# Project Title

Stock Quant

# Team

* Constantinos Vogiatzis [cvogiatz@iu.edu](mailto:cvogiatz@iu.edu)
* Shubham Saurabh [ssaurab@iu.edu](mailto:ssaurab@iu.edu)
* Seth Smithson [sesmit@iu.edu](mailto:sesmit@iu.edu)

# Application Architecture

* **Data storage:** The historical stock data will be fetched directly from Yahoo Finance using their public API, and it won't be stored on any separate cloud server or standalone server. The data will be used on-the-fly inside Pandas Dataframe for processing and creating visualizations, predictions, and displaying results.
* **Back-end languages:** Python will be the primary language used for building the back-end of the app. It will be used for fetching data from the API, processing the data, and creating the stock price prediction model using the Prophet library.
* **Accessing data and web app:** The app will securely access the Yahoo Finance API. The app will be built using the Streamlit framework, which supports user authentication, but this app is for public use and we do not need authentication. The stock data being used in this app is already publicly available. We will rely on the built-in security of the Streamlit.io hosting platform to avoid malicious attacks.
* **Front-end layout:** Streamlit, the framework used for building the app, automatically generates the front-end layout based on the Python code. Although HTML, CSS, or JavaScript can be used in Streamlit, the primary layout will be created using Python code and Streamlit's built-in widgets. If needed, we can integrate Bootstrap templates using Streamlit's HTML and Markdown rendering support or Streamlit Components for more advanced customization, but we don’t anticipate that to be necessary.
* **Application deployment:** The app will be deployed on a Streamlit-provided server using Streamlit Sharing, which is a platform specifically designed for hosting and sharing Streamlit apps. Streamlit.io is Platform as a Service (PaaS) being used for the same.
* **Interactivity:** Streamlit provides a range of built-in interactive widgets, such as dropdowns, sliders, date pickers, and buttons, which will be used to enable user interactivity within the app. Users will be able to select stocks, compare them graphically, choose the date range for model testing and training, and set the prediction horizon. The app will also display the model's accuracy on training and testing data.

# Architecture Diagram

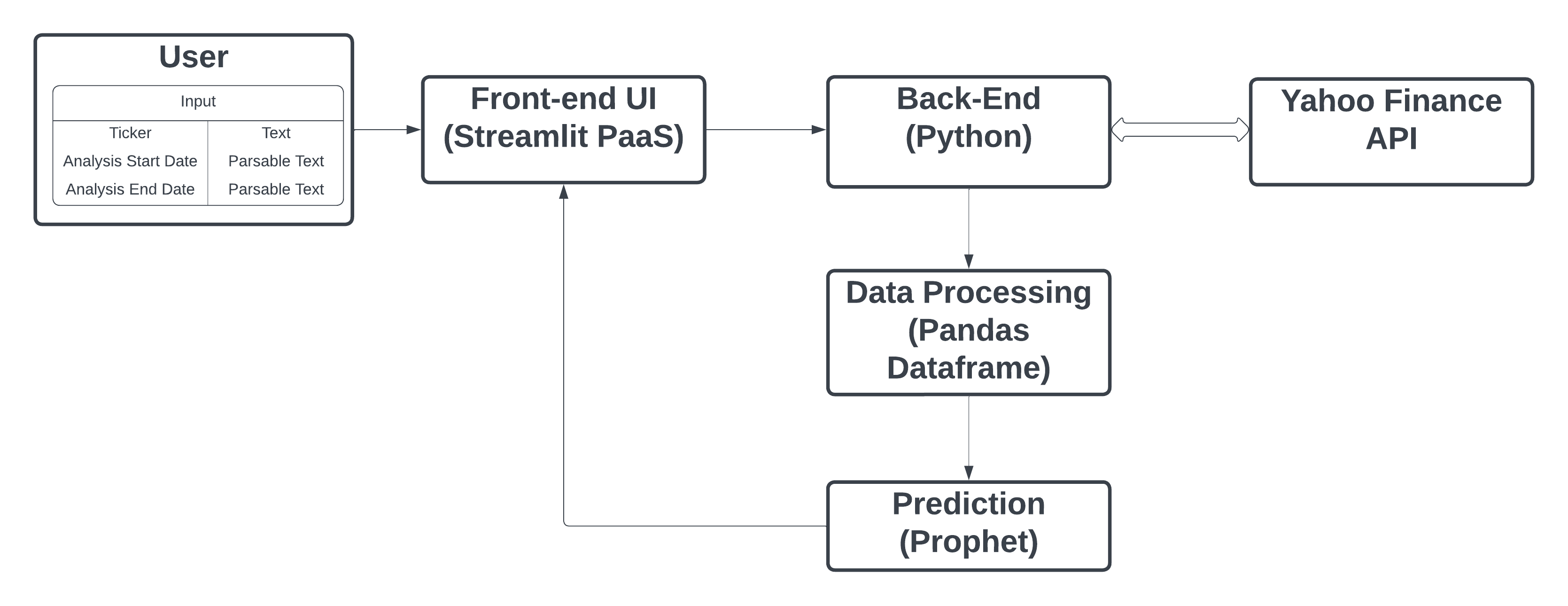


Diagram Explanation:

1. The User interacts with the Streamlit Front-end UI and selects the Ticker and Date range using Streamlit’s built-in widgets.
2. The Front-end communicates with the Python Back-end.
3. Python Back-end fetches stock data from the Yahoo Finance API.
4. The Yahoo Finance API is used as the primary data source, with data fetched directly and stored in a Pandas dataframe and processed on-the-fly.
5. The dataframe is loaded into the Prophet machine learning algorithm, which generates predictions which are then displayed in the Front-End UI for the user to examine.

# Web App Layout

The app is using a tabbed layout with three tabs:

1. **Stock Quant:** landing page describing the functionality.
2. **Compare Stocks:** allows the user to select up to 5 stocks and obtain key metrics and comparison plots between selected date ranges.
3. **Stock Price Forecast:** allows the user to specify the training date range to be used for analysis and obtain a forecast of the stock's price for a selected time period.

The app has a sidebar on the left which allows the user to navigate between the tabs. There are a total of three screens in the app as described above. The app will automatically detect and use either the dark or the light theme based on the user’s browser or system set-up.

## Screenshots

Figure 1 shows the landing page which a user will see when they first open the app. It mentions the purpose of the app and lists the functionalities being provided along with a short description of how to use it.

Text

Description automatically generated

Fig. 1 – Home page with purpose and directions

Figure 2 shows the Compare Stocks screen where the user sees the options to add up to 5 NYSE stock codes. The app prepopulates all 5 options with default values. The user can either make them blank or overwrite with their desired stock codes. The next input is for the user to specify their desired date range within which they want to make the comparisons. The app will warn the user in case of an invalid input.

The output on this screen will be a plot of all stocks along the same axis. These will be color-coded, and a legend will be shown. We also aim to provide some more comparison stats like maximum value, minimum value, and average value within the selected date range.

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 2 – Visualization and Comparison of Stocks

Figure 3 shows the third screen of the app where the user can specify any valid stock code, date range of training data, and date range of forecast. The outputs will be a plot of the forecasted data including a confidence interval band, and also a plot of the trend observed in the stock’s price.

A screenshot of a computer

Description automatically generated

Fig. 3 – Stock Price Forecasting

# Teamwork

All team members contributed to the project as planned. Primary responsibilities were as follows:

* Shubham – create the mock-up for the screenshots.
* Costas – write-up the architecture.
* Seth – prepare the architecture diagram.

The team met ~2 hours every week to review and finalize all the deliverables.

# Milestones

|  |  |  |
| --- | --- | --- |
| **Schedule** | **Milestone** | **Status** |
| Week 7 | Project planning + dataset description | Complete |
| Week 11 | TS analysis/forecast | Complete |
| Week 13 | Dashboard mock | Complete |
| Week 14 | Dashboard demo (5 min recording) | TBD |
| Week 15 | URL + code + final documentation | TBD |