Project Idea: Advanced Library Management System

Build a library management system to manage libraries, books, authors, and categories. Implement user registration, login, and password recovery. Allow users to borrow and return multiple books in one transaction, with notifications and real-time updates for book availability.

Must-Have Endpoints:

- **Library Management**:
  - List libraries
  - Filter libraries by book categories, authors
  - Calculate distances between users and nearby libraries
- **Authors**:
  - List authors with book counts
  - Filter by library and book category (book counts should update with filters)
- **Books**:
  - List books
  - Filter by category, library, and author
  - Return author and category names
- **Loaded Authors Endpoint**:
  - List authors with all their books objects
  - Each book should include its category object
  - Filter by category and library

Task Details:

1. Notifications:

- **Email Notifications**:
  - Send confirmation emails upon borrowing. Test locally with Mailhog, no actual email service required (AWS SES, Sendmail etc.)
  - Send daily reminders in the last 3 days of the borrowing period

2. Business Logic:

- **Borrowing Rules**:
  - Allow up to 3 books; return one to borrow a 4th
  - Users must specify a return date (max 1 month); late returns incur a daily penalty
- **Penalty Calculation**:
  - Calculate penalties based on overdue days

3. Nice to Have:

- **Deploy**: Deploy the project on free aws tier
- **Library Branches**: Manage multiple branches with unique locations
- **Caching**: Implement caching for frequently accessed data
- **Rate Limiting**: Apply rate limiting to API endpoints
- **Internationalization**: Support multiple languages
- **Permissions**: Implement user roles and permissions
- **Real-time Notifications**:
  - Use WebSockets (Django Channels) to notify when a book is returned and available
- **Dockerization**: Containerize the project using Docker
- **Task Queue:** Use Celery (or an alternative) for email sending