Implementation Aspects

Video Link *- https://www.loom.com/share/5187744613dd4874b6b560a081adc6a4*

Note: Watch the above video to understand the below implementation aspects in better

Define API

There are 5 endpoints (Default server url - http://localhost:4000)

- Register User - /api/users
- Login User - /api/users/login
- Upload Mood - /api/upload-mood
- Mood Distribution - /api/mood-frequency/:username
- Locations of Mood nearby - /api/nearby-locations/:mood?lat=:lat&lng=:lng

The API is also documented using swagger & can be viewed in http://localhost:4000/api-docs

Create dev project

Dev project is built using MERN stack (MongoDB, ExpressJS, ReactJS, NodeJS)

Layout code structure

Followed standard code & project structure

Design data model and key data structures

Data model (schema) is designed using mongoose schema and geoJson data structure has been used for retrieval purposes

Define data persistence using any data store of your choice.

Used Redux for data persistence

Define Implementation of operations

Each operation has its own endpoint. Please refer video for more information on operations

Input validation

Input validation is done on the client form level and validation on the server is done at the database level

Authentication

Created register & login server api endpoints and client components to perform authentication

Authorization

Used jwt tokens for subsequent authorization and access of resources

Design, Implementation and Assumptions

- Started the design with a database schema. Figured out geojson structure is necessary for the requirements (finding nearby locations).
- Started the design with **user unique id** as part of moods schema (like foreign key). Then changed it to just a username for client POC purposes.
- Also originally designed list of moods (Happy, Sad, Neutral) as a separate collection/table with a mood code. Didn't create one, for client POC purposes.
- Both the above things could have been added if for future to edit their name while keeping the references only attached to other collections.
- Added 5 different api's for various application requirements
- Used jwt token for authentication & authorization.
- Used winston logger for logging purposes
- Seperated the code into respective folders - controllers, middleware, models, routes for api & database purposes.
- Added swagger documentation for API endpoints.
- Created routes in the client for respective application requirements
- Created login & register components for user authentication
- Used redux for authorization state management
- Thought of using a visualization (map) to getting coordinates and used mapbox
- Used forms for validating the api request body

- For uploading mood & finding mood distribution instead of using logged in user information, I have added user name as part of form to add, retrieve information using single login instead of multiple logins.
- To show how it can be done, I have used logged in user information to get nearby happy locations for a selected location
- Also I made the api a little broader for getting nearby locations. You can request any "mood" in nearby locations for a given location.
- I have tried to provide information on how to use and navigate in the app on the home screen
- One thing I forgot to show on the video is you can able to click on the mood markers on the map. It'll show you the name of the location of the mood. You can add the locationName as part of your API call. Thought it'll be useful to enter your personal coordinates (Home, Work, etc,.)
- Seperated the code into respective folders - components, pages, routes, store for reusability & decoupling.
- Used MongoDB Atlas Cloud instead of local, so that you can enjoy the existing data. Use "**psaelpa**" username to retrieve mood distribution easily.

Final Comments

- I know the time allotted was for one day. But I might have used more than one day (probably 1.5 ~ 2 days) as I wanted to implement the client as well.
- Swagger documentation is not working for testing purposes. There was still authentication needed to integrate properly. But feel free to use it to view overview of the API endpoints & schema design
- Unit tests couldn't be added. As I have already run out of time. If you wish, I can spend couple more hours in writing test cases using Mocha Chai framework
- README only has information to start the app. Again due to time constraints, I have used this document to provide detailed information.
- I wasn't able to give the full tour of the application in the 5 minute video. Feel free to start the application & play around with the application. If anything breaks (it really shouldn't), kindly restart the servers. It is just a 1 day POC 🙂