Harun Gokcegoz

DHI1V.SO 509855

MOVIEPICKER

WEB APPLICATION PROJECT

Table of Contents

[Introduction 2](#_Toc107207941)

[Frontend Documentation 2](#_Toc107207942)

[Simplified Wireframe 2](#_Toc107207943)

[Framework Choices: 4](#_Toc107207944)

[Frontend Explanation of HTML/CSS/JS 4](#_Toc107207945)

[Backend Documentation 11](#_Toc107207946)

[Framework Choices 11](#_Toc107207947)

[Class Diagram 11](#_Toc107207948)

[Sequence Diagrams 12](#_Toc107207949)

[Backend Structure 14](#_Toc107207950)

[Requests 14](#_Toc107207951)

# Introduction

In this documentation, frontend and backend explanations of a full-stack web application will be made, and a document presentation of the developed application will be made by giving information about the designs and content. This web application is generally designed to suggest a movie to the user based on a mood selected by the user. It aims to eliminate the situation of searching for movies to watch , which is a headache for everyone for hours.

# Frontend Documentation

## Simplified Wireframe

**Homepage:**

Graphical user interface, text

Description automatically generated

**Moviepicker Page:**

Graphical user interface, application

Description automatically generated

**Movielist page:**

Graphical user interface, application

Description automatically generated

# Framework Choices:

|  |  |
| --- | --- |
| **HTML** | I used HTML because It is supported by all browser easily and it is easy to create something. |
| **CSS** | I used CSS because the design and graphics of the website are the most important things. |
| **JS** | I used JS because I wanted to create a interactive website. It helped to make it fancy and easy to use and also it is easy to connect to backend. |

# Frontend Explanation of HTML/CSS/JS

**Homepage:**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Registration and login buttons (not active yet) have been placed in the upper right corner of the homepage.

A navigation bar was designed just after the login buttons. Graphical improvements have been made to the links on this bar with the ": hover" tag. This navigation bar has been fixed to the top of the page with the "sticky" tag, making it accessible all over the page.

To show a visually beautiful design on the page, a background was designed for the site content and a 3D effect was given with the "shadow" effect in this way. A "aside" was placed, and links were created for the website's social media accounts and mobile application on this aside. These links are also visually enhanced with ": hover" tags, just like in the navigation bar.

On the other hand, on the side of the aside bar, a text area was created to provide information about the site, a background was given to this text area, and an A4-page look was given to the text in this way. The 3D effect has been given with the "shadow" tag. The texts were filled with lorem ipsums to make the site design easier.

I wanted to create visual content for the homepage by developing 3 different boxes under this text section, but since I could not spend much time on it, they seem a bit poor.

A text area was created under this box section, and a wrapper background was added to this text area and it was visually enhanced.

In the footer section, two buttons have been designed for contact and help, and the copyright, developed year, and designer signature, which has become a classic for every website, are displayed.

**Movie Picker Page:**

Graphical user interface, application

Description automatically generated

Graphical user interface, website

Description automatically generated

When "Movie Picker" is clicked on the main page, the user is greeted with a page like this.

A dropdown list has been prepared so that the user can choose their mood, and a "choose" button has been designed to select it.

A visual movie description draft was prepared for the movie to be generated just below the mood selection.

As soon as the user selects the mood and uses the choose button, a progress bar appears and when it reaches 100%, a movie recommendation is displayed according to the selected mood by user.

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

After the user has clicked the "choose" button, the selected movie appears on the screen. Here you will find information about the movie and a cover image. there is a review section under this movie information. Here, reviews about the film are shown, and in the "text field" located below it, the user can write a review about this film anonymously.

**Watch Before Die Page:**

Graphical user interface, application

Description automatically generated

This page shows 10 movies for users. A separate wrapper (background) was created for each movie to give a visual effect, and a 3D effect was created with the "shadow" effect. It was not written on js through database to improve html skills. It was created with HTML.

**Profile page:**

**Graphical user interface

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

A dashboard has been developed on this page where the user who logs in can change their account information, view the watch lists that will be created, log out, and get help. the contents on this page are not ready for use. It is prepared only for future developments**.**

# Backend Documentation

# Framework Choices

|  |  |
| --- | --- |
| SPRING BOOT | Although it is not very popular, I used SpringBoot because the java software is language-based (I have enough experience to develop something). |

# Class Diagram

A picture containing arrow

Description automatically generated

Three different entities were created here. Attributes were created to store the information of the movies displayed on the first entity. The second entity is the mood table. A many-to-one relationship was established between these two tables. A movie can have only one mood, while a mood can have multiple movies.

The other entity is the "review" that will be displayed for the movies. A one-to-many relationship has also been established between the movie table and the review table. A movie can have multiple reviews, while a review can only belong to one movie.

# Sequence Diagrams

**Diagram - 1:**

Chart, box and whisker chart

Description automatically generated

In this diagram, the user selects his mood and starts the process with the "choose" button. This button sends data to the backend via fetch API with GET. The backend finds the requested information and sends it back to the fetch API as a response. Here, code 201 is sent because the operation was successful. the movie that needs to be shown later is displayed on the screen with javascript codes

**Diagram 2:**

Chart, box and whisker chart

Description automatically generated

In this diagram, the user enters the review he wants to send into the text field and sends it to the fetch API with "send review". fetch API converts this data as JSON and sends it to the backend via POST. The backend reads the data and saves this data to the database with the necessary method. If invalid data is entered, an error message is displayed by the user.

# Backend Structure

Text

Description automatically generated

Here is the backend structure that I set up. this is an incorrect installation. Instead of opening packages for entities, I needed to open packages according to classes. However, when I realized that it was wrong, it was too late to change it.

# Requests

**GET:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **api/movies/get** | | | |
| It gets all movies from the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully gets all movies | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **api/moods/get** | | | |
| It gets all moods from the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully gets all moods | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **api/reviews/get** | | | |
| It gets all movies from the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully gets all reviews | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **api/movies/post** | | | |
| It posts a new specific movie to the backend | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully sends a movie | |
|  | | 400 | Invalid data | |

POST:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **api/moods/post** | | | |
| It posts a new specific mood to the backend | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully sends a mood | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **api/reviews/post** | | | |
| It posts a new specific review to the backend | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully sends a review | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **api/movies/id** | | | |
| It updates a specific movie in the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
|  | | name | (path) | It gets a new name if it needs to be updated |
|  | | category | path | It gets a new category if it needs to be updated |
|  | | description | path | It gets a new description if it needs to be updated |
|  | | releaseYear | path | It gets a new release year if it needs to be updated |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully updates a movie | |
|  | | 400 | Invalid data | |

PUT:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **api/moods/ id** | | | |
| It updates a specific mood in the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
|  | | name | (path) | It gets a new name if it needs to be updated |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully updates a mood | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **api/reviews/ id** | | | |
| It updates a specific mood in the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
|  | | review | (path) | It gets a new name if it needs to be updated |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully updates a review | |
|  | | 400 | Invalid data | |

DELETE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **api/movies/ id** | | | |
| It deletes a specific movie in the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully deletes a movie | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **api/moods/ id** | | | |
| It deletes a specific mood in the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully deletes a mood | |
|  | | 400 | Invalid data | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **api/reviews/ id** | | | |
| It deletes a specific mood in the database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | id | (path) | Based on id |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Successfully deletes a review | |
|  | | 400 | Invalid data | |