# Self-Learning Stock Trading Bot

1.) Ananya Shroff- UCE2023563
2.) Divyanshi Singh - UCE2023565
3.) Sanika Tavate - UCE2023566
4.) Trupti Patil-UEC2023163

# Introduction


- **The Challenge:** Financial markets are highly volatile and complex. Developing automated trading strategies that can adapt to changing market conditions and manage risk is extremely difficult.
- **The Goal**: To design and implement an intelligent agent that learns optimal, profitable trading strategies directly from market data through trial and error.
- **Our Project**: We present an end-to-end system featuring a Deep Q-Network (DQN), a type of reinforcement learning agent, trained for automated stock trading.
- **Key Feature**: The system includes a dynamic, real-time web-based visualization interface to simulate, analyze, and monitor the agent's performance interactively.

# Our Solution

We developed an integrated system with three main functions:

1. **Automated Learning:** A reinforcement learning agent (DQN) that trains on historical stock data to discover complex trading patterns and strategies on its own.
2. **Integrated Risk Management:** The trading environment is built with practical, automatic risk controls, including Stop-Loss (5%) and Take-Profit (8%) triggers, position size limits, and transaction cost modeling (0.1% per trade).
3. **Real-Time Visualization:** A web-based dashboard that simulates the agent's performance, streaming every trade, decision, and portfolio change live to the user for in-depth analysis.

# AIML Techniques Used

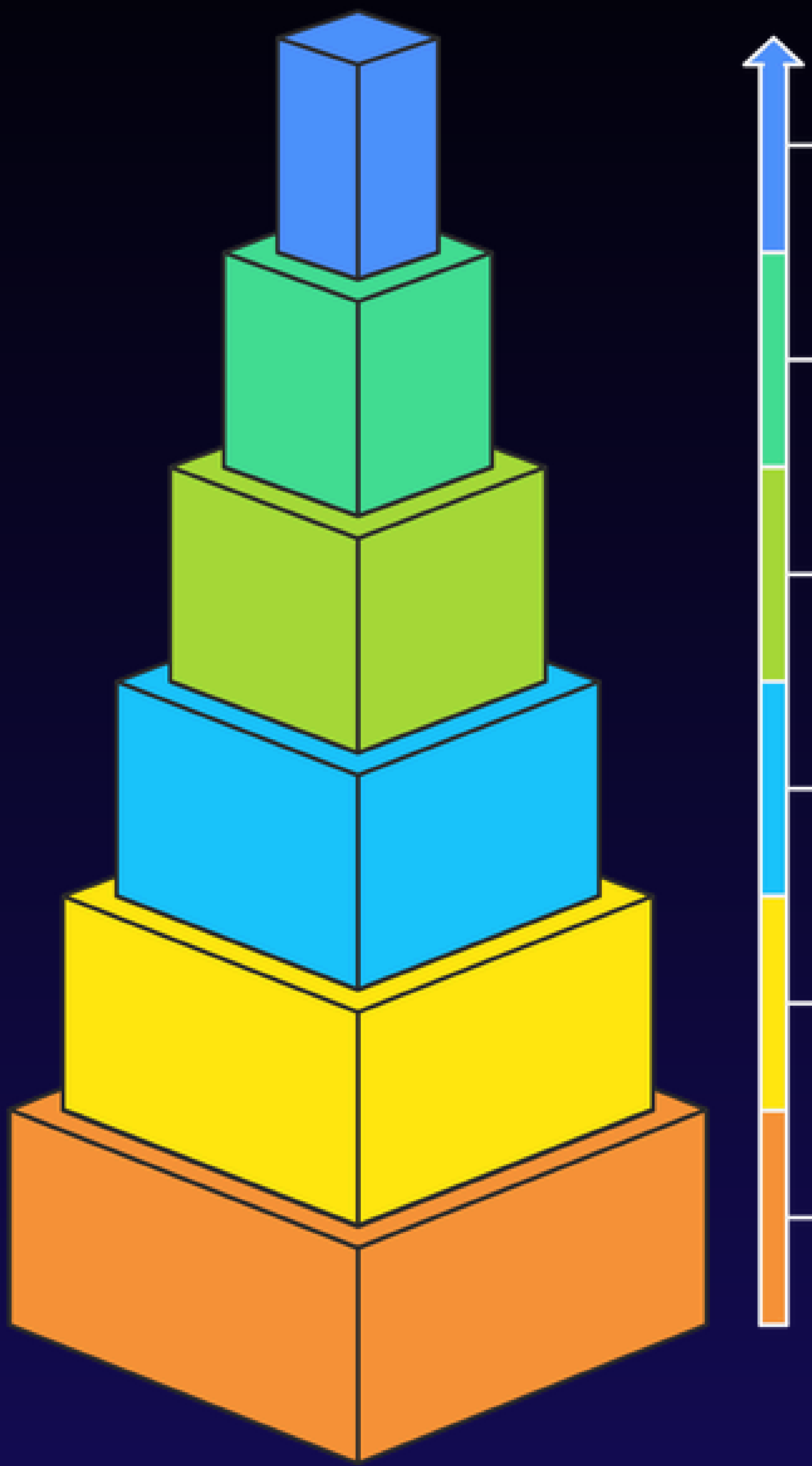
01 **Reinforcement Learning (RL):** The core methodology where an "agent" learns to make optimal decisions (actions) in an "environment" to maximize a cumulative "reward."

02 **Deep Learning:** Using a multi-layer neural network to approximate the complex Q-value function, which estimates the expected future reward for each possible action.

03
- **Deep Q-Network (DQN):** The specific algorithm that combines Reinforcement Learning with a Deep Neural Network. It uses key features like:
- Experience Replay Buffer: Stores past experiences (state, action, reward, next state) and samples from them randomly to break correlations and stabilize learning.
- Target Network: A separate, periodically updated copy of the main network used to create more stable learning targets.

04
- **Advanced Feature Engineering:**
- Temporal Lookback: A 30-timestep rolling window of historical returns, allowing the agent to see recent price momentum.
- Technical Indicators: 13 distinct, normalized indicators (RSI, MACD, Bollinger Bands, etc.) provide the agent with a rich understanding of the market state.
- Data Normalization: Using Z-scores and tanh scaling to ensure all features are in a stable range for the neural network.

# DQN Model Architecture



### Input Layer

43 features for market analysis

### Hidden Layers

2 layers with 256 neurons each

### Output Layer

3 Q-values for action selection

### Action Space

3 discrete actions: hold, buy, sell

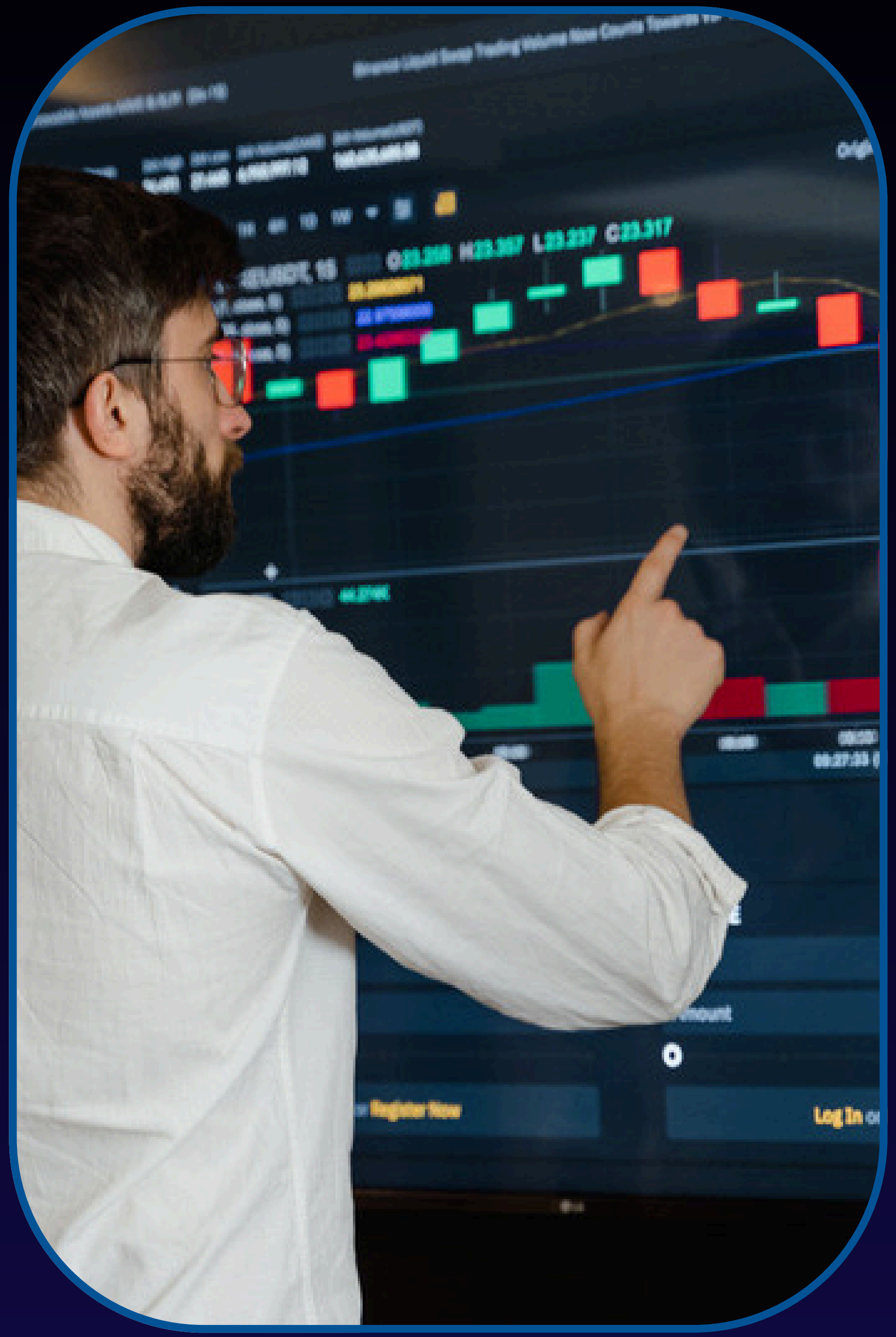
### Observation Space

43-D vector of market data

30 past normalized returns, 13 normalized features

### Reward Function

Maximizes portfolio value



# Tech Stack

## 1.Backend & AI:

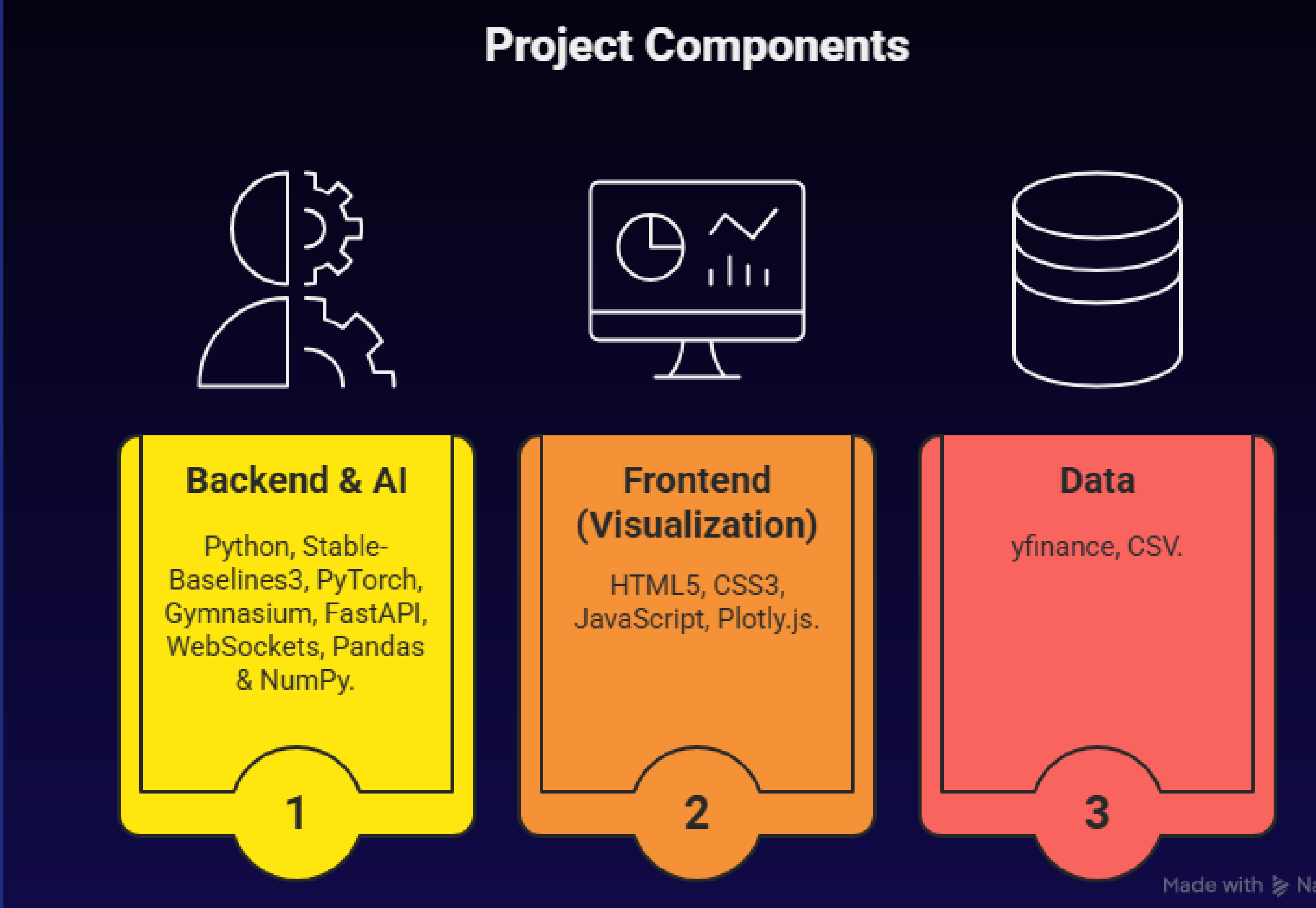
- Python 3.10+: Core programming language.
- Stable-Baselines3: The library used for the DQN algorithm.
- PyTorch: The deep learning framework powering the neural network.
- Gymnasium (OpenAI Gym): The toolkit used to build the custom trading environment.
- FastAPI: High-performance web framework for the backend API.
- WebSockets: For real-time, bidirectional communication with the frontend.
- Pandas & NumPy: For data manipulation and numerical operations.

## 2.Frontend (Visualization):

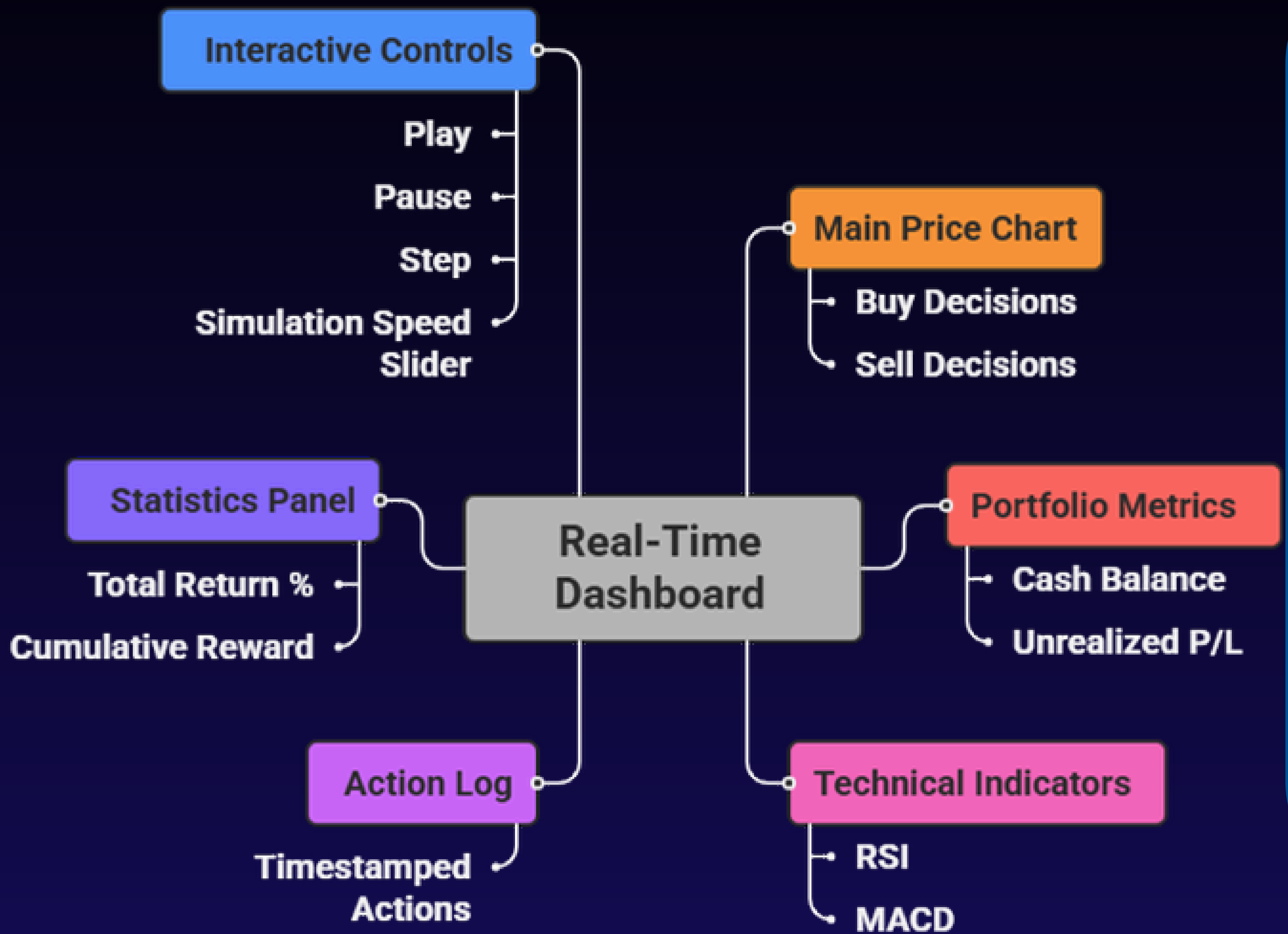
- HTML5, CSS3, JavaScript: The foundation of the web dashboard.
- Plotly.js: The interactive charting library used to render all graphs and charts.

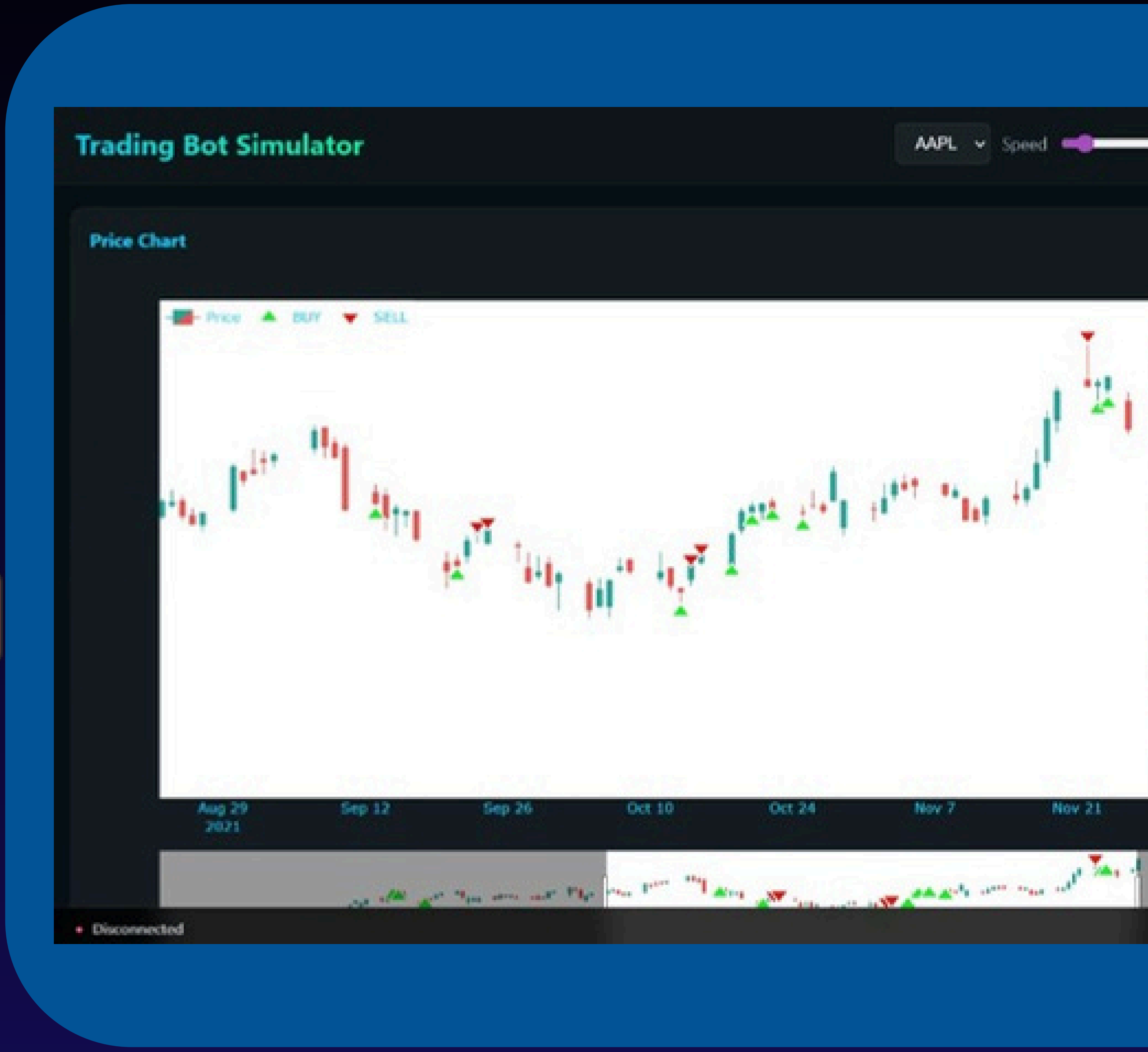
## 3.Data:

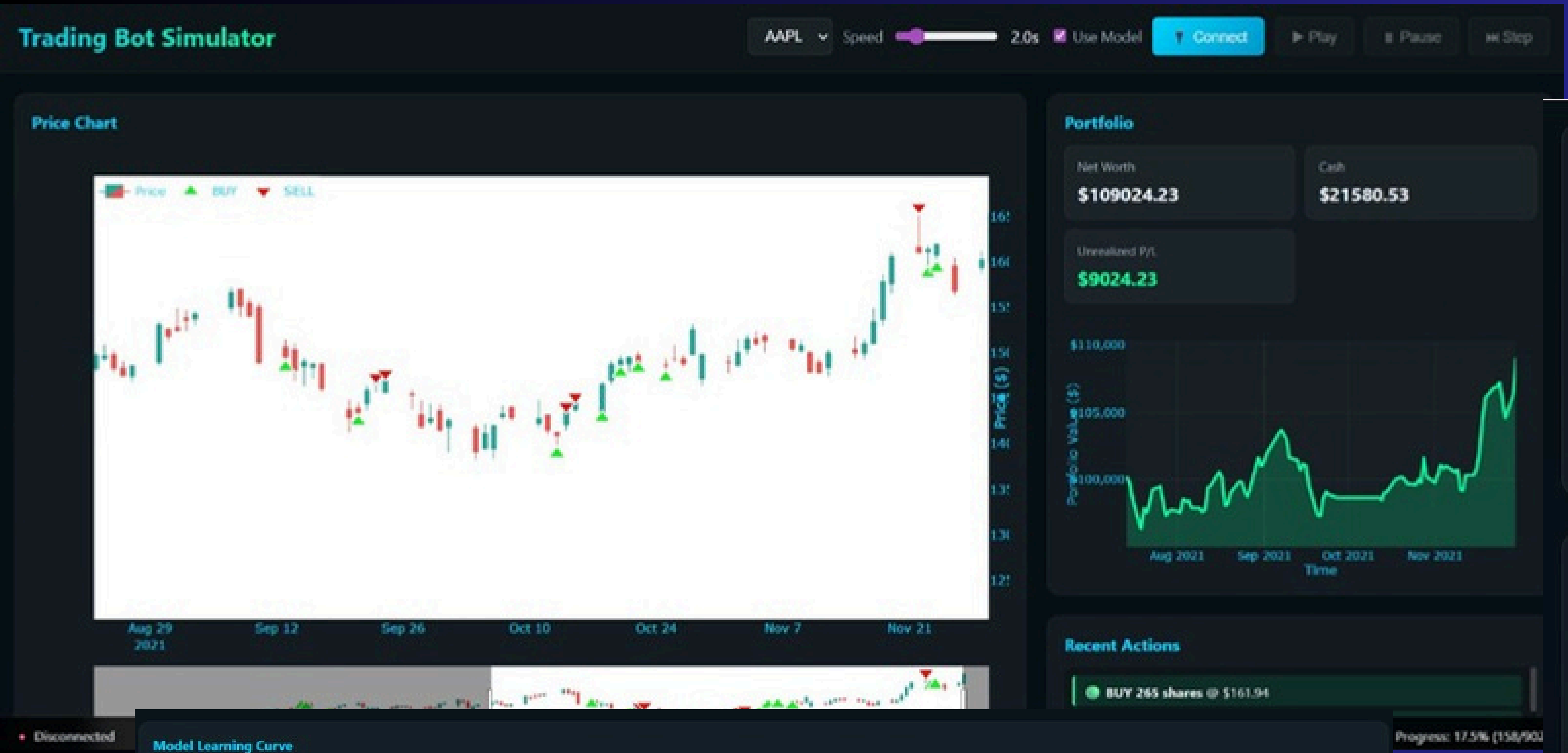
- yfinance: Python library to download historical stock data from Yahoo Finance.
- CSV: Storage format for processed data.

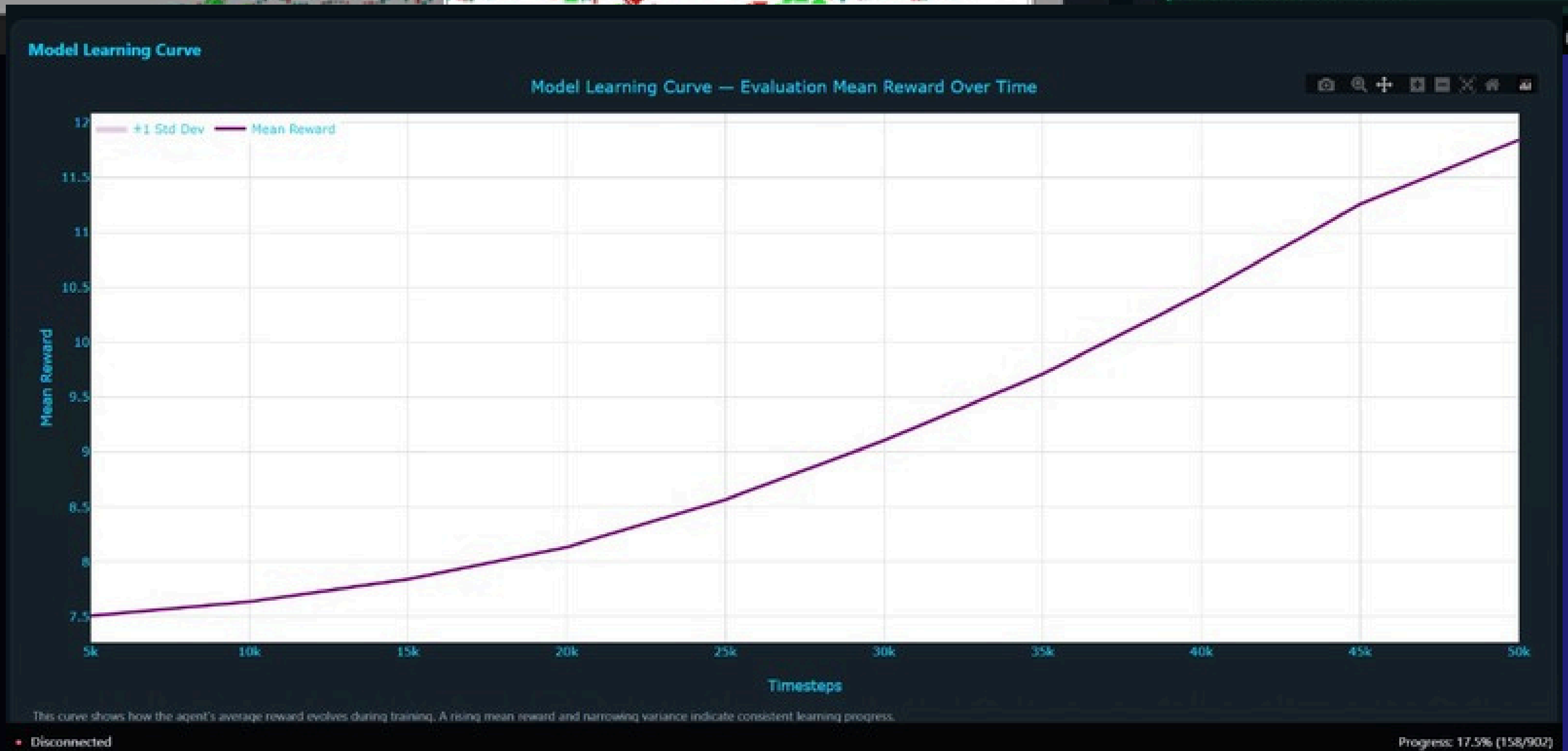


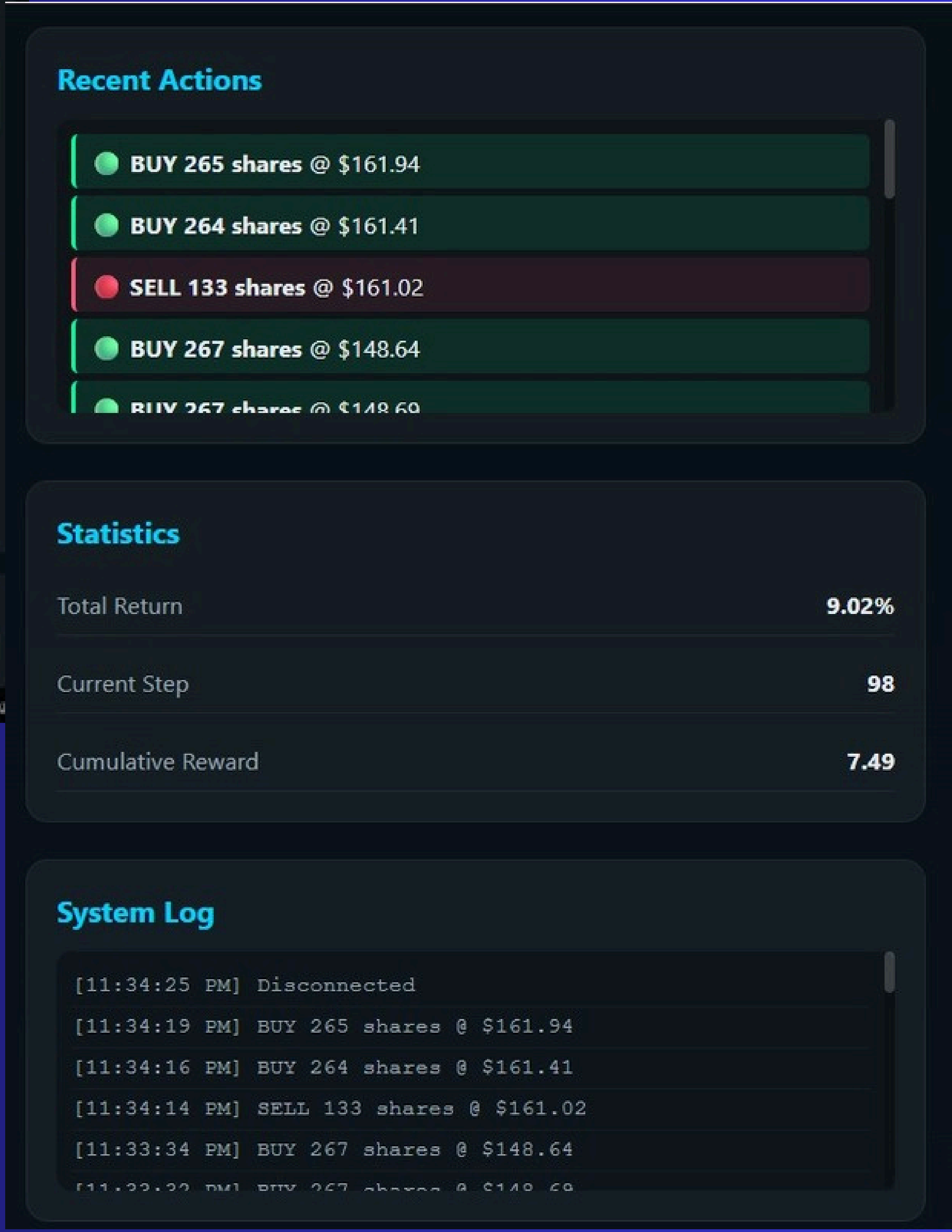
# Real-Time Dashboard Components



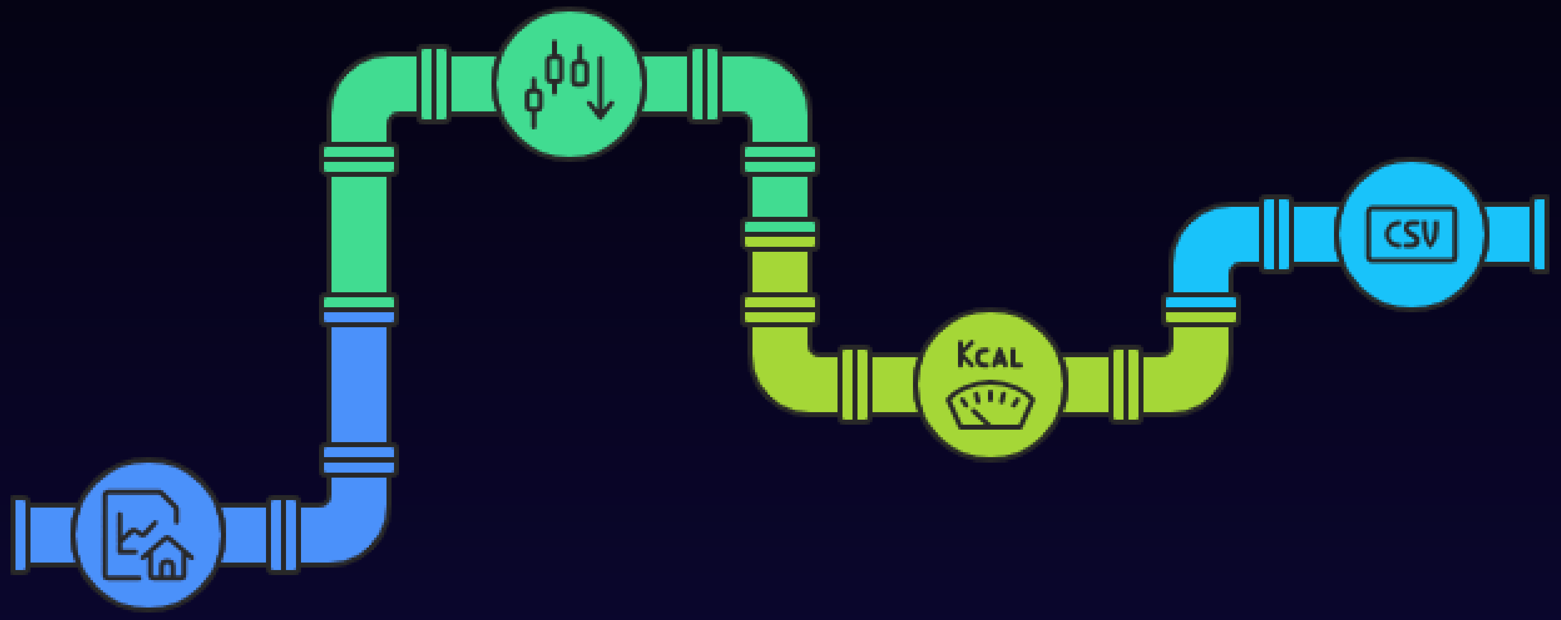

Trading Bot Simulator
AAPL
Speed
2.0s
Use Model
Connect
▶ Play
Pause
Step
Price Chart
Price
BUY
SELL
Price ($)
Aug 29 2021
Sep 12
Sep 26
Oct 10
Oct 24
Nov 7
Nov 21
Portfolio
Net Worth
$109024.23
Cash
$21580.53
Unrealized P/L
$9024.23
$110,000
$105,000
$100,000
Portfolio Value ($)
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Time
Recent Actions
BUY 265 shares @ $161.94
Disconnected


Model Learning Curve
Model Learning Curve — Evaluation Mean Reward Over Time
±1 Std Dev
Mean Reward
Mean Reward
12
11.5
11
10.5
10
9.5
9
8.5
8
7.5
5k
10k
15k
20k
25k
30k
35k
40k
45k
50k
Timesteps
This curve shows how the agent's average reward evolves during training. A rising mean reward and narrowing variance indicate consistent learning progress.
Disconnected
Progress: 17.5% (158/902)


Recent Actions
BUY 265 shares @ $161.94
BUY 264 shares @ $161.41
SELL 133 shares @ $161.02
BUY 267 shares @ $148.64
Statistics
Total Return
9.02%
Current Step
98
Cumulative Reward
7.49
System Log
[11:34:25 PM] Disconnected
[11:34:19 PM] BUY 265 shares @ $161.94
[11:34:16 PM] BUY 264 shares @ $161.41
[11:34:14 PM] SELL 133 shares @ $161.02
[11:33:34 PM] BUY 267 shares @ $148.64

# System Architecture

## Data Pipeline Process



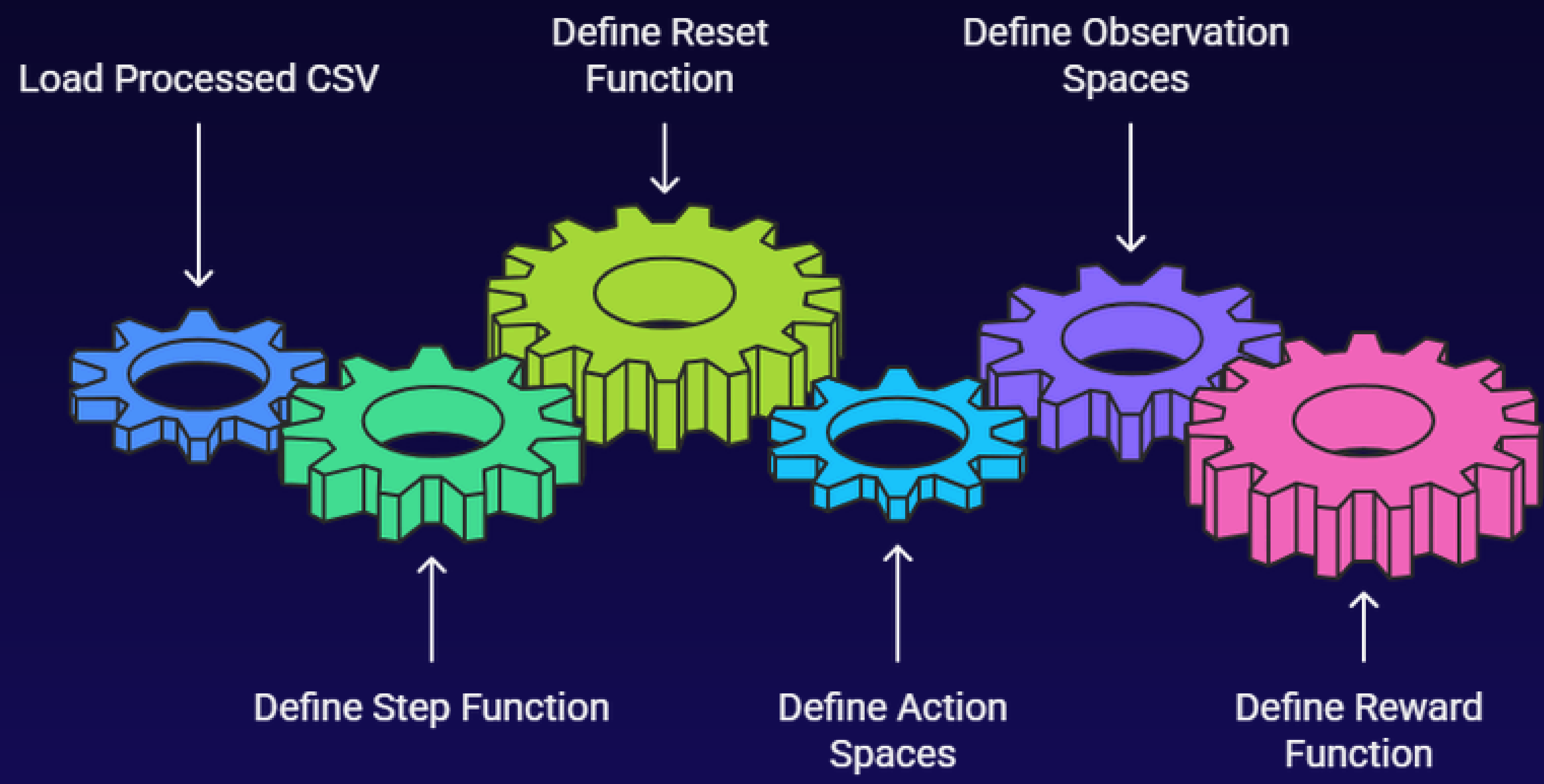
01
**Download Raw Data**
Fetches OHLCV data from Yahoo Finance

02
**Calculate Indicators**
Computes 13 technical indicators and returns

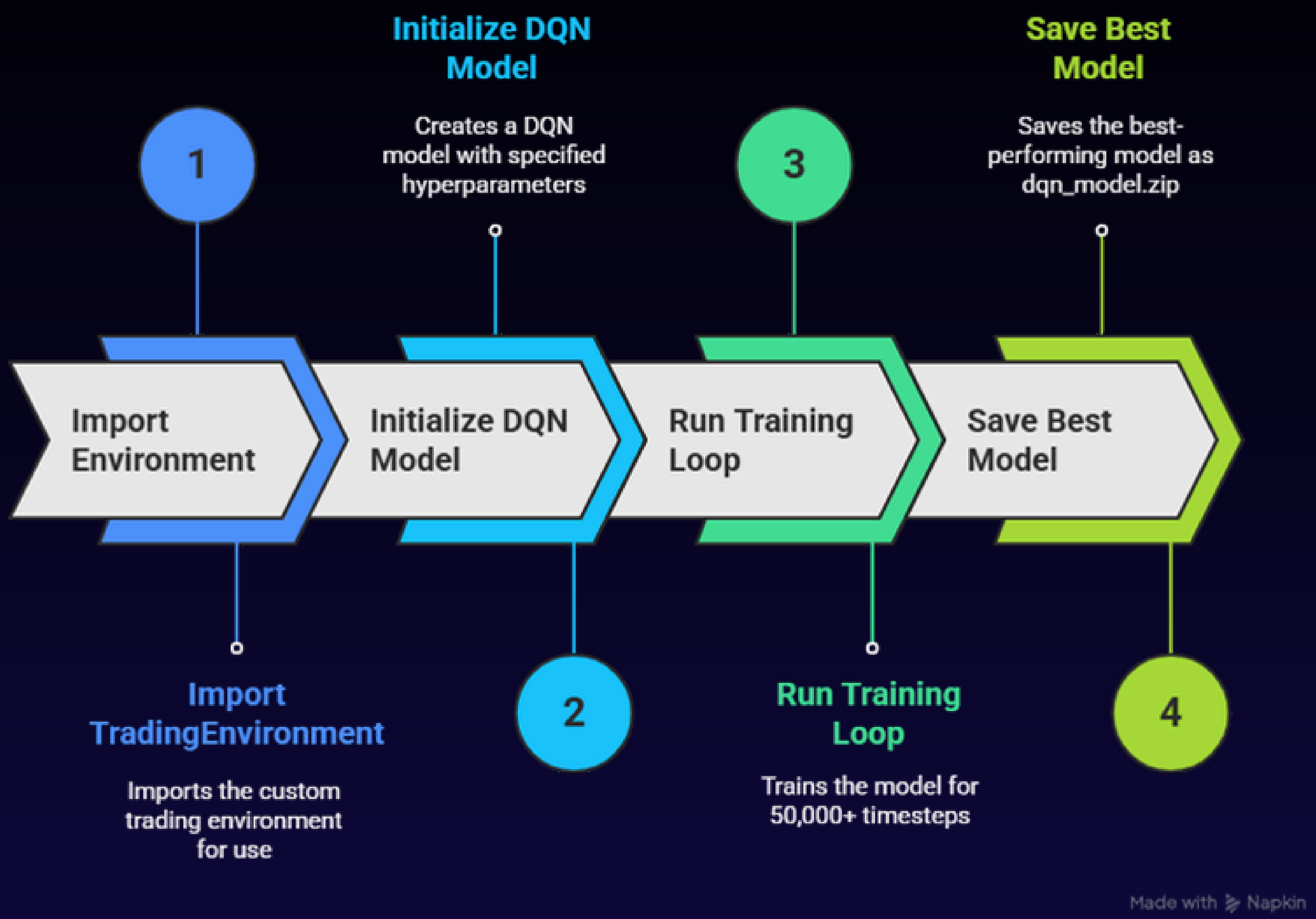
03
**Normalize Data**
Scales data to a standard range

04
**Save to CSV**
Stores processed data in a CSV file

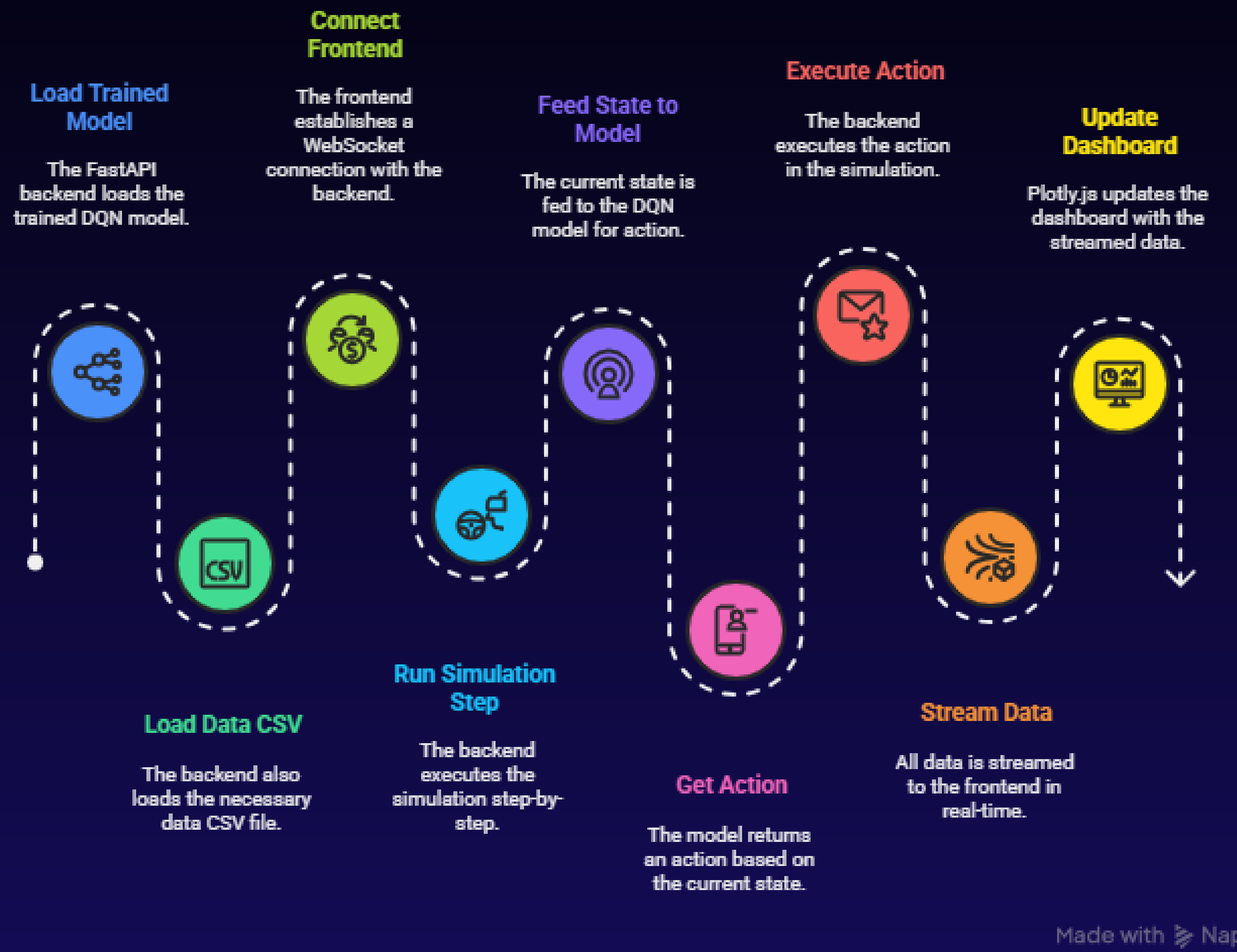
## Trading Environment Setup



## DQN Model Training Workflow




## Web Simulator Process



# Conclusion

We successfully built and demonstrated a complete, end-to-end system for developing and evaluating reinforcement learning-based trading agents.

- The Deep Q-Network model, combined with a rich feature set and integrated risk management, forms a robust foundation for automated trading.
- The real-time, interactive web dashboard proves to be an invaluable tool for analyzing agent behavior and building trust in the system.

- **Current Limitations:**
  - Single Asset: The agent can only trade one stock at a time.
  - Fixed Position Size: The agent can only Buy/Sell 1 share, not decide how much to trade.
  - Offline Only: The system runs on historical data, not live market data.
- **Future Work:**
  - Portfolio Management: Expand the agent to trade multiple assets simultaneously.
  - Variable Position Sizing: Allow the agent to learn the optimal amount of capital to risk per trade.
  - Live Trading Integration: Connect the system to a brokerage API for real-time paper and live trading.
  - Explore Other Models: Compare DQN's performance against other modern RL algorithms like PPO or A3C.
  - Alternative Features: Integrate new data sources, such as market news sentiment.

# References


- Mnih, V., et al. (2015). "Human-level control through deep reinforcement learning." Nature.
- Jiang, Z., et al. (2017). "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem." arXiv.
- Yang, H., et al. (2020). "Deep Reinforcement Learning for Stock Trading." IEEE.
- Raffin, A., et al. (2021). "Stable-Baselines3: Reliable Reinforcement Learning Implementations." Journal of Machine Learning Research (JMLR).



Thank You