeping up to date with the data and updating it within the project

Employee 3:

- Task: Set up design and layout of the website
- Responsibility: Set up the prototype of the website and check in with client on any changes for details

Phase 3: Development (4 weeks)

Employee 1:

- Task: Finish coding up their part on the website's capabilities, including user interface and user load
- Responsibility: Keep in contact with other team members and client on any details or notices needed

Employee 2:

- Task: Finish the login system for users
- Responsibility: Make sure it aligns with the client's requirements as well as making sure it transitions smoothly for the other programmers to use in their part

Employee 3:

- Task: Finish the movie page setup on site

- Responsibility: Design how a movie would look if a user were to click on it and view the details, contact the other programmers and client on any questions or conflicting details

Employee 4:

- Task: Finish the purchasing system
- Responsibilities: Making sure that the system is completely functional and completed by deadline for testing

Employee 5:

- Task: Finish the personal purchase details of tickets that will be a copy and sent to users who successfully completed purchases
- Responsibilities: Complete code for purchase details, checking in with other programmers and client for any mishaps or concerns.

Phase 4: Testing and Finalization (2 weeks)

Employee 1:

- Task: Test UI
- Responsibility: Look at the user interface and make sure everything is as stated in the plan. Fix anything that is wrong.

Employee 2:

- Task: Stress test the website
- Responsibility: In case of any serious defects becoming uncovered, report to the team and fix it immediately.

Employee 3:

- Task: Path testing
- Responsibility: Make sure that the test cases throughout the program are all executed a minimum of one time. If not, report to the team and fix the issue.

Employee 4 :

- Task: Run the website through a quality control through multiple browsers.
- Responsibility: Check how the performance and UI respond to multiple web browsers in respect of the expectations we set. If anything is poor, report to the team and fix it.

Phase 5: Deployment and Maintenance (1 week, continuous)

Employee 1:

- Task: Check on bandwidth effect on database
- Responsibility: Check for possible lag caused by multiple users to make sure its effect isn't severe.

Employee 2:

- Task: Make sure deployment of website runs smoothly
- Responsibility: For the first week keep a close eye on the website from the users POV, report anything wrong as soon as possible to the team.

Employee 3:

- Task: Keep website up to date
- Responsibility: Make sure everything is as optimal as can be. Fix anything that is broken and continuously monitor the performance of the website.

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 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

Our repository:
[CS250-Moviicket-SRS]

Our commits:
Shelley: UML Class Diagram
Adelina: Software Architecture Diagram
Amilcar: SRS Document