**✅ 1. Flow of the Whole Model (User Journey from Login to Buy)**

[User Login]

↓

[User Lands on Dashboard]

↓

[User Selects Stock or Adds to Watchlist]

↓

[Data Engine loads past + real-time OHLCV]

↓

[Two things happen in parallel]:

→ (A) AI Prediction Engine runs

→ (B) User Strategy Engine runs (if defined)

↓

[Signal Engine combines result]

↓

[Suggestions shown: "BUY" / "SELL"]

↓

[Alert sent (optional)]

↓

[User clicks "Trade Now"]

↓

[Order placed via Broker API (Zerodha, Dhan)]

↓

[Trade logged & visible in History]

**✅ 2. How Each Functionality Will Be Created (One by One)**

**1️⃣ Data Collection Engine**

* 📥 **What it does:** Fetch OHLCV data (5min, 1hr, daily etc.)
* 📦 **Source:** Yahoo Finance (start simple) → Broker API later (e.g. Zerodha)
* 🔧 **Tools:** yfinance, kiteconnect, or dhanhq
* 📂 **Output:** DataFrame like:

| date | open | high | low | close | volume |

FLOW

[User selects stock or opens dashboard]

↓

[Backend calls Broker API (e.g. Dhan)]

↓

[Fetch historical OHLCV data]

↓

[Format data into DataFrame/JSON]

↓

[Return to next steps:

→ Indicator Engine

→ ML Prediction

→ Frontend chart (React)

]

**Steps :**

✅ DhanHQ (Best for Starting)

* ✅ Free access for live price + historical data
* ✅ Easy API docs
* ✅ Real-time WebSocket streaming
* ✅ Simple to set up

📌 Docs: https://docs.dhan.co

✅ Zerodha Kite Connect (Very Stable)

* ✅ Most used in India
* ✅ High reliability
* ❌ Paid API (₹2000/month approx)
* ✅ Used by Groww, Smallcase, etc.

📌 Docs: https://kite.trade/docs/connect/v3/

✅ AngelOne SmartAPI

* ✅ Free real-time API
* ✅ F&O support
* ✅ Good for high-frequency apps

📌 Docs: https://smartapi.angelone.in

**2️⃣ Indicator/Feature Engine**

* 📥 **Input:** OHLCV DataFrame
* 📦 **Process:** Add RSI, MACD, SMA, EMA using ta or ta-lib
* 🔧 **Tools:** ta, pandas
* 📂 **Output:** Enhanced DataFrame with extra columns:

| close | RSI | MACD | SMA |

**3️⃣ AI Prediction Engine**

* 📥 **Input:** Price + indicator features
* 📦 **Model:** XGBoost / LSTM / Random Forest
* 🔧 **Tools:** scikit-learn, xgboost, keras
* 🎯 **Goal:** Predict next move: 1 = BUY, 0 = HOLD, -1 = SELL
* 📂 **Output:** Model’s prediction + confidence

{ "signal": "BUY", "confidence": 87% }

**4️⃣ Signal Generator**

* 📥 Input: Model result (or strategy result)
* ⚙️ Logic:

python

if prediction == 1:

signal = "BUY"

elif prediction == -1:

signal = "SELL"

else:

signal = "HOLD"

* 📂 Output: Signal saved in DB + shown in UI

**5️⃣ Backtesting Engine**

* 📥 Input: Historical OHLCV + strategy or model
* 🔄 Apply logic on past data (loop row-by-row)
* 📦 Output:
  + Win rate, PnL, Sharpe ratio, accuracy
  + Visual trades on chart
* 🔧 Tools: pandas, matplotlib, backtrader

**6️⃣ Suggestion Dashboard (Frontend)**

* 📈 Display candle chart
* 🧠 Show:
  + Latest signal: “BUY at ₹450”
  + Reason: “MACD crossover + low RSI”
  + Confidence: 82%
* 🔧 Tools: React, lightweight-charts, REST API

**7️⃣ Alert System**

* ⚙️ Run every X seconds
* ✅ If new signal → send:
  + Telegram / Email / SMS
* 🔧 Tools: python-telegram-bot, smtplib, apscheduler

**8️⃣ Auto Trading (Optional)**

* 🎯 On "BUY" signal:
  + Place order on broker via API
* 🔧 Tools: kiteconnect (Zerodha), angelone.smartapi
* ✅ Add risk management (max daily trades, stop loss, etc.)

**9️⃣ Custom Strategy Engine**

* 🧠 Option A: User writes Python logic (via code editor)
* 🧠 Option B: User uses drag-and-drop strategy builder (visual)
* 🔄 Run on backend with user data
* 📩 Save result in DB → show on dashboard

**✅ 3. How Functionalities Are Connected to Each Other**

Here’s a **functional dependency map**:

[Login System] → [User Dashboard]

↓

[Data Collection Engine]

↓

[Indicator Engine]

↓

┌────────────────────┬─────────────────────────────┐

│ │ │

↓ ↓ ↓

[AI Prediction] [Custom Strategy Engine] [Backtesting Engine]

↓ ↓ ↓

[Signal Generator]

↓

[Signal Database + UI]

↓

[Alert System] → Notify

↓

[Auto Trading] → Place orders

**🔗 Key Integration Points**

| **Link** | **Description** |
| --- | --- |
| Data → Indicators | Indicators are built on top of OHLCV |
| Indicators → AI/Strategy | Both AI & user logic use the indicators |
| AI/Strategy → Signal Engine | Signal logic comes from AI or user |
| Signal Engine → Dashboard | Shows final actionable suggestion |
| Signal Engine → Alerts | Triggers Telegram/email |
| Signal Engine → Auto Trading | If enabled, sends order |

**🛠 Suggested Folder Structure**

/backend/

└── /data/

└── fetcher.py # OHLC data

└── indicators.py

└── /models/

└── ai\_model.py

└── backtest.py

└── /strategies/

└── user\_engine.py

└── /signals/

└── generator.py

└── alert.py

└── api.py (FastAPI)

└── database.py

/frontend/

└── /components/

└── /pages/

└── strategy-builder/