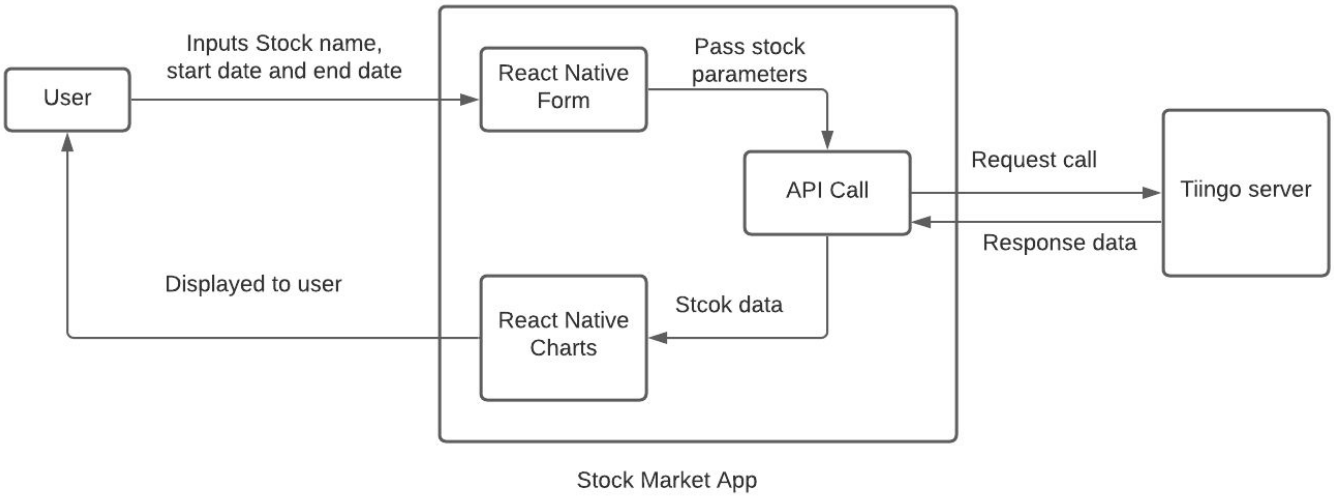
Stock Market Modelling App
REPORT

Contributors: Ankur Ingale, Arun Kumar, Garvit Mehta, Kommineni Nikhil, Tanya Chauhan

Problem Statement: To build a React Native mobile app which models stock data price.

High Level Diagram:



This project is based on Stock Market modelling. It is a mobile application developed using React Native which is a popular choice for cross platform development. The app currently fetches stocks data from an API and uses the fetched data to display a histogram of stock prices. The tools and technologies used in making the app as well as further improvements are explained below:

React Native is an open-source mobile application framework created by Facebook, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web , Windows and UWP by enabling developers to use React along with native platform capabilities. React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces.

The working principles of React Native are virtually identical to React except that React Native does not manipulate the DOM via the Virtual DOM. It runs in a background process (which interprets the JavaScript written by the developers) directly on the end-device and communicates with the native platform via a serialisation, asynchronous and batched Bridge. React components wrap existing native code and interact with native APIs via React's declarative UI paradigm and JavaScript. This enables native app development for whole new teams of developers, and can let existing native teams work much faster. React Native does not use HTML or CSS. Instead, messages from the JavaScript thread are used to manipulate native views. React Native also allows developers to write native code in languages such as Java for Android and Objective-C or Swift for iOS which make it even more flexible.

Some salient features of React Native are:

- **Written in JavaScript—rendered with native code** - React primitives render to native platform UI, meaning that the app uses the same native platform APIs other apps do. It allows us to create platform-specific versions of components so a single codebase can share code across platforms.
- **Native Development For Everyone** - React Native lets us create truly native apps and doesn't compromise your users' experiences. It provides a core set of platform agnostic native components like View, Text and Image that map directly to the platform's native UI building blocks.
- **Seamless Cross-Platform** - React components wrap existing native code and interact with native APIs via React's declarative UI paradigm and JavaScript. This enables native app development for whole new teams of developers, and can let existing native teams work much faster.
- **Fast Refresh** - It allows us to see the changes as soon as they are saved. With the power of JavaScript, React Native lets us iterate at a lightning speed.

Node.js

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.
This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our child_process.fork() API, and are designed to be easy to communicate with. Built upon that same interface is the cluster module, which allows you to share sockets between processes to enable load balancing over your cores.

When discussing Node.js, one thing that definitely should not be omitted is built-in support for package management using NPM, a tool that comes by default with every Node.js installation. The idea of NPM modules is quite similar to that of Ruby Gems: a set of publicly available, reusable components, available through easy installation via an online repository, with version and dependency management.

Some of the most useful npm modules today are:

- Express - Express.js—or simply Express—a Sinatra-inspired web development framework for Node.js, and the de-facto standard for the majority of Node.js applications out there today.
- hapi - a very modular and simple to use configuration-centric framework for building web and services applications
- Connect - Connect is an extensible HTTP server framework for Node.js, providing a collection of high performance “plugins” known as middleware; serves as a base foundation for Express.
- Socket.io and sockjs - Server-side component of the two most common websockets components out there today.
- pug (formerly Jade) - One of the popular templating engines, inspired by HAML, a default in Express.js.
- MongoDB and mongojs - MongoDB wrappers to provide the API for MongoDB object databases in Node.js.
- Redis - Redis client library.
- Lodash (underscore, lazy.js) - The JavaScript utility belt. Underscore initiated the game, but got overthrown by one of its two counterparts, mainly due to better performance and modular implementation.
- Forever - Probably the most common utility for ensuring that a given node script runs continuously. Keeps your Node.js process up in production in the face of any unexpected failures.
- Bluebird - A full featured Promises/A+ implementation with exceptionally good performance
- Moment - A JavaScript date library for parsing, validating, manipulating, and formatting dates.

Babel

Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments. Here are the main things Babel can do for you:
- Transform syntax
- Polyfill features that are missing in your target environment (through @babel/polyfill)
- Source code transformations (codemods)

Further Improvements

Some of the further improvements include:

We can add a back-end to support the calculation of various stock market indicators. Some of the simple yet powerful indicators are:

1. Simple Moving Average (SMA)
   The simple moving average is the average of the past k-days stock price values. It is to smoothen the price curve.
2. Exponential Moving Average (EMA)
   The exponential moving average is very similar to SMA but with added weights to the average calculation process. The weight of a price point data decreases exponentially as the data becomes older day by day.
3. Volume-Weighted Moving Average (VWAP)
   The volume-weighted moving average assigns weights to the price data points, proportional to the volume of the stocks sold at that price. This helps us estimate the average price for which the current stockholders bought each of their shares.

Meaningful insights can be drawn from these indicators. Generally, the stock is considered to be performing poorly when the 50-day moving average crosses the 200-day moving average in the downward direction.

The app can be further improved by adding stock price modeling features. Some of the modeling techniques are:

1. Black Scholes Models
   The stock price process can be modeled as a geometric Brownian motion. The parameters of the underlying Brownian motion can be calculated using the past data and the next price point can be predicted by sampling.
2. AR Models
   The stock price can be predicted using various regression techniques. Auto-Regression is one such technique that takes into account the time-series properties of the stock price process.