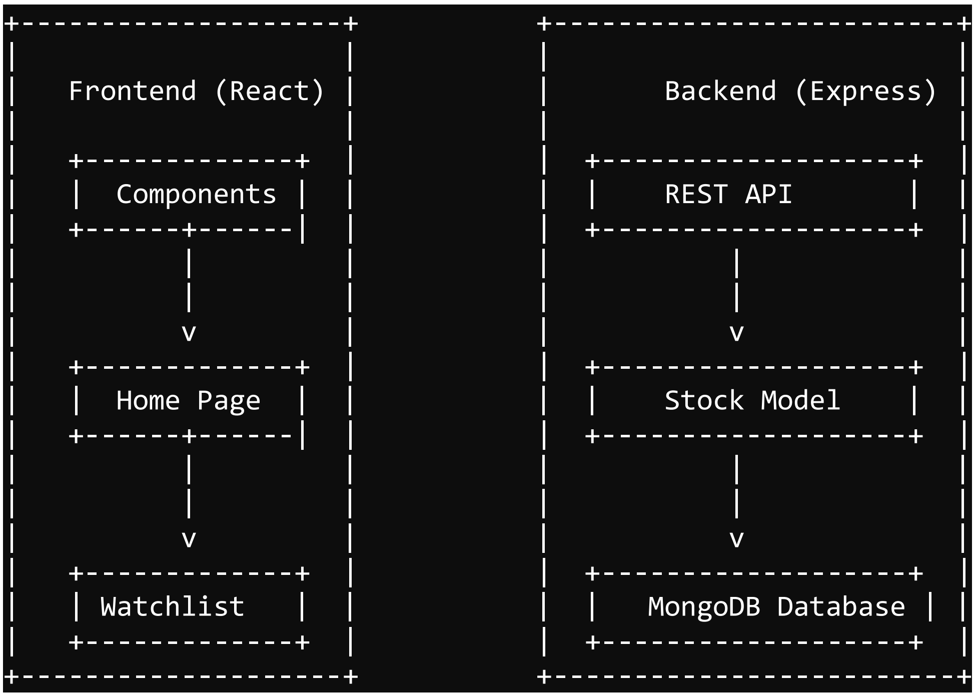
Architecture Diagram Description

The architecture diagram provides a visual representation of the Stock Market Portfolio application's components and their interactions. Here’s a detailed breakdown of each element in the diagram:



```
+--------------------+           +-------------------------+
|                    |           |                         |
|   Frontend (React) |           |       Backend (Express) |
|                    |           |                         |
|   +-------------+  |           |  +--------------------+  |
|   |  Components |  |           |  |    REST API        |  |
|   +------+------|  |           |  +--------------------+  |
|          |         |           |           |              |
|          |         |           |           |              |
|          v         |           |           v              |
|   +-------------+  |           |  +--------------------+  |
|   |  Home Page  |  |           |  |    Stock Model     |  |
|   +------+------|  |           |  +--------------------+  |
|          |         |           |           |              |
|          |         |           |           |              |
|          v         |           |           v              |
|   +-------------+  |           |  +--------------------+  |
|   | Watchlist   |  |           |  |   MongoDB Database |  |
|   +-------------+  |           |  +--------------------+  |
+--------------------+           +-------------------------+
```

- **Frontend (React)**: Manages the user interface and communicates with the backend via API requests.
- **Backend (Express)**: Processes API requests, interacts with the database, and sends responses.
- **Database (MongoDB)**: Stores and manages the application's data.

This architecture ensures a clear separation of concerns, where the frontend handles user interactions, the backend manages business logic and data processing, and the database stores persistent data.

1. Frontend (React)

Components:

- **Components**: This represents the various React components that make up the user interface. Components are modular and reusable pieces of the UI, such as buttons, lists, and forms.
  - **Home Page**: A component that likely serves as the main landing page or dashboard of the application.

    - **Watchlist**: A component that displays the stocks a user has added to their watchlist.

Flow:

- The frontend communicates with the backend to fetch data and update the user interface based on user actions.

2. Backend (Express)

REST API:

- **REST API**: Represents the set of endpoints exposed by the backend to handle requests from the frontend. These include:
    - **GET /api/stocks**: Retrieves a list of stocks from the database.
    - **POST /api/watchlist**: Adds a new stock to the watchlist in the database.

Stock Model:

- **Stock Model**: Defines the schema for stock documents in the MongoDB database. It specifies the structure of stock data, including fields like company, description, initial_price, and symbol.

Flow:

- The backend processes requests from the frontend, interacts with the database to perform CRUD operations, and sends responses back to the frontend.

3. Database (MongoDB)

MongoDB Database:

- **MongoDB Database**: Stores the application's data. In this case, it contains:
    - **Stock Collection**: Holds documents with information about each stock, such as company details, prices, and symbols.

Flow:

- The backend interacts with MongoDB to store and retrieve stock data based on API requests. MongoDB handles the data persistence and querying.

Visual Representation of Data Flow

1. **Frontend Interaction**:
    - The **React frontend** requests data from the **Express backend** via API endpoints.
    - The **Stocks component** fetches stock data from the /api/stocks endpoint and displays it to users.
    - When a user adds a stock to their watchlist, the **Watchlist component** makes a **POST** request to /api/watchlist to save the stock data.

2. **Backend Processing**:
   - The **Express backend** receives requests from the frontend and interacts with the **MongoDB database** to fetch or store data.
   - The backend processes **GET** requests to retrieve stock information and **POST** requests to add stocks to the watchlist.
3. **Database Operations**:
   - The **MongoDB database** stores the stock data and watchlist entries.
   - The backend uses Mongoose to define the **stock model** and perform database operations, ensuring data is saved and retrieved correctly.