Group 9

CRYPTO TRACKER WITH TREND PREDICTION

Advanced Computer Programming

AGENDA

Overview

Features

Architecture

Prediction

Demo

Conclusion

Overview: Data-Driven Insights

Our final project is a web-based dashboard that displays real-time cryptocurrency prices, historical trends, and short-term forecasts. It's built using Flask (backend), Chart.js (visualization), and Python libraries like requests, pandas, numpy, and xgboost for data processing and prediction.

Real-time Data

Live feeds for up-to-the-minute prices

Interactive Experience

User-friendly interface for dynamic exploration

Short-term price prediction

Using machine learning models (XGBoost), giving users insights into possible future prices.

Dashboard Overview

Our interactive dashboard provides immediate access to critical market information.

Asset Selector

Choose specific crypto assets for detailed views.

Timeframe Tabs

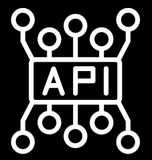
Select historical data ranges: 7 days, 1 month, 6 months, 1 year.

Interactive Tooltips

Reveal the exact date/time and price when you hover over any point on the chart.

Group 9

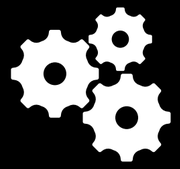
SYSTEM ARCHITECTURE



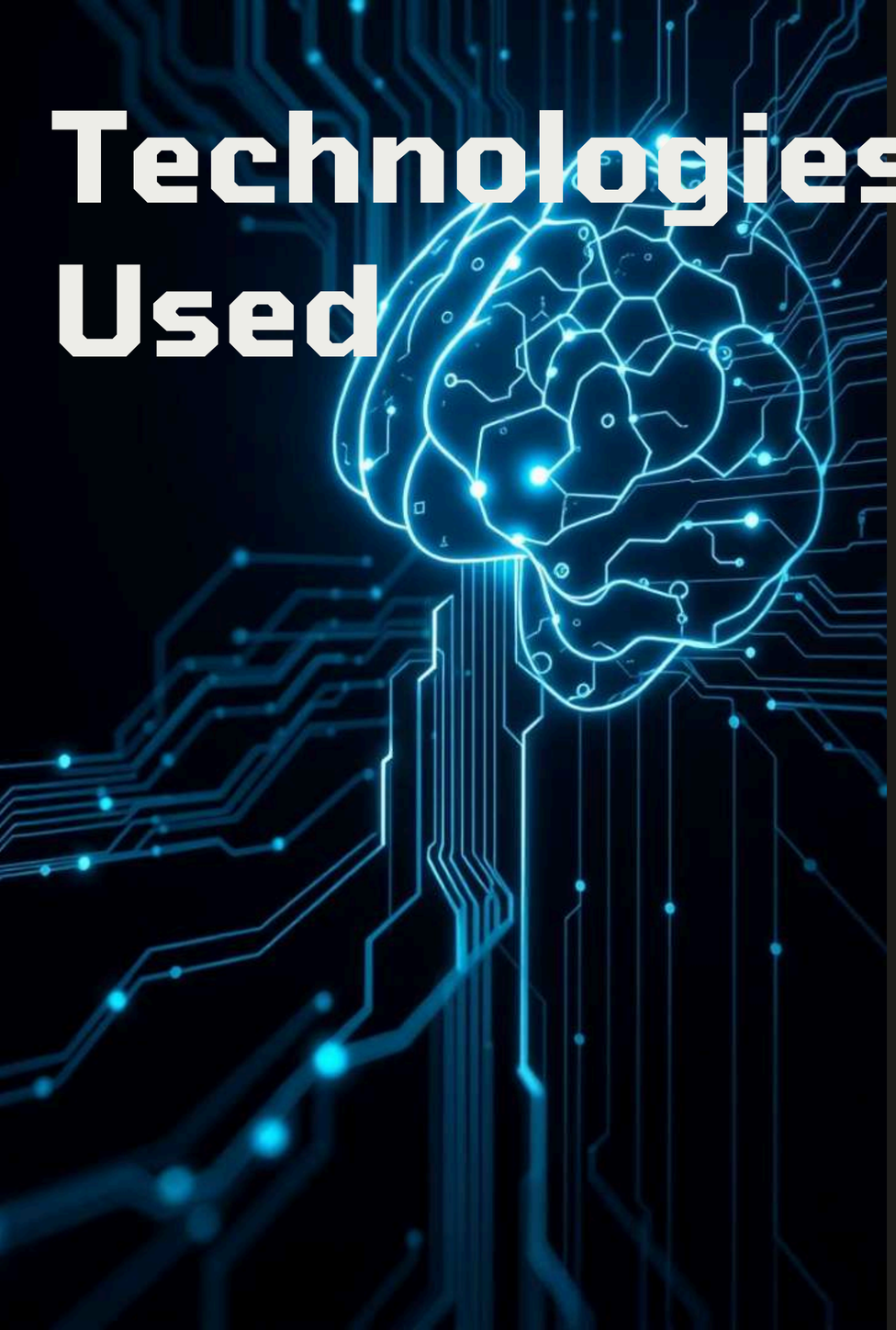
Built with HTML, CSS, and Chart.js to handle user interactions and dynamic visuals.

Python scripts use the CoinGecko API to retrieve and store historical data as CSVs.

Flask handles web routing, request handling, and dynamic template rendering.



A pre-trained XGBoost regression model is used to predict future prices based on engineered time series features.



Technologies Used

Flask

The core backend framework that connects data to UI.

requests

For real-time HTTP requests to the CoinGecko API.

pandas & numpy

or data wrangling, cleaning, and feature engineering.

xgboost

A high-performance gradient boosting library used for regression modeling.

scikit-learn

Used for pipeline integration, preprocessing, and model saving via joblib

Chart.js

A powerful JS library to create interactive, animated charts in the browser.

HTML/CSS/Jinja2

To build the visual and responsive components of the application.

Group 9

Real-Time Price Tracking

We use CoinGecko's market chart API to fetch real-time data (price, market cap, volume).

Flask renders the latest data each time the homepage loads.

The layout is fully responsive, adapting to desktop, tablet, or mobile screens.

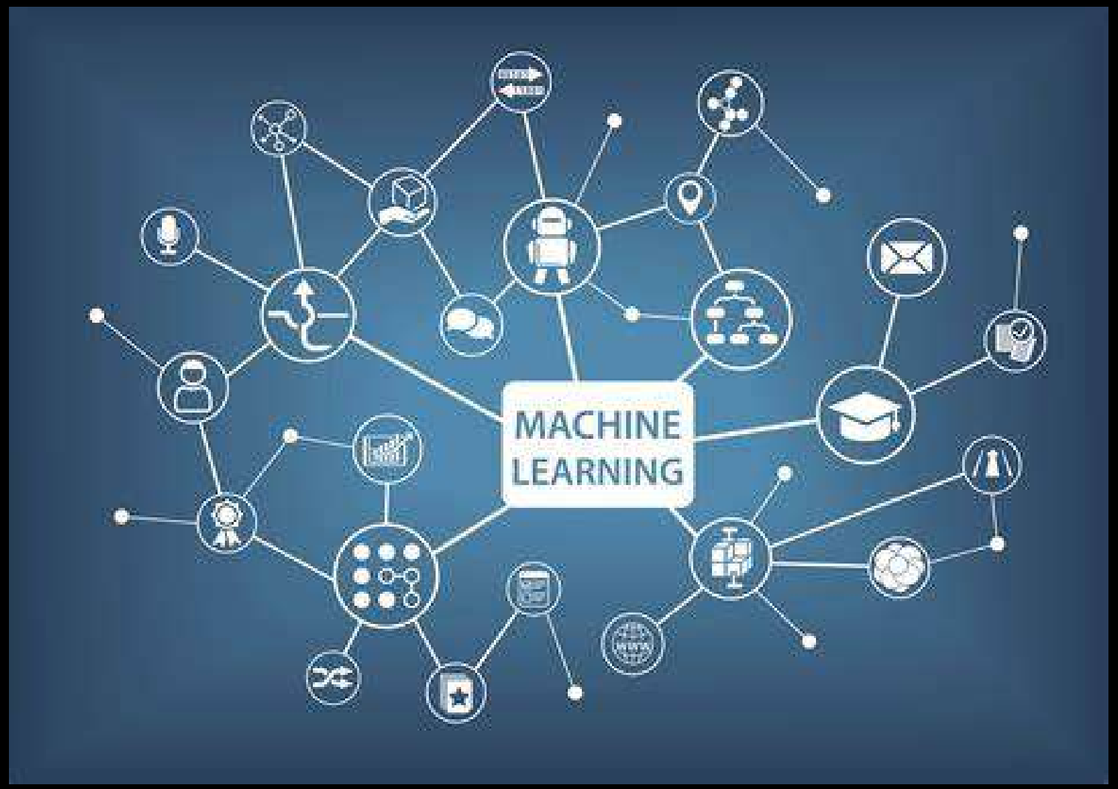
Historical Price Analysis

Visualize trends and patterns with our comprehensive historical data display.

7-day	98000	105560
1-month	101650	111560
6-month	76330	111560
1-Year	53920	111560

The area chart visualizes price movements, highlighting local minima and tracking recovery trajectories, like the recent 7-day "U-shaped" pattern.

Trend Prediction Logic



- Past prices turned into features using a sliding window
- XGBoost learns short-term trends and predicts future steps
- Adds Gaussian noise & trend drift for real-world feel
- Returns forecast with ±1.5% confidence bands

DEMO

How It Works

Here is the data flow in action:

Initialization

- Run **init_data.py** to cache historical data locally.

App Launch

- Start server with **python app.py.**
- Navigate to **http://127.0.0.1:5000.**

User Journey

- Homepage shows live prices and charts.
- User selects a coin on the prediction page.
- Model returns the next price forecast instantly.

All data flows are asynchronous and fast, ensuring a smooth user experience.

Challenges and Resolutions

The following challenges were encountered and future plans are underway to enhance the project.

Missing dependencies

- Import errors (e.g., xgboost, scikit-learn) resolved by using requirements.txt.

API Rate Limits

- CoinGecko call limits handled by caching data locally via init_data.py.

Model Integration

- Ensured feature consistency and used joblib for model loading and reuse.

Conclusions

Crypto Tracker with Trend Prediction successfully combines real-time tracking and basic forecasting in a single web interface. Through Flask, XGBoost, and Chart.js, we built a working system that mimics real crypto tracking websites. We learned to design modular code, work with time series data, implement predictive models, and deliver it all in a responsive user interface.

Future Improvements

To enhance this project further, we plan to:

- Support more coins and allow fiat conversions.
- Add dynamic time range selection for trend charts (e.g., 1M, 6M, YTD).
- Enable periodic auto-refreshing via scheduled background jobs (Celery, cron).
- Store user prediction history in a database for analytics or model feedback.
- Experiment with deep learning models (e.g., LSTM) for longer-term predictions.
- Deploy the app publicly using cloud services like Render, Heroku, or Vercel.

Group 9

THANK YOU

Advanced Computer Programming