E-Maktab RBAC API Documentation

This document describes the backend REST API for managing Authentication, Users, Roles,
and Permissions in the E-Maktab system. It is intended for frontend (React) developers.

All endpoints are JSON-based and follow standard REST patterns.
Base path used in examples: /api/accounts/
Adjust the base URL according to your deployment (e.g. https://example.com/api/accounts/).

1. Authentication & JWT

Authentication is handled using JSON Web Tokens (JWT) via djangorestframework-simplejwt.
Most APIs require a valid access token in the Authorization header.

1.1 Login (Obtain Token Pair)

Method: POST
URL: /api/accounts/auth/login/

Request body:

```
{
  "username": "teacher001",
  "password": "StrongPassword123"
}
```

Successful response:

```
{
  "refresh": "<refresh_token>",
  "access": "<access_token>",
  "user": {
```

```
    "id": 5,
    "username": "teacher001",
    "full_name": "Teacher One",
    "email": "teacher1@example.com",
    "phone": "01712345678",
    "roles": ["teacher"],
    "is_active": true
  }
}
```


Notes:
• The access token is used for authenticating API calls.
• The refresh token is used to obtain a new access token when it expires.
• The embedded user object can initialize frontend auth state (current user).

1.2 Refresh Access Token

Method: POST
URL: /api/accounts/auth/refresh/

Request body:


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


Response:


```
{
  "access": "<new_access_token>"
}
```


1.3 Authenticated Requests

For any protected endpoint (e.g. /api/accounts/users/), include the access token in the Authorization header:

Authorization: Bearer <access_token>

Example:
GET /api/accounts/users/
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...

1.4 Current User (Profile) Endpoint

If implemented, a convenience endpoint can be exposed to get the current authenticated user:

Method: GET
URL: /api/accounts/users/me/

This returns the same shape as a normal User object, based on the token's authenticated user.
Frontend can use this to restore user profile state on page reload using a stored token.

2. Common API Behavior

Response Envelope for List Endpoints

List endpoints (e.g. GET /users/, GET /roles/, GET /permissions/) use paginated responses with the following JSON structure:

```
{
  "data": [
    { /* item 1 */ },
    { /* item 2 */ }
  ],
  "meta": {
    "total": 75,
    "page": 1,
    "page_size": 20,
    "total_pages": 4
  },
  "links": {
    "next": "/api/accounts/users/?page=2&page_size=20",
    "prev": null
  }
}
```

Single-item endpoints (GET /users/{id}/, POST /users/, etc.) return a single JSON object without the pagination wrapper.

Common Query Parameters

The following query parameters are shared by many list endpoints:

• page: current page number (default 1)
• page_size: number of items per page (default 20, max 200)
• ordering: field name to sort by (e.g. ordering=username or ordering=-created_at)

3. Users API

Base endpoint: /api/accounts/users/

3.1 User Object Shape

Example User JSON:


```
{
  "id": 1,
  "username": "student001",
  "full_name": "Student One",
  "email": "student1@example.com",
  "phone": "01700000000",
  "gender": "male",
  "date_of_birth": "2000-01-01",
  "address": "Dhaka, Bangladesh",
  "profile_photo": "/media/profiles/1.jpg",
  "emergency_contact_name": "Father Name",
  "emergency_contact_phone": "01711111111",
  "institution": 1,
  "roles": [1, 3],
  "is_active": true,
  "created_at": "2025-11-30T10:00:00Z",
  "updated_at": "2025-11-30T11:00:00Z"
}
```


3.2 Create User

Method: POST
URL: /api/accounts/users/

Request body (JSON):

```
{
  "username": "teacher001",
  "password": "StrongPassword123",
  "full_name": "Teacher One",
  "email": "teacher1@example.com",
  "phone": "01712345678",
  "roles": [2],
  "is_active": true
}
```

Notes:
• password is write-only; it will not appear in responses.
• roles is an array of Role IDs.
• Other fields (gender, date_of_birth, etc.) are optional during creation.

Successful response: 201 Created, with the full User JSON (without password).

3.3 Retrieve User

Method: GET
URL: /api/accounts/users/{id}/

Example: GET /api/accounts/users/5/

Response: 200 OK, single User JSON object.

3.4 Update User (Full / Partial)

Method: PUT (full update) or PATCH (partial update)
URL: /api/accounts/users/{id}/

Example PATCH body:

```
{
  "full_name": "Updated Name",
  "phone": "01799999999",
  "roles": [1, 2]
}
```

Notes:
• If password is included, it will be re-hashed and updated.
• If roles is provided, it replaces the existing role list (set on the backend).

3.5 Delete User

Method: DELETE
URL: /api/accounts/users/{id}/

Example: DELETE /api/accounts/users/10/

Response: 204 No Content on success.

3.6 Bulk Delete Users

Method: POST
URL: /api/accounts/users/bulk_delete/

Request body:

```
{
  "ids": [3, 4, 7]
}
```

Response:

```
{
  "deleted": 3
}
```

3.7 List & Search Users

Method: GET
URL: /api/accounts/users/

Supported query parameters:

• page: page number (e.g. ?page=2)
• page_size: items per page (e.g. ?page_size=50)
• search: performs case-insensitive search on username, full_name, phone
  - Example: /api/accounts/users/?search=shaon
• role: filter by Role ID
  - Example: /api/accounts/users/?role=2
• is_active: filter by active status (true/false)
  - Example: /api/accounts/users/?is_active=true
• ordering: sort results by a field
  - Allowed fields: id, username, full_name, created_at
  - Example: /api/accounts/users/?ordering=username

- Descending: /api/accounts/users/?ordering=-created_at

Example list request:

GET /api/accounts/users/?
search=teacher&role=2&is_active=true&ordering=full_name&page=1&page_size=2
0

4. Roles API

Base endpoint: /api/accounts/roles/

4.1 Role Object Shape

Example Role JSON:


```
{
  "id": 2,
  "name": "teacher",
  "description": "Teacher role",
  "permissions": [1, 2, 3],
  "is_system": false,
  "created_at": "2025-11-30T10:00:00Z",
  "updated_at": "2025-11-30T11:00:00Z"
}
```


4.2 Create Role

Method: POST
URL: /api/accounts/roles/

Request body:


```
{
  "name": "course_coordinator",
  "description": "Can manage course offerings and announcements",
  "permissions": [4, 5, 6]
}
```


Notes:
• permissions is an array of Permission IDs.
• is_system is typically controlled from backend/bootstrapping, not by frontend.

4.3 Retrieve Role

Method: GET
URL: /api/accounts/roles/{id}/

4.4 Update Role

Method: PUT/PATCH
URL: /api/accounts/roles/{id}/

Example PATCH body:

```
{
  "description": "Updated description",
  "permissions": [1, 2, 7]
}
```

4.5 Delete Role

Method: DELETE
URL: /api/accounts/roles/{id}/

4.6 Bulk Delete Roles

Method: POST
URL: /api/accounts/roles/bulk_delete/

Request body:

```
{
  "ids": [5, 6]
}
```

Response: { "deleted": 2 }

4.7 List & Search Roles

Method: GET
URL: /api/accounts/roles/

Supported query parameters:

• page, page_size, ordering (same behavior as Users)
• search: search in name and description
  - Example: /api/accounts/roles/?search=teacher
• is_system: filter system roles vs custom roles

- Example: /api/accounts/roles/?is_system=true
• ordering: allowed fields: id, name, is_system, created_at
- Example: /api/accounts/roles/?ordering=name

5. Permissions API

Base endpoint: /api/accounts/permissions/

Permissions are read-only from the frontend. They are defined and maintained by the backend using a bootstrapping process (e.g. bootup/permissions.json). Frontend can only read and use them when creating or editing roles.

5.1 Permission Object Shape

Example Permission JSON:


```
{
  "id": 10,
  "code": "COURSES_VIEW",
  "module": "courses",
  "description": "View courses and offerings",
  "created_at": "2025-11-29T09:00:00Z",
  "updated_at": "2025-11-29T09:00:00Z"
}
```


5.2 List & Search Permissions

Method: GET
URL: /api/accounts/permissions/

Supported query parameters:

• page, page_size, ordering (same pagination style)
• search: search in code, description, and module
- Example: /api/accounts/permissions/?search=course
• module: filter by exact module name
- Example: /api/accounts/permissions/?module=courses
• ordering: allowed fields: id, code, module, created_at
- Example: /api/accounts/permissions/?ordering=module

5.3 Retrieve Single Permission

Method: GET
URL: /api/accounts/permissions/{id}/

Note: There are no create/update/delete endpoints for permissions exposed to the frontend. Only the backend/bootstrapping process is allowed to modify permissions.

6. Frontend Integration Tips (React)

• For paginated lists, always expect the response structure: data + meta + links.
• For forms (create/update), send JSON bodies matching the shapes shown above.
• Use query params for search, filters, and ordering instead of custom endpoints.
• For role management screens:
  - Load available permissions from /permissions/ (optionally filter by module).
  - Let the admin select permission IDs and send them in the Role create/update payload.
• For user management screens:
  - Load roles from /roles/ for dropdowns (multi-select).
  - Display roles as labels/tags using the IDs from the user object.
• Bulk delete operations use POST with a simple { "ids": [...] } JSON body.

This API shape is designed to be stable and versionable. If changes are required later, they should be additive (e.g. adding new fields) so existing frontend code continues to work.