Enterprise Internship (SVNC.00.262)

Unpaid Internship In Students of Satellite 07.01.2025

Sergei Ivanov

Information Technology Systems Development, University of Tartu

Supervisor: Andre Sääsk

January 7, 2026

Abstract

This report describes the design and implementation of a Discord-integrated project management system developed during an internship at a software company. The system was created to improve internal coordination by unifying organizational workflows, task notifications, and data persistence within a familiar communication platform. Implemented primarily in Rust, the solution consists of a Discord bot interface, a Jira-based reminder service, and a PostgreSQL-backed persistence layer, with optional data synchronization to Google Sheets for non-technical stakeholders. The architecture follows containerized, microservice-ready principles and emphasizes memory safety, concurrency, and secure API usage. Throughout the internship, the system was developed iteratively under weekly supervision, addressing real-world constraints such as restricted API access and deployment consistency. The final result is a scalable, production-ready automation tool that reduces email dependency and enhances team visibility into events, resources, and task status.

Keywords: Discord bot, Rust programming language, project management automation, Jira integration, PostgreSQL, microservices, Docker, REST API

Contents

Internship Report: Discord-Integrated Project Management System

1. Executive Overview

During my internship at [Company Name], I was responsible for the end-to-end development of a custom **Discord Bot ecosystem** designed to streamline organizational workflows. Operating as a solo developer under weekly managerial supervision, I architected a multi-service solution focused on three core pillars:

- **Organizational Automation:** A Discord interface for event and resource management.
- **Notification Orchestration:** A Jira-linked reminder system.
- **Data Persistence & Synchronization:** Integration with PostgreSQL and Google Sheets.

The project was implemented using **Rust**, prioritizing memory safety and high-concurrency performance.

2. System Architecture & Tech Stack

The system follows a microservice-ready architecture, containerized for scalable deployment.

- **Language:** Rust (utilized for its robust type system and seamless `Serenity/Poise` integration).
- **Web Framework:** `Axum` for handling RESTful logic and HTTP protocols.
- **Database:** `PostgreSQL` managed via the `SQLx` crate, supporting compile-time verified queries and automated migrations.
- **Infrastructure:** `Docker` and `Docker Compose` for environment parity; prepared for Kubernetes (K8s) orchestration.

3. Core Functionalities

3.1. Discord Interface

The bot serves as the primary UI for organizational tasks. I implemented a command-driven architecture supporting:

- **Resource Coordination:** Specialized commands for `Carpool` and `Accommodation` pooling to assist in team logistics.
- **Event Management:** Logic for creating/modifying `Polls` and `Events` with active state tracking (Register/Vote).
- **Subscription Logic:** A framework for users to link Discord IDs to Jira instances for personalized notifications.

3.2. Jira Integration Service

To solve the "Email Fatigue" issue—where critical updates are missed in overflowing inboxes—I developed a **Reminder System**. This service polls the Jira REST API at configurable intervals, identifies pending tasks, and pushes direct notifications to users.

- **Security:** Implemented as a read-only (GET) client to maintain a minimal security footprint.

3.3. Google Sheets Integration

Currently in the final implementation phase (pending OAuth2 token issuance), this module functions as a secondary data export layer. While PostgreSQL remains the **Source of Truth**, the Google Sheets API is utilized for stakeholder accessibility, allowing non-technical users to view carpool and event data in a familiar spreadsheet format.

4. Weekly Progress Log

4.1. Phase 1: Research & Discovery (Week 1)

Initial focus was directed toward feasibility studies and environment configuration. I conducted research into the Google Sheets API v4 and evaluated Rust-based client libraries to ensure future-proof integration.

4.2. Phase 2: API Architecture (Week 2)

Developed the Jira REST client. This involved negotiating restricted API scopes to adhere to the **Principle of Least Privilege**, ensuring the bot could only access necessary ticket metadata without compromising company security.

4.3. Phase 3: Bot Development & Local Testing (Week 3)

Initiated the core Discord bot using the `Poise` framework. Development was validated through local deployment on a dedicated staging server, allowing for iterative refinement based on managerial feedback.

4.4. Phase 4: Feature Finalization (Week 4)

Completed the implementation of the carpooling and poll logic. Conducted a comprehensive feature review meeting to demonstrate command reliability and data handling before finalizing the primary feature set.

4.5. Phase 5: Deployment & Environment Hardening (Week 5+)

During the December 2025 – January 2026 phase, I focused on infrastructure: * Standardized the `Dockerfile` for multi-stage builds. * Configured the `PostgreSQL` container with persistent volumes. * Migrated the 1,300+ line codebase to the internal GitLab repository for CI/CD integration.

5. Technical Challenges & Solutions

- **Access Latency:** Progress was initially hindered by credential bottlenecks. I resolved this by pivoting to **Mock Data** and **Interface-driven development**, allowing logic to be written before tokens were issued.
- **Persistence Logic:** Implementing CRUD with `SQLx` migrations ensured that the database schema remains synchronized across all developer environments.

6. Conclusion

The internship resulted in a functional, high-performance system that bridges the gap between project management (Jira) and team communication (Discord). By utilizing Rust, the company received a memory-safe deliverable that is ready for production deployment and future expansion.