Chapter 1

Individual Contribution

Chapter 2

Individual Contribution

Chapter 3

Individual Contribution

3.1 Problem Statement

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

3.2 Student Details

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

3.3 Module Title

Backend & Face-Recognition Engine

3.4 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.

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 4

Individual Contribution

4.1 Problem Statement

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

4.2 Student Details

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

4.3 Module Title

Full-Stack Support & Research

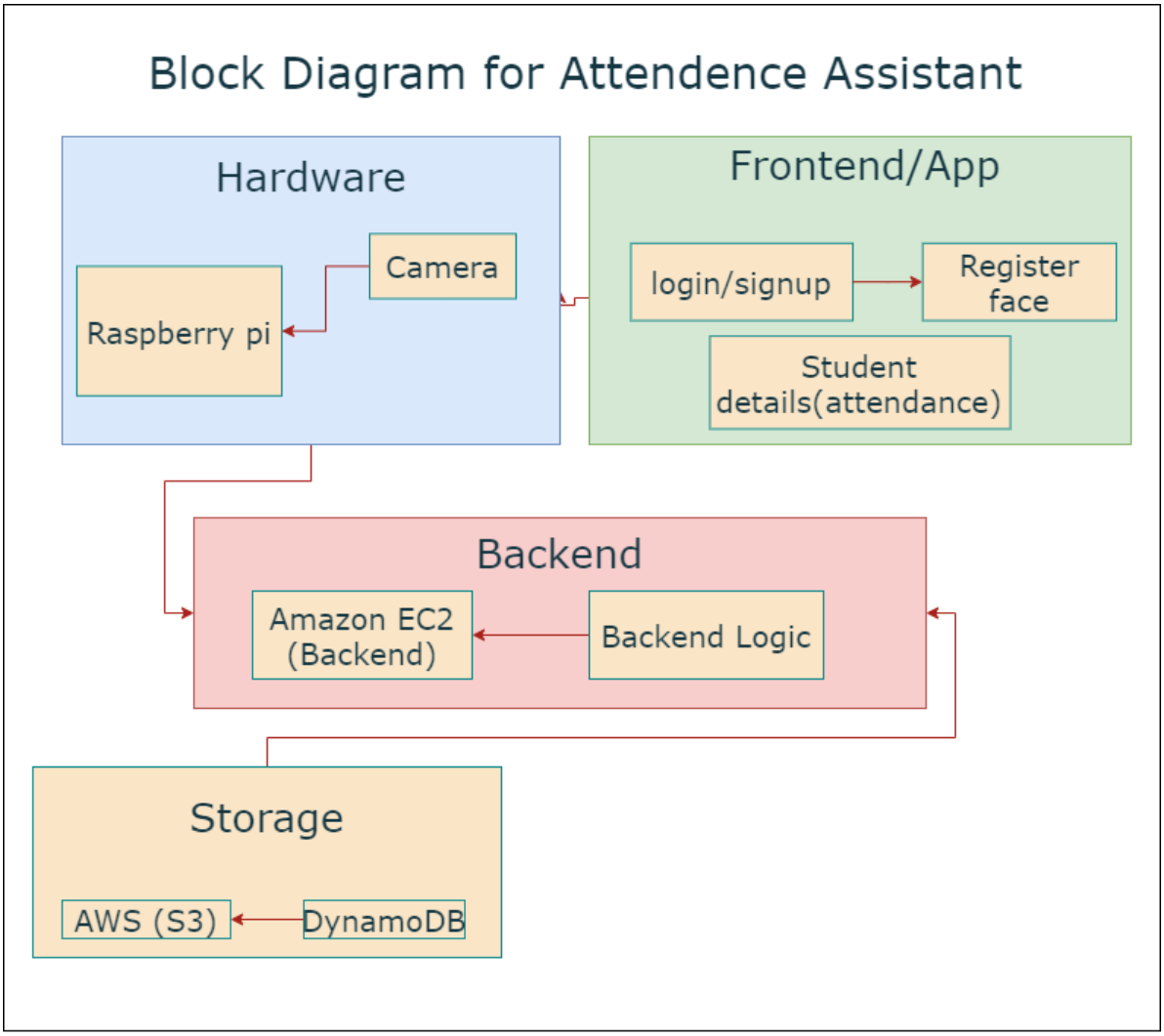

Figure 4.1: Block Diagram highlighting the modules supported by Parth Zarekar.

4.4 Project Module Scope

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

4.5 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 5

Individual Contribution

5.1 Problem Statement

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

5.2 Student Details

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

5.3 Module Title

Algorithm Research & Model Integration

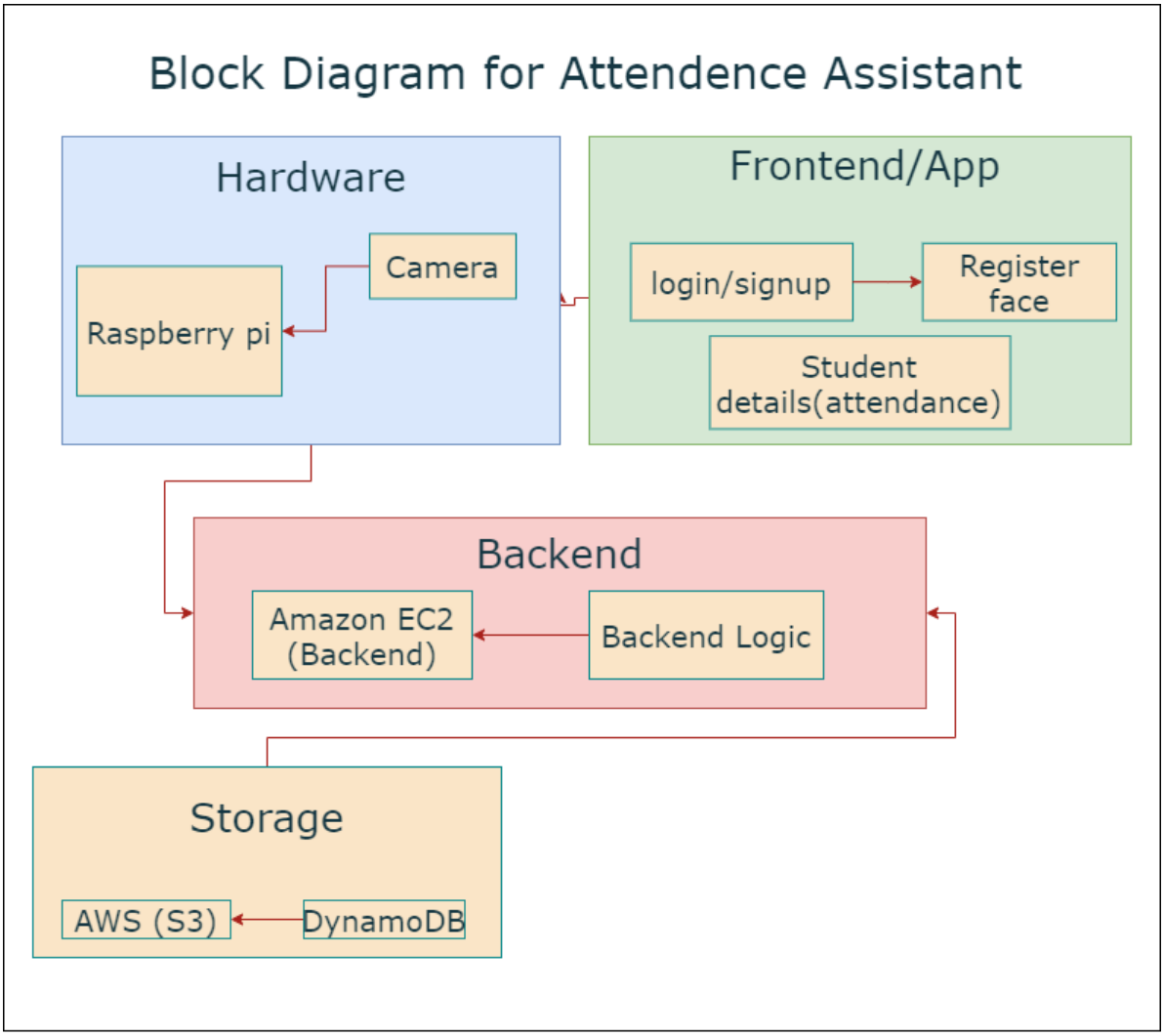

Figure 5.1: Block Diagram highlighting the algorithm research module (Sourab Karad's contribution).

5.4 Project Module Scope

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

5.5 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 6

Individual Contribution

6.1 Problem Statement

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

6.2 Student Details

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

6.3 Module Title

Flutter Front-End Application

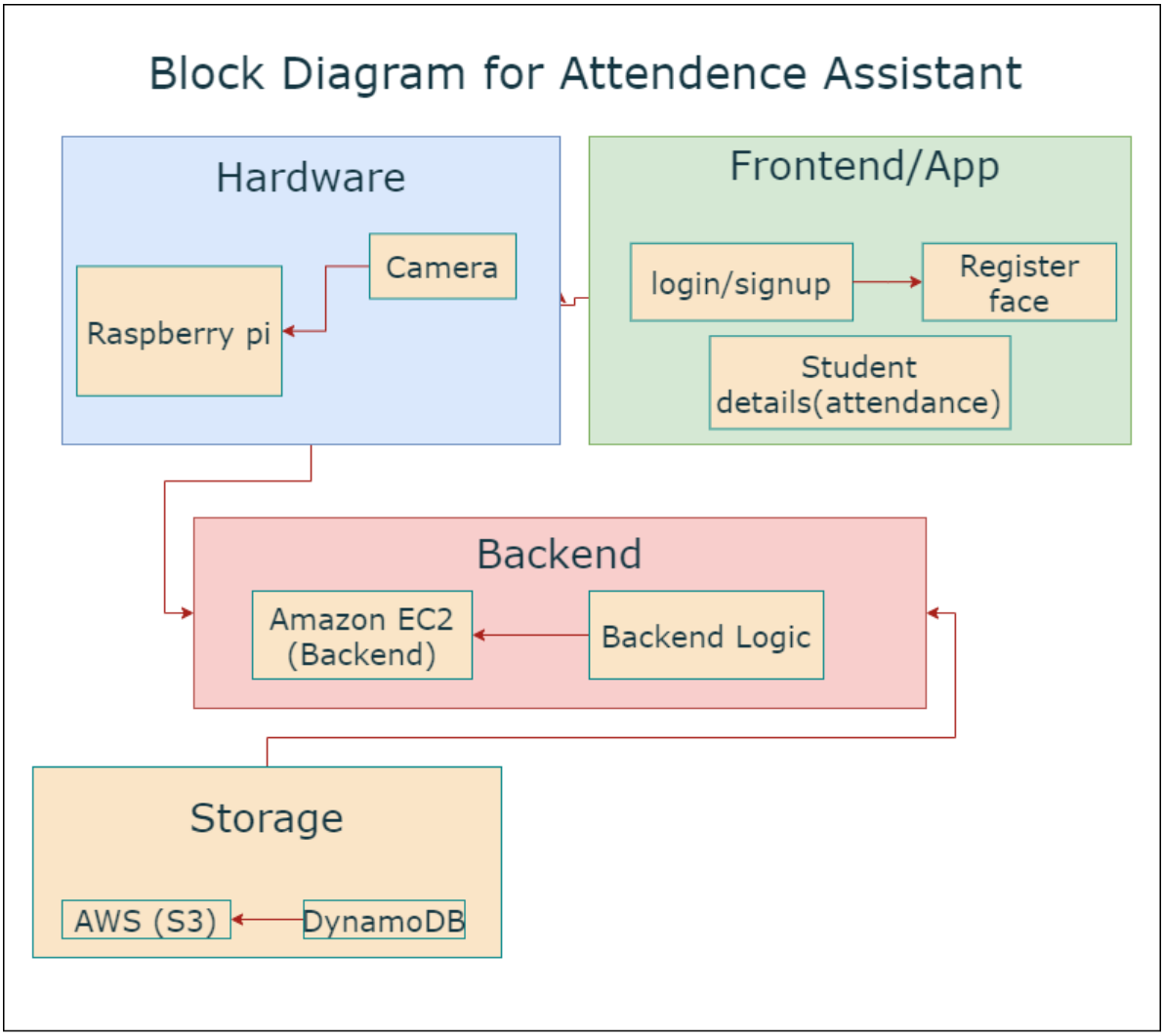

Figure 6.1: Block Diagram highlighting the frontend module (Saubhagya Singh's contribution).

6.4 Project Module Scope

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

6.5 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.