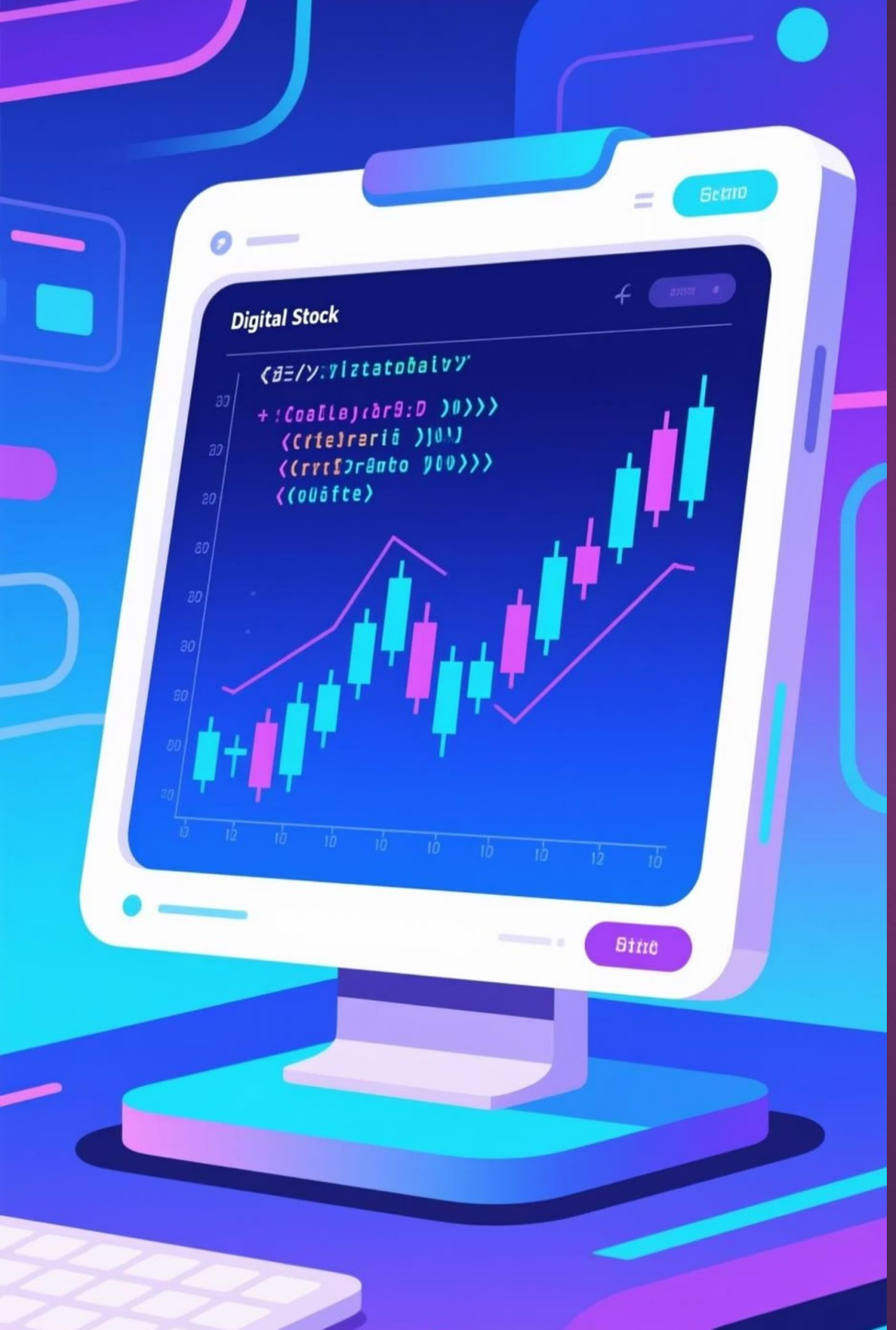


Building a Stock Portfolio Web App: Frontend & Backend with JavaScript and MySQL

A technical overview of the full-stack architecture, detailing the role of HTML, CSS, JavaScript, and MySQL in creating a secure and interactive financial management application.

Project Overview: What We’re Building

Full-Stack Website

A complete stock portfolio management application built from the ground up.

Core Functionality

Includes user registration, login, and key transaction features like buy, sell, invest, and withdraw.

Data Visualization

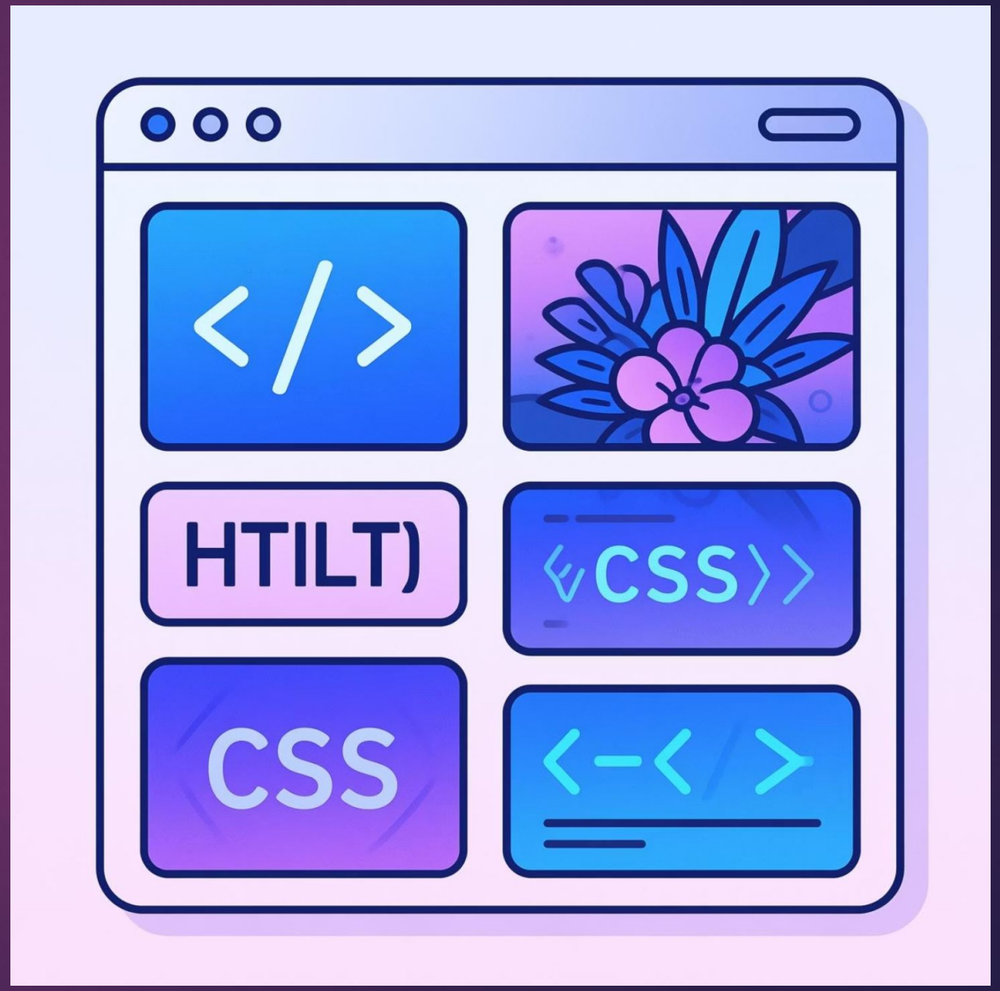
Real-time visualization of portfolio balance and transaction history using dynamic graphs.

- Technology Stack: HTML, CSS, JavaScript (Frontend & Backend), and MySQL for data persistence.

Frontend Structure: HTML & CSS for UI

- **HTML:** Provides the semantic foundation for all pages (registration, login, dashboard, transactions).
- **CSS:** Ensures responsive and clean design using Flexbox and Grid layouts, crucial for a professional user experience.
- **JavaScript:** Handles DOM manipulation, client-side form validation, and dynamic content rendering, bringing the interface to life.

The frontend is designed to be intuitive, ensuring users can navigate their portfolio effortlessly.



User Authentication Flow: Securing Access

Frontend Register

Backend Hashing

Login Session

Registration & Validation

Collect and validate user inputs (username, email, password) directly on the client side using JavaScript.

Password Security

Crucially, the backend hashes the password before storage in the MySQL database.

Login & State Management

Upon successful login, session management (e.g., using JWTs or cookies) is initiated to maintain the user's authenticated state.

Backend Integration: JavaScript & MySQL Core

The backend leverages Node.js with Express.js to create a robust and scalable server environment.

Server Framework

Express.js handles API routing and middleware functions efficiently.

Database Storage

MySQL persists all critical data: user profiles, transaction history, and stock holdings.

RESTful Endpoints

Clearly defined routes like `/register`, `/buy`, and `/portfolio` manage application logic.

Example: A `POST /buy` route atomically updates the user's cash balance and logs the transaction in the MySQL tables.

Managing Transactions & Portfolio Data

Every user action initiates a validated transaction cycle on the backend, ensuring data integrity.

1

Frontend Request

User submits buy/sell requests with stock symbol and quantity.

2

Balance Validation

Backend checks current funds or holdings for sufficient balance.

3

Database Update

MySQL tables (transactions and portfolio holdings) are updated in real-time.

4

Cash Management

Separate functions track investment and withdrawal actions, updating the primary cash balance.

5

History Tracking

All movements are logged with timestamps for a complete audit trail.

Visualizing Portfolio Performance

Data retrieved from the MySQL database is transformed into actionable visualizations using powerful JavaScript libraries.

- **Charting Libraries:** Tools like Chart.js or D3.js are integrated into the frontend to render complex financial data.
- **Graph Types:** We display transaction history, stock value trends, and total portfolio balance over time.
- **Dynamic Updates:** Charts are designed to refresh immediately after any transaction, providing a seamless, real-time experience.

A line chart can show portfolio value growth, a crucial metric for investors.

Code Architecture Highlights: The MVC Pattern

Employing the Model-View-Controller (MVC) pattern ensures organized, scalable, and maintainable code structure.

Model (MySQL)

Manages data storage: Database schema for users, stocks, and transactions.

Controller (Express.js)

Handles all business logic, processes requests, and interacts with both the Model and the View.

View (HTML/CSS/JS)

The user interface layer; dynamic rendering based on data from the Controller.

This separation of concerns significantly improves development speed and long-term product health.

Security & User Experience Considerations

Robust Security

Implement password hashing (e.g., bcrypt) to protect user credentials.

Data Protection

Mandatory input sanitization to prevent common vulnerabilities like SQL injection and XSS attacks.

Responsive Design

Ensure optimal usability across all device types, from desktop monitors to mobile phones.

Smooth User Experience (UX)

Utilize asynchronous JavaScript (AJAX/fetch) for quick, non-blocking updates, enhancing performance.

Summary & Next Steps

1

Interactive App

Successful creation of a secure, interactive stock portfolio application using full-stack JavaScript and MySQL.

2

Core Takeaways

Mastered user authentication, transactional consistency, and dynamic data visualization.

3

Future Enhancements

Integration of real-time stock market APIs, user notifications, and mobile app compatibility.

> Ready to build your own? Start with small modules and iterate!

Thank You

Done BY:

S. Dhanush
K. Harshith
Praveen Naik
Roop Singh