Final API Endpoints (with DB Schema)

Base URL: `https://<your-backend-domain>/api/v1/backend`
Auth: `Authorization: Bearer <JWT>` (admin/operator) for protected actions.

🧱 Database Schemas (SQLite)

1. `users`

Column	Type	Description	Requirement
`id`	UUID PRIMARY KEY	Unique user ID	Mandatory
`username`	TEXT UNIQUE	Login username	Mandatory
`email`	TEXT UNIQUE	Optional email	Optional
`password_hash`	TEXT	Hashed password	Mandatory
`full_name`	TEXT	User's full name	Mandatory
`role`	TEXT	`admin` or `operator`	Mandatory
`is_active`	BOOLEAN DEFAULT TRUE	Active status	Optional
`created_at`	TIMESTAMP DEFAULT now()	Creation time	Auto
`updated_at`	TIMESTAMP DEFAULT now()	Update time	Auto

---

2. `students_<year>` (Dynamic Partition)

Column	Type	Description	Requirement
`student_id`	TEXT PRIMARY KEY	e.g. 202500568	Mandatory
`first_name`	TEXT	Student first name	Mandatory
`last_name`	TEXT	Student last name	Optional
`email`	TEXT	Student email	Optional
`phone`	TEXT	Contact number	Optional
`created_at`	TIMESTAMP DEFAULT now()	Record creation time	Auto

---

3. `student_photos`

Column	Type	Description	Requirement
`id`	UUID PRIMARY KEY	Unique record ID	Mandatory
`student_id`	TEXT REFERENCES students_(student_id)	Student link	Mandatory
`front_path`	TEXT	Path to front face image	Mandatory
`left_path`	TEXT	Path to left image	Mandatory

Column	Type	Description	Requirement
right_path	TEXT	Path to right image	Mandatory
angled_left_path	TEXT	Path to angled left image	Mandatory
angled_right_path	TEXT	Path to angled right image	Mandatory
uploaded_by	UUID REFERENCES users(id)	Uploaded by user	Mandatory
created_at	TIMESTAMP DEFAULT now()	Upload timestamp	Auto

---

4. `enrollments`

Column	Type	Description	Requirement
id	UUID PRIMARY KEY	Enrollment record ID	Mandatory
student_id	TEXT REFERENCES students_(student_id)	Linked student	Mandatory
modality	TEXT	`face`, `body`, or `periocular`	Mandatory
embedding_vector	BYTEA	Serialized embedding vector	Mandatory
embedding_dim	INT	Embedding dimension	Mandatory
model_version	TEXT	Model used	Mandatory
created_at	TIMESTAMP DEFAULT now()	Record creation time	Auto

---

5. `attendance_events`

Column	Type	Description	Requirement
id	UUID PRIMARY KEY	Event ID	Mandatory
student_id	TEXT REFERENCES students_(student_id)	Student reference	Mandatory
camera_frame_time	TIMESTAMP	Frame timestamp	Mandatory
match_confidence	REAL	Confidence level	Mandatory
match_modality	TEXT	`face`, `fused`, etc.	Mandatory
matcher_model_version	TEXT	Version of matching model	Mandatory
created_at	TIMESTAMP DEFAULT now()	Created at	Auto

🔐 Authentication APIs

`POST /auth/login`

Authenticate an existing user.

Body:

```
{
  "username": "admin1",
  "password": "securepassword123"
}

```

Response `200`:

```
{
  "access_token": "<jwt_access>",
  "refresh_token": "<jwt_refresh>",
  "token_type": "bearer",
  "expires_in": 3600,
  "user": {
    "id": "uuid-1234",
    "username": "admin1",
    "role": "admin"
  }
}

```

Errors:

- `401` Invalid credentials
- `403` Inactive account

`POST /auth/refresh`

Refresh expired access token.

Body:

```
{
  "refresh_token": "<jwt_refresh>"
}

```

Response `200`:

```
{
  "access_token": "<new_jwt_access>",
  "token_type": "bearer",
  "expires_in": 3600
}
```

Errors:

- `401` Invalid/expired refresh token

`POST /auth/logout`

Invalidate a refresh token.

Body:

```
{
  "refresh_token": "<jwt_refresh>"
}
```

Response `200`:

```
{ "status": "logged_out" }
```

Errors:

- `401` Invalid token
- `400` Missing token

👩‍🎓 Student APIs

`POST /students`

Create new student record.

Headers: `Authorization: Bearer <JWT>`

Body:

```
{
  "student_id": "202500568",
  "first_name": "Steve",
  "last_name": "Owen",
  "email": "ishwan@example.com",
  "phone": "+91-XXXXXX",
  "metadata": { "department": "CSE" }
}

```

Response `201`:

```
{
  "student_id": "202500568",
  "status": "created",
  "created_at": "2025-11-05T10:30:00Z"
}

```

Errors:

- `400` Invalid payload
- `409` Student already exists

`POST /students/{student_id}/enroll`

Enroll a student by uploading images.

Headers:

- `Authorization: Bearer <JWT>`
- `Content-Type: multipart/form-data`

Multipart Fields:

Field	Type	Description	Requirement
`front`	file	Frontal face image	Mandatory
`left`	file	Left profile image	Mandatory
`right`	file	Right profile image	Mandatory
`angled_left`	file	Angled left image	Mandatory
`angled_right`	file	Angled right image	Mandatory

Request Body:

```
{
  "images": {
    "front": "<base64-encoded-image-or-filepath>",
    "left": "<base64-encoded-image-or-filepath>",
    "right": "<base64-encoded-image-or-filepath>",
    "angled_left": "<base64-encoded-image-or-filepath>",
    "angled_right": "<base64-encoded-image-or-filepath>"
  },
  "metadata": {
    "uploaded_by": "uuid-of-user",
    "remarks": "Initial enrollment batch",
    "timestamp": "2025-11-05T10:30:00Z"
  }
}

```

Example Response `201`:

```
{
  "student_id": "202500568",
```

```
  "photo_record_id": "uuid-photo-1234",
  "enrollments": [
    { "id": "uuid-enroll-1", "modality": "face", "model_version": "mobileface_v1", "embedding_dim": 128 },
    { "id": "uuid-enroll-2", "modality": "periocular", "model_version": "peri_v1", "embedding_dim": 64 },
    { "id": "uuid-enroll-3", "modality": "body", "model_version": "osnet_v0.25", "embedding_dim": 256 }
  ],
  "status": "enrolled",
  "faiss_update": "queued"
}
```

Errors:

- `400` Missing or invalid images
- `404` Student not found
- `500` Embedding generation failed

🔁 Real-Time Recognition Events

WebSocket Endpoint

`wss://<backend>/api/v1/backend/ws/events`

Example Recognition Event:

```
{
  "type": "recognition_event",
  "stream_url": "rtsp://192.168.1.100:554/stream1",
  "frame_time": "2025-11-05T12:20:00Z",
  "detections": [
    {
      "bbox": [120,220,300,400],
      "matched": true,
      "student_id": "202500568",
      "student_name": "Ishwan Roy",
      "match_confidence": 0.92,
      "match_modality": "fused",
      "models_used": { "face":"mobileface_v1", "body":"osnet_v0.25" },
      "thumbnail_url": "/cache/frames/frame_abc.jpg"
    }
  ]
}
```

`GET /students/{student_id}`

Fetch student and embedding information.

Response:

```
{
  "student_id": "202500568",
  "first_name": "Ishwan",
  "last_name": "Roy",
  "email": "ishwan@example.com",
  "photos": {
    "front": "/photos/2025/202500568/front.jpg",
    "left": "/photos/2025/202500568/left.jpg",
    "right": "/photos/2025/202500568/right.jpg"
  },
  "enrollments": [
    { "modality": "face", "model_version": "mobileface_v1", "embedding_dim": 128 },
    { "modality": "body", "model_version": "osnet_v0.25", "embedding_dim": 256 }
  ],
  "status": "enrolled"
}

```

📜 Full Endpoint List

#	Endpoint	Method	Description
1	`/auth/login`	POST	User login
2	`/auth/refresh`	POST	Refresh JWT token
3	`/auth/logout`	POST	Logout user
4	`/users/register`	POST	Register new admin/operator
5	`/students`	POST	Create new student record
6	`/students/{student_id}/enroll`	POST	Upload 5 photos and enroll
7	`/students/{student_id}`	GET	Get student info
8	`/ws/events`	WebSocket	Real-time recognition updates
9	`/models/list`	GET	List all active models

✅ **Final Notes:**

- RTSP streams are directly fetched by frontend; backend handles only recognition and event delivery.
- WebSocket ensures real-time attendance detection updates.
- All sensitive operations secured via JWT and role-based access control.