

# 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

Document Authorship.....	1
Table of Contents.....	1
1. Document Task Matrix.....	2
2. Introduction.....	3
2.1 Purpose.....	3
2.2 Project Description.....	3
3. Product Perspective.....	3
4. Product Functions.....	3
5. User Characteristics.....	3
6. Constraints.....	4
7. System Features (Use Case Based).....	4
7.1 Use Case 1: Complete Survey.....	4
7.2 Use Case 2: Determine User Level.....	4
7.3 Use Case 3: Generate Personalized Routine.....	5
7.4 Use Case 4: Receive Reminders.....	5
8. Non-Functional Requirements.....	5
8.1 Performance.....	5
8.2 Reliability.....	5
8.3 Portability.....	5
9. External Interface Requirements.....	5
10. Conclusion.....	5

---

## 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 1-4)	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.

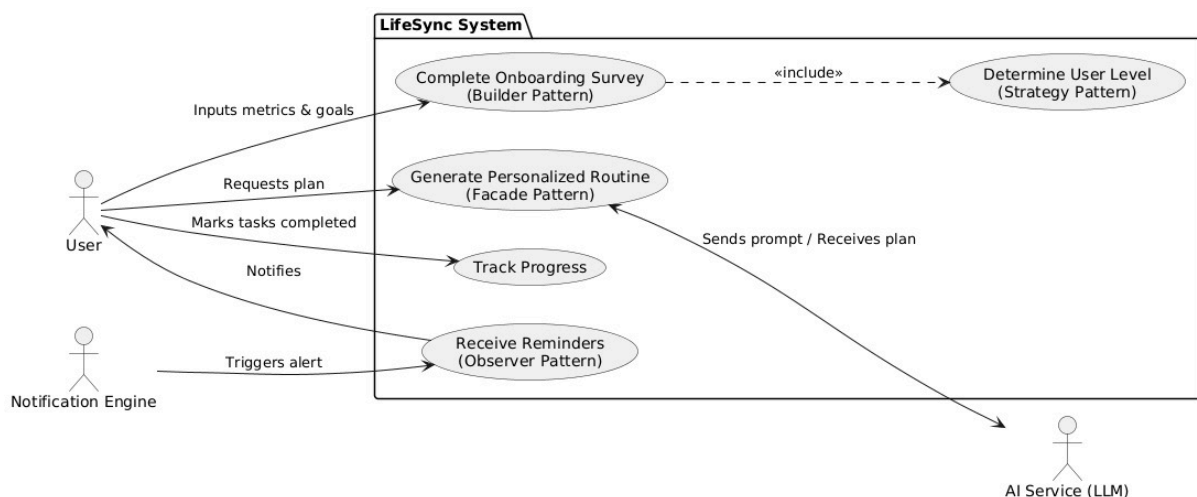
<b>Employee</b>	Professionals with sedentary roles focused on reminders and quick exercise routines.
-----------------	--

## 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:



### 7.1 Use Case 1: Complete Survey

**Title:** Complete Survey

**Main Actor:** User

**Goal:** Provide personal information for level determination

**Preconditions:**

- User has access to the system
- User has not completed survey before

**Main Flow:**

- User opens survey page
- System displays survey questions
- User answers questions
- User submits responses
- System stores responses

**Postconditions:**

- Survey data saved
- User profile initialized

**Alternate Flow:**

- If user submits incomplete survey → system prompts completion

## 7.2 Use Case 2: Determine User Level

**Title:** Determine User Level

**Main Actor:** System

**Goal:** Assign appropriate difficulty level

**Preconditions:**

- Survey completed
- Survey responses stored

**Main Flow:**

- System retrieves survey data
- System analyzes responses
- System assigns level (Beginner / Intermediate / Advanced)
- System updates user profile

**Postconditions:**

- User level stored in database

**Exception Flow:**

- If survey data missing → system requests survey completion

## 7.3 Use Case 3: Generate Personalized Routine

**Title:** Generate Routine

**Main Actor:** User

**Goal:** Receive personalized diet and exercise plan

**Preconditions:**

- **User level determined**
- **AI service available**

**Main Flow:**

- User requests routine generation
- System retrieves user level
- System sends structured prompt to AI model
- AI generates personalized routine
- System stores routine
- System displays routine

**Postconditions:**

- Routine stored
- Routine visible to user

**Exception Flow:**

- If AI service unavailable → system shows error message

## 7.4 Use Case 4: Receive Reminder

**Title:** Receive Reminder

**Main Actor:** System

**Goal:** Notify user about scheduled activities

**Preconditions:**

- User has active routine
- Reminder schedule configured

**Main Flow:**

- Scheduler checks reminder time
- System generates reminder notification
- User receives notification

**Postconditions:**

- Reminder logged

**Exception Flow:**

- If notification fails → system retries or logs error

## 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).