

# Software Requirements Specification

for

## Namal Complaint Management System

Version 1.0

Approved

Prepared by:

Dev Wizard Team

Sundeep Kumar	NUM-BSCS-2024-75
Rehan Tariq	NUM-BSCS-2024-66
Aimen Shafiq	NUM-BSCS-2024-04

Department of Computer Science  
Namal University, Mianwali

CSC-225 – Software Engineering

January 18, 2026

# Contents

<b>Revision History</b>	<b>3</b>
<b>1 Introduction</b>	<b>4</b>
1.1 Purpose . . . . .	4
1.2 Scope . . . . .	4
1.3 Definitions, Acronyms, and Abbreviations . . . . .	5
1.4 References . . . . .	5
1.5 Overview . . . . .	6
<b>2 Overall Description</b>	<b>7</b>
2.1 Product Perspective . . . . .	7
2.1.1 System Interfaces . . . . .	7
2.1.2 User Interfaces . . . . .	7
2.1.3 Hardware Interfaces . . . . .	7
2.1.4 Software Interfaces . . . . .	8
2.1.5 Communications Interfaces . . . . .	8
2.1.6 Memory Constraints . . . . .	8
2.2 Product Functions . . . . .	8
2.3 User Characteristics . . . . .	9
2.3.1 End Users (Students, Faculty, Staff) . . . . .	9
2.3.2 Maintenance Staff . . . . .	9
2.3.3 Administrators/Supervisors . . . . .	10
2.4 Constraints . . . . .	10
2.4.1 Regulatory Policies . . . . .	10
2.4.2 Hardware Limitations . . . . .	10
2.4.3 Interface Requirements . . . . .	10
2.4.4 Development Constraints . . . . .	10
2.4.5 Security Constraints . . . . .	10
2.5 Assumptions and Dependencies . . . . .	11
2.5.1 Assumptions . . . . .	11
2.5.2 Dependencies . . . . .	11
<b>3 Specific Requirements</b>	<b>12</b>
3.1 External Interface Requirements . . . . .	12
3.1.1 User Interfaces . . . . .	12
3.1.2 Hardware Interfaces . . . . .	14
3.1.3 Software Interfaces . . . . .	15
3.1.4 Communications Interfaces . . . . .	16
3.2 Functional Requirements . . . . .	16
3.2.1 User Authentication and Authorization . . . . .	16
3.2.2 Complaint Management . . . . .	18
3.2.3 Administrative Functions . . . . .	19
3.2.4 Maintenance Staff Functions . . . . .	21
3.2.5 Notification System . . . . .	21
3.2.6 Reporting and Analytics . . . . .	22
3.3 Performance Requirements . . . . .	23
3.4 Design Constraints . . . . .	24

---

3.4.1	Standards Compliance . . . . .	24
3.4.2	Hardware Limitations . . . . .	24
3.4.3	Technology Constraints . . . . .	25
3.5	Software System Attributes . . . . .	25
3.5.1	Reliability . . . . .	25
3.5.2	Security . . . . .	26
3.5.3	Maintainability . . . . .	27
3.5.4	Portability . . . . .	28
3.5.5	Usability . . . . .	28
3.6	Other Requirements . . . . .	29
3.6.1	Database Requirements . . . . .	29
3.6.2	Operational Requirements . . . . .	30
3.6.3	Legal and Compliance Requirements . . . . .	31
<b>A</b>	<b>Use Case Diagram</b>	<b>32</b>
<b>B</b>	<b>Context Diagram</b>	<b>33</b>
<b>C</b>	<b>Sample Screen Mockups</b>	<b>33</b>
<b>D</b>	<b>Traceability Matrix</b>	<b>34</b>
<b>E</b>	<b>Glossary of Terms</b>	<b>35</b>

## Revision History

Date	Version	Description	Author
January 18, 2026	1.0	Initial SRS Document	Dev Wizard Team

# 1 Introduction

## 1.1 Purpose

This Software Requirements Specification (SRS) document provides a complete description of the Namal Complaint Management System (NCMMS). The document specifies all functional and non-functional requirements for the system and is intended for:

- Development team members who will design and implement the system
- Project managers who will track project progress
- Testers who will verify and validate the system
- Requirement Provider (university administration) who will review and approve requirements
- Maintenance personnel who will maintain the system after deployment

## 1.2 Scope

The Namal Complaint Management and Maintenance System (NCMMS) is a web-based application designed to streamline the maintenance complaint management process at Namal University. The system will:

- Replace the current informal, verbal complaint reporting system
- Provide a centralized platform for complaint submission and tracking
- Enable efficient assignment and monitoring of maintenance tasks
- Improve transparency and accountability in complaint resolution
- Generate reports and analytics on maintenance performance

### **Benefits:**

- Reduced complaint resolution time through systematic tracking
- Enhanced user satisfaction through real-time status updates
- Improved resource allocation and workload management
- Data-driven insights for maintenance planning
- Complete audit trail of all maintenance activities

### **System will:**

- Allow users (students, faculty, staff) to submit categorized complaints
- Enable administrators to assign complaints to maintenance staff
- Provide real-time complaint status tracking

- Generate summary reports and performance metrics
- Send notifications for complaint updates

**System will not:**

- Handle financial transactions or budget management
- Manage inventory of maintenance materials
- Replace face-to-face emergency reporting procedures
- Provide external vendor management capabilities

### 1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
NCMMS	Namal Complaint Management and maintenance System
RP	Requirement Provider - University administration representative
JWT	JSON Web Token - Authentication mechanism
MERN	MongoDB, Express.js, React, Node.js - Technology stack
Admin	System administrator with full access privileges
Complaint	A reported maintenance issue requiring attention
Ticket	Unique identifier assigned to each complaint
Status	Current state of complaint (Open, In Progress, Resolved, Closed)
Priority	Urgency level of complaint (Low, Medium, High, Critical)
Category	Type of maintenance issue (Electrical, Plumbing, IT, Furniture, Civil, HVAC)
End User	Students, faculty, or staff members who submit complaints
Maintenance Staff	Personnel assigned to resolve complaints
Dashboard	Main interface displaying complaint statistics and information
SRS	Software Requirements Specification
API	Application Programming Interface

### 1.4 References

1. IEEE Std 830-1984, IEEE Guide to Software Requirements Specifications
2. React Documentation. (2024). Retrieved from <https://react.dev/>
3. Express.js Foundation. (2024). Retrieved from <https://expressjs.com/>

4. MongoDB, Inc. (2024). MongoDB Official Documentation. Retrieved from <https://www.mongodb.com/docs>
5. Cloudinary Ltd. (2024). API Documentation. Retrieved from <https://cloudinary.com/documentation>
6. Atlassian. (2024). Kanban Methodology Overview. Retrieved from <https://www.atlassian.com/agile/kanban>
7. NCMMS Project Proposal, Dev Wizard Team, November 2025
8. IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology

## 1.5 Overview

The remainder of this document is organized as follows:

- **Section 2: Overall Description** - Provides context for the system, including product perspective, functions, user characteristics, constraints, and assumptions
- **Section 3: Specific Requirements** - Details all functional and non-functional requirements, external interfaces, and system features
- **Appendix A: Use Case Diagram** - Visual representation of system interactions
- **Appendix B: Context Diagram** - System boundary and external entities

## 2 Overall Description

### 2.1 Product Perspective

The Namal Complaint Management System (NCMMS) is a new, self-contained web-based application designed specifically for Namal University. It operates independently and does not integrate with existing university systems in its initial release.

#### 2.1.1 System Interfaces

The system consists of the following major components:

- **Web Application Interface** - Browser-based user interface accessible via standard web browsers
- **Backend Server** - Node.js/Express.js server handling business logic
- **Database System** - MongoDB for data persistence
- **Cloud Storage** - Cloudinary for image/file storage
- **Authentication Module** - JWT-based security system

#### 2.1.2 User Interfaces

The system will provide responsive web interfaces optimized for:

- Desktop browsers (Chrome, Firefox, Safari, Edge)
- Tablet devices
- Mobile devices (smartphones)

Key interface components include:

- Login/Registration screens
- User dashboard
- Complaint submission form with image upload capability
- Complaint tracking interface
- Administrative panel for complaint management
- Maintenance staff work queue
- Reporting and analytics dashboard

#### 2.1.3 Hardware Interfaces

- **Client Side:** Any device with a modern web browser and internet connection
- **Server Side:** Web server capable of running Node.js applications
- **Storage:** Cloud storage for images and attachments via Cloudinary

### 2.1.4 Software Interfaces

- **Operating System:** Platform-independent (runs on any OS supporting Node.js)
- **Database:** MongoDB version 5.0 or higher
- **Web Framework:** React/Next.js for frontend, Express.js for backend
- **Runtime Environment:** Node.js version 18.x or higher
- **Cloud Services:** Cloudinary API for media management

### 2.1.5 Communications Interfaces

- HTTP/HTTPS protocols for client-server communication
- RESTful API architecture
- JSON data format for API requests and responses
- WebSocket connection for real-time notifications (optional)
- Email notifications via SMTP

### 2.1.6 Memory Constraints

- Maximum file upload size: 5 MB per attachment
- Server memory: Minimum 2 GB RAM recommended
- Database storage: Scalable based on complaint volume

## 2.2 Product Functions

The NCMMS provides the following major functions:

### 1. User Management

- User registration and authentication
- Role-based access control (Student/Faculty/Staff, Maintenance Staff, Administrator)
- Profile management

### 2. Complaint Management

- Submit new complaints with category, description, location, and images
- View and track complaint status
- Edit or cancel pending complaints
- Add comments or updates to existing complaints

### 3. Administrative Functions

- View all complaints with filtering and search capabilities

- Assign complaints to maintenance staff
- Update complaint status and priority
- Review and approve complaint resolution
- Manage user accounts and roles

#### 4. Maintenance Staff Functions

- View assigned complaints
- Update work progress and status
- Upload completion photos or documentation
- Request reassignment if necessary

#### 5. Reporting and Analytics

- Generate complaint resolution reports
- View performance metrics by category and staff member
- Export data in CSV format
- Display statistical dashboards

#### 6. Notification System

- Email notifications for complaint status changes
- In-app notifications
- Reminder notifications for pending complaints

## 2.3 User Characteristics

### 2.3.1 End Users (Students, Faculty, Staff)

- **Education Level:** Undergraduate to postgraduate level
- **Technical Expertise:** Basic computer and internet skills
- **Usage Pattern:** Occasional users who submit complaints as needed
- **Expectations:** Simple, intuitive interface with minimal training required

### 2.3.2 Maintenance Staff

- **Education Level:** Varies (technical school to undergraduate)
- **Technical Expertise:** Basic smartphone/computer operation skills
- **Usage Pattern:** Regular daily users to check assigned tasks
- **Expectations:** Clear work queue, easy status updates, mobile-friendly interface

### 2.3.3 Administrators/Supervisors

- **Education Level:** Undergraduate or higher
- **Technical Expertise:** Intermediate computer skills, familiar with management systems
- **Usage Pattern:** Daily users for monitoring and management
- **Expectations:** Comprehensive oversight tools, reporting capabilities, user management

## 2.4 Constraints

### 2.4.1 Regulatory Policies

- Must comply with university data protection policies
- User data must be handled according to privacy regulations
- Authentication must follow university security standards

### 2.4.2 Hardware Limitations

- Must function on standard university network infrastructure
- Should work on devices commonly available to students and staff
- Image uploads limited by available storage capacity

### 2.4.3 Interface Requirements

- Must be accessible via standard web browsers
- Should not require plugin installations
- Must work within university firewall restrictions

### 2.4.4 Development Constraints

- Development must follow Kanban methodology
- Must use MERN stack as specified
- Project timeline: Academic semester constraints
- Budget: Student project with limited resources

### 2.4.5 Security Constraints

- All passwords must be encrypted
- JWT tokens must be used for session management
- File uploads must be validated and scanned
- API endpoints must be protected against unauthorized access

## 2.5 Assumptions and Dependencies

### 2.5.1 Assumptions

- Users have access to internet-connected devices
- University will provide hosting infrastructure or cloud hosting budget
- Users have valid university email addresses for registration
- Maintenance staff are willing to adopt the digital system
- Network connectivity is generally reliable on campus
- Mobile devices used will have camera capability for photo uploads

### 2.5.2 Dependencies

- Availability of MongoDB database service
- Cloudinary API service availability and quota limits
- Email server (SMTP) for sending notifications
- Node.js runtime environment on hosting server
- GitHub for version control and deployment
- Vercel or similar platform for application deployment
- Continued availability of chosen third-party libraries and frameworks
- Approval and support from university administration
- Timely feedback from Requirement Provider during development

## 3 Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

##### UI-1: Login Screen

- Input fields for email and password
- "Remember Me" checkbox
- "Forgot Password" link
- "Login" button
- Link to registration page
- University logo and branding
- Responsive design for mobile and desktop

##### UI-2: Registration Screen

- Input fields: Full Name, University ID, Email, Phone Number, Password, Confirm Password
- Role selection dropdown (Student/Faculty/Staff)
- Department/Program selection
- Terms and conditions checkbox
- "Register" button
- Link back to login page

##### UI-3: User Dashboard

- Summary statistics: Total complaints, Open, In Progress, Resolved
- Recent complaints list with status indicators
- "Submit New Complaint" button prominently displayed
- Quick filters: All, My Complaints, Open, Closed
- Navigation menu: Dashboard, New Complaint, My Complaints, Profile, Logout

**UI-4: Complaint Submission Form**

- Category dropdown (Electrical, Plumbing, IT, Furniture, Civil, HVAC, Other)
- Priority selection (Low, Medium, High, Critical)
- Location dropdown or text field (Building, Room Number)
- Title field (max 100 characters)
- Description text area (max 1000 characters)
- Image upload button (up to 3 images, max 5MB each)
- Image preview thumbnails
- "Submit" and "Cancel" buttons
- Form validation with error messages

**UI-5: Complaint Detail View**

- Ticket ID and submission date
- Status badge with color coding
- Category and priority indicators
- Location information
- Complete description
- Uploaded images in gallery view
- Assigned staff member (if assigned)
- Timeline of status changes
- Comments section
- "Add Comment" functionality
- "Edit" and "Cancel" buttons (if applicable)

**UI-6: Admin Dashboard**

- Overall statistics: Total complaints, By category, By status, By priority
- Pending assignments counter
- Staff workload overview
- Recent complaints table with sorting and filtering
- Search functionality
- Quick action buttons: Assign, Update Status, View Details
- Navigation: Dashboard, All Complaints, Users, Staff, Reports

**UI-7: Staff Work Queue**

- List of assigned complaints
- Filter by status and priority
- Complaint cards showing key information
- "Update Status" quick action
- "View Details" link
- "Mark Complete" button with photo upload option

**UI-8: Reports Page**

- Date range selector
- Report type dropdown
- Filter options: Category, Status, Staff Member
- Visual charts: Bar chart, Pie chart, Line graph
- Summary statistics table
- "Export to CSV" button
- Print-friendly layout

**3.1.2 Hardware Interfaces****HW-1: Client Device Interface**

- The system shall be accessible from any device with:
  - Modern web browser (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
  - Minimum screen resolution: 320px width (mobile)
  - Internet connection (minimum 1 Mbps)
  - Camera capability for photo uploads (optional but recommended)

**HW-2: Server Hardware**

- The system shall operate on server hardware with:
  - Minimum 2 CPU cores
  - Minimum 2 GB RAM
  - Minimum 20 GB storage
  - Network interface card with internet connectivity

### 3.1.3 Software Interfaces

#### SW-1: MongoDB Database

- Database Management System: MongoDB version 5.0 or higher
- Purpose: Store all application data including users, complaints, comments, and logs
- Data Format: JSON/BSON documents
- Connection: Mongoose ODM library for Node.js

#### SW-2: Node.js Runtime

- Runtime Environment: Node.js version 18.x or higher
- Purpose: Execute server-side JavaScript code
- Package Manager: npm or yarn for dependency management

#### SW-3: Express.js Framework

- Web Framework: Express.js version 4.x
- Purpose: Handle HTTP requests, routing, and middleware
- Interface: RESTful API endpoints

#### SW-4: React/Next.js Framework

- Frontend Framework: React 18.x or Next.js 13.x
- Purpose: Build user interface components
- Rendering: Client-side and server-side rendering support

#### SW-5: Cloudinary Service

- Cloud Storage Service: Cloudinary API
- Purpose: Store and serve complaint images
- Interface: REST API via Cloudinary SDK
- Authentication: API key and secret
- Storage Limit: Based on subscription plan

#### SW-6: Email Service

- Email Provider: SMTP-compatible service (e.g., Gmail, SendGrid)
- Purpose: Send notification emails
- Protocol: SMTP
- Authentication: Username and password or API key

### 3.1.4 Communications Interfaces

#### COM-1: HTTP/HTTPS Protocol

- The system shall use HTTPS protocol for all client-server communication
- SSL/TLS encryption shall be enforced for production environment
- HTTP methods supported: GET, POST, PUT, PATCH, DELETE

#### COM-2: RESTful API

- API Architecture: REST (Representational State Transfer)
- Data Format: JSON for all request and response payloads
- Authentication: Bearer token (JWT) in Authorization header
- Base URL format: `https://[domain]/api/v1/[resource]`

#### COM-3: Email Communication

- Protocol: SMTP (Simple Mail Transfer Protocol)
- Port: 587 (TLS) or 465 (SSL)
- Email format: HTML with plain text fallback
- Content: Notification messages about complaint status changes

#### COM-4: WebSocket (Optional)

- Protocol: WebSocket for real-time notifications
- Library: Socket.io
- Purpose: Push instant updates to connected clients
- Fallback: HTTP polling if WebSocket not available

## 3.2 Functional Requirements

### 3.2.1 User Authentication and Authorization

**FR-1.1: User Registration** The system shall allow new users to register by providing:

- Full name (mandatory)
- University ID number (mandatory)
- University email address (mandatory)
- Phone number (mandatory)
- Password meeting security criteria (mandatory)
- User role selection: Student/Faculty/Staff (mandatory)
- Department/Program (mandatory)

**FR-1.2: Email Verification** The system shall send a verification email to the registered email address and require users to verify their email before accessing the system.

**FR-1.3: User Login** The system shall authenticate users using email and password credentials and issue a JWT token upon successful authentication.

**FR-1.4: Password Requirements** The system shall enforce password requirements:

- Minimum 8 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- At least one special character

**FR-1.5: Password Reset** The system shall allow users to reset forgotten passwords by:

- Requesting password reset via email
- Receiving a time-limited reset link (valid for 1 hour)
- Creating a new password meeting security criteria

**FR-1.6: Session Management** The system shall maintain user sessions using JWT tokens with:

- Token expiration time of 24 hours
- Automatic logout upon token expiration
- Refresh token capability for extended sessions

**FR-1.7: Role-Based Access Control** The system shall implement three user roles with distinct permissions:

- **End User:** Submit complaints, view own complaints, update profile
- **Maintenance Staff:** View assigned complaints, update status, upload completion photos
- **Administrator:** Full access to all features including user management, complaint assignment, and reporting

### 3.2.2 Complaint Management

**FR-2.1: Submit Complaint** The system shall allow authenticated end users to submit a new complaint with:

- Category selection from: Electrical, Plumbing, IT, Furniture, Civil, HVAC, Other (mandatory)
- Priority selection: Low, Medium, High, Critical (mandatory)
- Location: Building name and room number (mandatory)
- Title: Brief summary (max 100 characters) (mandatory)
- Description: Detailed explanation (max 1000 characters) (mandatory)
- Image attachments: Up to 3 images, max 5MB each (optional)

**FR-2.2: Generate Ticket ID** The system shall automatically generate a unique ticket ID for each complaint in the format: NCMMS-YYYY-NNNN (e.g., NCMMS-2025-0001).

**FR-2.3: Complaint Validation** The system shall validate complaint submissions:

- All mandatory fields must be filled
- Images must be in supported formats (JPG, PNG, GIF)
- Image file size must not exceed 5MB per file
- Description must not contain inappropriate content

**FR-2.4: Initial Complaint Status** The system shall set the status of newly submitted complaints to "Open".

**FR-2.5: View Own Complaints** The system shall allow users to view a list of all their submitted complaints with:

- Ticket ID
- Title
- Category
- Status with color coding
- Submission date
- Priority level

**FR-2.6: View Complaint Details** The system shall allow users to view complete details of their complaints including:

- All submission information
- Current status and assigned staff (if applicable)
- Status change timeline
- Comments and updates
- Uploaded images in full resolution

**FR-2.7: Edit Pending Complaint** The system shall allow users to edit complaints that have "Open" status within 24 hours of submission.

**FR-2.8: Cancel Complaint** The system shall allow users to cancel their own complaints if the status is "Open".

**FR-2.9: Add Comment** The system shall allow users to add comments to their own complaints at any time.

**FR-2.10: Search Complaints** The system shall allow users to search their complaints by:

- Ticket ID
- Title keywords
- Category
- Status
- Date range

### 3.2.3 Administrative Functions

**FR-3.1: View All Complaints** The system shall allow administrators to view all complaints in the system with filtering options:

- By status: All, Open, Assigned, In Progress, Resolved, Closed
- By category: All categories or specific category
- By priority: All priorities or specific priority
- By date range
- By assigned staff member

**FR-3.2: Assign Complaint** The system shall allow administrators to assign complaints to maintenance staff by:

- Selecting available staff members by expertise/category
- Setting expected completion date
- Adding assignment notes
- Automatically updating complaint status to "Assigned"

**FR-3.3: Reassign Complaint** The system shall allow administrators to reassign complaints to different staff members with a reason for reassignment.

**FR-3.4: Update Priority** The system shall allow administrators to change the priority level of any complaint.

**FR-3.5: Update Status Manually** The system shall allow administrators to manually update complaint status to any valid state with a comment explaining the change.

**FR-3.6: Close Complaint** The system shall allow administrators to close resolved complaints after verification.

**FR-3.7: Reopen Complaint** The system shall allow administrators to reopen closed complaints if issues persist.

**FR-3.8: Delete Complaint** The system shall allow administrators to delete complaints that are duplicate or spam, with confirmation dialog.

**FR-3.9: View Staff Workload** The system shall provide administrators with a dashboard showing:

- Number of complaints assigned to each staff member
- Complaints by status for each staff member
- Average resolution time per staff member
- Staff availability status

**FR-3.10: Manage Users** The system shall allow administrators to:

- View list of all registered users
- Search users by name, email, or ID
- View user details and complaint history
- Deactivate/activate user accounts
- Change user roles

**FR-3.11: Manage Staff** The system shall allow administrators to:

- Add new maintenance staff members
- Assign expertise categories to staff
- Set staff availability status
- Remove staff members from the system

### 3.2.4 Maintenance Staff Functions

**FR-4.1: View Assigned Complaints** The system shall provide maintenance staff with a work queue showing all complaints assigned to them, sorted by priority and due date.

**FR-4.2: Update Complaint Status** The system shall allow maintenance staff to update the status of assigned complaints to:

- In Progress - when work begins
- Resolved - when work is completed
- Need Support - when additional resources required

**FR-4.3: Add Work Notes** The system shall allow maintenance staff to add notes documenting work performed, materials used, and time spent.

**FR-4.4: Upload Completion Photos** The system shall allow maintenance staff to upload photos (up to 3 images, 5MB each) when marking a complaint as resolved.

**FR-4.5: Request Reassignment** The system shall allow maintenance staff to request reassignment of a complaint with a reason, which will notify administrators.

**FR-4.6: View Complaint History** The system shall allow maintenance staff to view their complaint resolution history and performance metrics.

### 3.2.5 Notification System

**FR-5.1: Complaint Submission Notification** The system shall send an email notification to the user upon successful complaint submission containing ticket ID and summary.

**FR-5.2: Assignment Notification** The system shall send notifications when:

- A complaint is assigned to maintenance staff (to staff member)
- A complaint is assigned (to the user who submitted it)

**FR-5.3: Status Change Notification** The system shall send email notifications to users when their complaint status changes (In Progress, Resolved, Closed).

**FR-5.4: Comment Notification** The system shall notify users via email when a new comment is added to their complaint by staff or administrators.

**FR-5.5: Reminder Notifications** The system shall send reminder notifications to maintenance staff for complaints approaching due date (24 hours before).

**FR-5.6: In-App Notifications** The system shall display in-app notification badges for:

- New assignments (for staff)
- Status updates (for users)
- New comments

### 3.2.6 Reporting and Analytics

**FR-6.1: Generate Complaint Summary Report** The system shall generate reports showing:

- Total complaints by date range
- Breakdown by category
- Breakdown by status
- Breakdown by priority
- Average resolution time

**FR-6.2: Staff Performance Report** The system shall generate performance reports for each maintenance staff member showing:

- Total assigned complaints
- Total resolved complaints
- Average resolution time
- Pending complaints
- Resolution rate percentage

**FR-6.3: Category Analysis Report** The system shall generate reports analyzing complaints by category over time to identify recurring issues.

**FR-6.4: Export Reports** The system shall allow administrators to export reports in CSV format.

**FR-6.5: Dashboard Visualizations** The system shall display visual charts and graphs:

- Bar chart: Complaints by category
- Pie chart: Complaints by status
- Line graph: Complaint trends over time
- Status indicators for key metrics

### 3.3 Performance Requirements

**PF-1: Response Time** The system shall respond to user actions within the following time limits:

- Page load time: Maximum 3 seconds for initial page load
- Form submission: Maximum 2 seconds to process and confirm
- Search results: Maximum 2 seconds to display results
- Report generation: Maximum 10 seconds for standard reports
- Image upload: Maximum 5 seconds per image

**PF-2: Concurrent Users** The system shall support at least 100 concurrent active users without performance degradation.

**PF-3: Database Performance** The system shall execute database queries with:

- Simple queries (single record): Maximum 100ms
- Complex queries (reports): Maximum 3 seconds
- Batch operations: Maximum 10 seconds

**PF-4: Throughput** The system shall handle:

- Minimum 50 complaint submissions per hour
- Minimum 200 complaint views per hour
- Minimum 100 status updates per hour

**PF-5: Storage Capacity** The system shall accommodate:

- Minimum 10,000 complaints
- Minimum 30,000 images (assuming 3 images per complaint)
- Minimum 5,000 user accounts
- Data retention: Minimum 3 years

**PF-6: Scalability** The system shall be designed to scale horizontally to handle increased load by adding additional server instances.

### 3.4 Design Constraints

#### 3.4.1 Standards Compliance

**DC-1: Web Standards** The system shall comply with:

- HTML5 standards for markup
- CSS3 standards for styling
- ECMAScript 2015+ standards for JavaScript
- WCAG 2.1 Level AA accessibility guidelines

**DC-2: REST API Standards** The system shall follow REST API best practices:

- Use appropriate HTTP methods (GET, POST, PUT, DELETE)
- Implement proper HTTP status codes
- Use JSON for data exchange
- Implement versioning in API URLs

**DC-3: Security Standards** The system shall implement:

- OWASP Top 10 security recommendations
- Encryption for sensitive data at rest and in transit
- Secure password hashing (bcrypt with minimum 10 rounds)
- JWT token-based authentication

#### 3.4.2 Hardware Limitations

**DC-4: Browser Compatibility** The system shall function correctly on:

- Chrome version 90 or higher
- Firefox version 88 or higher
- Safari version 14 or higher
- Microsoft Edge version 90 or higher

**DC-5: Mobile Responsiveness** The system shall provide responsive design supporting:

- Minimum screen width: 320px (mobile devices)
- Tablet devices: 768px - 1024px
- Desktop devices: 1025px and above

**DC-6: Network Bandwidth** The system shall function with minimum internet speed of 1 Mbps for basic operations and 3 Mbps for optimal performance including image uploads.

### 3.4.3 Technology Constraints

**DC-7: Technology Stack** The system shall be developed using:

- Frontend: React.js or Next.js
- Backend: Node.js with Express.js
- Database: MongoDB
- Authentication: JWT (JSON Web Tokens)
- Cloud Storage: Cloudinary
- Version Control: GitHub
- Deployment: Vercel or similar platform

**DC-8: Programming Language** The system shall be implemented using JavaScript/TypeScript for both frontend and backend.

**DC-9: Development Methodology** The system shall be developed following Kanban methodology with continuous integration and deployment practices.

## 3.5 Software System Attributes

### 3.5.1 Reliability

**AT-1: Availability** The system shall maintain 99% uptime during operational hours (24/7 excluding scheduled maintenance).

**AT-2: Error Handling** The system shall:

- Display user-friendly error messages for all error conditions
- Log all errors on the server for debugging purposes
- Provide graceful degradation when third-party services are unavailable
- Not expose sensitive system information in error messages

**AT-3: Data Backup** The system shall:

- Perform automatic daily database backups
- Retain backups for minimum 30 days
- Store backups in a separate location from primary database

**AT-4: Recovery** The system shall support recovery from failures with:

- Maximum Recovery Time Objective (RTO): 4 hours
- Maximum Recovery Point Objective (RPO): 24 hours

### 3.5.2 Security

**AT-5: Authentication Security** The system shall:

- Implement account lockout after 5 failed login attempts for 30 minutes
- Require strong passwords as specified in FR-1.4
- Use HTTPS for all communications
- Implement JWT tokens with expiration

**AT-6: Authorization** The system shall:

- Verify user permissions before allowing access to any resource
- Implement role-based access control (RBAC)
- Prevent privilege escalation attacks
- Log all authorization failures

**AT-7: Data Protection** The system shall:

- Encrypt passwords using bcrypt hashing
- Encrypt sensitive data in database
- Validate and sanitize all user inputs to prevent SQL injection and XSS attacks
- Implement CSRF protection for all state-changing operations

**AT-8: File Upload Security** The system shall:

- Validate file types and reject executables
- Scan uploaded files for malware
- Limit file sizes to prevent denial of service
- Store uploaded files in secure cloud storage

**AT-9: Session Security** The system shall:

- Invalidate sessions upon logout
- Implement session timeout after 30 minutes of inactivity
- Use secure, HTTP-only cookies for session management

**AT-10: Audit Trail** The system shall maintain audit logs for:

- User login/logout activities
- Complaint submissions and modifications
- Administrative actions
- Status changes
- User management actions

### 3.5.3 Maintainability

**AT-11: Code Quality** The system shall:

- Follow consistent coding standards and style guides
- Include inline comments for complex logic
- Maintain code documentation
- Achieve minimum 70% code coverage with unit tests

**AT-12: Modularity** The system shall:

- Implement modular architecture with clear separation of concerns
- Use reusable components and functions
- Minimize coupling between modules
- Document module interfaces and dependencies

**AT-13: Version Control** The system shall:

- Use Git for version control
- Follow branching strategy (main, development, feature branches)
- Require code reviews before merging to main branch
- Tag releases with semantic versioning

**AT-14: Database Maintenance** The system shall:

- Support database schema migrations
- Include scripts for data cleanup and maintenance
- Document database structure and relationships

### 3.5.4 Portability

**AT-15: Platform Independence** The system shall run on any platform supporting:

- Node.js runtime environment
- Modern web browsers
- No platform-specific dependencies

**AT-16: Data Portability** The system shall:

- Support export of data in standard formats (CSV, JSON)
- Provide APIs for data migration
- Document data schemas for external integration

### 3.5.5 Usability

**AT-17: User Interface Consistency** The system shall maintain consistent:

- Navigation patterns across all pages
- Visual design and color scheme
- Terminology and labeling
- Button placement and behavior

**AT-18: Ease of Use** The system shall:

- Allow new users to submit a complaint within 5 minutes without training
- Provide contextual help and tooltips where needed
- Display clear progress indicators for multi-step processes
- Minimize required clicks to complete common tasks

**AT-19: Error Prevention** The system shall:

- Validate inputs in real-time with immediate feedback
- Provide confirmation dialogs for destructive actions
- Auto-save draft data to prevent data loss
- Highlight required fields clearly

**AT-20: Accessibility** The system shall:

- Support keyboard navigation for all functions
- Provide adequate color contrast (WCAG AA standard)
- Include alt text for all images
- Support screen readers
- Allow text size adjustment

**AT-21: Help and Documentation** The system shall provide:

- FAQ section addressing common questions
- User guide for basic operations
- Video tutorials for key features
- Contact information for technical support

## 3.6 Other Requirements

### 3.6.1 Database Requirements

**OR-1: Data Model** The system database shall include the following collections/tables:

- Users: User profiles and authentication information
- Complaints: All complaint records
- Comments: User and staff comments on complaints
- Categories: Complaint categories and subcategories
- Staff: Maintenance staff profiles and expertise
- Notifications: Notification queue and history
- AuditLogs: System activity logs
- Settings: System configuration parameters

**OR-2: Data Integrity** The system shall:

- Enforce referential integrity between related collections
- Prevent orphaned records
- Validate data types and constraints at database level
- Use transactions for operations affecting multiple collections

**OR-3: Data Retention** The system shall:

- Retain complaint records for minimum 3 years
- Archive old complaints after 2 years to secondary storage
- Retain audit logs for minimum 1 year
- Soft-delete user data to maintain historical references

### 3.6.2 Operational Requirements

**OR-4: Installation** The system shall:

- Provide installation scripts for automated deployment
- Include configuration templates with sensible defaults
- Document all environment variables and dependencies
- Support containerized deployment using Docker

**OR-5: System Monitoring** The system shall:

- Log all critical operations and errors
- Provide health check endpoints for monitoring tools
- Track key performance metrics (response times, error rates)
- Alert administrators for critical failures

**OR-6: Backup and Recovery** The system shall:

- Support automated backup procedures
- Provide manual backup triggering capability
- Include database restore procedures
- Test backup integrity regularly

**OR-7: System Updates** The system shall:

- Support zero-downtime deployment for minor updates
- Provide rollback capability for failed updates
- Notify administrators of available updates
- Maintain backward compatibility for API versions

### 3.6.3 Legal and Compliance Requirements

**OR-8: Data Privacy** The system shall:

- Comply with university data protection policies
- Obtain user consent for data collection
- Allow users to view their stored personal data
- Provide data deletion capability upon request

**OR-9: Terms of Service** The system shall:

- Display terms of service during registration
- Require acceptance of terms before account creation
- Provide accessible privacy policy
- Document acceptable use policy

**OR-10: Intellectual Property** The system shall:

- Use only properly licensed third-party libraries
- Display appropriate copyright notices
- Document all open-source components and their licenses

## A Use Case Diagram

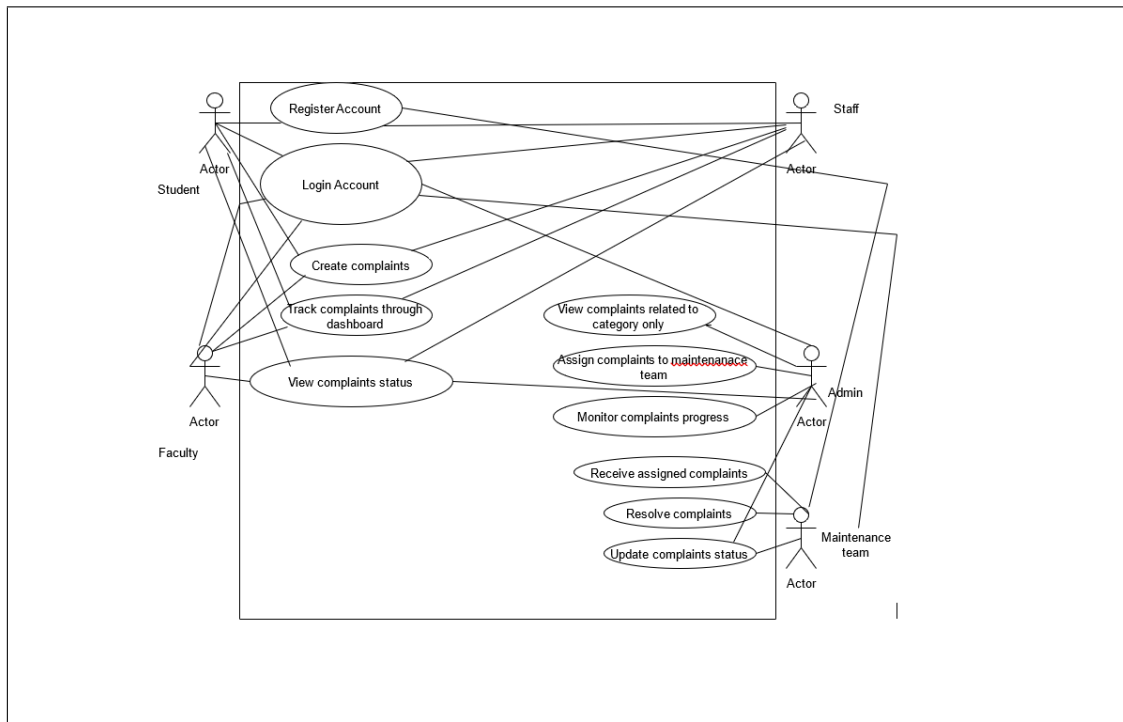


Figure 1: Use Case Diagram - Namal Complaint Management System

## B Context Diagram

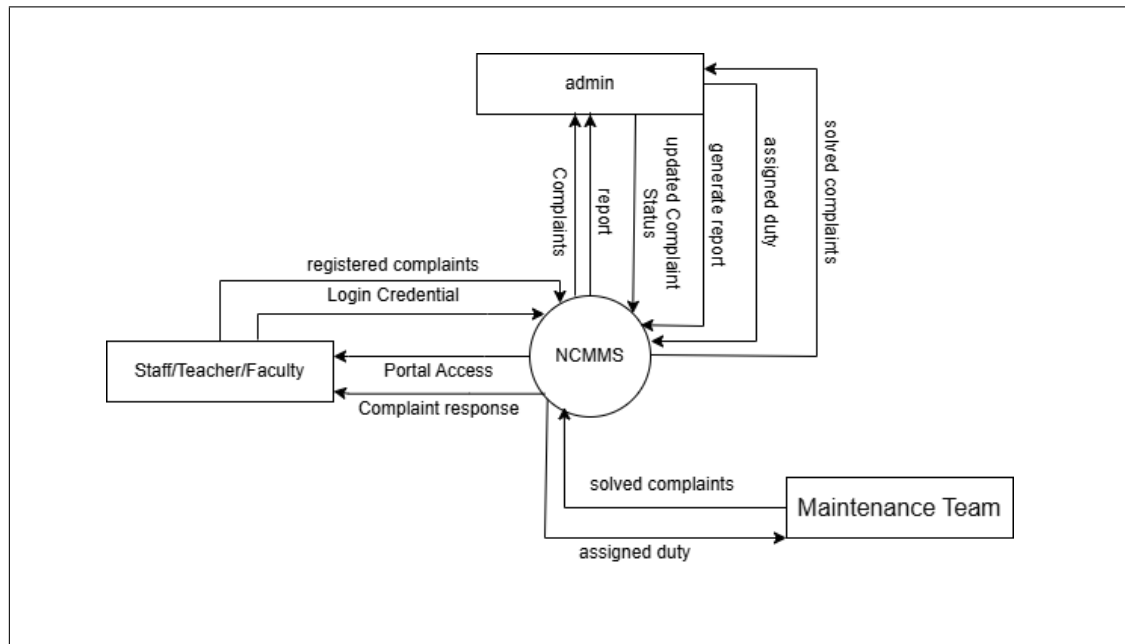


Figure 2: Context Diagram - NCMMS System Boundary and External Entities

## C Sample Screen Mockups

*This section should include wireframes or mockups of key screens:*

1. Login Screen
2. User Dashboard
3. Complaint Submission Form
4. Complaint Detail View
5. Admin Dashboard
6. Staff Work Queue
7. Reports Page

*Note: Screen mockups should be created using Figma as specified in the project proposal and included in this section.*

## D Traceability Matrix

Requirement ID	Requirement Description	Related Components
FR-1.1	User Registration	Authentication Module, User Database
FR-1.3	User Login	Authentication Module, JWT Service
FR-2.1	Submit Complaint	Complaint Module, File Upload Service
FR-2.2	Generate Ticket ID	Complaint Module, ID Generator
FR-3.2	Assign Complaint	Admin Module, Notification Service
FR-4.2	Update Status	Staff Module, Status Manager
FR-5.1	Submission Notification	Notification Service, Email Service
FR-6.1	Generate Reports	Reporting Module, Analytics Engine
PF-1	Response Time	Performance Optimization, Caching
AT-5	Authentication Security	Security Module, Encryption Service
AT-17	UI Consistency	Frontend Components, Style Guide

*Note: This is a sample traceability matrix. A complete matrix should map all requirements to their implementing components and test cases.*

## E Glossary of Terms

**API** Application Programming Interface - A set of protocols for building software applications

**Assignment** The act of allocating a complaint to a specific maintenance staff member

**Audit Trail** A chronological record of system activities

**Backend** Server-side application logic and database operations

**Category** Classification of complaint type (Electrical, Plumbing, IT, etc.)

**Cloudinary** Third-party cloud service for image storage and delivery

**Comment** A textual note added to a complaint by users or staff

**CRUD** Create, Read, Update, Delete - Basic database operations

**Dashboard** Main interface showing summary information and statistics

**End User** Students, faculty, or staff who submit complaints

**Express.js** Web application framework for Node.js

**Frontend** Client-side user interface of the application

**HTTPS** Secure version of HTTP protocol with encryption

**JWT** JSON Web Token - Compact, URL-safe token for authentication

**Kanban** Agile project management methodology using visual workflow

**MERN Stack** Technology stack using MongoDB, Express, React, and Node.js

**MongoDB** NoSQL document database

**Next.js** React framework for production-grade applications

**Node.js** JavaScript runtime environment for server-side programming

**ODM** Object Document Mapper - Maps objects to database documents

**Priority** Urgency level of a complaint (Low, Medium, High, Critical)

**React** JavaScript library for building user interfaces

**REST** Representational State Transfer - Architectural style for APIs

**Role** User permission level (End User, Staff, Administrator)

**Status** Current state of complaint (Open, Assigned, In Progress, Resolved, Closed)

**Ticket ID** Unique identifier for each complaint

**Workload** Number of complaints assigned to a staff member

**Work Queue** List of complaints assigned to maintenance staff