Screen Seat Ticketing

Software Design Specification

Version 1

02/29/2024

Group #8

Sawyer Jones, Will Ruff, & Alexander Prochazka

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

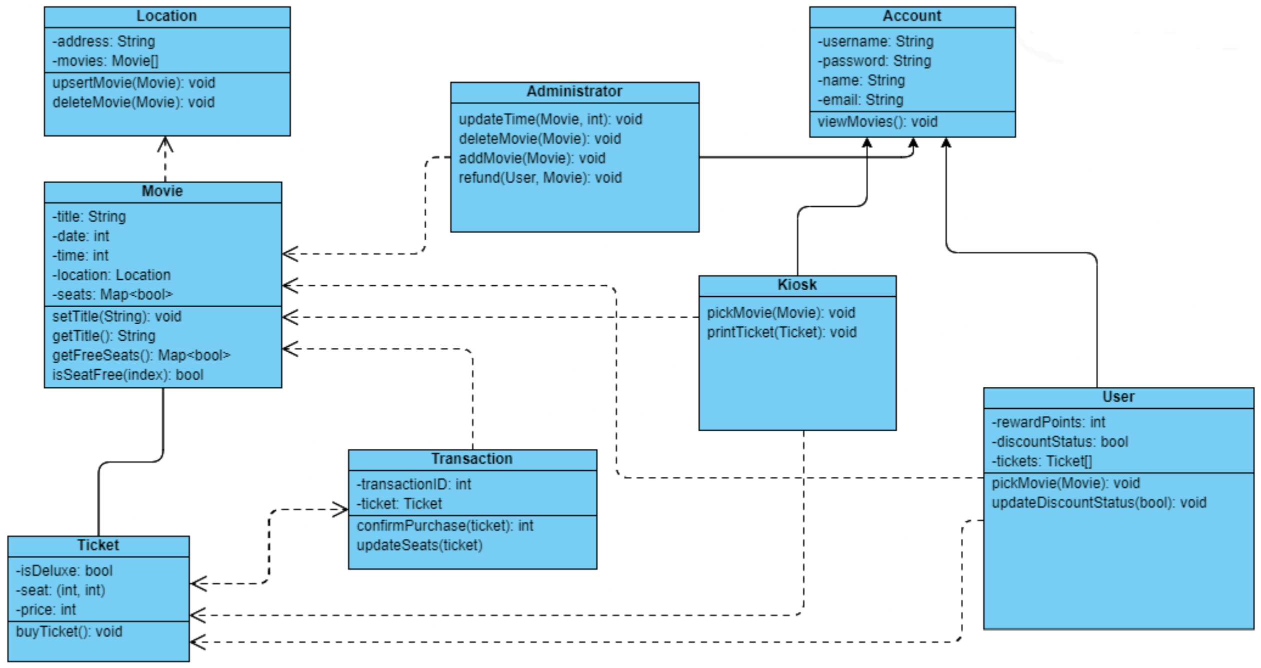
1. System Description

Screen Seat Ticketing allows users to conveniently purchase tickets for movies online. Users can browse available movies, view showtimes, select seats, and make secure payments through the website.

Data is kept on a database server. Three types of accounts, Users, Kiosks, and Administrators, all interact with the server. Movies are at a time and place, and each contains a map of tickets. Tickets are purchased, and orders are kept track of through a Transaction object.

2. Software Architecture Overview

UML Class Diagram:



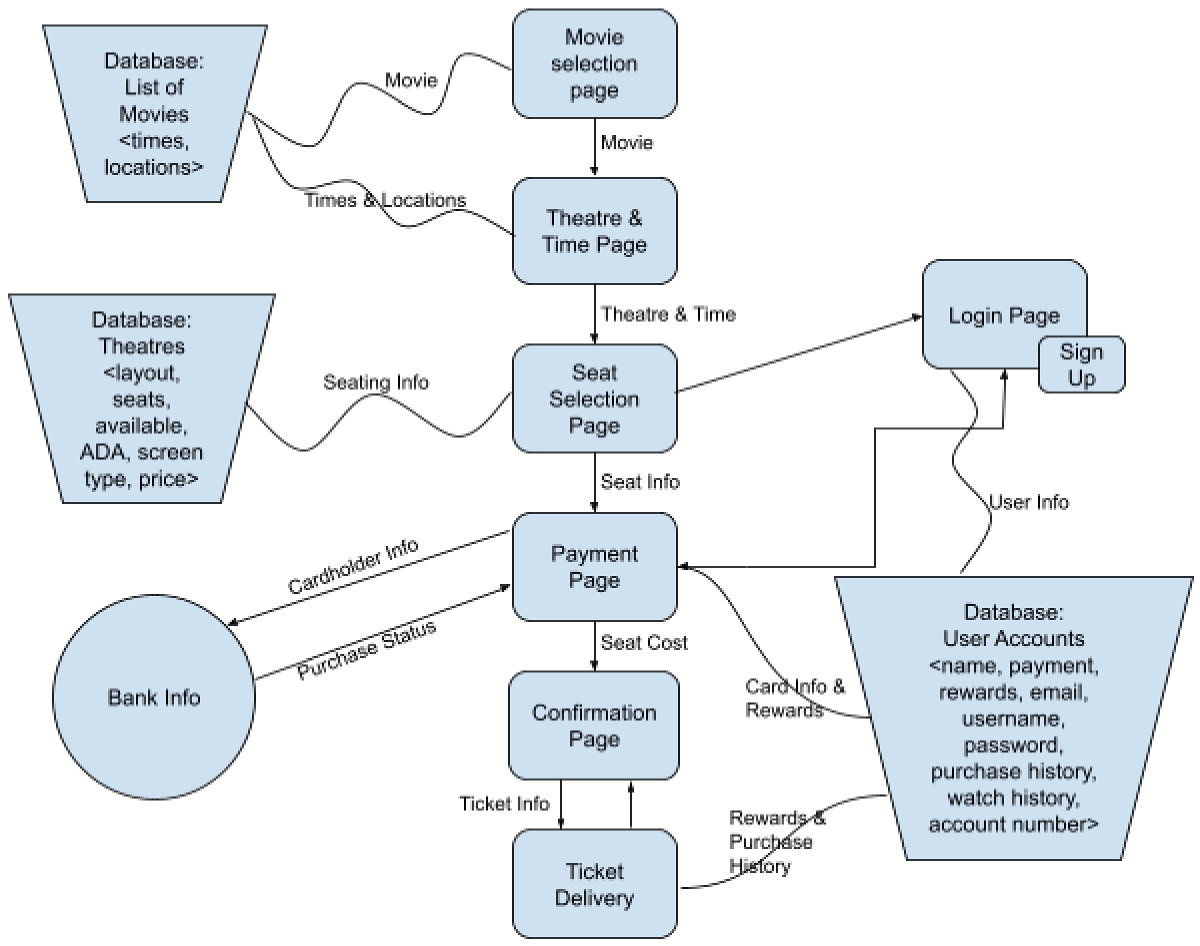
In our software, the Account class functions as the parent class to Administrator, Kiosk, and User classes. The Account class contains information about the user, such as username, password, name, and email. This information is critical to our software. Within this class we also have the viewMovies method, which lets users see all available movies. The User class has 3 inherent variables, rewardPoints, discountStatus, and tickets. These variables can be modified using the methods pickMovie and updateDiscountStatus. The tickets variable will be updated through the Ticket class.

The Ticket class has 3 variables as well, including isDeluxe, seat, and price. It also contains the buyTicket method that will allow the user to purchase tickets. A subclass of the Ticket class is the Movie class, which contains variables for the title, date, time, location, and seats available for each movie. We have getter methods for the title and the available seats, along with a setter for the title. We also have a boolean that can check if an individual seat is available. The last class involving the Movie class is the Location class. This class holds the address of the current theater, and an array of movies available at that location. We also give the location class the ability to add or delete movies from the available movie array.

Another class that interacts with the Ticket class is the Transaction class. This class contains secure variables that hold the transaction ID and the ticket(s) purchased. It also has methods that allow for seating to be updated, as well as a method to confirm if the ticket has been confirmed. We will also have a special class for our Ticket Kiosks, which will function in place of the User class when using an in-house kiosk. This class is a subclass of the Account class. This class will contain methods to select a movie and print the ticket once the transaction has been completed. The final child class of the Account class is the Administrator class. This class contains exclusively methods, which will allow us to modify movie showings. We will be able to update, add, or delete any of our theater's movies. It also has the ability to provide refunds to customers.

Ultimately, this UML diagram provides a visual representation of the relationships and structure found within our ticketing application. This diagram will help stakeholders and our team better understand our application.

Software Architecture Diagram:



The software architecture diagram contains descriptions of the databases that we will be using. We will utilize a database that will contain a list of movies and information relevant to those movies, and a database that holds our different theaters. The movie database will connect to the movie selection page. After selecting the movie, they will be brought to the theater & time page, where they will be able to select both a theater at a time. This information will be pulled back into the database that contains our list of movies. Next we will pull out seating information from our theater database, allow the user to select an open seat, and then update the seating information in the database.

This step will also pull data from our user accounts database, which will allow us to update the user's profile with their ticket information. Next we will bring the user to the payment page, which will be able to securely handle transactions. We will not store any sensitive information about customers. We will use an API to connect our site to banking services securely. After the transaction is successfully completed, we will bring the user to a confirmation page.

After confirming their purchase, the user will receive their ticket information, and we will update their account information within the user account database with the relevant information.

3. Development Plan and Timeline

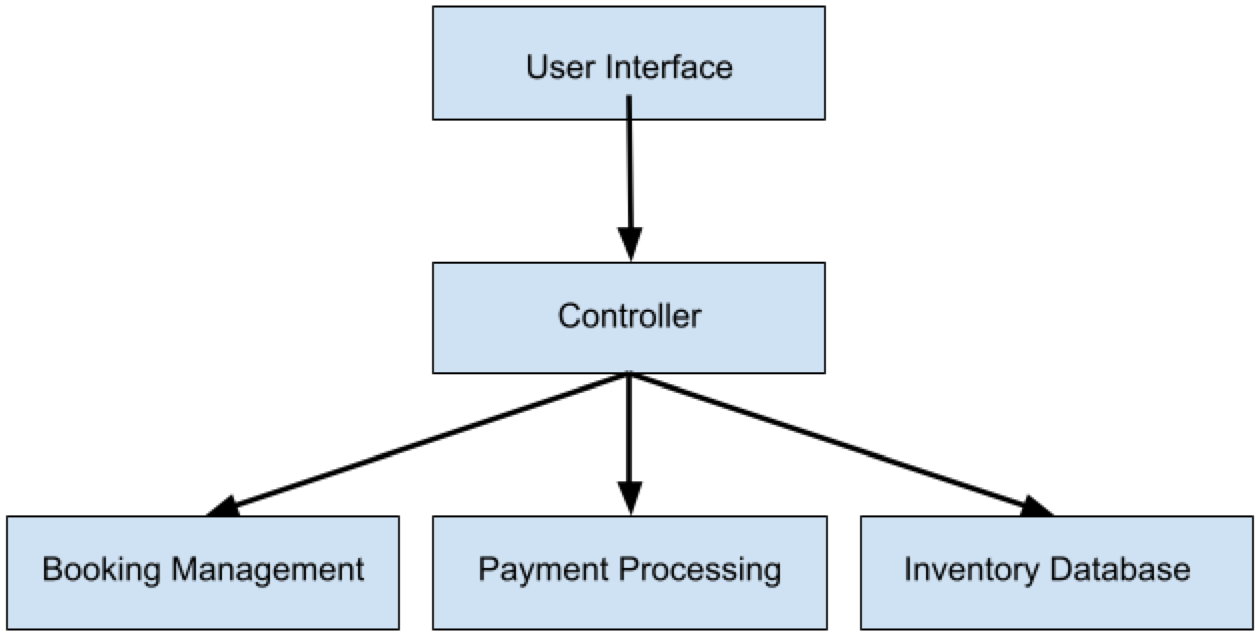
To start, the classes and objects will all be built, to the specifications of the UML diagrams. The UML diagram may need to be updated if unanticipated needs arise (new parameters, methods, entire objects).

Secondly, a database will be built, using the Location, Movie, Ticket, and Transaction classes as various types of entries. Account objects will exist as a separate list in the database. The methods contained within Account classes will be responsible for interacting with the database.

Next, the services will be connected to the database. This includes APIs for payment processing, and retrieving movie data. "Dummy services" may need to be built for testing purposes, until the full software is launched.

Finally, a front-end will be created. This website will be compatible on major computer and mobile browsers. It will display pertinent information based on the type of Account that is logged in. Buttons on the page will send queries to interact with the back-end that was designed before.

As a conservative estimate, this software will take six months to complete the design phase, from start to finish. Our team will consist of full-stack developers, so all members will be involved in all steps of the design process. Testing must be conducted at the end of each phase of design.



This diagram represents how data flows through our software. First, the user inputs data through our user interface. The data then flows through the data controller where it manages and sends it to the correct location. In booking management, we handle the booking process, including seat selection, reservation, and confirmation. In payment processing, we handle payments, ensuring secure transactions and handling user interactions with our payment interface. In the inventory database, we hold available information about what movies are being shown and when, along with a database of our user's account information.