**Problem Statement** :

These days, stock market trading is not for institutional investors or big investors alone. Individual investors and small investors are actively trading and managing their portfolios too. But most of the available platforms are either too complicated, expensive, or need internet connectivity with live servers. For teaching and training purposes, an easy-to-use, offline simulation system for stock trading is necessary to enable users to learn the process of registration, buying and selling, and portfolio management in a secure environment.

The project addresses this requirement by creating a Tkinter-based desktop application to mimic elementary stock market operations in Python. Rather than being connected to live stock exchanges, it holds and reads data locally from JSON files (users.json and registrations.json). This makes it suitable for students, new users, and instructional use.

**Need of the System**

To develop a simple and easy-to-use platform for mimicking stock market transactions.

To enable registration, login, and secure management of accounts by users.

To enable buying and selling of shares across industries (Automobile, Petroleum, Steel).

To keep activity records of users in JSON-based databases to make data management convenient.

To have an Admin Panel to manage and view all registered users.

**Importance of the System**

Educational Value – Enables students to grasp stock market principles without involving actual money.

Data Handling – Exhibits hands-on application of file handling and JSON data manipulation in Python.

GUI Learning – Introduces the students to Tkinter GUI programming.

Security & Validation – Enforces login, registration, and admin authentication functionality.

Scalability – The system can expand to add additional sectors, live APIs, and complex analytics in the future.

**Technologies Used**

**Front-End Technology**

* **Python Tkinter**
  + Used for designing the **Graphical User Interface (GUI)**.
  + Provides windows, labels, buttons, entry fields, and other widgets.
  + Makes the application user-friendly for registration, login, buy/sell, and admin management.
* **PIL (Pillow Library)**
  + Used to load and display images (like logos and icons) inside the GUI.
* **WebBrowser module (optional)**
  + Opens stock information links (Honda, Hyundai, Reliance, ONGC, etc.) directly in a web browser when user clicks on a company.

**Back-End Technology**

* **Python (Core Logic)**
  + Handles business logic like registration, login authentication, buying/selling shares, updating balances, etc.
* **JSON Files** (users.json and registrations.json)
  + Used as the **Database (Data Storage)**.
  + users.json → Stores user details, balance, and share portfolio.
  + registrations.json → Stores registration records for admin viewing.
* **File Handling in Python**
  + Reads and writes JSON files for saving and loading user data.

**Additional Technologies**

* **Operating Environment**: Runs on **Python 3.x** (cross-platform).
* **Libraries Used**:
  + tkinter → GUI
  + PIL (Image, ImageTk) → Image handling
  + json → Data storage and retrieval
  + os → File management
  + webbrowser → Open stock links

**System Requirements**

**1. Hardware Requirements**

* **Processor**: Minimum Intel i3 (or equivalent AMD); Recommended Intel i5/i7.
* **RAM**: Minimum 4 GB; Recommended 8 GB or higher.
* **Storage**: At least 200 MB of free disk space (for Python installation, JSON files, and images).
* **Display**: 1024×768 resolution or higher for proper GUI display.
* **Input Devices**: Keyboard and Mouse (for GUI navigation).

**2. Software Requirements**

* **Operating System**:
  + Windows 10/11 (recommended)
  + Linux or macOS (also supported, since Python & Tkinter are cross-platform).
* **Programming Language**:
  + **Python 3.10 or above** (your code runs on Python 3.12 as per traceback).
* **Python Libraries (Dependencies)**:
  + tkinter (built-in with Python).
  + PIL / Pillow (for image processing).
  + json (built-in, for data storage).
  + os (built-in, for file handling).
  + webbrowser (built-in, for opening stock links).
* **IDE / Editor**:
  + Any Python-supported IDE like **VS Code, PyCharm, Jupyter Notebook, or IDLE**.

**3. Database Requirements**

* **No external database required**.
* Uses **JSON files** (users.json and registrations.json) for storing user data and registration details.

**Software process model pertaining for Invest Karo**

The Incremental Software Process Model was chosen for this project because the application was developed in small functional parts, or increments, rather than as one large system. Each module (Registration, Login, Buy/Sell, Admin Panel, Portfolio Management) was designed, implemented, tested, and integrated step by step.

This approach was most suitable because:

**Step-by-Step Development**

First increment: Basic Registration and Login System.

Second increment: JSON-based database integration.

Third increment: Buying and Selling of shares.

Fourth increment: Portfolio management and balance update.

Fifth increment: Admin Panel and stock links integration.

**NOTE: It is evident from the above discussion that step-by-step function modules will be developed and tested separately, and integrated into a complete system. Therefore, Incremental model is best suited for the proposed project.**

**Flexibility**

This approach allowed for adding features, such as web browser links for stock info or sector-wise companies, without redesigning the entire system.

**User Feedback**

It was easy to test and improve at every stage by running small modules, like the login first and then the portfolio.

**Low Risk and Easy Maintenance**

Since the project is designed for academic purposes, incremental delivery made sure there was always at least a working prototype available.

**Why not Waterfall Model?**

Waterfall requires all requirements to be fixed before coding starts.

In this project, features were added gradually, starting with login, then portfolio, then admin. This gradual approach is not possible in Waterfall.

**Why not Spiral or Agile?**

Spiral is better for large-scale, high-risk projects with ongoing client involvement.

Agile requires team collaboration with sprints, while this is a small-scale academic project often developed individually.