**Full Stack Coding Challenge**

Welcome to the JotPsych Full Stack Coding Challenge! This test is designed to assess your ability to work with a Flask API, a React frontend, and your ability to plan and problem solve given time constraints. You will have 2 hours to complete the challenge.

**Objective**

Complete the given Flask API and React application, expand them based on the tasks outline below, and write about your approach and plans in a notes.md file. Keep notes on any major problems you encounter, your debugging steps, and how you come to solutions.

**Getting Started**

**Prerequisites**

* Python 3.x
* Node.js
* npm or yarn

**Setup Instructions**

1. **Backend (Flask API)**:
   * Navigate to the backend directory:

cd src/backend

* + Create a virtual environment and install dependencies:
  + python -m venv venv
  + source venv/bin/activate # On Windows use `venv\Scripts\activate`

pip install -r requirements.txt

* + Run the Flask API:

python app.py

1. **Frontend (React)**:
   * Navigate to the frontend directory:

cd src/frontend

* + Install dependencies:
  + npm install
  + # or

yarn install

* + Run the React application:
  + npm run dev
  + # or

yarn dev

**Tasks**

**Basic Tasks**

1. Finish implementing the user route in the backend to return user information to the frontend.
2. Implement the logic to have the /login and /register endpoints store the authentication token in such a way that the rest of the application can access it.
3. Create a /logout functionality in the frontend.
4. Use MUI (or other component library) to make the auth components *slightly* more stylish. (Spend very little time on this!)
5. Create a user profile component according to the wireframe in wireframes/profile\_screen.png to fetch and display user data after login. Add corresponding required data to /register endpoint and Register component.

**Version Management**

1. Look at the APIService.ts file in the frontend services folder. Reimplement all network requests in the frontend to use this service, so that the app-version header is sent with every single request.
2. Make it so that if the app-version header with a request is < 1.2.0, the server doesn't process the request but instead returns a message that prompts the user to update their client application.
3. Handle this update message in the frontend and display a message in the UI.
4. Optional: Mock the transition from the old version of the application to the new one, so initial load is a version below 1.2.0 and subsequent interactions send a version above.

**Audio Recording**

1. Use the MediaRecorder API (built into browser) to build a very simple recorder component that records up to a maximum of 15 seconds of audio (audio/webm codec).
2. After 15 seconds have been reached or the user stops the recording, send the audio blob to an /upload endpoint on the backend.
3. Mock a python function that mimics the process of sending this audio to a third party transcription service that returns a transcript of the audio. (BONUS: If you moved quickly through everything else and want to actually implement a local transcription model, go ahead!)
4. Save this transcription text to the user database as the user's motto and update the user profile to display their self-recorded motto.

**Encrypt it**

1. We don't want to store the patient's motto in plaintext. Encrypt it when you store it and decrypt it when you retrieve it.

**Make it Async**

1. Change the mock transcription function so that it takes a random amount of time between 5 and 15 seconds to return.
2. Make it so that the backend can handle multiple users submitting recordings at the same time, and still serve profile information to other users.

**Anything else**

1. Use the rest of your time to show us your skills and add something else to the application. Make the recorder more fully featured, add additional security, explain how you would build this for enterprise scale, etc. Show insight and initiative!

**Important Notes**

* Get as much done in the two hours as possible but **submit your work even if it's not complete**.
* If you are coming up to time and haven't finished, write out in notes.md what you would do to finish or thoughts you have about how to build other parts if you had more time.
* **Problem-solving skills are key**. You can use outside resources to help you complete the tasks.
* Emphasis will be given to the write-up you produce in notes.md about how you approached the tasks.

**Submission**

Please fork this repository and submit your completed code via a link to your forked repository.

Good luck!