

LAB 2 : Report

Sarvesh Waghmare 018319262

The Stock Analyzer application *are* designed to manage a stock portfolio, retrieve historical stock datas, generate analytical report, create visual chart, and handle data import/export via CSV file. The applications architecture *were* composed of *an* following key files:

- **stock.py:**
 - Define the core data model for an application.
 - Stock:
 - Represent an individual stock, encapsulate attribute such as symbol, name,
 - share, and a chronological lists of daily_data.
 - DailyData:
 - Store specific daily stock metric, include date, closing_price, and volumes.
- **stock_data.py:**
 - Contain module responsible for data acquisitions and importation.
 - retrieve_stock_web:
 - Fetch historical stock data from Yahoo! Finance, leverage selenium
 - for browser automation and BeautifulSoup for HTML parse.
 - import_stock_web_csv:
 - Facilitate a import of stock data from a CSV file into Stock object.
- **stock_console.py:**
 - Implement the console-based user interface for a application.
- **stock_GUI.py:**
 - Implement a graphical user interface (GUI) use the tkinter librarys.

Dependencies

The application *rely* on several external *library*:

- **selenium**
 - and BeautifulSoup4 (bs4) for web scrape functionality.
- **matplotlib**
 - for generate chart.

- **tkinter**
 - for a GUI version (typically include with standard Python distribution).
- **chromedriver**
 - (or a similar WebDriver for other browser) *are* required for selenium too
 - interface with an web browser.

Step-by-Step Walkthrough of Features

Feature 1: Manage Stocks

Allow users to add, update shares of, delete, and list stock in *they're* portfolio.

Console (stock_console.py)

Steps:

1. Main Menu:

- display_main_menu *print* options; user *enter* 1 for "Manage Stocks".
- Code: choice = self.display_main_menu() → if choice == "1": self.manage_stocks_menu().

```
Stock Analyzer ----
1 - Manage Stocks (Add, Update, Delete, List)
2 - Add Daily Stock Data (Date, Price, Volume)
3 - Show Report
4 - Show Chart
5 - Manage Data (Save, Load, Retrieve)
0 - Exit Program
Enter Menu Option: █
```

2. Manage Stocks Menu:

- manage_stocks_menu *present* option: Add (1), Update (2), Delete (3), List (4), Exit (0).
- Example: User *enter* 1 (Add Stock).
- Code: if choice == "1": self.add_stock().

3. Add Stock:

- add_stock *prompt* for ticker, name, shares (e.g., AAPL,GOOG).
- A Stock object *are* created and *append* to self.stock_list.
- Code: stock = Stock(symbol, name, shares); self.stock_list.append(stock).
- *Prompt* to add another or exits.

```

Manage Stocks ----
1 - Add Stock
2 - Update Shares
3 - Delete Stock
4 - List Stocks
0 - Exit Manage Stocks
Enter Menu Option: 1

Add Stock ----
Enter Ticker Symbol: APPL
Enter Company Name: Apple
Enter Number of Shares: 120
Stock Added - Enter to Add Another Stock or 0 to Stop: 0

```

4. Update Shares:

- User select 2 → update_shares_menu (Buy (1), Sell (2), Exit (0)).
- Buy Shares (buy_shares): Show stock list; *prompt* for symbol, shares (e.g., AAPL, 50). *Update* stock.shares += shares.
- Sell Shares (sell_shares): Similar to buy, but *validate* against owned share (e.g., AAPL, 30). *Update* stock.shares -= shares.

```

Update Shares ----
1 - Buy Shares
2 - Sell Shares
0 - Exit Update Shares
Enter Menu Option: 2

Sell Shares ----
Stock List: ['AAPL']
Which stock do you want to sell?: AAPL
How many shares do you want to sell?: 60
Updated shares for AAPL: 60.0

```

5. Delete Stock:

- User selects 3 → delete_stock. *Prompts* for symbol (e.g., AAPL). *Remove* stock from self.stock_list.
- Code: self.stock_list.pop(i).

```

Manage Stocks ----
1 - Add Stock
2 - Update Shares
3 - Delete Stock
4 - List Stocks
0 - Exit Manage Stocks
Enter Menu Option: 3

Delete Stock ----
Stock List: ['AAPL', 'GOOG']
Which stock do you want to delete?: GOOG
Stock GOOG deleted.

```

6. List Stocks:

- User selects 4 → list_stocks. *Print* a formatted table: SYMBOL NAME SHARES.
- Example Output: AAPL APPLE 120.

```

Manage Stocks ----
1 - Add Stock
2 - Update Shares
3 - Delete Stock
4 - List Stocks
0 - Exit Manage Stocks
Enter Menu Option: 4

Stock List ----
SYMBOL  NAME      SHARES
AAPL    APPLE      60.0
Press Enter to Continue ***

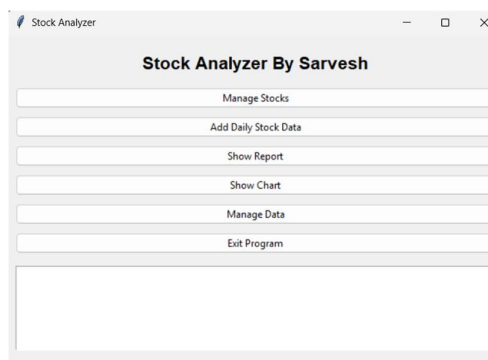
```

GUI (stock_GUI.py)

Steps:

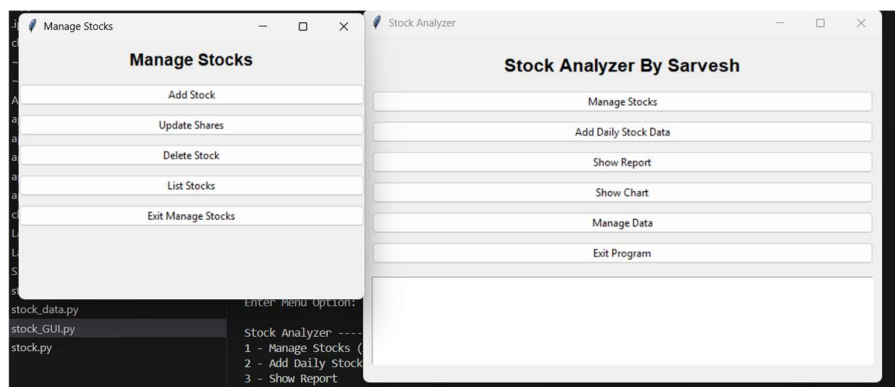
1. Main Window: User *click* the "Manage Stocks" button.

- Code: `ttk.Button(..., command=self.manage_stocks_menu) call manage_stocks_menu.`



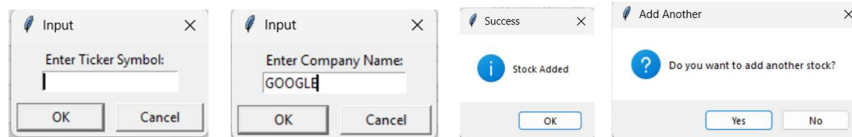
2. Manage Stocks Window:

- `manage_stocks_menu` *open* a Toplevel window with buttons: Add Stock, Update Shares, Delete Stock, List Stocks, Exit.
- User *click* "Add Stock".
- Code: `ttk.Button(..., command=lambda: [manage_window.destroy(), self.add_stock()])`.



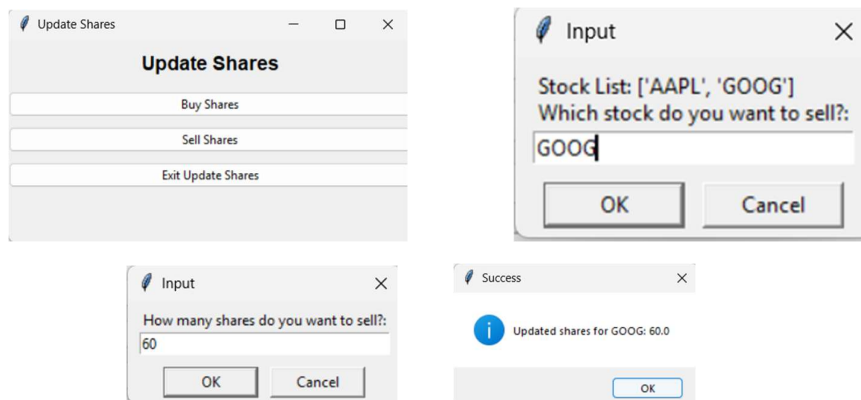
3. Add Stock:

- `add_stock` use `simplifiedialog` to get ticker, name, shares (e.g., AAPL, APPLE, 100).
- Create Stock object, *add* to `self.stock_list`. (Code: `stock = Stock(symbol, name, shares)`).
- Show success message; `messagebox.askyesno("Add Another", ...)` to add more.



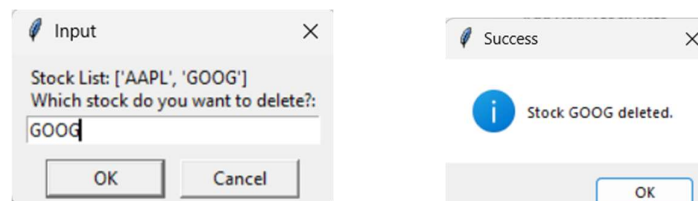
4. Update Shares:

- User *clicks* "Update Shares" → `update_shares_menu` *open* a new window.
- Buy Shares (`buy_shares`): Dialogs *prompt* for symbol, shares (e.g., AAPL, 50). *Updates* shares, *show* success.
- Sell Shares (`sell_shares`): Similar, with *validations*. *Update* shares (e.g., AAPL, 30).



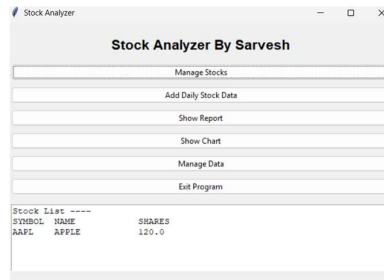
5. Delete Stock:

- User *clicks* "Delete Stock" → `delete_stock`. Dialog *prompt* for symbol. *Removes* stock, *show* success.



6. List Stocks:

- User *clicks* "List Stocks" → `list_stocks`. *Display* the stock list in the main window Text widget.
- Code: `self.display_output(output)` with formatted *tables*.



Feature 2: Add Daily Stock Data

Fetch historical stock data from Yahoo! Finance for a specific stock within a date *ranges*.

Console

Steps:

1. **Main Menu** → User *enter* 2 → add_daily_stock_data.
2. **Prompts for stock symbol, start date, end date** (e.g., AAPL, 05/02/25, 05/09/25).
3. *Call* stock_data.retrieve_stock_web(dateFrom, dateTo, [stock]) to fetch data.
4. *Displays* fetched data in a table format: Date Price Volume.
 - Example Output: 05/08/25 197.49 50369700.

```

Stock Analyzer ----
1 - Manage Stocks (Add, Update, Delete, List)
2 - Add Daily Stock Data (Date, Price, Volume)
3 - Show Report
4 - Show Chart
5 - Manage Data (Save, Load, Retrieve)
0 - Exit Program
Enter Menu Option: 2

Add Daily Stock Data ----
Stock List: ['AAPL']
Which stock do you want to use?: AAPL
Fetching data for: AAPL
Enter starting date (MM/DD/YY): 05/02/25
Enter ending date (MM/DD/YY): 05/09/25

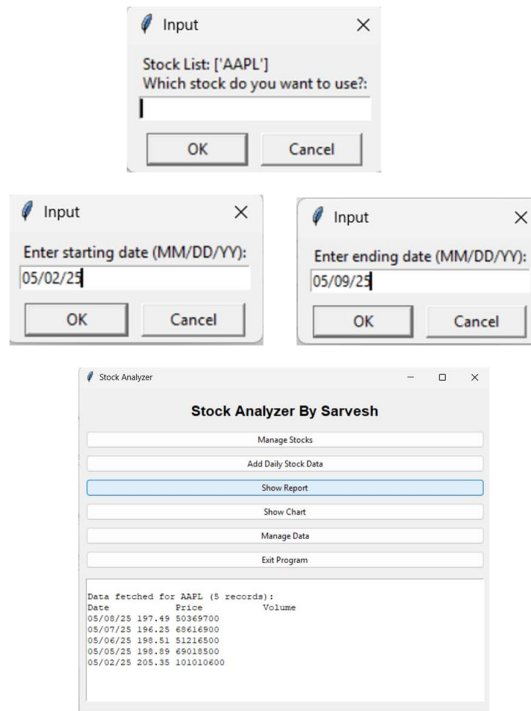
Data fetched for AAPL (5 records):
Date      Price      Volume
05/08/25   197.49    50369700
05/07/25   196.25    68616900
05/06/25   198.51    51216500
05/05/25   198.89    69018500
05/02/25   205.35    101010600
Press Enter to Continue ***

```

GUI

Steps:

1. **Main Window** → User *clicks* "Add Daily Stock Data" button → add_daily_stock_data.
2. **Dialog box prompt for symbol, start date, end date** (e.g., AAPL, 05/02/25, 05/09/25).
3. *Fetch* data using stock_data.retrieve_stock_web.
4. *Display* the data in the main Text widget via self.display_output(output).



Feature 3: Show Report

Generate and display a report of all stocks in the portfolio and they're daily historical data.

Console

Steps:

1. **Main Menu** → User enters 3 → show_report.
2. *Loop* through self.stock_list, printing detail for each stock *include* share count and daily data.
 - Example Output:
 - Report for: AAPL APPLE
 - Shares: 120
 - Date: 05/08/25 197.49 50369700.

```

Stock Analyzer ----
1 - Manage Stocks (Add, Update, Delete, List)
2 - Add Daily Stock Data (Date, Price, Volume)
3 - Show Report
4 - Show Chart
5 - Manage Data (Save, Load, Retrieve)
0 - Exit Program
Enter Menu Option: 3

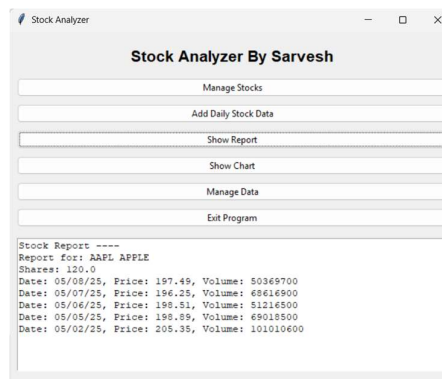
Stock Report ----
Report for: AAPL APPLE
Shares: 60.0
Date: 05/08/25, Price: 197.49, Volume: 50369700
Date: 05/07/25, Price: 196.25, Volume: 68616900
Date: 05/06/25, Price: 198.51, Volume: 51216500
Date: 05/05/25, Price: 198.89, Volume: 69018500
Date: 05/02/25, Price: 205.35, Volume: 101010600
---- Report Complete ----
Press Enter to Continue

```

GUI

Steps:

1. **Main Window** → User *click* “Show Report” button → show_report.
2. *Execute* the same underlying logic as a console version but *direct* the formatted output to the Text widget use self.display_output(output).



Feature 4: Show Chart

Generate a price chart for a selected stock use matplotlib.

Console

Steps:

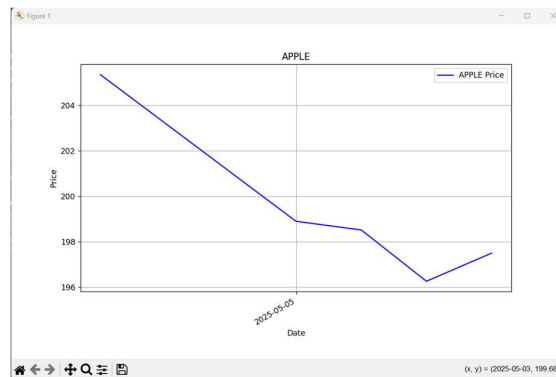
1. **Main Menu** → User *enter* 4 → show_chart.
2. **Prompts for stock symbol** (e.g., AAPL).
3. **Parse dates and prices** from stock.daily_data and use matplotlib to create a plot.
 - Code: plt.plot(dates, prices, 'b-', label=f"{stock.name} Price").
4. **Display the chart** in a new matplotlib window and typically save it as an image file (e.g., PNG).


```

Stock Analyzer -----
1 - Manage Stocks (Add, Update, Delete, List)
2 - Add Daily Stock Data (Date, Price, Volume)
3 - Show Report
4 - Show Chart
5 - Manage Data (Save, Load, Retrieve)
0 - Exit Program
Enter Menu Option: 4

Show Chart -----
Stock List: ['AAPL']
Which stock do you want to use?: AAPL

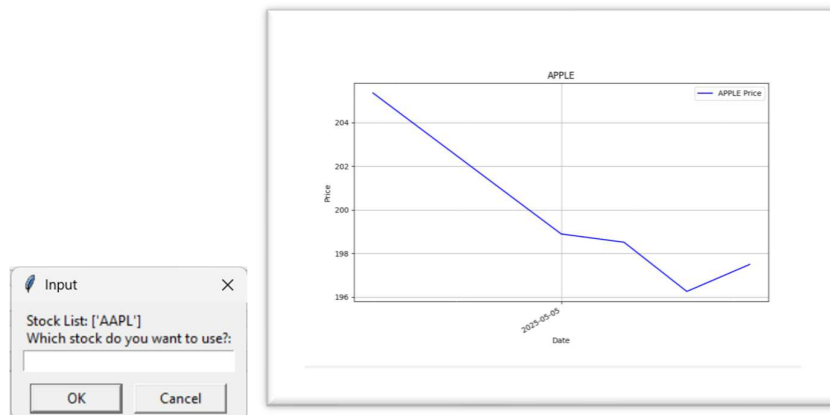
```



GUI

Steps:

1. **Main Window** → User *click* "Show Chart" button → show_chart.
2. **A dialog box *prompt* for the stock symbol** (e.g., AAPL).
3. *Utilize* the same matplotlib plotting logic as the console version, *display* the chart in a new windows.



Feature 5: Manage Data

Handle data operations such as saving to/loading from CSV, retrieving data from a web, and importing from a CSV file.

Console

Steps:

1. **Main Menu** → User enters 5 → `manage_data_menu`.
2. **Present submenu: Save (1), Load (2), Retrieve (3), Import (4), Exit (0).**
3. **Save Data to Database (`save_to_database`):**
 - Typically *fetch* current data for all stocks use `retrieve_stock_web`.
 - Saves data to a CSV file, prompting *an* user for a filename (e.g., `apple_4.csv`).

```
Manage Data ----
1 - Save Data to Database
2 - Load Data from Database
3 - Retrieve Data from Web
4 - Import from CSV File
0 - Exit Manage Data
Enter Menu Option: 1
Saving to database...
Data saved to temporary file. Please enter a filename to save permanently:
Enter filename: apple_4
Data saved to apple_4.csv
Press Enter to Continue
```

4. **Load Data from Database (`load_from_database`):**

- Prompts for filename, symbol, start/end dates (e.g., `apple_4.csv`, `AAPL`, `05/02/25`, `05/09/25`).
- Load and display the filtered data.

```
Manage Data ----
1 - Save Data to Database
2 - Load Data from Database
3 - Retrieve Data from Web
4 - Import from CSV File
0 - Exit Manage Data
Enter Menu Option: 2

---- Data Loaded from Database ----
Enter CSV filename to load: apple_4.csv
Enter stock symbol to load: AAPL
Enter starting date (MM/DD/YY): 05/02/25
Enter ending date (MM/DD/YY): 05/09/25

Data loaded for AAPL within 05/02/25 to 05/09/25:
Date      Price      Volume
05/08/25   197.49   50369700
05/07/25   196.25   68616900
05/06/25   198.51   51216500
05/05/25   198.89   69018500
05/02/25   205.35  101010600
05/08/25   197.49   50369700
05/07/25   196.25   68616900
05/06/25   198.51   51216500
05/05/25   198.89   69018500
05/02/25   205.35  101010600
Press Enter to Continue
```

5. **Retrieve Data from Web (`retrieve_from_web`):**

- Prompts for start/end dates (e.g., `05/02/25`, `05/09/25`).
- Fetch data for all stocks in *an* portfolio and report the number of records retrieved (e.g., Records Retrieved: 5).

```

Manage Data ----
1 - Save Data to Database
2 - Load Data from Database
3 - Retrieve Data from Web
4 - Import from CSV File
0 - Exit Manage Data
Enter Menu Option: 3

Retrieving Stock Data from Yahoo! Finance ----
This will retrieve data from all stocks in your stock list.
Enter starting date (MM/DD/YY): 05/02/25
Enter ending date (MM/DD/YY): 05/09/25
Records Retrieved: 5
Press Enter to Continue

```

6. Import from CSV File (import_csv):

- Prompts for filename and symbol (e.g., apple_4.csv, AAPL).
- Import data from a specified CSV and display it or a confirmation messages.

```

Manage Data ----
1 - Save Data to Database
2 - Load Data from Database
3 - Retrieve Data from Web
4 - Import from CSV File
0 - Exit Manage Data
Enter Menu Option: 4

Import CSV file from Yahoo! Finance ----
Enter CSV filename to load: apple_4.csv
Enter stock symbol to load: AAPL
Enter starting date (MM/DD/YY): 05/02/25
Enter ending date (MM/DD/YY): 05/09/25

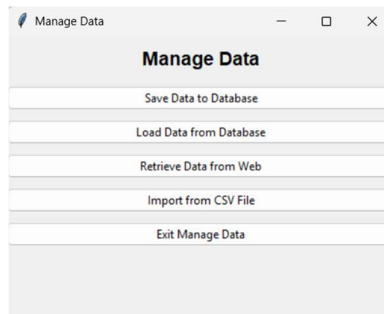
Data loaded for AAPL within 05/02/25 to 05/09/25:
Date          Price      Volume
05/08/25      197.49    50369700
05/07/25      196.25    68616900
05/06/25      198.51    51216500
05/05/25      198.89    69018500
05/02/25      205.35    101010600
05/08/25      197.49    50369700
05/07/25      196.25    68616900
05/06/25      198.51    51216500
05/05/25      198.89    69018500
05/02/25      205.35    101010600
Press Enter to Continue

```

GUI

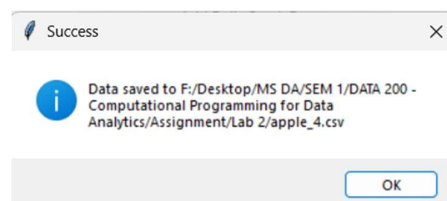
Steps:

1. **Main Window** → User clicks "Manage Data" button → manage_data_menu.
2. **Open** a new window with buttons: Save, Load, Retrieve, Import, Exit.



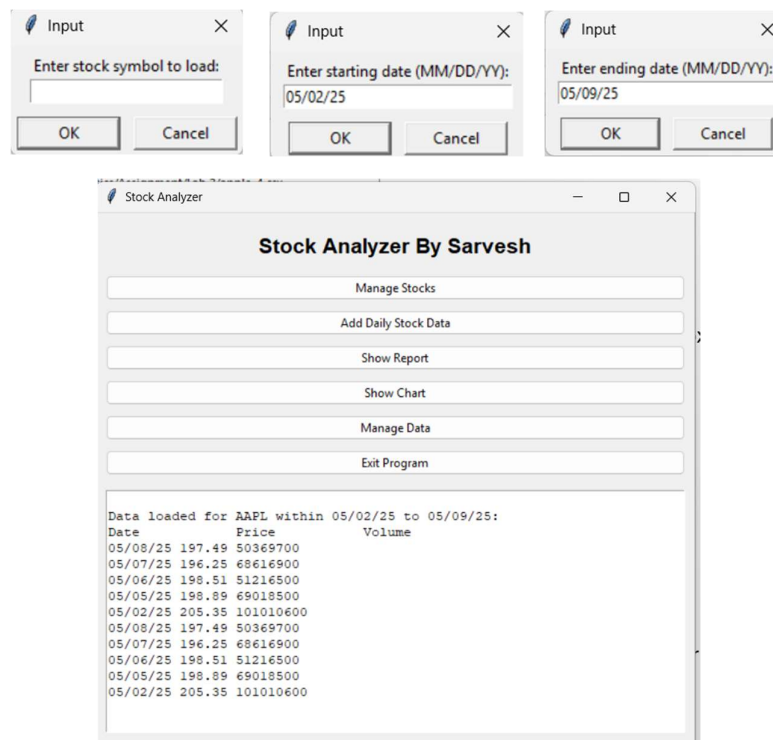
3. Save Data to Database:

- Fetches data, then *open* a standard file save dialog (filedialog.asksaveasfilename(...)) for *an* user to specify a filename and location (e.g., apple_4.csv).



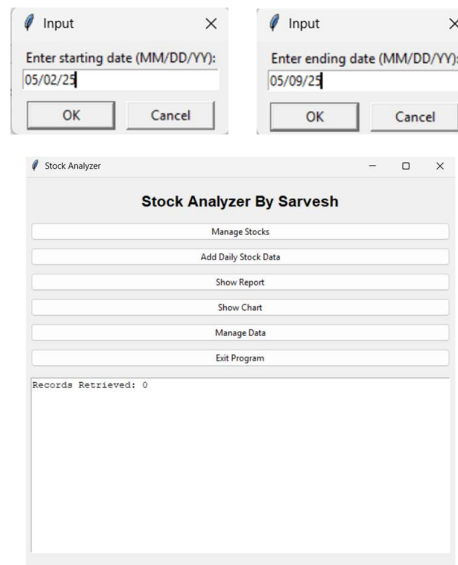
4. Load Data from Database:

- Use a file open dialog for *select* the CSV file, followed *by* dialog boxes for symbol and date range input.
- Displays* loaded data in a Text widget.



5. Retrieve Data from Web:

- Use dialog boxes to get start/end dates, *fetch* data, and *show* the record count, typically in a message *boxes*.



6. Import from CSV File:

- Use a file open dialog for CSV selection, a dialog for a stock symbol, and then *display* imported data or a confirmation in a Text widget or a message box.

