Design Document

High Level

Objective:

Create a web app that uses a trained PyTorch model to predict future forex rates based on user-provided sequences.

Design Choices:

Component	Choice	Rationale
Frontend	React	Declarative UI, state management with hooks, component-based architecture
Backend	FastAPI	High-performance, asynchronous Python web framework with automatic docs
ML Framework	PyTorch	Industry-standard for ML research and deployment
Data Exchange	JSON via REST	Language-agnostic, easy to debug, integrates well with React

Backend Endpoints:

- `GET /model/info`: Returns model version, creation date, and performance metrics.
- `POST /predict`: Accepts input sequence, currency pair, and prediction horizon, returns prediction with confidence interval.

Low Level

Endpoint Definitions:

1. **`GET /model/info`**

   Input: None

   Output (JSON):

   ```
   {
     "model_version": "v1.0",
   ```

```
  "created_at": "2024-04-25T15:00:00Z",
  "metrics": {
    "MAE": 0.0134,
    "RMSE": 0.0235
  }
}
```

2. **`POST /predict`**

   Input (JSON):

```
{
  "sequence": [73.25, 73.36, 73.48, 73.55, 73.60],
  "currency_pair": "USD_INR",
  "horizon": 2
}
```

   Output (JSON):

```
{
  "predicted_value": 73.82,
  "confidence_interval": {
    "lower_bound": 73.68,
    "upper_bound": 73.95
  },
  "prediction_timestamp": "2025-04-30T10:15:00Z"
}
```

IO Spec:

- Input sequence is normalized, reshaped (e.g., `(1, sequence_length, features)`), and sent to `model.eval()`.
- Model outputs are post-processed to compute a point estimate and confidence interval (e.g., ±1.96 * std).