Software Development
Software Requirements Specification

Version: 4.0

Nov. 2 2023

Group 14
Ryo Taono
Jorge Perez
Kane Cruz-Walker

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

Revision History

Date	Description	Author	Comments
<9-21-23>	<Version 1>	<Ryo, Jorge, Kane>	<First Revision>
<10-5-23>	<Version 2>	<Ryo, Jorge, Kane>	<Second 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

This document outlines the specifications and requirements needed to build a movie ticketing system for a company that has multiple theaters. It will provide detailed information about the system features clients (ticket buyers, employees, and developers) may use, as well as provide a reference for stockholders, developers, and managers to streamline the development process and ensure transparency amongst all parties.

1.1 Purpose

The system will provide intuitive solutions to streamline the ticket sales process for the company and their customers. For the customers, this product will provide an intuitive method for them to buy movie tickets, earn rewards for their purchases, view live updates of seatings, and movie information when it comes to enjoying the cinema. For the company this product will provide a low maintenance way of managing, tracking, and updating relative data in regards to current/incoming movies, various ticket prices depending on age, membership, and promotion, movie theater availability, and general services required for a consumer service.

1.2 Scope

This software, Ticket Management System (TMS), is designed for both online and in-person for Regal Cinemas, for the purpose of easing the purchase and redemption of movie tickets. The software will allow customers to purchase movie tickets, view movie times, available seats, and anything else we can think to add. We are hoping to provide an easy and quick user experience that will maximize user happiness and profits. Although we mentioned that examining the data collected can be used to maximize company's revenue, wrangling the data is beyond our scope.

- Sept 21 (Requirements Specifications)
- Oct 5 (Software Design Specification)
- Oct 19 (SDS: Test Plan)
- Nov 2 (Architecture Design w/ Data Mgmt.)
- Nov 9 (Security Quiz)
- Nov 30 (The Importance of Ethics)
- Nov 30 (Project Report)
- Dec 7 (Project Report Peer Review)

We estimate the total cost to be $100k

☐ Smooth buying tickets through the display of live seat availability

- ☐ Monetary troubles will be managed effectively by organizing purchase history and every purchase has the primary key to identify
- ☐ Advertise members effectively based on their input information (when ticket buyers create an account, the system will also ask their interest in the type of movies.)

Contacts:

- Ryo Taono:XXX-XXX-XXXX
- Jorge Perez:XXX-XXX-XXXX
- Kane Cruz-Walker:XXX-XXX-XXXX

1.3 Definitions, Acronyms, and Abbreviations

DBMS: database management system
SRS: Software Requirement Specification
TMS: Ticket Management System
PK: Primary Key
FK: Foreign Key
CRUD: Create, Remove, Update, Delete
DB: Database

1.4 References

1. IEEE Recommended Practice for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society, 06/25/1998
2. E-Store Project Software Requirements Specification Version <4.0>,
3. Software Engineering Software Requirements Specification (SRS) Document, Shock Force Software Team, 10/20/2009
4. Use Case, Massimo Felici, School of Informatics, 2004-2011

1.5 Overview

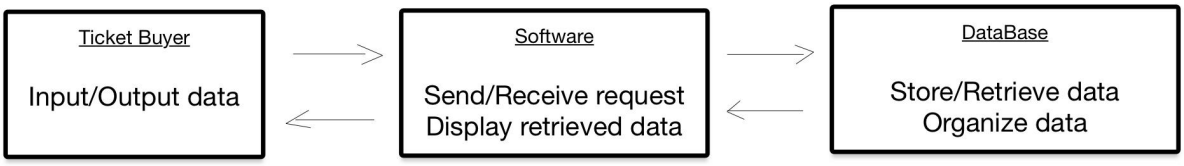
Following sections will contain a general description of TMS, its various functions, specific requirements, and user interaction. Section 2 will be the general description, Section 3 will include the specific requirements, Section 4 will include different models used to elaborate on specific requirements, and Section 5 will cover any potential updates and changes to the system.

2. General Description

This section will contain the specifics of the TMS. Its features, system requirements, and how it will benefit the company and enhance customer experience. It will describe the major features available for patrons of the theater to create a better viewing experience.

2.1 Product Perspective

- Ticket buyers expect to view seat availability, prices, movie features.
- The company expects to store/retrieve data via DBMS such as MySQL, Microsoft SQL…
- Display the insights of sales info from various perspectives (movie features, time, age,...)



2.1.1 User Interfaces

- ☐ Web Browser
- ☐ Mobile Application

2.1.2 Hardware Interfaces

- ☐ Computer
- ☐ Smartphone
- ☐ Ticketing machine (in-person)

2.1.3 Software Interfaces

- ☐ Operating system: We will be using Windows operating system for its ease of use and accessibility to new users.
- ☐ Database: To save movie details e.g. movie times, ratings, durations, leading actors, etc. Will also save customer rewards information and purchases and reservations made by customers
- ☐ Visual Basic: We have chosen the Visual Basic language due to its simplicity and workability with Windows

2.1.4 Communications Interfaces

This system will support all types of web browsers. The system will use simple electronic forms for purchases, reservations, etc.

2.2 Product Functions

- ☐ Input/retrieve/remove data (customer's info, movie features, shopping transactions)
- ☐ Display current/incoming movie info, seat availability, prices, theater map
- ☐ Facilitate the interaction between employees and customers
- ☐ Allow admins/managers to access important information with their password and make a change

2.3 User Characteristics

Users of this system will be able to retrieve movie times and information for current and upcoming movies and patron payment and reward information. The system will allow patrons to: purchase reserved seats and earn points on any purchases, see movie information such as movie times, run length, or genre, create a profile that will save their personal information, and see any past purchases. The system will allow employees to find patron information in order to redeem points and verify any purchase made, verify reserved patron's seats, as well as any permissions patrons also have and the ability to see a patron's profile.

2.4 General Constraints

- Constraints include general knowledge of inputting customers info, reviews, and movie info into the database. It may require basic programming skills (SQL, Excel..?)
- The system interface will be intuitive enough that customers will be able to easily access all features
- Customer personal information will be securely stored
- Customer reward information will be stored and available to view when buying tickets
- Persistent storage will be kept for movie information, length, rating, genre, etc.
- Kiosks will be well maintained to ensure that their touchscreens are functional at all times
- Kiosks will always be up-to-date with data from the servers to ensure accurate information is presented
-

2.5 Assumptions and Dependencies

A request for a refund/cancellation on a ticket, offer of a replacement ticket in case of no available compromise.
Calculation of reward points and appropriate reward to point ratios.
Patron may not have an updated version of the application.

3. Specific Requirements

This section will describe all the details of software requirements, including external interface requirements, nonfunctional and functional requirements, and Use Cases for all the users.

3.1 External Interface Requirements

A. Name of item: Touch Screen Ticketing Machine
B. Description of purpose: To reserve a set to watch a movie
C. Source of input: Fingers!
D. destination of output: Screen
E. Relationships to other inputs/outputs: Seat availability

3.2 Functional Requirements

Ticket Buyer

- ☐ Create/Update/Delete an account
- ☐ Buy/Cancel Tickets/goods (online shopping)
- ☐ Get a free/VIP membership
- ☐ View rewards
- ☐ Log in/out (allow social media integration)
- ☐ View purchase history
- ☐ Input/Update personal info
- ☐ Request questions/call
- ☐ View theater map
- ☐ View seat availability
- ☐ View current/incoming movies and their features and prices for all cases
- ☐ Search for something

Employee

- ☐ Log in/out with EmployeeID
- ☐ View transactions
- ☐ Verify tickets at the gate
- ☐ View customer's inquiries
- ☐ Access basic customer's info

Admins/Managers

- ☐ Log in/out with Admin/ManagerID
- ☐ Access credentials
- ☐ Give refunds back
- ☐ All basic employee permissions as well

3.3 Use Cases

3.3.1 Patron Account Creation

- ☐ Users can create an individual account following these steps
  - ☐ Providing a name, phone number, email address, and password
  - ☐ The software will then create an account in the system and send a verification email
  - ☐ After verifying their email users will be able to sign in and begin using their account

3.3.2 Employee Account Creation

- [ ] Employee accounts will be created and distributed following these steps
    - [ ] Upon hiring IT will create an employee account with employees work ID, name, phone number, and email address
    - [ ] The software will then create an account in the system and send a verification email
    - [ ] After verifying their email employees will be able to sign in and begin using their account

3.3.3 Guest Movie Reservations

- [ ] Movie reservations for non members will go as follows
    - [ ] Guests will select the movie, time, and seats they wish to purchase
    - [ ] They will be asked if they are a member, would like to sign up to be, or would like to continue to purchase as a guest.
    - [ ] If they select to purchase as a guest.
    - [ ] They will then provide a name, phone number, and email to receive their tickets and receipts
    - [ ] They will then be prompted to the payment page
    - [ ] Upon paying they will receive an email with their purchase and movie tickets
    - [ ] 24 hours before their movie they will receive a reminder for the movie they have purchased tickets for

3.3.4 Member Movie Reservations

- [ ] Movie reservations for members will go as follows
    - [ ] Guests will select the movie, time, and seats they wish to purchase
    - [ ] The system will save the movie, time, and seats to add them to the users cart once they sign in
    - [ ] The user will be asked to sign into their account via their credentials and be returned to the checkout page
    - [ ] If the user has a credit card on file they can simply click the purchase ticket
    - [ ] If the user does not have a card on file they will be prompted to the payment page
    - [ ] Upon paying they will receive an email with their purchase and movie tickets
    - [ ] Based on purchase amount the software will automatically update the members points accordingly
    - [ ] 24 hours before their movie they will receive a reminder for the movie they have purchased tickets for

3.3.5 Employee Movie Reservations

- [ ] Movie reservations for employees will go as follows
    - [ ] Employees will select the movie, time, and seats they wish to purchase

- ☐ The system will save the movie, time, and seats to add them to the users cart once they sign in
- ☐ The user will be asked to sign into their account via their credentials and be returned to the checkout page
- ☐ If the user has a credit card on file they can simply click the purchase ticket
- ☐ If the user does not have a card on file they will be prompted to the payment page
- ☐ Upon paying they will receive an email with their purchase and movie tickets
- ☐ Based on purchase amount the software will automatically update the members points accordingly
- ☐ 24 hours before their movie they will receive a reminder for the movie they have purchased tickets for

3.3.6 Subscriptions

- ☐ Upon account creation each account will be provided a free membership
  - ☐ A membership code that is not already taken should be generated and assigned to the account
- ☐ Should anyone wish to upgrade to a VIP membership the following steps should be followed
  - ☐ The user must be logged in
  - ☐ They can navigate to the membership section of their profile
  - ☐ From there they can see their current membership
  - ☐ There will be a button they can click to upgrade or downgrade membership
  - ☐ If they choose to upgrade take them to a checkout window to pay the membership fee
  - ☐ Once they pay, update their accounts membership to the appropriate status
  - ☐ If they choose to downgrade take not of when their membership term ends and set it up to downgrade when the term ends
  - ☐ Once their term ends update their account accordingly

3.3.7 Rewards

- ☐ All accounts will have a rewards tab in their user menu
- ☐ Users will be able to earn points for rewards based on how much they spend
- ☐ User can navigate to the rewards page to see rewards they have earned
- ☐ Should the theater wish to give users free promotional rewards a notification will go out to users telling them to check their rewards page to claim their free rewards

3.3.8 Login/Logout

- ☐ Users will be provided a section to login or create an account
- ☐ If a user wishes to login they provide a username and password that has already been approved and submitted to the database
- ☐ If a user wishes to create an account they must complete the onboarding procedure

- ☐ Provide first and last name, username, email, phone number, credit card to save to file (if desired)
- ☐ Users will be able to logout if they navigate to their user menu and click the log out button
- ☐ If user is buying a ticket they will be prompted to either login or continue as guest
  - ☐ If the user has an account and has a card on file the checkout process will require no further information from them
  - ☐ If the user has an account but no card on file the checkout process will require them to input payment before receiving their order
  - ☐ If the user wishes to continue as a guest, they must provide first name, last name, email, phone number, and credit card (not to be saved on file) for the one time purchase before receiving their order.

3.3.9 View purchase history

- ☐ A user will be able to track purchases they have made in the past if they purchased their tickets with their account
- ☐ Users can navigate to their account menu and find a previous purchases section
- ☐ By clicking on the previous purchases section they will all purchases with brief information
  - ☐ Date of purchase, movie, number of tickets
- ☐ They can then click on a specific item and see more details about that purchase
  - ☐ Date of purchase, movie, number of tickets, card used to make purchase

3.3.10 Input/Update Personal Information

- ☐ Upon account creation users will provide the name, email, phone number, and credit card they would like to use for their profile
- ☐ Should they wish to update their information
  - ☐ Date of purchase, movie, number of tickets, card used to make purchase

3.3.11 Request questions/calls

- ☐ Users can navigate to their profile menu and be provided with a ‘contact’ section
- ☐ Once in the contact section they will be provided with the theaters phone number or an option to start a chat with a representative

3.3.12 View Theater Map

- ☐ Users can view theater placements in 3D, 3D models will be updated in real time with every leftover soda, popcorn kernel, and any other trash left over.
- ☐ View reserved and open seats available

3.5 Non-Functional Requirements

- ☐ The maximum capacity of access at a time is 100 people.
- ☐ The last purchase can be made 10 mins after the movie starts.
- ☐ Make this software available for all platforms

- ☐ Employee will respond less than 5 mins to chat
- ☐ No more than 50 tickets can be bought at once
- ☐ Refund can be made no later than 10 mins after the movie started
- ☐ One person can only have one account (check with primary key)
- ☐ Saved credit/debit card must be checked if expired or not before approving purchase
- ☐ Notify a ticket buyer if one is buying multiple tickets and they are overlapping
- ☐ Tax on products should differ depending on the State
- ☐ Ticket prices may also differ based on State

3.5.1 Performance

Performance on the app will be based on the user's phone and its connection strength to our database as well as the phone's hardware.
In-theater kiosks will run on an Intel Core i5-13600K cpu. This will run all transactions in less than one second.
The capacity of simultaneous user is maximum 100

3.5.2 Reliability

The app/kiosk will constantly be updated allowing all movie times, prices, etc. to be accurate.
Kiosk updates will not exceed a minute per day

3.5.3 Availability

The movies and seats should be available at the reserved dates and times and should take into account both customers that have pre purchased and those buying at the kiosk.
The app will have direct database access to ensure that customers can always buy tickets or look up what movies may be available.

3.5.4 Security

- ☐ All purchase will be signed with a unique number
- ☐ Credit card info will not be displayed that it only shows last 4 digits and verify with CVV code for cookies
- ☐ All transactions will use Transport Layer Security to ensure that customer's data is completely safe and remains confidential.
- ☐ Any unfinished transactions will be canceled after five minutes of inactivity.
- ☐ Transactions will be confirmed by the customer after they have put in their payment information.
- ☐ Employees have limited access to data that Managers/Admins can access with their code.

3.5.5 Maintainability

The system will support regular updates to ensure compatibility with the latest web browser versions, bug fixes, and security patches.

3.5.6 Portability

The app will be available on ios and android for mobile devices
Kiosks will be in-theater only

<Start of Assignment 2>

3.7 Software Design Specification

3.7.1 Design Overview:

This project will create a software for a theater ticketing system, based on customer requirements as specified in the SRS document. This document will provide the streamline of internal queries between user interface and database as well as the architecture of web pages.

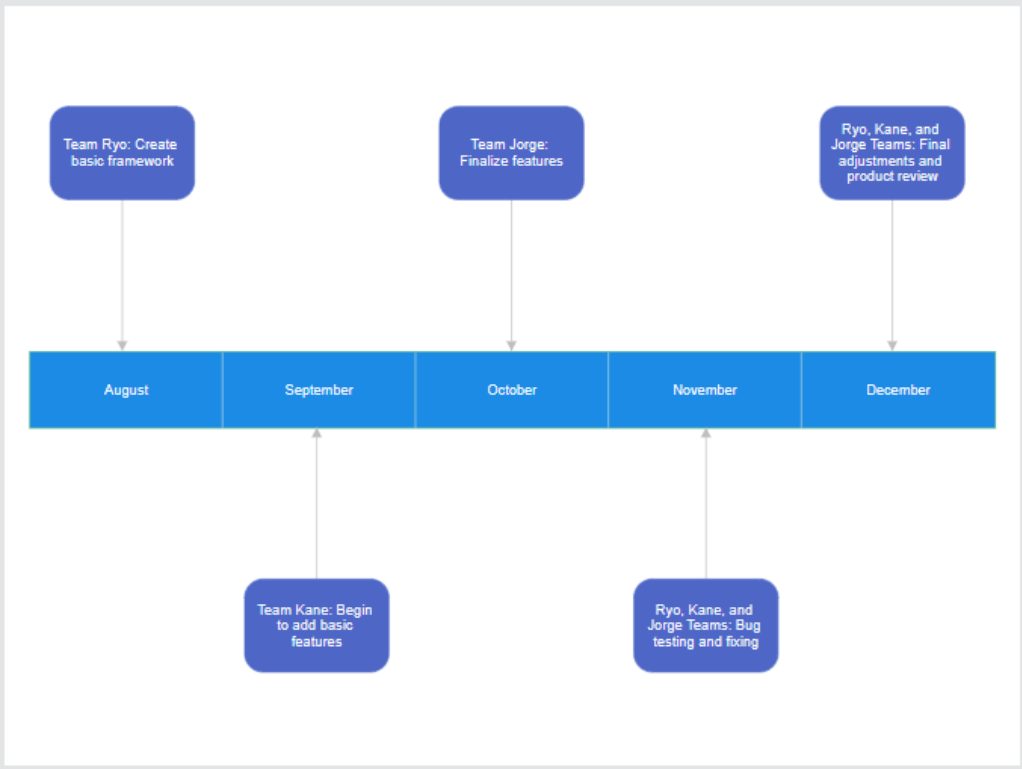
3.7.2 Object:

The purpose of this document is to clarify the model of the software to be developed that will enhance maintainability and correctness of the software. It provides an essential structure for developing the right software. This document covers the lower level abstraction.

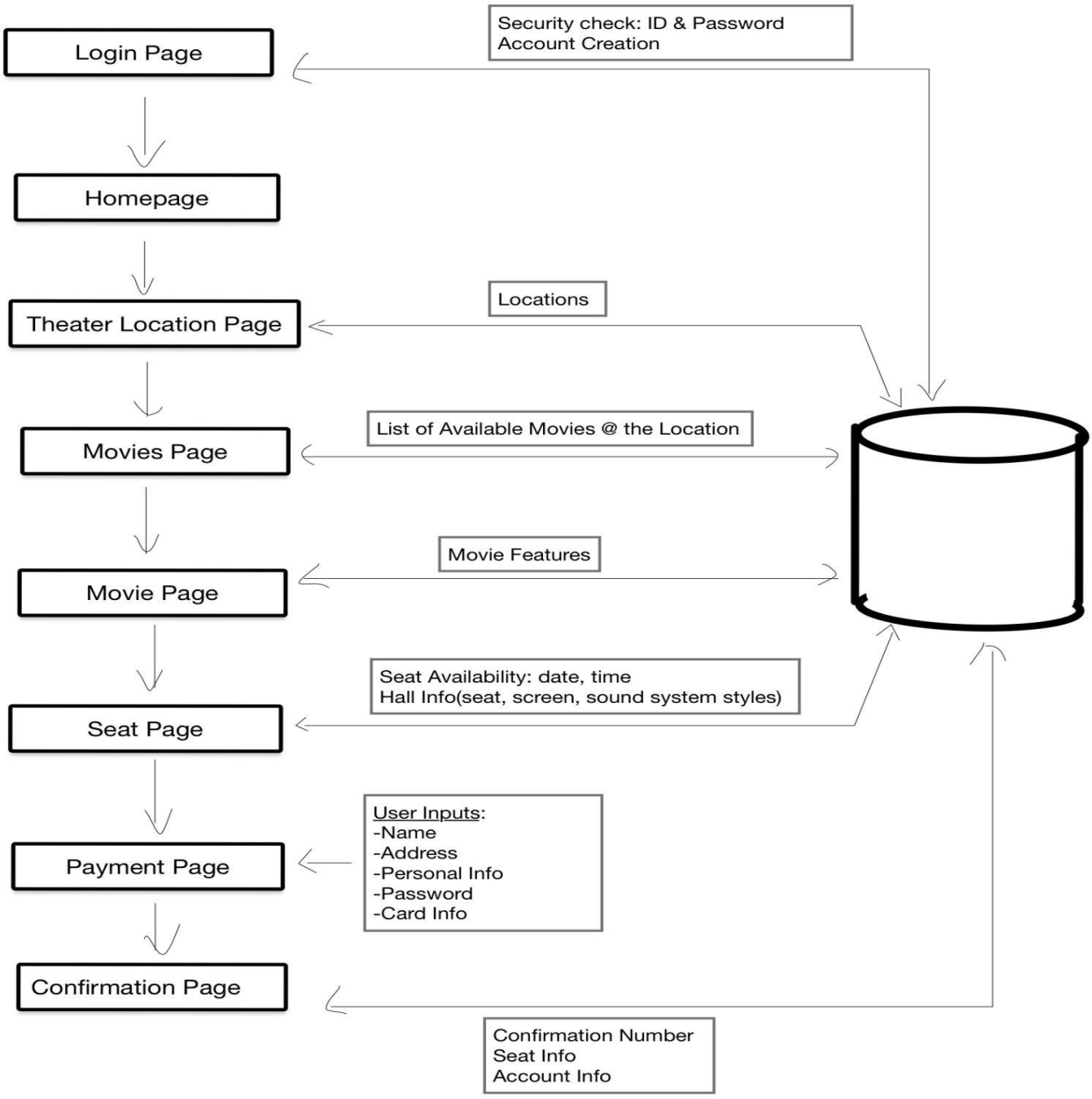
3.7.3 Development Plan:

Description: This project will be developed by multiple project managers, team leads, and teams. There will be a front end development team responsible for creating the UI, theme, and interface that users will be interacting with. There will be a back end team responsible for creating the database, and api architecture to allow for the front end to access data from the back end. Each team will have a team lead knowledgeable in their team's field that will work closely with the project managers to ensure transparency between the development process, desired outcomes, and the final project.

Timeline: Both front end and back end teams should first develop minimum viable products that demonstrate modular basic operations that will be required for use. Once the mvps have been developed while employing the desired architecture for scalability, each team should then branch into building out more sophisticated operations that will be required. The project manager and team leads will be sure to create and share a timeline with developers to ensure checkpoints are reached in a timely manner allowing for any mishaps to be addressed efficiently and with proper direction.



3.7.4 Web Page Architecture Diagram:



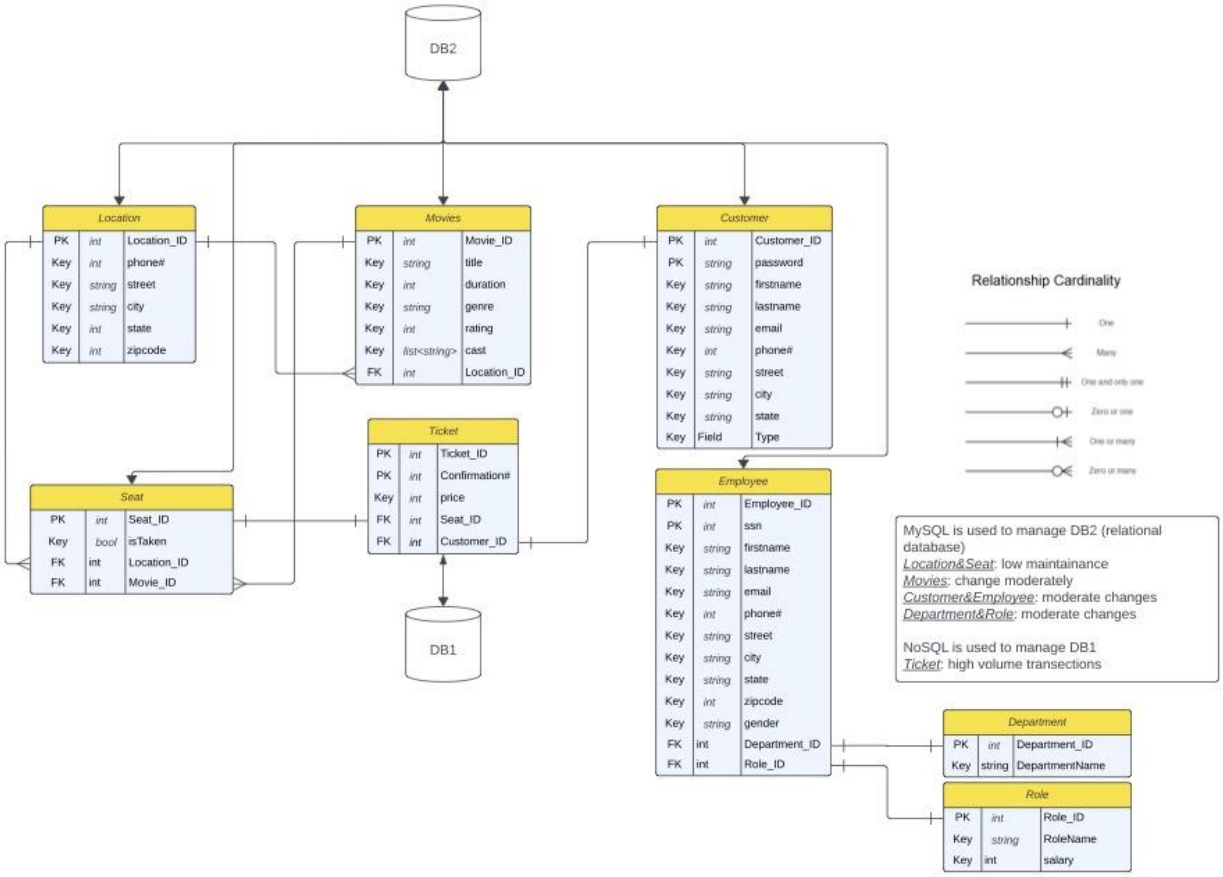
3.7.5 Description of SWA:

The above SWA diagram outlines how the components of the web application interact with our database and each other.

<Start Assignment 4>

Architecture Design with Data Management

3.7.6 Unified Modeling Language (UML) Class Diagram:



UML Update Description:

After better understanding the structure of our information, we created new classes and more representative values for them. We included new classes such as Location, Seat, Ticket, Department and Role. They all build off of previous classes we had such as Seats, Admin/Manager, TicketCounter, TheatreManagement, and Technical Staff. We also included more relevant data fields for our classes for personal and location information such as addresses and phone numbers. We realize some personal information fields will need to be hashed and secured, such as SSN values in the employee class. We also outlined the relationship between each class better by including the key of the various relationship types that can be found.

3.7.7 Overview of UML:

The above UML diagram outlines the classes that should be developed for the implementation of this project. Below you will see more details about each class, but one can study the above image to better understand how classes work together and what each class's intended purposes are.

3.7.8 Description of Classes:

Location: The class contains the details of theater
Movies: The movie class will be used to manage movie information such as details and showtimes.
Seat: Seat class will determine if a seat is available for reservation
Ticket: This class has features to allow users to reserve seats
Customer: This class will be used to give customers permissions
Employee: This class is used to store employees data and parent class of TechnicalStaff and Admin/Managers classes
Department: This class will store department information
Role: This class will store employment details

3.7.9 Description of Attributes:

Location:

- PK, int, Location_ID
- Key, int, phone#
- Key, string, street
- Key, string, city
- Key, string, state
- Key, int, zipcode

Movies:

- PK, int, Movie_ID
- Key, string, title
- Key, string, genre
- Key, int, rating
- Key, list<string>, cast
- FK, int, Location_ID

Seat:

- PK, int, Seat_ID
- Key, bool, isTaken
- FK, int, Location_ID
- FK, int, Movie_ID

Ticket:

- PK, int, Ticket_ID
- PK, int, Confirmation#
- Key, int, price
- FK, int, Seat_ID
- FK, int, Customer_ID

Employee:

- PK, int, Employee_ID
- PK, int, ssn
- Key, string, firstname
- Key, string, lastname

- Key, string, email
- Key, int, phone#
- Key, string, street
- Key, string, city
- Key, string, state
- Key, int, zipcode
- Key, string, gender
- FK, int, Department_ID
- FK, int, Role_ID

Customer:

- PK, int, Customer_ID
- PK, string, password
- Key, string, firstname
- Key, string, lastname
- Key, string, street
- Key, string, city
- Key, string, state
- Key, int, zipcode

Role:

- PK, int, Role_ID
- Key, string, RoleName
- Key, int, salary

Department:

- PK, int, Department_ID
- Key, string, DepartmentName

3.7.10 Description of Operations:

Location:

1. DisplayLocation() - Will allow users to see the theater's physical address

Movies:

1. DisplayMovie() - Will display movie information: runtime, genre, description, etc.

Seat:

1. DisplayAvailability() - Will determine if a seat has been reserved

Ticket:

1. DisplayTicket() - Displays ticket information like: movie, time, hall, seat
2. BookTicket() - Allows customers to reserve specific seats for a specific movie

Customer:

1. DisplayCustomer() - Will display customer information: name, reward points, credit card info, purchases, etc.

Employee:

1. DisplayEmp() - Will display employee information: name, position, details of work, etc.

Role:

1. DisplayRole() - This function will display the detail of all the role

Department:

1. DisplayDepartment() - This function will display the detail of all the department

3.7.11 Data Management Strategy:

Relational Database (DB2 in the UML diagram) is used for Location, Movies, Customer, Seat, Employee, Department, and Role classes as they are structured data and changes in these data are low to moderate.

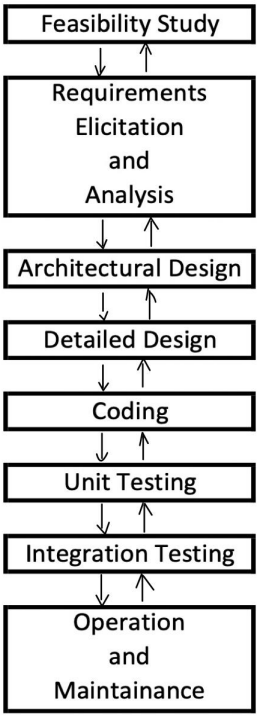
Non-relational Database (DB1 in the UML diagram) is used for ticket transactions as we expect the transaction to be high volume and the data will be used for real-time analysis.

3.7.12 Tradeoffs:

We chose to use two separate databases to be able to have a backup of our data in case a problem arises with one or the other. We also chose to implement a relational database for movie theater operations and non-relational database for purchasing, in hopes to improve the systems speed and efficiency for common tasks. We have decided to focus more on a simple rather than larger scope for our ticketing system to allow easier use for customers. The system will be easier for customers to use with a sacrifice in extra functionality. We have forgone multifactor authentication in favor of a more convenient experience for our customers. Our rewards program is point based over visit based to incentivize customers to spend more to earn more rewards.

3.7.13 Architecture Diagram:

Waterfall Model



<Start of Assignment 3>

3.8 Verification Test Plan

3.8.1 Test Plan Overview:

This test plan is meant to cover Unit, Functionality, and System tests that will be needed to ensure product quality. Unit tests will be created to ensure individual components operate properly and provide desired results. Functionality tests will be implemented to verify everything users interact with will function as desired. System tests will be used to ensure the system can complete its various functionalities while executing as intended. To see detailed unit tests please see attached document (Test Plan for Ticketing System), note that each TestCaseId identifies what type of test each one is, be it unit, functionality, or system. Our current test plan includes 4 unit tests, 5 functionality tests, and 5 systems tests.

3.8.2 Object:

Unit tests will be used to ensure all functions and methods accurately input and output desired data and complete the appropriate tasks. We have included unit tests for payment, movie availability at specific theaters, and seat availability functionality. The first four unit tests in our Test Plan document outline the specific steps for the mentioned tests. Other unit tests can be implemented to ensure employee accounts are created properly,

data changes in databases persist as intended, and any general task that manages any CRUD operations.

Functionality tests will be used to ensure complex processes execute correctly and provide the desired results. We have included functionality tests for the login procedures and payment history in our Test Plan document. When a user logs in they may provide proper credentials, have provided incorrect credentials, or have no credentials, all which have been accounted for with their unique detailed test case plans in our Test Plan document. Other functionality tests that can be implemented include tests for User Interface functionality, basic movie reservation operations such as reserving seats, receiving tickets, searching for movies, or even looking into movie details.

System tests will be used to ensure the implementation of smaller parts of the software operate accordingly once put together. In our Test Plan document we have included systems tests for system payment information, customer rewards programs, and validity of movie data in the database. Other system tests that can be implemented include load handling tests, data storage locations and speeds, proper data management and validity, and customer service functionalities.

Following IDs are excerpts from Excel Sheet “TestPlan for Ticketing System” & * implies numbers**

3.8.3 Unit Testing: Ryo will be testing all unit testing

- ☐ TC_TTS_Unit_Payment_***:
  - ☐ The purpose of this test is to ensure the timeout coding functionality of the payment page that it is expected to send the user to the homepage if the user stays on the payment page for more than 10 mins without any action.
- ☐ TC_TTS_Unit_Location_***:
  - ☐ The purpose of this test is to ensure the return values of the available movies at the selected location. The expected result should show all the available movies at the selected location.
- ☐ TC_TTS_Unit_Seating_***:
  - ☐ The purpose of this test is to ensure the return values with corresponding inputs are theater location, movie, and time & date, then, it is expected to return all the available seats.

3.8.4 Functional Testing: Kane will be in charge of Functional testing

- ☐ TC_TTS_Functional_Login_***:
  - ☐ The purpose of this test is to ensure that any account holder will be able to properly log into and access their movie theater account
- ☐ TC_TTS_Functional_PaymentHist_***:
  - ☐ This test will verify whether or not a customer’s payment was able to be accepted

3.8.5 System Testing: Jorge will be in charge of System testing

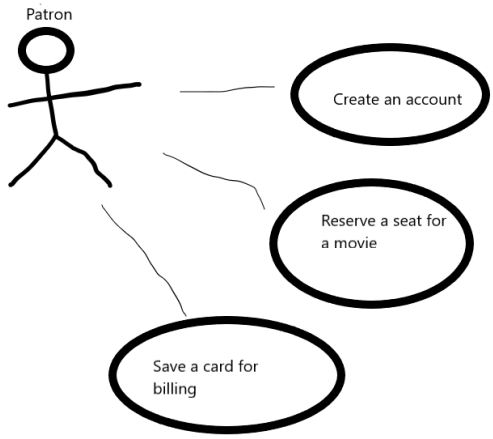
- ☐ TC_TTS_System_Payment_***:
  - ☐ The purpose of this test is to ensure that payment is able to be processed for any purchases made and that the customer receives confirmation for their purchase

- [ ] TC_TTS_System_Reward_***:
    - [ ] The purpose of this test is to ensure that customers properly receive rewards points based on their purchases and see their points total adjusted correctly
- [ ] TC_TTS_System_MovieList_***:
    - [ ] This test is designed to verify that the movie database is accurate and up-to-date with upcoming and currently playing movies.
- [ ] TC_TTS_System_MovieList_***:
    - [ ] This test is designed to test whether the database is missing a current or upcoming movie and whether it needs to be added to the database

A. Appendices

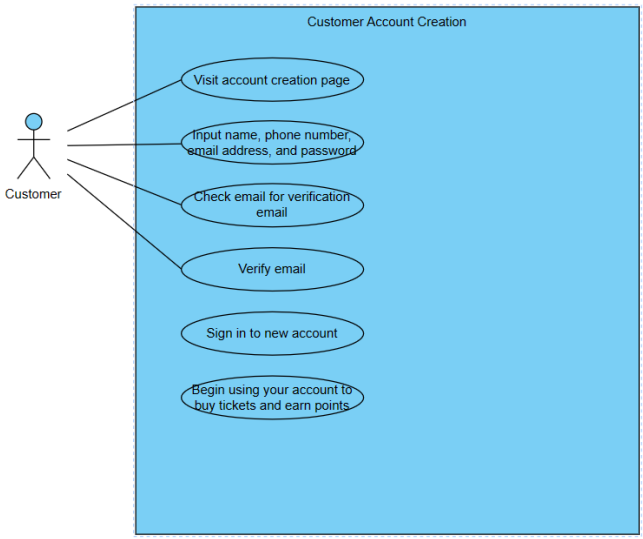
A.1 Appendix 1

Basic member reservation system



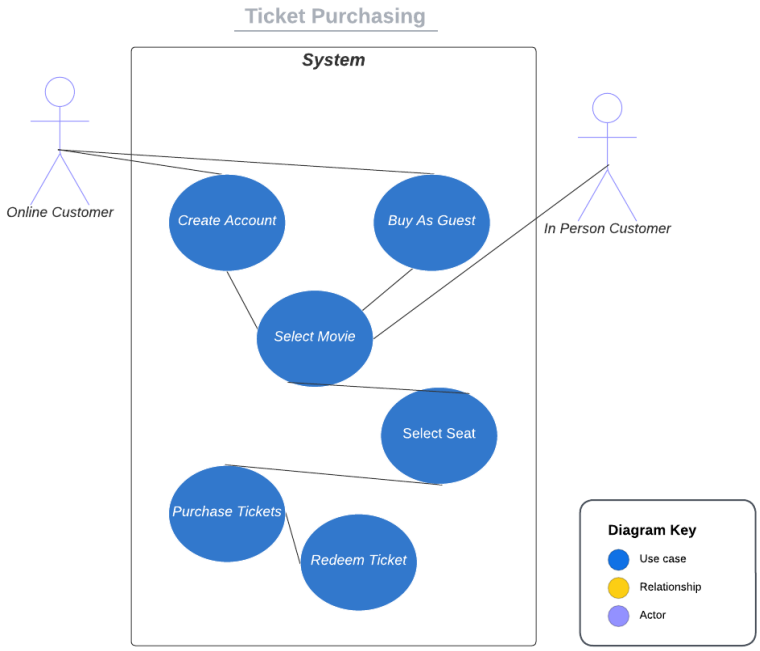
A.1 Appendix 2

Account creation diagram



A.1 Appendix 3

Movie reservation system



A.1 Appendix 4

Member account upgrade diagram