Low-Level Design (LLD) Document

Main Files and What They Do

1.crypto_liquidity_model1.py

- Loads two CSV files (coin_gecko_2022-03-16.csv and 2022-03-17.csv)
- Cleans the data by removing rows with missing values
- Scales all numerical columns using StandardScaler
- Creates new features:
    - price_change_score (based on 1h, 24h, and 7d price changes)
    - volume_to_marketcap (volume divided by market cap)
- Generates EDA plots (histograms, scatterplots, heatmap)
- Trains an XGBoost regression model
- Prints performance: RMSE, MAE, R^2 score
- Saves the model (xgb_model.pkl) and scaler (volume_scaler.pkl)

2.app.py

- Launches a web app using Streamlit
- Loads the saved model and scaler
- Takes 4 numeric inputs from the user:
    - Price
    - Price Change Score
    - Volume-to-Market Cap Ratio

  - Market Cap
- Predicts the 24h trading volume
- Shows the result on the screen

3.xgb_model.pkl

- This is the machine learning model trained with the script.
- It's used to predict liquidity in the Streamlit app.

4.volume_scaler.pkl

- A saved scaler used to convert 24h volume values into a normalized scale.
- Helps reverse the model output if needed.

5.reports

- Stores all the charts generated during EDA:
  - Correlation heatmap
  - Histograms of price, volume, score
  - Market cap vs volume scatterplot

Functions and Tools Used

The following are tools or methods used for each tasks ;

Task	Tools/Methods Used
Data loading	pandas.read_csv()
Data cleaning	dropna()
Scaling	StandardScaler()
Feature creation	Column math (df['col'] = ...)
Model	XGBRegressor() from xgboost
Evaluation	mean_squared_error, r2_score, etc.
UI	Streamlit (`st.number_input`, `st.button`)

Inputs and Outputs

- Inputs: User provides 4 numerical values in the app

- Output: Predicted 24-hour trading volume shown on screen

Summary

Each file has a clear job:

- One trains the model
- One runs the app
- Others store saved files and visuals