# **Blockchain Research Lab**

## **Assignment 2**

Deadline : 7th November 2022 Total Marks : 400

### **Problem Statement :**

Make the assigned project.

| **Trainee Name** | **Project** |
| --- | --- |
| Aditya Raj Sharma | Mail notification backend. |
| Prakhar Maheshwari | Student management backend. |
| Gracy Gupta | Book management. |
| Simran | Book management. |
| Harsh Saxsena | To-do-list backend |
| Jaswant Kushwaha | Student Management backend. |
| Jatin Shankar | Student Management backend. |
| Yash | To-do-list |

**Project Description :**

1. **Backend for managing student record (CRUD) :**
   1. A student record should atleast contain following fields :
      1. name
      2. student\_number
      3. roll\_number
      4. branch
      5. city
      6. cgpa
   2. Following APIs (routes) must atleast be present :
      1. **/students (method : GET) :** This should retrieve all students records
      2. **/students (method : POST) :** This route should create a new student record.
      3. **/students (method : DELETE) :** This route should delete all student records.
      4. **/student (method : GET | use params or query string : roll\_number) :** This route should fetch record of a particular student
      5. **/student (method : PUT | use params or query string : roll\_number) :** This route should replace a student record with another student record.
      6. **/student (method : PATCH | use params or query string : roll\_number) :** This route should update some fields of a student record.
      7. **/student (method : DELETE | use params or query string : roll\_number) :** This route should delete the record of a student.
      8. **students/toppers (request method : GET | use query string specifying n) :** This route should return sorted records of top n students based on cgpa.
      9. **students/bottomers (request method : GET | use query string specifying n) :** This route should return sorted records of bottom n students based on cgpa.
   3. Bonus marks for integrating the APIs (routes) with an interface.
   4. Bonus marks for making an API documentation.
   5. Bonus marks for implementing validations.
   6. Use NodeJS, express, mongoDB, mongoose.
   7. Follow REST apis format.
2. **Book management backend (CRUD):**
   1. A book record must atleast contain following fields :
      1. name
      2. isbn\_no
      3. author\_name
      4. genre
      5. inventory
   2. Following APIs routes atleast must be present:
      1. **/books (method : GET) :** This should retrieve all books records
      2. **/books (method : POST) :**  This route should create a new book record.
      3. **/books (method : DELETE) :** This route should delete all book records.
      4. **/books/find\_books\_needed (method : GET with query string specifying n) :** This route should return all books with availability less than n in inventory
      5. **/books/unavailable\_books (method : GET):** This route returns all books with inventory = 0
      6. **/book (method : GET | use params or query string : isbn\_no) :** This route should fetch record of a particular book
      7. **/book (method : PUT | use params or query string : isbn\_no) :** This route should replace a book record with another book record.
      8. **/book (method : PATCH | use params or query string : isbn\_no) :** This route should update some fields of a book record.
      9. **/book (method : DELETE | use params or query string : isbn\_no) :** This route should delete the record of a book.
      10. **/book/issue\_book (method : GET | use query string or params for isbn) :** This route decrements the inventory of alloted book.
   3. Bonus marks for integrating the APIs (routes) with an interface.
   4. Bonus marks for making an API documentation.
   5. Bonus marks for implementing validations.
   6. Use NodeJS, express, mongoDB, mongoose.
   7. Follow REST apis format.
3. **To-do list**
   1. a to-do-list task must atleast contain following fields :
      1. task\_id
      2. name
      3. description
      4. category
      5. priority (enum : [“low”, “med”, “high”])
   2. Following API (routes) must be present
   3. **/tasks/:category?priority=<priority> (method : GET | use param category, query for priority ) :** This route should fetch tasks of a particular category and query, It should fetch all tasks if priority or category is not provided.
   4. **/tasks (method : POST) :** This route should create a new task.
   5. **/tasks (method : DELETE) :**  This route should delete all tasks.
   6. **/task (method : GET | use params or query string : task\_id) :** This route should fetch record of a particular task
   7. **/task (method : PUT | use params or query string : task\_id) :** This route should replace a task record with another task record.
   8. **/task (method : PATCH | use params or query string : task\_id) :** This route should update some fields of a task record.
   9. **/task (method :DELETE | use params or query string : task\_id) :** This route should delete a task.
4. Bonus marks for integrating the APIs (routes) with an interface.
5. Bonus marks for making an API documentation.
6. Bonus marks for implementing validations.
7. Use NodeJS, express, mongoDB, mongoose.
8. Follow REST apis format.

**4. Mail notification backend**

1. a mailing llist must atleast contain following fields :
   1. name
   2. email
   3. subscription\_status (default : True)
2. Following APIs (routes) atleast must be present :
   1. **/recipients?subscription\_status=<true/false> (method: GET | use query subscription\_status)** : This route should fetch all recipients with subscription\_status = true/false. It should fetch all recipients if subscription\_status is not provided.
   2. **/recipients (method : POST) :** This route should create a new recipient.
   3. **/recipients (method : DELETE) :** This route should delete all recipients.
   4. **/recipient?email=<email> (method : GET | use query email) :** This route should fetch record of a particular recipient
   5. **/recipient?email=<email> (method : PATCH | use query email) :** This route should update some fields of a recipient record.
   6. **/recipient?email=<email> (method : DELETE | use query email) :** This route should delete the record of a recipient.
   7. **/recipient/subscribe?email=<email> (method : GET | use query email) :** This route should set subscription\_status of a recipient to true.
   8. **/recipient/unsubscribe?email=<email> (method : GET | use query email) :** This route should set subscription\_status of a recipient to false.
   9. **/send\_mail (method : POST | body : Subject, Body, HTML ) :** This route should send a mail to all recipients with subscription\_status = true. And ruturn a success\_list and a failure\_list of emails.
3. Bonus marks for integrating the APIs (routes) with an interface.
4. Bonus marks for making an API documentation.
5. Bonus marks for implementing validations.
6. Use NodeJS, express, mongoDB, mongoose, and nodemailer for sending mails.
7. Follow REST apis format.