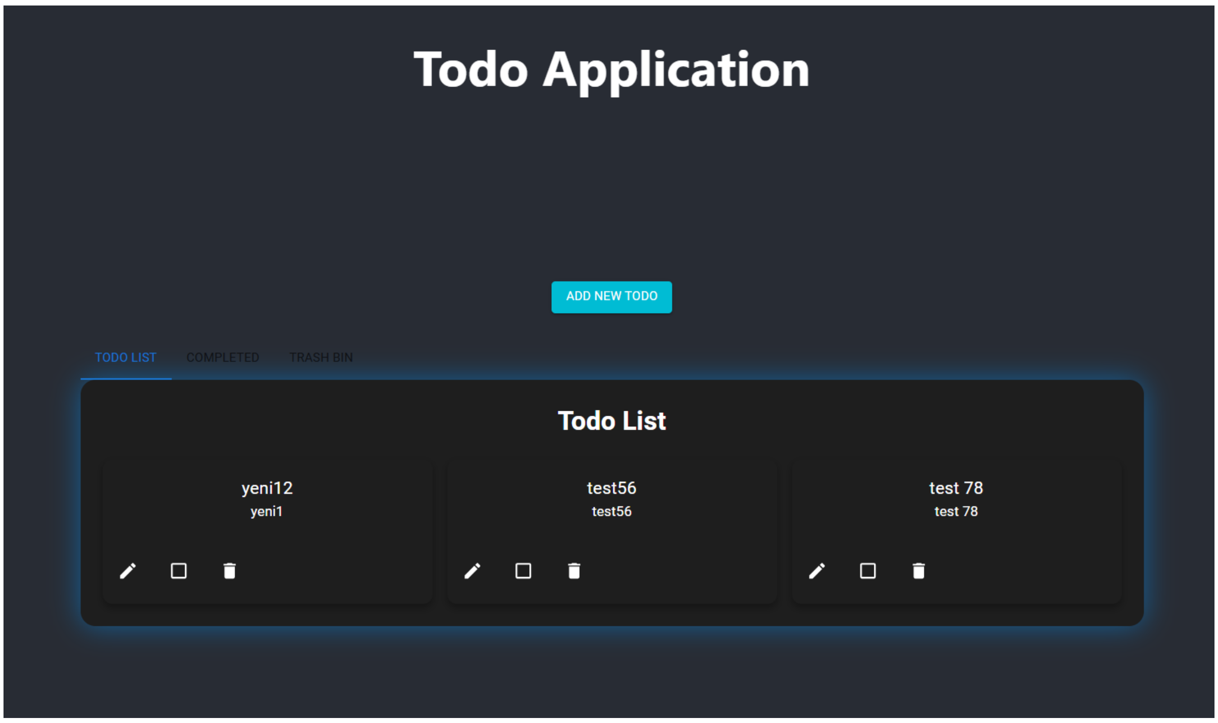
TODO APPLICATION



Introduction

This report provides an overview of the Todo Application, detailing its features, technologies used, architecture, and code structure. The Todo Application is designed to help users manage their tasks efficiently by allowing them to add, edit, delete, and track the completion status of their tasks.

Features

The Todo Application includes the following features:

- **Add New Todo**: Users can create new tasks.
- **Edit Todo**: Users can modify existing tasks.
- **Delete Todo**: Users can remove tasks.
- **Task Status Tracking**: Users can mark tasks as completed and move them to the "Completed" section.
- **Trash Bin**: Deleted tasks are moved to the trash bin for potential recovery.

Technologies Used

Client Side

- **React.js**: The client side of the application is built using React.js, a popular JavaScript library for building user interfaces. React.js allows for efficient, modular, and maintainable code, making it an ideal choice for building dynamic web applications.

Server Side

- **Node.js**: The server side is built using Node.js, a JavaScript runtime that allows for server-side scripting. Node.js enables the application to handle multiple requests efficiently.
- **Express.js**: The server is powered by Express.js, a web application framework for Node.js. Express.js simplifies the development of web servers and APIs, providing a robust set of features for web and mobile applications.
- **MongoDB Atlas**: MongoDB Atlas is used as the database for storing tasks. MongoDB is a NoSQL database that provides high performance, high availability, and easy scalability. MongoDB Atlas is a cloud-hosted version of MongoDB, offering automated backups, monitoring, and more.

Architecture

The application follows a client-server architecture, where the client and server interact through API endpoints. The client side is responsible for the user interface and user interactions, while the server side handles data processing, storage, and business logic.

Client-Server Interaction

1. **Creating a Task**: When a user creates a new task, the client sends a POST request to the server with the task details. The server then saves the task to the MongoDB Atlas database.
2. **Fetching Tasks**: The client fetches the list of tasks by sending a GET request to the server. The server retrieves the tasks from the database and sends them back to the client.
3. **Updating a Task**: When a user edits a task, the client sends a PUT request to the server with the updated task details. The server updates the task in the database.
4. **Deleting a Task**: When a user deletes a task, the client sends a DELETE request to the server. The server moves the task to the trash bin in the database.

Code Structure

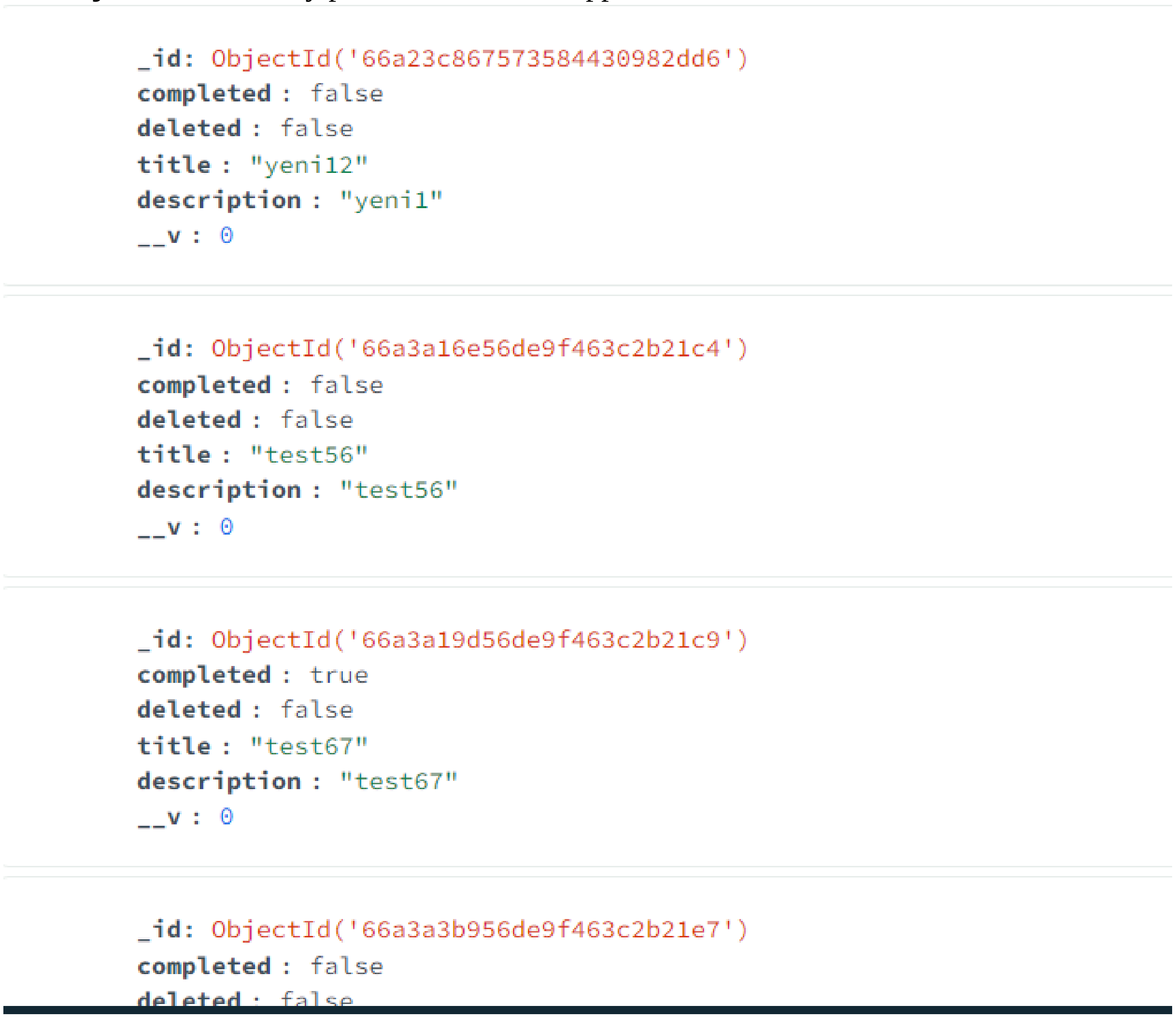
The codebase is organized into two main directories: `server` and `todo-app-client`.

Server

The `server` directory contains the backend code for the application. It includes the following subdirectories and files:

- **controllers**: Contains the `todoController.js` file, which handles the logic for task-related operations.
- **models**: Contains the `Todo.js` file, which defines the MongoDB schema for tasks.

- **routes**: Contains the `todoRoutes.js` file, which defines the API endpoints for task operations.
- **server.js**: The main entry point for the server application.



```
_id: ObjectId('66a23c867573584430982dd6')
completed : false
deleted : false
title : "yeni12"
description : "yeni1"
__v : 0
```

```
_id: ObjectId('66a3a16e56de9f463c2b21c4')
completed : false
deleted : false
title : "test56"
description : "test56"
__v : 0
```

```
_id: ObjectId('66a3a19d56de9f463c2b21c9')
completed : true
deleted : false
title : "test67"
description : "test67"
__v : 0
```

```
_id: ObjectId('66a3a3b956de9f463c2b21e7')
completed : false
deleted : false
```

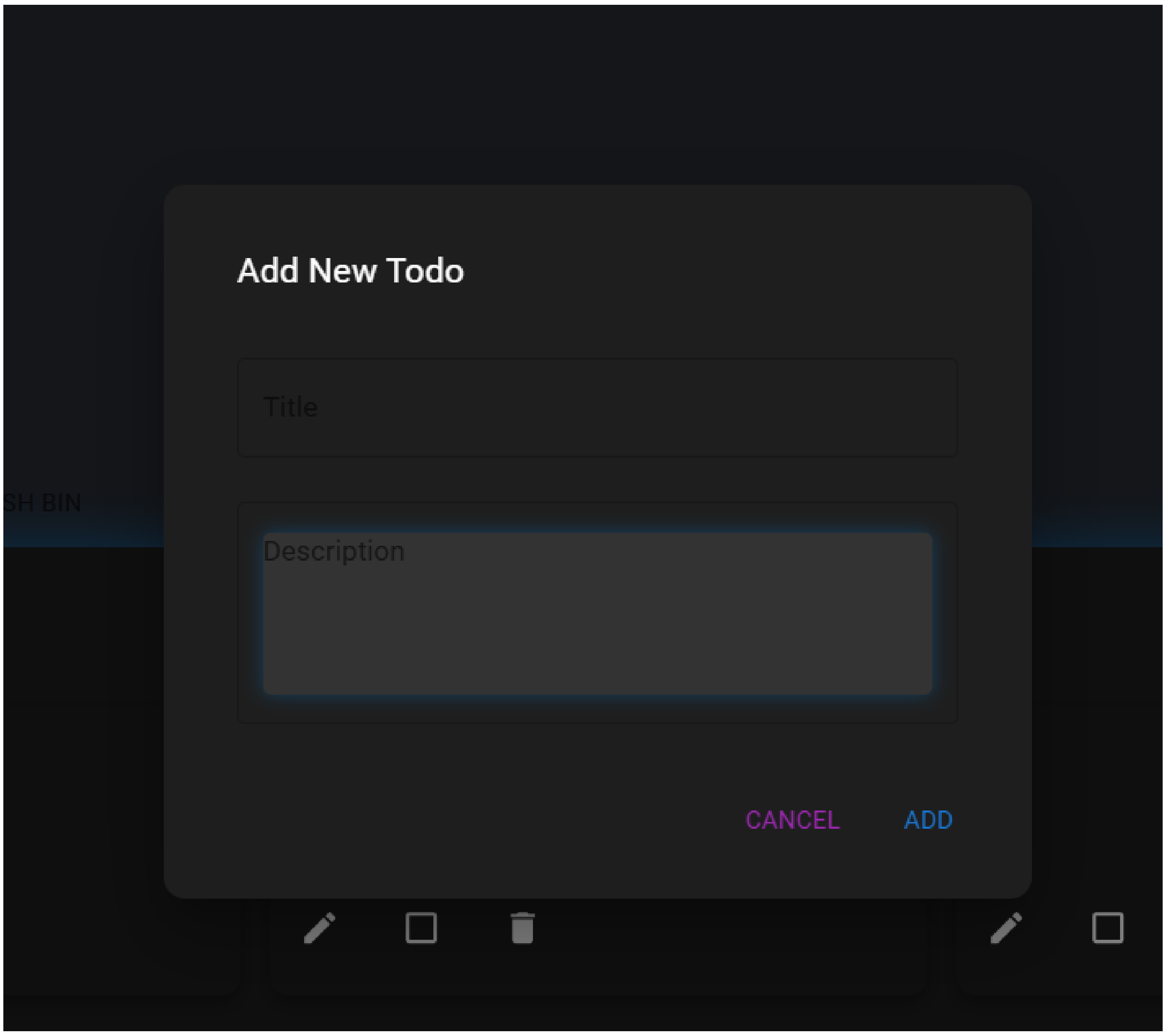
Client

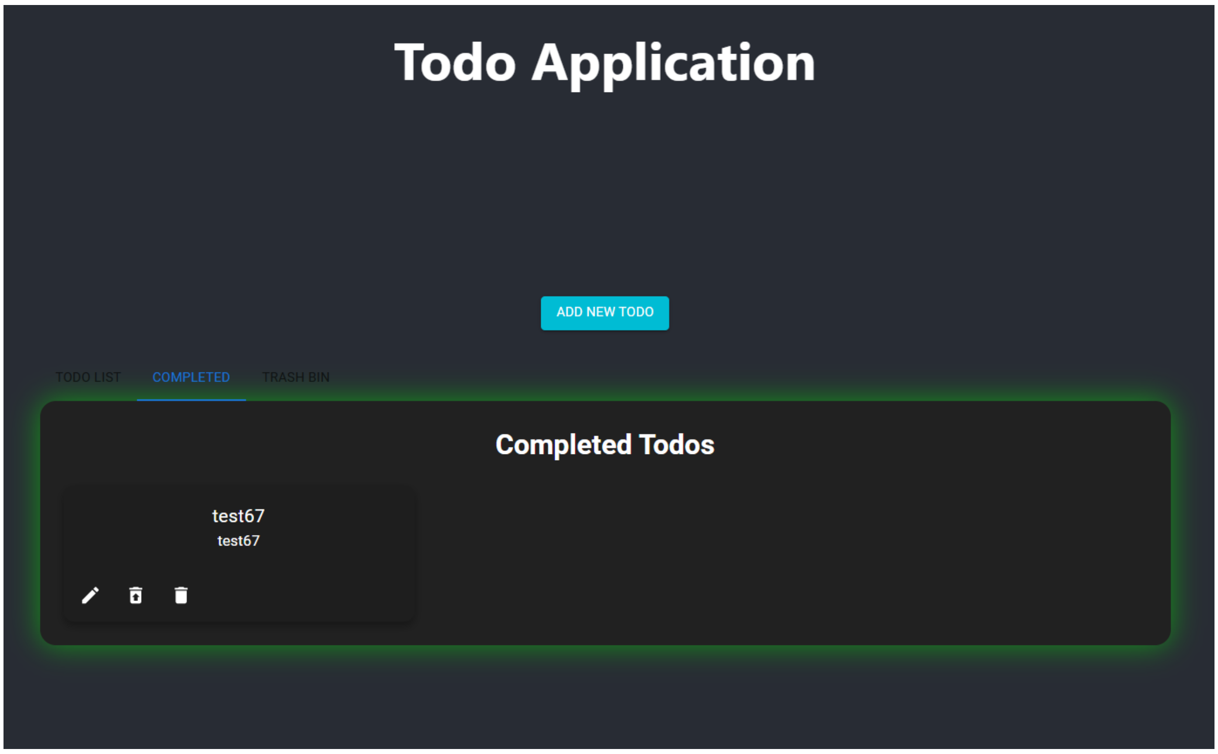
The `todo-app-client` directory contains the frontend code for the application. It includes the following subdirectories and files:

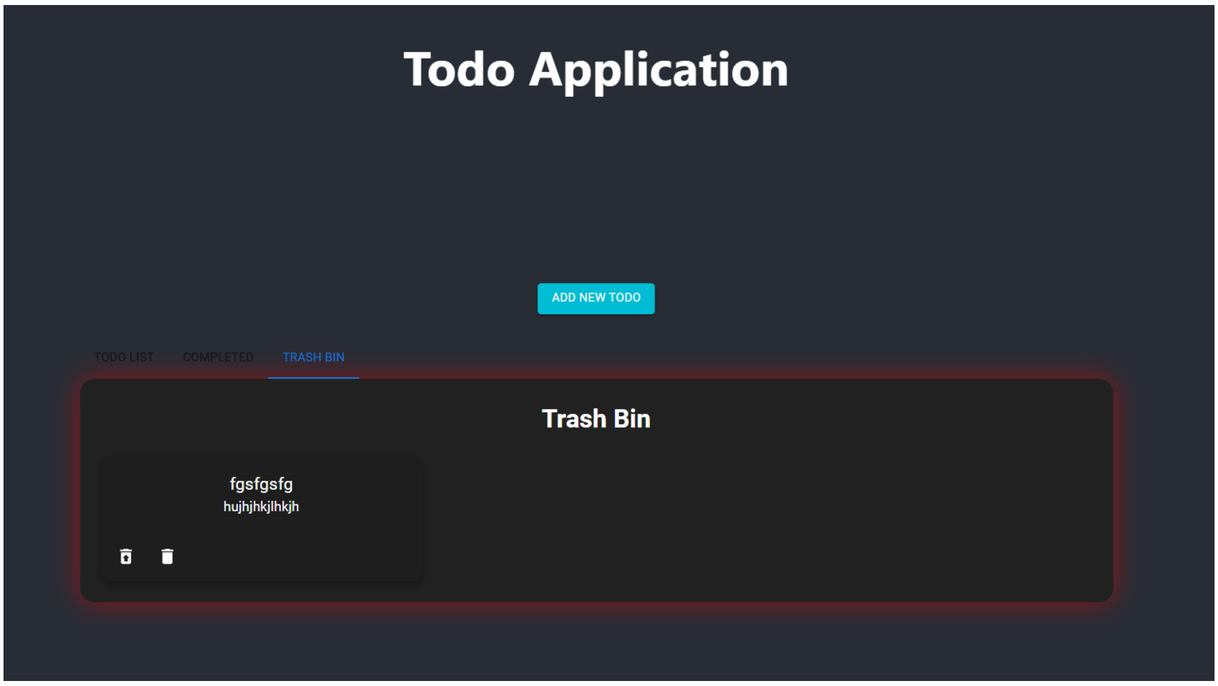
- **public**: Contains static assets and the `index.html` file.
- **src**: Contains the source code for the React application.
    - **components**: Contains React components such as `EditToDo.js`, `TodoForm.js`, and `TodoList.jsx`.
    - **api.js**: Handles API requests to the server.
    - **App.css**: Contains styles for the application.
    - **App.js**: The main React component that renders the application.
    - **index.js**: The entry point for the React application.
    - **reportWebVitals.js**: Used for measuring performance.

User Interface

The user interface of the Todo Application is designed to be intuitive and user-friendly. The main screen displays the list of tasks with options to add, edit, and delete tasks. The "Add New Todo" modal allows users to enter the title and description of a new task, as shown in the screenshot below:







Conclusion

The Todo Application is a robust task management tool built using modern web technologies. By leveraging React.js for the client side and Node.js with Express.js and MongoDB Atlas for the server side, the application offers a seamless and efficient user experience. This architecture ensures that the application is scalable, maintainable, and capable of handling a large number of tasks and users efficiently. The well-organized code structure further facilitates ease of development and maintenance.