Team 4

Jeru Sanders, Garry Callis, Stephen Washington, and Michael Gonzales

Project: Stage 2

Software Design Document

March 27, 2022

Table of Contents

Introduction

Purpose

The purpose of this project is to create software for the theatre "Los Portales." In this document we will go through an in-depth explanation of the system architecture, data design, and component design.

Scope

As previously mentioned, the purpose of this project is to create software for the theatre "Los Portales." Listed below are the requirements for the project.

- Display a screen with 8 rows and 12 seats per row
- If seats are available, the seats will be shown in green
- If seats are unavailable, they will be shown in red
- Admin must be able to add plays during the year based on dates and times
- Admin must be able to change the price of seats (certain seats can have different prices)
- Admin can change price by selecting one or more seats and changing them
- Registered customers can buy seats for any play (ex: name, age, address, telephone, and email)
- System displays seating for future plays
- Customer can click on previously selected seat to remove it from their list of seats
- Customers can add a list of selected seats to a shopping cart
- Customers can move from play to play to buy more seats
- When customers check out the systems will display:
    - Total due
    - Report on transactions
        - Transaction ID (consecutive number)

        - Name of play or plays
        - Dates/Time
        - Seat numbers
- Customer needs to make changes to shopping cart
- System must ask customer for Creditcard number and simulates the sale.
- System must inform user the sale was correct.
- An option to the customer, the system must send an email with purchase details
- Admin can generate a report with
    - How many seats were sold for a specific play and date
    - Requires username and password to access the management area
- System must be online and can be used with mobile devices

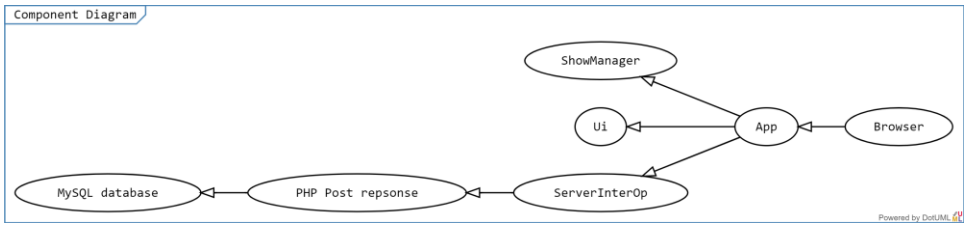
I hope this introduction has provided you with a general idea of the purpose and requirements of the project.

System Overview

To further your knowledge of the project development, we will briefly give an overview of the rest of this document. The next section, System Architecture, will discuss the modules that make up the client-side application, the properties each module contains, the functions that govern the behavior of the modules, and relationship between them and the functionality of the system. The section following that, Data Design, will go over specifics involving what data is stored, the format of the data, the format of the queries that the application will make, and the general structure/layout of that database. Human Interface Design will go over how the application is experienced by the user's perspective, both normal customers and administrators. There are also sketches to approximate what we expect the screens to look like.

System Architecture

Architecture design



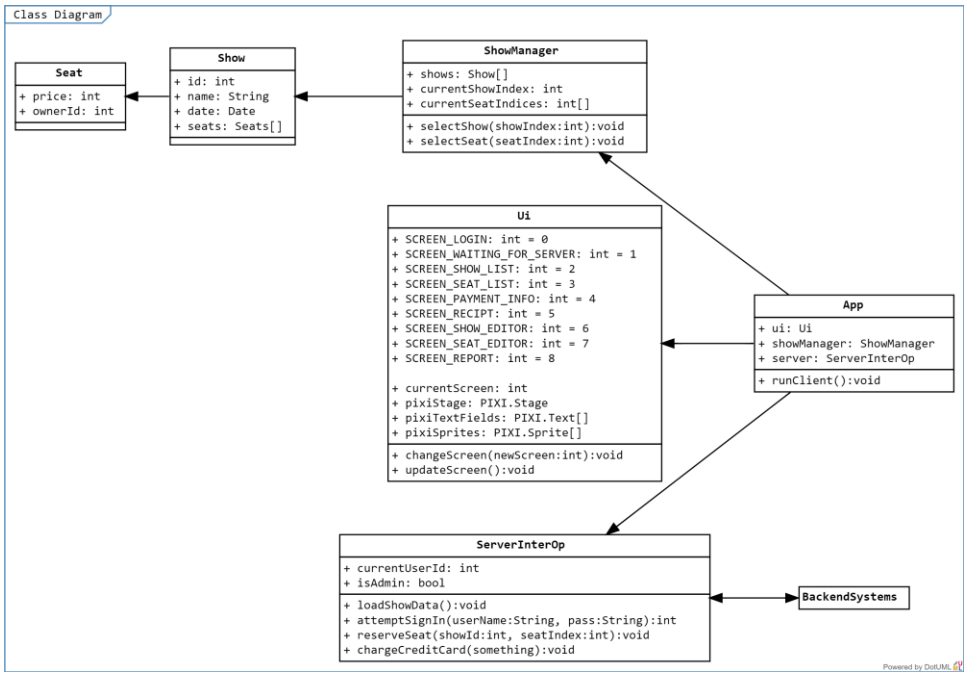
This is the client-side application for the Los Portales ticket sales platform.

The client-side application is written in JavaScript and will run in the user's web browsers.

Both customers and admins are users.

Customers will use this application to view current available shows and seats and purchase them. Admins are the owners are the venue and will log in to add and adjust shows.

The client will be a single page application written in "vanilla" JS prominently using the library PixiJs.



The following is a list of screens and their functions:

NONE	A sentinel screen
LOGIN	The screen where customers or admins log in
WAITING_FOR_SERVER	A screen for the client to wait for a server response on
SHOW_LIST	Shows all the available shows in a list (icon|title|date)
SEAT_LIST	Shows a grid of seats where the user can select multiple seats by touching/clicking them, then purchase them by clicking a buy button. Available seats are in white, unavailable ones are grey, selected are yellow
PAYMENT_INFO	Shows a screen where the user inputs their credit card, CSV, and exp date, then clicks buy. There should be visual icons of each of the major card providers, with all but MasterCard and VISA crossed out. (TODO: Figure how know what provider a card came from) (TODO: Figure out how to avoid two people paying at the same time for the same seat) Optionally: User can save or pick a saved card Optionally: Cash payment
RECIPT	Just shows the seat and show you paid with a green check mark or something. Has a button to go back to the show list.
SHOW_EDITOR	A screen that admins get to from the seat list, that allows them to modify the show name, date, icon. (TODO: How do get icons???)
SEAT_EDITOR	A screen that admins get to from the seat list, that allows them to modify the selected seats price.
REPORT	A screen that dumps a huge filterable log of all the events

Once the user logs on or chooses to browse as a guest, the client will need to download the latest show and seat availability data. This will happen by calling *ServerInterOp.loadShowData()*, this function will make an XMLHttpRequest to a page server.php?action=load, this page will reply with all show and seat data, the layout of which has not yet been determined (CSV strings).

Components Description

The app will consist of 4 global singletons.

- *Ui*: The module in charge of drawing to the screen and processing user input into actionable commands. Solely in charge of PixiJs.
- *ShowManager*: The module in holding the downloaded list of show and seats, all client-side show/seat data is held here. (This will probably be renamed to "Cart" later, when that functionality is documents)
- *ServerInterOp*: The module in charge of interacting with the server, downloads the show/seat data and stores it in the *ShowManager*, sends update and verification commands to the server.
- App: Just a structure that holds all 3 of the other modules.

Each of these singletons will be initialized during an *onload()* callback within the main page's header. The relevant code for the singletons will be loaded from client.js and *runClient()* will be called.

- The *runClient()* function will simply start the update loop, further initialization will happen upon the first iteration of the update loop.

The update loop contains one giant "switch/case-like" if-statement for each entry in the SCREENS enum, the ui module contains the *currentScreen(ui.currentScreen)* that dictates which branch is taken. Each branch is responsible for the display and input processing for its screen. Just before the branch is taken a boolean *firstFrameOfScreen* is declared and set to true if *ui.screenTime==0*, *ui.screenTime* is incremented at the end of each update loop.

Design Rationale

The intention is that each screen uses *firstFrameOfScreen* being true as a signal for it to call ui creation functions like *createText(),* and *createButton()*. Then, monitor the state of the created elements for user interaction.

When user action occurs, or some external event is received, *changeScreen()* can be used to change to a new screen. *Ui* elements do not need to be cleaned up, *Ui* elements will be automatically removed at the top of the update loop.

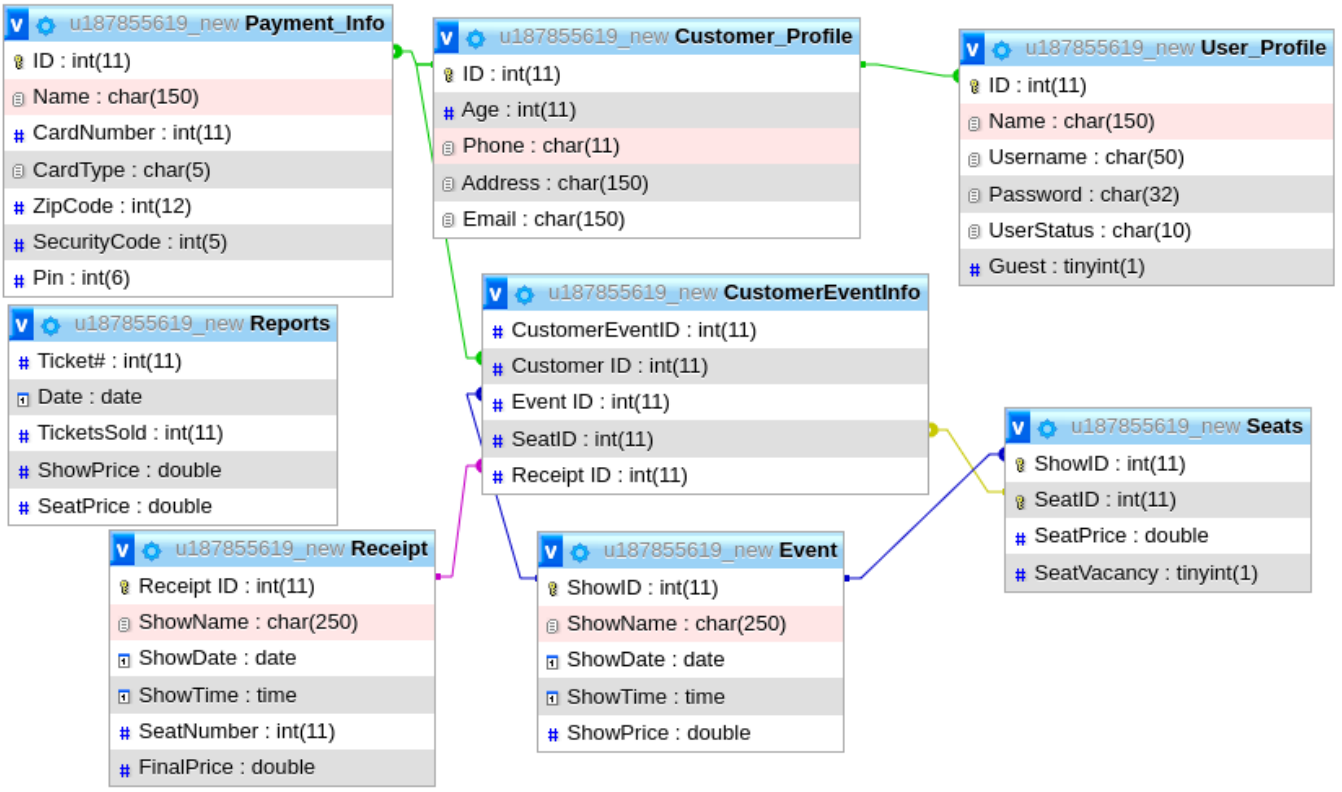
Database Schema

Tables, Fields, and Relationships

The database will be a MySQL database, consisting of 8 tables. The tables will be labeled as the following: User Profile, Customer Profile, Payment Information, Reports, Receipts, Events, Seats, and a customer event summary. The user table will be used for the login screen. It will share a one-to-one relationship with the customer table, so that after the login screen the username and password information can be linked to the customer profile. Customer profile will have more details about the customer that may aid them in their purchase. It will share a one-to-many relationship with payment information and customer event summary. Payment information will have credit/debit card information saved to aid the customer in their purchase by making payment information quickly accessible. The receipt table will store data after purchase about the customers seat, event, and show time. The receipt table can also be used to generate a ticket so that the customer has something to show the staff at the theatre when they arrive at the show. The event and seat tables will have specific information about the seats available for each show. Both share a many-to-many relationship with the customer event information table. The customer event information table is meant to be a central point for data about a particular show for a customer. This can be used to access data from multiple tables, so that all the information about a particular event for a customer is in one table. This table has a many-to-many relationship with multiple tables previously stated.

Databases

There will be two databases, a theatre_los_portales database, and a theatre_los_portales_test database. The theatre_los_portales_test database will be used for development and testing. Below are representations of the tables and their relationships for theatre_los_portales.



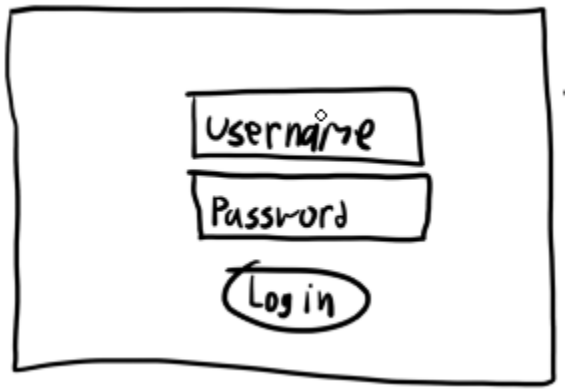
Files

We will use PHP files to access, edit, and store data on the database.

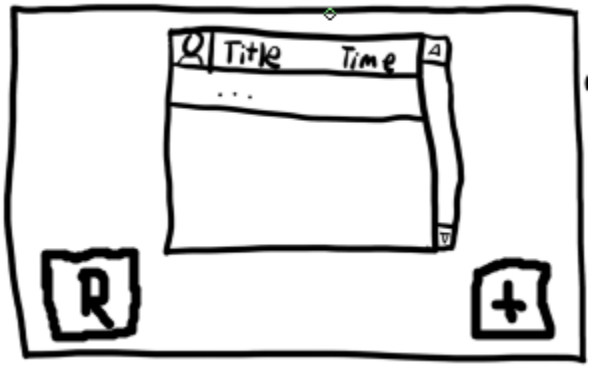
Human Interface Design

Customer's perspective:

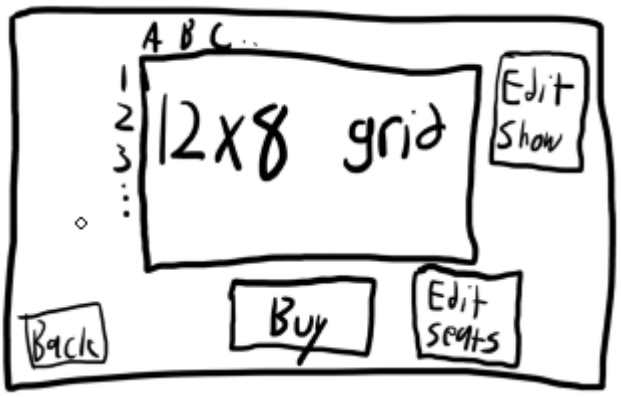
- Upon opening the app, the customer is prompted to log in to their account, they can either log in, or press a "browse as guest" button to proceed.



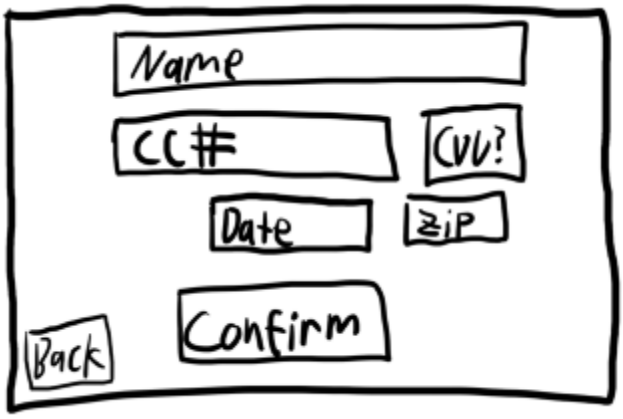
- After that the customer sees a list of all the shows that are scheduled for the future. The customer can scroll up and down this list and touch an entry to view the seat availability.



- On the seat listing page, the customer sees a grid of seats, color coded to their availability. Green=available, Gray=taken. The customer can touch a seat to select or deselect it, selected seats glow yellow, taken seats cannot be selected. Once the customer has selected 1 or more seats, a purchase button appears at the bottom of the screen.



- When the customer clicks this button, they are taken to the payment screen where they are shown their selected seat numbers, the price, and prompted to provide their credit card information, as an extra feature, we might add a "save card" feature.



- Once the customer confirms their CC information, a request is sent to the server to process the transaction and mark the seat as taken. The server relies on the client's application, with whether the transaction was successful or not.

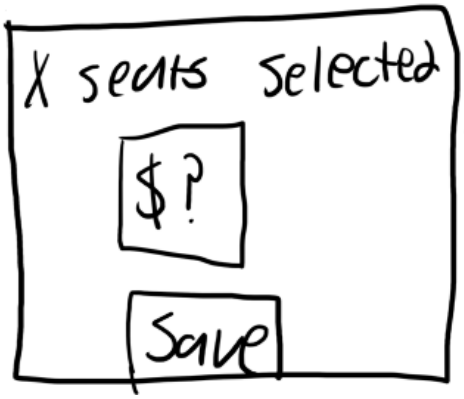
- The transaction may fail if the CC does not have the appropriate funds, or the seat was just taken. @todo Purchase failure screen
- If the purchase was successful, the customer is taken to the receipt screen where they are shown information about their purchase.

Admin perspective:

The admins perspective is mostly identical to the customers' perspective. The main difference is that on the seat grid screen, instead of a purchase button, admins see an "Edit seats" button, that takes them to a screen where they can set the price of the selected seats.

- They all see an "Edit show" button that allows them to edit the name and date of the currently selected show.

- On the show listing screen, they have an "Add show" button, that creates a new show, and moves them into the "Edit show" screen for that show. @todo Make a way to delete shows?



- Also on the show listing screen, they have a "Reports" button that takes them to a screen where they can see a filterable list of purchase history and information.