Project Delta

Professor Bonaventure

CSC 131

30 November 2025

Section 1: Software Specification

1.1 Project Overview

Project Delta Library Management is a web-based software application specifically designed to modernize and improve library database operations. This system consists of core functionalities, including a book catalog browser, user account management, book reservation, membership, enrollment, book loaning, and an admin dashboard used to track and manage books, users, and any associated fees. The overall goal of this project was to create a user-friendly, responsive, and fully functioning aesthetic platform allowing book enthusiasts and admins to browse and perform their necessary tasks. This project strictly follows the Waterfall model, which follows the five stages of software development known as specification, design, implementation, testing, and evolution.

1.2 Stakeholders

List of people who use/manage the system:

- **End Users / Members -** browse books, reserve books, return books, and manage a membership account
- **Library Admin -** staff responsible for managing the collection of books, users, loans, and overdue fees.

- **Development Team -** Project Delta development team, responsible for adding/removing functionalities, alpha testing

1.3 Functional Requirements

The mandatory requirements system must perform

1.3.1 User-Side Functionalities

- The system allows the user to create an account with a name, email, and other required personal information
- System allows users to log in and log out of the system securely
- Users must be able to browse the book catalog and view book summaries, author, and other important details
- Users must be able to search books by title, author, and/or category
- Users must be able to reserve/loan books and return them after borrowing
- Users must be able to track borrowing history, active loans, and due dates
- Users must have access to a wishlist where they add/remove desired books
- The system must send a notification reminder for overdue books
- Users must be able to contact the team through the Contact Page form

1.3.2 Admin-Side Functionalities

- Admin must be able to log in and log out of the administrative dashboard
- Admin must be able to add new books to the system
- Admin must be able to modify and delta books
- Admin must be able to view any/all registered users and their user status (membership or no membership)
- Admin must be able to manage book loans, including due dates and overdue fees

- Admin must be able to view and manage user reservations

1.4 Non-Functional Requirements

Quality of Software

1.4.1 Performance Requirements

- Pages should load in under ten seconds, about the network connection
- Search functionality must return results with minimal delay

1.4.2 Usability Requirements

- Interface must be consistent in styling, typography, and spacing throughout all pages
- Fonts must be readable and accessible
- Icons and buttons are meant to be clearly labeled with no point of confusion
- The application itself must be easy to navigate and have a sensible approach to the UI

1.4.3 Accessibility Requirements

- Users must be able to resize and adjust the window size
- Application must be accessible with keyboard navigation
- Colors must have high contrast for visibility purposes

1.4.4 Compatibility Requirements

- Must work on Windows, macOS, and Linux systems
- Must be compatible with modern browsers such as Chrome, Firefox, Safari, and Edge
- Must be responsive relatively for laptops and desktops

1.4.5 Security Requirements

- User login must be secure password handling
- Sensitive data must not be leaked in the client-side code
- System must validate all input fields (Contact form, login form, etc.)

1.5 Constraints

- The project must follow the Waterfall Model strictly
- All pages must have uniform styling, background colors, fonts, and themes
- No external commercial URLS
- Final submission must be able to run on Windows PC and Mac
- Avoid grammatical and spelling errors
- Must include About, Credits, Contact, and Home pages as required
- Contact form must send an email to the inbox
- Web-based application must follow React/HTML structure

1.6 Assumptions

- Users have an active internet connection
- Admin understands the basic computer operations
- System will run on a browser with JavaScript enabled
- Team will collaborate using GitHub branches

Section 2: System Architecture

2.1 Design Phase

- Focuses on structural, visual, and architectural planning of Project Delta
- Includes UI planning, component structure, and diagram model (UML, use-case, etc.)

2.1.1

- Frontend Framework: Next.js
- Styling: Custom CSS via global.css, layout components, responsive design language

- Navigation: Next.js App router with multiple pages, including Home, About Credit, and Contact
- Component Structure: Reusable head and footer also navigation bar, and layout wrapper
- Backend and Database: Backend consists of SQL and backend API integration with Node.js/JavaScript

Allows for maintainable, scalable, and easy updates to occur with this structure

2.1.2 User Interface/ User Experience Design

Ui design is to be easy to navigate and clean, professional, responsive, and visually consistent.

- Center-focused layout with spacing for readability
- Minimalist header with navigation links to all required pages
- Footer containing all social media icons
- Consistent typography across all pages
- High-contrast color palette for accessibility
- Aesthetic homepage for user experience

All pages use a shared layout with layout.tsx or similar pages under the folder ot maintain uniformity, including:

- Persistent navigation
- Global footer
- Shared global CSS

2.1.3 Page-Level Design

Home page

- Introduces the purpose of the application
- Includes buttons/accessible items to scroll through other pages

- Uses clean typography and centered content

About Page

- List each team member
- Includes education, skills, awards, and experience
- Clean profile layout and text

Credits Page

- Displays team photos
- List tasks/roles each member has done
- Designed using a card/grid layout

Contact Page

- Contains required input fields: name, email, phone, message
- Submit button allows for email functionality for the entire team
- Syteld for clarity and accessibility for the user

Footer

- Contains social media links to project accounts
- Appears on relevant pages, consistent with assignment requirements

2.1.4 System Modeling (High-Level Explanation)

- Use-Case Diagram: Shows interactions between users, admins, and the system
- High-Level Class Diagram: Shows main entities such as User, Book, Reservation, Loan, Admin
- Detailed Class Diagram, either pseudo code or algorithm, showing the process of the application

2.2.1 Technology Stack

- Front-end: Next.js, Page routing, UI rendering
- Styling: CSS, layout, colors, typography
- Icons: React Icons/ SVGS footer and UI icons
- Version Control: GitHub, branching, merging, collaboration
- Hosting (Optional), Vercel, Notion, Deployment for live access

2.2.2 Project Folder Structure

- Clear separation of pages
- Reusable components
- Clean code
- Easy collaboration with teammates

2.2.3 Layout and Global Styling

Layout.tsx is the skeleton of the entire app:

- Navigation bar
- Footer
- Page container
- Typography and spacing rules

Global.css provides:

- Base font styles
- Page background
- Link styling
- Grid Layout utilities
- Universal margin and padding rules

2.2.4 Page Implementation

Home Page

- Uses HTML inside React component
- Styled with global CSS to match aesthetic
- Includes hero text and imagery

About Page

- Structured using lists, headings, and a grid layout
- Present each member's biography

Credits Page

- Display team roles and tasks
- Uses a grid layout for team pictures
- Emphasizes readability and clean card layout

Contact Page

- Uses controlled input comments for data
- Validations prevent empty submissions
- Submissions will be configured to send emails to the team

2.2.5 Version Control Implementation

- Team uses GitHub branches main and ui-update for running the program
- Commits include: layout changes, styling updates, new pages, and component creation
- Team members pull the latest changes before working to avoid merge conflictions

2.2.6 Challenge and Solutions

- Styling inconsistencies: create global.css for universal styling
- Navigation issues: Centerize header component used across all pages
- Layout Spacing problems: implement container classes for standard padding
- Combine teammate work: Git branches are used to manage contributions

Section 3: Software Validation

3.1 Testing (PLACEHOLDERS)

- Unit tests
- Integration tests
- System tests
- Acceptance tests
- What tools used
- Test scenarios + expected outcomes
- Screenshots of testing (optional)

3.2 Deployment

Installation Prerequisites

Before running the project, ensure the following software is installed on your system:

1. Node.js & npm
    - Required to run both the frontend and backend scripts. Check versions using node -v and npm -v
2. IDE / Code Editor (optional)
    - Recommended for development or inspection of source code. Examples include Visual Studio Code or WebStorm
3. Database
    - MySQL server is required to store and manage library data.

4. Web Browser
    - A modern web browser (Chrome, Firefox, etc) is required to access the frontend interface

Steps to Launch

To run the system locally, follow these steps:

1. Clone the project repository:
    - Clone the repository from GitHub
        - git clone https://github.com/Mystical123/Library-Management-System-.git
    - Change the directory into the project folder
        - cd Library-Management-System-
2. Install dependencies
    - For backend:
        - cd backend
        - npm install
    - For frontend:
        - cd frontend
        - npm install
    - Start the backend server:
        - cd backend
        - npm start
    - Start the frontend server:
        - cd frontend

- npm run dev

3. Database Setup Instructions

   The library management system requires a preconfigured database

   1. Open your MySQL
   2. Create a new database
      a. CREATE DATABASE book_catalog;
   3. Import the provided database file (book_catalog.sql) into your database:
      a. mysql -u root -p book_catalog < book_catalog.sql
   4. Ensure the backend server is configured to connect to the correct database
      a. Note: the provided book_catalog.sql contains all required tables and initial data for the system to function correctly

4. Environment Variables / Configuration

   The backend server currently uses hardcoded database credentials in its connection setup.

   The following settings must match your local MySQL setup:

Setting	**Value**	**Notes**
DB_HOST	localhost	Database server host
DB_USER	root	Database username
DB_PASSWORD	(blank)	Leave empty if no password is set
DB_NAME	book_catalog	Database name

Section 4: Software Evolution

Project Delta Library Management is designed to be extensible and open to future improvements. The following enhancements are proposed to increase functionality, accessibility, and user experience:

1. Mobile Version
    - Develop a mobile application for iOS and Android platforms to allow users to browse books, reserve items, and manage accounts from their mobile devices.
2. Payment System
    - Implement a secure payment system to handle membership dues and fines for overdue books, allowing users to pay directly through the application.
3. Automated Overdue Payment Reminders
    - Introduce an automated system to notify users of overdue books and payments, or fines automatically.
4. Improved UI Accessibility
    - Enhance the user interface with features such as keyboard navigation, resizable fonts, high-contrast themes, and accessible layout designs.
5. Environment Configuration for Safety
    - Move sensitive variables, such as database credentials, API keys, and secret tokens, from being hardcoded in the SQL routes to a dedicated .env file. This would improve security, simplify configuration management, and allow safer deployment.
6. Enhanced Contact Form Handling
    - Currently, the contact form relies on EmailJS for email delivery. Migrating this functionality to Supabase would centralize data management, enhance reliability and provide better integration with user and database records.
7. Voice-Enabled Book Descriptions
    - Add audio descriptions of books to improve accessibility for visually impaired users and provide a richer browsing experience.
8. Activity Dashboard for Users
    - Develop a user dashboard that displays personalized statistics, recently viewed books, reading history, and quick links to key features to enhance navigation and engagement.
9. Dark Mode and Theme Customization
    - Add support for dark mode and additional customizable themes to improve visual comfort and accessibility for users across different lighting environments.
10. Improved Wishlist Functionality

- Expand the wishlist feature to allow users to organize books into multiple custom lists and optionally share them with friends or other library members.

11. Personalized Recommendation Engine
    - Implement a recommendation system that analyzes a user's browsing and borrowing history to suggest books tailored to their reading preferences.
12. Book Review and Rating System
    - Add the ability for users to leave reviews and ratings on books, allowing the community to share feedback and helping others discover popular or highly rated books.
13. Offline Browsing Support
    - Enable limited offline functionality so users can view previously loaded pages, cached book details, and their wishlist even without an active internet connection.
14. Enhanced Error Handling and System Feedback
    - Improve the application by adding user-friendly error messages, loading indicators, form validation feedback, and system alerts so users receive clear guidance when actions fail, succeed, or require additional input.
15. Admin Analytics Dashboard
    - Introduce an analytics dashboard for administrators that displays useful statistics such as the most borrowed books, active users, overdue trends, reservation fulfilment rates, and inventory summaries.

These planned enhancements aim to make Project Delta more user-friendly, secure, and accessible while increasing its reach and functionality across multiple platforms.