Software Requirements Specification (SRS)

Gym Management System

---

1. INTRODUCTION

1.1 Purpose

This document specifies the software requirements for a Gym Management System designed to manage gym members, trainers, workout plans, diet plans, attendance tracking, and progress monitoring.

1.2 Scope

The Gym Management System is a web-based application that enables:

- **Members**: Register, login, and view their fitness-related data
- **Trainers**: Manage member information, assign plans, and track progress
- **System**: Maintain comprehensive records of gym operations

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **DBMS**: Database Management System
- **API**: Application Programming Interface
- **JWT**: JSON Web Token
- **CRUD**: Create, Read, Update, Delete
- **REST**: Representational State Transfer

1.4 References

- IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
- REST API Design Guidelines
- Database Design Best Practices

---

2. OVERALL DESCRIPTION

2.1 Product Perspective

The Gym Management System is a standalone application consisting of:

- **Backend**: RESTful API built with Node.js, Express, TypeScript
- **Database**: PostgreSQL managed via Prisma ORM
- **Frontend**: React-based user interface
- **Authentication**: JWT-based token authentication

2.2 Product Functions

The system provides the following major functions:

2.2.1 Member Management

- Member registration and authentication
- Profile management
- View assigned workout and diet plans
- Track attendance history
- Monitor fitness progress

2.2.2 Trainer Management

- Trainer authentication (pre-registered)
- View all gym members
- Assign and update workout plans
- Assign and update diet plans
- Record member attendance
- Update member progress metrics

2.2.3 Data Management

- Secure storage of user credentials
- Management of workout and diet plan details
- Attendance record maintenance
- Progress tracking with metrics

2.3 User Classes and Characteristics

2.3.1 Member

- **Technical Expertise**: Basic computer literacy
- **Frequency of Use**: Daily to weekly

- **Functions**: View personal data, check plans
- **Security Level**: Standard user access

2.3.2 Trainer

- **Technical Expertise**: Moderate computer literacy
- **Frequency of Use**: Daily
- **Functions**: Manage member data, assign plans
- **Security Level**: Elevated user access

2.4 Operating Environment

- **Server**: Node.js runtime environment
- **Database**: PostgreSQL 12 or higher
- **Client**: Modern web browser (Chrome, Firefox, Safari, Edge)
- **Network**: Internet connection required

2.5 Design and Implementation Constraints

- Must use TypeScript for type safety
- Must implement JWT for authentication
- Must use RESTful API architecture
- Database must use Prisma ORM
- Must handle CORS for cross-origin requests

2.6 Assumptions and Dependencies

- Users have internet connectivity
- Database server is always available
- Users have valid email addresses
- Trainers are pre-registered by administrators

---

3. SPECIFIC REQUIREMENTS

3.1 Functional Requirements

3.1.1 Member Registration (FR-1)

- **ID**: FR-1
- **Description**: System shall allow new members to register
- **Input**: Name, Email, Password, Age, Gender, Phone

- **Process**: Validate input, hash password, create account
- **Output**: Member account created, JWT tokens issued
- **Priority**: High

3.1.2 Member Login (FR-2)

- **ID**: FR-2
- **Description**: System shall authenticate members
- **Input**: Email, Password
- **Process**: Verify credentials, generate JWT tokens
- **Output**: Access token, Refresh token
- **Priority**: High

3.1.3 Trainer Login (FR-3)

- **ID**: FR-3
- **Description**: System shall authenticate trainers
- **Input**: Email, Password
- **Process**: Verify credentials, generate JWT tokens
- **Output**: Access token, Refresh token
- **Priority**: High

3.1.4 View Member Profile (FR-4)

- **ID**: FR-4
- **Description**: Members can view their profile
- **Input**: Member JWT token
- **Process**: Verify token, retrieve member data
- **Output**: Member profile information
- **Priority**: Medium

3.1.5 View Workout Plans (FR-5)

- **ID**: FR-5
- **Description**: Members can view assigned workout plans
- **Input**: Member JWT token
- **Process**: Verify token, retrieve workout plans
- **Output**: List of workout plans with trainer details
- **Priority**: High

3.1.6 View Diet Plans (FR-6)

- **ID**: FR-6
- **Description**: Members can view assigned diet plans
- **Input**: Member JWT token
- **Process**: Verify token, retrieve diet plans
- **Output**: List of diet plans with trainer details
- **Priority**: High

3.1.7 View Attendance (FR-7)

- **ID**: FR-7
- **Description**: Members can view attendance history
- **Input**: Member JWT token
- **Process**: Verify token, retrieve attendance records
- **Output**: List of attendance records
- **Priority**: Medium

3.1.8 View Progress (FR-8)

- **ID**: FR-8
- **Description**: Members can view fitness progress
- **Input**: Member JWT token
- **Process**: Verify token, retrieve progress records
- **Output**: Progress data with metrics
- **Priority**: High

3.1.9 View All Members (FR-9)

- **ID**: FR-9
- **Description**: Trainers can view all gym members
- **Input**: Trainer JWT token
- **Process**: Verify token and role, retrieve all members
- **Output**: List of all members
- **Priority**: High

3.1.10 Assign Workout Plan (FR-10)

- **ID**: FR-10
- **Description**: Trainers can assign workout plans to members
- **Input**: Trainer JWT token, Member ID, Plan details
- **Process**: Verify token/role, create workout plan
- **Output**: Workout plan created

- **Priority**: High

3.1.11 Assign Diet Plan (FR-11)

- **ID**: FR-11
- **Description**: Trainers can assign diet plans to members
- **Input**: Trainer JWT token, Member ID, Diet details
- **Process**: Verify token/role, create diet plan
- **Output**: Diet plan created
- **Priority**: High

3.1.12 Record Attendance (FR-12)

- **ID**: FR-12
- **Description**: Trainers can record member attendance
- **Input**: Trainer JWT token, Member ID, Status
- **Process**: Verify token/role, create attendance record
- **Output**: Attendance recorded
- **Priority**: Medium

3.1.13 Update Progress (FR-13)

- **ID**: FR-13
- **Description**: Trainers can update member progress
- **Input**: Trainer JWT token, Member ID, Metrics, Notes
- **Process**: Verify token/role, create progress record
- **Output**: Progress updated
- **Priority**: High

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- **NFR-1**: API response time < 500ms for 95% of requests
- **NFR-2**: Support at least 100 concurrent users
- **NFR-3**: Database query execution < 200ms
- **NFR-4**: Token generation < 100ms

3.2.2 Security Requirements

- **NFR-5**: Passwords must be hashed using bcrypt (10 rounds)
- **NFR-6**: JWT tokens must expire (Access: 15min, Refresh: 7d)

- **NFR-7**: All API endpoints must use HTTPS in production
- **NFR-8**: SQL injection protection via Prisma ORM
- **NFR-9**: Role-based access control enforced

3.2.3 Reliability Requirements

- **NFR-10**: System uptime of 99.5%
- **NFR-11**: Database backups every 24 hours
- **NFR-12**: Error handling for all operations
- **NFR-13**: Transaction rollback on failures

3.2.4 Usability Requirements

- **NFR-14**: Intuitive user interface
- **NFR-15**: Clear error messages
- **NFR-16**: Responsive design for mobile devices
- **NFR-17**: Consistent UI/UX across pages

3.2.5 Maintainability Requirements

- **NFR-18**: TypeScript for type safety
- **NFR-19**: Modular code architecture
- **NFR-20**: Comprehensive code documentation
- **NFR-21**: RESTful API design

3.2.6 Scalability Requirements

- **NFR-22**: Support horizontal scaling
- **NFR-23**: Database connection pooling
- **NFR-24**: Stateless authentication (JWT)
- **NFR-25**: Serverless deployment capability

3.3 Database Requirements

3.3.1 Data Entities

1. **Members**: User profiles and credentials
2. **Trainers**: Trainer profiles and credentials
3. **Workout Plans**: Exercise routines
4. **Diet Plans**: Nutrition plans
5. **Attendances**: Attendance records
6. **Progress**: Fitness metrics

3.3.2 Data Integrity

- Primary keys for all entities
- Foreign key constraints
- Unique constraints on email fields
- Cascade delete for related records
- Not-null constraints on required fields

3.3.3 Data Security

- Encrypted password storage
- Secure database connections
- Regular backups
- Access control

---

4. SYSTEM FEATURES

4.1 Authentication System

Priority: High
Description: Secure user authentication using JWT tokens

Stimulus/Response:

- User provides credentials
- System validates and returns tokens

Functional Requirements:

- Support member and trainer login
- Generate access and refresh tokens
- Token expiration handling

4.2 Member Management

Priority: High
Description: Complete member lifecycle management

Stimulus/Response:

- Member registers/logs in
- System provides personalized dashboard

Functional Requirements:

- Profile management
- View assigned plans
- Track attendance and progress

4.3 Trainer Dashboard

Priority: High
Description: Tools for trainers to manage members

Stimulus/Response:

- Trainer accesses dashboard
- System displays member management tools

Functional Requirements:

- View all members
- Assign/update plans
- Track member progress

---

5. EXTERNAL INTERFACE REQUIREMENTS

5.1 User Interfaces

- Web-based interface
- Responsive design
- Forms for data entry
- Tables for data display
- Navigation menu
- Login/Logout functionality

5.2 Hardware Interfaces

- Standard web server
- Database server
- Network interface

5.3 Software Interfaces

- **Operating System**: Cross-platform (Windows, Linux, macOS)

- **Web Server**: Node.js Express
- **Database**: PostgreSQL
- **ORM**: Prisma
- **Authentication**: JWT

5.4 Communication Interfaces

- **Protocol**: HTTPS/HTTP
- **Data Format**: JSON
- **API Style**: RESTful
- **CORS**: Enabled for cross-origin requests

---

6. OTHER REQUIREMENTS

6.1 Legal Requirements

- Comply with data protection regulations
- User consent for data storage
- Privacy policy implementation

6.2 Ethical Requirements

- Secure handling of personal information
- Transparent data usage
- User data ownership

6.3 Safety Requirements

- Data backup and recovery
- Failover mechanisms
- Error logging and monitoring

---

7. APPENDIX

7.1 API Endpoints Summary

Authentication:

- POST /api/auth/signup - Member registration

- POST /api/auth/login - Member login
- POST /api/auth/trainer/login - Trainer login

Member Routes:

- GET /api/member/profile
- GET /api/member/my/workout
- GET /api/member/my/diet
- GET /api/member/my/attendance
- GET /api/member/my/progress

Trainer Routes:

- GET /api/trainer/members
- PUT /api/trainer/members/:id/workout
- PUT /api/trainer/members/:id/diet
- POST /api/trainer/members/:id/attendance
- PUT /api/trainer/members/:id/progress

7.2 Technology Stack

- **Backend**: Node.js, Express.js, TypeScript
- **Database**: PostgreSQL, Prisma ORM
- **Authentication**: JWT (jsonwebtoken)
- **Security**: bcryptjs for password hashing
- **Validation**: express-validator
- **Frontend**: React, TypeScript, Tailwind CSS
