Online Library — ER Diagram & API Blueprint

Author: Rohit (generated)
Purpose: Complete blueprint (ER diagram, table schemas, relationships, and API endpoints) for the Online Library project including production-ready features.

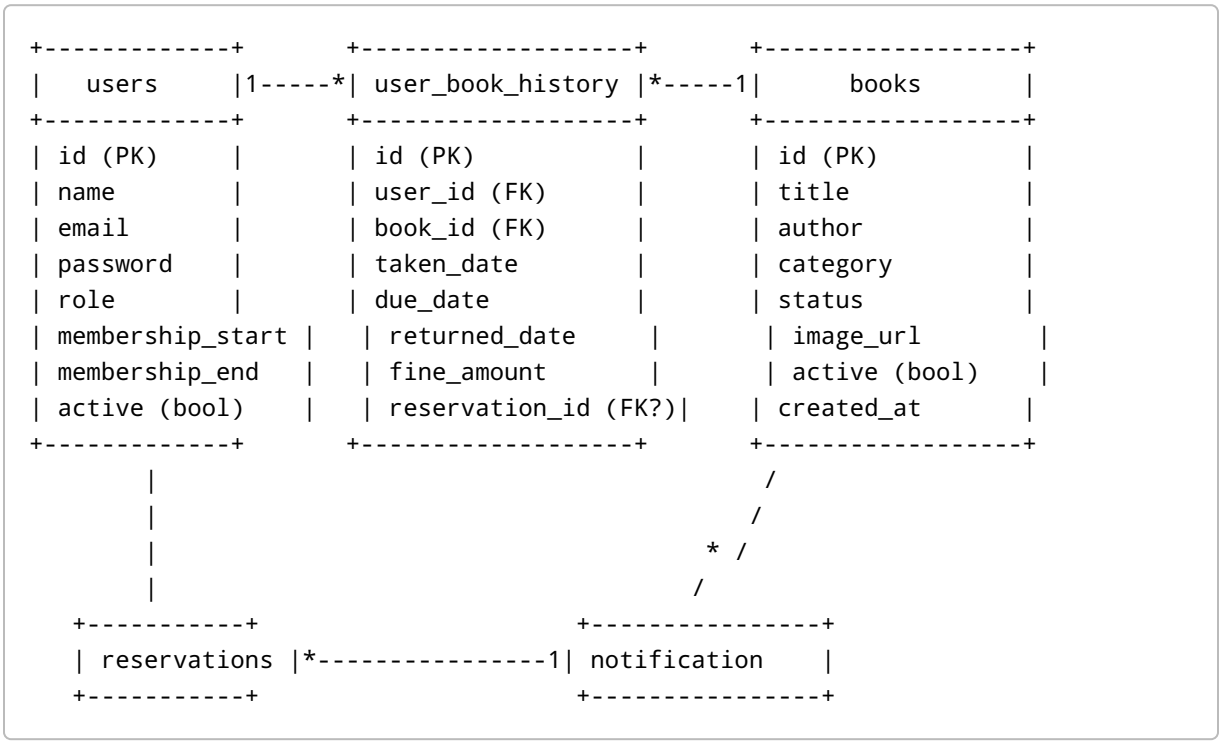
---

1. High-level overview

- Users can browse books, borrow/return them, view history and reports.
- Roles: `GUEST`, `MEMBER`, `LIBRARIAN` (admin).
- Core domains: Users, Books, Borrowing/History, Memberships, Reports, Reservations, Notifications.

---

2. ER Diagram (ASCII)



Notes: ASCII is simplified — see Entities section for full fields and constraints.

---

3. Entities / Table Schemas

users

- `id` BIGINT PK
- `name` VARCHAR NOT NULL
- `email` VARCHAR NOT NULL UNIQUE
- `password` VARCHAR NOT NULL (store hashed)
- `role` ENUM('GUEST','MEMBER','LIBRARIAN') default 'GUEST'
- `membership_start_date` DATE (nullable for guests)
- `membership_end_date` DATE (nullable for guests)
- `active` BOOLEAN default TRUE
- `created_at`, `updated_at`
- Index: `email` unique, index on `role`

books

- `id` BIGINT PK
- `title` VARCHAR NOT NULL
- `author` VARCHAR NOT NULL
- `category` VARCHAR NOT NULL
- `status` ENUM('AVAILABLE','TAKEN') default 'AVAILABLE'
- `image_url` VARCHAR (optional)
- `active` BOOLEAN default TRUE
- `created_at`, `updated_at`
- Index: `category`, `author`, `title` (for search)

user_book_history (borrows)

- `id` BIGINT PK
- `user_id` FK -> users(id)
- `book_id` FK -> books(id)
- `taken_date` DATE NOT NULL
- `due_date` DATE NOT NULL
- `returned_date` DATE NULL
- `fine_amount` DECIMAL(10,2) default 0
- `created_at`, `updated_at`
- Constraints: `returned_date` nullable; `due_date` > `taken_date`

reservations

- `id` BIGINT PK
- `user_id` FK -> users(id)
- `book_id` FK -> books(id)
- `reserved_at` TIMESTAMP
- `status` ENUM('PENDING','NOTIFIED','CANCELLED','FULFILLED')
- `expires_at` TIMESTAMP (optional)

notifications

- `id` BIGINT PK
- `user_id` FK -> users(id)
- `type` ENUM('MEMBERSHIP_EXPIRE','DUE_REMINDER','RESERVATION_AVAILABLE')
- `payload` JSON
- `sent` BOOLEAN default FALSE
- `created_at`, `sent_at`

audit_logs (optional, useful for assignment)

- `id`, `action`, `user_id`, `entity_type`, `entity_id`, `timestamp`, `metadata` JSON

---

4. Relationships

- `users` 1 — * `user_book_history` (a user can have many borrow records)
- `books` 1 — * `user_book_history` (a book can be borrowed many times historically)
- `users` 1 — * `reservations` (a user can reserve many books)
- `books` 1 — * `reservations` (a book can have many reservations)
- `users` 1 — * `notifications`

---

5. API Endpoints

General notes - All endpoints return `ApiResponse<T>`. - Use JWT auth for protected endpoints.
- Role-based access: annotate endpoints with `@PreAuthorize` where needed. - Pagination & sorting via `page`, `size`, `sort` params for list endpoints.

Auth / User

- `POST /auth/register` — register (name, email, password, membershipMonths optional)
- `POST /auth/login` — returns JWT token
- `GET /users` — (Librarian only) list users (filter by active/role).
- `GET /users/{id}` — get user detail and borrow history (auth required, librarian or same user)
- `PUT /users/{id}` — update user (librarian or user)
- `PUT /users/{id}/membership` — extend membership (months)

Books

- `POST /books` — add book (librarian) (multipart/form-data to include image)
- `GET /books` — list books with filters: `?category=&author=&name=&status=&page=&size=&sort=`
- `GET /books/{id}` — get book detail (including whether currently borrowed)
- `PUT /books/{id}` — update book (librarian)
- `PUT /books/{id}/status` — update status (librarian or system)

- `DELETE /books/{id}` — soft-delete book (librarian)

Borrow / Return

- `POST /users/{userId}/borrow/{bookId}?days=7` — borrow a book (checks membership and availability)
- Request validated: days positive, membership active
- `POST /users/{userId}/return/{borrowId}` — return by borrow record id
- `GET /users/{userId}/history` — borrow history (paginated)

Reservations

- `POST /books/{bookId}/reserve` — reserve a book (member only)
- `GET /users/{userId}/reservations` — list user reservations
- System job: when a book becomes available, set reservation `NOTIFIED` and create notification

Reports (Admin / Librarian)

- `GET /reports/top-category` — percentage per category
- `GET /reports/daily` — daily borrows
- `GET /reports/monthly` — monthly borrows
- `GET /reports/yearly` — yearly borrows
- `GET /reports/top-books` — top borrowed books
- `GET /reports/top-users` — top users
- `GET /reports/currently-borrowed` — all active borrows
- `GET /reports/overdue` — overdue borrows
- `GET /reports/user/{userId}/total-borrows` — count
- `GET /reports/book-trends` — book borrow trends
- `GET /reports/category-range?start=YYYY-MM-DD&end=YYYY-MM-DD` — range

Notifications & Scheduler

- Scheduler job: daily tasks
- Send membership expiry reminders (7 days before expiry)
- Send due-date reminders (1 day before due)
- Clean expired reservations
- `GET /notifications` — user notifications (paginated)
- `POST /notifications/{id}/ack` — mark as read/acknowledged

---

6. Request / Response examples (borrow)

Request

```
POST /users/42/borrow/123?days=14
Authorization: Bearer <jwt>
```

Success Response

```
{
  "success": true,
  "message": "Book borrowed successfully",
  "data": {
    "borrowId": 987,
    "dueDate": "2025-09-15"
  },
  "timestamp": "2025-09-01T10:00:00"
}
```

Error (membership expired)

```
{
  "success": false,
  "message": "Membership expired, please renew",
  "timestamp": "2025-09-01T10:01:00"
}
```

---

7. Validation & Errors

- Use `@Valid` DTOs, JSR-380 annotations on DTOs.
- Throw custom exceptions: `ResourceNotFoundException`, `BadRequestException`, `NoDataFoundException`, `UnauthorizedException`.
- `GlobalExceptionHandler` returns `ApiResponse.error(...)` with appropriate HTTP code.

---

8. Non-functional considerations

- **Caching**: use Spring Cache (Redis) for `books` lists and `reports`.
- **Security**: Spring Security + JWT, password hashing with BCrypt.
- **File storage**: store book images on disk under `/uploads` or S3 for production.
- **Database**: use proper indexes, and run migrations with Flyway/Liquibase.
- **Testing**: unit tests for services, integration tests for controllers (MockMvc), and e2e tests.
- **Observability**: structured logging, metrics (Prometheus), and tracing (optional).

---

9. Scheduler & Background jobs

- `MembershipScheduler` (daily): mark expired or send reminders.

- `NotificationScheduler` (hourly/daily): process reservations, send notifications.
- Make notification sending async (`@Async`) or push to message queue.

---

10. Deployment & Dev checklist

- Dockerize services, run locally with docker-compose (Postgres + Redis).
- Use environment variables for secrets.
- Health endpoints (`/actuator/health`).
- CI pipeline: lint, build, tests.

If you want, I can now: - Produce a **visual ER diagram** (PNG/SVG) you can download, or - Generate **Postman collection** / OpenAPI (YAML) for all endpoints, or - Create detailed DTO classes and controller method signatures for copy-paste.

Which one should I generate next?