Low Level Design

Swiggy Restaurant Dashboard

Contents

1. Introduction

1.1 What is Low-Level design document?

The Low-Level Design Document (LLDD) for the Swiggy Restaurant Dashboard outlines the internal logic design of the program code. It describes the class diagrams, methods, relations between classes, and program specifications. This document serves as a guide for programmers to directly code the program.

1.2 Scope

The Low-Level Design (LLD) process involves refining the component-level design step by step. It covers data structures, software architecture, source code, and performance algorithms. The data organization is defined during requirement analysis and refined during data design.

2. Architecture

Swiggy Restaurant Dashboard Architecture
The architecture of the Swiggy Restaurant Dashboard is designed to provide a user-friendly interface for restaurant owners and managers to analyze restaurant performance and manage orders.

The following diagram illustrates the architecture:

The dashboard system comprises the following components:

1. Frontend Application: Provides a user interface for interacting with the dashboard.
2. Backend Server: Handles data processing, storage, and communication with the frontend.
3. Database: Stores restaurant-related data, order history, and analytics.
4. User Authentication: Manages user access and authentication to ensure data security.
5. Analytics Engine: Processes data and generates insights for restaurant performance.
6. Deployment: Deploys the dashboard system to production servers.

3. Architecture Description

3.1 Data Description

The dataset includes restaurant-related information, order details, customer feedback, and menu items.
1. RestaurantID: Unique identifier for each restaurant.
2. RestaurantName: Name of the restaurant.
3. Location: Geographic location of the restaurant.
4. MenuItems: List of menu items offered by the restaurant.
5. OrderHistory: Historical data of orders placed by customers.
6. CustomerFeedback: Feedback and ratings provided by customers.

3.2 Data Collection

Data is collected from restaurant partners, customer orders, and feedback forms.
1. API Integration: Swiggy API is used to fetch real-time order and menu data.
2. Feedback Forms: Customer feedback is collected through forms in the frontend.

3.3 Data Transformation

Raw data is transformed and cleaned for analysis and storage.
1. Parsing: Raw API responses are parsed to extract relevant data.
2. Data Cleansing: Data inconsistencies are addressed to ensure accuracy.
3. Data Enrichment: Additional data, such as location details, is added to enhance analytics.

3.4 Data Storage

Restaurant and order data are stored in a relational database.
1. Tables: Restaurant, MenuItems, OrderHistory, CustomerFeedback.
2. Relationships: Establish relationships between tables for data integrity.

3.5 User Authentication

User authentication ensures secure access to the dashboard.
1. User Roles: Restaurant Owner, Manager, Admin.
2. Authentication Process: Users log in with credentials and receive access tokens.
3. Authorization: Users are granted permissions based on their roles.

3.6 Restaurant Analytics

Restaurant owners and managers can view various analytics:
1. Order Trends: Visualize order trends over time.
2. Menu Performance: Analyze the popularity of menu items.
3. Customer Satisfaction: Monitor feedback and ratings.
4. Location Insights: Explore performance by geographic location.

3.7 Deployment

The dashboard system is deployed to production servers.
1. Frontend: Hosted on a web server.
2. Backend: Deployed on a cloud-based server.
3. Database: Hosted on a reliable database server.

4. Unit Test Cases

TEST CASE DESCRIPTION EXPECTED RESULTS
Login Functionality

Test the login process with valid and invalid credentials.

Dashboard Interface

Verify that the dashboard interface displays correctly for different user roles.

Order Analytics

Check if order trends and analytics are displayed accurately.

Menu Performance

Ensure that menu performance analytics are generated correctly.

Data Storage

Test data insertion and retrieval from the database.

User Authentication

Verify that user roles and access control are enforced.