# Library API Documentation

## Overview

This API is developed using the [Gin](https://gin-gonic.com/) web framework in Go and serves as the backend for a library management system. It offers endpoints to manage users, books, events, reviews, and authentication. It supports standard HTTP methods including GET, POST, PUT, and DELETE.

## Table of Contents

1. Introduction
2. User Management
3. Book Management
4. Review System
5. Event Handling
6. Authentication
7. Additional Features
8. Error Handling and Response Format

## 1. Introduction

### Root Endpoint

* ANY /
  + Responds with HTTP 200 OK and an empty JSON object.

## 2. User Management

### Get All Users

* GET /users
  + Retrieves a list of all users.

### Get User by ID

* POST /user
  + Request Body: { "token": "<token>", "id": <user\_id> }
  + Returns user details.

### Get Current Profile

* POST /profile
  + Returns the currently authenticated user's profile.

### Edit Profile

* POST /edit
  + Updates the user's profile.

### Delete Account

* DELETE /user
  + Deletes the authenticated user's account.

### User History

* POST /user/history
  + Returns borrowing and activity history of a user. Used by a librarian to get any user’s history.
* POST /history
  + Returns borrowing and activity history of a user.

## 3. Book Management

### Get All Books

* GET /books
  + Returns a list of all books.

### Get Book by ID

* GET /book
  + Query parameter: id=<book\_id>

### Add Book

* POST /book
  + Adds a new book to the system.

### Remove Book

* POST /book/remove
  + Removes a book from the system.

### Update Book Details

* POST /book/quantity
* POST /book/update/id
  + Updates the book quantity and ID respectively.

### Search Books

* POST /searchbook
  + Searches for books based on title.

### Book Borrow/Return

* POST /book/borrow
* POST /book/return
  + Handles borrowing and returning books.

### Book Reservation

* POST /book/reserve
* POST /book/cancel/reservation
  + Handles reserving and canceling book reservations.

### Book Availability

* GET /book/availability
  + Checks if a book is available.

### Get Overdue Books

* POST /book/overdue
  + Returns a list of overdue books.

## 4. Review System

### Leave/Edit/Delete Review

* POST /review - Add a review.
* PUT /review - Edit a review.
* DELETE /review - Delete a review.

### Retrieve Reviews

* POST /book/review - Get reviews for a specific book.
* POST /review/user - Get reviews made by a user.
* POST /book/review/high - Highest-rated reviews.
* POST /book/review/low - Lowest-rated reviews.

### Rating Information

* POST /book/rating - Average rating for a book.
* POST /book/rating/details, POST /book/rating/details/sql - Detailed rating info.

### Voting

* POST /review/vote - Vote for a review.
* POST /review/votes, POST /review/votes/sql - Get votes for a review.

## 5. Event Handling

### Get Events

* GET /events - Retrieves all events.
* GET /event/upcoming - Retrieves upcoming events.

### Create/Invite to Event

* POST /event - Create a new event.
* POST /event/invite - Invite users to an event.

### Invited Events

* POST /event/invited - List of users invited to this event.

## 6. Authentication

### Sign Up

* POST /signup
  + Registers a new user.

### Log In

* POST /login
  + Authenticates the user and returns session/token data.

## 7. Additional Features

### Get Librarians

* GET /librarians
  + Retrieves a list of all librarians.

### Get Authors

* GET /authors
  + Returns a list of all book authors.

## 8. Error Handling and Response Format

All endpoints return JSON responses. On success, HTTP status 200 is returned with the corresponding data. On failure, appropriate status codes (e.g., 400, 401, 404, 500) are returned with an error message in the format:

{

"error": "Description of the error"

}

## Conclusion

This API is structured to support a full-featured library system. With endpoints grouped by responsibility (users, books, reviews, events), it supports RESTful best practices while offering extended functionality like review voting and event invitations.