# Stock Market Portfolio Application Documentation

## Introduction

The Stock Market Portfolio application is a MERN stack-based web tool designed for effective management and tracking of stock investments. It allows users to view stock details, add stocks to a watchlist, and monitor price changes. The application uses a MongoDB database to store stock information, which is critical for providing users with up-to-date and accurate data about their investments.

## Database Schema

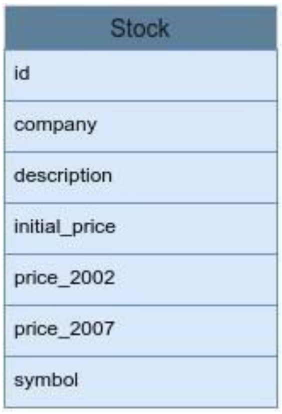
### MongoDB Schema:

The MongoDB schema is designed using Mongoose to structure the stock data. The schema includes a single collection, Stock, which serves both as the primary data store for stock details and for managing user watchlists. Here are the details of the schema:

- **id:** ObjectId (Primary Key)
  A unique identifier for each stock entry in the database.
- **company:** String
  The name of the company associated with the stock.
- **description:** String
  A brief description of the company or stock.
- **initial_price:** Number
  The price of the stock at the time of its initial listing.
- **price_2002:** Number
  The stock price in the year 2002.
- **price_2007:** Number
  The stock price in the year 2007.
- **symbol:** String
  The stock's ticker symbol.

## ER Diagram

The ER diagram reflects that the same Stock collection is used for both retrieving stock data and adding new entries (acting as a watchlist). There is no separate Watchlist collection; instead, the Stock collection serves this purpose.



- The same Stock collection is used for both retrieving stock data and adding new stock entries, acting as the watchlist. There is no separate Watchlist collection.

## Backend Code

**Server Setup:**

**javascript**

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const bodyParser = require("body-parser");
const app = express();
const PORT = process.env.PORT || 4000; // Changed the port number
// Middleware
app.use(cors());
app.use(bodyParser.json());
// MongoDB Connection URI
const mongoURI = "mongodb://localhost:27017/Stock_Market"; // Updated the database name
mongoose.connect(mongoURI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log("MongoDB connected"))
.catch(err => console.error("MongoDB connection error:", err));
// Schema and Model
const stockSchema = new mongoose.Schema({
  company: String,
  description: String,
  initial_price: Number,
  price_2002: Number,
  price_2007: Number,
  symbol: String,
});
```

```
const Stock = mongoose.model("Stock", stockSchema);
// Routes
app.get("/api/stocks", async (req, res) => {
  try {
    const stocks = await Stock.find();
    res.json(stocks);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: "Internal Server Error" });
  }
});
app.post("/api/watchlist", async (req, res) => {
  try {
    const {
      company,
      description,
      initial_price,
      price_2002,
      price_2007,
      symbol,
    } = req.body;
    const stock = new Stock({
      company,
      description,
      initial_price,
      price_2002,
      price_2007,
      symbol,
    });
    await stock.save();
res.json({ message: "Stock added to watchlist successfully" });
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: "Internal Server Error" });
    }
});

// Start Server
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

## MongoDB Queries

**Retrieve Data (GET):**

- **Endpoint:** GET /api/stocks
- **Description:** Fetches a list of all stocks from the Stock collection.
- **Postman Request:**
    - Method: GET
    - URL: http://localhost:4000/api/stocks
    - Headers: None needed
    - Body: Not applicable for GET requests

**Create Data (POST):**

- **Endpoint:** POST /api/watchlist
- **Description:** Adds a new stock to the Stock collection, which acts as the watchlist.
- **Request Body:**

json

Copy code

```
{
  "company": "Apple",
  "description": "Technology company",
  "initial_price": 150.25,
 "price_2002": 120.30,
  "price_2007": 175.50,
  "symbol": "AAPL"
}
```

- **Postman Request:**
    - Method: POST
    - URL: http://localhost:4000/api/watchlist
    - Headers: Content-Type: application/json
    - Body: Raw JSON data as shown above

## Data Storage and Retrieval

**Data Stored:**

The database stores detailed information about various stocks, which is crucial for users to manage and track their stock investments. Each stock entry in the database includes the following data:

- **_id:** A unique identifier (ObjectId) assigned to each stock entry. This field is automatically generated by MongoDB.
- **company:** The name of the company associated with the stock.
- **description:** A brief description of the company's operations or the stock itself.
- **initial_price:** The price of the stock at the time of its initial listing.
- **price_2002:** The stock price recorded in the year 2002.
- **price_2007:** The stock price recorded in the year 2007.
- **symbol:** The stock's ticker symbol used for trading.

**Data Entries:**

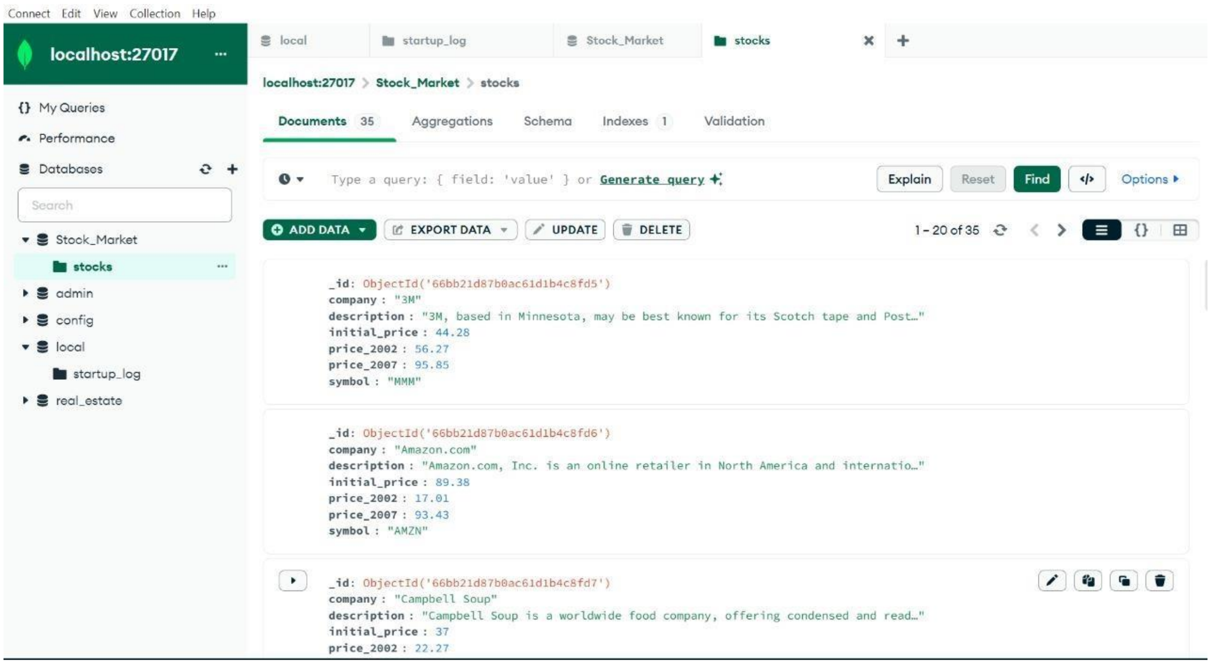
1. **Apple Inc.**
    - **company:** Apple Inc.
    - **description**: Technology company known for its iPhones, iPads, and Mac computers.
    - **initial_price:** 150.25
    - **price_2002:** 120.30
    - **price_2007:** 175.50
    - **symbol:** AAPL
2. **Microsoft Corporation**
    - **company**: Microsoft Corporation
    - **description:** Leading technology company specializing in software, hardware, and cloud services.
    - **initial_price:** 290.40
    - **price_2002:** 52.70
    - **price_2007:** 32.90

- **symbol:** MSFT

3. **Tesla, Inc.**
    - **company**: Tesla, Inc.
    - **description:** Electric vehicle and clean energy company.
    - **initial_price:** 720.50
    - **price_2002:** 14.40
    - **price_2007:** 33.70
    - **symbol:** TSLA
4. **Amazon.com, Inc.**
    - **company:** Amazon.com, Inc.
    - **description:** E-commerce and cloud computing giant.
    - **initial_price:** 3500.15
    - **price_2002:** 25.20
    - **price_2007:** 90.50
    - **symbol:** AMZN
5. **Google LLC (Alphabet Inc.)**
    - **company:** Google LLC
    - **description:** Multinational technology company known for its search engine and advertising services.
    - **initial_price:** 2700.75
    - **price_2002:** 100.10
    - **price_2007:** 400.50
    - **symbol:** GOOGL

**MongoDB Compass:**

Stocks Collection: The stocks data has stored in the stocks collection.



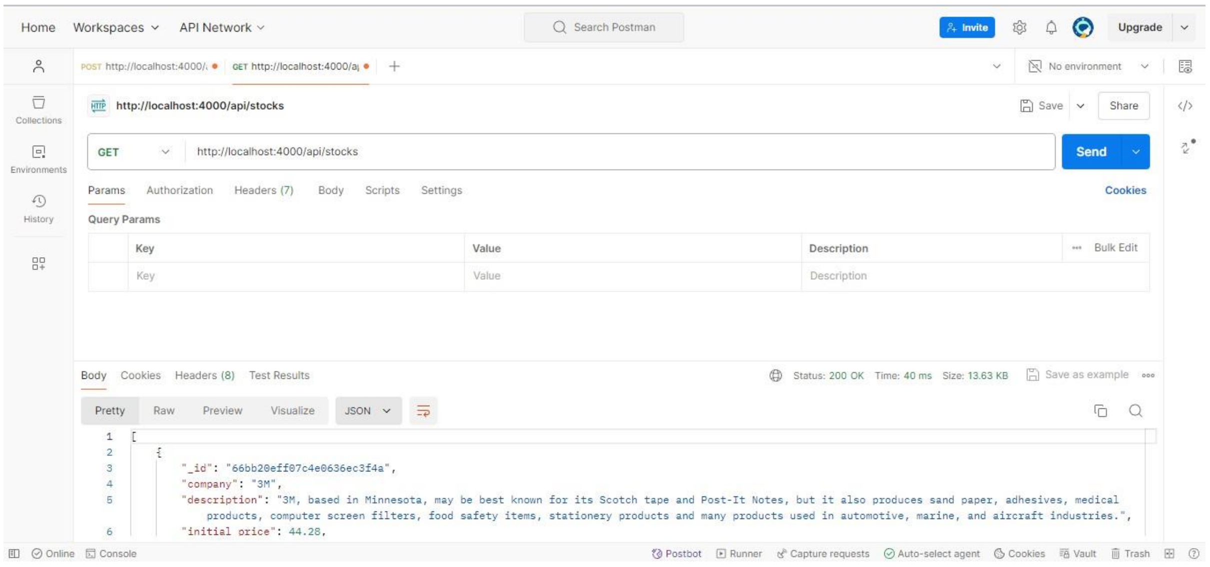
```
_id: ObjectId('66bb21d87b0ac61d1b4c8fd5')
company : "3M"
description : "3M, based in Minnesota, may be best known for its Scotch tape and Post…"
initial_price : 44.28
price_2002 : 56.27
price_2007 : 95.85
symbol : "MMM"

_id: ObjectId('66bb21d87b0ac61d1b4c8fd6')
company : "Amazon.com"
description : "Amazon.com, Inc. is an online retailer in North America and internatio…"
initial_price : 89.38
price_2002 : 17.01
price_2007 : 93.43
symbol : "AMZN"

_id: ObjectId('66bb21d87b0ac61d1b4c8fd7')
company : "Campbell Soup"
description : "Campbell Soup is a worldwide food company, offering condensed and read…"
initial_price : 37
price_2002 : 22.27
```

**Data Retrieval**

The GET /api/stocks endpoint retrieves all stock entries from the database and displays them on the Stocks page. Users can then add any stock to their watchlist by clicking the "Add to My Watchlist" button. This action triggers a POST request to the /api/watchlist endpoint, which adds the selected stock to the database.

**1.Retrieve Data (GET):**

**URL**: http://localhost:4000/api/stocks



GET http://localhost:4000/api/stocks

Status: 200 OK Time: 40 ms Size: 13.63 KB

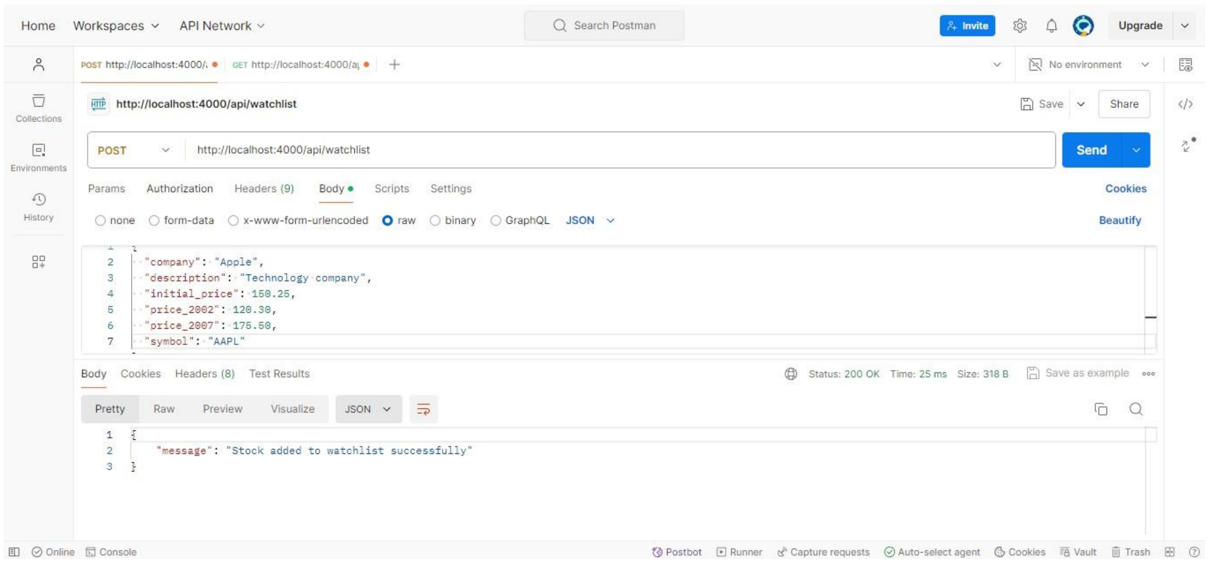
```
[
    {
        "_id": "66bb20eff07c4e0636ec3f4a",
        "company": "3M",
        "description": "3M, based in Minnesota, may be best known for its Scotch tape and Post-It Notes, but it also produces sand paper, adhesives, medical
           products, computer screen filters, food safety items, stationery products and many products used in automotive, marine, and aircraft industries.",
        "initial price": 44.28,
```

**2.Create Data (POST)**

**URL**: http://localhost:4000/api/watchlist

**Inserting the data using Postman:**

```
{
  "company": "Apple",
  "description": "Technology company",
  "initial_price": 150.25,
  "price_2002": 120.30,
  "price_2007": 175.50,
  "symbol": "AAPL"
}
```


POST http://localhost:4000/api/watchlist

Body: raw, JSON

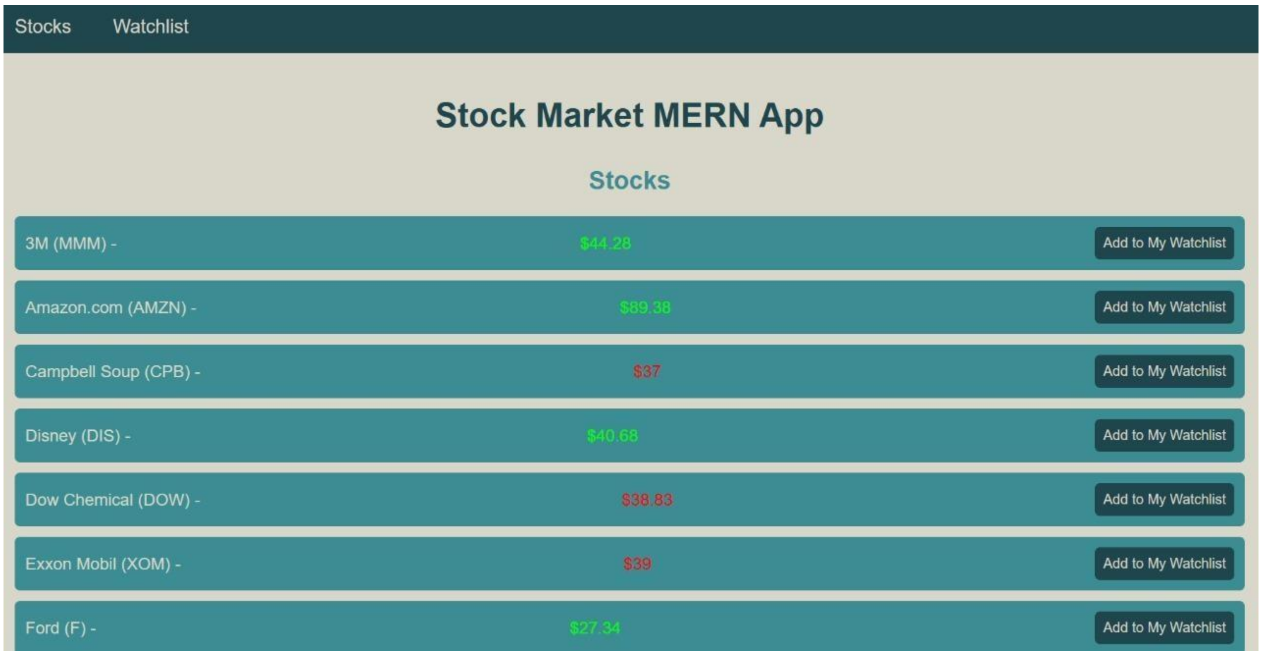
```
{
  "company": "Apple",
  "description": "Technology company",
  "initial_price": 150.25,
  "price_2002": 120.30,
  "price_2007": 175.50,
  "symbol": "AAPL"
```

Status: 200 OK Time: 25 ms Size: 318 B

```
{
    "message": "Stock added to watchlist successfully"
}
```

## Final Web Page:

The web page integrates with the backend to fetch stock data and update the watchlist. The Stocks page allows users to view available stocks and add them to their watchlist. The Watchlist page displays the stocks that have been added by the user.


Stocks Watchlist

Stock Market MERN App

Stocks

3M (MMM) - $44.28 Add to My Watchlist

Amazon.com (AMZN) - $89.38 Add to My Watchlist

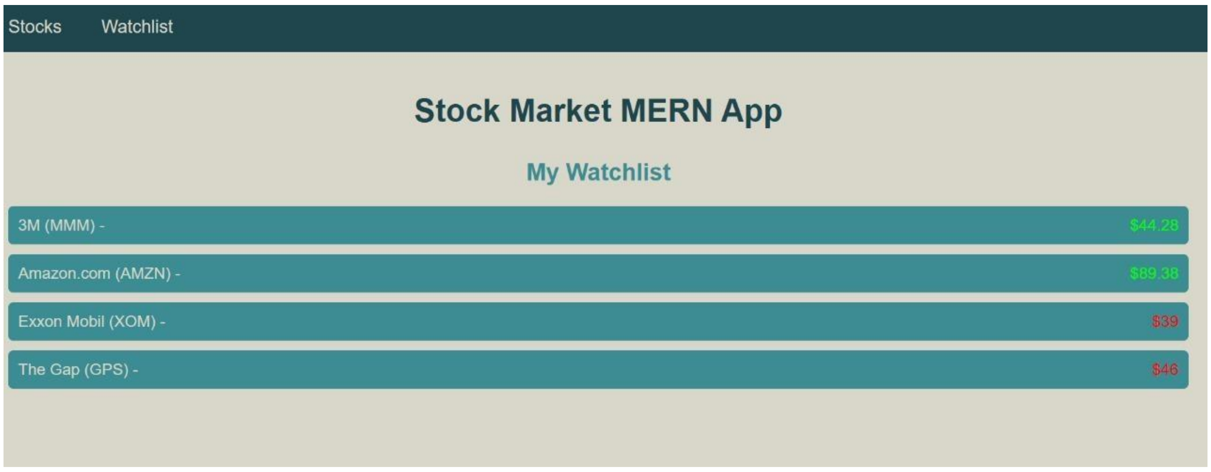
Campbell Soup (CPB) - $37 Add to My Watchlist

Disney (DIS) - $40.68 Add to My Watchlist

Dow Chemical (DOW) - $38.83 Add to My Watchlist

Exxon Mobil (XOM) - $39 Add to My Watchlist

Ford (F) - $27.34 Add to My Watchlist



Stock Market MERN App

My Watchlist

3M (MMM) - $44.28

Amazon.com (AMZN) - $89.38

Exxon Mobil (XOM) - $39

The Gap (GPS) - $46