Fullstack Microservice Assignment 1(Intern)

Frontend

Create a singlepage web application which allows users to manage the content

1. Landing page consists of a sign up form with email and password as fields
    a. Both fields in sign up form should be required
    b. Implement inline field validation and indicate errors
    c. Enable sign up button only when all fields are filled with correct validations
2. After signup, use a dashboard to show all available content with title and likes
    a. Dashboard can fetch data from top contents api in **content micro service.**
3. Use persistent storage to store user id, which is returned on sign up. Use this as a check for login as well for the scope of this assignment, if user id is not present in persistent storage then show sign up form.
4. When the user clicks on a content the application should display on the same screen the following attributes: story content, title, likes and publishing date
5. User can like a content and it should reflect in the real time.

Concerns:

1. UI application can be implemented using any framework [vue.js preferably / react]
2. UI should be responsive and should follow fundamentals of responsive design
3. Your design and code should meet these requirements and be sufficiently flexible to allow for future extensibility.
4. Design is not given in order to allow you to come up with the same, you can be as creative as possible, while keeping in mind other aspects of the assignment as well.

Backend Microservices

2 micro services - content, user-interaction service.

Content Service

1. Serving books as content. The content will have a story and title, considering the scope of this assignment.
2. Data ingestion should happen via csv, write a script to ingest data into the database(IdIdeally script should also be a part of your service). Content service should have at least the title, story, date published and the user id stored.
3. Top contents API - sorted on user-interaction[Sort on basis of Number of likes]
4. ***Testing****- An API to help us post the csv file, and it should automatically invoke the data ingestion process once it receives the csv file. User data should be sent to User Interaction Service and stored accordingly.*

User interaction service

1. Add Update API for User Like event(validate if user exists)
2. Add signup API for User which will take fields as userId and password
3. email address should consist of an email prefix and an email domain, both in acceptable formats.
4. password should follow following policy:
    a. a minimum of 1 upper case letter [A-Z] and
    b. a minimum of 1 lower case letter [a-z] and
    c. a minimum of 1 numeric character [0-9] and
    d. a minimum of 1 special character
    e. password must be at least 10 characters in length, but can be much longer.
5. return auto generated userId in response of this API and use it to validate user

Backend Concerns

1. Please dockerize everything(yes, including databases).
2. Micro-service architecture has to be followed strictly, databases should be separate for all 3 micro-services. [same **database instance** ca n be used for the purpose of this interview, but the database itself should be entirely separate for each micro-service].
3. No Code repetitions, common pieces of code should not be replicated anywhere.
4. Database schema/design
5. HLD and LLD
6. REST API conventions(We are open to graphQL as well).
7. Documentation, we would prefer something like swagger, but open to anything you come up with as long as it helps us understand your code better.
8. Consider using an Architecture diagram.
9. Worker - Should always be part of the service. In general workers can be CRON as well as queue consumers.

Code base

1. We prefer golang, but we are open to other languages as well.
2. Database - Again up to you, but do remember that we might need to run your application on local.
3. Ideally Code should be **dockerized** and neatly tied in using docker-compose[network mode can be host]. In case you are finding it difficult to use **docker-compose**, just dockerize all applications/db and send us commands to build/run your code. If possible put commands in a script, again not a deal breaker though.
4. We expect Postman collection or sample commands for all API's.
5. Again bonus points for considering developer empathy. https://apiguide.readthedocs.io/en/latest/principles/empathy.html#:~:text=Design%20with%20developer%20empathy&text=Perhaps%20the%20most%20important%20criteria,will%20remain%20undiscovered%20or%20unrealised
6. Please share the github link or zip file for the assignment via email.