Project Report SEP 4

# Abstract

Cinema Web Page is a web page which enables user to look through movies which will be played and after registration in our system the user is then able to book a seat for a specific movie and book parking if he wishes to. User will have overview of his bookings with aim for managing these reservations. Our web page also supports role of administrator. Administrators are able to create a movie and has an overview of all users which are currently in the database and also is capable of alternating of the users and deleting them also he is capable of overviewing all parking reservations and delete user reservation of an parking place.

# Introduction

In the current age it is hard for small cinemas to compete with huge multiplex cinemas. Modern cinemas must have web page to compete because majority of people are searching on the internet for free time activities. Our Cinema Web Page is supposed to be solution for this issue. It allows our registered users to book a movie and parking so they can just come to the cinema and enjoy they free time without any worry for lack of parking places and sold out movie projections.

# Analysis

Before the project start we made an internal analysis and created use cases. These use cases are created in regard of our goal which we presented in the Introduction. These use cases reflect suspected behaviours of our three types of actors.

* Visitor: is a non-registered user which has limited possibilities. Visitor is only available to observe movies which are played and register to become a user.
* User: is a registered person in our database. This user can log in and observe played movies, but he can also reserve seats in specific movie and reserve parking for himself. User is also available to see all his reservations of parking and movies.
* Administrator: is special type of User. Administrators are not created through registration but are manually inputted into the database. Administrator upon logging in can see all the users and is able to manipulate with them (change user or remove him), he can also create movie as well as modify it or remove it. Lastly, he can see all parking places and remove user from reserved parking place and make it available.

# Design

# Choice of technologies

In Cinema Web Page we were trying to use the best available technologies for web development. Bellow we will list all technologies we used with reasons why we choose them over other possibilities. This list will be divided into two subcategories which are: Front end and Back end.

# Back End

In regard of back end programming language for a web page we could choose between function programming languages and object-oriented programming languages this choice will then affect other frameworks and databases we use:

* Functional programming languages: Are very powerful for web page application. They are lightweight and really fast but sometimes they have issues with data consistency and requires multiple frameworks to operate accordingly. In this field we were mainly considering using Node.js but for a such a short period of time it would be really challenging to set up this back end correctly.
* Object-oriented programming languages: These languages are on the other hand more heavy and in a lot of ways slower then properly set Node.js backend, but they support a lot of good frameworks, have great documentation and mainly they are easy to set up and are more conventional approach towards web page development.

After consideration we decide to use Object-oriented programming language. Here we could choose from Java or C#. In terms of efficiency they are similar, therefore decision was made upon “friendliness” of these system and what frameworks they support. With this in mind we choose to go with C# and .NET core framework.

# Programming language: C#

As said above we choose this programming language with regards of powerful frameworks we can use with C#, good documentation and better user friendliness of this language.

# .NET Core framework 2.1.

.NET Core framework 2.1. enables us to create good API with build in templates. This framework also contains methods to handle http request/response properly.

# Entity Framework

Because of the time limitation we had to choose database which is easy to set up and easy to maintain. Entity framework is perfect for this job because it allows us to create just models of database and Entity framework will create the tables and relations for us. Also, when we need to change some table we can just change the model and it will do the rest of the work for us. Another big reason is built in methods for queries which will make our work database much easier.

Also, Entity Framework support multiple types of database from proper persistent SQL databases to in memory database which great for development. In our project we used the In Memory Database because we anticipated that database will be changed quite frequently during development and therefore it’s fine for us when database will be recreated every time we start Back End.

If you would wish to run database in proper SQL server in *Startup.cs* there are comments to help you switch between these two.

# Front End

After we decided on Back end we had to decide how we going to build our front end. There are even more options then on the front end. Three main ways we could go is create front end with ASP.NET, use one of many JavaScript frameworks or combine these two approaches.

* with ASP.NET: This way would probably the most straight forward because we had already ASP.NET back end API. But there is a serious down site for this approach. First of all, it’s not widely used for this purpose and it is slow compared to JS frameworks.
* JavaScript Frameworks: They are really fast and with some knowledge of functional programming very user friendly and with lots of packages which can be easily imported.
* combination of two: created with ASP.NET templates for JavaScript frameworks (Angular, React). This combination is perfect for fast creation of web pages, but it is really challenging to make changes of the template.

We decided to work with JavaScript frameworks here we choose Angular 6 (earliest at the time) over React. We choose Angular over React even though it’s DOM is great for rendering pages and show the changes without necessity to reload page but to do that we would need to establish Redux for passing along data between components and it would take too much time.

At first, we tried to use ASP.NET with template for Angular but there is only template for Angular 5. We tried to update this template version to Angular 6, but it didn’t work for us. Therefore, we moved to stand alone Angular 6 front end.

# Angular 6

Angular is the most used JavaScript framework for front end. It has a lot of functionalities integrated into it where he also gets an edge over React with Redux. It also contains TypeScript which enables us to write JavaScript code with classes and interfaces, this is important for us because we have to learn how to write in Angular 6 on the way.

# PrimeNG

PrimeNG is a library for user interface, we are using it to have unified UI and it also saves us a lot of time because we can use components such as calendar, inputs, tables and others.

# Other packages and libraries

All packages and libraries can be found in *Package.json* located in front end folder.