Chapter 1

Individual Contribution

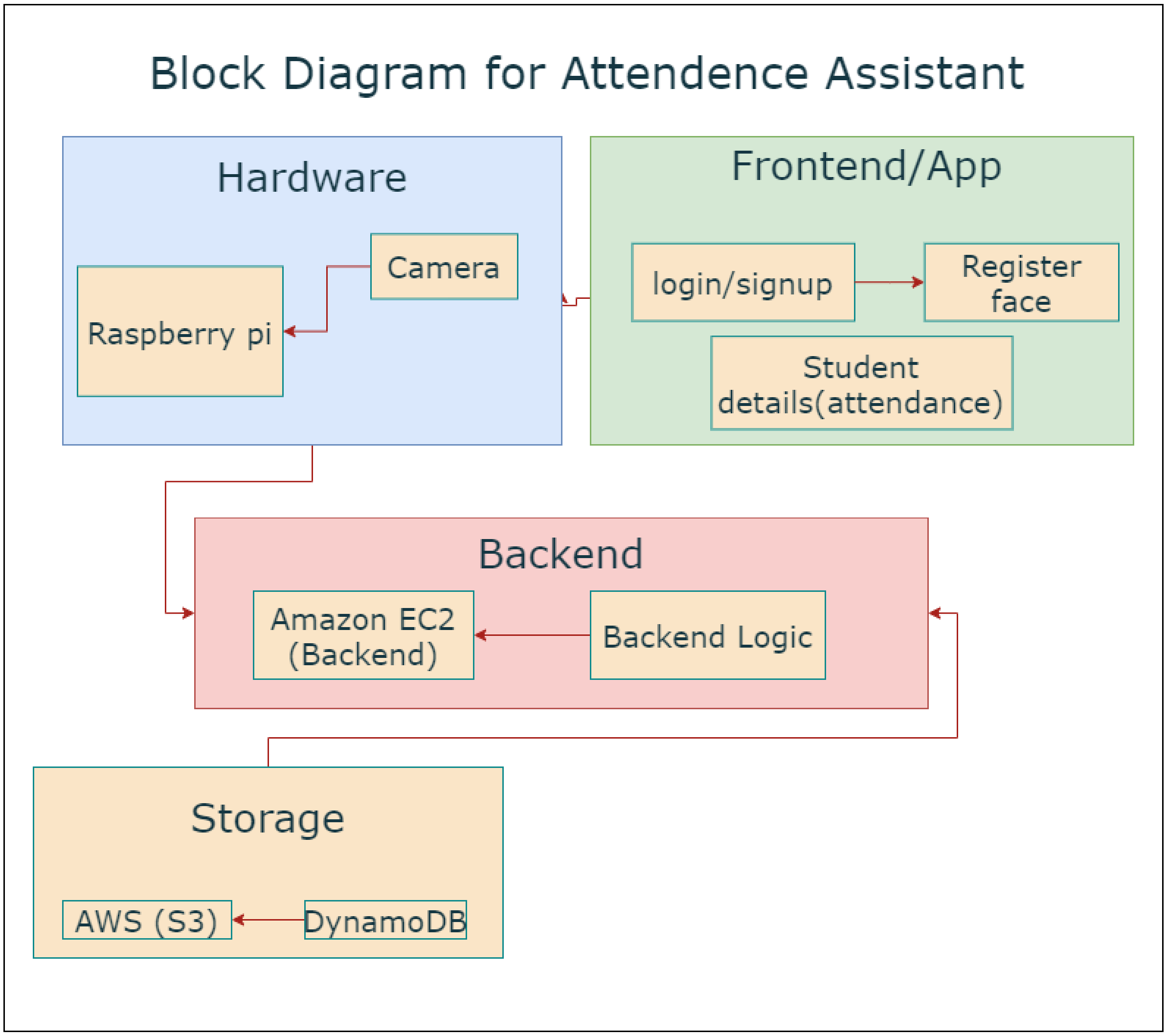


Figure 1.1: Block Diagram highlighting the modules supported by Parth Zarekar.

1.1 Problem Statement

Design and implement the backend API and face-recognition engine for the Attendance-Assistant system.

1.2 Student Details

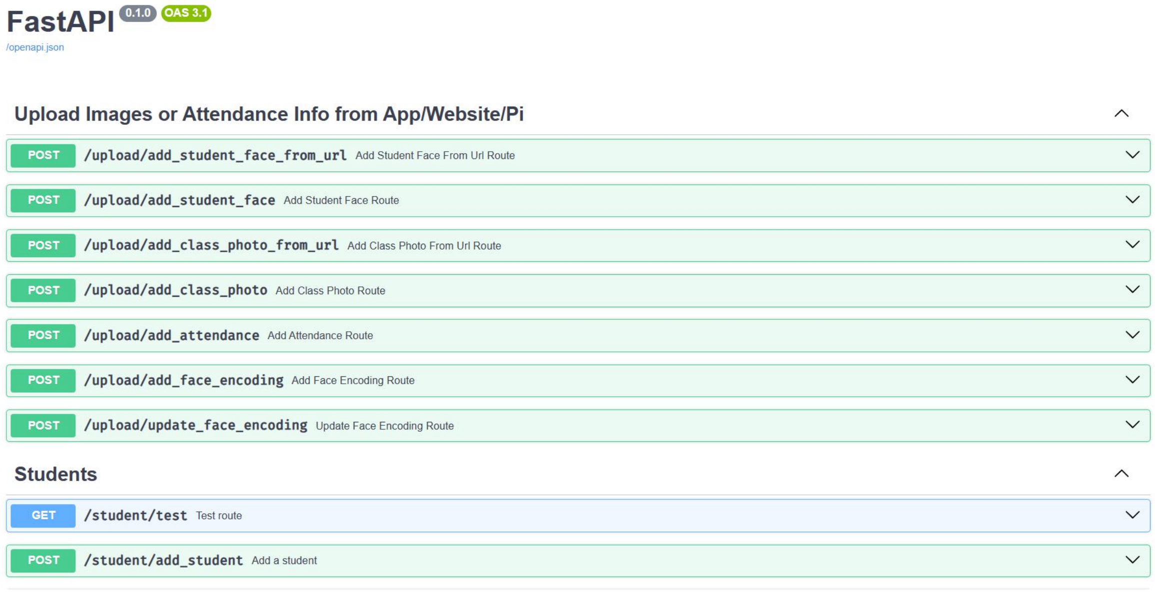
Krishnaraj Thadesar
PRN: 1032210888
Roll Number: 15
Panel: A

1.3 Module Title

Backend & Face-Recognition Engine

1.4 Project's Module Scope (Individual Perspective)

End-to-end implementation of all backend services, face-encoding storage and lookup, handling concurrent API calls from clients, all hosted locally via Docker.



FastAPI 0.1.0 OAS 3.1
/openapi.json

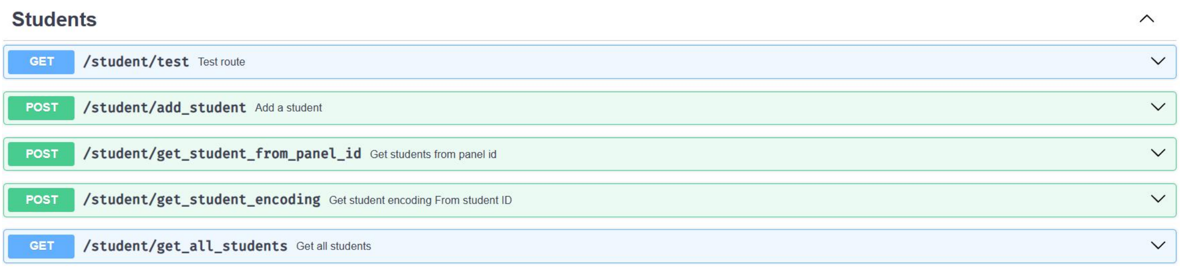
Upload Images or Attendance Info from App/Website/Pi

POST `/upload/add_student_face_from_url` Add Student Face From Url Route
POST `/upload/add_student_face` Add Student Face Route
POST `/upload/add_class_photo_from_url` Add Class Photo From Url Route
POST `/upload/add_class_photo` Add Class Photo Route
POST `/upload/add_attendance` Add Attendance Route
POST `/upload/add_face_encoding` Add Face Encoding Route
POST `/upload/update_face_encoding` Update Face Encoding Route

Students

GET `/student/test` Test route
POST `/student/add_student` Add a student

Figure 1.2: Swagger UI for API documentation (Krishnaraj Thadesar's contribution).



Students

GET /student/test Test route

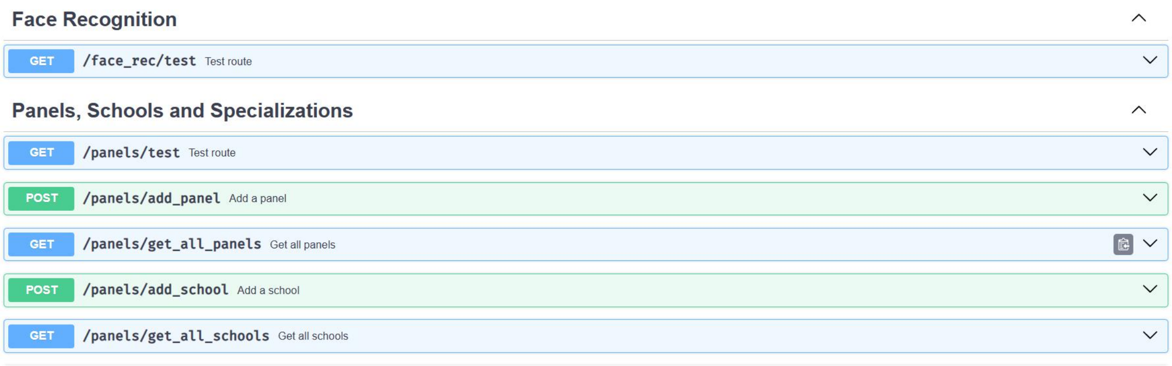
POST /student/add_student Add a student

POST /student/get_student_from_panel_id Get students from panel id

POST /student/get_student_encoding Get student encoding From student ID

GET /student/get_all_students Get all students

Figure 1.3: Swagger UI for API documentation (Krishnaraj Thadesar's contribution).



Face Recognition

GET /face_rec/test Test route

Panels, Schools and Specializations

GET /panels/test Test route

POST /panels/add_panel Add a panel

GET /panels/get_all_panels Get all panels

POST /panels/add_school Add a school

GET /panels/get_all_schools Get all schools

Figure 1.4: Swagger UI for API documentation (Krishnaraj Thadesar's contribution).

Module Interfaces

The FastAPI application exposes the following routes (defined in `main.py` and router files):

- **Add Attendance** `POST /api/v1/add_attendance` *Request body:*

  ```
  {
    "room_id": "Room ID",
    "subject_id": "Subject ID",
    "teacher_id": "Teacher ID",
    "panel_id": "Panel ID",
    "start_time": "10:00",
    "end_time": "11:00"
  }
  ```

- **Add Image** `POST /api/v1/add_image` *Request body:*

  ```
  {
    "room_id": "Room ID",
    "image": "Base64-encoded image"
  }
  ```

  Response:

```
{
  "status": "success",
  "message": "Image added successfully"
}
```

- **Add Specialization** `POST /api/v1/add_specialization`
- **Add School** `POST /api/v1/add_school`
- **Add Panel** `POST /api/v1/add_panel`
- **Add Student** `POST /api/v1/add_student`
- **Add Face Image** `POST /api/v1/add_face_image`
- **Add Face Encoding** `POST /api/v1/add_face_encoding`
- **Add Teacher** `POST /api/v1/add_teacher`
- **Add Semester** `POST /api/v1/add_semester`
- **Add Subject** `POST /api/v1/add_subject`
- **Get Students** `POST /api/v1/get_students`
- **Get Teachers** `POST /api/v1/get_teachers`

Module Dependencies

- `face_recognition` → `dlib`, `numpy`
- `FastAPI` → `uvicorn`, `pydantic`
- MongoDB driver (`motor`)

Module Design

Layered architecture: Controller → Service → Model → Persistence; singleton face-model loader; JWT authentication middleware.

Module Implementation

- Containerized services with Docker Compose.
- Approximately 1,200 lines of Python code.
- Integrated face_recognition pipeline with error handling.

Module Testing Strategies

- Unit tests via `pytest` (coverage >= 85%).
- Mocked face detection for CI.
- Postman end-to-end smoke tests.

Module Deployment

- Fully hosted on local Docker Compose setup.
- Single-command bring-up of all services (backend, database, model).
- Manual rollback by re-deploying previous Docker image versions.

Chapter 2

Individual Contribution

2.1 Problem Statement

Support the full-stack development cycle by contributing to UI design, API development, research, testing, and deployment for the Attendance-Assistant system.

2.2 Student Details

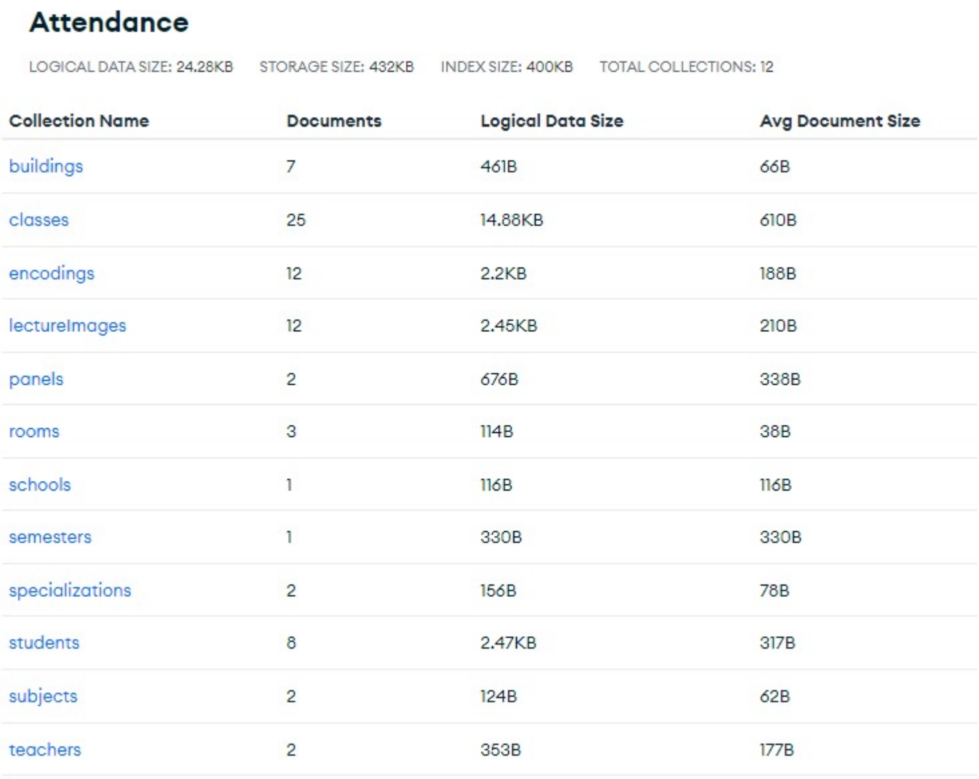
Parth Zarekar
PRN: 1032210846
Roll Number: 09
Panel: A

2.3 Module Title

Full-Stack Support & Research

2.4 Project Module Scope

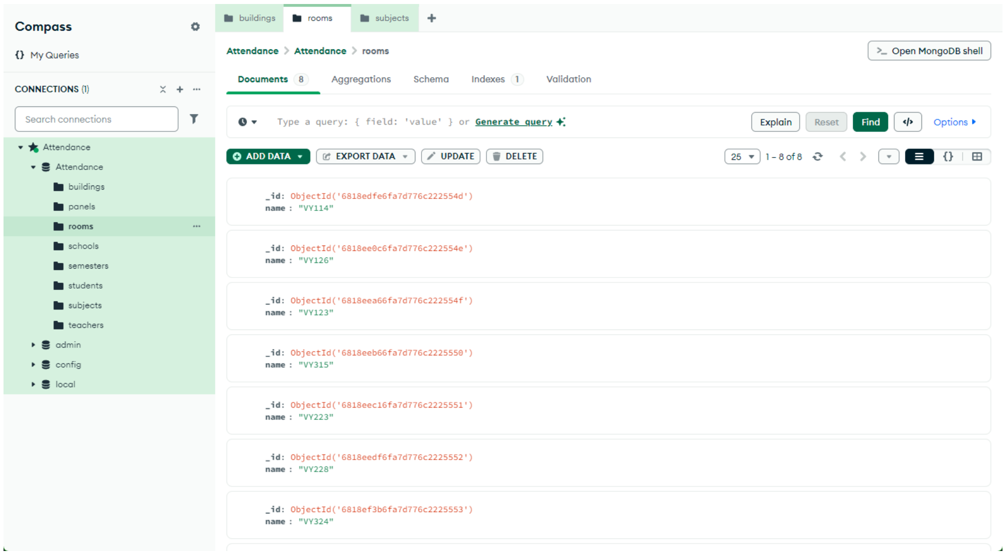
Assisted across UI design, backend API development, model-training research, paper drafting, testing, and deployment.



LOGICAL DATA SIZE: 24.28KB STORAGE SIZE: 432KB INDEX SIZE: 400KB TOTAL COLLECTIONS: 12

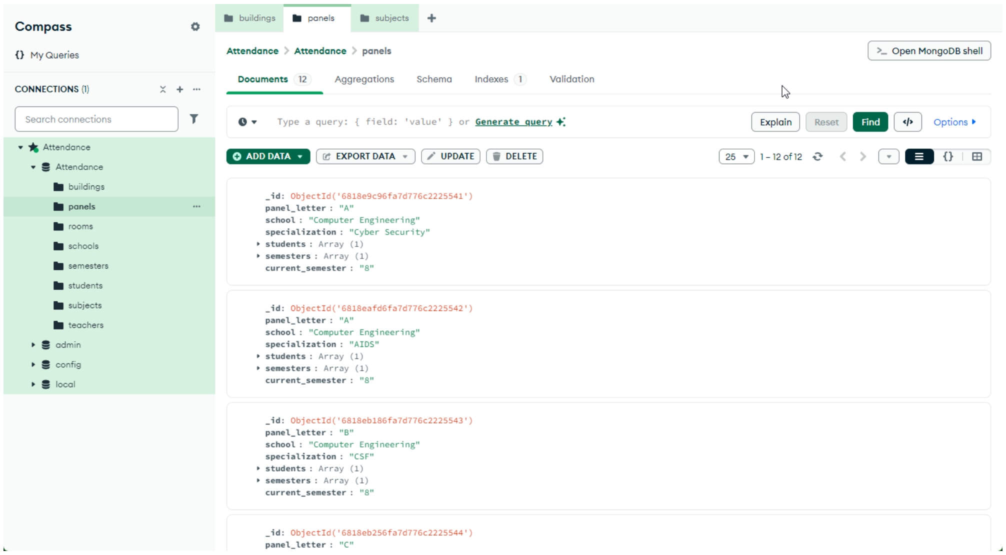
Collection Name	Documents	Logical Data Size	Avg Document Size
buildings	7	461B	66B
classes	25	14.88KB	610B
encodings	12	2.2KB	188B
lectureImages	12	2.45KB	210B
panels	2	676B	338B
rooms	3	114B	38B
schools	1	116B	116B
semesters	1	330B	330B
specializations	2	156B	78B
students	8	2.47KB	317B
subjects	2	124B	62B
teachers	2	353B	177B

Figure 2.1: MongoDB Collections (Parth Zarekar's area)



```
_id: ObjectId('6818edfe6fa7d776c222554d')
name : "VY114"

_id: ObjectId('6818ee0c6fa7d776c222554e')
name : "VY126"

_id: ObjectId('6818eea66fa7d776c222554f')
name : "VY123"

_id: ObjectId('6818eeb66fa7d776c2225550')
name : "VY315"

_id: ObjectId('6818eec16fa7d776c2225551')
name : "VY223"

_id: ObjectId('6818eedf6fa7d776c2225552')
name : "VY228"

_id: ObjectId('6818ef3b6fa7d776c2225553')
name : "VY324"
```

Figure 2.2: MongoDB Collections (Parth Zarekar's area)



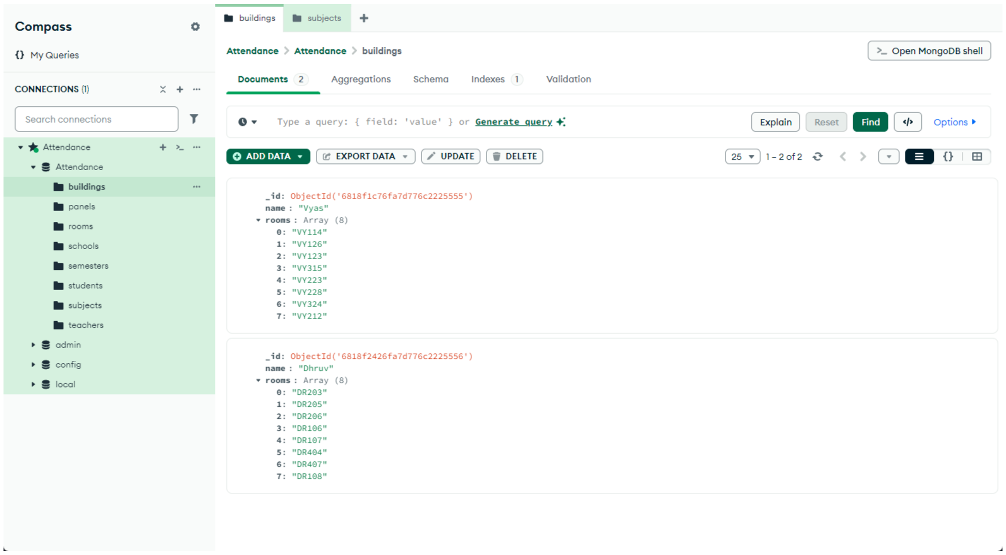
```
_id: ObjectId('6818e9c96fa7d776c2225541')
panel_letter : "A"
school : "Computer Engineering"
specialization : "Cyber Security"
students : Array (1)
semesters : Array (1)
current_semester : "8"
```

```
_id: ObjectId('6818eafd6fa7d776c2225542')
panel_letter : "A"
school : "Computer Engineering"
specialization : "AIDS"
students : Array (1)
semesters : Array (1)
current_semester : "8"
```

```
_id: ObjectId('6818eb186fa7d776c2225543')
panel_letter : "B"
school : "Computer Engineering"
specialization : "CSF"
students : Array (1)
semesters : Array (1)
current_semester : "8"
```

```
_id: ObjectId('6818eb256fa7d776c2225544')
panel_letter : "C"
```

Figure 2.3: MongoDB Collections (Parth Zarekar's area)



```
_id: ObjectId('6818f1c76fa7d776c2225555')
name : "Vyas"
rooms : Array (8)
  0: "VY114"
  1: "VY126"
  2: "VY123"
  3: "VY315"
  4: "VY223"
  5: "VY228"
  6: "VY324"
  7: "VY212"
```

```
_id: ObjectId('6818f2426fa7d776c2225556')
name : "Dhruv"
rooms : Array (8)
  0: "DR203"
  1: "DR205"
  2: "DR206"
  3: "DR106"
  4: "DR107"
  5: "DR404"
  6: "DR407"
  7: "DR108"
```

Figure 2.4: MongoDB Collections (Parth Zarekar's area)

2.5 Project Modules – Individual Contribution

1. **Frontend:** Provided feedback and enhancements on Figma wireframes and UI flows.
2. **Backend API:** Implemented core endpoints for image upload, face encoding, and attendance marking.
3. **Model Research:** Supported training experiments and benchmark comparisons for face-recognition models.
4. **Literature Research:** Drafted and edited sections of the project research paper on algorithm selection.
5. **Testing:** Created and executed end-to-end tests (API smoke tests, basic UI checks).

6. **Deployment:** Deployed Dockerized services to a basic AWS environment and configured DynamoDB storage.

Chapter 3

Individual Contribution

3.1 Problem Statement

Evaluate and benchmark multiple face-recognition algorithms; support model selection and integration.

3.2 Student Details

Sourab Karad
PRN: 1032211150
Roll Number: 40
Panel: A

3.3 Module Title

Algorithm Research & Model Integration

3.4 Project Module Scope

Implementation and evaluation of face-recognition methods; performance reporting and API stub delivery.

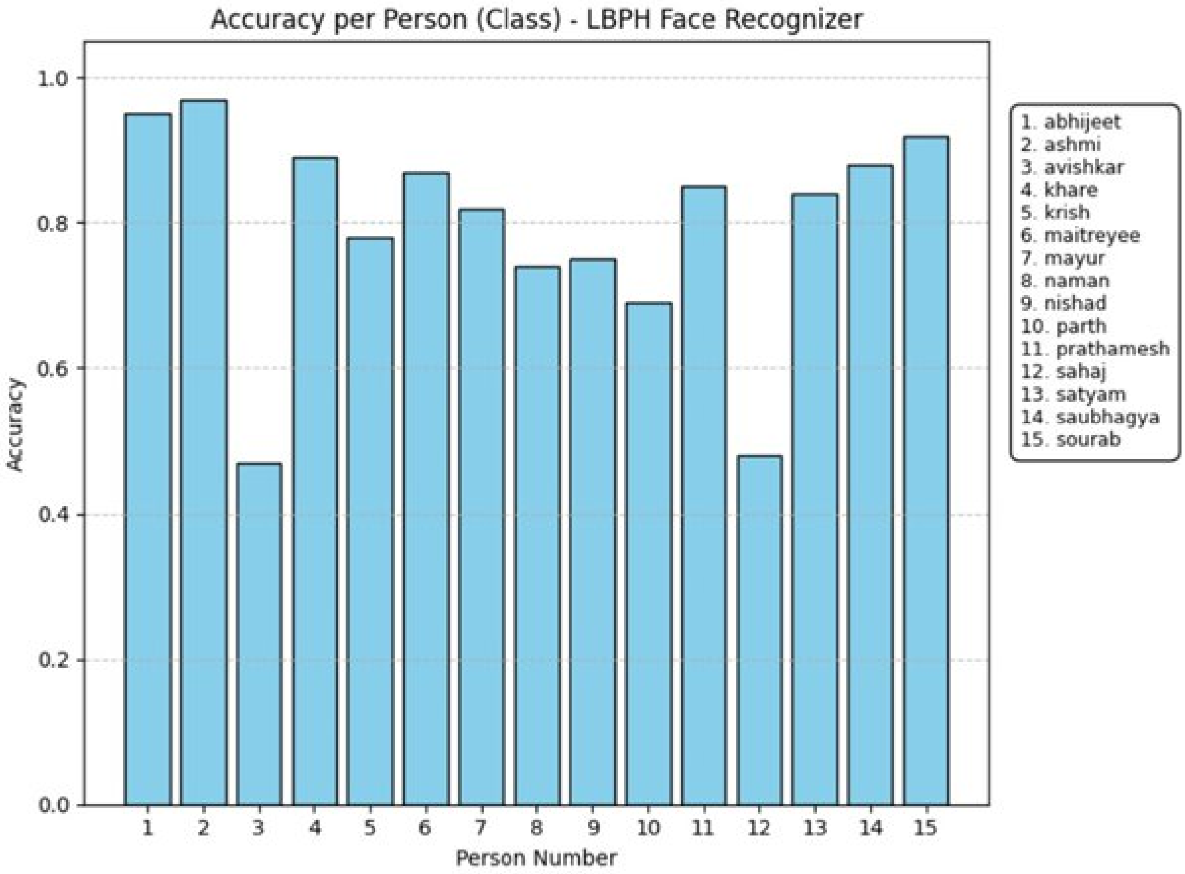


Figure 3.1: Accuracy per Person (Class) - LBPH Face Recognizer

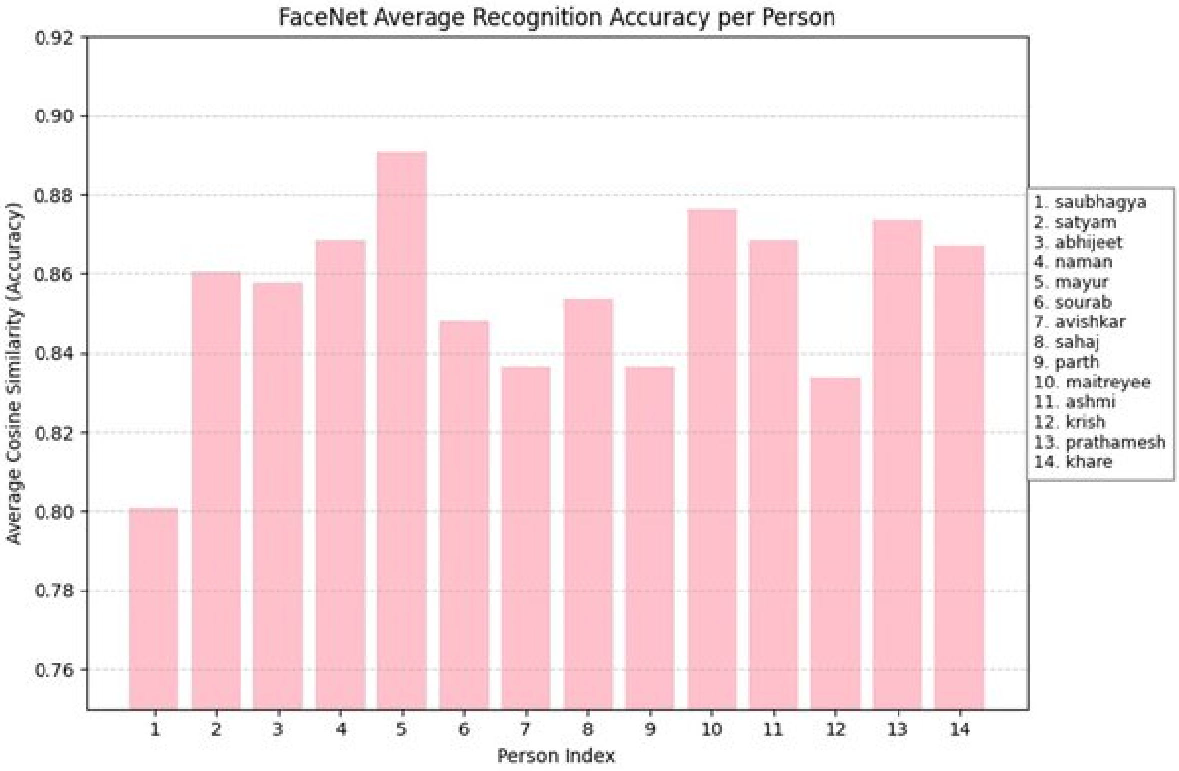


Figure 3.2: Accuracy per Person (Class) - Facenet Face Recognizer

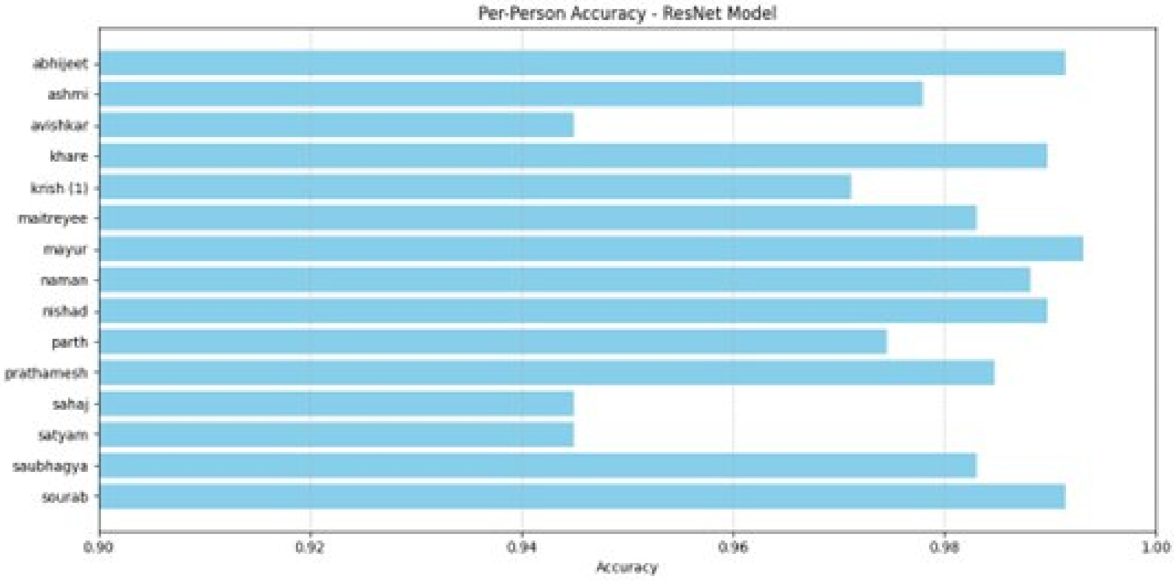


Figure 3.3: Accuracy per Person (Class) - Resnet Face Recognizer

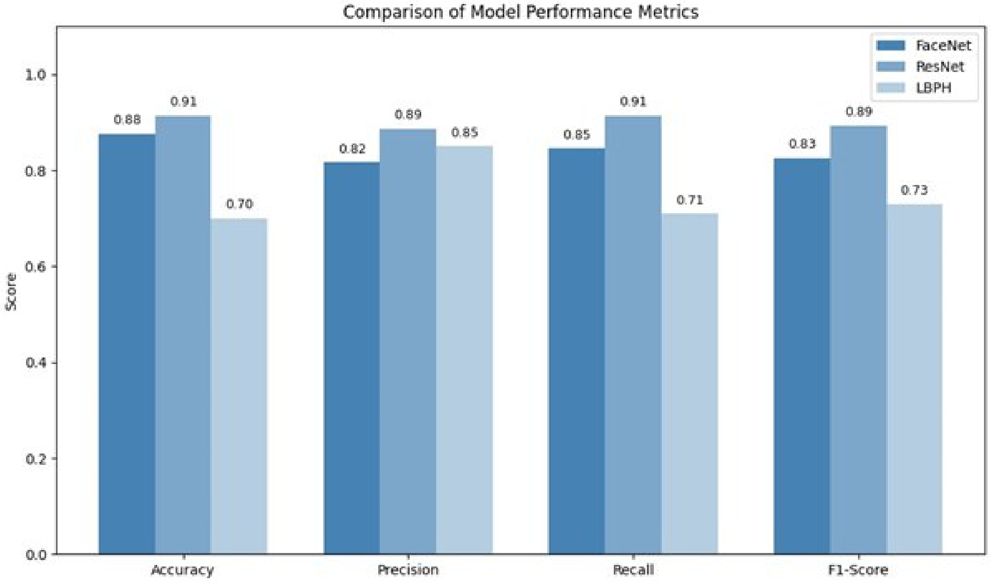


Figure 3.4: Final Model Comparison (Sourab Karad's results)

3.5 Project Modules – Individual Contribution

1. **Hardware & Software requirements:** GPU (RTX 2060), dlib, OpenCV, torch, scikit-learn, pandas.
2. **Module Interfaces:** `train_model.py`, `evaluate.py`; JSON output (`accuracy, precision, recall`).
3. **Module Dependencies:** torch→torchvision; face_recognition→dlib; numpy→pandas.
4. **Module Design:** Abstract base classes; modular trainer & evaluator.
5. **Module Implementation:** 800 LOC benchmarking harness; comparative plots in report.
6. **Testing Strategies:** 5-fold cross-validation; confusion matrices.
7. **Deployment:** Packaged ResNet model as pickle; provided Dockerfile snippet.

Chapter 4

Individual Contribution

4.1 Problem Statement

Design and build the cross-platform mobile app for attendance marking via facial capture.

4.2 Student Details

Saubhagya Singh
PRN: 1032211144
Roll Number: 38
Panel: A

4.3 Module Title

Flutter Front-End Application

4.4 Project Module Scope

Implement the Flutter-based UI for login, camera capture, attendance display, and offline support.

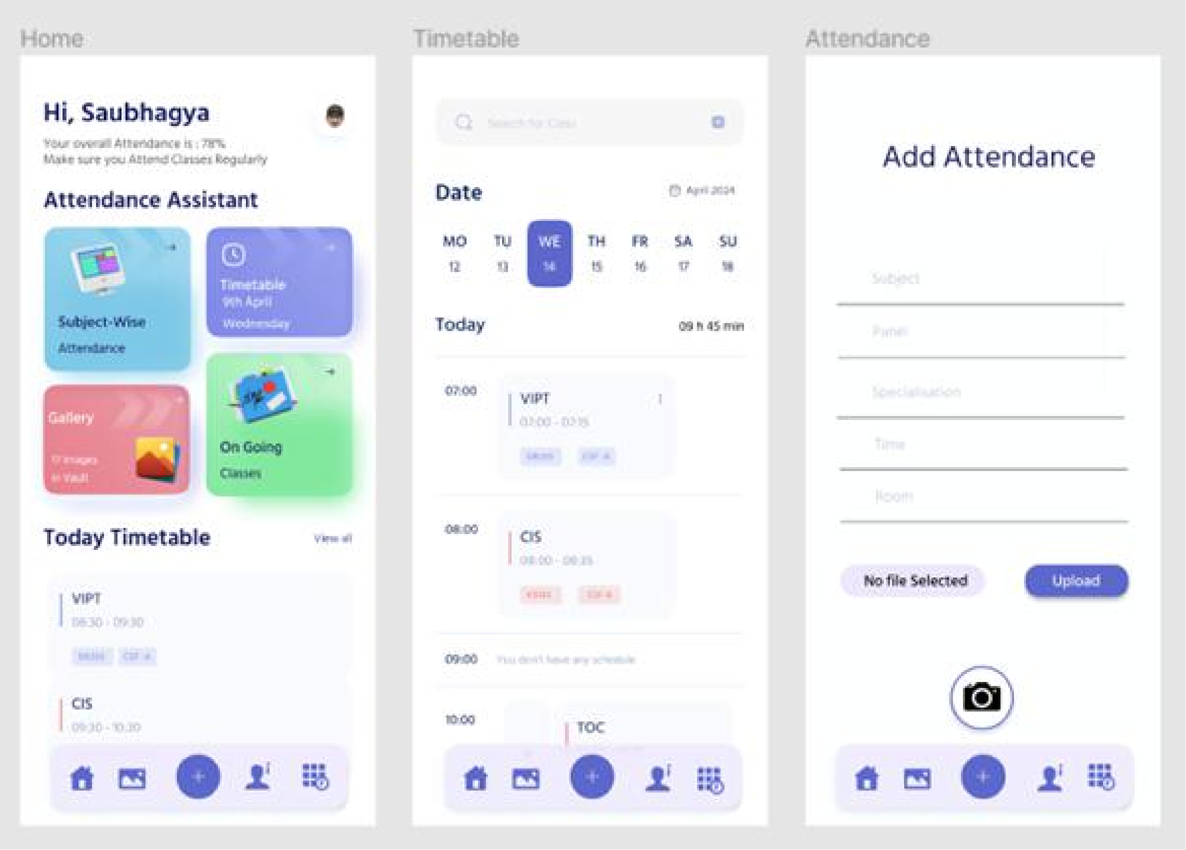


Figure 4.1: Frontend (Saubhagya Singh's contribution).

4.5 Project Modules – Individual Contribution

1. **UI Design:** Assisted in Figma wireframes and refined user flows.
2. **Flutter Development:** Built screens for login, camera preview, and attendance history.
3. **Camera Integration:** Integrated device camera plugin and handled image capture.
4. **Offline Support:** Added basic local caching to queue captures when offline.
5. **Testing:** Performed manual UI tests on both Android and iOS emulators.