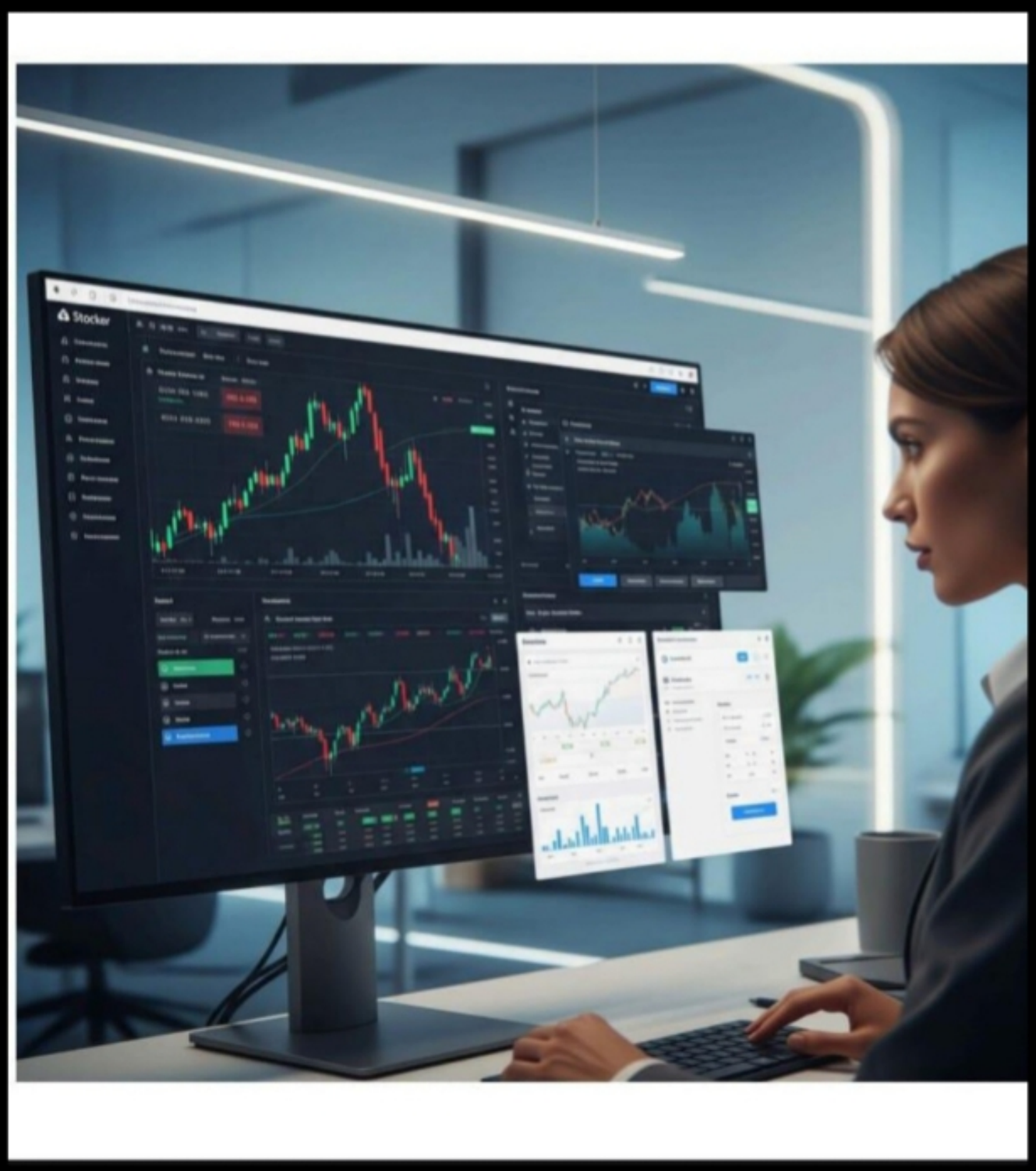
Real-Time Stock Trading Dashboard with AWS Service



1. Introduction

Stock trading platforms are often complex and risky for beginners because they involve real money and advanced trading concepts. Many new users want to learn how stock markets work but do not have access to a safe and simple learning environment.

This project aims to build a simulated real-time stock trading dashboard where users can practice trading using virtual money. The platform helps users understand stock trading concepts such as buying, selling, and

2. Problem Statement

Many aspiring investors lack access to user-friendly platforms that simulate stock trading without involving real money. Existing platforms are complex and intimidating for beginners, which makes learning difficult and risky.

3. Proposed Solution

The proposed system is a simulated real-time stock trading dashboard that allows users to:

Register and log in

Receive virtual currency

View simulated stock prices

Buy and sell stocks

Manage a virtual portfolio

This system eliminates real financial risk while providing practical trading experience.

4. Objectives of the Project

To provide a risk-free environment for learning stock trading

To simulate real-world trading operations

To demonstrate buy/sell logic implementation

To introduce cloud-based architecture concepts

5. Core Functionalities (MVP)

User registration and authentication

Virtual wallet balance allocation

Simulated stock price display

Buy and sell stock functionality

Portfolio tracking

Transaction history

Trade confirmation notifications (conceptual)

6. Scope of the Project

Current Scope

- Virtual trading system
- Simulated stock prices
- Basic portfolio management

Future Enhancements

- Real-time stock price APIs
- Advanced charts and analytics
- News feeds and watchlists
- Mobile application support

7. Technology Stack

Frontend

- HTML
- CSS
- Bootstrap

Backend

- Python (Flask Framework)

Database

- SQLite (development)
- DynamoDB (AWS – conceptual)

Cloud Services

- Amazon EC2
- AWS SNS
- AWS IAM

8. System Architecture

User → Web Browser → Flask Backend → Database → Response

When a user places a trade, the system:

Receives the request

Validates user balance

Updates portfolio

Logs the transaction

Sends confirmation

9. Database Design

Users Table

id

name

email

password

balance

Stocks Table

id

symbol

price

Portfolio Table

id

user_id

stock_id

quantity

Transactions Table

id

user_id

stock_id

transaction_type

quantity

price

10. Buy and Sell Logic

Buy Logic

When a user places a buy order:

The system checks if the user has sufficient balance

If sufficient, the amount is deducted

The stock is added to the user's portfolio

The transaction is recorded

Sell Logic

When a user places a sell order:

The system checks stock availability in the portfolio

Stock quantity is reduced

Balance is increased

The transaction is logged

11. Backend Code Example (Flask – Buy Logic)

Copy code

Python

```
@app.route("/buy", methods=["POST"])
def buy_stock():
    user_id = request.form["user_id"]
    stock_id = request.form["stock_id"]
    quantity = int(request.form["quantity"])

    price = get_stock_price(stock_id)
    total_cost = price * quantity

    if get_user_balance(user_id) >= total_cost:
        update_balance(user_id, total_cost)
        add_to_portfolio(user_id, stock_id, quantity)
        log_transaction(user_id, stock_id, "BUY", quantity, price)
        return "Buy Order Successful"
    else:
        return "Insufficient Balance”
```

12. Frontend Dashboard Design
The dashboard is designed to be simple and beginner-friendly. It displays stock prices, allows users to buy or sell stocks, and provides portfolio information.

13. Frontend Dashboard HTML Code
dashboard.html
Copy code
Html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Stock Trading
Dashboard</title>
    <link
href="https://cdn.jsdelivr.net/npm/boot
strap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
</head>
<body class="bg-light">

<nav class="navbar navbar-dark
bg-dark px-4">
    <span class="navbar-brand">Stock
Trading Dashboard</span>
    <a href="/logout" class="btn
btn-danger btn-sm">Logout</a>
</nav>

<div class="container mt-4">
    <h3>Available Stocks</h3>

    <table class="table table-bordered
bg-white">
        <thead class="table-dark">
            <tr>
                <th>Stock</th>
                <th>Price (₹)</th>
                <th>Quantity</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>TATA</td>
                <td>850</td>
                <td><input type="number"
class="form-control" value="1"></td>
                <td>
                    <button class="btn
btn-success btn-sm">Buy</button>
                    <button class="btn
btn-danger btn-sm">Sell</button>
                </td>
            </tr>
        </tbody>
    </table>
</div>

</body>
```

14. Portfolio Page Frontend Code

portfolio.html

Copy code

Html

```
<!DOCTYPE html>
<html>
<head>
    <title>My Portfolio</title>
    <link
href="https://cdn.jsdelivr.net/npm/boot
strap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
</head>
<body>

<div class="container mt-4">
    <h2>My Portfolio</h2>

    <table class="table table-striped">
      <thead class="table-dark">
        <tr>
          <th>Stock</th>
          <th>Quantity</th>
          <th>Current Price</th>
          <th>Total Value</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>TATA</td>
          <td>10</td>
          <td>850</td>
          <td>8500</td>
        </tr>
      </tbody>
    </table>
</div>

</body>
```

15. Frontend Styling (CSS)
Copy code
Css
```
body {
    font-family: Arial, sans-serif;
}

table input {
    width: 80px;
}
```

16. Advantages of the System
Risk-free learning environment
Simple and intuitive interface
Easy backend integration
Scalable cloud-ready design

17. Conclusion
The Real-Time Stock Trading Dashboard provides a safe and educational platform for learning stock market trading. By combining frontend development, backend logic, and cloud concepts, this project offers both technical and practical knowledge. The system can be extended to real-time trading