1. Design and Implementation

1.1 The REST API Specification

Endpoints

A. /index

B. /home, /home/:postId

C. /journal, /journal/:postId

D. /user, /user/:userId, /user/:userId/profile, /user/login, user/signup, /user/logout

E. /settings

Endpoint Relationships

A. /index can be accessed by anyone (GET), users or admin will enter their credentials here to route to /home, DELETE and POST are not allowed, PUT only by admin

B. /home

1. GET: once logged in and authenticated, user's post data will be mapped to a feed, this data is private to user
2. POST: user can create a new post to add to the "journal" collection
3. PUT: not allowed on this endpoint
4. DELETE: user can delete a post that is displayed on the feed, a filter method may be used here

C. /home/:postId

1. GET: the post selected will be called and opened in modal form, user allowed
2. POST: not allowed on this endpoint
3. PUT: user can edit their post, it will be saved to the db
4. DELETE: user can delete a post. deleteOne() will be used

D. /journal

1. GET: user logged in will be authenticated again (auto checked) in order to pull up their posts
2. POST: not allowed
3. PUT: user can filter the posts via form
4. DELETE: not allowed (except by admin)

E. /journal/:postId

1. GET: user allowed to see post selected to view from /journal
2. POST: not allowed
3. PUT: user can edit the post selected via form
4. DELETE: user can delete a single post, deleteOne()

F. /users

1. GET: only admin can see all user data, or entire users collection available here
2. POST: not allowed
3. PUT: not allowed
4. DELETE: only admin can delete all users

G. /users/:userId

1. GET: user can view all their data,
2. POST: not allowed
3. PUT: user can edit their data
4. DELETE: user can delete their account (includes all data associated with their ID)

H. /users/signup, /users/login, /users/logout

1. GET: not allowed
2. POST: allowed for all users(login/logout), and non-users(sign-up)
3. PUT: not allowed
4. DELETE: not allowed

I. /settings

1. GET: allowed for authenticated users
2. POST: not allowed
3. PUT: allowed by authenticated users
4. DELETE: not allowed

1.2 Database Schemas, Design and Structure

<u>Collections</u>

A. Users
    1. Firstname (String, default: “”, required)
    2. Lastname(String, default: “”, required)
    3. Username(String, default: “”, required)
    4. Age(Number, default: 0, required)
    5. Email(String, default: “”, required)
    6. Password(String, default: “”, required)
    7. User gets a token if FB OAuth
    8. Subdocuments:
        - journal (1/user)
            a. Contains array of sub-sub doc “post”
        - settings (1/user)
            a. Light/Dark mode (Boolean, default: false)
            b. Notifications (Mixed, default: “email”)

c. Cookies (Boolean, default: true)

1.3 Communication

A. Scenario: User who doesn't exist tries to login
    - Status: 401 "You are not authorized to view this page"
B. Scenario: User tries to access "/home" without being logged in
    - Status: 401 "You are not authorized, please make sure to provide your credentials to proceed"
C. Scenario: Tries GET, POST, PUT, or DELETE where not allowed
    - Status: 403 "GET/POST/PUT/DELETE not allowed"
D. Scenario: User tries to access an account not their own (not signed into) at users/:userId
    - Status: 401 "You are not authorized to view this page"

2. Conclusions

- (1)I expect to be able to use routes to navigate to and perform CRUD operations there as allowed. (2) I expect to be able to create new collections and documents in those collections as each endpoint and operation allows.

3. References

- Course material
- Mongoose DOCS
- https://medium.com/@alvenw/how-to-store-images-to-mongodb-with-node-js-fb3905c37e6d
- MongoDB docs