**Database Design Strategy Based on UML Diagram**

**✅ General Strategy**

We will adopt a **normalized relational database strategy**, based on:

1. Creating **tables for each class with attributes**
2. Defining **primary keys (PK)** and **foreign keys (FK)** for relationships
3. Using **junction tables for N:N relationships**
4. Handling inheritance (e.g., ServiceProvider, ServiceSeeker, Admin) using the **Table per Class** approach

**📊 Main Tables (Suggested Structure)**

**🔐 User**

User (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

email VARCHAR(100) UNIQUE,

password VARCHAR(255),

phone VARCHAR(20),

role ENUM('ServiceProvider', 'ServiceSeeker', 'Admin')

)

**🧑 ServiceProvider and ServiceSeeker**

ServiceProvider (

provider\_id INT PRIMARY KEY,

user\_id INT UNIQUE,

FOREIGN KEY (user\_id) REFERENCES User(user\_id)

)

ServiceSeeker (

seeker\_id INT PRIMARY KEY,

user\_id INT UNIQUE,

FOREIGN KEY (user\_id) REFERENCES User(user\_id)

)

**🛠️ Service**

Service (

service\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100),

description TEXT,

category\_id INT,

FOREIGN KEY (category\_id) REFERENCES Category(category\_id)

)

**🗂️ Category**

Category (

category\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100)

)

**🔄 Provider\_Service (N:N relationship)**

Provider\_Service (

provider\_id INT,

service\_id INT,

PRIMARY KEY (provider\_id, service\_id),

FOREIGN KEY (provider\_id) REFERENCES ServiceProvider(provider\_id),

FOREIGN KEY (service\_id) REFERENCES Service(service\_id)

)

**📟 ServiceRequest**

ServiceRequest (

request\_id INT PRIMARY KEY AUTO\_INCREMENT,

seeker\_id INT,

provider\_id INT,

service\_id INT,

request\_date DATETIME,

status ENUM('pending', 'accepted', 'completed', 'cancelled'),

FOREIGN KEY (seeker\_id) REFERENCES ServiceSeeker(seeker\_id),

FOREIGN KEY (provider\_id) REFERENCES ServiceProvider(provider\_id),

FOREIGN KEY (service\_id) REFERENCES Service(service\_id)

)

**💳 Payment**

Payment (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

request\_id INT,

amount DECIMAL(10,2),

payment\_date DATETIME,

status ENUM('paid', 'pending', 'failed'),

FOREIGN KEY (request\_id) REFERENCES ServiceRequest(request\_id)

)

**✨ Review and Rating**

Review (

review\_id INT PRIMARY KEY AUTO\_INCREMENT,

request\_id INT,

comment TEXT,

FOREIGN KEY (request\_id) REFERENCES ServiceRequest(request\_id)

)

Rating (

rating\_id INT PRIMARY KEY AUTO\_INCREMENT,

request\_id INT,

rating INT CHECK (rating BETWEEN 1 AND 5),

FOREIGN KEY (request\_id) REFERENCES ServiceRequest(request\_id)

)

**❤️ Favorite**

Favorite (

seeker\_id INT,

provider\_id INT,

PRIMARY KEY (seeker\_id, provider\_id),

FOREIGN KEY (seeker\_id) REFERENCES ServiceSeeker(seeker\_id),

FOREIGN KEY (provider\_id) REFERENCES ServiceProvider(provider\_id)

)

**🔍 SearchFilter and MultilingualSupport**

SearchFilter (

filter\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

filter\_criteria TEXT,

FOREIGN KEY (user\_id) REFERENCES User(user\_id)

)

MultilingualSupport (

key\_name VARCHAR(100) PRIMARY KEY,

lang\_code VARCHAR(10),

translated\_text TEXT

)

**👨‍💼 Admin**

Admin (

admin\_id INT PRIMARY KEY,

user\_id INT UNIQUE,

FOREIGN KEY (user\_id) REFERENCES User(user\_id)

)

**🖥️ AdminPanel**

AdminPanel (

setting\_id INT PRIMARY KEY AUTO\_INCREMENT,

admin\_id INT,

setting\_key VARCHAR(100),

setting\_value TEXT,

FOREIGN KEY (admin\_id) REFERENCES Admin(admin\_id)

)

**🔗 Relationship Summary**

* User is the base for ServiceSeeker, ServiceProvider, and Admin
* ServiceProvider ⇄ Service: many-to-many
* ServiceRequest links Seeker, Provider, and Service
* Payment, Review, Rating all depend on ServiceRequest
* Favorite: many-to-many between Seeker and Provider

Let me know when you're ready to generate the SQL scripts based on this model!