ST. CECILIA'S COLLEGE-CEBU, INC.

LASSO Supervised School

Ward II, Poblacion Minglanilla, Cebu

GYM MEMBERSHIP SYSTEM

Abarquez, Ronbell B.
BSIT - 2D

A. INTRODUCTION

This Java application is a Console-Based Membership Management System designed to handle core administrative and transactional tasks for a fitness center or gym. The system is built around a relational database model and uses a clear CRUD (Create, Read, Update, Delete) structure to manage three main entities: Users, Members, and Services.

The primary entry point is the main class, which manages user authentication (Login/Registration) and employs role-based access control (Staff vs. Admin). The functionality is modularized into three distinct classes Members, Services, and Management each responsible for interacting with its corresponding database table to ensure efficient and structured data handling.

B. SIGNIFICANCE

The code for your Gym Membership System is significant because it demonstrates three core competencies essential for professional application development:

> Modular Design and Maintenance: The use of dedicated classes like Members, Services, and Management enforces clear separation of concerns, making the entire application easy to read, debug, and maintain.
>
> Data Integrity and Persistence: The application actively ensures data quality by using SQL transactions and implementing validation checks (like verifying a Member ID exists before adding a service), which is vital for database accuracy.
>
> Role-Based Security: It implements a necessary security layer through password hashing and enforces role-based access control, restricting sensitive Management functions only to Admin users.

C. FEATURES AND STAKEHOLDERS

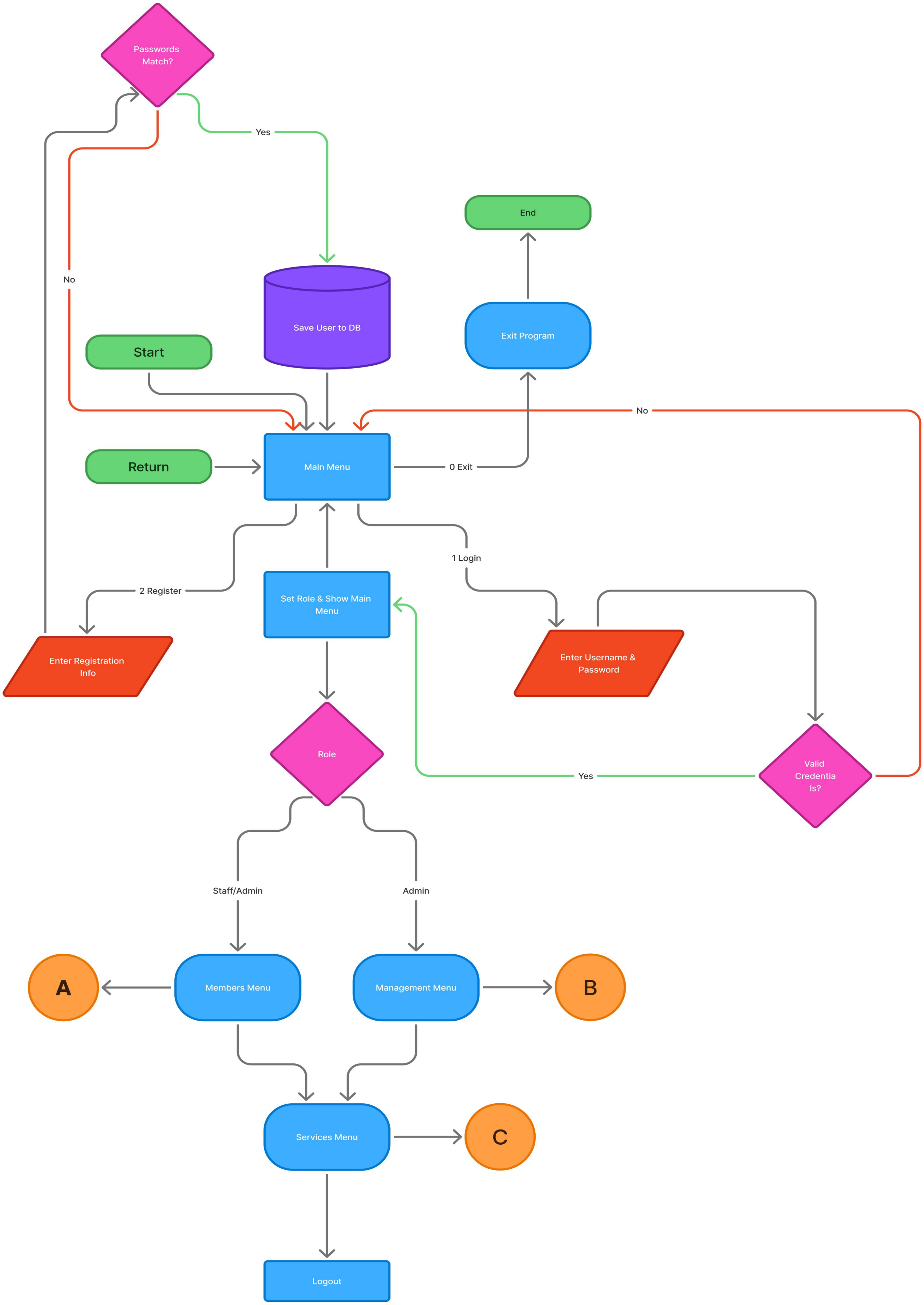
Features

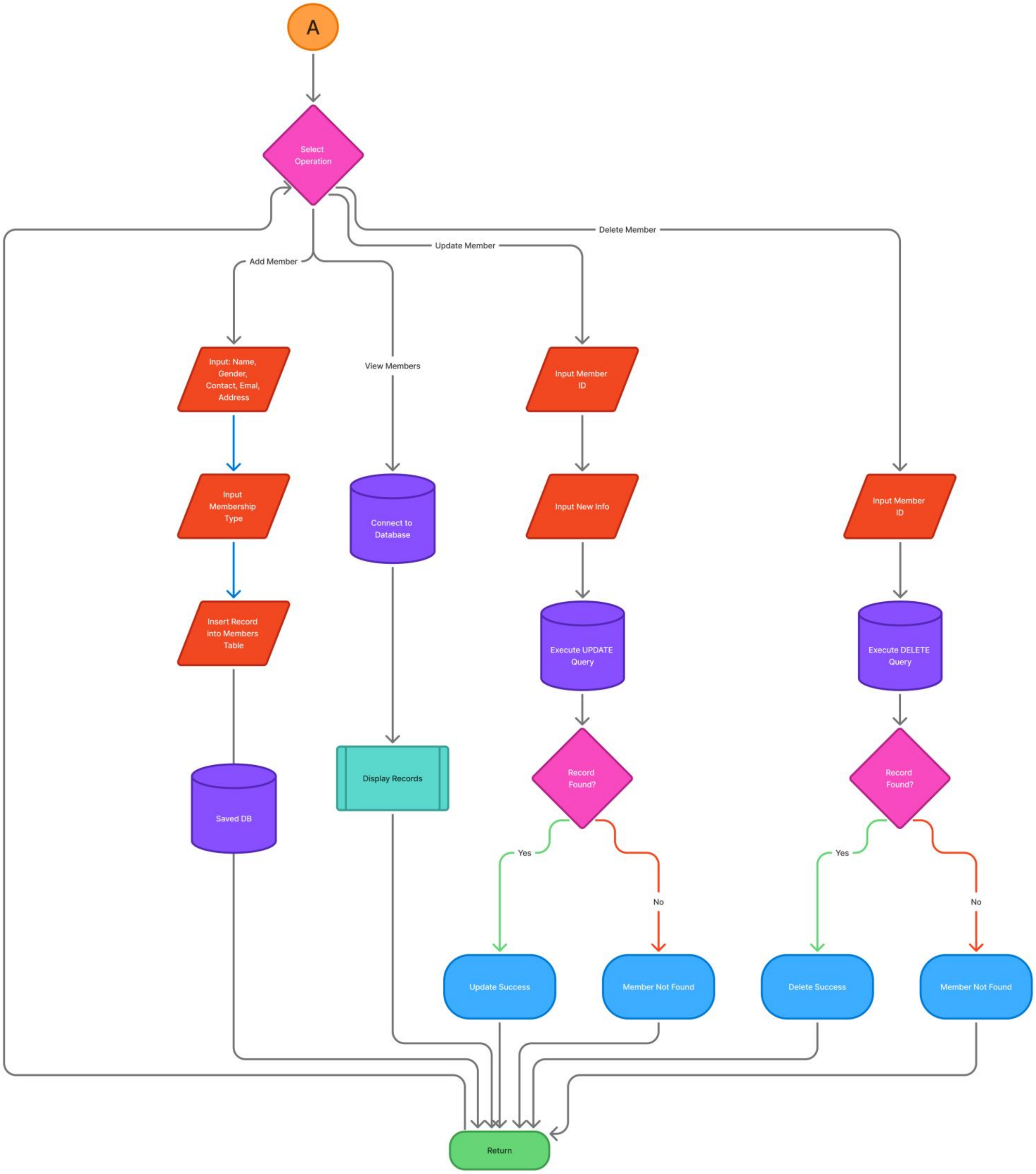
The Gym Membership System is a modular Java application that enables its Stakeholders the Admin and Staff to manage all core operations. It supports key Features including: Role-Based Access Control (RBAC) to govern permissions; CRUD operations for the Members module, allowing staff to easily register and manage member profiles; and Service Transaction recording, which maintains data integrity by validating the M_ID before logging payments. Exclusively for the Admin stakeholder, the system provides staff management CRUD to maintain sensitive employment details linked to user accounts, ultimately providing the Gym Owner with reliable data for business oversight.

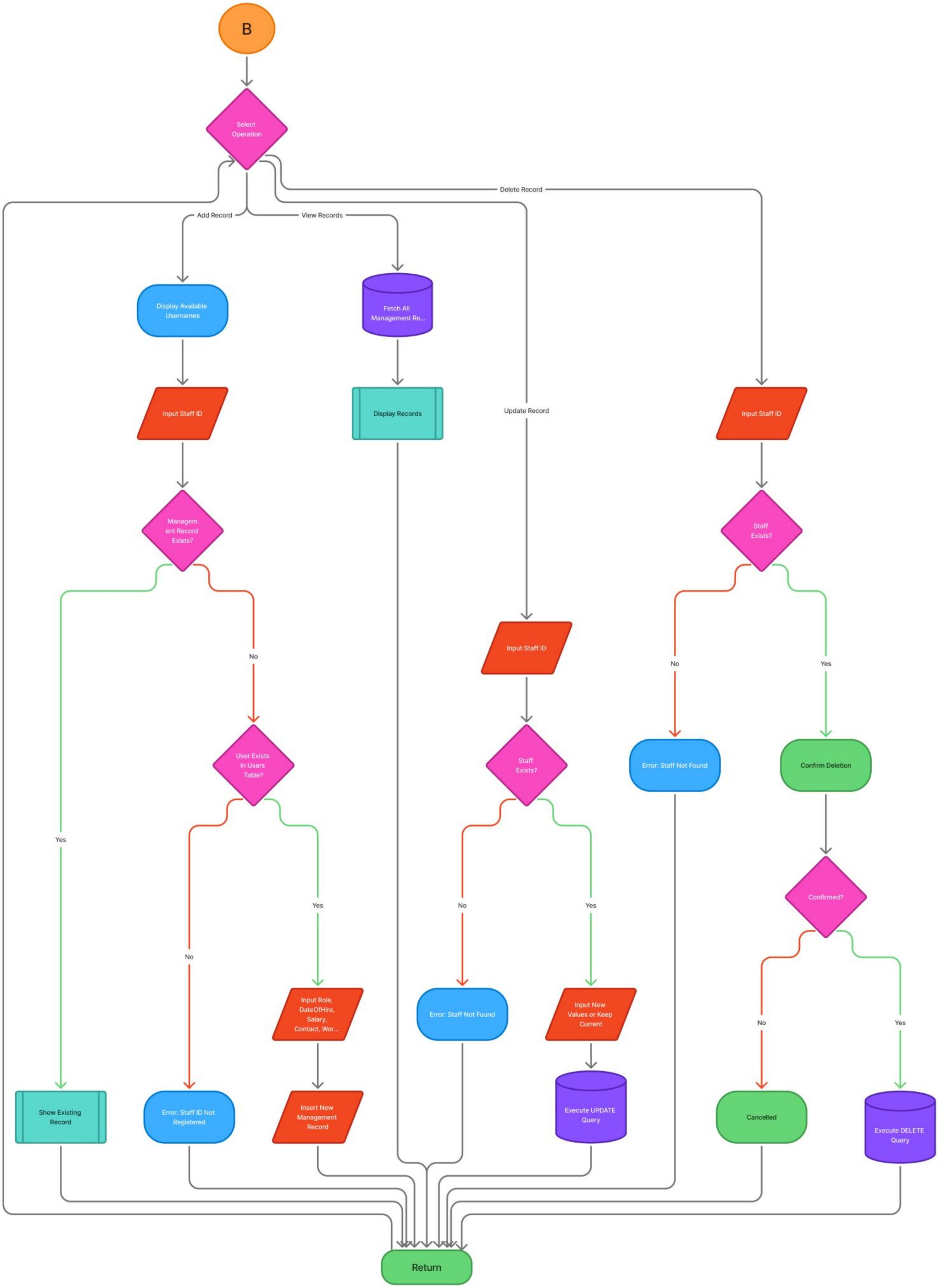
Stakeholder

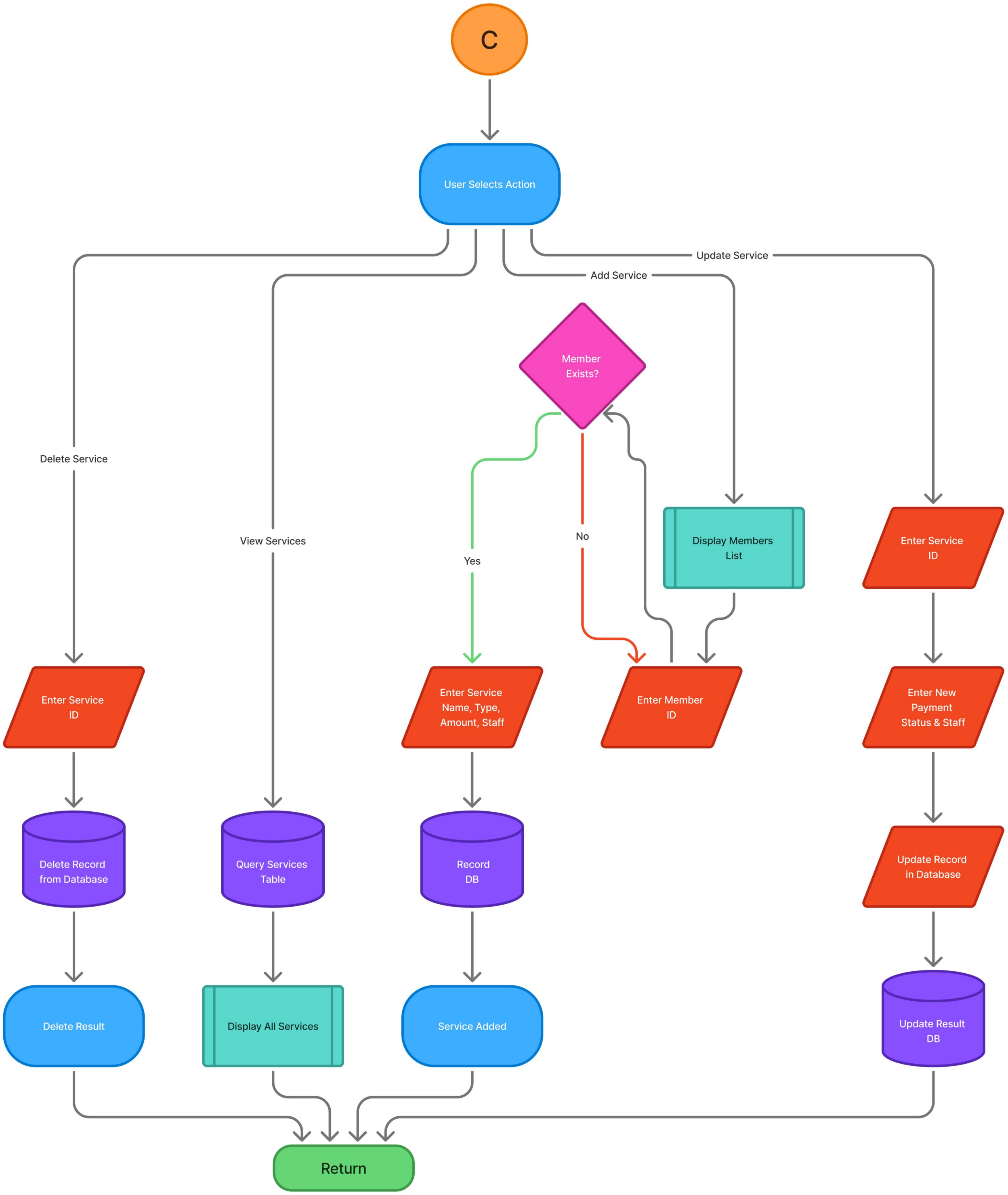
Admin Manages Security, Staff, and overall system operations.
Staff Manages Members and Services (day-to-day transactions).
Members Consumers of the gym's services, accurate tracking of their membership status and payments.
Gym Owner Business decision-maker, reliable data for revenue tracking and oversight.

D. FLOWCHART


Passwords Match?
Yes
No
End
Save User to DB
Exit Program
Start
No
Return
Main Menu
0 Exit
1 Login
2 Register
Set Role & Show Main Menu
Enter Registration Info
Enter Username & Password
Role
Valid Credentia ls?
Yes
Staff/Admin
Admin
A
Members Menu
Management Menu
B
Services Menu
C
Logout


A
Select Operation
Add Member
View Members
Update Member
Delete Member
Input: Name, Gender, Contact, Emal, Address
Input Membership Type
Insert Record into Members Table
Saved DB
Connect to Database
Display Records
Input Member ID
Input New Info
Execute UPDATE Query
Record Found?
Yes
No
Update Success
Member Not Found
Input Member ID
Execute DELETE Query
Record Found?
Yes
No
Delete Success
Member Not Found
Return


B
Select Operation
Add Record
View Records
Update Record
Delete Record
Display Available Usernames
Fetch All Management Re...
Input Staff ID
Display Records
Input Staff ID
Managem ent Record Exists?
Staff Exists?
Yes
No
No
Yes
Input Staff ID
User Exists in Users Table?
Staff Exists?
Error: Staff Not Found
Confirm Deletion
No
Yes
No
Yes
Confirmed?
Input Role, DateOfHire, Salary, Contact, Wor...
Error: Staff Not Found
Input New Values or Keep Current
No
Yes
Show Existing Record
Error: Staff ID Not Registered
Insert New Management Record
Execute UPDATE Query
Cancelled
Execute DELETE Query
Return


C
User Selects Action
Delete Service
View Services
Add Service
Update Service
Member Exists?
Yes
No
Display Members List
Enter Service ID
Enter Service ID
Enter Service Name, Type, Amount, Staff
Enter Member ID
Enter New Payment Status & Staff
Delete Record from Database
Query Services Table
Record DB
Update Record in Database
Delete Result
Display All Services
Service Added
Update Result DB
Return

E. Data Dictionary

Management Table

Key	**Field Name**	**Data Type**	**Field Length**	**Constraint**	**Description**
PK	**StaffID**	TEXT	*N/A*	**Primary Key, Not Null**	Employee Staff ID
FK	**StaffID**	TEXT	*N/A*	**Foreign Key**	References the Username field in the Users table
	Role_Position	TEXT	*N/A*	*Implied*	Employee's Role/Job Position
	DateOfHire	TEXT	*N/A*	*Implied*	The date the employee was hired
	Salary_PayRate	REAL	*N/A*	*Implied*	Employee's salary or pay rate
	ContactNumber	TEXT	*N/A*	*Implied*	Employee's contact phone number
	WorkEmail	TEXT	*N/A*	*Implied*	Employee's professional work email

Members Table

Key	**Field Name**	**Data Type**	**Constraint**	**Description**
PK	**M_ID**	INTEGER	**Primary Key, Auto-increment**	Unique Member ID, automatically generated
	Name	TEXT	Not Null	The member's full name
	Gender	TEXT	*Implied*	The member's gender
	Contact_No	TEXT	*Implied*	The member's contact phone number
	Email	TEXT	**Unique**	The member's unique email address
	Address	TEXT	*Implied*	The member's street address
	U_Type	TEXT	*Implied*	The member's user type
	Join_Date	TEXT	*Implied*	The date the member joined
	Membership_Status	TEXT	*Implied*	The current status of the membership (e.g., Active, Inactive)
	Membership_Type	TEXT	*Implied*	The category of membership (e.g., Premium, Basic)

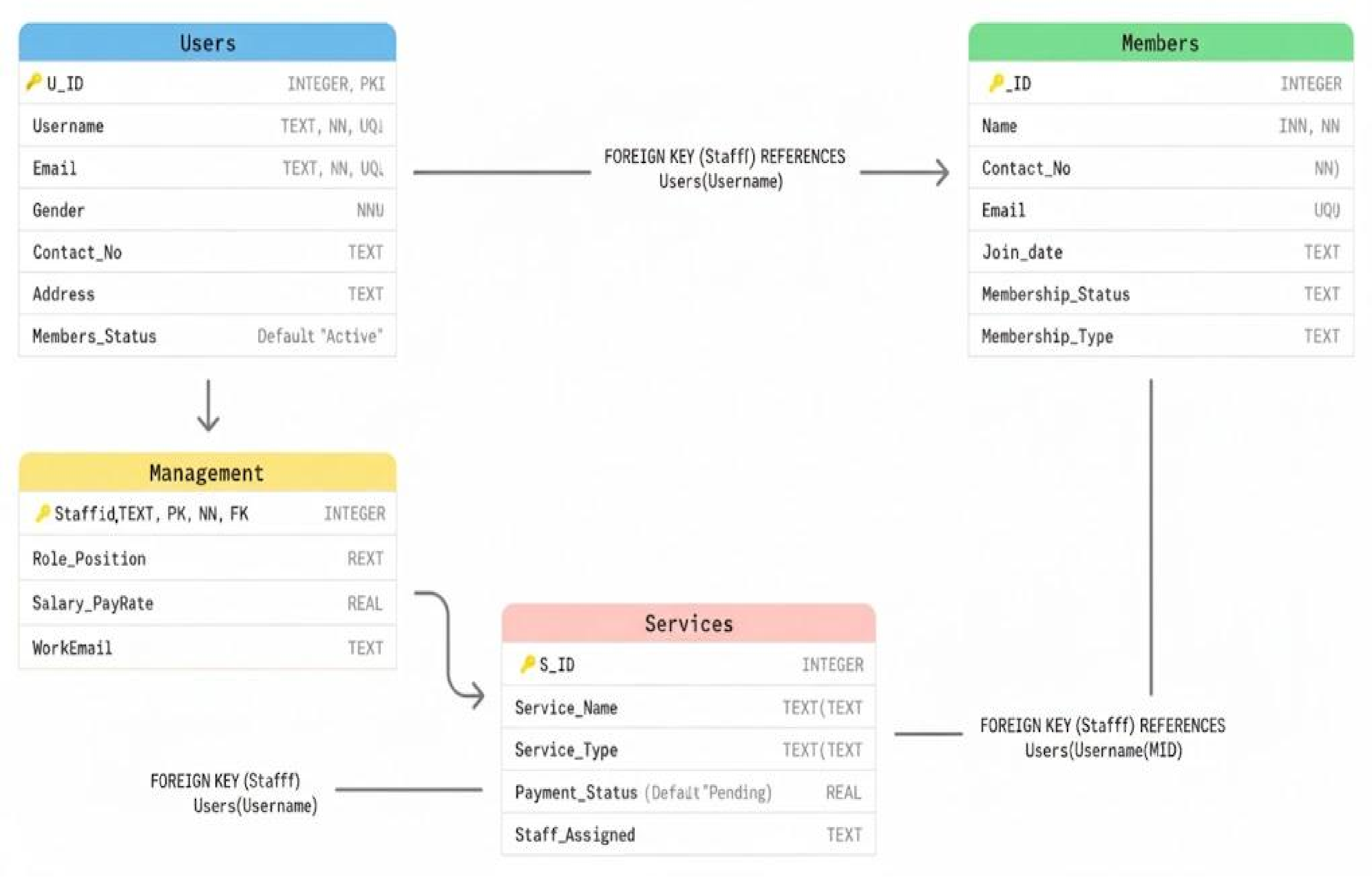
Services Table

Key	**Field Name**	**Data Type**	**Constraint**	**Description**
PK	**S_ID**	INTEGER	**Primary Key, Auto-increment**	Unique Service ID, automatically generated
FK	**M_ID**	INTEGER	**Not Null, Foreign Key**	The ID of the member who received the service. References M_ID in the Members table.
	Service_Name	TEXT	**Not Null**	The name of the specific service provided (e.g., "Gym Access", "Personal Training")
	Service_Type	TEXT	**Not Null**	The category or type of service (e.g., "Monthly Fee", "Add-on")
	Payment_Status	TEXT	Default: 'Pending'	The current status of the payment for the service
	Amount	REAL	**Not Null**	The monetary cost of the service
	Staff_Assigned	TEXT	*Implied*	The name or ID of the staff member assigned to this service/transaction

User Table

Key	Field Name	Data Type	Constraint	Description
PK	**U_ID**	INTEGER	**Primary Key, Auto-increment**	Unique User ID, automatically generated.
	Username	TEXT	**Not Null, Unique**	Unique name used for login.
	Email	TEXT	**Not Null, Unique**	Unique email address for the user.
	Password	TEXT	**Not Null**	User's password (should be securely hashed).
	U_Type	INTEGER	**Not Null**	Defines the user's account type (e.g., 1 for Admin, 2 for Member).
	Name	TEXT	*Implied*	The user's full name.
	Gender	TEXT	*Implied*	The user's gender.
	Contact_No	TEXT	*Implied*	The user's contact phone number.
	Address	TEXT	*Implied*	The user's physical address.
	Join_Date	TEXT	*Implied*	The date the user joined the system.
	Members_Status	TEXT	Default: 'Active'	The current status of the user's membership or account.

F. Entity Relationship Diagram



Users	
🔑 U_ID	INTEGER, PKI
Username	TEXT, NN, UQ|
Email	TEXT, NN, UQL
Gender	NNU
Contact_No	TEXT
Address	TEXT
Members_Status	Default "Active"

Members	
🔑 _ID	INTEGER
Name	INN, NN
Contact_No	NN)
Email	UQU
Join_date	TEXT
Membership_Status	TEXT
Membership_Type	TEXT

Management	
🔑 Staffid,TEXT, PK, NN, FK	INTEGER
Role_Position	REXT
Salary_PayRate	REAL
WorkEmail	TEXT

Services	
🔑 S_ID	INTEGER
Service_Name	TEXT(TEXT
Service_Type	TEXT(TEXT
Payment_Status (Defalt "Pending)	REAL
Staff_Assigned	TEXT

G. MANUAL (INSTRUCTIONS)

The system is designed using three dedicated Java modules to manage the gym's database.

1. Members Module

Create (addMember): Adds a new member to the system by collecting personal details and automatically setting the join date and 'Active' status.

Read (viewMembers): Displays a comprehensive list of all member records.

Update (updateMember): Allows changing a member's contact information, address, and membership status based on their M_ID.

Delete (deleteMember): Removes a member's record permanently using their **M_ID**.

2. Services Module

Create (addService): Records a new service, requires validation to ensure the entered M_ID exists in the Members table, and defaults the payment status to 'Pending'.

Read (viewServices): Displays all recorded service transactions.

Update (updateService): Primarily allows updating the payment status and the staff assigned to the service using the S_ID.

Delete (deleteService): Removes a service record using its S_ID.

3. Management Module (Admin Access Only)

Helpers: Includes functions to display a list of registered usernames (potential staff IDs) and check if a staff ID has an existing record.

Create (addManagementRecord): Adds a staff record (Role, Salary, Hire Date) by using a registered Username as the StaffID, ensuring that a user is registered before employment details are added.

Read (viewManagementRecords): Displays all staff employment records.

Update (updateManagementRecord): Allows the Admin to modify staff details (Role, Salary, etc.) by searching for the StaffID.

Delete (deleteManagementRecord): Deletes a staff employment record after explicit confirmation.

This design ensures clear separation of duties: Staff and Admin manage members/services, while only Admin manages staff records.