Kristu Jayanti College

AUTONOMOUS Bengaluru

Reaccredited 'A++' Grade by NAAC | Affiliated to Bengaluru North University

PROJECT REPORT ON

"My Books Library"

Submitted in partial fulfillment of the requirements for Software Engineering Mini Project Lab for 2nd Semester

Master of Computer Applications

Submitted by

Ms. Aasreen Khatoon ***24MCAB02***

Mr. Karthik M ***24MCAB31***

Under the guidance of

Dr. Bharathi V

Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited 'A++' Grade by NAAC | Affiliated to Bengaluru North University

CERTIFICATE

This is to certify that the project titled **"My Books Library"** has been satisfactorily completed by **Mr. Karthik M** with **Reg. No. 24MCAB31** in partial fulfillment of the requirements for ***Software Engineering Mini Project Lab*** with course code **MCC2P2B21,** for the 2nd Semester MCA course during the academic year 2024-2026 as prescribed by Bangalore North University.

Faculty In-charge **Head of the Department**

Valued by

Examiner 1:_____ Date:

Examiner 2:_____ Centre: Kristu Jayanti College

ACKNOWLEDGEMENT

First of all, we would like to thank the God Almighty for all the blessings he has showered on us. Our spiritual quotient gave us more strength and motivation that helped immensely.

We would like to thank **Rev. Fr. Dr. Augustine George,** our esteemed Principal, for providing us their constant guidance and support. I would also like to thank **Rev. Fr. Lijo P Thomas,** our Vice-Principal and Chief Finance Officer, for providing us with the best facilities.

We are extremely thankful to our **Dr. Kumar R**, Head, Department of Computer Science (PG) for giving us the essential support in the form of allocating comfortable project hours and necessary software resources.

We would like to extend our heartfelt thanks to **Dr. Bharathi V,** our project guide for providing us the necessary details related to project development and process identification enabling us to finish the project within the stipulated time.

We thank all other faculty members who helped us a lot in completing this project.

We thank our class mates, who have pointed out errors and guided us a lot and we thank each and every one who has helped us.

Synopsis

The digital reading experience is rapidly evolving, with users seeking more than just a platform to read e-books. My Books Library is a feature-rich web application designed to cater to book enthusiasts by offering a seamless and interactive reading experience. Unlike conventional e-book readers, this app integrates advanced functionalities such as In-App Notes and Annotations, Progress Tracking and Analytics, and Audio Integration. These features ensure that users can actively engage with their books, monitor their reading habits, and enjoy audiobooks, making the app a versatile tool for modern readers.

One of the standout features of My Books Library is its In-App Notes and Annotations, which allow users to highlight text, add personal notes, and categorize their annotations for easy reference. The Progress Tracking and Analytics feature enables users to monitor their reading habits through visual insights, set goals, and maintain motivation with gamification elements like badges and streaks. Additionally, the Audio Integration feature supports text-to-speech conversion and audiobook playback, ensuring a flexible and accessible reading experience. These functionalities collectively enhance engagement, encourage active learning, and provide users with a well-rounded digital library experience.

By integrating these advanced features, My Books Library transforms traditional e-reading into an interactive and user-centric experience. The app is designed to support diverse reading preferences, whether users prefer immersive reading with annotations, goal-oriented tracking, or listening to audiobooks on the go. With its comprehensive approach to digital reading, My Books Library serves as an essential tool for students, researchers, and book lovers, offering convenience, engagement, and accessibility in one unified platform.

CONTENTS

1.INTRODUCTION

The rise of digital reading has created a demand for advanced e-reading platforms that go beyond traditional book consumption. My Books Library is a mobile application designed to provide a seamless and interactive reading experience by integrating features such as In-App Notes and Annotations, Progress Tracking and Analytics, and Audio Integration. This application is a useful tool for students, researchers, and book lovers because it lets users read, annotate, track their progress, and listen to audiobooks. With its user-friendly interface and advanced functionalities, My Books Library enhances accessibility, engagement, and efficiency, transforming the way readers interact with digital books.

1.1 System Definition

The rapid evolution of digital technology has significantly transformed the way people consume and interact with books. With the increasing reliance on smartphones and tablets, traditional reading habits are gradually being replaced by digital alternatives. While conventional e-book readers fulfill the basic function of reading, today's readers expect much more—interactivity, personalization, accessibility, and analytics.

To address this demand, My Books Library emerges as a comprehensive digital reading platform that combines traditional reading with modern enhancements. Built using PHP, MySQL, and hosted via XAMPP, this application is designed to deliver a seamless and interactive user experience.

The system allows users not only to read e-books but also to annotate, track reading progress, and listen to audio versions of books. With a clean and intuitive user interface, the platform is suitable for a diverse audience including students, educators, researchers, and avid readers. The goal is to make digital reading more meaningful, efficient, and enjoyable through an integrated feature set.

Through features such as In-App Notes and Annotations, Progress Tracking and Analytics, Audio Integration, and AI-powered Book Recommendations, My Books Library transforms digital reading into a personalized, productive, and engaging experience.

1.2 Project Description

My Books Library is a full-fledged digital book management and reading platform developed using PHP and MySQL, hosted locally via XAMPP. Unlike conventional e-readers that offer basic reading utilities, this application introduces a dynamic reading environment that supports interaction, engagement, and user customization.

Users can import e-books in commonly supported formats like PDF or TXT and store them in their personalized digital library. The platform offers a text viewer for reading, an annotation tool for highlighting and note-taking, and a progress tracker that records reading activity such as time spent reading and percentage of book completed.

One of the standout features is the Audio Integration module, which utilizes text-to-speech APIs to convert written content into audio. This allows users to effortlessly switch between reading and listening modes, thereby catering to different learning styles and preferences.

Additionally, the system supports AI-powered recommendations, helping users discover books that align with their interests and past behavior. It also includes a secure user authentication system, allowing personalized experiences while maintaining data security.

All user data, including books, notes, reading logs, and preferences, is stored securely in a MySQL database, ensuring persistence and structured data management. The application logic is handled through PHP scripts, and the entire system is served through a XAMPP environment, allowing local hosting for testing and deployment.

This makes *My Books Library* not just a reading tool, but a complete ecosystem for digital learning and literary engagement.

1Modules in My Books Library

The functionality of *My Books Library* is divided into well-defined modules to support modularity, scalability, and maintainability. Each module is responsible for a specific set of tasks and interacts with others through cleanly defined interfaces and data flows. Below is an in-depth overview of each module:

1. User Authentication and Profile Management

- Built using PHP sessions and MySQL queries.
- Allows users to sign up, log in, and manage their accounts.
- Supports email-based registration and password recovery.
- Stores user preferences, reading history, and progress tracking data in personalized profiles.
- Uses form input validations and session management for security.

2. E-Book Management

- Users can upload e-books in supported formats (PDF, TXT).
- Backend PHP code parses and stores metadata in the MySQL database.
- Books are categorized by genre, author, or tags.
- Features search and filter functionality to quickly find desired content.
- Admin panel allows management (add/edit/delete) of books in the library.

3. In-App Notes and Annotations

- Users can highlight text while reading and add annotations to specific segments.
- Notes are saved to the database and linked to the corresponding book and user.
- Features include color-coded highlights, editable notes, and a personal notes dashboard.
- Annotations can be sorted by date, book, or custom tags.
- Uses AJAX and PHP to save notes without page reloads for a seamless experience.

4. Progress Tracking and Analytics

- Automatically logs the number of pages read, time spent, and books completed.

- Uses timers and user events (page scrolls, button clicks) to record session data.
- Progress is displayed in visual formats (bars, pie charts) using tools like Chart.js.
- Data is stored and retrieved using MySQL, providing accurate user analytics.
- Gamification elements like reading streaks and achievement badges are included to motivate users.

5. Audio Integration

- Text-to-speech feature converts readable content into audio using an API (e.g., Google Text-to-Speech).
- Allows users to play, pause, skip, or repeat content.
- Audio bookmarks let users save specific points to revisit.
- Playback speed and volume settings are customizable.
- Enhances accessibility for users who prefer auditory learning or have reading difficulties.

6. Library Synchronization and Backup

- Cloud backup support (optional with integration APIs like Google Drive or Dropbox).
- User data and library files are stored in structured MySQL tables.
- Database dumps can be taken from phpMyAdmin to backup and restore library data.
- Users can sync across devices (mobile, desktop) via cloud sync or session-based access.

7. Search and Recommendation System

- Advanced search engine with filters for title, author, category, and keyword.
- Implements simple AI logic in PHP (or via third-party APIs) to recommend books based on:

- Most read genres
- Recently completed books
- Reading time analysis

- Search results are ranked and highlighted for easy navigation.
- Promotes personalized discovery of new books, improving user retention.

2. SYSTEM STUDY

2.1 Existing System

In the current digital era, e-reading platforms have become increasingly popular due to the convenience they offer. However, most traditional platforms are primarily focused on providing only the basic functionalities required for reading digital books. These include basic text display, file storage, font customization, bookmarking, and simple navigation controls. Although these features suffice for casual reading, they fall short in terms of deeper reader engagement, productivity, and interactive learning.

While popular platforms such as Amazon Kindle, Google Play Books, Apple Books, and Kobo Books have made significant contributions to the world of digital reading, they still suffer from various limitations that prevent them from offering a fully immersive and productive experience.

Limitations of Current E-Reading Platforms

1. **Minimal User Engagement Tools:**
   - Most e-readers offer basic highlights or annotations, but these are often difficult to manage or organize. Users cannot categorize notes, tag them for future reference, or extract them easily for study or research purposes.
2. **Lack of Progress Analytics:**
   - Platforms generally fail to provide deep insights into a user's reading habits. Users are unable to view metrics such as total time spent reading, pages read per session, daily/weekly reading goals, or personalized reading streaks.
3. **Limited Audio Integration:**
   - While some apps offer audiobooks as separate products, the integration of text-to-speech (TTS) within e-books is rare or limited. Users must often switch between apps to transition from reading to listening, disrupting the reading flow.
4. **No Gamification Features:**

- Encouraging regular reading through gamified elements like badges, leaderboards, or streak counters is largely missing in most platforms. This results in a lack of motivation for users who enjoy rewards or progress milestones.

5. **Fragmented Experience Across Multiple Apps:**

- Readers often need to use multiple applications—one for reading, another for note-taking, one for audio, and another for tracking goals. This leads to inefficiency and data fragmentation.

6. **Absence of AI-Based Recommendations:**

- Most traditional systems recommend books based on category or popularity, not based on individual user preferences, past reading history, or behavioral patterns, which limits the personalized experience.

How My Books Library Improves Upon Existing Systems

My Books Library is designed to address the limitations of traditional platforms by integrating all essential and advanced functionalities into a single unified platform. Developed using PHP for server-side scripting, MySQL for structured data storage, and hosted via XAMPP, the system is flexible, extendable, and easy to deploy.

Key improvements over existing systems include:

- **Comprehensive In-App Note-Taking** with tagging, categorization, and color-coded highlights for better organization and reference.
- **Detailed Progress Tracking** including session duration, percentage completion, pages read per day, and streak tracking with graphical dashboards.
- **Seamless Text-to-Speech Integration**, allowing readers to switch between reading and listening instantly.
- **Gamification Features** like badges, goals, and rewards to motivate consistent reading habits.

- Personalized AI-Driven Recommendations based on reading history, genre preferences, and engagement patterns.
- Single Platform Access – No need for multiple apps; everything from reading to note-taking, analytics, and listening happens within one environment.

This holistic approach not only streamlines the reading process but also fosters higher engagement, greater productivity, and enhanced learning outcomes for all types of users.

2.2 Proposed System

The proposed system, *My Books Library*, is a powerful, scalable, and user-friendly digital reading application that integrates reading, annotating, progress tracking, and audio playback within a unified framework. The system is tailored to meet the modern reader's need for personalization, productivity, and accessibility.

Built using PHP for backend logic, MySQL for data handling, and XAMPP for local server management, the platform ensures a smooth development, testing, and deployment workflow. It is designed to work efficiently across mobile and web platforms, ensuring seamless user experiences across devices.

Core Features of the Proposed System

1. **In-App Notes and Annotations:**
   - Users can highlight specific portions of text, add notes, and organize annotations by custom tags.
   - Notes are stored in the MySQL database with timestamp, book reference, and location for easy retrieval.
   - Notes dashboard allows users to filter, edit, or delete annotations.
2. **Progress Tracking and Analytics:**
   - Tracks how many pages a user reads during each session.
   - Displays visual analytics including graphs of reading activity over time.

- Users can set reading goals (e.g., 10 pages/day) and unlock badges or streak rewards.
- Data is persistently stored and retrieved dynamically using PHP scripts.

3. **Audio Integration:**

- Implements Text-to-Speech (TTS) functionality to convert e-book content into audio.
- Users can control playback speed, volume, and navigate using audio bookmarks.
- Supports multi-tasking by enabling users to listen while taking notes or tracking progress.

4. **AI-Powered Book Recommendations:**

- Utilizes past reading history and stored user data to recommend books via a custom PHP algorithm.
- Enhances user engagement by showing relevant books based on preferences.

5. **Unified Dashboard and User Profile:**

- Users have a personalized dashboard showing:
  - Total books read
  - Ongoing reading sessions
  - Notes taken
  - Time spent
  - Upcoming goals
- Secure login system built using PHP sessions with password encryption.

6. **Cloud Synchronization (Optional):**

- Allows users to back up their books, notes, and progress data via cloud integrations.
- Ensures smooth access across multiple devices with the same login.

7. **Responsive UI Design:**

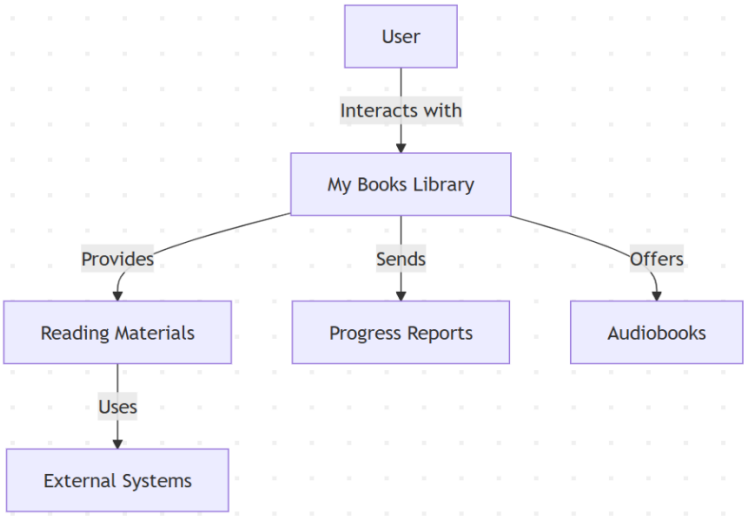
   - Designed to be mobile-friendly and accessible from any device.
   - Implemented using HTML, CSS, and Bootstrap within the XAMPP environment.

Benefits of the Proposed System

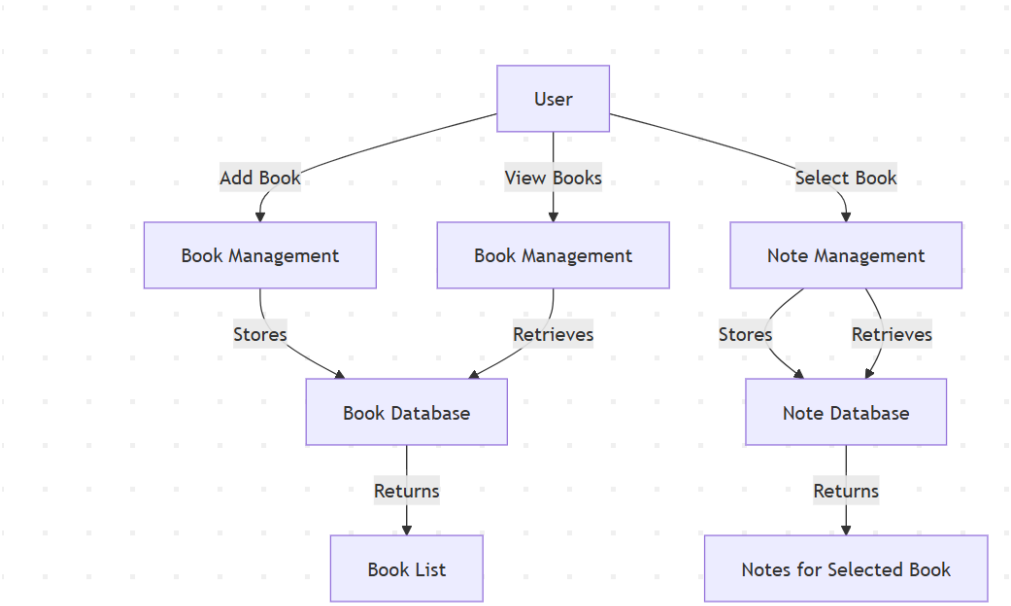
- **Efficiency:** Combines multiple tools into one system, saving users from juggling between apps.
- **Engagement:** Features like goals, gamification, and progress visualization make reading more enjoyable.
- **Accessibility:** Text-to-speech ensures support for visually impaired or multitasking users.
- **Personalization:** AI recommendations and profile management ensure each user gets a unique experience.
- **Educational Use:** Ideal for students and researchers needing advanced note-taking and annotation features.

2.3 Data Flow Diagram

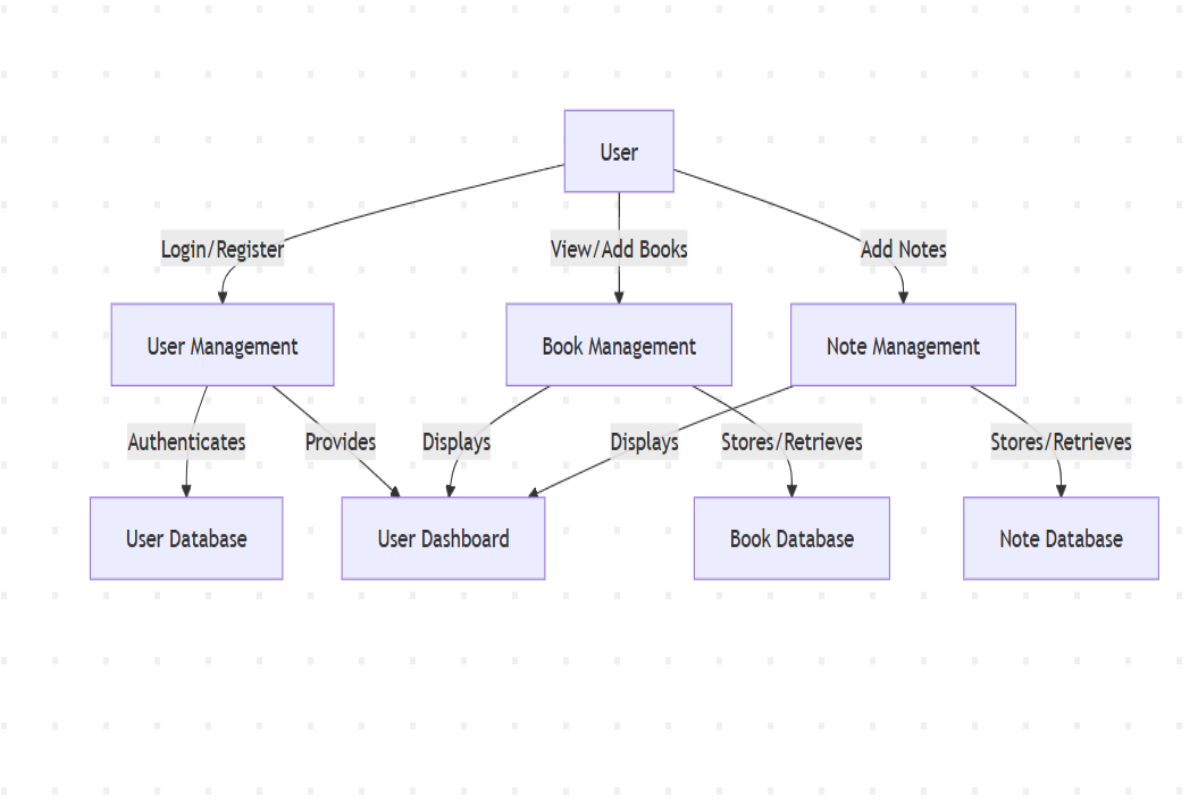
Level 0



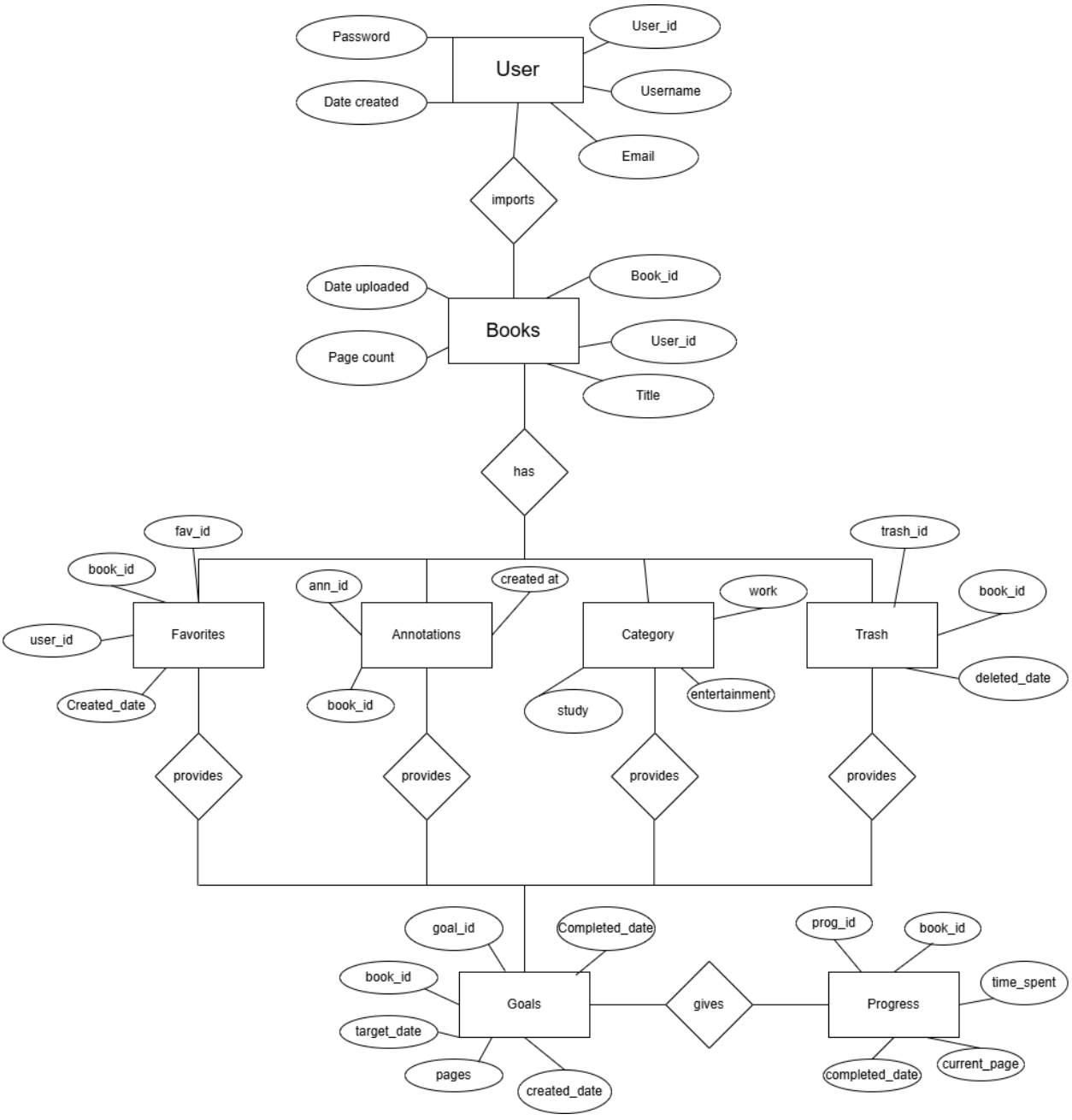
Level 1



Level 2


User
Login/Register
View/Add Books
Add Notes
User Management
Book Management
Note Management
Authenticates
Provides
Displays
Displays
Stores/Retrieves
Stores/Retrieves
User Database
User Dashboard
Book Database
Note Database

2.4 E-R Diagram



3. SYSTEM CONFIGURATION

The System Configuration outlines the hardware and software requirements necessary for the proper functioning of the Study Companion platform. A well-defined system configuration ensures that the project runs smoothly, efficiently, and without performance issues.

3.1 Hardware Configuration

The hardware requirements define the minimum and recommended specifications needed for both development and deployment of the Study Companion platform.

Minimum Requirements:

- **Processor:** Intel Core i3 (or equivalent AMD processor)
- **RAM:** 4GB
- **Storage:** At least 20GB of free disk space
- **Operating System:** Windows (7, 8, 10, 11), Linux, or macOS
- **Internet Connection:** Required for course content streaming and online functionality

Recommended Requirements:

- **Processor:** Intel Core i5 or higher (for faster processing)
- **RAM:** 8GB or more (for handling multiple users simultaneously)
- **Storage:** SSD with at least 50GB of free space (for better performance)
- **Operating System:** Latest versions of Windows, Linux, or macOS
- **Internet Connection:** High-speed internet (for smooth video playback and payments)

The recommended specifications ensure a better user experience, especially for handling multiple users, streaming course videos, and running database operations smoothly.

3.2 Software Configuration

The software configuration includes all the necessary technologies, frameworks, and tools used to build and run the Study Companion system.

Front-End (User Interface Development):

- **HTML** – To structure the webpages.
- **CSS** – For styling and designing the UI.
- **Bootstrap** – To make the website **responsive** and user-friendly on different devices.
- **Back-End (Server-Side Development):**
- **PHP** – Handles all logic, user authentication, course management, and payment integration.
- **Apache Server (via XAMPP)** – A local server used for testing and hosting the application.

Database Management:

- **MySQL** – Used to store user accounts, course details, enrollments, quizzes, and certificates.
- **phpMyAdmin** – A web-based database management tool used to manage the MySQL database.

Development & Deployment Tools:

- **XAMPP** – A package that includes Apache (server), MySQL (database), and PHP for local development.
- **VS Code**– Code editors for writing and debugging PHP, HTML, CSS, and JavaScript.
- **GitHub** – For version control and collaboration.
- **Security & Authentication:**

- **Email Verification for Password Reset** – Allows users to recover their accounts securely.
- **Session Management in PHP** – Prevents unauthorized access.
- **Encryption for Passwords** – Ensures user credentials are stored securely.

4. Details of Software

The development of the My Books Library application involved the careful selection of modern web technologies to ensure robust functionality, a clean and interactive user interface, and a seamless backend for data management and operations. This section elaborates on the front-end and back-end architecture, covering the technologies, features, and tools used throughout the development process.

4.1 Front-End Overview

Technology Used

The front-end of the My Books Library application has been developed using a combination of HTML, CSS, JavaScript, and PHP (used in both frontend templating and server communication). These technologies work together to deliver a dynamic, responsive, and interactive user experience.

- **HTML (HyperText Markup Language):** Used to structure the content and layout of the web pages. HTML is the backbone of all page components, including headers, footers, buttons, input fields, and book containers.
- **CSS (Cascading Style Sheets):** Applied to style and visually enhance the HTML content. CSS is responsible for all the visual elements such as color schemes, layout alignment, spacing, transitions, animations, and responsiveness across various devices.
- **JavaScript:** Used for adding interactivity and client-side logic such as form validation, dynamic content rendering (e.g., real-time progress bar updates), modals, tab switching, and AJAX calls to communicate with PHP scripts in the backend without reloading the page.
- **PHP (Templating in Frontend):** In addition to backend logic, PHP is embedded in the front-end files to dynamically fetch and display data such as book lists, user names, reading statistics, and recommendations.

Features of the Front-End

The front-end layer acts as the primary interface through which users interact with the My Books Library system. It has been designed to be visually engaging, responsive, and accessible for all types of users.

1. User Authentication Interface

- Login and registration pages that allow users to sign in via email.
- Error prompts and input validation using JavaScript.
- Password reset or recovery options.

2. Interactive Dashboard

- Displays visual charts for books read, pages read, session time, and reading streaks.
- A clean overview of currently reading, completed books, and suggested titles.
- Toggle options for light/dark mode for comfortable reading.

3. Book Viewer and Reader Interface

- Supports embedded reading of uploaded PDF, TXT, or EPUB files.
- Integrated annotation tools for highlighting, note-taking, and commenting.
- Progress bar indicating current page number and percentage completed.

4. Notes & Annotations Panel

- Sidebar showing all notes related to the book being read.
- Filters and search options to find notes by tags or keywords.
- Editable note cards with delete/update functionality.

5. Audio Playback Interface

- Custom player with play, pause, rewind, forward, volume, and speed control.

- Displays synced text for text-to-speech features.
- Allows bookmarking specific parts of audio sessions.

6. Responsive UI Design

- Built with media queries and CSS flex/grid layouts to ensure proper display on mobile, tablet, and desktop.
- Adaptive UI elements that resize or reorient based on the screen size.

Development Tools Used (Front-End)

- **Visual Studio Code (VS Code):**
  - Primary IDE used for writing, editing, and managing HTML, CSS, JS, and PHP code.
  - Extensions like *Live Server*, *PHP Intelephense*, and *Prettier* used to improve productivity and code quality.
- **Browser Developer Tools:**
  - Used for inspecting elements, debugging JavaScript, and live testing CSS.

4.2 Back-End Overview

Technology Used

The backend of My Books Library handles data storage, logic processing, and business rules. It is primarily developed using the PHP scripting language, powered by MySQL as the relational database system, and managed using phpMyAdmin. The entire application is hosted and run locally through XAMPP.

- **PHP (Hypertext Preprocessor):**
  - Handles server-side operations such as user authentication, book upload, session tracking, annotation storage, and API responses for AJAX.
  - Acts as the middleware between the front-end and the database.

- **MySQL:**
  - Used to manage and store structured data related to users, books, reading sessions, annotations, audio bookmarks, and analytics.
  - Ensures data integrity, fast retrieval, and reliable storage.
- **phpMyAdmin:**
  - A web-based GUI tool to interact with the MySQL database.
  - Used for creating tables, setting primary and foreign keys, managing user roles, and performing queries.
- **XAMPP (Apache + MySQL + PHP + Perl):**
  - Acts as the local server environment to host and run the entire project.
  - Allows seamless testing and debugging before deployment.

Features of the Back-End

1. User Management

- Registration, login, logout, and password encryption handled using PHP.
- Sessions managed using PHP's session handlers to maintain user state.
- Profile data is securely stored and fetched from the database using SQL queries.

2. Book Management

- Users can upload PDF/EPUB/TXT files via a form, which are stored in the server directory and referenced in the MySQL database.
- Book metadata such as title, author, upload date, and genre is stored in structured tables.

3. Annotation and Note Handling

- Each annotation is associated with a specific user, book, and page number.
- Stored in a relational structure allowing efficient sorting, searching, and filtering.
- PHP scripts handle CRUD operations (Create, Read, Update, Delete) for annotations.

4. Audio and Text-to-Speech Integration

- Audio files (if included) are stored on the server and linked via file paths.
- TTS content is dynamically generated and delivered through PHP-generated audio.

5. Reading Progress and Analytics

- Backend continuously logs:
  - Pages read
  - Session start and end time
  - Completion percentage
- PHP scripts analyze these logs and calculate daily/weekly activity, stored in reading_logs tables.

6. AI-Based Recommendations (Basic Logic)

- A PHP script fetches previously read genres and authors.
- Based on historical data, the system recommends new books stored in the library.

Development Tools Used (Back-End)

- **Visual Studio Code:**
  - Used for writing and debugging PHP scripts. r.
  - Includes extensions like PHP Debug, MySQL, and Code Runne

- **phpMyAdmin:**
  - Used to manage the database via a visual dashboard.
  - Simplifies table creation, relationship mapping, and query execution.
- **XAMPP Control Panel:**
  - Used to start/stop Apache and MySQL servers.
  - Provides access to logs, security settings, and module configurations.
- **Postman (For API Testing):**
  - Optional tool used to test backend PHP endpoints for data exchange and response formatting.

Together, the front-end and back-end technologies used in My Books Library create a powerful, full-stack digital reading application. The front-end provides a clean, engaging, and user-friendly interface, while the back-end ensures efficient data handling, secure user management, and real-time functionality. The use of PHP, MySQL, and XAMPP not only simplifies development but also makes the system highly extensible for future features like cloud integration, AI-driven summarization, or real-time collaboration.

5. SYSTEM DESIGN

5.1 Architectural Design

Introduction to Architectural Design

Architectural design is one of the most critical phases in the system development lifecycle. It defines the high-level structure of the software, specifying how different components of the system interact, how data flows, and how each layer is separated to promote modularity, maintainability, and scalability. In the case of My Books Library, the system has been designed using a three-tier architecture—a widely accepted software architecture model that separates concerns between the Presentation Layer, Application (Logic) Layer, and Data Layer.

This design approach not only ensures better organization and code maintainability but also enhances the application's performance, security, and ease of testing.

Three-Tier Architecture Overview

The **three-tier architecture** divides the entire application into three independent but connected layers:

1. Presentation Tier (User Interface Layer)

Purpose:

This is the layer that the user interacts with directly. It handles the display of content and receives input from users.

Technologies Used:

- **HTML** for structuring content
- **CSS** for design and layout
- **JavaScript** for interactivity

- **PHP** for dynamic content rendering
- **AJAX** for asynchronous communication with the backend

Features in This Layer:

- Login/Register interface
- Dashboard showing analytics and stats
- Book reader UI with annotation/highlight tools
- Audio player with playback controls
- Notes and bookmarks display
- Progress bar showing reading completion percentage
- Search and recommendation UI

How It Interacts:

- Sends user inputs (e.g., login credentials, uploaded books, notes) to the application layer using forms or AJAX requests.
- Receives processed data from the application layer (e.g., fetched book list, analytics, progress status) and displays it on the UI.

2. Application Tier (Business Logic Layer)

Purpose:

This layer is the brain of the system. It processes user requests, applies business logic, and communicates with the database.

Technologies Used:

- **PHP** (Server-side scripting)
- Session and cookie management

- Authentication logic
- File handling (book uploads, audio)
- Custom functions for progress tracking, TTS integration, and annotations

Responsibilities:

- Validating user input (login, registration, book uploads)
- Managing sessions and user state
- Handling CRUD operations for books, notes, annotations
- Tracking reading activity and generating analytics
- Returning JSON/XML or HTML back to the front-end via AJAX
- Processing data before sending it to the database layer or UI

Example Flow:

- When a user adds a note, the front-end sends the note data to a PHP script.
- The script validates and structures the data, then calls database functions to store the note.
- A confirmation message or updated note list is returned to the front-end.

3. Data Tier (Database Layer)

Purpose:

This layer is responsible for storing, retrieving, and managing the persistent data of the application.

Technologies Used:

- **MySQL** for relational database management
- **phpMyAdmin** for database administration

- **XAMPP** as the local environment to run MySQL services

Tables in Database (example structure):

- users – stores user credentials and profile data
- books – stores metadata of uploaded books
- annotations – stores highlights, notes, and tags
- reading_progress – tracks progress, time spent, sessions
- audio_bookmarks – stores bookmarks during audiobook listening
- recommendations – caches AI-generated book suggestions

Responsibilities:

- Ensures data persistence and integrity
- Supports efficient querying and data retrieval
- Manages relationships between tables using foreign keys
- Protects data using privileges and authentication

How It Interacts:

- The application tier sends SQL queries to insert, update, fetch, or delete data.
- The database responds with results (e.g., list of books, notes, reading history).
- The application layer formats this data and sends it back to the front-end for display.

Benefits of Three-Tier Architecture in My Books Library

- **Separation of Concerns:**

Each layer is responsible for a specific function, making the system modular and easier to manage.

- **Scalability:**

Because each layer is independent, future features like cloud backup or advanced AI modules can be added without rewriting the entire application.

- **Maintainability:**

If a bug is found or a change is needed, developers can fix or update one tier without affecting the others significantly.

- **Security:**

Sensitive operations like login, session handling, and database queries are hidden from the user, reducing risks like SQL injection and unauthorized access.

- **Performance Optimization:**

Caching logic can be implemented in the application layer. Indexes and optimized queries improve database speed.

The architectural design of My Books Library ensures a systematic, organized, and secure flow of data and functionality. The three-tier architecture supports scalability, modularity, and efficient management, aligning perfectly with modern web application development standards. Each layer has been implemented with suitable tools and technologies, ensuring seamless communication between users, the business logic, and the underlying data storage system.

With this design, My Books Library can be extended to support additional features like real-time collaboration, cloud sync, or AI-based summarization, making it a sustainable and future-ready digital reading platform.

5.2 Input Design

Input design is one of the most critical components of system design. It refers to the process of designing the way users provide data or instructions to a system. An effective input design enhances usability, reduces user effort, ensures data accuracy, and ultimately improves the overall user experience. In *My Books Library*, a digital reading platform, input design plays a crucial role in how users interact with the system—whether it's logging in, uploading books, annotating content, listening to audio, or tracking reading progress.

The input design of this application focuses on intuitiveness, efficiency, and user-friendliness while supporting diverse input formats and interaction modes. Let's explore the key input components in the system.

1. User Authentication

The first point of interaction between the user and the system is the authentication process. *My Books Library* supports multiple login options to provide flexibility and convenience.

Input Options:

- **Email and Password Login**: Traditional login method where users enter registered email and password.
- **Google Sign-In**: Allows users to sign in using their Google account through OAuth, reducing the need to remember separate credentials.
- **OTP-Based Login (Optional)**: For quick mobile-based access, users can log in using a One-Time Password sent to their email or mobile number.

Design Considerations:

- Validation of inputs (e.g., email format, password strength)
- Clear error messages for incorrect inputs
- Secure handling of credentials using PHP sessions and hashing (e.g., bcrypt)

Purpose and Impact:

- Streamlined onboarding and returning-user experience
- Improves accessibility for users who prefer federated login systems like Google
- Ensures secure and fast access to personalized reading data

2. Book Importing

Uploading books into the digital library is a core function. The system supports importing different file types for flexibility.

Input Methods:

- File Upload via Local Storage: Users select book files in formats like PDF, EPUB, or TXT from their device.
- Drag-and-Drop Upload: A user-friendly drag-and-drop feature to add books quickly.
- Cloud Integration (Future Scope): Allows importing books from Google Drive or Dropbox.

Design Considerations:

- File validation (format, size limit, duplicates)
- Preview of uploaded content
- Responsive design to support mobile uploads

Impact:

- Makes it easy for users to add their personal collection to the app
- Reduces dependency on third-party reading apps
- Enhances convenience, especially for academic and personal reading collections

3. Notes & Annotations

This feature transforms reading from a passive activity to an interactive one. Users can actively engage with the content by taking notes and highlighting key sections.

Input Mechanism:

- **Text Highlighting**: Users select a portion of text and highlight it using different colors.
- **Note Writing**: Users type custom notes linked to specific pages or sections.
- **Categorization/Tagging**: Notes can be categorized by themes like "Important," "Doubt," or "Summary."

Features in Input Design:

- WYSIWYG-style note editors (with bold, italics, etc.)
- Keyboard and touch support for typing/editing notes
- Auto-save feature to prevent data loss
- **Impact:**
- Empowers students and researchers to organize and retain information
- Improves academic reading and study effectiveness
- Facilitates future revisions by grouping similar notes

4. Audio Playback Controls

A major feature of the app is its audio integration, enabling users to listen to audiobooks or text-to-speech content. The input design here focuses on providing precise and customizable controls.

Input Options:

- **Play/Pause Toggle**: Single tap to start or stop playback

- **Speed Adjustment**: Slider or dropdown to choose from 0.5x, 1x, 1.5x, or 2x speeds
- **Volume Control**: In-app volume slider to independently adjust sound
- **Bookmark Input**: Users can add bookmarks at specific timestamps
- **Jump Controls**: Buttons to jump forward/backward by 10 or 30 seconds

Design Considerations:

- Button size and spacing for easy touch input
- Real-time feedback (e.g., showing current speed or position)
- Accessibility support for visually impaired users

Purpose and Impact:

- Provides a personalized audio experience
- Enhances accessibility for users with reading difficulties
- Supports multitasking (e.g., listening while commuting)

5. Reading Progress Tracking

This is a passive yet intelligent input design, where the system captures user activity automatically and converts it into usable data.

Tracked Inputs (Automatically Captured):

- **Pages Read**: Detected based on scroll events or page navigation
- **Session Duration**: Tracked using login timestamps and inactivity timeouts
- **Books Completed**: Triggered when user reaches final page or clicks "Mark as Read"
- **Streaks and Daily Goals**: Based on login frequency and reading patterns

User Input (Optional):

- Setting custom goals (e.g., "Read 20 mins/day")
- Manually marking books or chapters as complete

Design Considerations:

- Input must not distract or interrupt the reading experience
- Background tracking using JavaScript timers or session data
- Progress indicators must update in real-time (e.g., progress bars)

Significance:

- Helps users stay motivated with visual feedback
- Useful for goal-driven readers, especially students
- Enables gamification (e.g., awards for daily streaks)

Overall Input Design Philosophy

The input design in *My Books Library* is developed with the following principles in mind:

- **User-Centric**: Inputs are designed to minimize effort and cognitive load
- **Error-Free**: Validations and feedback systems ensure clean and accurate data entry
- **Consistency**: Input methods are consistent across modules, improving usability
- **Accessibility**: Designed for users of all age groups and technical backgrounds

5.3 Output Design

Output design is a critical aspect of any software application as it determines how users perceive and interact with the system's data. Effective output design not only enhances usability but also boosts user engagement by providing clear, organized, and visually appealing information. In My Books Library, the output design is thoughtfully crafted to deliver a seamless reading experience while giving users insights into their reading behaviors, study patterns, and audio progress.

This section elaborates on the various types of outputs delivered by the system, the methods of their representation, and the role they play in enriching the overall user experience.

1. Dashboard & Analytics

The Dashboard is the first major output interface users encounter upon logging in. It acts as a personalized control center that provides insightful reading metrics and user engagement data.

Features of Dashboard Outputs:

- **Books Completed**: A visual indicator (usually in card format) showing how many books the user has finished reading. It may include progress rings or completion bars for ongoing books.
- **Total Reading Time**: Summarized in hours or minutes, this is typically shown using clocks, charts, or animated counters.
- **Reading Streaks**: Graphs or calendars indicate how consistently the user has read (e.g., reading for 5 days in a row).
- **Goal Tracking**: Users see how close they are to daily, weekly, or monthly goals via percentage bars or motivational meters.

Visualization Tools:

- Bar graphs, line charts, donut charts, and calendar heat maps are used to provide clarity and comparison over time.

Design Focus:

- Clean UI with a minimalist layout
- Color-coded metrics to indicate goal status (e.g., red = below target, green = goal met)
- Responsive dashboard view for mobile and desktop

Significance:

This visual output boosts motivation and allows users to self-monitor their habits, making reading more goal-oriented and productive. It's particularly helpful for students or competitive readers who wish to measure performance over time.

2. Notes & Highlights Panel

One of the standout features of *My Books Library* is its in-app note-taking and annotation system. The output panel for this feature is designed to allow quick retrieval and organization of user-generated content.

Features of Notes Output Panel:

- **Categorized Notes**: Users can view notes sorted by categories such as "Important," "Review Later," or "Summary."
- **Color-Coded Highlights**: Different colored highlights help users distinguish between various types of annotations (e.g., yellow for general, blue for critical).
- **Expandable View**: Clicking or tapping on a note expands it to show the full content and the page or section it belongs to.
- **Search Functionality**: Filters and search boxes help users quickly locate a specific note or keyword across the library.

User Experience Design:

- Sticky notes-like appearance for familiarity
- Smooth scrolling and pagination for large lists

- Hover effects (on desktop) or press-and-hold (on mobile) for quick actions like “Edit,” “Delete,” or “Copy”

Significance:

This output system serves academic readers and researchers, helping them revisit and organize study material efficiently. It transforms passive reading into an interactive and productive activity.

3. Audiobook Player Interface

For users who prefer listening to books, the audiobook player is an essential output component. It outputs real-time playback data and user controls in a compact, intuitive interface.

Player Outputs Include:

- **Playback Progress Bar**: Shows how much of the audio has been listened to and how much is remaining.
- **Current Timestamp**: Displays current listening time (e.g., “23:48 / 2:10:05”).
- **Bookmarks Display**: Highlights saved time positions with labels (e.g., “Chapter 2 Start”).
- **Speed Indicators**: Shows current speed like 1.0x, 1.5x, etc.
- **Volume Status**: Visual volume bar that reflects the current level

UI/UX Features:

- Floating player design that persists while navigating other sections of the app
- Dark/light mode compatibility
- Waveform or animated visualizer for aesthetic feedback during playback

Significance:

The audiobook interface outputs real-time feedback, making the listening experience smooth and controlled. It is designed with accessibility in mind for visually impaired users or those who prefer multitasking.

4. Personalized Book Recommendations

One of the most intelligent and user-focused output sections is the AI-driven recommendation panel. This output presents custom book suggestions based on the user's reading patterns, completed books, genres of interest, and notes.

Features of Recommendation Outputs:

- **Suggested Books Section**: Carousel or grid display of books recommended by the system.
- **"Because You Read..." Messages**: Contextual output that explains why a book is recommended (e.g., "Based on your interest in mystery novels").
- **Trending Reads**: Popular books across the platform to spark interest
- **Recently Viewed**: A section that outputs books the user recently interacted with for quick access

How the Output is Generated:

- Based on keywords in annotations, time spent on genres, completed books, and bookmarked sections
- Machine learning algorithms (in future scope) can improve accuracy by learning user behavior

Significance:

This output system adds personalization, helping users discover relevant content without having to search manually. It increases engagement by making content discovery seamless and enjoyable.

5. General UI Output Principles

Across all modules, the application maintains a consistent and clean **output display philosophy**, which includes:

- **Minimal Clutter**: Ensuring only the most relevant data is shown at any time
- **Visual Hierarchy**: Using size, color, and placement to guide user attention
- **Responsiveness**: Outputs scale and reflow gracefully across devices (phone, tablet, desktop)
- **Tooltips & Guidance**: On-hover or tap-based explanations for first-time users

Technologies Used for Output Rendering:

- **HTML5/CSS3**: For structured content display
- **JavaScript**: For dynamic, real-time output (e.g., dashboards, player feedback)
- **PHP**: Server-side rendering of personalized outputs like recommendations or notes
- **MySQL**: Outputs data stored in the database such as books completed, user notes, etc.

The output design of My Books Library plays a vital role in transforming reading from a solitary activity into an interactive, guided, and insightful experience. From visually rich dashboards to intelligent book recommendations and real-time audio feedback, each output component is crafted to serve a specific user need.

By adhering to principles of clarity, simplicity, and responsiveness, the output system ensures that users receive the right information at the right time with minimal effort.

5.4 Database Design

User:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	user_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	username	varchar(50)	utf8mb4_general_ci		No	Reader			Change Drop More
3	email	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	created_at	timestamp			No	current_timestamp()			Change Drop More

Book:

	bk_id	user_id	title	file_data	file_name	file_type pdf/epub/etc	file_size in bytes	thumbnail First page as thumbnail	category	page_count	upload_date	cover_image
Edit Copy Delete	17	1	24MCAB31_JAVA RECORD karthick.pdf		67e6625e3dc05_24MCAB31_JAVA RECORD karthick.pdf		0	NULL	study	NULL	2025-03-28 09:48:30	NULL
Edit Copy Delete	19	1	AlandML.pdf-compressed.pdf		67e6670050624_AlandML.pdf-compressed.pdf		0	NULL	study	NULL	2025-03-28 10:08:16	NULL
Edit Copy Delete	20	1	cant_help_fall_in_love.pdf.pdf		67e80c38955d9_cant_help_fall_in_love.pdf.pdf		0	NULL	study	NULL	2025-03-29 16:05:28	NULL

Favourites:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	book_id	int(11)			No	None			Change Drop More
3	user_id	int(11)			No	None			Change Drop More
4	created_at	timestamp			No	current_timestamp()			Change Drop More

Annotations:

ann_id	book_id	type	content	page_number	color For highlights	created_at

Trash:

trash_id	book_id	deleted_at

Reading goals:

go_id	book_id	target_date	pages_per_day	time_per_day in minutes	created_at	completed

Reading progress:

prog_id	book_id	current_page	last_read	time_spent in minutes	is_completed

6. SOURCE CODE

Login page:

```
<?php
// Start session at the beginning
session_start();
// Include database connection
require_once "db.php";
$errorMsg = "";
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["login"])) {
    // Sanitize inputs
    $email = mysqli_real_escape_string($conn, $_POST["email"]);
    $pass = $_POST["password"]; // Don't escape password - we'll hash it
     // Prepare statement to prevent SQL injection
   $stmt = $conn->prepare("SELECT `user_id`, `username`, `email`, `password` FROM user
WHERE email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $res = $stmt->get_result();
    if ($res->num_rows > 0) {
$ro = $res->fetch_assoc();
        // Verify password (assuming passwords are hashed in database)
        if (password_verify($pass, $ro["password"])) {
            $_SESSION["user_id"] = $ro["user_id"];
            $_SESSION["username"] = $ro["username"];

            header("Location: dashboard.php");
            exit();
        } else {
            $errorMsg = "Invaluser_id email or password";
        }
```

```
    } else {
        $errorMsg = "No account found";
    }
    $stmt->close();
}
// Close connection at the end of script
$conn->close();
?>
```

Homepage.php

```
<?php
session_start();
require_once 'db.php';

?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
                <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<linkrel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-latest.min.js"></script>
    <link rel="stylesheet" type="text/css" href="styles.css">

</head>
<body>
```

```
<!-- Original Sidebar Navigation (unchanged) -->
<div class="sidebar">
  <!-- My Library Section -->
  <h2 onclick="toggleDropdown('libraryDropdown')">My Library <i class="fa fa-chevron-down"></i></h2>
  <div id="libraryDropdown" class="dropdown-content">
      <a href="homepage.php?q=1" class="<?php if(@$_GET['q']==1) echo'active'; ?>">Books</a>
      <a href="homepage.php?q=2" class="<?php if(@$_GET['q']==2) echo'active'; ?>">Favorites</a>
      <a href="homepage.php?q=3" class="<?php if(@$_GET['q']==3) echo'active'; ?>">Notes</a>
      <a href="homepage.php?q=4" class="<?php if(@$_GET['q']==4) echo'active'; ?>">Highlights</a>
      <a href="homepage.php?q=5" class="<?php if(@$_GET['q']==5) echo'active'; ?>">Trash</a>
  </div>
  <!-- Shelf Section -->
    <h3 onclick="toggleDropdown('shelfDropdown')">Shelf <i class="fa fa-chevron-down"></i></h3>
  <div id="shelfDropdown" class="dropdown-content">
    <a href="#">Study</a>
    <a href="#">Work</a>
    <a href="#">Entertainment</a>
  </div>
  <!-- Progress Tracking Section -->
  <h3 onclick="toggleDropdown('progressDropdown')">Progress Tracking <i class="fa fa-chevron-down"></i></h3>
  <div id="progressDropdown" class="dropdown-content">
      <a href="homepage.php?q=6" class="<?php if(@$_GET['q']==6) echo'active'; ?>">Reading Goals</a>
```

```
        <a href="#">Current Progress</a>
        <a href="#">Completed Books</a>
    </div>
</div>
<div class="main">
    <!-- Rest of your main content remains exactly the same -->
    <div class="header">
        <div class="search-bar">
            <input type="text" placeholder="Search books..." id="searchInput">
            <button onclick="searchBooks()"><i class="fa fa-search"></i></button>
        </div>
        <div class="header-buttons">
            <a href="import.php"
               class="btn btnn-primary"
                 style="background-color: rgb(89, 59, 107); color:white; position: absolute; top:
20px; left: 1200px;">
               Import 📚
            </a>
            <a href="logout.php"
               class="btn btnn-primary"
                style="background-color: rgb(212, 38, 38); width: 100px ;color: white ; position:
absolute; top: 20px; left: 1300px;">
               Logout
            </a>
        </div>
    </div>
    <div class="booklist">
        <?php
        // Include the appropriate content based on q parameter
        if (!isset($_GET['q']) || $_GET['q'] == 1) {
            include('book.php');  // Main books listing
```

```
        }?>
  <?php
        // Include the appropriate content based on q parameter
        if (!isset($_GET['q']) || $_GET['q'] == 2) {
            include('favourite.php');  // Main books listing
        }?>
        <?php
        // Include the appropriate content based on q parameter
        if (!isset($_GET['q']) || $_GET['q'] == 3) {
            include('notes.php');  // Main books listing
        }?>
        <?php
        // Include the appropriate content based on q parameter
        if (!isset($_GET['q']) || $_GET['q'] == 4) {
            include('highlights.php');  // Main books listing
        }?>
        <?php
        // Include the appropriate content based on q parameter
        if (!isset($_GET['q']) || $_GET['q'] == 5) {
            include('trash.php');  // Main books listing
        }?>
    </div>
  </div>
  <script>
  // Fixed dropdown toggle function (unchanged)
  function toggleDropdown(id) {
    var dropdown = document.getElementById(id);
    dropdown.classList.toggle("show");

    // Toggle chevron icon
    var header = dropdown.previousElementSibling;
```

```
    var icon = header.querySelector('i');
    if (dropdown.classList.contains("show")) {
        icon.classList.remove("fa-chevron-down");
        icon.classList.add("fa-chevron-up");
    } else {
        icon.classList.remove("fa-chevron-up");
        icon.classList.add("fa-chevron-down");
    }
}
// Close dropdowns when clicking outside (unchanged)
window.onclick = function(event) {
    if (!event.target.matches('.sidebar h2') && !event.target.matches('.sidebar h3')) {
        var dropdowns = document.getElementsByClassName("dropdown-content");
        for (var i = 0; i < dropdowns.length; i++) {
            var openDropdown = dropdowns[i];
            if (openDropdown.classList.contains('show')) {
                openDropdown.classList.remove('show');
                var icon = openDropdown.previousElementSibling.querySelector('i');
                icon.classList.remove("fa-chevron-up");
                icon.classList.add("fa-chevron-down");
            }
        }
    }
// Your existing search function (unchanged)
function searchBooks() {
    var input = document.getElementById('searchInput').value.toLowerCase();
    var books = document.querySelectorAll('.book');
    books.forEach(function(book) {
        var title = book.querySelector('h3').textContent.toLowerCase();
        book.style.display = title.includes(input) ? '' : 'none';
    });
```

```
}
</script
</body>
</html>
<?php
// Close connection at the end
mysqli_close($conn);
?>
```

Books.php

```
<?php
session_start();
$link = mysqli_connect("localhost", "root", "", "mybooklibrary");
if (mysqli_connect_errno()) {
   die("Database connection failed: " . mysqli_connect_error());
}
// Updated query to use your books table structure
$user_id = 1;
$stmt = mysqli_prepare($link, "SELECT bk_id, title, file_name, upload_date FROM books
WHERE user_id = ? ORDER BY upload_date DESC");
mysqli_stmt_bind_param($stmt, "i", $user_id);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt)
while ($row = mysqli_fetch_assoc($result)) {
   $books[] = $row;
}
mysqli_stmt_close($stmt);
mysqli_close($link);
?>
<!DOCTYPE html>
<html>
```

```
<head>
    <title>My Library</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
 <linkrel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap
.min.css">
<linhref="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=
swap" rel="stylesheet">
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            background: linear-gradient(to right, rgb(199, 118, 224), #cfdef3);
            color: #333;
            margin: 0;
            padding: 20px;
        }
        .container {
            max-width: 1200px;
            margin: 0 auto;
            background: #fff;
            border-radius: 10px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
            padding: 30px;
        }
        h1 {
            color: rgb(19, 2, 18);
            margin-bottom: 30px;
        }
        .book-card {
            border: 1px solid #ddd;
            border-radius: 5px;
```

```
        padding: 15px;
        margin-bottom: 20px;
        background: #f9f9f9;
        display: flex;
        justify-content: space-between;
        align-items: center;
    }
    .book-info {
        flex-grow: 1;
    }
    .book-actions a {
        margin-left: 10px;
    }
    .btn-primary {
        background-color: #4C787E;
        border-color: #4C787E;
    }
    .btn-primary:hover {
        background-color: #006666;
        border-color: #006666;
    }
    .btn-danger {
        background-color: #dc3545;
        border-color: #dc3545;
    }
    .btn-danger:hover {
        background-color: #c82333;
        border-color: #bd2130;
    }
    .btn-back {
        margin-bottom: 20px;
```

```
        display: inline-block;
    }
    .upload-date {
        color: #666;
        font-size: 0.9em;
        margin-top: 5px;
    }
    .no-books {
        text-align: center;
        padding: 30px;
        color: #666;
    }
    /* PDF Viewer Styles */
    .pdf-viewer {
        display: none;
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: rgba(0,0,0,0.8);
        z-index: 1000;
        padding: 20px;
        box-sizing: border-box;
    }
    .pdf-container {
        width: 80%;
        height: 80%;
        margin: 5% auto;
        background: white;
    }
```

```
.close-pdf {
    position: absolute;
    top: 20px;
    right: 20px;
    color: white;
    font-size: 30px;
    cursor: pointer;

/* Loading spinner */
.pdf-loading {
    display: none;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    color: white;
    text-align: center;
}
.spinner {
    border: 4px solid rgba(255, 255, 255, 0.3);
    border-radius: 50%;
    border-top: 4px solid #fff;
    width: 40px;
    height: 40px;
    animation: spin 1s linear infinite;
    margin: 0 auto 10px;
}
@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

```
    </style>
</head>
<body>
    <div class="container">
        <a href="homepage.php" class="btn btn-secondary btn-back">Back to Home</a>
        <a href="import.php" class="btn btn-primary btn-back">Import New Book</a>
        <h1>My Library</h1>
        <?php if (empty($books)): ?>
            <div class="no-books">
                <p>You haven't uploaded any books yet.</p>
                <a href="import.php" class="btn btn-primary">Import Your First Book</a>
            </div>
        <?php else: ?>
            <div class="book-list">
                <?php foreach ($books as $book): ?>
                    <div class="book-card">
                        <div class="book-info">
                            <h4><?php echo htmlspecialchars($book['title']); ?></h4>
                            <div class="upload-date">
                        Uploaded on: <?php echo date('F j, Y, g:i a', strtotime($book['upload_date'])); 
                            </div>
                        </div>
                        <div class="book-actions">
                         <a href="view_pdf.php?id=<?php echo $book['bk_id']; ?>" class="btn btn-
primary">View</a>
                        <a href="delete_book.php?id=<?php echo $book['bk_id']; ?>" class="btn btn-
danger" onclick="return confirm('Are you sure you want to delete this book?')">Delete</a>
                        </div>
                    </div>
                <?php endforeach; ?>
            </div>
```

```
        <?php endif; ?>
    </div>
    <!-- JavaScript Libraries -->
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
 <scriptsrc="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script
>
</body>
</html>
```

Book.php

```
<?php
require 'db.php';
// TEMPORARY USER ID - REPLACE WITH YOUR AUTH SYSTEM
$user_id = 1;
// Handle Favorite Toggle via AJAX
if($_SERVER['REQUEST_METHOD']==='POST'&&isset($_POST['ajax_toggle_favorite
']')) {
    $response = ['success' => false];
    $bk_id = (int)$_POST['bk_id'];
    // Check if book exists
    $check_book = $conn->prepare("SELECT bk_id FROM books WHERE bk_id = ?");
    $check_book->bind_param("i", $bk_id);
    $check_book->execute();
    $check_book->store_result();
    if ($check_book->num_rows > 0) {
        // Check if already favorited
        $check_fav = $conn->prepare("SELECT fav_id FROM favorites WHERE book_id =
? AND user_id = ?");
        $check_fav->bind_param("ii", $bk_id, $user_id);
        $check_fav->execute();
        $check_fav->store_result();
```

```
        if ($check_fav->num_rows > 0) {
            // Remove from favorites
             $remove_fav = $conn->prepare("DELETE FROM favorites WHERE book_id = ?
  AND user_id = ?");
            if ($remove_fav->bind_param("ii", $bk_id, $user_id) && $remove_fav->execute())
  {
                $response = ['success' => true, 'is_favorite' => false];
            }
        } else {
            // Add to favorites
            $add_fav = $conn->prepare("INSERT INTO favorites (book_id, user_id) VALUES
  (?, ?)");
            if ($add_fav->bind_param("ii", $bk_id, $user_id) && $add_fav->execute()) {
                $response = ['success' => true, 'is_favorite' => true];
            }
        }
    }
    header('Content-Type: application/json');
    echo json_encode($response);
    exit();
}
// Handle Book Deletion (Move to Trash)
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['delete_book'])) {
    $bk_id = (int)$_POST['bk_id'];
    // Move to trash
    $move_to_trash = $conn->prepare("INSERT INTO trash (book_id) VALUES (?)");
    $move_to_trash->bind_param("i", $bk_id);
    $move_to_trash->execute();

    // Now delete from books table
```

```
    $delete_stmt = $conn->prepare("DELETE FROM books WHERE bk_id = ? AND user_id
= ?");
    $delete_stmt->bind_param("ii", $bk_id, $user_id);
    $delete_stmt->execute();
    // Also remove from favorites
    $delete_fav = $conn->prepare("DELETE FROM favorites WHERE book_id = ?");
    $delete_fav->bind_param("i", $bk_id);
    $delete_fav->execute();
    header("Location: book.php");
    exit();
}
// Handle Category Update
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['update_category'])) {
    $bk_id = (int)$_POST['bk_id'];
    $category = $_POST['update_category'];
    $update_stmt = $conn->prepare("UPDATE books SET category = ? WHERE bk_id = ?
AND user_id = ?");
    $update_stmt->bind_param("sii", $category, $bk_id, $user_id);
    $update_stmt->execute();

    header("Location: book.php");
    exit();
}
// Fetch user's books with favorite status
$stmt = $conn->prepare("
        SELECT b.bk_id, b.title, b.file_name, b.file_type, b.cover_image, b.category,
b.upload_date,
    (SELECT COUNT(*) FROM favorites f WHERE f.book_id = b.bk_id AND f.user_id =
?) as is_favorite
    FROM books b
```

```
    WHERE b.user_id = ?
    ORDER BY b.upload_date DESC
");
$books = [];
if ($stmt) {
    $stmt->bind_param("ii", $user_id, $user_id);
    $stmt->execute();
    $result = $stmt->get_result();
    while ($row = $result->fetch_assoc()) {
        $books[] = $row;
    }
    $stmt->close();
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Book Library</title>
        <link    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
            <link            rel="stylesheet"            href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.0/font/bootstrap-icons.css">
    <style>
        .container {
            max-width: 1800px;
            padding: 20px;
        }
        .books-grid {
```

```
    display: grid;
    grid-template-columns: repeat(4, minmax(250px, 1fr));
    gap: 20px;
    width: 100%;
}
@media (max-width: 1400px) {
    .books-grid {
        grid-template-columns: repeat(3, minmax(250px, 1fr));
    }

@media (max-width: 992px) {
    .books-grid {
        grid-template-columns: repeat(2, minmax(250px, 1fr));
    }
}
@media (max-width: 576px) {
    .books-grid {
        grid-template-columns: 1fr;
    }
}
.book-card {
    transition: all 0.3s ease;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
    height: 100%;
}
.book-card:hover {
    transform: translateY(-5px);
```

```
        box-shadow: 0 10px 20px rgba(0, 0, 0, 0.15);
    }
      .book-preview {
        height: 200px;
        background-color: #f8f9fa;
        display: flex;
        align-items: center;
        justify-content: center;
        border-bottom: 1px solid #dee2e6;
      }
      .book-icon {
        font-size: 3.5rem;
        color: #dc3545;
      }
      .book-title {
        display: -webkit-box;
        -webkit-line-clamp: 2;
        -webkit-box-orient: vertical;
        overflow: hidden;
        min-height: 3em;
        font-size: 1.1rem;
      }
      .action-btn {
        width: 40px;
        height: 40px;
        border-radius: 50% !important;
        padding: 0;
        display: inline-flex;
        align-items: center;
        justify-content: center;
      }
```

```
        .card-footer {
            padding: 0.75rem !important;
        }
    </style>
</head>
<body>
    <div class="container py-4">
    <a href="trash.php" class="btn btn-outline-secondary">
    <i class="bi bi-trash"></i> Trash
</a>
        <?php if (!empty($books)): ?>
            <div class="books-grid">
                <?php foreach ($books as $book):
                    $displayName = htmlspecialchars($book['title']);
                    $isFavorite = $book['is_favorite'] > 0;
                ?>
                    <div class="book-card">
                        <div class="book-preview">
                            <?php if ($book['file_type'] === 'application/pdf'): ?>
                                <i class="bi bi-file-earmark-pdf book-icon"></i>
                            <?php elseif ($book['file_type'] === 'application/epub+zip'): ?>
                                <i class="bi bi-journal-bookmark book-icon"></i>
                            <?php else: ?>
                                <i class="bi bi-book book-icon"></i>
                            <?php endif; ?>
                        </div>
                        <div class="card-body p-3">
                            <h5 class="card-title book-title" title="<?= $displayName ?>">
                                <?= $displayName ?>
                            </h5>
                            <p class="card-text text-muted small mb-2">
```

```
                <i class="bi bi-calendar"></i>
                <?= date('M d, Y', strtotime($book['upload_date'])) ?>
            </p>
            <p class="card-text text-muted small">
                <i class="bi bi-tag"></i>
                <?= ucfirst($book['category']) ?>
            </p>
        </div>
        <div class="card-footer bg-white">
            <div class="d-flex justify-content-between">
                <!-- View Button -->
                <a href="view_pdf.php?bk_id=<?= $book['bk_id'] ?>"
                   class="btn btn-sm btn-outline-primary action-btn"
                   title="View Book">
                    <i class="bi bi-eye"></i>
                </a>
                <!-- Favorite Button -->
                <button type="button"
                        class="btn btn-sm btn-outline-danger action-btn toggle-favorite"
                        data-bk-id="<?= $book['bk_id'] ?>"
                       title="<?= $isFavorite ? 'Remove from favorites' : 'Add to favorites'
                    <i class="bi bi-heart<?= $isFavorite ? '-fill text-danger' : '' ?>"></i>
                </button>

                <!-- Delete Button -->
                <form method="post" class="d-inline">
                    <input type="hidden" name="bk_id" value="<?= $book['bk_id'] ?>">
                    <button type="submit"
                            name="delete_book"
                            class="btn btn-sm btn-outline-secondary action-btn"
                            title="Delete Book"
```

```
                                onclick="return confirm('Are you sure you want to delete this
book?');">
                            <i class="bi bi-trash"></i>
                        </button>
                    </form>
                    <!-- Category Dropdown -->
                    <div class="dropdown d-inline">
                        <button class="btn btn-sm btn-outline-info action-btn dropdown-
toggle"
                            type="button"
                            data-bs-toggle="dropdown"
                            aria-expanded="false"
                            title="Change Category">
                            <i class="bi bi-tag"></i>
                        </button>
                        <ul class="dropdown-menu">
                            <li>
                                <form method="post">
                                <input type="hidden" name="bk_id" value="<?= $book['bk_id']
?>">
                                <button type="submit" name="update_category" value="study"
class="dropdown-item">
                                    <i class="bi bi-journal-bookmark"></i> Study
                                </button>
                                </form>
                            </li>
                            <li>
                                <form method="post">
                                <input type="hidden" name="bk_id" value="<?= $book['bk_id']
?>">
```

```
                                <button type="submit" name="update_category" value="work"
class="dropdown-item">
                                    <i class="bi bi-briefcase"></i> Work
                                </button>
                            </form>
                        </li>
                        <li>
                            <form method="post">
                            <input type="hidden" name="bk_id" value="<?= $book['bk_id']
?>">
                                                <button type="submit" name="update_category"
value="entertainment" class="dropdown-item">
                                    <i class="bi bi-joystick"></i> Entertainment
                                </button>
                            </form>
                        </li>
                    </ul>
                </div>
            </div>
        </div>
    <?php endforeach; ?>
    </div>
<?php else: ?>
    <div class="alert alert-info text-center py-4">
        <i class="bi bi-folder-x" style="font-size: 2rem;"></i>
        <h4 class="mt-3">No Books Found</h4>
            <p>You haven't added any books yet. Click the "Import Book" button to get
started.</p>
    </div>
<?php endif; ?>
```

```
 </div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script
>
   <script>
      document.addEventListener('DOMContentLoaded', function() {
         // Initialize tooltips
         var tooltipTriggerList = [].slice.call(document.querySelectorAll('[title]'));
         var tooltipList = tooltipTriggerList.map(function (tooltipTriggerEl) {
            return new bootstrap.Tooltip(tooltipTriggerEl);
         });
         // Handle favorite toggling via AJAX
         document.querySelectorAll('.toggle-favorite').forEach(button => {
            button.addEventListener('click', function() {
               const bkId = this.getAttribute('data-bk-id');
               const icon = this.querySelector('i');
               const isFavorite = icon.classList.contains('bi-heart-fill');
               fetch('book.php', {
                  method: 'POST',
                  headers: {
                     'Content-Type': 'application/x-www-form-urlencoded',
                  },
                  body: `ajax_toggle_favorite=1&bk_id=${bkId}`
               })
               .then(response => response.json())
               .then(data => {
                  if (data.success) {
                     icon.classList.toggle('bi-heart');
                     icon.classList.toggle('bi-heart-fill');
                     icon.classList.toggle('text-danger');
```

```
                    // Update tooltip title
                        const newTitle = data.is_favorite ? 'Remove from favorites' : 'Add to
favorites';
                    button.setAttribute('title', newTitle);
                            bootstrap.Tooltip.getInstance(button).setContent({'.tooltip-inner':
newTitle});
                }
</script>
</body>
</html>
<?php
$conn->close();
?>
```

Import.php

```
<?php
// Enable error reporting for debugging
error_reporting(E_ALL);
ini_set('display_errors', 1);
// File upload handling
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['file'])) {
    $allowed_types = ['pdf', 'epub', 'mobi'];
    $file_name = $_FILES['file']['name'];
    $file_tmp = $_FILES['file']['tmp_name'];
    $file_ext = strtolower(pathinfo($file_name, PATHINFO_EXTENSION));
    // Validate file type
    if (!in_array($file_ext, $allowed_types)) {
        $success_message = "Only PDF, EPUB, and MOBI files allowed.";
    }
    // Validate file size
    elseif ($_FILES['file']['size'] > 20000000) { // 20MB max
```

```
    $success_message = "File too large (max 20MB).";
}
// Check for upload errors
elseif ($_FILES['file']['error'] !== UPLOAD_ERR_OK) {
    $success_message = "Upload error: " . $_FILES['file']['error'];
}
// Proceed with upload if all validations pass
else {
    // Use absolute path for upload directory
    $upload_dir = __DIR__ . '/uploads/';
    // Create upload directory if it doesn't exist
    if (!file_exists($upload_dir)) {
        if (!mkdir($upload_dir, 0777, true)) {
            $success_message = "Failed to create upload directory.";
    }
    // Check if directory is writable
    if (!is_writable($upload_dir)) {
        $success_message = "Upload directory is not writable.";
    } else {
        // Generate unique filename to prevent conflicts
        $new_filename = uniqid() . '_' . basename($file_name);
        $destination = $upload_dir . $new_filename;
        // Move the uploaded file
        if (move_uploaded_file($file_tmp, $destination)) {
            // Connect to database
            $link = mysqli_connect("localhost", "root", "", "mybooklibrary");
            if (mysqli_connect_errno()) {
                $success_message = "Database connection failed: " . mysqli_connect_error();
            } else {
                // Set a default user ID (replace this with your actual user authentication)
                $user_id = 1; // CHANGE THIS TO YOUR ACTUAL USER ID SYSTEM
```

```
                // Prepare and execute SQL statement
                $stmt = mysqli_prepare($link, "INSERT INTO books (user_id, title, file_name,
upload_date) VALUES (?, ?, ?, ?)");
                $upload_date = date('Y-m-d H:i:s');
                 mysqli_stmt_bind_param($stmt, "isss", $user_id, $file_name, $new_filename,
$upload_date);
                if (mysqli_stmt_execute($stmt)) {
                    $success_message = "File uploaded successfully!";
                } else {
                    $success_message = "Error saving file to database: " . mysqli_error($link);
                    // Remove the uploaded file if database insert failed
                    unlink($destination);
                }
                mysqli_stmt_close($stmt);
                mysqli_close($link);
            }
        } else {
            $success_message = "Error moving uploaded file. Check permissions.";
        }
?>
<!DOCTYPE html>
<html>
<head>
    <title>Import Files</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
 <linkrel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
 <linkhref="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&displa
y=swap" rel="stylesheet">
    <style>
```

```
body {
    font-family: 'Roboto', sans-serif;
    background: linear-gradient(to right, rgb(199, 118, 224), #cfdef3);
    color: #333;
    margin: 0;
    padding: 0;
}
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
.card {
    background: #fff;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    padding: 30px;
    width: 100%;
    max-width: 600px;
}
.btn-library {
    display: block;
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    background-color: #5cb85c;
    color: white;
    text-align: center;
    border-radius: 5px;
    text-decoration: none;
```

```
            font-weight: 500;
            transition: background-color 0.3s;
        }
        .btn-library:hover {
            background-color: #4cae4c;
            color: white;
            text-decoration: none;
        }
        .button-container {
            display: flex;
            justify-content: space-between;
            margin-top: 20px;
        }
        .form-group {
            margin-bottom: 20px;
        }
        .alert {
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="card">
            <h2 style="margin-top: 0;">Import Files</h2>
            <a href="books.php" class="btn-library">View My Library</a>
            <?php if (!empty($success_message)): ?>
<div class="alert alert-<?php echo strpos($success_message, 'successfully') !== false ? 'success' : 'danger'; ?>">
                <?php echo htmlspecialchars($success_message); ?>
            </div>
```

```
        <?php endif; ?>
        <form method="post" enctype="multipart/form-data">
            <div class="form-group">
                <label for="fileInput">Choose File (PDF, EPUB, MOBI)</label>
                        <input type="file" name="file" id="fileInput" class="form-control"
accept=".pdf,.epub,.mobi" required>
                <small class="form-text text-muted">Maximum file size: 20MB</small>
            </div>
            <div class="button-container">
                <a href="homepage.php" class="btn btn-secondary">Back</a>
                <button type="submit" class="btn btn-primary">Import File</button>
            </div>
        </form>
    </div>
</body>
</html>
```

Viewpdf.php

```
<?php
// view_pdf.php
require 'db.php';
// Get book ID from URL
$book_id = isset($_GET['bk_id']) ? (int)$_GET['bk_id'] : 0;
// Get book data from database
$stmt = $conn->prepare("SELECT file_name FROM books WHERE bk_id = ?");
$stmt->bind_param("i", $book_id);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows === 0) {
    die("Book not found in database");
}
```

```
$book = $result->fetch_assoc();
$file_path = __DIR__ . '/uploads/' . $book['file_name'];
// Check if file exists
if (!file_exists($file_path)) {
    die("File not found on server");
}
// Set proper headers for PDF
header("Content-Type: application/pdf");
header("Content-Length: " . filesize($file_path));
header("Content-Disposition: inline; filename=\"" . basename($book['file_name']) . "\"");
// Output the file
readfile($file_path);
exit;
?>
```

Db.php

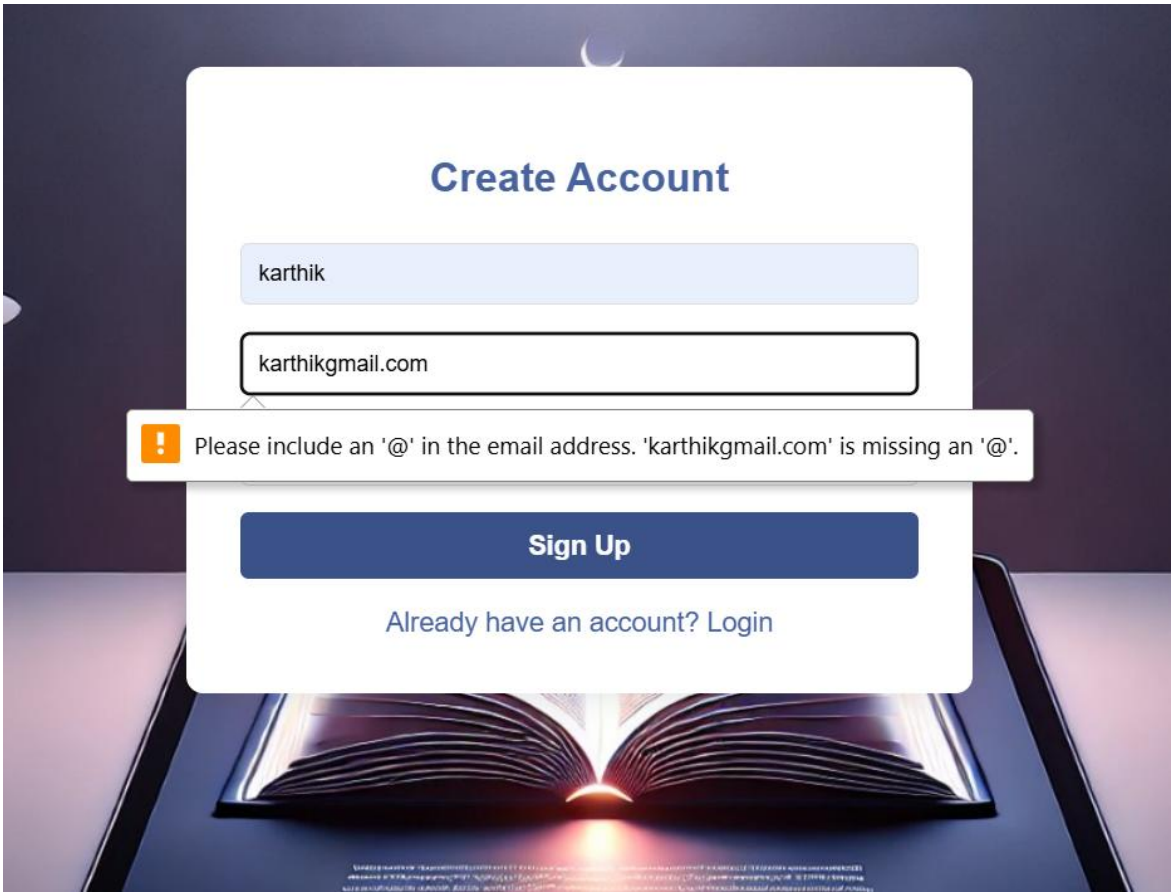
```
<?php
// db_connect.php - Simple MySQLi connection file
// Database configuration
$servername = "localhost";
$username = "root";
$password = "";
$database = "mybooklibrary";
// Create connection
$conn = new mysqli($servername, $username, $password, $database);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// No closing PHP tag to prevent accidental output
```

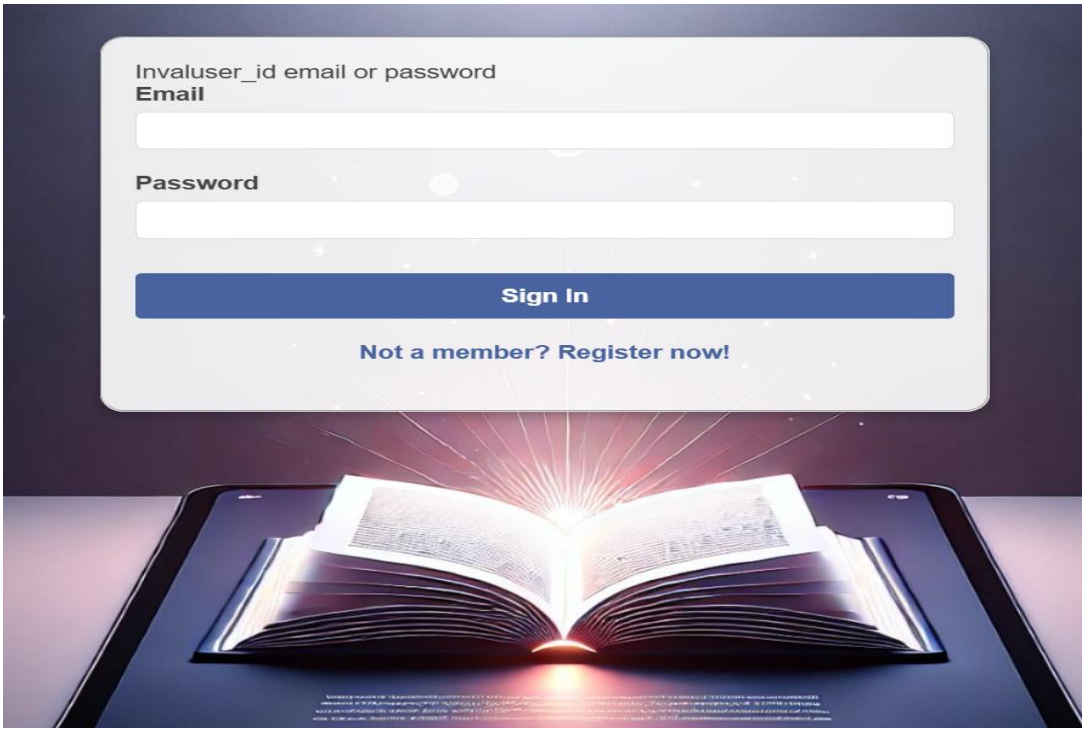
7. TESTING

1. Unit Testing

Objective: Test individual functions or components to ensure they work as expected.

- **Example Tests**:
    - **Authentication Module**: Verify login and registration processes (e.g., valid/invalid credentials, password validation).
    - **Book Uploading**: Check file format validation (e.g., only PDF or ePub allowed).
    - **Annotation System**: Ensure text highlighting and note-saving functionality works correctly.





2. Integration Testing

Objective: Test the interaction between modules to confirm they work together seamlessly.

- **Example Tests**:
    - Verify that uploaded books appear correctly in the user library.
    - Ensure annotations made during reading are linked to the correct book and saved in the database.
    - Test reading progress tracking and its reflection on the dashboard.

3. System Testing

Objective: Test the entire application to ensure it meets all functional requirements.

- **Example Tests**:
    - User navigation across modules (library, reading view, dashboard).
    - Uploading multiple books and verifying system performance.
    - Checking the functionality of audio integration and playback controls.

4. User Interface (UI) Testing

Objective: Ensure the design and layout are user-friendly and visually consistent.

- **Example Tests**:
    - Verify responsive design across various devices (desktop, tablet, mobile).
    - Test button placements, readability of text, and accessibility features (e.g., contrast and font size).
    - Check for broken links and navigation consistency.

5. Performance Testing

Objective: Evaluate the system's speed, scalability, and stability.

- **Example Tests**:
    - Test load time for large book uploads.
    - Simulate multiple users accessing the platform simultaneously to check for performance degradation.
    - Measure the speed of retrieving user dashboards and analytics.

6. Security Testing

Objective: Identify vulnerabilities to ensure data integrity and user privacy.

- **Example Tests**:
    - Test SQL injection and cross-site scripting (XSS) protection.
    - Verify password encryption and secure login mechanisms.
    - Ensure user sessions expire after inactivity to prevent unauthorized access.

8. IMPLEMENTATION

Implementation is one of the most crucial phases in software development, where the design is translated into a fully working application through programming, integration of components, testing, and deployment. For the My Books Library project, this phase involved the development and deployment of a web-based digital reading platform, enabling users to interact with a comprehensive e-book ecosystem.

The system was developed using PHP for the server-side logic, MySQL as the relational database management system, and HTML, CSS, and JavaScript for the front-end development. The local development and testing were handled using XAMPP and phpMyAdmin.

The implementation process followed the Agile Software Development Life Cycle (SDLC), enabling iterative development, feedback incorporation, and continuous improvements throughout the development process.

9.1 Development Process

The development of My Books Library was carried out in clearly defined stages. Each step focused on converting functional requirements into robust, user-friendly, and scalable modules.

1. Requirement Gathering

The first step was identifying the specific needs of the end users such as students, researchers, and general readers. The key features finalized during this phase included:

- In-app annotations and notes
- Progress tracking and analytics
- Audiobook integration
- User authentication and profile management
- Cloud-like book access (via local file uploads)

- AI-based book recommendation (future scope)

2. Design & Prototyping

Wireframes and mockups were created using tools like **Figma** to visualize the application's look and feel before development began. The designs focused on:

- Intuitive UI/UX with responsive layout
- A reader interface with highlighting and annotation options
- Simple navigation menus
- Dashboard and analytics layout for user statistics

3. Front-End Development

The front-end was built using:

- **HTML5** for content structure
- **CSS3** for styling, responsiveness, and visual components
- **JavaScript** for interactive elements like progress bars, note popups, dynamic recommendations, and animations

Front-end components included:

- Login and registration pages
- Home/dashboard with visual analytics
- Book reading interface with annotation tools
- Audiobook player with interactive controls
- Notes panel and personalized recommendation section

4. Back-End Development

The back-end logic was developed using **PHP**, a widely used scripting language suitable for handling server-side logic, file management, and integration with databases. Key responsibilities of the back-end included:

- Handling user authentication (login, register, logout)
- Managing session data and user preferences
- Uploading and storing books (PDF, EPUB, TXT)
- Handling annotations and storing them in a relational format
- Generating progress analytics from reading behavior

Database:

- The system uses **MySQL** as the RDBMS.
- **phpMyAdmin** was used for database design and management during development.
- Tables included users, books, annotations, reading_sessions, notes, and recommendations.

5. Feature Integration

Several core and auxiliary features were integrated during this phase:

- **Annotation System**: Enables users to highlight text, save notes, and categorize them.
- **Progress Tracker**: Tracks number of pages read, total time spent, and visual progress indicators.
- **Audio Support**: Though advanced audio conversion is a future scope, audio integration through pre-recorded files and basic playback was implemented.
- **Recommendation Module**: Suggests books based on tags, categories, and past reads using basic filtering logic in PHP/MySQL.

Each feature was integrated with the existing PHP routes and functions, using session management for personalization and maintaining a smooth flow between front-end and back-end modules.

6. Testing & Debugging

Testing was conducted across multiple phases:

- **Unit Testing**: Each PHP function was tested individually to ensure expected outputs.
- **Integration Testing**: Interaction between front-end inputs and back-end outputs was validated.
- **User Acceptance Testing (UAT)**: Conducted by potential users to gather feedback and improve UX.
- **Cross-browser Testing**: Ensured that the app functions consistently on Chrome, Firefox, Edge, etc.
- **Responsive Testing**: The layout and components were tested for responsiveness across desktop and mobile screens.

7. Deployment

Since the project was developed locally using XAMPP, the deployment was first done in a **localhost environment**. Once tested successfully, the final deployment process can involve:

- **Hosting using shared or cloud-based hosting** platforms supporting PHP and MySQL.
- Configuration of a custom domain (e.g., www.mybookslibrary.com).
- Database export/import via **phpMyAdmin** for deployment.
- FTP or Git for code uploads and version control.

A web-based deployment ensures cross-platform access for users across mobile, tablet, and desktop.

8. Challenges Faced During Implementation

- **File Handling**: Handling and displaying various file formats (PDF, EPUB, TXT) in-browser required third-party libraries and proper security handling.
- **Audio Integration**: Due to limitations in direct text-to-speech from PHP, the initial phase included static audio file support.
- **Session Management**: Maintaining login states and reading progress accurately across sessions required careful session and cookie management in PHP.
- **User Data Privacy**: Ensuring that uploaded files and notes remained secure and accessible only to authorized users.

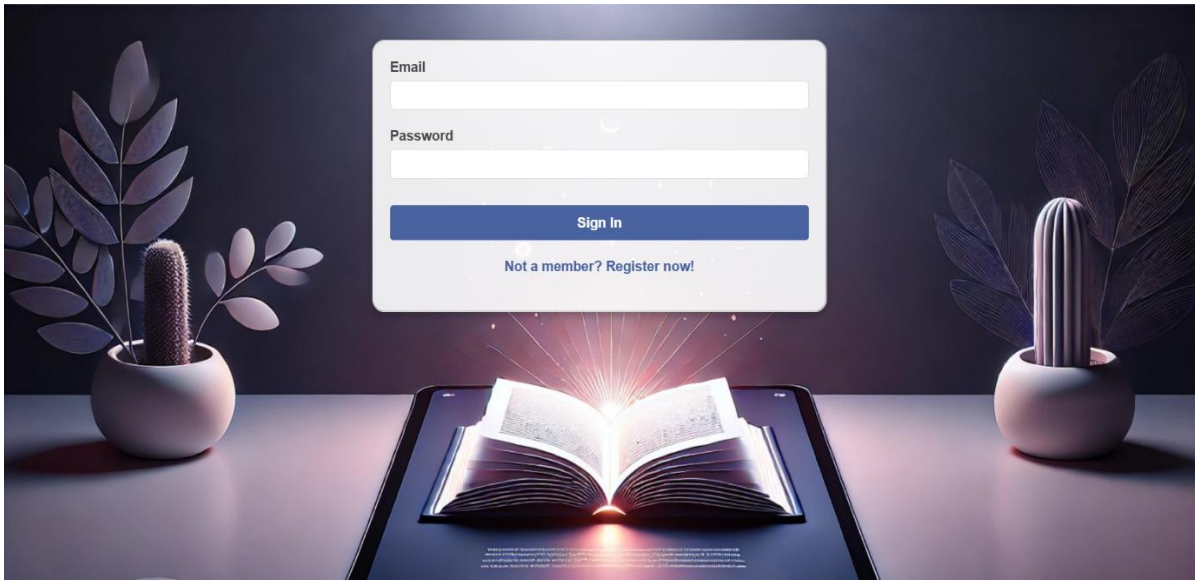
9. Tools and Environment

- **Languages**: PHP, JavaScript, HTML, CSS
- **Database**: MySQL
- **Development Environment**: Visual Studio Code
- **Server**: Apache (via XAMPP)
- **Database GUI**: phpMyAdmin
- **Version Control**: Git (optional for backup and sharing)

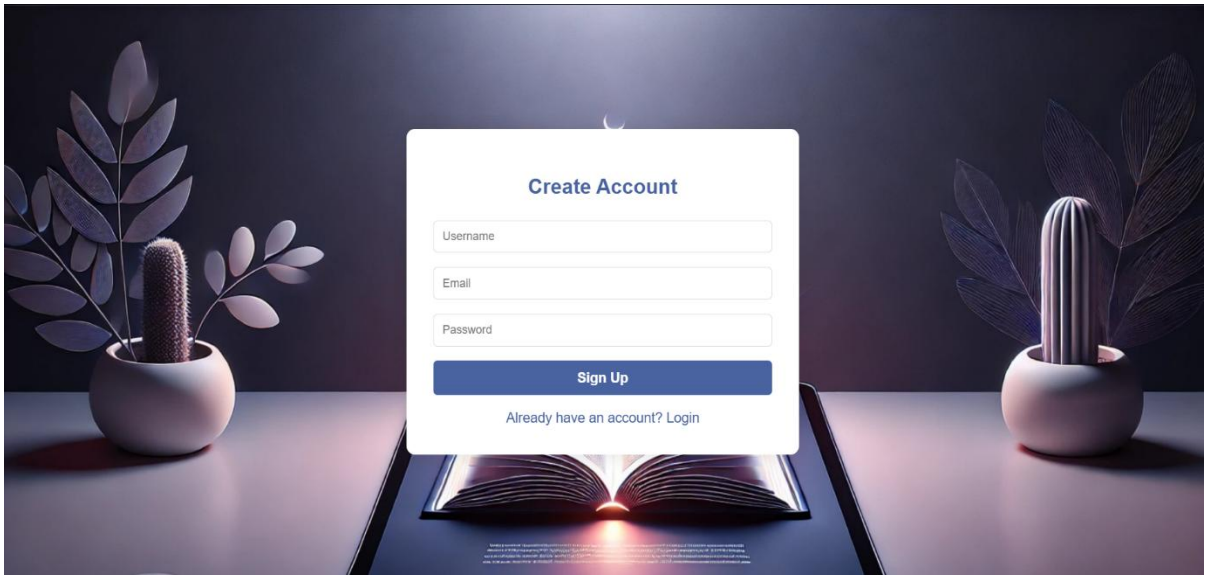
The implementation of *My Books Library* was a structured and multi-phase process, ensuring the delivery of a robust, efficient, and user-centric reading platform. Through a combination of reliable web technologies (PHP, MySQL, HTML/CSS/JS), proper architecture, and iterative development, the project successfully meets the modern demands of digital reading. With features like notes, annotations, audio playback, and visual dashboards, the implementation phase brought the original vision of the platform into a working reality—ready to be further scaled and enhanced.

9.SCREENSHOTS

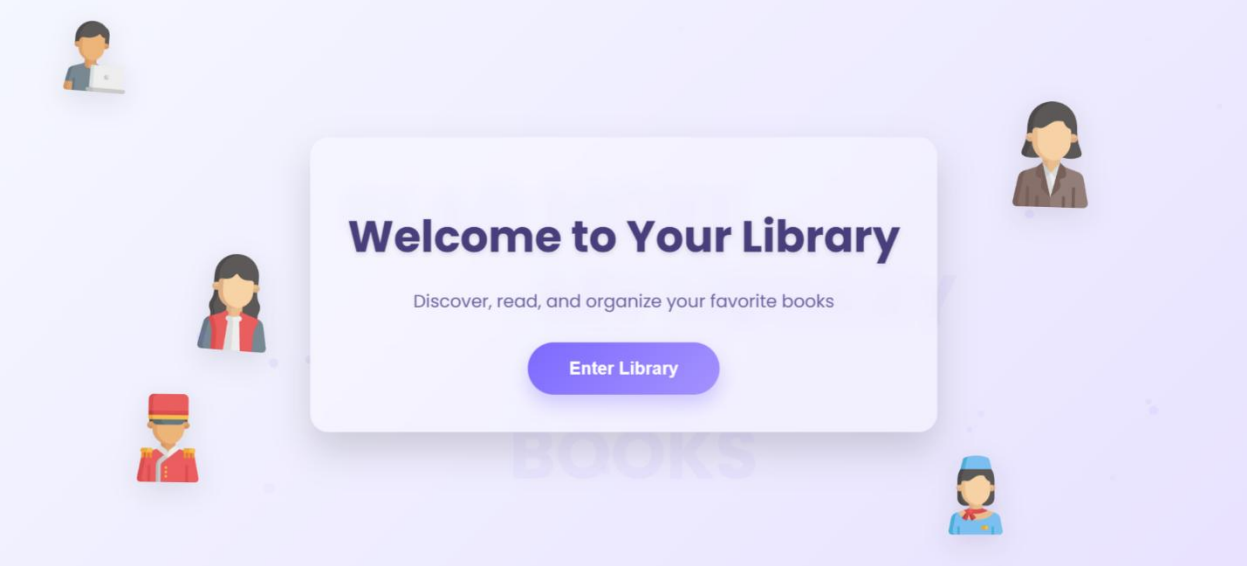
Sign in page



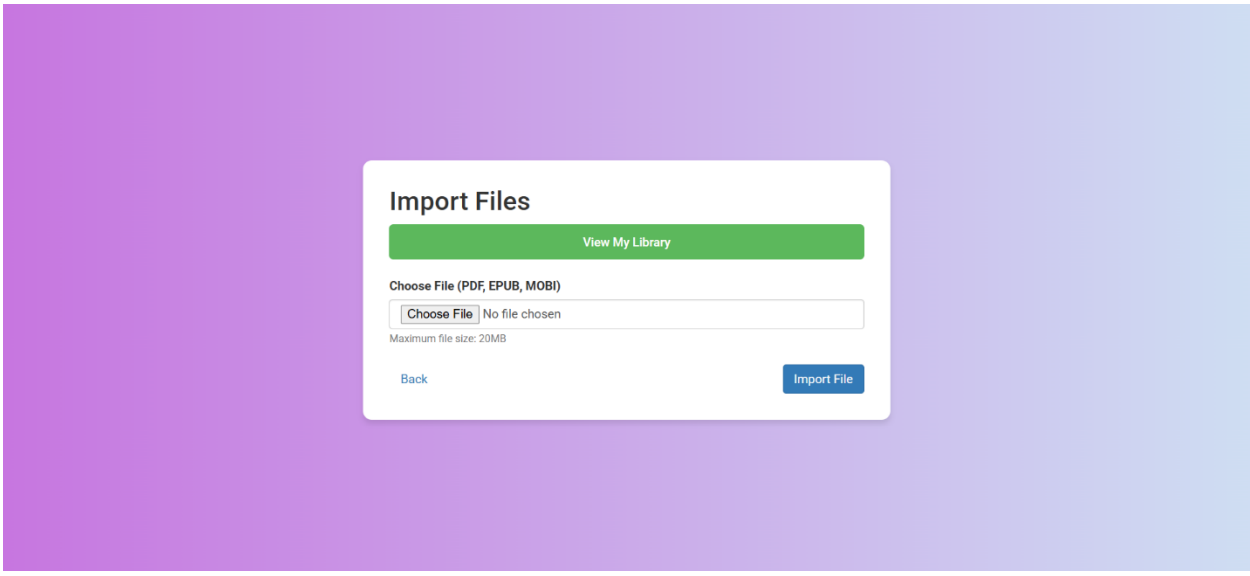
Sign up page



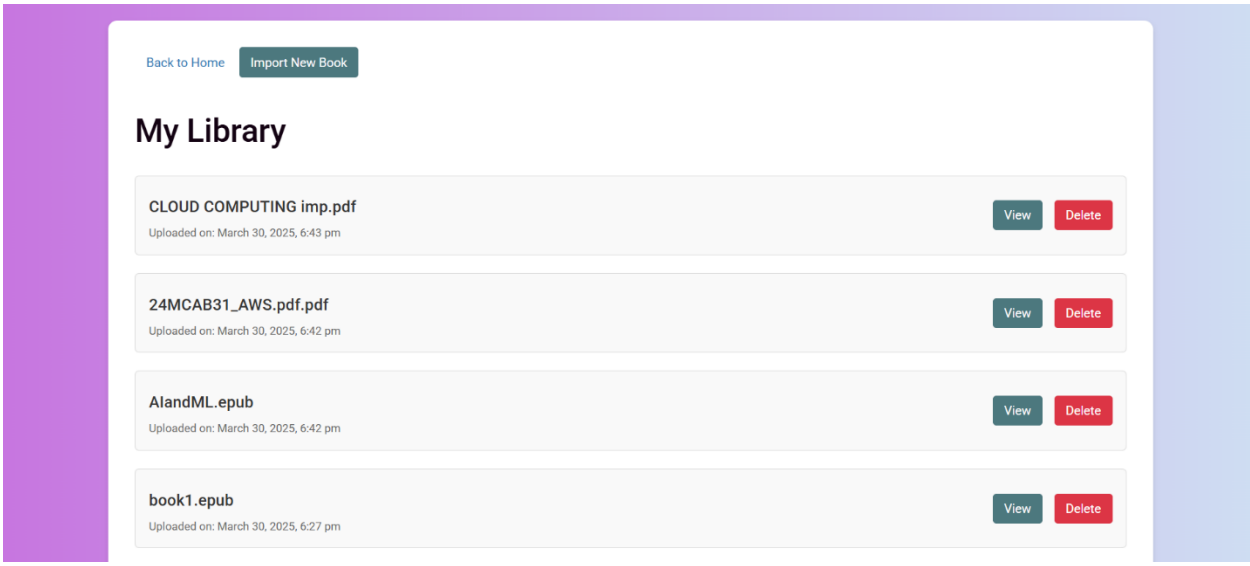
Dashboard page



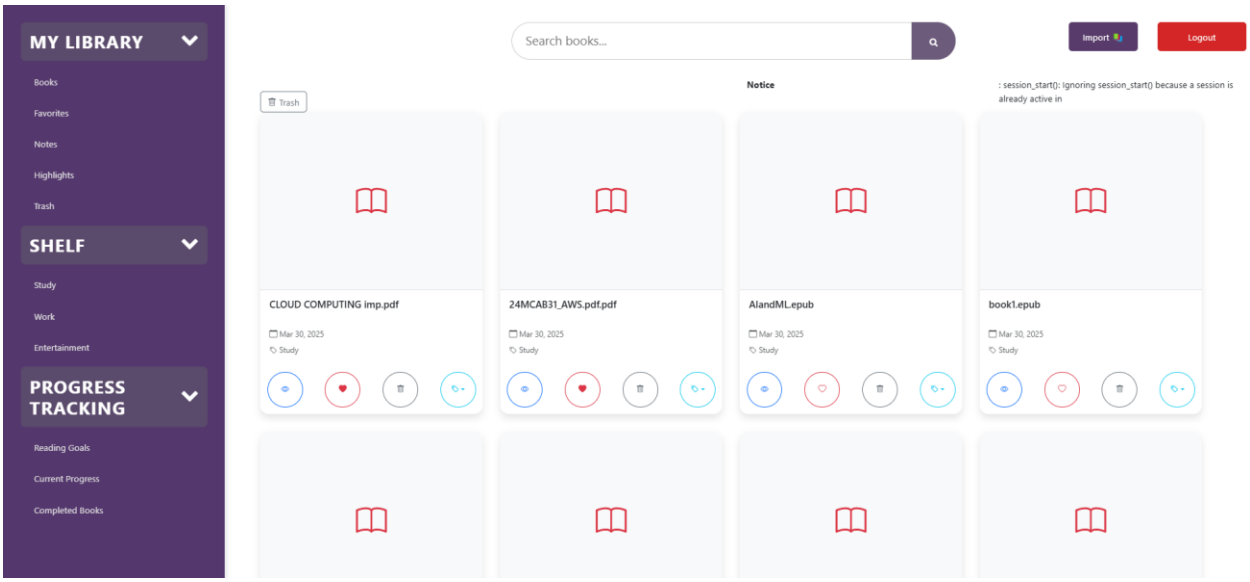
Import page



Library page



Home page



10. CONCLUSION

The *My Books Library* project revolutionizes digital reading by blending traditional book features with modern technology. Designed for students, educators, researchers, and book enthusiasts, it offers a seamless, interactive platform for enhanced reading experiences. Beyond basic e-reader functions, it incorporates features like in-app note-taking, progress tracking, personalized recommendations, and audio support, creating a highly engaging and educational experience.

Built using PHP, MySQL, HTML, CSS, and JavaScript, the system ensures a secure, dynamic, and responsive platform. Developed iteratively using Agile methodology, each sprint focused on specific modules like user authentication, progress analytics, and annotations. Features such as categorized notes, a personalized dashboard, and audio integration set it apart, catering to diverse user needs, including auditory learners and visually impaired individuals.

The project prioritizes a user-friendly interface, secure data management, and future scalability. Planned enhancements include AI-driven recommendations, cloud synchronization, and social features like book clubs. *My Books Library* demonstrates how thoughtful software design can redefine reading, evolving into a comprehensive learning companion and knowledge hub for the digital age.

11. BIBLIOGRAPHY

- **OpenAI – ChatGPT**: Utilized for assistance in generating code, logic design, database schema creation, and refining documentation during the development of the project.
  https://chat.openai.com
- **W3Schools**: Referenced for examples and syntax related to HTML, CSS, JavaScript, PHP, and MySQL to implement frontend and backend functionalities.
  https://www.w3schools.com
- **MDN Web Docs**: Used for understanding advanced JavaScript concepts, DOM manipulation techniques, and ensuring browser compatibility for the frontend interface.
  https://developer.mozilla.org
- **Stack Overflow**: Consulted for debugging common errors in PHP and MySQL, as well as community-driven solutions to optimize backend operations.
  https://stackoverflow.com
- **Bootstrap Official Documentation**: Leveraged for designing a responsive user interface and implementing UI components like navigation bars, cards, and pagination.
  https://getbootstrap.com
- **GeeksforGeeks**: Explored detailed tutorials on PHP and MySQL integration, as well as best practices for CRUD operations.
  https://www.geeksforgeeks.org
- **PHP.net Manual**: Consulted for in-depth understanding of PHP functions, error handling, and secure data handling methods.
  https://www.php.net
- **SQL Tutorial**: Used for designing and optimizing queries for the MySQL database, including joins, indexes, and normalization techniques.
  https://www.sqltutorial.org