Revlient Student App API Documentation

Uniform Response Format

All API responses follow a consistent format:

```
{
  "message": "Success message or Error reason",
  "data": { /* serialized data */ }
}
```

- The **message** field contains a textual success message or the reason for an error.
- The **data** field contains the serialized output from the API (i.e., `serializer.data`).

Additional Notes:

- All **GET** requests that return a list (excluding single detail views) will use a paginated response. This paginated format includes keys such as:
  - `count` – Total number of items.
  - `next` – URL for the next page (if available).
  - `previous` – URL for the previous page (if available).
  - `results` – List of items for the current page.

This uniform structure ensures consistency across the API and simplifies integration with the frontend.

1. Validate Access Token API

Endpoint:

`POST /api/auth/validate-token/`

Description:

Validates an access token and returns the user's role ID.

Authentication:

No authentication required.

Request Body:

- `access` (string, required): The JWT access token.

Role ID Mapping(Fixed):

```
{
    "Superadmin": 872,
    "Admin": 239,
    "Counsellor": 890,
    "Student": 456,
    "Agent": 423
}
```

2. Login API

Endpoint:

`POST /api/auth/login/`

Description:

Authenticates a user using email and password; returns access and refresh tokens.

Authentication:

No authentication required.

Request Body:

- `email` (string, required)
- `password` (string, required)

Notes:

- `access` token validity is for 60 minutes and `refresh` token validity is for 12 hours.

3. Register Student API

Endpoint:

`POST /api/auth/register-student/`

Description:

Public endpoint to create a new **Student** user. This creates:

- a `Students` record (student profile), and
- a linked `CustomUser` record with role **Student**.

Authentication:

Not required (public signup).

Request Headers:

- `Content-Type: application/json` (or `multipart/form-data` if uploading `profile_pic`)

Request Body (JSON or form-data):

- `first_name` (string, required)
- `last_name` (string, required)
- `email` (string, required, unique)

- `phone_number` (string, required)
- `password` (string, required, min 6 chars)
- `referral_id` (string, **conditionally required***)
- `company` (string, **conditionally required***)
- `profile_pic` (file, optional — only with `multipart/form-data`)

> * **Routing rule (very important for frontend):**
>
> - At least **one** of `referral_id` or `company` **must** be provided.
> - If `referral_id` is **present**: the student is assigned to the staff (Superadmin/Admin/Counsellor/Agent) owning that referral.
> - If `referral_id` is **empty or not provided**: the **frontend MUST automatically add** `"company": "gloriaglobalventures"` to the request body **before** sending.
> - If `company` is provided (e.g., `"gloriaglobalventures"`), the student is assigned to the **Superadmin** of that company.

Sample Request (JSON):

```
{
  "first_name": "Ram",
  "last_name": "Singh",
  "email": "ramsingh@example.com",
  "phone_number": "6574857682",
  "password": "secret123",
  "referral_id": "3GNQSRW6"
}
```

OR (when no referral_id is present, frontend must send company):

```
{
  "first_name": "Ram",
  "last_name": "Singh",
  "email": "ramsingh@example.com",
  "phone_number": "6574857682",
  "password": "secret123",
  "company": "gloriaglobalventures"
}
```

Sample Response:

```
{
  "message": "Student user created successfully.",
  "data": {
    "user": {
      "id": "3807c675",
      "full_name": "Ram Singh",
      "email": "ramsingh@example.com",
      "phone_number": "6574857682",
      "role_id": 456,
      "role": "Student",
      "student_id": "56130e9e",
      "created_by": "Super Admin",
      "password_changed": false,
      "profile_pic": null,
      "company_name": "gloriaglobalventures",
      "is_blocked": false
    },
    "student": {
      "id": "56130e9e",
      "name": "Ram Singh",
      "email": "ramsingh@example.com",
      "phone_number": "6574857682",
      "location": null,
      "address": null,
      "nationality": null,
      "aadhar_card_no": null,
      "religion": null,
      "caste": null,
      "community": null,
      "enquiry": null,
      "lead_source": "Registered via Student Portal",
      "referral_id": "3GNQSRW6",
      "staff_assigned": "Jithu Admin",
      "admission_status": "not_admitted",
      "fee_status": "unpaid",
      "approval_status": "not_approved",
      "course_status": "not_started",
      "is_attended": false,
      "admitted_by": null,
      "date_of_admission": null,
      "course": null,
      "college": null,
      "father_name": null,
      "father_contact_no": null,
      "father_occupation": null,
      "mother_name": null,
      "mother_contact_no": null,
      "mother_occupation": null,
      "annual_family_income": null,
      "school_name_10th": null,
      "school_name_12th": null,
      "gender": null,
      "blood_group": null,
      "date_of_birth": null,
      "age": null,
      "passport_photo": null,
      "SSLC": null,
      "plus_two": null,
      "aadhar": null,
      "other_documents": null,
      "uniform_fee": null,
      "extra_fee": null,
      "first_year": null,
      "second_year": null,
      "third_year": null,
      "fourth_year": null,
      "fifth_year": null,
      "total_fees": "0.00"
    }
  }
}
```

Possible Error Responses:

- `400 Bad Request`

```
{ "message": "email: User with this email already exists.", "data": null }
```

```
{
    "message": "routing: Provide either referral_id, staff_assigned, or
company to associate with an organization.",
    "data": null
}
```

```
{
    "message": "referral_id: Invalid referral_id. No matching staff found.",
    "data": null
}
```

```
{
    "message": "company: No Superadmin found for company
'gloriaglobalventures'.",
    "data": null
}
```

Notes for Frontend:

- Always ensure **either** `referral_id` **or** `company` is present (auto-append `"company": "gloriaglobalventures"` when `referral_id` is missing/empty).
- For file uploads (`profile_pic`), use `multipart/form-data` .
- On success, both the `user` and `student` objects are returned for immediate UI population (profile + portal).
- Response format is **uniform**:

```
{ "message": "<status message>", "data": { ... } }
```

4. Get Current User Details API

Endpoint:

`GET /api/auth/user-details/`

Description:

Retrieves details of the currently authenticated user.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

Request Body:

None.

Sample Response:


```
{
    "id": "eea3da58",
    "full_name": "Ram Manohar",
    "email": "rammanohar@example.com",
    "phone_number": "6574857680",
    "role_id": 456,
    "role": "Student",
    "student_id": "ea21d376",
    "created_by": "Super Admin",
    "password_changed": false,
    "profile_pic": null,
    "company_name": "gloriaglobalventures",
    "is_blocked": false
}
```


5. Refresh Access Token API

Endpoint:

`POST /api/auth/refresh-token/`

Description:

Generates a new access token (and possibly a new refresh token) using a valid refresh token.

Authentication:

No authentication required.

Request Body:

- `refresh` (string, required): The JWT refresh token.

Notes:

- `access` token validity is for 60 minutes.

6. Logout API

Endpoint:

`POST /api/auth/logout/`

Description:

Logs out the user by blacklisting the refresh token.

Authentication:

Requires a valid access token in the `Authorization` header.

Request Headers:

- `Authorization` (string, required)

Request Body:

- `refresh` (string, required): The JWT refresh token.

Notes:

- `Must redirect to the login page and also needs to remove the current access token from the frontend cache.`

7. Reset Password API

Endpoint:

```
POST /api/auth/reset-password/
```

Description:

Requests a password reset by sending an OTP to the registered email.

Authentication:

Requires a valid access token in the `Authorization` header.

Request Headers:

- `Authorization` (string, required)

Request Body:

- `email` (string, required): The registered email.

8. Forget Password API

Endpoint:

```
POST /api/auth/forget-password/
```

Description:

Initiates a password reset for any user by sending an OTP to their email.

Authentication:

No authentication required.

Request Body:

- `email` (string, required): The registered email.

9. Change Password API

Endpoint:

`POST /api/auth/change-password/`

Description:

Resets the user's password using a valid OTP.

Authentication:

No authentication required.

Request Body:

- `otp` (string, required): The OTP received via email.
- `new_password` (string, required)
- `confirm_password` (string, required)

Notes:

- Needs to check if the new_password and confirm_password are same in the frontend itself.

10. Delete Student Account API

Endpoint:

`DELETE /api/auth/delete-user/<user_id>/`

Description:

Allows a **Student** user to permanently delete their own account.

- Both the `CustomUser` record and the linked `Students` record will be deleted.
- A student can **only delete their own account**.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

Request Body:

None.

Behavior (Student role):

- If the logged-in user is a **Student**, they can only delete their **own account**.
- If a student attempts to delete another user's account, the request will be denied.

Sample Request:

```
DELETE /api/auth/delete-user/3807c675/
Authorization: Bearer <access_token>
```

Sample Success Response:

```
{
    "message": "Your account & Student data has been deleted successfully."
}
```

Sample Error Responses:

- Trying to delete another student’s account:

```
{
    "message": "Students can only delete their own account."
}
```

- User not found:

```
{
    "message": "User not found."
}
```

11. Edit User Details API

Endpoint:

`GET /api/admin/edit-user/<user_id>/`

`PATCH /api/admin/edit-user/<user_id>/`

Description:

For **Student** users, this endpoint lets them **view and update their own profile** only.

- Editable fields (Student role): `first_name` , `last_name` , `email` , `phone_number` , `profile_pic`
- `referral_id` is **read-only** (auto-generated for staff only; not editable).
- Empty string values for editable fields are **ignored** (existing DB value is kept).
- `profile_pic` can be **removed** by sending an empty string or `"null"` .

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Access Rules (Student role):

- A Student may **only GET/PATCH their own user** (`<user_id>` must equal the authenticated Student’s `id`).
- Attempts to access or edit another user will be rejected.

Request Headers:

- `Authorization` (string, required)
- `Content-Type: application/json` (for PATCH without file)
- `Content-Type: multipart/form-data` (for PATCH with `profile_pic`)

➤ GET: Retrieve User Profile

Request Body:

None.

Sample Request:

```
GET /api/admin/edit-user/3807c675/
Authorization: Bearer <access_token>
```

Sample Response (Student view fields):

```
{
  "id": "3807c675",
  "full_name": "Ram Singh",
  "email": "ramsingh@example.com",
  "phone_number": "6574857682",
  "role_id": 456,
  "role": "Student",
  "student_id": "56130e9e",
  "created_by": "Super Admin",
  "password_changed": false,
  "profile_pic": null,
  "company_name": "gloriaglobalventures",
  "is_blocked": false
}
```

Possible Errors:

- `403 Forbidden` — Student tried to access another user's details.

➤ PATCH: Update User Profile

Editable fields:

`first_name` , `last_name` , `email` , `phone_number` , `profile_pic`

Rules:

- **Phone number** must contain **at least 10 digits**.
- **Empty string** for any editable field ⇒ **keeps existing value** (ignored).
- To **remove** `profile_pic` , send `profile_pic` as empty string `""` or `"null"` in form-data/JSON.
- Any attempt to send `referral_id` is ignored (read-only).

A) Sample Request (JSON — without file)

```
PATCH /api/admin/edit-user/3807c675/
Authorization: Bearer <access_token>
Content-Type: application/json
```

```
{
  "first_name": "Ram",
  "last_name": "Singh",
  "email": "ramsingh@example.com",
  "phone_number": "6574857682"
}
```

B) Sample Request (multipart/form-data — with new `profile_pic`)

```
PATCH /api/admin/edit-user/3807c675/
Authorization: Bearer <access_token>
Content-Type: multipart/form-data


first_name: Ram
last_name: Singh
email: ramsingh@example.com
phone_number: 6574857682
profile_pic: <image file>
```

C) Sample Request (remove `profile_pic`)

```
PATCH /api/admin/edit-user/3807c675/
Authorization: Bearer <access_token>
Content-Type: multipart/form-data


profile_pic: ""
```

Sample Success Response:

```
{
  "message": "User details updated successfully",
  "data": {
    "id": "3807c675",
    "full_name": "Ram Singh",
    "email": "ramsingh@example.com",
    "phone_number": "6574857682",
    "role_id": 456,
    "role": "Student",
    "student_id": "56130e9e",
    "created_by": "Super Admin",
    "password_changed": false,
    "profile_pic": "https://your-domain/media/profile_pics/ram.jpg",
    "company_name": "gloriaglobalventures",
    "is_blocked": false
  }
}
```

Possible Error Responses:

- `403 Forbidden`

  ```
  { "message": "You do not have permission to update this user." }
  ```

- `400 Bad Request` (phone number validation)

  ```
  { "message": "Phone number must contain at least 10 digits." }
  ```

- `500 Internal Server Error`

  ```
  { "message": "An unexpected error occurred." }
  ```

12. Student Self Profile API (GET & PATCH)

Endpoint:

`GET /api/admin/student/self/me/`

`PATCH /api/admin/student/self/me/`

Description:

Retrieve and update the **logged-in Student's** profile.

- **GET** returns full student details.
- **PATCH** updates allowed fields.
- Only available to authenticated users with role = **Student** and a linked `Students` record.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)
- `Content-Type: application/json` (for PATCH without files)
- `Content-Type: multipart/form-data` (for PATCH with files like `passport_photo` , certificates)

➤ GET: Fetch Student Profile

Request Body:

None.

Sample Request:

```
GET /api/admin/student/self/me/
Authorization: Bearer <access_token>
```

Choice Fields:

```
student_status = [ "interested", "not_interested", "pending", "accepted",
"follow_up", "not_answered" ]

admission_status = [ "not_admitted", "admitted" ]

fee_status = [ "unpaid", "paid" ]

approval_status = [ "not_approved", "approved" ]

course_status = [ "not_started", "ongoing", "cancelled", "completed" ]

gender = [ "male", "female", "other" ]

blood_group = [ "A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-" ]
```

Sample Response:

```
{
  "id": "56130e9e",
  "name": "Ram Singh",
  "email": "ramsingh@example.com",
  "phone_number": "6574857682",
  "location": null,
  "address": null,
  "nationality": null,
  "aadhar_card_no": null,
  "religion": null,
  "caste": null,
  "community": null,
  "enquiry": null,
  "lead_source": "Registered via Student Portal",
  "referral_id": "3GNQSRW6",
  "staff_assigned": "Jithu Admin",
  "student_status": "pending",
  "admission_status": "not_admitted",
  "fee_status": "unpaid",
  "approval_status": "not_approved",
  "course_status": "not_started",
  "is_attended": false,
  "admitted_by": null,
  "date_of_admission": null,
  "course": null,
  "college": null,
  "father_name": null,
  "father_contact_no": null,
  "mother_name": null,
  "mother_contact_no": null,
  "gender": null,
  "blood_group": null,
  "date_of_birth": null,
  "age": null,
  "passport_photo": null,
  "SSLC": null,
  "plus_two": null,
  "aadhar": null,
  "other_documents": null,
  "uniform_fee": null,
  "extra_fee": null,
  "first_year": null,
  "second_year": null,
  "third_year": null,
  "fourth_year": null,
  "fifth_year": null,
  "total_fees": "0.00"
}
```


Possible Error Responses:

- 403 Forbidden

  ```
  { "message": "Only students can access this endpoint.", "data": null }
  ```

- 404 Not Found

  ```
  { "message": "Student profile not linked to your account.", "data": null }
  ```

  or

  ```
  { "message": "Student profile not found.", "data": null }
  ```

➤ PATCH: Update Student Profile

Description:

Update allowed fields of the student profile.

- If `name`, `email`, or `phone_number` are sent as **empty string** `""` or **null**, the **existing DB value is preserved** (field is ignored).
- If **all provided keys** are empty string / null, returns **"No Data to Update."**
- File fields replace existing files when a new file is uploaded.

Allowed Updatable Fields (examples):

- Basic: `name`, `email`, `phone_number`, `location`, `address`, `enquiry` (JSON)
- Academic/Personal: `course`, `college`, `gender`, `blood_group`, `date_of_birth`, `nationality`, `aadhar_card_no`, `religion`, `caste`, `community`, `father_name`, `father_contact_no`, `mother_name`, `mother_contact_no`, `father_occupation`, `mother_occupation`, `annual_family_income`, `school_name_10th`, `school_name_12th`
- Files: `passport_photo`, `SSLC`, `plus_two`, `aadhar`, `other_documents`

Rules & Validations:

- Empty/ `null` for **required base fields** (`name`, `email`, `phone_number`) ⇒ **ignored** (keeps DB value).
- If **every provided value** is empty/ `null` ⇒ `400 Bad Request` with `"No Data to Update."`
- `enquiry` accepts JSON string or JSON object.
- Uploading a new file replaces the old file (old file is cleaned up per model logic).
- To keep an existing file, simply **do not send that field**.

Sample Request (JSON):

```
PATCH /api/admin/student/self/me/
Authorization: Bearer <access_token>
Content-Type: application/json



{
  "location": "Kochi",
  "course": "BBA",
  "enquiry": { "interest": "Management", "year": 2025 },
  "phone_number": ""
}
```

> Here `phone_number` is `""` so it is **ignored**, keeping the current phone number intact.

Sample Request (multipart/form-data with file):

```
PATCH /api/admin/student/self/me/
Authorization: Bearer <access_token>
Content-Type: multipart/form-data



course: B.Com
passport_photo: <image file>
```

Sample Success Response:

```
{
  "message": "Student details updated successfully.",
  "data": {
    "id": "56130e9e",
    "name": "Ram Singh",
    "email": "ramsingh@example.com",
    "phone_number": "6574857682",
    "location": "Kochi",
    "address": null,
    "nationality": null,
    "aadhar_card_no": null,
    "religion": null,
    "caste": null,
    "community": null,
    "enquiry": { "interest": "Management", "year": 2025 },
    "lead_source": "Registered via Student Portal",
    "referral_id": "3GNQSRW6",
    "staff_assigned": "Jithu Admin",
    "student_status": "pending",
    "admission_status": "not_admitted",
    "fee_status": "unpaid",
    "approval_status": "not_approved",
    "course_status": "not_started",
    "is_attended": false,
    "admitted_by": null,
    "date_of_admission": null,
    "course": "BBA",
    "college": null,
    "father_name": null,
    "father_contact_no": null,
    "mother_name": null,
    "mother_contact_no": null,
    "gender": null,
    "blood_group": null,
    "date_of_birth": null,
    "age": null,
    "passport_photo": "https://your-domain/media/photos/ram.jpg",
    "SSLC": null,
    "plus_two": null,
    "aadhar": null,
    "other_documents": null,
    "uniform_fee": null,
    "extra_fee": null,
    "first_year": null,
    "second_year": null,
    "third_year": null,
    "fourth_year": null,
    "fifth_year": null,
    "total_fees": "0.00"
  }
}
```


Possible Error Responses:

- 403 Forbidden


  ```
  { "message": "Only students can access this endpoint.", "data": None }
  ```


- 404 Not Found


  ```
  { "message": "Student profile not linked to your account.", "data": null }
  ```


- 400 Bad Request

```
{ "message": "No Data to Update.", "data": null }
```

or validation errors:

```
{ "message": "email: Enter a valid email address.", "data": null }
```

13. List All Colleges (with Images & Courses + Bookmarking) API

Endpoint:

`GET /api/admin/colleges/list/`

Description:

Lists all colleges visible to the authenticated user (scoped by organization hierarchy), including:

- College details
- **images**: full list of college images (id + absolute URL)
- **courses**: list with `id`, `course_name`, `course_description`, `fee_structure` (absolute URL if present), and `is_bookmarked` (for the current user)

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

Query Parameters (optional):

- `search` (string): case-insensitive match on `college_name` or `college_location`
- `course` (string): exact match on `course_name`
- Pagination (standard): `page`, `page_size`

Notes on Visibility (Org Hierarchy):

- **Student** → colleges created by their **Superadmin**

Sample Response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "result": [
    {
      "id": "6479",
      "college_name": "Viswajyothi College of Engineering",
      "college_description": "New engineering college",
      "college_location": "Muvattupuzha",
      "college_address": "",
      "college_brochure": null,
      "video_url": null,
      "images": [
        {
          "id": "489",
          "image": "http://127.0.0.1:8000/media/college_images/Laundry_2.jpg"
        },
        {
          "id": "735",
          "image": "http://127.0.0.1:8000/media/college_images/landry_3.jpg"
        }
      ],
      "courses": [
        {
          "id": "6862",
          "course_name": "CSE",
          "course_description": null,
          "fee_structure": "",
          "is_bookmarked": false
        },
        {
          "id": "6993",
          "course_name": "ECE",
          "course_description": null,
          "fee_structure": "",
          "is_bookmarked": false
        }
      ]
    }
  ]
}
```


Possible Error Responses:

- `401 Unauthorized`

```
{ "message": "Authentication required to view colleges.", "data": null }
```

Implementation Notes:

- `is_bookmarked` is computed by checking the current user's `CourseBookmark` records for each course.
- `fee_structure` (if present) is returned as an **absolute URL**.
- Response is **paginated** using your custom pagination (`count` , `next` , `previous` , `result`).

14. Get Single College Details (with Images & Courses + Bookmarking)

Endpoint:

`GET /api/admin/college-details/<college_id>/`

Description:
Returns one college's full details scoped to the same Superadmin organization as the requester, including:

- Base college fields
- **images**: list of image objects with absolute `image` URLs
- **courses**: list with `id`, `course_name`, `course_description`, `fee_structure` (absolute URL if present), and `is_bookmarked` (for the current user)

Authentication:
Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

URL Path Params:

- `college_id` (string, required) – the 4-digit college ID

Notes on Visibility (Org Hierarchy):

- **Student** → colleges created by their **Superadmin**

Sample Request:

```
GET /api/admin/college-details/6479/
Authorization: Bearer <access_token>
```

Sample Response:


```
{
  "id": "6479",
  "college_name": "Viswajyothi College of Engineering",
  "college_description": "New engineering college",
  "college_location": "Muvattupuzha",
  "college_address": "",
  "college_brochure": null,
  "video_url": null,
  "images": [
    { "id": "489", "image": "http://127.0.0.1:8000/media/college_images/Laundry_2.jp
    { "id": "735", "image": "http://127.0.0.1:8000/media/college_images/landry_3.jpg
  ],
  "courses": [
    {
      "id": "6862",
      "course_name": "CSE",
      "course_description": null,
      "fee_structure": "",
      "is_bookmarked": false
    },
    {
      "id": "6993",
      "course_name": "ECE",
      "course_description": null,
      "fee_structure": "",
      "is_bookmarked": true
    }
  ]
}
```


Possible Error Responses:

- `401 Unauthorized`

```
{ "message": "Authentication required.", "data": null }
```

- `404 Not Found`

```
{ "message": "Not found.", "data": null }
```

Implementation Notes:

- `images` are produced via `CollegeImageSerializer` (absolute URLs).
- `courses` are built from `college.courses.values(...)`, with `fee_structure` transformed to an **absolute URL** when present.
- `is_bookmarked` is computed from the current user's `CourseBookmark` records.
- Access is restricted by role to the same Superadmin scope in `get_queryset()`.

15. My College Course API

Endpoint:

`GET /api/admin/my-college-course/`

Purpose:

This API is used to populate the **college** and **course** dropdown fields when assigning a student to a specific course in a college.

Authentication Required:

✅ Yes – Must be a logged-in user.

Provide a valid **JWT token** in the `Authorization` header.

Headers:

Name	Type	Required	Description
Authorization	string	✅ Yes	Bearer token (`Bearer <access_token>`)

🎯 Logic and Behavior

This API accepts **one** of the following sets of query parameters. Depending on the parameter, it returns either a list of colleges or a list of courses.

Parameter Priority	Description
✅ `college_name`	Returns a list of course names offered by the specified college.
✅ `course_name`	Returns a list of college names that offer the specified course.
✅ `college_search`	Returns a list of college names that match the search string (used for autocomplete).
✅ `course_search`	Returns a list of course names that match the search string (used for autocomplete).

> ⚠️ If `college_name` or `course_name` is provided, then `college_search` and `course_search` are **ignored**.

🔍 Query Parameters

Name	Type	Required	Description
`college_name`	string	Optional	Return courses offered by this college (case-insensitive partial match allowed).
`course_name`	string	Optional	Return colleges offering this course (case-insensitive partial match allowed).
`college_search`	string	Optional	Return list of college names matching this string (for live search/autocomplete).
`course_search`	string	Optional	Return list of course names matching this string (for live search/autocomplete).

✅ Example 1 – Get Courses from a College

Request:

```
GET /api/admin/my-college-course/?college_name=Christ College
Authorization: Bearer <your_token>
```

Response:

```
[
  "B.Com",
  "BBA",
  "BCA"
]
```

✅ Example 2 – Get Colleges Offering a Course

Request:

```
GET /api/admin/my-college-course/?course_name=BCA
Authorization: Bearer <your_token>
```

Response:

```
[
  "Christ College",
  "MES College"
]
```

✅ Example 3 – Search College Names for Autocomplete

Request:

```
GET /api/admin/my-college-course/?college_search=chr
Authorization: Bearer <your_token>
```

Response:

```
[
  "Christ College",
  "Christian College of Engineering"
]
```

✅ Example 4 – Search Course Names for Autocomplete

Request:

```
GET /api/admin/my-college-course/?course_search=b
Authorization: Bearer <your_token>
```

Response:

```
[
  "BBA",
  "BCA",
  "B.Sc"
]
```

🚫 Invalid or No Parameters

- If no query parameters are provided or none match, the API returns an empty list:

```
[]
```

16. List Bookmarked Courses (Student)

Endpoint:

`GET /api/admin/students/bookmarks/`

Description:

Returns the **current student's** bookmarked courses (paginated), including:

- Course details (`id` , `course_name` , `course_description` , `fee_structure`)
- Related **college** details (`id` , `college_name` , `college_location`)
- All **college_images** (each with `id` and absolute `image` URL)
- `created_at` of the bookmark in `dd-mm-yyyy HH:MM:SS` format

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).
Only users with role **Student** can access this endpoint.

Request Headers:

- `Authorization` (string, required)

Query Parameters (optional):

- Pagination: `page` , `page_size`

Sample Response:


```
{
  "count": 1,
  "next": null,
  "previous": null,
  "result": [
    {
      "id": "3621",
      "course": {
        "id": "6993",
        "course_name": "ECE",
        "course_description": null,
        "fee_structure": null,
        "college": {
          "id": "6479",
          "college_name": "Viswajyothi College of Engineering",
          "college_location": "Muvattupuzha"
        },
        "college_images": [
          { "id": "489", "image":
"http://127.0.0.1:8000/media/college_images/Laundry_2.jpg" },
          { "id": "735", "image":
"http://127.0.0.1:8000/media/college_images/landry_3.jpg" }
        ]
      },
      "created_at": "28-08-2025 00:42:37"
    }
  ]
}
```


Possible Error Responses:

- `401 Unauthorized`

  ```
  { "message": "Authentication credentials were not provided.", "data": null }
  ```

- `403 Forbidden` (non-student)

  ```
  { "message": "Only students can access bookmarks.", "data": null }
  ```

Notes:

- Uses standard pagination (`count` , `next` , `previous` , `result`).
- `college_images` is returned for each bookmarked course's college as a list of `{id, image}` objects with absolute URLs.
- `created_at` is formatted server-side as `dd-mm-yyyy HH:MM:SS` .

17. Toggle Course Bookmark (Student)

Endpoint:

`POST /api/admin/student/bookmarks/toggle/`

Description:

Toggles a bookmark for the **current student** on a given course.

- If the course is **not yet bookmarked**, this creates the bookmark and returns its details.
- If the course **is already bookmarked**, this removes the bookmark and returns `data: null` .

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Only users with role **Student** can access this endpoint.

Request Headers:

- `Authorization` (string, required)
- `Content-Type: application/json`

Request Body:

```
{
  "course": "6993"   // Course ID (string)
}
```

Sample Response — Created:

```
{
  "message": "Course bookmarked successfully.",
  "data": {
    "id": "3621",
    "course": {
      "id": "6993",
      "course_name": "ECE",
      "course_description": null,
      "fee_structure": null,
      "college": {
        "id": "6479",
        "college_name": "Viswajyothi College of Engineering",
        "college_location": "Muvattupuzha"
      },
      "college_images": [
        { "id": "489", "image":
"http://127.0.0.1:8000/media/college_images/Laundry_2.jpg" },
        { "id": "735", "image":
"http://127.0.0.1:8000/media/college_images/landry_3.jpg" }
      ]
    },
    "created_at": "28-08-2025 00:42:37"
  }
}
```


Sample Response — Removed:


```
{
  "message": "Bookmark removed successfully.",
  "data": null
}
```


Possible Error Responses:

- `401 Unauthorized`


  ```
  { "message": "Authentication credentials were not provided.", "data": null }
  ```


- `403 Forbidden` (non-student)


  ```
  { "message": "Only students can access bookmarks.", "data": null }
  ```


- `400 Bad Request` (validation)


  ```
  { "message": "course: This field is required.", "data": null }
  ```


- `404 Not Found` (invalid course)


  ```
  { "message": "course: Course not found.", "data": null }
  ```


18. List Student Announcements (messages only)

Endpoint:

`GET /api/admin/student/announcements/list/`

Description:

Returns **only the announcement messages** (strings) for the student's organization, ordered by newest first. Designed for a top banner so students don't miss updates.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

Request Body:

None.

Sample Request:

```
GET /api/admin/student/announcements/list/
Authorization: Bearer <access_token>
```

Sample Response:

```
[
  "New college added.",
  "The college fees increased."
]
```

Possible Error Responses:

- `401 Unauthorized`

```
{ "message": "Authentication credentials were not provided.", "data": null }
```

Notes:

- Scope is resolved to the **Superadmin** of the current user; messages are fetched from that organization only.
- Response is a **plain JSON array of strings**, no pagination.

19. List FAQs

Endpoint:

`GET /api/admin/faqs/`

Description:

Fetches all FAQs. Supports **search** and **active** filters with pagination.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Query Parameters (optional):

- `search` — substring to match in `question` or `answer` (case-insensitive).
- `active` — filter by status: `true` or `false`.
- Pagination: `page`, `page_size`.

Request Body:

None.

Sample Response:


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "result": [
    {
      "id": "cadb2acc",
      "question": "What is this?",
      "answer": "It is a Monkey....",
      "is_active": true,
      "created_at": "26-08-2025 23:48:19",
      "updated_at": "27-08-2025 00:03:54",
      "created_by_full_name": "Super Admin"
    },
    {
      "id": "5c6eeb1d",
      "question": "What is this?",
      "answer": "It is a Monkey....",
      "is_active": true,
      "created_at": "26-08-2025 23:53:38",
      "updated_at": "26-08-2025 23:53:38",
      "created_by_full_name": "Super Admin"
    }
  ]
}
```


Notes:

- Results are ordered by `updated_at` (newest first).
- `created_at` and `updated_at` are formatted as `dd-mm-yyyy HH:MM:SS`.
- Use `active=true` to show only FAQs currently visible to end users.

20. Get Single FAQ

Endpoint:

`GET /api/admin/faqs/{id}/`

(e.g., `/api/admin/faqs/cadb2acc/`)

Description:

Retrieve the full details of a single FAQ by its ID.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Path Parameters:

- `id` (string, required) — FAQ identifier (e.g., `cadb2acc`)

Request Body:

None.

Sample Request:

```
GET /api/admin/faqs/cadb2acc/
```

Sample Response:

```
{
  "id": "cadb2acc",
  "question": "What is this?",
  "answer": "It is a Monkey....",
  "is_active": true,
  "created_at": "26-08-2025 23:48:19",
  "updated_at": "27-08-2025 00:03:54",
  "created_by_full_name": "Super Admin"
}
```

Possible Error Responses:

- `404 Not Found`

  ```
  { "message": "FAQ not found.", "data": null }
  ```

21. List Notifications (role-aware)

Endpoint:

`GET /api/admin/notifications/`

Description:

Returns notifications visible to the current user based on their role.

- **Student:** Sees notifications from their Superadmin with visibility `student` and `all`.

Supports **search** (by message text) and pagination.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

Query Parameters (optional):

- `search` — case-insensitive substring match on `message`.
- Pagination: `page`, `page_size`.

Request Body:

None.

Sample Request:

```
GET /api/admin/notifications/?search=fees
Authorization: Bearer <access_token>
```

Sample Response:


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "result": [
    {
      "id": "104",
      "message": "New fee schedule published for 2025 intake.",
      "visible_to": "student",
      "created_at": "28-08-2025 09:30:12",
      "updated_at": "28-08-2025 09:31:05",
      "created_by_full_name": "Super Admin"
    },
    {
      "id": "103",
      "message": "Orientation week starts Monday.",
      "visible_to": "all",
      "created_at": "27-08-2025 17:05:40",
      "updated_at": "27-08-2025 17:05:40",
      "created_by_full_name": "Super Admin"
    },
    {
      "id": "099",
      "message": "Submit pending documents by Friday.",
      "visible_to": "student",
      "created_at": "26-08-2025 12:14:09",
      "updated_at": "26-08-2025 12:14:09",
      "created_by_full_name": "Jithu Admin"
    }
  ]
}
```


Possible Error Responses:

- `401 Unauthorized`


```
{ "message": "Authentication credentials were not provided.", "data": null }
```


Notes:

- Results are ordered by `updated_at` (newest first).
- Date-time fields use `dd-mm-yyyy HH:MM:SS`.
- The visibility filter is **role-driven** and handled server-side per the hierarchy rules shown above.

22. Get Single Notification

Endpoint:

`GET /api/admin/notification/{notification_id}/`

(e.g., `/api/admin/notification/002/`)

Description:

Retrieve full details of a single notification by its ID.

Authentication:

Requires a valid access token in the `Authorization` header (`Bearer <token>`).

Request Headers:

- `Authorization` (string, required)

Path Parameters:

- `notification_id` (string, required) — 3-digit identifier (e.g., `"002"`)

Request Body:

None.

Sample Request:

```
GET /api/admin/notification/002/
Authorization: Bearer <access_token>
```

Sample Response:

```
{
  "id": "002",
  "message": "Welcome student to our community",
  "created_at": "27-08-2025 16:59:28",
  "updated_at": "27-08-2025 16:59:28",
  "created_by": "Super Admin",
  "visible_to": "student"
}
```

Possible Error Responses:

- `404 Not Found`

  ```
  { "message": "Notification not found.", "data": null }
  ```