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.

Ticket System

Software Requirements Specification

Version I

09/21/2023

Group 13
Ronessa Mose, Samantha Georges, David Prieto, & Swaraj Khatri

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

Revision History

Date	Description	Author	Comments
09/21/2023	Version 1	Samantha, Swaraj, David, Ronessa	As a whole, we corrected various errors and finalized our SRS for the first submission.
10/05/2023	Version 2	Samantha, Swaraj, David, Ronessa	Added the Software Design Specification to the document.

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

6. **Software Design Specification**

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 annotations are largely taken from the IEEE Guide to SRS).

1.1 Purpose

The purpose of this document is to create a ticketing system for multiple theaters within the San Diego area that will be accessible online. It will also detail its requirements, functionality, and constraints.

1.2 Scope

This system is created for AMC theaters, it will allow users through web browser purchase tickets, and phone applications will not be made available. There is a limit of 20 tickets per user while handling a traffic of 10 million customers simultaneously on the website, while simultaneously having the tickets be unique per customer as an NFT to avoid botting and scalping. The administrator will be able to access any data pertaining to the website in order to assist those in need of support.

1.3 Definitions, Acronyms, and Abbreviations

1.2 Scope pertains to 'NFT'. NFT stands for non-fungible token, in which each ticket sold through the ticketing system is unique and cannot be replaced.
2.1 Product Perspective pertains to 'IMDB'. IMDB stands for Internet Movie Database, which is an online database that contains information and statistics on movies, TV shows, and other sorts of entertainment.
2.4 General Constraints pertains to 'UI'. UI stands for User Interface, which is the interaction between a human and a computer over a communication device such as a phone or any digital devices.
3.2.3 Anti-Bot Measures pertains to 'CAPTCHA'. CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart". It is a type of security measure against bots by asking for a human response.
3.2.9.3 Processing pertains to 'API'. API stands for Application Programming Interface, which is a type of software that would enable applications to communicate with each other.

1.4 References

The customer answered two questionnaires for the ticketing system. Respective 25 for the first questionnaire and 15 for the second questionnaire. Two other documents included were a comprehensive overview of the questionnaire's answers.

1.5 Overview

The system is a ticketing system that will become a website that will be able to store securely the information of the users such as payment information, name, purchase history, and email. In addition to avoiding scalpers and bots from abusing the website, while allowing the administrator to view ratings of certain movies and which movies are available. Furthermore, the document will go into depth on how the website will be accessed, designed, and how it will operate.

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 system shall make it easier for those who wish to purchase the tickets through an easier means than going to the physical location of a movie theater by utilizing a website to buy tickets, check the prices and the ratings based upon other critique websites reviews such as IMDB or Rotten Tomatoes on the available movie.

2.2 Product Functions

This system should allow the user to purchase up to 10 tickets past show time, and be able to buy 20 maximum tickets at a time. Users, while purchasing the tickets, will be able to reserve specific seating, as well purchases will be made available two weeks prior to showtime, and ten minutes after. This system will also be able to block bots from purchasing and scalping tickets. Loyalty accounts can be created and loyalty points can be generated through each purchase through the system.

2.3 User Characteristics

The users include the administrator and/or the owner of the website which will be able to access information of the users via entering an administrator mode. Other types of users will be the catered audience of movie enjoyers due to them being the ones purchasing said movie tickets through the system. For the system, the users should have a basic knowledge of using an internet browser and be able to read and input information regarding the users themselves and their payment information.

2.4 General Constraints

The constraints given by the customer are the 500K budget and testing is only within the span of four months. This system will be available only via web browser and not through mobile application, the UI should be easy to understand. The website should be able to withstand the traffic of 1 million active users. Tickets must be unique per-user. This system will only be

available and applicable to theaters within the San Diego area. Prices must be consistent across the system, and each ticket must be unique per user.

2.5 Assumptions and Dependencies

This system runs assuming that the user is on a stable internet connection. If such is not present, the user buying said tickets will not be able to proceed with payment or to continue with the process of reserving seats. In case such an issue occurs, the ticketing system should be able to hold a user's last selected session without charging the user; and the reserved seats will be held for a certain amount of time for the user.

3. Specific Requirements

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

Each requirement in this section should be:

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

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

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface will be accessible through any internet browser, such as Chrome, Safari, Microsoft Edge, etc.

3.1.2 Hardware Interfaces

Because the software is available for all internet browsers, the only hardware interface requirement is that it is capable of connecting to the internet.

3.1.3 Software Interfaces

1. The ticketing system will take in responses from the user (theater location) and display the drop-down menu.
2. The ticketing system must share all of the available showings and movie types across platforms and to the drop-down menu.
3. The ticketing system must confirm whether or not users belong to the membership program.

4. The ticketing system must communicate with the payment software to ensure that the correct billing information is inputted.
5. The payment software must keep a record of all purchases.

3.1.4 Communications Interfaces

The electronic ticketing system will work with any type of internet browser. This is a simple electronic form that is used for ticket purchases.

3.2 Functional Requirements

3.2.1 Server Management

3.2.1.1 Introduction

This section of the document outlines the Server Management feature of the Ticketing System.

3.2.1.2 Inputs

1. The Server Management feature must be able to take in the inputs of at least 1,500 users at any given time. This includes theater location, movie selection, showtime, movie type, payment information, etc.

3.2.1.3 Processing

1. The Server Management feature must be able to interact with the Showtime and Ticket Database feature. Should be able to process the inputs of at least 1,500 users at the same time.

3.2.1.4 Outputs

1. The Server Management feature will not output any information to the user but may relay server and client information to administrators.

3.2.1.5 Error Handling

1. In the case of the software being down, the Server Management feature will display an error message on the home page, alerting users that the site is temporarily unavailable or undergoing maintenance.
    a. If the software becomes overwhelmed by too many users (over 1,500), the Server Management feature will display the same message as above and report the error to administrators.

3.2.2 Web Browser Based Software

3.2.1.1 Introduction

This section of the document outlines the Web Browser Based Software feature of the Ticketing System. In summary, the software will be available for any web browser (Chrome, Microsoft Edge, Safari, etc.)

3.2.2.2 Inputs

1. As a browser application, the user interface will feature a main menu that will allow the user to first select their theater location.
2. It will then lead to a drop-down menu where they may select their movie, showtime, & movie type (deluxe or regular).
3. Once all selections have been made, the interface will prompt the user to state whether they are a member and then proceed to checkout after their input.

3.2.2.3 Processing

1. As a web application, the software will require uninterrupted access to the internet in order to take user inputs, communicate with databases, and operate as expected.

3.2.2.4 Outputs

1. Will output available theater locations, movie showtimes, pricing, and payment receipts to users.

3.2.2.5 Error Handling

1. If the application is unable to run, it will communicate with both the Server Management feature and report the issue to administrators.

3.2.3 Anti-Bot Measures

3.2.3.1 Introduction

This section of the document outlines the Anti-Bot Measures feature of the Ticketing System. This feature will help prevent the usage of bots to purchase large amounts of tickets for high-demand films.

3.2.3.2 Inputs

1. Before checkout, the software will utilize a CAPTCHA test in the purchasing process to ensure that bots are not buying tickets for high-demand films.
2. The user must review the CAPTCHA image and submit an input copying the distorted letters of the test.

3.2.3.3 Processing

1. The software must compare the input to the expected input. If the input is incorrect, it will prevent the user from completing their purchase until they enter the correct information.

3.2.3.4 Outputs

1. If the user inputs the incorrect information, it will prompt them to try again with a different CAPTCHA question.
2. If the user inputs the correct information, let them proceed to checkout.

3.2.3.5 Error Handling

1. If the user experiences an issue with the CAPTCHA system, such as being unable to correctly input the information, they may seek the feedback option, which will be available at the bottom of the page at every step of the ticketing system process.

3.2.2.4 Interface with Database

3.2.4.1 Introduction

The software is going to require a database that is connected to the user interface in an attempt to combat any type of interference between clients.

3.2.4.2 Inputs

1. The software will prompt the user to set up an account and take these inputs:
    a. Name
    b. Email

    c. Billing information (optional)
    d. Membership (optional)
2. Take input of the movie that the user would like to watch.
3. Take input of the time.
4. Take input of the quality.
5. Take input of the theater if it is open.
6. Take the input of the seat inside the theater.

3.2.4.3 Processing
1. Make sure the billing process is accurate.
2. Use boolean values (true or false).
3. If the seat is open, allow the user to click on it and confirm.
4. Once confirmed, send that seat to the database and set it equal to false.
5. If all the seats in the theater are equal to false, meaning not available, close that theater.

3.2.4.4 Outputs
1. When confirmed, it should print a receipt for the user.

3.2.4.5 Error Handling
1. Any errors must be repaired by the administrators.

3.2.5 Movie Review

3.2.5.1 Introduction
The Movie Review software will retrieve the rating of the specific film and report the critics' reviews to the user.
3.2.5.2 Input
1. The software will prompt the user to choose a movie.
2. The software will then allow the user to choose whether to purchase the ticket or to view ratings for the movie.

3.2.5.3 Processing
1. This will require an API that would connect administrators to a data system that holds ratings for movies.
    a. The two movie rating websites that the software would need to access are Rotten Tomatoes and/or IMDB.

3.2.5.4 Output
1. The software would display random user ratings and critic reviews from Rotten Tomatoes/IMDB.

3.2.5.5 Error Handling
2. If there is an error within the Movie Review feature, complaints can be sent through the Feedback System which will be reviewed by the administrators.

3.3 Use Cases

The actor should be able to see only parts of the overall structure. One thing that is quintessential is that the actor should be able to interact with the interface easily but not go into the database.

3.3.1 Use Case #1

Use Case #1 should be the actor being able to set up an account with the system on the main interface and it would ask the actor, in this case the movie-goer to fill out information about themselves. The actor must put in their name and age and would be given the option for other information, such as billing information and email address.

3.3.2 Use Case #2

Use Case #2 should allow the actor to choose a certain movie the actor should be able to be old enough to watch the movie hence it would ask them their age in Use Case #1, the actor after selecting a movie will be given the option to select a time and the quality before any of that, however, there needs to be some type of function put in place that shows all the people who have already selected seating for the certain movie. This is to see if there is any additional room moreover if the actor would like the seats there. All that should be stored in a database.

3.3.3 Use Case #3

Use case #3 should be centered around payment. Now, the actor could choose to pay with cash, in this sense this would be done manually and at the front. However, if the actor wishes to pay with a card they would be given the option to submit all of the billing information or do it through Bitcoin and Paypal. This information would not be stored and as soon as the information is passed, the information would be deleted and a receipt would be printed.
…

3.4 Classes / Objects

3.4.1 <Class / Object #1>

3.4.1.1 Attributes
3.4.1.2 Functions
<Reference to functional requirements and/or use cases>

3.4.2 <Class / Object #2>

…

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

For the overall power, it will require a 500Mhz processor followed by 256 MB of RAM and 1.5 GB of hard drive space. Most of the functions will be done through the cloud.

3.5.2 Reliability

The electronic ticketing system will be connected to a storage database with a substitute backup system in case of failure which will be able to recover saved information.

3.5.3 Availability

The software will be on all browsers, so any internet browser that a computer can run, additionally there will be a kiosk machine with similar software located at the theaters.

3.5.4 Security

The security system needs to be designed in which it assures customers that there will be no data breach, this is more important than anything else because it is going to be handling transactions.

3.5.5 Maintainability

The domain name must be registered, so the administrators would have to contact a cloud service provider and build it on top of that.

3.5.6 Portability

The software is going to be on a computer, so portability should not be an issue. The system will be designed so that 1,500 people can all access it at the same time.

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

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

3.8 Logical Database Requirements

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

3.9 Other Requirements

Catchall section for any additional requirements.

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

4.1 Sequence Diagrams

4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

5. Change Management Process

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

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

6. Software Design Specification

6.1 System Description

The Ticketing System utilizes various classes, methods, and functions to achieve the client's goals for their software. Through the use of diagrams, the Software Design Specification offers an outline and description of the system for software engineers to interpret into code.

6.2 Software Architecture Overview

1. Class: Location – Bundles in movies being shown and operating hours.
    a. List_Of_Movies is an Array that holds the string value of the names of the movies.
    b. HoursOpen stores an int value giving us the operating hours
        i. Employees should be able to insert/delete and change everything in the scope.
2. Class: EmployeeInfo – Bundles in the data that the employer would have access to.
    a. Employees would give their name as a string value if the value is correct then it would grant access to the operation.

        i. The employee would be able to change the functions in the location class. They'd be able to change the time and movies.
3. Class: Movie – Bundles up all the information about the movies and seats.

    a. There would be the movie name, movie time, and movie date. This would all be values.
        i. It will get the name of the movie.
        ii. It will get the index of the seat.
        iii. It will have a function to see if the seat is open or empty.
4. Class: Customer – This would bundle all the information about the customer.
    a. The information that will be taken in are the username, email address, credit card information of the user, and account balance.
        i. The functions will allow the user to register.
        ii. It will also give users the option to login and it will take them to the next page.
5. Class: Administrator – This class grants permissions to employees and customers.
    a. This allows employees access to change the name of the movie, seating, and location.

6. Class: Tickets - The ticket class allows the user to gather all of the information needed: from seating number, ticket price, and movie availability.
    a. The seating number function gives the user the information of where inside the theater.

        i. Let the user know where to sit.

    b. The price function gives the user the total ticket price.

        i. The user will know the cost.

    c. Allows the user to know which movie they purchased to watch.

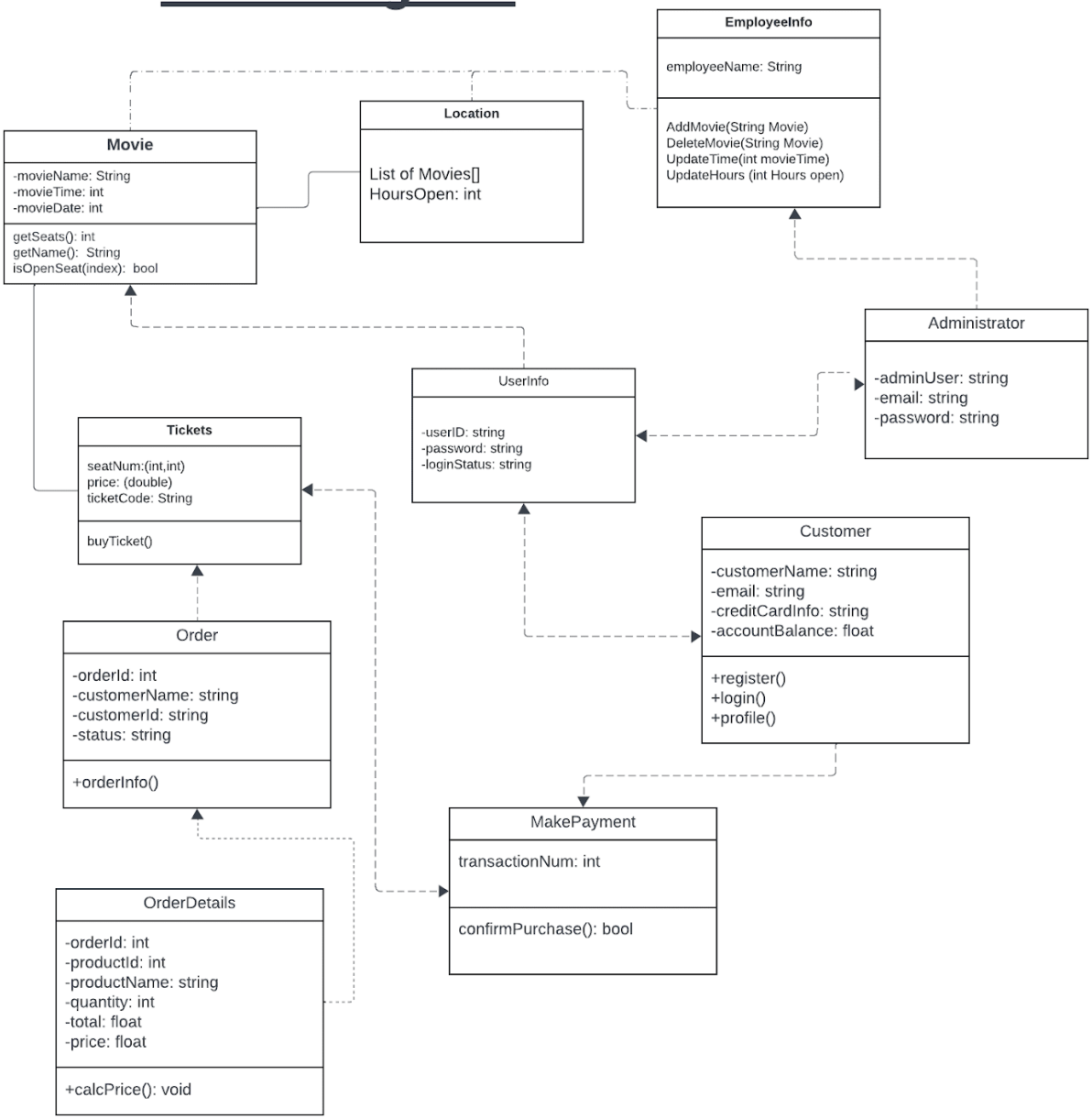
        i. Will inform the user of the movie.

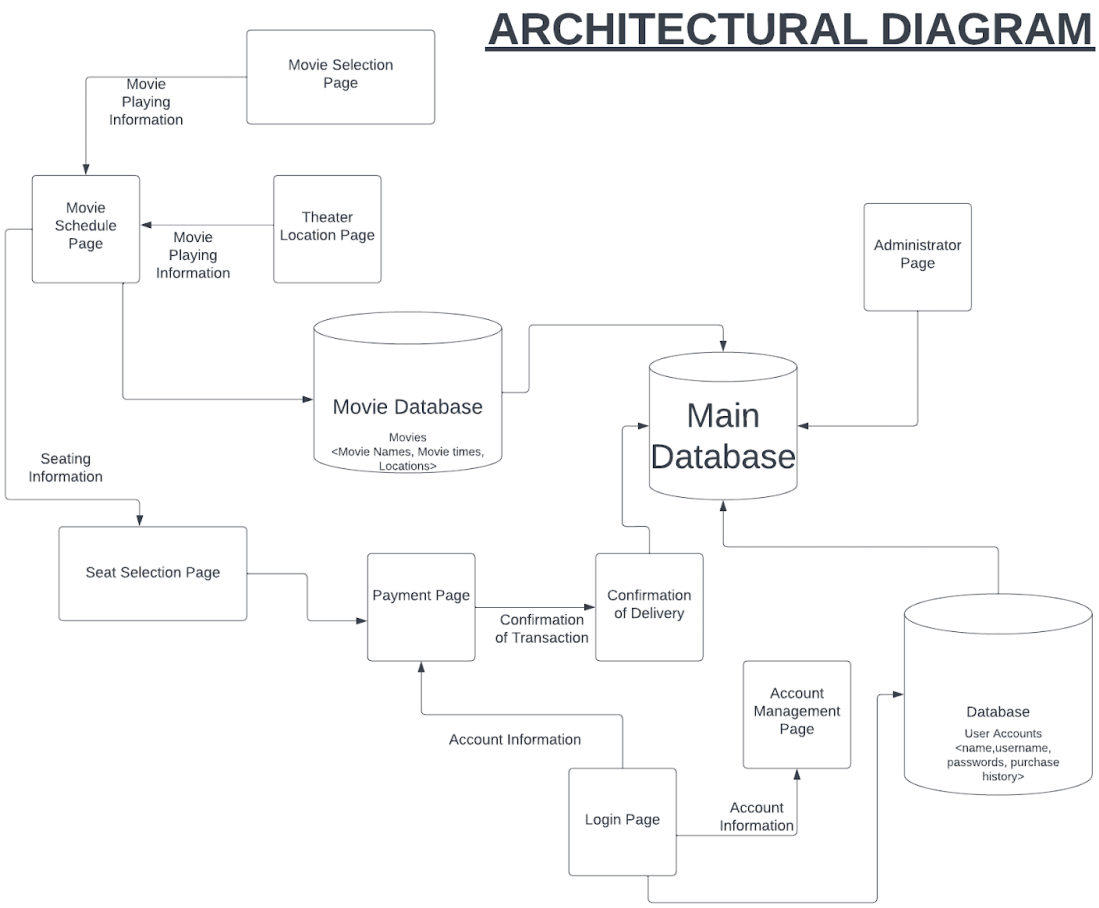
7. Class: UserInfo – This class provides employees with specific user information including ID, password, and login status.
    a. This class works with the Customer class to implement additional user login and status details.
8. Class: MakePayment – Combines all the information to handle the payment.
    a. It is going to assign a number for the transaction that should work as a receipt.
        i. Confirm purchase will just be a boolean value that passes in and gives customers a receipt.

9. Class: The Order class allows the user to receive unique login information based on their: ID number, name, and status.
    a. This is going to take the customer information, ID, and status.
        i. This will be a function that gives all of the information about what was selected that would be imputed to the user.

10. Class: OrderDetails – This class provides administrators and employees with the order ID, product ID, product name, quantity, price, and total price.
    a. calcPrice(): This method will calculate and return the total price of the product.

UML Diagram


EmployeeInfo
employeeName: String
AddMovie(String Movie)
DeleteMovie(String Movie)
UpdateTime(int movieTime)
UpdateHours (int Hours open)
Location
List of Movies[]
HoursOpen: int
Movie
-movieName: String
-movieTime: int
-movieDate: int
getSeats(): int
getName(): String
isOpenSeat(index): bool
Administrator
-adminUser: string
-email: string
-password: string
UserInfo
-userID: string
-password: string
-loginStatus: string
Tickets
seatNum:(int,int)
price: (double)
ticketCode: String
buyTicket()
Customer
-customerName: string
-email: string
-creditCardInfo: string
-accountBalance: float
+register()
+login()
+profile()
Order
-orderId: int
-customerName: string
-customerId: string
-status: string
+orderInfo()
MakePayment
transactionNum: int
confirmPurchase(): bool
OrderDetails
-orderId: int
-productId: int
-productName: string
-quantity: int
-total: float
-price: float
+calcPrice(): void



Development plan and timeline

6.3 Partitioning of tasks

- UML Diagram: Entails the planning out of various classes and functions revolving around user information, employee information, administration, the payment process, and ticketing.
- Architectural Diagram: Map out all of the major components of the system.
- Overview: Describe classes, attributes, and operations involved in the Ticketing System's SDS.

6.4 Team member responsibilities:

UML Diagram	Architectural Diagram	Overview & System

		Description
Ronessa Mose Swaraj Khatri David Prieto Samantha Georges	Ronessa Mose Swaraj Khatri David Prieto	Swaraj Khatri Samantha Georges David Prieto Ronessa Mose