TED UNIVERSITY

CMPE 492 / SENG 492 Senior Project

General AI Safety Systems Low-Level Design Report

Spring 2025

Team Members

Mustafa PINARCI 18853734706 Computer Engineering

Ege İZMİR 12584814676 Computer Engineering

Egemen Doruk SERDAR 71155167474 Software Engineering

Mustafa Boğaç MORKOYUN 44764509874 Software Engineering

Supervisor: Gökçe Nur YILMAZ

Jury Members

Eren ULU

Tansel DÖKEROĞLU

Tolga Kurtuluş ÇAPIN

Contents

1. Introduction

1.1 Object Design Trade-Offs

During the object design stage of the General AI Safety System, we encountered significant tradeoffs concerning aspects such as performance, scalability, and security. We have diligently managed the following considerations:

Performance vs. Accuracy: Using the HOG method with dlib library for face detection ensures reliable performance while maintaining real-time processing capabilities. The face recognition model leverages dlib-face-recognition-resnet-model-v1 for accurate identification.

Maintainability vs. Complexity: We opted for a modular subsystem design to make future maintenance and scalability easier. Although this increases initial complexity, it simplifies future upgrades and troubleshooting.

Accessibility vs. Security: We implemented Role-Based Access Control (RBAC) along with advanced encryption methods to protect data, while still allowing access for those who are authorized.

Stability vs. Innovation: Innovative techniques such as behavioral analysis were integrated, along with established software patterns, like the MVC for subsystem design, to maintain system reliability.

1.2 Interface Documentation Guidelines

All interfaces within the system adhere to well-defined documentation standards to promote clarity and uniformity among components. RESTful APIs are utilized for communication between subsystems.

Guidelines Adopted:

1. **Standardization:**

- RESTful API endpoints, including GET, POST, PUT, DELETE methods, comply with industry practices for interoperability.
- Data types and response formats are distinctly outlined following OpenAPI specifications.,

2. **Clarity:**

- Every API endpoint includes detailed documentation of the method type, input parameters, expected outputs, and error management.
- UML class diagrams illustrate the connections and duties of primary objects.

3. **Security:**

- All inter-component communications are encrypted using TLS 1.3.
- Sensitive data is encrypted using AES-256 during storage.

4. **Consistency:**

- Naming conventions are consistent across packages and class interfaces.

1.3 Engineering Standards

To ensure robust and maintainable design, we adhere to the following standards:

1. **UML (Unified Modeling Language):**
   - Class diagrams for object representation.
   - Sequence diagrams for subsystem interactions.
2. **IEEE 830-1998:**
   - Followed for structuring the software requirements and specifications.
3. **ACM Code of Ethics and Professional Conduct:**
   - Applied to ensure ethical and responsible use of AI technologies.
4. **ISO/IEC 27001:**
   - Implemented for information security management.
5. **GDPR Compliance:**
   - Ensured adherence to data protection laws in data handling processes.

1.4 Definitions, Acronyms, and Abbreviations

- **AI:** Artificial Intelligence
- **GPS:** Global Positioning System
- **GDPR:** General Data Protection Regulation
- **RBAC:** Role-Based Access Control
- **UI:** User Interface
- **TLS:** Transport Layer Security
- **AES:** Advanced Encryption Standard

2. Packages

The system is divided into four main packages, each responsible for specific functionality.

2.1 Attendance Management Package

This package is dedicated to tracking and recording student attendance through AIpowered facial recognition technology.

Classes and Responsibilities:

FaceRecognition:

- Captures and processes student images.
- Detects faces using the Histogram of Oriented Gradients (HOG) method implemented in the dlib library.
- Extracts facial landmarks using the shape_predictor_68_face_landmarks.dat weight file.
- Recognizes faces using the dlib_face_recognition_resnet_model_v1.dat weight file.
- Returns a Boolean indicating verification success.

AttendanceLogger:

- Logs verified student entries and exits in a secure database.
- Associates timestamps with student records.

Data Flow:

- FaceRecognition captures student images when they board.
- The image is processed and matched against the student database.
- Successful matches are logged via AttendanceLogger.

Error Handling:

- If verification fails, the system logs the event and alerts the driver.
- Handles corrupted images and poor lighting conditions with preprocessing.

2.2 Route Optimization Package

This package calculates optimized bus routes in real-time using advanced algorithms like A* and Dijkstra's.

Classes and Responsibilities:

RoutePlanner:

- Computes the most efficient route between start and end locations.
- Supports dynamic re-routing based on traffic data.

TrafficMonitor:

- Integrates with external APIs to monitor traffic and weather.
- Updates the RoutePlanner with new conditions.

Data Flow:

- TrafficMonitor collects live traffic data.
- RoutePlanner recalculates the optimal route and sends it to the driver.

Error Handling:

- Implements fallback routes if data from TrafficMonitor is unavailable.

- Logs errors if pathfinding algorithms fail.

2.3 Notification Package

This package ensures timely communication of safety warnings and updates to interested parties.

Classes and Responsibilities:

AlertManager:

- Creates and manages safety alerts.
- Supports multiple alert levels (INFO, WARNING, CRITICAL).

UserNotifier:

- Sends notifications to users through SMS, email, and in-app messages.

Data Flow:

- AlertManager generates alerts based on system events.
- UserNotifier dispatches these alerts to relevant users.

Error Handling:

- Implements retry logic for failed notifications.
- Logs undelivered messages for later review.

2.4 Data Management Package

This package is responsible for the safe storage, encryption, and recovery of essential information.

Classes and Responsibilities:

DataEncryptor:

- Encrypts and decrypts of sensitive information using AES256 standards.

StorageManager:

- Handles long-term storage and maintains data reliability.

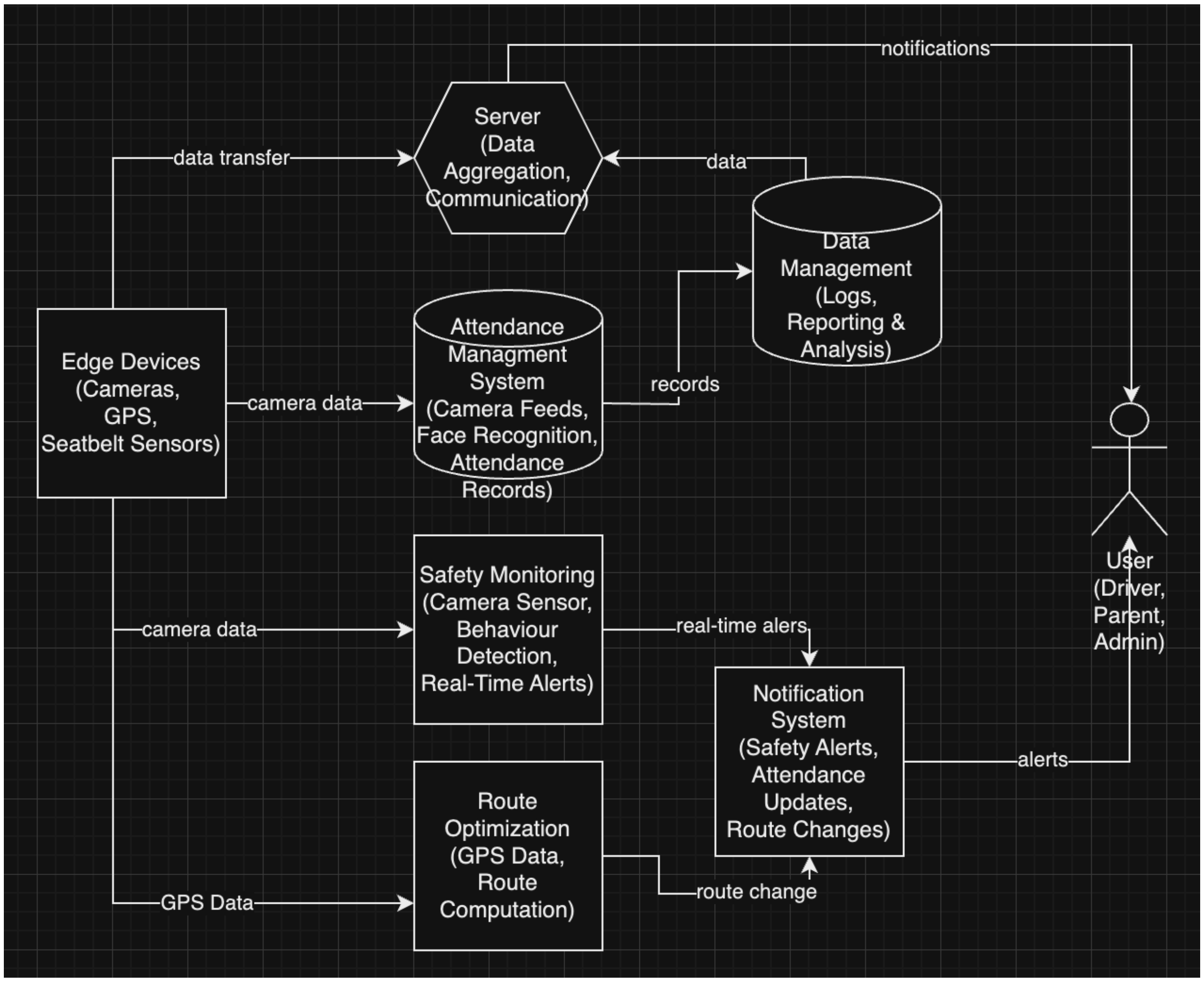
Data Flow:

- DataEncryptor secures data before storage.
- StorageManager keeps saved encrypted data and ensures limited access.

Error Handling:

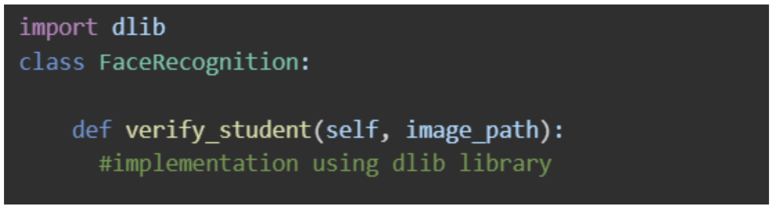
- Logs encryption errors and data corruption.
- Implements regular integrity checks.

2.5 UML Class Diagram


Server (Data Aggregation, Communication)
notifications
data transfer
data
Data Management (Logs, Reporting & Analysis)
Edge Devices (Cameras, GPS, Seatbelt Sensors)
Attendance Managment System (Camera Feeds, Face Recognition, Attendance Records)
camera data
records
User (Driver, Parent, Admin)
Safety Monitoring (Camera Sensor, Behaviour Detection, Real-Time Alerts)
camera data
real-time alers
Notification System (Safety Alerts, Attendance Updates, Route Changes)
alerts
Route Optimization (GPS Data, Route Computation)
GPS Data
route change

3. Class Interfaces

3.1 Component 1: FaceRecognition



```
import dlib
class FaceRecognition:

    def verify_student(self, image_path):
      #implementation using dlib library
```

verifyStudent(image_path): Compares a provided image with the database to verify a student’s identity using HOG-based face detection, facial landmark extraction, and face recognition via dlib.

3.2 Component 2: RoutePlanner

```java
public class RoutePlanner {
    public Route calculateOptimalRoute(Location start, Location end) {
        // Uses A* and Dijkstra's algorithm
    }
}
```

calculateOptimalRoute(start, end): Determines the most efficient path between two locations using real-time traffic data and route optimization algorithms to minimize travel time.

3.3 Component 3: AlertManager

```java
public class AlertManager {
    public void sendAlert(String message, User recipient) {
        // Sends safety alerts
    }
}
```

sendAlert(message, recipient): Sends a system-generated alert to the specified recipient, ensuring timely communication for critical updates or emergency notifications.

3.4 Component 4: DataEncryptor

```java
public class DataEncryptor {
    public String encryptData(String data);
    public String decryptData(String encryptedData);
}
```

encryptData(data): Encrypts sensitive data using a secure cryptographic algorithm to protect it from unauthorized access.

decryptData(encryptedData): Decrypts previously encrypted data, ensuring that only authorized users can access the original information.

4. Glossary

- **Face Recognition:** Technology that utilizes AI to identify and verify individual identities based on facial characteristics using HOG and dlib-based models.
- **Dynamic Routing:** A realtime approach to adjusting routes using algorithms based on traffic and environmental variables.

- **Behavioral Analysis:** Monitoring powered by AI to detect and report unsafe behaviors during transit.
- **RBAC:** RoleBased Access Control is a security strategy that limits access based on users' roles.
- **Notification System:** A system for providing realtime alerts and updates to key stakeholders.

5. References

1. ACM Code of Ethics and Professional Conduct. (https://www.acm.org/code-of-ethics)
2. "Object-Oriented Software Engineering, Using UML, Patterns, and Java," 2nd Edition.
3. GDPR Guidelines. (https://www.edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_3_2018_territorial_scope_after_public_consultation_en_1.pdf)
4. ISO/IEC 27001 Information Security Management Standards. (https://www.iso.org/standard/27001)
5. NHTSA Child Safety Guidelines. (https://www.nhtsa.gov/road-safety/child-safety)
6. HOG Face Detection Overview: (https://dlib.net/face_detector.py.html)
7. dlib Face Recognition Model Documentation: (https://dlib.net/face_recognition.py.html)