ITE 18 – Application Development and Emerging Technologies

Web/Mobile Application – Final Project Submission

Student Name(s): Michael D. Tiposo
Project Title: BookKeeper - Library Management System
Date of Submission: December 22-26, 2025
Course/Section: ITE18-EJ1

1. Project Overview

1.1 Project Description

BookKeeper is a comprehensive **full-stack web application** designed for library management, featuring a complete borrowing system with digital book reading capabilities. The system consists of a **React-based frontend UI** and a **Laravel backend API**. The application solves the problem of manual library operations by providing a centralized platform where administrators can manage books, borrowers, and transactions, while members can browse, borrow, and read books digitally.

1.2 Target Users

• Library Administrators: Manage books, borrowers, categories, suppliers, and oversee borrowing requests

• Library Members: Browse books, submit borrowing requests, read digital content, make payments

• Library Staff: Process requests, manage inventory, handle payments

1.3 Key Features

• User authentication and role-based access (Admin, Member)

• Complete book inventory management with categories and suppliers

• Digital book reading with pagination

• Borrowing system with 10-day periods and 3-book limits

• Payment processing for fines with multiple payment methods

• Request approval workflow

• Real-time notifications system

• Responsive React-based user interface

• RESTful API integration between frontend and backend

1.4 Technology Stack

• **Frontend**: React.js, TypeScript, HTML5, CSS3, Tailwind CSS

• **Backend**: Laravel (PHP 8.2+)

• **Database**: MySQL

• **API: RESTful** JSON APIs with Laravel Sanctum

• **Tools**: Git, Composer, Node.js, npm, Laravel Artisan, Postman

2. App Plan

2.1 Project Scope

In Scope:

- Full-stack web application (React frontend + Laravel backend)
- User authentication and authorization with role-based access
- Complete book management system
- Digital borrowing and reading system
- Payment processing for library fines
- Admin and member dashboards
- Real-time notifications

Out of Scope:

- Native mobile applications
- Physical book tracking (RFID/barcodes)
- Advanced reporting and analytics
- Multi-branch library support

2.2 Objectives & Goals

• Develop a user-friendly React frontend with modern UI/UX

• Build a secure and scalable Laravel API backend

• Implement complete library management workflows

• Provide digital book reading experience

• Ensure responsive design for all devices

2.3 User Stories & Use Cases

User Story 1:

As a library member, I want to browse available books so that I can discover new reading materials.

Acceptance Criteria:

[✔] Books are displayed in grid/list view with covers

[✔] Search and filter functionality works

[✔] Book details show availability status

User Story 2:

As an administrator, I want to manage the book inventory so that the catalog remains current.

Acceptance Criteria:

[✔] Admin dashboard allows book creation and updates

[✔] Category and supplier management included

[✔] Book-author relationships maintained

User Story 3:

As a library member, I want to borrow books digitally so that I can read them online.

Acceptance Criteria:

[✔] Members can request book loans

[✔] Admin approval workflow implemented

[✔] Digital reading available for approved loans

2.4 System Architecture

The system follows a **client-server architecture** with modern web technologies. The React frontend handles user interaction, state management, and API communication. The Laravel backend processes requests, manages business logic, interacts with the MySQL database, and returns JSON responses. Authentication is handled via Laravel Sanctum tokens, and the system supports role-based access control.

2.5 Development Timeline

Phase	Description	Timeline
Phase 1	Planning & UI Design	Week 1
Phase 2	Backend Setup & Database	Week 2
Phase 3	API Development	Week 3
Phase 4	Frontend Development	Week 4-5
Phase 5	Integration & Testing	Week 6

3. UI/UX Design

3.1 Design Philosophy

The UI follows a modern, clean, and professional design approach suitable for educational institutions. Emphasis is placed on usability, accessibility, and efficient workflows. The design uses a dual-portal system with distinct interfaces for administrators and members.

3.2 Color Scheme

- **Primary Color**: Blue (#2563eb) – Used for primary actions and admin interface
- **Secondary Color**: Green (#16a34a) – Used for success states and member interface
- **Accent Color**: Orange (#ea580c) – Used for warnings and highlights
- **Neutral Colors**: Gray scale for backgrounds and text

3.3 Typography

- **Font Family**: Inter (sans-serif)
- **Heading Sizes**: 24px–32px for main headings
- **Body Text**: 14px–16px for content
- **Small Text**: 12px for metadata

3.4 UI Components

- **Navigation**: Sidebar navigation for both admin and member portals

• **Data Tables**: Sortable, filterable tables for admin management

• **Cards**: Book cards with cover images and metadata

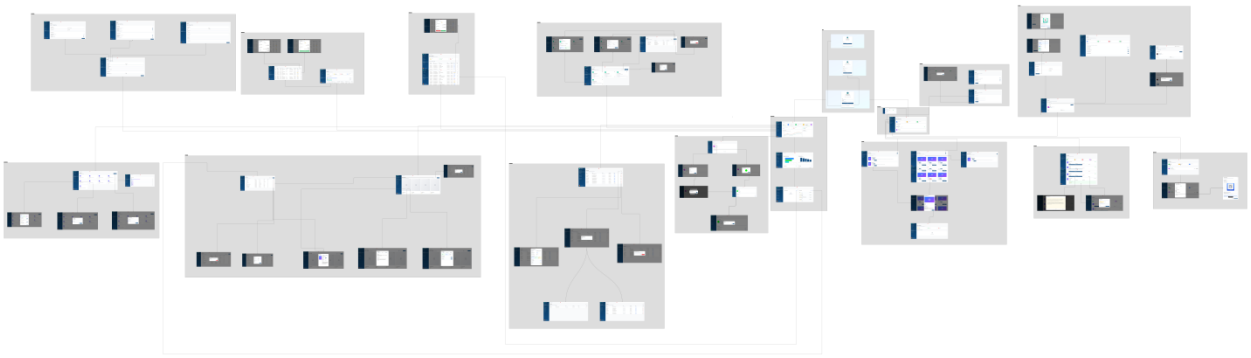
• **Forms**: Validated forms for data entry

• **Modals**: Dialogs for confirmations and detailed views

• **Notifications**: Toast notifications and dropdown

3.5 Wireframes & Mockups

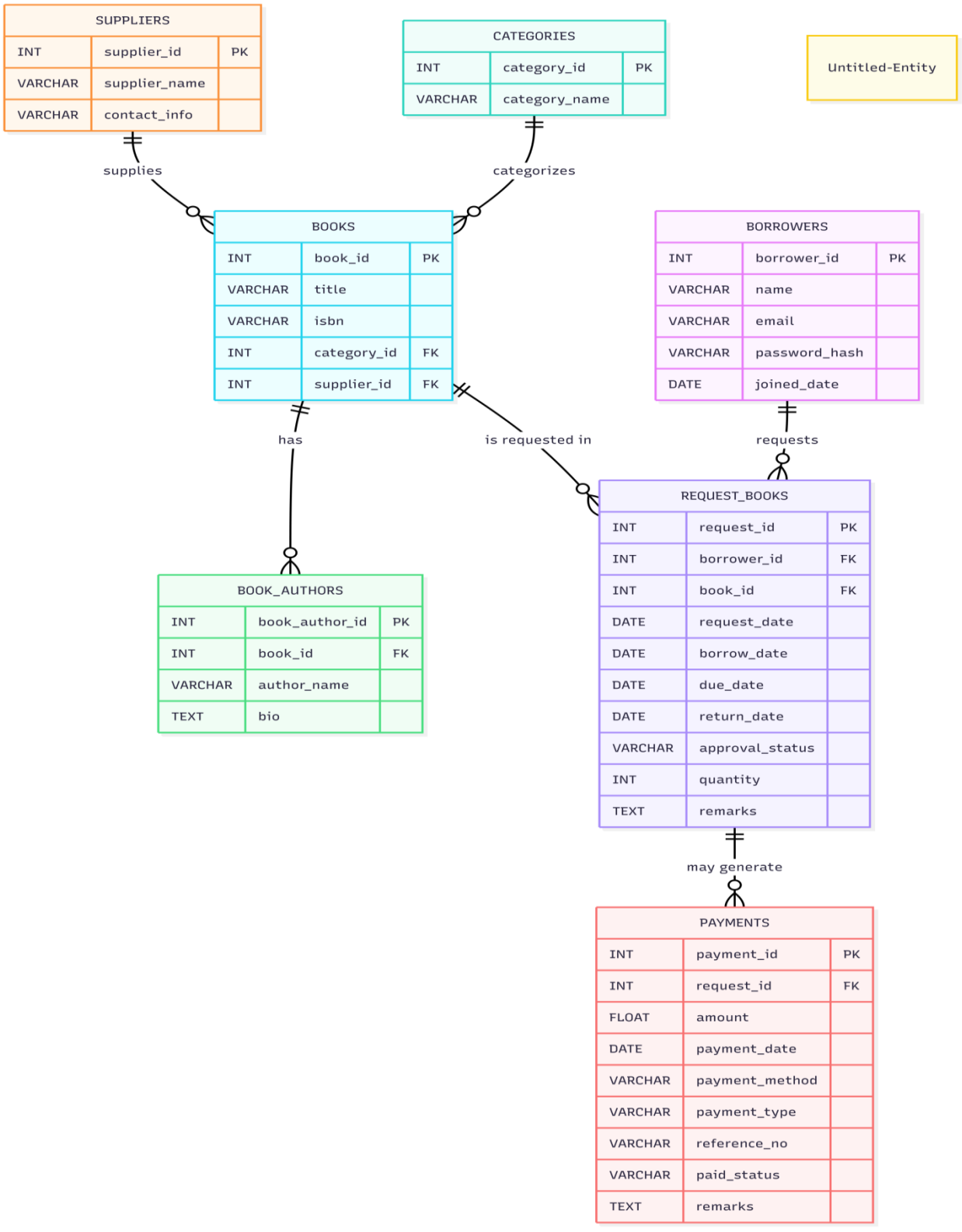
https://www.figma.com/board/L9kMMwhcl0uvv9jPpFC236/Library-Management-System?node-id=0-1&t=YnmVZvRH80pqvgvU-1



4. Database Architecture (ERD)

4.1 Entity Relationship Diagram

The database structure supports a complete library management system with the following core entities interconnected to manage books, borrowers, and transactions.


SUPPLIERS
INT supplier_id PK
VARCHAR supplier_name
VARCHAR contact_info
CATEGORIES
INT category_id PK
VARCHAR category_name
Untitled-Entity
supplies
categorizes
BOOKS
INT book_id PK
VARCHAR title
VARCHAR isbn
INT category_id FK
INT supplier_id FK
BORROWERS
INT borrower_id PK
VARCHAR name
VARCHAR email
VARCHAR password_hash
DATE joined_date
has
is requested in
requests
REQUEST_BOOKS
INT request_id PK
INT borrower_id FK
INT book_id FK
DATE request_date
DATE borrow_date
DATE due_date
DATE return_date
VARCHAR approval_status
INT quantity
TEXT remarks
BOOK_AUTHORS
INT book_author_id PK
INT book_id FK
VARCHAR author_name
TEXT bio
may generate
PAYMENTS
INT payment_id PK
INT request_id FK
FLOAT amount
DATE payment_date
VARCHAR payment_method
VARCHAR payment_type
VARCHAR reference_no
VARCHAR paid_status
TEXT remarks

4.2 Entity Descriptions

Borrowers

- Stores member information, authentication credentials, and role assignments
- Members can request books, make payments, and access digital content
- Supports both admin and member roles

Books

- Stores book information including title, ISBN, and relationships
- Central entity for the library catalog
- Links to categories, suppliers, and authors

Categories

- Defines book categories for organization
- Supports hierarchical classification

Suppliers

- Stores information about book suppliers/vendors
- Tracks procurement relationships

Book Authors

- Junction entity linking books to their authors
- Supports multiple authors per book

Request Books

- Core borrowing entity managing the request-to-return workflow
- Tracks request dates, approval status, borrow/return dates
- Calculates due dates and fines automatically

Payments

- Handles financial transactions including fines and fees

- Supports multiple payment methods with proof upload
- Tracks payment status and verification

4.3 Relationships

- **Borrowers** → Request Books: One-to-Many (member can have multiple requests)
- **Books** → Request Books: One-to-Many (book can be requested multiple times)
- **Categories** → Books: One-to-Many (category can contain multiple books)
- **Suppliers** → Books: One-to-Many (supplier can provide multiple books)
- **Books** → Book Authors: One-to-Many (book can have multiple authors)
- **Borrowers** → Payments: One-to-Many (member can make multiple payments)

4.4 Database Normalization

The database follows Third Normal Form (3NF) with proper foreign key relationships and constraints to ensure data integrity.

5. Application Features & Functionality

5.1 Authentication & User Management

Description: Secure user system with role-based access control

Functionality:
- User registration and login with email/password
- Laravel Sanctum token-based authentication
- Role-based access (Admin/Member)
- Password hashing and validation
- Session management with token refresh
- Profile management

5.2 Book Inventory Management (Admin)

Description: Complete book catalog management system

Functionality:
- CRUD operations for books
- Category and supplier management
- Author assignment to books

• Book cover image upload and display
• ISBN validation and uniqueness
• Inventory tracking

5.3 Borrowing System

Description: Complete digital borrowing workflow

Functionality:
• Book request submission by members
• Admin approval/rejection workflow
• Automatic due date calculation (10 days)
• Borrowing limits (3 books per member)
• Digital book access for approved loans
• Return processing with rating requirement

5.4 Digital Book Reading

Description: In-browser book reading experience

Functionality:
• Paginated book content display
• Navigation between pages
• Reading progress tracking
• Access restricted to active borrows
• Responsive reading interface

5.5 Payment Processing

Description: Fine and fee payment system

Functionality:
• Automatic fine calculation (₱5/day late)
• Multiple payment methods (GCash, Maya, GoTyme)
• Payment proof upload
• Admin payment verification
• Payment status tracking

5.6 Request Management (Admin)

Description: Borrowing request processing

Functionality:
• View all pending requests
• Approve/reject requests
• Automatic borrow date assignment
• Activity logging

5.7 Notifications System

Description: Real-time user notifications

Functionality:
• Request status updates
• Payment reminders
• Due date alerts
• System announcements
• Notification management (read/unread)

5.8 Dashboard Analytics (Admin)

Description: Library performance metrics

Functionality:
• Total books, borrowers, active borrows
• Revenue tracking
• Popular books analytics
• System health monitoring

6. Security & Error Handling

6.1 Security Measures

• Laravel Sanctum for API authentication
• Password hashing with bcrypt
• Role-based middleware protection
• Input validation and sanitization
• CORS configuration
• SQL injection prevention via Eloquent ORM

6.2 Error Handling

• Comprehensive API error responses
• Frontend error boundaries

- User-friendly error messages
- Logging for debugging and monitoring

7. Installation & Setup Instructions

7.1 Prerequisites

- PHP 8.2 or higher
- Composer
- Node.js 18+ and npm
- MySQL 8.0+
- Git

7.2 Installation Steps

1. Clone the repository
2. Backend setup:
3. Configure database in .env
4. Run migrations:
5. Frontend setup:

7.3 Running the Application

Backend:

Access at: http://127.0.0.1:8000

Frontend:

Access at: http://localhost:5173

8. Testing

8.1 Test Cases

- User authentication – Pass
- Book CRUD operations – Pass
- Borrowing workflow – Pass
- Payment processing – Pass
- Admin dashboard – Pass
- Digital reading – Pass

8.2 Known Issues & Limitations

- Single-branch library support only
- No advanced search filters
- Limited reporting features

9. Code Quality & Documentation

9.1 Code Structure

Backend Structure (Laravel):

```
pagekeeperrrrrrr/
├── app/
|   ├── Http/Controllers/Api/
|   |   ├── AuthController.php
|   |   ├── BorrowersController.php
|   |   ├── BooksController.php
|   |   ├── CategoriesController.php
|   |   ├── SuppliersController.php
|   |   ├── RequestBooksController.php
|   |   └── PaymentsController.php
|   ├── Models/
|   |   ├── borrowers.php
|   |   ├── books.php
|   |   ├── categories.php
|   |   ├── supplier.php
|   |   ├── request_books.php
|   |   ├── payments.php
|   |   └── book_authors.php
|   └── Middleware/
```

├── database/migrations/

├── routes/api.php

└── ...

Frontend Structure (React/TypeScript):

Library Management System front/

├── src/

│ ├── components/

│ │ ├── layout/

│ │ │ ├── Sidebar.tsx

│ │ │ ├── MemberSidebar.tsx

│ │ │ └── Header.tsx

│ │ ├── ui/ (shadcn/ui components)

│ │ └── shared/

│ ├── pages/

│ │ ├── admin/

│ │ │ ├── Dashboard.tsx

│ │ │ ├── Books.tsx

│ │ │ ├── Borrowers.tsx

│ │ │ └── ...

│ │ └── member/

│ │ ├── MemberDashboard.tsx

│ │ ├── BrowseBooks.tsx

│ │ └── ...

│ ├── contexts/

│ │ ├── AuthContext.tsx

│ │ └── NotificationContext.tsx

│ ├── lib/

│ │ ├── api.ts

│ │ └── systemHealthCheck.ts

│ └── ...

├── public/

└── ...

9.2 Code Standards

Backend (PHP/Laravel):

- Classes: PascalCase (BorrowersController, Book)
- Methods: camelCase (createBorrower, updateBook)
- Variables: camelCase ($borrower, $bookData)
- Database: snake_case (borrower_id, book_title)

Frontend (TypeScript/React):

- Components: PascalCase (BooksList, MemberDashboard)
- Functions/Variables: camelCase (fetchBooks, userData)
- Interfaces: PascalCase (Book, Borrower)
- Files: PascalCase for components

9.3 API Endpoints

Authentication:

- POST /api/auth/register - Register new member
- POST /api/auth/login - User login
- POST /api/auth/logout - Logout
- GET /api/auth/me - Get authenticated user

Books:

- GET /api/books - List books (with filters)

- POST /api/books - Create book
- GET /api/books/{id} - Get book details
- PUT /api/books/{id} - Update book
- DELETE /api/books/{id} - Delete book

Borrowers:

- GET /api/borrowers - List borrowers
- POST /api/borrowers - Create borrower
- GET /api/borrowers/{id} - Get borrower
- PUT /api/borrowers/{id} - Update borrower
- DELETE /api/borrowers/{id} - Delete borrower

Categories:

- GET /api/categories - List categories
- POST /api/categories - Create category
- PUT /api/categories/{id} - Update category
- DELETE /api/categories/{id} - Delete category