📓SPRING BOOT API DESIGN REFERENCE (STRUCTURE + FIELD-LEVEL SPEC)

🔗CustomerController

`POST /api/customers` — Register a new customer *(Level: ⚫Basic)*

Request:

- `name`: String
- `email`: String
- `joinedDate`: LocalDate (optional)

Success Response:

- `id`: Long
- `name`: String
- `email`: String
- `joinedDate`: LocalDate
- `active`: Boolean

Failures:

- 400 `BAD_REQUEST`: Missing name/email
- 409 `CONFLICT`: Email already exists

`GET /api/customers/{id}` — Fetch customer details *(⚫Basic)*

Path Param: `id`: Long

Response:

- Full Customer DTO (see above)

Failures:

- 404 `NOT_FOUND`: Customer does not exist

`PUT /api/customers/{id}` — Update customer info *(⚫Basic)*

Path Param: `id`: Long

Request:

- `name`: String (optional)
- `email`: String (optional)

Response:

- Updated Customer DTO

Failures:

- 404 `NOT_FOUND`: Customer not found

`GET /api/customers/{id}/books` — List currently issued books (⚫*Moderate)*

Path Param: `id`: Long

Response: List of:

- `issueId`, `bookId`, `title`, `issueDate`, `dueDate`, `status`

`GET /api/customers/{id}/history` — Complete borrowing history (⚫*Moderate)*

Response: List of all `issueId`, `status`, `returnDate`, `fine`

`DELETE /api/customers/{id}` — Delete customer (⚫*Moderate)*

Constraint: Only if no active books

Failures:

- 403 `FORBIDDEN`: Active books found

`GET /api/customers/active` — Customers with active issues (⚫*Basic)*

Response: List of:

- `id`, `name`, `email`, `totalBooksIssued`

`GET /api/customers/defaulters` — Customers with pending fines (⚫*Moderate)*

Response: List of:

- `id`, `name`, `pendingFine`, `lastPaymentDate`

`POST /api/customers/{id}/deactivate` — Deactivate customer (⚫*Moderate)*

Response:

- `message`: String
- `active`: false

📚BookController

`GET /api/books` — Get all books with availability (⚫*Basic)*

Query (optional):

- `available`: Boolean (default: true)

Response: List of BookDTO:

- `id`, `title`, `author`, `isbn`, `availableCopies`, `totalCopies`, `averageRating`

`GET /api/books/{id}` — Book details (⚫*Basic)*

Path Param: `id`: Long

`POST /api/books` — Add a book (⚫*Basic)*

Fields:

- `title`, `author`, `isbn`, `totalCopies`

`PATCH /api/books/{id}/adjust-stock?delta=-2` (⚫*Moderate)*

Path: `id`: Long **Query:** `delta`: int

Failures:

- 400 `BAD_REQUEST`: Reducing below issued count

`GET /api/books/out-of-stock` (⚫*Basic)*

Response: Books with `availableCopies` == 0

`GET /api/books/recommended` ***(⚪ Advanced)***

Response: Sorted by `averageRating`, personalized (optional)

`GET /api/books/search?title=java&author=gosling&available=true` ***(⚫ Moderate)***

Query Params:

- `title`, `author`, `available`

Response: Matching BookDTOs

`GET /api/books/filters?minRating=4.5&copies>5` ***(⚪ Advanced)***

Query Params:

- `minRating`: double
- `copies`: int (min stock)

🔦IssueController

`POST /api/issues` **— Issue book *(⚫ Moderate)***

Request:

- `bookId`, `customerId`, `issueDate`, `dueDate`

Failures:

- 409 `CONFLICT`: No available copies
- 403 `FORBIDDEN`: Customer inactive or has dues

`GET /api/issues/active` ***(⚫ Moderate)***

Response: Active issues list

`GET /api/issues/overdue` ***(⚫ Moderate)***

Response: List of `issueId`, `bookId`, `customerId`, `dueDate`, `daysOverdue`

`POST /api/issues/batch` ***(⚪Advanced)***

Request:

- `customerId`, `bookIds[]` **Query Param (optional):** `extendDays`

Response:

- `issued[]`, `skipped[]`

🕐ReturnController

`PUT /api/returns/{issueId}` ***(⚫Moderate)***

Path Param: `issueId`

Response:

- `fineApplied`, `returnDate`, `isLate`, `status`

`POST /api/returns/bulk` ***(⚪Advanced)***

Body:

- `issueIds[]`

🕖ReviewController

`POST /api/books/{bookId}/review` ***(⚪Basic)***

Body:

- `rating`: 1-5, `comment`, `customerId`

Response:

- `reviewId`, `bookId`, `rating`, `comment`, `postedAt`

`GET /api/reviews/top` ***(⚫Moderate)***

Response: Top books with averageRating > 4.5

💳PaymentController

`POST /api/payments` *(**⚫Basic)***

Request:

- `customerId`, `amount`, `paymentDate`, `mode`

Response:

- `paymentId`, `status`, `note`

`GET /api/payments/monthly?month=07&year=2025` ***(⚫Moderate)***

Response:

- List of payments made that month

📊ReportController

`GET /api/reports/daily-summary` ***(⚫Moderate)***

Response:

- `issuedToday`, `returnedToday`, `finesCollected`

`GET /api/reports/defaulter-list` ***(⚪Advanced)***

Response:

- `customerId`, `name`, `pendingFine`, `booksOverdue`

🔍SearchController

`GET /api/search/books?query=java` ***(⚫Basic)***

Response: BookDTOs matching `title`, `author`, or `isbn`

`GET /api/search/issues?bookId=10&customerId=5` ***(⚫Moderate)***

Response: Issues for that customer/book

`GET /api/search/reviews?rating=5&sort=latest` ***(● Moderate)***

Query Params:

- `rating`, `sort`

🛠️SystemController

`GET /api/system/status` ***(● Basic)***

Response:

- `status`: UP/DOWN, `timestamp`, `message`

`POST /api/system/seed` ***(● Moderate)***

Response:

- `message`, `entitiesCreated`

Error Structure (Standard)

```
{
  "code": "BAD_REQUEST",
  "message": "Customer is inactive",
  "timestamp": "2025-07-13T18:40:12"
}
```

Levels:

- ●Basic = CRUD, simple retrieval
- ●Moderate = Includes logic, validation, query filtering
- ●Advanced = Aggregates, batch ops, analytics, personalization