Software Requirements Specification (SRS)

# Project Title: FreshFold – A Smart Laundry Management System

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to define the requirements and specifications for FreshFold, a web-based Laundry Management System tailored for college campuses. It facilitates students, staff, and administrators by automating the laundry process, improving efficiency, and enhancing service transparency.

## 1.2 Scope

The system includes:  
- Online laundry request submission  
- Real-time laundry status tracking  
- Role-based dashboards (Student, Staff, Admin)  
- Reporting of damaged or missing items  
- Notification system for laundry progress updates  
- Admin-level reports and user management tools  
  
The system does not include:  
- Physical pickup or delivery logistics  
- Integration with third-party payment gateways  
- Native mobile apps (currently web-based only)

## 1.3 Definitions, Acronyms, and Abbreviations

- UI: User Interface  
- DBMS: Database Management System  
- SRS: Software Requirements Specification  
- CRUD: Create, Read, Update, Delete

# 2. System Analysis

## 2.1 Problem Definition

Manual laundry tracking is inefficient and lacks visibility. Students face delays and confusion, and staff lack tools to manage workloads or respond effectively to complaints. Administrators cannot access meaningful data for decision-making. FreshFold solves these with a centralized, digital platform.

## 2.2 Advantages of Proposed System

- Eliminates manual errors  
- Tracks laundry in real time  
- Enhances user satisfaction through transparency  
- Improves administrative control  
- Saves time and paper

## 2.3 Feasibility Analysis

## a) Technical Feasibility

## PHP 8+, MySQL, HTML, JavaScript, CSS

## Compatible with XAMPP/WAMP

## Easily scalable and maintained

## b) Operational Feasibility

## Clean UI for all user roles

## Minimal training needed

## Mobile and desktop responsive

## c) Economic Feasibility

## Open-source tools used

## No licensing costs

## Reduces paper and human resource costs

## 2.4 Recommended Implementation

Agile development methodology will be used. XAMPP is the local server environment. Git for version control. The project will use responsive frontend design for cross-device compatibility.

## 2.5 Techniques Used

- Role-based authentication  
- Email notifications  
- Encrypted passwords  
- SQL JOINs and indexing for optimized data retrieval

# 3. Software Requirements Specification

## 3.1 Introduction

This section details the required functionalities and performance criteria for FreshFold. It guides development, testing, and deployment.

## 3.2 Purpose

To create a robust and secure platform to manage student laundry requests, ensure real-time tracking, and improve interaction between students, staff, and admins.

## 3.3 Scope

FreshFold will:  
- Support three roles: Student, Staff, Admin  
- Be accessible via modern browsers  
- Allow users to track laundry progress  
- Handle complaint submission and resolution

## 3.4 Technical Overview

- Frontend: HTML5, CSS3, JavaScript  
- Backend: PHP 8+  
- Database: MySQL 8  
- Server: Apache via XAMPP/WAMP  
- Notifications: PHP mail or SMTP  
- Browsers Supported: Chrome, Firefox, Edge, Safari

# 4. External Interface Requirements

## 4.1 User Interfaces

- Student: Submit laundry, track status, receive alerts, report issues  
- Staff: Manage bags, update statuses, resolve complaints  
- Admin: Manage users, monitor actions, generate reports

## 4.2 Software Interfaces

- PHP ↔ MySQL for database operations  
- Email API for notifications  
- Session management and form validation

## 4.3 Hardware Interfaces

Runs on any browser-supported device (laptop, mobile, tablet)

## 4.4 Communication Interfaces

- HTTPS for secure transmission  
- Email alerts through SMTP

# 5. Functional Requirements

|  |  |  |
| --- | --- | --- |
| FR ID | Functionality | Actor |
| FR-01 | User Registration and Login | All Roles |
| FR-02 | Submit Laundry Form | Student |
| FR-03 | Track Bag Status | Student |
| FR-04 | Receive Notifications | Student |
| FR-05 | Report Missing/Damaged Items | Student |
| FR-06 | Manage Bags & Update Status | Staff |
| FR-07 | Respond to Complaints | Staff |
| FR-08 | Manage Users | Admin |
| FR-09 | Generate Reports | Admin |

# 6. Non-Functional Requirements

|  |  |
| --- | --- |
| NFR | Requirement |
| Usability | Clean, intuitive UI accessible across devices |
| Performance | Core tasks should respond in ≤ 2 seconds |
| Scalability | Should support expansion across hostels/campuses |
| Security | Role-based access, encrypted credentials, secure sessions |
| Availability | 99% uptime with regular backups |
| Portability | Browser-based access via any platform |
| Maintainability | Modular code for ease of testing, debugging, and future enhancements |

# 7. Database Design

# 7.1 Key Tables & Attributes

From freshfold\_database.sql:

* **users**: user\_id, username, email, password\_hash, user\_type, hostel\_block, room\_number
* **laundry\_requests**: request\_id, student\_id, bag\_number, pickup\_date, status, total\_items
* **laundry\_items**: item\_id, request\_id, item\_type, quantity, status
* **complaints**: complaint\_id, student\_id, complaint\_type, description, status
* **notifications**: notification\_id, user\_id, title, message, type
* **settings**: setting\_key, setting\_value

**7.2 Relationships**

* One student → many laundry requests
* One request → many items
* One student → many complaints
* One complaint → handled by one staff
* One user → many notifications

## 7.3 Relational Schema

### Users Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| user\_id | INT (PK, AUTO\_INCREMENT) | Unique ID for each user |
| username | VARCHAR(50) | Unique login username |
| Email | VARCHAR(100) | User's email address (unique) |
| password\_hash | VARCHAR(255) | Hashed password |
| full\_name | VARCHAR(100) | Full name of the user |
| Phone | VARCHAR(15) | Contact number |
| user\_type | ENUM('student', 'staff', 'admin') | Role of the user |
| hostel\_block | VARCHAR(10) | Student's hostel block |
| room\_number | VARCHAR(10) | Student's room number |
| created\_at | TIMESTAMP | Account creation timestamp |
| updated\_at | TIMESTAMP | Last updated timestamp |
| is\_active | BOOLEAN | Account status (active/inactive) |

### Laundry Requests Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| request\_id | INT (PK, AUTO\_INCREMENT) | Unique ID for each request |
| student\_id | INT (FK → Users.user\_id) | ID of the student |
| bag\_number | VARCHAR(20) | Unique bag number |
| pickup\_date | DATE | Requested pickup date |
| expected\_delivery | DATE | Expected delivery date |
| special\_instructions | TEXT | Any special notes |
| total\_items | INT | Total number of items |
| Status | ENUM(...) | Current status of laundry |
| payment\_status | ENUM(...) | Payment status |
| payment\_amount | DECIMAL(10,2) | Total payment amount |
| created\_at | TIMESTAMP | Record creation time |
| updated\_at | TIMESTAMP | Last update time |

### Laundry Items Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| item\_id | INT (PK, AUTO\_INCREMENT) | Unique ID for each item |
| request\_id | INT (FK → Laundry\_Requests.request\_id) | Associated request |
| item\_type | VARCHAR(50) | Type of clothing item |
| item\_description | VARCHAR(200) | Item description |
| Quantity | INT | Quantity of the item |
| price\_per\_item | DECIMAL(8,2) | Price per item |
| Status | ENUM(...) | Item status |

### Status History Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| history\_id | INT (PK, AUTO\_INCREMENT) | Unique ID for status change |
| request\_id | INT (FK → Laundry\_Requests.request\_id) | Related request |
| old\_status | VARCHAR(20) | Previous status |
| new\_status | VARCHAR(20) | Updated status |
| updated\_by | INT (FK → Users.user\_id) | User who updated |
| Remarks | TEXT | Remarks on the update |
| updated\_at | TIMESTAMP | Update time |

### Complaints Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| complaint\_id | INT (PK, AUTO\_INCREMENT) | Unique complaint ID |
| student\_id | INT (FK → Users.user\_id) | Complaint by student |
| request\_id | INT (FK → Laundry\_Requests.request\_id) | Related request |
| complaint\_type | ENUM(...) | Type of complaint |
| description | TEXT | Complaint details |
| Status | ENUM(...) | Complaint resolution status |
| Priority | ENUM(...) | Urgency of complaint |
| assigned\_to | INT (FK → Users.user\_id) | Staff assigned |
| resolution | TEXT | Resolution notes |
| created\_at | TIMESTAMP | When raised |
| resolved\_at | TIMESTAMP | When resolved |

### Notifications Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| notification\_id | INT (PK, AUTO\_INCREMENT) | Unique notification ID |
| user\_id | INT (FK → Users.user\_id) | Recipient user |
| Title | VARCHAR(100) | Notification title |
| Message | TEXT | Notification content |
| Type | ENUM(...) | Notification type |
| is\_read | BOOLEAN | Read status |
| created\_at | TIMESTAMP | Created timestamp |

### Settings Table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| setting\_id | INT (PK, AUTO\_INCREMENT) | Unique setting ID |
| setting\_key | VARCHAR(100) | Setting identifier |
| setting\_value | TEXT | Value of the setting |
| description | TEXT | Setting explanation |
| updated\_at | TIMESTAMP | Last update time |

**7.4 Normalization**

All tables conform to **3NF**:

* 1NF: Atomic fields
* 2NF: No partial dependencies
* 3NF: No transitive dependencies