Crowd handle ordering system

System Design Description:

The system will provide public Wi-Fi access with the following functionality:

1. **Public Wi-Fi Access**:
    - Users can connect to a public Wi-Fi network.
    - Upon connecting, users are redirected to a web page that lists registered shops.
2. **Shop Registration**:
    - Shop vendors can register themselves on the platform.
    - Vendors also connect to the same Wi-Fi network to manage their shop and orders.
3. **User Interaction**:
    - Users can browse the list of registered shops displayed on the web page.
    - Users can place orders for items from the listed shops directly through the web page.
4. **Order Management for Shopkeepers**:
    - Shopkeepers can view a list of all orders placed by users.
    - They can accept orders and update the order status in real time.
    - Shopkeepers can update order stages (e.g., "Order Accepted," "Preparing," "Ready for Pickup," etc.).
5. **User Notifications**:
    - Users receive real-time notifications about their order status on their mobile devices.
    - Once the order is ready, users are notified with an acknowledgment message.

System Design Overview

The system will be a web-based application integrated with public Wi-Fi, consisting of the following major components:

1. **Public Wi-Fi Gateway**
2. **Web Application**
3. **Backend System**
4. **Database**
5. **Notifications System**
6. **Order Management System**

Architecture

The system will follow a **3-tier architecture**:

1. **Presentation Layer**:
    - User Interface (Web Page) accessible via public Wi-Fi.

- Shopkeeper's dashboard for managing orders.

2. **Application Layer**:
   - Backend APIs for managing users, shops, and orders.
   - Order management and status tracking.
3. **Data Layer**:
   - Databases to store user, shop, and order information.

System Components

1. Public Wi-Fi Gateway

- **Functionality**: Redirects all connected users to the main application web page.
- **Implementation**: Use a captive portal to redirect users to a centralized URL after connecting to the Wi-Fi.
- **Technology**:
  - Wi-Fi Router with Captive Portal Support (e.g., MikroTik, Cisco).
  - Integration with the backend system to retrieve shop data dynamically.

2. Web Application

- **For Users**:
  - Display a list of registered shops.
  - Allow users to browse shop menus/items.
  - Enable users to place orders.
- **For Shopkeepers**:
  - Dashboard to view and manage orders.
  - Order acceptance, status updates, and notifications.
- **Frontend Technology**:
  - **Frameworks**: React.js, Vue.js, or Angular.
  - **Responsive Design**: Optimized for mobile and desktop.

3. Backend System

- **APIs**:
  - Authentication: Handles user and shopkeeper authentication.
  - Shop Management: Manages registered shops and their menus.
  - Order Management: Handles order placement, status updates, and notifications.
- **Technology Stack**:
  - **Programming Language**: Node.js, Python (Django/Flask), or Java (Spring Boot).
  - **API Framework**: RESTful APIs or GraphQL.
  - **Authentication**: OAuth 2.0 or JWT (JSON Web Token).

4. Database

- **Tables**:
  - **Users Table**: Stores user profiles and preferences.
  - **Shops Table**: Stores shop details, menus, and availability status.
  - **Orders Table**: Tracks orders with their status, timestamps, and user/shop references.
  - **Notifications Table**: Manages pending and sent notifications.
- **Database Options**:
  - **Relational Database**: PostgreSQL, MySQL (for structured data).
  - **NoSQL Database**: MongoDB (for storing menu items and logs).

5. Notifications System

- **User Notifications**:
  - Sends real-time notifications to users about order status (e.g., accepted, in preparation, ready).
- **Technology Options**:
  - **Push Notifications**: Firebase Cloud Messaging (FCM) or OneSignal for mobile notifications.
  - **Real-Time Updates**: WebSockets or Server-Sent Events (SSE).

6. Order Management System

- **Features**:
  - Allows shopkeepers to view incoming orders in real time.
  - Provides shopkeepers the ability to update order stages (e.g., "Accepted," "Preparing," "Ready").
  - Tracks order history for both users and shopkeepers.
- **Order Status Workflow**:
  - **Order Placed → Order Accepted → In Preparation → Ready for Pickup → Order Completed**
  - Notifications are sent to users at each stage.
- **Implementation**:
  - Backend services with event-driven architecture (e.g., using Apache Kafka or RabbitMQ).

Workflow

1. User Flow:

1. User connects to the public Wi-Fi.
2. User is redirected to a captive portal with the web application.
3. User browses the list of registered shops and places an order.
4. User receives real-time notifications about the status of their order.
5. Once the order is ready, the user is notified for pickup.

2. Shopkeeper Flow:

1. Shopkeeper connects to the same Wi-Fi network and logs into their dashboard.
2. Shopkeeper views and manages incoming orders.
3. Shopkeeper updates the order status at various stages.
4. Notifications are sent to users in real time.

8. Challenges and Solutions

Challenge 1: Handling Real-Time Updates

- **Solution**: Use WebSockets for instant notifications or long-polling fallback.

Challenge 2: Scalability of Wi-Fi and Application

- **Solution**: Use load balancers and a distributed database system.

Challenge 3: Ensuring Secure Access

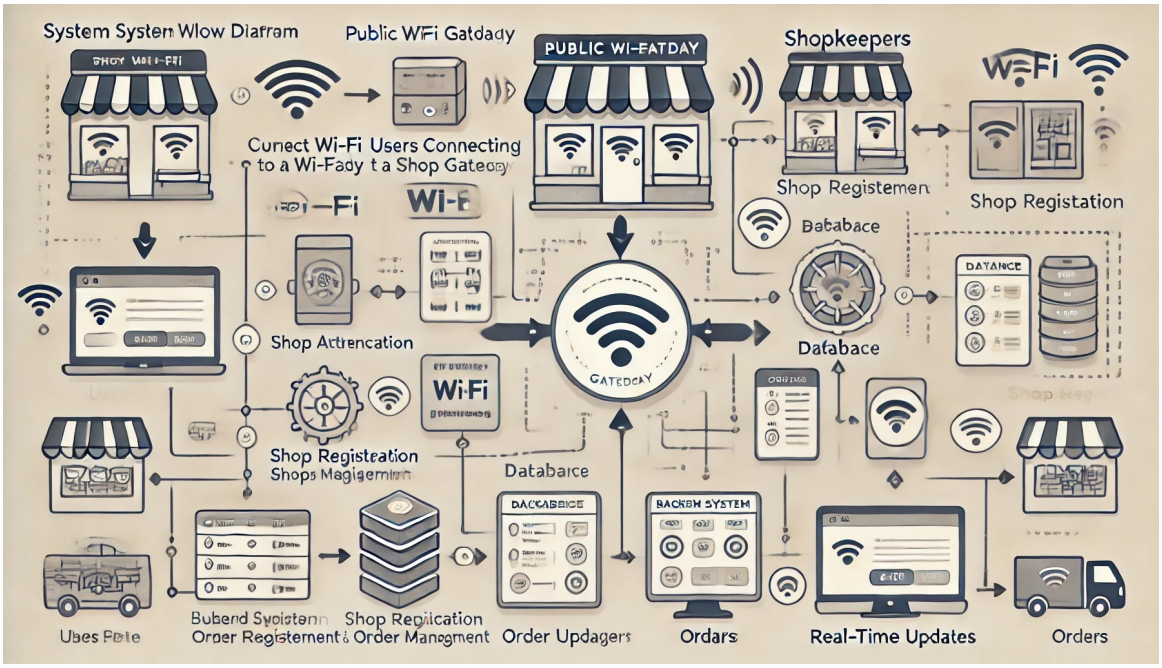
- **Solution**: Encrypt all communications (SSL/TLS), use role-based access control.

Technological Stack

Layer	Technology/Tool
Frontend	React.js, Vue.js, or Angular
Backend	Node.js, Django, or Spring Boot
Database	PostgreSQL/MySQL + MongoDB
Wi-Fi Gateway	MikroTik, OpenWrt Captive Portal
Notifications	Firebase Cloud Messaging (FCM), WebSockets
Hosting	AWS, Azure, or Google Cloud

User Device --> Wi-Fi Gateway --> Web Application
|
--> Backend System --> Database
--> Notification System
Shopkeeper Device --> Wi-Fi Gateway --> Order Management Dashboard

Reference diagram:



Tables Schema:

1. **Users Table** (Stores user and shopkeeper information)

```sql
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15) UNIQUE,
    password_hash TEXT NOT NULL,
    role ENUM('user', 'shopkeeper') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Shops Table (Stores shop details and registered shopkeepers)

```sql
CREATE TABLE shops (
    shop_id SERIAL PRIMARY KEY,
    shop_name VARCHAR(150) NOT NULL,
    owner_id INT NOT NULL,
    location VARCHAR(255),
    contact VARCHAR(15),
    wifi_network VARCHAR(100), -- To validate shopkeeper's connection to Wi-Fi
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (owner_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

3. Menu Items Table (Stores items available in a shop)

```sql
CREATE TABLE menu_items (
    item_id SERIAL PRIMARY KEY,
    shop_id INT NOT NULL,
    item_name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    availability BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (shop_id) REFERENCES shops(shop_id) ON DELETE CASCADE
);
```

4. Orders Table (Stores user orders and their statuses)

```sql
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    shop_id INT NOT NULL,
    total_price DECIMAL(10,2) NOT NULL,
    status ENUM('pending', 'accepted', 'preparing', 'ready', 'completed', 'cancelled'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (shop_id) REFERENCES shops(shop_id) ON DELETE CASCADE
);
```

5. Order Items Table (Stores items included in an order)

```sql
CREATE TABLE order_items (
    order_item_id SERIAL PRIMARY KEY,
    order_id INT NOT NULL,
    item_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (item_id) REFERENCES menu_items(item_id) ON DELETE CASCADE
);
```

6. Notifications Table (Stores notifications sent to users)

```sql
CREATE TABLE notifications (
    notification_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    order_id INT,
    message TEXT NOT NULL,
    is_read BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE
);
```

7. Wi-Fi Sessions Table (Tracks user connections to Wi-Fi)

```sql
CREATE TABLE wifi_sessions (
    session_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    ip_address VARCHAR(50),
    mac_address VARCHAR(50),
    connected_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    disconnected_at TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

8. Payments Table (Optional: For handling online transactions)

```sql
CREATE TABLE payments (
    payment_id SERIAL PRIMARY KEY,
    order_id INT NOT NULL,
    user_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    payment_method ENUM('cash', 'credit_card', 'upi', 'wallet') NOT NULL,
    status ENUM('pending', 'successful', 'failed') DEFAULT 'pending',
    transaction_id VARCHAR(100) UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

Relationships Between Tables

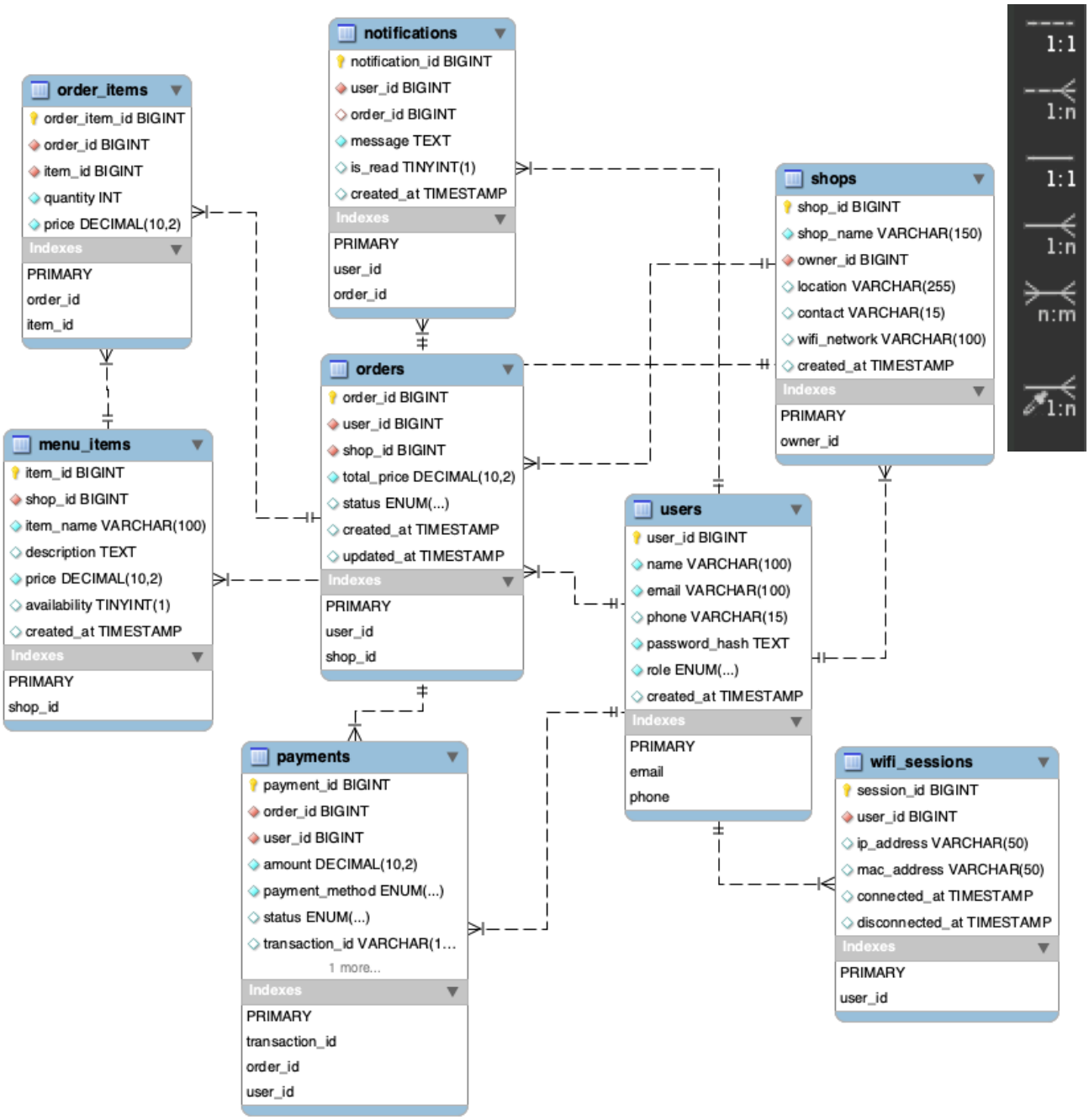
- **Users & Shops** → A shopkeeper (user) owns a shop (`users → shops` via `owner_id`).
- **Shops & Menu Items** → A shop has multiple menu items (`shops → menu_items` via `shop_id`).
- **Users & Orders** → A user can place multiple orders (`users → orders` via `user_id`).
- **Orders & Order Items** → Each order contains multiple items (`orders → order_items`).
- **Orders & Notifications** → Notifications are sent for order status updates (`orders → notifications`).

Table Create Query

Table Name	Query
Users	CREATE TABLE users (user_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100) UNIQUE NOT NULL, phone VARCHAR(15) UNIQUE, password_hash TEXT NOT NULL, role ENUM('user', 'shopkeeper') NOT NULL, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
Shops	CREATE TABLE shops (shop_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, shop_name VARCHAR(150) NOT NULL, owner_id BIGINT UNSIGNED NOT NULL, location VARCHAR(255), contact VARCHAR(15), wifi_network VARCHAR(100), -- To validate shopkeeper's connection to Wi-Fi created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (owner_id) REFERENCES users(user_id) ON DELETE CASCADE);
menu_items	CREATE TABLE menu_items (Item_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, shop_id BIGINT UNSIGNED NOT NULL, item_name VARCHAR(100) NOT NULL, description TEXT, price DECIMAL(10,2) NOT NULL, availability BOOLEAN DEFAULT TRUE, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (shop_id) REFERENCES shops(shop_id) ON DELETE CASCADE);

Table Name	Query
wifi_sessions	CREATE TABLE wifi_sessions (session_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, user_id BIGINT UNSIGNED NOT NULL, ip_address VARCHAR(50), mac_address VARCHAR(50), connected_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, disconnected_at TIMESTAMP, FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE);
orders	CREATE TABLE orders (order_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, user_id BIGINT UNSIGNED NOT NULL, shop_id BIGINT UNSIGNED NOT NULL, total_price DECIMAL(10,2) NOT NULL, status ENUM('pending', 'accepted', 'preparing', 'ready', 'completed', 'cancelled') DEFAULT 'pending', created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP, FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE, FOREIGN KEY (shop_id) REFERENCES shops(shop_id) ON DELETE CASCADE);
order_items	CREATE TABLE order_items (order_item_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, order_id BIGINT UNSIGNED NOT NULL, item_id BIGINT UNSIGNED NOT NULL, quantity INT NOT NULL, price DECIMAL(10,2) NOT NULL, FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE, FOREIGN KEY (item_id) REFERENCES menu_items(item_id) ON DELETE CASCADE);
notifications	CREATE TABLE notifications (notification_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, user_id BIGINT UNSIGNED NOT NULL, order_id BIGINT UNSIGNED, message TEXT NOT NULL, is_read BOOLEAN DEFAULT FALSE, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE, FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE);
payments	CREATE TABLE payments (payment_id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, order_id BIGINT UNSIGNED NOT NULL, user_id BIGINT UNSIGNED NOT NULL, amount DECIMAL(10,2) NOT NULL, payment_method ENUM('cash', 'credit_card', 'upi', 'wallet') NOT NULL, status ENUM('pending', 'successful', 'failed') DEFAULT 'pending', transaction_id VARCHAR(100) UNIQUE, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE, FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE);

ER Diagram


notifications
notification_id BIGINT
user_id BIGINT
order_id BIGINT
message TEXT
is_read TINYINT(1)
created_at TIMESTAMP
Indexes
PRIMARY
user_id
order_id
order_items
order_item_id BIGINT
order_id BIGINT
item_id BIGINT
quantity INT
price DECIMAL(10,2)
Indexes
PRIMARY
order_id
item_id
shops
shop_id BIGINT
shop_name VARCHAR(150)
owner_id BIGINT
location VARCHAR(255)
contact VARCHAR(15)
wifi_network VARCHAR(100)
created_at TIMESTAMP
Indexes
PRIMARY
owner_id
orders
order_id BIGINT
user_id BIGINT
shop_id BIGINT
total_price DECIMAL(10,2)
status ENUM(...)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
PRIMARY
user_id
shop_id
menu_items
item_id BIGINT
shop_id BIGINT
item_name VARCHAR(100)
description TEXT
price DECIMAL(10,2)
availability TINYINT(1)
created_at TIMESTAMP
Indexes
PRIMARY
shop_id
users
user_id BIGINT
name VARCHAR(100)
email VARCHAR(100)
phone VARCHAR(15)
password_hash TEXT
role ENUM(...)
created_at TIMESTAMP
Indexes
PRIMARY
email
phone
payments
payment_id BIGINT
order_id BIGINT
user_id BIGINT
amount DECIMAL(10,2)
payment_method ENUM(...)
status ENUM(...)
transaction_id VARCHAR(1...
1 more...
Indexes
PRIMARY
transaction_id
order_id
user_id
wifi_sessions
session_id BIGINT
user_id BIGINT
ip_address VARCHAR(50)
mac_address VARCHAR(50)
connected_at TIMESTAMP
disconnected_at TIMESTAMP
Indexes
PRIMARY
user_id
1:1
1:n
1:1
1:n
n:m
1:n