Web development course 2019

# MSc Advanced Computing Systems (2019 class)



Final Assignment :

Web development semester assignment

**Post-graduate Students:**

|  |  |
| --- | --- |
| **Full Name** | **AM** |
| Manos Adamopoulos | 18001 |
| Nikolaos Benakis | 18016 |
| Panagiotis Onasis | 18028 |

**­**

**Purpose**

The aim of this assignment is to develop an end to end dynamic web application. This specific application is an online booking system for cinema seats and tickets. It provides a seamless experience to the user, following the best practices for the best possible user interface and user experience.

**Technology stack**

The application stack is :

**DevOps/ deployment platforms, PAAS:**

* digital ocean, docker, azure, surge

**Database:**

* mySql

**Backend:**

* .net core 2.0.2 / Entity Framework
* C#

**Frontend:**

* React.js 16.8, HTML5/SASS, bootstrap 4, jest, webpack
* CDN cloudinary

**Design Patterns Implemented:**

* **Backend**: dependency injection, pub sub, repository, factory
* **Frontend**: pub sub, Hocs, hooks

**Requirements/ use cases:**

Popcorn is an online cinema booking system that has a schedule calendar with available tickets and reservations, a seamless web system to book seats and tickets. Also there is a subsystem admin panel for actions like create, edit, delete, update users/roles/auditoriums/movies . All the data are stored in our database hosted in cloud digital ocean. Our system fulfills three user types.

1. Admin
2. Superuser
3. User
4. Visitor

Each user role has it’s privileges. For instance, a visitor can only access movie theaters, movies, movie showings, search for reservations and sign up/ sign in. On the other hand super user can update our data but has no privilege to change user roles, no one does but admin. Admin has all the privileges to update, create, delete our data.

**Visitor - user journey:**

In this section , we describe a typical user flow in our booking system.   
Given the visitor is at the home page and clicks on sign in button at the navigation bar , then a popup modal appears. This modal has a form with two validated fields email, password. If visitor has already account by pressing the login button he can access the application as a user or superuser or admin. If he does not has already account, he can click on the register section on Modal’s footer . Then a register modal appears with the following fields : first name, last name, email, password, birthdate, upload photo. After he fill the validated fields and submits the form then if everything is validated he can access the application as a user.

Following the authentication flow , then the user can click to a movie card and then a modal appears with more details given for the movie. User interacts with modal by closing it or by pressing the book button. Following the latter scenario the user is navigating in a booking screen. This booking screen shows the available theaters sorted by the movie showings of the selected movie. User has the option to click in the theater card button named schedule in order to see the calendar of the movie showing. Then the card is flipped and a calendar appears in the shape of a table card. User can choose a specific date, by clicking on this date a seats scheme will begin to load in the screen. After the load time, a seats scheme is ready to inform the user about which seats are already reserved and which are available for future reservations. After the user seat choice , user can press the submit button and then a confirmation modal appears. The confirmation modal informs the user about the previous steps he has completed, for example the movie title, date, seats information , price. The price is standard for every seat. Confirmation modal can either be closed or submitted. If submitted then a payload request is sent to our application server and the client is waiting for the reservation status. If success , then a success modal with green positive image appears to inform the user about the reservation status. Also the user can download a pdf with reservation details and he has the option to send this pdf via email to his friends. If our request failed or something went wrong during the reservation submission then a red flag appears and informs the user what went wrong about the reservation pipeline.  
User can also edit his personal data via myProfile page. There he can edit the data that he submitted upon registration form such as birthdate, names, email, password etc.

User or visitor via navigated searchReservation page, he can search a reservation by email or by phone. There he can check if there is any reservation made by him or his friend. If the results are valid then a card with reservation details appears among with personalized message about the user.

**Superuser - admin journey**

Super user and admin extend the user journey . More specifically they have access to a subsystem named adminPanel. They can create, update, delete everything about the movie showings, auditoriums, movies, users(only admin). They can also collect some data about the total reservation by all users.

The following screens illustrate the above journey descriptions.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A picture containing monitor, screenshot

Description automatically generated A screenshot of a cell phone

Description automatically generated

**A screenshot of a computer

Description automatically generated  
  
A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated  
A screenshot of a cell phone

Description automatically generated** **A screenshot of a cell phone

Description automatically generated** A screenshot of a cell phone

Description automatically generated **A screenshot of a cell phone

Description automatically generated  
  
A screenshot of a computer

Description automatically generated**

**Following the assignment bullet points this is our USE CASE UML script**

* **@startuml**
* **title PopCornCinema Use Case Diagram**
* **left to right direction**
* **'skinparam packageStyle rectangle**
* **actor visitor**
* **actor customer**
* **actor superuser**
* **actor admin**
* **'rectangle System {**
* **visitor --> (sign up)**
* **visitor --> (view movies)**
* **visitor --> (view auditoriums)**
* **visitor --> (view movieshowings)**
* **visitor --> (search for reservation by email)**
* **superuser ---> (admin log in)**
* **(admin log in) <--- admin**
* **(show booking reports) .--> (admin log in) :extend**
* **(add movieshowing) <--. (admin log in): include**
* **(update movieshowing) .--> (admin log in):extend**
* **(delete movieshowing) .--> (admin log in):extend**
* **(add movie) <--. (admin log in): include**
* **(update movie) .--> (admin log in):extend**
* **(delete movie) .--> (admin log in):extend**
* **(add user) <--. (admin log in): include**
* **(update user) .--> (admin log in):extend**
* **(delete user) .--> (admin log in):extend**
* **(add auditorium) <--. (admin log in): include**
* **(update auditorium) .--> (admin log in):extend**
* **(delete auditorium) .--> (admin log in):extend**
* **(log in) .> (authentication) : include**
* **(admin log in) .-> (authentication) : include**
* **(log in) <--- customer**
* **(log in) .-> (select movie) : include**
* **(log in) .-> (select auditorium) : include**
* **(log in) .-> (select date) : include**
* **(log in) .-> (select time) : include**
* **(log in) .-> (select seats) : include**
* **(log in) .-> (confirm ticket) : include**
* **(confirm ticket) .-> (print ticket) : include**
* **(confirm ticket) .-> (email ticket) : include**
* **'}**
* **@enduml**

**Following the assignment bullet points this is our Sequence diagram UML script**

* **@startuml**
* **participant Admin order 10**
* **participant SignedInUser order 20**
* **participant Visitor order 30**
* **participant Auditoriums order 40**
* **participant Movies order 50**
* **participant MovieShowings order 60**
* **participant BookMovieShowing order 70**
* **participant EmailTicketToOthers order 80**
* **autonumber**
* **'Admin -> Auditoriums : AddAuditorium()**
* **'Admin <-- Auditoriums : return status**
* **'Admin -> Auditoriums : UpdateAuditorium()**
* **'Admin <-- Auditoriums : return status**
* **'Admin -> Auditoriums : DeleteAuditorium()**
* **'Admin <-- Auditoriums : return status**
* **'**
* **'Admin -> Movies : AddMovie()**
* **'Admin <-- Movies : return status**
* **'Admin -> Movies : UpdateMovie()**
* **'Admin <-- Movies : return status**
* **'Admin -> Movies : DeleteMovie()**
* **'Admin <-- Movies : return status**
* **Admin -> MovieShowings : AddMovieShowing()**
* **Admin <-- MovieShowings : return status**
* **Admin -> MovieShowings : UpdateMovieShowing()**
* **Admin <-- MovieShowings : return status**
* **Admin -> MovieShowings : DeleteMovieShowing()**
* **Admin <-- MovieShowings : return status**
* **SignedInUser -> MovieShowings : GetAllMovieShowings()**
* **MovieShowings <-- SignedInUser : ReturnAllMovieShowings**
* **SignedInUser -> SignedInUser: LogIn()**
* **activate SignedInUser #DarkSalmon**
* **Visitor -> Visitor: Register()**
* **activate Visitor #DarkSalmon**
* **SignedInUser -> BookMovieShowing : BookMovieShowing**
* **activate BookMovieShowing #DarkSalmon**
* **deactivate SignedInUser**
* **deactivate Visitor**
* **SignedInUser <-- BookMovieShowing : Confirmation**
* **deactivate BookMovieShowing**
* **SignedInUser -> EmailTicketToOthers : Write emails**
* **SignedInUser <-- EmailTicketToOthers : Success**
* **@enduml**