# Comprehensive Exercise Report

Team <<TeamRed>> of Section <<DIP392>>

<<Eymen Ucdal (231ADB006)

Tufan Yilmaz (231ADB089)

Ata Mert Pekcan (231ADB010)

Ali Can Eygay (231ADB027)

Renas Alp (211ADB112)

Ataberk Akcin (211AIB121)

Babak Gasimzade (221ADB125) >>

NOTE: You will replace all placeholders that are given in <<>>

# Requirements/Analysis

Week 2

## Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
    - << The project is a face recognition-based access control system that verifies employees' faces before granting access to a workplace, replacing traditional RFID card-based entry to improve security and prevent unauthorized access.>>
        - << The system must detect and recognize faces using a camera.
        - It should check against a database of registered employees.
        - If a face is recognized, the system should trigger a "Gate Open" signal (or simulate it via a web interface).
        - If the face is not recognized, access should be denied and logged.
        - The system should have an admin panel to add/remove employee records.
        - All access attempts should be logged for security tracking.
        - >>
- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
    - << "  Should the system support multiple entry points?
    - Yes, but we are focusing on a single entry point for this project.
    - How should the admin register new employees into the system?
    - Through a simple web interface where an admin uploads images of employees.
    - What happens if an employee is not recognized?
    - They will be denied entry, and an alert can be logged.
    - Do we need liveness detection to prevent spoofing (e.g., using a photo)?
    - Not required, but it would be a good feature for future versions.
    - Should the system work in low-light conditions?
    - It should be tested under different lighting conditions.
    - >>
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - << https://docs.opencv.org/  https://flask.palletsprojects.com/
    - >>
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - << Employees: Use the system for workplace entry.
    - Security/Admin Staff: Manage employee records and view access logs.
    - IT Support Staff: Troubleshoot technical issues and maintain the system.
      >>
- Describe how each user would interact with the software
    - << Employees:
    - Approach the entrance gate (or simulated web interface).
    - Camera scans their face and matches it with stored records.
    - If matched, the system grants access and logs the entry.
    - If not matched, the system denies entry and logs the attempt.

    - Admin/Security Staff:
    - Login to the web dashboard (via Flask web interface).
    - Add new employees (upload images for face recognition).
    - View access logs (successful and denied entries).
    - Remove employees who are no longer with the company.
    - >>
- What features must the software have? What should the users be able to do?
    - << **Face Recognition System:**
    - Detect faces using a webcam or security camera.
    - Compare detected faces against a database of registered employees.
    - **Access Control & Simulation:**
    - If recognized, show "Access Granted" and simulate gate opening.
    - If not recognized, show "Access Denied" and log the attempt.
    - **Web Interface:**
    - Admin dashboard for registering new employees.
    - Access log history for security tracking.
    - Optional live camera feed preview.
    - **Logging & Security:**
    - Store entry attempts (timestamp, employee ID, success/failure).
    - Prevent unauthorized access attempts.
    - >>
- Other notes:
    - << The initial prototype will simulate gate opening via a web interface.
    - If needed, the system can later be expanded to work with real IoT hardware (Raspberry Pi + relay).
    - Testing will include different lighting conditions and face angles.
    - Future improvements: Liveness detection, multi-entry point support.
    - >>

# Software Requirements

<< The Face Recognition-Based Access Control System is designed to improve workplace security by replacing traditional RFID card-based entry with face recognition technology. Employees will no longer need to carry access cards, reducing the risk of lost, stolen, or misused access credentials. The system will use a live camera feed to capture faces, match them against a database of registered employees, and determine whether access should be granted or denied. If access is granted, the system will simulate the gate opening (through a web-based interface for this prototype).

An admin panel will allow security personnel to manage employee records, register new users, and monitor access logs to track all entry attempts. This project will serve as a proof of concept and could later be expanded to integrate with physical gate control systems.

Functional Requirements
Face Recognition & Access Control
FR-1: The system shall capture a live image from the camera when an employee approaches.
FR-2: The system shall detect and extract faces from the captured image.
FR-3: The system shall compare the detected face against a database of registered employees.
FR-4: If a match is found, the system shall grant access and simulate gate opening.
FR-5: If a match is not found, the system shall deny access and log the attempt.

Web-Based Admin Panel
FR-6: The system shall provide an admin dashboard accessible via a web browser.
FR-7: Admins shall be able to add new employees to the system by uploading face images.
FR-8: Admins shall be able to remove employees from the system.
FR-9: Admins shall be able to view a log of all access attempts, including timestamps and results (granted/denied).
FR-10: Admins shall be able to search logs by date or employee name.

Security & Logging
FR-11: The system shall log all access attempts in a database, storing the date, time, employee ID (if recognized), and access status.
FR-12: The system shall flag repeated failed access attempts for security review.

**Use Cases**
Use Case 1: Employee Entry Process
Preconditions:

The employee is registered in the system.
The camera and face recognition system are operational.
Main Flow:

The employee approaches the entry point.
The system captures a live image from the camera.
The system detects and extracts the employee's face.
The system compares the detected face with stored records.
If a match is found, access is granted and the gate is opened.
The system logs the access attempt.
Sub Flows:

[S1] If multiple faces are detected, the system processes only the largest (closest) face.
Alternative Flows (Error Handling):

[E1] If no face is detected, the system prompts the user to step into the camera’s view.
[E2] If the system fails to recognize the face, it denies access and logs the attempt.
Use Case 2: Admin Adds a New Employee
Preconditions:

The admin is logged into the system.
Main Flow:

The admin navigates to the employee registration page.
The admin enters the employee's name and uploads a photo.
The system processes and stores the face embedding in the database.
The system confirms that the employee has been added.
Alternative Flows:

[E1] If the face is not clear, the system asks the admin to upload a better image.
[E2] If the employee is already registered, the system prevents duplicate registration.
Use Case 3: Admin Views Access Logs
Preconditions:

The admin is logged into the system.
Main Flow:

The admin navigates to the Access Logs page.
The system displays all recorded access attempts.
The admin can filter logs by date or employee name.
Alternative Flows:

[E1] If no records are found for a search query, the system displays "No matching entries found."
.>>

# Black-Box Testing

Instructions: Week 4

## Journal

***Remember:*** Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
    - <<The input consists of:
    - Employee face images captured by a live camera feed.
    - Employee registration data (name, employee ID, and uploaded face image).
    - Admin commands (adding/removing users, searching logs).>>
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - <<The system generates the following outputs:
    - Access Granted Message: If the face matches an employee in the database, the system simulates the gate opening.
    - Access Denied Message: If the face is not recognized, access is denied, and the attempt is logged.
    - Admin Dashboard Updates: Admins can see logs of entry attempts and manage users.>>
- What equivalence classes can the input be broken into?
    - <<Equivalence classes for testing:
    - Valid Input - Registered Employee: Face is detected, recognized, and access is granted.
    - Invalid Input - Unregistered Face: Face is detected but not in the database; access is denied.
    - No Face Detected: The system prompts the user to adjust their position.
    - Multiple Faces Detected: The system processes the largest (closest) face.
    - Admin Operations: Adding/removing employees, viewing logs.>>
- What boundary values exist for the input?
    - << Boundary conditions to test:
    - Face Recognition Confidence Level: If confidence is close to the threshold (e.g., 90% vs. 91%), does the system grant or deny access?
    - Lighting Conditions: Test under bright light, dim light, and shadows.
    - Camera Angles: Test with different head positions (frontal, slightly turned, extreme side).
    - Multiple Face Entries: If two employees enter at once, does the system detect and process only the closest face?>>
- Are there other cases that must be tested to test all requirements?
    - <<Additional test cases:
    - False Positives: The system mistakenly grants access to an unregistered face.
    - False Negatives: The system fails to recognize a registered employee.
    - Database Failures: What happens if the database connection is lost?
    - Web Panel Security: Ensure only admins can access user management features.>>
- Other notes:
    - <<Ensure the system performs within acceptable response time limits (≤ 2 seconds per recognition).
    - Consider adding liveness detection in future updates to prevent spoofing (e.g., using a printed photo).

- Logging and reporting must be clear and accessible for admin users.>>

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

| Test ID | Description | Expected Results | Actual Results |
|---|---|---|---|
| TC-01 | Recognized employee face | ✅ Access Granted | ✅ / ❌ |
| TC-02 | Unrecognized face | ❌ Access Denied | ✅ / ❌ |
| TC-03 | No face detected | ❌ Prompt user to adjust position | ✅ / ❌ |
| TC-04 | Face partially visible | ❌Request clearer image | ✅ / ❌ |
| TC-05 | Employee wearing mask/glasses | ✅ Recognize employee | ✅ / ❌ |
| TC-06 | Two faces in the frame | ✅ Process closest face | ✅ / ❌ |
| TC-07 | Poor lighting conditions | ✅ / ❌ Recognition based on accuracy | ✅ / ❌ |
| TC-08 | Admin adds new employee | ✅ Employee added successfully | ✅ / ❌ |
| TC-09 | Admin removes employee | ✅ Employee removed successfully | ✅ / ❌ |
| TC-10 | System logs entry attempt | ✅ Entry recorded in logs | ✅ / ❌ |

# Design

Instructions: Week 6

## Journal

***Remember:*** You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- List the nouns from your requirements/analysis documentation.
    - << Extracted nouns that may represent classes or attributes:
    - Employee (Represents a worker using the system)
    - Face (Captured and analyzed by the system)
    - Access Log (Records successful and failed entry attempts)
    - Admin (Manages employees and system settings)
    - Camera (Captures face images for recognition)
    - Database (Stores employee and log data)
    - Recognition System (Processes images and determines access)>>
- Which nouns potentially may represent a class in your design?
    - <<
        - Class — Description
        - Employee — Represents an employee with stored face data
        - FaceRecognition — Handles face detection and matching against the database
        - AccessControl — Determines if access is granted or denied
        - Admin — Manages employees and reviews access logs
        - AccessLog — Stores details of all access attempts
        - DatabaseManager — Handles interactions with the database
        - Camera — Captures live images for processing>>
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - <<
        - Class — Attributes (Fields)
        - Employee — employeeID, name, faceData
        - FaceRecognition — cameraFeed, detectedFace, matchConfidencee
        - AccessControl — accessStatus, timestamp, logEntry
        - Admin — username, password, manageEmployees(), viewLogs()
        - AccessLog — entryID, employeeID, status, timestamp
        - DatabaseManager — connection, fetchEmployee(), updateLog()
        - Camera — cameraID, resolution, captureImage()>>
- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
    - <<Option 1: Monolithic System (All-in-One Class)
    - Pros:
    - Simpler to implement
    - Easier for a single developer

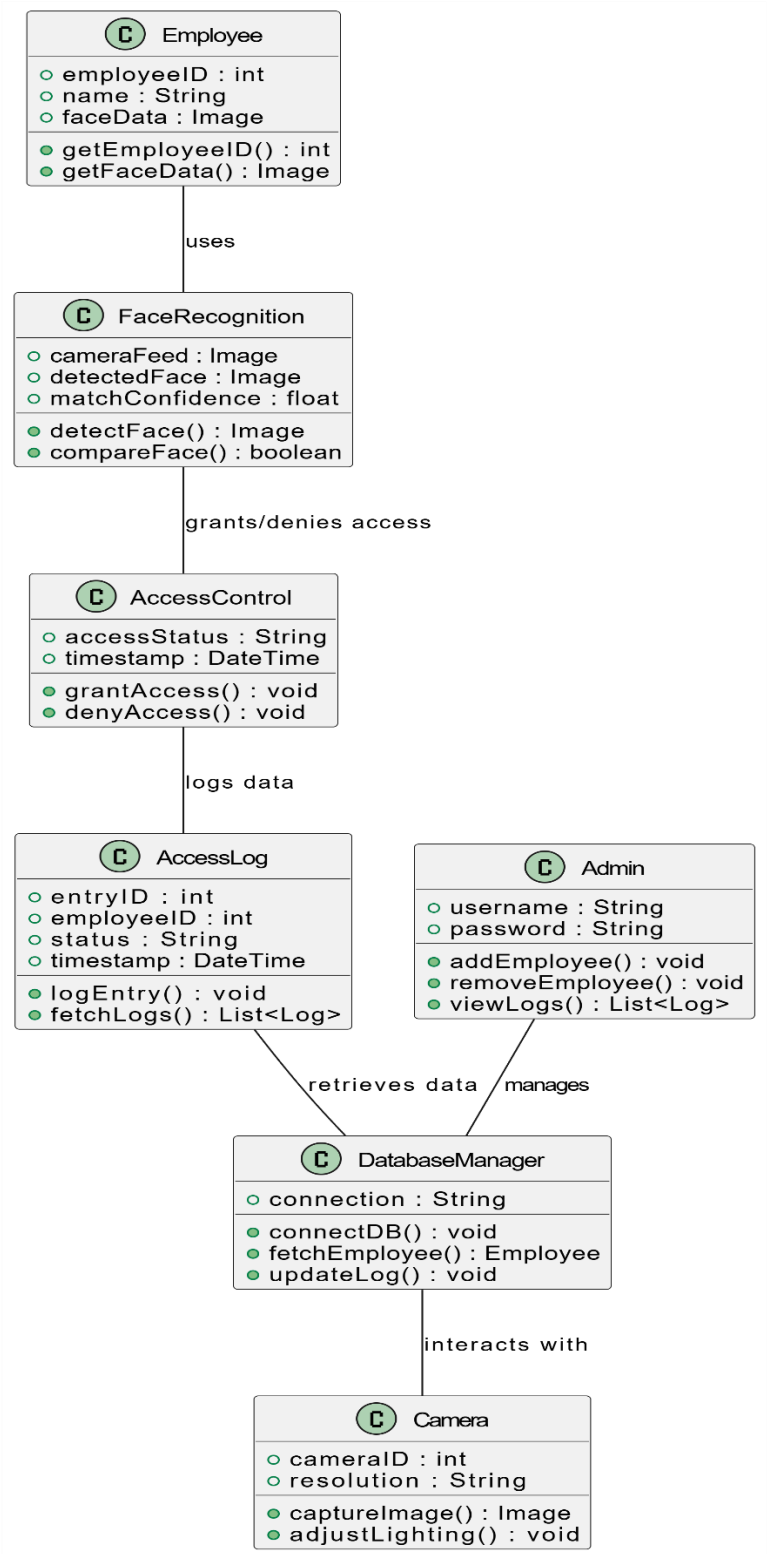
    - Cons:
    - Harder to scale for multiple entry points
    - Code becomes harder to maintain
    - Option 2: Modular System (Multiple Classes for Different Roles)
    - Pros:
    - More scalable (supports multiple entry points)
    - Better separation of concerns (cleaner architecture)
    - Easier to debug and extend
    - Cons:
    - Requires careful integration between modules
    - Slightly more complex setup>>
- Which design do you plan to use? Explain why you have chosen this design.
  <<We selected Option 2 (Modular System) because:
    - It allows for future scalability (e.g., adding liveness detection).
    - It separates functionalities, making it easier to maintain and debug.
    - The system can be expanded for multiple gates in the future.>>
- List the verbs from your requirements/analysis documentation.
    - <<Key actions in the system:
    - Capture face → (Handled by Camera class)
    - Recognize employee → (Handled by FaceRecognition class)
    - Grant or deny access → (Handled by AccessControl class)
    - Log entry attempt → (Handled by AccessLog class)
    - Manage employees → (Handled by Admin class)
    - Retrieve stored face data → (Handled by DatabaseManager class)>>
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.

| << Class | Methods |
|---|---|
| Employee | getEmployeeID(), getFaceData() |
| FaceRecognition | detectFace(), compareFace() |
| AccessControl | grantAccess(), denyAccess() |
| Admin | addEmployee(), removeEmployee(), viewLogs() |
| AccessLog | logEntry(), fetchLogs() |
| DatabaseManager | connectDB(), fetchEmployee(), updateLog() |
| Camera | captureImage(), adjustLighting()>> |

- Other notes:
    - <<Insert notes>>

# Software Design

<<Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will used in the software. Your design should include UML class diagrams along with method headers. ***Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.***>>



# Implementation

Instructions: Week 8

## Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
    - <<Insert answer>>
- Other notes:
    - <<Insert notes>>

# Implementation Details

<<Use your notes from above to write code and complete this section of the formal documentation with a README for the user that explains how he/she will interact with the system.>>

# Testing

Instructions: Week 10

## Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
    - <<Insert answer>>
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - <<Insert class>>
        - <<Insert needed tests>>
    - <<Insert class and tests for each class>>
- Other notes:
    - <<Insert notes>>

# Testing Details

<<Use your notes from above to write your test programs and complete this section of the formal documentation by creating a list of your test programs along with descriptions of what they are testing. You will also complete the black-box test plan by running the program and filling in the Actual Results column.>>

# Presentation

Instructions:Week 12

## Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
    - <<Insert answer>>
- Describe your requirement assumptions/additions.
    - <<Insert answer>>
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - <<Insert answer>>
- How did the extension affect your design?
    - <<Insert answer>>
- Describe your tests (e.g., what you tested, equivalence classes).
    - <<Insert answer>>
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - <<Insert answer>>
- What functionalities are you going to demo?
    - <<Insert answer>>
- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
    - <<Insert answer>>
- Other notes:
    - <<Insert notes>>

<<Use your notes from above to complete create your slides and plan your presentation and demo.>>