Software Design Specification

Theatre Booking System

Team Lead
Stephen Villanueva

Team Members
Giuseppe Scalise
Robert Leal
Trey Allen

Table of Contents

1. Introduction

1.1 Purpose

This design will detail the implementation of the requirements as defined in the Software Requirements Specification for Theatre Booking System.

1.2 System Overview

This system will be an online interface used for purchasing tickets. The system will have multiple types of users, those being administrators and customers. A user must go through a registration process before performing any actions. A customer user will be capable of purchasing tickets, and an administrator will be capable of uploading, removing, and modifying plays. These plays will have a limited number of seats, and consequently, a limited number of tickets available. The system will be able to identify the current capacity of a play, the remaining number of seats, the price of a ticket, the duration of a play, and several more characteristics, including genre and title. The system will be able to process transactions regarding the purchase of tickets, as well as potential refunds. The system will utilize PHP and JavaScript, with a database in SQL.

1.3 Design Map

Contained in this Software Design Document, the reader will find a description of the system architecture, as well as the design and motivation for methodology of the system's components and methods. The database schema, W.I.P. user interface, and requirements matrix can also be found.

1.4 Definitions and Acronyms

PHPMyAdmin - 3rd party database software
MySQL - 3rd party database software

2. Design Considerations

As there will be two different types of users, there will be one parent class called "User", which is inherited by two subclasses, "Administrator" and "Customer". These users will have common shared functions, like functions used in the retrieval of user information, identification, and login.

2.1 Assumptions

The physical theatre location will have internet access, as well as a connection to the webserver from which our system is hosted on.

2.2 Constraints

None that we are aware of.

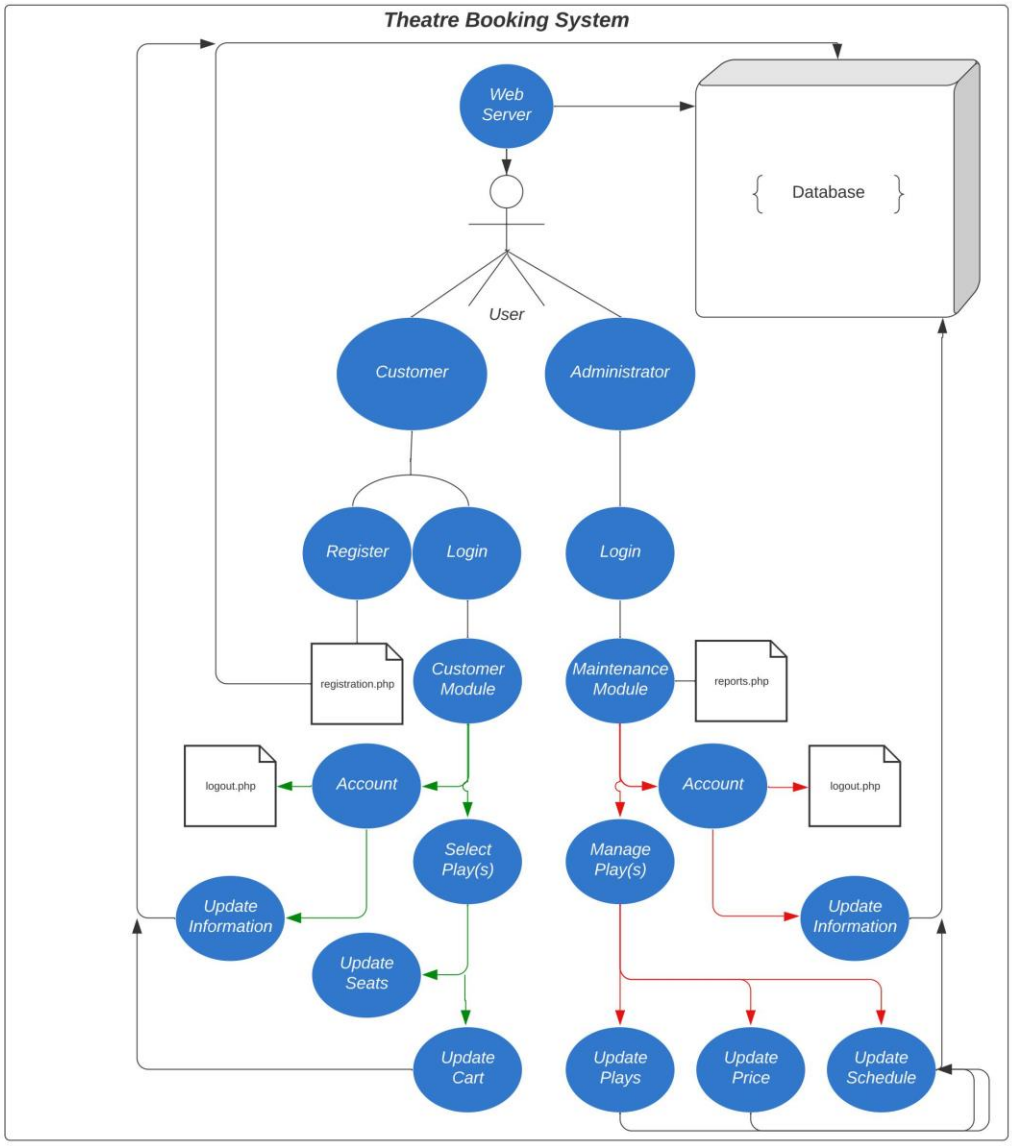
2.3 System Environment

The system will be contained in a webserver, which is also where the database is contained. The system will primarily be written in PHP and JavaScript, and will be accessed from the client at the workplace. The database used to store data will be MySQL and PHPMyAdmin. Employees at the workplace will connect to the webserver via desktop computers.

2.4 Risks and Volatile Areas

The system will be storing sensitive information, including credit-card numbers and addresses. Although the system has implemented protection against database infiltration in several ways, including input sanitization, it is worth noting that sensitive customer information and data are being stored – and furthermore, protected.

3. Architecture



3.1 Overview

The system's architecture follows a client-server model. The system's architecture can be accurately described by it's use-case diagram, found on page 3. The user first connects to our webserver, which houses our database. Our database interacts with our existing codebase using PHPMyAdmin and MySQL. Upon connection to the webserver, the user will login as either a customer or administrator – both of which are subclasses of parent class User. Depending on which type a user is logged in as, they will be directed to different modules. These modules are Customer Module and the Maintenance module. From here, customers can select plays, and administrators can manage plays. Additionally, both types of users can choose to update their account information -- interacting with our database. Upon customer interaction with plays, they can choose to purchase tickets by updating their cart. Administrators can choose to update available plays, update prices, update play scheduling, and possibly refund users. Both types of users will interface with a file called "logout.php", which will allow the users to log out.

The major responsibility of both users is to inevitably interact with our database. The Customer user achieves this by updating their account information, viewing a list of updated seats, and by updating their cart and making purchases. An Administrator user achieves this by updating their account information, and by managing plays.

3.2 Components

3.2.1 Customer User

The customer will be inherited from the parent User class. As mentioned in 3.1, the Customer will be able to edit their account information, select plays to view, and purchase tickets.

3.2.2 Administrator User

The administrator will be inherited from the parent User class. As mentioned in 3.1, the Administrator will be able to edit their account information, select plays to modify, and potentially refund tickets.

3.2.3 Plays

Plays will belong to their own class, where they will be given attributes such as genre, title, length, and ticket prices. Administrator users will perform actions on these plays, such as updating their ticket price or availability.

3.2.4 Cart

The shopping cart will belong to its own class, where Customer users will be able to perform actions on it, including adding tickets to the cart, removing tickets to the cart, clearing the card, and purchasing.

3.2.5 Tickets

Tickets will belong to their own class. Objects of this class will be assigned to Customers upon purchasing a ticket.

3.3 Module and Component Strategies

For the system, an object-oriented approach is being utilized. This appeared to be the most effective approach, as many of the system's functional requirements are similar between different components.

4. Database Schema

4.1 Tables, Fields and Relationships

Our tentative list of tables includes USERS, TRANSACTIONS, PLAYS.
The USERS table will have fields UserID, Admin, CardNumber.
The TRANSACTIONS table will have fields UserID, PlayID, TicketID, Date, TicketPrice
The PLAYS table will have the fields PlayID, Genre, Title, Length, Tickets, TicketPrice.

The USERS and TRANSACTIONS tables will share the UserID field.
The TRANSACTIONS and PLAYS tables will share the PlayID and TicketPrice fields.

4.1.1 Databases

LosPortalesDataProduction, LosPortalesDataTest for development and testing.

4.1.2 New Tables

Tables USERS, TRANSACTIONS, and PLAYS, as described in 4.1.

4.1.3 New Fields(s)

Table Name	Field Name	Data Type	Allow Nulls	Field Description
USERS	UserID	Varchar(8)	No	Unique ID for each distinct user.
USERS	Admin	Bit	No	If Bit is 0, the user is a customer. If 1, the user is an administrator.
USERS	CardNumber	Varchar(16)	Yes	If the user accepts, this field will store their card number.
TRANSACTIONS	TicketPrice	Float	No	The price in dollars for any given ticket
TRANSACTIONS	UserID	Varchar(8)	No	Unique UserID for each distinct user.
TRANSACTIONS	PlayID	Varchar(8)	No	Unique ID for each distinct play.
TRANSACTIONS	TicketID	Varchar(16)	No	Unique ID for each ticket sold.
TRANSACTIONS	Date	datetime	No	The date and time that each ticket transaction took place.
PLAYS	PlayID	Varchar(8)	No	Unique ID for each distinct play
PLAYS	Genre	String	No	Genre for any given play
PLAYS	Title	String	No	Title for any given play
PLAYS	Length	Float	No	Length in minutes of any given play
PLAYS	Tickets	Int	No	Number of tickets offered for any give play
PLAYS	TicketPrice	Float	No	The price in dollars for any given ticket

4.1.4 Fields Change(s)

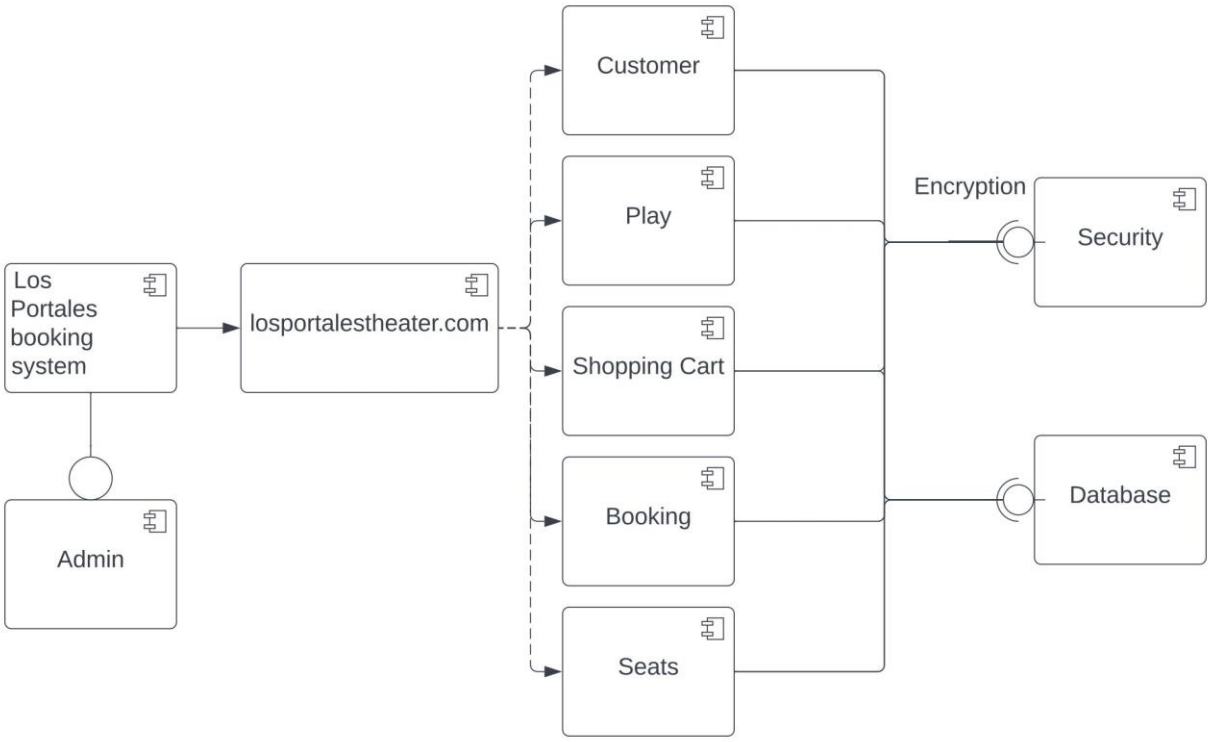
No field changes required.

4.1.5 All Other Changes

Currently, no other database changes needed.

5. Component Design

5.1 Component Diagram



5.1.2 Class Hierarchy

The system implements inheritance with the two user types. There is a parent class User, which is inherited by subclasses Customer and Administrator. The remaining classes belong to their own classes; these classes include Ticket, TheatreHall, Play, and Cart.

5.1.3 Object Functions

Objects of the class User will have a function for registering new users, called registerUser().

Objects of the classes Customer and Administrator will inherit this registration function. Customer objects will have the ability to book tickets, with a function called setBooking(). Administrators will be able to modify plays and users, with functions setTitle(), setGenre(), setDuration(), setPlayID(), addPlay(), removePlay(), updatePlay(), setPlay(), updateCustomer(), and updatePlay().

Objects of the class Ticket will be constructed to have unique attributes, including TicketID, SeatNum, Price, Date, Time, and Play. These attributes will change depending on the price of the ticket, the date and time of the play, and the play for which the ticket is assigned to.

The class TheatreHall represents rooms in which plays are presented. Objects of the class TheatreHall are capable of displaying how many seats remain in a play.

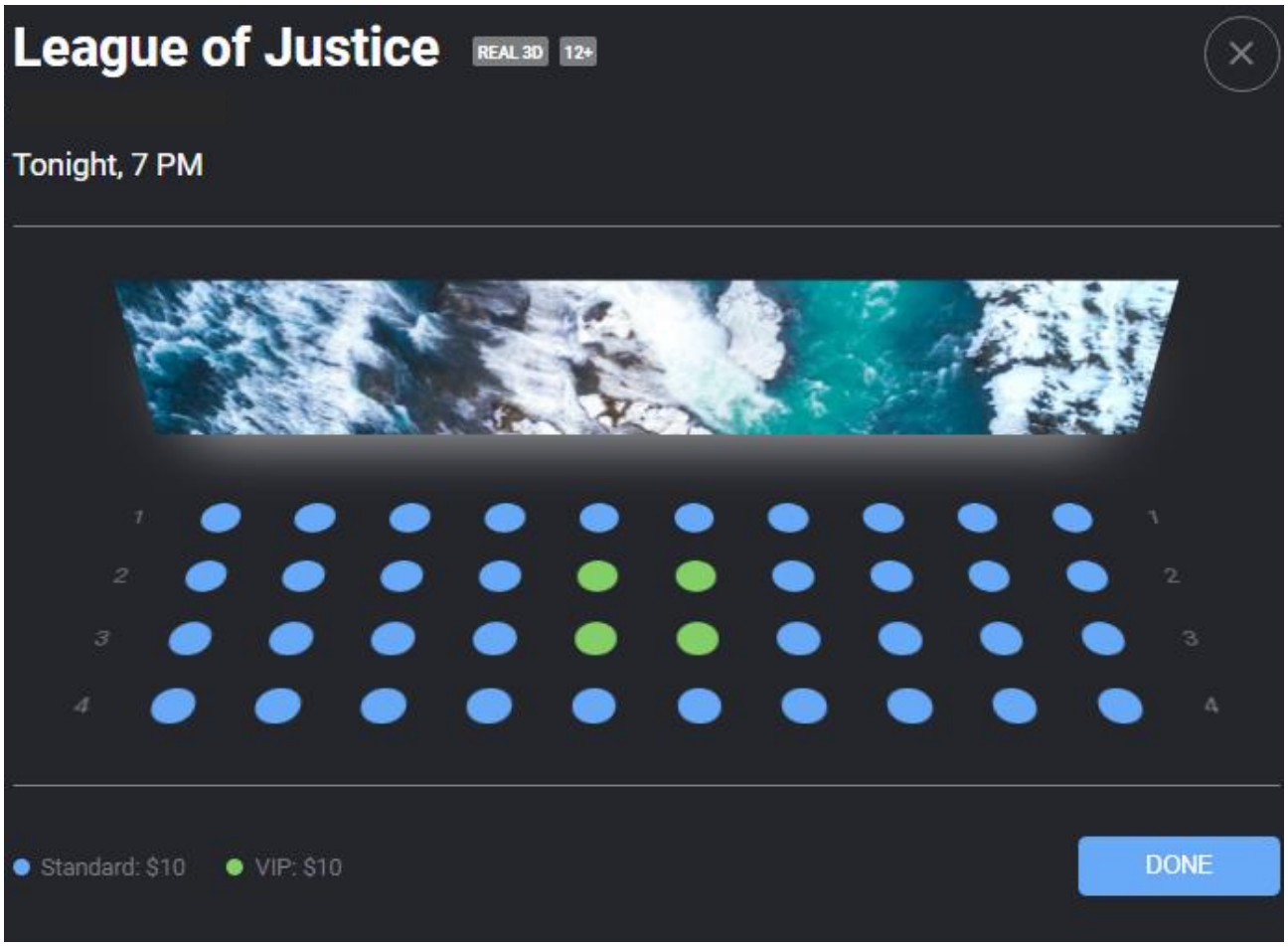
6. Human Interface Design

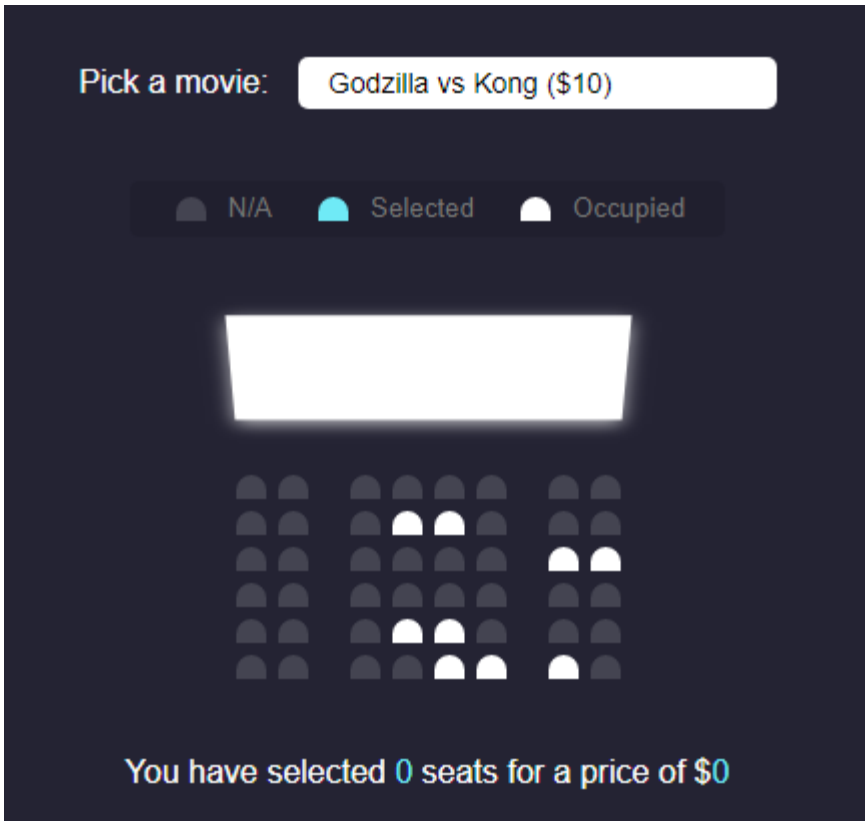
6.1 Application Controls

Each webpage will share a navigation bar at the top, which have the theatre logo, and several redirects, including to the homepage, to the “About” page, to the “Contact” page, and to login or logout, depending on if you are logged in or not.

6.2 Major User Interface Screens

Pending design finalization. Progress has been made on an interface for seat selection; different options shown below.




Pick a movie:
Godzilla vs Kong ($10)
N/A
Selected
Occupied
You have selected 0 seats for a price of $0

7. Requirements Matrix

Project Name:		Los Portales Theatre	
Reviewer/Approver		Stephen Villanueva	
Traceability #	Requirement ID	Technical Requirement ID	Test Case ID
2.2.1	2.2.1	N/A	N/A
2.2.2	2.2.2	N/A	N/A
2.2.3	2.2.3	N/A	N/A