Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core components, data structures, and logic for **SafeServe - AI Food Safety Inspector**. SafeServe is a web and IoT solution leveraging generative AI to analyze sensor data from food environments, generate compliance reports, and provide actionable alerts for hospitality businesses and regulators.

---

1. System Components

Component	Description	Key Responsibilities
IoT Sensor Gateway	Collects and transmits sensor data	Data ingestion, device authentication
Backend API Server	Central logic, AI analysis, user/session management	Data processing, AI integration, API endpoints
AI Analysis Engine	Processes data, generates reports/alerts	Data analysis, report/alert generation
Web Frontend	User interface for monitoring, reports, management	Data visualization, user actions
MongoDB Database	Stores users, sensor data, reports, alerts	Persistent storage, query support
Auth Service	Handles user authentication and authorization	Secure login, role management

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`SensorDataIngestor`	Handles incoming sensor data	`ingest(data)` , `validate(data)`
`AIAnalyzer`	Runs AI models on sensor data	`analyze(data)` , `generate_report()`
`ReportManager`	Manages compliance reports	`create()` , `export(format)`
`AlertManager`	Handles alert creation/dispatch	`check_thresholds()` , `notify()`
`UserManager`	User/session management	`authenticate()` , `authorize()`
`SensorDevice`	Represents a physical sensor	`id` , `type` , `status` , `lastReading`

Relationships:

- `SensorDataIngestor` receives data from `SensorDevice`
- `AIAnalyzer` processes data, interacts with `ReportManager` and `AlertManager`
- `UserManager` secures access to reports and alerts

---

3. Data Structure Overview

Model	Fields
User	`_id`, `username`, `password_hash`, `role`, `email`, `created_at`
SensorDevice	`_id`, `location`, `type`, `status`, `lastReading`, `owner_id`
SensorData	`_id`, `device_id`, `timestamp`, `data` (JSON: temp, humidity, etc.)
Report	`_id`, `generated_at`, `device_ids`, `summary`, `details`, `compliance_status`, `owner_id`
Alert	`_id`, `created_at`, `device_id`, `type`, `severity`, `message`, `resolved`

---

4. Algorithms/Logic

Sensor Data Processing & Alerting (Pseudocode):

```
def ingest_sensor_data(data):
    if not validate(data):
        log_error("Invalid data")
        return
    store_in_db(data)
    analysis = AIAnalyzer.analyze(data)
    if analysis['alert']:
        AlertManager.notify(analysis['alert'])
    if analysis['report_needed']:
        ReportManager.create(analysis['report'])
```

Report Generation:

- Aggregate recent sensor data
- Run AI compliance checks
- Summarize findings, flag violations
- Store and make exportable (PDF/CSV)

---

5. Error Handling

Scenario	Handling Approach
Invalid sensor data	Log error, discard data, notify admin if frequent
AI model failure	Fallback to rule-based checks, log incident
Database connection error	Retry, alert admin if persistent
Unauthorized access	Return 401/403, log attempt
Device offline	Mark as inactive, alert user
Report export failure	Return error message, allow retry

End of Document