SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Document Authorship

- **Document Title**: SRS
- **Project Name**: LifeSync
- **Date**: 04.02.2026
- **Team Members**: Elif Beyza Turan, Beyza Değirmenci, Fatma Zehra Paksoy, Kerem Elma, Mehmet Eski
- **Contributors to this document**: Beyza Değirmenci, Fatma Zehra Paksoy

---

Table of Contents

---

1. Document Task Matrix

This matrix outlines the specific contributions of each team member to the preparation of this Software Requirements Specification (SRS) document:

Section / Task	**Responsible Member**	**Contribution**
1. Document Task Matrix	**Beyza & Fatma**	Organized the team contribution structure.
2. Introduction & 3. Product Perspective	**Beyza Değirmenci**	Defined purpose, scope, and architectural perspective.
4. Product Functions & 8. Non-Functional Req.	**Beyza Değirmenci**	Detailed functional requirements and performance benchmarks.
5. User Characteristics & 6. Constraints	**Beyza Değirmenci**	Analyzed target users and system constraints.
7. System Features (Use Cases 1-5)	**Fatma Zehra Paksoy**	Authored use case descriptions and logic.
Use Case Diagram (Visual)	**Fatma Zehra Paksoy**	Designed the visual diagram and pattern mapping.
9. External Interface Requirements	**Fatma Zehra Paksoy**	Documented UI components and software interfaces.
Final Review & Formatting	**Beyza & Fatma**	Cross-checked document consistency and final review.

2. Introduction

2.1 Purpose

The purpose of this document is to define the software requirements for the LifeSync system. It outlines functional and non-functional requirements, system constraints, and the integration of Design Patterns into the architectural framework.

2.2 Project Description

LifeSync is a digital wellness assistant that determines a user's fitness level based on onboarding data and generates personalized diet and exercise routines via a language model API integration. The system promotes sustainable healthy habits through automated reminders and progress tracking.

3. Product Perspective

The system is designed as a standalone MVP (Minimum Viable Product).

- **Architecture**: It follows a layered architecture where AI services are managed via the **Facade Pattern** and notification mechanisms utilize the **Observer Pattern**.
- **Integration**: External integration is limited to the Language model API integration for routine generation.

4. Product Functions

The system performs the following core functions:

- **FR-01 (Authentication)**: Secure signup and login functionality.
- **FR-02 (User Profiling)**: Collection of metrics (age, height, weight), goals, and dietary restrictions through an onboarding survey.
- **FR-03 (Level Determination)**: Assigning "Beginner, Intermediate, or Advanced" status using the **Strategy Pattern** based on survey analysis.
- **FR-04 (AI Routine Generation)**: Creating custom weekly diet and exercise plans via LLM integration.
- **FR-05 (Dashboard)**: A central interface for users to view their plans and track their current status.
- **FR-06 (Notification System)**: Scheduler-based reminders for workouts and meal times.

5. User Characteristics

User Type	**Features & Expectations**
General User	Users with low technical knowledge who expect a simple, intuitive interface.

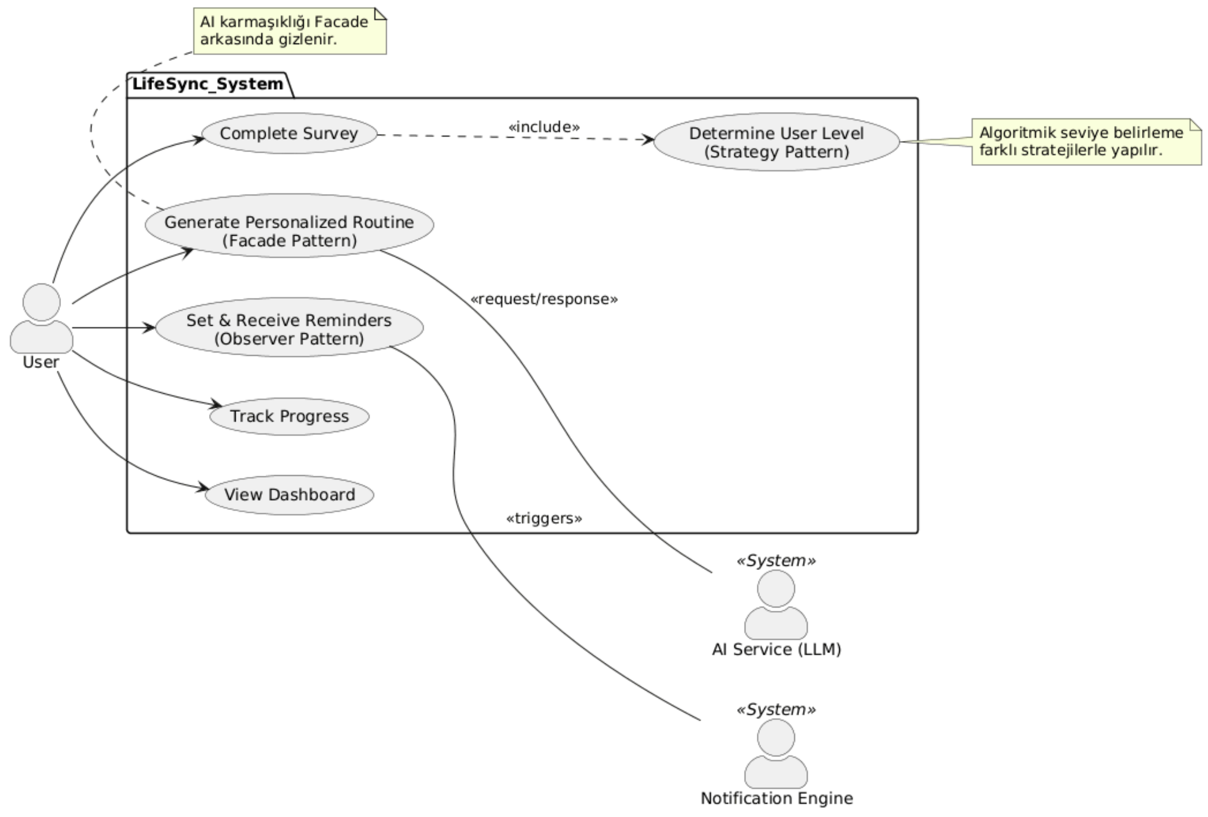
Student	Users requiring time management and structured routines amidst a busy schedule.
Employee	Professionals with sedentary roles focused on reminders and quick exercise routines.

6. Constraints

- The system does **not** provide medical diagnoses or professional health advice.
- Access to AI services requires an active internet connection.
- Development is limited by a 10-week academic timeline.
- Real-time biometric data integration (e.g., smartwatches) is currently out of scope.

7. System Features (Use Case Based)

The system's workflow and its relationship with design patterns are visualized in the following diagram:


AI karmaşıklığı Facade arkasında gizlenir.
LifeSync_System
Complete Survey
«include»
Determine User Level (Strategy Pattern)
Algoritmik seviye belirleme farklı stratejilerle yapılır.
Generate Personalized Routine (Facade Pattern)
«request/response»
Set & Receive Reminders (Observer Pattern)
User
Track Progress
View Dashboard
«triggers»
«System»
AI Service (LLM)
«System»
Notification Engine

7.1 Use Case 1: Complete Survey

- **Actor**: User
- **Goal**: Collect user metrics to build a profile
- **Design Pattern**: **Builder Pattern** (for step-by-step profile object construction).

7.2 Use Case 2: Determine User Level

- **Actor**: System
- **Goal**: Categorize user difficulty level
- **Design Pattern**: **Strategy Pattern** (to allow different analysis algorithms).

7.3 Use Case 3: Generate Personalized Routine

- **Actor**: User / AI Service
- **Goal**: Generate an AI-powered wellness plan
- **Design Pattern**: **Facade Pattern** (hiding AI API complexity).

7.4 Use Case 4: Receive Reminders

- **Actor**: System / Notification Engine
- **Goal**: Notify user of scheduled tasks
- **Design Pattern**: **Observer Pattern** (automatic notification updates).

7.5 Use Case 5: Track Progress

- **Actor**: User
- **Goal**: Monitor routine completion and visualize personal health trends
- **Flow**: User views dashboard -> Marks tasks as completed -> System updates progress metrics.

8. Non-Functional Requirements

8.1 Performance

- **AI Speed**: Users must receive their plan within **30 seconds**.
- **Dashboard Load**: The dashboard must load in **under 2 seconds**.

8.2 Reliability

- **Accuracy**: The system aims for a low hallucination rate in AI responses through strict prompt engineering.
- **Stability**: Data persistence must be maintained without loss.

8.3 Portability

- The system must be web-based and responsive for mobile and desktop browsers.

9. External Interface Requirements

- **User Interface**: Dashboard, Survey Forms, and Calendar View.
- **Software Interfaces**:
    - **Backend**: Node.js / Express.js.
    - **Frontend**: React.
    - **Database**: PostgreSQL.
    - **AI Model**: Language model API integration.

10. Conclusion

This document provides the technical framework for the LifeSync project. These requirements ensure that the final product effectively leverages AI capabilities while maintaining high standards of Object-Oriented Design (OOD).