Digital Library System API Documentation with Database Fields & DTOs

1. Authentication & User Management

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
Register a new user	POST	/api/users/register	Create a new user account	id, username, email, password_hash, role, created_at, updated_at	{ username, email, password, role }
Login	POST	/api/users/login	Authenticate user and return JWT/token	email, password_hash	{ email, password }
Logout	POST	/api/users/logout	Invalidate user session/token	user_id, token, expired_at	{ userId, token }
Get user profile	GET	/api/users/{userId}	Retrieve user details	id, username, email, role, created_at, updated_at	{ id, username, email, role }
Update user profile	PUT	/api/users/{userId}	Update user information	username, email, role, updated_at	{ username, email, role }
Change password	PUT	/api/users/{userId}/password	Update user password	password_hash, updated_at	{ oldPassword, newPassword }
Delete user	DELETE	/api/users/{userId}	Remove a user from the system	id	{ id }
List all users	GET	/api/users	Retrieve a list of users (admin only)	id, username, email, role, created_at, updated_at	[{ id, username, email, role }]
Assign role	POST	/api/users/{userId}/role	Assign or update user role (admin only)	role, updated_at	{ role }

2. Digital Book Management

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
Add a new book	POST	/api/books	Create a new book entry	id, title, author, category_id, description, file_path, created_at, updated_at	{ title, author, categoryId, description, file }
Get book details	GET	/api/books/{bookId}	Retrieve information about a specific book	id, title, author, category_id, description, file_path, created_at, updated_at	{ id, title, author, categoryId, description, filePath }
Update book	PUT	/api/books/{bookId}	Modify book metadata or content	title, author, category_id, description, file_path, updated_at	{ title, author, categoryId, description, file }
Delete book	DELETE	/api/books/{bookId}	Remove a book from the library	id	{ id }
List all books	GET	/api/books	Get all books, with optional filters	id, title, author, category_id, description, created_at, updated_at	[{ id, title, author, categoryId, description }]
Search books	GET	/api/books/search?query=keyword	Search books by title, author, or category	id, title, author, category_id, description	[{ id, title, author, categoryId, description }]
Upload book file	POST	/api/books/{bookId}/upload	Upload PDF, EPUB, or other formats	file_path, updated_at	{ file }
Download book file	GET	/api/books/{bookId}/download	Download the book file	file_path	{ filePath }

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
Add book category	POST	/api/categories	Create a new category	id, name, description, created_at, updated_at	{ name, description }
List categories	GET	/api/categories	Get all book categories	id, name, description	[{ id, name, description }]

3. Access & Reading Management

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
Borrow a book	POST	/api/access/{bookId}/borrow	User borrows a book	id, user_id, book_id, borrowed_at, due_date, returned_at	{ userId, bookId }
Return a book	POST	/api/access/{bookId}/return	User returns a borrowed book	returned_at	{ userId, bookId }
Check access	GET	/api/access/{bookId}/status	Check if user can access a book	user_id, book_id, borrowed_at, returned_at	{ userId, bookId }
Start reading	POST	/api/reading/{bookId}/start	Begin reading a book (track session)	id, user_id, book_id, start_time, end_time	{ userId, bookId }
End reading	POST	/api/reading/{bookId}/end	Finish reading session	end_time	{ userId, bookId }
Get reading history	GET	/api/reading/history	List all books read by the user	user_id, book_id, start_time, end_time	[{ bookId, startTime, endTime }]
Bookmark page	POST	/api/reading/{bookId}/bookmark	Save current page/location	id, user_id, book_id, page_number, created_at	{ userId, bookId, pageNumber }

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
List bookmarks	GET	/api/reading/{bookId}/bookmarks	Retrieve bookmarks for a book	user_id, book_id, page_number, created_at	[{ pageNumber, createdAt }]
Highlight text	POST	/api/reading/{bookId}/highlight	Add a highlight to a section	id, user_id, book_id, text, page_number, created_at	{ userId, bookId, text, pageNumber }
Get highlights	GET	/api/reading/{bookId}/highlights	Retrieve all highlights for a book	user_id, book_id, text, page_number, created_at	[{ text, pageNumber, createdAt }]

4. Reporting & Logs

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
Get user activity log	GET	/api/logs/users/{userId}	Fetch all actions performed by a user	id, user_id, action, timestamp, details	[{ action, timestamp, details }]
Get book access log	GET	/api/logs/books/{bookId}	Track who accessed a specific book	id, book_id, user_id, action, timestamp	[{ userId, action, timestamp }]
Generate report	GET	/api/reports/{reportType}	Generate report	report_id, report_type, generated_at, data	{ reportType }
Download report	GET	/api/reports/{reportType}/download	Download report as CSV or PDF	report_id, report_type, generated_at, file_path	{ reportType, filePath }
View system logs	GET	/api/logs/system	View backend/system activity	id, level, message, timestamp	[{ level, message, timestamp }]

Action	HTTP Method	Endpoint	Description	Database Fields	DTO
Filter logs	GET	/api/logs?type=activity&date=2026-01-25	Filter logs by type or date range	id, type, message, timestamp	{ type, date }
Delete logs	DELETE	/api/logs/{logId}	Remove specific logs (admin only)	id	{ logId }