كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

Helwan University
Faculty of Computers and Artificial Intelligence
Computer Science Department

Mind's Eye

Mohamed Mohamed Nader	**20180532**
Yehya Sayed Yehya Ahmed	**20180691**
Omar Amgad Othman	**20180365**
Mohamed Hesham Al-Sadek	**20180547**
Aya Muhammed Taha	**20180150**

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

Dr/ Mohamed El-Saieed

June 2022

Abstract

- With all this success and technological development and everything that the Internet and AI is capable of doing today, it is strange how much technology has helped people of all kinds, yet people with disabilities do not get half the quality of life or attention. We did not consider people with chronic diseases such as Alzheimer's and people with disabilities such as the visually impaired and those who suffer from blindness who has different ability dealing with their daily tasks.

- So, we developed new project that will help them in their life, where the application is concerned by providing appropriate features for reading and presenting information in different and new ways to make it easy and clear for the visually impaired people.

- Our project two parts, the first one is for Alzheimer's people and the second one is for the visually impaired people. There is also a hardware OCR glass that will paired to the cell phone so that it can help the person to caption the photo easier without the need to use the phone physically.

- This project aims to help the blind people to be able to read any text from a book, paper, banner, or any text written around them and the people diagnosed with Alzheimer to be able to remember the people around them.

- We used Ai technologies like OCR, Face Detection, Image Captioning, and Object Detection. These features we use it to help the visually impaired people to have a fair live in this technological development.

- It was targeted that to make the disables have an easy way to learn and to make decisions in this life so, we do our best to get this approach through our app.

Keywords

Alzheimer’s, Cognitive Services, Image Captioning, Object Detection, OCR, and Visually Impaired.

Acknowledgement

- At first, we thank ALLAH for giving us the power to achieve this work, and for the blessings from ALLAH to all of us, El 7amdullelah.

- We would like to express my special thanks to our mentor Dr/ Mohamed Al-Saieed for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to us during the project’s completion. In this aspect, we are eternally grateful to you.

Table of Contents

Chapter1: Introduction

1.1: Overview

- According to World Health Organization, around 285 million people live with visual impairments worldwide, of which 45 million are blind. If we are going to talk about visually impaired people in Egypt, according to the World Health Organization (WHO), about 2.2 million people with visual impairment in Egypt 900,000 of which are totally blind.

- However, having someone for guidance 24/7 makes the blind person feel dependent, not to mention the effects it has on his privacy; he often has to explain the reasons why he wants to go outdoors to the person accompanying him.

- So, our application targets many features to help the visually impaired and blinded people which are:

1. Image captioning: to describe an image
2. Face detection: to help in the recognition of the people around
3. Labels detection: to help in detecting the landmarks
4. Object detection: to help in detecting the objects included in an image
5. OCR: to help in reading the printed text located in an image
6. Currency recognition: to help in recognizing the local currency

- Talkback feature has been added to our application to make the usability of the application easier for visually impaired people. Talkback is Google's free screen reader for Android devices. The software responds to familiar touch and swipes commands, allowing users to interact with websites and apps. and its gestures can be customized.

- There is another part in the app which is for the people diagnosed with Alzheimer to help them remember the people using the face recognition feature to analyze the faces and store it in the database and provides the user with putting information for each recognized face.

1.2: Problem Statement

- The main problem is that the disabled people they feel dependent, the visually impaired people always need to understand the surroundings.

- So, our project target to help the visually impaired people with the new technology features so that they can feel like normal people in society.

- Also, we try to help Alzheimer people to communicate with the people that they know easily and recognize them faster from the application.

1.3: Scope and Objectives

- The disable people deserve to have a normal life, some privacy, normal day, normal habits like reading and learning.

- Our project aims to combine some important technologies to help the visually impaired people and Alzheimer people with AI technologies.

- Our main motive of this project is to help the visually impaired people, not accurately but to make their life a little bit easier and become self-dependent. In this project, the glasses we will be using would be able to take pictures via camera.

- The captured photo will be sent over the Bluetooth to our mobile application and from there he/she can choose any feature to help in a specific task, we added a memory card to the chip so that it will help the doctors to figure out patients usage to the chip.

- We will also help Alzheimer people to recognize the people that they know faster by using face recognition model to make the communication to them easier.

- we are targeting the visually impaired and the blind to have a normal, and easy daily life; and help them navigate independently.

- we also helping the people with short time memory to help them recognize the people around them faster with the application.

1.4: Work Methodology

- One of the more traditional project management methodologies, Waterfall is a linear, sequential design approach where progress flows downwards in one direction—like a waterfall. Originating in the manufacturing and construction industries, its lack of flexibility in design changes in the earlier stages of the development process is due to it becoming exuberantly more expensive because of its structured physical environments.

- The methodology was first introduced in an article written in 1970 by Winston W. Royce (although the term 'Waterfall' wasn't used), and emphasizes that you're only able to move onto the next phase of development once the current phase has been completed. The phases are followed in the following order:

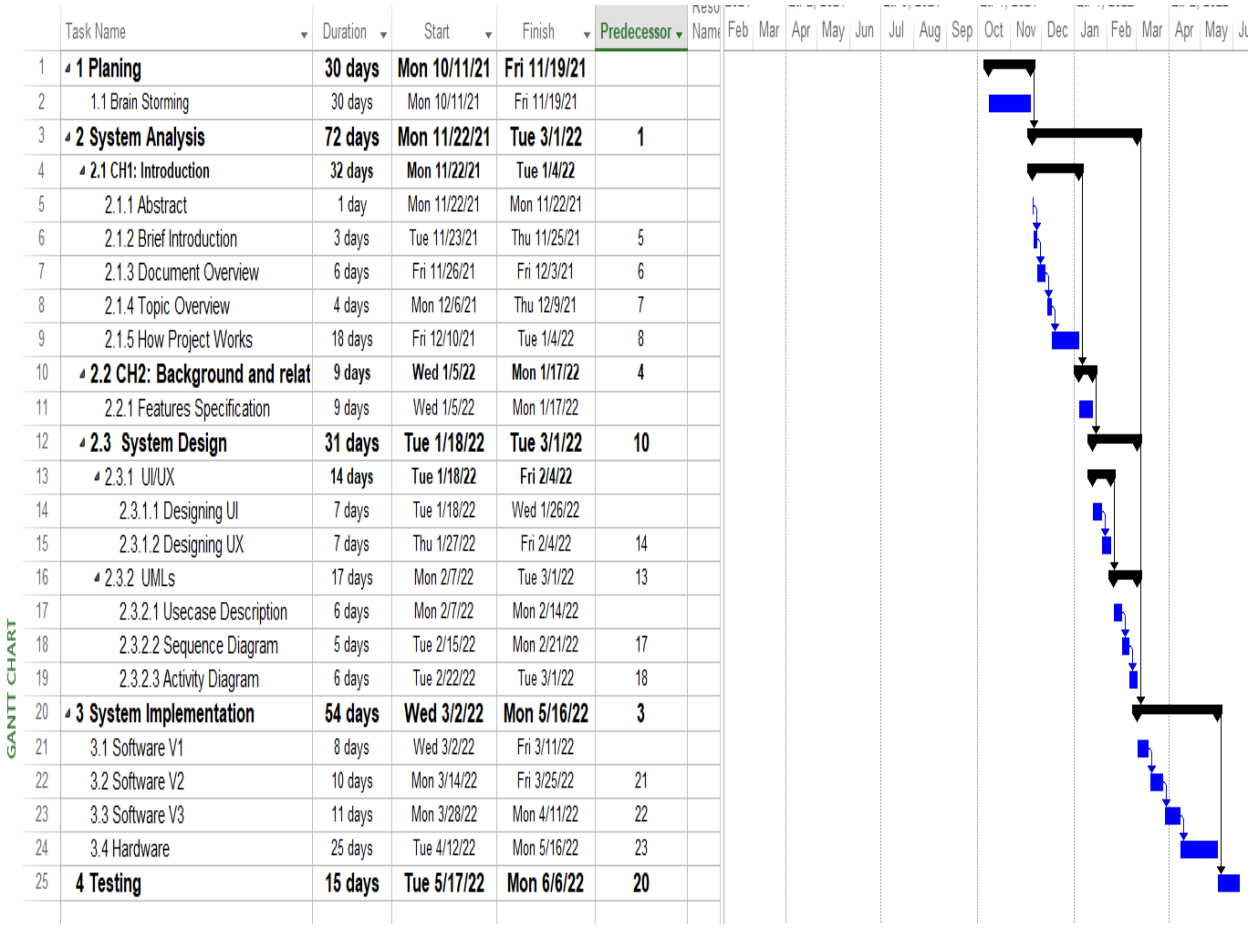
1. System and software requirements
2. Analysis
3. Design
4. Coding
5. Testing
6. Operations

- Waterfall is a project management methodology that stresses the importance of documentation. The idea is that if a worker was to leave during the development process, their replacement can start where they left off by familiarizing themselves with the information provided on the documents.

- Pre-Agile saw the Waterfall methodology being used for software development, but there were many issues due to its non-adaptive design constraints, the lack of customer feedback available during the development process, and a delayed testing period.

- Best suited for: larger projects that require maintaining stringent stages and deadlines, or projects that have been done various times over where chances of surprises during the development process are relatively low.

1.5: Work plan (Gantt Chart)



	Task Name	Duration	Start	Finish	Predecessor
1	**1 Planing**	**30 days**	**Mon 10/11/21**	**Fri 11/19/21**	
2	1.1 Brain Storming	30 days	Mon 10/11/21	Fri 11/19/21	
3	**2 System Analysis**	**72 days**	**Mon 11/22/21**	**Tue 3/1/22**	**1**
4	**2.1 CH1: Introduction**	**32 days**	**Mon 11/22/21**	**Tue 1/4/22**	
5	2.1.1 Abstract	1 day	Mon 11/22/21	Mon 11/22/21	
6	2.1.2 Brief Introduction	3 days	Tue 11/23/21	Thu 11/25/21	5
7	2.1.3 Document Overview	6 days	Fri 11/26/21	Fri 12/3/21	6
8	2.1.4 Topic Overview	4 days	Mon 12/6/21	Thu 12/9/21	7
9	2.1.5 How Project Works	18 days	Fri 12/10/21	Tue 1/4/22	8
10	**2.2 CH2: Background and relat**	**9 days**	**Wed 1/5/22**	**Mon 1/17/22**	**4**
11	2.2.1 Features Specification	9 days	Wed 1/5/22	Mon 1/17/22	
12	**2.3 System Design**	**31 days**	**Tue 1/18/22**	**Tue 3/1/22**	**10**
13	2.3.1 UI/UX	**14 days**	**Tue 1/18/22**	**Fri 2/4/22**	
14	2.3.1.1 Designing UI	7 days	Tue 1/18/22	Wed 1/26/22	
15	2.3.1.2 Designing UX	7 days	Thu 1/27/22	Fri 2/4/22	14
16	2.3.2 UMLs	17 days	Mon 2/7/22	Tue 3/1/22	13
17	2.3.2.1 Usecase Description	6 days	Mon 2/7/22	Mon 2/14/22	
18	2.3.2.2 Sequence Diagram	5 days	Tue 2/15/22	Mon 2/21/22	17
19	2.3.2.3 Activity Diagram	6 days	Tue 2/22/22	Tue 3/1/22	18
20	**3 System Implementation**	**54 days**	**Wed 3/2/22**	**Mon 5/16/22**	**3**
21	3.1 Software V1	8 days	Wed 3/2/22	Fri 3/11/22	
22	3.2 Software V2	10 days	Mon 3/14/22	Fri 3/25/22	21
23	3.3 Software V3	11 days	Mon 3/28/22	Mon 4/11/22	22
24	3.4 Hardware	25 days	Tue 4/12/22	Mon 5/16/22	23
25	**4 Testing**	**15 days**	**Tue 5/17/22**	**Mon 6/6/22**	**20**

1.6: Similar Systems

- Microsoft Seeing AI

  Amazing application by Microsoft. It offers lots of amazing features. Yet, it has key drawbacks that make it far from perfect:

  - It requires internet connection
  - It relies on the smartphone itself, which is a double-edged weapon–while utilizing an already-existing smartphone, it requires holding it all day
  - It only supports iPhone users
  - It doesn't support Arabic

- Lookout by Google

  Identifies important items in your environment and reports the information it believes is relevant. This might include things like exit signs, the location of bathroom, people or objects nearby, and even a text in book.

- Be My Eyes

  This application pairs blind people with sighted volunteers who help them identify objects using smartphone app and camera.

- Cash Reader

  Whether you need to hand out cash or count bills given to you, check out the Cash Reader app. This tool not only speaks the denomination but also vibrates and displays it in large contrasting numbers on screen for those discrete situations.

- TapTapSee: Identify Objects Through Photos: TapTapSee is designed to help the blind and visually impaired identify objects they encounter in their daily lives. Simply double tap the screen and take a photo of anything, at any angle. You'll hear the app speak the identification back to you (Note: Requires Voiceover to be turned on)

- My Vision Helper: My Vision Helper is an app that focuses on the following: magnification, color contrast enhancement, and optical character recognition (OCR). It is also integrated with Apple Speech Recognition, which means it can be operated (almost) exclusively via voice commands.

- Voice Brief: Voice Brief reads content aloud, such as the weather, blog posts, or emails. This app is useful for both the blind and sighted alike.

Chapter 2: Background and Related Work

2.1 Background:

In this chapter we will explain the used technologies and the most relevant prior works referred to this project.

We used Azure's Computer vision services in our project as it provides APIs that use Azure's image recognition capabilities to produce descriptions of what inside the image

2.2: Microsoft Azure: Computer Vision

- Computer Vision is a field of computer science that aims to enable computers to identify and understand objects and people in images.
- Computer vision focuses on the performance and the automation of the tasks that replicate human capabilities.
- Computer Vision seeks to replicate both the way humans see and the way humans make sense of what they see.
- Computer Vision can analyze an image and generate a human-readable phrase that describes its contents. The algorithm returns several descriptions based on different visual features, and each description is given a confidence score. The final output is a list of descriptions ordered from highest to lowest confidence.
- The practical range of applications of computer vision technology makes it the centerpiece of many modern innovations and solutions. Computer vision can run in the cloud or on-premises.

2.2.1: How computer vision works

- Computer vision applications use input from sensing devices, artificial intelligence, machine learning and deep learning to clone the way the human vision system works. Computer vision applications run on trained algorithms based on large amounts of visual data or images in the cloud. They recognize patterns in this visual data and use these patterns to determine the content of other images.

2.2.2: How an image is analyzed with computer vision (Generally)

- A sensing device captures an image. The sensing device is often just a camera, but could be a video camera, medical imaging device or any other type of device that captures an image for analysis.
- The image is then sent to an interpreting device. The interpreting device uses pattern recognition to break the image down, compare the patterns in the image against its library of known patterns and determine if any of the content in the image is a match. The pattern could be something general, like the appearance of a certain type of object or it could be based on unique identifiers such as facial features.
- A user requests specific information about an image and the interpreting device provides the information requested based on its analysis of the image.

2.2.3: How an image is analyzed with computer vision (In Our Project)

- After the camera attached to the glass capture the picture, the picture will be transferred to our app that will be downloaded in the cell phone.
- The image then will be received by the app that will use the Azure API to analyze the image using pattern recognition to break the image down, compare the patterns in the image against its library of known patterns and determine if any of the content in the image is a match. The pattern could be something general, like the appearance of a certain type of object or it could be based on unique identifiers such as facial features.

2.2.4: Deep learning and computer vision

- Modern computer vision applications aren't using then old way statistical methods for analyzing images anymore and increasingly relying on what is known as deep learning. With deep learning, a computer vision application runs on a type of algorithm called a neural network. This can provide a more detailed analysis of the image. In addition, deep learning allows a computer vision program to retain the information from each analyzed image, making it more accurate as you see it.
- Deep learning has achieved superhuman level performance in computer vision that can do object classification, object detection, or semantic segmentation of different features. Natural language processing models, on the other hand, work well for tasks such as named entity recognition, text classification, etc.

2.3: Optical character recognition

2.3.1: What is OCR (Optical Character Recognition)?

- OCR is an electronic or mechanical conversion of images of typed, handwritten, or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast). Think of it as the process of turning the analog data into digital

2.3.2: How does Optical Character Recognition Work?

- Let's have a look at three basic steps of optical character recognition: image pre-processing; character recognition; and the post-processing of the output.

Step 1: Image Pre-Processing in OCR

- OCR software often pre-processes images to improve the chances of successful recognition. The aim of image pre-processing is an improvement of the actual image data. In this way, unwanted distortions are suppressed, and specific image features are enhanced. These two processes are important for the following steps.

Step 2: Character Recognition in OCR

- For the actual character recognition, it is important to understand what "feature extraction" is. When the input data is too large to be processed, only a reduced set of features is selected. The features selected are expected to be the important ones while those that are suspected to be redundant are ignored.

By using the reduced set of data instead of the initial large one, the performance is increased.

- For the process of OCR, this is important as the algorithm must detect specific portions or shapes of a digitized image.

Step 3: Post-Processing in OCR

- Post-processing is another error correction technique that ensures the high accuracy of OCR. The accuracy can be further improved if the output is restricted by a lexicon. That way, the algorithm can fall back to a list of words that are allowed to occur in the scanned document for example.
- OCR is not only used to identify proper words but can also read numbers and codes. This is useful for identifying long strings of numbers and letters, such as serial numbers used in many industries.
- To better deal with different types of input OCR, some providers started to develop specific OCR systems. These systems can deal with the special images, and to improve the recognition accuracy, even more, they combined various optimization techniques.
- For example, they used business rules, standard expressions, or rich information contained in the color image. This strategy of merging various optimization techniques is called “application-oriented OCR” or “customized OCR”. It is used in applications such as business card OCR, invoice OCR, and ID card OCR.
- The system identifies letters and numbers in images and convert that text into machine-encoded text that can be read by other computer applications or edited by users.

2.4: Image captioning

2.4.1: What is image captioning?

- Image Captioning connects computer vision and natural language processing to describe the content of an image.
- Image Captioning is the process of the generation of the description of an image in text form after analyzing the image. The captioning is done based on objects, locations, face recognized, and actions detected in the image.

2.4.2: How image captioning works?

- When the user chooses the image description feature the app will use retrofit to send a request to the API to describe the image and returns the description in English with high accuracy.

2.4.3: Use the API

- The image description feature is part of the Analyze Image API. You can call this API through a native SDK or through REST calls. Include Description in the visualFeatures query parameter. Then, when you get the full JSON response, simply parse the string for the contents of the "description" section.

2.4.4: What is Retrofit?

- Retrofit is a type-safe HTTP networking library used for Android and Java. Retrofit was even better since it was super-fast, offered better functionality, and even simpler syntax. Most developers since then have switched to using Retrofit to make API requests.

2.4.5: The functions used in code

- These functions use the APIs from Microsoft Azure

- ImageCaptioning(img):

This function takes the image as an input and provides a full description for it.

- detect_document(img):

This function takes the image as an input and provides us with all the printed text inside of this image.

- translate_text(text, target):

This function takes the text generated and translates it into Arabic.

2.5: Face detection:

- Face detection is AI-based computer technology that is used to extract and identify human faces from digital images. When integrated with biometric security systems (particularly, facial recognition ones), this kind of technology is what makes it possible to monitor and track people in real-time. In applications that use facial tracking, analysis, and recognition, face detection typically works as the first step and has a significant impact on how sequential operations within the app will perform.
- Face detection helps with facial analysis by identifying the parts of a video or an image that should be focused on when determining gender, age, and emotions. Similarly, with facial recognition systems (which create "faceprint" maps of facial features), face detection data is included in the system's algorithms. And why? Face detection helps determine which parts of the video or image are needed to produce a faceprint.

2.5.1: What is the Azure Face service

- The Azure Face service provides AI algorithms that detect, recognize, and analyze human faces in images. Facial recognition software is important in many different scenarios, such as identity verification, touchless access control, and face blurring for privacy.

2.5.2: Face detection and analysis:

- Face detection is required as a first step in all the other scenarios. The Detect API detects human faces in an image and returns the rectangle coordinates of their locations. It also returns a unique ID that represents the stored face data. This is used in later operations to identify or verify faces.
- Optionally, face detection can extract a set of face-related attributes, such as head pose, age, emotion, facial hair, and glasses. These attributes are general predictions, not actual classifications. Some attributes are useful to ensure that your application is getting high-quality face data when users add themselves to a Face service. For example, your application could advise users to take off their sunglasses if they're wearing sunglasses.
- Face detection is the process of locating human faces in an image and optionally returning different kinds of face-related data.

2.5.3: Face rectangle:

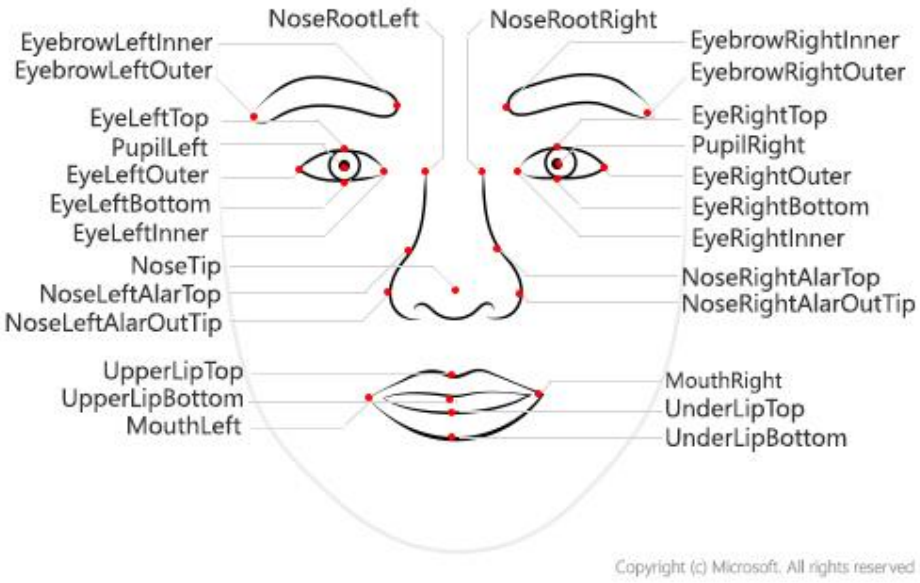
- Each detected face corresponds to a faceRectangle field in the response. This is a set of pixel coordinates for the left, top, width, and height of the detected face. Using these coordinates, you can get the location and size of the face. In the API response, faces are listed in size order from largest to smallest.

2.5.4: Face ID:

- The face ID is a unique identifier string for each detected face in an image. No image will be stored. Only the extracted face feature(s) will be stored on server. The faceId is an identifier of the face feature and will be used in Face - Identify, Face - Verify, and Face - Find Similar. The stored face features will expire and be deleted at the time specified by faceIdTimeToLive after the original detection call.

2.5.5: Face landmarks:

- Face landmarks are a set of easy-to-find points on a face, such as the pupils or the tip of the nose. By default, there are 27 predefined landmark points. The following figure shows all 27 points:



- The coordinates of the points are returned in units of pixels.

2.5.6: Used Attributes from Azure Face API:

- Age. The estimated age in years of a particular face.
- Emotion. A list of emotions with their detection confidence for the given face. Confidence scores are normalized, and the scores across all emotions add up to one. The emotions returned are happiness, sadness, neutral, anger, contempt, disgust, surprise, and fear.
- Gender. The estimated gender of the given face. Possible values are male, female, and genderless.

2.5.7: Functions:

face_client.face.detect_with_stream()

- Detect human faces in an image, return face rectangles, and optionally with faceIds, landmarks, and attributes.
- No image will be stored. Only the extracted face feature will be stored on server. The faceId is an identifier of the face feature and will be used in Face - Identify, Face - Verify, and Face - Find Similar. The stored face feature(s) will expire and be deleted at the time specified by faceIdTimeToLive after the original detection call.
- Optional parameters include faceId, landmarks, and attributes. Attributes include age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure, noise, mask, and qualityForRecognition. Some of the results returned for specific attributes may not be highly accurate.
- JPEG, PNG, GIF (the first frame), and BMP format are supported. The allowed image file size is from 1KB to 6MB.
- Up to 100 faces can be returned for an image. Faces are ranked by face rectangle size from large to small.
- For optimal results when querying Face - Identify, Face - Verify, and Face - Find Similar ('returnFaceId' is true), please use faces that are: frontal, clear, and with a minimum size of 200x200 pixels (100 pixels between eyes).
- The minimum detectable face size is 36x36 pixels in an image no larger than 1920x1080 pixels. Images with dimensions higher than 1920x1080 pixels will need a proportionally larger minimum face size.
- Different 'detectionModel' values can be provided. To use and compare different detection models, please refer to How to specify a detection model

- Different 'recognitionModel' values are provided. If follow-up operations like Verify, Identify, Find Similar are needed, please specify the recognition model with 'recognitionModel' parameter. The default value for 'recognitionModel' is 'recognition_01', if latest model needed, please explicitly specify the model you need in this parameter. Once specified, the detected faceIds will be associated with the specified recognition model. More details, please refer to Specify a recognition model.

2.5.8: URI Parameters

- Image : JPEG, PNG, GIF (the first frame), and BMP format are supported. The allowed image file size is from 1KB to 6MB.
- Endpoint : Supported Cognitive Services endpoints (protocol and hostname, for example: https://westus.api.cognitive.microsoft.com).
- detectionModel : Name of detection model. Detection model is used to detect faces in the submitted image. A detection model name can be provided when performing Face - Detect or (Large)FaceList - Add Face or (Large)PersonGroup - Add Face. The default value is 'detection_01', if another model is needed, please explicitly specify it.
- We used the Detection_03 model because currently it has the most accurate.
- recognitionModel : Name of recognition model. Recognition model is used when the face features are extracted and associated with detected faceIds, (Large)FaceList or (Large)PersonGroup. A recognition model name can be provided when performing Face - Detect or (Large)FaceList - Create or (Large)PersonGroup - Create. The default value is 'recognition_01', if latest model needed, please explicitly specify the model you need.
- We used 'recognition_04' model.

- returnFaceAttributes : Analyze and return the one or more specified face attributes in the comma-separated string like "returnFaceAttributes=age,gender". The available attributes depends on the 'detectionModel' specified. 'detection_01' supports age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure, noise, and qualityForRecognition. While 'detection_02' does not support any attributes and 'detection_03' only supports mask and qualityForRecognition.
- Additionally, qualityForRecognition is only supported when the 'recognitionModel' is specified as 'recognition_03' or 'recognition_04'. Note that each face attribute analysis has additional computational and time cost.

2.6: Object detection:

- In our application, the object detection feature is a service provided by Microsoft Azure cognitive services using computer vision using image analysis.
- In azure computer vision using image analysis services, object detection is similar to tagging.
- What is tagging in the image analysis service of computer vision?
- Computer Vision can return content tags for thousands of recognizable objects, living beings, scenery, and actions that appear in images. Tags are not organized as taxonomy and do not have inheritance hierarchies. A collection of content tags forms the foundation for an image description displayed as human-readable language formatted in complete sentences. When tags are ambiguous or not common knowledge, the API response provides hints to clarify the meaning of the tag in the context of a known setting.
- After uploading an image, the computer vision algorithm can output tags based on the objects identified in the image.

2.6.1: Functions

In object detection we use in python function called "detect_object_in_stream". This function described in the ComputerVisionClientOperationsMixin class. It:

- Performs object detection on the specified image.
- Has two input methods are supported -- (1) Uploading an image or (2) specifying an image URL, in our application we are uploading an image from the camera to a server, not by specifying an image URL.
- Has an output as a JSON file or we can use some parameters the ones we only want as we are going to output the text as a written text and voice so we don't need to use the whole JSON file, in our case we use only object_property parameter that introduces the object name, and confidence parameter to introduce the accuracy.

2.7: Google Cloud vision API

- Cloud Vision allows developers to easily integrate vision detection features within applications, including image labeling, face, and landmark detection, optical character recognition (OCR), and tagging of explicit content.
- Cloud vision currently have the following features: face detection, landmark detection, logo detection, label detection, text detection, and more.
- Cloud vision can detect objects and labels automatically, it can detect and classify multiple objects and labels including the location of each object in the image.
- Vision API offers powerful pre-trained machine learning models through REST and RPC APIs. Assign labels to images and quickly classify them into millions of predefined categories. Detect objects and faces, read printed and handwritten text, and build valuable metadata into your image catalog.

2.8: Detect labels

- In our application, the detect labels feature is a google cloud vision service.
- The Vision API can detect and extract information about entities in an image, across a broad group of categories.
- Labels can identify general objects, locations, activities, animal species, products, and more. If you need targeted custom labels, Cloud AutoML Vision allows you to train a custom machine learning model to classify images.
- Labels are returned in English only, but with the use of google cloud translation API, we can translate them to any language we want.
- The Vision API can perform label detection on a local image by sending the content of the image file as a base64 encoded string in the body of the request.
- The Vision API can also perform label detection directly on an image file located in Google Cloud Storage or on the Web without the need to send the contents of the image file in the body of the request.

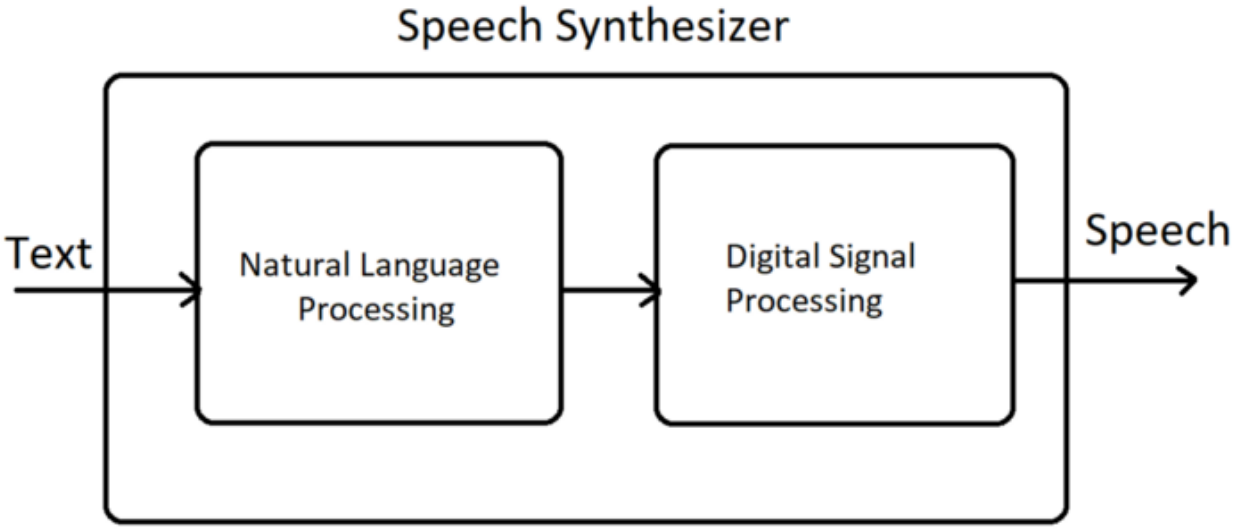
2.8.1: Functions

detect labels(img):

- a function that takes the image as an input and then go to google cloud base service which is label detection and detect labels in the image and then return the response.

2.9: Text-to-Speech:

- The text-to-speech (TTS) is the process of converting words into a vocal audio form. The program, tool, or software takes an input text from the user, and using methods of natural language processing understands the linguistics of the language being used, and performs logical inference on the text. This processed text is passed into the next block where digital signal processing is performed on the processed text. Using many algorithms and transformations this processed text is finally converted into a speech format. This entire process involves the synthesizing of speech. Below is a simple block diagram to understand the same.



- The gTTS module can be used extensively on other languages such as French, German, Hindi, Arabic, etc., as well. This is extremely useful when there is a communication barrier, and the user is unable to convey his messages to people. Text-to-speech is a great help to the visually impaired people or people with other disabilities as it can help them by assisting in the text to speech

translation. There are also many ideas possible with the gTTS module and it can be used for other languages as well.

- **gTTS** (Google Text-to-Speech)is a Python library and CLI tool to interface with Google Translate text-to-speech API. We will import the gTTS library from the gtts module which can be used for speech translation.
- gTTS – Google Text-to-Speech.
- An interface to Google Translate's Text-to-Speech API.

Parameters

1. **text** (*string*) – The text to be read.
2. **tld** (*string*) – Top-level domain for the Google Translate host, i.e *https://translate.google.<tld>*. Different Google domains can produce different localized 'accents' for a given language. This is also useful when `google.com` might be blocked within a network but a local or different Google host (e.g. `google.cn`) is not. Default is `com`.
3. **lang** (*string, optional*) – The language (IETF language tag) to read the text in. Default is `en`.
4. **slow** (*bool, optional*) – Reads text more slowly. Defaults to `False`.
5. **lang_check** (*bool, optional*) – Strictly enforce an existing `lang`, to catch a language error early. If set to `True`, a `ValueError` is raised if `lang` doesn't exist. Setting `lang_check` to `False` skips Web requests (to validate language) and therefore speeds up instanciation. Default is `True`.
6. **pre_processor_funcs** (*list*) –
7. A list of zero or more functions that are called to transform (pre-process) text before tokenizing. Those functions must take a string and return a string.
8. **tokenizer_func** (*callable*) –A function that takes in a string and returns a list of string (tokens).

2.10: Talkback Feature:

2.10.1: TalkBack Overview:

- TalkBack is Google's free screen reader for Android devices. The software responds to familiar touch and swipe commands, allowing users to interact with websites and apps. When activated, TalkBack announces where the user's focus is located, enabling people to control their phones, tablets, and other Android devices without using visual cues. In certain apps, users can input other touch and voice commands.
- As part of our series of articles on assistive technologies, we'll review some of TalkBack's unique features — and provide tips for using the software to evaluate mobile accessibility.

2.10.2: How Google TalkBack Improves Mobile Experiences:

- TalkBack helps the user browse the content of the screen and understand where to click. As a user drags their finger along their screen and they point to a button or a link on a website, the tool will ask if the user wants to click and provide them with a variety of options for proceeding. The software interprets user gestures for navigation. For example, rather than using the device's "back" button when browsing the web, the user can swipe down into the left to navigate to the previous page.
- TalkBack is part of the Android Accessibility Suite, and according to Google, it's installed on more than 5 billion devices. Here are some of the screen reader's basic features:

  - **Gesture Support.** Users navigate by dragging their finger(s) along the screen in a predefined pattern that corresponds to commonly used commands.

- **Voice Commands.** By using TalkBack with Google Assistant, users can navigate with dozens of predefined voice commands.
- **Braille Support.** Users can enter 6-dot braille to control TalkBack. Currently, the screen reader only supports Unified English Braille.

In addition to TalkBack, Google provides a variety of assistive technologies through the Android Accessibility Suite including live captions, hearing aid support, display magnification, and Action Blocks (customizable buttons that trigger routine actions on the Android home screen).

By enabling these tools, Android users can customize their devices to perform tasks quickly and comfortably. However, like all assistive technologies, the Android Accessibility Suite relies on content creators. To ensure that your mobile app or website works with TalkBack and other tools, you'll need to follow the best practices of digital accessibility

2.10.3: TalkBack gestures:

TalkBack gestures help you navigate and perform frequent actions on your Android device.

Single-finger & multi-finger gestures :

- Single-finger gestures are available on all devices.
- Single-finger and multi-finger gestures are available on updated Android R, Pixel 3, and other OEM devices like Samsung Galaxy.

To check if your device supports multi-finger gestures, while TalkBack is on, three-finger tap.

- **If you get the TalkBack menu:** Your device supports both multi-finger gestures and one-finger gestures.
- **If you don’t get the TalkBack menu:** Your device only supports one-finger gestures. To open the TalkBack menu, swipe down then right, or swipe up and right.

2.10.4: Customize TalkBack gestures:

You can assign most TalkBack gestures to most actions. For example, assign the “Next” reading control to swipe down. You can also reset assignments to their default settings.

Important: Two-finger swipe gestures can’t be reassigned.

To customize gestures:

1. Open the TalkBack menu.
    - **On devices with multi-finger gestures:** Three-finger tap. Or, in one motion, swipe down then right.
    - **On devices without multi-finger gestures (prior to updated Android R with TalkBack 9.1):** In one motion, swipe down then right.

Tip: If your Android device has a fingerprint sensor, you can use fingerprint gestures with TalkBack. Select **TalkBack settings** › **Customize gestures**. Choose the gesture that you want to assign to a different action. Choose the action you want to assign to the gesture.

Common gestures

- 

Drag one finger

Explore your screen and hear audible feedback for what is being touched.

- 

Double-tap anywhere on the screen

Opens or activates the item that you last touched (vocalized).

- 

Swipe up or down using two fingers

Scroll within lists or pages if selected. Equivalent to a vertical swipe.

- 

Swipe left or right using two fingers

Change pages and screens when possible. Equivalent to a horizontal swipe.

- 

Swipe right using one finger

Move the focus to the next item.

- 

Swipe left using one finger.

Move the focus to the previous item.

- 

Swipe up or down using one finger

Cycle through navigation mode: “pages”, “by default (elements)”, “characters”, “words”, “lines” et “paragraphs”.

2.11: Alzheimer Section:

2.11.1: Overview about Alzheimer's disease:

- Alzheimer's disease is a progressive neurologic disorder that causes the brain to shrink (atrophy) and brain cells to die. Alzheimer's disease is the most common cause of dementia — a continuous decline in thinking, behavioral and social skills that affects a person's ability to function independently.
- The early signs of the disease include forgetting recent events or conversations or close people. As the disease progresses, a person with Alzheimer's disease will develop severe memory impairment and lose the ability to carry out everyday tasks.
- Medications may temporarily improve or slow progression of symptoms. These treatments can sometimes help people with Alzheimer's disease maximize function and maintain independence for a time. Different programs and services can help support people with Alzheimer's disease and their caregivers.
- So, we develop in our application a section to help people with Alzheimer's disease to (remember the day's tasks and save the important conversations and appointments) remember and recognize their close people and save a new person who he/she wants to remember later by taking a single picture and save his/her name and remind them to take their medication in time.

2.11.2: Face recognition:

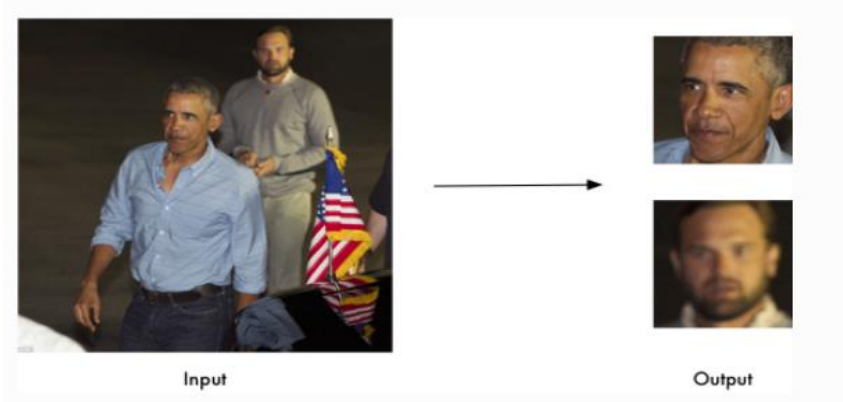
- This feature helps people with Alzheimer's disease to remember the people who they know and interact with in their daily life by taking one single picture of a person's face and saving his/her name and their general data, after this, he/she can take a photo of any person he/she wants to make sure that he/she knew this person or not by face recognition feature, by using face recognition python library.

2.11.3: How it works:

- Recognize and manipulate faces from Python or from the command line with the world’s simplest face recognition library.

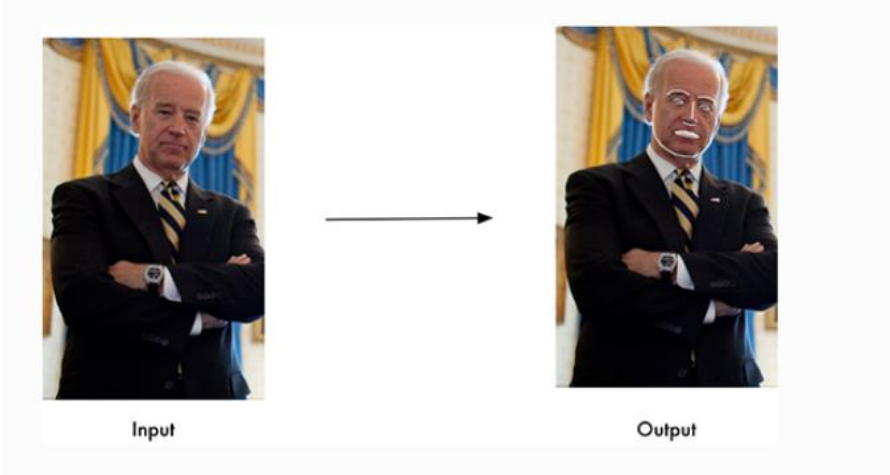
1- Find faces in pictures

- Find all the faces that appear in a picture:



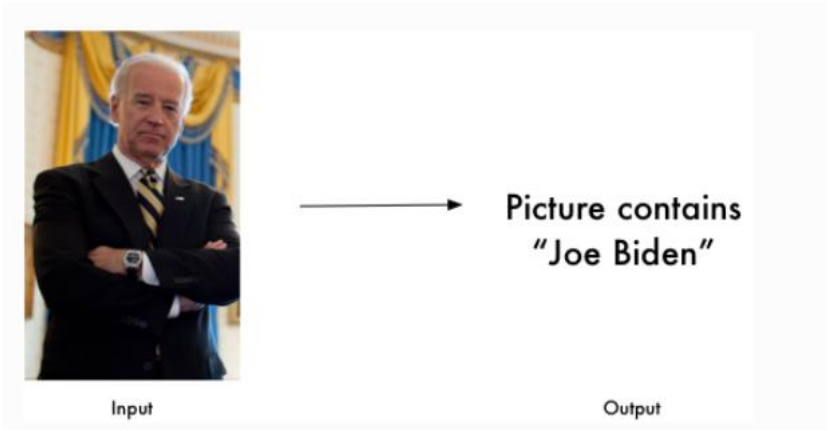
2- Find and manipulate facial features in pictures:

- Get the locations and outlines of each person’s eyes, nose, mouth, and chin.

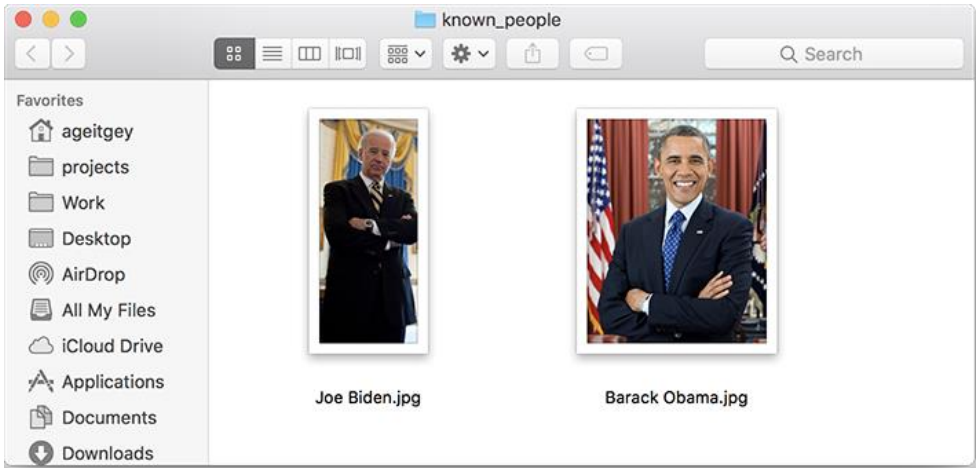


3- Identify faces in pictures:

- Recognize who appears in each photo.



- First, you need to provide a folder with one picture of each person you already know. There should be one image file for each person with the files named according to who is in the picture:



- Then in you simply run the command `face recognition`, passing the folder of known people and the folder (or single image) with unknown people and it tells you who is in each image:

2.10.4: All function used in this feature:

face_recognition.load_image_file () :

- That loads the image into a NumPy array

- face_recognition.face_locations(): (Face Detection step)

find/localize the faces of her resulting in a list of face boxes. We pass two parameters to the method:

1. rgb: Our RGB image.

2. model: Either CNN or HOG (this value is contained within our command line arguments dictionary associated with the "detection_method" key). The CNN method is more accurate but slower. HOG is faster but less accurate.

- face_recognition.face_encodings() : (Feature Extraction step)
  - Now that we have cropped the face out of the image, we extract features from it. Here we are going to use face embeddings to extract the features out of the face. A neural network takes an image of the person's face as input and outputs a vector which represents the most important features of a face. In machine learning, this vector is called embedding(encoding) and thus we call this vector as face embedding (encoding).

  - While training the neural network, the network learns to output similar vectors for faces that look similar. For example, if I have multiple images of faces within different timespan, of course, some of the features of my face might change but not up to much extent. So, in this case the vectors associated with the faces are similar or in short, they are very close in the vector space.

  - we're going to turn the bounding boxes of the person's face into a list of 128 numbers which represent the most important features of a face. This is known as encoding the face into a vector.
  - Then we can save the names and encodings into a file for future recall by using pickle.dumps() from pickle library.

- face_recognition.compare_faces():

    - first, we will load our pickled encodings and face names from disk. We'll need this data later during the actual face recognition step.

    - Now that we have face embeddings for every face in our data saved in pickled file, the next step is to recognize a new test image that is not in our data. So, the first step is to compute the face embedding(encoding) for the image using face_recognition.face_encodings() we used above and then compare this embedding(encoding)with the rest of the embeddings(encodings) we have. We recognize the face if the generated embedding is closer or similar to any other embedding.

2.12: Currency Recognition (Classification):

We made this feature to help blind or visually impaired people to classify what the type of the currency, they have. By capture one image to the single type of currency and our application will tell them what the category of the Egyptian Currency they have.

So, we used Azure Custom vision service to build this feature with custom data set.

2.12.1: What is Custom Vision:

Azure Custom Vision is an image recognition service that lets you build, deploy, and improve your own image identifier models. An image identifier applies labels to images, according to their detected visual characteristics. Each label represents a classification or object. Unlike the Computer Vision service, Custom Vision allows you to specify your own labels and train custom models to detect them.

2.12.2: What it does?

The Custom Vision service uses a machine learning algorithm to analyze images. You, the developer, submit groups of images that have and don't have the characteristics in question. You label the images yourself with your own custom labels (tags) at the time of submission. Then the algorithm trains to this data and calculates its own accuracy by testing itself on those same images. Once you've trained the algorithm, you can test, retrain, and eventually use it in your image recognition app to classify images. You can also export the model itself for offline use.

Classification and object detection:

Custom Vision functionality can be divided into two features. Image classification applies one or more labels to an entire image. Object

detection is similar, but it returns the coordinates in the image where the applied label(s) can be found.

The Custom Vision service is optimized to quickly recognize major differences between images, so we can start prototyping our model with a small amount of data. 50 images per label are generally a good start. However, the service is not optimal for detecting subtle differences in images (for example, detecting minor cracks or dents in quality assurance scenarios).

2.12.3: What it includes:

The Custom Vision Service is available as a set of native SDKs as well as through a web-based interface on the Custom Vision portal. we can create, test, and train a model through either interface or use both together.

So, we used build Custom Vision web portal to create an image classifier model. Once we build a model, you can test it with new images and eventually integrate it into your own image recognition app.

So, we follow next steps:

1- Create Image classifier model
2- Upload our data set and tagged them
3- Train our model
4- Evaluate the classifier model

2.11.4: Our Data set:

We have seven categories of Egyptian currency to Classify and each category we have about 1000 images:

One pound

Five pounds

Ten pounds

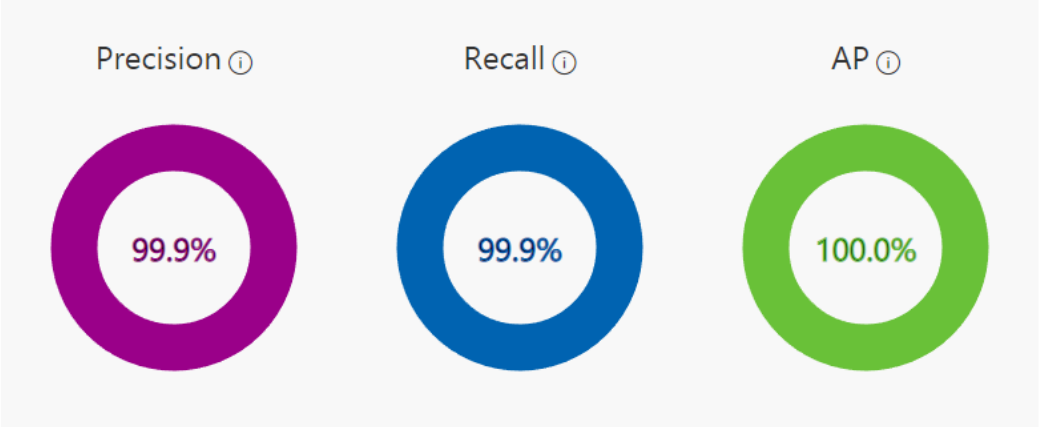
Twenty pounds

Fifty pounds

Hundred pounds

Two hundred pounds

We trained our classifier model about 8 hours. And we evaluate our model and this our result



Functions we used to integrate the Custom Vision model to our python code:

- classify_image_with_no_store()

used this function to classify and test new image

2.12.5: URI Parameters:

Name	In	Required	Type	Description
`Endpoint`	path	True	string	Supported Cognitive Services endpoints.
`projectId`	path	True	string uuid	The project id.
`publishedName`	path	True	string	Specifies the name of the model to evaluate against.
`application`	query		string	Optional. Specifies the name of application using the endpoint.

2.13: Heroku Server

- Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

- Heroku is fully managed, giving developers the freedom to focus on their core product without the distraction of maintaining servers, hardware, or infrastructure. The Heroku experience provides services, tools, workflows, and polyglot support—all designed to enhance developer productivity.

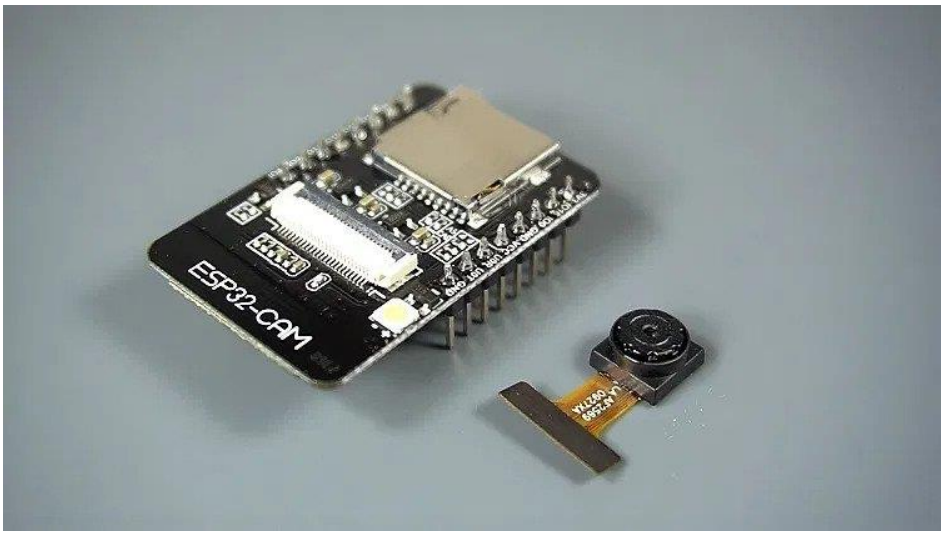
2.13.1: Heroku Languages

- fully managed container-based cloud platform, to make it easy for you to run apps written in a variety of programming languages, including our first-class languages Ruby, Java, PHP, Python, Node, Go, Scala and Clojure. Languages are at the heart of what we do — because the languages we support are at the heart of your applications.

2.14: Hardware

2.14.1: Introducing the ESP32-CAM

- The ESP32-CAM is a small size, low power consumption camera module with the ESP32-S chip that costs approximately $10 based on ESP32 WIFI development board. It comes with an OV2640 camera, and it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients.
- The ESP32-CAM suitable for home automation and smart devices, industrial wireless control, wireless video monitoring, QR wireless identification, Wi-Fi image upload, wireless face recognition and other IoT applications. It is an ideal solution for IoT applications.



Features

- Onboard ESP32-S module, supports WiFi + Bluetooth
- OV2640 camera with built-in flash lamp
- Onboard microSD card slot, supports up to 4G TF card for data storage
- Supports WiFi video monitoring and WiFi image upload
- Supports multi sleep modes, deep sleep current as low as 6mA
- Control interface is accessible via pin-header, easy to be integrated and embedded into user product

Specifications

- WIFI module: ESP-32S
- Processor: ESP32-D0WD
- Built-in Flash: 32Mbit
- RAM: Internal 512KB + External 4M PSRAM
- Antenna: Onboard PCB antenna
- WiFi protocol: IEEE 802.11 b/g/n/e/i
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE
- WIFI mode: Station / SoftAP / SoftAP+Station
- Security: WPA/WPA2/WPA2-Enterprise/WPS
- Output image format: JPEG (OV2640 support only), BMP, GRAYSCALE
- Supported TF card: up to 4G
- Peripheral interface: UART/SPI/I2C/PWM
- IO port: 9
- UART baudrate rate: default 115200bps
- Power supply: 5V
- Transmitting power:
- 802.11b: 17 ±2dBm(@11Mbps)
- 802.11g: 14 ±2dBm(@54Mbps)
- 802.11n: 13 ±2dBm(@HT20,MCS7)
- Receiving sensitivity:
- CCK,1Mbps: -90 dBm

- CCK,11Mbps: -85 dBm
- 6Mbps(1/2 BPSK): -88 dBm
- 54Mbps(3/4 64-QAM): -70 dBm
- HT20,MCS7(65Mbps, 72.2Mbps): -67 dBm
- Power consumption:
- Flash off: 180mA@5V
- Flash on and brightness max: 310mA@5V
- Deep-Sleep: as low as 6mA@5V
- Modern-Sleep: as low as 20mA@5V
- Light-Sleep: as low as 6.7mA@5V
- Operating temperature: -20 ℃ ~ 85 ℃
- Storage environment: -40 ℃ ~ 90 ℃, <90%RH
- Dimensions: 40.5mm x 27mm x 4.5mm

Applications

The ESP32-CAM suit for IOT applications such as:

1. Smart home devices image upload
2. Wireless monitoring
3. Intelligent agriculture
4. QR wireless identification
5. facial recognition

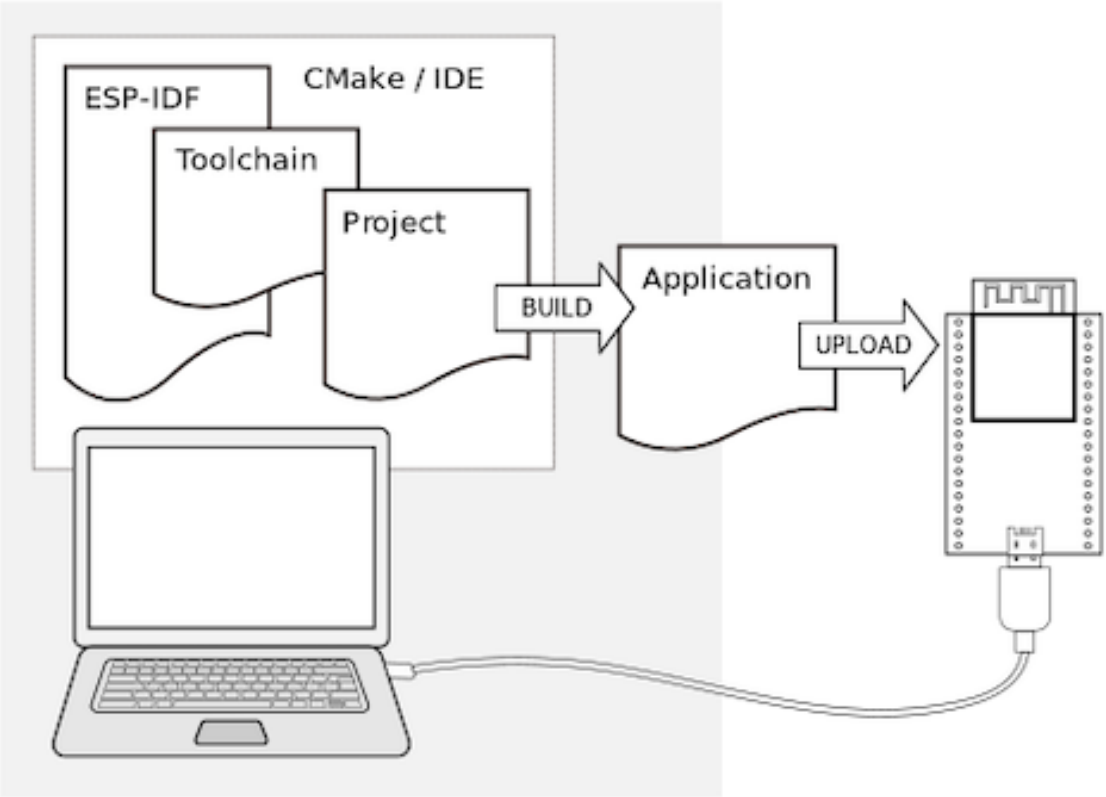
2.14.2: What we need

2.14.2.1: Hardware

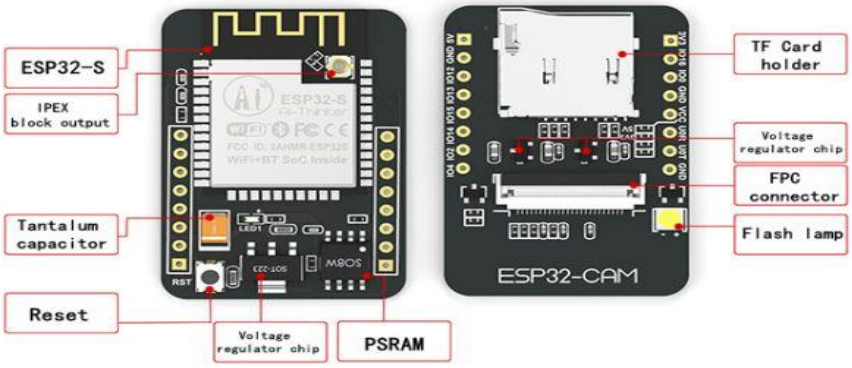
- An ESP32 board.
- USB cable - USB A / micro USB B.
- Computer running Windows, Linux, or macOS.

2.14.2.2: Software

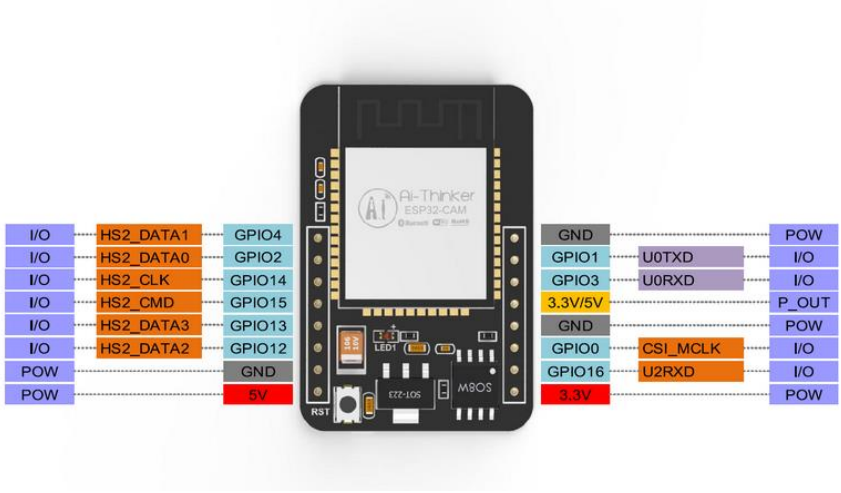
- Toolchain to compile code for ESP32
- Build tools - CMake and Ninja to build a full Application for ESP32
- ESP-IDF that essentially contains API (software libraries and source code) for ESP32 and scripts to operate the Toolchain



2.13.3: What is on Board



2.14.4: Pinouts



There are three GND pins and two pins for power: either 3.3V or 5V.
GPIO 1 and GPIO 3 are the serial pins. You need these pins to upload code to your board. Additionally, GPIO 0 also plays an important role, since it determines whether the ESP32 is in flashing mode or not. When GPIO 0 is connected to GND, the ESP32 is in flashing mode.
The following pins are internally connected to the microSD card reader:

- GPIO 14: CLK
- GPIO 15: CMD
- GPIO 2: Data 0
- GPIO 4: Data 1 (also connected to the on-board LED)
- GPIO 12: Data 2
- GPIO 13: Data 3

2.14.5: Sd card importance in our app:

- The photos taken by the chip will be saved to a memory so that we can access the data for the Alzheimer's people by checking the photos that are frequently taken by them
- So that we can use this data to help in providing the doctors with needed data to help them in providing new treatments for Alzheimer

Chapter 3: System Design

3.1: System features

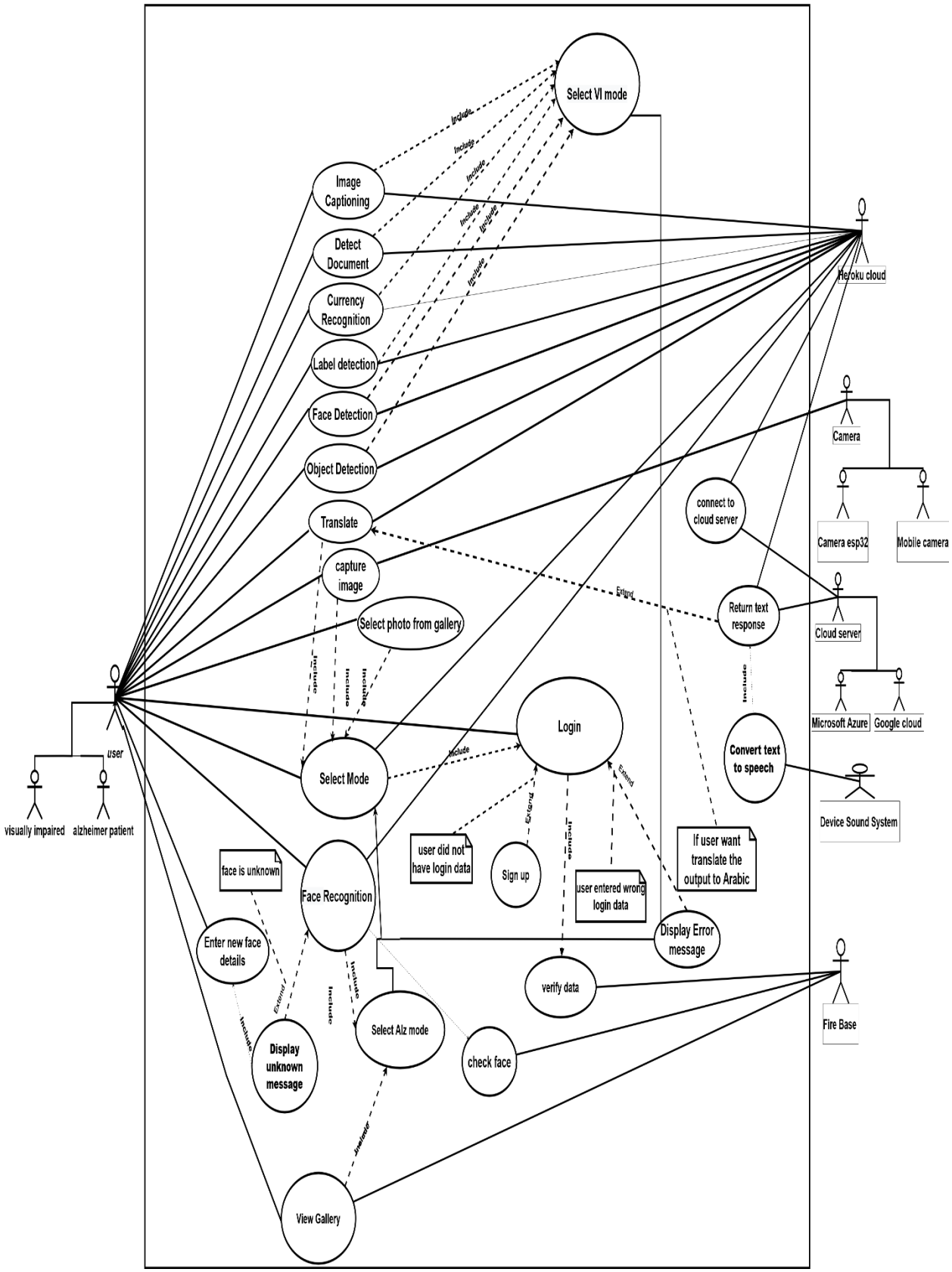
- Two main sections in our application Alzheimer's mode and visually impaired mode.
- Visually impaired mode: we use some AI technologies to help the visually impaired people in these sections you can used on of these features by just take a photo and forward it to the model:
    1. Text Reader (OCR)
    2. Face Detection
    3. Image Captioning
    4. Object Detection
    5. Label Detection
    6. Currency Detection
- Alzheimer mode: In this mode we use A face recognition model to make Alzheimer easily remember their beloved people.

3.2 Use case Diagram:

- use case diagram is the primary form of system/software requirements for a new software program underdeveloped.
- Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.
- A use case diagram is usually simple. It does not show the detail of the use cases:
- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.
- As said, a use case diagram should be simple and contains only a few shapes. If yours contain more than 20 use cases, you are probably misusing use case diagram.
- Purpose of Use Case Diagram
- Use case diagrams are typically developed in the early stage of development and people often apply use case modeling for the following purposes:

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases

Developed by analysts together with domain experts


Select VI mode
Image Captioning
Detect Document
Currency Recognition
Label detection
Face Detection
Object Detection
Translate
capture image
Select photo from gallery
Select Mode
Login
Face Recognition
Select Alz mode
View Gallery
Enter new face details
Display unknown message
face is unknown
check face
verify data
user did not have login data
Sign up
user entered wrong login data
Display Error message
If user want translate the output to Arabic
Return text response
Convert text to speech
connect to cloud server
Heroku cloud
Camera
Camera esp32
Mobile camera
Cloud server
Microsoft Azure
Google cloud
Device Sound System
Fire Base
user
visually impaired
alzheimer patient
include
extend

3.3: Use case description:

- A scenario describes a sequence of interactions between the system and some actors.
- In Each use case there is a set of possible scenarios. Where the main scenario is the successful scenario where nothing goes wrong, and the use case is achieved.
- For each use case there is a set of possible scenarios. A scenario is an instance of a use case. A scenario describes a sequence of interactions between the system and some actors. Here are two examples of scenarios.
- A member of a lending library wishes to borrow a book and is allowed to do that if they have no outstanding loans.
- Another member wishes to borrow a book but has exceeded the quota for the number of books that can be borrowed.
- In each scenario the member wishes to borrow a book, but both the circumstances and outcomes of events are different in each instance.
- A use case includes a complex set of requirements that the system must meet to cope with every eventuality.
- The main success scenario shows the steps normally followed to achieve the stated goal of the use case. But there can be other scenarios for the same use case, each one having different outcomes depending upon circumstances.

3.2.1: Login use case

Use case name	Login
Use case ID	UC1
Actor	User (visually impaired user or user who suffers from Alzheimer's)
Pre-condition	The user has login data stored in the system.
Post-condition	The user can access the system.
Main scenario	1- Open the application. 2- Fill in email and password. 3- Click on the login button. 4- The filled data will be sent to the firebase server to get verified. 5- The firebase server will respond back for the verified data. 6- The user logged in to the system successfully.
Alternative scenario	- if the email or password are not verified in the firebase server: The system will display an error message, and the user will be prompted to re-enter login data. - If the filled data aren't stored in the system: The user must sign up to the system first and then log in.

3.2.2: Select mode use case

Use case name	Select mode
Use case ID	UC2
Actor	User (visually impaired user or user who suffers from Alzheimer's)
Pre-condition	The user is already logged in to the system.
Post-condition	The user became able to access all the functionalities of the system.
Main scenario	1- The user is logged in to the system. 2- He can choose between two modes: "Visually impaired" or "Alzheimer's". 3- The user can access all functions provided in the mode he has chosen.
Alternative scenario	N/a

3.2.3: Capture image use case

Use case name	Capture image
Use case ID	UC3
Actor	User (visually impaired user or user who suffers from Alzheimer's) and camera
Pre-condition	- The user is already logged in to the system. - The user has already selected the mode he wants.
Post-condition	N/a
Main scenario	1- The user is logged in to the system and chose the mode he wants. 2- The user can use any function he wants that's provided in the mode he has chosen. 3- The user will get access to the camera to capture an image from it.
Alternative scenario	N/a

3.2.4: Caption image use case

Use case name	Caption image
Use case ID	UC4
Actor	Visually impaired user and HEROKU server
Pre-condition	- The user is already logged in to the system. - The user has already selected the visually impaired mode. - The user has already captured an image.
Post-condition	The user will get a description of the image he has captured.
Main scenario	1- The user is logged in to the system and chose the visually impaired mode. 2- The user selected the image captioning function. 3- The user captured an image. 4- The image has been sent to the HEROKU server to send it to the server that supports this function. 5- In this case, the image will be sent to the Azure cognitive services server. 6- Azure cognitive services will do the process to the image and return the text response to the HEROKU server. 7- HEROKU server will send the response back to the application to display it to the user as a text and voice response. 8- The user will get the response as a text and voice describing the image.
Alternative scenario	- If the user selected to translate the output he will get the response in Arabic as a text and voice describing the image.

3.2.5: Detect document use case

Use case name	Detect document
Use case ID	UC5
Actor	Visually impaired user and HEROKU server.
Pre-condition	- The user is already logged in to the system. - The user has already selected the visually impaired mode. - The user has already captured an image.
Post-condition	The user will be able to know the text in the document image he captured.
Main scenario	1- The user is logged in to the system and chose the visually impaired mode. 2- The user selected detect document function. 3- The user captured an image. 4- The image has been sent to the HEROKU server to send it to the server that supports this function. 5- In this case, the image will be sent to the Google cloud platform server. 6- Google cloud platform will do the process to the image and return the text response to the HEROKU server. 7- HEROKU server will send the response back to the application to display it to the user as a text and voice response. 8- The user will get the response as a text and voice saying the text in the image he captured.
Alternative scenario	- If the user selected to translate the output he will get the response in Arabic as a text and the voice of the text detected in the image.

3.2.6: Recognize currency use case

Use case name	Recognize currency
Use case ID	UC6
Actor	Visually impaired user and HEROKU server
Pre-condition	- The user is already logged in to the system. - The user has already selected the visually impaired mode. - The user has already captured an image.
Post-condition	The user will get the banknote category of the captured currency image.
Main scenario	1- The user is logged in to the system and chose the visually impaired mode. 2- The user selected the currency recognition function. 3- The user captured an image. 4- The image has been sent to the HEROKU server to send it to the server that supports this function. 5- In this case, the image will be sent to the Azure cognitive services server. 6- Azure cognitive services will do the process to the image and return the text response to the HEROKU server. 7- HEROKU server will send the response back to the application to display it to the user as a text and voice response. 8- The user will get the response as a text and voice of the category of the banknote in the captured image.
Alternative scenario	- If the user selected to translate the output he will get the response in Arabic as a text and the voice of the category of the banknote in the captured image.

3.2.7: Label detection use case

Use case name	Label detection
Use case ID	UC7
Actor	Visually impaired user and HEROKU server.
Pre-condition	- The user is already logged in to the system. - The user has already selected the visually impaired mode. - The user has already captured an image.
Post-condition	The user will be able to know the labels of the objects detected in the captured image.
Main scenario	1- The user is logged in to the system and chose the visually impaired mode. 2- The user selected detect document function. 3- The user captured an image. 4- The image has been sent to the HEROKU server to send it to the server that supports this function. 5- In this case, the image will be sent to the Google cloud platform server. 6- Google cloud platform will do the process to the image and return the text response to the HEROKU server. 7- HEROKU server will send the response back to the application to display it to the user as a text and voice response. 8- The user will get the response as a text and the voice of the detected objects in the captured image as labels.
Alternative scenario	- If the user selected to translate the output, he will get the response in Arabic as a text and voice the detected objects in the captured image as labels.

3.2.8: Detect faces use case

Use case name	Detect faces
Use case ID	UC8
Actor	Visually impaired user and HEROKU server
Pre-condition	- The user is already logged in to the system. - The user has already selected the visually impaired mode. - The user has already captured an image.
Post-condition	The user will be able to detect human faces and know the number of people in the image, their gender, their emotions, and their estimated age of them.
Main scenario	1- The user is logged in to the system and chose the visually impaired mode. 2- The user selected the currency recognition function. 3- The user captured an image. 4- The image has been sent to the HEROKU server to send it to the server that supports this function. 5- In this case, the image will be sent to the Azure cognitive services server. 6- Azure cognitive services will do the process to the image and return the text response to the HEROKU server. 7- HEROKU server will send the response back to the application to display it to the user as a text and voice response. 8- The user will get the response as a text and voice of the number of humans in the image, their gender, their emotions, and their estimated age for them.
Alternative scenario	- If the user selected to translate the output he will get the response in Arabic as a text and voice of the category of the number of humans in the image, their gender, their emotions, and the estimated age for them.

3.2.9: Detect object use case

Use case name	Detect object
Use case ID	UC9
Actor	Visually impaired user and HEROKU server
Pre-condition	- The user is already logged in to the system. - The user has already selected the visually impaired mode. - The user has already captured an image.
Post-condition	The user will be able to know what objects in the captured image are
Main scenario	1- The user is logged in to the system and chose the visually impaired mode. 2- The user selected the detect object function. 3- The user captured an image. 4- The image has been sent to the HEROKU server to send it to the server that supports this function. 5- In this case, the image will be sent to the Azure cognitive services server. 6- Azure cognitive services will do the process to the image and return the text response to the HEROKU server. 7- HEROKU server will send the response back to the application to display it to the user as a text and voice response. 8- The user will get the response as a text and voice of the detected objects in the captured image.
Alternative scenario	- If the user selected to translate the output he will get the response in Arabic as a text and voice of the detected objects in the captured image.

3.2.10: Face recognition use case

Use case name	Face recognition
Use case ID	UC10
Actor	The user who suffers from Alzheimer's and HEROKU server
Pre-condition	- The user is already logged in to the system. - The user has already selected Alzheimer's mode. - The user has already captured an image.
Post-condition	The user will be able to who the people in front of him are and what is the relation between them and the user.
Main scenario	1- The user is logged in to the system and chose Alzheimer's mode. 2- The user selected the face recognition function. 3- The user captured an image. 4- The image will be sent to the firebase server to check if it's saved in our system or not and if it is saved the data will be sent to the function- the function is a library in python we have uploaded it to the HEROKU server-. 5- HEROKU server will send the response back to the application to display it as a text to the user. 6- The user will get the response as a text consisting of the detected human name and the relation between him and the user.
Alternative scenario	- If the detected face is not stored in the system then the user will insert a new face by adding his information and the function will do the training on the image of the detected face and it will be stored in the firebase server.

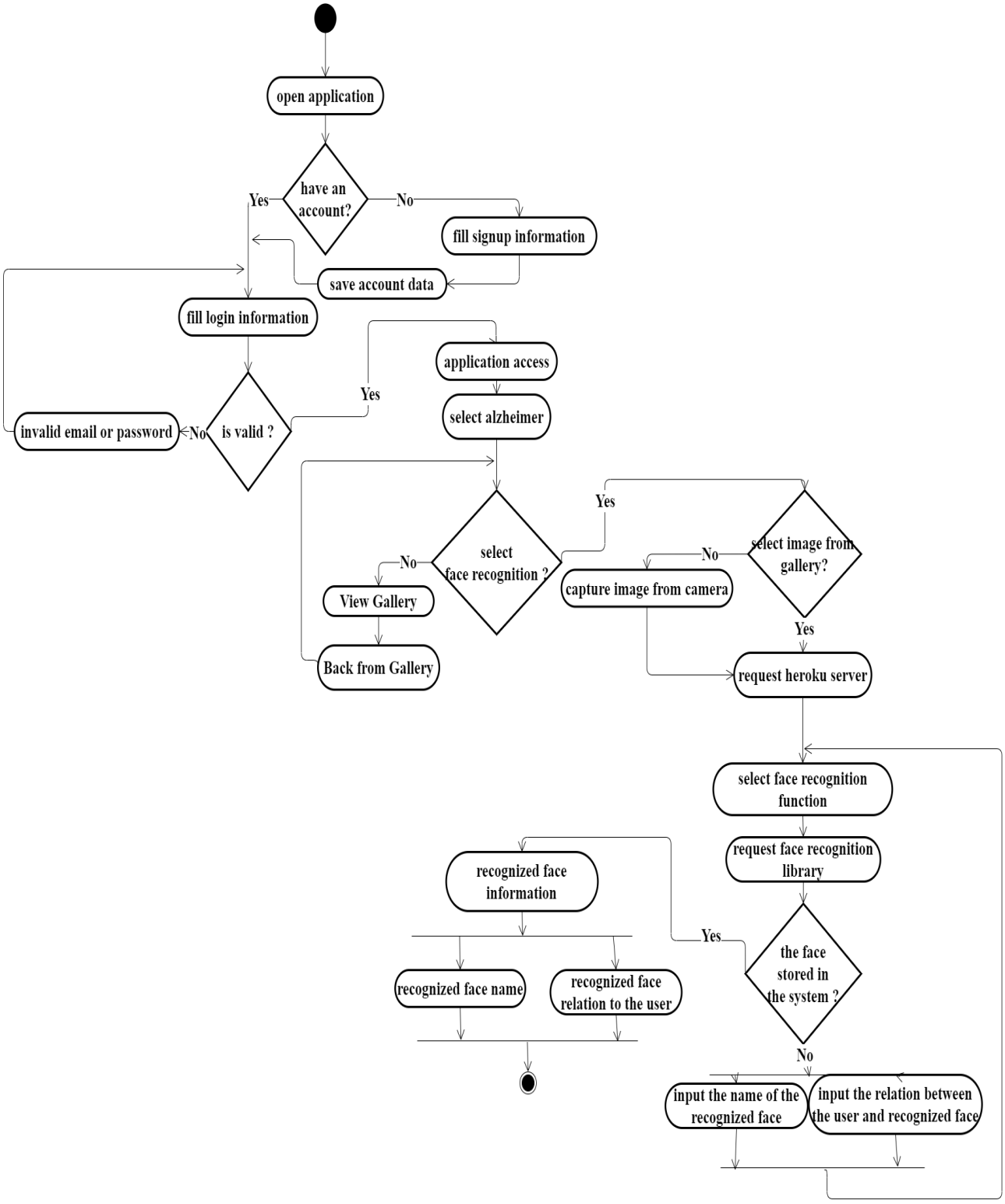
3.2.11: Return text response use case

Use case name	Return text response
Use case ID	UC11
Actor	HEROKU server and cloud server (Azure cognitive services or Google cloud platform)
Pre-condition	N/a
Post-condition	The server will return the response back to the application as a text.
Main scenario	1- The captured image will be processed in the cloud server. 2- The cloud server will send the output to the HEROKU server as a text response. 3- HEROKU server will send the response back to the application to display it as a text to the user.
Alternative scenario	N/a

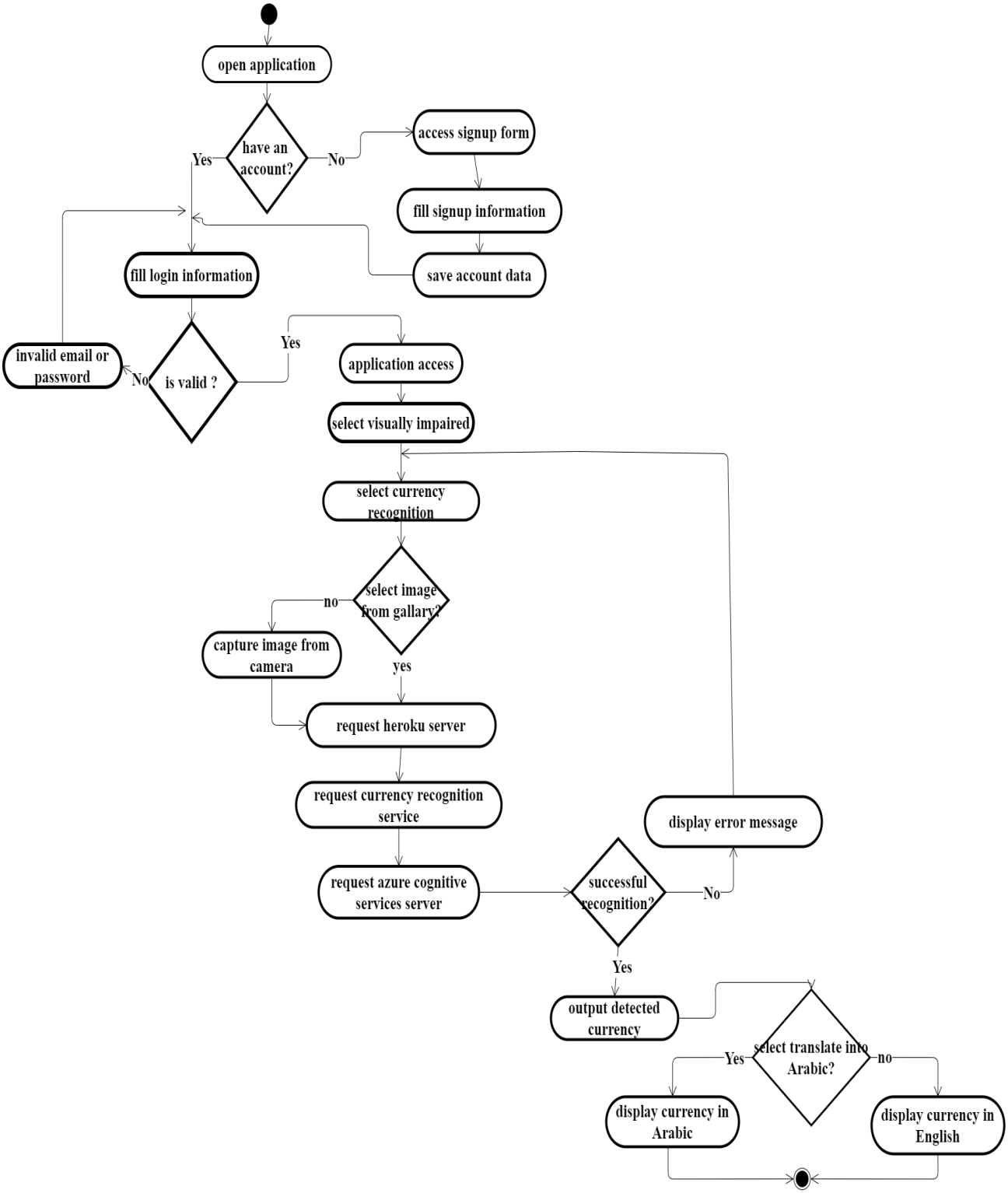
3.3: Activity Diagrams:

- Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.
- Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc
- The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.
- Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.
- It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.
- The purpose of an activity diagram can be described as
    1. Draw the activity flow of a system.
    2. Describe the sequence from one activity to another.
    3. Describe the parallel, branched and concurrent flow of the system.

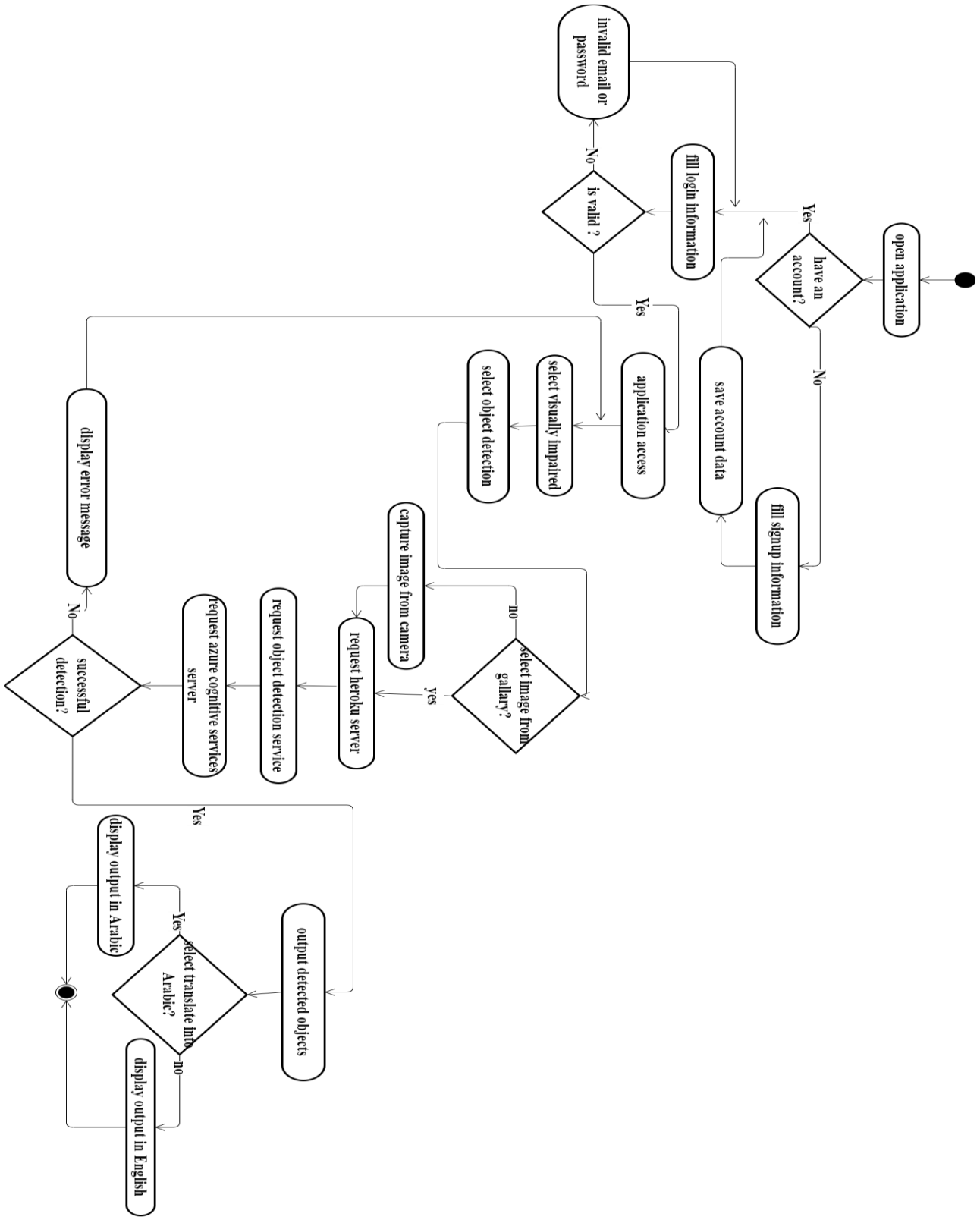
3.3.1: Face Recognition



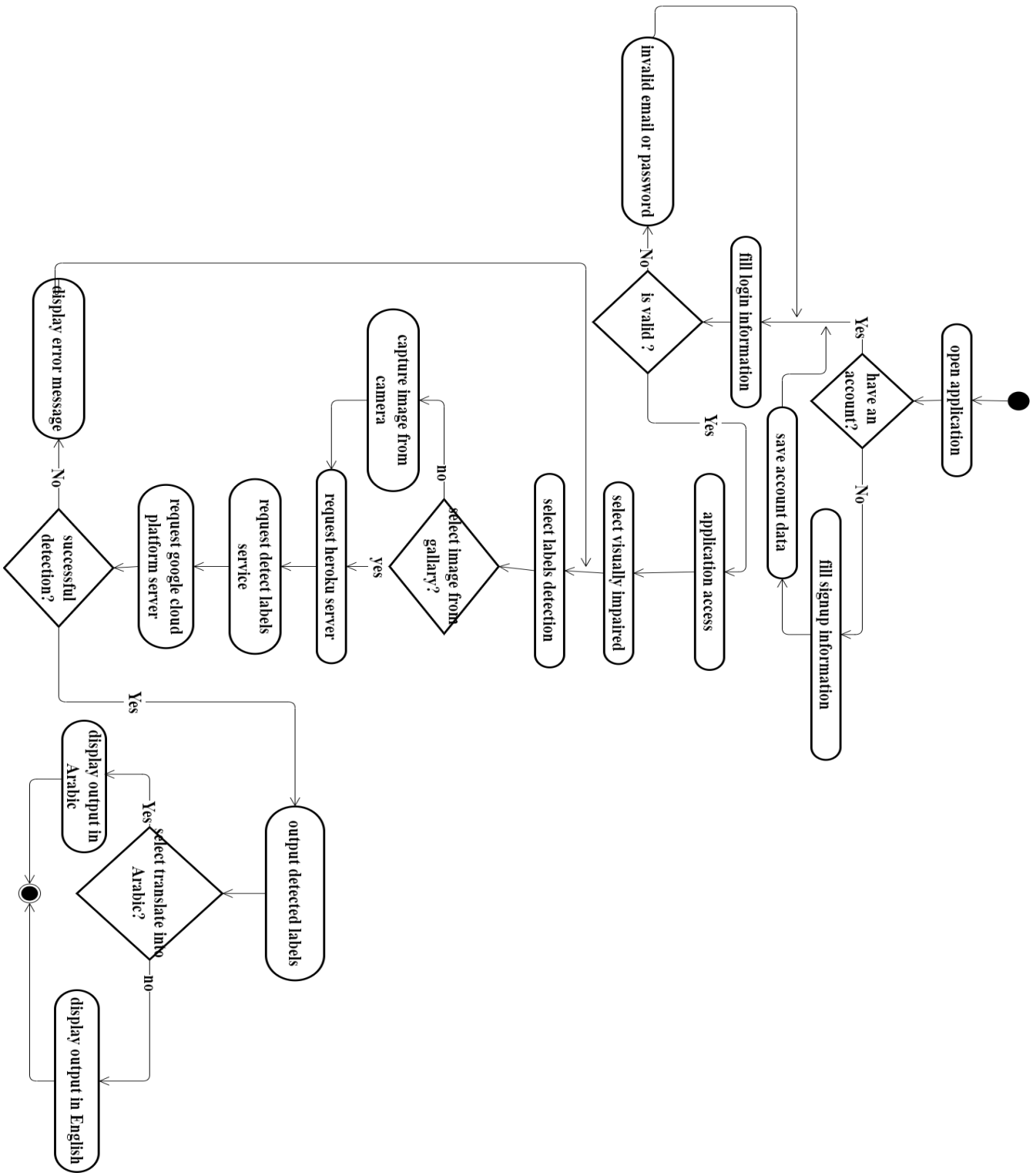
3.3.2: Currency Recognition



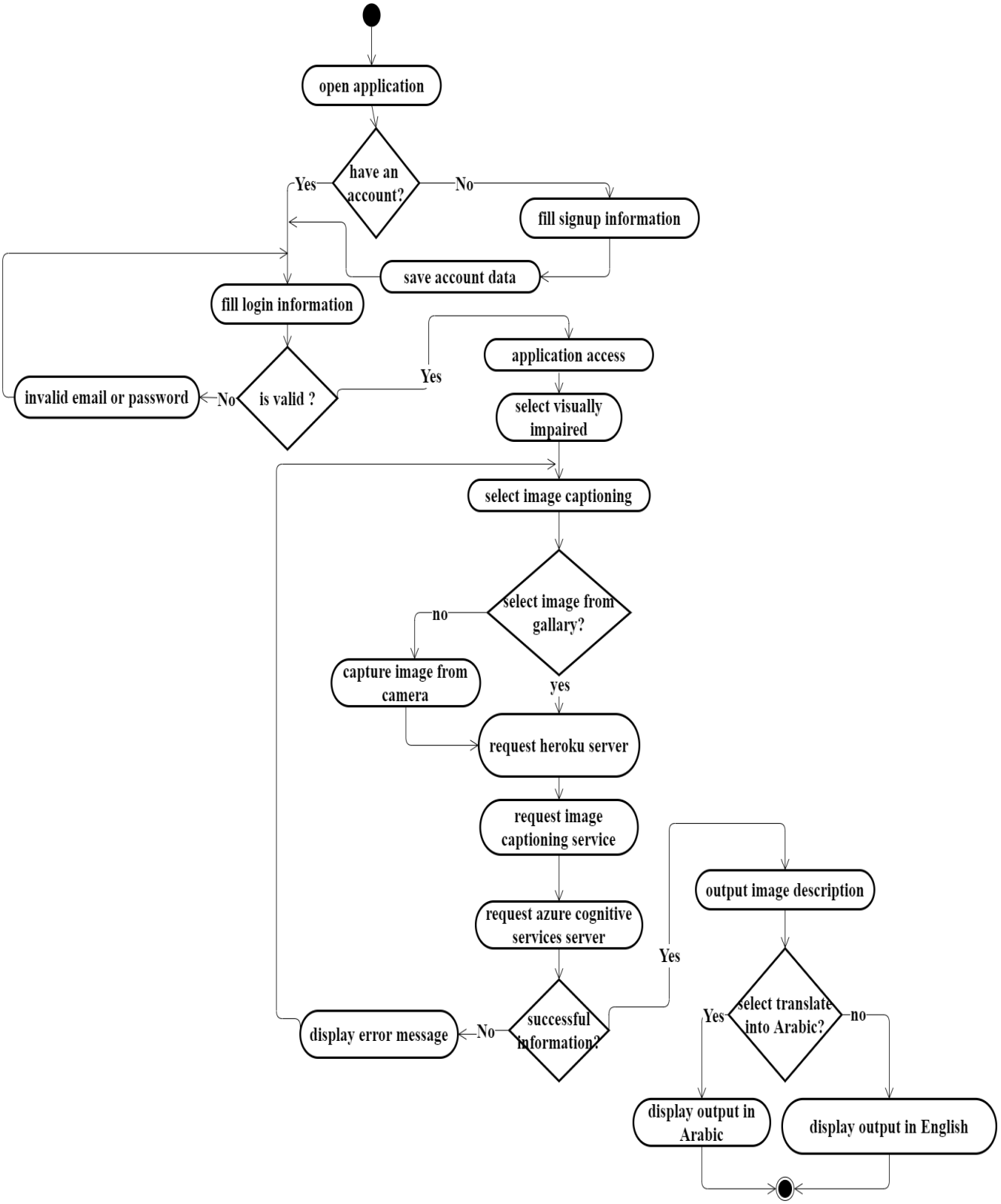
3.3.3: Object Detection



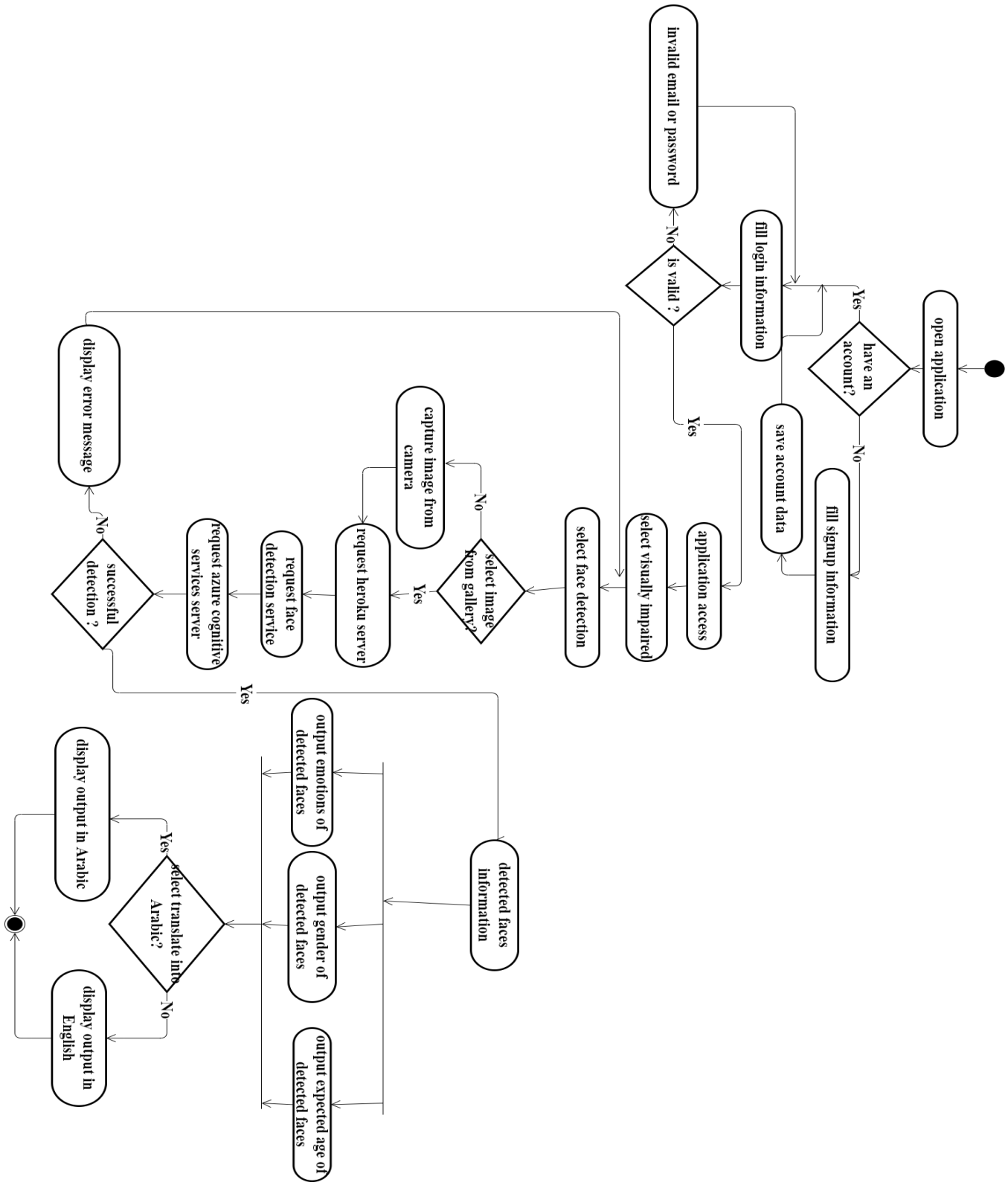
3.3.4: Label Detection



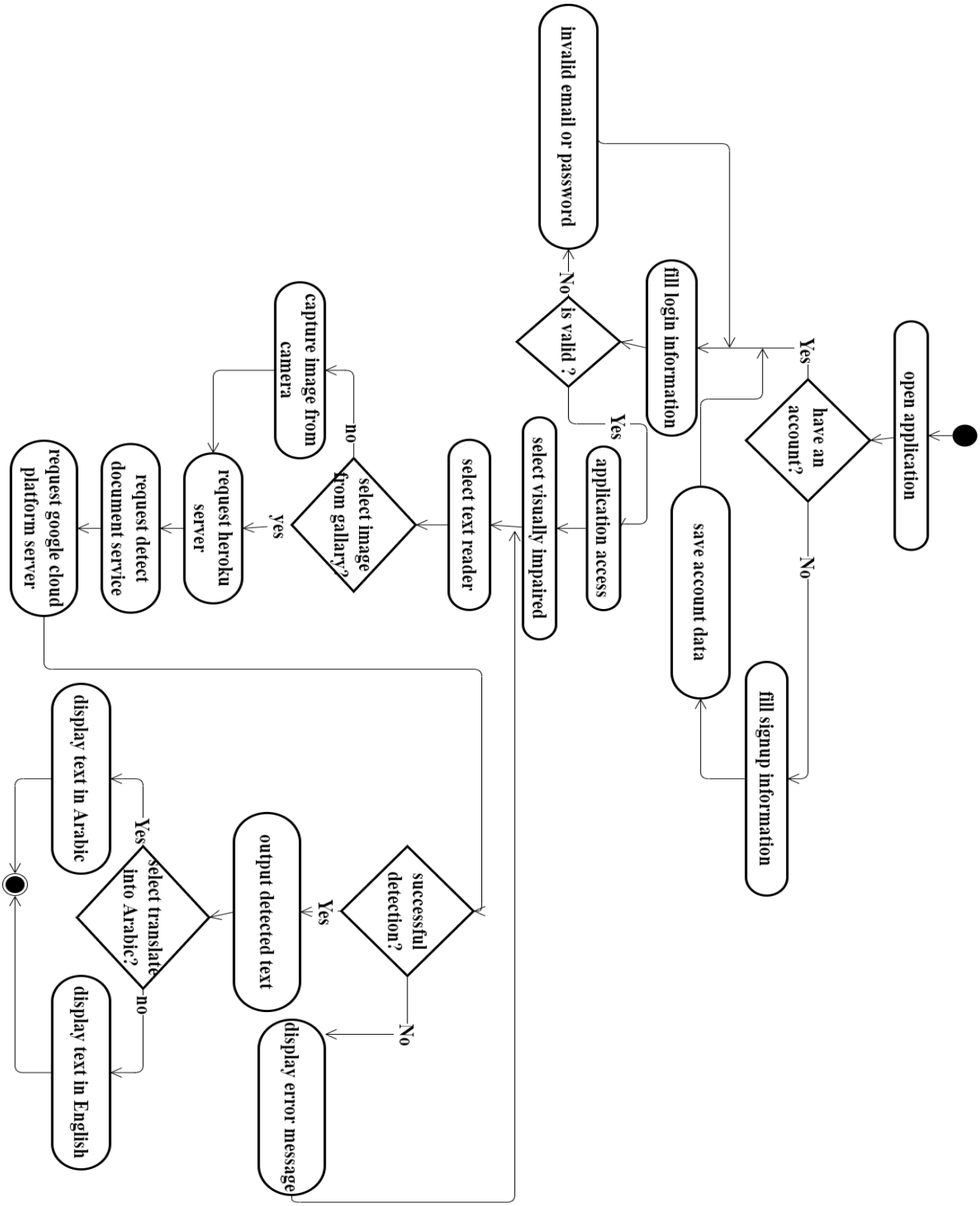
3.3.5: Image Captioning



3.3.6: Face Detection



3.3.7: Document Detection

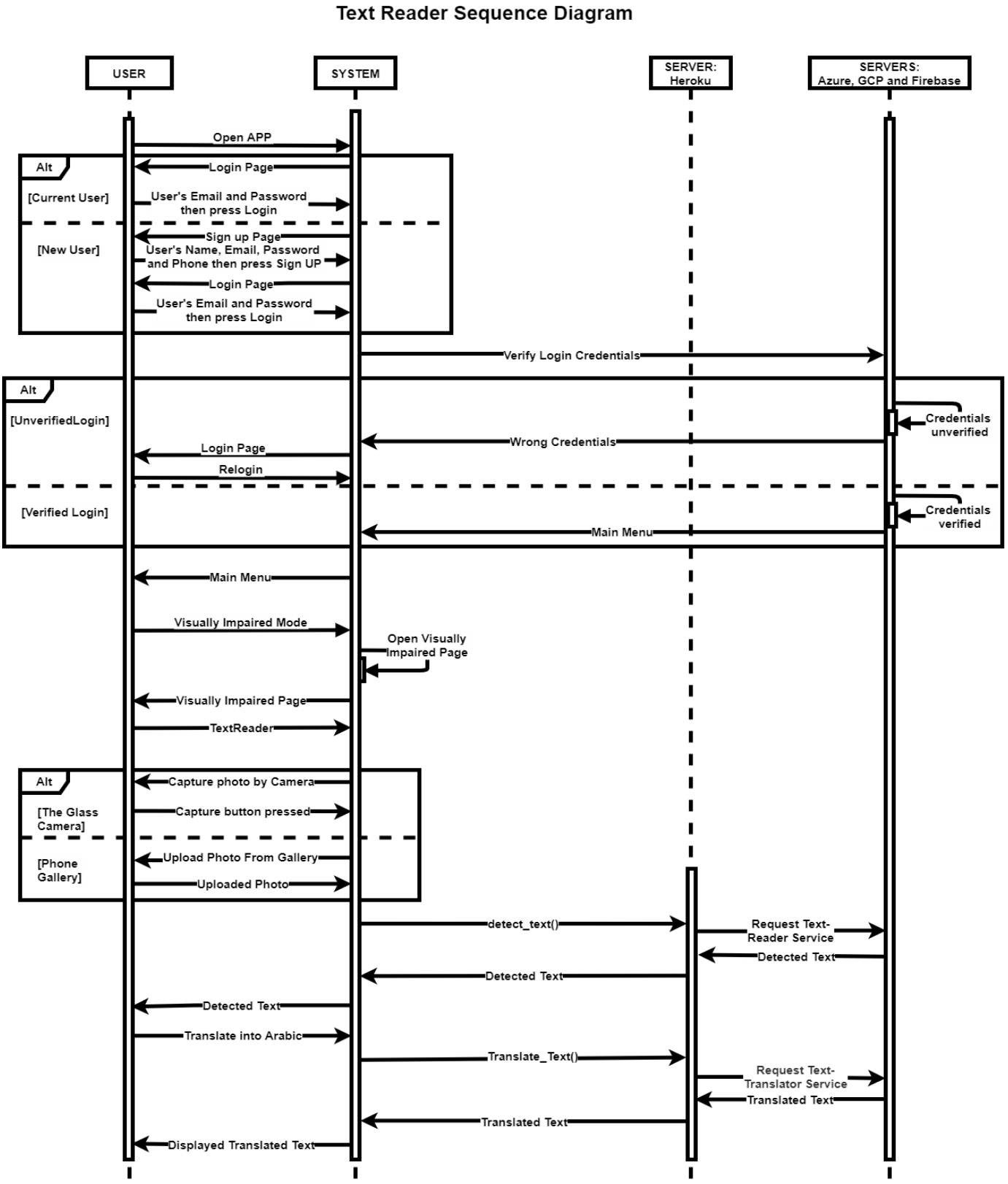


3.4: Sequence diagram(s)

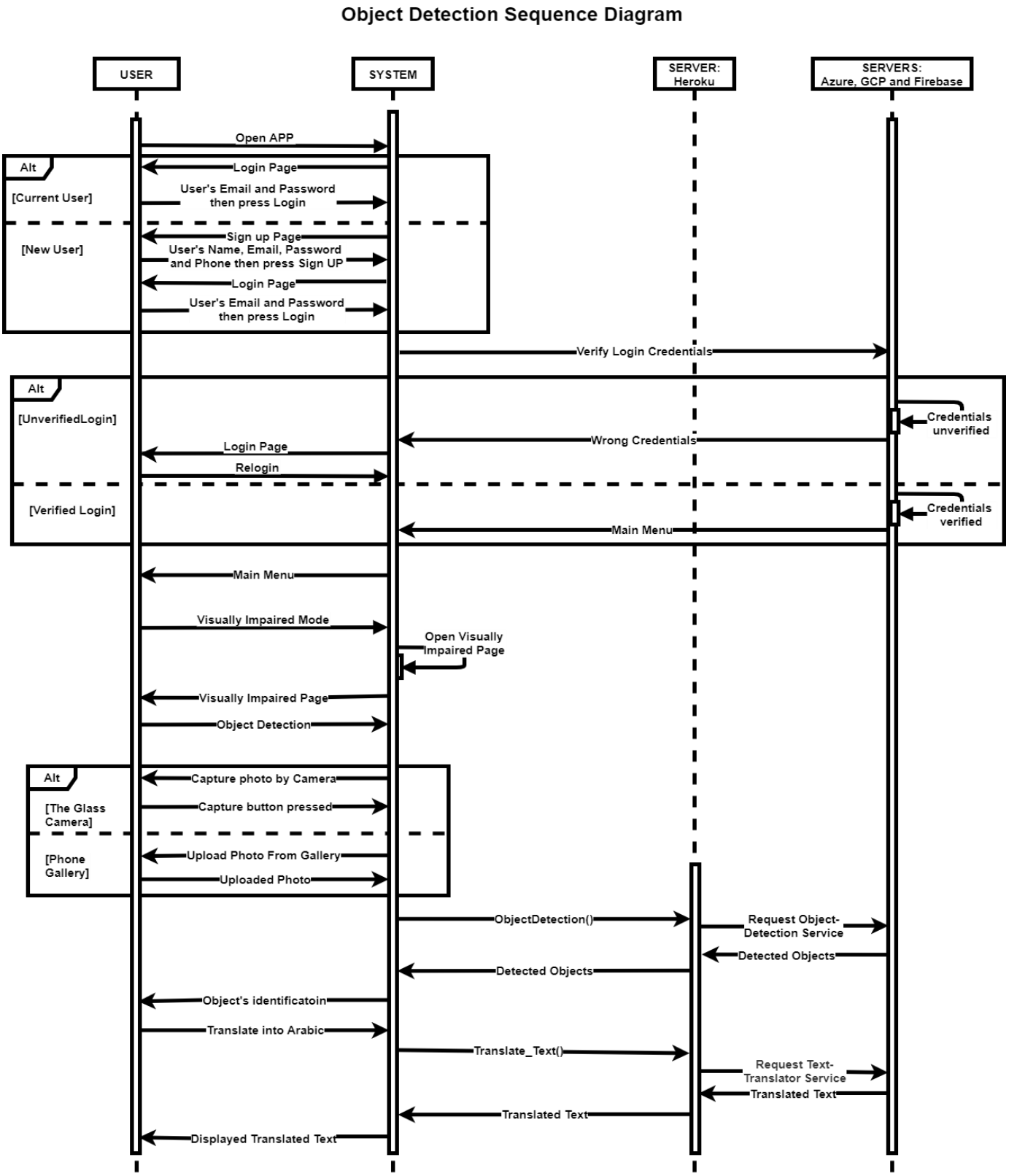
- To understand what a sequence diagram is, it's important to know the role of the Unified Modeling Language, better known as UML. UML is a modeling toolkit that guides the creation and notation of many types of diagrams, including behavior diagrams, interaction diagrams, and structure diagrams.
- A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.
- Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

    1. Represent the details of a UML use case.
    2. Model the logic of a sophisticated procedure, function, or operation.
    3. See how objects and components interact with each other to complete a process.
    4. Plan and understand the detailed functionality of an existing or future scenario.

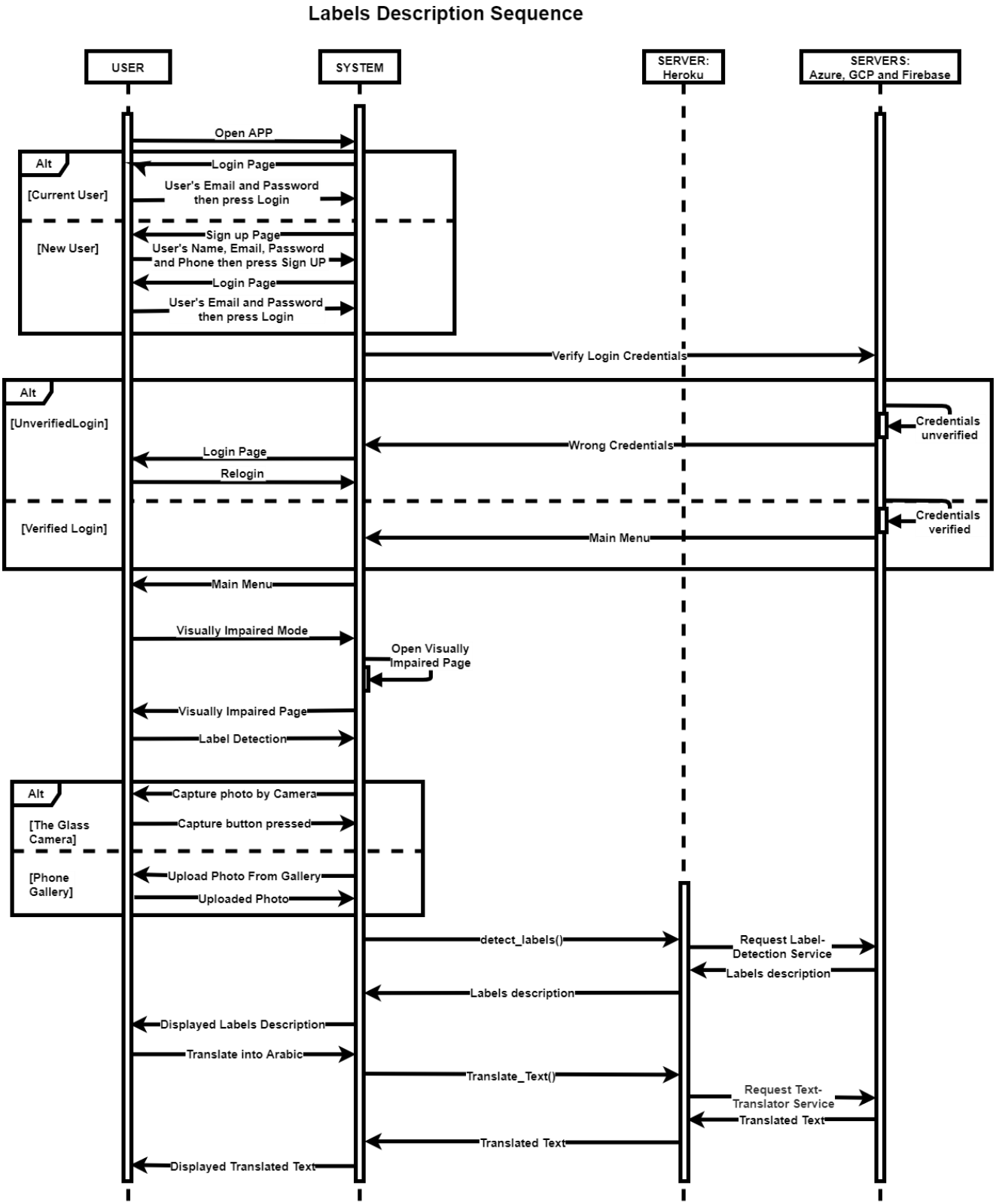
3.4.1: Text Reader



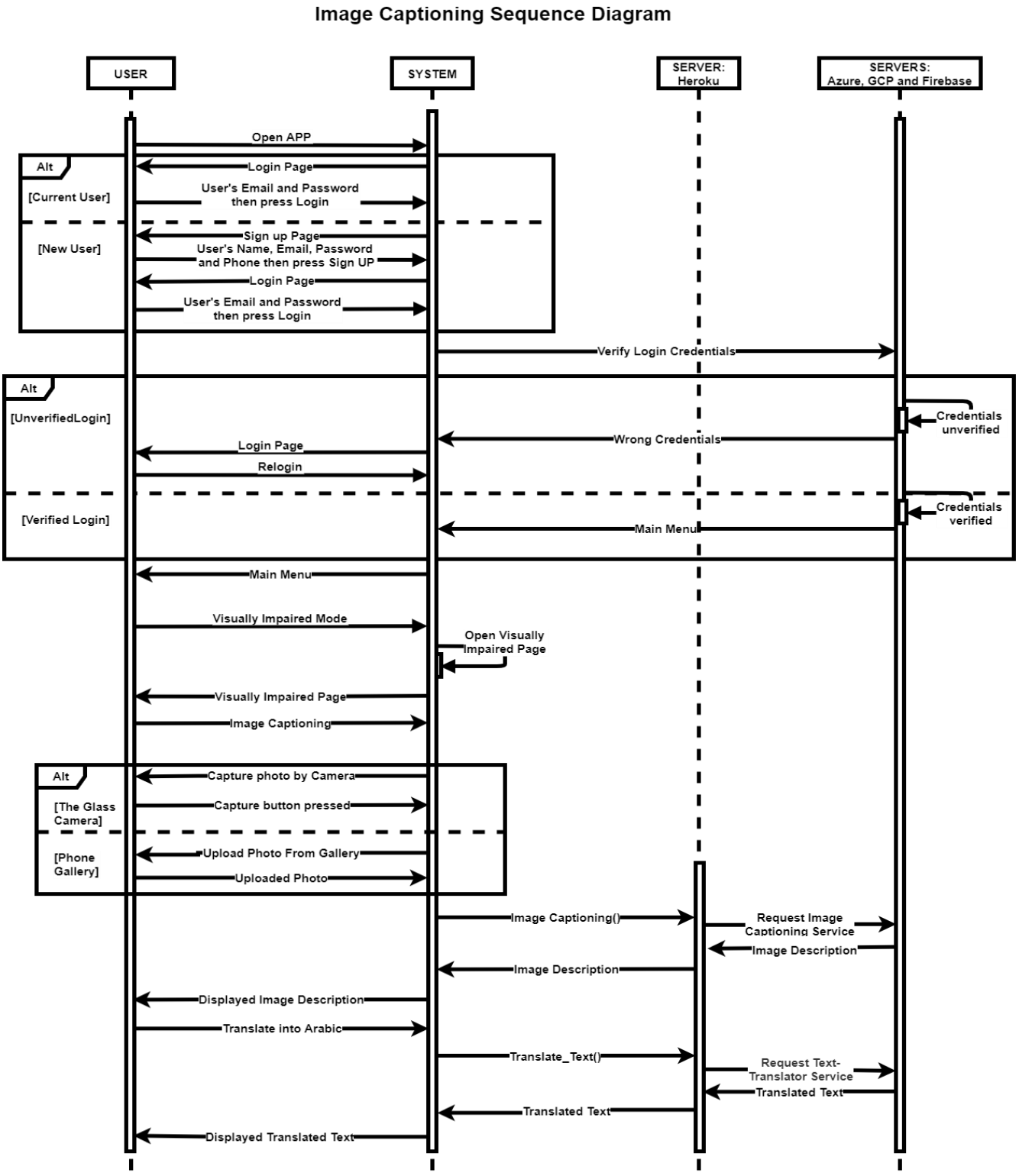
3.4.2: Object Detection



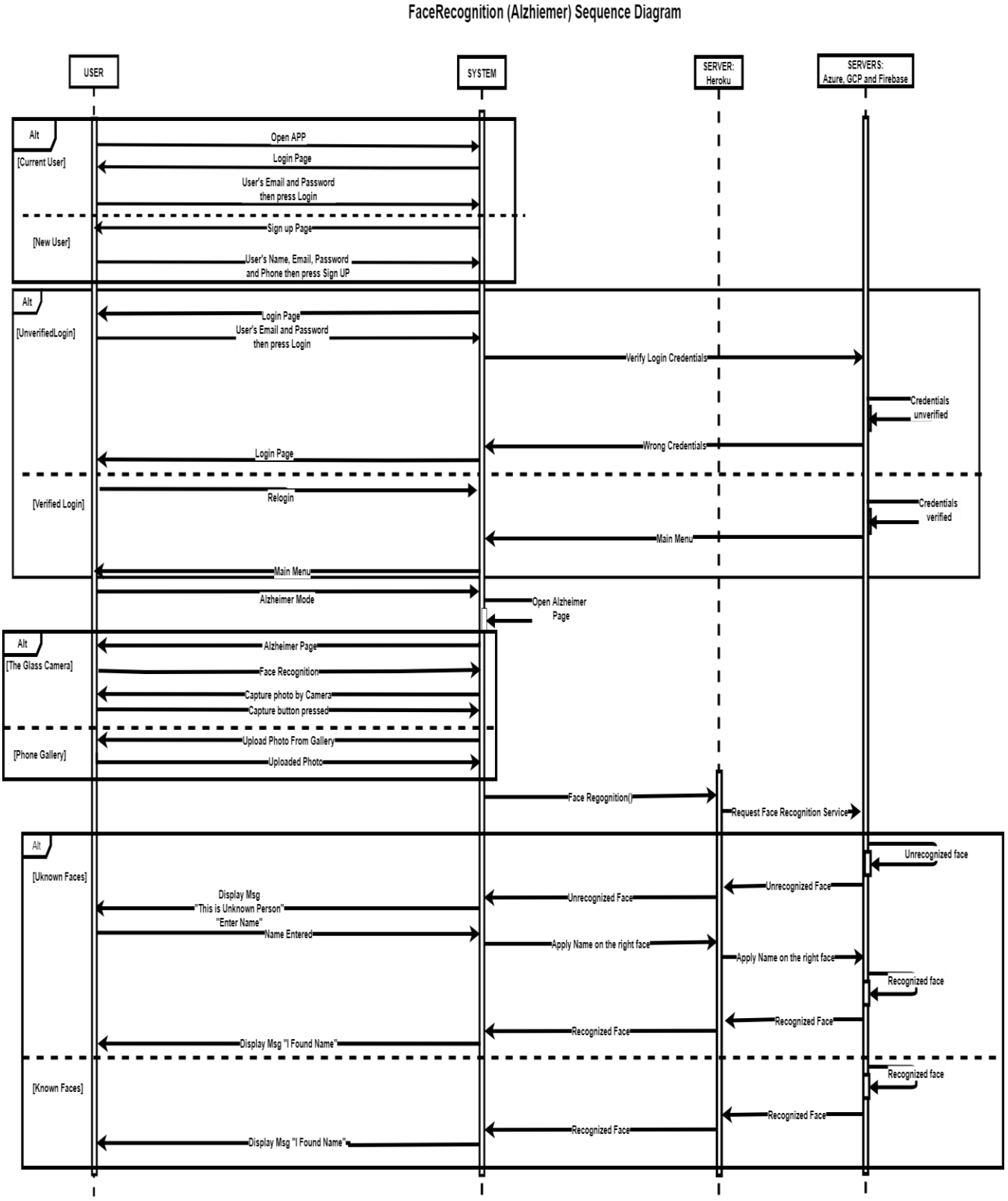
3.4.3: Labels Detection



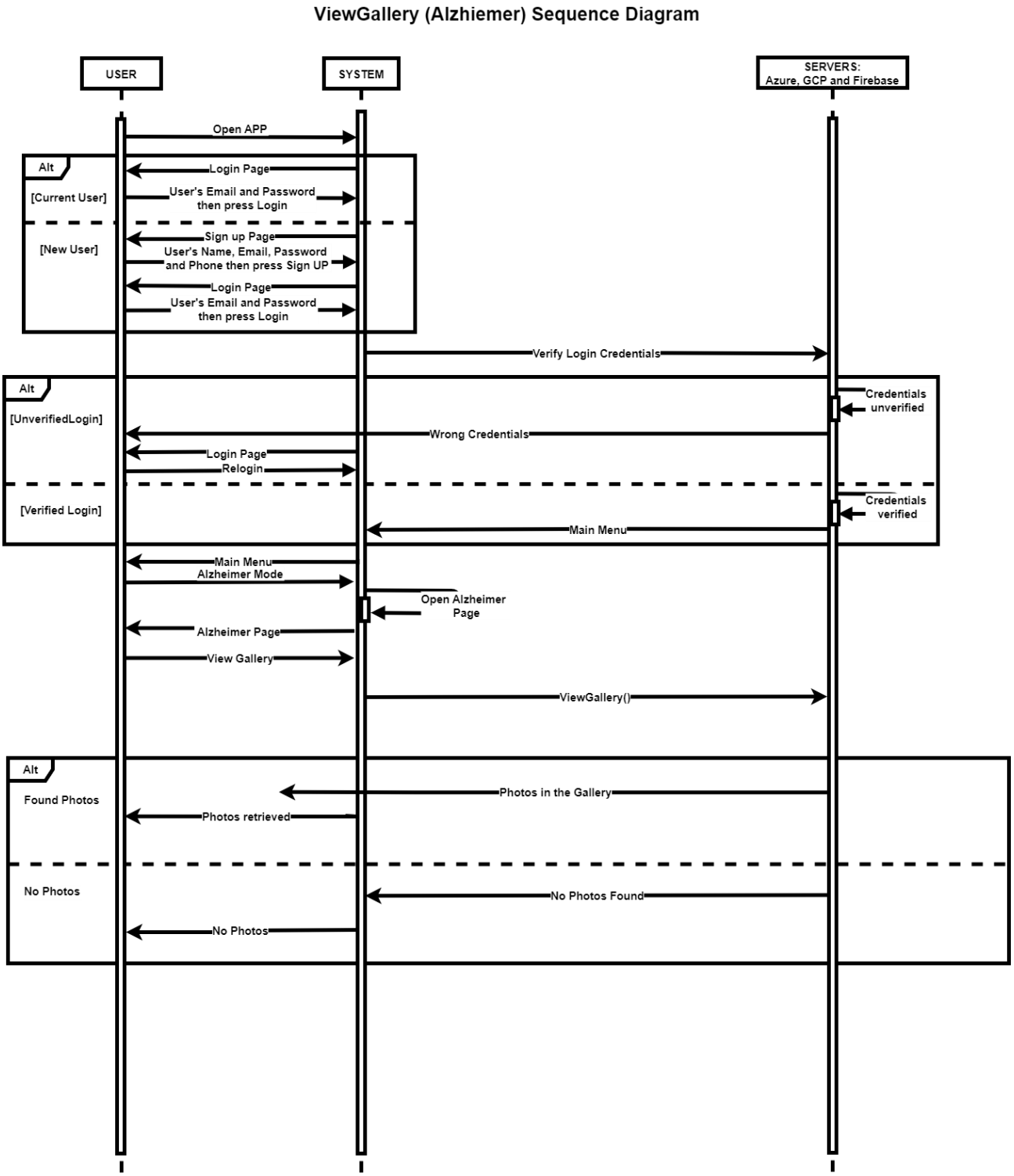
3.4.4: Image Captioning



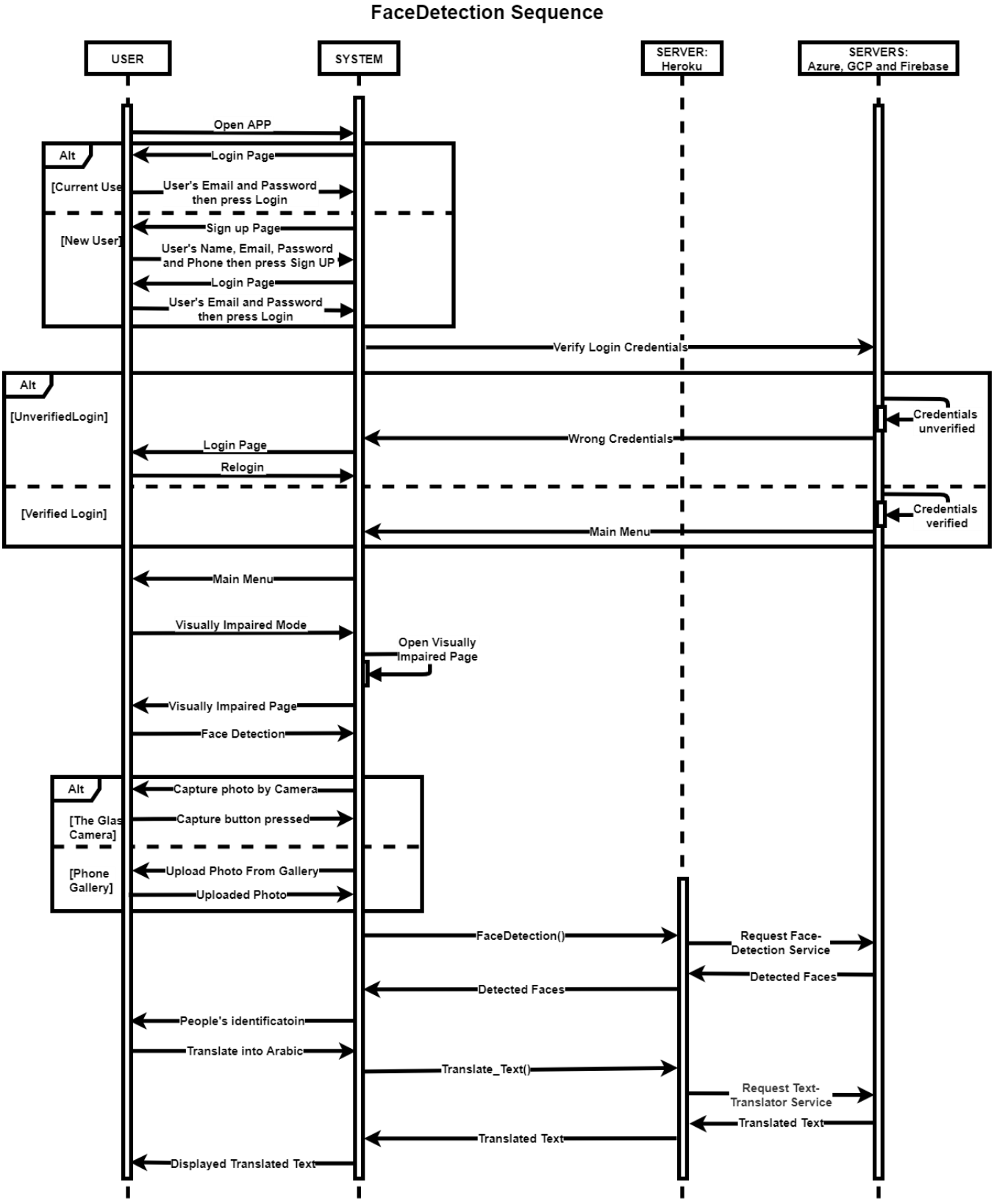
3.4.5: Face Recognition



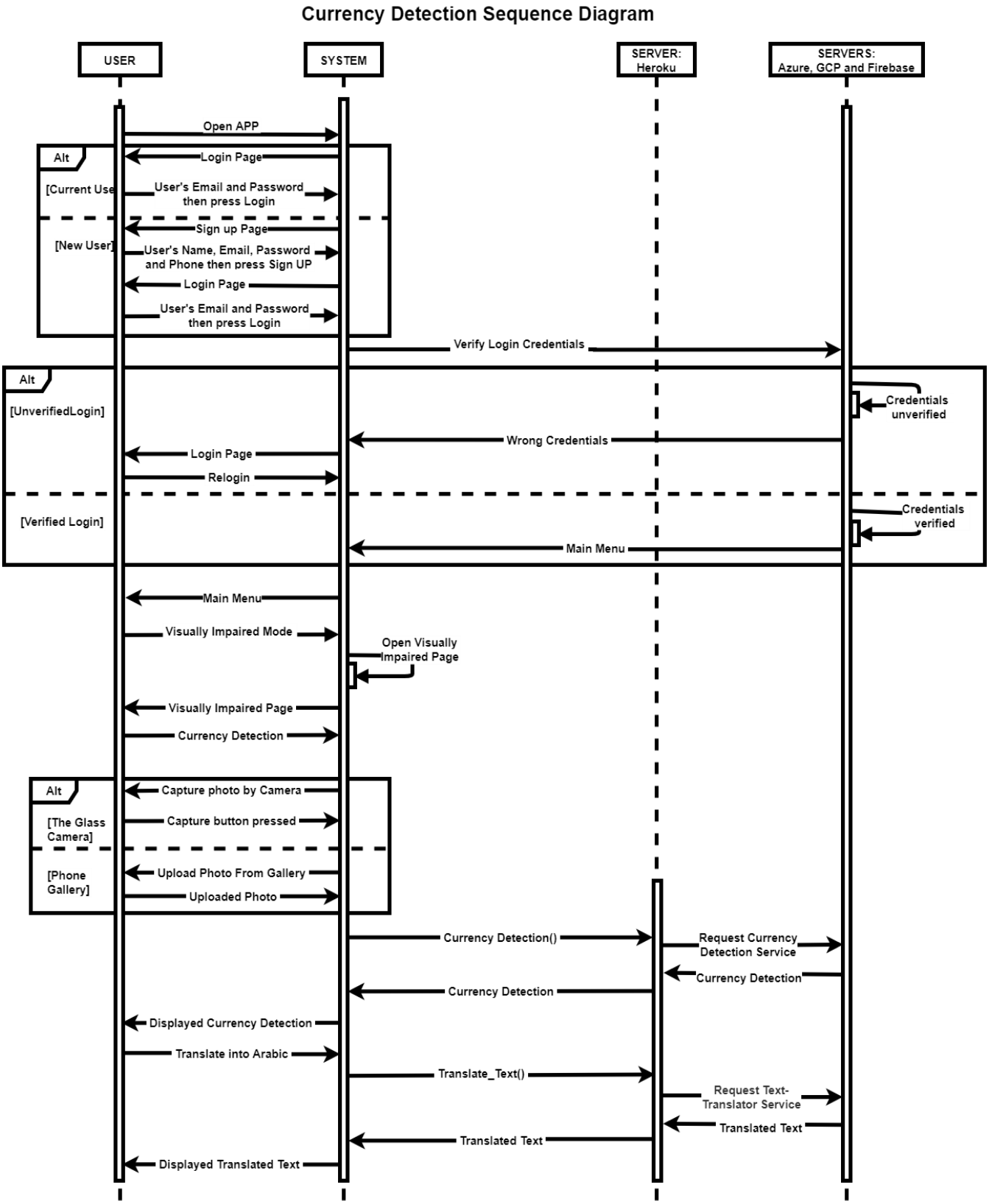
3.4.6: View Gallery:



3.4.7: Face Detection



3.4.8: Currency Detection

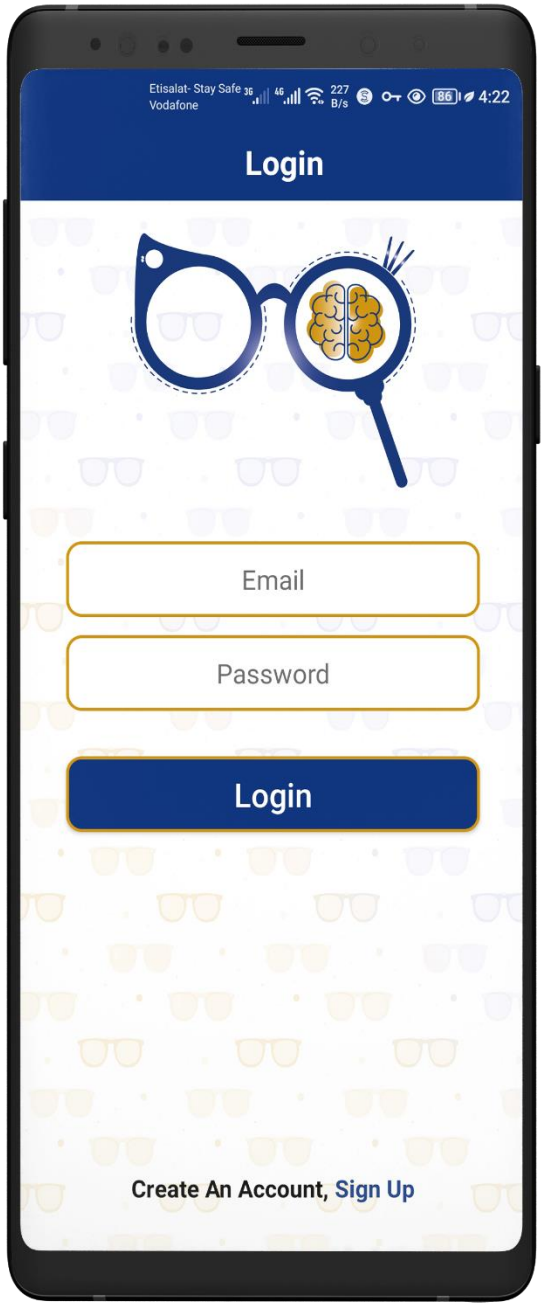


Chapter 4: Implementation and Results

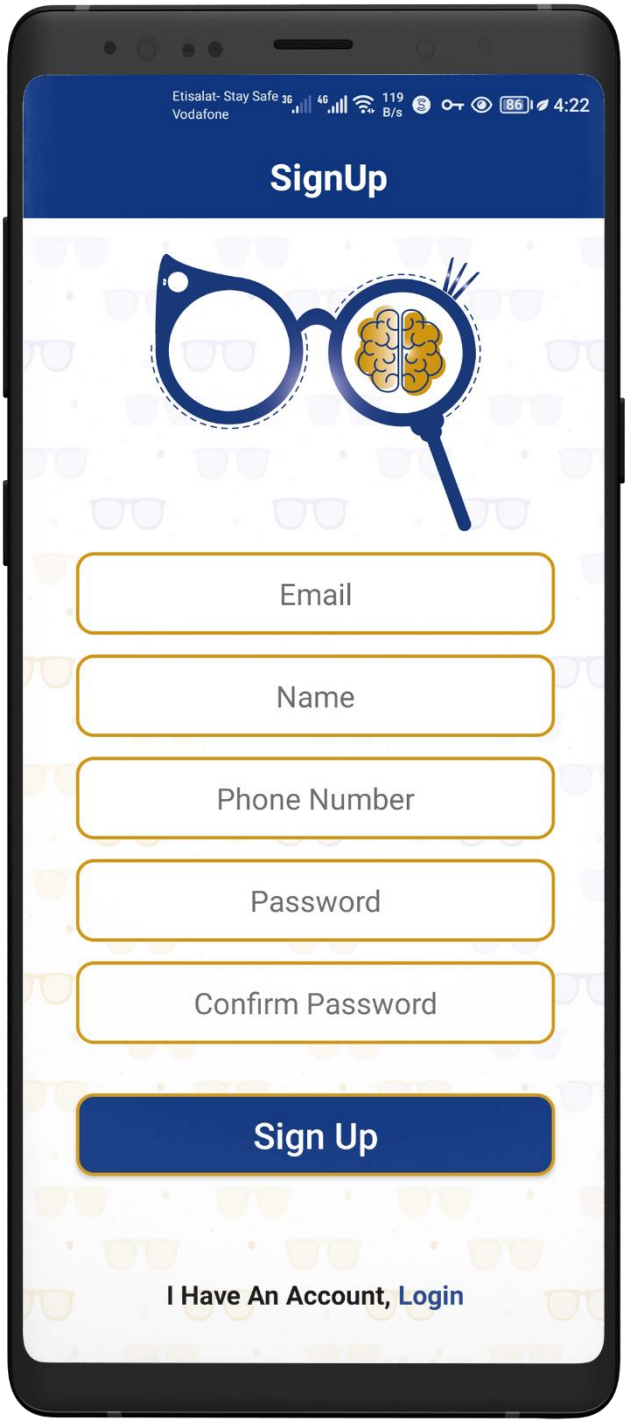
4.1: Implementation Overview

4.1.1: Splash Screen

4.1.2: Login Form



4.1.3: Sign up Form



4.1.4: Select mode Screen

4.1.5: Visually Impaired Features Screen

4.1.6: Alzheimer Feature Screen

4.2: Code Sample Screenshots

4.2.1: Image Captioning Function

```
def ImageCaptioning(img):
    try:
        result = ""
        max_description = 3
        response = cv_client.describe_image_in_stream(img, max_description)
        for caption in response.captions:
            print('Image Description: {0}'.format(caption.text))
            result = 'I can see {0}'.format(caption.text)
            print('Confidence {0}'.format(caption.confidence * 100))
        return result
    except:
        return "There was an Error while Scanning"
```

4.2.2: Object Detection Function

```
def ObjectDetection(img):
    try:
        result = ""
        counter = 0
        detect_object = cv_client.detect_objects_in_stream(img)
        if (len(detect_object.objects) == 0):
            print("No objects detected.")
            result = "I cannot recognize any object "
            return result
        else:
            print(len(detect_object.objects))
            result = "I can recognize "
            for object in detect_object.objects:
                print("'{}' with confidence {:.2f}%".format(object.object_property, object.confidence * 100))
                if (counter != len(detect_object.objects) - 1):
                    result += "'{}', ".format(object.object_property)
                else:
                    result += " and '{}' ".format(object.object_property)
                counter += 1
            result = re.sub(r'[\']', "", result)
            return result
    except:
        return "There was an Error while Detecting"
```

4.2.3: Face Detection Function

```
def FaceDetection(img):
    try:
        response_detected_faces = face_client.face.detect_with_stream(
            img,
            detection_model='detection_01',
            recognition_model='recognition_04',
            return_face_attributes=['emotion', 'gender', 'age'],

        )

        print(response_detected_faces)

        if not response_detected_faces:
            raise Exception('No face detected')

        print('Number of people detected: {0}'.format(len(response_detected_faces)))

        counter = 0
        result = ""
        for face in response_detected_faces:
            gender = face.face_attributes.gender
            emotion = face.face_attributes.emotion
            age = face.face_attributes.age
            neutral = '{0:.0f}%'.format(emotion.neutral * 100)
```

```
            happiness = '{0:.0f}%'.format(emotion.happiness * 100)
            anger = '{0:.0f}%'.format(emotion.anger * 100)
            sandness = '{0:.0f}%'.format(emotion.sadness * 100)
            surprised = '{0:.0f}%'.format(emotion.surprise * 100)
            fear = '{0:.0f}%'.format(emotion.fear * 100)

            print("person " + str(counter))
            print("Gender: " + gender)
            print("Age: " + str(int(age)))
            print("neutral: " + neutral)
            print("Happy: " + happiness)
            print("Angery: " + anger)
            print("Sad: " + sandness)
            print("surprised: " + surprised)
            print("fear: " + fear)

            list = {'Neutral': emotion.neutral, 'Happy': emotion.happiness, 'Angry': emotion.anger,
                    'Sad': emotion.sadness,
                    'Surprised': emotion.surprise, 'Fear': emotion.fear}

            final_emotion = max(list, key=list.get)
            print("Emotion of this person is " + final_emotion)
            counter += 1
            result += "\nPerson " + str(counter) + " Gender: " + gender + " Age: " + str(
                int(age)) + " Emotion of this person is " + final_emotion + "\n"
        return result
    except:
        return "There was an Error while Detecting"
```

4.2.4: Label Detection Function

```
def detect_labels(img):
    try:
        counter = 0
        result = "I can recognize "
        client = vision.ImageAnnotatorClient()

        image = vision.Image(content=img)

        response = client.label_detection(image=image)
        labels = response.label_annotations

        if (len(labels) == 0):
            print("No objects detected.")
            result = "I cannot recognize any label "
            return result
        else:
            for label in labels:
                if (counter != len(labels) - 1):
                    result += label.description + ", "
                else:
                    result += " and " + label.description
                counter += 1
            return result
    except:
        return "There was an Error while Detecting"
```

4.2.5: Text Reader Function

```
def detect_document(img):
    try:
        client = vision.ImageAnnotatorClient()
        image = vision.Image(content=img)
        response = client.document_text_detection(image=image)
        result = response.full_text_annotation.text
        if response.error.message:
            raise Exception(
                '{}\nFor more info on error messages, check: '
                'https://cloud.google.com/apis/design/errors'.format(
                    response.error.message))
        return result
    except:
        return "There was an Error while Scanning"
```

4.2.6: Translate Text

```
def translate_text(text, target):
    try:
        translate_client = translate.Client()

        if isinstance(text, six.binary_type):
            text = text.decode("utf-8")

        result = translate_client.translate(text, target_language=target)

        print(result)

        print(u"Text: {}".format(result["input"]))
        print(u"Translation: {}".format(result["translatedText"]))
        print(u"Detected source language: {}".format(result["detectedSourceLanguage"]))

        return result["translatedText"]
    except:
        return "There was an Error while Translating"
```

4.2.7: Training Face Recognition

```
@app.route("/FaceRecognitionTraining", methods=['POST'])
def Training_Faces():
    if not request.json or 'urls' not in request.json:
        abort(400)

    all_urls = json.loads(request.json['urls'])

    known_names = []
    known_faces = []

    for key, value in all_urls.items():
        response = urllib.request.urlopen(value.replace('\\', ''))
        image = face_recognition.load_image_file(response)

        encoding = face_recognition.face_encodings(image)[0]

        known_faces.append(encoding)
        known_names.append(key)

    data = {}
    for key in known_names:
        for value in known_faces:
            data[key] = value.tolist()
            known_faces.remove(value)
            break

    new_data = json.dumps(data)
    result_dict = {"output": new_data}
    return result_dict
```

4.2.8: Recognize Face Function

```
@app.route("/FaceRecognitionTesting", methods=['POST'])
def Recognize_Face():
    if not request.json or 'encodings' not in request.json:
        abort(400)

    if not request.json or 'image' not in request.json:
        abort(400)

    if request.json['encodings'] != "":
        all_face_encodings = json.loads(request.json['encodings'])
    else:
        result_dict = {"output": "I Found Unknown Person"}
        return result_dict

    im_b64 = request.json['image']

    img_bytes = base64.b64decode(im_b64.encode('utf-8'))

    img = io.BytesIO(img_bytes)

    known_names = list(all_face_encodings.keys())
    known_faces = np.array(list(all_face_encodings.values()))

    image = face_recognition.load_image_file(img)

    locations = face_recognition.face_locations(image, model=MODEL)

    encodings = face_recognition.face_encodings(image, locations)

    for face_encoding, face_location in zip(encodings, locations):

        results = face_recognition.compare_faces(known_faces, face_encoding, TOLERANCE)
        match = None
```

```
        if True in results:
            match = known_names[results.index(True)]
            result = "I Found " + match

        check = all(element == False for element in results)

        if check:
            result = "I Found Unknown Person"

    result_dict = {"output": result}
    return result_dict
```

4.2.9: Currency Recognition

```
def Currency_Recognition(img):
    try:
        Prediction_Key = "a875eed7217c4bae8d68ac7dc8d5e1ef"
        endpoint = "https://southcentralus.api.cognitive.microsoft.com/"
        project_id = "a2856ab9-3ca2-481d-95f9-60106f6000e9"
        iteration_name = "Iteration4"
        Credentials = ApiKeyCredentials(in_headers={"Prediction-Key": Prediction_Key})
        predictor = CustomVisionPredictionClient(endpoint, Credentials)

        currnecy_dic = {}

        with io.BufferedReader(img) as image_file:
            results = predictor.classify_image_with_no_store(project_id, iteration_name, image_file.read()
                                                             )

            for prediction in results.predictions:
                currnecy_dic[prediction.tag_name] = round(prediction.probability, 2)
                # print("\t" + prediction.tag_name +": {0:.2f}%".format(prediction.probability * 100))

        predicted_currency = max(currnecy_dic, key=currnecy_dic.get)

        return predicted_currency
    except:
        return "There was an Error while Detecting"
```

Chapter 5: Discussion, Conclusion, and Future Work

5.1 Discussion:

- It was targeted that helping the visually impaired people to improve their life independency, so, we try to use some AI technologies to make this achievement.

- Now from the app Alzheimer's can use our face recognition model to recognize the people who are close to them by just taking a photo and upload it to the model and then enter their data so that after this they can recognize them easily by just send another photo to our model and then wait for the response which is information about this person.

- Also, visually impaired people can use our technologies that we try to use it to help them with so that they can read text, recognize object, Scene description, and translate our text also and hear this voice in their language.

- Unfortunately, we cannot do all of these technologies perfectly because of our limited resources we use a server and cognitive services so that these technologies we cannot use it for free all the time so we decide to buy a cheap server for our work and in the future, we will try to make more development and enhancements for our project.

5.2 Summary & Conclusion:

- For now, our app used a lot of features which are:
  1. Face Detection
  2. Face Recognition
  3. Object Detection
  4. Text Reader
  5. Image Captioning
  6. Currency Recognition
- So, we used these features to try to improve visually impaired and Alzheimer's life so that they can learn easily, have more privacy in their life and also didn't feel themselves disabled people.
- Our main goal now is to try to make features more accurate and easier to use from the users and try to update the application with the technological development.

5.3: Future Work

1. We can make some enhancements that improve our app's reliability.
2. We can make the visually impaired user not to login, but Alzheimer user should login because he/she has a database of some pictures for the people he/she knows.
3. Solve some server issues because of pricing, buying a server with higher capabilities so that we can run more complex commands and models.
4. Make the response faster than it was by just using another services.
5. Resize our pictures so that it won't take more time to process on it from servers.
6. With the technological development we think that there will be some new cognitive services if this will serve our application and make it faster, we will add it.
7. In Currency recognition we just recognize that there's currency and its class but we cannot count how much he/she have so we can improve it.

8. We'll try to add more languages to the application.

References

1. https://developer.android.com/guide/topics/ui/accessibility/testing
2. https://a11y-guidelines.orange.com/en/mobile/android/talkback/#top
3. https://www.boia.org/blog/google-talkback-an-overview-of-androids-free-screen-reader
4. https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/overview
5. https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/getting-started-build-a-classifier
6. https://www.electroniclinic.com/esp32-bluetooth-wifi-together-for-smart-house-home-technology/
7. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html
8. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/api-conventions.html
9. https://face-recognition.readthedocs.io/en/latest/face_recognition.html
10. https://face-recognition.readthedocs.io/en/latest/readme.html#python-code-examples
11. https://www.geeksforgeeks.org/python-face-recognition-using-gui/
12. https://www.mygreatlearning.com/blog/face-recognition/
13. https://www.analyticsvidhya.com/blog/2021/11/build-face-recognition-attendance-system-using-python/
14. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
15. https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-identity

16. https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
17. https://developer.android.com/reference/android/speech/tts/TextToSpeech
18. https://www.alz.org/alzheimers-dementia/difference-between-dementia-and-alzheimer-s
19. https://cloud.google.com/docs
20. https://cloud.google.com/vision/docs
21. https://cloud.google.com/vision/docs/labels
22. https://docs.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services
23. https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview
24. https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-image-analysis
25. https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-object-detection
26. SE-1 activity diaram & usecase diagram lecture " https://drive.google.com/file/d/1_BkUk1LNQRzTVRQb8owCkwXEsNsWkoZO/view?usp=sharing "
27. https://zenkit.com/en/blog/7-popular-project-management-methodologies-and-what-theyre-best-suited-for/
28. https://www.lucidchart.com/pages/uml-sequence-diagram
29. https://www.tutorialspoint.com/uml/uml_activity_diagram.html
30. https://en.wikipedia.org/wiki/Optical_character_recognition
31. https://anyline.com/news/what-is-ocr#:~:text=Optical%20Character%20Recognition%20%28OCR%29%20defines%20the%20process%20of,as%20the%20process%20of%20turning%20analog%20data%2C%20digital.
32. https://proceedings.neurips.cc/paper/2019/file/680390c55bbd9ce416d1d69a9ab4760d-

Paper.pdf#:~:text=Image%20captioning%20models%20typically%20follow%20an%20encoder
33. https://developers.arcgis.com/python/guide/how-image-captioning-works/
34. https://www.mathworks.com/discovery/object-detection.html
35. https://viso.ai/deep-learning/object-detection/
36. https://thecleverprogrammer.com/2020/12/24/what-is-object detection/