Doc

Fitness Training Management System

Software Requirements & System Design Documentation
(Console-Based Application – SQLite Storage)

---

1. Introduction

The **Fitness Training Management System** is a console-based application designed to manage fitness training workflows between **Trainees**, **Trainers**, and **Admins**.
The system enables structured workout planning, trainer–trainee communication, and administrative control while ensuring data integrity.

The application also supports **exercise animations** to visually demonstrate correct exercise execution within a console environment.

---

2. System Objectives

- Manage fitness training relationships efficiently
- Enforce role-based access and responsibilities
- Provide guided workout plan creation
- Help trainees understand exercises through animations
- Maintain data accuracy and consistency
- Support future scalability and maintainability

---

3. User Roles Overview

3.1 Trainee (Client)

Receives workout plans, views exercise animations, and communicates with an assigned trainer.

3.2 Trainer

Creates workout plans, assigns exercises, and manages trainees.

3.3 Admin

Controls system users, exercises, and reports.

---

4. Functional Requirements

4.1 General System Functions

- Welcome screen with Login, Signup, and Exit options
- Graceful exit from any screen
- Role-based main menus
- “Back” option in all submenus
- ASCII table display for lists
- Persistent data storage
- Default admin account on first launch

4.2 Authentication and Authorization

- Signup for Trainee and Trainer roles
- Required information:
  - Unique username
  - Unique email
  - Password with confirmation
  - Name
  - Age (minimum **12 years**)
  - Gender
- Secure login
- Role-based menu routing
- Profile update and password change
- Logout functionality

4.3 Trainee Functional Requirements

- View and search trainers
- View trainer profiles
- Assign or leave a trainer
- View assigned workout plan
- Browse exercise library
- **View exercise animations**

- Manage personal exercise lists
- Send and receive messages with trainer
- Export workout plans to text files

4.4 Trainer Functional Requirements

- View assigned trainees
- Access trainee profiles
- Create and edit workout plans
- Select exercises with animations
- Assign workout plans
- Message trainees
- Dashboard summary

4.5 Admin Functional Requirements

- Default admin account on first launch
- Create additional admins
- View, edit, search, and delete users
- Manage exercise library and animations
- Generate system reports

---

5. Non-Functional Requirements

5.1 Usability

- Clear and consistent navigation
- Descriptive prompts and errors
- Simple console interaction

5.2 Performance

- Fast response time
- Efficient animation loading

5.3 Security

- Role-based access control

- User-specific permissions

5.4 Reliability

- Strong data integrity
- Safe deletion of related data
- Graceful handling of missing animations

5.5 Maintainability

- Modular system design
- Easy feature expansion

---

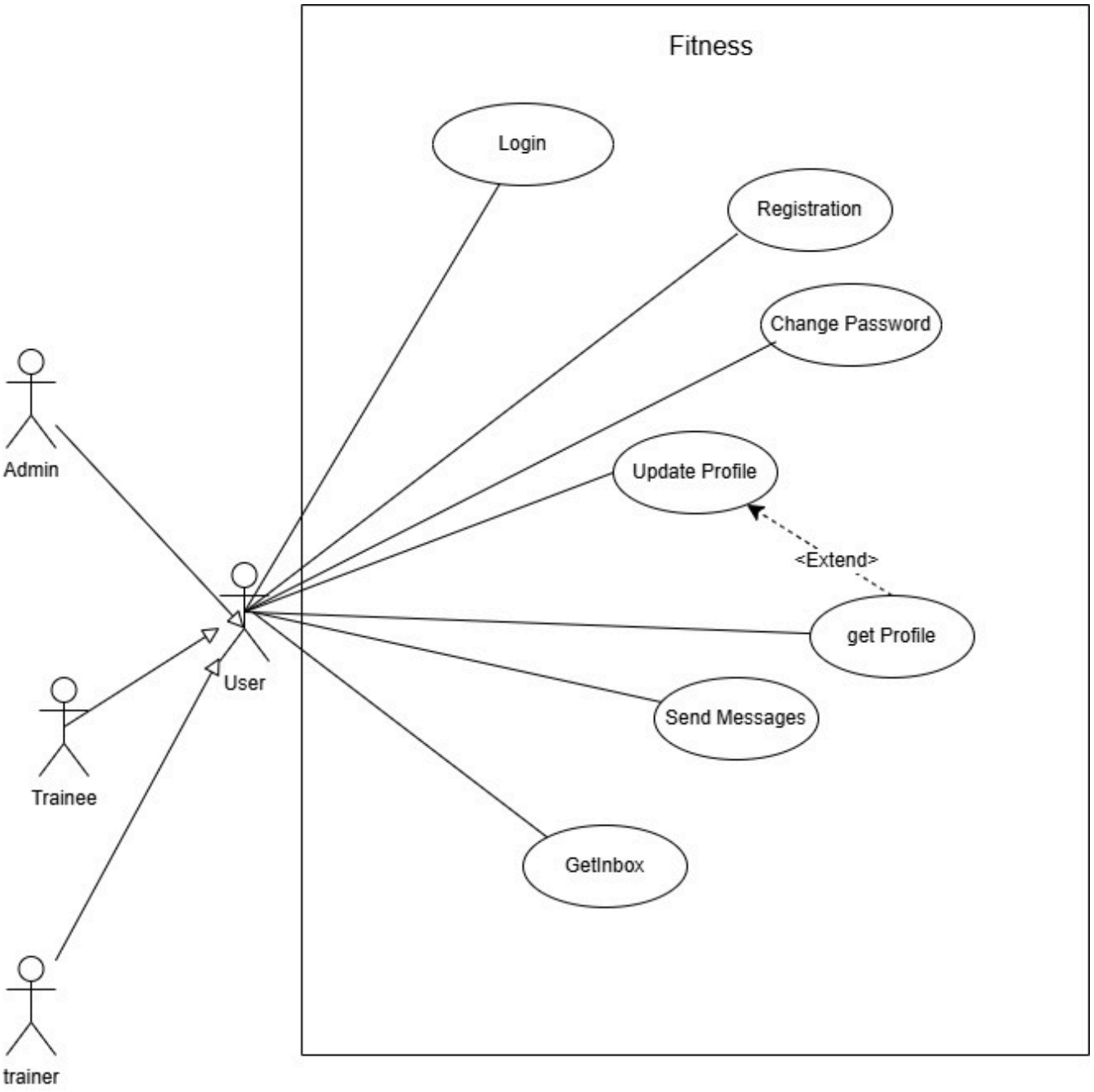
6. Data Constraints & Business Rules

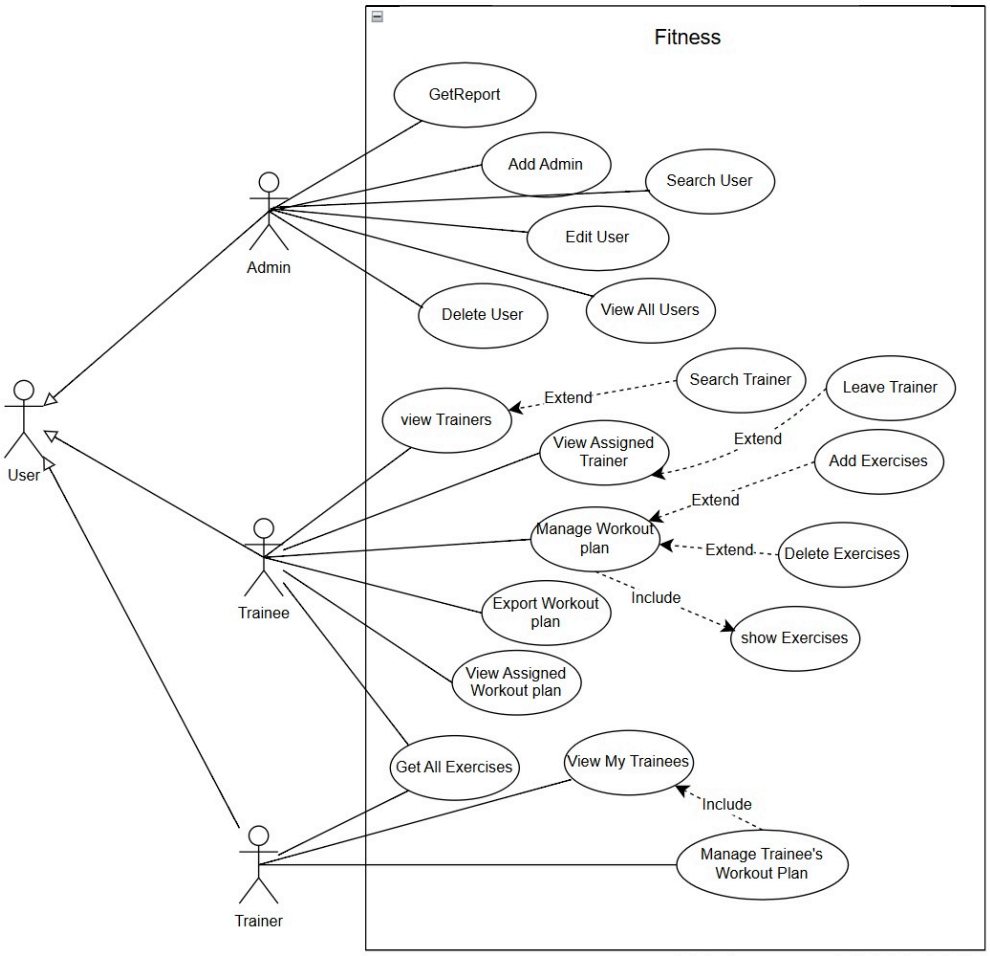
- User age must be **12 years or older**
- Usernames and emails must be unique
- A trainee can have only one trainer at a time
- A trainer can have multiple trainees
- Workout plans cannot contain duplicate exercises
- Exercises may optionally include animations

---

7. System Design Diagrams

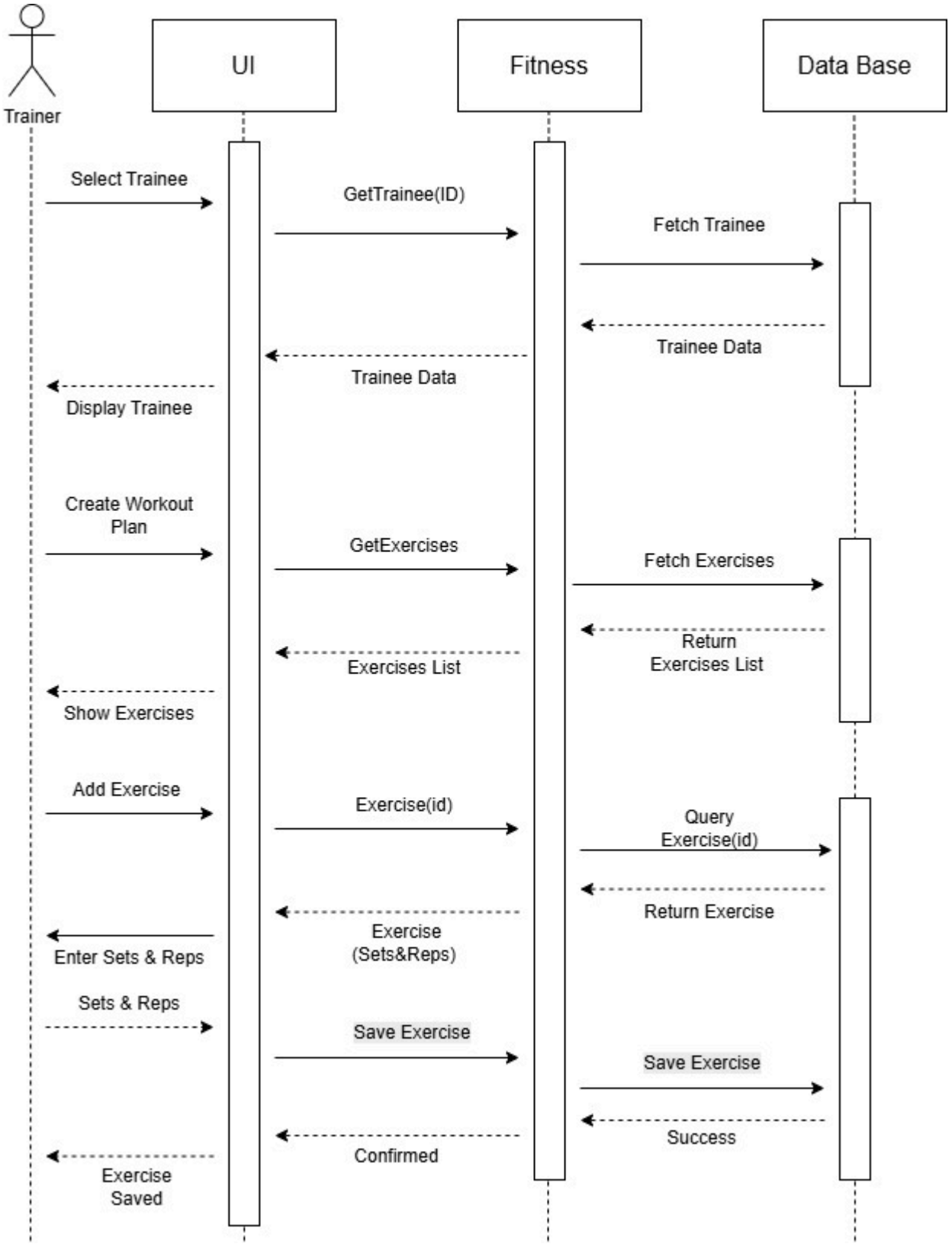
7.1 Use Case Diagram


Fitness
Login
Registration
Change Password
Update Profile
<Extend>
get Profile
Send Messages
GetInbox
Admin
Trainee
trainer
User

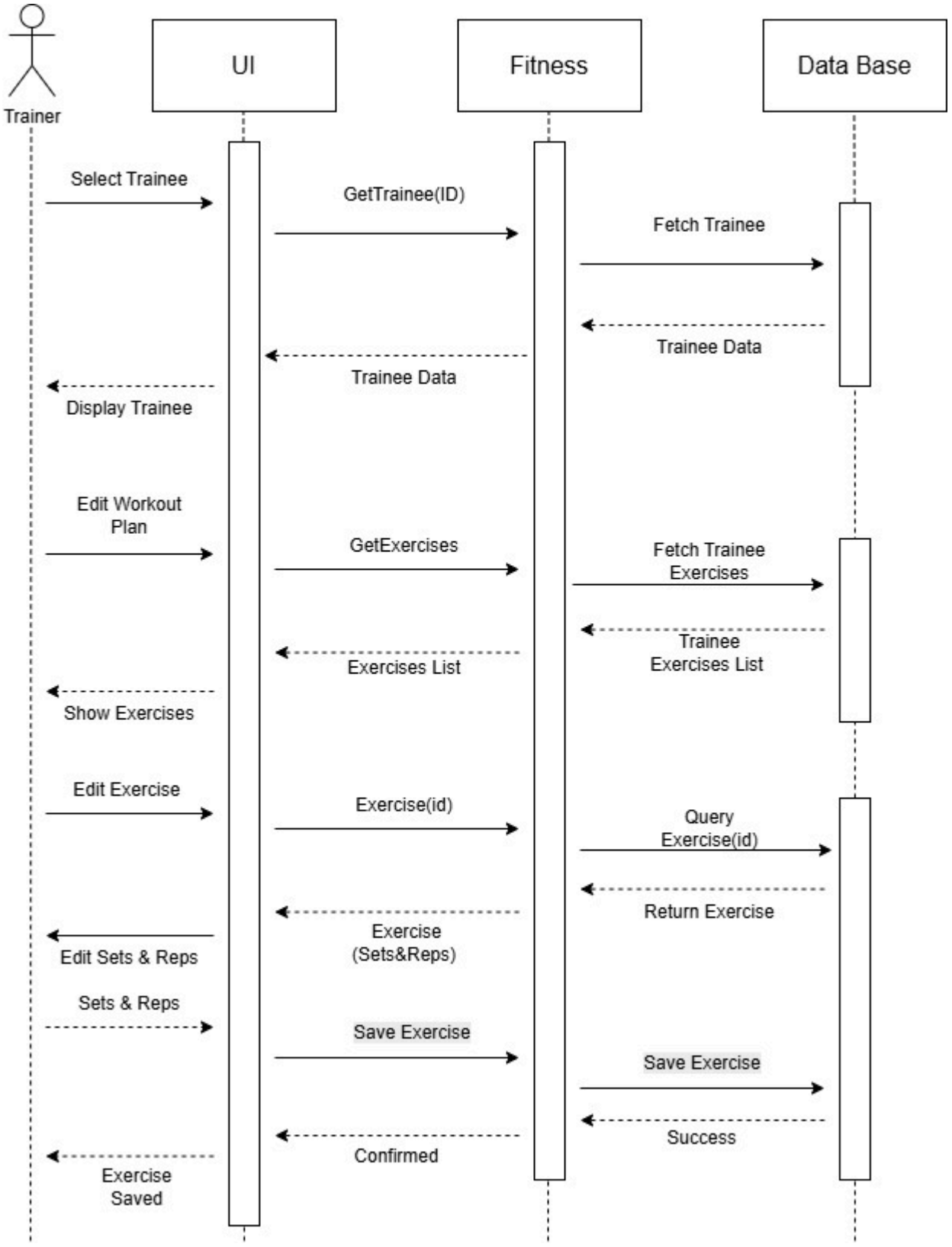


7.2 Sequence Diagrams

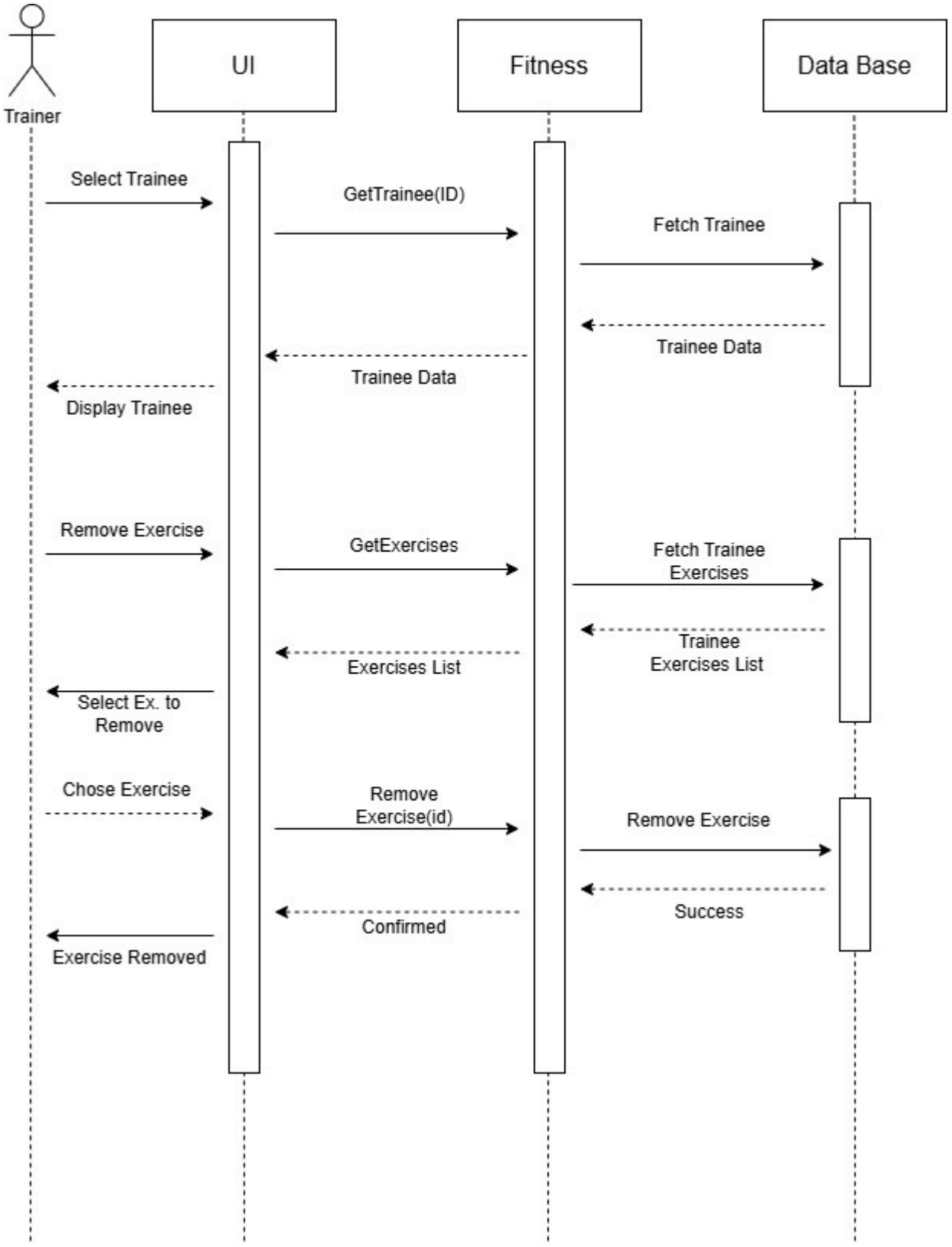
Create Workout Plan



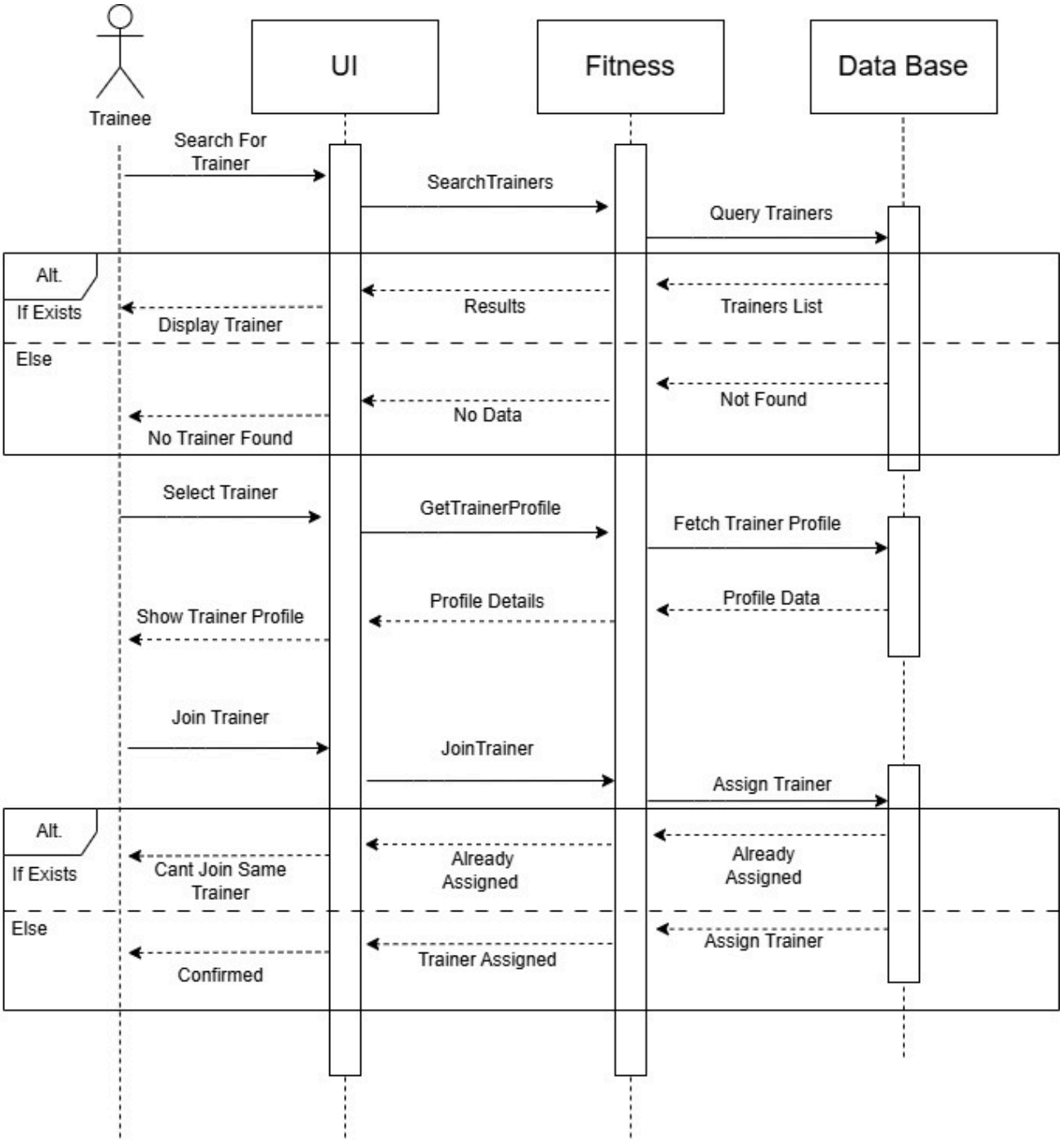
Edit Existing Plan



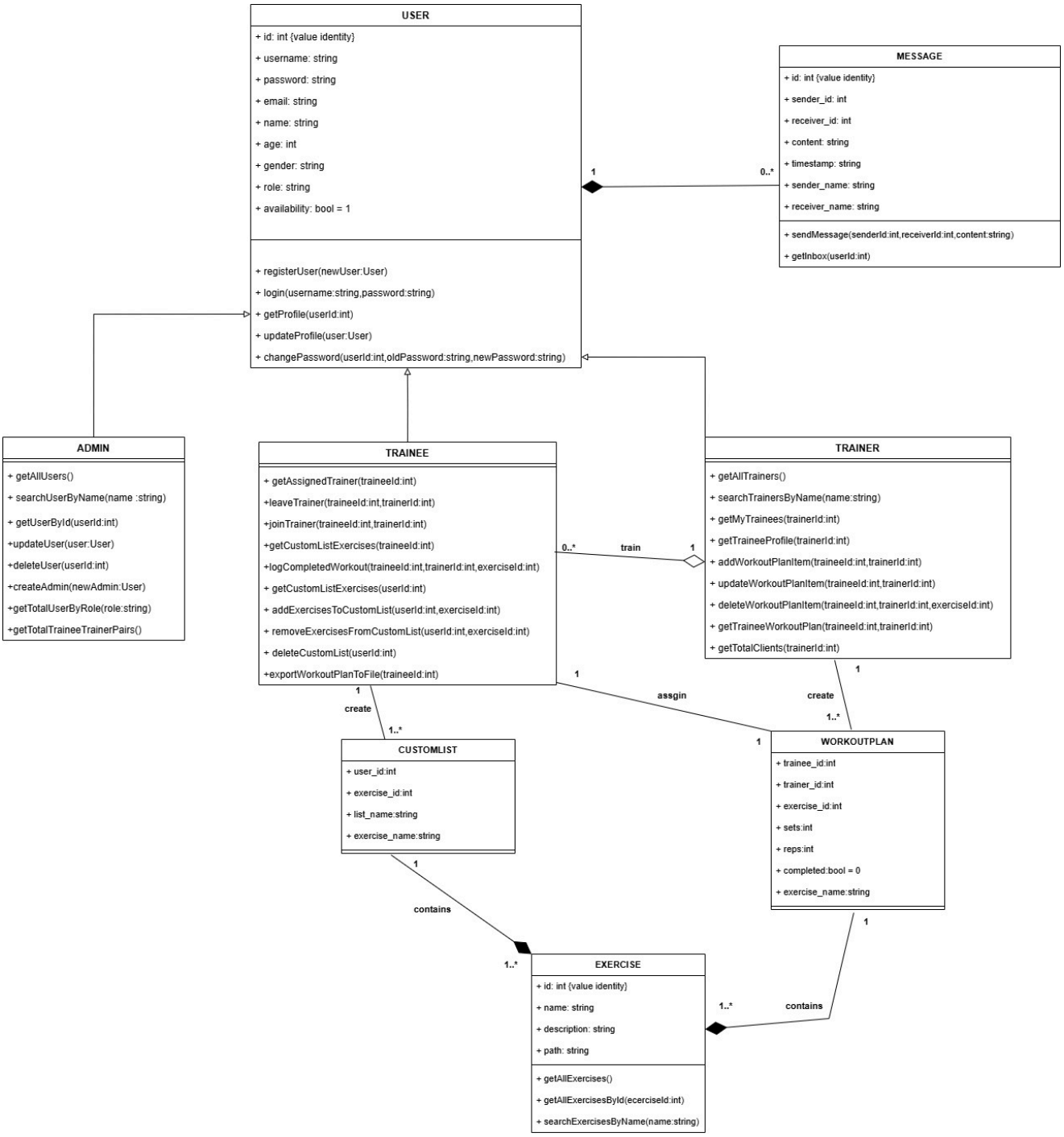
Remove Exercise From Workout Plan



Search And Join

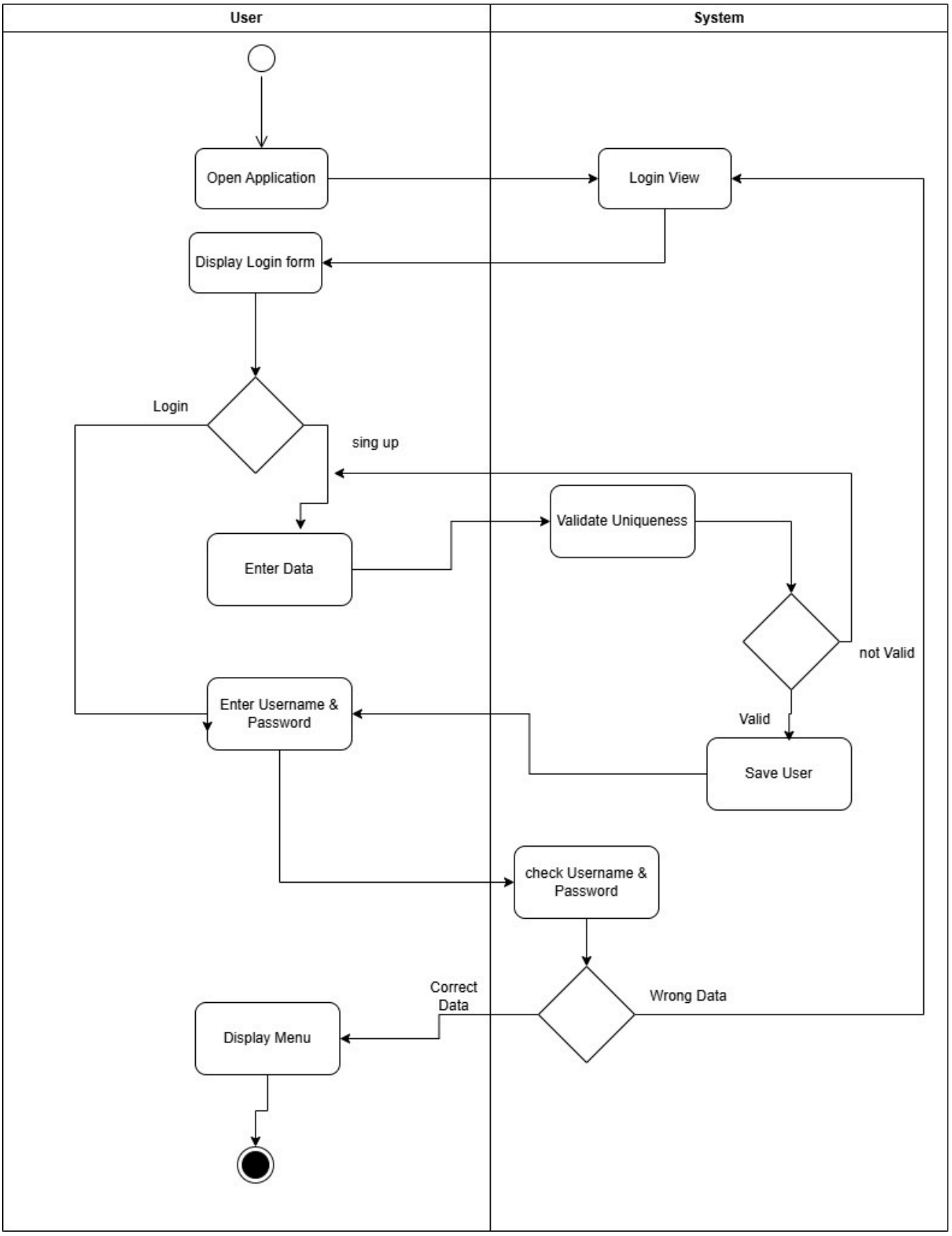


7.3 Class Diagram

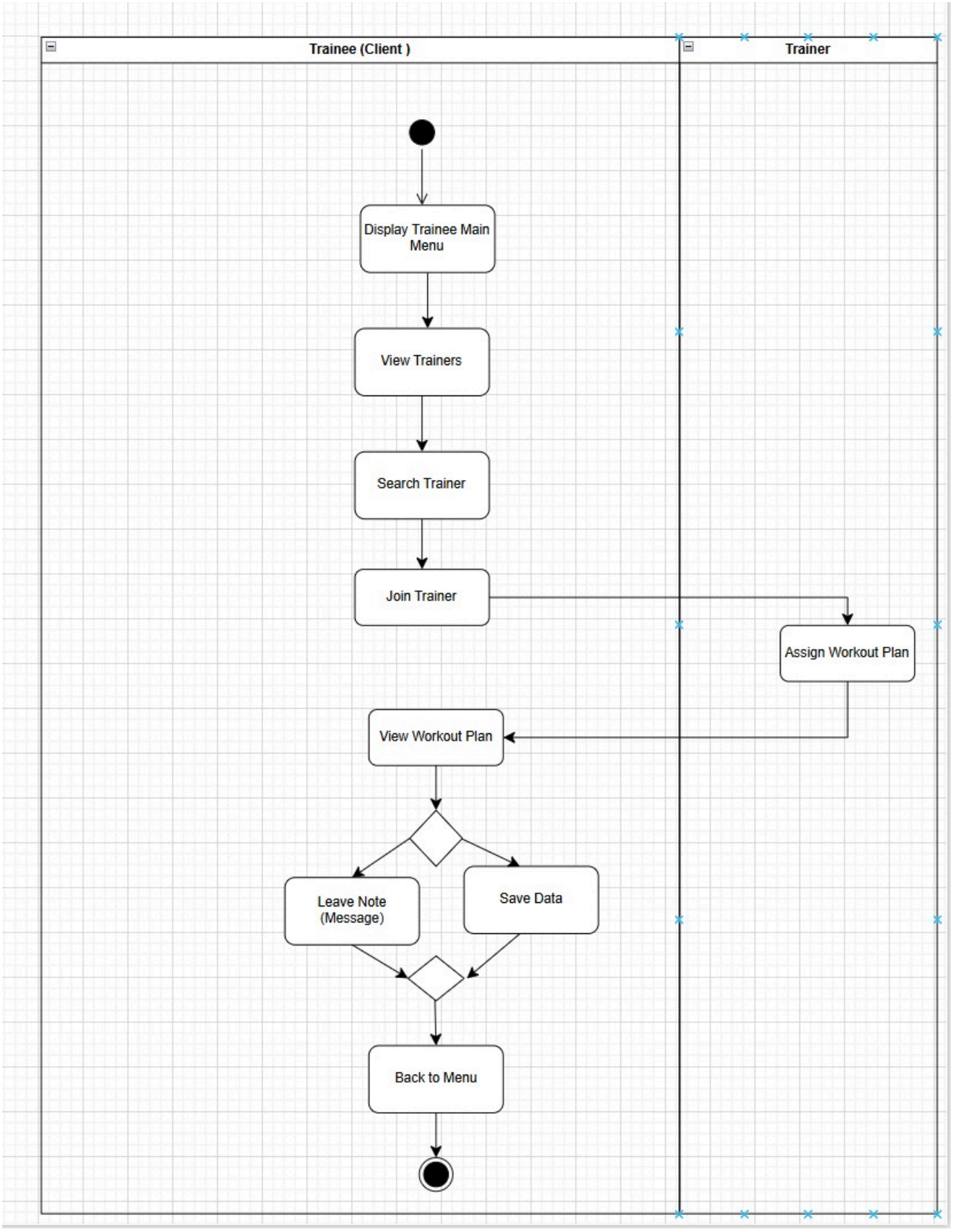


7.4 Activity Diagrams

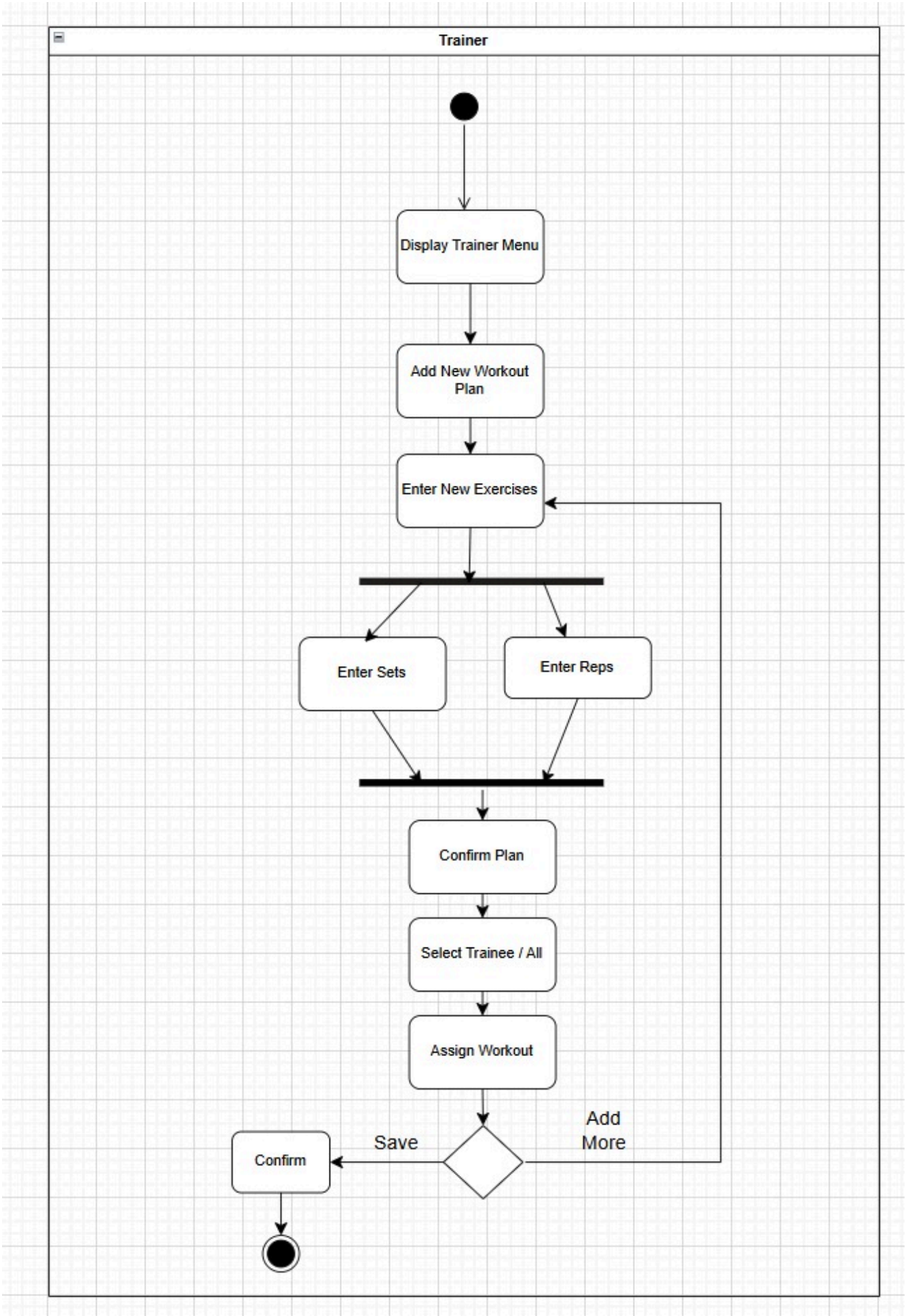
Login



Join And View Workout Plan



Add Exercises For Workout Plan



8. Software Development Methodology

8.1 Development Methodology Used

The system follows the **Agile Software Development Methodology**.

- Agile methodology divides the project into short, iterative phases called **sprints**
- It fits our case because the application features are **not fully known in advance**
- Allows **fast delivery** of working versions of the application
- Makes it easy to **adapt to changes** and add new features based on feedback

Agile allows rapid improvement of features such as workout planning, animations, and messaging based on user feedback.

---

9. Technologies Used

9.1 Programming Language

- **C++** – Core application logic and console interface

9.2 Build System

- **CMake** – Project configuration and build automation

9.3 Version Control

- **Git** – Source code versioning and collaboration

9.4 Database

- **SQLite** – Lightweight relational database for persistent storage

---

10. Future Enhancements

- Enhanced animation playback
- Workout progress tracking
- Exercise categorization
- Performance analytics
- Audit logs
