COLLEGE CODE : 9222

COLLEGE NAME : THENI KAMMAVAR SANGAM COLLEGEOFTECHNOLOGY

DEPARTMENT : B.tech(IT)

STUDENT NM-ID : 3971DB2B07CAADD9434475F74066FF6C

ROLL NO : 23it015

DATE : 02-10-2025

Completed the project named as

Phase__ TECHNOLOGY PROJECT

NAME : LIBRARY BOOK MANAGEMENT

SUBMITTED BY,

NAME : JEYASURYA K

MOBILE NO : 6374954151

Library Book Management

MVP Implementation

1. Project Setup

Objective: Establish the foundational structure of the application to enable smooth development and deployment.

Choose Technology Stack:
Decide on frontend and backend technologies.

For example:

- *Frontend: React.js, Vue.js, or Angular*
- *Backend: Node.js with Express, Django, or Flask*
- *Database: MongoDB, PostgreSQL, or MySQL*

Initialize Project:
Use package managers to scaffold the project.

For example:

Programing code:

```
npx create-react-app library-management
cd library-management
npm install axios react-router-dom
```

Folder Structure:
Organize files for scalability and clarity:

```
/src
  /components
  /pages
  /services
  /utils
  /styles
```

Setup Routing:
Implement basic routing to navigate between pages (Home, Books, Borrow, Admin).

Install Development Tools:
Set up ESLint, Prettier for code formatting and linting.
Configure environment variables for API keys or database URLs.

Version Control Initialization:
Initialize Git and create the initial commit to track the project from the start.

2. Core Features Implementation

Objective: Develop the minimum viable product functionalities that allow users to manage library books effectively.

User Interface:

- *Design a clean and responsive UI using CSS frameworks like Bootstrap or Material UI.*
- *Implement navigation menus and layout components.*

Book Catalog:

- *Display a list of books with essential details such as Title, Author, ISBN, Genre, and Availability.*
- *Pagination or infinite scroll for large catalogs.*

Book Management:

- *Admin users can add new books through a form with validation.*
- *Edit existing book details and update availability status.*
- *Delete books if necessary.*

Search & Filter:

- *Implement search bar functionality to search by Title, Author, or ISBN.*
- *Filters by Genre or Availability status.*

Borrowing System:

- *Users can borrow available books.*
- *Automatically update book availability status.*
- *Track due dates and send reminders (basic notification).*

User Roles & Authentication:

- *Simple login system (could be mocked or integrated with OAuth for MVP).*
- *Admin users have access to book management features; regular users can view and borrow books.*

3. Data Storage (Local State / Database)

Objective: Manage and persist data both locally and remotely.

Local State Management:

- *Use React's useState or useReducer to manage component state for UI updates.*
- *Use Context API or Redux for global state (e.g., user info, book list).*

Backend & Database Integration:

- *Create RESTful API endpoints for CRUD operations on books and users.*

Example API routes:

- *GET /books – Fetch list of books.*
- *POST /books – Add new book.*
- *PUT /books/:id – Update book details.*
- *DELETE /books/:id – Remove a book.*

Database Design:

- *Tables/Collections:*

- *Books (id, title, author, isbn, genre, availability, borrowedBy, dueDate)*
- *Users (id, username, passwordHash, role)*

Data Persistence:

- *Ensure data is saved to the database reliably.*
- *Use ORMs like Sequelize (Node.js) or Django ORM to abstract database queries.*

Error Handling:

- *Handle API failures gracefully and notify users of issues.*

4. Testing Core Features

Objective: Ensure the application works as expected and remains reliable as it grows.

Unit Testing:

- *Test individual components like the BookList, BookForm, and BorrowButton.*
- *Use Jest or Mocha for JavaScript testing.*

Integration Testing:

- *Test communication between frontend components and backend API.*
- *Mock API calls with tools like MSW (Mock Service Worker).*

End-to-End (E2E) Testing:

- *Simulate real user interactions such as login, searching books, borrowing books.*
- *Use Cypress or Selenium for automated E2E tests.*

Test Cases Examples:

- *Adding a new book updates the catalog correctly.*
- *Borrowing a book marks it as unavailable.*
- *Unauthorized users cannot access admin features.*

Continuous Testing:

- *Integrate tests into the CI/CD pipeline for automatic test execution on every commit.*

5. Version Control (GitHub)

Objective: Manage the project codebase effectively with version control best practices.

Initialize Git:

- *Run git init and create a .gitignore file to exclude node_modules, environment files, etc.*

Create Remote Repository:

- *Set up a repository on GitHub for collaboration and backup.*

Branching Strategy:

- *Use feature branches (feature/book-management), bugfix branches (bugfix/login-error), and a protected main branch.*
- *Merge via Pull Requests to enable code reviews.*

Commit Messages:

- *Follow clear and consistent commit message conventions,*

  e.g.,

  ```
  feat: add book search functionality
  fix: correct availability status update bug
  docs: update README with setup instructions
  ```

Collaboration:

- *Enable Issues and Projects on GitHub to track tasks and bugs.*
- *Document contribution guidelines for new contributors.*

CI/CD Integration (Optional):

- *Connect GitHub repository with CI tools like GitHub Actions to automate tests and deployments.*