Equipment API Documentation

Base URL

/api/equipment

Authentication

This API uses role-based access control. Certain endpoints require specific roles as indicated below.

Endpoints

1. List All Equipment

Retrieve a list of all equipment items.

Endpoint: `GET /api/equipment`

Authorization: None required

Response:

- **Status Code:** `200 OK`
- **Body:** Array of Equipment objects

Example Response:

json

```
[
  {
    "id": 1,
    "name": "Laptop",
    "category": "Electronics",
    "quantity": 15,
    "status": "Available"
  },
  {
    "id": 2,
    "name": "Projector",
    "category": "Presentation",
    "quantity": 5,
    "status": "Available"
  }
]
```


---

2. Get Equipment by ID

Retrieve a specific equipment item by its ID.

Endpoint: `GET /api/equipment/{id}`

Authorization: None required

Path Parameters:

- `id` (Long) - The unique identifier of the equipment

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Equipment object

Example Response:

json


```
{
  "id": 1,
  "name": "Laptop",
  "category": "Electronics",
  "quantity": 15,
  "status": "Available"
}
```


Error:

- **Status Code:** `404 Not Found`
- **Body:**


json

```
{
  "message": "Equipment not found with id: 1",
  "timestamp": "2025-11-09T10:30:00"
}
```

---

3. Create New Equipment

Add a new equipment item to the system.

Endpoint: `POST /api/equipment`

Authorization: `ROLE_ADMIN` required

Request Body:


json

```
{
  "name": "Laptop",
  "category": "Electronics",
  "quantity": 15,
  "status": "Available"
}
```

Response:

Success:

- **Status Code:** `201 Created`
- **Body:** Created Equipment object

Example Response:


json

```
{
  "id": 3,
  "name": "Laptop",
  "category": "Electronics",
  "quantity": 15,
  "status": "Available"
}
```


Error:

- **Status Code:** `403 Forbidden` - User does not have ADMIN role
- **Status Code:** `400 Bad Request` - Invalid request body

---

4. Update Equipment

Update an existing equipment item.

Endpoint: `PUT /api/equipment/{id}`

Authorization: `ROLE_ADMIN` required

Path Parameters:

- `id` (Long) - The unique identifier of the equipment to update

Request Body:


json


```
{
  "name": "Updated Laptop",
  "category": "Electronics",
  "quantity": 20,
  "status": "Maintenance"
}
```


Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Updated Equipment object

Example Response:


json

```
{
  "id": 1,
  "name": "Updated Laptop",
  "category": "Electronics",
  "quantity": 20,
  "status": "Maintenance"
}
```


Error:

- **Status Code:** `403 Forbidden` - User does not have ADMIN role
- **Status Code:** `404 Not Found` - Equipment with specified ID not found
- **Status Code:** `400 Bad Request` - Invalid request body

---

5. Delete Equipment

Remove an equipment item from the system.

Endpoint: `DELETE /api/equipment/{id}`

Authorization: `ROLE_ADMIN` required

Path Parameters:

- `id` (Long) - The unique identifier of the equipment to delete

Response:

Success:

- **Status Code:** `204 No Content`
- **Body:** Empty

Error:

- **Status Code:** `403 Forbidden` - User does not have ADMIN role
- **Status Code:** `404 Not Found` - Equipment with specified ID not found

---

Common HTTP Status Codes

Status Code	Description
200 OK	Request successful, resource returned
201 Created	Resource successfully created
204 No Content	Request successful, no content to return
400 Bad Request	Invalid request format or parameters
403 Forbidden	User lacks required permissions
404 Not Found	Requested resource does not exist
500 Internal Server Error	Unexpected server error

Authorization Roles

Role	Description	Access
ROLE_ADMIN	Administrator	Full access to all endpoints including create, update, and delete operations
Public	Any authenticated or unauthenticated user	Read-only access (GET operations)

Notes

- All POST and PUT requests must include a valid JSON body with the appropriate Content-Type header: `Content-Type: application/json`
- ID fields are auto-generated and should not be included in POST requests
- Equipment fields such as `name`, `category`, `quantity`, and `status` may vary based on your Entity definition

User API Documentation

Base URL

```
/api/users
```

Authentication

This API uses JWT token-based authentication with role-based access control. Authentication endpoints (signup/login) are public, while other endpoints require specific roles.

Endpoints

1. User Signup

Register a new user account.

Endpoint: `POST /api/users/signup`

Authorization: Public (no authentication required)

Request Body:

```json
{
  "username": "johndoe",
  "email": "john.doe@example.com",
  "password": "SecurePassword123!",
  "firstName": "John",
  "lastName": "Doe"
}
```

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** JWT token for authentication

Example Response:

```json
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaW
}
```

Error:

- **Status Code:** `400 Bad Request` - Invalid request data or user already exists
- **Status Code:** `409 Conflict` - Email already registered

---

2. User Login

Authenticate an existing user.

Endpoint: `POST /api/users/login`

Authorization: Public (no authentication required)

Request Body:

```json
{
  "email": "john.doe@example.com",
  "password": "SecurePassword123!"
}
```

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** JWT token for authentication

Example Response:

```json
```

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaW
}
```


Error:

- **Status Code:** `401 Unauthorized` - Invalid email or password
- **Status Code:** `400 Bad Request` - Missing required fields

---

3. Get All Users

Retrieve a list of all registered users.

Endpoint: `GET /api/users`

Authorization: `ROLE_ADMIN` required

Headers:

```
Authorization: Bearer <your_jwt_token>
```

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Array of User objects

Example Response:

```
json
```

```
[
  {
    "id": 1,
    "username": "johndoe",
    "email": "john.doe@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "ROLE_USER",
    "createdAt": "2025-01-15T10:30:00"
  },
  {
    "id": 2,
    "username": "janedoe",
    "email": "jane.doe@example.com",
    "firstName": "Jane",
    "lastName": "Doe",
    "role": "ROLE_STAFF",
    "createdAt": "2025-01-16T14:20:00"
  }
]
```


Error:

- **Status Code:** `403 Forbidden` - User does not have ADMIN role
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

4. Get User by ID

Retrieve a specific user by their ID.

Endpoint: `GET /api/users/{id}`

Authorization: `ROLE_ADMIN` or `ROLE_STAFF` required

Headers:

```
Authorization: Bearer <your_jwt_token>
```

Path Parameters:

- `id` (Long) - The unique identifier of the user

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** User object

Example Response:

```
json

{
  "id": 1,
  "username": "johndoe",
  "email": "john.doe@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "role": "ROLE_USER",
  "createdAt": "2025-01-15T10:30:00"
}
```

Error:

- **Status Code:** `403 Forbidden` - User does not have ADMIN or STAFF role
- **Status Code:** `404 Not Found` - User with specified ID not found
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

5. Update User

Update an existing user's information.

Endpoint: `PUT /api/users/{id}`

Authorization: `ROLE_ADMIN` required

Headers:

```
Authorization: Bearer <your_jwt_token>
```

Path Parameters:

- `id` (Long) - The unique identifier of the user to update

Request Body:

```json
{
  "username": "johndoe_updated",
  "email": "john.updated@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "role": "ROLE_STAFF"
}
```

Response:

Success:

- **Status Code:** 200 OK
- **Body:** Updated User object

Example Response:

```json
{
  "id": 1,
  "username": "johndoe_updated",
  "email": "john.updated@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "role": "ROLE_STAFF",
  "createdAt": "2025-01-15T10:30:00",
  "updatedAt": "2025-11-09T15:45:00"
}
```

Error:

- **Status Code:** 403 Forbidden - User does not have ADMIN role
- **Status Code:** 404 Not Found - User with specified ID not found
- **Status Code:** 400 Bad Request - Invalid request body
- **Status Code:** 401 Unauthorized - Missing or invalid JWT token

---

Remove a user from the system.

Endpoint: `DELETE /api/users/{id}`

Authorization: `ROLE_ADMIN` required

Headers:

```
Authorization: Bearer <your_jwt_token>
```

Path Parameters:

- `id` (Long) - The unique identifier of the user to delete

Response:

Success:

- **Status Code:** `200 OK` or `204 No Content`
- **Body:** Empty

Error:

- **Status Code:** `403 Forbidden` - User does not have ADMIN role
- **Status Code:** `404 Not Found` - User with specified ID not found
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

Common HTTP Status Codes

Status Code	Description
`200 OK`	Request successful, resource returned
`201 Created`	Resource successfully created
`204 No Content`	Request successful, no content to return
`400 Bad Request`	Invalid request format or parameters
`401 Unauthorized`	Missing or invalid authentication credentials
`403 Forbidden`	User lacks required permissions for the resource
`404 Not Found`	Requested resource does not exist
`409 Conflict`	Resource conflict (e.g., duplicate email)

Status Code	Description
`500 Internal Server Error`	Unexpected server error

Authorization Roles

Role	Description	Access
Public	Unauthenticated users	Signup and Login only
ROLE_USER	Regular authenticated user	Basic access (varies by implementation)
ROLE_STAFF	Staff member	Can view user details (read access)
ROLE_ADMIN	Administrator	Full access including user management (create, read, update, delete)

Role-Based Access Summary

Endpoint	Method	Public	USER	STAFF	ADMIN
`/signup`	POST	✅	✅	✅	✅
`/login`	POST	✅	✅	✅	✅
`/`	GET	❌	❌	❌	✅
`/{id}`	GET	❌	❌	✅	✅
`/{id}`	PUT	❌	❌	❌	✅
`/{id}`	DELETE	❌	❌	❌	✅

Authentication Flow

1. **Signup/Login**: Use `/api/users/signup` or `/api/users/login` to obtain a JWT token
2. **Store Token**: Save the returned token securely (e.g., in localStorage or httpOnly cookie)
3. **Authenticate Requests**: Include the token in the Authorization header for protected endpoints:

```
Authorization: Bearer <your_jwt_token>
```

4. **Token Expiration**: If a request returns `401 Unauthorized`, the token may have expired - login again to get a new token

---

Notes

- All POST and PUT requests must include a valid JSON body with the appropriate Content-Type header: `Content-Type: application/json`
- Passwords are securely hashed before storage (never stored in plain text)
- JWT tokens contain user identity and role information
- Token expiration time depends on your JWT configuration
- The User entity may contain additional fields based on your implementation (e.g., phone number, address, profile picture)

Borrow Request API Documentation

Base URL

```
/api/borrow
```

Authentication

This API uses JWT token-based authentication with role-based access control. All endpoints require authentication with appropriate roles.

Headers Required:

```
Authorization: Bearer <your_jwt_token>
```

Endpoints

1. Create Borrow Request

Submit a new equipment borrow request.

Endpoint: `POST /api/borrow/request`

Authorization: `ROLE_STUDENT` or `ROLE_STAFF` required

Request Body:

```json
{
  "equipmentId": 5,
  "quantity": 2,
  "startDate": "2025-11-15T09:00:00",
  "endDate": "2025-11-20T17:00:00"
}
```

Response:

Success:

- **Status Code:** `201 Created`

- **Body:** Created BorrowRequest object

Example Response:

```json
{
  "id": 101,
  "userId": 42,
  "equipmentId": 5,
  "quantity": 2,
  "startDate": "2025-11-15T09:00:00",
  "endDate": "2025-11-20T17:00:00",
  "status": "PENDING",
  "requestDate": "2025-11-09T10:30:00",
  "approvedBy": null,
  "issuedBy": null,
  "comment": null
}
```

Error:

- **Status Code:** `403 Forbidden` - User does not have STUDENT or STAFF role
- **Status Code:** `400 Bad Request` - Invalid request data or equipment not available
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

2. Get My Borrow Requests

Retrieve all borrow requests submitted by the logged-in user.

Endpoint: `GET /api/borrow/my`

Authorization: `ROLE_STUDENT` or `ROLE_STAFF` required

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Array of BorrowRequest objects

Example Response:

```json
[
  {
    "id": 101,
    "userId": 42,
    "equipmentId": 5,
    "quantity": 2,
    "startDate": "2025-11-15T09:00:00",
    "endDate": "2025-11-20T17:00:00",
    "status": "APPROVED",
    "requestDate": "2025-11-09T10:30:00",
    "approvedBy": 3,
    "issuedBy": null,
    "comment": "Approved for lab use"
  },
  {
    "id": 98,
    "userId": 42,
    "equipmentId": 12,
    "quantity": 1,
    "startDate": "2025-11-10T09:00:00",
    "endDate": "2025-11-12T17:00:00",
    "status": "RETURNED",
    "requestDate": "2025-11-08T14:20:00",
    "approvedBy": 3,
    "issuedBy": 3,
    "comment": "Equipment returned in good condition"
  }
]
```

Error:

- **Status Code:** `403 Forbidden` - User does not have STUDENT or STAFF role
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

3. Approve Borrow Request

Approve a pending borrow request.

Endpoint: `PUT /api/borrow/{id}/approve`

Authorization: `ROLE_STAFF` or `ROLE_ADMIN` required

Path Parameters:

- `id` (Long) - The unique identifier of the borrow request

Request Body:

```
json

{
  "comment": "Approved for educational purposes. Please return on time."
}
```

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Updated BorrowRequest object

Example Response:

```
json

{
  "id": 101,
  "userId": 42,
  "equipmentId": 5,
  "quantity": 2,
  "startDate": "2025-11-15T09:00:00",
  "endDate": "2025-11-20T17:00:00",
  "status": "APPROVED",
  "requestDate": "2025-11-09T10:30:00",
  "approvedBy": 3,
  "approvalDate": "2025-11-09T15:45:00",
  "issuedBy": null,
  "comment": "Approved for educational purposes. Please return on time."
}
```

Error:

- **Status Code:** `403 Forbidden` - User does not have STAFF or ADMIN role
- **Status Code:** `404 Not Found` - Borrow request not found

- **Status Code:** `400 Bad Request` - Request already processed or invalid status
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

4. Issue Equipment

Mark equipment as issued to the borrower.

Endpoint: `PUT /api/borrow/{id}/issue`

Authorization: `ROLE_STAFF` or `ROLE_ADMIN` required

Path Parameters:

- `id` (Long) - The unique identifier of the borrow request

Request Body: None required

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Updated BorrowRequest object

Example Response:

```json
{
  "id": 101,
  "userId": 42,
  "equipmentId": 5,
  "quantity": 2,
  "startDate": "2025-11-15T09:00:00",
  "endDate": "2025-11-20T17:00:00",
  "status": "ISSUED",
  "requestDate": "2025-11-09T10:30:00",
  "approvedBy": 3,
  "approvalDate": "2025-11-09T15:45:00",
  "issuedBy": 3,
  "issueDate": "2025-11-15T09:15:00",
  "comment": "Approved for educational purposes. Please return on time."
}
```

Error:

- **Status Code:** `403 Forbidden` - User does not have STAFF or ADMIN role
- **Status Code:** `404 Not Found` - Borrow request not found
- **Status Code:** `400 Bad Request` - Request not approved or already issued
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

5. Mark Equipment as Returned

Record that borrowed equipment has been returned.

Endpoint: `PUT /api/borrow/{id}/return`

Authorization: `ROLE_STAFF` or `ROLE_ADMIN` required

Path Parameters:

- `id` (Long) - The unique identifier of the borrow request

Request Body: None required

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Updated BorrowRequest object

Example Response:

```
json
```

```
{
  "id": 101,
  "userId": 42,
  "equipmentId": 5,
  "quantity": 2,
  "startDate": "2025-11-15T09:00:00",
  "endDate": "2025-11-20T17:00:00",
  "status": "RETURNED",
  "requestDate": "2025-11-09T10:30:00",
  "approvedBy": 3,
  "approvalDate": "2025-11-09T15:45:00",
  "issuedBy": 3,
  "issueDate": "2025-11-15T09:15:00",
  "returnDate": "2025-11-20T16:30:00",
  "comment": "Equipment returned in good condition"
}
```


Error:

- **Status Code:** `403 Forbidden` - User does not have STAFF or ADMIN role
- **Status Code:** `404 Not Found` - Borrow request not found
- **Status Code:** `400 Bad Request` - Equipment not issued or already returned
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

6. Reject Borrow Request

Reject a pending borrow request.

Endpoint: `PUT /api/borrow/{id}/reject`

Authorization: `ROLE_STAFF` or `ROLE_ADMIN` required

Path Parameters:

- `id` (Long) - The unique identifier of the borrow request

Request Body:

```
json
```

```
{
  "comment": "Equipment not available for requested dates. Please choose alternative dates."
}
```

Response:

Success:

- **Status Code:** 200 OK
- **Body:** Updated BorrowRequest object

Example Response:

```
json

{
  "id": 102,
  "userId": 45,
  "equipmentId": 8,
  "quantity": 1,
  "startDate": "2025-11-12T09:00:00",
  "endDate": "2025-11-14T17:00:00",
  "status": "REJECTED",
  "requestDate": "2025-11-09T11:00:00",
  "approvedBy": null,
  "rejectedBy": 3,
  "rejectionDate": "2025-11-09T16:00:00",
  "issuedBy": null,
  "comment": "Equipment not available for requested dates. Please choose alternative dates."
}
```

Error:

- **Status Code:** 403 Forbidden - User does not have STAFF or ADMIN role
- **Status Code:** 404 Not Found - Borrow request not found
- **Status Code:** 400 Bad Request - Request already processed
- **Status Code:** 401 Unauthorized - Missing or invalid JWT token

---

7. Get Pending Requests

Retrieve all pending borrow requests (for staff/admin review).

Endpoint: `GET /api/borrow/pending`

Authorization: `ROLE_STAFF` or `ROLE_ADMIN` required

Response:

Success:

- **Status Code:** `200 OK`
- **Body:** Array of pending BorrowRequest objects

Example Response:

```
json
```

```
[
  {
    "id": 103,
    "userId": 48,
    "equipmentId": 7,
    "quantity": 3,
    "startDate": "2025-11-16T09:00:00",
    "endDate": "2025-11-18T17:00:00",
    "status": "PENDING",
    "requestDate": "2025-11-09T12:30:00",
    "approvedBy": null,
    "issuedBy": null,
    "comment": null
  },
  {
    "id": 104,
    "userId": 52,
    "equipmentId": 12,
    "quantity": 1,
    "startDate": "2025-11-17T09:00:00",
    "endDate": "2025-11-19T17:00:00",
    "status": "PENDING",
    "requestDate": "2025-11-09T13:15:00",
    "approvedBy": null,
    "issuedBy": null,
    "comment": null
  }
]
```


Error:

- **Status Code:** `403 Forbidden` - User does not have STAFF or ADMIN role
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

---

8. Get Issued Requests

Retrieve all currently issued (non-returned) equipment.

Endpoint: `GET /api/borrow/issued`

Authorization: `ROLE_STAFF` or `ROLE_ADMIN` required

Response:

Success:

- **Status Code:** 200 OK
- **Body:** Array of issued BorrowRequest objects

Example Response:

```json
[
  {
    "id": 100,
    "userId": 40,
    "equipmentId": 3,
    "quantity": 1,
    "startDate": "2025-11-08T09:00:00",
    "endDate": "2025-11-15T17:00:00",
    "status": "ISSUED",
    "requestDate": "2025-11-07T10:00:00",
    "approvedBy": 3,
    "approvalDate": "2025-11-07T14:30:00",
    "issuedBy": 3,
    "issueDate": "2025-11-08T09:10:00",
    "comment": "Approved for project work"
  },
  {
    "id": 101,
    "userId": 42,
    "equipmentId": 5,
    "quantity": 2,
    "startDate": "2025-11-09T09:00:00",
    "endDate": "2025-11-12T17:00:00",
    "status": "ISSUED",
    "requestDate": "2025-11-08T15:30:00",
    "approvedBy": 3,
    "approvalDate": "2025-11-08T16:00:00",
    "issuedBy": 3,
    "issueDate": "2025-11-09T09:05:00",
    "comment": "Handle with care"
  }
]
```

Error:

- **Status Code:** `403 Forbidden` - User does not have STAFF or ADMIN role
- **Status Code:** `401 Unauthorized` - Missing or invalid JWT token

Borrow Request Lifecycle

```
PENDING → APPROVED → ISSUED → RETURNED
  ↓
REJECTED
```

1. **PENDING**: Initial status when request is created
2. **APPROVED**: Staff/Admin approves the request
3. **ISSUED**: Equipment is physically handed to the borrower
4. **RETURNED**: Equipment is returned by the borrower
5. **REJECTED**: Request is denied (terminal state)

Common HTTP Status Codes

Status Code	Description
`200 OK`	Request successful, resource returned
`201 Created`	Borrow request successfully created
`400 Bad Request`	Invalid request data or invalid state transition
`401 Unauthorized`	Missing or invalid authentication credentials
`403 Forbidden`	User lacks required permissions
`404 Not Found`	Borrow request not found
`500 Internal Server Error`	Unexpected server error

Authorization Roles

Role	Description	Permissions
ROLE_STUDENT	Student user	Can create and view own requests
ROLE_STAFF	Staff member	Can manage all requests (approve, issue, reject, return) + all student permissions

Role	Description	Permissions
ROLE_ADMIN	Administrator	Same as STAFF (full access to borrow management)

Role-Based Access Summary

Endpoint	Method	STUDENT	STAFF	ADMIN
`/request`	POST	✅	✅	✅
`/my`	GET	✅	✅	✅
`/{id}/approve`	PUT	❌	✅	✅
`/{id}/issue`	PUT	❌	✅	✅
`/{id}/return`	PUT	❌	✅	✅
`/{id}/reject`	PUT	❌	✅	✅
`/pending`	GET	❌	✅	✅
`/issued`	GET	❌	✅	✅

Workflow Examples

Student Workflow

1. **Create Request**: Student submits borrow request via `POST /api/borrow/request`
2. **Track Status**: Student checks request status via `GET /api/borrow/my`
3. **Collect Equipment**: Once approved and issued, student collects equipment
4. **Return Equipment**: Student returns equipment to staff/admin

Staff/Admin Workflow

1. **Review Pending**: Staff views pending requests via `GET /api/borrow/pending`
2. **Approve/Reject**: Staff approves via `PUT /api/borrow/{id}/approve` or rejects via `PUT /api/borrow/{id}/reject`
3. **Issue Equipment**: Staff issues equipment via `PUT /api/borrow/{id}/issue`
4. **Track Issued**: Staff monitors issued equipment via `GET /api/borrow/issued`
5. **Mark Returned**: Staff marks equipment as returned via `PUT /api/borrow/{id}/return`

Notes

- All requests must include a valid JWT token in the Authorization header
- User ID is automatically extracted from the JWT token (no need to include in request body)
- Dates should be in ISO 8601 format: `YYYY-MM-DDTHH:mm:ss`
- Comments are optional but recommended for approval/rejection decisions
- Equipment availability is checked during request creation
- Status transitions must follow the lifecycle (e.g., cannot issue a pending request without approval)
- Students can only view their own requests; Staff/Admin can view all requests