Google Summer of Code 2025 - OpenStreetMap Project Proposal
Temporary Road Closures Database and API

Raj Rajeshwar Singh Bisen

April 5, 2025

General Information

- **Name:** Raj Rajeshwar Singh Bisen
- **OSM Account Name:** raj_rajeshwar_singh_bisen
- **Current Occupation:** Bachelor of Technology in Computer Science and Engineering at SNU, GN, India
- **Contact Information:**
  - Email: thatonebipanda@gmail.com
  - GitHub: AnarchistHoneybun

Technical Background

Programming and Technical Skills

Languages

- **Advanced Level:** C, Java, Python, Rust, HTML/CSS, Bash
- **Proficient Level:** PowerShell, C++, JavaScript, TypeScript, LaTeX
- **Functional Understanding:** Haskell, Solidity, Zig, Go

Software and Tools

- **Development Environments:** VSCode, JetBrains IDEs, Visual Studio
- **Version Control:** Git, GitHub
- **Operating Systems:** Arch Linux, Windows

Libraries and Frameworks

- **Web Development:** React, Next.js
- **Scientific Computing:** NumPy, SciPy, Matplotlib, TensorFlow
- **Rust Ecosystem:** Ratatui, Tokio, Charm CLI

Databases and Backend Technologies

- **Relational Databases:** PostgreSQL, MariaDB, MySQL
- **Cloud Databases:** Supabase, Firebase
- **Database Management:** Advanced querying, database design, and optimization

Project-Specific Skills

Considering the Temporary Road Closures Database and API project, I have particularly relevant skills:

- Extensive experience with PostgreSQL, MariaDB, and other SQL databases
- Proficiency in developing robust APIs using Python, Node, and Rust
- Experience with cloud database solutions like Supabase and Firebase
- Capable of creating user interfaces for data entry and visualization

Previous Programming Projects

CredHealthBackend: Blockchain-Powered Dynamic Health Insurance System

- **Purpose:** A hackathon project integrating health metrics (sleep, walking, and food tracking) with insurance policies, dynamically adjusting premiums based on user habits via blockchain smart contracts.
- **Role:** Team of 3; I led the blockchain backend development and co-designed the mobile app frontend.
- **Technical Implementation:**
  - **Blockchain:**
    - Developed Solidity smart contracts to mint insurance policies as NFTs, with functions to adjust health scores and premiums.
    - Deployed a local testnet using Ganache; contracts handled real-time health data fluctuations.
  - **Backend:**
    - Built a Node.js API layer to bridge the React Native frontend (CredHealth repo) and blockchain.
    - Stored food images in AWS S3 and processed them with Python (quantity/quality analysis).
  - **Security:**
    - Ensured data integrity via blockchain immutability; limited PI health information to user device to minimize risk of leak.
- **Challenges & Solutions:**
  - **Gas Fees & Scalability:** Optimized contract functions to minimize transactions during testing.

- **Outcome:**
  - Functional prototype deployed on a local network; demoed on-device with dynamic premium adjustments.
  - Won special mention (4th place overall) in the software projects category.
- **Technologies:** Solidity, Ganache, Node.js, React Native, AWS S3, Python (OpenCV/TensorFlow for image analysis), Git.

Kupyna Hash Function Implementation for RustCrypto

- **Purpose:** Implemented Kupyna (DSTU 7564:2014), a Ukrainian national standard hash function, as part of RustCrypto's mission to provide pure-Rust cryptographic primitives. Based on the IACR specification, this contribution expands RustCrypto's algorithm support for developers requiring standards-compliant hashing.
- **Role:** One of two contributors; proposed and developed the implementation with my collaborator, later adapting it to match RustCrypto's trait system for seamless integration.
- **Technical Implementation:**
  - **Algorithm:**
    - Translated the Kupyna specification into idiomatic Rust.
    - Aligned with RustCrypto's `Digest` trait for interoperability with other crates.
  - **Testing:**
    - Verified correctness against official test vectors from the IACR paper.
    - Integrated into RustCrypto's CI/CD pipeline for automated testing and other checks.
  - **Performance:**
    - Actively optimizing throughput (e.g., leveraging new Galois Multiplication methods and working directly on 64 bit chunks in ongoing work).
- **Challenges & Solutions:**
  - **Trait Integration:** Studied RustCrypto's existing hash implementations (e.g., Groestl) to ensure API consistency.
- **Outcome:**
  - Successfully merged into RustCrypto/hashes and published as part of the official suite.
  - Downstream-ready for use in authentication, blockchain, or other cryptographic applications.
- **Technologies:** Rust, GitHub Actions (CI/CD).

Contributions to the Target Project

As per the GSoC guidelines, I contributed to the **iD editor** (a core OpenStreetMap web application) to demonstrate my ability to work with OSM's ecosystem. While my proposed GSoC project involves creating a new system for Temporary Road Closures (with no existing repository), my Pull Request #10903 to iD showcases relevant technical and collaborative skills for web-based OSM tools.

- **Feature Implementation:** Addressed Issue #10870 by implementing an auto toggle for highlighting unsaved changes on iD editor startup. This helps users quickly identify pending edits from their previous session, improving workflow efficiency.
- **Technical Scope:**
  - Modified the JavaScript codebase to persist and render unsaved change markers.
  - Leveraged iD's state management system to track edits across sessions.
- **Development Process:**
  - Set up the iD project locally, navigating its build system and architecture.
  - Analyzed the editor's event-driven workflow to integrate the feature without disrupting core functionality.
- **Relevance to Proposed Project:** Though my GSoC project would be new, this contribution demonstrates:
  - Experience with web apps.
  - Ability to work within OSM's technical and community standards.
  - Skill in enhancing user-facing tools—critical for the planned Road Closures API's frontend components.

This contribution underscores my readiness to develop web-based tools for OSM, even as I transition to a new codebase for my GSoC project.

Project Proposal

Project Title

Temporary Road Closures Database and API

Project Overview

This project aims to create a prototype system for managing temporary road closures in OpenStreetMap (OSM). The system will include a database for storing closure data, a user-friendly API for navigation apps, and interfaces for data submission with authentication-backed confidence scoring to prevent spam.

Problem Statement

Currently, OSM lacks an efficient way to handle temporary road closures (e.g., construction, events) that last hours or days rather than weeks or months. The main OSM database's update cycle makes it impractical for such short-term changes. This project solves this by creating a separate, real-time system for temporary closures with verified user submissions.

Proposed Solution

- **Database:** PostgreSQL with PostGIS extension - Robust spatial database support needed for geographic queries.
- **API:** Rust-based Actix-Web server - High performance and safety for the API layer (avoids JVM concerns from guidelines).
- **Authentication:** OAuth with OSM accounts - Leverages existing OSM authentication with confidence scoring for submissions.
- **Data Format:** OpenLR - Standardized format for location references, ensuring compatibility with navigation apps.
- **Web UI:** Map-based interface using OSM tiles with form for closure details.
- **Mobile Integration:** Prototype integration with OSMAnd - Popular OSM navigation app with extensible architecture.

Expected Outcomes

- **Core Deliverables:**
  - Functional PostgreSQL database schema for road closures
  - REST API supporting OpenLR format with OSM authentication
  - Web interface with authenticated data submission
  - OSMAnd prototype integration
- **Stretch Goals:**
  - Mobile data collection app
  - Advanced confidence scoring system for submissions
  - Time-based query capabilities
  - Integration with additional navigation apps
- **Fallback Plan:** If OpenLR integration proves too complex, implement a simpler json-based format initially.

Technical Implementation Details

- **System Architecture:** Three-tier architecture (DB, API, UI) deployed on Heroku/Render for easy prototyping.
- **Database Schema:** Tables for closures (geometry, time window, reason), users, and verification status with confidence scores.
- **API Design:** RESTful endpoints for CRUD operations with OpenLR encoding/decoding and OAuth integration.
- **Testing:** Near 100% test coverage for non-UI code using TDD approach.
- **CI/CD:** GitHub Actions for automated testing and SonarCloud integration.

Learning Objectives

- Gain deeper understanding of OSM ecosystem and spatial data handling
- Understand OpenLR specification and implementation
- Implement authentication and confidence scoring systems

Project Timeline

Availability

- **Weekly Availability:** app. 60 hours/week available for GSoC, including hours distributed for work on the timeline, and separate hours for research/implementation trials as needed
- **Focus Period:** Full-time commitment with no other obligations

Project Schedule

Week	Tasks	Hours
1	Research: Study OpenLR spec, OSM auth integration, existing solutions	30
2-3	Database design and implementation with PostGIS	60
4-5	API development (core endpoints, OpenLR support, OAuth integration)	60
6	Buffer Week: Mid-point review and adjustments	30
7-8	Web UI implementation (data submission interface with auth)	60
9-10	OSMAnd integration prototype	60
11	Testing, documentation, and final refinements	30
12	Buffer Week: Final review and community presentation	30

Total Commitment

350 hours (includes buffer time split across the timeline to handle additional requirements/catch up on tasks taking more time than expected)

Communication Plan

Meeting Schedule

- Regular updates every 2-3 days via email thread
- Bi-weekly progress reports on OSM community page
- Sync calls scheduled as needed by mentor

Progress Reporting

- Daily updates via GitHub project board
- Code reviews through GitHub PRs with SonarCloud checks
- Bi-weekly summary reports with test coverage metrics

Use of AI Tools

- **Usage:** Limited to code completion (GitHub Copilot).
- **Verification:** All AI-generated code will be manually reviewed and tested
- **Quality Control:** SonarCloud integration to maintain code quality standards