Stock Tracker PWA - Documentation

The Stock Tracker is a progressive web app that lets you monitor stock prices in real time, set alerts for price thresholds, and view a graph of stock performance over time. It uses modern tools like React and TypeScript, and it's built with the MVC architecture to keep the code clean, organized, and maintainable. Below, I'll guide you through how it works and how everything fits together.

How the App is Organized

The app was developed using the MVC way of organizing code, this means that it's split into three main sections: Models, Views, and Controllers

The Models

Stock.ts defines that a stock has those exact properties. If we ever try to work with a stock that doesn't fit this format, TypeScript will catch the error.

The Views

The views are what make the app look nice and work smoothly for the user. Here are the main ones:

LeftForm: This is a form where you can pick a stock to watch and set an alert price. It's really simple: a dropdown menu to select the stock, a number input for the alert price, and a submit button. When you hit submit, it sends this data to the controller, which handles adding the stock to the list and subscribing to updates.

TopCards: These are the cards at the top of the app that show each stock you're tracking. If the current price is below your alert, the card is red; if it's above, it's green. This helps you instantly know if any of your stocks are performing poorly.

StockGraph: This is the part of the app where you can see how the stock prices have changed over time. It uses a charting library to make a simple, interactive graph.

The Controllers

The controllers are responsible for ensuring the app behaves as it should. Here's what they do:

Adding Stocks: When you submit the form to add a stock, the controller takes the stock symbol and alert price, adds it to the list of stocks you're watching, and subscribes to updates from the WebSocket.

Processing Updates: When the WebSocket sends a stock update, the controller updates the stock's current price and calculates how much it has changed. It also checks if the price is below the alert, and if it is, it sends you a notification.

Saving Data: To make sure you don’t lose your data when you close the app, the controller saves all the stock information to localStorage. When you reopen the app, it loads this data back in so you can pick up right where you left off.

WebSocket Service

The WebSocket is how the app gets real-time stock data. StockSerrvice.ts connects to the WebSocket and subscribes to updates for the stocks you’re watching. It has three main functions:

Subscribing to a Stock: This tells the WebSocket to start sending updates for a specific stock.

Unsubscribing: If you stop watching a stock, this tells the WebSocket to stop sending updates for it.

Listening for Updates: This is where the app listens for new stock data and passes it to the controller to process.

Notification Service

The notifications are handled by the notificationSerrvice.ts files and does two things:

Asking for Permission: When you first open the app, it asks for permission to send you notifications. This only happens once.

Sending Alerts: If a stock’s price drops below your alert level, it sends a notification to let you know.

What Happens When You Use the App

Here’s a quick summary of what happens when you use the app:

You pick a stock and set an alert price in the form. The app adds it to the list of stocks you’re watching and subscribes to updates for it.

The WebSocket sends real-time updates for the stock. The app processes these updates, updates the stock cards and graphs, and checks if any alerts need to be sent.

If you close and reopen the app, it loads the saved data from localStorage so you don’t lose your tracking history.