# **Software Requirement Specification (SRS)**

**Project Name:** University Research Management System (URMS)  
 **Platform:** Oracle APEX with Oracle Database  
 **Prepared for:** Learning & Skill Development

## **1. Introduction**

### **1.1 Purpose**

The URMS is designed to manage and track research activities within a university. It provides a centralized platform for students, faculty, and administrators to handle research projects, publications, funding, and lab resources. The system is aimed at improving collaboration, automating workflows, and generating real-time insights.

### **1.2 Scope**

* Manage research projects, funding, labs, publications, and participants.
* Track student and faculty contributions.
* Automate approvals for project assignments and funding.
* Generate dashboards and reports for research metrics.
* Role-based access for students, faculty, and admins.

### **1.3 Definitions / Acronyms**

* **URMS** – University Research Management System
* **APEX** – Oracle Application Express
* **DB** – Database
* **CRUD** – Create, Read, Update, Delete operations

## **2. Overall Description**

### **2.1 Product Perspective**

URMS is a standalone system but can integrate with university authentication systems. It is web-based using Oracle APEX.

### **2.2 User Classes and Characteristics**

|  |  |  |
| --- | --- | --- |
| **User Type** | **Access** | **Characteristics** |
| **Admin** | Full CRUD on all tables, dashboard, reports | University admin staff, tech-savvy |
| **Faculty** | CRUD on own projects, approve students/funding, view reports | Professors/researchers, moderate DB knowledge |
| **Student** | View projects, apply for participation, submit milestones | Students, beginner DB users |

### **2.3 Operating Environment**

* Oracle Database 19c+
* Oracle APEX 22+
* Modern web browsers (Chrome, Firefox, Edge)
* Optional email notifications

### **2.4 Design Constraints**

* Must implement **role-based access control**
* Must normalize tables (3NF recommended)
* Use **stored procedures** for sensitive operations and sequences for auto-ID generation.
* Use **triggers** to log changes or enforce constraints

### **2.5 Assumptions**

* Users have university credentials
* Projects and funding details will be entered by faculty/admin
* Labs have limited capacity

## **3. System Features / Functional Requirements**

### **3.1 User Management**

* Admin can create/update/delete users (students/faculty).
* Users log in with role-based access.
* Passwords are hashed (e.g., Passwords are stored as salted hashes in pass (RAW(64)) with salt in pass\_salt (RAW(32)).).

### **3.2 Project Management**

* Faculty can create/update/delete projects.
* Projects have title, description, start/end dates, lab assignment, status.
* Students can apply to join projects (Optional).
* Admin approves project creation requests if needed (Optional).

### **3.3 Student & Faculty Assignments**

* Many-to-many mapping: students → projects, faculty → projects.
* Stored procedures to auto-assign IDs, validate conflicts, or cap lab capacity.
* Notifications to students/faculty on assignment approval/rejection.

### **3.4 Funding Management**

* Track funding source, amount, and allocated projects.
* Admin or faculty can approve/reject funding requests.
* Trigger to prevent exceeding total project budget.

### **3.5 Publications Management**

* Track publications per project: title, authors, journal/conference, date.
* Many-to-many mapping: students/faculty ↔ publications.
* Generate reports by author, project, or year.

### **3.6 Labs & Resources**

* Lab info: name, capacity, availability.
* Trigger to ensure lab capacity isn’t exceeded.

### **3.7 Dashboards & Reports**

* Interactive reports for projects, funding, publications.
* Charts showing funding distribution, project progress, student involvement.
* Export reports to PDF/Excel.

### **3.8 Audit & Logs**

* Trigger-based audit log for insert/update/delete operations on key tables.
* Maintain user ID, timestamp, operation type, old/new values.

## **4. Non-functional Requirements**

### **4.1 Performance**

* Must handle at least 500 concurrent users.
* Reports should load within 5 seconds.

### **4.2 Security**

* Role-based access enforced in APEX and DB.
* Data encryption for passwords.
* Audit trails for sensitive operations.

### **4.3 Usability**

* Intuitive APEX forms & dashboards.
* Mobile-friendly.

### **4.4 Maintainability**

* Modular stored procedures and triggers.
* Well-documented DB schema.

### **4.5 Reliability**

* Backup & restore policies in place (Oracle RMAN optional).
* Proper validation to prevent data inconsistencies.

## **5. Database Design**

### **5.1 ER Diagram (Textual Version)**

* USERS: user\_id (PK), name, email, role, password, created\_at
* PROJECTS: project\_id (PK), title, description, start\_date, end\_date, lab\_id (FK), status
* STUDENT\_PROJECTS: student\_id (FK), project\_id (FK), role\_in\_project, approved\_flag
* FACULTY\_PROJECTS: faculty\_id (FK), project\_id (FK)
* FUNDING: funding\_id (PK), project\_id (FK), source, amount, approved\_flag
* PUBLICATIONS: pub\_id (PK), title, project\_id (FK), publication\_date
* PUB\_AUTHORS: pub\_id (FK), user\_id (FK), role (author/co-author)
* LABS: lab\_id (PK), name, capacity, availability\_status
* AUDIT\_LOGS (OPTIONAL) : log\_id (PK), table\_name, operation\_type, old\_values, new\_values, user\_id, timestamp

## **6. Triggers & Stored Procedures**

* Auto-generate project\_id, pub\_id.
* Validate lab capacity on project assignment.
* Approval workflow triggers for funding and student assignments.
* Audit log triggers for insert/update/delete.

## **7. Reports**

* Student-wise project participation.
* Funding distribution per department/project.
* Publication count by author/year.
* Lab usage report.

## **8. Potential Advanced Features**

* Milestone tracking & notifications.
* Project progress charts with dynamic filtering.
* Role-based dynamic dashboards (different widgets for students, faculty, admins).