PHASE 1 - SCHOOL EQUIPMENT LENDING PORTAL

API Documentation:

Authentication Routes

BASE URL : http://localhost:5000/api

1. Register User

Endpoint: POST /api/auth/register
Description: Registers a new user (Student, Staff, or Warden).

Request Body:

```
{
  "name": “Chandler Bing",
  "email": "chandlerbing@example.com",
  "password": "securePassword123",
  "role": "student"
}
```

Response:

```
{ "_id": "690eb40af443a6a3f2a79ca8",
"name": "Chandler Bing",
"email": "chandlerbing@example.com",
"role": "student",
"token": "jwttoken" }
```

---

2. Login

Endpoint: POST /api/auth/login
Description: Authenticates a user and returns a JWT token.

Request Body:

```
{
  "email": "chandlerbing@example.com ",
  "password": "securePassword123"
}
```

Response:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI..."
}
```

Equipment Routes

1. Get All Equipment

Endpoint: GET /api/equipment
Description: Fetch all available equipment.
Access: All roles

Response:

```
[
  {
    "_id": "6740a65b...",
    "name": "Cricket Bat",
    "category": "Sports",
    "condition": "Good",
    "quantity": 5
  }
]
```

---

2. Add New Equipment

Endpoint: POST /api/equipment
Description: Add new equipment to the system.
Access: Warden only

Request Body:

```
{
  "name": "Convex Lens",
  "category": "Lab",
  "condition": "New",
  "quantity": 10,
  “available” : 10
}
```

Response:

```
{
"name": "Convex Lens",
"category": "Lab",
"condition": "New",
"quantity": 10,
"available": 10,
"_id": "690eb7f9f443a6a3f2a79cb1",
"createdAt": "2025-11-08T03:24:41.899Z",
"__v": 0
}
```

---

3. Update Equipment

Endpoint: PUT /api/equipment/:id
Description: Update existing equipment details.
Access: Warden only

Request Body:

{
  "condition": "Fair",
}

Response:

{
"_id": "690eb7f9f443a6a3f2a79cb1",
 "name": "Convex Lens",
"category": "Lab",
"condition": "Fair",
"quantity": 10,
"available": 10,
"createdAt": "2025-11-08T03:24:41.899Z",
 "__v": 0
}

---

4. Delete Equipment

Endpoint: DELETE /api/equipment/:id
Description: Remove equipment from the system.
Access: Warden only

Response:

{
  "message": "Equipment deleted successfully"
}

Borrow Request Routes

1. Submit Borrow Request

Endpoint: POST /api/borrow
Description: Student submits a request to borrow equipment.
Access: Student only

Request Body:

{
  "equipmentId": "6740a65b...",
  "quantity": 1
}

Response:


```
{
  "equipment": "68fc64accbc32a9d41fd3a4f",
  "student": "68fc56af1ad571146d256c81",
  "quantity": 1,
  "status": "Pending",
  "_id": "690ed3f82482096c7a609ac9",
  "requestDate": "2025-11-08T05:24:08.155Z",
  "__v": 0
}
```


---

2. Get My Borrow Requests

Endpoint: GET /api/borrow/my
Description: Retrieve all borrow requests submitted by the logged-in student.
Access: Student

Response:


```
[
  {
  "_id": "690ed3f82482096c7a609ac9",
  "equipment": {
    "_id": "68fc64accbc32a9d41fd3a4f",
    "name": "Camera",
    "category": "Project"
  },
  "student": "68fc56af1ad571146d256c81",
  "quantity": 1,
  "status": "Pending",
  "requestDate": "2025-11-08T05:24:08.155Z",
  "__v": 0
}
]
```


---

3. Get All Borrow Requests

Endpoint: GET /api/borrow
Description: View all student borrow requests.
Access: Warden

Response:


```
[
{
  "_id": "690ed3f82482096c7a609ac9",
  "equipment": {
    "_id": "68fc64accbc32a9d41fd3a4f",
```

```
    "name": "Camera",
    "category": "Project"
  },
  "student": {
    "_id": "68fc56af1ad571146d256c81",
    "name": "Phoebe Buffay",
    "email": "phoebebuffay@example.com"
  },
  "quantity": 1,
  "status": "Pending",
  "requestDate": "2025-11-08T05:24:08.155Z",
  "__v": 0
}
]
```


---

4. Approve or Reject Request

Endpoint: PUT /api/borrow/approve
Description: Approve or reject a borrow request.
Access: Warden

Request Body:


```
{
  "requestId": "690ed3f...",
  "approve": true
}
```


Response:


```
{
  "_id": "690ed3f82482096c7a609ac9",
  "equipment": "68fc64accbc32a9d41fd3a4f",
  "student": "68fc56af1ad571146d256c81",
  "quantity": 1,
  "status": "Approved",
  "requestDate": "2025-11-08T05:24:08.155Z",
  "__v": 0,
  "warden": "68fc56961ad571146d256c7e",
  "dueDate": "2025-11-08T05:31:24.228Z"
}
```


---

5. Get Approved Requests

Endpoint: GET /api/borrow/approved
Description: Retrieve all approved requests for equipment issue.
Access: Staff

Response:

```
[
 {
 "_id": "690ed3f82482096c7a609ac9",
 "equipment": {
  "_id": "68fc64accbc32a9d41fd3a4f",
  "name": "Camera",
  "category": "Project"
 },
 "student": {
  "_id": "68fc56af1ad571146d256c81",
  "name": "Phoebe Buffay",
  "email": "phoebebuffay@example.com"
 },
 "quantity": 1,
 "status": "Approved",
 "requestDate": "2025-11-08T05:24:08.155Z",
 "__v": 0,
 "dueDate": "2025-11-08T05:31:24.228Z",
 "warden": "68fc56961ad571146d256c7e"
}
]
```


---

6. Mark as Returned

Endpoint: PUT /api/borrow/return
Description: Mark equipment as returned after use.
Access: Staff

Request Body:


```
{
 "requestId": "6740a7c9..."
}
```


Response:


```
{
 "_id": "690ed3f82482096c7a609ac9",
 "equipment": "68fc64accbc32a9d41fd3a4f",
 "student": "68fc56af1ad571146d256c81",
 "quantity": 1,
 "status": "Returned",
 "requestDate": "2025-11-08T05:24:08.155Z",
 "__v": 0,
 "dueDate": "2025-11-08T05:31:24.228Z",
 "warden": "68fc56961ad571146d256c7e",
 "returnDate": "2025-11-08T05:39:15.077Z"
}
```

To Note

- All protected routes require a valid **JWT token** in the Authorization header: Authorization: Bearer <token>
- Roles are validated on the server before allowing access to endpoints.

Database Schema / ER Diagram:

DataModel

User

Field	Type	Description
_id	ObjectId	Primary key
name	String	Full name
email	String	Unique email
password	String	Encrypted password
role	String	Enum: student/staff/warden

Equipment

Field	Type	Description
_id	ObjectId	Primary key
name	String	Equipment name
category	String	Sports / Lab / Music / Project
condition	String	New/Good/Fair/Poor
quantity	Number	Total items available
available	Number	Current available count
createdAt	Date	Auto-calculated when created

BorrowRequest

Field	Type	Description
_id	ObjectId	Primary key
student	ObjectId (ref: User)	Requesting student
equipment	ObjectId (ref: Equipment)	Equipment requested
quantity	Number	Number of items requested
status	String	Pending / Approved / Rejected / Overdue
dueDate	Date	Auto-calculated when approved
requestDate	Date	Auto-calculated when created
issueDate	Date	Auto-calculated when approved
warden	ObjectId (ref: User)	Warden who approved/rejected
staff	ObjectId (ref: User)	Staff who marks returned

remarks	String	Additional Notes

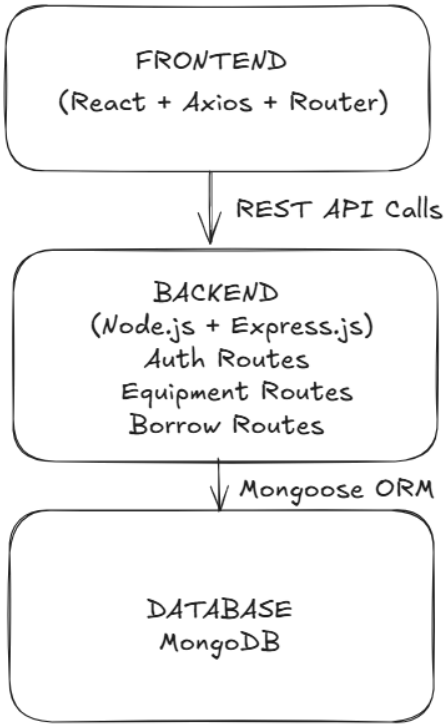
ER Diagram:

User (1) ——< BorrowRequest >——(1) Equipment

Relationship	**Type**
User (Student) → BorrowRequest	1 : N
Equipment → BorrowRequest	1 : N
User (Warden) → BorrowRequest (approval)	1 : N
User (Staff) → BorrowRequest (return)	1 : N

Architecture Diagram:

MERN Stack Architecture Overview



Assumptions:

1. Login Simulation & Token Design: The authentication mechanism uses JWT tokens returned upon successful login; stored in localStorage and sent in HTTP headers for subsequent API calls. All protected endpoints validate the token and user role.
2. Role-based Views: Navigation and permissions are determined by the role in the JWT: "student", "warden" and "staff".

3. Equipment Quantity Management: The field `available` in Equipment reflects how many items can be lent out.
4. Date Handling: BorrowRequest has fields for request, issue, due, and return dates.
5. Approval Workflow: Requests are created by students, approved/rejected by wardens, and issued/returned by staff.
6. UI Authentication State: React state/context manages authentication state and role-guarded routes.