FitLife Gym Web Application
Software Architecture Documentation

1. Introduction

This document presents a detailed software architecture description of the FitLife Gym Web Application.
The project is developed as part of a Software Architecture course and focuses on applying Microservices Architecture concepts using Flask.
The main goal is to design a scalable, maintainable, and modular backend system.

2. Problem Statement

Traditional monolithic gym management systems suffer from tight coupling and limited scalability.
FitLife Gym aims to solve this by separating responsibilities into independent services, making the system easier to extend and maintain.

3. System Objectives

- Provide secure authentication and authorization
- Allow admins to manage users
- Display fitness plans dynamically
- Support future services like payments
- Demonstrate microservices concepts clearly

4. Stakeholders

- End Users (Gym Members)
- Admin Users
- System Developer
- Course Instructor

5. Architectural Style

The system follows a Microservices Architecture.
Each service is responsible for a single business capability.
Flask is used to implement lightweight services.
Inside each service, MVC principles are applied.

6. System Architecture Overview

The application consists of multiple independent services:

- Authentication Service
- Admin Service
- Meal Plan Service
- Payment Service (Planned)

Each service can be deployed independently and communicates through HTTP.

7. Authentication Service

Responsibilities:
• User Registration
• User Login
• Session Management
• Admin Detection

Endpoints:
POST /register
POST /sign
GET /logout

Security:
• Password hashing
• Session-based authentication

8. Admin Service

Responsibilities:
• View users
• Add users
• Delete users

Admin Access:
Only users with is_admin = True can access admin functionalities.

9. Meal Plan Service

Responsibilities:
• Display meal plans
• Separate plans based on goals
• Fetch data from database

Plans:
• Gain Muscle
• Lose Fat

10. Payment Service (Planned)

Responsibilities:
• Handle subscription payments
• Simulate payment process
• Connect packages with users

11. Database Design

Database: MySQL

Users Table:
- id (Primary Key)
- firstname
- email (Unique)
- password_hash
- is_admin

The database is shared between services with strict responsibility boundaries.

12. Technology Stack

- Backend: Python (Flask)
- Database: MySQL
- ORM: SQLAlchemy
- Frontend: HTML, CSS, Bootstrap
- Security: Werkzeug

13. Security Considerations

- Passwords are hashed
- Sessions are used for authentication
- Admin routes are protected
- SQL Injection is prevented using ORM

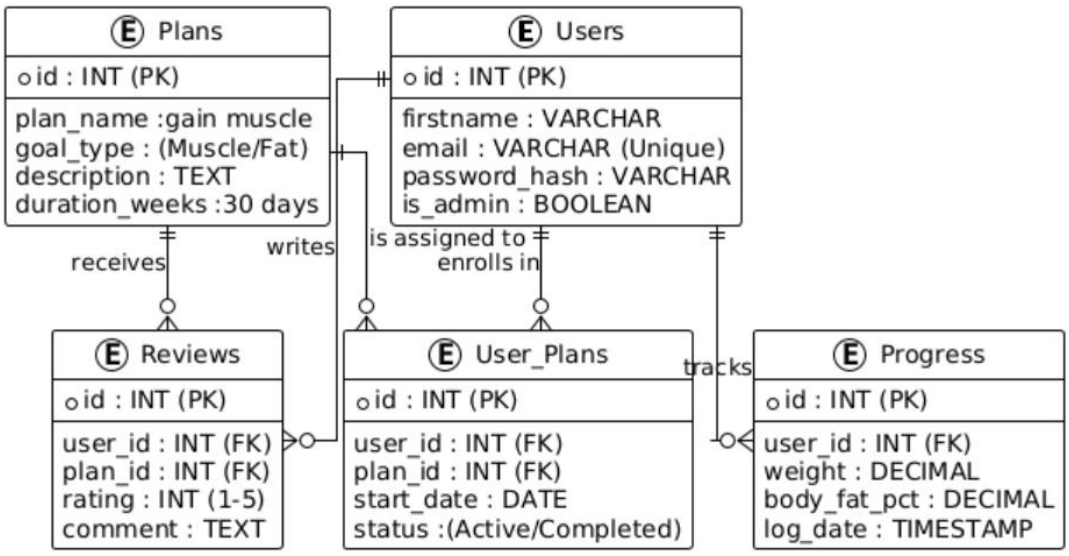
14. Future Enhancements

- JWT authentication
- Full payment gateway integration
- Separate databases per service
- Docker and containerization

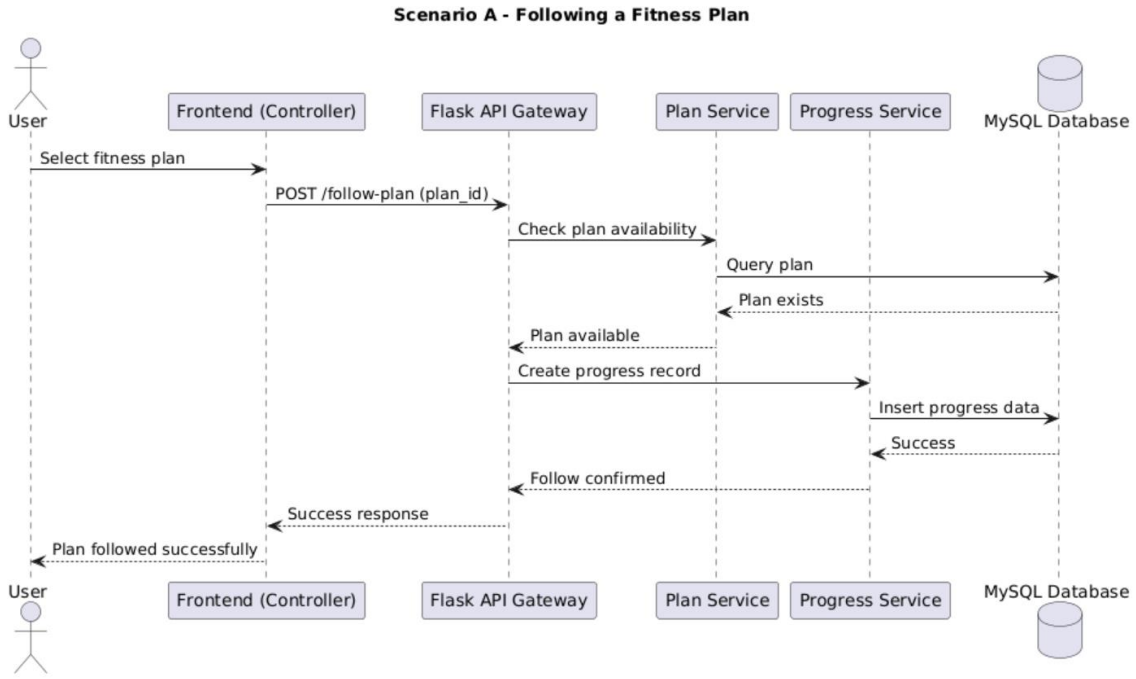
15. Conclusion

This project demonstrates a clear understanding of software architecture principles.
By applying microservices, the FitLife Gym system achieves modularity, scalability, and maintainability.

Database Schema (Simple ER Diagram)



Scenario A —Following a Plan



Scenario B — Rating / Reviewing