Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core components, data structures, and logic for **CivicSync - Public Service Request Optimizer**. CivicSync is a government web platform leveraging agentic-AI to analyze, prioritize, and route citizen service requests, predict resource needs, optimize response times, and provide actionable reports for city officials. The system includes secure authentication, real-time dashboards, and collaborative tools for public sector teams.

---

1. System Components

Component	Technology	Key Responsibilities
Web Frontend	Vue.js	User interface, dashboards, request submission
API Backend	Python (FastAPI)	Business logic, authentication, data orchestration
AI Agent Service	Python	Analyze, prioritize, route requests, predict resources
Database	PostgreSQL	Persistent storage of users, requests, logs
Auth Service	Python/JWT	Secure authentication, role management
Collaboration Module	Python/Vue.js	Team chat, notes, assignment tracking
Reporting Engine	Python	Generate and export actionable reports

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`ServiceRequest`	Citizen service request model	`id`, `type`, `status`, `priority`, `location`, `created_at`
`User`	System user (citizen/official)	`id`, `name`, `role`, `email`, `password_hash`
`AIRequestAnalyzer`	AI agent interface	`analyze(request)`, `predict_resources(request)`
`AuthManager`	Auth/session management	`login()`, `logout()`, `validate_token()`
`DashboardController`	Dashboard data provider	`get_stats()`, `get_active_requests()`
`CollaborationService`	Team collaboration	`post_message()`, `assign_request()`
`ReportGenerator`	Report creation/export	`generate_report()`, `export(format)`

Relationships:

- `ServiceRequest` linked to `User` (creator, assignee)
- `AIRequestAnalyzer` processes `ServiceRequest`
- `CollaborationService` references `ServiceRequest` and `User`

---

3. Data Structure Overview

Table/Model	Fields
`users`	`id`, `name`, `email`, `password_hash`, `role`, `created_at`
`service_requests`	`id`, `user_id`, `type`, `description`, `status`, `priority`, `location`, `created_at`, `updated_at`
`assignments`	`id`, `request_id`, `assignee_id`, `assigned_at`
`messages`	`id`, `request_id`, `user_id`, `content`, `timestamp`
`reports`	`id`, `generated_by`, `created_at`, `data`

---

4. Algorithms/Logic

Service Request Prioritization & Routing (Pseudocode):

```
def process_new_request(request):
    features = extract_features(request)
    priority = ai_agent.predict_priority(features)
    resources = ai_agent.predict_resources(features)
    assignee = route_to_team(priority, location=request.location)
    save_request(request, priority, assignee, resources)
```

Dashboard Update (Summary):

- On new/updated request, push real-time stats to dashboard via WebSocket.

---

5. Error Handling

Scenario	Handling Approach
Invalid authentication token	Return 401 Unauthorized, prompt re-login
AI agent service unavailable	Fallback to default rules, log error
Database connection failure	Retry with backoff, alert admin if persistent
Invalid request data	Return 400 Bad Request with error details
Permission denied	Return 403 Forbidden, log access attempt

End of Document