Library Management System –API Design (v1)

Raw documentation link: doc

---

1) Conventions & Standards

1. **Base URL**
   `/` (example paths below are relative to the API root). You may later version as `/api/v1/…` without changing payloads.
2. **Auth**
   - Login issues a **JWT access token** (Bearer).
   - Send it in: `Authorization: Bearer <token>`.
   - Roles: `user`, `admin`.
3. **IDs & Dates**
   - All IDs are strings or integers (server-decides; examples use strings).
   - Dates are ISO-8601 strings (e.g., `2025-08-27`).
   - Times may be added later as ISO-8601 datetime.
4. **Success Status Codes**
   - `200 OK` (GET/PUT/PATCH), `201 Created` (POST create), `204 No Content` (DELETE succeed, or updates with no body).
5. **Error Model (uniform)**

```
{
  "error": {
    "code": "string",      // MACHINE_READABLE
    "message": "string",   // human-friendly summary
    "details": {}           // optional object for field errors, limits,
etc.
  }
}
```

   - Common codes: `INVALID_CREDENTIALS`, `UNAUTHORIZED`, `FORBIDDEN`, `NOT_FOUND`, `CONFLICT`, `VALIDATION_ERROR`, `RATE_LIMITED`, `INTERNAL_ERROR`.
6. **Lists, Count, Pagination**
   - All list endpoints may include these **optional** query params (non-breaking):

- `page` (default `1`), `page_size` (default `20`, max `100`).
- `q` (search term), `sort` (e.g., `title,-created_at`).

- Response wraps data with `meta.total` for count. (This preserves your need for counts while keeping list responses useful.)

Standard list wrapper:

```
{
  "data": [ /* array of items */ ],
  "meta": { "total": 123, "page": 1, "page_size": 20 }
}
```

7. **Backward-compatible aliases & typo fixes**
   - Canonical field **`book_availability`**; accept legacy misspelling `book_availibility` on input, still return canonical in responses.
   - Where your original paths used forms like `details<book_id>` or `all<book_id>`, the canonical form uses `/books/{book_id}` style. **Aliases** keep the originals working (documented below).

---

2) Auth

2.1 Login (admin provides credentials to users)

POST `users/login`
Request

```
{
  "user_id": "U1001",
  "password": "••••••••"
}
```

If you prefer email login, allow `{ "email": "...", "password": "..." }` as an alternative input.

Response – 200

```
{
  "access_token": "jwt-token-string",
  "token_type": "Bearer",
  "user": {
    "user_id": "U1001",
    "user_name": "Jamil Ahmed",
    "user_email": "jamil@example.com",
```

```
    "User_photo": .jpg/ .png/.jpeg
    "role": "user"
  }
}
```

Errors - `401 UNAUTHORIZED` → `INVALID_CREDENTIALS`
- `422 UNPROCESSABLE ENTITY` → `VALIDATION_ERROR` (missing fields)

---

3) Core Models

3.1 User

```
{
  "user_id": "U1001",
  "user_name": "Jamil Ahmed",
  "user_email": "jamil@example.com"
}
```

3.2 Book (public shape)

```
{
  "book_id": "B42",
  "book_title": "Clean Code",
  "book_photo": "https://.../clean-code.jpg",
  "book_availability": true
}
```

Accept `book_availibility` on input; always respond with `book_availability`.

3.3 Book (full/admin/detail)

```
{
  "book_id": "B42",
  "book_title": "Clean Code",
  "book_author": "Robert C. Martin",
  "book_category_id": "C2",
  "book_rating": 4.8,
  "book_photo": "https://.../clean-code.jpg",
  "book_details": "...",
  "book_availability": true,
  "category_title": "Software Engineering"   // populated from category table
  "book_count": 30
}
```

3.4 Category

```
{
  "category_id": "C2",
  "category_title": "Software Engineering"
}
```

3.5 Borrow Record

```
{
  "borrow_id": "BR1009",
  "user_id": "U1001",
  "user_name": "Jamil Ahmed",    // denormalized convenience
  "book_id": "B42",
  "borrow_date": "2025-08-20",
  "return_date": "2025-09-03",
  "borrow_status": "borrowed",    // one of: returned | overdue | borrowed
  "request_status": "pending"     // one of: accept | pending | reject
}
```

---

4) Users

4.1 List Users (and count)

GET `users/`
Response – 200

```
{
  "data": [
    { "user_id": "U1001", "user_name": "Jamil Ahmed", "user_email":
"jamil@example.com" }
  ],
  "meta": { "total": 57, "page": 1, "page_size": 20 }
}
```

> **Preserves your requirement to "count all users"** via `meta.total`.
> **Alias kept:** `GET users/all` may return the same structure.

Errors
`401 UNAUTHORIZED` if not logged in (or `403 FORBIDDEN` if restricted by role policy you define).

---

5) Public Books (Home Page)

5.1 List Books (title, photo, availability)

GET `books/`
Query (optional): `page`, `page_size`, `q`, `sort`
Response – 200

```
{
  "data": [
    {
      "book_id": "B42",
```

```
      "book_title": "Clean Code",
      "book_photo": "https://.../clean-code.jpg",
      "book_availability": true
    }
  ],
  "meta": { "total": 340, "page": 1, "page_size": 20 }
}
```

5.2 Book Details

GET `books/{book_id}`
Aliases: `books/details{book_id}`
Response – 200 → **Book (full)** model.

Errors
`404 NOT_FOUND` → `NOT_FOUND` if `book_id` unknown.

---

6) Borrowing Flow

6.1 Borrow a Book (from details page)

POST `books/{book_id}/borrow`

Request (one of):

```
{
  "user_id": "U1001",
  "borrow_date": "2025-08-20",    // this date is set by current date time
  "return_date": "2025-09-03"   // this date is set by current date + borrow_day_lim
}
```

> *`book_id` comes from the path. For backward compatibility you may accept `book_id` in the body, but the path value wins if they differ.*

Headers (optional):
`Idempotency-Key: <uuid>` to prevent duplicate submissions.

Response – 201

```
{
  "borrow_id": "BR1009",
  "user_id": "U1001",
  "book_id": "B42",
  "borrow_date": "2025-08-20",
  "return_date": "2025-09-03"
}
```

Business validations (error responses): - `409 CONFLICT` → `BOOK_UNAVAILABLE` if `book_availability=false`. - `409 CONFLICT` → `BORROW_LIMIT_REACHED` if exceeds

`borrow_book_lim`. - `422 UNPROCESSABLE ENTITY` → `INVALID_DATE_RANGE` if `borrow_date > return_date`. - `403 FORBIDDEN` if user not allowed (e.g., role or penalties).

6.2 All Borrow Records (borrowed + returned)

GET `books/all_borrow`

Response – 200 (list wrapper)

```
{
  "data": [
    {
      "borrow_id": "BR1009",
      "user_id": "U1001",
      "user_name": "Jamil Ahmed",
      "book_id": "B42",
      "borrow_date": "2025-08-20",
      "return_date": "2025-09-03",
      "borrow_status": "borrowed",
      "request_status": "pending"
    }
  ],
  "meta": { "total": 124, "page": 1, "page_size": 20 }
}
```

6.3 Update Request Status (admin)

PATCH `books/all_borrow/request{borrow_id}`
Body

```
{ "request_status": "accept" }
```

Response – 200 → updated Borrow Record.
Errors: `404 NOT_FOUND`, `422 VALIDATION_ERROR` (invalid enum), `403 FORBIDDEN`.

6.4 Update Borrow Status (admin)

PATCH `books/all_borrow/borrow{borrow_id}`
Body

```
{ "borrow_status": "returned" }
```

Response – 200 → updated Borrow Record.
Errors: `404 NOT_FOUND`, `422 VALIDATION_ERROR`, `403 FORBIDDEN`.

> *Note:* `borrow_status` transitions are usually constrained (e.g., `borrowed` → `returned` or auto `overdue` based on today vs `return_date`). On write, validate transitions; otherwise return `409 CONFLICT` → `INVALID_STATUS_TRANSITION`.

---

7) Admin Dashboards (“Donation Request Page” in your notes)

These endpoints provide **counts** and **lists** for borrow requests in different states. To keep your original feature set, each state has: - a **count** endpoint (returns only the number), and - a **list** endpoint (returns the detailed records).

> **Canonical** admin path below is `admin/books/...` exactly as you drafted, split into `…/count` (for numbers) and the base path for lists. If you prefer not to add `/count`, you can instead return a list wrapper with `meta.total`, but both are documented for compatibility.

7.1 Pending

- **GET** `admin/books/pending/count` →

  ```
  { "count": 12 }
  ```

- **GET** `admin/books/pending` → list of **Borrow Record** items (wrapper with `meta`).

7.2 Accepted

- **GET** `admin/books/accepted/count` → `{ "count": 99 }`
- **GET** `admin/books/accepted` → list of **Borrow Record** items.

7.3 Rejected

- **GET** `admin/books/rejected/count` → `{ "count": 5 }`
- **GET** `admin/books/rejected` → list of **Borrow Record** items.

7.4 Returned

- **GET** `admin/books/return/count` → `{ "count": 8 }`
- **GET** `admin/books/return` → list of **Borrow Record** items.

7.5 Overdue

- **GET** `admin/books/overdue/count` → `{ "count": 3 }`
- **GET** `admin/books/overdue` → list of **Borrow Record** items.
  > *Overdue = records whose `return_date` < today and `borrow_status` ≠ `returned`.*

7.6 Global Totals

- **GET** `books/all/count` → `{ "count": 340 }`
- **GET** `users/all/count` → `{ "count": 57 }`

If you prefer a single style, you can always return list wrappers with `meta.total` from the list endpoints and omit `/count`. Both patterns are documented so you **don’t lose any feature**.

---

8) Manage Books (admin)

8.1 List All Books (admin/full)

GET `books/all`
Response – 200

```
{
  "data": [
    {
      "book_id": "B42",
      "book_title": "Clean Code",
      "book_author": "Robert C. Martin",
      "book_category_id": "C2",
      "category_title": "Software Engineering",
      "book_rating": 4.8,
      "book_photo": "https://.../clean-code.jpg",
      "book_details": "...",
      "book_availability": true
    }
  ],
  "meta": { "total": 340, "page": 1, "page_size": 20 }
}
```

8.2 Create Book (admin)

POST `books/all`
Request

```
{
  "book_title": "Clean Code",
  "book_author": "Robert C. Martin",
  "book_category_id": "C2",
  "book_rating": 4.8,
  "book_photo": "https://.../clean-code.jpg",
  "book_details": "...",
  "book_availability": true
}
```

Response – 201 → Book (full) with server-assigned `book_id`.

Errors: `422 VALIDATION_ERROR` (missing fields), `404 NOT_FOUND` (category ID doesn't exist).

8.3 Update Book (partial or full; `book_id` immutable)

PATCH `books/all{book_id}`
Aliases: canonical is `PATCH books/{book_id}` (keep both).
Request – any subset of full Book fields except `book_id`.

Response – 200 → updated Book (full).
Errors: `404 NOT_FOUND`, `422 VALIDATION_ERROR`.

8.4 Delete Book (by `book_id`)

DELETE `books/all{book_id}`
Aliases: canonical `DELETE books/{book_id}` (keep both).
Response – 204 (no body).
Errors: `404 NOT_FOUND`, `409 CONFLICT` (e.g., trying to delete a book that is currently borrowed).

---

9) Categories (admin)

9.1 List Categories

GET `books/category/all`
Response – 200

```
{
  "data": [ { "category_id": "C2", "category_title": "Software Engineering" }
],
  "meta": { "total": 12, "page": 1, "page_size": 20 }
}
```

9.2 Update Category (partial)

PATCH `books/category/{category_id}`
Request

```
{ "category_title": "Software Engg." }
```

Response – 200 → Category.
Errors: `404 NOT_FOUND`, `422 VALIDATION_ERROR`.

9.3 Delete Category (guardrails)

DELETE `books/category/{category_id}`
Rule: Only allowed if **no available books** belong to this category.
Response – 204
Errors:
- `409 CONFLICT` → `CATEGORY_IN_USE` if any book with this `category_id` has `book_availability=true`.
- `404 NOT_FOUND` if unknown `category_id`.

---

10) Admin Settings

Single-row settings table: these endpoints get/set named limits. (The structure mirrors your draft: three distinct keys.)

10.1 Borrow Day Limit

- **GET** `admin/settings/borrow_day_lim` →

  ```
  { "borrow_day_lim": 14 }
  ```

- **POST** `admin/settings/borrow_day_lim`
  Request

  ```
  { "borrow_day_lim": 14 }
  ```

  Response – 200 same as GET.

10.2 Borrow Day Extension Limit

- **GET** `admin/settings/borrow_day_ext_lim` → `{ "borrow_day_ext_lim": 7 }`
- **POST** `admin/settings/borrow_day_ext_lim`
  Request/Response use the same key.

10.3 Borrow Book Limit (per user)

- **GET** `admin/settings/borrow_book_lim` → `{ "borrow_book_lim": 3 }`
- **POST** `admin/settings/borrow_book_lim`
  Request/Response use the same key.

Errors (all settings): `403 FORBIDDEN` (non-admin), `422 VALIDATION_ERROR` (non-numeric or negative), `500 INTERNAL_ERROR` (DB write failure).

---

11) Security & Authorization Summary

- **User endpoints** (`users/`, `books/`, `books/{id}`, `books/{id}/borrow`) require a valid token.

- **Admin endpoints** (`admin/**`, `books/all*`, `books/category/**`, `books/all_borrow/**`) require `role=admin`.

- Use `403 FORBIDDEN` for valid token but insufficient role; `401 UNAUTHORIZED` for missing/expired token.

---

12) Validation Rules (key highlights)

- **Borrow**

    - `borrow_date` ≤ `return_date` (inclusive) else `422 INVALID_DATE_RANGE`.
    - Respect `borrow_book_lim` per user; else `409 BORROW_LIMIT_REACHED`.
    - Deny if `book_availability=false` → `409 BOOK_UNAVAILABLE`.
- **Category delete**
    - Deny if any book in that category is currently available → `409 CATEGORY_IN_USE`.
- **Book update**
    - `book_id` immutable → `422 VALIDATION_ERROR` if provided and mismatched.

---

13) Field Enums

- `borrow_status ∈ { "returned", "overdue", "borrowed" }`
- `request_status ∈ { "accept", "pending", "reject" }`

Invalid values → `422 VALIDATION_ERROR` with `details.field` info.

---

14) Example Errors

14.1 Not Found

`GET books/B9999`

```
{
  "error": {
    "code": "NOT_FOUND",
    "message": "Book not found",
    "details": { "book_id": "B9999" }
  }
}
```

14.2 Validation Error

`PATCH books/allBR1009` (bad enum)

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid borrow_status",
    "details": { "borrow_status": "done" }
  }
}
```

14.3 Conflict

`DELETE books/allB42` (book currently borrowed)

```
{
  "error": {
    "code": "CONFLICT",
    "message": "Book cannot be deleted while borrowed",
    "details": { "book_id": "B42", "borrow_id": "BR1009" }
  }
}
```

---

15) Backward-Compatibility Notes

- **Paths**: You can keep original forms like `books/details<book_id>` and `books/all<book_id>` as **aliases** to the canonical `/books/{book_id}` routes.
- **Typos**: Accept `book_availibility` on input but always respond with `book_availability`.
- **Counts**: If you prefer to keep *pure count* endpoints as you drafted (e.g., `admin/books/pending` → number only), add `/count` as the canonical endpoint while still serving the legacy path. Alternatively, always return list wrappers with `meta.total`.

---

16) Quick cURL Samples

- **Login**

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{"user_id":"U1001","password":"secret"}' \
  http://localhost:8000/users/login
```

- **List Books (public)**

```
curl -H "Authorization: Bearer $TOKEN"
http://localhost:8000/books?page=1&page_size=12
```

- **Borrow a Book**

```
curl -X POST \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -H "Idempotency-Key: 7b8f6b0a-5db8-4f5a-8c9a-49b8f2b2b111" \
  -d
'{"user_id":"U1001","borrow_date":"2025-08-20","return_date":"2025-09-03"}' \
  http://localhost:8000/books/B42/borrow
```

- **Update Request Status (admin)**

```
curl -X PATCH \
  -H "Authorization: Bearer $ADMIN_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"request_status":"accept"}' \
  http://localhost:8000/books/all_borrow/requestBR1009
```

```sql
-- Users Table
CREATE TABLE users (
   user_id SERIAL PRIMARY KEY,
   user_name VARCHAR(100) NOT NULL,
   user_email VARCHAR(150) UNIQUE NOT NULL,
   user_photo TEXT,
   password VARCHAR(255) NOT NULL,
   role VARCHAR(50) NOT NULL DEFAULT 'student',
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Index for fast lookup by email & name
CREATE INDEX idx_users_email ON users(user_email);
CREATE INDEX idx_users_name ON users(user_name);

-- Categories Table
CREATE TABLE categories (
   category_id SERIAL PRIMARY KEY,
   category_title VARCHAR(100) NOT NULL
);

-- Index for category title search
CREATE INDEX idx_categories_title ON categories(category_title);

-- Books Table
CREATE TABLE books (
```

```
    book_id SERIAL PRIMARY KEY,
    book_title VARCHAR(200) NOT NULL,
    book_author VARCHAR(150) NOT NULL,
    book_category_id INT REFERENCES categories(category_id) ON DELETE CASCADE,
    book_rating DECIMAL(2,1) DEFAULT 0,
    book_photo TEXT,
    book_details TEXT,
    book_availability BOOLEAN DEFAULT TRUE,
    book_count INT DEFAULT 1,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Indexes for books
CREATE INDEX idx_books_title ON books(book_title);
CREATE INDEX idx_books_author ON books(book_author);
CREATE INDEX idx_books_category ON books(book_category_id);

-- Borrow Records Table
CREATE TABLE borrow_records (
    borrow_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id) ON DELETE CASCADE,
    book_id INT REFERENCES books(book_id) ON DELETE CASCADE,
    borrow_date DATE DEFAULT CURRENT_DATE,
    return_date DATE,
    borrow_status VARCHAR(50) DEFAULT 'borrowed',
    request_status VARCHAR(50) DEFAULT 'pending'
```

```
);


-- Indexes for borrow queries
CREATE INDEX idx_borrow_user ON borrow_records(user_id);
CREATE INDEX idx_borrow_book ON borrow_records(book_id);
CREATE INDEX idx_borrow_status ON borrow_records(borrow_status);
CREATE INDEX idx_borrow_request_status ON borrow_records(request_status);


-- Settings Table (Single Row Configuration)
CREATE TABLE settings (
   setting_id SERIAL PRIMARY KEY,
   borrow_day_limit INT NOT NULL,
   borrow_day_extension_limit INT NOT NULL,
   borrow_max_limit INT NOT NULL,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Library Management System – Database Schema

1. Users Table

```
Table: users
-------------------------------
user_id          | SERIAL PK
user_name        | VARCHAR(100) NOT NULL
user_email       | VARCHAR(150) UNIQUE NOT NULL
user_photo       | TEXT
password         | VARCHAR(255) NOT NULL
role             | VARCHAR(50) NOT NULL DEFAULT 'student'
created_at       | TIMESTAMP DEFAULT CURRENT_TIMESTAMP

Indexes:
- idx_users_email ON user_email
- idx_users_name ON user_name
```

---

2. Categories Table

```
Table: categories
-------------------------------
category_id       | SERIAL PK
category_title    | VARCHAR(100) NOT NULL
```

```
Indexes:
- idx_categories_title ON category_title
```

---

3. Books Table

```
Table: books
--------------------------------
book_id           | SERIAL PK
book_title        | VARCHAR(200) NOT NULL
book_author       | VARCHAR(150) NOT NULL
book_category_id  | INT FK → categories(category_id) ON DELETE CASCADE
book_rating       | DECIMAL(2,1) DEFAULT 0
book_photo        | TEXT
book_details      | TEXT
book_availability | BOOLEAN DEFAULT TRUE
book_count        | INT DEFAULT 1
created_at        | TIMESTAMP DEFAULT CURRENT_TIMESTAMP

Indexes:
- idx_books_title ON book_title
- idx_books_author ON book_author
- idx_books_category ON book_category_id
```

---

4. Borrow Records Table

```
Table: borrow_records
------------------------------
borrow_id        | SERIAL PK
user_id          | INT FK → users(user_id) ON DELETE CASCADE
book_id          | INT FK → books(book_id) ON DELETE CASCADE
borrow_date      | DATE DEFAULT CURRENT_DATE
return_date      | DATE NOT NULL
borrow_status    | VARCHAR(50) DEFAULT 'borrowed'   -- Enum: borrowed |
returned | overdue
request_status   | VARCHAR(50) DEFAULT 'pending'    -- Enum: pending |
accept | reject

Indexes:
- idx_borrow_user ON user_id
- idx_borrow_book ON book_id
- idx_borrow_status ON borrow_status
- idx_borrow_request_status ON request_status
```

---

5. Settings Table

```
Table: settings
------------------------------
setting_id                 | SERIAL PK
borrow_day_limit           | INT NOT NULL
borrow_day_extension_limit | INT NOT NULL
borrow_max_limit           | INT NOT NULL
```

```
updated_at                  | TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
library_backend/
├── app/
│   ├── __init__.py
│   ├── main.py
│   ├── config.py
│   ├── dependencies.py
│   ├── core/
│   │   ├── __init__.py
│   │   ├── security.py
│   │   └── exceptions.py
│   ├── models/
│   │   ├── __init__.py
│   │   ├── user.py
│   │   ├── book.py
│   │   ├── category.py
│   │   ├── borrow.py
│   │   └── settings.py
│   ├── schemas/
│   │   ├── __init__.py
│   │   ├── user.py
│   │   ├── book.py
│   │   ├── category.py
│   │   ├── borrow.py
│   │   └── settings.py
│   ├── crud/
│   │   ├── __init__.py
│   │   ├── user.py
```

```
│   │   ├── book.py
│   │   ├── category.py
│   │   ├── borrow.py
│   │   └── settings.py
│   ├── api/
│   │   ├── __init__.py
│   │   ├── auth.py
│   │   ├── users.py
│   │   ├── books.py
│   │   ├── categories.py
│   │   ├── borrow.py
│   │   └── admin.py
│   └── utils/
│       ├── __init__.py
│       ├── minio_utils.py
│       └── pagination.py
├── requirements.txt
└── README.md
```

sudo apt install uvicorn