**Abstract**

I built multi-purpose web application using face detection called “**Mentor-Student Management System**” to enhance mentor – student relationship and to monitor students’ regularity and discipline particularly to prevent unauthorized persons attending to the interactions.

Features in my web application are:

1. Drowsiness Detection
2. Preventing Crime
3. Attendance Tracking

I have designed it in such a way that it is scalable, and we can keep on adding other required new features.

**Design and Technologies Used**

I built web application in 3-tier architecture . So, it mainly consists of 3 parts:

1. Frontend
   1. MSAL for authentication
   2. React
   3. TypeScript
2. Backend
   1. Flask
   2. Python
   3. TensorFlow
   4. NumPy
   5. OpenCV
3. Database
   1. Azure Table Storage

I am using **Agile Methodology** for this so at any point in time I will have a working model.

**Front end**

1. We are using **MSAL** for **Authentication**.
2. And for building web application we are using React + TypeScript.
3. There are mainly 2 parts in UI
   1. Add Faces
   2. Manage Classes and Students
4. All the data a user can see is the only data they added.
   1. When calling backend API we send username if the current user in the body of request
   2. If a user A added face id of a person, then user B don’t get that face id.
   3. Everyone can only see the data that they added
5. User can give input in multiple ways
   1. Via Video camera
   2. Uploading image
6. Features of Add Faces:
   1. Identify faces and get face data
      1. Helps to identify criminals in any place like bus stands.
      2. Helps to identify missing people
      3. Helps to do face ID check of people.
      4. Helps to identify number of drowsy people in a class or meeting.
      5. If they have a criminal background or are under “wanted” list, then we can alert.
      6. If we are using this application in school, then attendance of that student is marked for that day by calling Backend API. We can detect who is drowsy in the class.
      7. After detecting an unknown face:
         1. User can add details of the unknown faces and submit, to update the database or add new face data in database.
         2. If we are using this at:
            1. Home: then inform tenants that an unknown person entered the house.
            2. School: inform principal regarding this.
   2. Add or update Face data
   3. Clear all face data added by current user
   4. Face Data:
      1. Name of the person
      2. Is person criminal
      3. Is person drowsy
7. Features of Manage Classes and Students
   1. Create new class
   2. Select a class
      1. Get attendance report of the class
         1. Shows date wise segregated attendance report with the list of students attended on that day
      2. Register a student to class
         1. Take attendance of registered candidates
         2. Check if registered candidates are drowsy or not
      3. Unregister a student from class
      4. View unauthorized or unknown people in that class
      5. Identify if any criminals are there in that class

**Backend API**

1. We are using **Flask Framework and Python** for backend server.
2. Front end make API calls to the backend. Backend API get the images from frontend and process them.
3. Tasks of backend API
   1. Face Management
   2. Class management
   3. Student Management
4. In backend we are primarily using 2 ways to detect faces:
   1. **Azure Face client library API**
   2. Python Library- “**face\_recognition**”

**Tasks of backend API**

1. Tasks of backend API
   1. Face Management
   2. Class management
   3. Student Management
2. Face Management
   1. Identify faces
      1. Checks database and gets all the rows created by current user, that is all the rows whose partition key is username of the user and compares face encoding stored in database with current face encoding from image.
      2. Returns the following for each face identified in the image
         1. Name
         2. Person Id
         3. Is person drowsy or not
         4. Is person criminal or not
         5. Face location in original image
         6. Cropped image in base 64 format
   2. Register faces
      1. For each face identified in the image
         1. Checks Azure table storage database and Azure face client library database for current face
         2. If current face doesn’t exist in azure face client library then
            1. Adds the current face to azure face client library
            2. Returns the Person ID
         3. If current face doesn’t exist in Azure table storage or is we are updating
            1. Gets the encoding of current face from face\_recognition library and converts it into string
            2. Gets the person id from azure face client library
            3. Adds the name, criminal status(received from user),encoding, person id in database

Partition key is username

Row key is person id

* 1. Clearing all face data
     1. Deletes all the faces added by current user in database.

1. Class management
   1. Creating a new class
      1. Adds new class to database
         1. Partition key-username
         2. Row Key-class name
         3. We also add new field called IsClass, to identify that this row represents class
   2. Getting list of classes created by current user
      1. Returns list of names of all the classes created by current user
   3. Mark attendance of a class via image
      1. Steps involved in attendance taking
         1. Identifying all the faces in the image
         2. Getting person ids of all the people
         3. Getting list of registered ids in the current class
         4. Finding all the registered students in the given image in the class
         5. Adding attendance to Database
            1. Partition key-username
            2. Row key- class name+ “---” +date+”---“+”class”
            3. Attendance-list of all person ids converted into a string by joining them using “---”
   4. Get Attendance report of the class
      1. Returns a map where keys are date of classes and values are list of names of the students
2. Student Management
   1. Registering a student to a class
      1. Adds person id to the students field.
   2. Unregistering a student from a class
      1. Removes person id from the students field
   3. Getting all registered student ids of a class
      1. Returns list of person ids registered to the current class

**Face Detection in Backend**

1. In backend we are primarily using 2 ways to detect faces:
   1. **Azure Face client library API**
   2. Python Library- “**face\_recognition**”
2. Advantages of using face detection Python Library:
   1. Very easy to use because of rich inbuilt functions.
   2. Easy to find face similarities and face detection.
   3. Returns encoding for every face which is unique based on their facial features.
   4. These encodings can be used to identify each person individually.
   5. Free to use
3. Disadvantages of using face\_recognition:
   1. No inbuilt support for database
   2. So, we need to use an additional database for storing people’s information
4. Advantages of using Azure Face client library API:
   1. It predicts the user’s age, gender.etc. Which we can use for Smarter Advertising.
      1. Let us say we are using this application in a bus stand or railway station, then basing on the majority of the people’s gender and age group we can show ads in that place selectively.
   2. It comes with an inbuilt server and database, so there is no need to have additional Database
5. Disadvantages of Azure Face client library API:
   1. Free version comes with many restrictions
   2. Maximum 10 requests per minute
   3. Complicated to use.

**Working in Backend**

1. In backend we have mainly 2 steps:
   1. Detecting Faces
   2. Using Machine learning models with those faces detected in above step.
2. Now after we get outcomes from those 2 detecting models; Azure face client library API and face\_recognition Library we are using those predictions in next step.
3. After face\_recognition models identify any face:
   1. API are calling database and get list of all users and their face encodings and compare their encoding with current encoding and identify the person.
   2. Basing on the threshold we give we are controlling the accuracy with which we want to identify the person. And return the user’s name or any other details.
4. **Attendance Tracking**:
   1. All the registered known faces detected in the previous step are marked present in the selected class.
   2. And multiple images of the class are taken to make sure every person is marked properly.
   3. Let us say we take 3 markings of a same class on same day, then union of all the persons identified will be marked as final attendance.
   4. Every time we are marking the attendance we are checking the latest attendance taken on that day (if exists) then we are doing union of current ids and previous ids and updating current row.
5. **Preventing Crime**:
   1. If any criminal is detected or is under “wanted” list, then are informing user that criminal is detected.
6. **Drowsiness Detection**:
   1. After the first step from the results of face detection we get the location of faces in the image which we are using to crop the image and send it to pre trained ML models to predict if they are drowsy or not.
   2. ML models returns the probabilities with which both eyes are drowsy, if the probabilities exceeds the threshold, then we are considering them as drowsy and informing user the same

**Database**

1. We are using **Azure storage tables** for database.
2. The partition key of the row is from which user account these are being stored i.e., username.
3. We are storing the following things in database:
   1. Face Management
      1. Name of the person
      2. Face Encoding
      3. Criminal Status
      4. Who added the face to database- as Partition Key
      5. Person Id- as Row Key
   2. Class Management
      1. Who added the class to database- as Partition Key
      2. Name of the class – as row key
      3. Students- list of person ids joined and converted into string
   3. Attendance Management
      1. Who added the class to database- as Partition Key
      2. Name of the class+ date – as row key
      3. Attendance- list of person ids of registered students who attended the class on that date
4. We are storing results from the azure face client library API and face\_recognition Library in the database.
5. As Azure Face client library already has a database, we just store the person id in our database along with their name.
6. With face\_recognition as they don’t provide any database, we store the encodings of the person’s face in the database.

**Flowcharts** Above figure shows high level view of my project.

Above figure shows detailed view of UI and Flask API(Backend)

Above figure shows detailed view of Models

**Flask API(Backend) Response**

**Get Predications or Identify:**



Above screenshots are the JSON responses of the flask API. When we call API to get predictions.

We receive a JSON where we have multiple keys ranging from 0 to n-1 , where n is the number of faces identified in the image. Each value represents face data of i-th face identified in the uploaded image or frame taken from video camera.

**Future Scope**

1. **Acne Detection**:
   1. Using the images received from Front end we will detect the faces using face\_recognition library.
   2. We then crop the images and send faces to Acne Detection Model (by **Microsoft collaborates with Nestlé Skin Health SHIELD**).
   3. Providing patients with an easy way to assess and track the progress of their acne treatment.
2. **Face Mask Detection**:
   1. Using images received from Frontend we will send the images to Face Detection Models which returns faces of the people without masks.
   2. Identities of these people retuned by models will be sent to Azure face client library to get details of the people without masks.
3. **Selective Advertising**:
   1. Basing on the predictions from Azure Face client library API we will get age, gender.etc. details of the person. Which we can for advertising as mentioned already.