**Project Report: Movie Review Website**

**1. Introduction**

The Movie Review Website project aims to provide users with a platform to explore movies and share their reviews. This report provides an in-depth analysis of the project, covering both frontend and backend components, as well as deployment details.

**2. Project Overview**

The project consists of a frontend web application and a backend server. The frontend allows users to browse movies, search for specific titles, and view movie details along with reviews. The backend manages the storage and retrieval of movie data and reviews, handling HTTP requests from the frontend.

**3. Frontend**

**3.1. Technologies Used**

- HTML: Used for structuring web pages.

- CSS: Used for styling the user interface.

- JavaScript: Used for dynamic interactions and fetching data from the backend.

- cors: Cross Origin Resource Sharing.

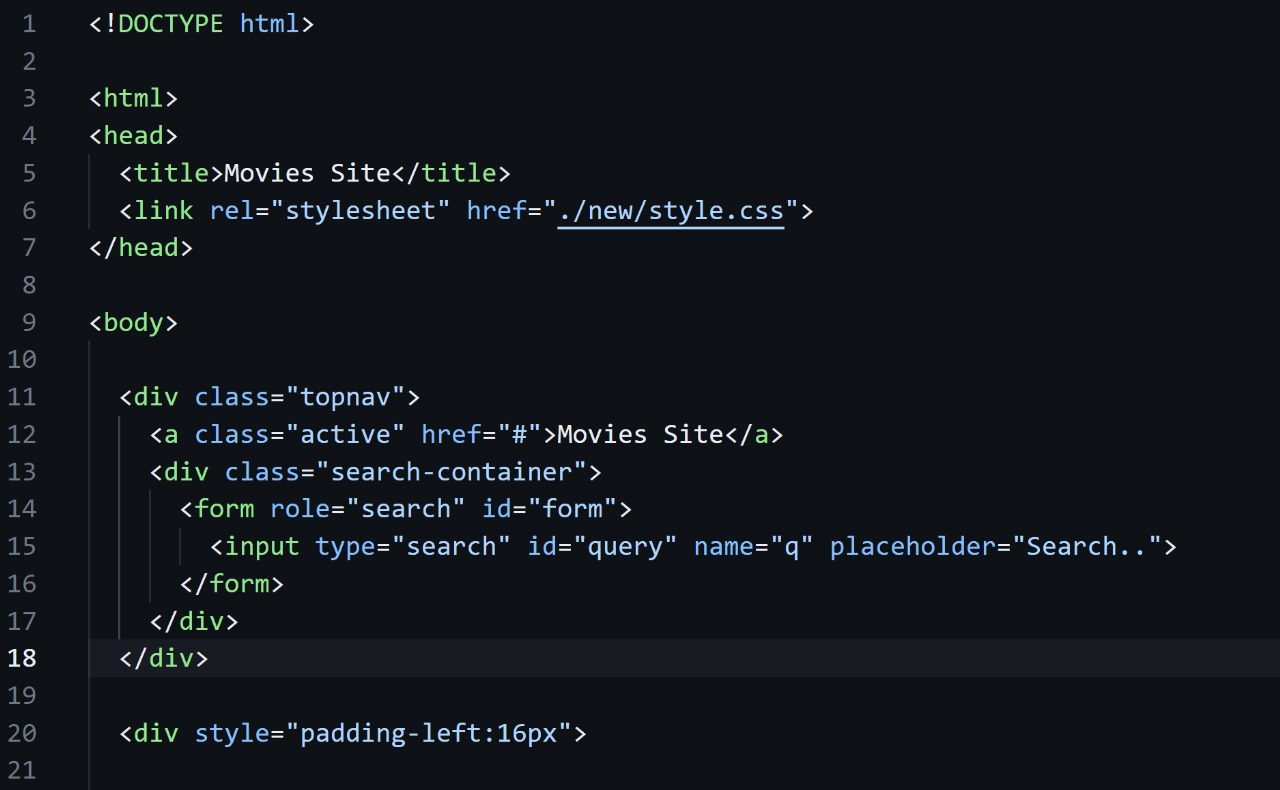
-Express.js: JavaScript framework for backend.

-MongoDB: For database.

-mongoose: MongoDB library.

**3.2. Files**

- **index.html**: The main webpage displaying a list of movies and a search bar.



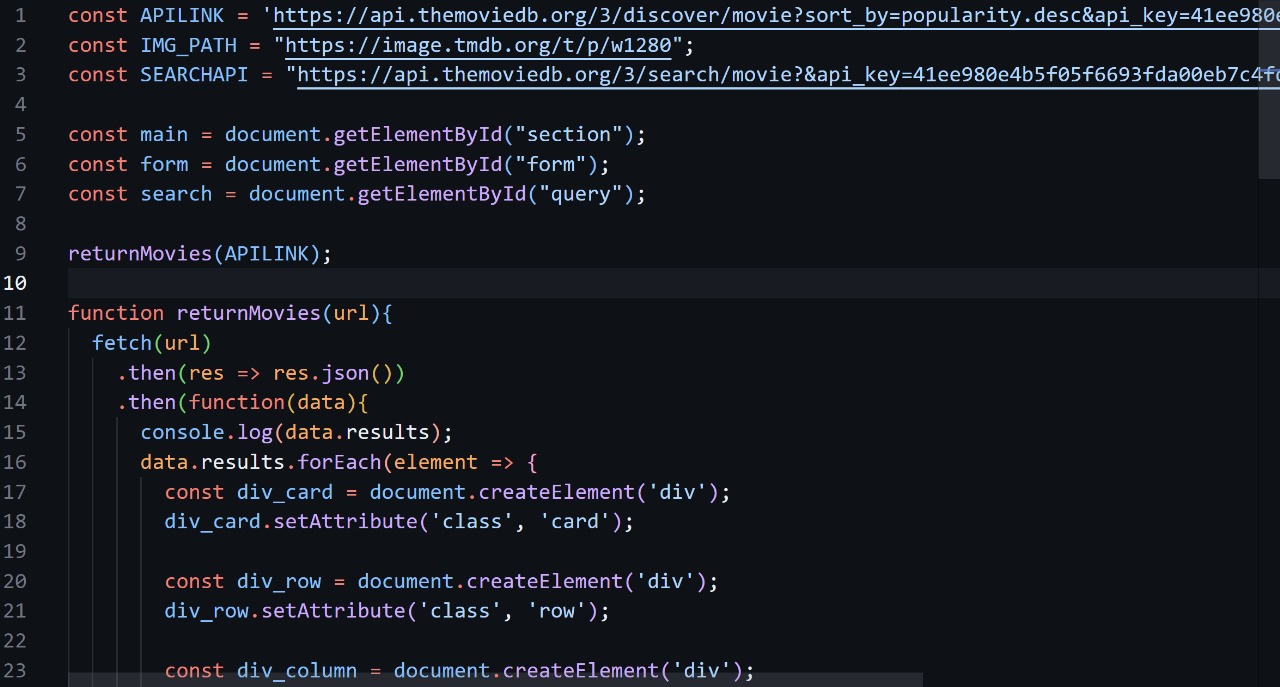


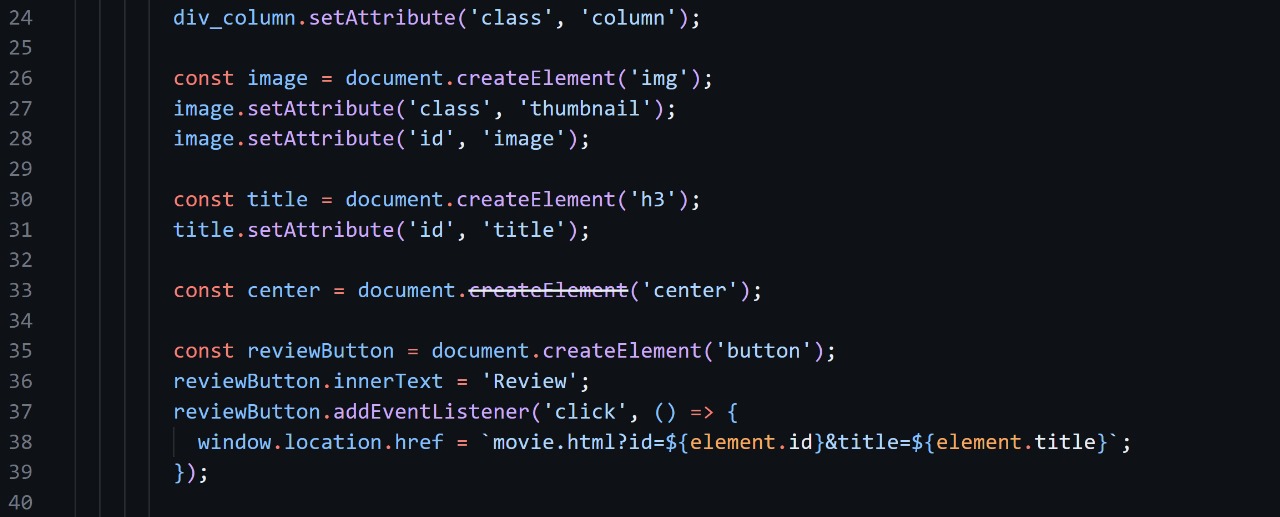
**- movie.html**: Displays reviews for a specific movie.

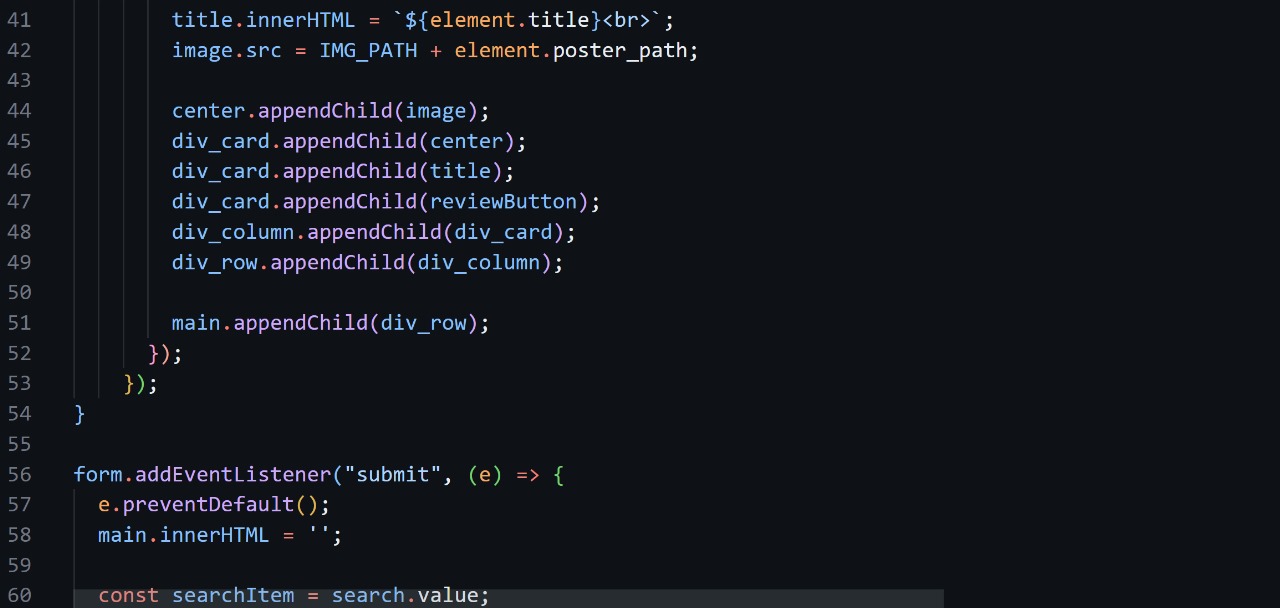


- **style.css**: Contains styling rules for the frontend.

- **script.js**: Handles fetching movies from an external API and search functionality.

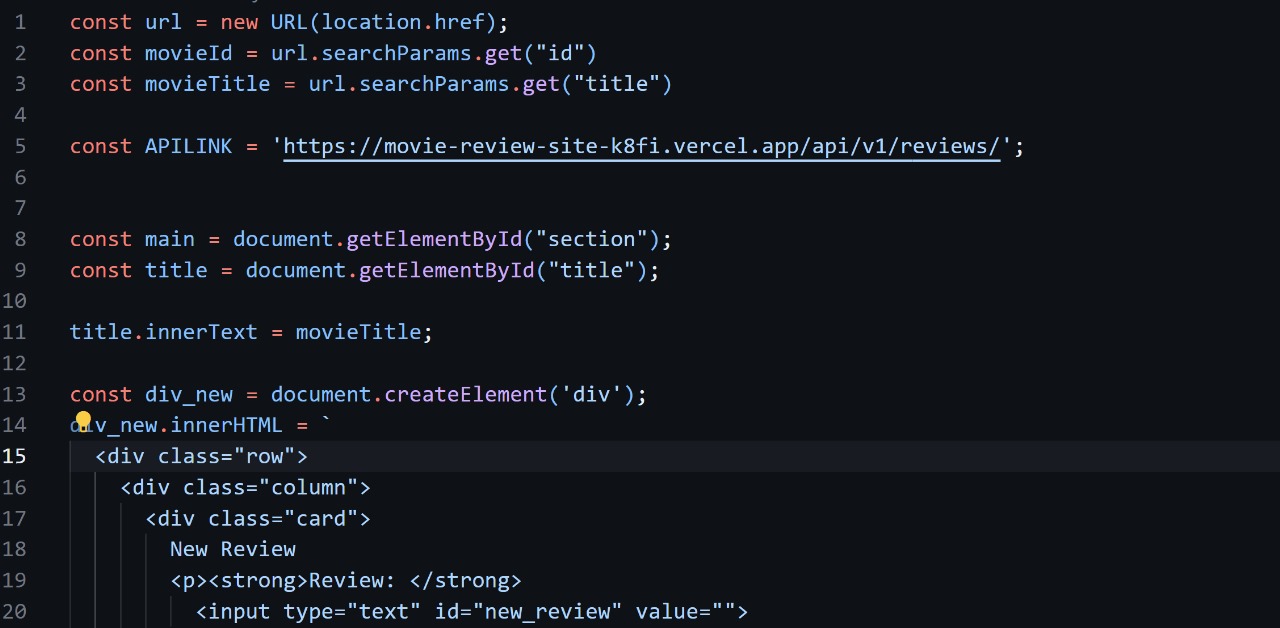


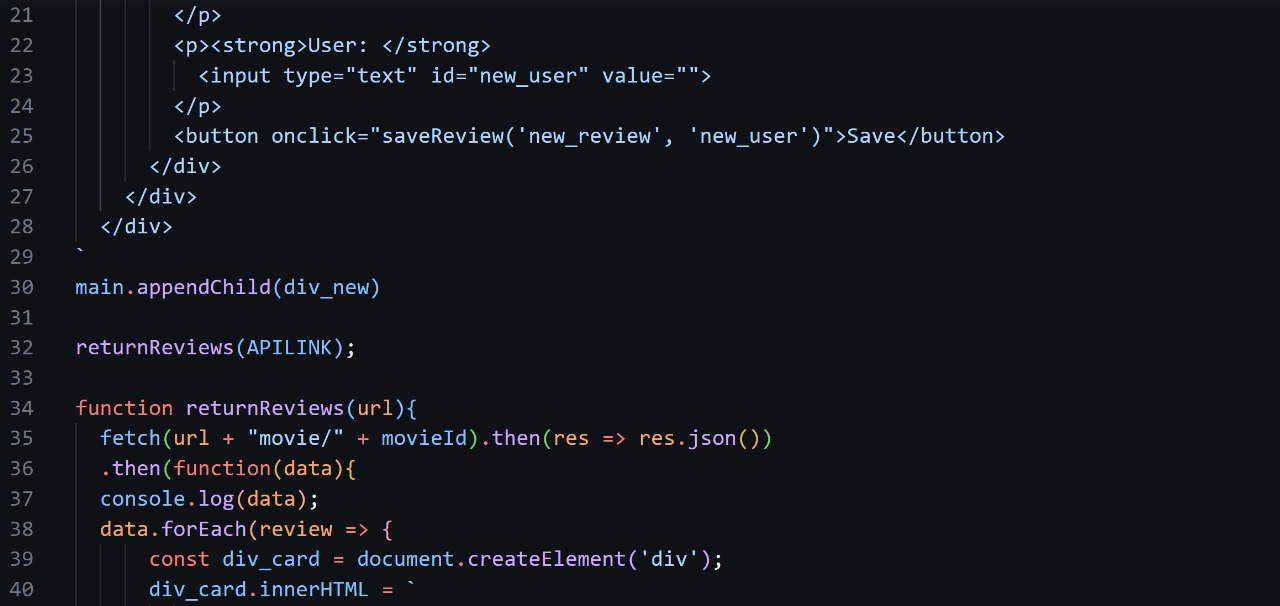


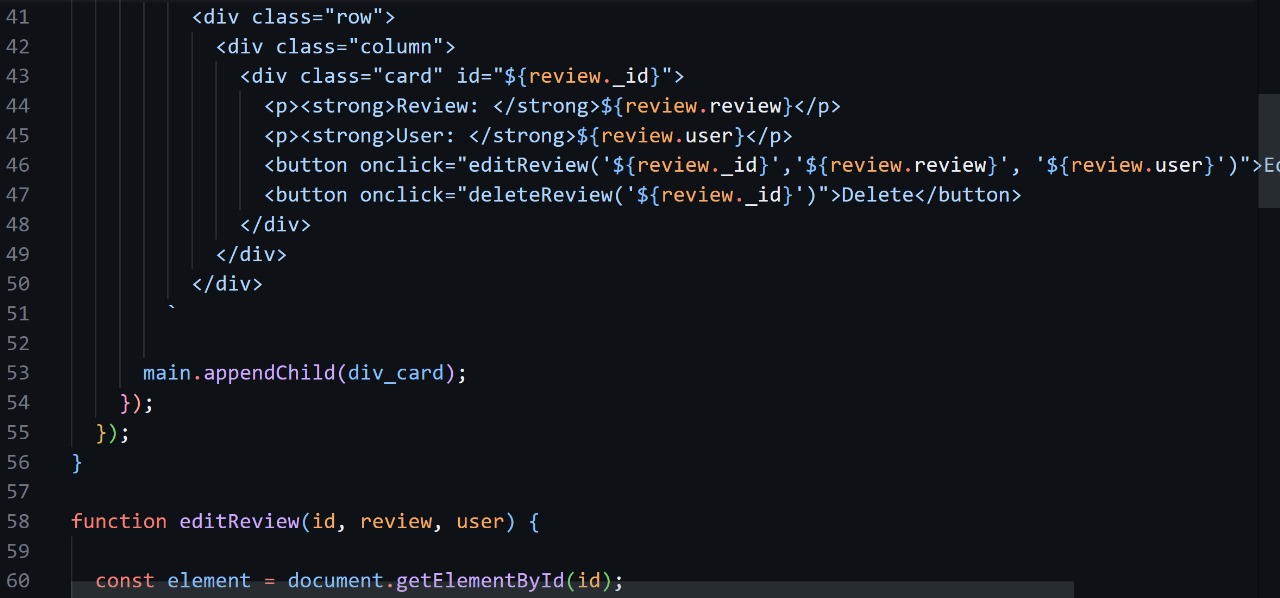




- **movie.js**: Handles fetching and managing reviews for a specific movie.

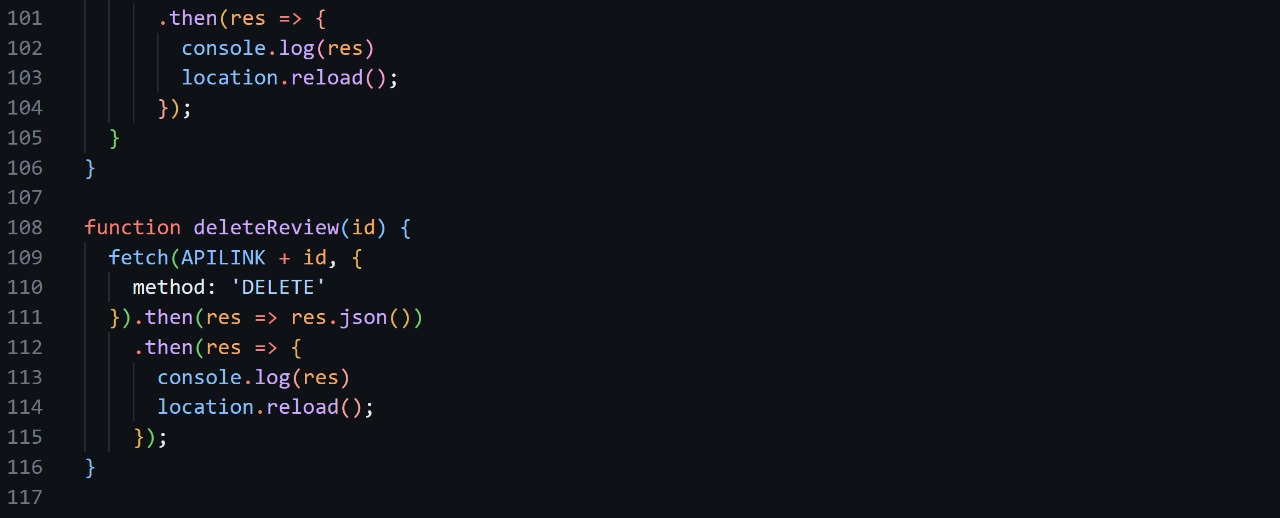












**3.3. Functionality**

- Users can browse movies and search for specific titles.

- Clicking on a movie displays its details and allows users to see reviews.

- Users can submit new reviews for movies.

**4. Backend**

**4.1. Technologies Used**

- Node.js: JavaScript runtime for building server-side applications.

- Express.js: Web application framework for Node.js.

- MongoDB: NoSQL database used to store movie data and reviews.

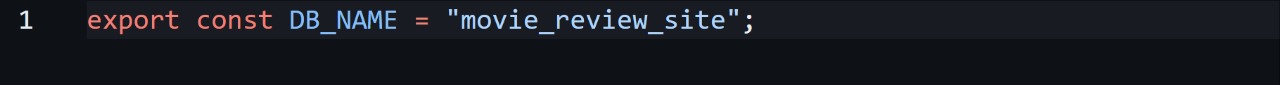
- Mongoose: ODM library for MongoDB, used to model application data.

**4.2. Files**

- **connectdb.js**: Establishes a connection to the MongoDB database.



- **constant.js**: Defines constants used in the project, such as the database name.

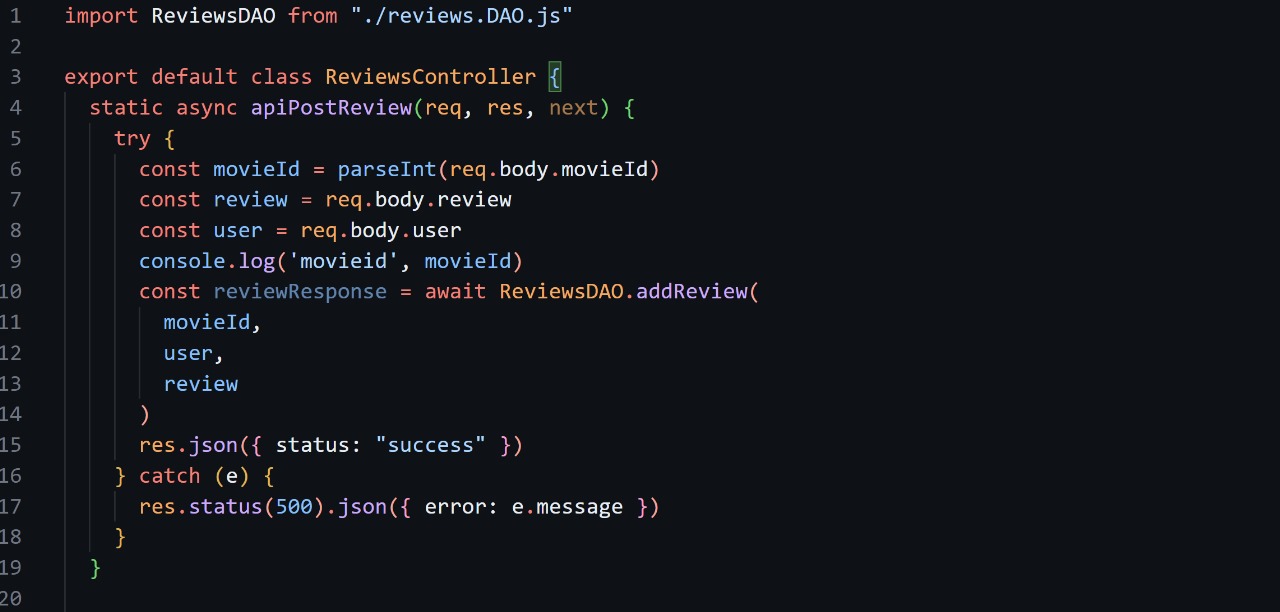


- **index.js**: Entry point of the backend application, configures the server and database connection.





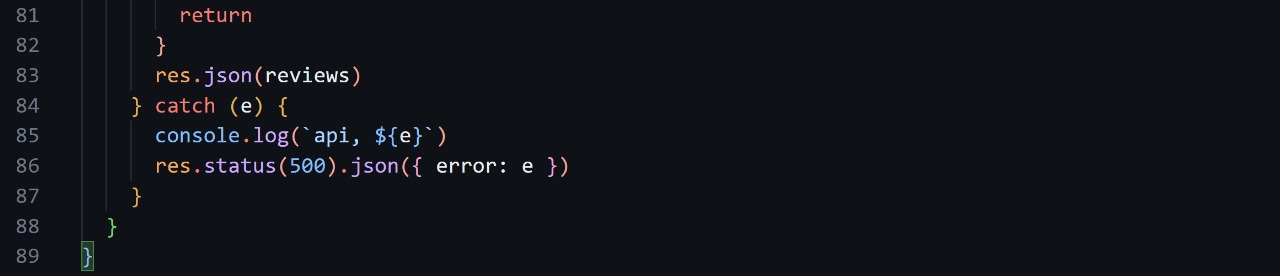
- **reviews.controller.js**: Handles HTTP requests related to reviews.







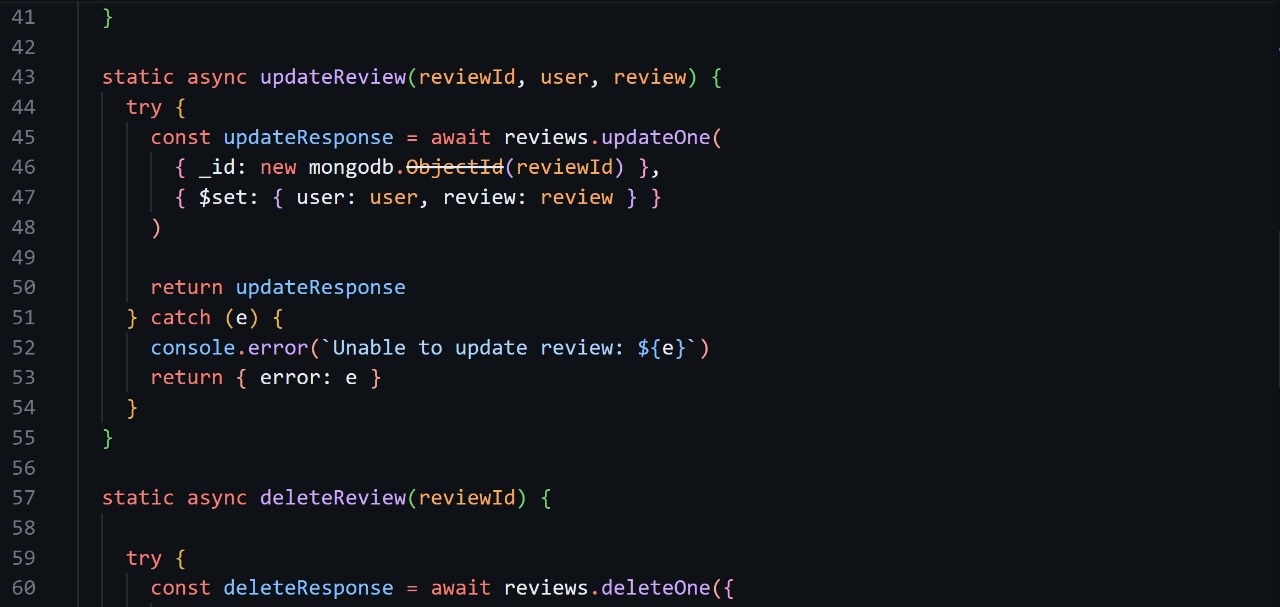


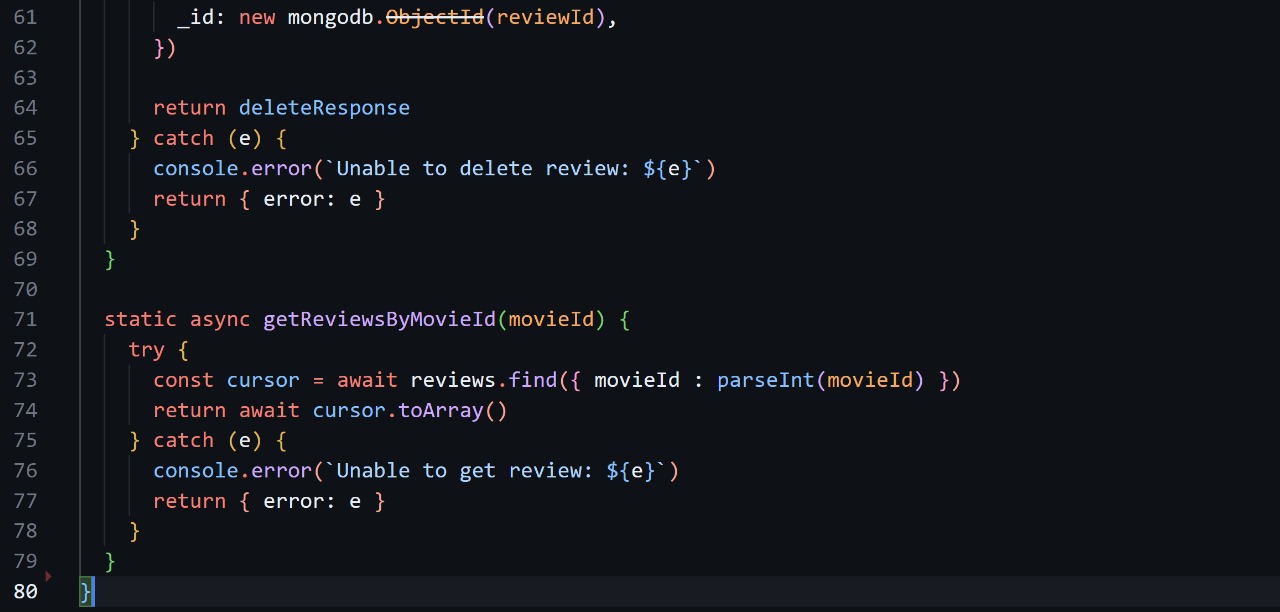


- **reviews.DAO.js**: Data access object for interacting with the MongoDB database.









- **reviews.route.js**: Defines routes for review-related HTTP requests.



- **server.js**: Creates an Express application and defines middleware and routes.



**4.3. Functionality**

- Provides APIs for CRUD operations on movie reviews.

- Allows users to add, retrieve, update, and delete reviews.

- Handles requests from the frontend and communicates with the database.

**5. Deployment**

The project is hosted on Vercel, a cloud platform that provides hosting for web projects. The codebase is stored on GitHub, allowing for version control and collaboration. Continuous deployment is set up, ensuring that any changes pushed to the GitHub repository are automatically deployed to the Vercel platform.

**6. Conclusion**

The Movie Review Website project successfully combines frontend and backend technologies to create a platform for users to explore movies and share their opinions. The modular architecture and use of modern web technologies make the project scalable and maintainable. With further enhancements and refinements, the website has the potential to become a popular destination for movie enthusiasts to discover and discuss their favourite films.