IT 309 Software Engineering

Project Documentation

Film/series Review Web Application: **Spoiler Alert**

*Department of Engineering and Natural Sciences, Faculty of Information Technologies, International Burch University, Sarajevo*



***Prepared by: Presented to:***

Mirza Kurtovic Assist. Prof. Dr. Nermina Durmic

Nermin Kapetanovic Aldin Kovačević, Teaching Assistant

***Date of submission:*** 21.06.2023***.*** 23:54:59

Table of Contents

[1. Introduction 3](#_Toc138282513)

[1.1. About the Project 3](#_Toc138282514)

[1.2. Project Functionalities and Screenshots 3](#_Toc138282515)

[2. Project Structure 4](#_Toc138282516)

[2.1. Technologies 4](#_Toc138282517)

[2.1.1. Database 5](#_Toc138282518)

[2.1.2. Backend 5](#_Toc138282519)

[2.1.3. Frontend 5](#_Toc138282520)

[2.1.4. Coding Standards 5](#_Toc138282521)

[2.2. Database Entities 5](#_Toc138282522)

[2.3. Design Patterns 6](#_Toc138282523)

[2.4. Tests 7](#_Toc138282524)

[3. Conclusion 7](#_Toc138282525)

[4. References 7](#_Toc138282526)

# 1. Introduction

## 1.1. About the Project

The name of our project is Spoiler Alert. Spoiler Alert is a movie/series review website aimed at film enthusiasts, that will help its users find the best content for them. The website aims to create a community where users will be able to read and post reviews, search and filter content that best suits them. The website features a modern, friendly UI, that makes the tedious process of searching for new movies a piece of cake. The website is aimed at movie enthusiasts and people of all ages who are in search of their next Saturday night entertainment.

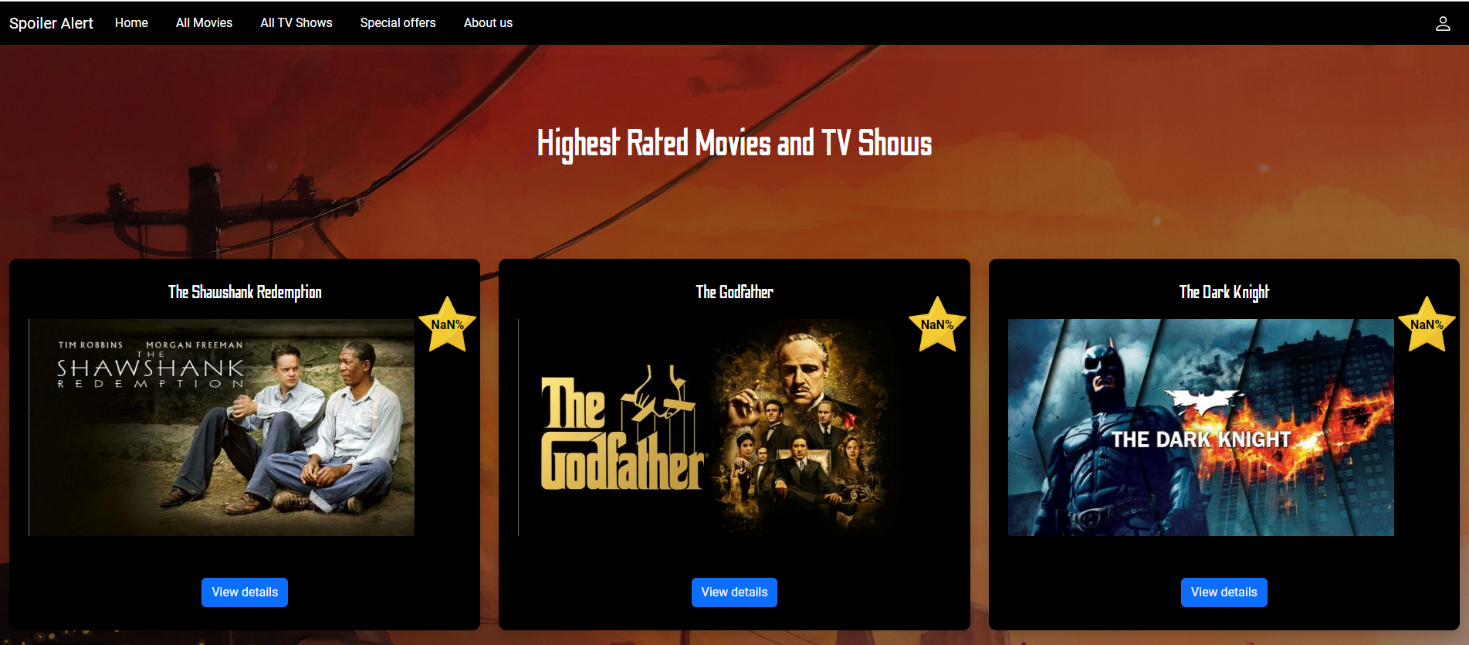
GitHub repository link: <https://github.com/mirzaskola/SEProject2023>

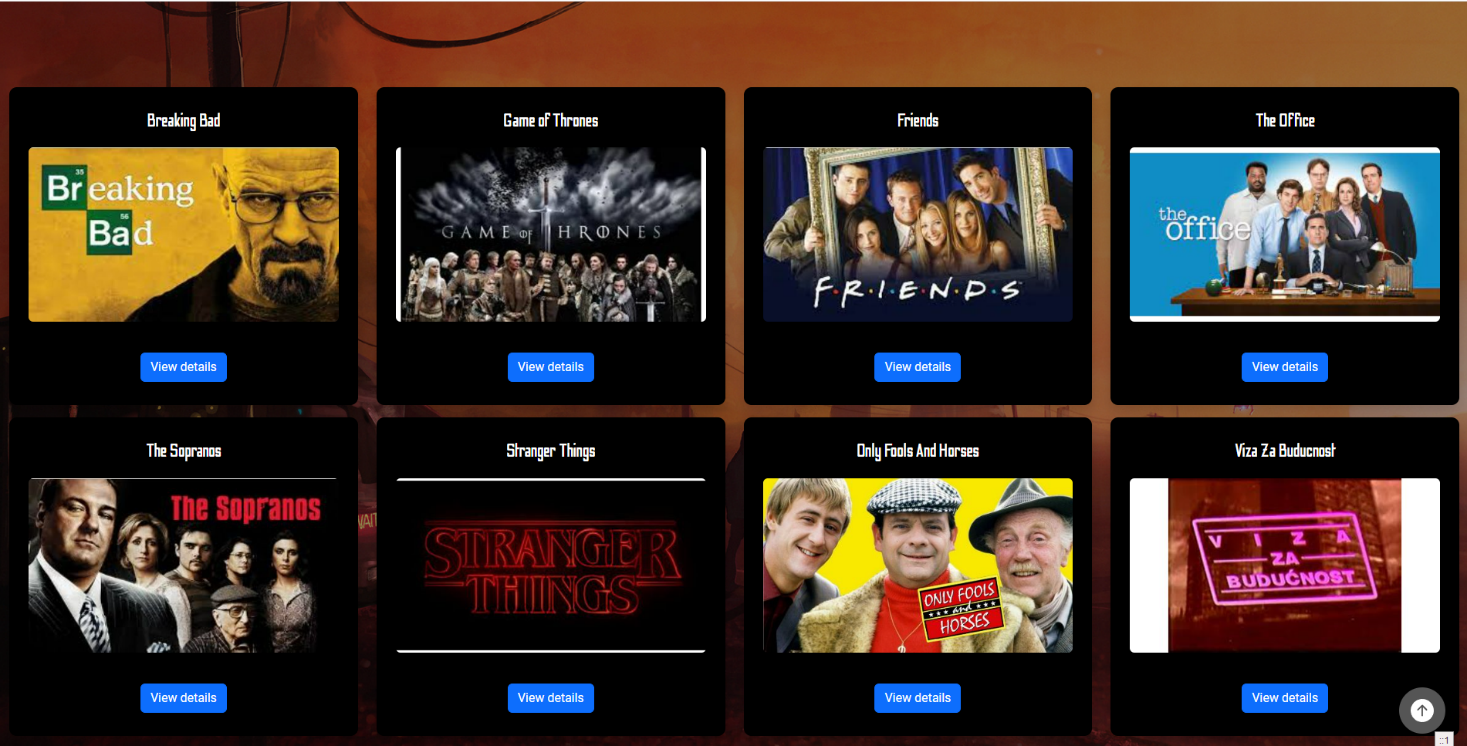
Link to website: <http://209.38.247.151/SEProject2023/>

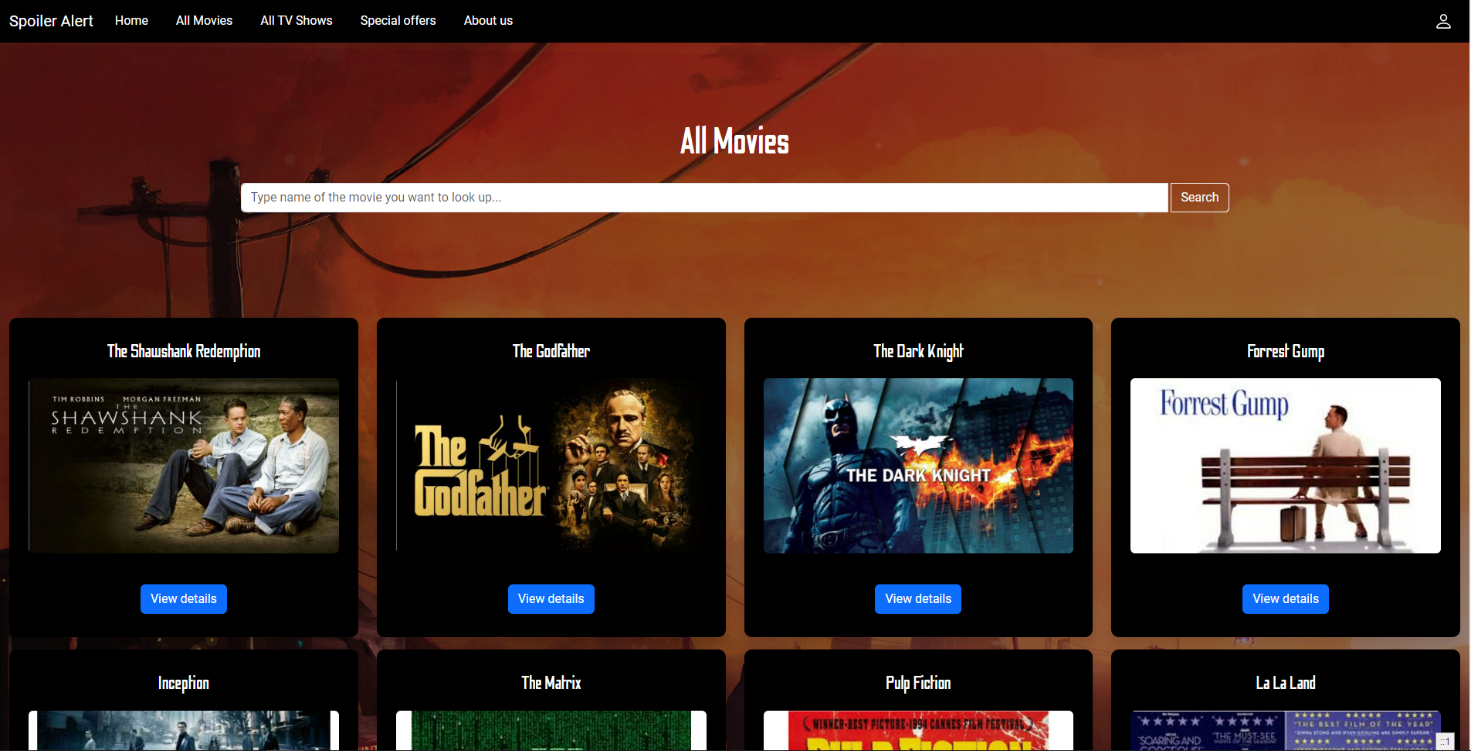
## 1.2. Project Functionalities and Screenshots

The main features of the project are:

* Login/register system with tokenization
* Leave a review/comment feature
* Content searching
* Profile management system
* Admin panel (not working properly, but available)







# 2. Project Structure

## 2.1. Technologies

The project is a web application that was implemented using the technologies and frameworks listed below.

### 2.1.1. Database

The database is a relational database implemented using MySQL for database creation and manipulation. The entities implemented are listed in section 2.2.

### 2.1.2. Backend

The backend consists of 3 layers:

* DAO Layer
* Business Logic Layer
* API Layer (Routes)

which were built using PHP and the FlightPHP framework.

### 2.1.3. Frontend

The frontend was built using HTML/CSS, Bootstrap, jQuery and JavaScript.

### 2.1.4. Coding Standards

The coding standards we have used to implement the project are PHP’s PSR-12 standard within all 3 layers of the backend and JavaScript’s “JavaScript Standard Style” for the frontend. Links for the standards will be included in the references section.

## 2.2. Database Entities

Our database contains 5 entities:

* user
* content – Contains all information regarding movies and TV shows
* genre
* offer – Stores offers/discounts from our “partners” (Not a scam, please click all the links)
* rating – Individual ratings from users for the movies/shows

## 2.3. Design Patterns

The design patterns we have decided to use are:

* DAO pattern: used in the backend, in the file *api/dao* and *api/interfaces:*
* ContentDao.php
* GenreDao.php
* OfferDao.php
* RatingDao.php
* UserDao.php
* ContentDaoInterface.php
* GenreDaoInterface.php
* OfferDaoInterface.php
* RatingDaoInterface.php
* UserDaoInterface.php
* Builder pattern: used in the backend, in the folder *api/builders/* and *api/interfaces*:
* ContentBuilder.php
* MovieBuilder.php
* ShowBuilder.php
* ContentBuilderInterface.php

Both patterns use the files in *api/models* to transfer data from one layer to the other.

Since our backend is designed in a layered structure, the DAO pattern was used to implement the lowest layer of data persistence in order to handle communication with the database using SQL queries in code and convert database entities into objects that can be then passed and manipulated by the layers above. This pattern made it easy to implement necessary CRUD operations for the database entities and helped develop the layers above by simply calling functions. Functions from the DAO pattern were also used to implement the Builder Pattern.

The Builder Pattern was used to implement the Business Logic Layer (Service layer) of the “content” entity. It is a complex structure that requires multiple functions to be called in order to be parsed on the frontend. The builder pattern was an intuitive solution for handling this problem and avoiding messy, spaghettified code. The content is built step-by-step to make the object easier to build, more readable and maintainable.

## 2.4. Tests

We have written 2 Selenium tests in Python to test out the main features of the project on the frontend:

* Login/logout test – The login page is loaded, (the username and password are already in the form for ease of testing). Logs in, returns the token generated and asserts the token is not missing or empty. Afterwards the script logs out the user and asserts there is no token in the browser’s local storage.
* Leave a review test – Loads the homepage of the site, takes the first title in the Highest Rated section, attempts to leave a review and asserts the review is present in the “details” modal.

The tests are located in the root project folder (test\_review.py, test\_login.py)

# 3. Conclusion

The journey to implementing our project taught us a great deal of methods and techniques that we were able to adapt and use to build Spoiler Alert. Even though the scope of the project is not large, we believe that this implementation well represents our capabilities in terms of what we have learnt throughout the course. The most difficult part was selecting the adequate design patterns to use within the project, due to the fact that, as we read, many of the patterns we have learnt about, have emerged as an industry standard for larger and much more complex projects than ours, so the challenge was to adapt the pattern to the project, not vice-versa Sadly, we have not implemented all the features that we have wanted to, mainly focusing on the scope of the course, and focusing on proper implementation of patterns, so in terms of overall satisfaction we wish we could have done more, but we hope this will be enough as a proof of concept.

# 4. References

PSR-12 documentation: <https://www.php-fig.org/psr/psr-12/>

JavaScript Standard Style documentation: <https://standardjs.com/rules.html>

DAO design pattern documentation: <https://www.digitalocean.com/community/tutorials/dao-design-pattern>

Builder design pattern documentation: <https://refactoring.guru/design-patterns/builder>