📌 PROJECT SPEC FOR COPILOT

Unified Stock & Crypto Trading Platform (Paper Trading, AI-Lite)

1️⃣ PROJECT OVERVIEW

Build a **full-stack web application** that simulates **stock and crypto trading** using **virtual money**.
The platform includes authentication, wallet, market explorer, trading, portfolio analytics, AI-assisted insights, and payment integration (test mode).

Purpose: Educational / demo platform
No real trading

2️⃣ TECH STACK (FIXED)

- Frontend: **React + Tailwind CSS**
- Backend: **FastAPI (Python)**
- Database: **MySQL**
- Auth: **JWT**
- Payments: **Stripe or Razorpay (test mode)**
- Charts: **Chart.js or Recharts**
- Market Data:
  - Crypto → CoinGecko API
  - Stocks → Alpha Vantage / Twelve Data

3️⃣ USER ROLES

USER

- Register / Login
- View market data
- Add virtual money
- Buy / Sell assets
- View portfolio
- View AI recommendations

ADMIN (optional)

- View users
- View total trades

- View total virtual money

4️⃣ PAGES TO BUILD (FRONTEND)

Public

- /login
- /register

Protected

- /dashboard
- /markets
- /trade/:assetId
- /portfolio
- /wallet
- /payments
- /about

5️⃣ DATABASE SCHEMA (IMPORTANT FOR COPILOT)

USERS

id (INT, PK)

name (VARCHAR)

email (VARCHAR, UNIQUE)

password_hash (VARCHAR)

role (ENUM: user, admin)

created_at (TIMESTAMP)

WALLETS

id (INT, PK)

user_id (FK -> users.id)

balance (DECIMAL)

updated_at (TIMESTAMP)

ASSETS

id (INT, PK)

symbol (VARCHAR)

name (VARCHAR)

type (ENUM: stock, crypto)

HOLDINGS

id (INT, PK)

user_id (FK)

asset_id (FK)

quantity (DECIMAL)

avg_buy_price (DECIMAL)

ORDERS

id (INT, PK)

user_id (FK)

asset_id (FK)

order_type (ENUM: buy, sell)

quantity (DECIMAL)

price (DECIMAL)

status (ENUM: completed)

created_at (TIMESTAMP)

TRANSACTIONS

id (INT, PK)

user_id (FK)

amount (DECIMAL)

type (ENUM: deposit, trade)

created_at (TIMESTAMP)

---

6 BACKEND API CONTRACT

AUTH

POST /auth/register

POST /auth/login

MARKET DATA

GET /markets/stocks

GET /markets/crypto

WALLET

GET /wallet

POST /wallet/add-money

TRADING

POST /trade/buy

POST /trade/sell

GET /orders/history

PORTFOLIO

GET /portfolio

AI INSIGHTS

GET /ai/recommendation?asset_id=

PAYMENTS

POST /payment/create-order

POST /payment/verify

7️⃣AI LOGIC (RULE-BASED, SIMPLE)

For each asset:

- Calculate:
    - Moving Average (short vs long)
    - RSI

- Output:

```
{
  "signal": "BUY | HOLD | SELL",
  "confidence": "Low | Medium | High",
  "reason": "RSI is low and price is above MA"
}
```

Label this as **“AI-Assisted Insight”**

8️⃣UI REQUIREMENTS

- Dashboard cards:
  - Wallet balance
  - Portfolio value
  - Total P/L
- Market table:
  - Name
  - Price
  - % change
  - Buy button
- Charts:
  - Price history
  - Portfolio allocation
- Clean fintech UI
- Mobile responsive

9️⃣SECURITY & CONFIG

- JWT auth
- Protected routes
- Password hashing
- Environment variables:
  - DB credentials
  - API keys

    - Payment keys
- CORS enabled

🔟 DISCLAIMER (FOOTER TEXT)

“This platform is for educational and simulation purposes only.
No real trading is performed.”

1️⃣1️⃣ DEPLOYMENT REQUIREMENTS

- Frontend deployed (Vercel / Netlify)
- Backend deployed (Render / Railway)
- MySQL hosted (Railway / PlanetScale)
- All API calls working in production

✅ END RESULT EXPECTATION

- User can:
    - Register
    - Add money
    - View stocks & crypto
    - Buy / sell virtually
    - View portfolio & AI insights
- Fully deployed and publicly accessible