Optimized Field Engineer Task Management System for Telecommunications Infrastructure Maintenance

# INTRODUCTION

SwiftLink Telecom Services (STS) dispatches field engineers to perform a range of tasks including laying new backbone network, rectifying network faults, and maintaining telecom backbone infrastructure across various locations such as exchanges, consumer premises, junction boxes, and fiber network poles. These tasks are initiated through a ticketing system and must be prioritized and scheduled based on factors like service type (installation or fault), work location, potential hazards, traffic conditions, and proximity to the engineer's residence.

To facilitate this process, a comprehensive system is developed. This system receives task requests from the users, schedule assignments for engineers, and manages task allocations. The engineers should be able to update their statuses, indicating completion, failure, or deferred. The system also sends notifications to users regarding task statuses.

Moreover, the system integrates with Hazard and Risk Compliance provided by the engineer who visited the site. This system will assess potential hazards at work sites keeping the engineer’s safety in concern. The hazard and risk can be updated by the admin. It also leverages leaflet API for route mapping from the engineer’s location to the fault location.

To accommodate operational demands, the system should support a throughput of 1000 requests per hour. It can store and replay orders in the event of internal or user/engineer failures.

Furthermore, engineers should be able to access the application from various devices including laptops, tablets, and mobile phones. The holidays are considered while assigning tasks to the engineers and are marked respectively in the calendar of admin. These operations mainly span the geographies of Bangalore.

# REQUIREMENTS

##  Functional

* FR1. Roles include User, Engineer, and Admin.
* FR2. Common login page for users, engineers and admins.
* FR3. Users authenticate before accessing ticket-raising functionality.
* FR4. Users, Engineers, and Admins provide specific information for registration.
* FR5. Feature of security questions in case the user, admin or technician forgets their login credentials.
* FR6. As a user, I should be able to raise a ticket for a new Installation or a Fault.
* FR7. As a user, I should be able to view my current/previously raised tickets.
* FR8. As a user, I should be able to view and update my profile.
* FR9. As a user, I should be notified via email once I have raised a ticket.
* FR10. As a user, I should be notified via email once my ticket has been successfully resolved.
* FR11. As an admin I should be able to view the technician details, tickets raised by user.
* FR12. As an admin, I should be able to view the priority of tickets assigned to engineer based on service types, including fault and installation.
* FR13. As an admin, I should be able to view the assigned tickets to the technician based on distance from the fault location, holidays of the technician, number of tasks assigned to them.
* FR14. As an admin, I should be notified on my dashboard if a ticket has been marked as failed or deferred.
* FR15. As an admin I should be able to reassign the deferred ticket to another technician.
* FR16. As a technician, I should be able to view the details of the ticket raised assigned by the admin such as the fault location, ticket date, etc.
* FR17. As a technician, I should be notified on my dashboard once a ticket has been allotted to me.
* FR18. As a technician, I should be able to update the ticket status to accept or reject
  + FR18.1. Upon acceptance of ticket, I should be able to update the task as SUCCESS, FAILURE, DEFFERED.
* FR19. As a technician, I should be able to accept or reject tasks based on availability and workload.
* FR20. As a technician, I should be able to view hazards for any site, add, update and delete them.
* FR21. As an admin, I should be able to view hazards for any site, add, update and delete them.

##  Non-Functional

* NFR1. Protect user credentials, sensitive information, and ticket data from unauthorized access or disclosure.
* NFR2. Implement access controls and authentication mechanisms.
* NFR3. Efficient handling of authentication and ticket management processes.
* NFR4. Scalability of scheduling processes and engineer assignments.
* NFR5. Scale system to handle concurrent status updates and compliance data requests.
* NFR6. User-friendly interfaces for login, ticket submission, and report exploration.
* NFR7. Reliable acceptance, processing, and storage of ticket data.
* NFR8. Real-time retrieval of hazard information for engineers before task execution.
* NFR9. Engineer task actions and status updates should be reliably processed and recorded, with mechanisms to recover from failures or errors.
* NFR10. User interfaces accessible from laptops, tablets, and mobile devices.
* NFR11. Interfaces designed with intuitive navigation, specifically tailored for field engineers' needs.

##  Architectural Constraint:

* Eureka: is used for Service Registration and Discovery
* Leaflet is used to calculate the distance between user and technician.
* The user data (password) is encrypted with the help of Encrypt-js library.

# SCOPE

##  In-Scope

* + System for task management for field engineers which includes resolving raised tickets.
  + The system needs to accept task requests through a ticketing system and prioritize them based on factors such as the type of service (installation or fault), work location, hazards, traffic congestion, and distance from the engineer's residence.
  + Tasks must be scheduled and allocated to engineers based on the prioritization criteria mentioned above.
  + Field engineers should be able to acknowledge assigned tasks, update task statuses (completion or failure), and provide notifications to upstream systems regarding status updates and any business exceptions encountered.
  + The system should integrate with Hazard and Risk Compliance applications to identify potential hazards at work sites.
  + Engineers should be able to access the system from various devices such as laptops, tablets, and mobile phones.
  + Operations primarily cover the geographies of Bengaluru.

##  Out-off Scope

* + The system should support a throughput of 1000 requests per hour.
  + It should be able to integrate with Google services.
  + Interface with data warehouse to send task details for analytical requirements**.**