Requirement Analysis Document

ROAD REPAIR AND TRACKING SYSTEM (RRTS)

Team :
Tech Wizards

Name	Roll Number	Mail Id
M.Likhitha	R200105	rr200105@rgukt.rkv.ac.in
S.Pushpalatha	R200049	rr200105@rgukt.rkv.ac.in
K.Sreeja	R200816	rr200816@rgukt.rkv.ac.in
V.keshav	R200505	rr200505@rgukt.rkv.ac.in
T.Shiva Shankar	R200304	rr200304@rgukt.rkv.ac.in

Table of Contents

Introduction:

The Road Repair and Tracking Software (RRTS) is designed to automate and streamline the Public Works Department's management of road repair tasks in a large city. By integrating resident complaints and supervisor assessments, the system ensures efficient prioritization and scheduling of repairs. It also provides real-time updates on resource availability and generates detailed reports for city officials.

Purpose:

The purpose of RRTS is to enhance the efficiency of road repair operations by automating complaint logging, prioritization, and scheduling based on severity and resource availability. It aims to ensure timely repairs by dynamically adjusting schedules in response to changes in manpower and machine availability. Additionally, it provides comprehensive statistics and reports to city officials for informed decision-making and resource allocation.

Intended Audience:

- **Customers:** Residents who report road issues.
- **Stakeholders:** Public Works Department staff and city officials.

Product Vision

- **Vision Statement:** Develop a user-friendly and accessible RRTS to provide efficient and timely road repair services.

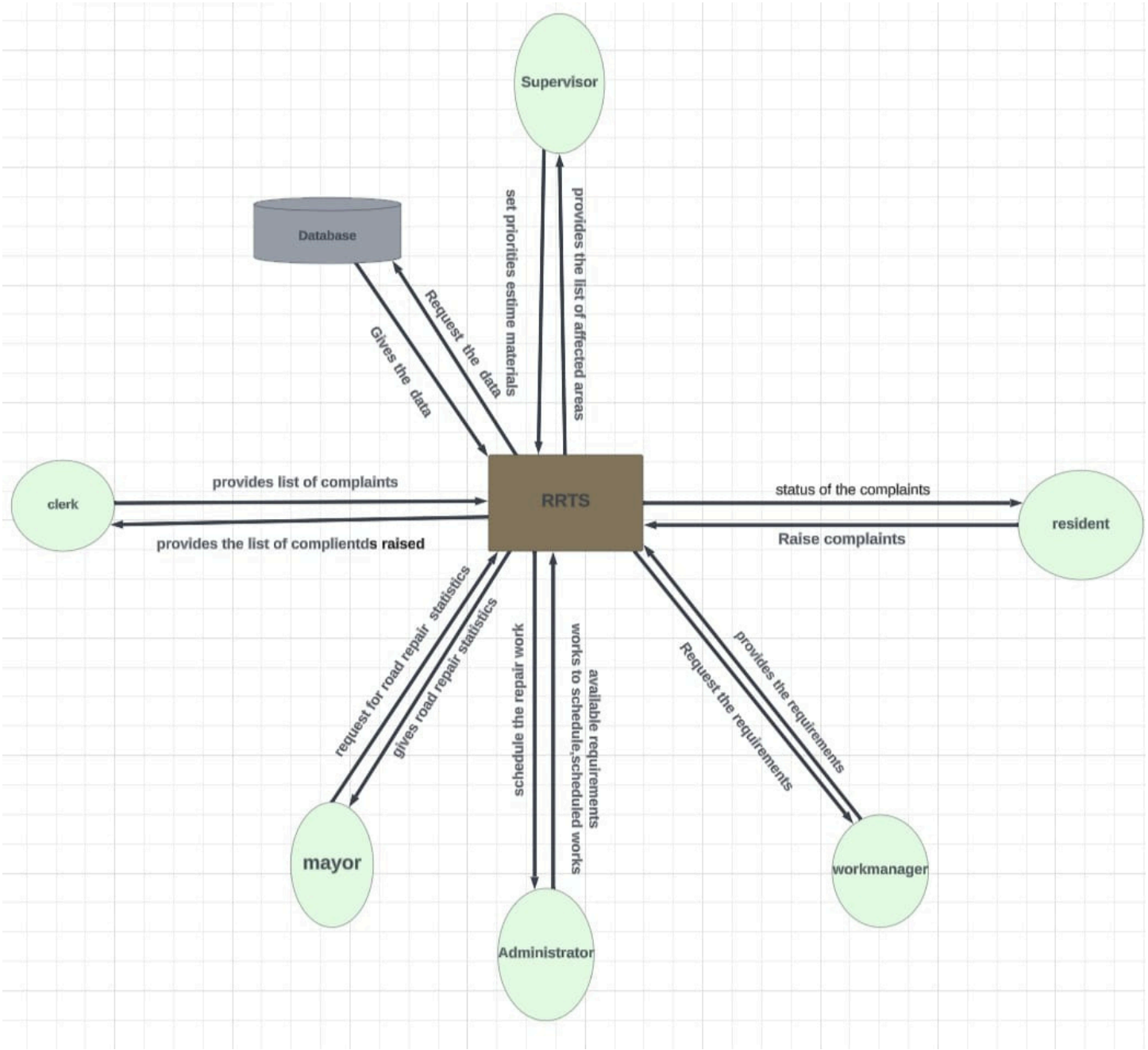
Technologies

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Servlets,JDBC
- **Tools:** Git,Canva

System Context:

- **Complaint Logging:** Residents report road issues via phone or written complaints, which are entered into the system by clerks.
- **Supervisor Assessment:** Supervisors receive daily complaint lists, assess severity, prioritize repairs, and estimate resource requirements.
- **Scheduling:** The system schedules repairs based on priority and resource availability, updating dynamically with any changes.
- **Resource Management:** Administrators update manpower and machine availability, triggering rescheduling as needed.
- **Reporting:** City officials access comprehensive reports on repair status, resource utilization, and repair history for informed decision-making.

Context Diagram



Functional Requirements

1. **Complaint Management**
   - Ability for clerks to enter resident complaints into the system.
   - Generate area-wise lists of fresh complaints for supervisors daily.
2. **Supervisor Assessment**
   - Allow supervisors to view, assess, and prioritize complaints.
   - Enable supervisors to input estimates for materials, machines, and personnel required.
3. **Scheduling**
   - Automatically schedule repairs based on priority, severity, and resource availability.

    - Reschedule repairs dynamically when there are changes in resource availability.
4. **Resource Management**
    - Allow administrators to update availability of manpower and machines.
    - Maintain real-time data on current and committed resources.
5. **Reporting**
    - Generate reports on repair statistics, including completed repairs, outstanding repairs, and resource utilization.
    - Provide city officials with access to various customizable reports.
6. **Notifications**
    - Send notifications to relevant parties (clerks, supervisors, administrators) about updates or changes in repair schedules.
7. **User Management**
    - Manage different user roles (clerk, supervisor, administrator, city official) and their access rights.

Non-Functional Requirements

1. **Performance**
    - The system should handle up to 1000 concurrent users without performance degradation.
    - Response time for any user action should be under 2 seconds.
2. **Reliability**
    - Ensure 99.9% system uptime.
    - Implement data backup procedures to prevent data loss.
3. **Scalability**
    - The system should be scalable to accommodate an increasing number of users and complaints as the city grows.
    - Modular design to allow for easy addition of new features.
4. **Security**
    - Implement role-based access control to ensure data privacy.
    - Encrypt sensitive data both in transit and at rest.
    - Conduct regular security audits and vulnerability assessments.
5. **Usability**
    - User interfaces should be intuitive and user-friendly.
    - Provide training and documentation for all user roles.
6. **Maintainability**
    - Code should follow best practices and be well-documented to facilitate maintenance and updates.

- Implement automated testing for continuous integration and deployment.

7. **Compatibility**
   - The system should be compatible with major web browsers (Chrome, Firefox, Edge, Safari).
   - Ensure mobile responsiveness for access via tablets and smartphones.
8. **Compliance**
   - Adhere to relevant local and national regulations regarding data protection and privacy.
   - Ensure accessibility compliance to accommodate users with disabilities.