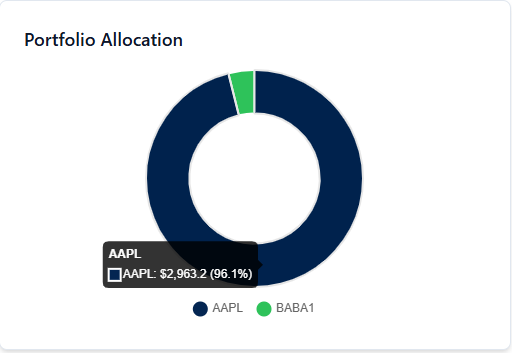
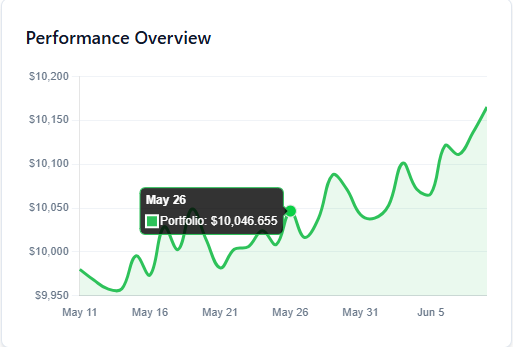
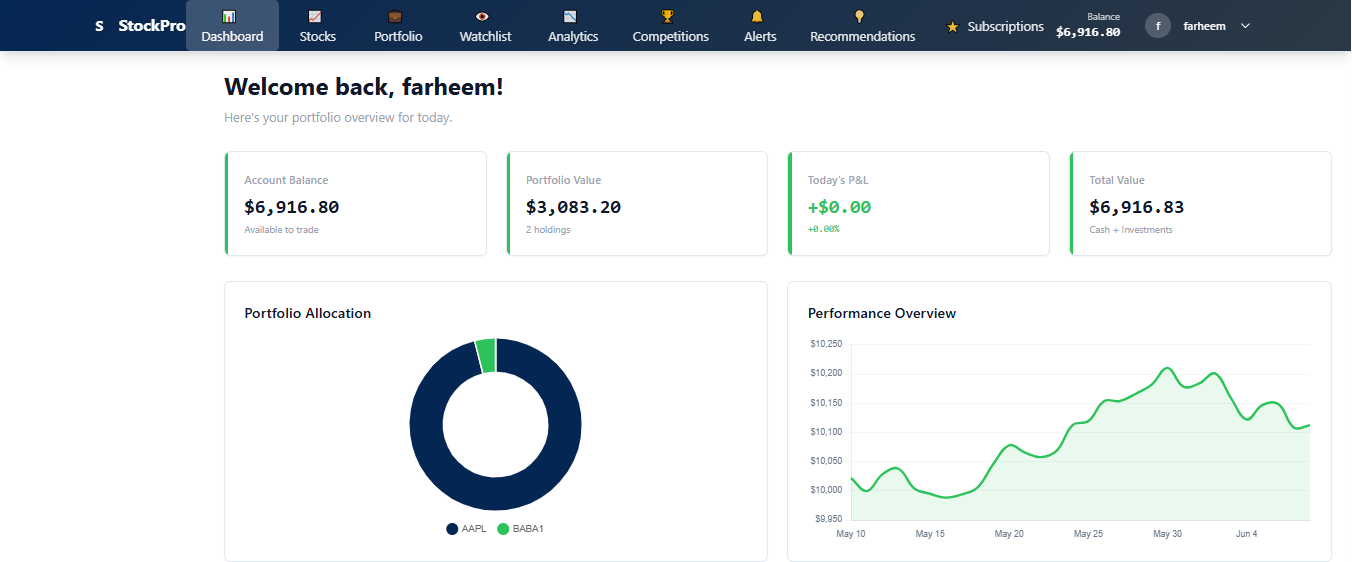
****

**PROJECT PROPOSAL**

**TITLE : STOCKPRO**





MADE BY:

ARISHA MUMTAZ(2312358)

FARHEEN IMAM(2312363)

**Introduction:**

StockPro is a comprehensive web-based stock trading platform designed to provide users with an intuitive and feature-rich environment for managing their investment portfolios. In today's fast-paced financial markets, retail investors need accessible tools to track, analyze, and trade stocks effectively. Our platform bridges this gap by offering a complete ecosystem for both novice and experienced traders.

The platform serves as a virtual trading environment where users can:

**Stock Market Simulation**: Users can trade with virtual money, making it perfect for learning and testing strategies without financial risk.

**Real-time Data Tracking**: Access to comprehensive stock information including current prices, market capitalization, volume, and performance metrics.

**Portfolio Management**: Complete portfolio tracking with profit/loss calculations, asset allocation visualization, and performance analytics.

**Social Trading Features**: Competitive trading through organized competitions and leaderboards to engage the trading community.

**Educational Resources**: Built-in analytics and recommendations to help users make informed investment decisions.

By combining modern web technologies with robust financial data management, StockPro creates an environment where users can develop their trading skills, compete with peers, and gain valuable market experience in a risk-free environment.

## **Project Overview**

A comprehensive stock trading and investment platform that allows users to trade stocks, manage portfolios, participate in trading competitions, receive personalized recommendations, and access premium features through subscriptions.

## Database Tables (12 Tables)

### 1. users1

* user\_id (PK, INT, AUTO\_INCREMENT)
* username (VARCHAR(50), UNIQUE, NOT NULL)
* email (VARCHAR(100), UNIQUE, NOT NULL)
* password\_hash (VARCHAR(255), NOT NULL)
* first\_name (VARCHAR(50))
* last\_name (VARCHAR(50))
* phone (VARCHAR(20))
* date\_of\_birth (DATE)
* account\_balance (DECIMAL(15,2), DEFAULT 10000.00)
* risk\_tolerance (ENUM('Conservative', 'Moderate', 'Aggressive'))
* subscription\_tier (ENUM('Free', 'Premium', 'Pro'), DEFAULT 'Free')
* created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)
* last\_login (TIMESTAMP)
* is\_active (BOOLEAN, DEFAULT TRUE)

### 2. stocks

* stock\_id (PK, INT, AUTO\_INCREMENT)
* symbol (VARCHAR(10), UNIQUE, NOT NULL)
* company\_name (VARCHAR(100), NOT NULL)
* sector\_id (FK, INT)
* current\_price (DECIMAL(10,2), NOT NULL)
* market\_cap (BIGINT)
* volume (BIGINT)
* day\_change (DECIMAL(8,2))
* day\_change\_percent (DECIMAL(5,2))
* pe\_ratio (DECIMAL(8,2))
* dividend\_yield (DECIMAL(5,2))
* year\_high (DECIMAL(10,2))
* year\_low (DECIMAL(10,2))
* last\_updated (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP)

### 3. sectors

* sector\_id (PK, INT, AUTO\_INCREMENT)
* sector\_name (VARCHAR(50), UNIQUE, NOT NULL)
* sector\_description (TEXT)
* performance\_ytd (DECIMAL(5,2))
* created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

### 4. portfolio

* portfolio\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* stock\_id (FK, INT, NOT NULL)
* quantity\_owned (INT, NOT NULL)
* average\_buy\_price (DECIMAL(10,2), NOT NULL)
* total\_invested (DECIMAL(15,2), NOT NULL)
* current\_value (DECIMAL(15,2))
* unrealized\_gain\_loss (DECIMAL(15,2))
* first\_purchase\_date (TIMESTAMP)
* last\_updated (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP)

### 5. transactions

* transaction\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* stock\_id (FK, INT, NOT NULL)
* transaction\_type (ENUM('BUY', 'SELL'), NOT NULL)
* quantity (INT, NOT NULL)
* transaction\_price (DECIMAL(10,2), NOT NULL)
* total\_amount (DECIMAL(15,2), NOT NULL)
* commission\_fee (DECIMAL(8,2), DEFAULT 0.00)
* transaction\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)
* order\_id (FK, INT)
* realized\_gain\_loss (DECIMAL(15,2))

### 6. orders

* order\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* stock\_id (FK, INT, NOT NULL)
* order\_type (ENUM('MARKET', 'LIMIT', 'STOP\_LOSS'), NOT NULL)
* action (ENUM('BUY', 'SELL'), NOT NULL)
* quantity (INT, NOT NULL)
* limit\_price (DECIMAL(10,2))
* stop\_price (DECIMAL(10,2))
* status (ENUM('PENDING', 'EXECUTED', 'CANCELLED', 'EXPIRED'), DEFAULT 'PENDING')
* order\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)
* execution\_date (TIMESTAMP)
* expires\_at (TIMESTAMP)

### 7. watchlists

* watchlist\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* stock\_id (FK, INT, NOT NULL)
* alert\_price\_high (DECIMAL(10,2))
* alert\_price\_low (DECIMAL(10,2))
* notes (TEXT)
* added\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

### 8. stock\_alerts

* alert\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* stock\_id (FK, INT, NOT NULL)
* alert\_type (ENUM('PRICE\_HIGH', 'PRICE\_LOW', 'VOLUME\_SPIKE', 'NEWS'), NOT NULL)
* target\_value (DECIMAL(10,2))
* is\_triggered (BOOLEAN, DEFAULT FALSE)
* message (TEXT)
* created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)
* triggered\_at (TIMESTAMP)

### 9. trading\_competitions

* competition\_id (PK, INT, AUTO\_INCREMENT)
* competition\_name (VARCHAR(100), NOT NULL)
* description (TEXT)
* start\_date (TIMESTAMP, NOT NULL)
* end\_date (TIMESTAMP, NOT NULL)
* initial\_balance (DECIMAL(15,2), DEFAULT 100000.00)
* entry\_fee (DECIMAL(8,2), DEFAULT 0.00)
* prize\_pool (DECIMAL(15,2))
* max\_participants (INT)
* status (ENUM('UPCOMING', 'ACTIVE', 'COMPLETED'), DEFAULT 'UPCOMING')
* created\_by (FK, INT)
* created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

### 10. competition\_participants

* participant\_id (PK, INT, AUTO\_INCREMENT)
* competition\_id (FK, INT, NOT NULL)
* user\_id (FK, INT, NOT NULL)
* virtual\_balance (DECIMAL(15,2))
* current\_portfolio\_value (DECIMAL(15,2))
* total\_return (DECIMAL(15,2))
* return\_percentage (DECIMAL(8,2))
* rank (INT)
* joined\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

### 11. recommendations

* recommendation\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* stock\_id (FK, INT, NOT NULL)
* recommendation\_type (ENUM('BUY', 'SELL', 'HOLD'), NOT NULL)
* confidence\_score (DECIMAL(3,2))
* reason (TEXT)
* target\_price (DECIMAL(10,2))
* algorithm\_used (VARCHAR(50))
* created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)
* expires\_at (TIMESTAMP)
* is\_viewed (BOOLEAN, DEFAULT FALSE)

### 12. user\_subscriptions

* subscription\_id (PK, INT, AUTO\_INCREMENT)
* user\_id (FK, INT, NOT NULL)
* subscription\_type (ENUM('Premium', 'Pro'), NOT NULL)
* start\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)
* end\_date (TIMESTAMP, NOT NULL)
* payment\_amount (DECIMAL(8,2), NOT NULL)
* payment\_method (VARCHAR(50))
* auto\_renewal (BOOLEAN, DEFAULT TRUE)
* status (ENUM('ACTIVE', 'EXPIRED', 'CANCELLED'), DEFAULT 'ACTIVE')
* created\_at (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

## **Table Relationships**

### Primary Relationships:

1. **users**1→ **portfolio** (1:M) - One user can have multiple portfolio entries
2. **stocks** → **portfolio** (1:M) - One stock can be in multiple portfolios
3. **users1** → **transactions** (1:M) - One user can have multiple transactions
4. **stocks** → **transactions** (1:M) - One stock can have multiple transactions
5. **sectors** → **stocks** (1:M) - One sector contains multiple stocks
6. **users**1→ **orders** (1:M) - One user can place multiple orders
7. **stocks** → **orders** (1:M) - One stock can have multiple orders
8. **orders** → **transactions** (1:1) - One executed order creates one transaction
9. **users1** → **watchlists** (1:M) - One user can have multiple watchlist items
10. **stocks** → **watchlists** (1:M) - One stock can be in multiple watchlists
11. **users1** → **stock\_alerts** (1:M) - One user can have multiple alerts
12. **stocks** → **stock\_alerts** (1:M) - One stock can trigger multiple alerts
13. **trading\_competitions** → **competition\_participants** (1:M) - One competition has multiple participants
14. **users1** → **competition\_participants** (1:M) - One user can participate in multiple competitions
15. **users**1→ **recommendations** (1:M) - One user can receive multiple recommendations
16. **stocks** → **recommendations** (1:M) - One stock can be recommended to multiple users
17. **users1** → **user\_subscriptions** (1:M) - One user can have multiple subscription records
18. **users1** → **trading\_competitions** (1:M) - One user can create multiple competitions

## **Key Features Enabled by This Design**

### Core Trading Features:

* Real-time portfolio tracking with profit/loss calculations
* Advanced order types (Market, Limit, Stop-Loss)
* Transaction history with commission tracking
* Watchlists with price alerts

### Advanced Features:

* **Sector Analysis**: Group stocks by sectors for better analysis
* **Trading Competitions**: Virtual trading contests with leaderboards
* **AI Recommendations**: Personalized stock recommendations based on user preferences
* **Premium Subscriptions**: Tiered access to advanced features
* **Alert System**: Customizable price and volume alerts
* **Risk Management**: User risk tolerance tracking for personalized suggestions

### Analytics Capabilities:

* Portfolio performance tracking across different time periods
* Sector-wise investment distribution
* Competition rankings and performance metrics
* User engagement analytics through subscription and activity tracking

## Sample Queries Enabled:

1. **Portfolio Performance**: Calculate total portfolio value and gains/losses for each user
2. **Sector Analysis**: Show investment distribution across different sectors
3. **Trading Activity**: Track most active traders and popular stocks
4. **Competition Leaderboards**: Rank participants by returns in trading competitions
5. **Personalized Recommendations**: Generate stock suggestions based on user's portfolio and risk tolerance
6. **Alert Management**: Track and trigger price alerts for users

**Subscription Analytics**: Monitor premium feature usage and subscription trend.

**Objectives:**

The main objectives of StockPro are:

**a.** To create an intuitive and responsive web interface for stock trading and portfolio management.

**b.** To implement a secure user authentication and authorization system with multiple subscription tiers.

**c.** To develop comprehensive dashboards tailored for different user experience levels and subscription types.

**d.** To provide complete CRUD operations for managing stocks, portfolios, orders, and user accounts.

**e.** To offer advanced search and filtering functionality for stocks and market sectors.

**f.** To implement real-time data visualization and analytics for informed decision-making.

**g.** To create a competitive trading environment through organized competitions and ranking systems.

**Technologies/Frameworks:**

StockPro is developed using a modern full-stack web development approach:

**Frontend Technologies:**

* **HTML:**  Modern semantic markup for structured content presentation
* **CSS**: Advanced styling with responsive design principles and animations
* **JavaScript** : Dynamic client-side functionality and API interactions
* **Chart.js**: Interactive data visualization for portfolio analytics and market trends

**Backend Technologies:**

* **Node.js**: JavaScript runtime environment for server-side development
* **Express.js**: Fast and minimalist web application framework for Node.js
* **RESTful API Architecture**: Clean and standardized API endpoints for data operations

**Database:**

* **PostgreSQL**: Advanced open-source relational database management system
* **Database Design**: Normalized schema with 12+ interconnected tables for optimal data integrity

**Development Tools:**

* **Git**: Version control system for collaborative development
* **GitHub**: Repository hosting and project management
* **VS Code**: Integrated development environment

## Database Schema & Features

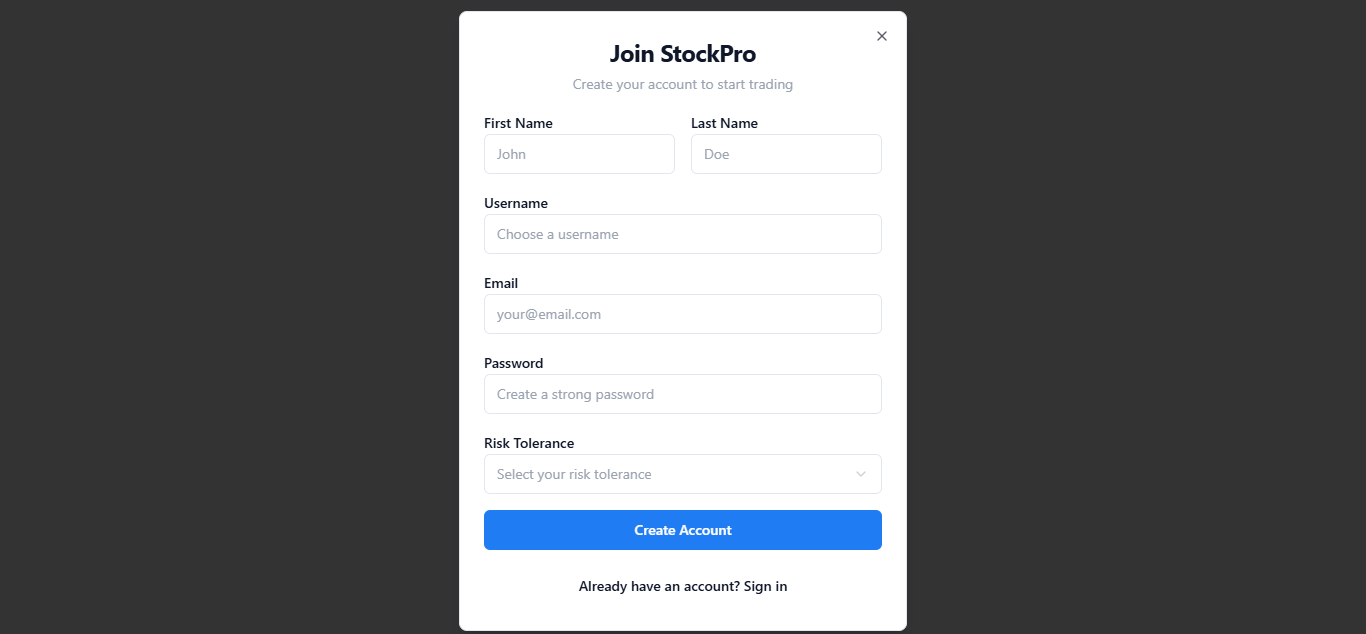
### ****Core Tables (12+ Tables):****

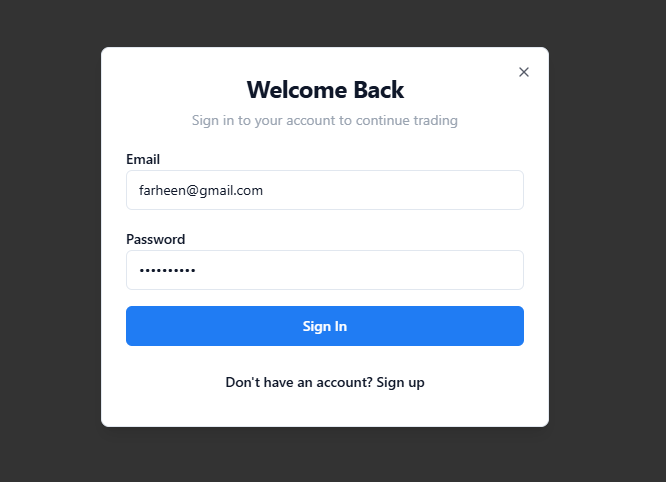
1. **users1**: Complete user management with authentication, profile data, and subscription tiers
2. **sectors**: Stock market sector categorization and performance tracking
3. **stocks**: Comprehensive stock information with real-time pricing data
4. **portfolio**: User portfolio holdings with profit/loss calculations
5. **orders**: Trading order management with multiple order types
6. **transactions**: Complete transaction history and audit trail
7. **watchlists**: User-customized stock monitoring lists
8. **stock\_alerts**: Price-based notification system
9. **trading\_competitions**: Virtual trading contest management
10. **competition\_participants**: Competition enrollment and performance tracking
11. **recommendations**: AI-driven stock recommendations
12. **user\_subscriptions**: Subscription tier management and billing

### ****Key Features:****

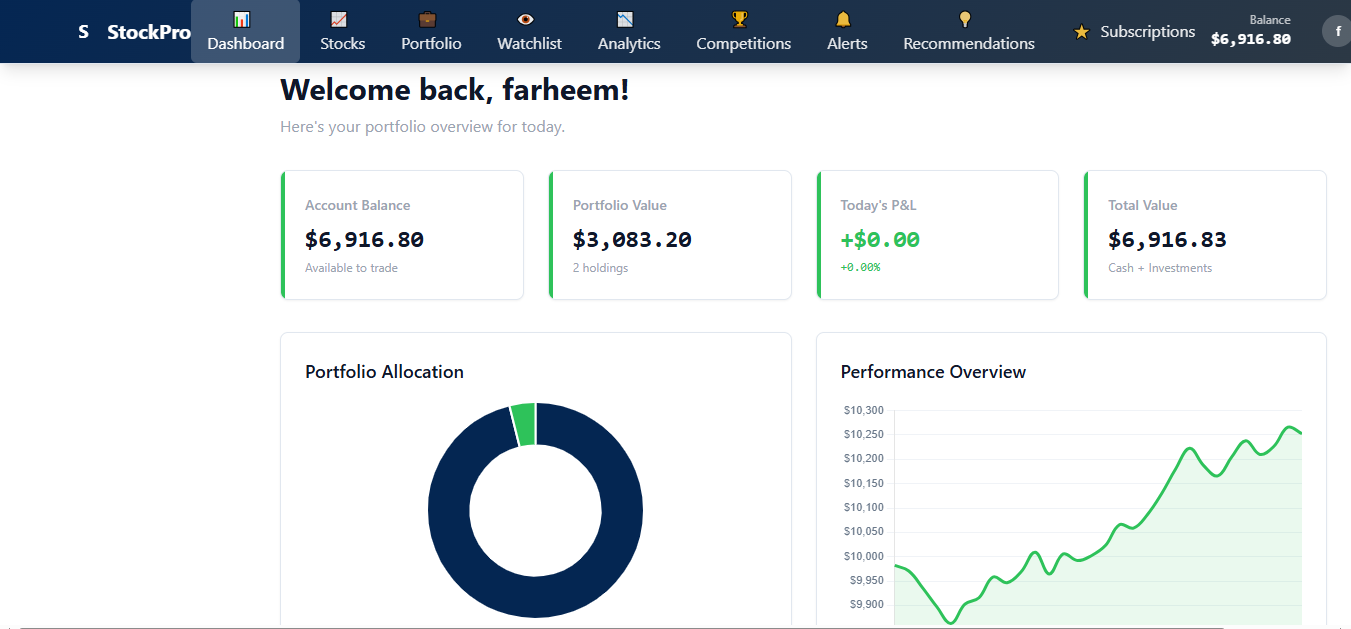
#### **User Management:**

* Secure signup/login with email verification
* Comprehensive user profiles with risk tolerance assessment
* Multi-tier subscription system (Free/Premium/Pro)
* Account balance tracking and management



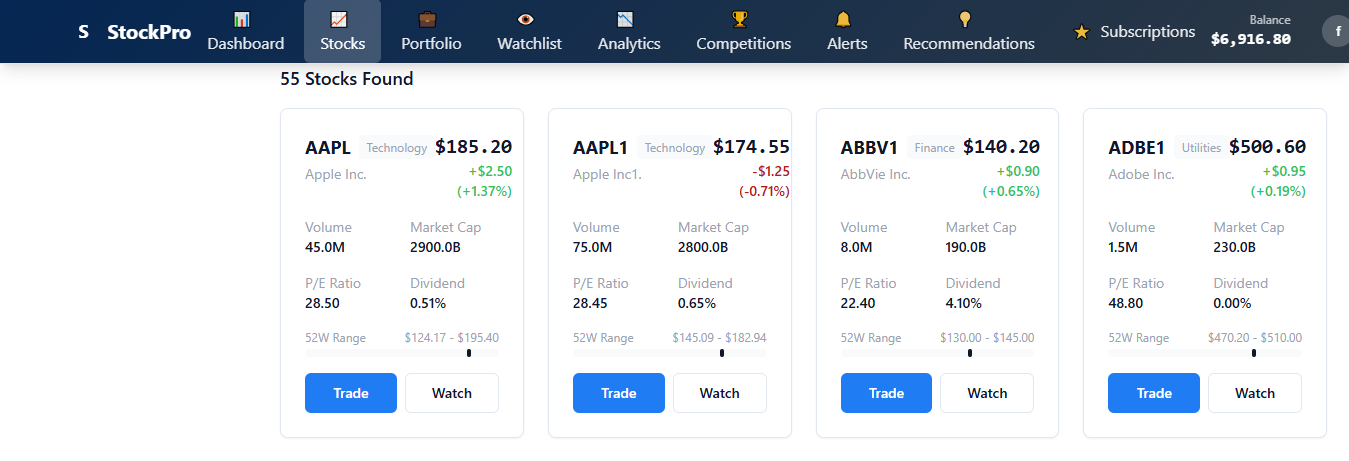


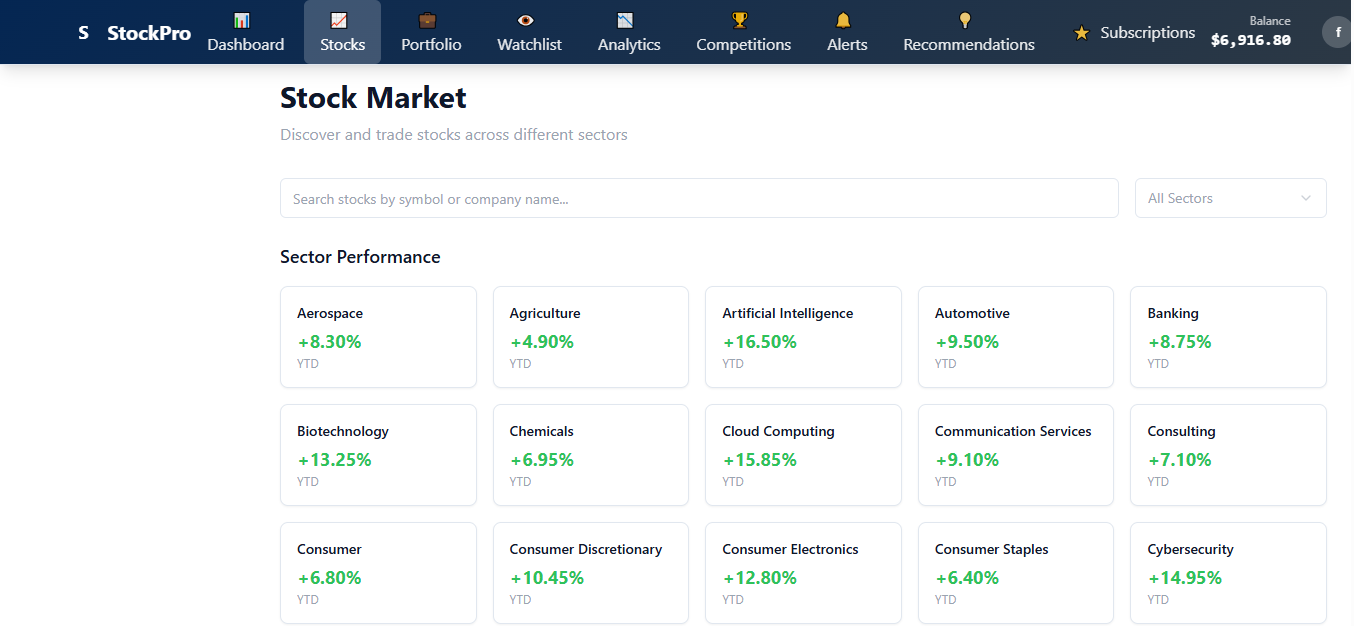
**DASHBOARD:**



#### **Stock Trading System:**

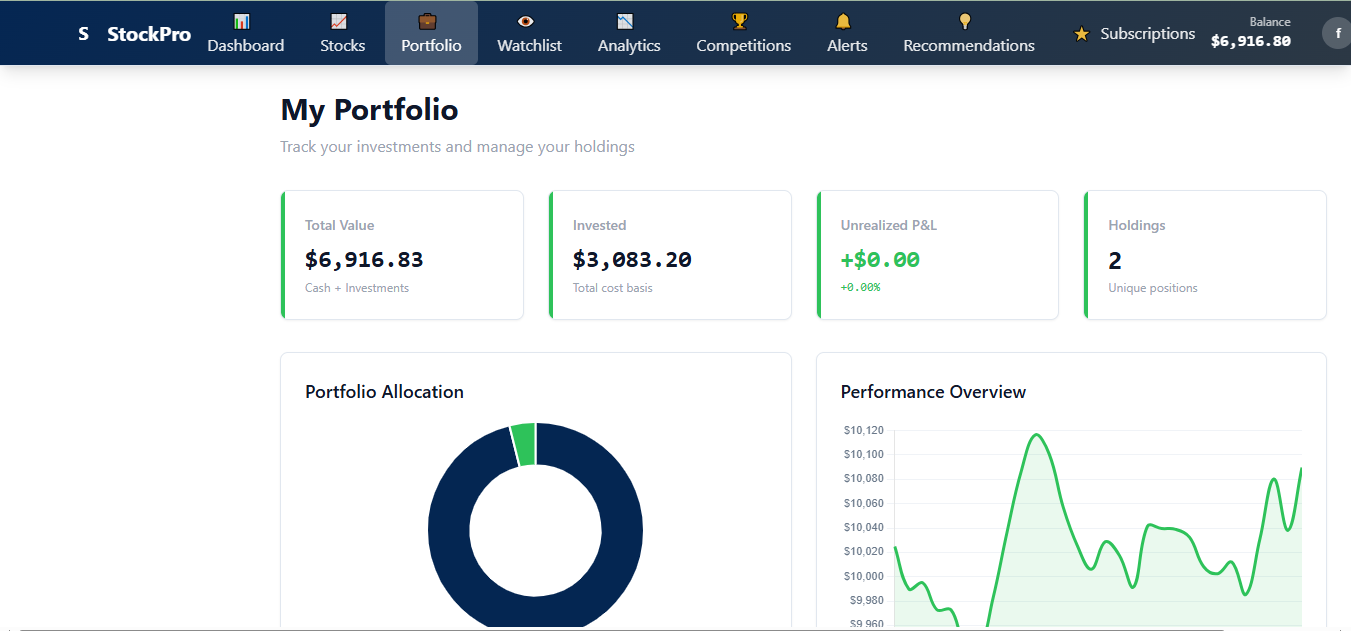
* Stock data for 100+ companies across multiple sectors
* Real-time price updates and market data
* Multiple order types: Market, Limit, and Stop-Loss orders
* Complete buy/sell transaction processing

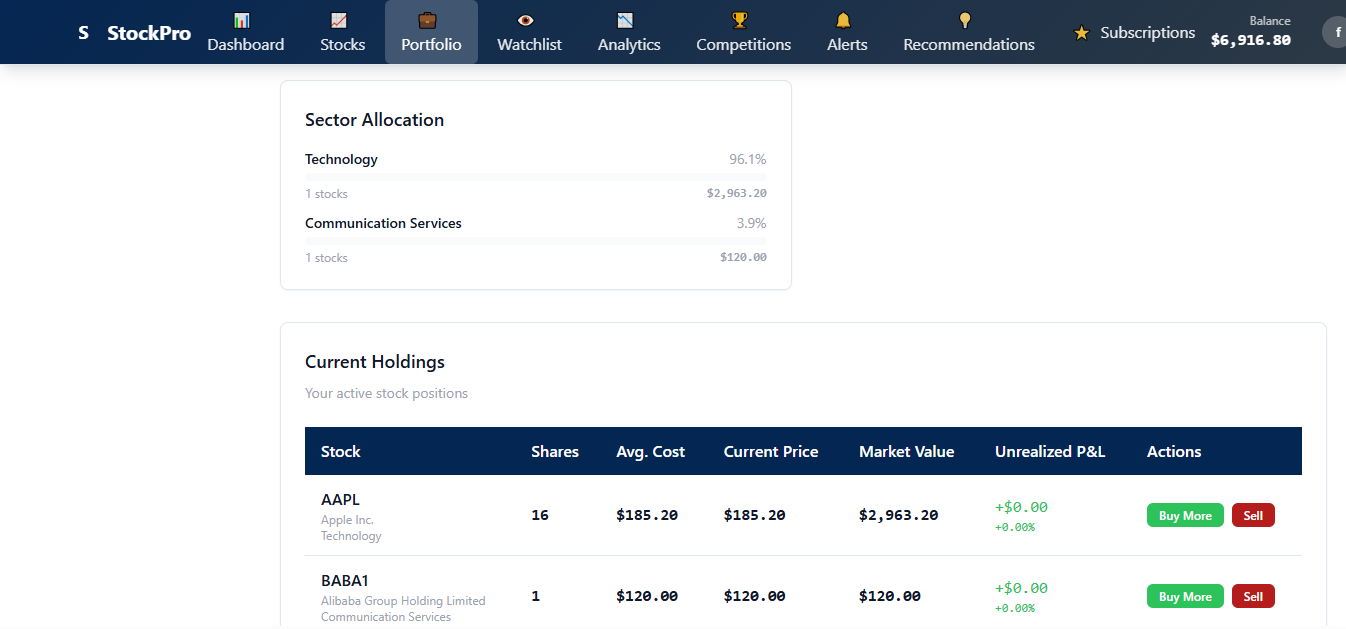




#### **Portfolio Management:**

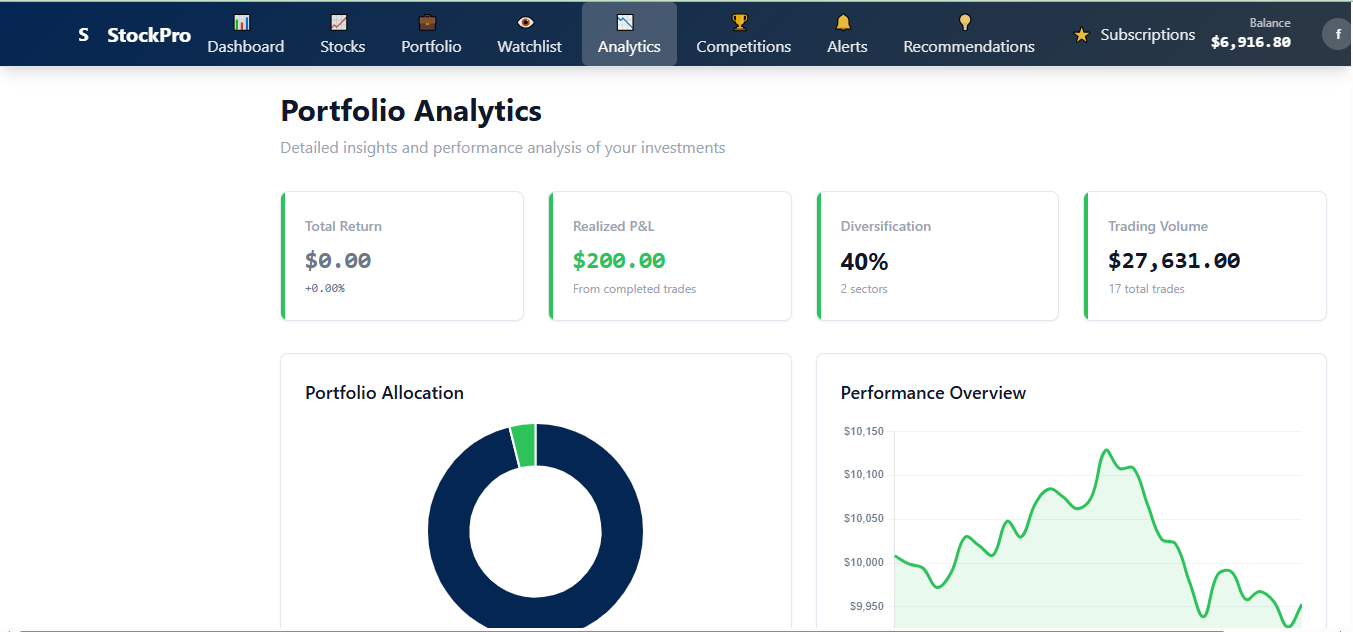
* Real-time portfolio valuation and performance tracking
* Detailed profit/loss analysis (realized vs. unrealized)
* Asset allocation visualization with interactive pie charts
* Historical performance tracking and analytics

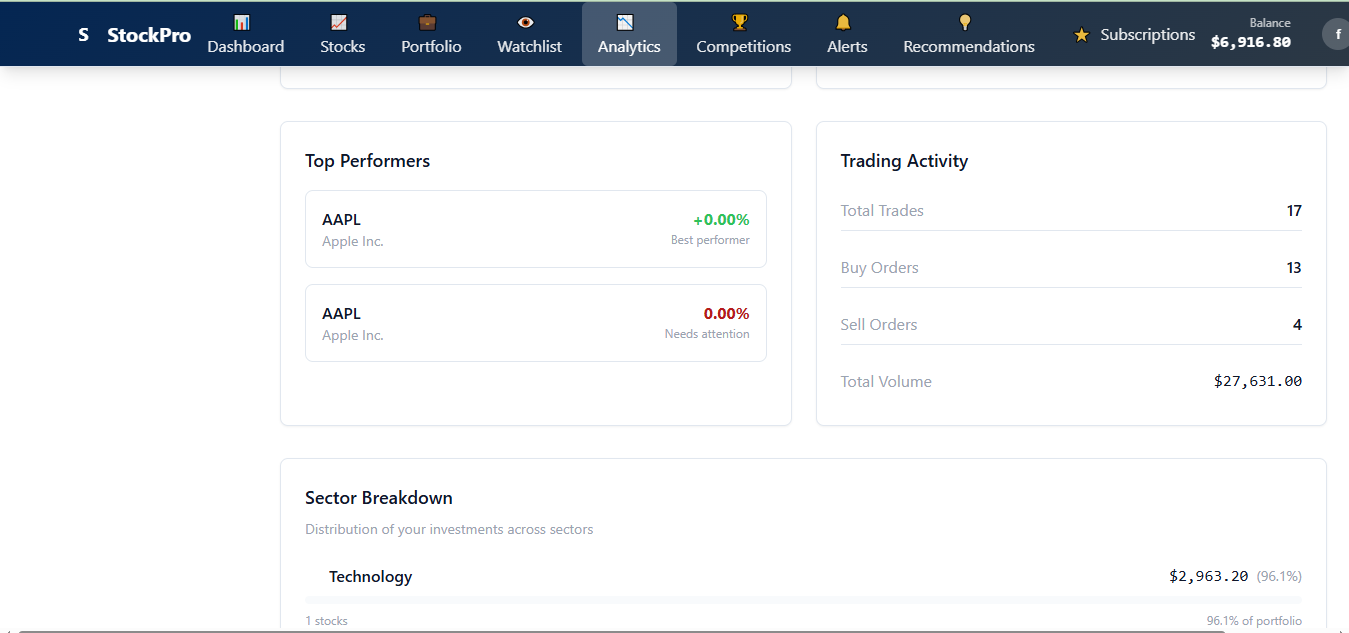


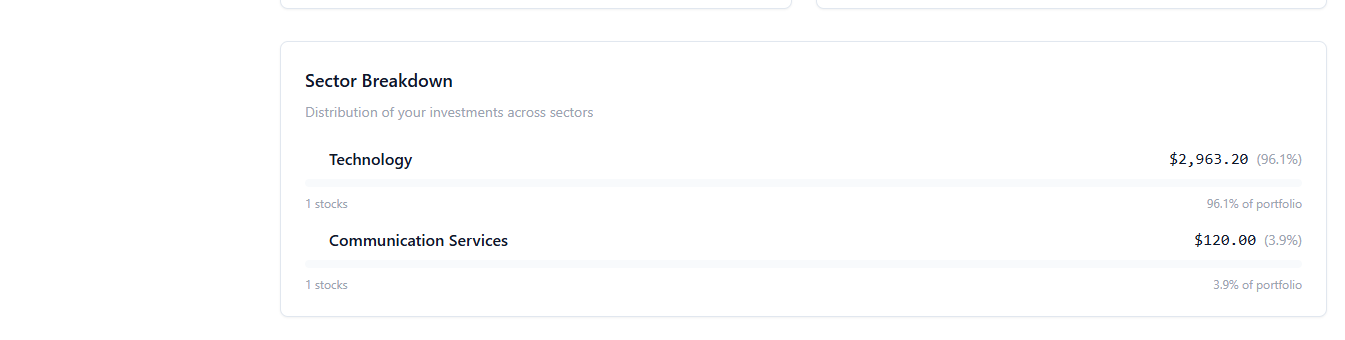


#### **Market Analysis Tools:**

* Sector-wise performance visualization
* Interactive price charts and technical indicators
* Customizable watchlists with price alerts
* Stock recommendation engine with confidence scoring



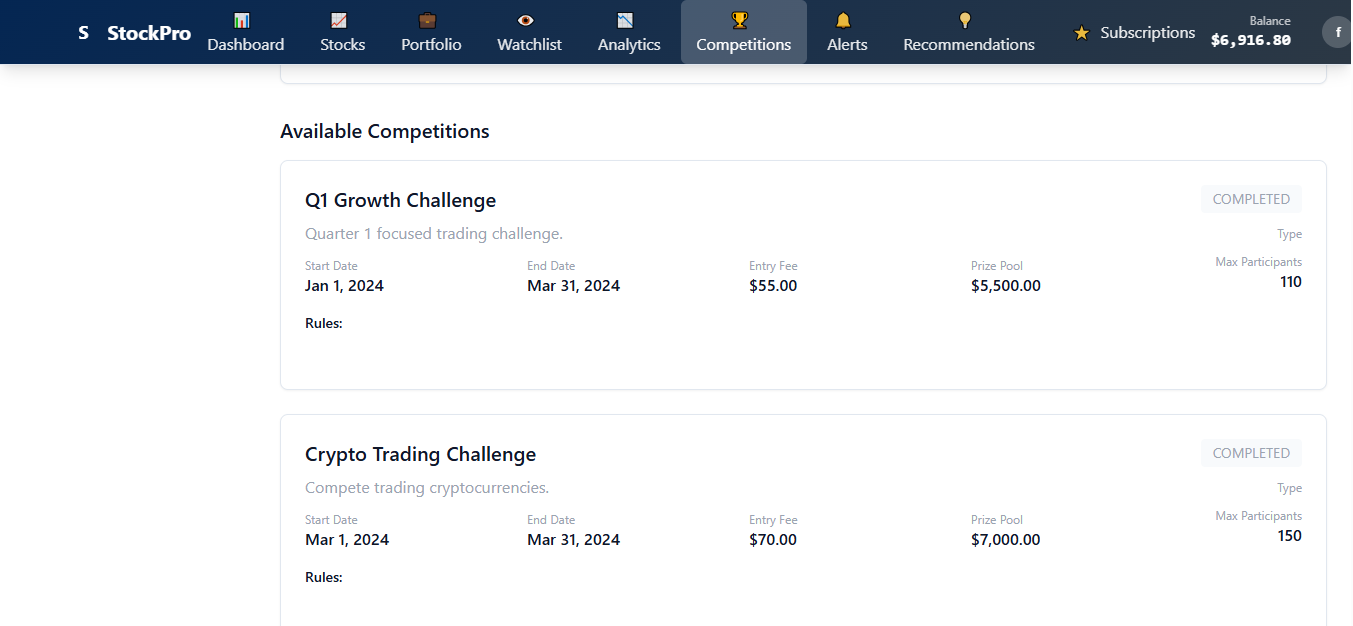


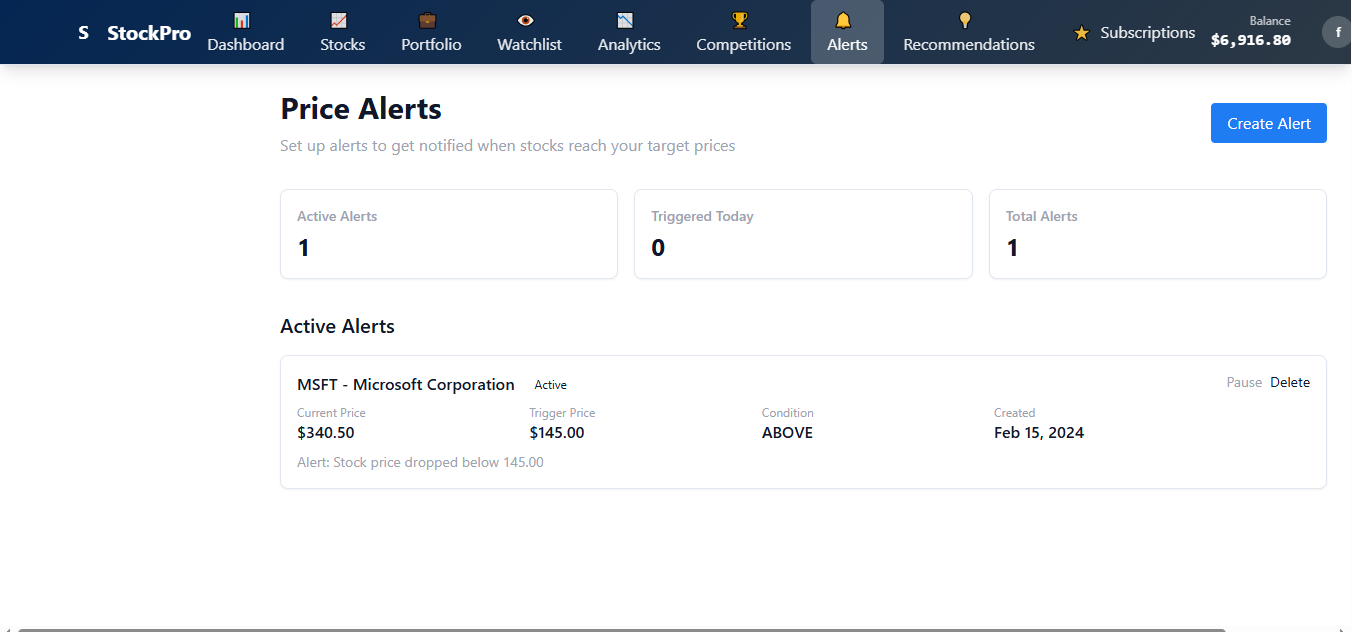


#### **Social Trading Features:**

* Virtual trading competitions with entry fees and prize pools
* Real-time leaderboards and ranking systems
* Community engagement through competitive trading
* Performance comparison and social trading insights

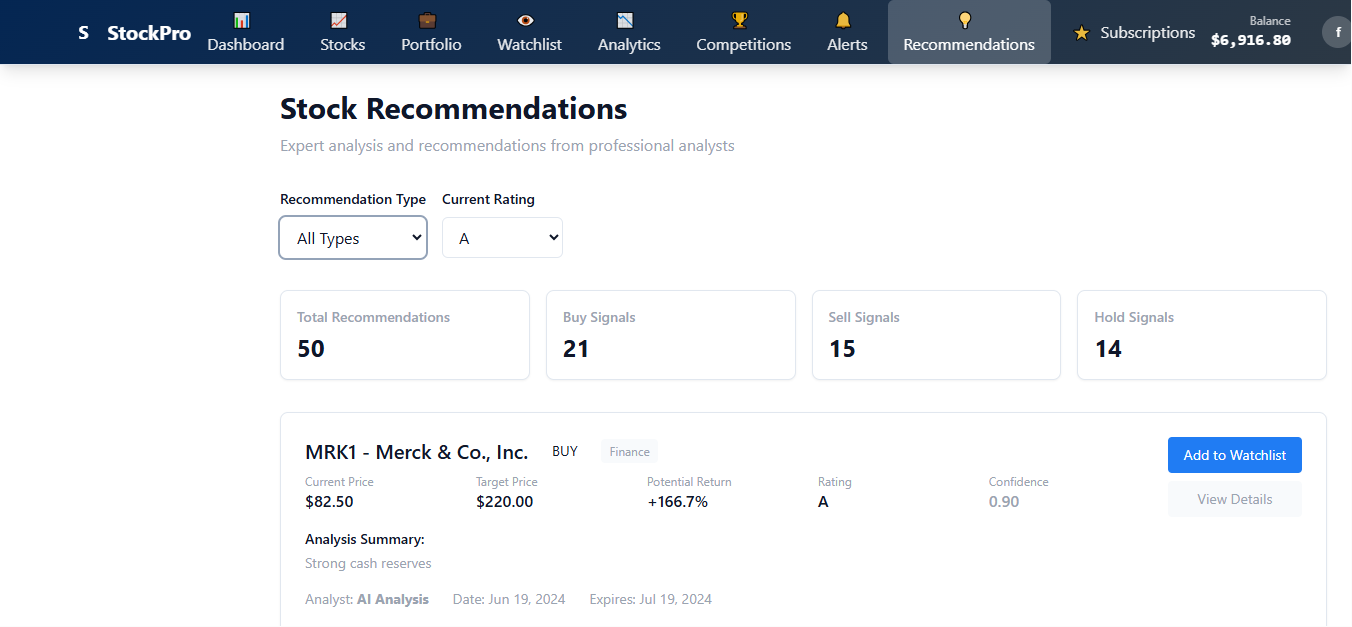


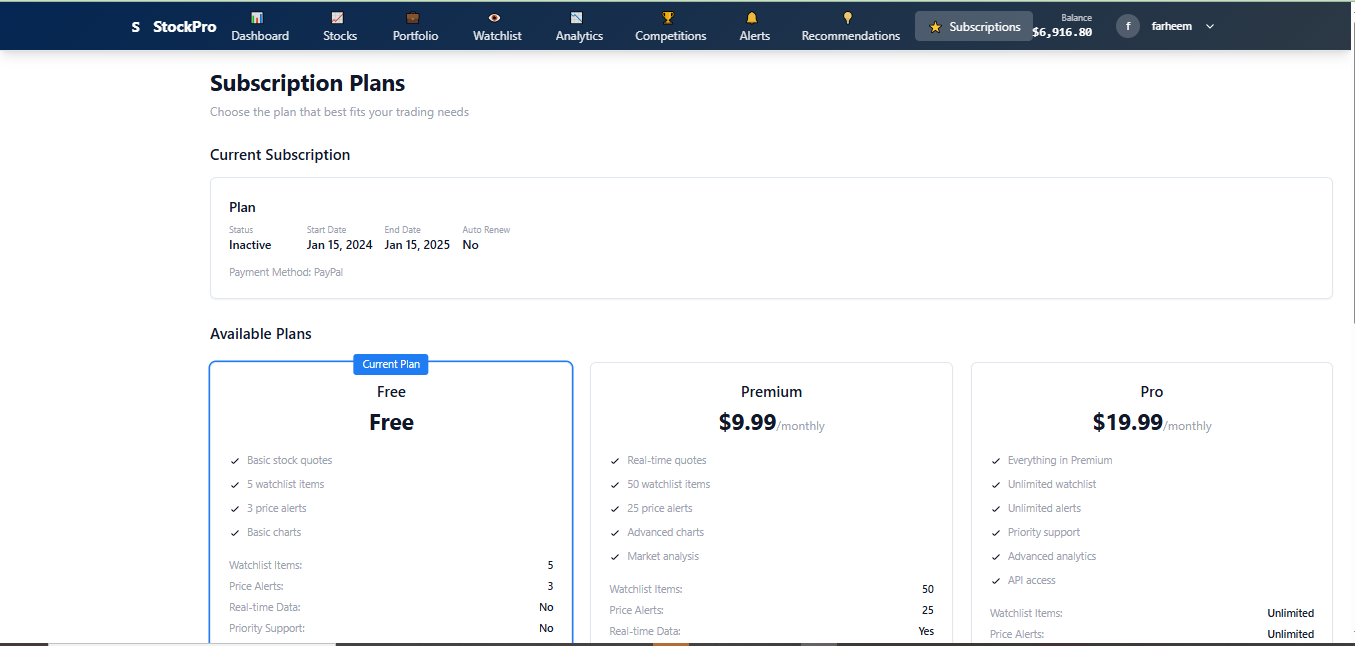


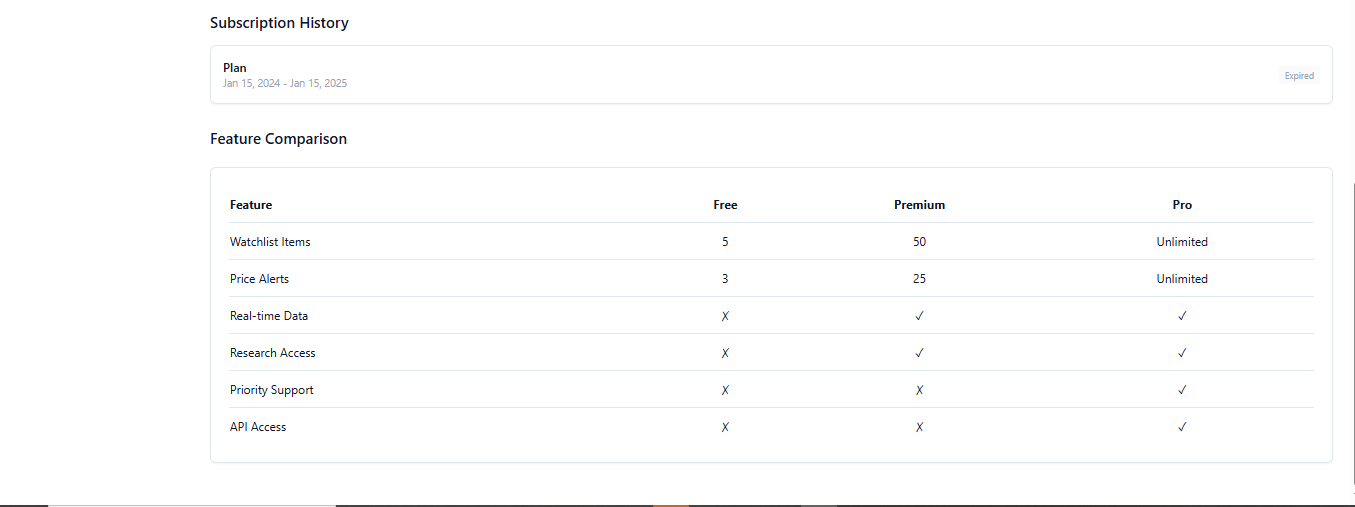


#### **Advanced Analytics:**

* Interactive charts using Chart.js for data visualization
* Portfolio performance over time analysis
* Sector allocation and diversification metrics
* Risk assessment and portfolio optimization suggestions







## Implementation Architecture

### ****Frontend Architecture:****

The client-side implementation follows modern JavaScript practices with modular design:

* **Responsive Design**: Mobile-first approach ensuring optimal user experience across all devices
* **Interactive UI**: Dynamic content updates without page refreshes
* **Data Visualization**: Real-time charts and graphs for portfolio and market analysis
* **User Experience**: Intuitive navigation and clear visual hierarchy

### ****Backend Architecture:****

The server-side implementation uses Express.js with RESTful API design:

* **API Endpoints**: Comprehensive CRUD operations for all database entities
* **Authentication**: Secure session management and user authorization
* **Data Processing**: Real-time calculations for portfolio values and market metrics
* **Error Handling**: Robust error management with user-friendly responses

### ****Database Design:****

PostgreSQL database with optimized schema design:

* **Normalization**: Properly normalized tables to eliminate data redundancy
* **Relationships**: Foreign key constraints ensuring data integrity
* **Indexing**: Strategic indexing for optimal query performance
* **Data Types**: Appropriate data types for financial calculations and precision

## API Implementation

### ****Level 1 Implementation (60% - Basic Requirements):****

**12+ Database Tables** as per financial scenario requirements  
**GET (Read) API Methods** implemented for all tables:

* /api/users - Retrieve user information
* /api/stocks - Get stock data and market information
* /api/portfolio/:userId - Fetch user portfolio holdings
* /api/transactions/:userId - Get transaction history
* /api/orders/:userId - Retrieve user orders
* /api/watchlists/:userId - Get user watchlists
* /api/competitions - Fetch trading competitions
* /api/sectors - Get sector information
* /api/recommendations/:userId - Get personalized recommendations

**Simple Reports** based on data analysis and portfolio performance

### ****Level 2 Implementation (61-75% - Enhanced Features):****

**POST (Create) API Methods** for all tables:

* User registration and profile creation
* Stock purchase and sell orders
* Watchlist management
* Competition participation
* Alert creation and management
* Transaction processing
* Portfolio updates

**Data Insertion Handling** through proper API validation and error management

### ****Level 3 Implementation (76-100% - Advanced Features):****

**Graphical Data Visualization** implementation:

* Interactive portfolio allocation pie charts
* Stock price trend line charts
* Sector performance bar charts
* Competition leaderboard visualizations
* Portfolio performance over time graphs

**Meaningful Data Insights**:

* Portfolio diversification analysis
* Risk assessment visualizations
* Market trend analysis
* Competition performance comparisons
* Profit/loss trend analysis

## Security Implementation

### ****Authentication & Authorization:****

* Secure password hashing.
* Session-based authentication with secure session management
* Role-based access control for different subscription tiers
* Input validation and sanitization to prevent attacks

## User Experience Design

### ****Interface Design Principles:****

* Clean and intuitive layout optimized for financial data presentation
* Responsive design ensuring consistent experience across desktop, tablet, and mobile devices
* Color-coded visual indicators for profit/loss and market trends
* Fast loading times with optimized database queries and caching strategies

### ****Accessibility Features:****

* Screen reader compatibility for visually impaired users
* Keyboard navigation support
* High contrast mode for better visibility
* Clear and descriptive labels for all interactive elements

## Testing & Quality Assurance

### ****API Testing:****

* Comprehensive testing of all GET and POST endpoints
* Data validation testing for all input forms
* Error handling verification for edge cases
* Performance testing for concurrent user scenarios

### ****Database Testing:****

* Data integrity testing across all table relationships
* Transaction rollback testing for failed operations
* Concurrent access testing for multi-user scenarios
* Backup and recovery testing procedures

## Deployment & Infrastructure

### ****Development Environment:****

* Local development setup with PostgreSQL database
* Git version control with feature branch workflow
* Code review process for quality assurance
* Automated testing integration

### ****Production Considerations:****

* Database optimization for production workloads
* Server-side caching for improved performance
* Load balancing strategies for high availability
* Monitoring and logging for system health tracking

## Timeline & Development Phases

**Phase 1 - Planning & Design (Week 1-2):**

* Requirements analysis and system design
* Database schema design and ERD creation
* UI/UX mockups and user flow design
* Technical architecture planning

**Phase 2 - Core Development (Week 3-8):**

* Database setup and table creation
* Backend API development and testing
* Frontend interface development
* User authentication implementation

**Phase 3 - Feature Enhancement (Week 9-10):**

* Advanced features implementation
* Data visualization integration
* Competition system development
* Performance optimization

**Phase 4 - Testing & Documentation (Week 11-12):**

* Comprehensive testing across all features
* Bug fixes and performance improvements
* Documentation completion
* Deployment preparation

## Resources Required

### ****Software Requirements:****

* Node.js runtime environment (v14 or higher)
* PostgreSQL database server
* Modern web browsers for testing and development
* Git for version control management

### ****Development Libraries:****

* Express.js for server framework
* pg (node-postgres) for database connectivity
* Chart.js for data visualization
* bcrypt for password security
* express-session for session management

**Timeline:**

Design: 1 week

Implementation: 8 weeks

Testing: 2 weeks

Documentation: 1 week

**Conclusion:**

StockPro represents a comprehensive solution for virtual stock trading and portfolio management, successfully implementing all required academic project criteria while providing practical value to users interested in learning about financial markets. The platform demonstrates proficiency in full-stack web development, database design, and modern software engineering practices.

The project successfully achieves:

* **60%+ Grade Requirements**: Complete documentation, ERD, 12+ tables, and GET API implementations
* **75%+ Grade Requirements**: Full CRUD operations with POST method implementations
* **100% Grade Requirements**: Advanced data visualization and meaningful analytical insights

Through its implementation, StockPro showcases the integration of modern web technologies to create a feature-rich financial application that serves both educational and practical purposes in the fintech domain.

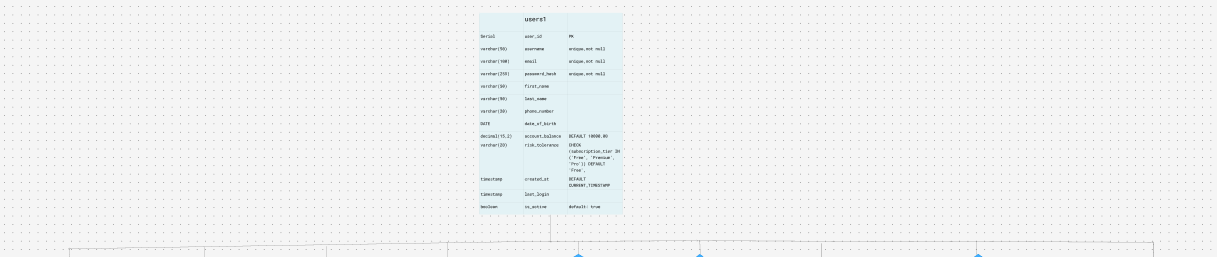
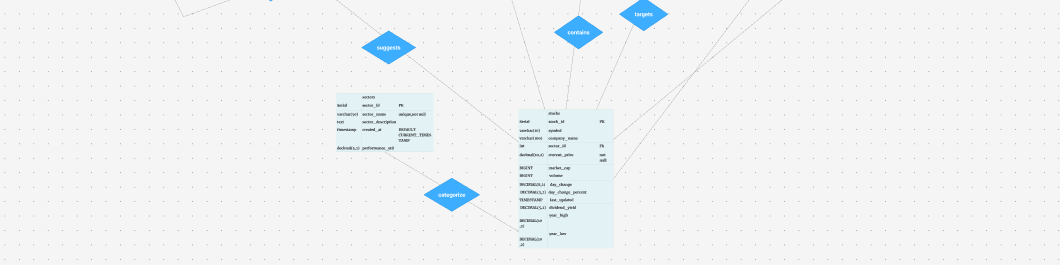
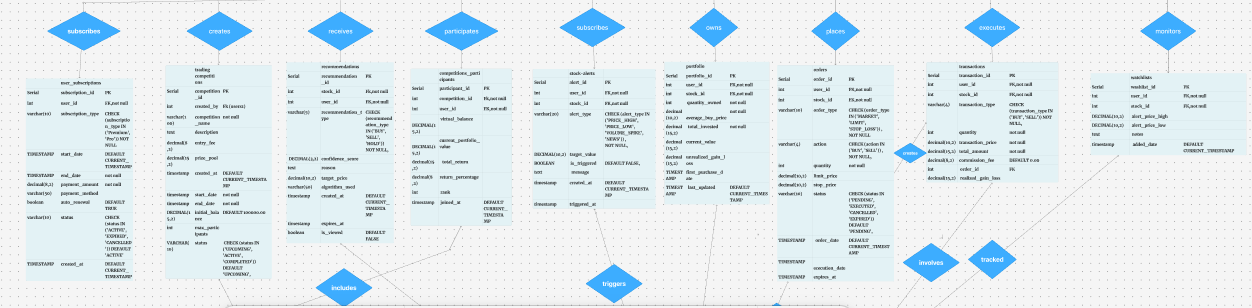
**References:**

* **Express.js Documentation**: <https://expressjs.com/>
* **PostgreSQL Documentation**: <https://www.postgresql.org/docs/>
* **Chart.js Documentation**: <https://www.chartjs.org/docs/>
* **Node.js Official Documentation**: <https://nodejs.org/docs/>

**Repository**: <https://github.com/Arisha004/STOCKS>

**THE END …………………………………………………………………………………**

***ERD:***

VIEW HERE: <https://www.figma.com/board/BO60rYzYsR1LlLMcfkGsg7/Untitled?node-id=0-1&p=f&t=FMJHwehp9j8R6l1h-0>

Note:

Relation of order-> transactions

1:1