**MoodSensingApp**

MoodSensingApp applications is a set of APIs which allow users to analyze their mood based on the facial expressions provided in the image.

**Features**

- User registration: Users can create an account with a username and password.

- User authentication: Users can authenticate themselves using their credentials, using JWT for authentication

- Image emotion analysis: Users can upload images, and the app will analyze the emotions displayed in those images.

- Closest happy location: Given the user’s current location, return the closest location where the user has been happy.

**Requirements to run the application -**

- Google Cloud Vision API key: Obtain the key from the Google Cloud Console and set the `GOOGLE\_APPLICATION\_CREDENTIALS` environment variable to the path of the key file.

**How to run the application –**

* Unzip the solution
* Add the google credentials keys to the environment variable to GOOGLE\_APPLICATION\_CREDENTIALS by adding it to GoogleFaceRecognitionService class constructor as

Environment.SetEnvironmentVariable("GOOGLE\_APPLICATION\_CREDENTIALS", "vision\_api\_key.json");

* Build the project
* Run the app

**Implementation Design –**

1. Architecture:

- Presentation Layer: Handles HTTP requests and responses using ASP.NET Core, including controllers and API endpoints.

- Application Layer: Contains the services that orchestrate the business logic and interact with repositories.

- Repository Layer: Provides data access and storage operations using repositories and database context.

- Database: Stores the application data, such as user information, mood captures, and locations.

2. Service Components:

- User Management: Allows users to register, authenticate, and generate JSON Web Tokens (JWT) for authentication and authorization purposes.

- Mood Capture: Enables users to capture their mood and store it along with their location information.

- Location Service: Retrieves the user's current location using the IP address and determines the closest happy location based on mood captures.

- Google Face Recognition Service: Uses the Google Cloud Vision API to analyze facial emotions from images and determine the mood.

3. Authentication and Authorization:

- The app uses JWT (JSON Web Tokens) for authentication and authorization.

- Users can register with a unique username and password.

- Passwords are securely hashed and stored with salt for improved security.

- The app generates JWTs upon successful authentication, which are then used for subsequent authorized API requests.

4. API Endpoints:

- User-related endpoints: Register user, authenticate user, generate JWT.

- Mood capture endpoints: Add mood capture, retrieve mood captures for a user.

- Location-related endpoints: Get the closest happy location for a user.

5. Third-Party Integration:

- The app integrates with the Google Cloud Vision API for facial emotion recognition and analysis.

- API credentials and access keys are securely stored and accessed during image analysis.

- The app calls the IP-API to get the user location.

- The app uses SQLite db for the persistent storage.

6. Model Validation:

- Model validations are implemented at request models.

- Created custom validation so that user can upload only image in file type.

7. Exception Handling:

- The app uses custom middleware for exception handling which returns internal server error if the environment is not development.

8. Testing:

- Unit tests are implemented for the service classes using a testing framework like NUnit

- Mocking frameworks like Moq are used to mock dependencies.

**Assumptions:**

* The app assumes that the user's current location can be determined using their IP address.
* The app assumes that the user will upload an image of a person, and in the image their should

Be only one person.

**Unfinished Work:**

* Unit test cases are covered for few services and controllers, however few controllers and services UTs are yet to be completed due to time constraints.
* Swagger documentation is not proper. Will include xml comments of actions method in swagger.
* Logging is not present. If possible, want to implement serilog and will save the logs in file.