# Software Requirements Specification Template

Software Engineering

The following annotated template shall be used to complete the Software Requirements Specification (SRS) assignment.

**Template Usage:**
Text contained within angle brackets ('<', '>') shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either 'Smart Home' or 'Sensor Network'.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

## Group Github:

https://github.com/PeavyMan/CS-250-Skyflicks

# SkyFlicks Drive-In

Software Requirements Specification

Version 1

2/15/2024

Group 7
Josue Favela-Pacheco, Jayden Dy, Marcus Bowman

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

## Revision History

| Date | Description | Author | Comments |
|---|---|---|---|
| <date> | <Version 1> | <Your Name> | <First Revision> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 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

*The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document. (Note: the following subsection annotates are largely taken from the IEEE Guide to SRS).*

## 1.1 Purpose

The purpose of this SRS document is to outline and specify the requirements for the drive-in movie ticketing system and to describe its complete functionality.

## 1.2 Scope

1) **Database management system** (DBMS) is used to manage and store any information on the website using tables of information. The first table of information that will be tracked inside of this software product will be the user's name, email, phone, order details. This DBMS will be responsible for holding all of the users information when they purchase a ticket. This product is essential to the functionality of the website as the AERS will use this product to confirm user details. These orders must also be linked to the account of the user purchasing the ticket. In addition, this is essential for administrators to diagnose user problems inside of the CSSS. The second table of information that will be tracked is used to manage, maintain, and create specific user accounts. This database system will also include a royalty program and will award points to users who spend more money. As such, the database system must account for a user's unique username, password, reward points, and email associated with the account. The last table of information inside of the database management system is used to account for the movies currently airing. Inside each movie should have a name, genre, and age rating. It’s important to note that this database system must only be accessible by the administrators and not be available to the public.

2) The **automatic email report service** (AERS) is designed to automate the communication process with users regarding their orders. It will send emails to the recipients containing order information such as order number, order date, order recipient, and detailed order specifics. It works with the DBMS and queries the information to generate the report. The primary purpose is to enhance user satisfaction by giving users a report they can refer back to. This is an automatic service and its source code should only be accessible by the developers who work on developing and maintaining it.

3) The **filter** allows users to efficiently search for movies based on name, genre, and age rating. It will offer a user-friendly interface with a GUI to enhance easy access to desired content. It’s important to note that all movies shown onto the website must be movies that are currently airing by using the DBMS to query for movies of a specified user request.

4) **Customer Support and Service System** (CSSS) is a support system to handle customer inquiries, complaints, and feedback related to the Drive-In Movie Ticketing System. The CSSS has a service that runs in the back end and is only accessible to approved administrators to provide support. The front end of the CSSS is accessible for everyone who has an account to get support from the live staff.

## 1.3 Definitions, Acronyms, and Abbreviations

| | |
|---|---|
| **DBMS** | Database management system |
| **SRS** | Software requirements specification |
| **AERS** | Automatic email report service |
| **CSSS** | Customer support and service system |
| **HTTPS** | Hypertext transfer protocol secure |

## 1.4 References

Documents referenced:
**IEEE Recommended Practice for Software Requirements Specifications**, 20 October 1998, The Institute of Electrical and Electronics Inc.
This document can be obtained here: IEEE Document
**NATO 1968 (NATO Science Committee):**
This document can be obtained here: NATO
**No Sliver Bullet**, 1986, University of North Carolina at Chapel Hill
This document can be obtained here: No Sliver Bullet

## 1.5 Overview

Section 2 - General description: Provides an overview of the project and how it will function including a product perspective, product functions, user characteristics, general constraints, and assumptions and dependencies. It will provide the reader with a better understanding of requirements.

Section 3 - Specific requirements: The meat and bones of this document and outlines all of the software requirements in depth. It includes topics such as external interface requirements, functional requirements, use cases, classes/objects, non functional requirements, inverse requirements, design constraints, and logical database requirements.

## 2. General Description

*This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.*

### 2.1 Product Perspective

*This software is related to every checkout process in online stores and shopping. This software is also related to review websites due to being able to review movies. This software is also related to GPS or map software due to needing to locate the nearest theater to the user.*

### 2.2 Product Functions

*The software will provide a faster and simpler way of purchasing movie tickets. This software will also provide a way to buy movie tickets for drive-in movie theaters. Within the software, you can read and write reviews on films and movies. In the software, the ability to order food is an option.*

### 2.3 User Characteristics

*The users of this software are going to be movie watchers. For the system, the user will have to be 13 years old or older to purchase a ticket. The user needs to have an electronic device that has access to a browser. The user is going to need to be able to access the internet and provide credit card details for purchase.*

### 2.4 General Constraints

*Some constraints include being able to access the website. The user is going to need to have an electronic device that can access the internet and browsers to access the website. Other constraints are not making the system too powerful so it can run and operate on most users' devices. Another limitation is the user not being near a theater to purchase a ticket. The website is going to need to be able to handle at least 1000 people on the website.*

### 2.5 Assumptions and Dependencies

We are going to assume that the user hardware has a compatible operating system such as Windows or MacOS. If the specific OS is not available we are going to need to change our software requirements. We are also going to assume that the user has a modern web browser including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, as well as mobile browsers on iOS and Android devices. The impact of the user not having a web browser mentioned would affect the software requirements. We are going to assume that the customer has reliable internet connectivity to access the website. If the user does not have reliable internet then the website won’t be able to function in its intended way.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

- The User interface design should enable the customer to purchase and book a movie ticket with ease. The focus will be on usability, simplicity, and rapid access to useful information.
- The GUI will have a responsive design that can adjust to various screen sizes and devices. It will have easy-to-use action buttons for choosing movies, showtimes, seats, and food, as well as marked movie posters. Dropdown menus and time widgets are examples of interactive elements that make it simple to choose your preferred movies.
- To make sure the website works for all users, the website will support several languages. Movie titles, descriptions, and other textual content will be dynamically translated based on user preferences.
- The method for purchasing tickets should be simple, assisting customers in choosing their preferred movie, showtime, theater, and seating arrangement. Users will be able to easily and securely complete the transaction by entering their payment information through a secure checkout process.

### 3.1.2 Hardware Interfaces

- On desktop and laptop computers, the movie ticketing website will be available through web browsers such as Microsoft Edge, Mozilla Firefox, Google Chrome, and Safari. It is necessary to guarantee compatibility with operating systems, such as Windows, macOS, and Linux.
- Users will use standard input devices, such as keyboards and mice, to interact with the website. The website has to support standard input functions like text entry and scrolling to provide a consistent user experience across various input methods.
- The website will display content on a range of output devices, such as laptop screens and monitors. It should be compatible with all different size screens, resolutions, and aspect ratios, providing a pleasant viewing experience.
- For users to access movie listings, showtimes, theater information, and to complete ticket purchases, the website must have an active internet connection.

### 3.1.3 Software Interfaces

- To enable safe online ticket purchases, the movie ticketing website will integrate with third-party payment processing services. Communication with payment gateway APIs offered by payment service providers like Stripe, PayPal, or comparable services will be a part of this integration.
- The website must accept several payment options, such as debit and credit cards, and digital wallets (like Apple Pay and Google Pay).
- To dynamically populate movie listings, showtimes, theater information, and other related content, the movie ticketing website must integrate with external movie database APIs. This integration will involve interacting with movie database APIs like IMDb.

#### 3.1.4 Communications Interfaces

- To guarantee secure communication between clients (web browsers or mobile devices) and the web server hosting the website, the movie ticketing website must make use of the HTTPS protocol.
- To offer extra features or services, like email notifications, chatbots for customer service, or social media authentication (e.g., Facebook or Google login), the movie ticketing website may integrate with third-party services and APIs.
- Real-time updates for specific features or functionalities, like countdown timers for ticket reservation timeouts, notifications for special promotions or events, and live seat availability and pricing information updating, may be implemented on the movie ticketing website.

## 3.2 Functional Requirements(Marcus)

*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 <Data provided by UDMS(user data) shall be stored in the DBMS(our database)>

3.2.1.1 Introduction

- The movie ticket system will maintain information about the user's full name, street address, credit/debit card information(if they decide to create an account), movie showtimes,prices,skyCoins and ticket bookings.

3.2.1.2 Inputs

- Very high criticality ; all this information will be <u>private:</u>
- User information will be stored in the database that is encrypted

3.2.1.3 Processing

- Low bandwidth/wifi availability could present a technical challenge

3.2.1.4 Outputs

- The above stated factor is a risk that will be encountered. We prevented this from becoming a bigger issue if it factors in by reducing dependency on internet service.

3.2.1.5 Error Handling

- This requirement is the base of the project; all other functionalities depend on this working

#### 3.2.2 <Data shall be accessible through queries and stored data should be able to be manipulated through forms>

3.2.1.1 Introduction

- Database users should have the capability to generate reports based on the data being stored in the database. Also items and other personal information should able to be added and updated through the use of forms

3.2.1.2 Inputs

- Very high criticality; too keep away hackers from trying to access data, p

3.2.1.3 Processing

- If technical difficulties arise, our customer service will step in and try to solve the problem as soon as it's been reported

3.2.1.4 Outputs

- Given access into private data, this requirement is satisfied

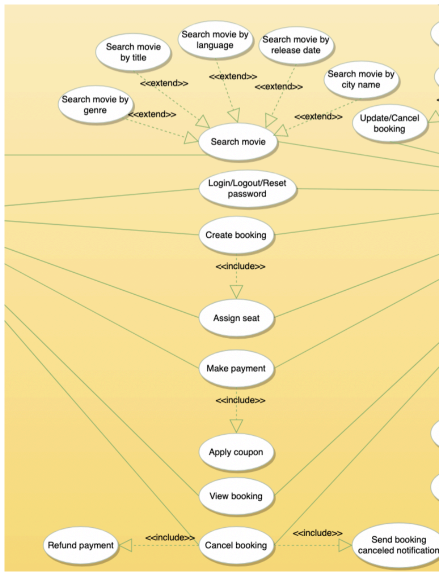
3.2.1.5 Error Handling

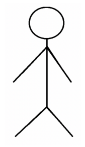
- This requirement depends on requirement number one.

## 3.3 Use Cases(Marcus)

### 3.3.1 Use Case #1:

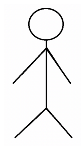
The Customer/Guest will be able to access the app through these sequence of steps:



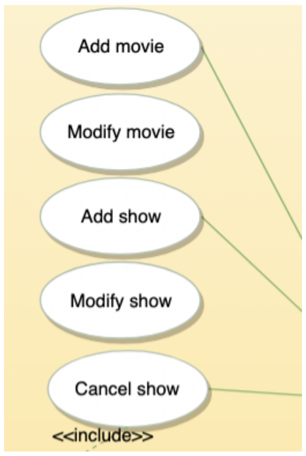


Guest —-----> (Enters the app)

### 3.3.2 Use Case #2



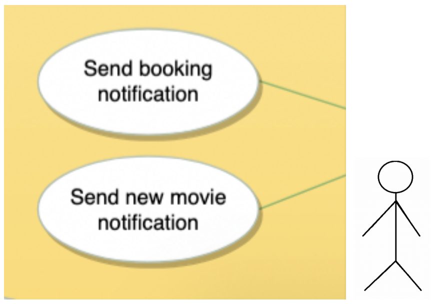
admin←—-



Admin: On the clock employee, can modify any data that needs to be switched out. Mostly just adding,modifying or canceling movies

**Use Case #3**

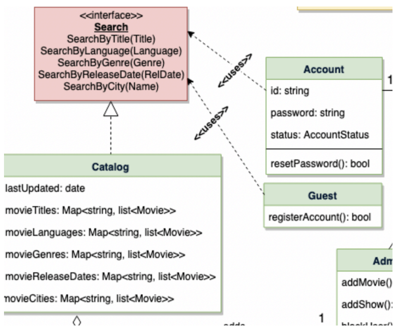
- Our system algorithm will automatically send booking notifications, send any new movie notifications



←—-System

## 3.4 Classes / Objects(Marcus)

### 3.4.1 <Class / Object #1>


<<interface>>
Search
SearchByTitle(Title)
SearchByLanguage(Language)
SearchByGenre(Genre)
SearchByReleaseDate(RelDate)
SearchByCity(Name)
<<uses>>
<<uses>>
Account
id: string
password: string
status: AccountStatus
resetPassword(): bool
Guest
registerAccount(): bool
Catalog
lastUpdated: date
movieTitles: Map<string, list<Movie>>
movieLanguages: Map<string, list<Movie>>
movieGenres: Map<string, list<Movie>>
movieReleaseDates: Map<string, list<Movie>>
movieCities: Map<string, list<Movie>>
addMovie()
addShow():
1
1

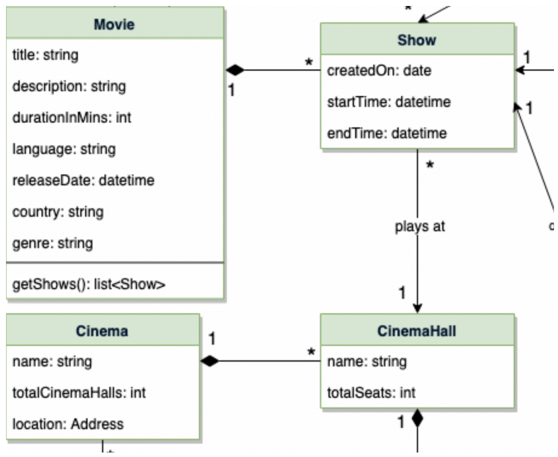
3.4.1.1 Attributes

- We made sure to add all main attributes to make this movie ticket system function properly, an excessive amount of classes will need to be created to best organize this. Making sure to add <<"relationships">> objects to classes is very important, this will help us not get lost and make mistakes

3.4.1.2 Functions

- This class updates the date everyday, shows the movie listings, shows the list of movies based on genre, shows list of release movie dates and also movies based on cities.

### 3.4.2 <Class / Object #2>



3.4.1.1 Attributes

- We made sure to simplify our classes as best as possible to prevent any errors or bugs, and also adding<<relationships>> objects to classes.

3.4.1.2 Functions

- Descriptions of the movie, cinema show and which cinema hall is the movie being shown at are very important to be kept organized, we made sure that every class and attribute was purposely made for usage.

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

- The system shall load any user interface screen within 3 seconds assuming the user has proper network connection.
- When a user filters for a specified movie requirement inside of the catalog, the user interface must transition to show the updated catalog within 5 seconds.

### 3.5.2 Reliability

- There should be a 99.9% success rate of processing transactions that provide required information.
- Should any unsuccessful transaction occur, there must **always** be an error message to tell the user the problem.
- 100% of user transactions should come with a confirmation email within 10 minutes of purchase from the AERS. This email should also come with a unique order number.

### 3.5.3 Availability

- Maintenance/downtime for ticketing systems should be limited to 15 minutes every day (must be at a time when usage is low).
- The servers of this system should be able to support at least 10,000 active users at a time.

### 3.5.4 Security

- 100% of all user information should be encrypted/hashed inside of the DBMS to provide an extra layer of security should hackers get access.
- Security audits should be run once a week over every product to ensure hackers don't have access to classified user information.
- The website should be run over the HTTPS protocol to secure data.

### 3.5.5 Maintainability

- Automatic updates/patches must run in the background at every use and should prompt the user if they want to update.
- 100% of all code should have some sort of documentation to ensure clarity of what's beig done.

### 3.5.6 Portability

- The system should be optimized to run across any user browser and must be able to support mobile devices that have access to the internet.
- The system should be able to support and translate plaintext to the top 5 most popular languages.

## 3.6 Inverse Requirements

- The system shall not allow tickets to be sold once a specified number of tickets have been sold
- The system shall not generate duplicate order numbers
- The user shall not be able to modify or cancel their tickets 2 hours before the movies start time
- The system shall not show events of show times that have already passed.
- The system shall not assign designated spots as the drive in will give priority to whoever shows up first.

## 3.7 Design Constraints

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

*Compliance and Standards: The website must comply with the payment and bank industry. This means the website must include the encryption of credit and debit card information. The website must also comply with disabled people to ensure they are well accommodated. Company Policies: Return Policy, the website/company will not accept any refunds or cancelation 14 days* *before the movie's airing. Technical constraints: the group of engineers building this website is small, only 3 engineers, thus this will limit the amount of work each one of us could do. Financial constraints: the team does not have sufficient funds to run the website up to its full capabilities. Talent constraints: the group of engineers working on this website are only undergrad students, the group does not yet have the full experience of a graduate-level engineer. Hardware Limitations: Server capacity, the website must be designed to be able to operate between the limits of the servers, this also includes during peak times. Technological Considerations: The website must be compatible with the latest versions of major browsers, including Chrome, Firefox, Safari, and Edge. Load Times: The website must be optimized for fast loading times, with a target of loading the main content within 2 seconds under normal conditions.*

### 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.*

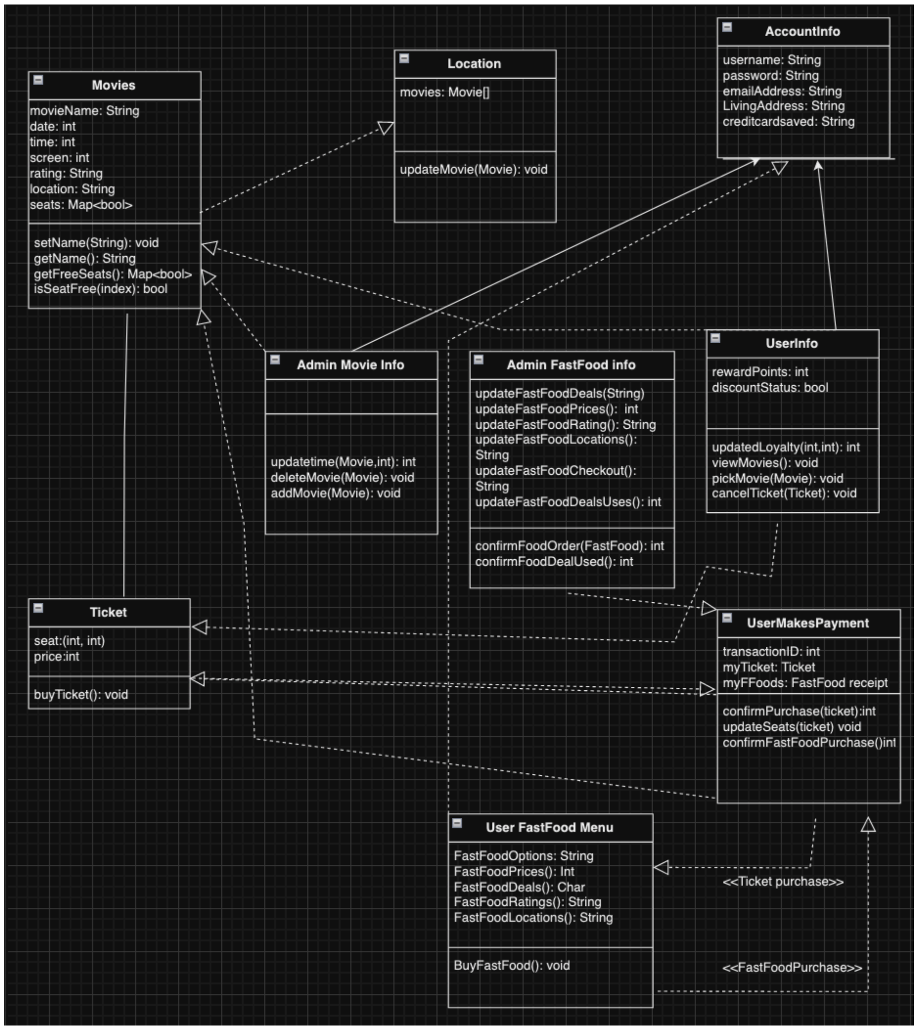
## 4. Analysis Models

*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 System Description

SkyFlicks is a drive-in movie ticketing website that is designed to provide an easy and seamless user-first interface for any customers who would enjoy the experience of a drive-in movie theater. The website enables users to browse newly released movies, select showtimes, purchase tickets, and access information about movies, all in one place. The purpose of Skyflicks is to provide a way of purchasing and booking tickets to drive-in movie screenings online. It aims to streamline the ticket purchasing process and reduce the amount of time the customers spend waiting in line or accessing this information on this website from other sources. It is here to provide a convenient way for moviegoers to plan their trip to the drive-in theaters. Some key features of the website include movie listing, ticket booking, user accounts, and information and support. Users can view the whole catalog that is available at drive-in movie theaters. They can see movies that are currently showing or movies that are coming soon, with complete descriptions, ratings, trailers, and showtimes of each movie. The website will also provide a secure and seamless process for selecting and purchasing a ticket online. The website will support multiple forms of payment methods, including credit/debit cards and online payment systems like PayPal. The website will also support customers creating accounts to manage and track their bookings, check payment history, receive rewards, and receive recommendations on movies they will enjoy based on previous viewing. The website will also try to provide information on all of the drive-in movie theaters including location, parking maps/ guidelines, snacks and food offered, and contact information of the theaters.

## 4.2 UML Class Diagram



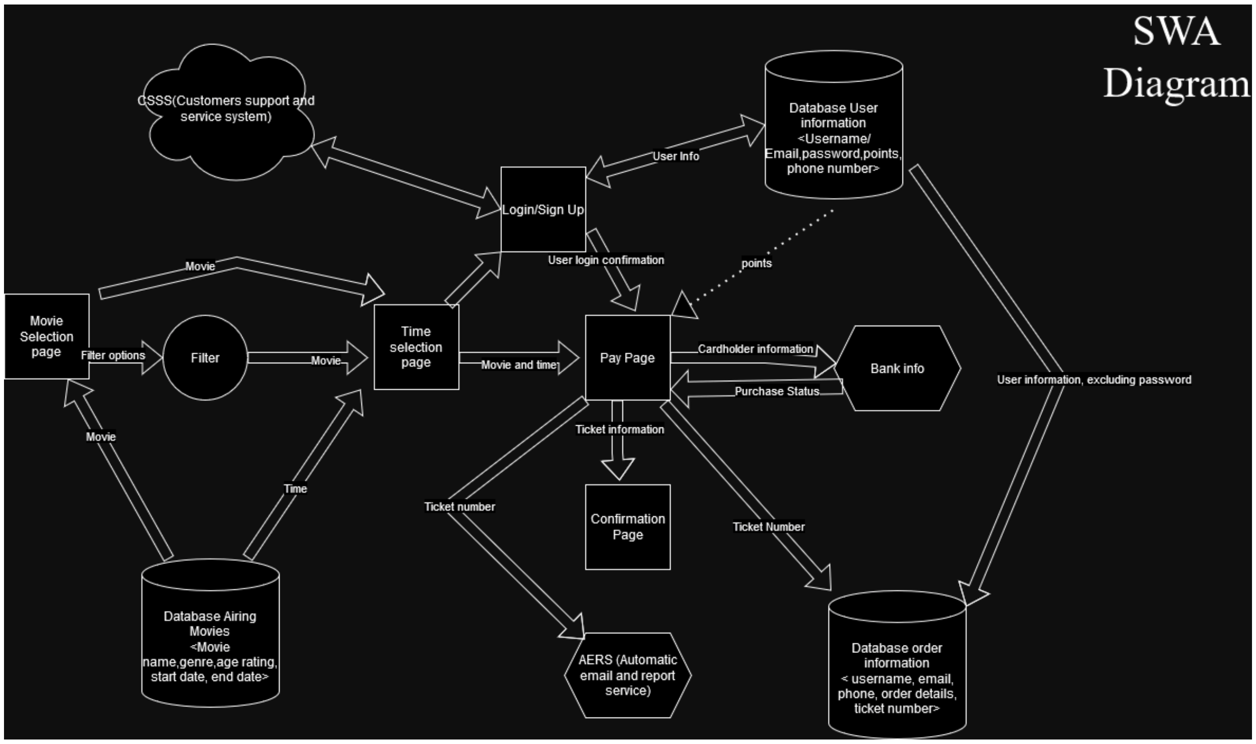
- Description of classes:
- <u>Movies</u>: Will show the user the movieName, date, time, location, and seats.
- <u>Ticket</u>: Contains which seats were picked and the price.
- <u>AdminFastFoodInfo</u>: Contains methods that allows the Admin to take control and easily change the FastFood Menu when needed.
- <u>UserFastFoodMenu</u>: Contains FastFood Options, Prices, Deals, Ratings, Locations and
- <u>UserMakesPayment:</u> Contains the transactionID:(unique receipt), myTickets(User's Tickets) and Fastfood Receipt( That's if the user decided they buy food through the app)
- <u>Location</u>: Show the movies that are being shown at that Location, through the Movie array method and can be updated if needed to.
- <u>Admin Movie Info:</u> Contains methods that allow for you to update the movie listings and times.

- Account Info: Contains username, password, email Address and Living Address; User that decide that they want to stay as a Guest will have to put in this information every time since they're data won't be saved
- UserInfo: Contains the rewardPoints, discounts, viewMovies and pickMovies. These options are only available if the user decides to create an skyFlicks account instead of a guest account.

● Description of attributes:
- buyTickets(): This attribute contains methods that ultimately come together to provide you with the opportunity to pick the seats once you picked the movie you want to see, and at the end gives you the total price.
- ConfirmFastFoodOrder: This attribute contains methods that ultimately come together to provide you the purchase confirmation and receipt of your food.

● Description of operations:
1. When the New User/User opens the app it will automatically show "Now Showing Listings" of the top 10 movies in the geolocation the user is in.( User begins in Location Class)
2. The geolocation of the user will show an Array of movies that are showing near him. Once he/she selects what movie they want to watch.
3. They now can see which locations have the movie they want to see, once they pick the location they want to watch the movie at. they will be directed to the Ticket Class where they will select the seats and get told the total price.
4. If they want to go ahead and purchase the movie tickets, they will move to the UserMakesPayment Class where you put in your credit/debit card information to purchase your tickets. Once you click "purchase" and the transaction goes through, the system will automatically send you a unique 12 digit code which serves as your receipt.
5. Once the Users buy tickets to a movie, they are now allowed to purchase local fast food deals, our movie ticket system app will partner up with local fast food shops and chains to provide exclusive deals that only skyFlicks account members can use. Guest Users can only use the fast foods deals on Tuesdays and Thursdays, however weekends and holidays are restricted.
6. If the user decides they want to buy food they can pick from a variety of options, once they pick the deal they want they will be redirected back to the UserMakesPayment to pay for your food.
7. Once you've paid the system will send you a confirmation code confirming your food purchase(Receipt), you are now ready to go. Make sure to pick up your food and get to the showing on time. Enjoy the show.

## 4.3 Software Architecture diagram


SWA Diagram
CSSS(Customers support and service system)
Database User information <Username/ Email,password,points, phone number>
User Info
Login/Sign Up
User login confirmation
points
Movie
Movie Selection page
Filter options
Filter
Movie
Time selection page
Movie and time
Pay Page
Cardholder information
Bank info
Purchase Status
User information, excluding password
Ticket information
Confirmation Page
Ticket number
Ticket Number
Movie
Time
Database Airing Movies <Movie name,genre,age rating, start date, end date>
AERS (Automatic email and report service)
Database order information < username, email, phone, order details, ticket number>

**SWA Description:**
When the user first loads the website of our movie ticketing system, the database loads the movies that are currently airing onto the UI. The user can either proceed by clicking on the desired movie they wish to see or they can utilize the filter function which will narrow down the movies on their screen. The filter option includes settings such as a specific name, genre, or age rating. After the user has correctly identified the movie that they wish to see they will be prompted by a time selection page which will ask the user to select a time or date for which they wish to see the movie. The time slots are also sent over by the movies database. After they have specified the time and date, they will be prompted by a login/signup page which is required to proceed. If using a currently existing account, the Database for user information will be queried. If signing up for the first time, the new user sign-in information will be provided as a new slot inside of the User Information database. After our user has successfully logged in, they will be prompted with a payment page which will show the details of their order along with asking for cardholder information. On this page, the user will also be able to view their points from the User Information database and redeem it for a discount of their purchase. Once they submit their payment, the bank checks and verifies if the purchase can be successfully completed and they send a confirmation of the status. Upon a successful status code from the bank, the user is shown a confirmation page that shows their ticketing number. The AERS also sends the user an email associated with their account the ticketing information such as their ticketing number, time, and movie. The database for order information is then updated for every order that occurs. Another feature of the SWA diagram is the CSSS in which it will prompt the user to sign up/log in to receive help from a specialist.

### 4.4 Development plan and Timeline

Josue Favela-Pacheco was responsible for the Design constraints, System Description, and Development plan and timeline. Marcus Bowman was responsible for the UML Diagram and description. Jayden Dy was responsible for the SWA Diagram and description.

## 5. Change Management Process

*Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.*

## A. Appendices

*Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.*

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*

### A.1 Appendix 1

### A.2 Appendix 2

SOFTWARE DESIGN SPECIFICATION DIRECTIONS:

**This is a Group submission assignment**

The second deliverable in the course project (after the requirement specification) is the software design specification that describes the software organization and development plan. In contrast to the software requirements specification that is intended to be read by the client, the software design document is meant for the developers that implement the software components and later maintain the software product. Specifically, the software design specification document should describe the overall software architecture and implementation, at a level of detail similar to a UML Class Diagram. The classes, data structures, together with all major functions and their parameters should be specified.

Some software system descriptions are provided. You are also free to propose your own software system. You will work on the system you select for this assignment for the remainder of the semester.

Submissions will be done through your project group github page. Push your software design specification as a **pdf** or **txt** file to your group project repository, and submit the link to the file. Only one submission per group is required.

Note: to receive credit for the assignment, each group member must push at least one commit to the github repository.

The document submitted by your team should include the following content:

**Title page**

- Software title
- Team members

**System Description**

- Brief overview of system

**Software Architecture Overview**

- Architectural diagram of all major components
- UML Class Diagram
- Description of classes
- Description of attributes
- Description of operations

* descriptions should be detailed and specify datatypes, function interfaces, parameters, etc..

**Development plan and timeline**

- Partitioning of tasks
- Team member responsibilities