**Pipeline Architecture and Document**

**1. Data Collection**

* Source: Historical cryptocurrency data (price, volume, market cap, etc.).
* Tools: pandas, yfinance, or existing dataset from Kaggle.
* Dataset spans multiple currencies and includes daily records.

**2. Data Preprocessing**

* **Missing Values**: Handled using interpolation and forward fill.
* **DateTime Formatting**: Ensured datetime index is sorted for time series models.
* **Irrelevant Features**: Dropped unused columns like "symbol", "Unnamed", etc.
* **Scaling**: Used MinMaxScaler for normalization (for models like LSTM/ANN).

**3. Exploratory Data Analysis (EDA)**

* Distribution analysis for liquidity-related features.
* Time series plots for volume, price, volatility.
* Correlation heatmaps to understand inter-feature relations.
* Stationarity check using rolling mean/ADF Test.

**4. Feature Engineering**

* Created advanced features like:
  + Rolling averages (7, 14, 30 days).
  + Price change %, volatility ratio.
  + Liquidity Index (volume/market cap).
  + Lag features for modeling temporal effects.
* **Dimensionality Reduction** (Optional): PCA if multicollinearity exists.

**5. Modeling Phase**

* **Train/Test Split**: TimeSeriesSplit used to avoid data leakage.
* **Algorithms Used**: Linear Regression, LSTM (for deep sequence learning).
* **Baseline Model**: Linear regression to benchmark performance.
* **Advanced Model**: LSTM using Keras with sequential layers.

**6. Model Evaluation**

* Evaluation metrics used:
  + MAE (Mean Absolute Error)
  + R² Score
* Visual comparison of actual vs. predicted liquidity values using line plots.

**7. Model Optimization**

* **Hyperparameter Tuning** via GridSearchCV for traditional models.
* **Early Stopping**, Dropout, and learning rate tuning for LSTM.
* Re-trained models after tuning to validate stability.

**8. Model Testing & Forecasting**

* Tested model on unseen future data.
* Analyzed prediction accuracy and interpreted model performance trends.

**9. Deployment Plan**

* Flask for real-time prediction.
* API input: new market data → model → liquidity prediction output.
* Used joblib/pickle to save model for deployment.

**Architecture Flow Diagram (Textual Form)**

Data Source (CSV)

↓

Preprocessing (cleaning, scaling)

↓

EDA + Feature Engineering

↓

Model Selection

↓

Model Training

↓

Evaluation & Tuning

↓

Final Prediction

↓

Deployment (Flask)

↓

Deployment (Render.com)

**10. Hosting:**

The application has been deployed live using Render with a Flask backend.

Live URL: https://cryptocurrency-liquidity-prediction-a1qg.onrender.com