Aditya Kumar adityakumar0718@gmail.com

Full-Stack Developer Assessment

Face Recognition Web Application

Application Architecture and Design Choices:

The application follows a client-server architecture, with the frontend built using React.js and the backend implemented using Node.js and Express.js. MongoDB is used as the database for storing user information and other relevant data. The application is hosted on AWS.

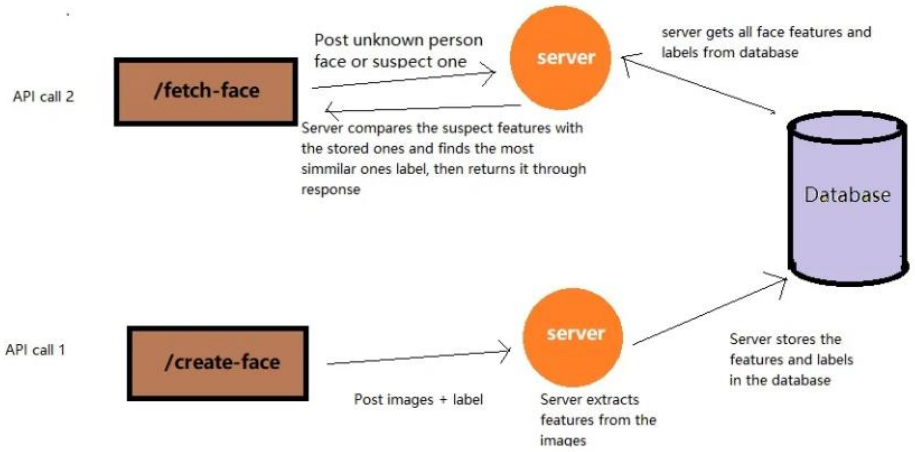

Fig: Application Architecture

Frontend:

- React.js was selected for its component-based architecture, enhancing modularity and reusability.
- Notable packages utilized:
  - axios: A promise-based HTTP client for facilitating AJAX requests to the server.
  - react-webcam: A React component enabling access to the user's webcam and capturing video streams.
- The application consists of three distinct pages:
- The login screen where we get the camera access and start the authentication process
- We have added a button to retake picture in case the picture is hazy or the face is not recognised.
- The result page, indicating the success or failure of the login attempt.
- The add face data screen, facilitating the addition of a new user.
- The registration and login process is intentionally designed to be user-friendly, boasting an intuitive user interface.

Backend:

- The code relies on several packages including Express.js, face-api.js, mongoose, canvas, and express-fileupload for handling various functionalities such as HTTP requests, face,cors

recognition, database operations, image processing, and file uploads. All errors in the code are logged for better debugging and troubleshooting.

- The server listens on port 5001 once it successfully connects to the MongoDB Atlas database.
- Backend serves static files from a React app's build folder to manage client-side routing. This is facilitated by using the express.static middleware.
- An asynchronous function, LoadModels(), is defined to load face recognition models from disk using the faceapi library.
- A MongoDB schema named faceSchema is defined to structure the data for storing face information, including fields like label and descriptions.
- '/post-face' is an API endpoint used for processing images to extract face descriptors using the loaded models, which are then stored in the MongoDB database.
- '/check-face' is another API endpoint designed to retrieve face descriptors from the database based on uploaded images. These descriptors are then compared with the descriptors of known faces to recognize and match faces.

Deployment:

1. Selection of EC2 instance type
2. Creation of EC2 instance
3. Configuration of security groups
4. SSH key pair setup
5. Installation of PM2 globally on the EC2 instance
6. Configuration of PM2 for managing Node.js application
7. Install npm nodejs expressjs and other modules on EC2
8. Install nginx and configure nginx as reverse proxy
9. Create a folder on EC2 and clone the git repository and start the project using pm2 and check for any errors through ‘pm2 logs 0’.
10. Then go to https://freedns.afraid.org/ and get a free domain from here and in the destination of the domain name add the Public IP address of the EC2 instance.
11. The website link of the project: http://faceauth.us.to/

Challenges faced:

- Integrating React.js, Node.js, Express.js, and face recognition AI posed initial hurdles. Researched Medium articles and YouTube tutorials for solutions.
- Faced challenges in storing face data in MongoDB. Opted to store the face descriptors instead of images for efficiency.
- Attempted AWS Lambda deployment but faced limitations due to large zip file size. Explored EC2 instance deployment for a workaround.
- Canvas module dependency in face-api.js caused compatibility issues with Vercel and Netlify. Chose EC2 deployment for compatibility.
- Despite successful local deployment, camera functionality issues persisted on the EC2 instance. Though the application and all its functionalities work perfectly on my local machine, I am currently debugging to resolve the issue.

Aditya Kumar adityakumar0718@gmail.com

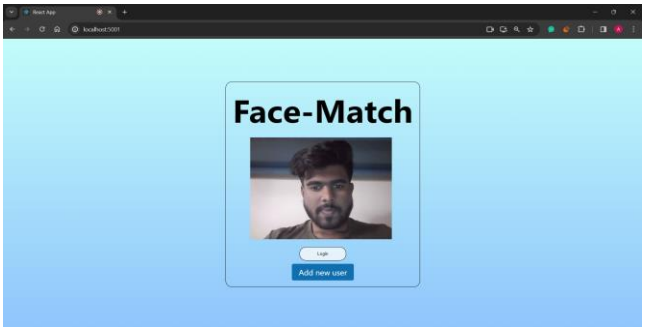

Figure 1: Login Screen

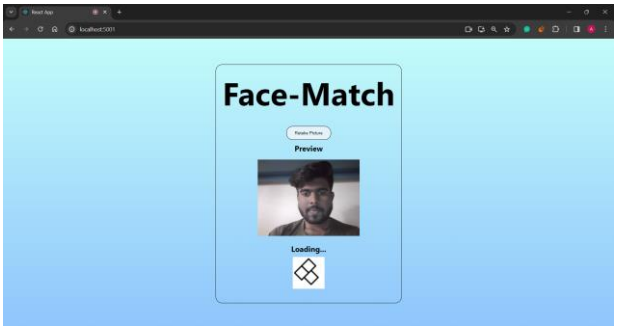

Figure 2: Loading Screen

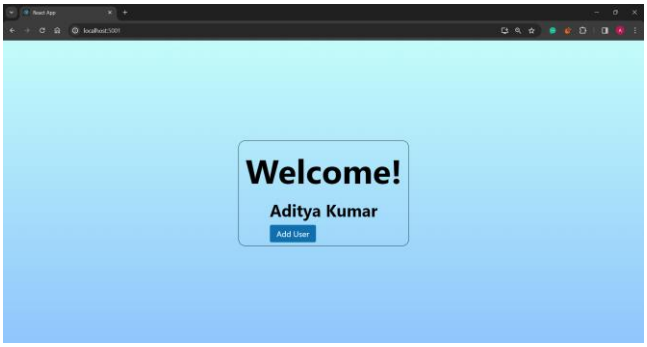

Figure 3: Result Page

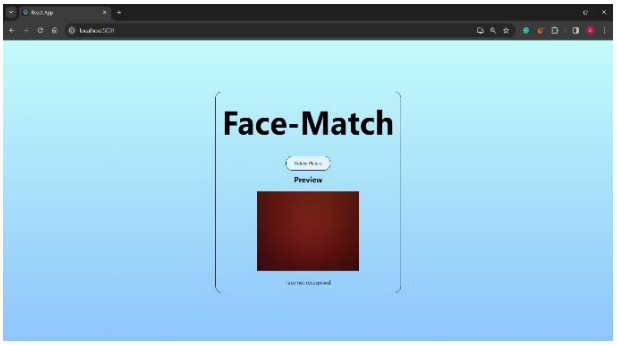

Figure 4: In case Face not recognised it prompts to retake picture

Figure 5: Add New User Page

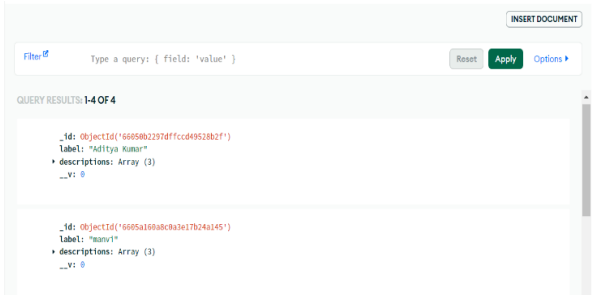

Figure 6: Database Collection from MongoDB