Face Recognition & Tracking App

- **6.1 How the System Works**

The face recognition and tracking system operates through a series of coordinated steps involving hardware, software, and cloud services.
Users first authenticate using Firebase, which supports Google, phone number, and email logins.
The ESP32-CAM, equipped with a GPS module, captures images and streams live video.
These captures and streams, along with location data, are uploaded to a server.
The server processes the images for face recognition using machine learning algorithms.
If a match is found, the server sends a notification to the user with the detected face's location (latitude and longitude).
Users can view live streams, receive notifications, and interact with a chatbot for system queries through a mobile app.
The app also displays the camera's location on a map and calculates the distance between the user and the camera, as well as tracking individuals on the map.

- ## 6.2 Authentication with Google, Phone Number, and Email (using Firebase)
  - **Google Authentication:**

    Users can log in using their Google accounts through Firebase Authentication.

    Provides a quick and secure way to access the app without needing to create a new account.

  - **Phone Number Authentication:**

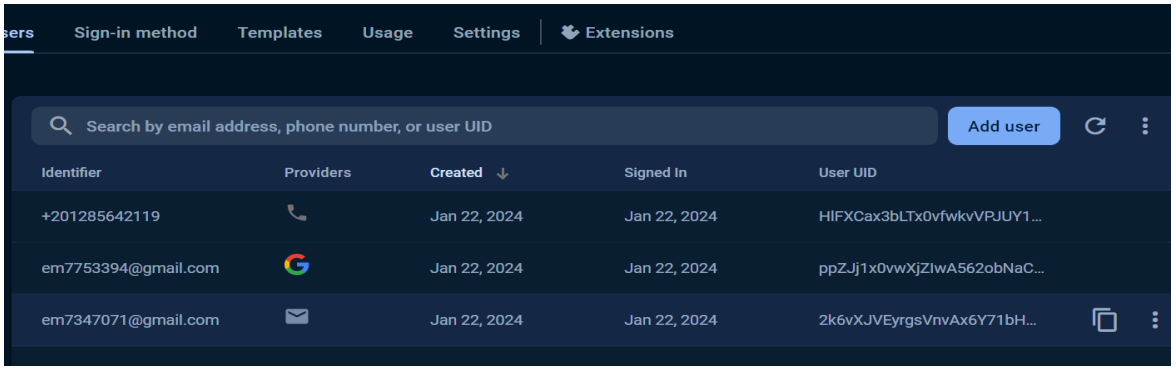
    Users receive an SMS code to verify their phone number.

    Ensures that only verified users can access the system, adding an extra layer of security.

  - **Email Authentication:**

    Traditional login method using email and password.

    Managed by Firebase Authentication to ensure secure and reliable access control.



- ## ESP Camera Integration:
  - **Image Capture:**

    The ESP32-CAM captures images at set intervals or on-demand.

    These images are crucial for the face recognition process.

  - **GPS Tracking:**

    An integrated GPS module logs the camera's location.

    Location data is sent to the server along with the images to track where each image was captured.

  - **Live Streaming:**

```
onPressed: () {
    _launchURL(
      "https://583f-41-34-175-243.ngrok-free.app/mjpeg/1");
    },
```

    The ESP32-CAM streams live video to the server. I need the IP address of your ESP32-CAM.

    Users can view this live feed in real-time through the mobile app, enabling continuous monitoring.

- **Data Upload:**

```
var storage = AzureStorage.parse(
    'DefaultEndpointsProtocol=https;AccountName=dataset7;AccountKey=yQc8xFTR6LdfUSg6J19C0Z6DhzrgrS9rnjT6QkA+WPl5QNAGTtrFP84a7prOkXSQUziyXi1Q1VYq+ASt5v
String container = "eman";
```

Captured images and live streams are uploaded to the server.

Ensures all data is centralized for processing and storage.

To upload images and videos to Azure Blob Storage, you'll need the following information:

1. Connection String: This is a key that grants access to your Azure storage account.
2. Container Name: This is the name of the blob container where your images and videos will be stored.

- **Server-Side Processing:**
  - **Face Recognition:**

    The server processes the uploaded images to recognize faces using machine learning algorithms

    Matches detected faces against a database to identify known individuals.

  - **Notification Service:**

    When a match is found, the server sends a notification to the user.

    Notifications include the location of the detected face (latitude and longitude) to inform users

- **User Interface:**
  - **Live Stream View:**

    Users can view the live feed from the ESP camera through the mobile app.

    Provides real-time monitoring capabilities.

  - **Notification Alerts:**

    Real-time alerts notify users when a recognized face is detected.

    Includes location details to help users take immediate action.

  - **Person Tracking**:

    Users can track persons on the map based on the GPS data received from the cameras.

- **Chatbot Integration:**
  - **Query Handling:**

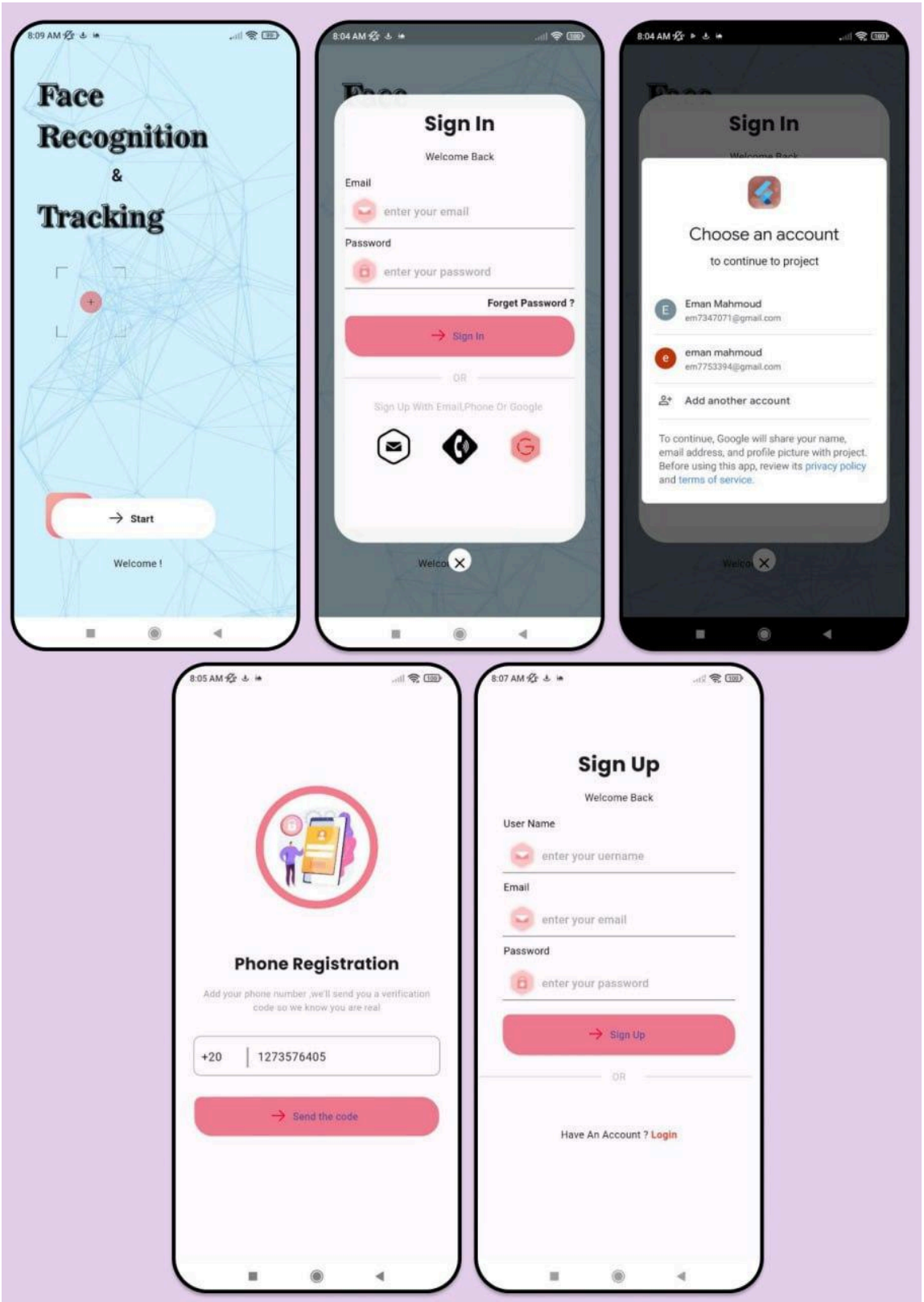
    Users can interact with a chatbot to ask questions about the system or data.

    The chatbot provides information and assistance, enhancing user experience.

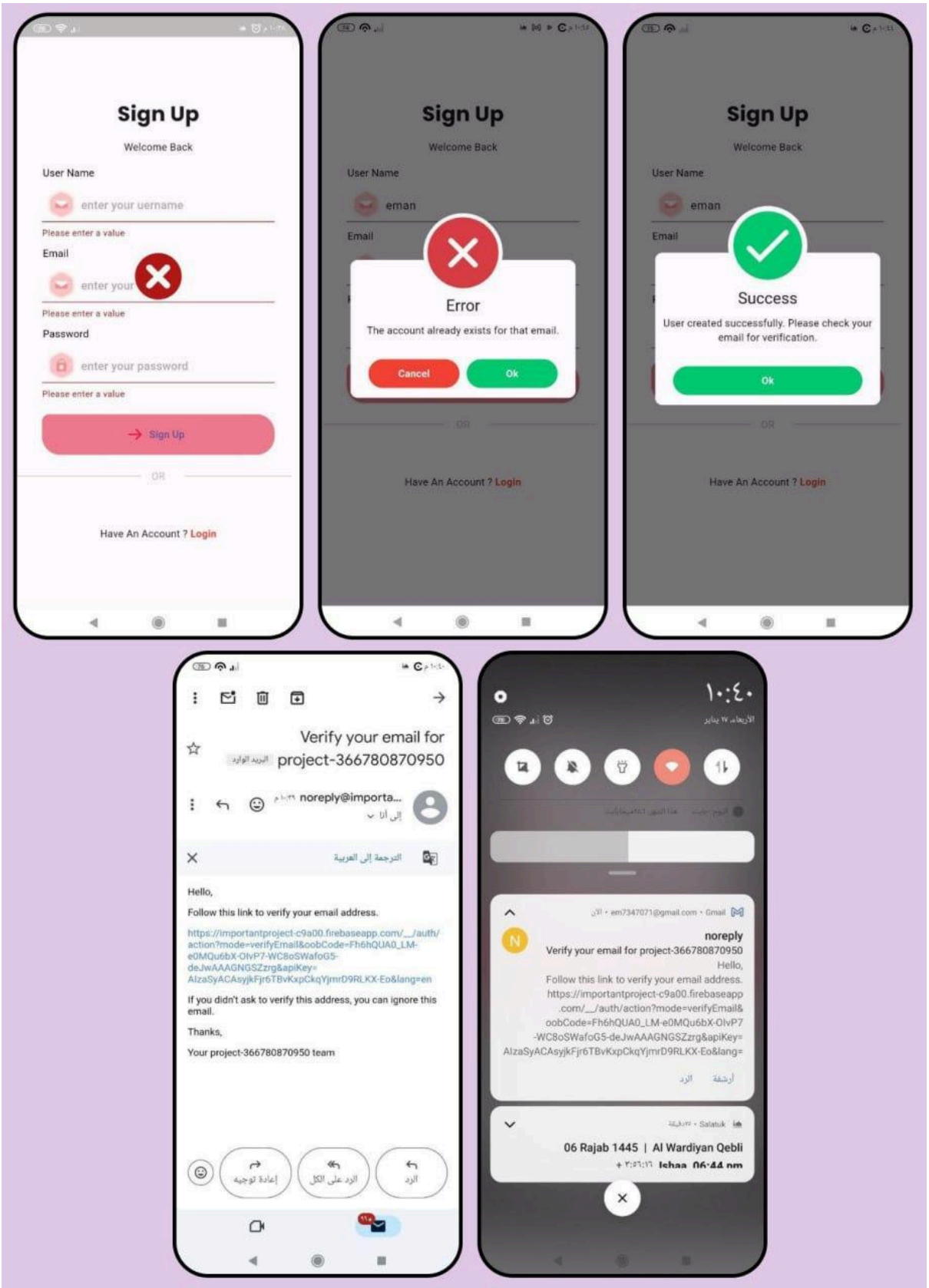
  - **Information Retrieval:**

    The chatbot can access and provide data stored on the server.

    Ensures users can easily obtain the information they need without navigating through the app.


Face
Recognition
&
Tracking
Start
Welcome !
Sign In
Welcome Back
Email
enter your email
Password
enter your password
Forget Password ?
Sign In
OR
Sign Up With Email,Phone Or Google
Sign In
Choose an account
to continue to project
Eman Mahmoud
em7347071@gmail.com
eman mahmoud
em7753394@gmail.com
Add another account
To continue, Google will share your name, email address, and profile picture with project. Before using this app, review its privacy policy and terms of service.
Phone Registration
Add your phone number ,we'll send you a verification code so we know you are real
+20
1273576405
Send the code
Sign Up
Welcome Back
User Name
enter your uername
Email
enter your email
Password
enter your password
Sign Up
OR
Have An Account ? Login

- My app offers seamless access with **Google Authentication**, secure verification via **Phone Number** Authentication, and traditional login convenience through **Email Authentication.**



- **Sign-Up and Email Verification Process:**

  - **Sign-Up Process**

    1. Sign Up with Empty Fields:

    Scenario:

    User attempts to sign up with empty fields.

    System Behavior:

    Prevents the account creation process.

    Displays an alert or message instructing the user to fill in all required fields before proceeding.

    User Feedback: "Please fill in all the required fields."

    2. Email Existence Check:

    Scenario:

    User enters an email address during sign-up.

    System Behavior:

    Performs an email existence check.

    If the email is already registered:

    Notifies the user immediately.

    Advises the user to use a different email address to proceed.

    User Feedback:

    "This email is already registered. Please use a different email address."

    3. Successful Sign-Up:

    scenario:

    User completes the sign-up process with valid and unique details.

    System Behavior:

    Provides a confirmation message indicating the successful creation of the account.

    Sends a welcome notification guiding the user to check their email for verification.

    User Feedback:

    Your account has been successfully created. Please check your email to verify your account."

User Email Verification Process

1.Start and Verify Email:

Scenario: User completes the sign-up process.

System Behavior:

Sends a verification email with a special link to the user's provided email.

User Feedback:

"A verification link has been sent to your email. Please check your inbox and click the link to verify your email."

2. Confirm Your Account:

Scenario: User receives the verification email.

System Behavior:

User clicks the verification link in the email, confirming their email address.

User Feedback: "Your email has been successfully verified."

3.Unlock Your Account:

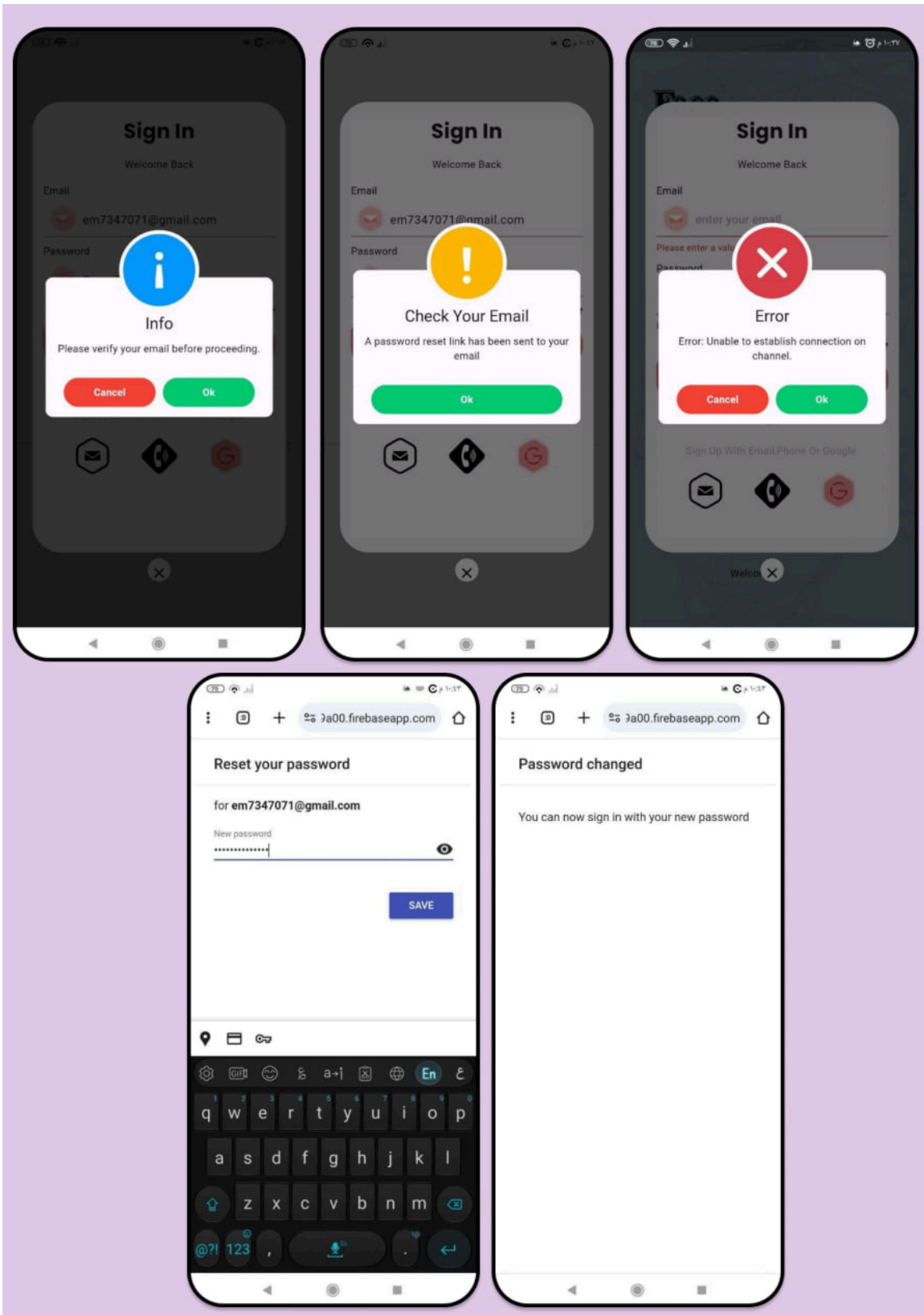
Scenario : User clicks the verification link.

System Behavior :

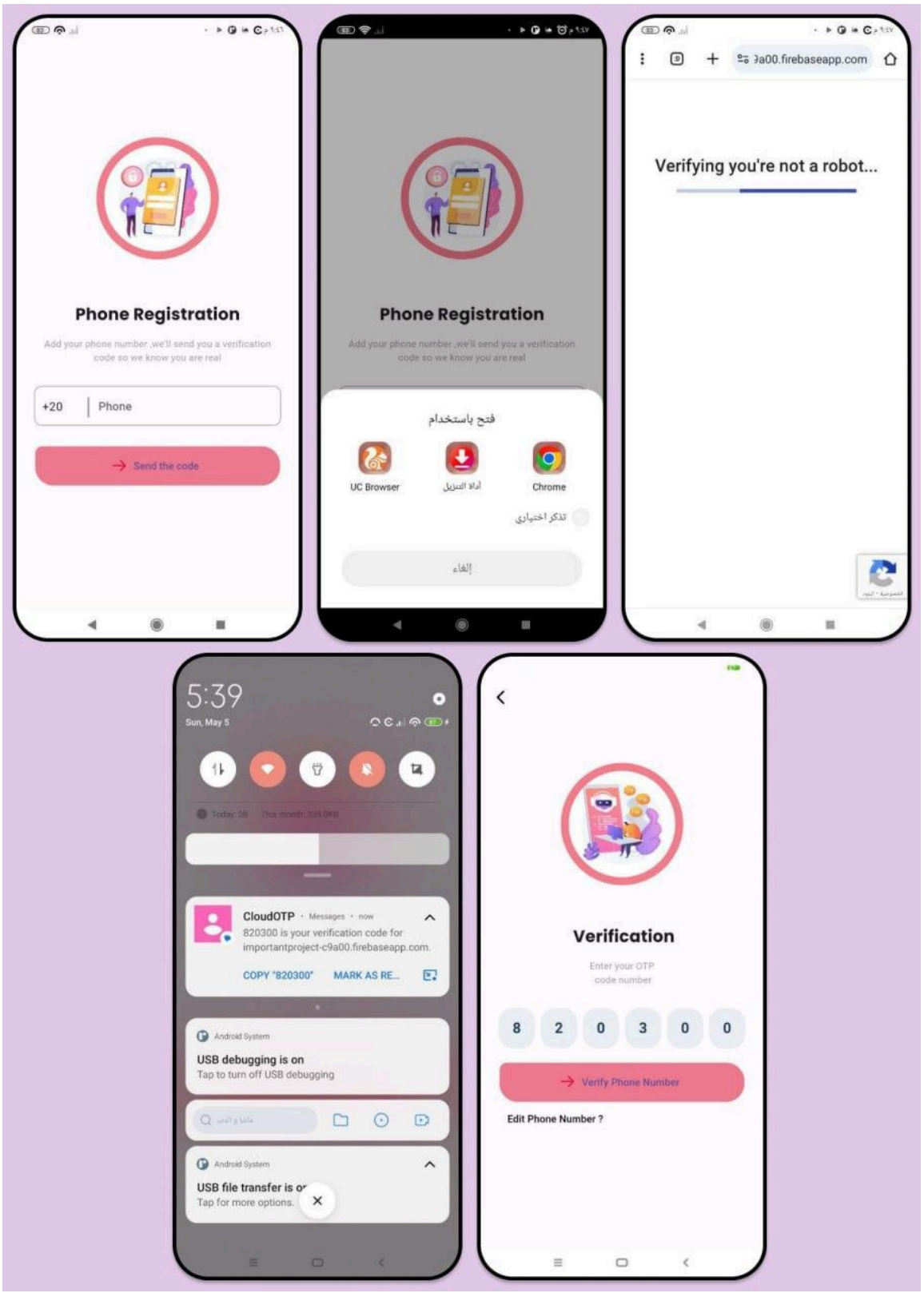
Grants the user full access to their account.

Ensures the platform's security and authenticity of users.

User Feedback: "Your account has been verified and unlockedYou now have full access."

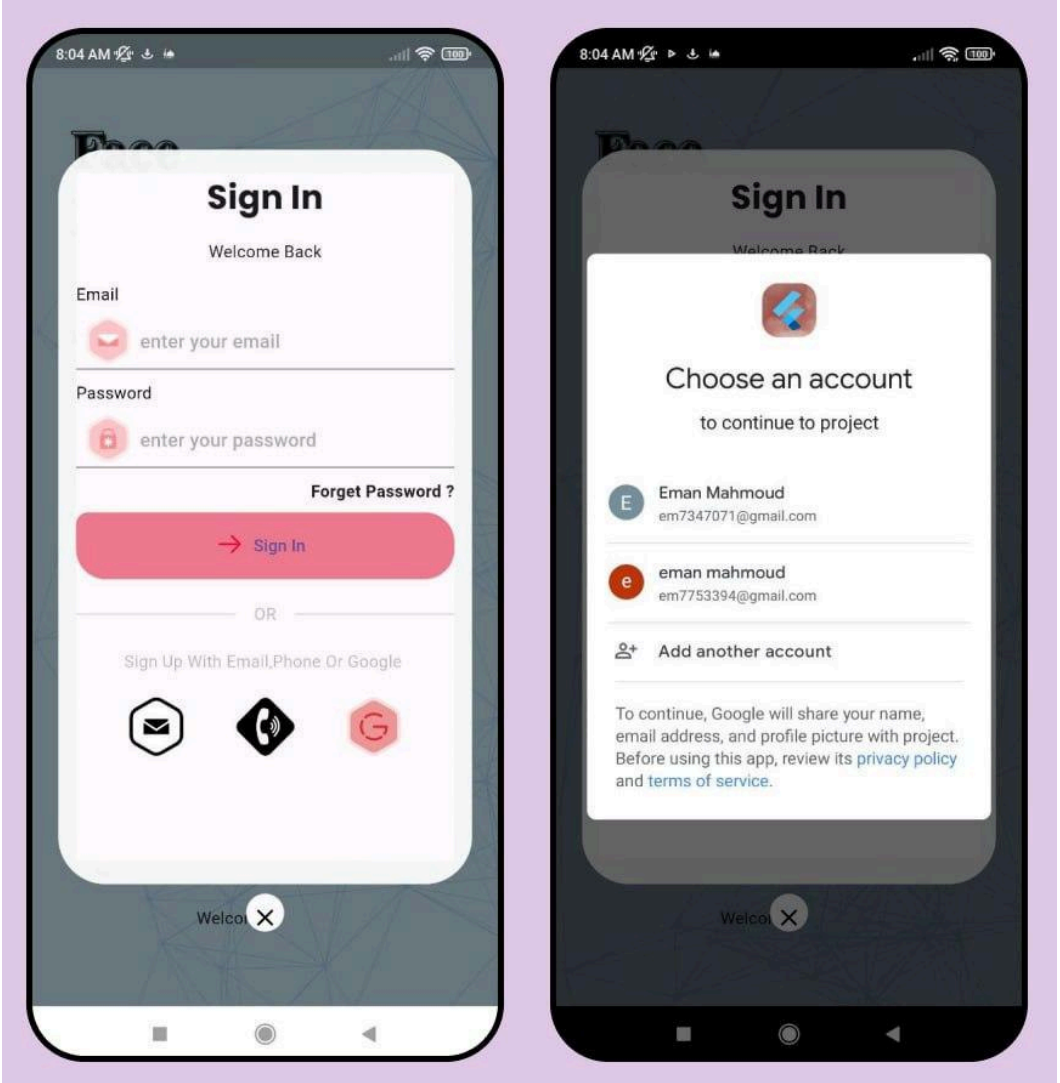

Sign In
Welcome Back
Email
em7347071@gmail.com
Password
Info
Please verify your email before proceeding.
Cancel
Ok
Sign In
Welcome Back
Email
Password
Check Your Email
A password reset link has been sent to your email
Ok
Sign In
Welcome Back
Email
enter your email
Please enter a valu
Password
Error
Error: Unable to establish connection on channel.
Cancel
Ok
Sign Up With Email,Phone Or Google
Welcome
Reset your password
for em7347071@gmail.com
New password
SAVE
Password changed
You can now sign in with your new password

- **Sign-in and Email Verification Process**
    1. If the user doesn't click the verification link from Gmail after sign-up, attempting to sign in without verification triggers an alert instructing the user to check their email and click the verification link for account activation.
    2. In case of a sign-in error, such as incorrect credentials, the system displays an alert guiding the user to rectify their credentials.
    3. When attempting to sign in with an unregistered email, an alert notifies the user that the account doesn't exist and suggests signing up.
    4. If the user forgets their password, clicking "Forgot Password" triggers an alert instructing them to check their email for reset instructions.
    5. Simultaneously, the system sends a reset password email containing a unique link for secure password reset.
    6. Clicking the link directs the user to a secure password reset page within the app.
    7. Upon successful password reset, an alert confirms the change to the user.


Phone Registration
Add your phone number ,we'll send you a verification code so we know you are real
+20
Phone
Send the code
Phone Registration
Add your phone number ,we'll send you a verification code so we know you are real
فتح باستخدام
UC Browser
أداة التنزيل
Chrome
تذكر اختياري
إلغاء
Verifying you're not a robot...
5:39
Sun, May 5
CloudOTP · Messages · now
820300 is your verification code for importantproject-c9a00.firebaseapp.com.
COPY "820300"
MARK AS RE...
Android System
USB debugging is on
Tap to turn off USB debugging
Android System
USB file transfer is or
Tap for more options.
Verification
Enter your OTP code number
8 2 0 3 0 0
Verify Phone Number
Edit Phone Number ?

- **Sign-in with Phone Number and CAPTCHA Verification**

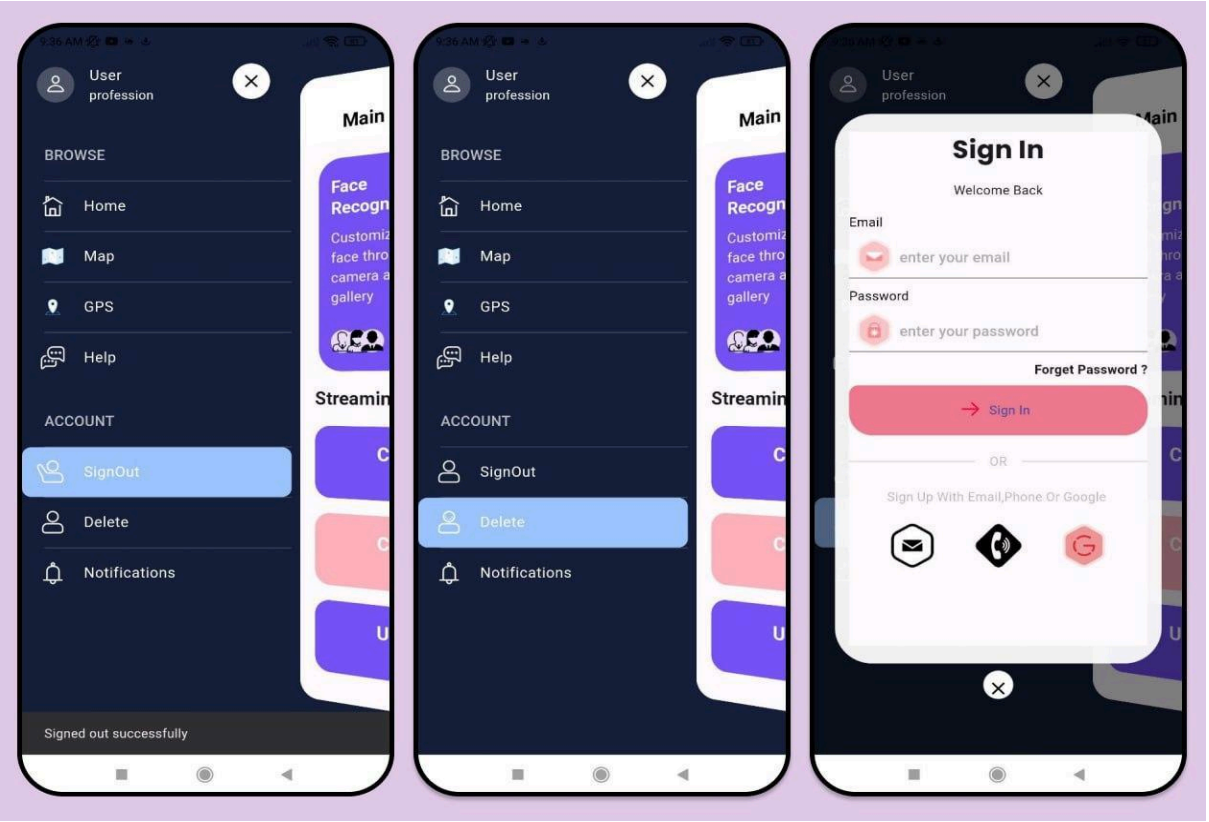
1. Users opt for phone number sign-up for convenience.
2. An anti-robot verification step, like a CAPTCHA, is employed before sending the verification code.
3. Users complete the CAPTCHA or verification task to confirm their humanity.
4. Upon successful verification, a secure verification code is sent via SMS.
5. Users enter the received code to confirm their phone number.
6. Successful verification ensures the authenticity of the phone number for seamless sign-up.



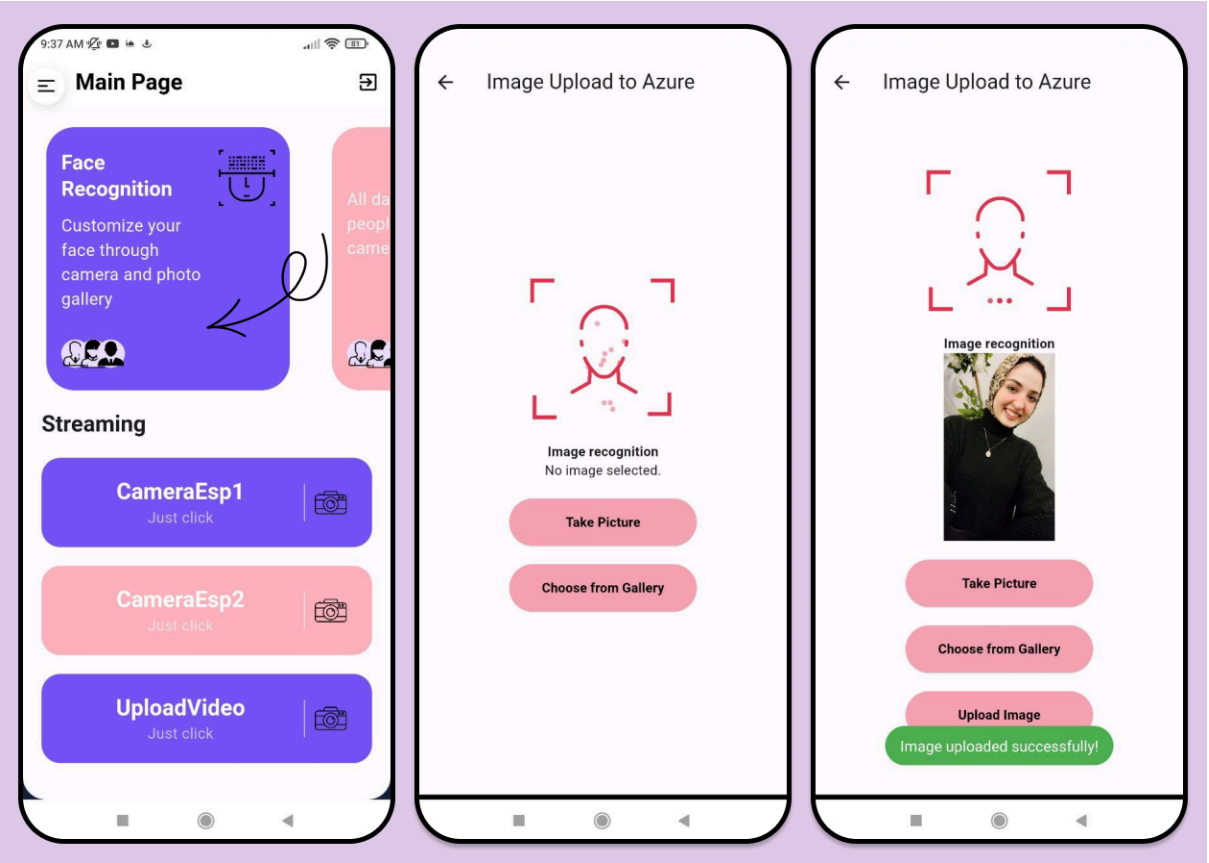
- **Sign-in with Google Account Integration**

1. Users start sign-in by selecting "Sign In with Google."

2. This redirects them to the Google Sign-In page.
3. Users choose their account for secure access to the app.

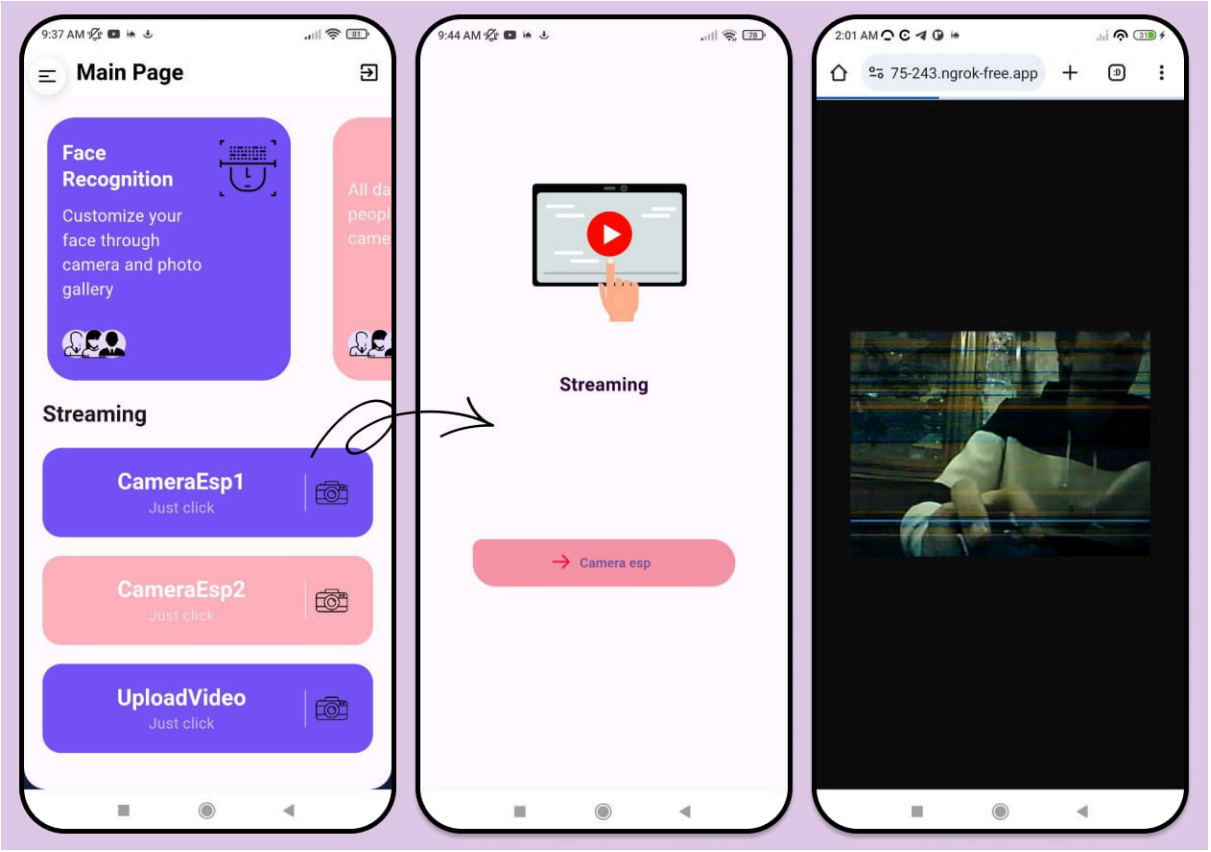


- **<u>Sign Out:</u>**
  Clicking the "Sign Out" icon will log you out of your current account. You'll receive an alert confirming the successful sign-out.
- **<u>Delete Account:</u>**
  If you click the "Delete" icon, your account will be permanently deleted. Afterward, you'll be redirected to the sign-in page to log in with a new account.



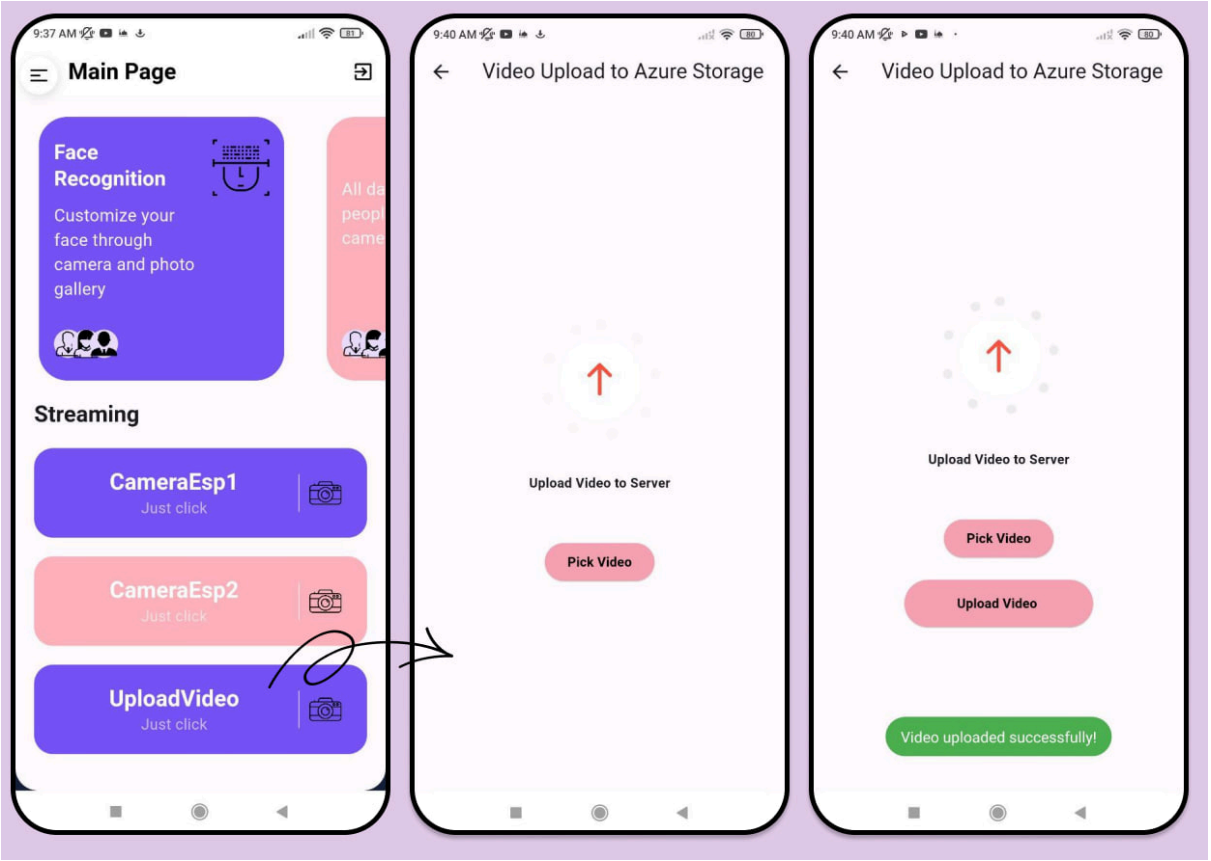
- **6.3 Uploading Images for Recognition:**

  When a user captures an image with the camera, it's sent to Azure Blob Storage for storage and processing. The server generates a unique blob name for the image and stores it in Azure Blob Storage. A link to the uploaded image is constructed using the blob's URL. This link or blob name can be stored in a database or returned to the user for reference.



- **6.4 Live Streaming with ESP Camera:**

  ESP connects to your application server to receive commands or send data.
  It establishes a communication link, typically using protocols like MQTT or HTTP.
  Commands or data are transmitted between ESP and the server for processing or control.
  This connection allows ESP to interact with your application, enabling functionalities like data retrieval, device control, or notifications.

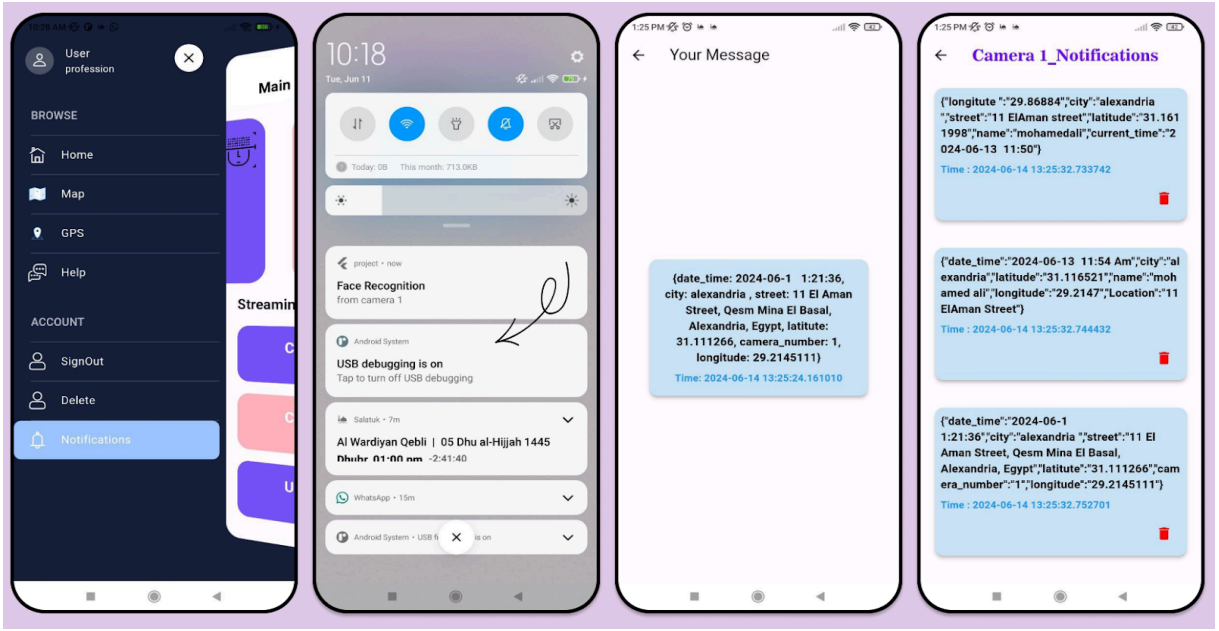


- ## 6.5 Uploading Recorded Video from Live Stream to Server:

  Recorded video files, capturing live streaming sessions, are uploaded to Azure Blob Storage.

  Each video segment or compiled video file is stored as a separate blob with a unique blob name.

  Links to these files are constructed using the blob's URL for easy access and sharing.



- ## 6.6 Notification From Server:

  When a camera recognizes a person, the server sends a notification to your app containing specific details about the recognition event. This information includes the name of the recognized person, the timestamp of the recognition, the geographical coordinates of the location, the address or location details, and the identifier of the camera that captured the photo.

- **Your app is designed to receive these notifications regardless of its current state:**

  1. App is Open:
  If the app is running in the foreground, it immediately displays the notification and stores the details.

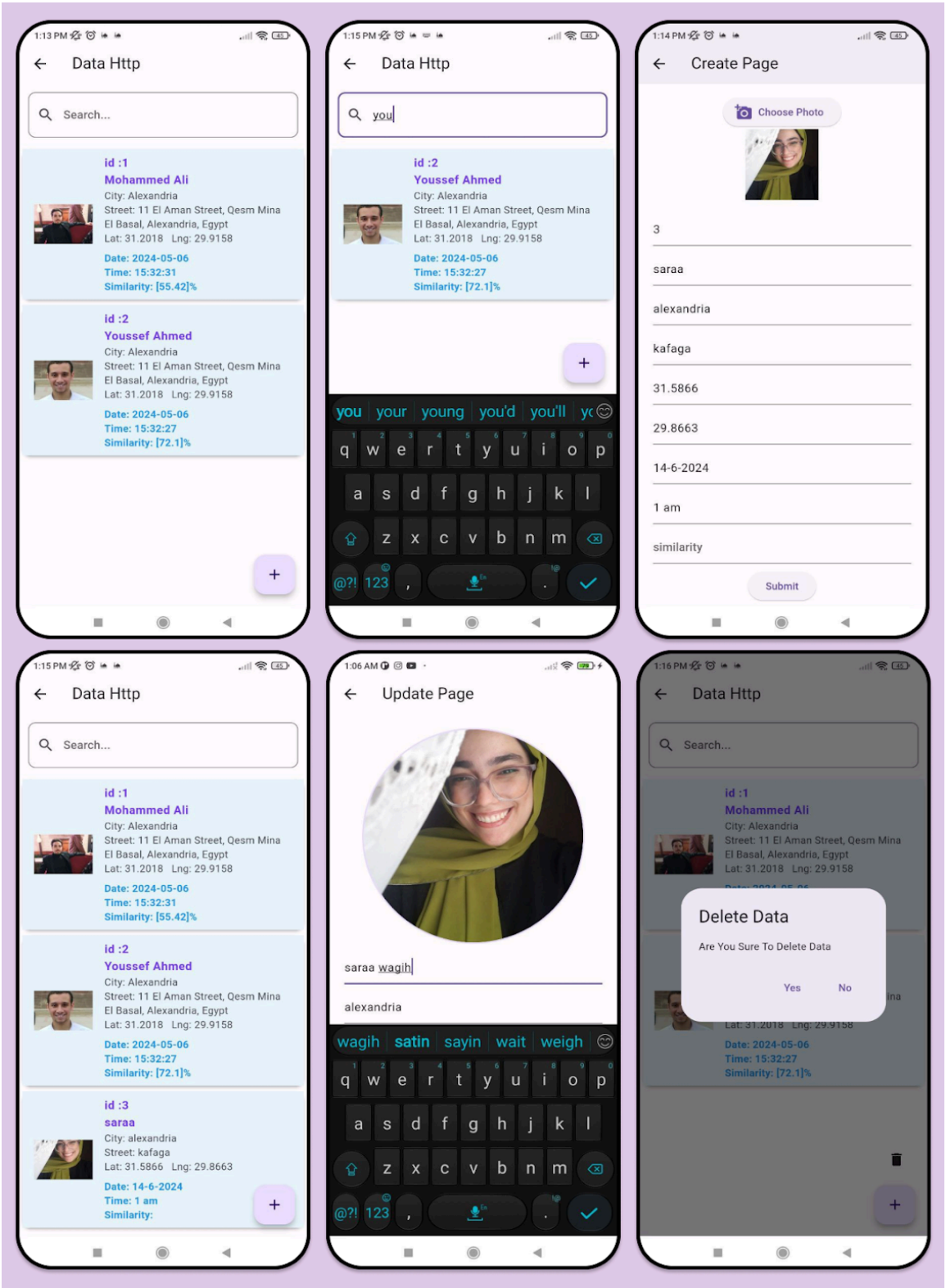
  2. App is in the Background:
  If the app is running in the background, the notification is still delivered. Depending on your app's configuration, it can appear as a push notification, and the data is processed and stored as soon as the app is brought to the foreground.

  3. App is Closed/Terminated:
  If the app is not running at all, the notification is delivered as a push notification. When the user opens the notification, the app launches, and the notification details are processed and stored.

- ## Notifications Management with SharedPreferences

  1. Storing Notifications Locally:
  SharedPreferences : is used to store these notifications persistently on the user's device.
  Notifications are saved in a key-value format, where the key is a unique identifier (such as a timestamp or UUID) and the value is the notification data.

  2. Accessing Stored Notifications:
  Users can access and view the stored notifications within the app.
  A history of notifications can be displayed in a user-friendly manner, such as in a list view, allowing users to easily track and review past events.


Data Http
Search...
id :1
Mohammed Ali
City: Alexandria
Street: 11 El Aman Street, Qesm Mina El Basal, Alexandria, Egypt
Lat: 31.2018 Lng: 29.9158
Date: 2024-05-06
Time: 15:32:31
Similarity: [55.42]%
id :2
Youssef Ahmed
City: Alexandria
Street: 11 El Aman Street, Qesm Mina El Basal, Alexandria, Egypt
Lat: 31.2018 Lng: 29.9158
Date: 2024-05-06
Time: 15:32:27
Similarity: [72.1]%
you
Create Page
Choose Photo
3
saraa
alexandria
kafaga
31.5866
29.8663
14-6-2024
1 am
similarity
Submit
id :3
saraa
City: alexandria
Street: kafaga
Lat: 31.5866 Lng: 29.8663
Date: 14-6-2024
Time: 1 am
Similarity:
Update Page
saraa wagih
alexandria
Delete Data
Are You Sure To Delete Data
Yes
No

- **Overview**

  The application is designed to manage data retrieved from an API hosted on Azure. It handles records of individuals recognized by a camera system, providing functionalities to create, read, update, and delete (CRUD) these records, along with search capabilities to easily find specific entries.

- **6.7 Retrieving Data of Recognized Persons from API**
  - **API Integration:**

    The application integrates with an API hosted on Azure to fetch data.
  - **Data Included**:

    The data typically includes

    ID: A unique identifier for each record.
  - **Name**:

    The name of the recognized individual.
  - **Location Information**:

    City, street, latitude (Lat), and longitude (Lng) details where the individual was recognized.
  - **Recognition Timestamp:**

    Date and time when the recognition occurred.
  - **Similarity Percentage:**

    A percentage indicating the confidence level of the system in recognizing the individual.

- **User Interface Elements**
  - **List View:**

    Displays a list of recognized individuals, each entry includes ID, name, location, timestamp, and similarity percentage.

  - **Floating Action Button:**

    A "+" button at the bottom right of the screen allows users to add new entries.

- **Creating New Entries**
  - **Add Entry:**

    Users can click the "+" button to open a form for inputting new data. This data includes the individual's name and location details, which is then sent to the server via the API and added to the list.

- **Updating Existing Entries**
  - **Edit Functionality:**

    To update an existing entry, users select the entry from the list, opening a form pre-populated with current data. After making changes, the updated data is sent back to the server to overwrite the existing record.

- **Deleting Entries**
  - **Delete Confirmation:**

    Users delete an entry by clicking the trash icon next to it. A confirmation dialog appears asking if they are sure about the deletion to prevent accidental removal. If confirmed, the entry is deleted from the server and removed from the list.

- **Searching for Entries**
  - **Search Bar:**

    A search bar at the top of the screen allows users to search for specific entries.

  - **Real-Time Filtering:**

    As users type into the search bar, the application filters the list in real-time to show only entries that match the search term.

- **Application Use Cases**

  The application is useful in several scenarios, including:
  - **Security Systems:**

    Managing and tracking individuals recognized by security cameras.
  - **Attendance Tracking**:

    Monitoring attendance using facial recognition in educational institutions or workplaces.
  - **Customer Management:**

    Enhancing service in retail or hospitality sectors by recognizing and managing customer data.

- **Workflow**

  1. Recognition Event:

  When a camera system recognizes an individual, their data, including a similarity percentage, is sent to the server

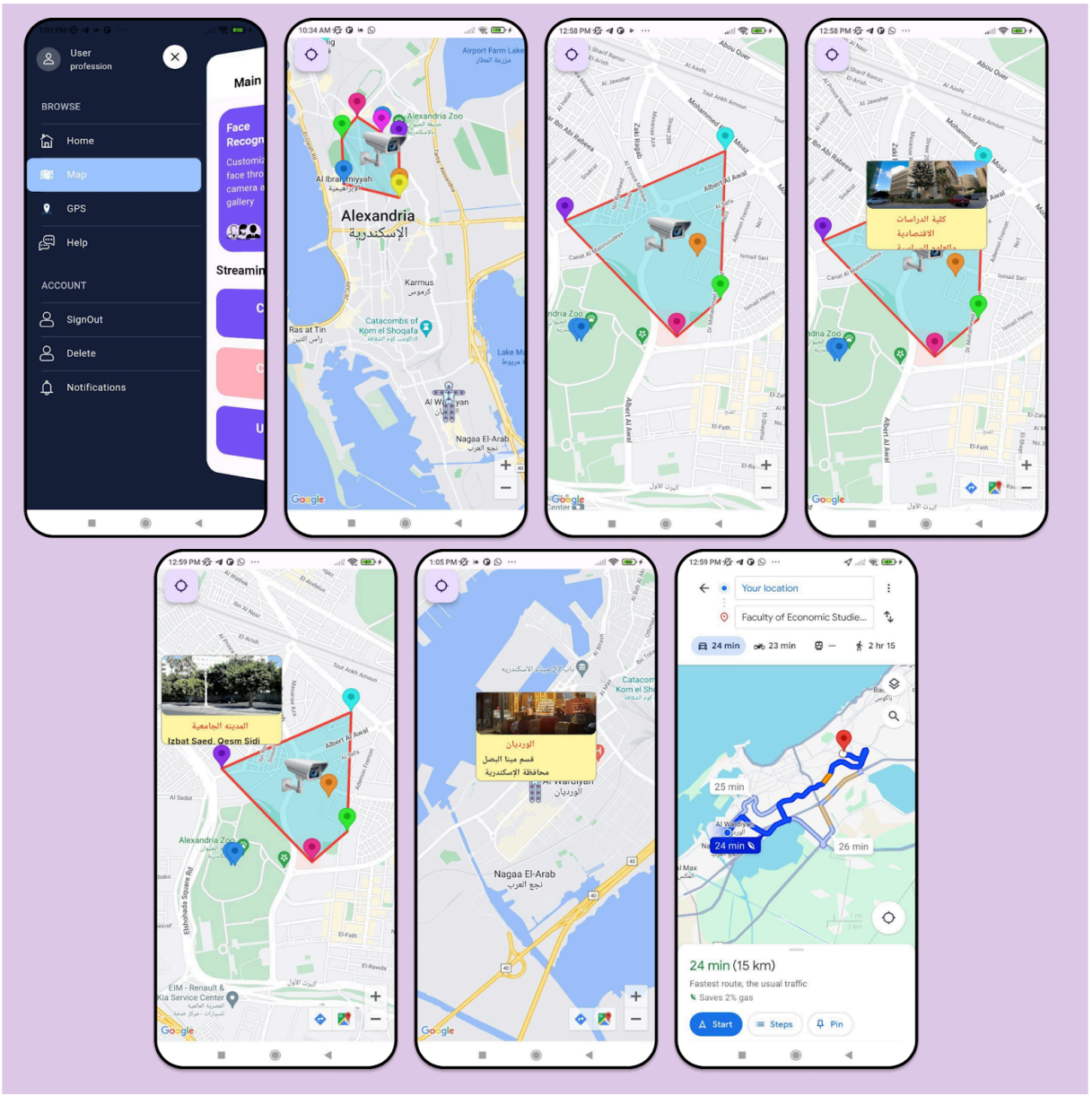
  2. Data Management:

  The application retrieves this data from the server and displays it to the user. Users can manage this data through the application's interface.

  3. **CRUD Operations**:

  Users can create new records, update existing ones, delete unwanted records, and search for specific individuals efficiently.

  4. Confirmation Dialogs: To prevent accidental data loss, the application uses confirmation dialogs before performing critical actions like deleting a record.



- **6.8 Displaying Camera Location and Nearby Places on a Map with Flutter**

- **Key Features**
  Real-time updates of camera location, integration with Google Maps for interactive map functionalities, and display of nearby places using Google Places API.

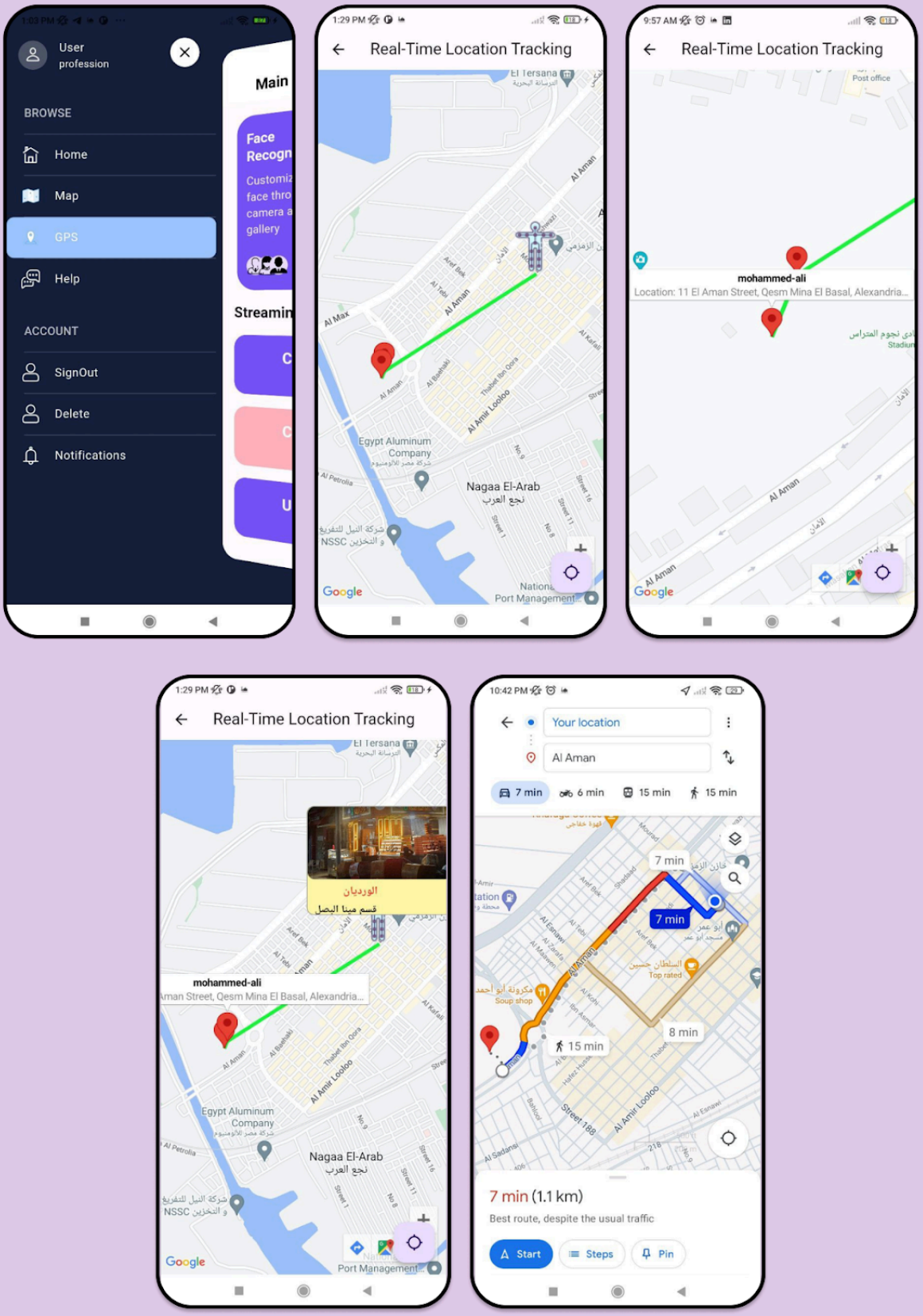
- **Benefits**
  1. Enhanced User Experience:
  Users can view their current location and explore nearby places conveniently.
  2. Integration:
  Seamless integration of map functionalities within Flutter application.
  3. Scalability:
  Can be expanded with additional features like route planning and real-time notifications

- **Key Components for Dynamic Map Features in Flutter**

Markers: Essential for pinpointing specific locations on the map, such as the device's camera position and nearby places. Customizable with icons and labels for clear identification.

Define Polyline Points: Determine the points that will define the polyline shape around the camera's location. This could be a circular area or a polygonal shape.

InfoWindow: Enhances user interaction by displaying additional details when markers are tapped, such as place names or custom information related to the location.


User
profession
BROWSE
Home
Map
GPS
Help
ACCOUNT
SignOut
Delete
Notifications
Main
Face Recogn
Streamin
Real-Time Location Tracking
Egypt Aluminum Company
Nagaa El-Arab
نجع العرب
Google
Real-Time Location Tracking
mohammed-ali
Location: 11 El Aman Street, Qesm Mina El Basal, Alexandria...
Google
Real-Time Location Tracking
الورديان
قسم مينا البصل
mohammed-ali
Egypt Aluminum Company
Nagaa El-Arab
نجع العرب
Google
Your location
Al Aman
7 min
6 min
15 min
15 min
7 min
7 min
8 min
15 min
7 min (1.1 km)
Best route, despite the usual traffic
Start
Steps
Pin

- **6.9 Getting Current Location and Real-Time Data from Firebase for Tracking Users**
    - **Gps icon:**

Clicking on the map will display your current location.

- **Focus Rectangle:**

Tapping the focus rectangle will center the map on a person's updated location on the Google Map.

1. Add Dependencies

Add the necessary packages to your `pubspec.yaml` file to enable Google Maps, geolocation, and Firebase Realtime Database functionalities.
google_maps_flutter: For integrating Google Maps into your app.
geolocator: For obtaining the device's current location.
firebase_core: To initialize Firebase in your Flutter app.
firebase_database: For accessing Firebase Realtime Database.

2. Initialize Firebase

Set up Firebase in your Flutter app. This includes:
Registering your app with Firebase.
Adding the Firebase configuration files to your project.
Initializing Firebase in your app's entry point (`main.dart`).

3. Get Current Location

Use the Geolocator package to fetch the device's current location. This involves:
Requesting location permissions from the user.
Using the Geolocator's methods to get the current position (latitude and longitude).

4. Set Up Google Maps

Integrate Google Maps into your app by:
Setting up a Google Maps API key.
Adding a `GoogleMap` widget to your app's UI to display the map.
Configuring map settings, such as initial camera position and markers.

5. Real-Time Data from Firebase

Use the Firebase Realtime Database to receive and track location data:

Establish a connection to the Firebase Realtime Database.
Create database references to listen for changes in the tracked individuals' location data.
Use listeners to update the map whenever new data is received

6. Display Locations on Map

Plot the tracked individuals' locations on the Google Map:
Convert the latitude and longitude data received from Firebase into `LatLng` objects.
Add markers to the map at the corresponding locations.
Update the markers in real-time as new data comes in.

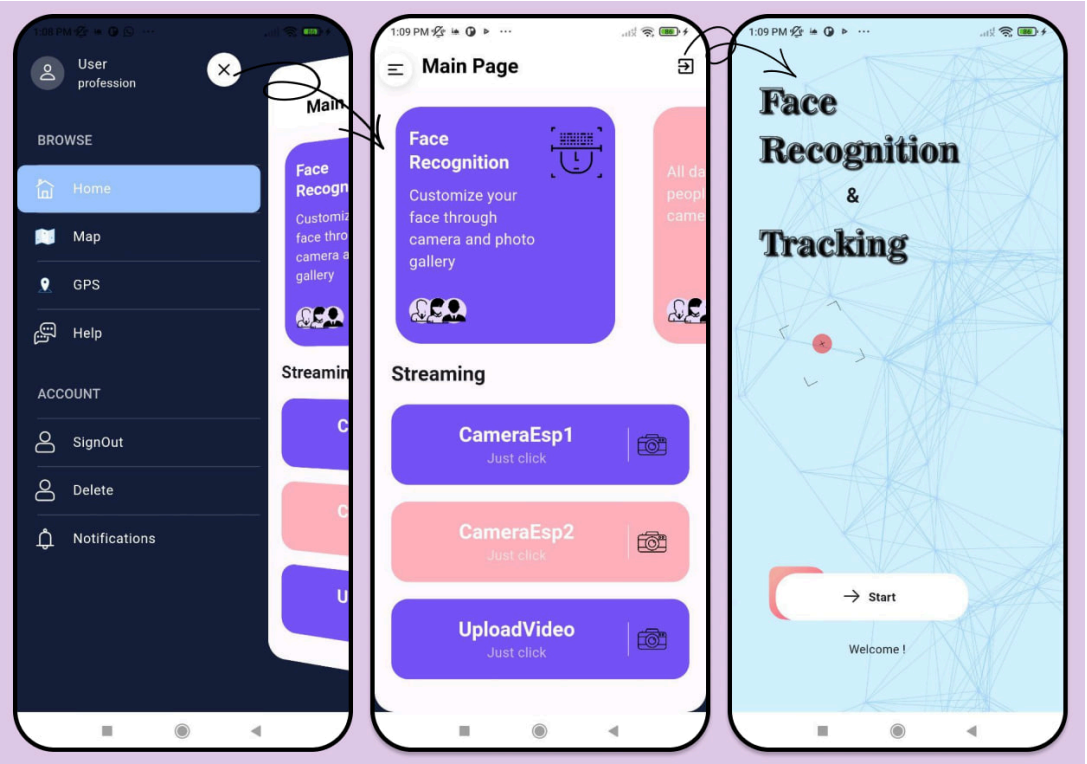
7. Show User's Location & Space - Time Between user and individuals' locations on Map

```
Future<Position> getLatAndLong() async {
  return await Geolocator.getCurrentPosition().then((value) => value);
}
```

Display the user's current location on the map:

- Use the current location data obtained from the Geolocator.
  Add a marker or a custom icon to indicate the user's position.
  Optionally, move the camera to focus on the user's current location.

- Flutter app can track and display the locations of individuals in real-time on a Google Map, while also showing the user's current location. The Firebase Realtime Database provides continuous updates, ensuring that the location data on the map is always up-to-date. This setup is particularly useful for applications requiring real-time location tracking, such as monitoring delivery services, tracking family members, or other similar use cases.



- **Navigation Between Sidebar Menu and Home Page**

1.X Icon:

Clicking the "Home" icon will navigate you to the MainPage.

2. Left Circle:

When you click the circle on the left side, it will navigate you to the sidebar menu.

3. Right Arrow:

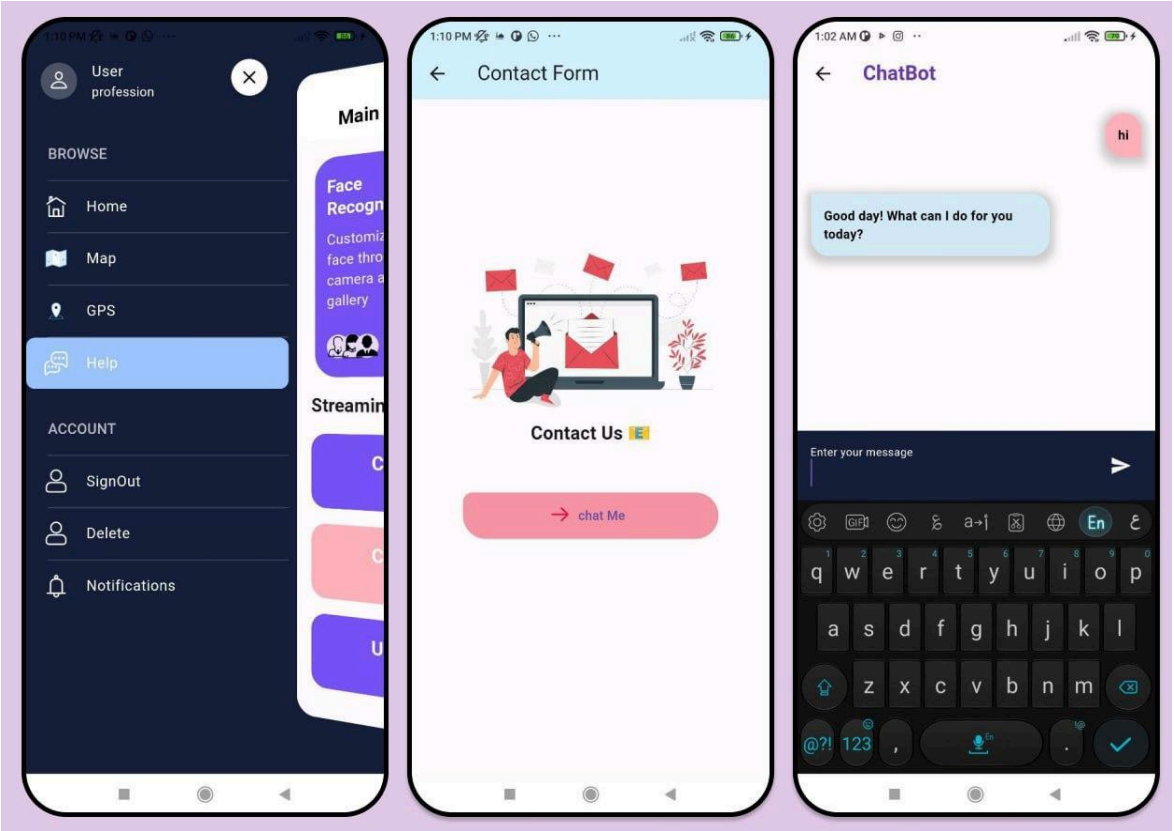
Clicking the arrow on the right side will navigate you out to the Welcome screen.

package:packages

particles_fly: ^0.0.8

rive: ^0.9.1

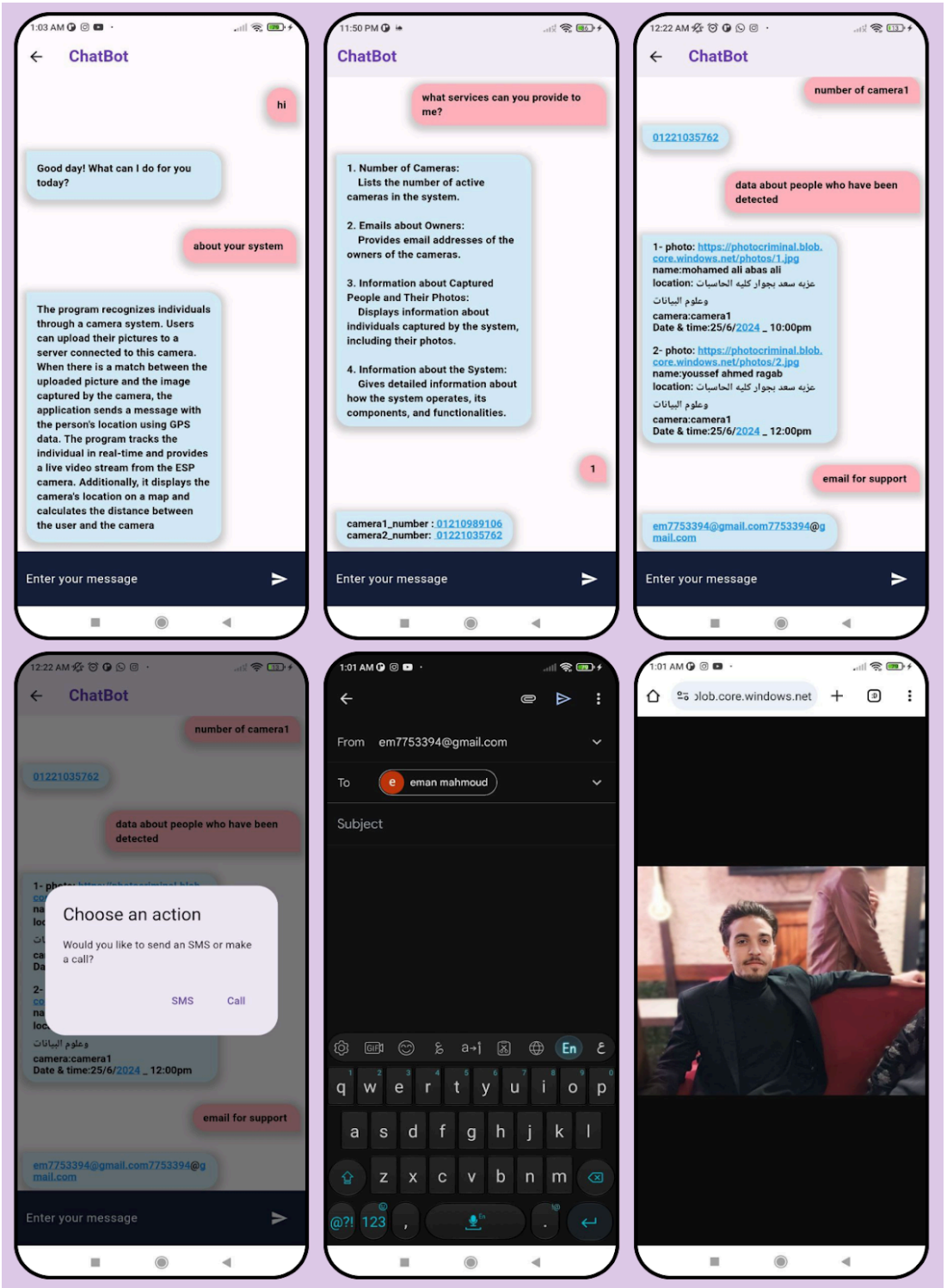
lottie: ^2.1.0



- ## 6.10 Chatbot in Face Recognition App
  - ### Purpose of the Chatbot

    User Assistance:

    Answers user queries related to the system
    (e.g., how to upload images, how recognition works).
    Information Retrieval: Provides users with information about past notifications, system status, and other relevant data.
    System Interaction: Offers real-time updates and instructions, enhancing user experience.


ChatBot
hi
Good day! What can I do for you today?
about your system
The program recognizes individuals through a camera system. Users can upload their pictures to a server connected to this camera. When there is a match between the uploaded picture and the image captured by the camera, the application sends a message with the person's location using GPS data. The program tracks the individual in real-time and provides a live video stream from the ESP camera. Additionally, it displays the camera's location on a map and calculates the distance between the user and the camera
Enter your message
ChatBot
what services can you provide to me?
1. Number of Cameras:
Lists the number of active cameras in the system.
2. Emails about Owners:
Provides email addresses of the owners of the cameras.
3. Information about Captured People and Their Photos:
Displays information about individuals captured by the system, including their photos.
4. Information about the System:
Gives detailed information about how the system operates, its components, and functionalities.
1
camera1_number : 01210989106
camera2_number: 01221035762
Enter your message
ChatBot
number of camera1
01221035762
data about people who have been detected
1- photo: https://photocriminal.blob.core.windows.net/photos/1.jpg
name:mohamed ali abas ali
location: عزبه سعد بجوار كليه الحاسبات وعلوم البيانات
camera:camera1
Date & time:25/6/2024 _ 10:00pm
2- photo: https://photocriminal.blob.core.windows.net/photos/2.jpg
name:youssef ahmed ragab
location: عزبه سعد بجوار كليه الحاسبات وعلوم البيانات
camera:camera1
Date & time:25/6/2024 _ 12:00pm
email for support
em7753394@gmail.com7753394@gmail.com
Enter your message
ChatBot
number of camera1
01221035762
data about people who have been detected
Choose an action
Would you like to send an SMS or make a call?
SMS
Call
camera:camera1
Date & time:25/6/2024 _ 12:00pm
email for support
em7753394@gmail.com7753394@gmail.com
Enter your message
From em7753394@gmail.com
To e eman mahmoud
Subject
olob.core.windows.net

- **Key Features**

  1. Query Handling:
  Answers questions about system operation, steps to follow, and troubleshooting tips

  2. Data Information:
  Provides details about recent Datas
  (e.g., names, times, locations of recognized individuals).

  3. System Status:
  Responds to inquiries about the current status of the camera and the recognition system.

  4. Guidance and Instructions:
  Guides users on how to use different app features (e.g., uploading pictures, viewing live streams).

- **Integrating with Dialogflow API**

  1. Create a Dialogflow Agent:
  Go to the Dialogflow Console.
  Create a new agent and configure its basic settings.

  2. Define Intents:
  Create intents for common user queries (e.g., "How does the system work?", "Show me the latest notifications", "What is the current status of the camera?").

  3. Train the Agent:
  Provide example phrases for each intent to improve recognition accuracy.
  Train the agent regularly.

  4. Integrate Dialogflow with Flutter:
  Use the `dialogflow_flutter` package.
  Authenticate using a service account key.
  Implement functions to send user queries to Dialogflow and handle responses.

  5. Handle Responses:
  Process Dialogflow responses to provide meaningful information.
  Use responses to trigger specific actions within the app (e.g., fetching notification details from SharedPreferences).

- **Scenario**

  User Input: The user types "Can you tell me about your system?" into the chatbot.
  Dialogflow Request: The app sends this query to Dialogflow.

Intent Matching: Dialogflow matches the query to the corresponding intent.
Response Generation: Dialogflow generates a response and sends it back to the app.
User Response: The chatbot displays the response explaining how the system works.

- ## Additional Features

1. Email Navigation:
Clicking an email address navigates the user to their email client to send an email.

2. SMS and Call Navigation:
Prompt: Clicking a phone number prompts the user to choose between sending an SMS or making a phone call.
SMS Option: If SMS is chosen, the app navigates to the SMS app to compose a message.
Call Option: If a phone call is chosen, the app navigates to the phone dialer with the number prefilled.

3. Photo Link Navigation:
Clicking a photo navigates the user to a link or page displaying the photo.

- ## Querying Services

When a user asks the chatbot "What are the services?", the chatbot provides **the following:**

1. Number of Cameras:
Lists the number of active cameras in the system.

2. Emails about Owners:
Provides email addresses of the owners of the cameras.

3. Information about Captured People and Their Photos:
Displays information about individuals captured by the system, including their photos.

4. Information about the System:
Gives detailed information about how the system operates, its components, and functionalities.

- ## 6.11 package used:

dependencies:
awesome_dialog: ^3.0.0
camera: ^0.10.5+6
cloud_firestore: ^4.13.3

cloud_functions: ^4.5.6
cupertino_icons: ^1.0.2
firebase_auth: ^4.15.0
firebase_core: ^2.25.3
firebase_messaging: ^14.7.6
firebase_storage: ^11.5.3
flutter_svg: ^1.1.6
geocoding: ^2.1.1
geolocator: ^10.1.0
get_it: ^7.2.0
google_maps_flutter: ^2.5.1
google_sign_in: ^6.1.2
image_picker: ^1.0.7
intl: ^0.19.0
lottie: ^2.1.0
uuid: ^4.3.3
particles_fly: ^0.0.8
path_provider: ^2.1.1
pin_code_fields: ^8.0.1
pinput: ^3.0.1
provider: ^6.1.1
rive: ^0.12.4
custom_info_window: ^1.0.1
firebase_app_check: ^0.2.1+12
flutter_local_notifications: ^16.3.2
fluttertoast: ^8.2.4
url_launcher: ^6.1.5
azblob:
http:
mime: ^1.0.5
path: ^1.8.3
shared_preferences: ^2.0.15
video_player: ^2.8.3
dialog_flowtter: ^0.3.3
flutter_polyline_points: ^1.0.0
permission_handler: ^11.3.1