# Team 6 GitHub Documentation

## 1. Abstract

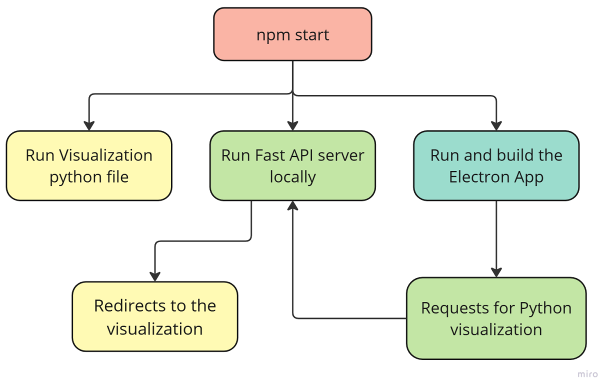
*A brief summary of the implementation to the project and objectives:*

- The main idea of the team was to develop and implement necessary visualizations to monitor stocks. Developed several stock charts with selectable options. Added options to overlay bullish and bearish technical indicators over the stock charts. The charts are completely interactable. Since the charts are implemented in python, the team worked on integrating the python script into the node application with the help of FastAPI and Concurrently packages.

## 2. Features Implemented

*A list of features developed, including descriptions and significance:*

- Visualizations:
  - Developed several stock charts including different types of Candlesticks, HLC Area, High-Low, Heikin Ashi, Renko, and Range.
  - Added indicators for each stock chart: Head and Shoulders, Inverse Head and Shoulders, Support, and Resistance.
  - Options to choose between any combination of stock chart and indicators.
  - Options to choose between various stocks.
  - Ability to select different time periods of the stock and zoom into any part of the chart.
- Integration of python code as API:
  - The charts were written in Plotly and Dash in python scripts. Implemented a local FastAPI server to run the Dash app. The stock charts run on port 8080. The FastAPI server runs on port 8000.
  - Starting scripts simultaneously run the server, the electron app and the python code using Concurrently. This also allows for python scripts from other groups to run on the local server.



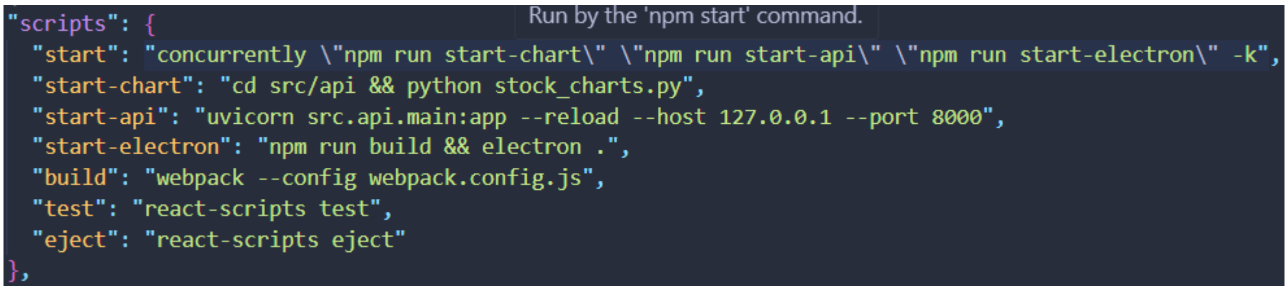
npm start
Run Visualization python file
Run Fast API server locally
Run and build the Electron App
Redirects to the visualization
Requests for Python visualization
miro

## 3. Code Highlights

*Key pieces of code or logic that is worth mentioning:*

- Most of the visualization logic rests in determining the range for time periods to view the stock and updating the chart with every user interaction. Refer the **`update_chart()`** function in **`stock_charts.py`** located at **`AutoProphet\auto-prophet\src\api\stock_charts.py`**

- Running multiple scripts of python and electron was possible with the Concurrently package, the figure below shows the startup scripts of the electron app.



```
"scripts": {
  "start": "concurrently \"npm run start-chart\" \"npm run start-api\" \"npm run start-electron\" -k",
  "start-chart": "cd src/api && python stock_charts.py",
  "start-api": "uvicorn src.api.main:app --reload --host 127.0.0.1 --port 8000",
  "start-electron": "npm run build && electron .",
  "build": "webpack --config webpack.config.js",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
```

## 4. Challenges and issues

*Possible challenges and issues arised or may arise in the future:*

- The python script, despite being implemented, requires users to have python installed on their machines.

## 5. Solutions and Fixes

*Strategies to fix possible challenges:*

- Possible solutions would be using something like Inno installers to bundle the app and python as an installer for users.
- Another approach would be to bundle necessary python required to run the scripts.

## 6. Future Work and suggestions

*Suggestions for further development or improvement of the project or the feature:*

- Implementing Database to store and retrieve custom visualizations made by users.
- Bundling python with the app to run on different operating systems.

## 7. Repository and Documentation Links

*URLs to any relevant documentation:*

- The code for the the entire feature is on the branch: https://github.com/jeffreywallphd/AutoProphet/tree/FALL2024-dev-almost-complete-BA5200
- The prototype version of the stock charts can be found at: https://github.com/jeffreywallphd/AutoProphet/tree/FALL2024-BA5200-Team6/prototypes/Fall%202024/Team%206
- FastAPI documentation to integrate other python scripts in the app: https://fastapi.tiangolo.com/