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.

Theater Ticketing

Software Requirements Specification

Version 1

2/1/2024

Group 15
Anna Krassowizki, Aidan Armas, Noam Joseph

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

Revision History

Date	Description	Author	Comments
2/15/24	Version 1	Anna, Aidan, and Noam	First Revision of the SRS Document

Document Approval

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

Signature	Printed Name	Title	Date
	Anna Krassowizki	Software Engineer	2/15/24
	Aidan Armas	Software Engineer	2/15/24
	Noam Joseph	Software Engineer	2/15/24
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of the SRS is to explain the software application that is being developed and the different requirements and constraints that are to be built within the mobile movie ticketing app.

1.2 Scope

Theater Ticketing will be an app software that will be sold to different theater companies. This software will allow these companies to input their databases into our software, in which they provide their theaters in a given area, and all the information relevant to that theater. Our users will be able to use this software to locate the nearest theater of that brand, and book tickets for a movie of their choice with multiple amenities, such as pre-reserved concessions, outside food vouchers, and other features that make our application better than our competitors.

1.3 Definitions, Acronyms, and Abbreviations

User - A person who will be operating the application. This includes the person on the consumer end as well as the employees and administrators that will be servicing the application

Database - The storing and management system that will need to be maintained and added to by the purchasing business.

1.4 References

1. IEEE Recommended Practice for Software Requirements Specifications, 20 October 1998, Software Engineering Standards Committee

2. E-Store Project Software Requirements Specification Version <4.0>, 15 April 2007, Marvel Electronics and Home Entertainment

1.5 Overview

The rest of this document provides insight into how the application will function including its many benefits. This application will provide an innovative and improved movie theater experience. The next section will include constraints and detailed functions as well as hardware and software requirements necessary for the user.

2. General Description

2.1 Product Perspective

This software system provides Theater Ticketing along with other services to enhance the movie watching experience. It will be an application that has compatibility with both the Apple and Android Operating Systems.

2.2 Product Functions

This application will allow you to purchase a theater ticket with features that include scheduling concessions pickup and allowing purchase of vouchers to bring outside food into the theater. After any purchase on our application, the user will receive a confirmation email which includes their receipt and the QR codes to their tickets and vouchers. The application will also include every theater that is owned by the company in the area of the user, and provide ratings for each theater along with the distance the theater is from the user's location/zip code.

Internet connection will be required to use the application, but after purchasing tickets users will have the ability to enter tickets and vouchers into their Apple / Google wallet, letting users avoid needing internet connection thereafter. Users will also need a digital form of payment that aligns with the ones allowed on the website. These will include Debit and Credit cards, bank account, Venmo, PayPal, Apple Pay, Google Pay, or Cryptocurrency. The user will also be required to enter an existing email address when making an account, so that after any purchase on our application, the user can receive a confirmation email which includes their receipt and the QR codes to their tickets and vouchers.

2.3 User Characteristics

This product is meant for general public use and is to be simple enough for users aged 13 and above to use. Users wishing to use the application need a basic knowledge of English or Spanish. Because our product will be built for Apple and Android products, the user is required to own a device that is compatible with these systems and has a stable connection to the internet. Other devices like Nokia will not be compatible with our application.

2.4 General Constraints

The constraints the developers will face is implementation of our applications compatible for both Apple and Android products. The application is created using JavaScript to allow for easy implementation. This product is meant for use in the United States, posing a constraint for developers in the regions to be implemented. This constraint must be created to avoid payment troubles along with language barriers for the user. This application will have a life cycle of 6 months posing a time constraint on the development and engineering teams. The stakeholders are looking for this application to be done by the end of the year, making 6 months the perfect amount of time for revisions. The application will also be using a security API to promise safe transactions. The database that will be used to store the saved payment methods, will have to have additional securities to resist security breaches. The first iteration of the program should allow for a strain of 5,000 users at a time, to allow for quick and speedy processing for users.

2.5 Assumptions and Dependencies

An assumption made is that JavaScript will be implementable on both Apple and Android's operating systems. We have also made the assumption that our beginning clientele will be under 5,000 at any point, and the application is capable of handling this many users at once. We have also assumed that we will be giving the brands that buy our software administrative permissions, so they can implement their own databases whenever necessary.

This application will depend on internet availability and secure databases to store payment information of users. The application will also need the user to share their location or zip code, and trust their payment information in a secure and safe payment system.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Our application will implement a GUI to give the user many different options and abilities to allow filtering and specific option selection. The users and administrators will have no access to API, however will have a different option of GUI depending on their given role.

3.1.2 Hardware Interfaces

The system will allow for compatibility and access for Apple and Android products. The application will be stored on the hard drive of the user's device(either the solid-state drive or hard disk drive).

3.1.3 Software Interfaces

The application will only be compatible on the Apple and Android operating systems. The database will be exported/imported to and from Microsoft Excel. This will be the primary system used for the database administrator for management. The excel document as well as the database will communicate to make sure all data is accurate and up to date.

3.1.4 Communications Interfaces

The application will be dealing with bank accounts and different payment methods. Because of this, the application will communicate with the internet through Hypertext Transfer Protocol, to ensure a secure protocol.

3.2 Functional Requirements

3.2.1 Online Ticket Purchasing and Seat Reservation

3.2.1.1 Introduction

This function is vital in buying tickets and reserving seats in the selected movie theater.

3.2.1.2 Inputs

Inputs include the filtering of movie theaters by things like location, ratings, and brand. Furthermore, movies are filtered by genre, length, and ratings. After selecting a movie, users will

need to input the type of ticket they will be purchasing(child, adult, senior), the quantity, and the location of the seat(s).

3.2.1.3 Processing

The system will process reservation by inputting the selection into the database and making this selection invalid for any future users. This will also input the given data to the movie theater's system and give them a type of ticket and amount of total tickets bought.

3.2.1.4 Outputs

The user is emailed a QR code that is connected to their account which contains their ticket, as well as given a receipt and proof of payment. Vouchers and any other concessions will also be included in the QR code. These tickets in the email can be transferred to their Apple Wallet or Google Wallet for ease of access for users. The seat is now unavailable to other users.

3.2.1.5 Error Handling

To avoid the error of a double purchase, after selecting seats and continuing to the payment process, the database will temporarily reserve the seat for 5 minutes to not allow another user to attempt to purchase the same seat. The ticket purchasing system for each movie will close 5 minutes after the local screening time has begun to stop potential confusion and error resulting in refunding.

3.2.2 User Login Feature

3.2.2.1 Introduction

As the user opens the app a screen with two options will appear. One of the options will be to log in into an existing account, the other one will be making a new account and signing up for the app.

3.2.2.2 Inputs

If the user has an existing account they will enter their email address and the password if not the app will navigate the user to a new page to make an account. On the new page the app will ask the user to enter personal information such as their name, email address and zip code.

3.2.2.3 Processing

If the user entered their login information correctly it will lead the user to the main page of our application. When the user has finished entering their personal information while making a new account it will keep their information in the system and will navigate them to the purchase ticket process after successfully creating an account.

3.2.2.5 Error Handling

If the app does not recognize the email address or password for an existing account the page will notify the user. The app will not specify which information was inserted incorrectly to promise security. The user will have the ability to reset their password using the email address they used while creating the account.

3.2.3 Safe and Reliable Payment Processing and Transactions

3.2.3.1 Introduction

This function allows the user to provide the application with multiple payment methods, which they can use during checkout.

3.2.3.2 Inputs

Inputs for this feature will include the user selecting a payment method to input, and these payment methods consist of paying by card, bank account, paypal, etc. The user is encouraged to select at least one payment method to provide the application with. If they choose not to do so, during checkout they will need to provide us with information for any of our payment methods.. Whenever the user has selected the things they would like to purchase, they can go to checkout, choose the payment method that they want to use, and pay for the products with that payment method. If the user has any of the payment methods set up beforehand, then they can simply select the payment method and pay directly, otherwise they will need to fill out their information on the spot.

3.2.3.3 Processing

Our system will take the inputs and store the payment information in a secure database, if the user decides to save the payment method on our application. If they choose to do a one-time payment, their information will not be saved, and just be used for the one-time transaction.

3.2.3.4 Outputs

When the user saves a payment method onto our application, they will receive an email that asks them to verify they have added this payment method, and once verified the payment method will be secured in the application's database. Whenever they use a payment method, whether it is for a one time payment or a pre-saved method, they will also receive a confirmation email with their receipt and confirmation their transaction went through.

3.2.3.5 Error Handling

If the user provides inaccurate payment information, the screen will reload and erase all information, for security purposes. If the user attempts to connect a specific payment method more than 5 times and is unsuccessful, they will be locked out of the payment system for 15 minutes, and then they can attempt to connect it again.

3.2.4 Easy to Schedule Concession Pickup

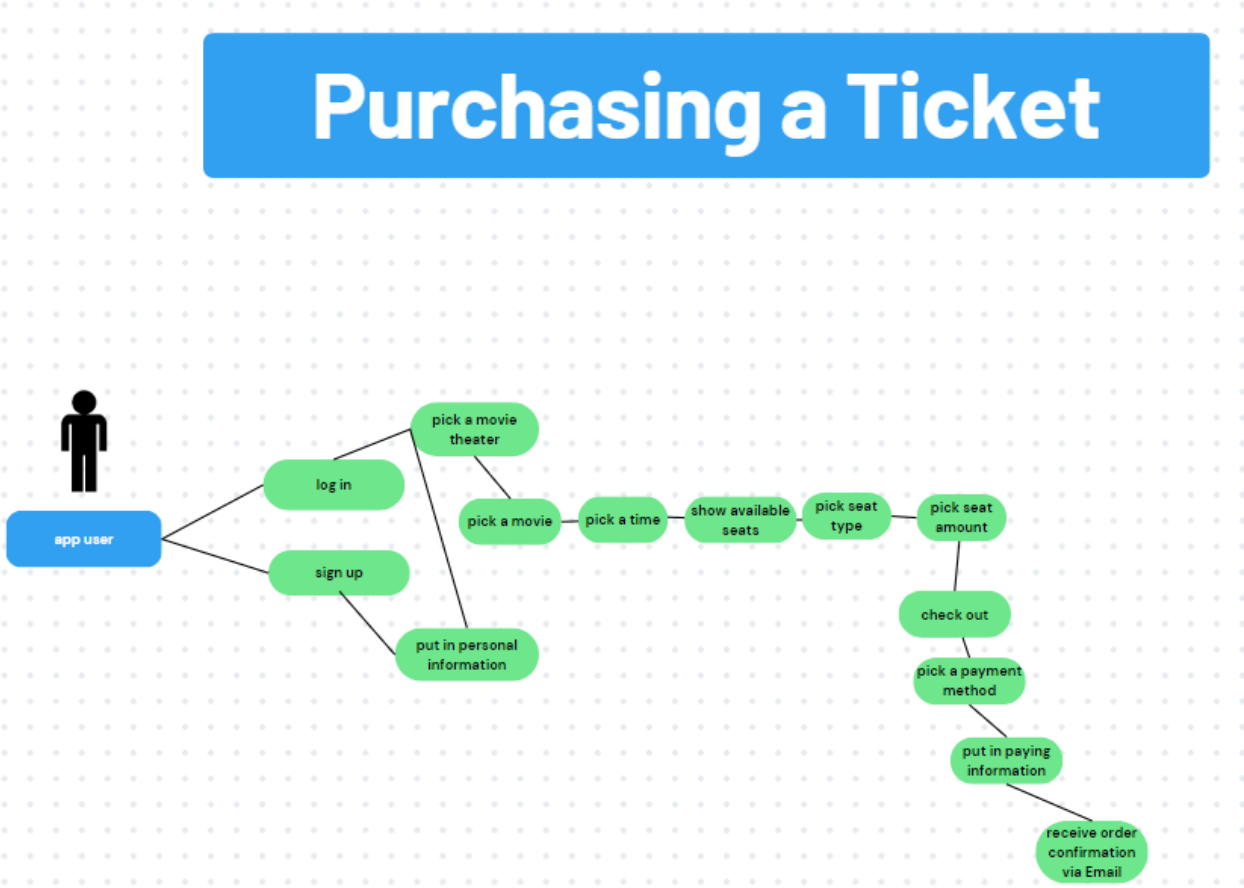
3.2.5 Ability to Buy Outside Food/Concession Vouchers

3.2.6 English and Spanish Localization

3.2.7 Allow our Customers Administrator Permissions

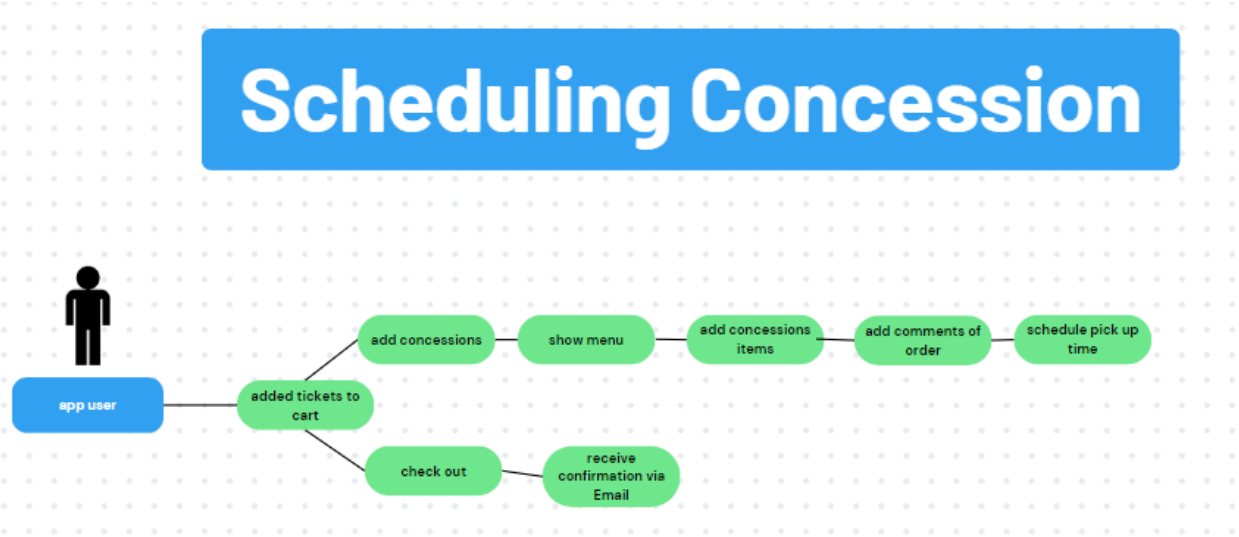
3.2.8 Easy to Use User Location Sharing

3.3 Use Cases



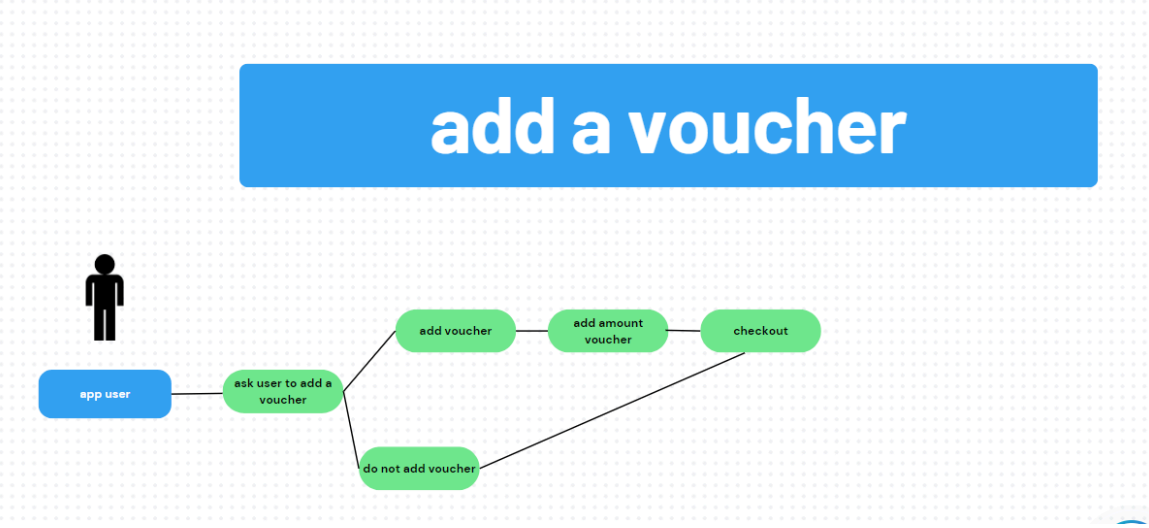
First the user login into their account if the user does not have an account the user will enter personal information to create an account. Then the app will present all the movie theaters that are near his area. After the user has picked a movie theater they will pick the movie they are interested in watching along with seats and the type of seat. Once the user picks the tickets the app will navigate the user to checkout and will ask the user for an electronic payment method. When the payment was done successfully the user will receive a confirmation email of their purchase.

3.3.2 Use Case #2



This diagram shows the procedure of scheduling a concession pick up. The user can schedule a concession pickup once they have picked their movie tickets. If the user chooses not to add a concession the next page will lead the user to checkout. If a user is interested in adding concessions the app will show a menu of all the concession options available. The user will add the concessions, the amount of each item and will add comments on the order if necessary. Lastly the user will pick a time for the pick up of the concession.

3.3.3 Use Case #3



An additional feature our app will perform is the ability to purchase a voucher to have the ability to bring outside food into the theater. First the app will ask the user if they would like to add a voucher if not they will continue to the checkout page. If yes the user will enter the amount of vouchers and will proceed to checkout.

…

3.4 Classes / Objects

3.4.1 Movie Ticket

3.4.1.1 Attributes

Attributes for the movie ticket will include ticket type (child, adult, senior), seat number, the ticket holder's name, the movie showing, theater number.

3.4.1.2 Functions

There will be a QR code on the ticket, for the theater staff to scan which allows the user to enter the theater. Also allows the user to locate their theater, and the seat number.

3.4.2 Food Voucher

3.4.2.1 Attributes

The food voucher will include a QR code that is scannable when the customer walks into the theater. Will also include the movie this is purchased for.

3.4.2.2 Functions

The QR code will be scanned on the way into the theater, and allows the customer to bring outside food into the theater.

3.4.3 User Login

3.4.3.1 Attributes

The user login will consist of their first and last name, email address, their password, phone number, and age (to confirm they are 13 or older), and 3 security questions.

4.3.3.2 Functions

This information will be stored in a secure database, and only accessible to the user if they request it. Access to information will require the user to enter their password and answer 2/3 of their security questions correctly.

3.5 Non-Functional Requirements

3.5.1 Performance

This application will be adequately optimized to support wait times for users on stable internet connection to be under 5 seconds. For further information about how "adequate" the internet will be known, refer to definitions in Section 1.3. Because payment processing will be used, the application will aim to limit processing time to under 5 seconds. This performance aim is also codependent on the banking companies systems as well.

3.5.2 Reliability

As stated before, the first iterations of the application will be able to handle 5000 concurrent users at any moment. This reliability includes scrolling through queries, as well as during the payment processing. This application also aims to limit maintenance to once every 2 months, with a guarantee of downtime that is limited to 2 hours.

3.5.3 Availability

The system will be stored on multiple servers making downtime as minimal of a threat as possible, as one server closing down will only cause for less users at a concurrent time. The other servers will take on the users from the down server until maintenance is completed.

3.5.4 Security

Users will be given the option to save their payment method. To protect this from data breaches and hackers, sensitive data will be sent to a database center and disconnected from the online database. If the user fails to correctly input payment information more than 5 times, they will be locked out of the payment page for 10 minutes, and they can try again afterwards. Once the application detects an active user that has previously saved sensitive data, it will call back the payment method for temporary use until the user is inactive once again. (Other security measures include stopping hackers from achieving ADMIN status and manipulating website properties.)

3.5.5 Maintainability

Our development team will be maintaining the application for the first 3 months after the developmental life-cycle. In this process the code will be optimized and further improved. One key point in the maintainability of the software is the use of roles and permissions. Base employee's will be given the employee role while Administration as defined by the company of use will be given. Staff with the administration role will have access to advanced permissions like (add)

3.5.6 Portability

The application is to be implemented and usable on Android and Apple products. Developers will not be keeping portability in mind during the development process and instead focus on the optimization of use on the two operating systems to be sustained. This heavily limits the client base and portability, however was asked for by the stakeholders.

3.6 Inverse Requirements

The application won't work on operating systems that are not Apple or Android.The app also does not ask for verification of age. The application does not have communication interfaces that allow for printing. Our application does not set a limit for how much outside food can be brought in using the app’s outside food voucher.

3.7 Design Constraints

The constraints of our design and application are that it will only work on Apple and Android operating systems. It will be developed using javascript, and there are no memory requirements. It needs a stable internet connection to be used, and the user should be able to use the application with ease, so nothing should be too complicated to understand. The design should be kept as simple as possible and include comments occasionally throughout for future improvements and debugging.

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.

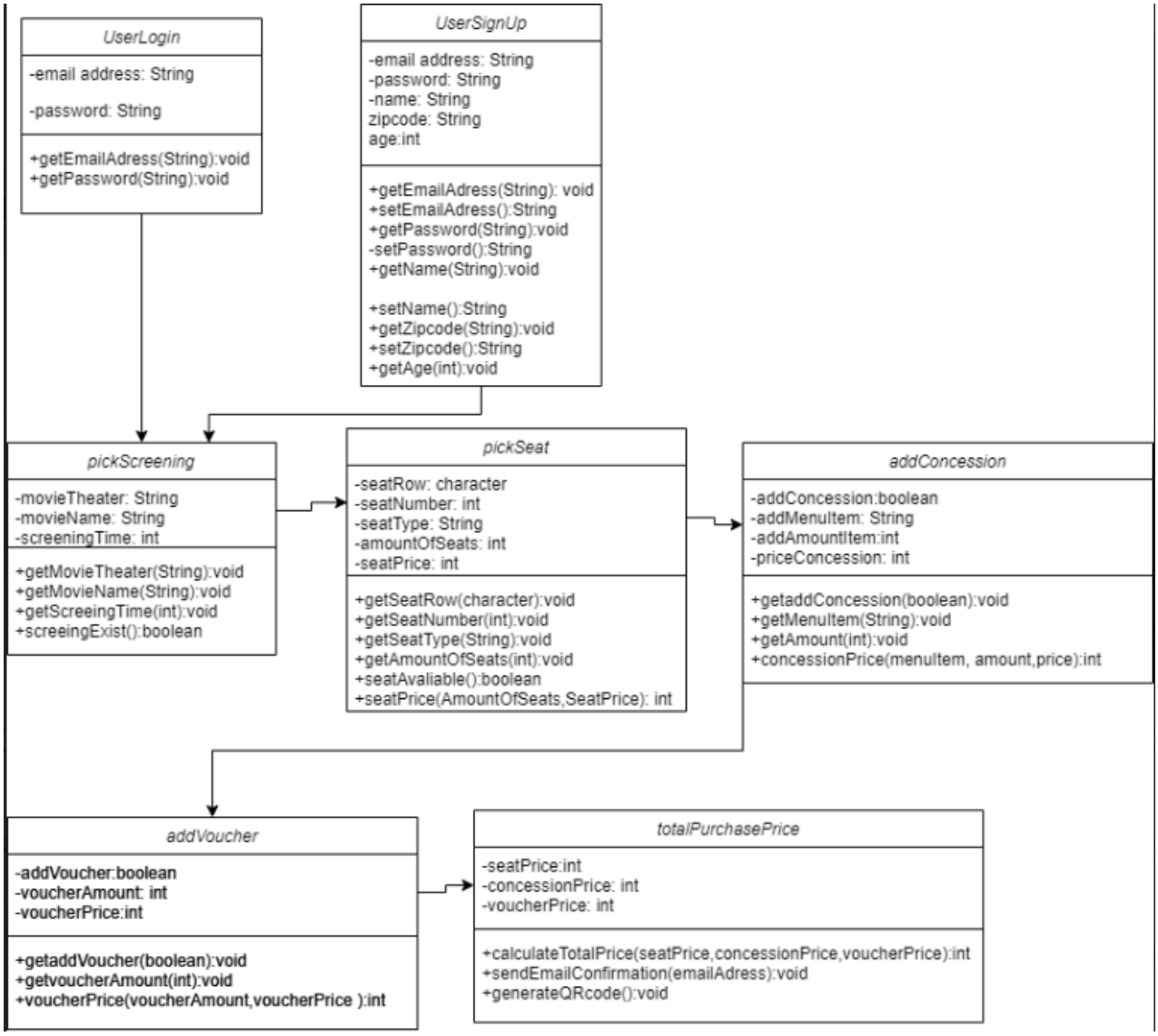
3.9 Other Requirements

Catchall section for any additional requirements.

4. Analysis Models

4.1 Sequence Diagrams

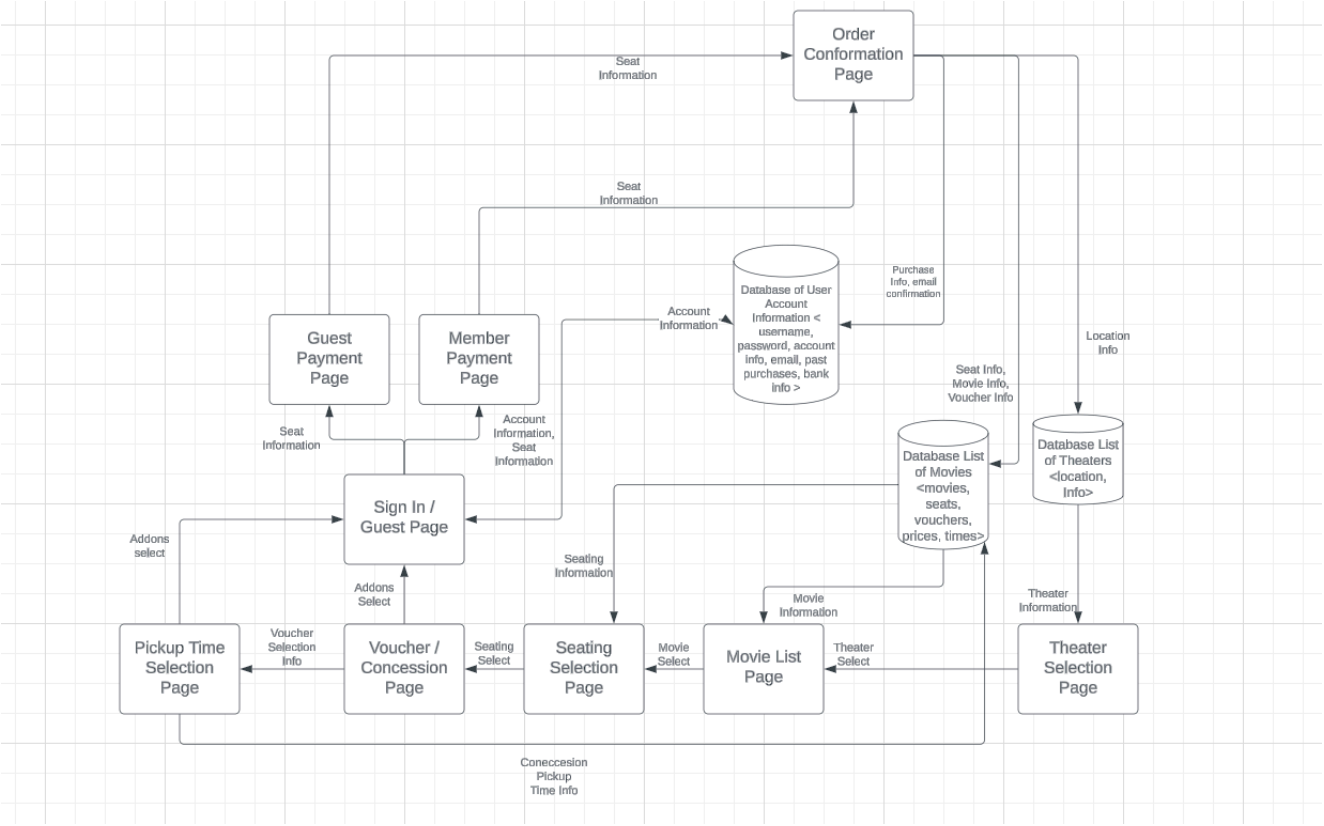
UML Diagram:



The UML diagram above describes the classes that are needed to perform various functionalities. The userLogin class represents the user credentials as String data type. If the user does not have an account the userSignUp class will be implemented. This class gets data such as email address, name, zip code , age and password as strings and ints data type and sets the information in the database while creating an account. Once a user created an account or logged in they are able to purchase a movie ticket. To purchase a ticket the pickScreeing class will be implemented along with pickSeat. pickScreeing allows the user to pick a movie theater, movie and screening time. This class gets the data from the user and checks if the user preferred screening exists and available. Next the user gets to choose their seat, seat type and the amounts of seats using the pickSeat class. To check if the seat is available boolean expression will be implemented if available expression will return true if not will return false. This class also calculates the cost of the tickets based on the variables in the class. Our application allows the user to schedule

concession pickup which will be implemented using the addConsession class. When the user wants to add a concession and schedule ahead the method addConcession will return true otherwise it will return false. This class includes menu items that are represented by string, amount of concession represented by int, price of each item represented by int and the total price that will be calculated using the concessionPrice() which will return an int. The user can also add a voucher that will allow the user to bring outside food to the theater. The addVoucher class will return true if the user is adding a voucher, false if not. The class will add the amount of vouchers and will calculate the total price. The last class to get implemented is the totalPurchase class that will add up the prices of each item such as ticket, concession and voucher and will receive the price of each item from the classes listed above. This class will also send an email confirmation to the user along with making a QR code the user can use at the theater.

4.2 Data Flow Diagrams (DFD)



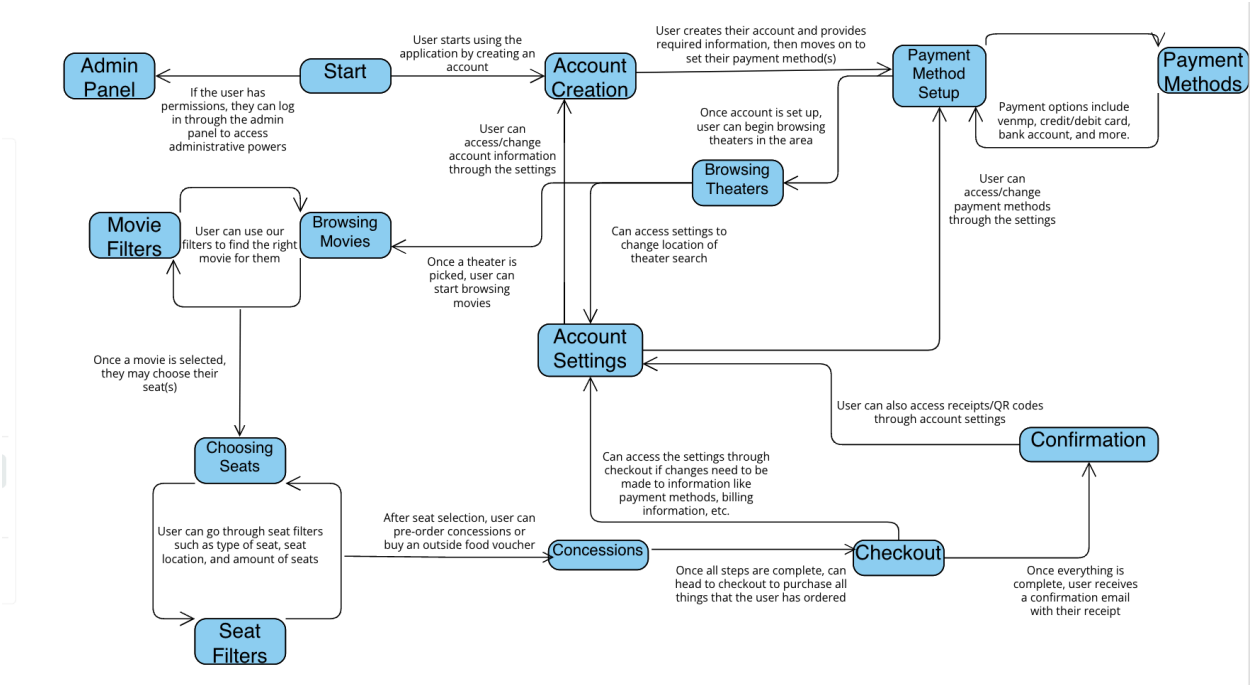
Shareable Link for Better View:
https://lucid.app/lucidchart/778638fd-7a71-4345-95fe-313fc1ce61b7/edit?viewport_loc=-1290%2C-354%2C2377%2C1116%2C0_0&invitationId=inv_9096052a-147a-47e0-abc0-7f4410345808

The Software Architecture diagram shows the databases that will be connected, as well as how each page and information between each page will connect. The home page is a theater selection page, pulling data from the list of theaters. This then connects to the movie list after a theater is selected. The movie list page pulls data from the list of movies being shown, as well as other

necessary information. After a movie is selected a seat selection page is shown and the list of movies database is once again accessed giving seat information. After seating selection, the user is prompted to select concession purchases as well as vouchers for food, a crucial part of our design. If no concessions are selected, the user is taken to the sign in / guest page, if concessions are selected, a pickup time page is displayed. After choosing guest or signing in through access to the account information database, users are prompted with the payment page of their previous selection. The payment is then confirmed with an automatic email sent to users, and the order data is sent to the list of theaters database, list of movies database, and account information database to update.

The software architecture diagram has been updated to contain needed databases that have been approved by the development team. These databases include a database of account information, a database of a list of movies, and a database of movie theaters. The diagram also includes the different methods to be included in which will be fully connected.

4.3 State-Transition Diagrams (STD)

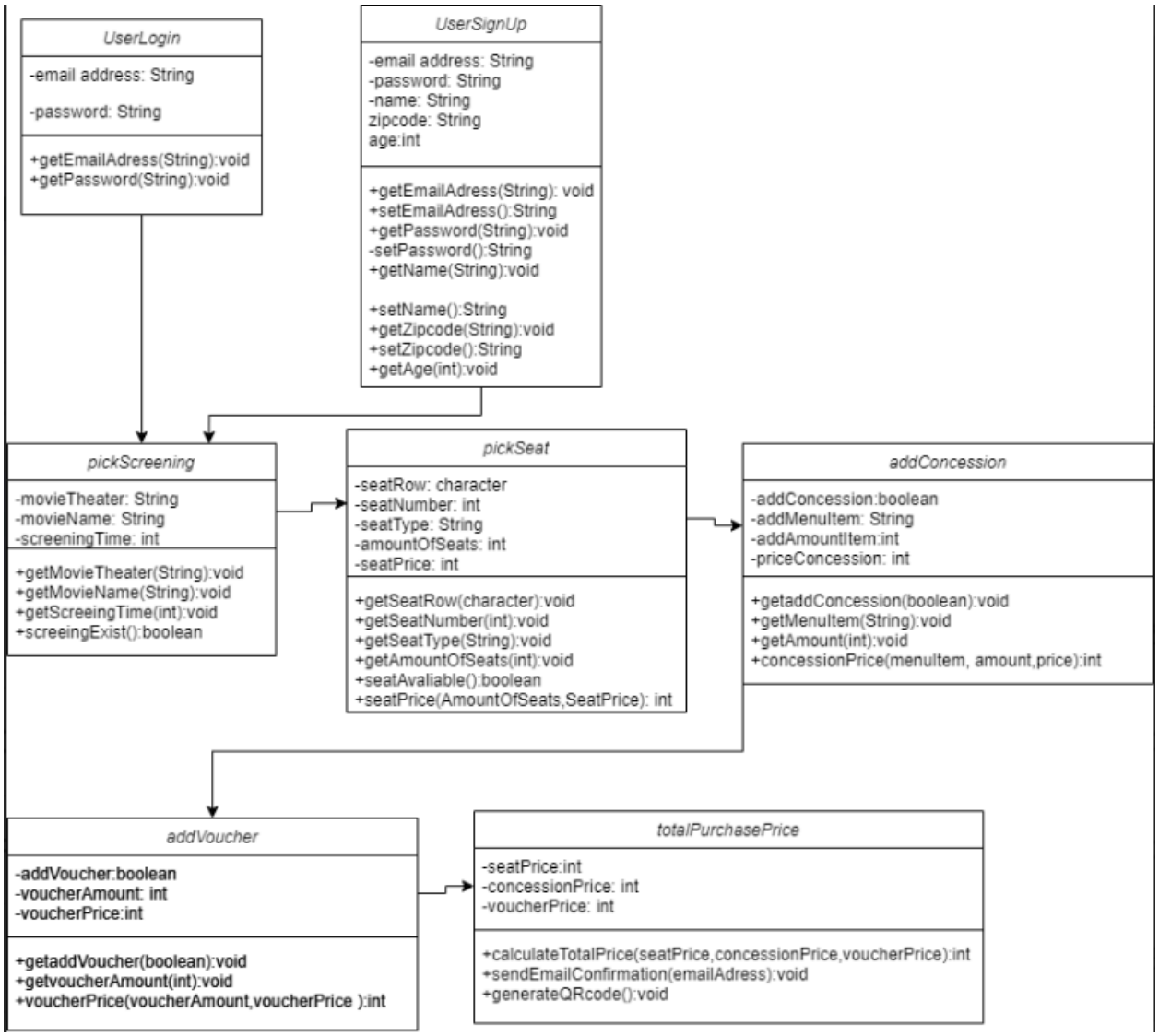


This diagram details the steps users can take to go through our app, and use it properly. It highlights the details of every step, and how to get to that step. Some steps have multiple features, like when browsing a movie you can filter it through many filters, or when selecting a seat you can filter the types of seats of the seat amounts. The blue boxes in this diagram represent the stages that the user can go through, and the arrows represent the transitions from one state to another.

5. Change Management Process

The SRS is a living document that is being updated daily to fit different criteria and constraints made known by shareholders and potential buyers. During the instance of a major scope change, the document will be created new and templated to adequately handle previous missing info and system changes(This process has been used once during a beginning scope change). Changes will only be added by the original developers and creators of the SRS, these being named in Section 1 of the Revision history. These names include Aidan Armas, Noam Joseph, and Anna Krassowizki. During a transition of management, these names will be redefined and changed to fit new owners' requirements. Changes will go through an approval process beginning with the three lead developers, then making their way to Cole Simpson, and finally Dr. Gus Hanna.

6. Software Test Plan



The UML diagram above describes the classes that are needed to perform various functionalities. The userLogin class represents the user credentials as String data type. If the user does not have an account the userSignUp class will be implemented. This class gets data such as email address, name, zip code , age and password as strings and ints data type and sets the information in the database while creating an account. Once a user created an account or logged in they are able to purchase a movie ticket. To purchase a ticket the pickScreeing class will be implemented along with pickSeat. pickScreeing allows the user to pick a movie theater, movie and screening time. This class gets the data from the user and checks if the user preferred screening exists and available. Next the user gets to choose their seat, seat type and the amounts of seats using the pickSeat class. To check if the seat is available boolean expression will be implemented if available expression will return true if not will return false. This class also calculates the cost of the tickets based on the variables in the class. Our application allows the user to schedule

concession pickup which will be implemented using the addConsession class. When the user wants to add a concession and schedule ahead the method addConcession will return true otherwise it will return false. This class includes menu items that are represented by string, amount of concession represented by int, price of each item represented by int and the total price that will be calculated using the concessionPrice() which will return an int. The user can also add a voucher that will allow the user to bring outside food to the theater. The addVoucher class will return true if the user is adding a voucher, false if not. The class will add the amount of vouchers and will calculate the total price. The last class to get implemented is the totalPurchase class that will add up the prices of each item such as ticket, concession and voucher and will receive the price of each item from the classes listed above. This class will also send an email confirmation to the user along with making a QR code the user can use at the theater. The modifications that were made in the UML diagram are additional functions that will be performed by the total purchase class. This class will now generate a QR code and will send an email confirmation.

Test plan 1:

The first course of test planning will be an automated test process that is done by the database management system at all times. The objective of these tests are to make sure that inputs by the user are correct and not causing errors through concurrent operations and queries. The approach will take in input for each function/feature to be listed below, and will give a desired output compared to the output that was the result of the user's input. This testing method will allow the developers to see the difference from expected outputs and given outputs. The features to be tested using this automatic process done by the database management system are as follows:

- The Login Function
- Payment Method Information
- Location Function
- Account Creation

Test plan 2:

The second course will be a manual testing process using a blind test group that will use functions and features to assist in the processing and debugging of different functions. The aim of this plan is to get hands-on experience to understand how the different databases and queries will interact with one another to reduce and stop error from occurring. The testing method will test key features of the movie selection process and "list of movies" database. The functions to be tested are as follows:

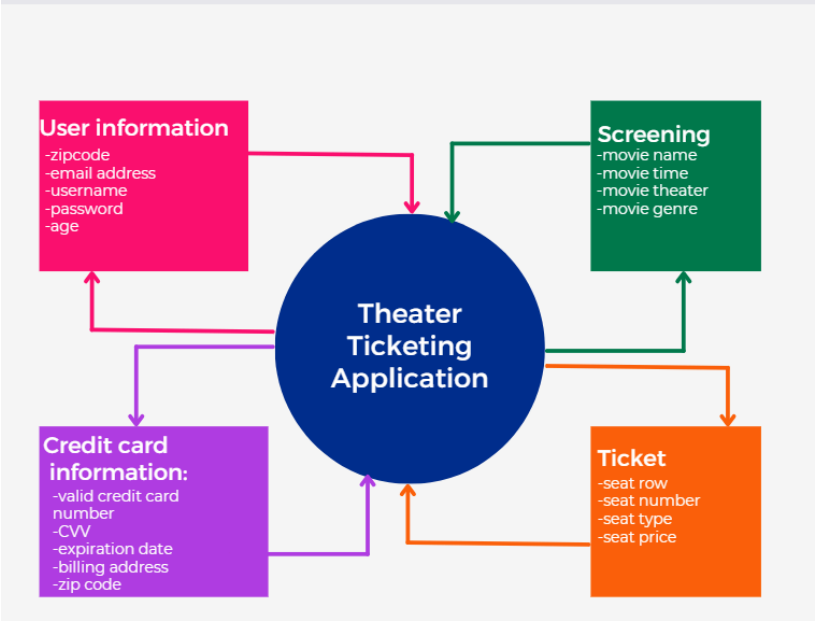
- Movie Selection
- Seat Selection
- Account Settings
- Concession Scheduling
- Outside Food Vouchers
- Checkout Confirmation

Test Cases:

https://docs.google.com/spreadsheets/d/1B4HC50U9a4XbrE-SQMUtF_Ft2ly0Jbm4c-shZ9FHJZ0/edit#gid=0

A	B	C	D	E	F	G	H	I	J
Test Case Template									
TestCaseId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By
logintest#1	login_function	P0	Verify successful login with v	existing account	1. navigate to the login page 2. Enter user name 3. Ente	navigate to the	navigated to the	Pass	Anna Krassowizki
paymenttest#1	payment_function	P1	Verify valid payment informati	electronic payme	1. Ask the user for payment method 2. Check if the user p	If the test passe	If the test passe	Pass	Anna Krassowizk
locationTest#1	location_function	P2	verify if the user provided a va	list of valid zipco	1. Get a zipcode from the user 2. check if the zipcode is va	valid zipcode na	valid zipcode na	Pass	Anna Krassowizk
accounttest#1	account_creation_functio	P3	Verify user provided accurate	have a valid ema	1. Collect necessary information. 2. Verify all information is	All information p	All tests passed	Pass	Noam Joseph
selectiontest#1	movie_selection_functior	P4	Make sure all filters are able t	have a valid acc	1. Use different filters such as genre, length, an ratings to	Movie selection	Movie was able	Pass	Noam Joseph
seatselectiontest#1	seat_selection_function	P5	Once user selects a movie, va	have a movie se	1. Select the movie you would like to watch. 2. Navigate to	Movie selected v	The movie was p	Pass	Noam Joseph
settingstest#1	account_settings_functio	P6	Verify the user can go to their	Have a valid acc	1. Go to the user settings display. 2. Be able to navigate th	User is able to n	The user was ab	Pass	Noam Joseph
concessiontest#1	concession_selection_fu	P7	Verify user can buy and sche	Selected a movi	1. Add Concessions wanted. 2. If timeslots available choo	User is able to s	User is able to sel	Pass	Aidan Armas
foodvoucher#1	food_voucher_function	P8	Verify user can purchase food	Movie and seat s	1. Select Outside food voucher 2. Pay for items, receiving	Proper QR code	Proper QR code	Pass	Aidan Armas
checkoutconfirmation#1	checkout_confirm_functi	P9	Verify user is given a confirma	Items paid for ar	1. Enter email when prompted 2. Press pay for all items. 3	Confirmation ID	Confirmation ID	Pass	Aidan Armas

Data Management Strategy:



This diagram displays a few of our databases that we use to store very important data. When the user goes through creating their account, they will go through the process of entering a bunch of information that is necessary for our app to function correctly. All of this information will be stored into the databases, and will only be accessed when called upon. When the user creates their account, they will be asked for their name, username, email address, zip code, password, age, and more, and this will all be stored in the User Information database. This information will only be accessed when the user requests to change/view their information, or if the application requires it to perform a function. The user will also be required to enter at least one payment method to be stored into the payment method database. For example, if the user enters a credit card, they will be asked for the credit card number, CVV, expiration date, billing address, and zip code, and this will all be stored in a credit card database within the payment information database. This will ONLY be accessed when the user is making a payment and wants to use this payment method, or if they wish to change their current payment methods.

When the user is browsing for a movie to watch, they will go through many filters and settings to find the right movie for them. Our application has a lot of movie options at a lot of different theaters, so databases must be created in order to keep all these movies on our application. This database will contain every movie's name, time, theater, genre, ratings, etc, and will be provided to the user once they ask for extended information about the movie they select or are browsing. The user, after selecting a movie, will also be required to select the type of tickets they want. We have a ticket database that contains tickets for every movie, and this database will contain tickets that have a seat number, seat row, seat type, and seat price, that will also be displayed to the user whenever they want to select a ticket. These are some of our databases that we use to protect and access important information, along with other security methods we provide on our application.

Data Strategy:

This application will be developed using SQL because of its wide range of market use and a large scalability for different sizes of potential buyers. SQL also provides for a large scale data analysis while also keeping the simplicity and easy to use construct. Lastly, data integrity and security is vital to the holding of sensitive data, something SQL does best. The databases will be split into large databases through SQL, then subdivided further. The three main databases are a list of movies, a list of movie theaters, and account information. Within these databases are different methods and items to connect the three using open database connectivity through the application. This allows for quick and efficient querying and data manipulation. SQL is a better option for this application due to the ease of quick data management.

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