**High-Level Design Document: Swiggy**

**Table of Contents**

1. **Introduction**
   * Purpose of the Document
   * Overview of Swiggy
2. **System Architecture**
   * Client-Side
     + Consumer App (iOS, Android)
     + Restaurant Partner App
     + Delivery Partner App
   * Server-Side
     + Load Balancers
     + Web Servers
     + Application Servers
     + Database Servers
3. **Major Components and Modules** 3.1 **User Management Module**
   * Customer Registration and Login
   * Restaurant Partner Registration and Login
   * Delivery Partner Registration and Login

3.2 **Order Management Module**

* + Consumer Placing Orders
  + Restaurant Receiving and Processing Orders
  + Delivery Partner Assigning and Delivering Orders

3.3 **Menu Management Module**

* + Restaurant Uploading and Updating Menus
  + Customer Browsing Menus

3.4 **Payment Processing Module**

* + Handling Consumer Payments
  + Payouts to Restaurant Partners
  + Earnings for Delivery Partners

3.5 **Rating and Review Module**

* + Customers Rating and Reviewing Restaurants
  + Feedback for Delivery Partners

1. **Data Flow**
   * Consumer Placing Orders
   * Order Processing Flow
   * Payment Flow
   * Ratings and Reviews Flow
2. **Interfaces**
   * Consumer App Interface
   * Restaurant Partner Interface
   * Delivery Partner Interface
   * APIs for Communication
   * Payment Gateway Interfaces
3. **Technology Stack**
   * Consumer App: Swift (iOS), Kotlin (Android)
   * Restaurant and Delivery Partner Apps: React Native
   * Web Servers: Nginx, Apache
   * Application Servers: Node.js, Express.js
   * Database Servers: MySQL, MongoDB
4. **Constraints and Assumptions**
   * Assumption: Users have internet access for placing orders.
   * Constraint: Compliance with data protection and payment processing regulations.
5. **Introduction:**
   * **Purpose of the Document:** Describes why the document is created, typically to provide an understanding of the Swiggy system's high-level design.
   * **Overview of Swiggy:** Offers a brief introduction to the Swiggy platform.
6. **System Architecture:**
   * **Client-Side:**
     + **Consumer App (iOS, Android):** Describes the architecture of the mobile apps used by customers.
     + **Restaurant Partner App:** Explains the architecture of the mobile app for restaurant partners.
     + **Delivery Partner App:** Outlines the architecture of the mobile app for delivery partners.
   * **Server-Side:**
     + **Load Balancers:** Describes the role and function of load balancers.
     + **Web Servers:** Explains the purpose of web servers.
     + **Application Servers:** Describes the servers handling application logic.
     + **Database Servers:** Outlines the role of servers managing databases.
7. **Major Components and Modules:**
   * **User Management Module:**
     + **Customer Registration and Login:** Describes how customers register and log in.
     + **Restaurant Partner Registration and Login:** Explains the registration and login process for restaurant partners.
     + **Delivery Partner Registration and Login:** Outlines the registration and login process for delivery partners.
   * **Order Management Module:**
     + **Consumer Placing Orders:** Describes how consumers place orders.
     + **Restaurant Receiving and Processing Orders:** Explains how restaurants handle received orders.
     + **Delivery Partner Assigning and Delivering Orders:** Outlines the process of assigning and delivering orders.
   * **Menu Management Module:**
     + **Restaurant Uploading and Updating Menus:** Describes how restaurants manage their menus.
     + **Customer Browsing Menus:** Explains how customers browse available menus.
   * **Payment Processing Module:**
     + **Handling Consumer Payments:** Describes the process of handling payments from consumers.
     + **Payouts to Restaurant Partners:** Explains how payouts are managed for restaurant partners.
     + **Earnings for Delivery Partners:** Outlines the process of calculating and managing earnings for delivery partners.
   * **Rating and Review Module:**
     + **Customers Rating and Reviewing Restaurants:** Describes how customers provide feedback on restaurants.
     + **Feedback for Delivery Partners:** Explains the feedback process for delivery partners.
8. **Data Flow:**
   * **Consumer Placing Orders:** Illustrates the flow of data when a consumer places an order.
   * **Order Processing Flow:** Describes the flow of data during the processing of an order.
   * **Payment Flow:** Outlines the flow of data during the payment process.
   * **Ratings and Reviews Flow:** Illustrates the flow of data when customers provide ratings and reviews.
9. **Interfaces:**
   * **Consumer App Interface:** Describes the user interface for consumers.
   * **Restaurant Partner Interface:** Explains the user interface for restaurant partners.
   * **Delivery Partner Interface:** Outlines the user interface for delivery partners.
   * **APIs for Communication:** Describes the application programming interfaces (APIs) used for communication.
   * **Payment Gateway Interfaces:** Explains interfaces with payment gateways.
   * **Database Servers:** Lists the database server technologies (MySQL, MongoDB).

**Technology Stack:**

1. **Backend (Server-Side):**
   * **Java with Spring Framework:** Utilized for building the backend logic and handling server-side operations.
   * **Spring Boot:** Specifically, Spring Boot may be used to simplify the development of the Spring-based applications.
   * **Spring MVC (Model-View-Controller):** A part of the Spring framework used for creating scalable and flexible web applications.
2. **Database:**
   * **MySQL:** A relational database management system used for storing and managing data.
   * **Hibernate:** Potentially used as an Object-Relational Mapping (ORM) tool to simplify database interactions in the Java application.
3. **Frontend (Client-Side):**
   * *Note: While you mentioned Java full stack, the frontend technologies might vary. If you are using Java on the frontend, you might use JavaFX or other Java-based UI frameworks. Alternatively, if you're using other languages or frameworks on the frontend, you can specify them here.*
4. **Web Servers:**
   * **Apache Tomcat:** A popular web server that may be used to deploy and run Java web applications.
5. **Build Tools:**
   * **Maven or Gradle:** Build automation tools often used with Java projects.
6. **Version Control:**
   * **Git:** Used for version control to track changes in the source code.
7. **APIs for Communication:**
   * **RESTful APIs:** Communication between the frontend and backend is often facilitated using RESTful APIs.
8. **Testing:**
   * **JUnit:** A widely used testing framework for Java.
   * **Spring Test:** For testing Spring applications.
9. **IDE (Integrated Development Environment):**
   * **Eclipse or IntelliJ IDEA:** Common IDEs used for Java development.
10. **Deployment:**
    * **Docker:** Containerization tool that aids in deploying applications consistently across different environments.
    * **AWS, Azure, or other Cloud Services:** Cloud platforms for hosting and scaling the application.
11. **Constraints and Assumptions:**
    * **Assumption: Users have internet access for placing orders.**
    * **Constraint: Compliance with data protection and payment processing regulations.**

**Conclusion:**

Conclusion The above template provides a high-level overview of the Swiggy system, its architecture, major components, data flow, interfaces, technology stack, and constraints.