Software Requirements Specification (SRS) for Faculty Data Management System (FDMS)

Version: 1.0

Prepared by: Priyanka, Suma, Abhishek Bhan , Phanindra

Date: 26th February 2025

1. Introduction

1.1 Purpose

- The Faculty Data Management System (FDMS) is a web-based system that provides a complete solution for the collection, maintenance, and reporting of faculty data in institutions of higher learning. It particularly addresses the needs of NAAC (National Assessment and Accreditation Council) and NBA (National Board of Accreditation), with the aim of ensuring institutions maintain best practices in faculty development, research, and teaching.

1.2 Document Conventions

- Main Body: Arial, 12 pt
- Headings: Arial, 16 pt bold
- Subheadings: Arial, 14 pt bold

1.3 Meant Audience and Reading Recommendations

- This report is meant for:
- Developers (Backend, Frontend)
- Database Administrators
- College Administration
- Faculty Members

1.4 Product Scope

- FDMS optimizes faculty administration by effectively tracking certifications, course assignments, and performance data. It reduces manual record-keeping, increases the efficiency of operations, and helps in accreditation adherence by automating essential faculty information processes.

1.5 References

- NAAC/NBA Accreditation Guidelines
- Institutional Policies regarding Faculty Performance

2. General Description

2.1 Product Point of View

- FDMS includes three major user portals:
- HOD Portal: Faculty department-specific information and approval processes.
- Administrator Portal: Faculty records centralized management across departments.
- Faculty Portal: Personal profile management, certification renewal, and performance monitoring.

2.2 **Product Functions**

- FDMS offers the following core functions:
- User Authentication: Secure login process for HODs, administrators, and faculty.
- Workload and Performance Tracking: Track teaching loads and faculty performance data.
- Automated Reporting: Produce reports for accreditation purposes and internal assessments.
- Role-Based Access Control: Provide role-based secure access.

2.3 User Classes and Characteristics

- FDMS addresses various user categories with unique needs and rights:
- Administrators: Control system-level operations, e.g., roles and permissions.
- HOD's: View department-level faculty details, authorize changes, and make reports.
- Faculty Members: Update own information, post certifications, and monitor load.
- Accreditation Committees: Employ automated report-based compliance validation.

2.4 Operating Environment

- FDMS is designed to run in a web-based system with the following specifications:
- Server: Node.js backend on port 3000.
- Database: MONGODB relational database for organized faculty data storage.
- Frontend: HTML,CSS,JavaScript, React.js for a dynamic and user-friendly interface.
- Operating System: Supports Windows and macOS.
- Browsers Supported: Chrome, Firefox, Edge.

2.5 Design and Implementation Constraints

- Development and deployment of FDMS needs to take the following constraints into account:
- Security Requirements: Role-based access control and encrypted data storage.
- Compliance Standards: Should be in line with institutional and accreditation guidelines.
- Integration with Existing Systems: Integration with the college database.
- Scalability: Scaling to accommodate growing faculty data over time.
- Technology Stack: Should be Node.js for the backend and React.js for the frontend.

2.6 User Documentation

- The following documentation will be available:
- User Manual: In-depth guide for administrators, HODs, and teachers.
- Online Help: Built-in help sections in the application.
- API Documentation: Technical documentation for database integration and system extensions.

2.7 Assumptions and Dependencies

- Successful FDMS implementation relies on:
- Availability of Institutional Database Access: Needed for real-time retrieval of faculty data.
- Network Stability: Since it is a web application, stable internet access is needed.
- Institutional IT Support: Needed for deployment, upkeep, and user training.
- Third-party Libraries and Frameworks: Dependencies on Node.js and database management systems.

3. External Interface Requirements

3.1 User Interfaces

- FDMS features a friendly and responsive Graphical User Interface (GUI) intended to allow effortless interaction with the system by the users. Some of the primary UI features are:
- Dashboard: Offers quick insight into faculty records and major notifications.
- Navigation Bar: Comprises access to faculty profiles, tracking of workload, and reporting modules.
- Standardized Forms: For certification, qualification, and research activity updates.
- Error Handling: Descriptive and informative error messages with tooltips and validation hints.
- Accessibility Features: Adherence to accessibility standards for ease of use.

3.2 Hardware Interfaces

- FDMS is a web application and does not need special hardware other than common computing equipment. It is, however, optimized for:
- Desktop and Laptop Computers with a current web browser.
- Tablets and Mobile Devices for on-the-move faculty management.
- Servers using Node.js for backend processes and SQL databases for storage.
- Network Infrastructure to facilitate smooth communication between the database and users.

3.3 Software Interfaces

- FDMS supports integration with several software components, such as:
- Database: Relational SQL database for storing structured data.
- Backend Framework: Node.js for API development and authentication.
- Frontend Framework: React.js for an interactive and user-friendly UI.
- Operating Systems: Supports Windows, Linux, and macOS.

3.4 Communications Interfaces

- FDMS has secure and effective communication protocols, such as:
- HTTP/HTTPS: Provides encrypted and secure web-based interactions.
- RESTful APIs: Supports smooth data exchange between frontend and backend.
- Email Notifications: For profile update, certification expiration, and approval alerts.
- Role-Based Access Control (RBAC): Provides secure user access based on role.

4. System Features

4.1 Faculty Profile Management

4.1.1 Description and Priority

- This functionality enables institutions to update and keep faculty profiles, such as personal information, qualifications, research work, and administrative positions. Priority: High

4.1.2 Stimulus/Response Sequences

- User logs in → System authenticates credentials and shows the dashboard.
- User chooses 'Faculty Profile' → System retrieves and shows stored faculty information.
- User saves profile information → System checks and saves changes.
- Error Handling: Shows proper messages for invalid or incomplete information.

4.1.3 Functional Requirements

- REQ-1: The system will permit users to edit personal and professional information.
- REQ-2: The system will check input data prior to submission.
- REQ-3: The system will support role-based access control for changing profiles.

4.2 Workload Management for Faculty

4.2.1 Description and Priority

- Manages and assigns teaching loads, research assignments, and administrative duties. Priority: High

4.2.2 Stimulus/Response Sequences

- Administrator assigns workload → System stores workload information.
- Faculty views assigned workload → System fetches workload details.
- Faculty requests changes → System directs request for approval.

4.3 Accreditation Compliance Reporting

4.3.1 Description and Priority

- Creates reports needed for NAAC/NBA accreditation compliance. Priority: High

4.3.2 Stimulus/Response Sequences

- User chooses 'Generate Report' → System generates and shows report preview.
- User downloads report → System exports report in PDF/Excel format.
- User filters report parameters → System adjusts report output.

4.3.3 Functional Requirements

- REQ-1: The system will create reports from faculty performance measures.
- REQ-2: The system should provide filtering and report customization.
- REQ-3: The system should have export capabilities in various formats (PDF, Excel).

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system should accommodate multiple users' access without a degradation of performance.
- The system should provide response times less than 2 seconds for data retrieval operations.
- The system should process large datasets efficiently with minimal latency.

5.2 Safety Requirements

- Data integrity will be ensured to avoid loss of faculty records.
- Backups will be carried out regularly to avoid accidental data loss.
- Unauthorised alterations to faculty data will be tightly controlled.

5.3 Security Requirements

- Login shall be enforced with multi-factor authentication (MFA).
- User data will be encrypted with AES-256 encryption.
- Secure role-based access control will limit unauthorised access.

5.4 Software Quality Attributes

- Reliability: The system will be available 99.9% of the time.
- Scalability: The architecture should be scalable to accommodate more users and features.
- Usability: The UI will be easy to use and need little training for the user.

5.5 Business Rules

- Authorized users only can modify faculty records.
- Adjustments to faculty workload must be authorized by an administrator.
- Reports produced must comply with accreditation compliance reporting.

6. Other Requirements

6.1 Database Requirements

- The system will implement a SQL-based relational database (e.g., PostgreSQL or MySQL).
- The database will hold faculty information, workload assignments, research records, and accreditation reports.
- Data redundancy should be reduced via effective indexing and normalization methods.
- The system must provide automated nightly backups to safeguard against data loss.
- Role-based access control should be implemented on the database level to secure confidential data.

6.2 Requirements for Internationalization

- The system must be supported in multiple languages for wider user accessibility (e.g., regional languages, English).
- Date, time, and number formats will be user-dependent.
- Character set encoding should use UTF-8 for easy-to-represent text.

6.3 Legal and Compliance Needs

- The system must be in accordance with data privacy laws like GDPR (General Data Protection Regulation) and India's Data Protection Legislations.
- Reports generated by accreditation reports must follow NAAC/NBA documentation guidelines.
- The system will have audit logs to record changes in faculty records for transparency.
- The terms and conditions need to be accepted by users prior to accessing the system.

6.4 Reuse and Extensibility

- The system will be built with modular architecture, which will allow easy incorporation of new features.
- APIs will be exposed for integration with third-party systems (e.g., HR management systems).
- The codebase will adhere to industry-standard best practices for maintainability.

Appendix A: Glossary

- FDMS – Faculty Data Management System
- NAAC – National Assessment and Accreditation Council
- NBA – National Board of Accreditation
- GUI – Graphical User Interface
- RBAC – Role-Based Access Control
- SQL – Structured Query Language
- MFA – Multi-Factor Authentication

- GDPR – General Data Protection Regulation

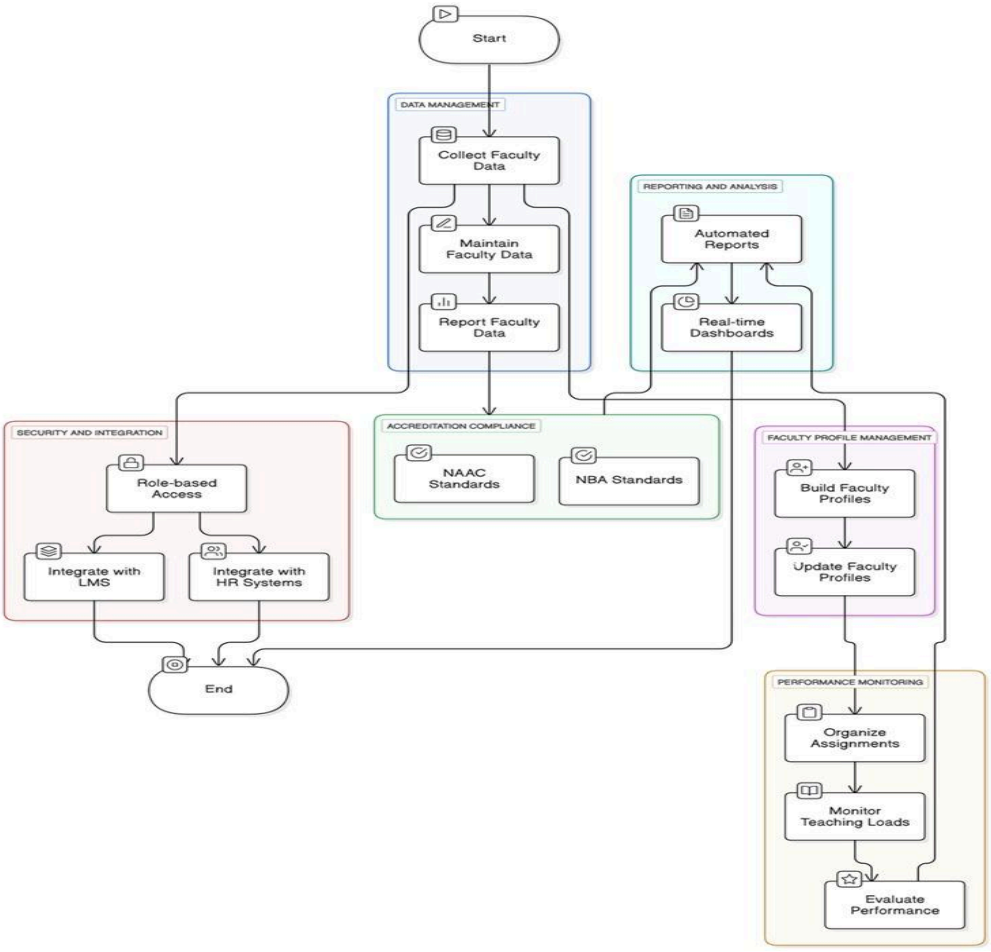
Appendix B: Analysis Models

- (Incorporate models that are appropriate, e.g., Entity-Relationship Diagrams (ERD), Data Flow Diagrams (DFD), and Class Diagrams.)

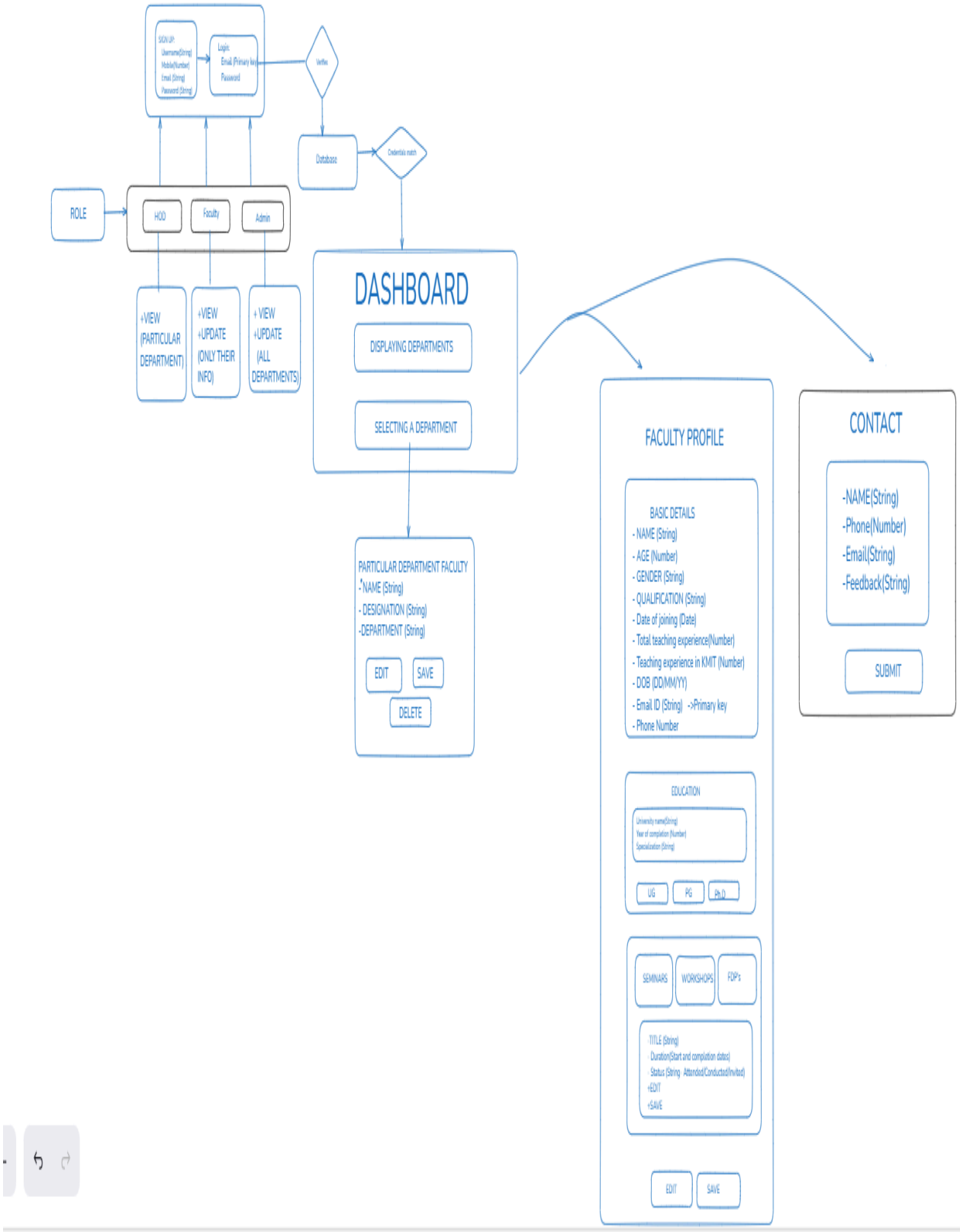
Appendix C: To Be Determined (TBD) List

- Selection of particular third-party authentication providers (e.g., Google, Microsoft).
- Selection of data visualization libraries for tracking faculty performance.
- Final determination of export formats supported in reports (e.g., CSV, JSON).

ARCHITECTURE DIAGRAM



CLASS DIAGRAM



DATABASE DESIGN SCHEMAS

FACULTY PROFILE

```
const profileSchema = new mongoose.Schema({
  name: { type: String, required: true },
  age: { type: Number },
  gender: { type: String, enum: ["Male", "Female", "Other"] },
  dateOfBirth: { type: Date },
  dateOfJoining: { type: Date },

  designation: { type: String },
  qualification: { type: String },
  jntuID: { type: String, unique: true },

  email: { type: String, unique: true, required: true },
  phone: { type: String },
  officeAddress: { type: String },
  imageUrl: { type: String },

  department: { type: String }, // Primary Department
  departments: [String], // List of departments

  subject1: { type: String },
  subject2: { type: String },
  experience: { type: Number },

  researchInterests: [String],

  ugSpecialization: { type: String },
  ugYearOfCompletion: { type: Number },
  ugUniversity: { type: String },

  pgSpecialization: { type: String },
  pgYearOfCompletion: { type: Number },
  pgUniversity: { type: String }
});
```

Feedback

```
// Feedback Schema
const contactSchema = new mongoose.Schema({
  fullName: String,
  email: String,
  phone: String,
  feedback: String,
  date: { type: Date, default: Date.now }
});

const Contact = mongoose.model("Contact", contactSchema);
```

Users(HOD/ADMIN/FACULTY)

```
// User Schema
const userSchema = new mongoose.Schema({
  name: String,
  phone: String,
  email: String,
  password: String,
  role: String, // Admin, Faculty, HoD
});
```

RESEARCHES

```
const ResearchSchema = new mongoose.Schema({
    faculty: { type: mongoose.Schema.Types.ObjectId, ref: "Faculty" }, // Reference to Faculty
    title: { type: String, required: true },
    publicationType: { type: String, enum: ["Journal", "Conference"], required: true },
    yearPublished: { type: Number, required: true },
    doi: { type: String }
});
```