DESIGN DOCUMENT

For

Road Repair and Tracking System

Prepared by: **Archisman Chakraborty(224CS1004)**
Submitted to: **Dr. Judhistir Mahapatro**

Date: **29th August, 2024**

Contents

1. Introduction

The Road Repair and Tracking System (RRTS) is a software solution aimed at automating the road repair management process for the Public Works Department (PWD) of a city corporation. This design document details the architecture, system components, modules, interfaces, and data structures to provide a blueprint for developing and implementing the system.

2. System Overview

The RRTS follows a three-tier architecture comprising the *Presentation Layer*, *Business Logic Layer*, and *Data Layer*.

- **Presentation Layer**: This layer provides the user interface and is responsible for handling user interactions. It includes web-based interfaces for different user roles—clerks, supervisors, administrators, and the city mayor.
- **Business Logic Layer**: This layer contains the core functionality and processes of the system, such as complaint management, repair prioritization, scheduling, resource allocation, and reporting. It handles data processing and business rules to ensure the correct functioning of the application.
- **Data Layer**: This layer manages data storage and retrieval operations. It uses a relational database to store information on complaints, repairs, resources, users, and reports. The data layer ensures data integrity, security, and consistency.

3. Requirements Tracing

The purpose of this section is to provide a clear mapping between the requirements of the Road Repair and Tracking System (RRTS) and the corresponding design elements that implement these requirements. This ensures that all specified functionalities are covered and allows for better project tracking, validation, and future maintenance.

3.1. Functional Requirements Tracing

Requirement ID	Requirement Description	Design Element(s)
FR-01	The system shall allow clerks to enter road repair complaints into the system received over the phone or through written complaints.	Complaint Management Module, User Interface for Clerks
FR-02	The system shall generate an area-wise list of fresh complaints every morning for respective supervisors.	Complaint Reporting Module, Scheduled Report Generation

Requirement ID	Requirement Description	Design Element(s)
FR-03	Supervisors shall be able to examine complaints, assess severity, and assign priorities based on road conditions and locality type.	Supervisor Dashboard, Complaint Assessment and Priority Module
FR-04	The system shall provide functionality to estimate raw material requirements, machines, and personnel required for each repair task.	Resource Estimation Module, Repair Scheduling System
FR-05	The system shall automatically schedule repair tasks based on priority and availability of resources, machines, and personnel.	Automated Scheduling Module, Resource Management System
FR-06	The system shall allow administrators to update the status and availability of manpower, machines, and materials at any time.	Resource Management Module, Administrator Control Panel
FR-07	The system shall reschedule repair tasks dynamically when there are changes in the availability of resources.	Dynamic Rescheduling Component, Resource Management System
FR-08	The system shall allow the mayor to request various reports, such as the number of repairs completed and outstanding, and utilization of resources over time.	Reporting and Analytics Module, Custom Report Generation Interface

3.2. Non-Functional Requirements Tracing

Requirement ID	Requirement Description	Design Element(s)
NFR-01	The system shall have a user-friendly interface for easy navigation and task completion.	User Interface Design, UX Principles
NFR-02	The system shall be secure and ensure data privacy and integrity.	Authentication and Authorization, Data Encryption Mechanisms
NFR-03	The system shall provide high availability and be able to handle high volumes of data and concurrent users.	Scalable Architecture Design, Load Balancing and Caching
NFR-04	The system shall be designed for easy	Modular Design Approach,

Requirement ID	Requirement Description	Design Element(s)
	maintenance and future enhancements.	Microservices Architecture
NFR-05	The system shall provide quick response times for data retrieval and processing.	Optimized Database Queries, Efficient Data Management

3.3. Mapping of Requirements to Modules

Module/Component	Requirement(s) Addressed
Complaint Management Module	FR-01, FR-02
Supervisor Dashboard	FR-03
Complaint Assessment and Priority Module	FR-03
Resource Estimation Module	FR-04
Automated Scheduling Module	FR-05
Resource Management Module	FR-06, FR-07
Dynamic Rescheduling Component	FR-07
Reporting and Analytics Module	FR-08
Administrator Control Panel	FR-06

3.4. Traceability Matrix

The following traceability matrix helps map each requirement to the corresponding design element and module:

Requirement ID	Design Element	Module/Component	Status
FR-01	User Interface for Clerks	Complaint Management Module	Planned
FR-02	Scheduled Report Generation	Complaint Management Module	Planned
FR-03	Supervisor Dashboard	Complaint Assessment and Priority Module	Planned
FR-04	Resource Estimation UI	Resource Estimation Module	Planned

Requirement ID	Design Element	Module/Component	Status
FR-05	Automated Scheduling Logic	Automated Scheduling Module	Planned
FR-06	Admin Resource Update Interface	Administrator Control Panel	Planned
FR-07	Dynamic Scheduling Algorithm	Dynamic Rescheduling Component	Planned
FR-08	Custom Report Generation Interface	Reporting and Analytics Module	Planned
NFR-01	User Interface Design Principles	All Modules	Planned
NFR-02	Authentication and Data Encryption	Security Subsystem	Planned

4. Module Description

The system is divided into several modules, each responsible for a specific set of functionalities:

1. **Complaint Management Module**
   - **User Interface**:
     - **Components**: Form for complaint entry, complaint list view, status update interface.
     - **Interactions**: Clerks can enter new complaints, update statuses, and generate reports.
   - **Backend Logic**:
     - **Database Tables**: Complaints, Users.
     - **Key Functions**: AddComplaint(), UpdateComplaintStatus(), GenerateComplaintReport().
   - **Data Flow**:
     - **Input**: Complaint details from the UI.
     - **Processing**: Store and update complaint information in the database.
     - **Output**: Updated complaint lists and status reports.
2. **Supervisor Dashboard**

- **User Interface**:
    - **Components**: Complaint review panel, severity assessment tool, priority assignment interface.
    - **Interactions**: Supervisors review complaints, assess severity, and set priorities.
- **Backend Logic**:
    - **Database Tables**: Complaints, Repairs.
    - **Key Functions**: ReviewComplaint(), AssessSeverity(), AssignPriority().
- **Data Flow**:
    - **Input**: Complaint details and supervisor inputs.
    - **Processing**: Update complaint priorities and create repair tasks.
    - **Output**: Updated repair schedules and priority lists.

3. **Resource Estimation Module**

- **User Interface**:
    - **Components**: Resource estimation form, resource requirements calculator.
    - **Interactions**: Supervisors input repair details to estimate resource needs.
- **Backend Logic**:
    - **Database Tables**: Resources, Estimates.
    - **Key Functions**: EstimateResources(), CalculateRequirements().
- **Data Flow**:
    - **Input**: Repair details and locality type.
    - **Processing**: Calculate required resources based on predefined rules.
    - **Output**: Estimated resource requirements for each repair task.

4. **Automated Scheduling System**

- **User Interface**:
    - **Components**: Scheduling interface, task allocation view.
    - **Interactions**: Automatic scheduling based on priority and resource availability.
- **Backend Logic**:
    - **Database Tables**: Repairs, Resources.
    - **Key Functions**: ScheduleRepair(), AllocateResources().
- **Data Flow**:

- **Input**: Repair priorities and resource availability.
- **Processing**: Generate repair schedules and allocate resources.
- **Output**: Scheduled repair tasks and resource allocation details.

5. **Resource Management Module**
   - **User Interface**:
     - **Components**: Resource status update form, resource tracking dashboard.
     - **Interactions**: Administrators update and monitor resource availability.
   - **Backend Logic**:
     - **Database Tables**: Resources.
     - **Key Functions**: UpdateResourceStatus(), TrackResourceUsage().
   - **Data Flow**:
     - **Input**: Resource status updates from the UI.
     - **Processing**: Update and track resource availability.
     - **Output**: Updated resource status and usage reports.
6. **Reporting and Analytics Module**
   - **User Interface**:
     - **Components**: Report generation interface, analytics dashboard.
     - **Interactions**: Generate and view various reports.
   - **Backend Logic**:
     - **Database Tables**: Reports.
     - **Key Functions**: GenerateReport(), ViewReport().
   - **Data Flow**:
     - **Input**: Report generation requests.
     - **Processing**: Compile and format report data.
     - **Output**: Generated reports in various formats.
7. **Administrator Control Panel**
   - **User Interface**:
     - **Components**: User management interface, system configuration settings.
     - **Interactions**: Administrators manage system settings and user roles.
   - **Backend Logic**:

    - **Database Tables**: Users, SystemSettings.
    - **Key Functions**: ManageUsers(), ConfigureSystem().
- **Data Flow**:
    - **Input**: System configuration changes and user management actions.
    - **Processing**: Apply configuration changes and manage user roles.
    - **Output**: Updated system settings and user management records.

5. User Interface Design

The user interface (UI) will be designed to be intuitive and easy to navigate. Each user type will have a specific interface tailored to their needs:

- **Clerk Interface**: Simple forms for complaint entry, search, and update. Allows clerks to view complaint status and history.
- **Supervisor Interface**: Dashboards for viewing area-wise complaints, prioritization tools, and resource status.
- **Administrator Interface**: Tools for managing resources, user roles, and viewing overall repair schedules.
- **Mayor Interface**: High-level reports on repair status, resource utilization, and overall city maintenance statistics.

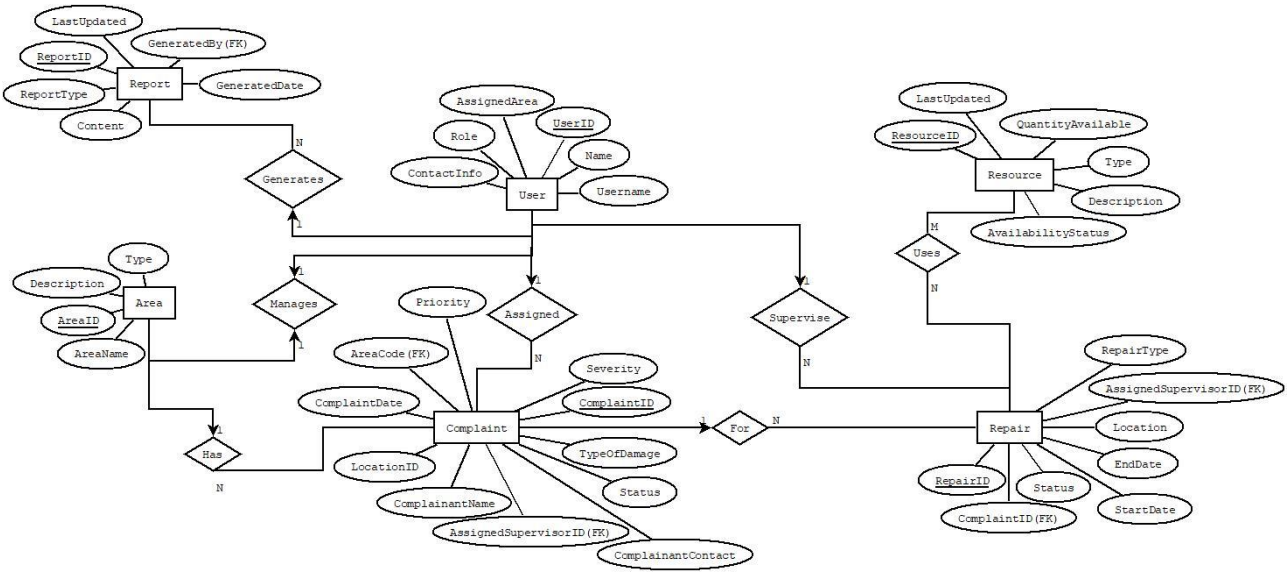
6. Database Design

The system uses a **relational database** to store and manage data. The primary entities and their relationships are as follows:

5.1. Key Entities

1. **Complaint**: Stores complaint details, including complaint ID, date, location, type of damage, severity, complainant details, status, and priority.
2. **Repair**: Stores information about scheduled repairs, including repair ID, location, assigned supervisor, resources allocated, start and end dates, and status.
3. **User**: Stores user information, including user ID, name, role (clerk, supervisor, administrator, mayor), contact information, and login credentials.
4. **Resource**: Stores information about available resources, including resource ID, type (manpower, machinery, materials), availability status, and current allocation.
5. **Report**: Stores generated report details, including report ID, type, generation date, and the data included.

5.2. Entity-Relationship (ER) Diagram



6. Data Definitions

The **Road Repair and Tracking System (RRTS)** involves multiple data entities that represent various aspects of the system, including complaints, users, resources, repairs, and reports. This section defines these entities, their attributes, and how they are used within the system.

1. Entity: Complaints

- **Description**: Represents the complaints lodged by residents regarding road repairs.
- **Attributes**:
    - ComplaintID (Integer, Primary Key): Unique identifier for each complaint.
    - ResidentName (String, 100): Name of the resident lodging the complaint.
    - ResidentContact (String, 15): Contact information of the resident.
    - ComplaintDetails (Text): Detailed description of the complaint.
    - ComplaintDate (DateTime): Date and time when the complaint was lodged.
    - Location (String, 200): Specific location of the road that needs repair.
    - AreaCode (String, 10): Code representing the area where the complaint is located.
    - Status (String, 20): Current status of the complaint (e.g., New, In Progress, Resolved).
    - Priority (String, 10): Priority level assigned by the supervisor (e.g., High, Medium, Low).

2. Entity: Users

- **Description**: Represents all users interacting with the system, including clerks, supervisors, administrators, and the mayor.
- **Attributes**:
    - UserID (Integer, Primary Key): Unique identifier for each user.
    - Name (String, 50): Name of the user.
    - Username (String, 50): Username of the user.
    - Role (String, 20): Role of the user (e.g., Clerk, Supervisor, Administrator, Mayor).
    - Password (String, 100): Encrypted password for authentication.
    - ContactInfo (String, 50): Contact information of the user.
    - AssignedArea (String, 10, Nullable): Area code assigned to the supervisor (applicable only for supervisors).

3. Entity: Resources

- **Description**: Represents the resources available for road repair tasks, including manpower, machinery, and materials.
- **Attributes**:
    - ResourceID (Integer, Primary Key): Unique identifier for each resource.
    - Type (String, 20): Type of resource (e.g., Manpower, Machinery, Material).
    - Description (String, 50): Name or designation of the resource.
    - AvailabilityStatus (String, 20): Current availability status (e.g., Available, In Use, Under Maintenance).
    - QuantityAvailable (Integer): Quantity of the resource available.
    - LastUpdated (DateTime): Timestamp when the resource status was last updated.

4. Entity: Repairs

- **Description**: Represents the repair tasks scheduled by the system based on complaints and resource availability.
- **Attributes**:
    - RepairID (Integer, Primary Key): Unique identifier for each repair task.
    - ComplaintID (Integer, Foreign Key): Reference to the associated complaint.
    - AssignedSupervisorID (Integer, Foreign Key): ID of the supervisor overseeing the repair.
    - StartDate (DateTime): Scheduled start date for the repair task.

- EndDate (DateTime, Nullable): Actual completion date of the repair task.
- Location (Text): Location of the repair
- RepairType (Text): Type of repair task.
- Status (String, 20): Current status of the repair (e.g., Scheduled, In Progress, Completed).

5. Entity: Reports

- **Description**: Represents various reports generated by the system for analysis and decision-making.
- **Attributes**:
  - ReportID (Integer, Primary Key): Unique identifier for each report.
  - ReportType (String, 30): Type of report (e.g., Completed Repairs, Resource Utilization).
  - GeneratedBy (Integer, Foreign Key): User ID of the person who generated the report.
  - GeneratedDate (DateTime): Date and time when the report was generated.
  - Content (Text): Serialized data or file path containing the report details.
  - LastUpdated (DateTime): Date and time when the report was updated last.

6. Entity: RepairHistory

- **Description**: Represents the history of each repair task for tracking purposes.
- **Attributes**:
  - HistoryID (Integer, Primary Key): Unique identifier for each history entry.
  - RepairID (Integer, Foreign Key): Reference to the repair task.
  - ChangeDate (DateTime): Date and time of the status change.
  - PreviousStatus (String, 20): The previous status of the repair task.
  - NewStatus (String, 20): The new status after the change.
  - ChangedBy (Integer, Foreign Key): User ID of the person who made the change.
  - Comments (Text, Nullable): Comments or notes regarding the status change.

Data Relationships

- **Complaints ↔ Repairs**: One-to-Many. A single complaint can lead to multiple repair tasks (if divided into stages), but each repair task is linked to a specific complaint.

- **Users (Supervisors) ↔ Repairs**: One-to-Many. Each repair task is supervised by one supervisor, but a supervisor can manage multiple repair tasks.
- **Users (Clerks) ↔ Complaints**: One-to-Many. Clerks can register multiple complaints, but each complaint is entered by one clerk.
- **Repairs ↔ Resources**: Many-to-Many. Each repair task may require multiple resources, and each resource can be allocated to multiple repair tasks.
- **Users ↔ Reports**: One-to-Many. A user can generate multiple reports, but each report is generated by one user.
- **Repairs ↔ RepairHistory**: One-to-Many. A repair task can have multiple entries in the repair history, each representing a change in status.

7. Technology Stack

- **Frontend**: HTML, CSS, JavaScript, React.js
- **Backend**: Python, Django Framework
- **Database**: PostgreSQL or MySQL
- **APIs**: RESTful APIs for integration with external systems
- **Hosting**: Cloud-based infrastructure (e.g., AWS, Azure) for scalability and reliability
- **Security**: SSL/TLS for data transmission, role-based access control, and data encryption for sensitive information

8. Security Considerations

- **Data Privacy**: Compliance with data protection laws by securing personal data and obtaining user consent.
- **Access Control**: Implement role-based access controls to ensure only authorized users access sensitive functions and data.
- **Audit Trails**: Maintain audit logs for key actions performed by users to ensure accountability.
- **Backup and Recovery**: Regular backups and disaster recovery plans to prevent data loss.

9. Performance Considerations

- **Scalability**: The system should be scalable to handle increased load as the city expands and the number of complaints grows.
- **Optimization**: Efficient database queries and optimized code to minimize response times and ensure a smooth user experience.
- **Load Balancing**: Implement load balancing strategies to distribute the load evenly across servers.

10. Testing Strategy

- **Unit Testing**: Testing individual modules and components for functionality and correctness.
- **Integration Testing**: Testing interactions between different modules to ensure seamless integration.
- **User Acceptance Testing (UAT)**: Testing the system with real users (clerks, supervisors, administrators) to validate usability and functionality.
- **Performance Testing**: Load and stress testing to ensure the system performs well under varying conditions.

11. Deployment Plan

- **Phase 1**: Deploy the system in a pilot area to validate its functionality and gather user feedback.
- **Phase 2**: Incorporate feedback, make necessary adjustments, and roll out the system city-wide.
- **Phase 3**: Continuous monitoring and optimization, including user training and support.

12. Maintenance and Support

- **Regular Updates**: Implement periodic updates to address bugs, add new features, and improve security.
- **Helpdesk and Support**: Establish a support system to address user queries and technical issues promptly.
- **Documentation**: Provide comprehensive user manuals and technical documentation for future maintenance and development.

13. Conclusion

The design of the Road Repair and Tracking System (RRTS) aims to provide a robust, scalable, and efficient solution for managing road repairs in a large city. This document serves as a blueprint for development, ensuring that all technical, functional, and user requirements are met.