Stonks Market

Maxim Dokukin, Ryan Fernald, Sean Hsieh, Ashkan Nikfarjam, Varun Ranjan

12/02/2024

Problem Statement

- ➔ Managing stock investments is complex for beginners.
- ➔ Real trading involves financial risk, deterring learning.
- ➔ Lack of virtual platforms for simulated trading of real market data.

- ➔ Research shows that 97% of Brazilian day traders lost money over a 300-day period, highlighting the challenges faced by inexperienced traders (Business Insider).

The Solution

Virtual Trading System:

➔ Learn trading with no real risk.

Features:

➔ Real-time S&P 500 stock data.

➔ Virtual dollar wallet for risk-free investments.

➔ Analytics to track performance and simulate strategies.

Key Benefit:

➔ Build trading skills in a real-market environment without financial loss.

Key Features and Functionality

User Management:

- ➔ Sign up and maintain balances in virtual dollars.

Stock Market Trading:

- ➔ Buy/Sell stocks using S&P 500 tickers.
- ➔ Track market prices and performance.
- ➔ Historical Prices: View price trends for better decisions.

Watchlist:

- ➔ Track favorite stocks.

Virtual Wallet:

- ➔ Deposit/withdraw funds.
- ➔ Record transactions securely.

News Section:

- ➔ Keep up-to-date with market news.

Application Architecture

Frontend:

➔ React (user interface).

Backend:

➔ Flask API for data management.

Database:

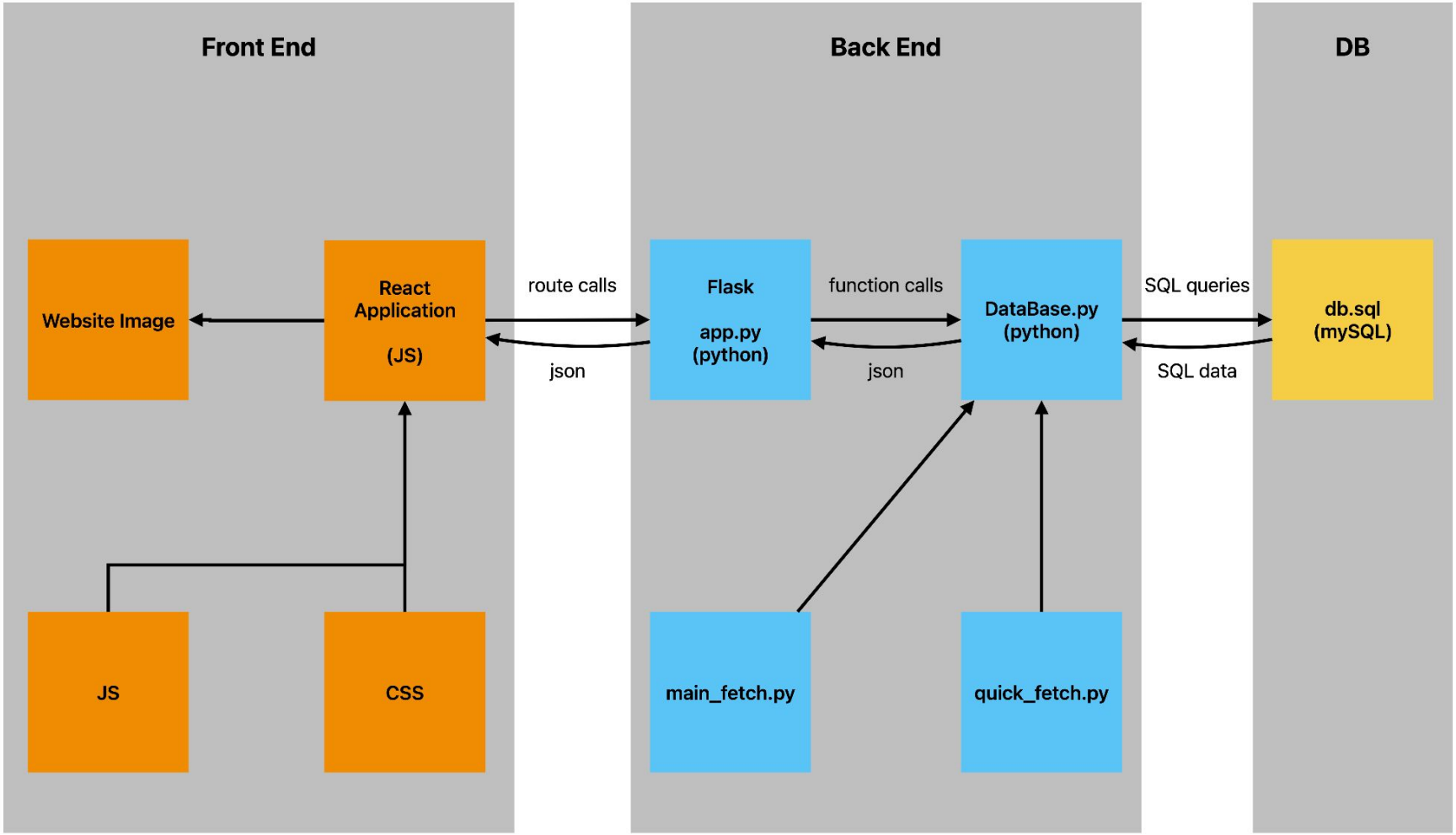
➔ MySQL with BCNF schema.

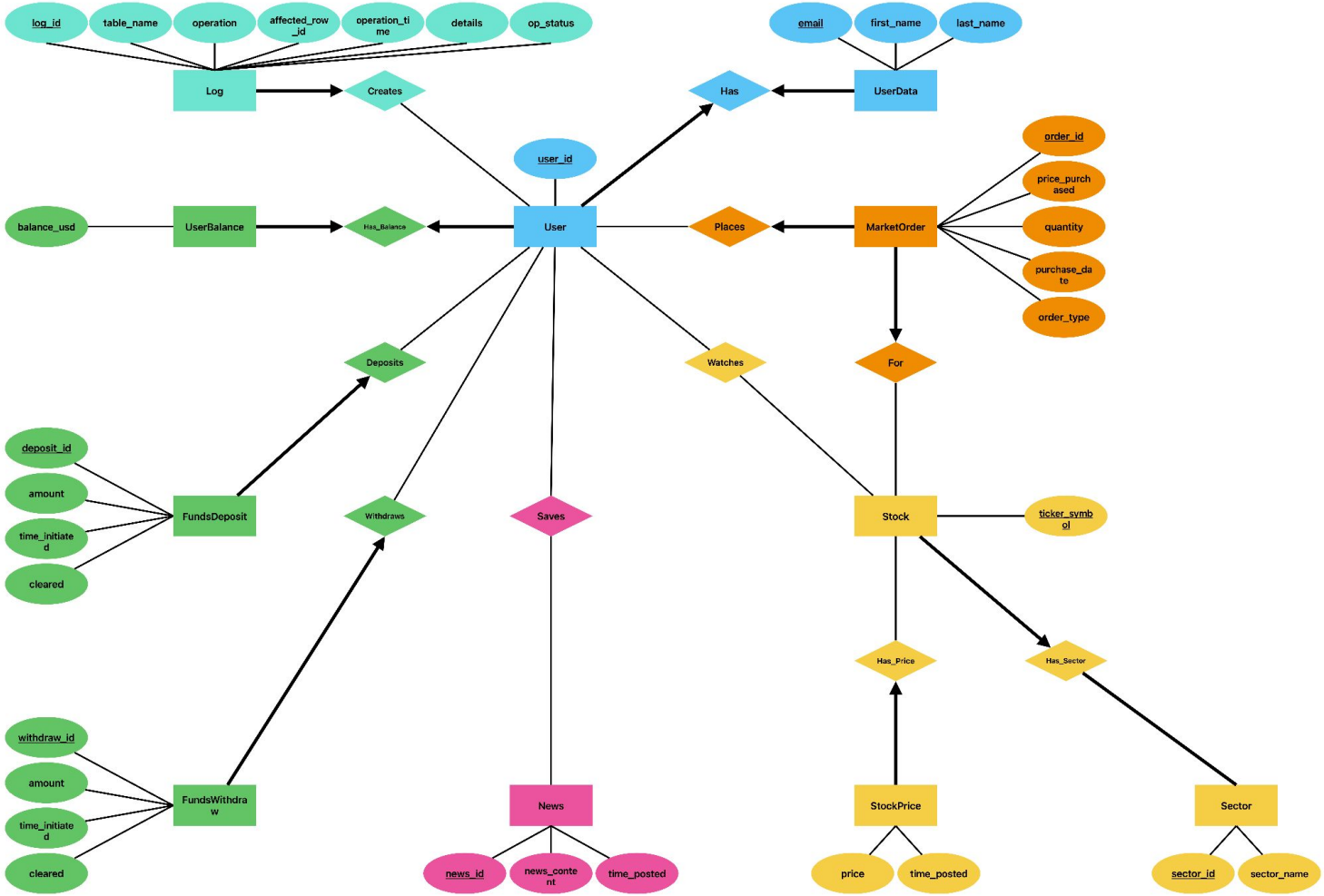
Workflow:

➔ Users interact with frontend.

➔ Backend processes requests and updates the database.

➔ Data pulled from a mock S&P 500 data source (e.g., API integration or stored tables).


Front End
Back End
DB
Website Image
React Application (JS)
route calls
json
Flask app.py (python)
function calls
json
DataBase.py (python)
SQL queries
SQL data
db.sql (mySQL)
JS
CSS
main_fetch.py
quick_fetch.py


log_id
table_name
operation
affected_row_id
operation_time
details
op_status
Log
Creates
email
first_name
last_name
UserData
Has
user_id
order_id
price_purchased
quantity
purchase_date
order_type
balance_usd
UserBalance
Has_Balance
User
Places
MarketOrder
Deposits
Watches
For
deposit_id
amount
time_initiated
cleared
FundsDeposit
Withdraws
Saves
Stock
ticker_symbol
Has_Price
Has_Sector
withdraw_id
amount
time_initiated
cleared
FundsWithdraw
News
StockPrice
Sector
news_id
news_content
time_posted
price
time_posted
sector_id
sector_name

Firebase

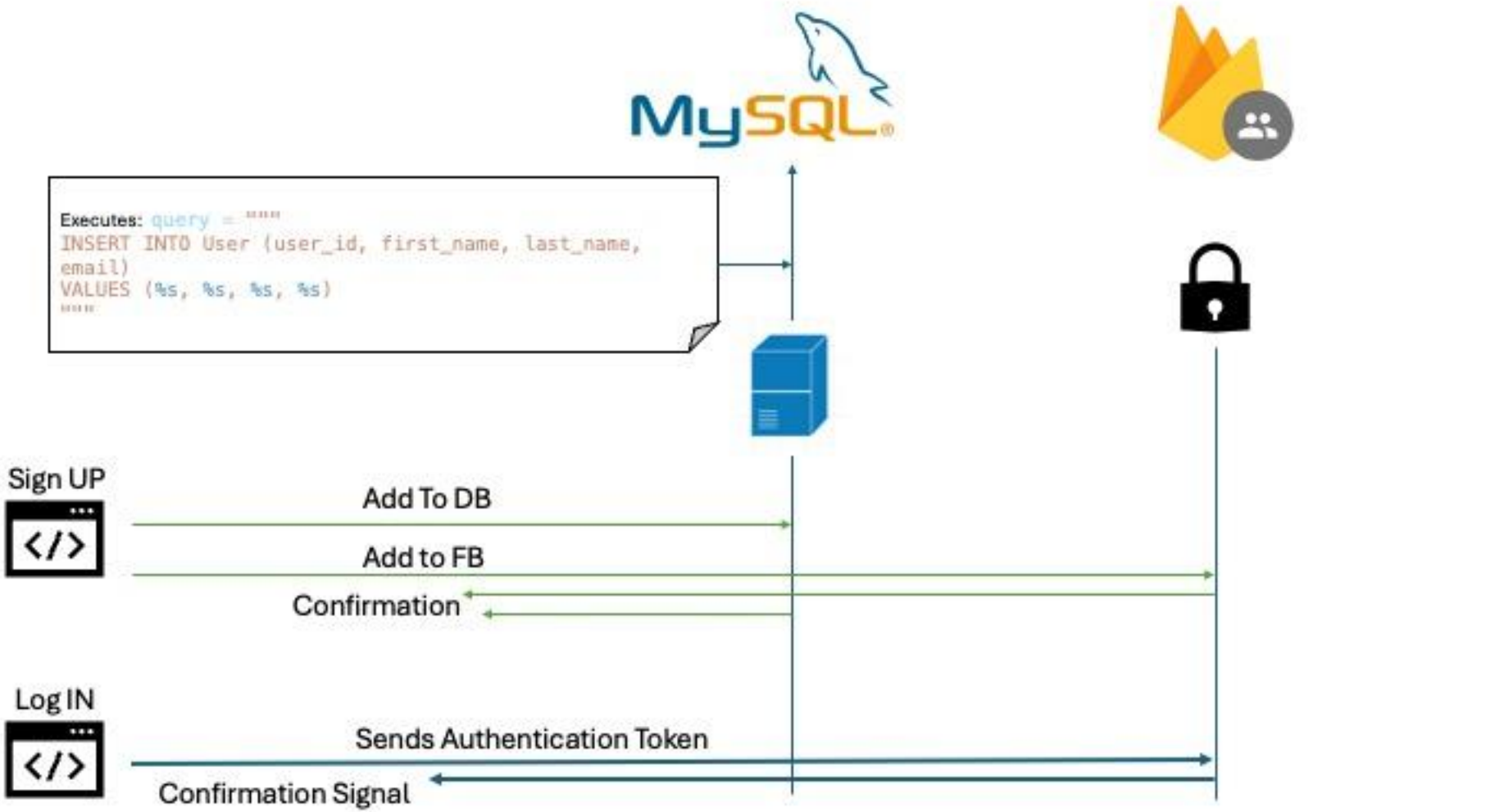
➔ A cloud-base platform offering various services for building, deploying and scale small and mobile app.

Authentication:

➔ Easy to use SDK, and ready to use UI libraries for user authentication.

Real-Time DB:

➔ A cloud-hosted NoSQL database that allows users to store and sync data in real time.


MySQL
Executes: query = """
INSERT INTO User (user_id, first_name, last_name,
email)
VALUES (%s, %s, %s, %s)
"""
Sign UP
Add To DB
Add to FB
Confirmation
Log IN
Sends Authentication Token
Confirmation Signal

Major Design Decisions

Database Normalization:

- ➔ Ensured BCNF for consistency.
- ➔ Tables: User, Stock, MarketOrder, UserBalance, Watchlist.

Frameworks:

- ➔ React for responsiveness.
- ➔ Flask for modular API.

Mock Real-Time Data:

- ➔ Stock prices stored and updated in StockPrice.

Scalability:

- ➔ Used cascading rules to handle deletions (e.g., SET NULL for dependent entries).

Application Demo

Sign Up and Login:

- ➔ Create a user (user_id, email).
- ➔ Add funds to virtual wallet using FundsDeposit.

Trade Stocks:

- ➔ Place a BUY order.
- ➔ View updated portfolio in UserBalance and MarketOrder.

Track Stocks:

- ➔ Add a stock to the Watchlist.
- ➔ Check price trends in StockPrice.

Market News:

- ➔ Save a news article to SavedNews.

Challenges Encountered

Database Normalization:

- ➔ Ensured no redundancy while maintaining relationships.
- ➔ Cascading Effects:
- ➔ Handling foreign key constraints for MarketOrder and Watchlist.

API Integration:

- ➔ Mocking real-time S&P 500 data.

User Wallet Updates:

- ➔ Ensured atomicity in deposits, withdrawals, and transactions.

Future Work

Real-Time Data:

➔ Integrate with live S&P 500 APIs.

Advanced Analytics:

➔ Portfolio performance.

➔ Risk assessment tools.

Gamification:

➔ Leaderboards and rewards for user engagement.

Enhanced Security:

➔ Implement two-factor authentication.

Conclusion

"Our platform bridges the gap between theory and practice, making stock trading accessible, safe, and fun."

Achievements:

- ➔ Fully functional virtual trading system.
- ➔ Interactive user interface.
- ➔ Reliable backend with normalized data.

- ➔ Thank your audience!
- ➔ Questions?