



# Database Design

## Sany3y

Version 1.0

*Nov of 2025*

# Table of Contents

<b>Team Members .....</b>	<b>3</b>
<b>Introduction .....</b>	<b>3</b>
<b>Database Objectives.....</b>	<b>3</b>
<b>Entities and Attributes.....</b>	<b>4</b>
<b>Relationships.....</b>	<b>6</b>
<b>ERD.....</b>	<b>7</b>
<b>Summary .....</b>	<b>7</b>

## Team Members

ID	Name	Email
21030343	Mohammed Atef	<a href="mailto:mohamed45452020@gmail.com">mohamed45452020@gmail.com</a>
21060536	Steven Amin	<a href="mailto:stevenameen02@gmail.com">stevenameen02@gmail.com</a>
21040560	Omnia Ashraf	<a href="mailto:omniaamohamed102@gmail.com">omniaamohamed102@gmail.com</a>
21056226	Kenzi Shibli	<a href="mailto:kebzishibl@gmail.com">kebzishibl@gmail.com</a>

## Introduction

This document outlines the database design for the Sany3y full-stack website. It defines entities, relationships, constraints, and data structures required to support the platform's operations.

## Database Objectives

- Ensure data consistency and integrity.
- Support user, technician, booking, and admin operations.
- Provide efficient querying for dashboard analytics.
- Enable scalable growth for future platform expansion.

## Entities and Attributes

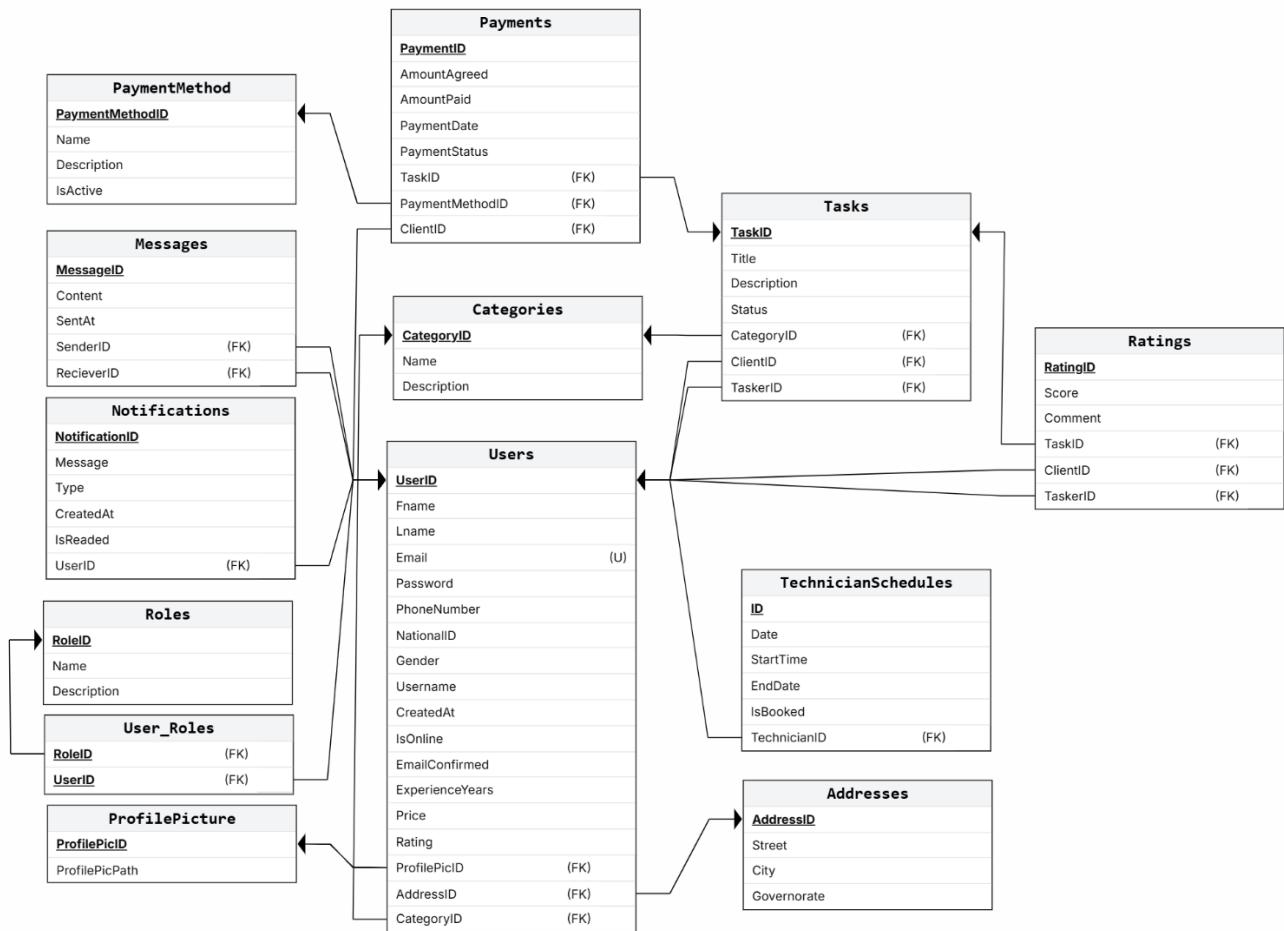
Entity	Description	Key Attributes
<b>ProfilePicture</b>	Stores the paths of users' profile pictures	<ul style="list-style-type: none"> <li>• ProfilePicID (PK)</li> <li>• ProfilePicPath</li> </ul>
<b>Addresses</b>	Stores address details (street, city, governorate)	<ul style="list-style-type: none"> <li>• AddressID (PK)</li> <li>• Street</li> <li>• City</li> <li>• Governorate</li> </ul>
<b>Categories</b>	Represents the categories of services or skills in the system	<ul style="list-style-type: none"> <li>• CategoryID (PK)</li> <li>• Name</li> <li>• Description</li> </ul>
<b>Users</b>	Stores information about system users (clients/technicians)	<ul style="list-style-type: none"> <li>• UserID (PK)</li> <li>• Email (Unique)</li> <li>• Fname</li> <li>• Lname</li> <li>• Password</li> <li>• PhoneNumber</li> <li>• NationalID</li> <li>• Gender</li> <li>• Username</li> <li>• CreatedAt</li> <li>• ProfilePicID (FK)</li> <li>• AddressID (FK)</li> <li>• CategoryID (FK)</li> </ul>
<b>Tasks</b>	Tasks created between clients and technicians	<ul style="list-style-type: none"> <li>• TaskID (PK)</li> <li>• Title</li> <li>• Description</li> <li>• Status</li> <li>• CategoryID (FK)</li> <li>• ClientID (FK)</li> <li>• TaskerID (FK)</li> </ul>
<b>Notifications</b>	Notifications sent to users	<ul style="list-style-type: none"> <li>• NotificationID (PK)</li> <li>• Message</li> <li>• Type</li> <li>• CreatedAt</li> <li>• IsReaded</li> <li>• UserID (FK)</li> </ul>

<b>Messages</b>	Messages exchanged between clients and technicians	<ul style="list-style-type: none"> <li>• MessageID (PK)</li> <li>• Content</li> <li>• SentAt</li> <li>• SenderID (FK)</li> <li>• ReceiverID (FK)</li> </ul>
<b>Ratings</b>	Ratings given by clients to technicians after completing tasks	<ul style="list-style-type: none"> <li>• RatingID (PK)</li> <li>• Score</li> <li>• Comment</li> <li>• TaskID (FK)</li> <li>• ClientID (FK)</li> <li>• TaskerID (FK)</li> </ul>
<b>PaymentMethod</b>	Available payment methods in the system	<ul style="list-style-type: none"> <li>• PaymentMethodID (PK)</li> <li>• Name</li> <li>• Description</li> <li>• IsActive</li> </ul>
<b>Payments</b>	Records payments related to tasks	<ul style="list-style-type: none"> <li>• PaymentID (PK)</li> <li>• AmountAgreed</li> <li>• AmountPaid</li> <li>• PaymentDate</li> <li>• PaymentStatus</li> <li>• TaskID (FK)</li> <li>• PaymentMethodID (FK)</li> <li>• ClientID (FK)</li> </ul>
<b>Roles</b>	User roles (Admin / Client / Technician)	<ul style="list-style-type: none"> <li>• RoleID (PK)</li> <li>• Name</li> <li>• Description</li> </ul>
<b>User_Roles</b>	Many-to-many relationship between users and roles	<ul style="list-style-type: none"> <li>• RoleID + UserID (Composite PK)</li> <li>• RoleID (FK)</li> <li>• UserID (FK)</li> </ul>
<b>TechnicianSchedules</b>	Schedules for technicians showing availability	<ul style="list-style-type: none"> <li>• ID (PK)</li> <li>• Date</li> <li>• StartTime</li> <li>• EndDate</li> <li>• IsBooked</li> <li>• TechnicianID (FK)</li> </ul>

## Relationships

Relationship	Type	Description
<b>ProfilePicture → Users</b>	One-to-Many	Each user has one profile picture, but a profile picture could theoretically be linked to multiple users (depending on your design, usually 1-to-1).
<b>Addresses → Users</b>	One-to-Many	Each user has one address, but an address can belong to multiple users.
<b>Categories → Users</b>	One-to-Many	Each user belongs to one category (e.g., skill), each category can have many users.
<b>Categories → Tasks</b>	One-to-Many	Each task belongs to a category; a category can have many tasks.
<b>Users → Tasks (Client)</b>	One-to-Many	Each client can create many tasks; each task has one client.
<b>Users → Tasks (Tasker/Technician)</b>	One-to-Many	Each technician can be assigned to many tasks; each task has one technician.
<b>Users → Notifications</b>	One-to-Many	Each user can receive many notifications.
<b>Users → Messages (Sender)</b>	One-to-Many	Each user can send many messages.
<b>Users → Messages (Receiver)</b>	One-to-Many	Each user can receive many messages.
<b>Tasks → Ratings</b>	One-to-Many	Each task can have ratings.
<b>Users → Ratings (Client)</b>	One-to-Many	Each client can rate multiple tasks.
<b>Users → Ratings (Tasker)</b>	One-to-Many	Each technician can be rated multiple times.
<b>PaymentMethod → Payments</b>	One-to-Many	Each payment method can be used for many payments.
<b>Tasks → Payments</b>	One-to-Many	Each task can have many payments.
<b>Users → Payments (Client)</b>	One-to-Many	Each client can make many payments.
<b>Roles → User_Roles</b>	One-to-Many	Each role can be assigned to many users.
<b>Users → User_Roles</b>	One-to-Many	Each user can have multiple roles.
<b>Users → TechnicianSchedules</b>	One-to-Many	Each technician can have multiple schedule entries.

# ERD



## Summary

This database design supports the core functionalities of the Sany3y platform, ensures data integrity, allows scalability, and maintains efficient query performance.