Customer Order Prediction High-Level Design (HLD)

Document Version Control

Date Issued	Version	Description	Author
01/10/2024	1.0	Initial HLD	[TARUN KUMAR]
02/10/2024	1.1	Updated sections and metrics	[TARUN KUMAR]

Contents

1. Abstract

The Customer Order Prediction system utilizes historical transaction data to forecast future customer purchases. By analyzing past buying patterns, businesses can optimize inventory levels, thus reducing costs and improving customer satisfaction.

---

2. Definitions

Term	Description
Customer Order	A record of items purchased by a customer in a transaction.
Prediction Model	A machine learning model used to forecast future orders.
API	Application Programming Interface for model exposure.

---

3. Introduction

3.1 Purpose

The purpose of this High-Level Design (HLD) document is to provide detailed insights into the architecture and design of the Customer Order Prediction system, ensuring all stakeholders understand the system components and their interactions.

3.2 Scope

This document covers the architecture, data flow, components involved in data collection, processing, and the predictive modeling of customer orders.

3.3 Why this High-Level Design Document?

The HLD serves as a blueprint for the development team, detailing:

- System architecture and design considerations.
- Interactions between components.
- User interface expectations and performance requirements.

---

4. General Description

4.1 Product Perspective

The Customer Order Prediction system is a machine learning-based application that analyzes transaction data to predict future purchases and assist in inventory management.

4.2 Problem Statement

Retailers face challenges in accurately forecasting customer orders, leading to stockouts or overstocking. This project aims to develop an AI solution that enhances prediction accuracy and inventory management.

4.3 Proposed Solution

The proposed solution involves implementing a predictive model that analyzes historical transaction data to forecast customer orders. This system will alert businesses about future demand, enabling better stock management.

4.4 Technical Requirements

- **Programming Languages**: Python for model development and API creation.
- **Libraries**: Pandas, NumPy, Scikit-learn for data processing and modeling.

4.5 Data Requirements

The model requires historical transaction data including:

- At least 1000 records with relevant features.
- Data should be clean, well-structured, and annotated where necessary.

4.6 Tools Used

- **Development**: Python, Pandas, Scikit-learn, Flask/FastAPI.
- **Visualization**: Matplotlib, Seaborn.
- **Deployment**: AWS or any cloud service for hosting the API.

4.7 Hardware Requirements

- A server capable of running Python applications.
- Sufficient RAM and CPU to handle data processing and model training.

---

5. Design Details

5.1 Process Flow

1. Data Collection
2. Data Cleaning and Transformation

3. Feature Engineering
4. Model Training
5. Prediction and Evaluation
6. Deployment of the Model

5.2 Model Training and Evaluation

The model will be trained on historical transaction data. Evaluation metrics will include Mean Squared Error (MSE) to assess performance.

5.3 Deployment Process

- The trained model will be saved using joblib or pickle.
- An API will be developed using Flask or FastAPI to facilitate predictions.

5.4 Event Log

The system will log events to track actions taken during data processing and model predictions for debugging and monitoring purposes.

5.5 Error Handling

Error messages will be generated for any inconsistencies or issues encountered during data processing or API requests.

5.6 Performance

The predictive model should achieve a low Mean Squared Error (MSE) and operate efficiently in terms of resource utilization.

5.7 Reusability

Code components and functions should be modular to facilitate reuse in other projects or applications.

5.8 Application Compatibility

The system will be compatible with existing e-commerce platforms and easily integrate with other services via the API.

5.9 Resource Utilization

The application will be optimized to use available processing power effectively, especially during model training and API requests.

---

6. Dashboards

6.1 KPIs (Key Performance Indicators)

1. Prediction accuracy (MSE).
2. Inventory turnover rates.
3. Alerts for low stock levels.
4. Response time for predictions via API.

---

7. Conclusion

The Customer Order Prediction system is designed to enhance inventory management by accurately forecasting customer orders. By leveraging historical data and advanced modeling techniques, the system provides actionable insights for e-commerce businesses.

---

8. References

1. Scikit-learn Documentation: https://scikit-learn.org/
2. Flask Documentation: https://flask.palletsprojects.com/
3. AWS Documentation: https://aws.amazon.com/documentation/