CSE 341 Final project Proposal

# General Info

NAMES

Clementine Iradukunda

Farai Dandara

Cristobal Alfonso Henriquez

Joseph Uchenna Israel  
Andrew Obinna Onyekwere

Application name

Library management system

Contents:

Contents

[General Info](#_gjdgxs) **1**

[Application Info](#_oxlmg6u5hyfx) **2**

[What will the API do?](#_1fob9te) 2

[How will your API utilize a login system?](#_tyjcwt) 2

[What database will you use?](#_2et92p0) 2

[How will the data be stored in your database?](#_2s8eyo1) 2

[How would a frontend be able to manage authentication state based on the data you provide?](#_3dy6vkm) 2

[What pieces of data in your app will need to be secured? How will you demonstrate web security principles in the development of this app?](#_1t3h5sf) 2

[What file structure and program architecture will you use for this project (how will you organize your node project)? Why?](#_4d34og8) 2

[What are potential stretch challenges that you could implement to go above and beyond?](#_17dp8vu) 2

[**API Endpoint Planning**](#_mral43ww1pvt) **2**

[Project Scheduling and Delegation](#_1ksv4uv) **3**

[How will you divide up work in your team to ensure the following tasks all get completed?](#_3rdcrjn) 3

[Potential Risks and Risk Mitigation Techniques](#_26in1rg) **4**

[What are the risks involved with you being able to finish this project in a timely manner?](#_lnxbz9) 4

[How will you mitigate or overcome these risks?](#_35nkun2) 4

# Application Info

## What will the API do?

The API will handle the backend operations for a Library Management System. It will manage key entities such as Users (library members), Books, Lending Records, and Staff. The API will enable functionalities such as creating and managing users, adding and searching for books, keeping track of lending transactions, and storing staff data. It will also support login, authentication, and user management, ensuring secure access to sensitive data.

## How will your API utilize a login system?

The API will utilize a login system based on authentication tokens (such as JWT or OAuth). Upon successful login, the system will issue a token that will be used for subsequent requests to verify the identity and permissions of the user. For example, library members and staff will be authenticated before performing any CRUD operations on books, lending records, or staff data. User credentials (username, password) will be securely stored, and passwords will be hashed to ensure security.

## What database will you use?

We will use A MongoDB as the database for this API

## How will the data be stored in your database?

The data will be stored in collections in MongoDB.

* **Users**: Stores user details like name, email, membership status, etc.
* **Books**: Stores book details such as title, author, ISBN, availability status, etc.
* **Lending Records**: Stores records of books lent out to users, including dates and due dates.
* **Staff**: Stores information about library staff, including role and contact details.

## How would a frontend be able to manage authentication state based on the data you provide?

The frontend will use the authentication token received after a successful login to manage the authentication state. Once the token is stored (typically in localStorage or sessionStorage), the frontend can send the token as part of the Authorization header in subsequent requests to verify the user's identity. If the token is valid, the user will have access to the application; if not, they will be prompted to log in again. The frontend will check for the existence of the token upon page load to determine if the user is authenticated.

## What pieces of data in your app will need to be secured? How will you demonstrate web security principles in the development of this app?

* The password Storage, and we can do that by using a hashed password using bcrypt.
* JWT: The token will be secured and transmitted by HTTPS.
* -Input validation and sanitization.

## What file structure and program architecture will you use for this project (how will you organize your node project)? Why?

/src

/config # Configuration files (e.g., environment variables, database config)

/controllers # API endpoint controllers (business logic)

/models # Database models (ORM models for tables)

/routes # Routes to handle incoming requests

/middlewares # Middleware functions (authentication, validation, error handling)

/services # Business logic services (e.g., lending management)

/utils # Utility functions (helpers, encryption, validation)

/validators # Input data validation

/tests # Unit and integration tests

/server.js # Main entry point for the application

## What are potential stretch challenges that you could implement to go above and beyond?

We can use File Uploads. And file Downloads

API Endpoint Planning

For the Library Management System API, the following endpoints will be needed for each database entity:

#### User

* POST /users: Create a new user (library member or staff).
* GET /users/login: User login (returns a token).
* GET /users/logout: User logout (invalidate token).
* GET /users/{userId}: Get details for a specific user.
* PUT /users/{userId}: Update user details.
* DELETE /users/{userId}: Delete a user.

#### Book

* POST /books: Add a new book to the library.
* GET /books: Get a list of all books (with optional filters such as title, author, etc.).
* GET /books/{bookId}: Get details of a specific book.
* PUT /books/{bookId}: Update book details (e.g., status, location).
* DELETE /books/{bookId}: Delete a book from the system.

#### Lending Records

* POST /lending: Create a new lending record (assign a book to a user).
* GET /lending: Get all lending records (filter by user, date, etc.).
* GET /lending/{recordId}: Get a specific lending record.
* PUT /lending/{recordId}: Update a lending record (e.g., extend due date).
* DELETE /lending/{recordId}: Delete a lending record.

#### Staff

* POST /staff: Add a new staff member.
* GET /staff: Get a list of all staff members.
* GET /staff/{staffId}: Get details for a specific staff member.
* PUT /staff/{staffId}: Update staff details.
* DELETE /staff/{staffId}: Remove a staff member from the system.

# Project Scheduling and Delegation

Plan out what tasks will get completed with each lesson remaining in the semester (Only edit highlighted text).

|  |  |
| --- | --- |
| Week 04 Tasks | *Project Proposal* |
| Week 05 Tasks | * *Create Git Repo* * *Push to Heroku* * *API DOCUMENTATION is complete and available at route ‘/api-docs’*   Deploy to Render |
| Week 06 Tasks | Check and fix errors . |
| Week 07 Tasks | Week 7 *Video Presentation…* |

## 

## How will you divide up work in your team to ensure the following tasks all get completed?

Since we are few each one of us will get couple of tasks.

* Users collection and all its endpoints - Joseph Israel
* Staff collection and all its endpoints – Andrew Obinna Onyekwere
* Book collection and all its endpoints - Clementine Iradukunda
* Lending records collection and all its end point - Cristobal Henriquez
* Securing and Authenticating the Staff endpoint – Andrew Obinna Onyekwere
* Securing and Authenticating the User endpoint – Joseph Israel
* Deploying the project to Render – Andrew Obinna Onyekwere
* Node.js project creation – Cristobal Henriquez
* Create git repo and share with group – Clementine Iradukunda
* MongoDB setup – Andrew Obinna Onyekwere
* API Swagger documentation for all API route - Farai Dandara
* Video presentation of node project, all routes functioning, mongoDB data being modified, and API documentation. Individually, Last video will be done as a team.

# Potential Risks and Risk Mitigation Techniques

## What are the risks involved with you being able to finish this project in a timely manner?

* Time management so that we make sure that everything is ready and running before the deadline .
* Trying to Debug some error that are unforeseen.

## How will you mitigate or overcome these risks?

Make sure we each work on our part and fix errors before the time of meeting.