MoviePlug Ticketing System

Software Design Specification

Version 1

October 5, 2023

Group 5

Ruben Guillen

Justin Pudiquet

Rosa Lisa Silipino

Prepared for

CS 250- Introduction to Software Systems

Instructor: Gus Hanna, Ph.D.

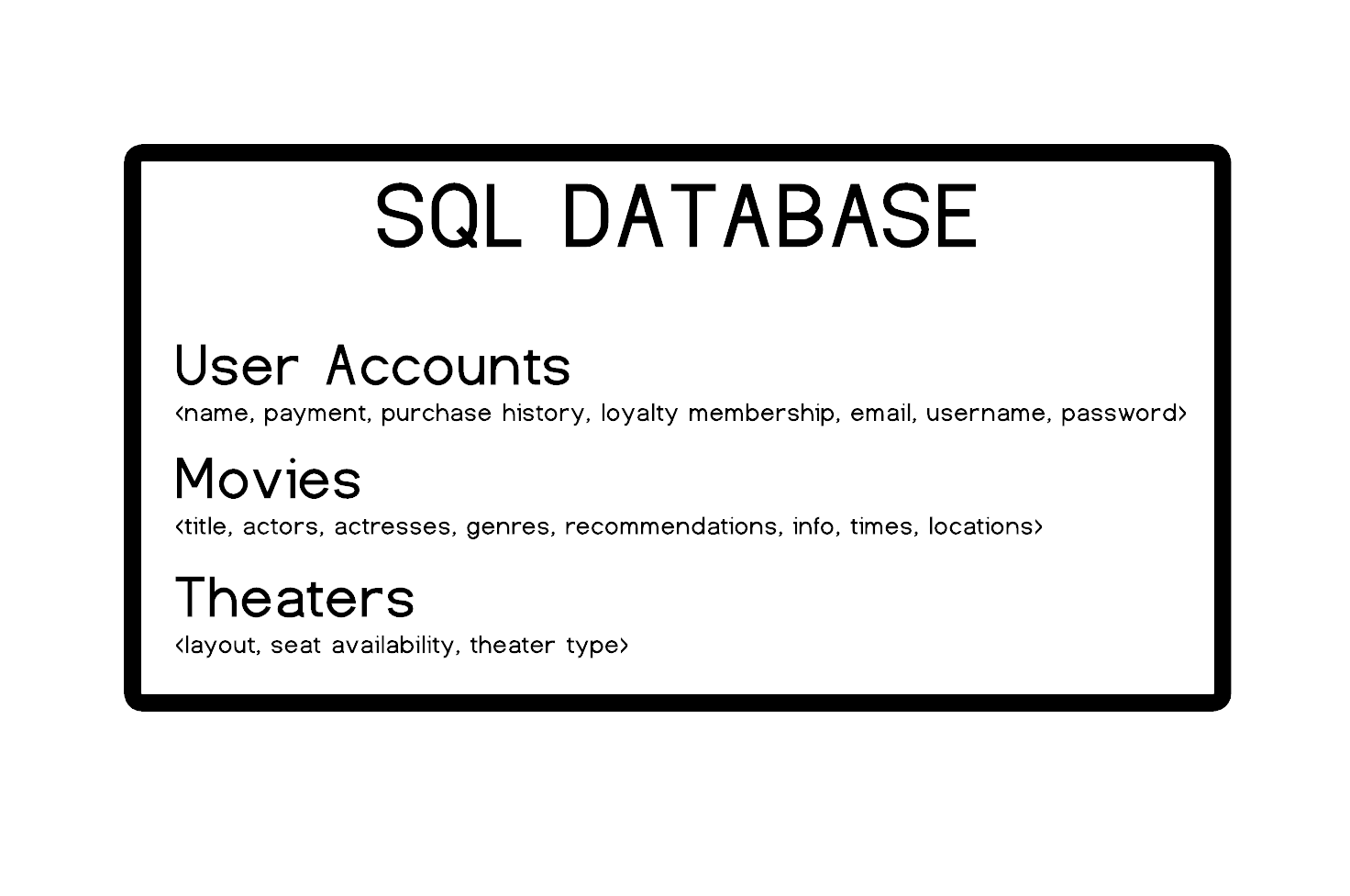
Fall 2023

# System Description

The purpose of MoviePlug Ticketing Systemis to ease customers' movie ticket purchasing experience. By providing a convenient online web-based platform that essentially acts as a hub for customers purchasing movie tickets. It has access to 20 partnered theaters across the San Diego region offering access to both regular and deluxe theater seating reservations. The system shall interface with several APIs and a relational database that streamlines processes such as movie searches, ticket purchases, seat reservations, payment, and reviews. Customer accounts are available and loyal customers reap rewards through their loyalty points saved to their account, with their commitment to our product. Overall, the system will provide a user-friendly portal for movie-goers that offers the ability to skip long lines on movie night.

# Software Architecture Overview Architectural Diagram

## SWA Description



## UML Class Diagram

## 

## Description of Classes

* *User() - Contains user information and validates information with DB.*
* *Booking() - Contains booking information like booking id, the user who is booking and screenings.*
* *Payment() - Contains user input such as card information and uses 3rd party API to process payment, and transaction information.*
* *Ticket() - Contains all ticket details such as user name, booking confirmation, and assigned seat if applicable.*
* *Seat() - Contains the user designated seat provided that the theater in question is an assigned seating theater.*
* *Theater() - Contains theater branch information such as name, theater id and capacity among other things.*
* *Screening() - Contains information in regards to actual screening date, time, theater location, film name.*
* *Movie() - Contains actual movie information such as title, genre, date released, and movie identifier (internal purposes).*

## Description of Attributes

* User()
  + user\_id - string - user display name
  + username - string user login
  + password - string user password
  + email - string email address
* Booking()
  + booking\_id - string - booking confirmation
  + user - - strings - information coming from the User() class containing user info
  + screening - multiple data types coming from the Screening() class
* Payment()
  + payment\_id - string - payment confirmation id returned from 3rd party API
  + amount - int - total amount being charged for ticket(s)
  + payment\_date - int - date transaction was made in xx/xx/xxxx xx:xx:xx format
* Ticket()
  + ticket\_id - int - ticket id unique to theater, movie, and user
  + booking - multiple data types coming from the booking class
  + seat - string - passed from the Seat() class
* Seat()
  + seat\_id - string - composed from the theater information as well as seat number
  + theater - string/int - information passed from the Theater() class
  + seat\_number - int - seat number within a given row
  + is\_occupied - bool - flag that determines if seat is already taken
* Movie()
  + movie\_id - int - unique ID that corresponds to movie within theater (internal use)
  + title - string - movie title
  + genre - string - movie genre
  + release\_date - int - contains date of when the movie will be released in theaters in xx/xx/xxxx format
* Screening()
  + screening\_id - unique ID that is composed of movie information such as location, movie title, movie times (internal use)
  + movie - string - information brought over from the Movie() class contains movie
  + theater - object - contains theater information
  + date\_time - int - Date and time of the screening
* Theater()
  + theater\_id - int - Unique identifier for the theater (internal use)
  + name - string - Name of the theater
  + capacity - int - Maximum seating capacity of the theater

## 

## 

## Description of Operations

* User()
  + Register() - Registers user from user input information
  + Login() - Using user input information validates credentials with DB information
* Booking()
  + Add\_Ticket() - Adds ticket to the purchase, marks seat as unavailable (if applicable), updates theater capacity, and will queue for printing at a later step.
* Payment()
  + Process\_Payment() - 3rd party API that processes payment

# Development Plan and Timeline

## 

## Partitioning of Tasks

To ensure the four month timeline is met these have been assigned to the following teams:

1. **Planning:**
   1. **Project Manager** - POC: Justine Rosario Rubenero
2. **Research**:
   1. **UX/UI Team** - POC: Shanique Doodlebob
3. **Gather Requirements:**
   1. **Project Manager** - POC: Justine Rosario Rubenero
   2. **UX/UI Team** - POC: Shanique Doodlebob
4. **Design:**
   1. **UX/UI Team** - POC: Shanique Doodlebob
5. **Build:**
   1. **Front-End Development Team** - POC: Terrance Clifford III
   2. **Back-End Development Team** - POC: Chris P. Baycoon
6. **Testing:**
   1. **Front-End Development Team** - POC: Terrance Clifford III
   2. **Back-End Development Team** - POC: Chris P. Baycoon
   3. **QA Team** - POC: Naomi Yerba
   4. **Tester Team** - POC: Wallace Steffingshire
7. **Final Review:** 
   1. **Project Manager** - POC: Justine Rosario Rubenero
   2. **Software Architect** - POC: Davy Jones
   3. **QA Team** - POC: Naomi Yerba
8. **User Testing:**
   1. **UX/UI Team** - POC: Shanique Doodlebob
   2. **QA Team** - POC: Naomi Yerba
   3. **Tester Team** - POC: Wallace Steffingshire
9. **Refining:**
   1. **Software Architect** - POC: Davy Jones
   2. **Front-End Development Team** - POC: Terrance Clifford III
   3. **Back-End Development Team** - POC: Chris P. Baycoon
10. **Launch:**
    1. **Project Manager** - POC: Justine Rosario Rubenero
    2. **Software Architect** - POC: Davy Jones

## Team Member Responsibilities

* **Project Manager**: Justine Rosario Rubenero
  + Develop a project plan and lead product planning meetings with teams.
  + Develop functional specifications.
  + Plan, estimate, and manage resources and the project budget.
  + Conduct initial interviews with stakeholders.
  + Evaluate and ensure timeline processes.
  + Launch and release.
* **UX/UI Team |*****UX/UI Lead***: Shanique Doodlebob
  + Conduct user interviews.
  + Design website wireframe.
  + Research competition.
  + Gather Requirements
* **Front-End Development Team | *FE Lead*:** Terrance Clifford III
  + Build the features laid out in the Sprint.
  + Update the status of the software project to the Project Manager and Lead.
* **Back-End Development Team** | ***BE Lead:*** Chris P. Baycoon
  + Integrate web services and APIs.
  + Build servers, frameworks, databases.
  + Integrate with front-end capabilities.
* **QA Team | *QA Lead:*** Naomi Yerba
  + Evaluate execution of processes and/or production.
  + Identify and document deviations in SOPs.
  + Provide feedback about the results of the quality assurance.
* **Tester Team | *Test Lead:*** Wallace Steffingshire
  + Understand requirements, and constraints to create and run test cases.
  + Build and run test cases to detect bugs and report them.
* **Software Architect:** Davy Jones
  + Define the technical and functional architecture of the overall system.
  + Guide developers in the design and implementation of the solution.
  + Ensure functionality of the final product.