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.

INOX Theater Ticketing

Software Requirements

Specification <Version>

<Date>

Group 14

Isaiah Austin,
Dhruvanshi Mokani,
Sreehith Nareddy

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

INOX

Revision History

Date	Description	Author	Comments
2/1/2024	Version 1.0	Isaiah Austin,	Initial Revision

		Dhruvanshi Mokani,Sreehith Nareddy	
2/8/2024	Version 1.1	Isaiah Austin	Sections 2 & 3
2/15/2024	Version 1.2	Isaiah Austin, Dhruvanshi Mokani,Sreehith Nareddy	Sections 2 & 3
2/29/2024	Version 1.3	Isaiah Austin	Sections 4.1 & 5

Document Approval

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

Signature	Printed Name	Title	Date
	Isaiah Austin	Software Eng.	
	Dhruvanshi Mokani	Software Eng.	
	Sreehith Nareddy	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

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

1.1 Purpose

What is the purpose of this SRS and the (intended) audience for which it is written

The purpose of this document is to specify the requirements and functionality of the software being produced.

1.2 Scope

This subsection should:
(1) Identify the software product(s) to be produced by name; for example, Host DBMS, Report Generator, etc
(2) Explain what the software product(s) will, and, if necessary, will not do (3) Describe the application of the software being specified. As a portion of this, it should: (a) Describe all relevant benefits, objectives, and goals as precisely as possible. For example, to say that one goal is to provide effective reporting capabilities is not as good as saying parameter-driven, user-definable reports with a 2 h turnaround and on-line entry of user parameters.
(b) Be consistent with similar statements in higher-level specifications (for example, the System Requirement Specification) , if they exist.What is the scope of this software product.

The product 'IXT' (1) is a website for INOX theaters that displays movies and will allow customers to purchase a user-defined amount of tickets at any INOX theater within the United States, as well as create an account for ease of access and numerous other functions.

1.3 Definitions, Acronyms, and Abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

(1) 'IXT' - INOX Ticketing
(2) 'PII' - Personal Identifiable Information, including credit card information, first and last name, email address, phone number, and date of birth.
(3) 'HTTPS' - Hypertext Transfer Protocol Secure

1.4 References

This subsection should:
(1) Provide a complete list of all documents referenced elsewhere in the SRS, or in a separate, specified document.
(2) Identify each document by title, report number - if applicable - date, and publishing organization.
(3) Specify the sources from which the references can be obtained.
This information may be provided by reference to an appendix or to another document.

(4) Motion Picture Association Film Ratings (motionpictures.org)

1.5 Overview

This subsection should:
(1) Describe what the rest of the SRS contains

The portion of this document that follows shall provide specific details, requirements, and abilities of the functionality of IXT and all development concerns. The early sections begin with broader requirements before identifying key and specific elements of the software, of which an engineer requires to produce this product.

(2) Explain how the SRS is organized.

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 subsection of the SRS puts the product into perspective with other related products or projects. (See the IEEE Guide to SRS for more details).

2.2 Product Functions

This subsection of the SRS should provide a summary of the functions that the software will

perform.

The product displays movie listings for the United States, allowing users to browse current and upcoming showings and availability, select seating, and purchase a user-defined amount of tickets for their desired theater. Users will be able to purchase and redeem gift cards, and create an account to save personal information. Lastly, users will be able to purchase, redeem and send INOX gift cards, and contact customer support for refunds, etc.

2.3 User Characteristics

This subsection of the SRS should describe those general characteristics of the eventual users of the product that will affect the specific requirements. (See the IEEE Guide to SRS for more details).

Users of IXT will need basic computer competency in order to navigate the website and input payment information if necessary. Additionally they will need to be able to have access to an email or phone number in order to receive order confirmations.

2.4 General Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options for designing the system. (See the IEEE Guide to SRS for a partial list of possible general constraints).

Constraints include:

- Security concerns over receiving user PII.
- Restriction to web development languages
- Bandwidth to support heavy traffic
- API to Stripe in order to process transactions

2.5 Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

Assumptions include a computer and browser capable of accessing the HTTPS (3) IXT website.

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 the carefuly organize the requirements presented in this section so that they may 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

All UI will be graphical interface pages.

- Home
  - Page to be linked to by search engines
  - Displays newest or featured/popular movies by user click, more info on bottom of page
  - Buttons with access to pages: accounts, gift cards, movie selection, contact us
- Account
  - Displays fields to enter username and password
  - Buttons with access to pages: create account, home page, accounts
- Create Account
  - Displays fields to enter desired username and password, and PII.
  - Buttons with access to pages: Home page, accounts
- Gift Cards
  - Displays gift card buy option, and field to check balance
  - Buttons with access to pages: Gift card checkout, home page, accounts
- Gift Card Checkout
  - Displays fields to enter gift card amount, checkbox if it's a gift to email to someone else, and additional field if that box is checked to enter recipient address. Lastly all PII in order to complete purchase, and displays accepted payment methods.
  - Buttons with access to pages: Gift card, home page, confirmation
- Movie Selection
  - Displays movie poster, name, runtime, and ratings from MPA (4)
  - Drop-down to search for theater and buttons to select date and showtime + quality
  - Buttons with access to pages: Home, accounts, seat selection
- Seat Selection
  - Displays theater layout with all available seats and ticker counter.
  - Allows selection of multiple seats, which increases ticket counter
  - Buttons with access to pages: Home, movie selection, accounts, movie checkout
- Movie Checkout
  - Displays selected movie information (name, theater, time, seats) Provides fields to insert PII. And displays accepted payment methods

    - Buttons with access to pages: Home, seat selection, accounts, confirmation
- Accounts (if signed in)
    - Displays PII, and button to drop down purchase history, change email, password, and sign out
    - Buttons with access to pages: Home, accounts
- Confirmation
    - Displays order details and button to print confirmation
    - Buttons with access to pages: Home, accounts
- Contact Us
    - Displays customer support number, email and a button to request a refund
    - Buttons with access to pages: Home, accounts

3.1.2 Hardware Interfaces

Requires access to internet, wireless or cabled.

Optional connection to printer

Browser capable of viewing web development language elements.

3.1.3 Software Interfaces

- Movie Information Database
    - Microsoft SQL Server 2022
- Website Telemetry
    - Google Analytics
- Payment Processing
    - Stripe
- Operating System
    - Windows

3.1.4 Communications Interfaces

User communication to the website will be done through HTTPS

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 Location Selection

3.2.1.1 Introduction

The software shall allow a user to provide their location in order to automatically fill a list of theaters closest to them

3.2.1.2 Inputs

User location access permission

3.2.1.3 Processing

Browser geolocation function

3.2.1.4 Outputs

Curated list of theaters from least distance away from user to greatest

3.2.1.5 Error Handling

Request users manually enter address or request location permissions if not previously activated.

3.2.2 Handle Electronic Transactions

3.2.2.1 Introduction

The software shall allow users to complete electronic transactions for purchasing movie tickets or INOX gift cards.

3.2.2.2 Inputs

User PII

3.2.2.3 Processing

Stripe API and database to update taken seats

3.2.2.4 Outputs

Redirect the user to the purchase confirmation page with details and QR code.

3.2.2.5 Error Handling

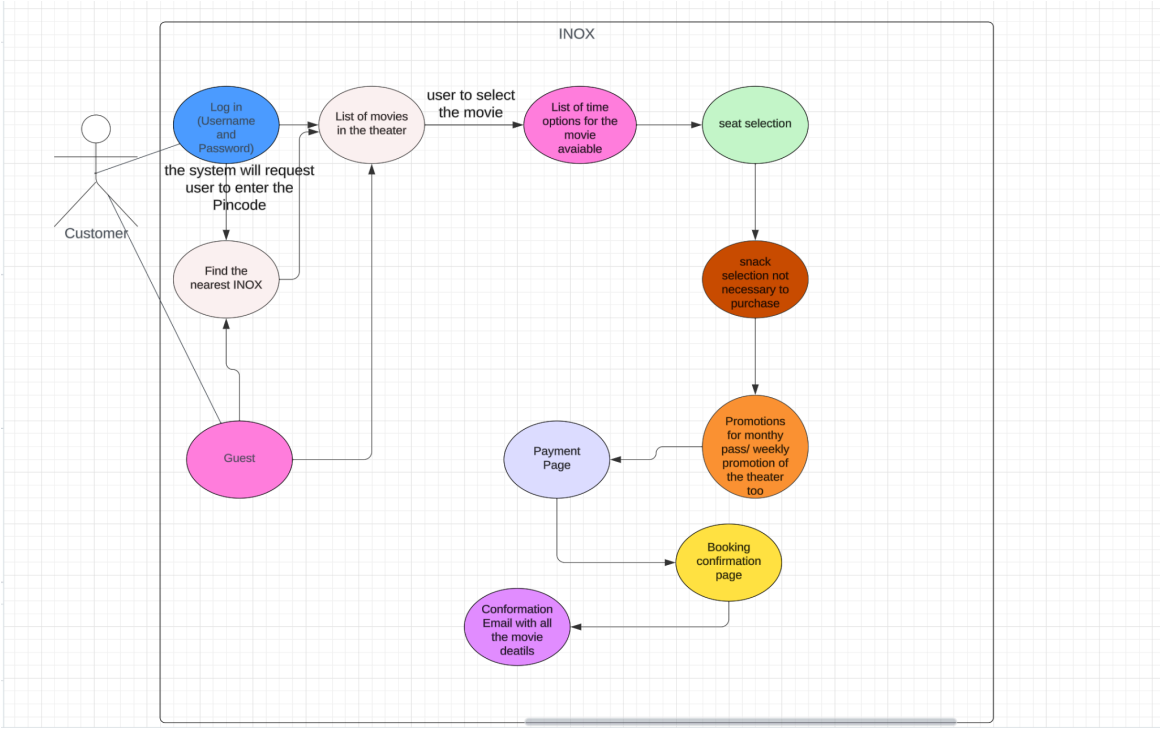
Any errors that are encountered should try to be troubleshooted by using Stripe's functions, otherwise providing refunds or resolving our payment gateway.

3.3 Use Cases

3.3.1

The system will work on a very simple algorithm.When the user uses the website, he'll have to log in and the users can also log in to the website as a guest. They'll have to allow their location to the website so the system can detect the nearest theater and show the movies which are available at that time. Then the user can select the movie they want to watch and then select the seat that they like. Later, comes a page where they can purchase snack if they want and a monthly pass to the theater if they frequently movies and they don't want to go through the process, which means when a user buys the monthly pass they can directly select they movie they want to watch and then they'll have to enter their pass number so, they'll not have to go through the payment gateway everytime they book the tickets. Moreover, on the last step the user will get a ticket confirmation email which confirms their booking with confirmation number ,movie name,time, location, number of tickets purchased and the screen number in it.

3.3.2 Use Case #2



3.3.3 Use Case 3

As we are talking about use case scenarios, if at any point the user has any queries or they face any problem we can troubleshoot it on the website as well and can also call the customer care for an immediate solution to their problem which they are facing in any aspect of the website.

3.4 Classes / Objects

3.4.1 <Class / Object #1>

3.4.1.1 Attributes

The Attributes of this theater ticketing system as very simple and user friendly. The attribute used in the website are as follows:- List of Movies, time schedule for movie, seat selection, Payment, booking confirmation.

3.4.1.2 Functions

As soon as the customer opens the website the person has to log in or can also use it as a guest if the user doesn't want to log in if they are in a rush.
(i) List of movies :- Using this Attribute, the user can browse for the movies which are available in the theater.
(ii) List of time options for the movie available:- Using this can help find time slots for any movie.
(iii) Seat selection:- This helps to select any random seat in the theater.
(iv) Snack selection:- It gives an option of purchasing snacks online before the start of the show and is optional.
(v) Promotion:- Promotions of the monthly pass and theater would be active if provided.
(vi) Payment page:- It redirects you to the payment page which is totally secure and processes the payment in safe order.
(vii) Booking Confirmation page:- After the payment you get a confirmation with all the details on the current page and to your email id and phone number if provided.

<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 transaction 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

The system should be able to handle a high volume of ticket sales during peak hours, such as before popular shows or events, without significant slowdowns or downtime.

3.5.2 Reliability

The system should be highly reliable, ensuring that tickets are accurately reserved and purchased, and that the system is available when customers need it.

3.5.3 Availability

The system should have high availability, minimizing downtime for maintenance and ensuring that customers can access the ticketing platform at any time, including during scheduled performances.

3.5.4 Security

The system should employ robust security measures to protect sensitive customer data, such as personal information and payment details, from unauthorized access or breaches.

3.5.5 Maintainability

This refers to the ease with which the theater ticketing system can be maintained, updated, and modified over time. Key aspects of maintainability may include testing as well.

3.5.6 Portability

This concerns the ease with which the theater ticketing system can be deployed and run on different hardware platforms, operating systems, and environment.

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

The main requirement are the users who help in purchasing tickets for random shows playing in the theater, selecting seats, cancellation of ticket, monitoring the whole sales and revenue, configuring pricing tiers for different sections of the theater, Configure pricing tiers for different sections of the theater and many more basic users needs. Followed by the system functionality which includes purchasing tickets, managing booking, seat selection and customer support. They deeply help in on modes providing the customer to book tickets safely and securely. And finally coming to the integration requirements which includes website and third party integrations. Website integration mainly helps to seamlessly integrate with the theater's website for ticket purchasing. And the third party integration helps integrating with CRM systems for managing ones personal information. This inverted requirements document presents a detailed overview of the theater ticketing system's features and functionalities, with an emphasis on the needs and expectations of users, stakeholders, and the overall system architecture.

3.7 Design Constraints

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

The software shall be built on computers using Linux or Windows

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.

Databases shall be covered by Microsoft SQL Server, and rely on that softwares security systems for keeping our data safe.

3.9 Other Requirements

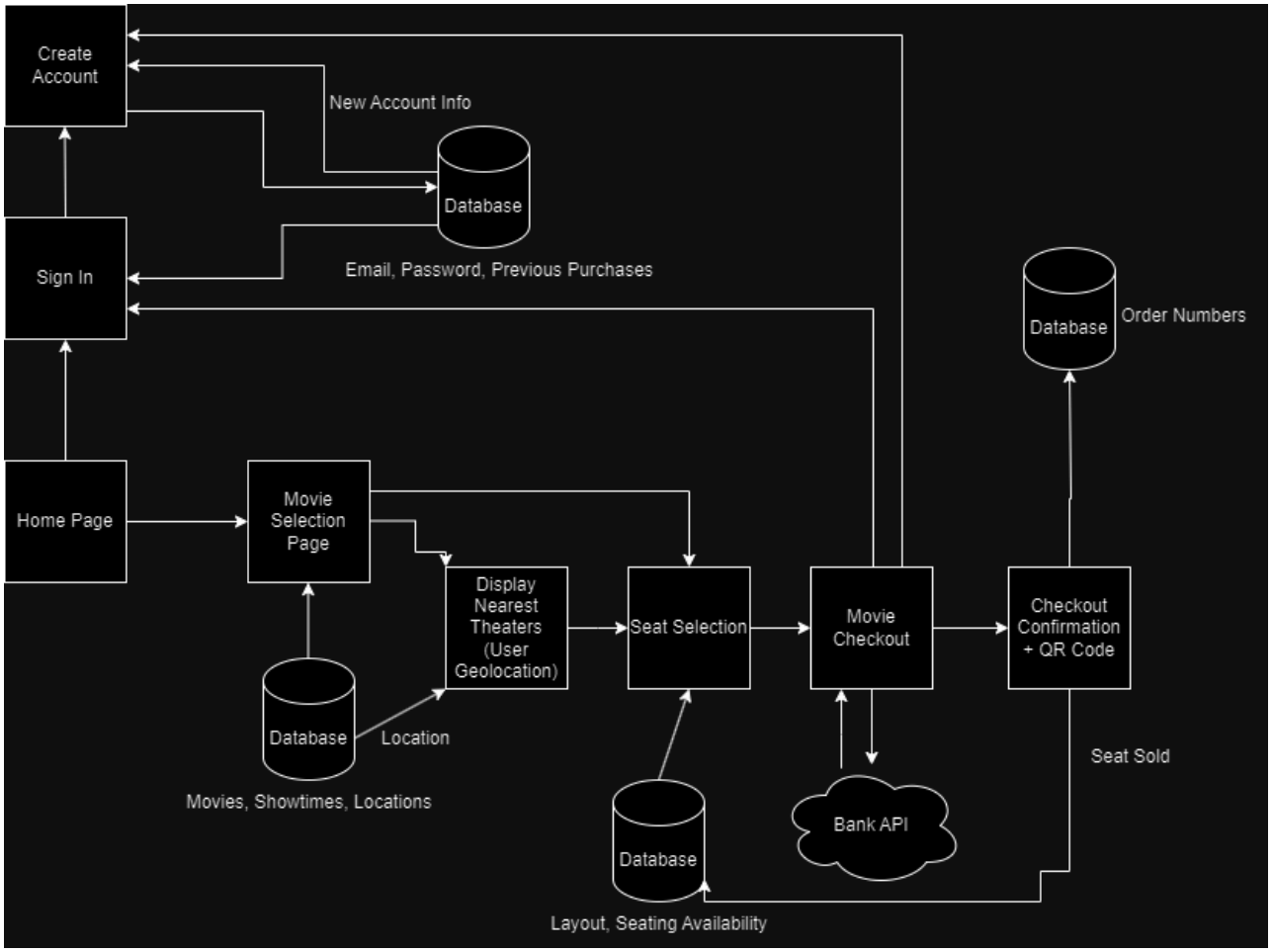
Catchall section for any additional requirements.

4. Analysis Models

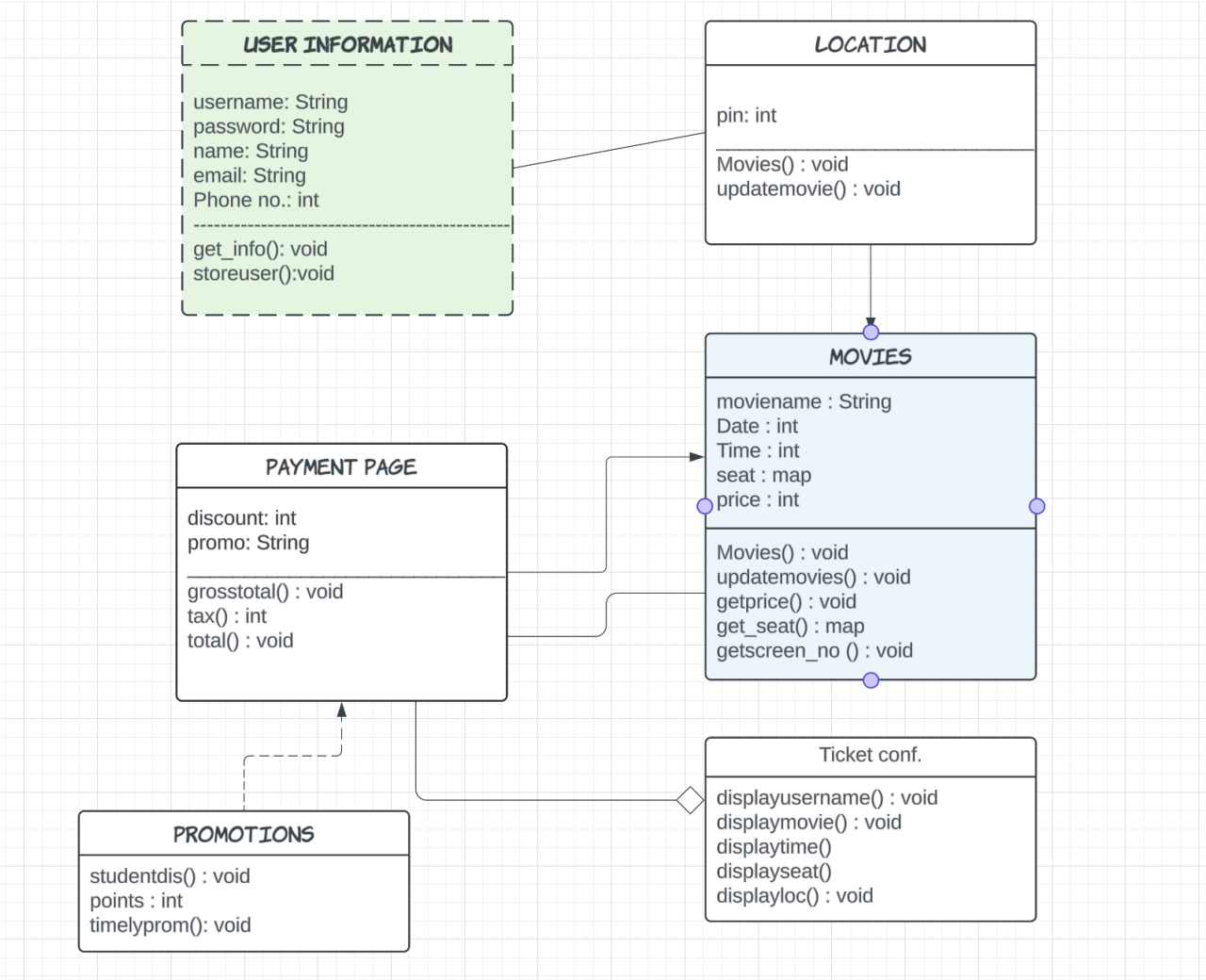
List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

This section shall display diagrams that enhance an understanding of the integral parts of the software and its requirements.

4.1 Architectural Diagrams (*Updated in Section 7.0)*



4.2 UML Class Diagram



4.3 Description of Classes

Starting with the User information:

- 'Username' (String): The username of the user.
- 'password' (String): The password of the user.
- 'name' (String): The name of the user.
- 'email' (String): The email address of the user.
- 'Phone no.' (int): The phone number of the user.

Payment page:

- 'discount' (int): The discount offered to the user.
- 'promo' (String): The promotion code for the user.

Promotions:

- `'studentdis()'` (void): A method to apply student discount.
- `'points'` (int): The number of points earned by the user.
- `'timelyprom()'` (void): A method to apply timely promotion.

Location:

- `'pin'` (int): The pin code of the user's location.

Movies:

- `'moviename'` (String): The name of the movie.
- `'Date'` (int): The release date of the movie.
- `'Time'` (int): The running time of the movie.
- `seat` (map): A map to store the seat details.
- `price` (int): The price of the movie ticket.

Ticket Confirmation:

- `'displayusername'()` (void): A method to display the username.
- `'displaymovie'()` (void): A method to display the movie name.
- `'displaytime'()` (void): A method to display the movie time.
- `'displayseat'()` (void): A method to display the seat details.
- `'displayloc'()` (void): A method to display the location details.

4.5 Description of Attributes

User information has two methods:

- `'get_info()'` (void): A method to get the user information.
- `'storeuser()'`(void): A method to store the user information.

Payment page has three methods:

- `'grosstotal()'` (void): A method to calculate the gross total.
- `'tax()'` (int): A method to calculate the tax.
- `'total()'` (void): A method to calculate the total amount.

Location has a method :

- `'Movies'()` (void): A method to display the movies.

- `'updatemovie'()` (void): A method to update the movie details.

Movies has five methods:

- `'Movies'()` (void): A method to display the list of movies.
- `'updatemovies'()` (void): A method to update the movie details.
- `'getprice'()` (void): A method to get the price of the movie ticket.
- `'get_seat'()` (map): A method to get the seat details.
- `'getscreen_no'()` (void): A method to get the screen number.

Ticket Confirmation has five methods :

- `'displayusername'()` (void): A method to display the username.
- `'displaymovie'()` (void): A method to display the movie name.
- `'displaytime'()` (void): A method to display the movie time.
- `'displayseat'()` (void): A method to display the seat details.
- `'displayloc'()` (void): A method to display the location details.

4.6 Description of Operations

- There are several operations going on in the UML Diagram.

  First of all its is “User Information”. It stores all the information of the user which includes username, their account password and also their personal information.

  Second there is “Location”, it asks the user to turn on their location in order to find the nearest location of the theater. Later, it captures the location to show the accurate results.

  Third would be “Movies”, which will display all the available movies to the users in the nearest location with their time and later on after deciding movie the “Movies” tab will also ask the user to select their seats as well.

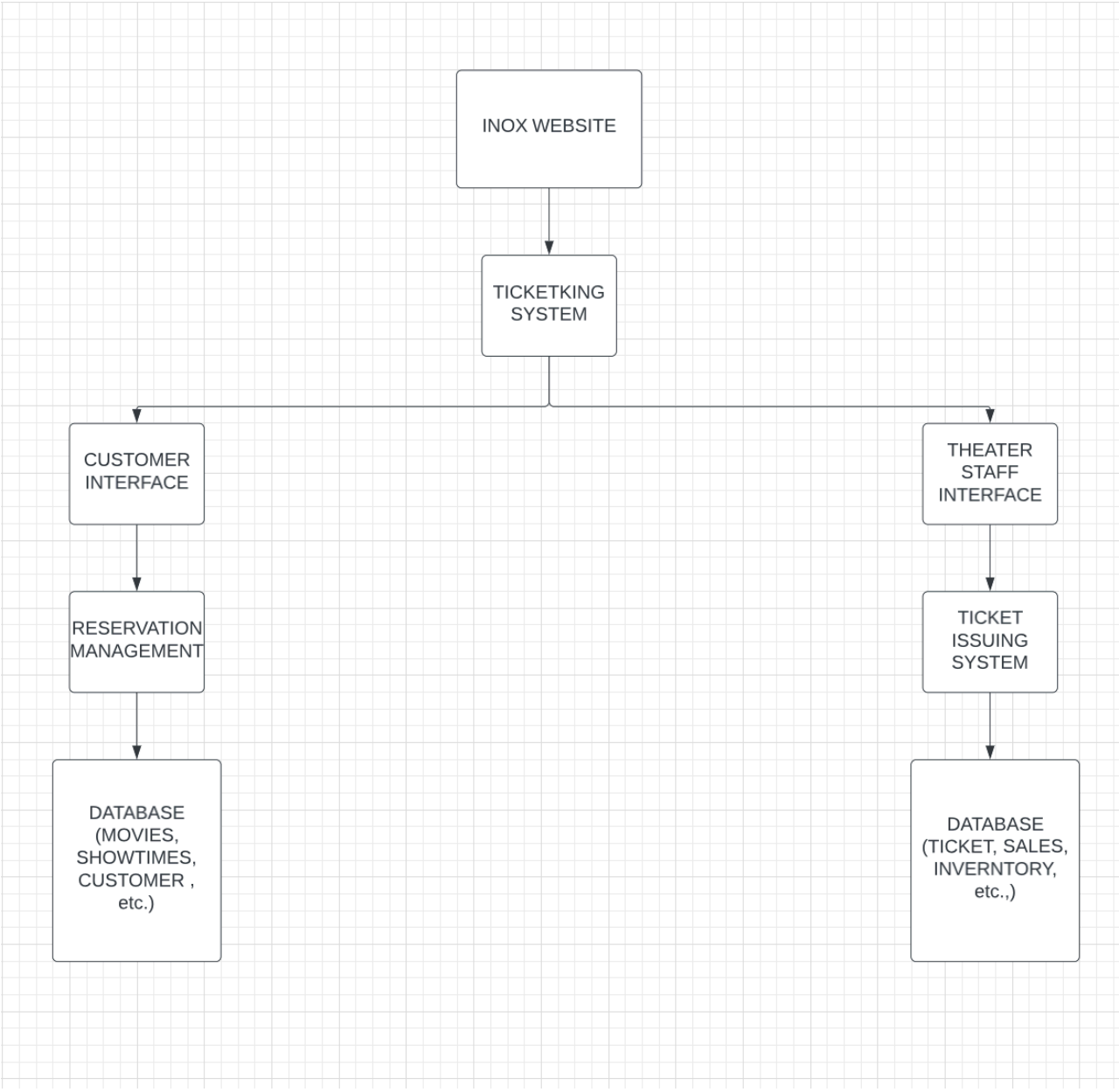
  Following would be a payment tab where the user would see the amount that they are suppose to pay to confirm their booking, where they would see the total amount which would be billed according to their respective choices.

  And then the last one would be “Confirmation” where the users would see all of the details of the movies and their booking which would avoid confusion in customer regarding the booking.

4.7 Data Flow Diagrams (DFD)

- The Inox website is responsible for managing the ticket booking process by acting as the main system for the whole thing.

- The customer interface allows customers to interact with the system. They can browse available movies, select showtimes, choose seats, and make bookings.
- The theater staff interface is used by theater staff to manage ticket sales, reservations, and inventory. They can view available seats, process ticket sales, and handle customer inquiries.
- Reservation management handles the reservation process initiated by customers. It manages seat availability, reserves seats for customers, and updates the database accordingly.



4.4 Development Plan And Timeline

The tasks and estimated time taken for each member are as follows:

Isaiah (https://github.com/ia4381/CS250-SRS):

- 4.1 Architectural Diagrams (40 min)
- 5.0 Change Management Process (5 min)
- 4.4 Development Plan And Timeline (10 min)

Sreehith:

- 4.4 Description of Classes (10 min)
- 4.5 Description of Attributes (10 min)
- 4.7 Data Flow Diagrams (25-30 min)

Dhruvanshi:

- 4.2 UML Class Diagram (45 min)
- 4.6 Description of Operations (25 min)

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.

Changes will be requested using a business management software, such as 'Flowlu' where alterations can be posted and described. The software engineers will then meet and review potential changes to determine their feasibility and decide what will be affected and how. A secondary document shall be created to allow engineers to freely update it. All changes must be marked by a highlight, with a comment of what was deleted. Any engineer(s) can be assigned the change upon completion of any previous tasks and any final changes will need to be approved by the software manager before being updated on the official document.

6. Verification

6.1 Introduction

This portion of the document will detail any functionality regarding user input and how it will be tested

6.2 Scope

The following are some functionality requirements of the software that will be tested:

Unit Testing:

1. Allow users to create account
2. Purchases are recorded
3. Record user information and history

Integration Testing:

1. Store users account information
2. Purchases are stored in database

3. Allow users to update information and add to history

Systems Testing:

1. Allow users to login
2. Purchases are confirmed with email and order number
3. Automatically detect rewards points and discounts

6.3 Test Plan

The aforementioned functionality will be tested through the following tests and strategy:

6.3.1 Test Plan 1

These tests revolve around user inputs and the respective outputs. They will be executed using white box functionality testing

Test 1: AcctCreation_1

This test is targeted at unit testing and will confirm that the userInformation class that houses all of the users information is capable of being stored and returned. This covers the requirement of storing users account information.

The testers input will be to call the storeInfo() method and then call the getInfo() method which should return default values from the constructor

Test 2: AcctCreation_2

This test is targeted at integration testing and will confirm that users can create an account and have it stored in the database. This covers the requirement of users creating an account.

The testers will go to the webpage and enter in default information for testing purposes, the output should be that information stored within the database that holds account information.

Test 3: AcctCreation_3

This test is targeted at System testing and will confirm that users can successfully login after creating an account. This covers the requirement to allow users to login to their account

The testers will go to the webpage and login with their aforementioned credentials, the output should be a redirection to the home page with an indication that they have been logged in

6.3.2 Test Plan 2

Test 1: PaymentInfo_1

This test case helps to verify the gross total, tax and total amount of the ticket after the selection of movie and seat selection. The user gets the exact value of the ticket inclusive of tax.

The tester input will be the grosstotal() method, tax() method and the total() method which gives the confirmation output.

Test 2: PaymentInfo_2

This test case helps to add promotion using student discount, reward points and the timely promotions. These are applicable once they meet their requirements. They will be added before the payment method to get the final amount.

The tester input will be the studentdis() method, points() methods and the timelyprom() method.

Test 3: VerificationInfo_1

This test case helps the user look at the details which include the username, time, location, seat and movie. And the user will receive the confirmation mail which includes all the above details

after the payment process.

The tester input will be the displayusername(), displaymovie(), displaytime(), displayseat() and displayloc().

6.3.3 Test Plan 3

Test 1: locat_1

Users will enter the pincode for the place they want to watch a movie, for the system to locate the nearest theater for them. The methods that were tested were Movies(), updatemovie(). The webpage then will locate and show options to choose the theater. It Should be redirected to a page with the list of movies available for the time being at the chosen theater.

User is sent to a different webpage after entering the pincode and making selection for the theater.

Test 2: user_1

Stores user information. It will collect personal data from the user like name, phone number, email.However it is not to be shared with anyone except the user and the information is stored in database as a record for future reference.

Test 3: Mov_1

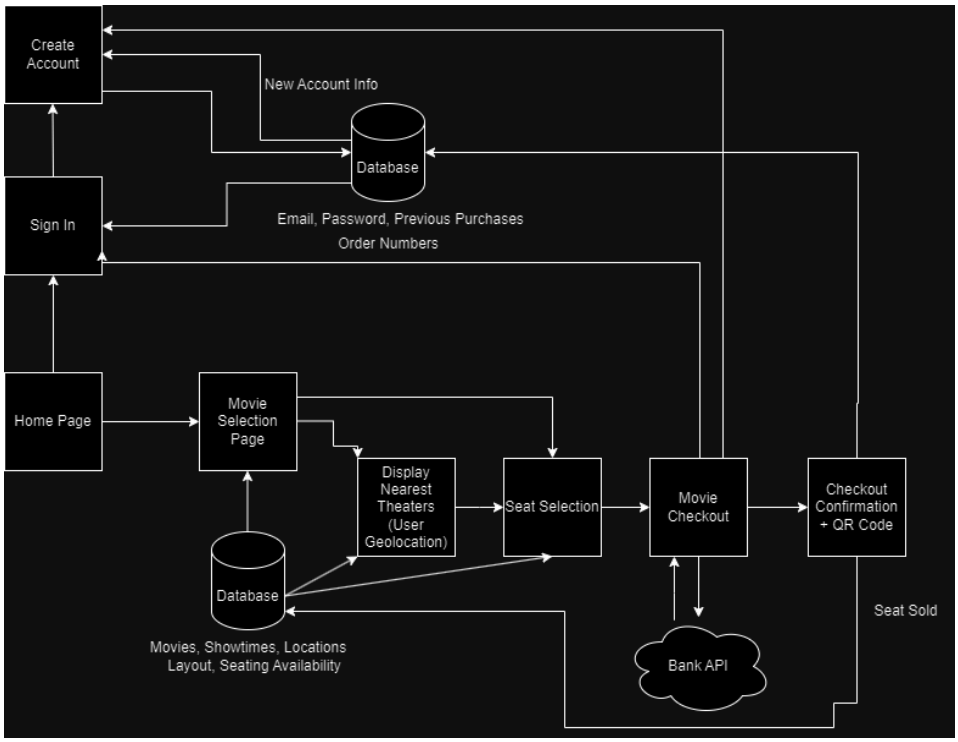
Collects information of the selections made by the user.launches to a page where the user can select what movie they want to watch. A web page where users can select the movie and their seats for the same and the data is stored until the customer gets the ticket confirmation. Stored in database for record of the user

Test 4: promo_1

It is a different webpage where the users can choose and apply discounts if there are any to their purchase. Provides promotion for the user if there are any available. A page where users can get offers for snacks and movies on discounted rate

7. Architecture Design

7.1 Software Architecture Diagram



7.1.1 Changes

- Merged order numbers into a database that houses account information for efficiency.
- Merged layout and seat availability database with the main database that stores all information about movies for efficiency.

7.2 Data Management Strategy

7.2.1 Database 1 Design:

The first database that stores account information at the top will be a NoSQL database. It is cheaper to maintain and it is easier to scale, and great for a large amount of data, meaning when a large volume of people create accounts and complete orders the database can be updated with ease. This is especially important since some users may create accounts and then abandon them for indefinite periods of time without requesting deletion, so this database will need the space and scalability to support that, in order to avoid deactivating unused accounts.

Cautions:

The main drawback of a NoSQL database is the lack of security features, which will be absolutely imperative when storing users account information. In order to compensate for this there will be more resources poured into designing the database to be as secure as possible, such as using MongoDB as they are ranked highly and have built-in security features. If a secure NoSQL database is not feasible then an alternative, SQL database, will be considered, which will increase costs.

7.2.2 Database 2 Design:

The second database on the bottom will be a SQL database. The reason for this is due to the fact that movie information will require multiple rows and columns of data associated with it. Additionally, adding movies or theaters will be less common compared to users creating accounts and therefore it will be more worth it to put everything in a rigid system that won't be scaled nearly as much. Lastly, this type of database is preferred for transactions which will be in high volume and important for customers reserving and purchasing seats.

Cautions:

As we are using SQL database for the second design which should be careful while handling and managing the data because as it has multiple rows and columns which is holding movies and seats information there can be a possible chance that the information might get jumbled and then there can be a miscommunication between its user and the website so, the information should be properly collected and uploaded in the website to avoid any misinterpretations.

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