**Software Requirements Specification**

**Project Title:** Gym Management System

**Version:** 1.0

**Project Code:** [To be assigned]

**Supervisor:** [To be added]

**Co-Supervisor:** [To be added]

**Project Team:**

* Muhammad Yousuf Rehan (22K-4457)
* Azmeer Un Nabi (22K-4235)
* Ateeb Azam (22K-4205)

**Submission Date:** [To be added]

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Name of Person** | **Date** | **Description of Change** |
| 1.0 | Muhammad Yousuf Rehan | 11th May 2025 | Document Created |

**Distribution List**

|  |  |
| --- | --- |
| Name | Role |
| [Supervisor Name] | Supervisor |
| [Co-Supervisor Name] | Co-Supervisor |

**Document Sign-Off**

|  |  |  |
| --- | --- | --- |
| Version | Sign-off Authority | Sign-off Date |
| 1.0 |  |  |

**1. Introduction**

**1.1 Purpose of Document**

This Software Requirements Specification (SRS) describes the requirements for the Gym Management System (GMS). It includes functional and non-functional requirements, system context, constraints, and interactions.

**1.2 Intended Audience**

* Developers and Testers
* Supervisors and Evaluators
* Project Stakeholders

**1.3 Abbreviations**

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| GUI | Graphical User Interface |
| DBMS | Database Management System |
| CRUD | Create, Read, Update, Delete |
| UML | Unified Modeling Language |

**1.4 Document Convention**

* Font: Times New Roman
* Font Size: 12pt normal text, 14pt headings

**2. Overall System Description**

**2.1 Project Background**

Managing daily operations in gyms is challenging with manual registers and unintegrated tools. This project aims to provide a consolidated software system to automate gym tasks.

**2.2 Project Scope**

The system will include:

* Member and Trainer Management
* Attendance Logging
* Payment Records
* Equipment and Package Management

**2.3 Not In Scope**

* Biometric attendance
* Mobile app version
* Real-time sensor-based tracking

**2.4 Project Objectives**

* Automate gym workflows
* Minimize manual errors
* Enable centralized record keeping

**2.5 Stakeholders**

* Gym Owner / Admin
* Trainers
* Members
* Software Developers

**2.6 Operating Environment**

* Platform: Windows OS
* Language: Java (Swing)
* Database: MySQL
* Java Runtime Environment

**2.7 System Constraints**

* Internet not required (offline desktop app)
* Only one admin login session at a time
* Data access only via GUI

**2.8 Assumptions & Dependencies**

* Admin is tech-literate
* JRE and MySQL installed
* Assumes one physical location per system

**3. External Interface Requirements**

**3.1 Hardware Interfaces**

* Keyboard and Mouse for input
* Monitor for output display

**3.2 Software Interfaces**

* Java Runtime Environment
* MySQL Database
* JDBC Connector

**3.3 Communications Interfaces**

* None (Offline system)

**4. Functional Requirements**

**4.1 Functional Hierarchy**

* Admin Login
  + Manage Trainers
  + Manage Members
  + Record Attendance
  + Generate Reports
  + Manage Packages
  + View Equipment

**4.2 Use Cases**

**4.2.1 Use Case: Add Member**

* **Use Case ID:** UC001
* **Actors:** Admin
* **Feature:** Member Management
* **Pre-condition:** Admin is logged in

|  |  |  |
| --- | --- | --- |
| Step# | Action | Software Reaction |
| 1 | Admin selects “Add Member” | Form is displayed |
| 2 | Admin enters member info | Data validated |
| 3 | Admin clicks Save | Member saved to DB |

* **Alternate Scenario:**
  + 2a: If info is incomplete → Show error message
* **Post Conditions:**
  + Member is added to system and shown in list
* **Use Case Cross Referenced:** UC002 (Edit Member), UC003 (Delete Member)

**Test Cases**

**Test Case 1: Admin Login**

* **Input:** Valid credentials
* **Expected Output:** Dashboard opens
* **Result:** Pass/Fail

**Test Case 2: Invalid Login**

* **Input:** Incorrect username/password
* **Expected Output:** Error message
* **Result:** Pass/Fail

**Test Case 3: Add Trainer**

* **Input:** New trainer info
* **Expected Output:** Trainer appears in list
* **Result:** Pass/Fail

**Test Case 4: Update Trainer**

* **Input:** Edited trainer record
* **Expected Output:** Updated info saved
* **Result:** Pass/Fail

**Test Case 5: Delete Trainer**

* **Input:** Click delete on trainer
* **Expected Output:** Trainer removed
* **Result:** Pass/Fail

**Test Case 6: Add Member**

* **Input:** New member details
* **Expected Output:** Member appears in list
* **Result:** Pass/Fail

**Test Case 7: Update Member Info**

* **Input:** Change contact number
* **Expected Output:** Changes reflected
* **Result:** Pass/Fail

**Test Case 8: Delete Member**

* **Input:** Select and delete member
* **Expected Output:** Member removed
* **Result:** Pass/Fail

**Test Case 9: Add Equipment**

* **Input:** Equipment details
* **Expected Output:** Equipment added
* **Result:** Pass/Fail

**Test Case 10: Delete Equipment**

* **Input:** Remove one item
* **Expected Output:** Equipment removed
* **Result:** Pass/Fail

**Test Case 11: Add Package**

* **Input:** Package name, fee, duration
* **Expected Output:** Package added
* **Result:** Pass/Fail

**Test Case 12: Update Package**

* **Input:** Edit fee or name
* **Expected Output:** Changes saved
* **Result:** Pass/Fail

**Test Case 13: View All Members**

* **Input:** Click on member tab
* **Expected Output:** All records shown
* **Result:** Pass/Fail

**Test Case 14: Record Attendance**

* **Input:** Select member, mark present
* **Expected Output:** Attendance saved
* **Result:** Pass/Fail

**Test Case 15: Search Member by Name**

* **Input:** Enter name in search box
* **Expected Output:** Filtered results shown
* **Result:** Pass/Fail

**Test Case 16: Generate Member Report**

* **Input:** Export to PDF
* **Expected Output:** PDF file generated
* **Result:** Pass/Fail

**Test Case 17: Generate Payment Summary**

* **Input:** Select date range
* **Expected Output:** Payment list shown
* **Result:** Pass/Fail

**Test Case 18: Logout Admin**

* **Input:** Click logout
* **Expected Output:** Return to login screen
* **Result:** Pass/Fail

**Test Case 19: Validate Member Email**

* **Input:** Wrong format email
* **Expected Output:** Validation error shown
* **Result:** Pass/Fail

**Test Case 20: Duplicate Username Handling**

* **Input:** Re-enter existing username
* **Expected Output:** Warning message
* **Result:** Pass/Fail

**Test Case 21: Form Reset Button**

* **Input:** Click reset in Add Member form
* **Expected Output:** All fields cleared
* **Result:** Pass/Fail

**Test Case 22: Admin Password Change**

* **Input:** Current and new password
* **Expected Output:** Password updated
* **Result:** Pass/Fail

**Test Case 23: Add Payment Record**

* **Input:** Member, amount, date
* **Expected Output:** Payment saved
* **Result:** Pass/Fail

**Test Case 24: Delete Payment Entry**

* **Input:** Remove one payment row
* **Expected Output:** Entry deleted
* **Result:** Pass/Fail

**Test Case 25: GUI Component Loading**

* **Input:** Open multiple tabs
* **Expected Output:** Components load without crash
* **Result:** Pass/Fail

**5. Non-functional Requirements**

**5.1 Performance Requirements**

* Form transitions under 1 second
* Up to 1000 member records handled smoothly

**5.2 Safety Requirements**

* Automatic logout after 10 minutes inactivity
* Regular DB backups (manual)

**5.3 Security Requirements**

* Login required for access
* Password encryption in DB
* Role-based access (Admin only)

**5.4 User Documentation**

* User manual PDF
* Tooltips and labels on forms
* Installation guide

**6. References**

* Java Oracle Docs: https://docs.oracle.com/en/java/
* MySQL Docs: https://dev.mysql.com/doc/
* UML Standards: https://www.uml-diagrams.org/

**7. Appendices**

* Screenshots
* ERD Diagram
* Test Case Sheet
* SQL Dump
* Deployment Steps

**End of Document**