# GYM MANAGEMENT SYSTEM

Project Report

Group 03
Senuth Wickramasinghe
Yogesh Gurung
Ayush Subedi

# Table of Contents

# Executive Summary

The Gym Management System (GMS) is designed to automate and streamline daily operations in a gym environment. It allows members to register, subscribe to weekly or monthly membership plans, and optionally add services such as diet plans, workout routines, and meal preparation guidance. For administrators, the system provides efficient management of user accounts, plans, services, and payments.
All data is stored in a secure, well-structured relational database, ensuring accuracy, reliability, and scalability. By automating subscription status updates, billing, and reporting, the GMS reduces manual workload, prevents errors, and enhances communication between gyms and their members. This results in improved customer experience and smoother operations for gym staff.

# Design Analysis Process

## Stakeholders and Their Needs

- **Gym Members:** Require storage of personal details, membership subscriptions, optional services, and payment history.

- **Staff (Administrators and Trainers)** Require backend access to user accounts, plan/service configurations, subscription tracking, and financial reporting.

## User Stories

**Gym Members (Primary Users)**

1. As a gym member, I want to subscribe to a membership plan and add optional services so that I can access gym facilities and reach my fitness goals.
2. As a gym member, I want to view my payment history and subscription details so that I can track my expenses and confirm my membership status.
3. As a gym member, I want to book or cancel fitness classes so that I can manage my schedule conveniently

**Staff (Administrators and Trainers)**

1. As an administrator, I want to manage user accounts by adding, editing, or deactivating them so that member information stays accurate and up to date.
2. As an administrator, I want to configure membership plans and optional services so that the gym can offer flexible options to members.
3. As an administrator, I want to generate reports on payments and subscriptions so that I can analyze revenue and monitor gym performance.

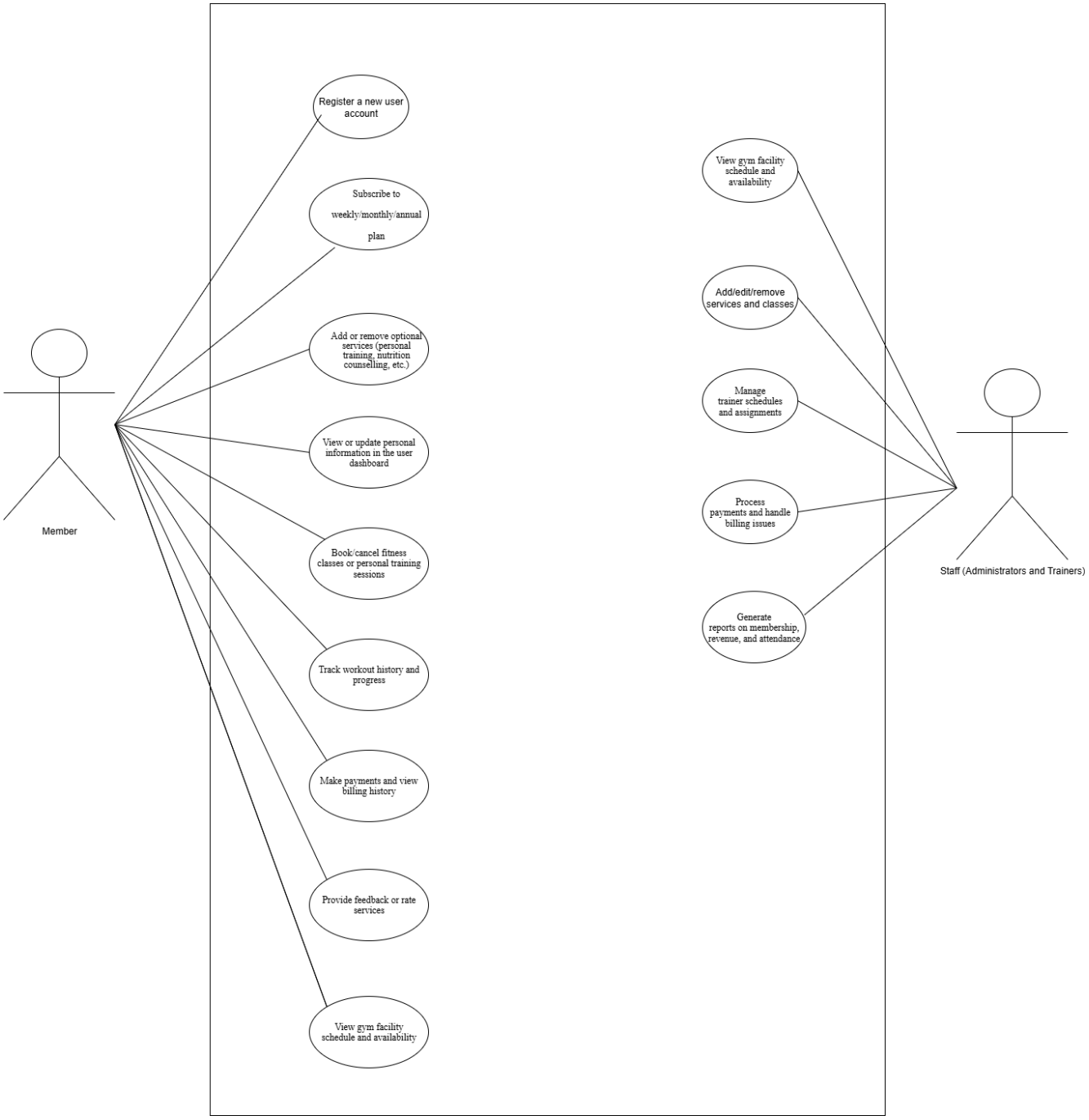
## Use Cases & Diagram

**Member Use Cases (Primary Users)**

1. **Register a new user account** – A new user creates a personal account by entering details such as name, email, and password.
2. **Subscribe to a weekly/monthly/annual plan** – A member chooses a subscription package (Normal or Premium) with defined validity periods.
3. **Add or remove optional services** – Members customize their subscriptions with add-ons such as personal training or nutrition counseling.
4. **View or update personal information in the user dashboard** – Members manage their personal profile and preferences.
5. **Book/cancel fitness classes or personal training sessions** – Members reserve or cancel slots for gym activities or personal coaching.
6. **Track workout history and progress** – Members view attendance, activity records, and progress logs stored in the system.
7. **Make payments and view billing history** – Members process subscription payments and check past transactions.
8. **Provide feedback or rate services** – Members share experiences and rate trainers, classes, or overall services.
9. **View gym facility schedule and availability** – Members check schedules for classes, trainers, and available gym facilities.

**Staff Use Cases (Trainers / Administrators)**

1. **Add/edit/remove services and classes** – Administrators manage the available fitness services and optional packages.
2. **Manage trainer schedules and assignments** – Admins allocate trainers to classes or personal sessions.
3. **Process payments and handle billing issues** – Admins verify payments, resolve disputes, and manage financial records.
4. **Generate reports on membership, revenue, and attendance** – The system provides analytical reports for management decisions.
5. **View gym facility schedule and availability** – Administrators ensure the timetable is accurate and facilities are optimally utilized.


Register a new user account
Subscribe to weekly/monthly/annual plan
Add or remove optional services (personal training, nutrition counselling, etc.)
View or update personal information in the user dashboard
Book/cancel fitness classes or personal training sessions
Track workout history and progress
Make payments and view billing history
Provide feedback or rate services
View gym facility schedule and availability
View gym facility schedule and availability
Add/edit/remove services and classes
Manage trainer schedules and assignments
Process payments and handle billing issues
Generate reports on membership, revenue, and attendance
Member
Staff (Administrators and Trainers)

## System Development Life Cycle (SDLC) Approach

For the development of the Gym Management System, our team adopted the **Iterative Model** of the System Development Life Cycle (SDLC). This model was chosen because it allows development to proceed in cycles, where the system is built and improved step by step. Each cycle involves planning, designing, developing, testing, and evaluating before moving into the next iteration.

### Why We Chose the Iterative Model

1. **Flexibility in Requirements:** At the start of the project, not all requirements were fully defined. The iterative model allowed us to gradually refine features, such as optional services and reporting, as the project evolved.
2. **Early Prototyping:** By building early versions of the system (e.g., focusing first on registration and membership plans), we were able to visualize functionality early and validate our approach with the team.
3. **Risk Management:** Breaking the project into iterations reduced the risk of major failures, since errors could be detected and fixed in early stages before affecting the entire system.
4. **Stakeholder Feedback:** The model encouraged feedback from the group after each cycle, helping us adjust system features and database entities to meet practical needs.

### How It Worked in This Project

- **First Iteration:** Focused on identifying actors, entities, and core membership functionalities. A brainstorming event was held to finalize ERD tables and user roles.
- **Second Iteration:** Expanded the design to include optional services (diet/workout plans), payment processing, and subscription tracking. The ERD and data dictionary were refined during this stage.
- **Third Iteration:** Addressed administrative functions such as reporting, data management, and inventory. We added cases for administrators and tested how they integrate with the database.
- **Final Iteration:** Combined all modules into a complete system design. The proposal document was updated, and diagrams were created to ensure clarity and alignment with project goals.

# Functional and Non-Functional Requirements

## Functional:

**User Registration & Login**

1. Members must be able to create an account with personal details.
2. Secure login with username/email and password.

**Membership Management**

1. Ability to subscribe to weekly, monthly, or annual plans.
2. Option to upgrade, downgrade, or cancel subscriptions.

**Service Management**

1. Add/remove optional services (personal training, diet plan, etc.).
2. Allow members to book or cancel classes.

**Payment & Billing**

1. Process payments (cash, card, or online).
2. Generate and store payment receipts.
3. Show billing history to members and staffs.

**Class & Schedule Management**

1. Manage available gym classes, trainers, and time slots.
2. Allow staff to manage schedules and assign trainers.

## Non-Functional:

**Performance**

1. The system must support at least 500+ concurrent users.
2. All queries should return results within 2 seconds.

**Security**

1. Passwords must be encrypted (e.g., SHA-256).
2. Secure login with session timeouts.
3. Role-based access (admins vs members).

**Availability & Reliability**

1. System uptime should be at least 99%.
2. Automatic backup of database daily.

**Usability**

1. Simple and user-friendly interface.
2. Accessible on desktop and mobile devices.

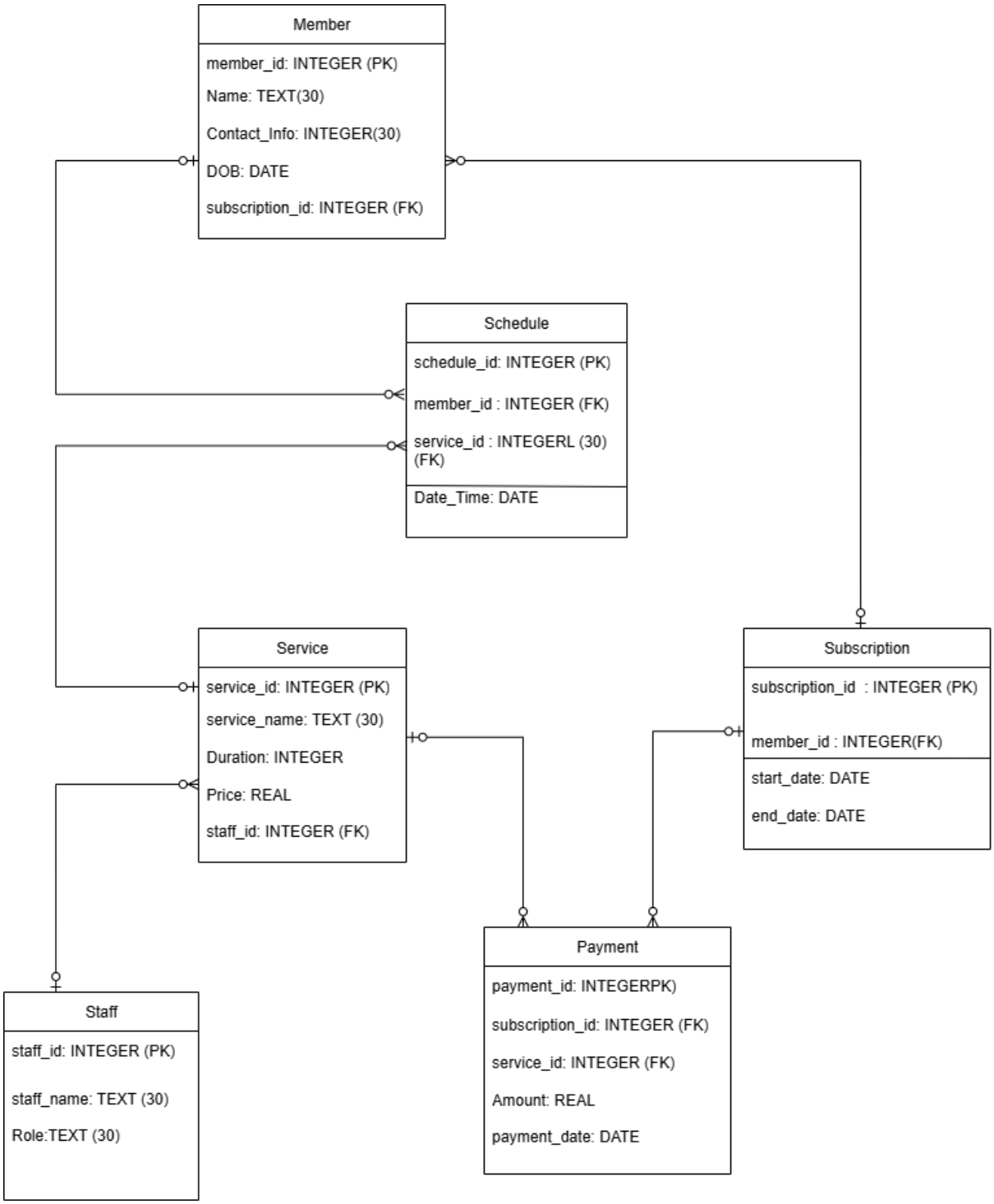
**Scalability**

1. The system should be able to expand to support multiple gyms/branches in the future.

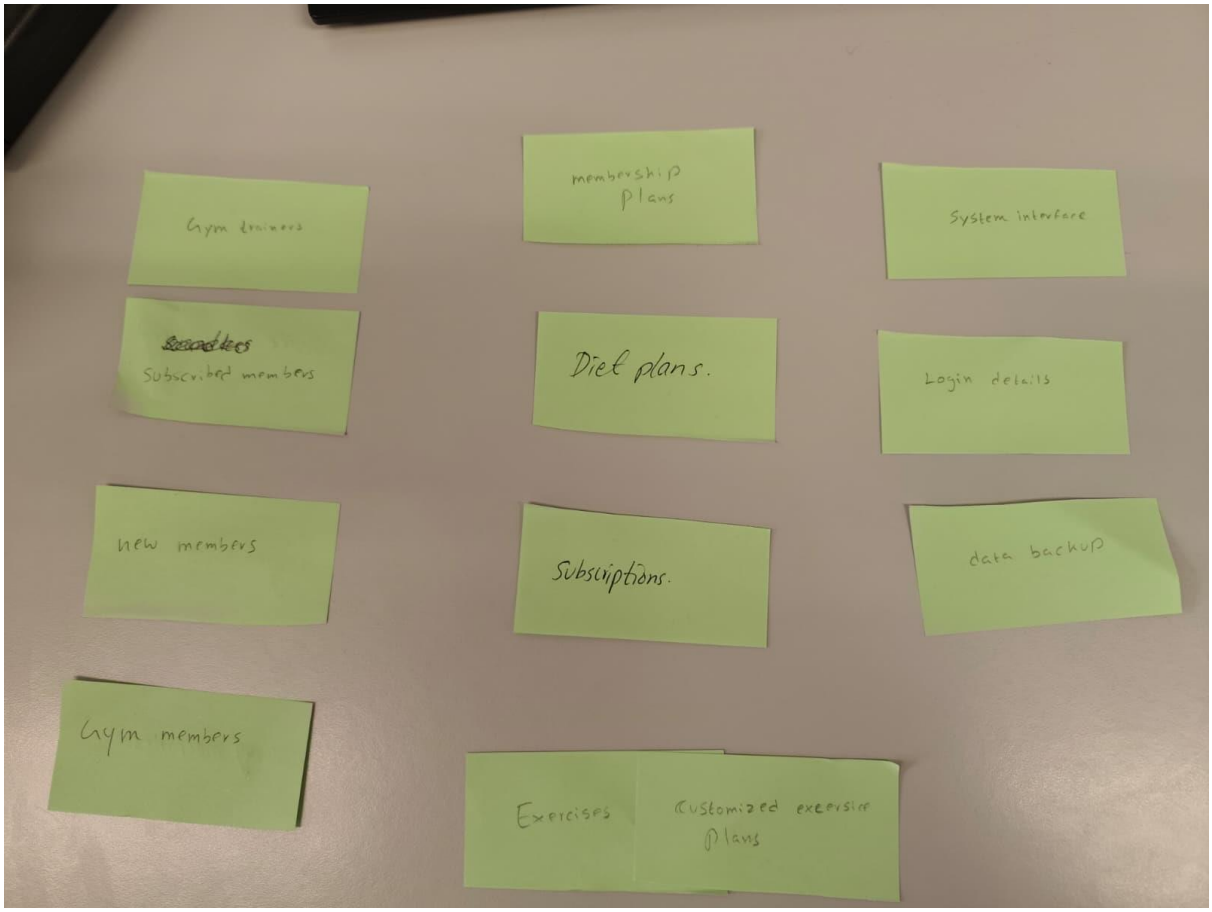
**Portability**

1. The system should work on different operating systems (Windows, Linux, Mac).

# Entity Relationship Diagram



# Brainstorming Evidence




As part of the design analysis process, our team conducted a brainstorming event to identify all possible entities and components of the Gym Management System. The session aimed to explore every element that could interact with or exist within the system before narrowing them down into finalized actors and database tables.

During this activity, we categorized the identified items into three main groups:

1. **Actors** – These are the human users or stakeholders who directly interact with the system. Examples identified included gym members, administrators, and trainers/nutritionists.
2. **Services** – These represent the business functions or features offered within the system, such as membership plans, payment processing, add-on services (diet plans, workout routines, etc.), and class scheduling.
3. **Technical Entities** – These are the system-related elements required for backend operations, including user accounts, authentication, subscriptions, and database records.

# Table Designs – Data Dictionary

## Member Table

| Field Name | Description | Data Type | Key Field | Constraints | Example |
|---|---|---|---|---|---|
| **member_id** | Unique identifier for each user | INTEGER | Primary Key | Auto Increment, Not Null | 101 |
| **name** | Full name of the user | TEXT(30) | – | Not Null | John Smith |
| **contact_info** | Phone number of the user | TEXT(30) | – | Unique, Not Null | +64-212345678 |
| **DOB** | Date of birth | DATE | – | Nullable | 1995-08-22 |
| **subscription_id** | Subscription linked to payment | INETEGER | Foreign Key | Not Null | 201 |

## Schedule Table

| Field Name | Description | Data Type | Key Field | Constraints | Example |
|---|---|---|---|---|---|
| **schedule_id** | Unique ID for schedule entry | INTEGER | Primary Key | Auto Increment, Not Null | 00586 |
| **member_id** | Reference for the member assigned | INTEGER | Foreign Key | Not Null | 101 |
| **service_id** | Reference for the service being scheduled | INTEGER | Foreign Key | Not Null | 99965 |
| **date_time** | Schedule date and time for service | DATE | – | Not Null | 09/15/2025 17.30p.m |

## Subscription Table

| Field Name | Description | Data Type | Key Field | Constraints | Example |
|---|---|---|---|---|---|
| **subscription_id** | Unique identifier for subscription | INTEGER | Primary Key | Auto Increment, Not Null | 201 |
| **member_id** | Member who holds the subscription. | INTEGER | Foreign Key | References User(User_ID), Not Null | 101 |
| **start_date** | Date subscription begins | DATE | – | Not Null | 2025-01-01 |
| **end_date** | Date subscription ends | DATE | – | Not Null | 2025-01-07 |

## Service Table

| Field Name | Description | Data Type | Key Field | Constraints | Example |
|---|---|---|---|---|---|
| **service_id** | Unique identifier for service | INTEGER | Primary Key | Auto Increment, Not Null | 301 |
| **service_name** | Name of optional service | TEXT(100) | – | Not Null | Personal Diet Plan |
| **duration** | Duration of the service in minutes | INTEGER | – | Not Null | 45 |
| **price** | Cost of the service | REAL | – | Not Null | $20.00 |
| **staff_id** | Unique ID for each staff member | INTEGER | Foreign Key | Increment, Not Null | 1000078 |

## Payment Table

| Field Name | Description | Data Type | Key Field | Constraints | Example |
|---|---|---|---|---|---|
| **payment_id** | Unique ID for each payment | INTEGER | Primary Key | Increment, Not Null | 501 |
| **subscription_id** | Subscription linked to payment | INETEGER | Foreign Key | References Subscription(Subscription_ID), Not Null | 201 |
| **amount** | Payment amount | REAL | – | Not Null | 50.00 |
| **payment_date** | Date of payment | DATE | – | Not Null | 2025-01-10 |
| **service_id** | Reference for the service being scheduled | INTEGER | Foreign Key | Not Null | 99965 |

## Staff Table

| Field Name | Description | Data Type | Key Field | Constraints | Example |
|---|---|---|---|---|---|
| **staff_id** | Unique ID for each staff member | INTEGER | Primary Key | Increment, Not Null | 1000078 |
| **staff_name** | Reference name identifier for staff members | TEXT (30) | _ | Not Null | Jhonny Smith |
| **Role** | Position of the staff member | TEXT (30) | – | Not Null | Trainer |

## Contributions

- **Ayush Subedi: :** Summary, Use cases, Member & Schedule Table
- **Senuth Wickramasinghe:** ERD, User stories, Use case Diagram, Staff & Payment Table
- **Yogesh Gurung:** Functional & Non-Functional requirements, Service & Subscription Table