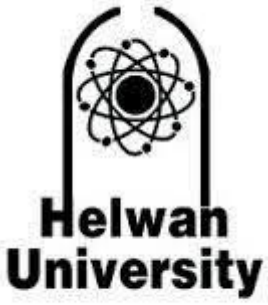
Autonomous vehicles system

Faculty of Science
Department of computer science



Final Year Project

Autonomous vehicles system

By:

Asmaa Mohy

Mahmoud Mohamed Labib

Ibrahem Mohamed Abdulmoneam

Supervised By:

Dr. Mourad Raafat

Supervisor(s) :

Date of examination:

Acknowledgement

I would like to express my very great appreciation to Dr Mourad Raafat for his patient guidance, enthusiastic encouragement and useful critiques of this graduation project (Autonomous vehicles system) and for his valuable technical and mental support on this project.

I wish to thank our parents for their support and encouragement throughout our study for the graduation project.

At last, we would like to thank all the people who helped, supported and encouraged us to successfully finish the graduation project whether they were in the university.

Also, we would like to express our appreciation toward our department for providing us with the information and the tools that we need

Abstract

There are 96% of all car accidents are caused by human error which made us think of the car capable of automatic driving it is expected that within 30 years, almost all cars will be autonomous, and this is what Tesla, Google are trying to create.

In this project, an experiment in making a self-driving car using deep learning and mobile app interface to make it easy to use by all people, where the car responding to the user pickup points and give them a ride to the destination location fully automated with the ability to detect and follow lanes and recognize traffic signs and people on the road.

We will use the Python language for programming our Robot; Python has a standard library in development and a few for AI.

We will train the Car Model to navigate the lane autonomously without having to explicitly write logic to control it. Where we use just the videos of the road and the correct steering angles for each video frame to train the Car Model to drive itself. Where we can use a deep Convolution Neural Network(CNN) to detect road features and make the correct steering decisions. To speed up make decisions we can use a faster region convolution neural network (Faster R-CNN)

In embedded systems development safety and reliability are important quality characteristics. It is thus required to determine the reliability and safety of a complete system including hardware and software. It is reasonable to analyze particular failures that may cause, for example, safety critical situations. The basic idea of our approach is the automated generation of so-called fault trees based on the source code of the software. These may be combined with fault trees based on the electronic circuit design of the hardware. We have implemented a prototype of a fault tree generation tool that is capable to generate fault trees based on C++ code. The fault tree generation tool for electronic circuits has already been used to analyze failure situations in industrial automation applications. If a structural approach is not applicable, stochastic techniques may be used. We developed the reliability assessment tool RAT that supports reliability analysis of software systems.

Here it comes the end-user part, the time that the end-user takes part in, the Mobile Application; we will develop an app using Flutter and Dart technologies to serve a mobile application that makes our robot more interactive, like ordering a ride and confirming pickup and starting the trip and also end it.

Table of Contents

LIST OF FIGURES

Introduction

"**1.3 Million People Die in a Car Crash Every Year**"

We want to start with a pretty staggering statistic, and that is that 1.3 million people die in car crashes every year. That, on average, is over 3,000 deaths a day. While I'm talking here, 100 people are going to die in a car crash. Now, this very, very staggering number doesn't even include the 20 million people every year that will be injured or disabled from cars. Cars are pretty lethal. If you live in the United States and you're a younger person, it is the most likely way that you will die.

Which part is responsible??

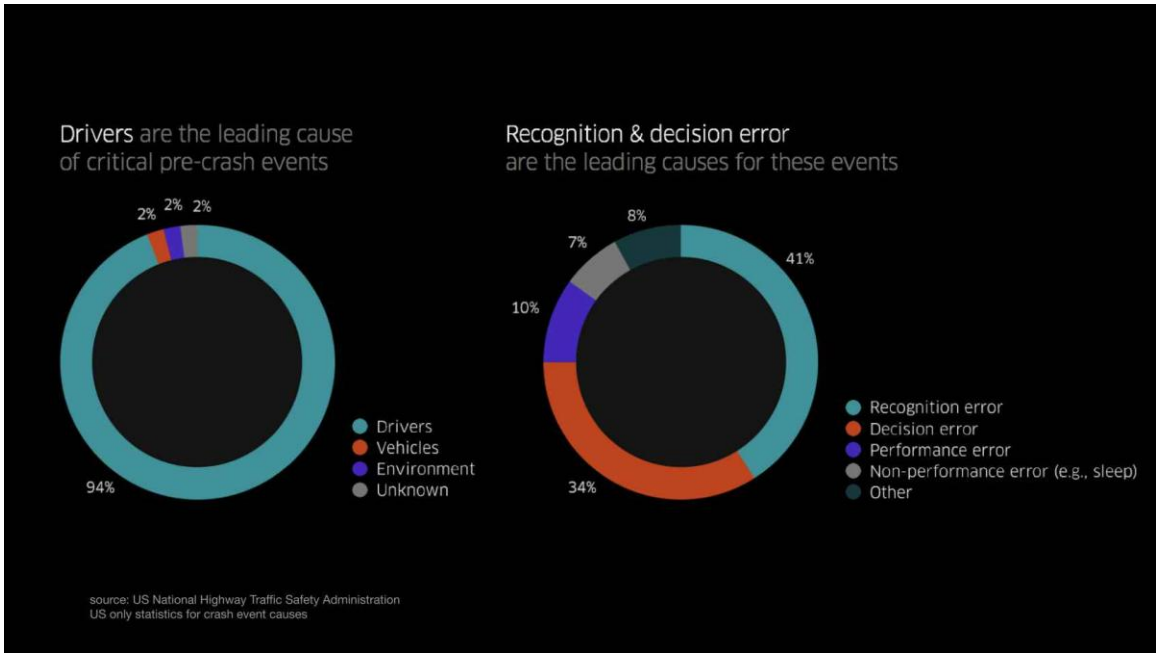


Figure 1-1: leading cause of critical pre-crash events

For brevity, There are 96% of all car accidents are caused by humanerror which made us think of the car
capable of automaticdriving ,It is expected that within 30 years almost all cars will be autonomous, and this is what Tesla,Google, Uber are trying to create.

Researchers forecast that by 2025 we'll see approximately 8 million autonomous or semi-autonomous vehicles on the road. Before merging onto roadways, self-driving cars will first have to progress through 6 levels of driver assistance technology advancements.

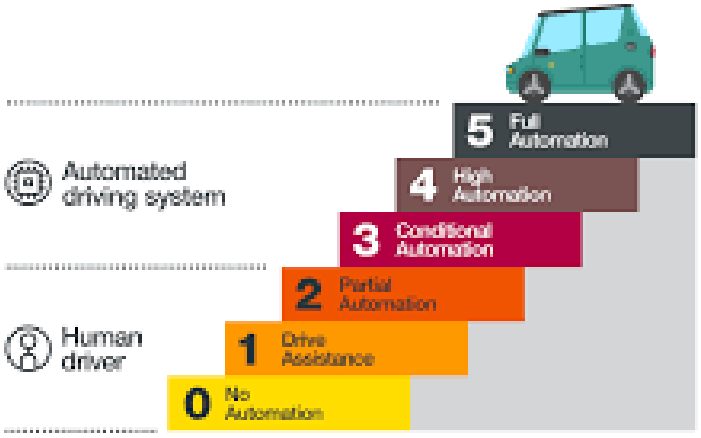


Figure 1-2:6 levels of driver

Self-driving system overview

- **Sensors:** hardware the gathers raw data about the environment
- **Perception:** computer vision techniques for example(object detection and classification)
- **Planning** :Have tasks are Behavioral planning ,trajectories generation.
- **Control :**Have tasks are Acceleration , brake ,steering .

In this project, an experiment in making a self-driving car using deep learning, where the car only isable recognize traffic signs .

Basics

- **Perception step:**

We focus on classify and recognize various traffic signs found in roadways. Traffic signs classification is one of the major part of Self-Driving Car system. In the automotive industry, machine learning and deep learning are playing an important role in the development of advanced driver assistance system for improving driver safety. Using Deep Learning it can be developed very easily because it is considered as one of the most emerging technologies. Through this project I'll guide you about step by step development of Traffic sign classifier.

To complete this project successfully you need to use the neural network construction and basic image processing techniques,
The dataset we'll be using is German Traffic Sign Recognition Benchmark(GSTRB). This dataset contains 43 different classes of images.

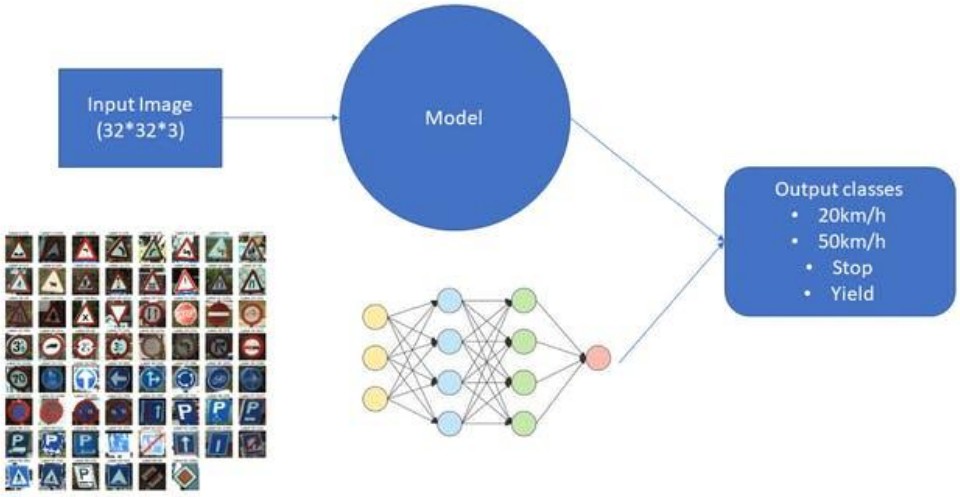


Figure 1-3:Overview of model

AI(Artificial Intelligence) may be defined as the branch of computer science that is concerned with the automation of intelligent behavior other definition The automation of activities that we associate with human thinking(e.g., decision-making, learning…).
One of AI fields which is important and this project related with it is **Computer vision.**
Computer vision is a field that includes methods foracquiring, processing, analyzing,andunderstandingimages and, in general, high dimensional data fromthe real world in order to produce numerical orsymbolic information, e.g., in the forms of decisions.

1. **AI , ML and DL Definitions:**

Artificial intelligence and machine learning are the part of computer science that are correlated with each other.

These two technologies are the most trending technologies which are used for creating intelligent systems.

Machine learning is a subset of AI which allows a machine to automatically learn from past data without programming explicitly.

The goal of ML is to allow machines to learn from data so that they can give accurate output.

we teach machines with data to perform a particular task and give an accurate result.

Machine learning is working to create machines that can perform only those specific tasks for which they are trained.

It includes learning and self-correction when introduced with new data.

Machine learning can also be divided into mainly three types that are:

- Supervised learning.
- Unsupervised learning.
- Reinforcement learning.

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:

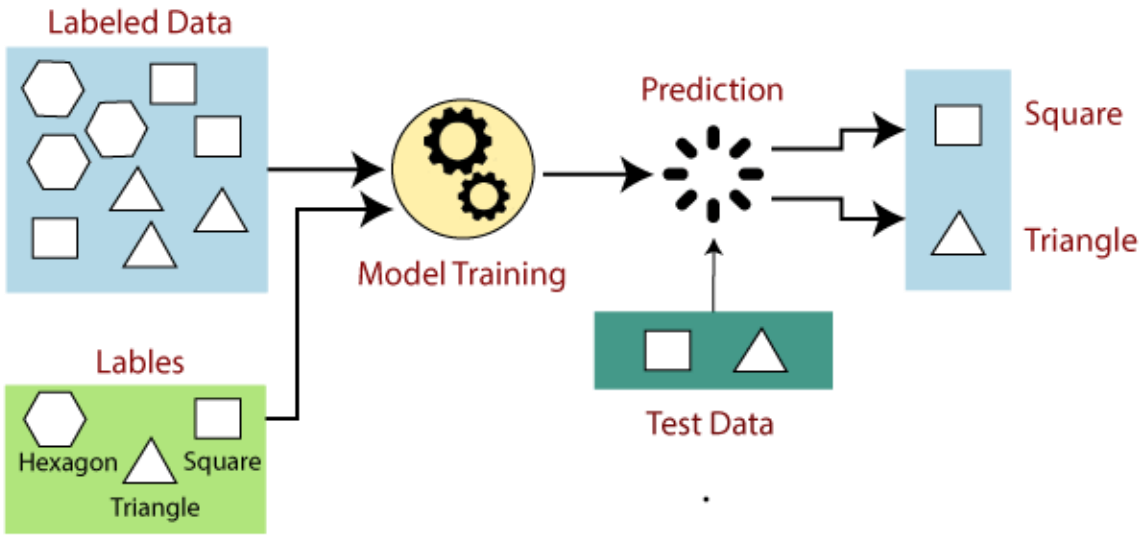
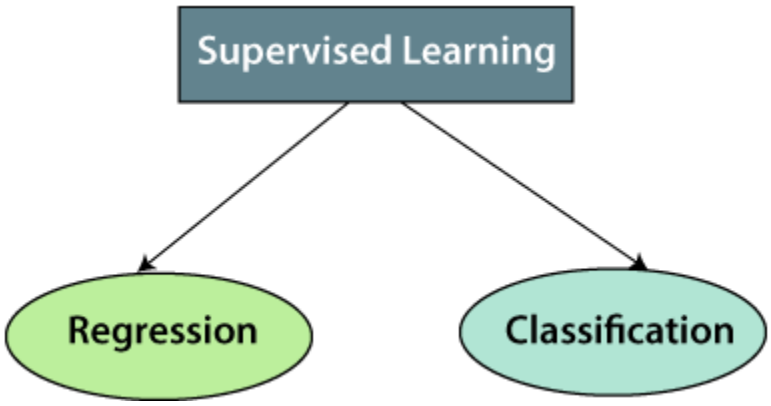


Figure 1-4: Supervised learning

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:

- Regression
- Classification



The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

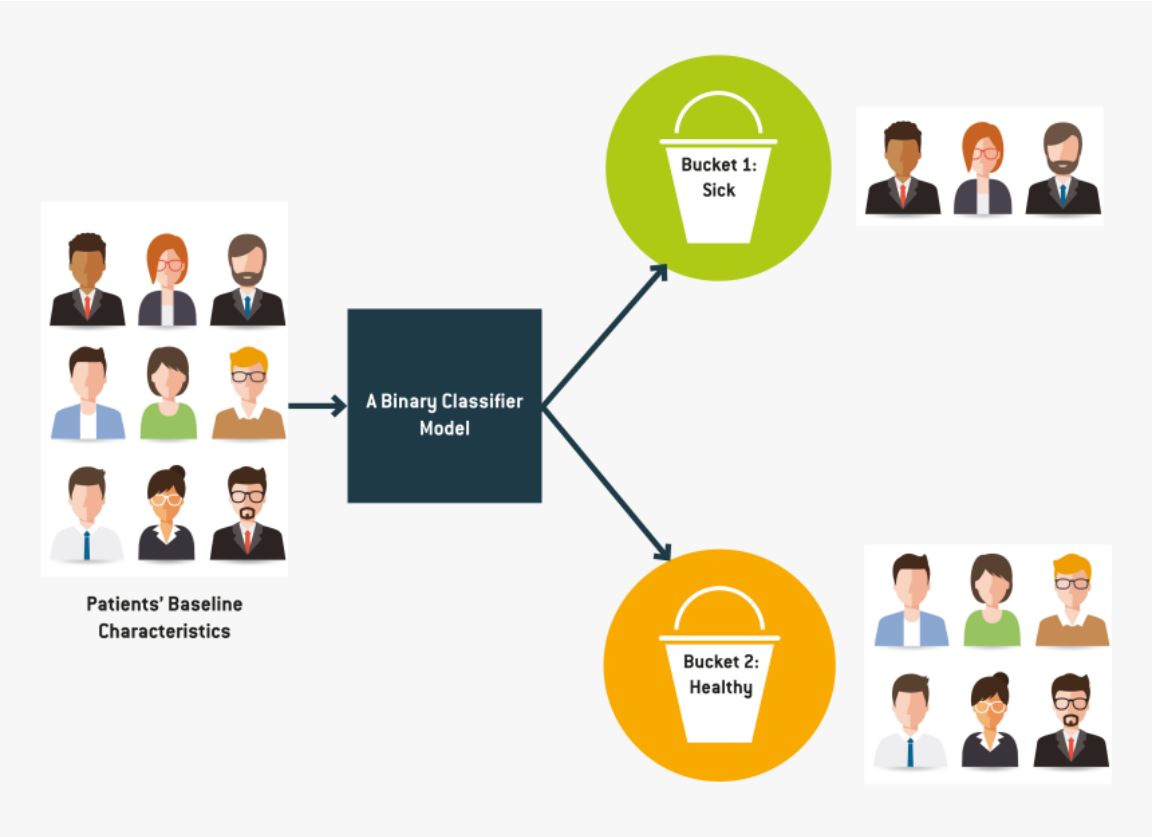


Figure 1-6:Binary classifier model.

Deep Learning is the subset of machine learning or can be said as a special kind of machine learning.

It works technically in the same way as machine learning does, but with different capabilities and approaches.

It is inspired by the functionality of human brain cells, which are called ***neurons***, and leads to the concept of ***artificial neural networks***.

It is also called a ***deep neural network*** or deep neural learning.

In deep learning, models use different layers to learn and discover insights from the data.

Some popular applications of deep learning are self-driving cars.

Some popular deep learning models are:

- Convolutional Neural Network
- Recurrent Neural Network
- Classic Neural Networks.

We can understand the working of deep learning with the same example of identifying cat vs. dog.

The deep learning model takes the images as the input and feed it directly to the algorithms without requiring any manual feature extraction step.
The images pass to the different layers of the artificial neural network and predict the final output.

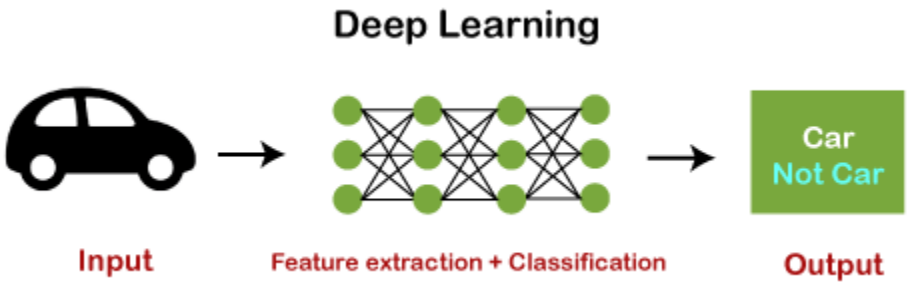


Figure 1-7: Deep Learning model

Deep Learning algorithms highly depend on a large amount of data, so we need to feed a large amount of data for good performance.
Deep Learning takes a long execution time to train the model, but less time to test the model.
The deep learning model needs a huge amount of data to work efficiently, so they need GPU's and hence the high-end machine.

Artificial neural network learning algorithm:

is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

Common Types of neural network :

- Artificial Neural Networks (ANN)
- Convolution Neural Networks (CNN)
- Recurrent Neural Networks (RNN)

In this project , we use CNN to recognize Traffic sign

So we focus on

CNN network:

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers and fully connected layers.

The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.

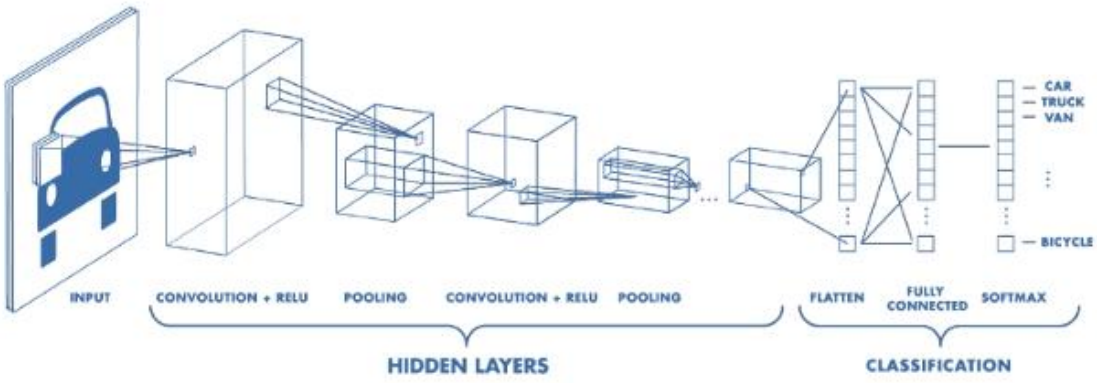


Figure 1-8: CNN network.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

Pooling Layer:

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

Fully Connected Layer:

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect.

Transfer learning:

is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources

required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

A range of high-performing models have been developed for image classification and demonstrated on the annual **ImageNet** Large Scale Visual Recognition Challenge, or ILSVRC.

This challenge, often referred to simply as ImageNet, given the source of the image used in the competition, has resulted in a number of innovations in the architecture and training of convolutional neural networks. In addition, many of the models used in the competitions have been released under a permissive license.

These models can be used as the basis for transfer learning in computer vision applications.

In computer vision, transfer learning is usually expressed through the use of pre-trained models. A pre-trained model is a model that was trained on a large benchmark dataset to solve a

problemsimilar to theone that we want to solve.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	–
ResNet101	171 MB	0.764	0.928	44,707,176	–
ResNet152	232 MB	0.766	0.931	60,419,944	–
ResNet50V2	98 MB	0.760	0.930	25,613,800	–
ResNet101V2	171 MB	0.772	0.938	44,675,560	–
ResNet152V2	232 MB	0.780	0.942	60,380,648	–
ResNeXt50	96 MB	0.777	0.938	25,097,128	–
ResNeXt101	170 MB	0.787	0.943	44,315,560	–
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	–
NASNetLarge	343 MB	0.825	0.960	88,949,818	–

Figure 1-9: ImageNet models

In this project we used **Mobilenetnetwork**to scale to hardware (raspberry pi) , but it is not efficient as CNN network.

Measurement of performance:

1. **confusion matrix:** is a tabular way of visualizing the performance of your prediction model. Each entry in a confusion matrix denotes the number of predictions made by the model where it classified the classes correctly or incorrectly.

2. **Accuracy:** It gives you the overall accuracy of the model, meaning the fraction of the total samples that were correctly classified by the classifier.
3. **Precision**: It tells you what fraction of predictions as a positive class were actually positive.
4. **Recall**: It tells you what fraction of all positive samples were correctly predicted as positive by the classifier.
5. **F1-score**: It combines precision and recall into a single measure. Mathematically it's the harmonic mean of precision and recall.

$$F_1\text{−score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

In this project there are 43 class , so we use Confusion Matrix for Multi-Class Classification , for example :
if we take class Apple, then let's see what are the values of the metrics from the confusion matrix.

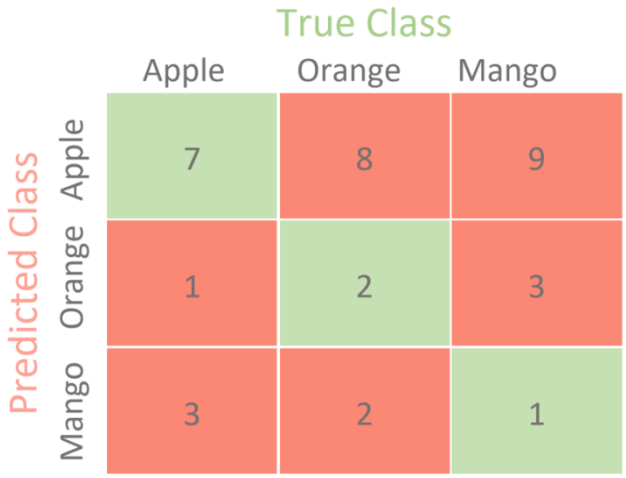


Figure 1-10: confusion matrix.

Then:

- **TP = 7**
- **TN = (2+3+2+1) = 8**
- **FP = (8+9) = 17**

- **FN = (1+3) = 4**

We can combine the F1-score of each class to have a single measure for the whole model. There are a few ways to do that:

1. **Micro F1**:This is called micro-averaged F1-score. It is calculated by considering the total TP, total FP and total FN of the model. It does not consider each class individually, It calculates the metrics globally.
   Precision = Recall = Micro F1 = Accuracy
2. **Macro F1** : This is macro-averaged F1-score. It calculates metrics for each class individually and then takes unweighted mean of the measures.
3. **Weighted F1**: The last one is weighted-average F1-score. Unlike Macro F1, it takes a weighted mean of the measures. The weights for each class are the total number of samples of that class.

Embedded systems Definition.

. All systems that contain one or more processing units usually it is micro-controller to do **specificfunctionalities**(this processor is not for general processing like Pc's)and give responses upon receiving inputs.

. the word embedded reflect the fact that these systems are usually an integral part of a larger system, known as the embedding system. Multiple embedded systems can co-exist in an embedding system.

. Embedded Systems are **application-specific**systems which contain hardware and software tailored for a particular task and are generally part of a larger system.

. For more explain, answer this question “Can personal computer be considered as an embedded system as it integrates hardware and software to perform functions? Why?”

Answer → No, pc cannot be considered as an embedded system because:

1- It uses a General-Purpose Processor.
2- The system is built independently from the software runs on it.

Embedded systems characteristics.

- **Reliability**
  - •Personal computer programs such as word processors and games do not need to achieve the same standard of reliability that a microcontroller application must. Errors in programs such as word processors may result in errors in a document or loss of data
  - •An error in an embedded system application such as a TV remote control or compact disc player will result in a product that does not work and consequently does not sell.
  - • An error in embedded system application such as an antilock braking system or autopilot could be fatal.

- **Efficiency**
  - • Considered in real time applications.
  - • A real time application is one in which must be able to act at a speed corresponding with the occurrence of an actual process. Must compute certain results in certain real-time without delay.
  - • Some Embedded systems may have real-time performance constraints that must be met, for reasons such as safety and usability. Others may have low or no timing performance requirements, allowing the system hardware to be simplified to reduce costs.
- **Tightly-constrained**
  - Embedded Systems normally come with constraints in hardware resources:
  - Processing (speed)
  - Memory (Data)

- Storage (Code Size)
- Most of the time it targets real time objectives, this means,
  - It needs to be fast and efficient (Response Time).
  - It needs to be predictable (execution time known ahead, and almost constant)
- Power limited(battery operated devices).
- Cost
- Size

- **Single-functionality**
  Dedicated to performing a specific function and include processors dedicated to a specific function.

- **Complex functionality**
  Often have to run sophisticated algorithms or multiple algorithms Cell phone, laser printer, automotive, etc.

- **Safety-critical**
  Must not endanger human life and the environment.

- **Maintainability**
  The ability to modify the system after its initial release and enhance its performance like execution time, code and memory size.

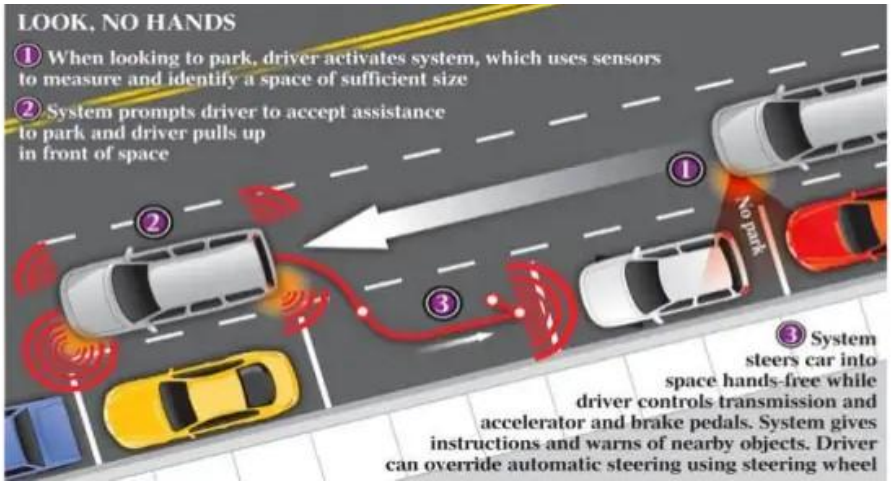
- **Interactive System**
  An Embedded system should be as interactive as possible so that it is easily understandable, Operated and Handled by a user and provide ease of task.

- **Strong association between the HW and SW**

Embedded systems Applications

- **Automotive**

- Automotive, Electronics represents 40% of total cost of a car. • 90% of new car features require software.



- **ECUs are used in different functions of the car :**
  - Engine Control
  - Transmission Control
  - Fuel Efficiency Control
  - Electric Power Steering
  - Speed Control • Brake Control
  - Suspension System Control
  - Battery Management

- Seat Control
- Door Control
- Electric Windows Control

- **Robotics**

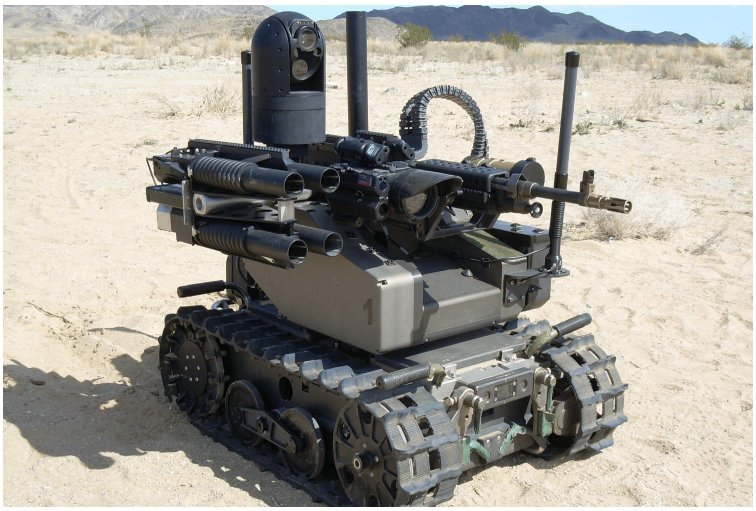




- Product: NASA's Mars Sojourner Rover 1996, low-costspacecraft. Using 8-bit Intel Microprocessor 80C85.


- **Image Processing**

- **Military**



Embedded system Design

- Trade off between SW & HW For a certain application:

- **Software characteristics** :

• Advantage

1- Highly configurable
2- Shorter development cycle
3- Easier in versions updates
4- Cheaper

• Disadvantages

1- Constrained with processor speed which may satisfy real time application and may not.

- **Hardware characteristics**

• Advantage

1- Better performance in high speed real time application Shorter development cycle

• Disadvantages

1- Longer development cycle.
2- Customized for specific application, not updatable (unchangeable).

- Design goals
  1- Performance (overall speed execution time and throughput).
  2- Functionality and user interface.
  3- Manufacturing cost.
  4- Power consumption.
  5- Safety.

Embedded system HW

- Micro-processor main components
  1- Central Processing Unit (CPU)
  2- Memory units.
  3- Input and Output ports (GPIO or DIO).
  4- Timers.
  5- Watch dog timer
  6- Buses.
  7- Interrupt circuit.

Mobile Application

The process of creating software for Smartphone and digital assistants, usually for Android and IOS, is known as mobile application development. The application can be preinstalled, be downloaded from a mobile app store, or be accessible via a mobile web browser. Java, Swift, C#, and Flutter are some of the programming utilized in this type of software development.

In our case we will use Flutter using Dart Language; Flutter is a Cross-platform that can make IOS, Android Apps With the same source code

Development with Flutter

Flutter is a cross-platform software development framework first introduced by Google in 2015 and released in May of 2017. Flutter has significantly expanded in recent years, expanding its capabilities to include not only IOS and Android mobile development, but also web and desktop applications. Let's take a look at the issue and see why it's so popular nowadays.

Dart: the language used by Flutter

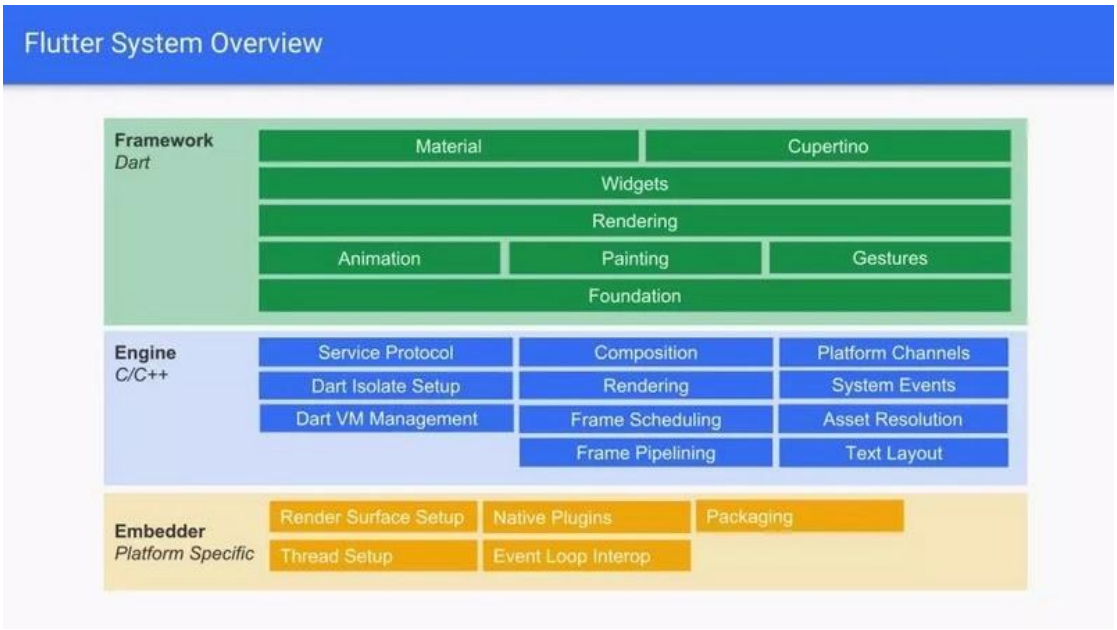
Let's start by looking at the Dart programming language that was utilized to create the Flutter framework. Dart is a Google-developed object-oriented programming language that was initially released in 2011. Dart has continually evolved since then, providing new features. Among others, it is worth mentioning the “dart2native” feature that allows compiling it for Windows, Linux, and macOS platforms as a desktop application. As of the writing this article, the desktop solution is not production-ready but the outlook seems promising. Aside from this, the Dart program can be composed into a self-contained executable file or compiled to JavaScript.

In general, the Dart language is easy to study. Its syntax is pretty similar to Java, Swift, or Kotlin languages. Moreover, the Dart software development kit (SDK) is shipped with a stand-alone Dart Virtual Machine (VM) that allows you to build the code in a command-line interface (CLI) environment and if you are not familiar with CLI then you can play around with the Dart in DartPad. Dart Pad is an online editor that provides access to Dart’s API and allows you to compile the Dart code.

Flutter Framework

As was previously mentioned, the Flutter framework was first unveiled by Google in 2015. Its code name was “Sky” and it ran on the Android operating system. The first stable release was delivered on the 4 of December 2018 and on May 6, 2020, the Dart SDK version 2.18 and Flutter version 1.17 were released delivering a build with Metal API integration. This has provided performance improvements to the iOS platform.

The framework is written using C, C++, and Dart languages and uses Google’s Skia Graphics Engine for user interface rendering. This graphics engine is used for such known products as Google Chrome, Chrome OS, Chromium OS, Mozilla Firefox, Mozilla Thunderbird, Android, Firefox OS, and now the Flutter.



Flutter runs using Dart virtual machine (VM) on Windows, Linux, and macOS operating systems. The Dart VM uses a just-in-time (JIT) code compilation which provides a development-time-saving feature such as hot-reloading. While the developer writes and debugs the mobile application, the JIT compilation injects new code into the running application. It provides a stateful hot-reload feature, where, in most cases source code changes can be immediately reflected in the running application without requiring a restart or any loss of state. This saves the developers’ a lot of time in the end. When it comes to releasing the application, the Dart VM uses the ahead-of-time (AOT) compilation which further converts Dart code into a native platform-dependent machine code making Flutter’s high performance on mobile devices possible.

The Flutter framework was designed using some principles that should be described separately. These principles are “Everything’s a widget”, “Composition > Inheritance”, “Widget tree”.

In Flutter almost everything is a widget and it is the basic building block of the application. Compared to other frameworks Flutter does not have separate controllers, views, or layouts. Almost every aspect of Flutter development is covered by a unified building block - the widget. A widget can be a unique button, style element, or a separate pop-up screen, etc.

The composition approach is better than inheritance. Often widgets are composed of other smaller widgets and that is the composition based approach. Using a Flutter API allows you to combine multiple widgets to elicit the exact behavior you need.

A widget tree concept is basically an implementation of nested widgets that represent user interface components. These widgets could be stateless or stateful and the difference between them is in keeping with the widget’s state. A useful feature that helps in managing the applications’ states.

The most interesting thing about Flutter is the user interface components that were delivered with the latest releases. Google went by its own path and created two sets of widgets, Cupertino (iOS) and Material (Android). These sets of widgets are responsible for the user interface (UI) and include every component that you may need for Android and iOS development. These widgets are not connected with the native API of iOS or Android like in React Native but work as standalone Flutter components with appropriate rendering speed and animation. That is one of the key selling features of Flutter. Users will not be affected by poor user experience, because of it.

Flutter Usage on our project

We want to create a Flutter app that is similar to Uber But instead of a real driver come to pick you up, the **Autonomous vehicles** will reach out to pick you, and give you the ride **autonomously.**

What are APIs?

Application Programming Interfaces (APIs) are constructs made available in programming languages to allow developers to create complex functionality more easily. They abstract more complex code away from you, providing some easier syntax to use in its place.

As a real-world example, think about the electricity supply in your house, apartment, or other dwellings. If you want to use an appliance in your house, you plug it into a plug socket and it works. You don't try to wire it directly into the power supply — to do so would be really inefficient and, if you are not an electrician, difficult and dangerous to attempt.

What can APIs do?

There are a huge number of APIs available in modern browsers that allow you to do a wide variety of things in your code. You can see this by taking a look at the MDN APIs index page.

Common browser APIs

In particular, the most common categories of browser APIs you'll use (and which we'll cover in this module in greater detail) are:

- **APIs for manipulating documents** loaded into the browser. The most obvious example is the DOM (Document Object Model) API, which allows you to manipulate HTML and CSS — creating, removing and changing HTML, dynamically applying new styles to your page, etc. Every time you see a popup window appear on a page or some new content displayed, for example, that's the DOM in action. Find out more about these types of API in Manipulating documents.
- **APIs that fetch data from the server** to update small sections of a webpage on their own are very commonly used. This seemingly small detail has had a huge impact on the performance and behavior of sites — if you just need to update a stock listing or list of available new stories, doing it instantly without having to reload the whole entire page from the server can make the site or app feel much more responsive and "snappy". APIs that make this possible include `XMLHttpRequest` and the Fetch API. You may also come across the term **Ajax**, which describes this technique. Find out more about such APIs in Fetching data from the server.
- **APIs for drawing and manipulating graphics** are now widely supported in browsers — the most popular ones are Canvas and WebGL, which allow you to programmatically update the pixel data contained in an HTML `<canvas>` element to create 2D and 3D scenes. For example, you might draw shapes such as rectangles or circles, import an image onto the canvas, and apply a filter to it such as sepia or grayscale using the Canvas API, or create a complex 3D scene with lighting and textures using WebGL. Such APIs are often combined with APIs for creating animation loops (such

as window.requestAnimationFrame()) and others to make constantly updating scenes like cartoons and games.

- **Audio and Video APIs** like HTMLMediaElement, the Web Audio API, and WebRTC allow you to do really interesting things with multimedia such as creating custom UI controls for playing audio and video, displaying text tracks like captions and subtitles along with your videos, grabbing video from your web camera to be manipulated via a canvas (see above) or displayed on someone else's computer in a web conference, or adding effects to audio tracks (such as gain, distortion, panning, etc).
- **Device APIs** are basically APIs for manipulating and retrieving data from modern device hardware in a way that is useful for web apps. Examples include telling the user that a useful update is available on a web app via system notifications (see the Notifications API) or vibration hardware (see the Vibration API).
- **Client-side storage APIs** are becoming a lot more widespread in web browsers — the ability to store data on the client-side is very useful if you want to create an app that will save its state between page loads, and perhaps even work when the device is offline. There are a number of options available, e.g. simple name/value storage with the Web Storage API, and more complex tabular data storage with the IndexedDB API.

Json API

JSON API documents work as any other API format; you send a request to an endpoint and receive your document. The JSON API specification defines how resources are structured. This structure helps in normalizing how you consume the API.

Compound documents

To reduce the number of HTTP requests, servers MAY allow responses that include related resources along with the requested primary resources. Such responses are called “compound documents”.

A compound document is a resource which includes the data of included relations. For example, when requesting an article as shown earlier it may include the data of the author so you don’t need a second call to fetch the author of the article.

This is awesome for a few reasons. Getting the full data of a resource you want to consume in one go is unheard of unless specific endpoints are programmed. On the server side of things, caching and invalidating a request like this is easy.

API Usage in the scope of Our Project

First of all, we need to know how many platforms we have and want to communicate

The **Autonomous vehicles**	The **Server/Database**	**Flutter Application**

Here are the 3 main platforms we want to communicate, what they should communicate about, let's see

If the user wants a ride from **Al-Shrouq City** to **Helwan University** He should select the final destination he wants from the app and adjust the point on the map and press start trip

By pressing start trip these are the communications happening at this moment

Flutter Application Sends **Longitudes** and **latitudes** for the current and destination point to

The **Server/Database** specifically on **Trips** Table of the **MySQL** Database

All **Autonomous vehicles** searching for the closest user to them on the Database Using the API and when the **Autonomous vehicle** finds the closest user that already started a trip, it connects to him automatically and goes to his Current point (location), then

Autonomous vehicle Sends **new state** of the trip which is **waiting for the user to enter,**

At the same time the **Flutter Application**will be waiting for a **response** from the **API** about the state changing and whenthe **state** changed to **waiting for the user to enter,** Pops-up a message to the user asking him to enter the car next to him.

This is briefly how the communication works between these 3 platforms, will cover it in-depth in the next few chapters.

Technology Used In Project

- Machine Learning Technologies
    1. **Python**
        - **Python** is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.
    2. **TensorFlow**
        - **TensorFlow** is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.
    3. **openCV**
        - **openCV**is an open-source library that includes several hundreds of computer vision algorithms.
    4. **Machine Learning Classification**
        - **Classification** refers to a predictive modeling problem where a class label is predicted for a given example of input data.
    5. **NumPy**
        - **NumPy** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices
    6. **Pandas**
        - **Pandas**is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.
    7. **sklearn**
        - **sklearn** library for the Python programming language. It features various classification, regression and clustering algorithms.
    8. **Matplotlib/seaborn**
        - Library for creating static, animated, and interactive visualizations in Python.

- Embedded Systems Technologies

1. **Python**
    - **Python** is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.
2. **Embedded C**
    - Embedded C is a set of language extensions for the C programming language by the C Standards Committee to address commonality issues that exist between C extensions for different embedded systems.
3. **Embedded Linux**
    - **Embedded Linux** Operating systems based on the Linux kernel are used in embedded systems such as consumer electronics..
4. **Raspberry PI**
    - **Raspberry Pi** is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi project originally leaned towards the promotion of teaching basic computer science in schools and in developing countries.
5. **Arduino Mega**
    - **NumPy** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices

- Mobile Application Technologies

1. **Flutter**
    - **Flutter** is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows.
2. **Dart**
    - **Dart** is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications.
3. **PHP**
    - **PHP** is a general-purpose scripting language especially suited to web development.
4. **MySQL**
    - **MySQL** is an open-source relational database management system.

5. **JSONAPI**
    - **JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write.

Example of Autonomous vehicles system

- **Waymo**

Waymo LLC is an American autonomous driving technology development company. It is a subsidiary of Alphabet Inc, the parent company of Google. Waymo operates a commercial self-driving taxi service in the greater Phoenix, Arizona area called "Waymo One", with Chandler, Arizona fully mapped. In October 2020, the company expanded the service to the public, and it is the only self-driving commercial service that operates without safety backup drivers in the vehicle. Waymo also develops driving technology for use in other vehicles, including delivery vans and Class 8 tractor-trailers for delivery and logistics.

Methodology

We used in out project the “Agile Software methodology”

- Because Agile Methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

Four values of Agile

1- individuals and interactions over processes and tools;

- The first value in the Agile Manifesto is “Individuals and interactions over processes and tools.” Valuing people more highly than processes or tools is easy to understand because it is the people who respond to business needs and drive the development process. If the process or the tools drive development, the team is less responsive to change and less likely to meet customer needs. Communication is an example of the difference between valuing individuals versus process. In the case of individuals, communication is fluid and happens when a need arises. In the case of process, communication is scheduled and requires specific content.

2- working software over comprehensive documentation;

- Historically, enormous amounts of time were spent on documenting the product for development and ultimate delivery. Technical specifications, technical requirements, technical prospectus, interface design documents, test plans, documentation plans, and approvals required for each. The list was extensive and was a cause for the long delays in development. Agile does not eliminate documentation, but it streamlines it in a form that gives the developer what is needed to do the work without getting bogged down in minutiae. Agile documents requirements as user stories, which are sufficient for a software developer to begin the task of building a new function.The Agile Manifesto values documentation, but it values working software more.

3- customer collaboration over contract negotiation; and.

- Negotiation is the period when the customer and the product manager work out the details of a delivery, with points along the way where the details may be renegotiated. Collaboration is a different creature entirely. With development models such as Waterfall, customers negotiate the requirements for the product, often in great detail, prior to any work starting. This meant the customer was involved in the process of development before development began and after it was completed, but not during the process. The Agile Manifesto describes a customer who is engaged and collaborates throughout the development process, making. This makes it far easier for development to meet their needs of the

customer. Agile methods may include the customer at intervals for periodic demos, but a project could just as easily have an end-user as a daily part of the team and attending all meetings, ensuring the product meets the business needs of the customer.

4- responding to change over following a plan.

- Traditional software development regarded change as an expense, so it was to be avoided. The intention was to develop detailed, elaborate plans, with a defined set of features and with everything, generally, having as high a priority as everything else, and with a large number of many dependencies on delivering in a certain order so that the team can work on the next piece of the puzzle.

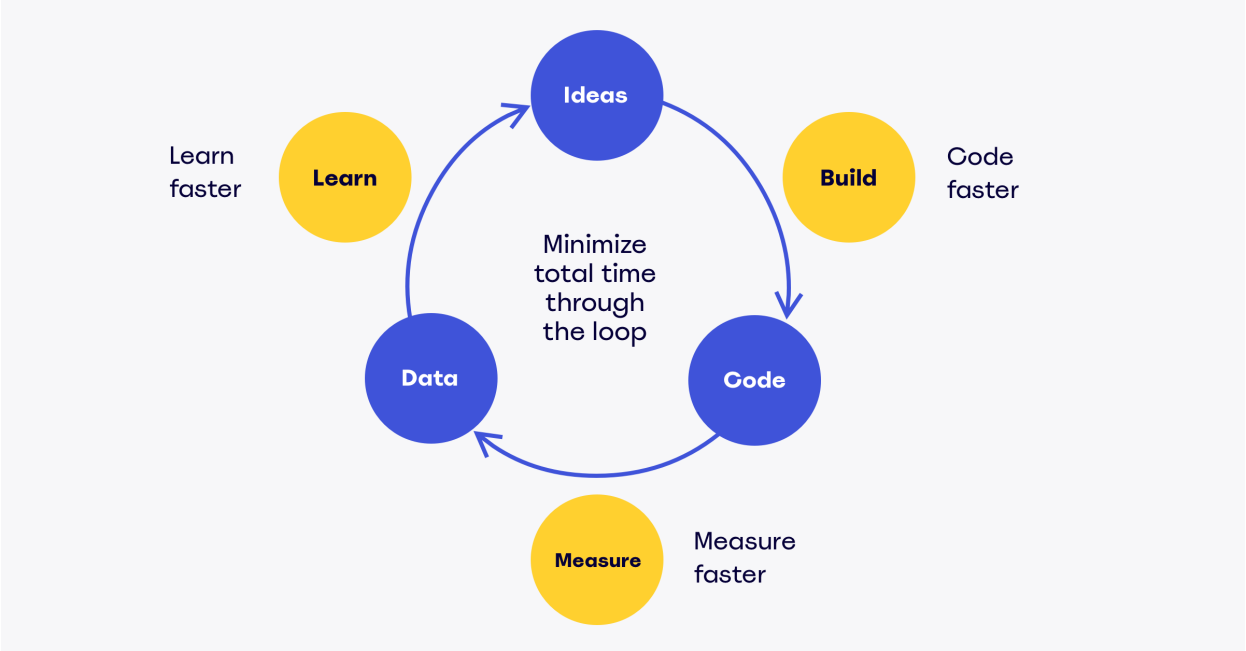
- With Agile, the shortness of an iteration means priorities can be shifted from iteration to iteration and new features can be added into the next iteration. Agile's view is that changes always improve a project; changes provide additional value.

■ Agile Methodology 12 Principles

- The Twelve Principles are the guiding principles for the methodologies that are included under the title "The Agile Movement." They describe a culture in which change is welcome, and the customer is the focus of the work. They also demonstrate the movement's intent as described by Alistair Cockburn, one of the signatories to the Agile Manifesto, which is to bring development into alignment with business needs.

1- Customer satisfaction through early and continuous software delivery – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
2- Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.
3- Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.
4- Collaboration between the business stakeholders and developers throughout the project – Better decisions are made when the business and technical team are aligned.
5- Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.
6- Enable face-to-face interactions – Communication is more successful when development teams are co-located.
7- Working software is the primary measure of progress – Delivering functional software to the customer is the ultimate factor that measures progress.

8- Agile processes to support a consistent development pace – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
9- Attention to technical detail and design enhances agility – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
10- Simplicity – Develop just enough to get the job done for right now.
11- Self-organizing teams encourage great architectures, requirements, and designs – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
12- Regular reflections on how to become more effective – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.



- How our Team are working Together in project
  - In our project we have three component
    1. Computer Vision
    2. Embedded Systems
    3. Mobile Application

Computer Vision

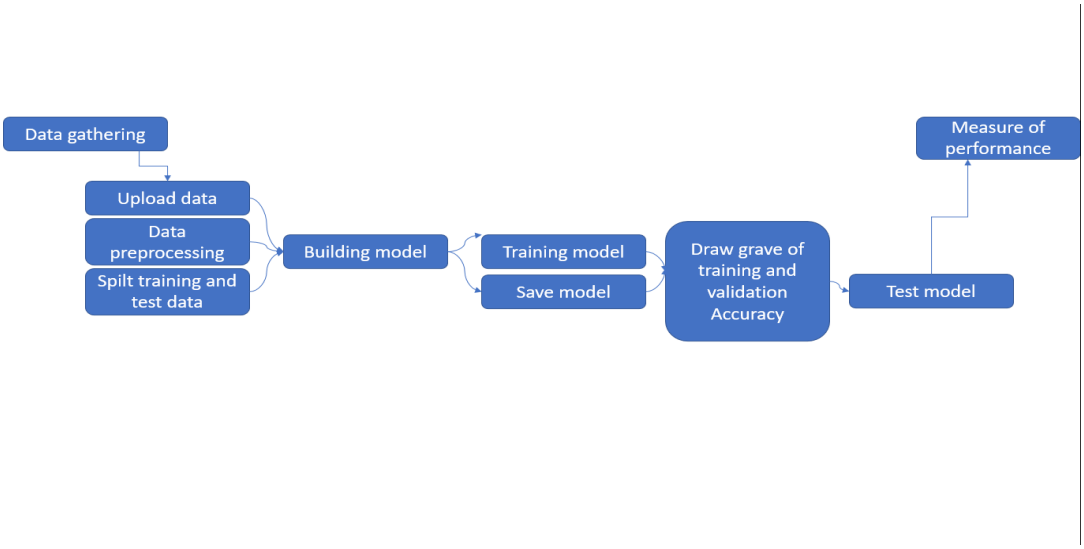


Figure 3-1:Our model.

- Data gathering
  - ➢ We use Kaggle platform to gather data for train model and test We use dataset with name : GTSRB - German Traffic Sign Recognition Benchmark The German Traffic Sign Benchmark is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011. We cordially invite researchers from relevant fields to participate: The competition is designed to allow for participation without special domain knowledge. Our benchmark has the following properties:
    - Single-image, multi-class classification problem
    - More than 50,000 images in total
    - Large, lifelike database
- Upload data
  - ➢ We use cloud computing such as colab platform To upload data and build our model Explore the dataset: Data contain 3 folders and 3 files
    - Our ‘train’ folder contains 43 folders each representing a different class. The range of the folder is from 0 to 42.
    - Our dataset contains a test folder and in a test.csv file, we have the details related to the image path and their respective class labels.
- Data preprocessing
  - ➢ We need to resize all image with size 30x30 and convert the list into numpy arrays for feeding to the model.

- Split training and test data
  - using the sklearn package to split training and testing data.
- Building model
  - we use a lot of method to build our model
    - **Using normal CNN** : Using sequential model in Keras package by The architecture of our model is:
      - 2 Conv2D layer (filter=32, kernel_size=(5,5), activation="relu")
      - MaxPool2D layer (pool_size=(2,2))
      - Dropout layer (rate=0.25)
      - 2 Conv2D layer (filter=64, kernel_size=(3,3), activation="relu")
      - MaxPool2D layer (pool_size=(2,2))
      - Dropout layer (rate=0.25)
      - Flatten layer to squeeze the layers into 1 dimension
      - Dense Fully connected layer (256 nodes, activation="relu")
      - Dropout layer (rate=0.5)
      - Dense layer (43 nodes, activation="softmax") This way we use in our project since it reduce high accuracy and scale to classify
    - **Using mobileNet** : Using functional API way to create model in Keras package by transfer learning techniques where The architecture of our model is:
      - Base model : the mobilenet model and discards the last 1000 neuron layer.
      - Head model : 3dense layers so that the model can learn more complex functions and classify for better results.
      - Final layer : final layer with softmax activation to multi-classes This way is not scale with project to can not predict classes in direct .

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, None, None, 3)]	0
conv1 (Conv2D)	(None, None, None, 32)	864
conv1_bn (BatchNormalization	(None, None, None, 32)	128
conv1_relu (ReLU)	(None, None, None, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, None, None, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, None, None, 32)	128
conv_dw_1_relu (ReLU)	(None, None, None, 32)	0
conv_pw_1 (Conv2D)	(None, None, None, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, None, None, 64)	256
conv_pw_1_relu (ReLU)	(None, None, None, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, None, None, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, None, None, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, None, None, 64)	256
conv_dw_2_relu (ReLU)	(None, None, None, 64)	0
conv_pw_2 (Conv2D)	(None, None, None, 128)	8192
conv_pw_2_bn (BatchNormaliza	(None, None, None, 128)	512
conv_pw_2_relu (ReLU)	(None, None, None, 128)	0
conv_dw_3 (DepthwiseConv2D)	(None, None, None, 128)	1152
conv_dw_3_bn (BatchNormaliza	(None, None, None, 128)	512
conv_dw_3_relu (ReLU)	(None, None, None, 128)	0
conv_pw_3_relu (ReLU)	(None, None, None, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, None, None, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, None, None, 128)	1152
conv_dw_4_bn (BatchNormaliza	(None, None, None, 128)	512
conv_dw_4_relu (ReLU)	(None, None, None, 128)	0
conv_pw_4 (Conv2D)	(None, None, None, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_pw_4_relu (ReLU)	(None, None, None, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, None, None, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_dw_5_relu (ReLU)	(None, None, None, 256)	0
conv_pw_5 (Conv2D)	(None, None, None, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, None, None, 256)	1024

conv_pw_5_relu (ReLU)	(None, None, None, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, None, None, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, None, None, 256)	2304
conv_dw_6_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_dw_6_relu (ReLU)	(None, None, None, 256)	0
conv_pw_6 (Conv2D)	(None, None, None, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_6_relu (ReLU)	(None, None, None, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_7_relu (ReLU)	(None, None, None, 512)	0
conv_pw_7 (Conv2D)	(None, None, None, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_7_relu (ReLU)	(None, None, None, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_8_relu (ReLU)	(None, None, None, 512)	0
conv_pw_8 (Conv2D)	(None, None, None, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_8_relu (ReLU)	(None, None, None, 512)	0
conv_pw_8 (Conv2D)	(None, None, None, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_8_relu (ReLU)	(None, None, None, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_9_relu (ReLU)	(None, None, None, 512)	0
conv_pw_9 (Conv2D)	(None, None, None, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_9_relu (ReLU)	(None, None, None, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_dw_10_relu (ReLU)	(None, None, None, 512)	0
conv_pw_10 (Conv2D)	(None, None, None, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_pw_10_relu (ReLU)	(None, None, None, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_dw_11_relu (ReLU)	(None, None, None, 512)	0
conv_pw_11 (Conv2D)	(None, None, None, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_pw_11_relu (ReLU)	(None, None, None, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, None, None, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_dw_12_relu (ReLU)	(None, None, None, 512)	0
conv_pw_12 (Conv2D)	(None, None, None, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, None, None, 1024)	4096
conv_pw_12_relu (ReLU)	(None, None, None, 1024)	0

```
conv_dw_13 (DepthwiseConv2D) (None, None, None, 1024)  9216
conv_dw_13_bn (BatchNormaliz (None, None, None, 1024)  4096
conv_dw_13_relu (ReLU)       (None, None, None, 1024)  0
conv_pw_13 (Conv2D)          (None, None, None, 1024)  1048576
conv_pw_13_bn (BatchNormaliz (None, None, None, 1024)  4096
conv_pw_13_relu (ReLU)       (None, None, None, 1024)  0
global_average_pooling2d (Gl (None, 1024)              0
dense (Dense)                (None, 1024)              1049600
dense_1 (Dense)              (None, 1024)              1049600
dense_2 (Dense)              (None, 512)               524800
dense_3 (Dense)              (None, 43)                22059
=================================================================
Total params: 5,874,923
Trainable params: 2,646,059
Non-trainable params: 3,228,864
_________________________________________________________________
```

Figure 3-2: Architecture of mobileNet model

- **Using mobileNet with data augmentation keras** : Using sequential model way to create model in Keras package by transfer learning techniques where The architecture of our model is:

    - Base model : pre-trained model is the mobilenet model .
    - GlobalAveragePooling2D layer
    - 3 dense layers and final layer with softmax activation
      This way is scale with project but reduce less than accuracy

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Function (None, None, None, 1024)  3228864
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              1049600
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 43)                22059
=================================================================
Total params: 4,825,323
Trainable params: 1,596,459
Non-trainable params: 3,228,864
_________________________________________________________________
```

Figure 3-3: Architecture of mobileNet with data augmentation model

- Training model
  - After building the model architecture, we then train the model , we tried with batch size 32 and 64. Our model performed better with 64 batch size. And after 5 epochs the accuracy was stable.
- Save model
  - we need save model to deploy it on hardware
- Draw graph of training and validation Accuracy
  - To ensure there are not overfit
- Test model
- Measure of performance

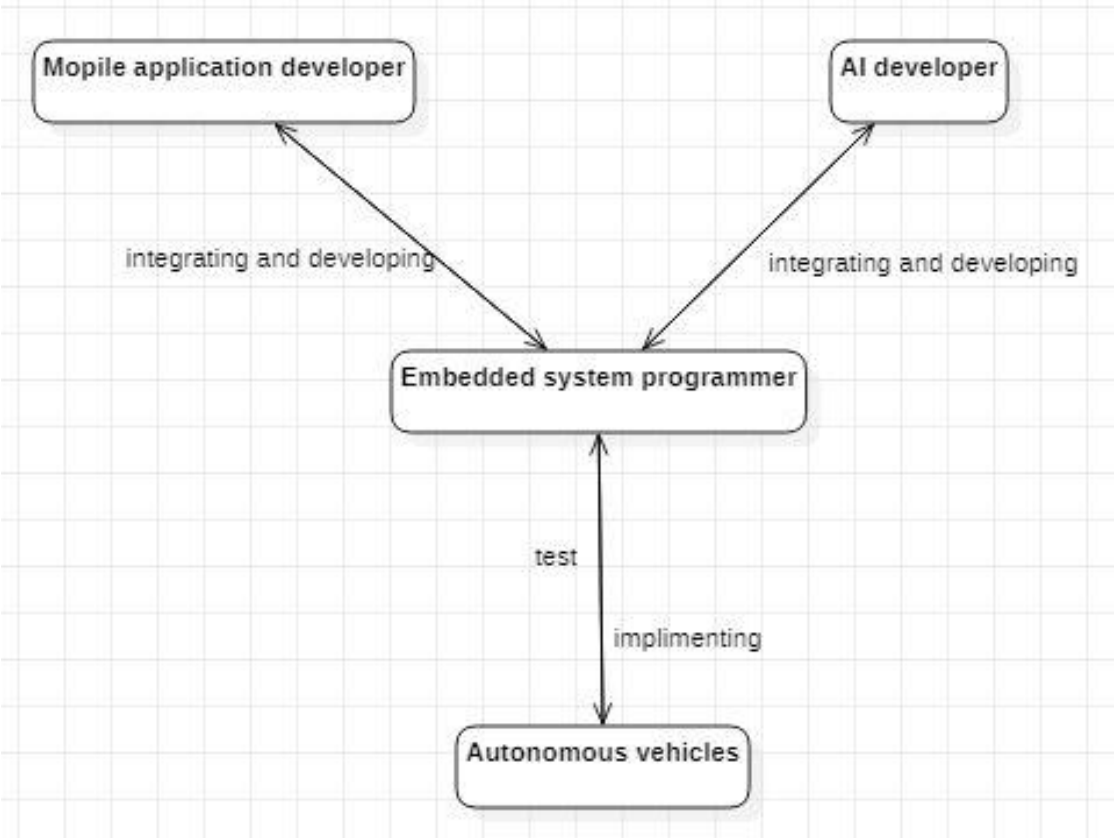
We need to performance of model , so we a lot of way to measure performance .

Embedded Systems

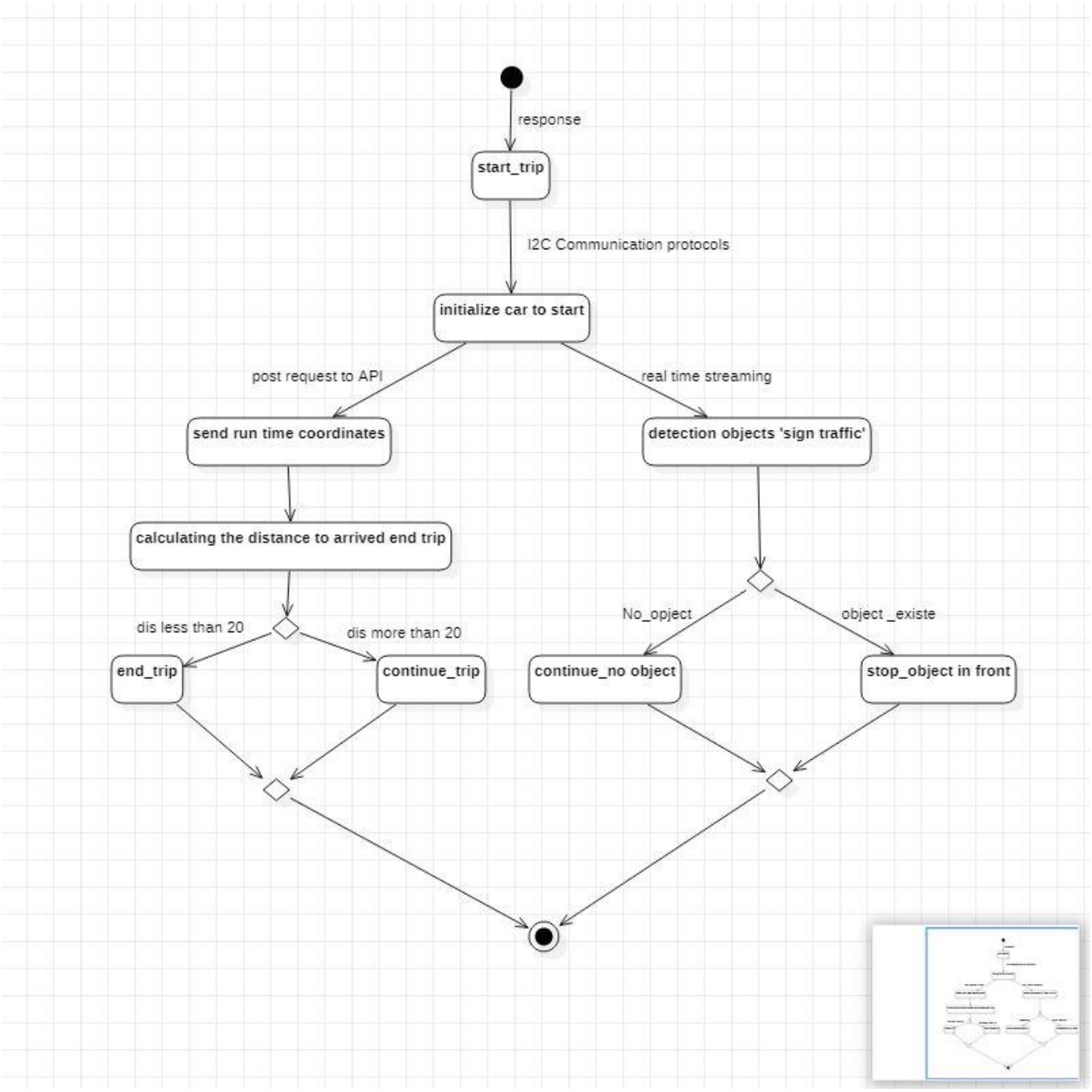
- Data gathering
  - gathering requirements
    - In this step the Embedded Systems Programmer start collecting all data and requirements from Ai developer and mobile app in team even can starting to implementation step .
- Data processing
  - Analysisdata
    - In this step the Embedded Systems Programmer start analysis data and creating design “software architecture”.
- Implementation
    - In this step the Embedded Systems Programmer starting to implementing the design “software architecture” , and I am ready for any modifying in requirements or design .

- Testing

- In This stepthe Embedded Systems Programmer start testing the autonomous vehicle with the Ai developer and mobile app developer, ifthey need to put new feature the embedded programmer is ready.



- Diagram of operational testing ‘Activitydiagram ’


response
start_trip
I2C Communication protocols
initialize car to start
post request to API
real time streaming
send run time coordinates
detection objects 'sign traffic'
calculating the distance to arrived end trip
dis less than 20
dis more than 20
No_opject
object _existe
end_trip
continue_trip
continue_no object
stop_object in front

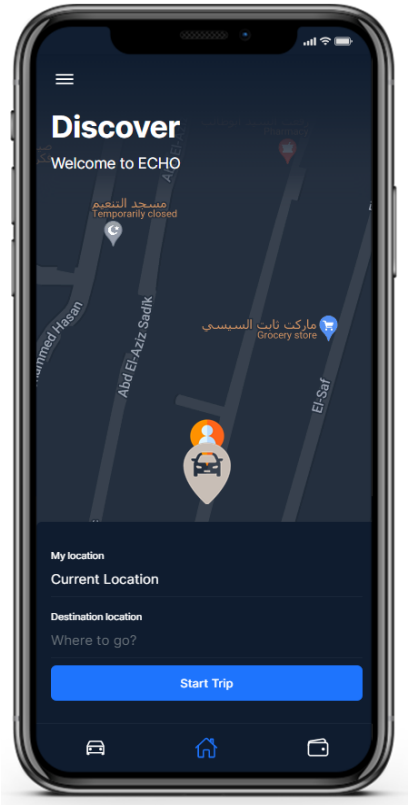
Mobile Application

Planning: The system is simply Mobile Application that can get you a ride with self derived car in small Modern City.

Analysis : The application Targets Local Citizens and Visitors to enjoy the ride to whatever they want in the small Modern City like school/grocery stores/hospitals, and provide the Flexibility to use it everywhere in the city.

The application provides a riding service, when you open the app for the first time

- **LOGIN SCREEN**
  - Here the user can register or go to Main screen (If they are already registered) using E-Mail and gets verification code
- **Verification screen**
  - Contains one field that you can put the Verification code that you got from E-Mail and single button (Verify) to click
- **Personal data screen**
  - Filling your information like your name in this screen
- **Main Screen**
  - A map with a card design on the bottom to search the place you want to go, Below this box are the suggestions places depends on what you typed, When you press **Start Trip** or click on one of the suggestions a **Pin** appears on the Map having **Your Destination Point** and **other pin** is going to be Your **Pickup Point**
  - By Clicking on **Your Destination Point** you are free to change **Your Destination pin**
  - **Confirm Your Ride** card shows the details of your ride and the price And your balance on the wallet , Having one button
    - **Confirm Pickup**
      - You confirm that you are in the car to start the ride

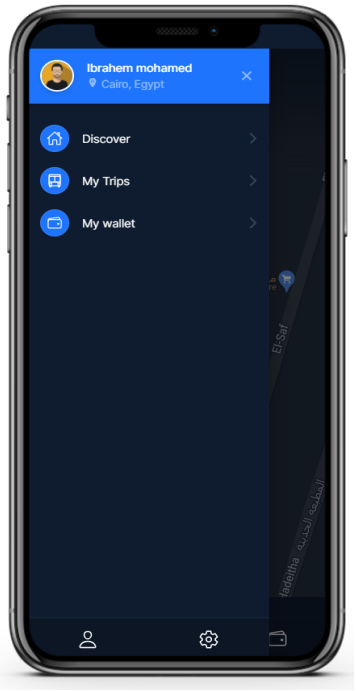


- **My Wallet**
  - **Contains** Your balance 30 EGP as example
  - **Add Funds** Button takes you to **Add Funds Screen**

- **Add Funds Screen**
  - **Add** your **visa Details** on the fields
  - **Add Funds** Button to confirm your visa data and complete the transaction and then fall back to **My Wallet Screen**

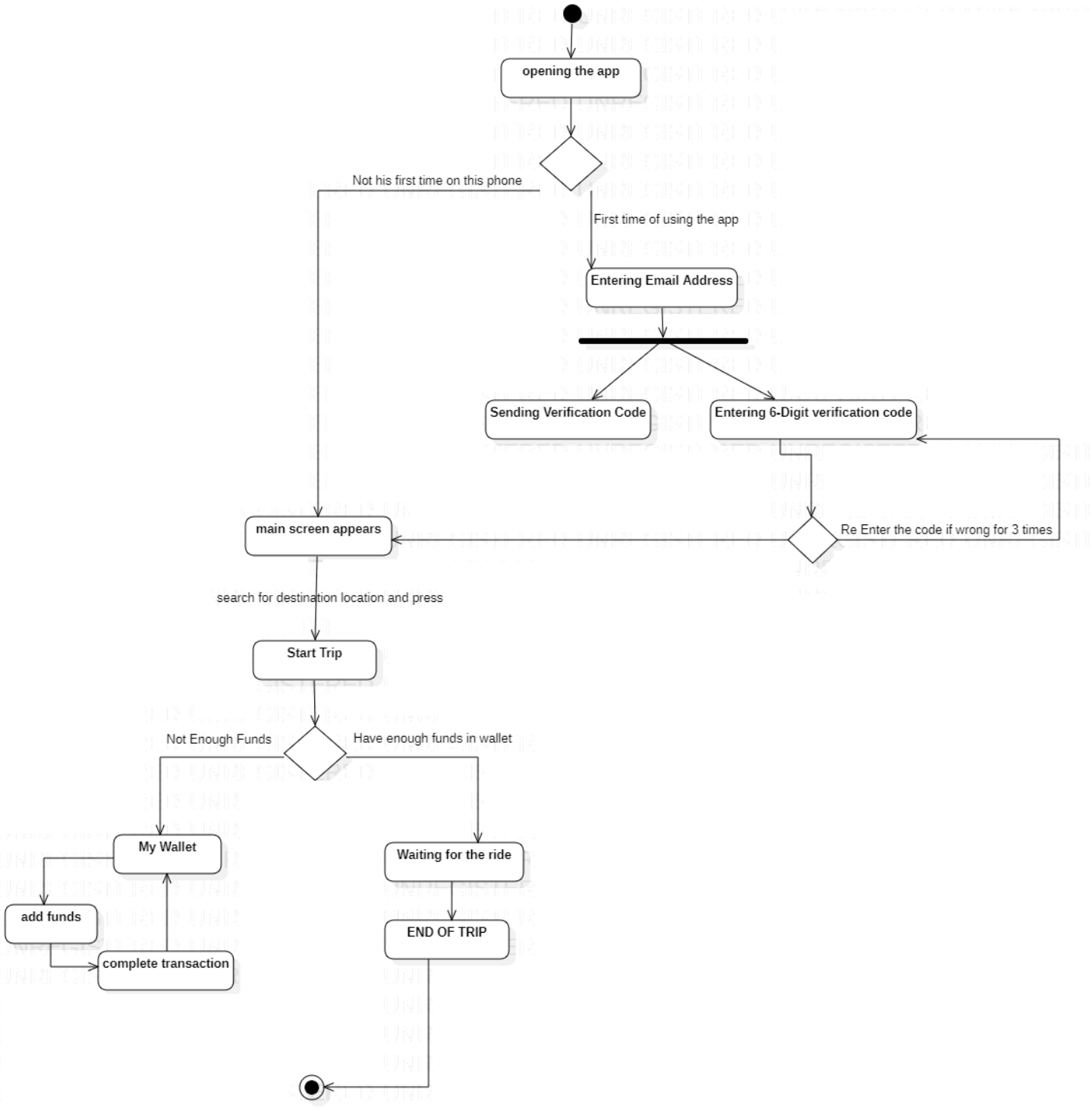
BONUS Content

- **Side Bar**
  - View Your **Profile**& your **History** of trips.
  - **My Wallet** button – takes you to **My Wallet Screen**
  - **Discover** Button – takes you to **main screen**
  - **Settings** button –takes you to **App Settings**



Design Phase

Implementing the analysis into a UML Activity Diagram



Experimental

Upload data

- Import all packages
- upload data

```
data = []
labels = []
classes = 43
cur_path = "/content/"


#Retrieving the images and their labels
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)

    for a in images:
        try:
            image = Image.open(path + '/'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")
```

Figure 4-1: upload data

Data preprocessing

We have stored all the images and their labels into lists (data and labels).

We need to convert the list into numpy arrays for feeding to the model.

The shape of data is (39209, 30, 30, 3) which means that there are 39,209 images of size 30×30 pixels and the last 3 means the data contains colored images (RGB value).

```
#Converting lists into numpy arrays
data = np.asarray(data).astype('float32')
labels = np.array(labels).astype('float32')

print(data.shape, labels.shape)
```

Split training and test data

With the sklearn package, we use the train_test_split() method to split training and testing data.

```
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

Figure 4-2: Split training and test data

From the tensorflow.keras.utils package, we use to_categorical method to convert the labels present in y_train and t_test into one-hot encoding.

```
#Converting the labels into one hot encoding
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
```

Building model

To classify the images into their respective categories, we will build a CNN model (Convolutional Neural Network). CNN is best for image classification purposes.

```
#Building the model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

Figure 4-3: Building model

We can see The architecture of our model using methode summary().

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        2432
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 22, 22, 32)        25632
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 32)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 11, 11, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 9, 9, 64)          18496
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               147712
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 43)                11051
=================================================================
Total params: 242,251
Trainable params: 242,251
Non-trainable params: 0
_________________________________________________________________
```

We compile the model with Adam optimizer which performs well and loss is categorical_crossentropy because we have multiple classes to categorise

```
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Training model

After building the model architecture, we then train the model using model.fit(). I tried with batch size 32 and 64. Our model performed better with 64 batch size. And after 5 epochs the accuracy was stable.

```
epochs = 5
history = model.fit(X_train, y_train, batch_size=64, epochs=epochs, validation_data=(X_test, y_test))
model.save("my_model.h5")
```

```
Epoch 1/5
491/491 [==============================] - 144s 260ms/step - loss: 4.1697 - accuracy: 0.2743 - val_loss: 0.4773 - val_accuracy: 0.8906
Epoch 2/5
491/491 [==============================] - 128s 260ms/step - loss: 0.7407 - accuracy: 0.7895 - val_loss: 0.2054 - val_accuracy: 0.9455
Epoch 3/5
491/491 [==============================] - 126s 256ms/step - loss: 0.3885 - accuracy: 0.8900 - val_loss: 0.0912 - val_accuracy: 0.9784
Epoch 4/5
491/491 [==============================] - 125s 255ms/step - loss: 0.2756 - accuracy: 0.9223 - val_loss: 0.0822 - val_accuracy: 0.9778
Epoch 5/5
491/491 [==============================] - 125s 255ms/step - loss: 0.2734 - accuracy: 0.9267 - val_loss: 0.0694 - val_accuracy: 0.9811
```

Figure 4-4: Training model

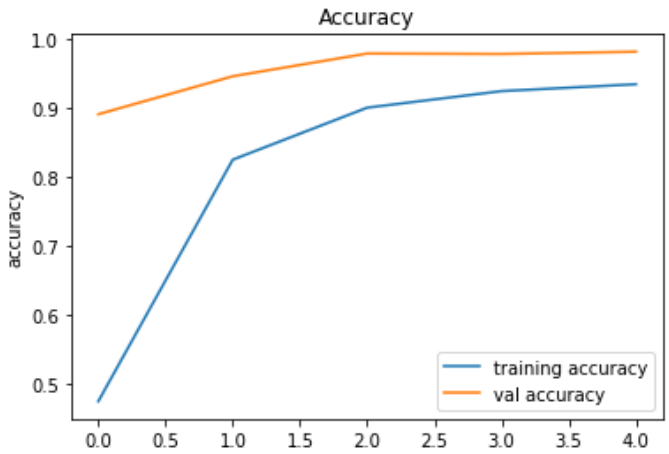
Save model

We saved model by save() method.

Draw graphs of training and validationAccuracy

Our model got a 98% accuracy on the training dataset.
With matplotlib, we plot the graph for accuracy and the loss.

```
#plotting graphs for accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```




```
plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

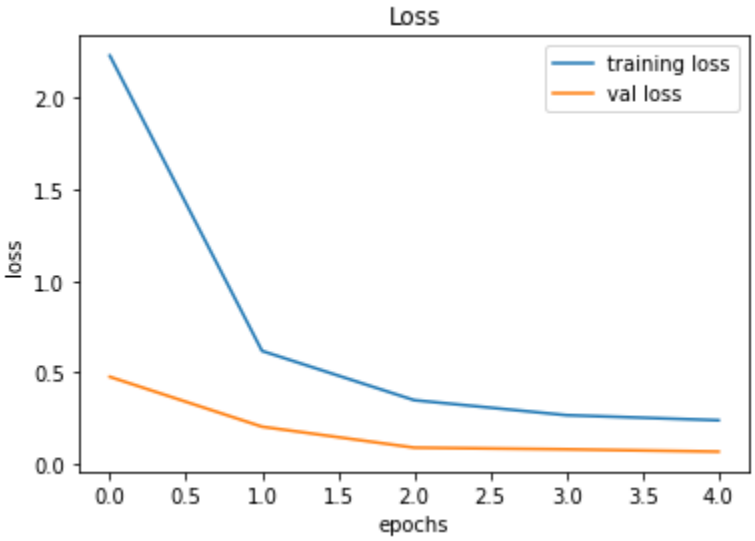


Figure4-5: Graphs of training and validation Accuracy

Test model

- Upload test data

```
#testing accuracy on test dataset
from sklearn.metrics import accuracy_score

y_test = pd.read_csv('/content/Test.csv')

labels = y_test["ClassId"].values
imgs = y_test["Path"].values

data=[]

for img in imgs:
    path = '/content/'
    image = Image.open(path+img)
    image = image.resize((30,30))
    data.append(np.array(image))

X_test=np.array(data)
```

Figure4-6: Upload test data

- Predict and measure accuracy

```
pred = model.predict_classes(X_test)

#Accuracy with the test data
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, pred))
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01.
  warnings.warn('`model.predict_classes()` is deprecated and '
0.9533650039588282
```

Figure4-7: Predict and measure accuracy

Measure of performance

- Using confusion matrix

```
#importing confusion matrix
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(labels, pred)
print('Confusion Matrix\n')
print(confusion)
```

```
Confusion Matrix

[[ 57   2   0 ...   0   0   0]
 [  0 701   2 ...   0   0   0]
 [  0   4 742 ...   0   0   0]
 ...
 [  0   0   0 ...  80   0   0]
 [  0   0   0 ...   0  47   0]
 [  0   0   0 ...   0   0  86]]
```

- With seaborn package , we can draw heatmap for confusion Matrix.

```
import seaborn as sn
df_cm = pd.DataFrame(confusion, columns=np.unique(labels), index = np.unique(labels))
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
plt.figure(figsize = (20,20))
sn.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16})# font size
```

Figure4-8:Drawconfusion matrix

- Classification report

```
from sklearn.metrics import classification_report
ClassificationReport = classification_report(labels, pred)
print('Classification Report is : \n', ClassificationReport )
```

Figure4-9: Draw Classification report

- Another ways for measurement

```
#importing accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
print('\nAccuracy: {:.2f}\n'.format(accuracy_score(labels, pred)))

print('Micro Precision: {:.2f}'.format(precision_score(labels, pred, average='micro')))
print('Micro Recall: {:.2f}'.format(recall_score(labels, pred, average='micro')))
print('Micro F1-score: {:.2f}\n'.format(f1_score(labels, pred, average='micro')))

print('Macro Precision: {:.2f}'.format(precision_score(labels, pred, average='macro')))
print('Macro Recall: {:.2f}'.format(recall_score(labels, pred, average='macro')))
print('Macro F1-score: {:.2f}\n'.format(f1_score(labels, pred, average='macro')))

print('Weighted Precision: {:.2f}'.format(precision_score(labels, pred, average='weighted')))
print('Weighted Recall: {:.2f}'.format(recall_score(labels, pred, average='weighted')))
print('Weighted F1-score: {:.2f}'.format(f1_score(labels, pred, average='weighted')))
```

Figure4-10: Another ways of measurement

Embedded Systems Experimental

- **Initialize GPS module**

  - In the first, I import all library need to connect with the satellites and get the coordinates ‘ latitude ’ and ‘ longitude ’ of the reallocation of car.
  - The first lib I need it’s (serial) then (pynmea2), (time)
  - Then I need to send my location to the Mobile application within API this is mean that I need to import more lib. I import (requests) and (JSON)
  - Now we ready to write our code to get our coordinates or our location using GPS module,
  - In the __ fig. num__ you can see all steps to get ‘lat’ and ‘lng’ using python programing language.
  - After get the coordinates , I will start send It to the mobile application even start tracking the car in Map.
  - To send my coordinate to API , I use Post request to send my data because it is more secure .

Figure4-11: response start_trip from API

```
port = '/dev/ttyAMA0'
ser = serial.Serial(port, baudrate=9600, timeout=0.5)
dataout = pynmea2.NMEAStreamReader()
newdata = ser.readline()

if newdata[0:6] == '$GPRMC':
    newmsg = pynmea2.parse(newdata)
    lat = newmsg.latitude
    lng = newmsg.longitude
    Serial=getserial()
    gps = 'Latitude=' + str(lat) + 'and Longitude=' + str(lng)
    #print (gps)
    API_ENDPOINT = 'https://egyptian-idco.com/ECHO/API/json.php'
    data = {'lat': str(round(lat, 9)),
            'lon': str(round(lng, 9)),
            'sn' : str(Serial)}
    r = requests.post(url=API_ENDPOINT, data=data)
```

- **Check Response**

  - This this step I will check the API parameter even knowing if any customer make order to start trip or no.

- To write code by ‘PYTHON’ and response data from API written by ‘JSON’ must get json data and convert it to text then load the text data into python.
- You can follow the __fig. num__ to see how to write the code.

- **Response Start trip**

```
trip_url = 'https://egyptian-idco.com/ECHO/API/trip.php'
false_data = {}
trip_res = requests.post(url=trip_url, data=false_data)

json_CallBack = trip_res.text
json_encode =json.loads(json_CallBack)
f_lat =json_encode['final_location']['lat']
f_lon =json_encode['final_location']['lon']
start_trip = json_encode['response']
```

Figure4-12: check is the response return 1

- If the response return start trip this is mean the customer order car and the nearest car from the customer will start trip to go to him location.
- Then we will send character to the Arduino mega from raspberry pi using I2C communication protocol even the car start moving because the Arduino responsible to control every thing in car that allow car moving.

- In the __fig. num__ you can see how to write code even send sample or character from raspberry pi to Arduino through I2C bus .

```
if(start_trip == "starttrip"):
    #bus.write_byte(addr,'S')
    print ("start")
```

Figure4-13: action Start_trip in arduino

- And this is code to exciting the action on car from Arduino mega.

```
while ( Wire.available()) {
  char c = Wire.read();
  Serial.print(c);
  switch(c){
    case'S':
         int first = null;
         if(first == null){
          frontled_ON();
          backled_ON();
          delay(500);
          backled_OFF();
          first = 1;
          }
          if(first == 1){
            forward(150);
            }
         }
         break;
```

Figure4-14: action Start_trip in arduino

- **Calculating the distance**
  - In this step we using trigonometry to calculating the difference between the current(start) location and final location that it determined before start trip .
  - And to calculate this I need to import from math library (sin ,cos , sqrt, atan2 , radians)
  - And to see how to write this code you can look __fig. num__

```
def Get_Distance(f_lat, f_lon ,s_lat,s_lon):
# approximate radius of earth in km


  R = 6373.0
  distance =0.0000000000
  lat1 = radians(s_lat)
  lon1 = radians(s_lon)
  lat2 = radians(f_lat)
  lon2 = radians(f_lon)

  dlon = lon2 - lon1
  dlat = lat2 - lat1

  a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
  c = 2 * atan2(sqrt(a), sqrt(1 - a))

  distance = (R * c)*float(1000)
  return distance
```

Figure4-14: creating function to calculate the Distance

- **Check distance**
  - Here we need to check the distance when the car moving even know the car arrived or stay in the trip.
  - And to do this must calling Get_Distance() function to return the dis with meter unite .
  - And to see how this is write look __fig. num__ .

```
D= ( Get_Distance(f_lat,f_lon,lat,lng))
```

- **Decision distance**

  - Here we manipulating the return data from Get_Distance() to put the rules or decisions like.
  - If the different between to location less than 20 meter in this case the trip is Ended to see how to write the code look the __fig. num__.

```
if (D)< float(20.0):
    bus.write_byte(addr,'F')
    print ("stop")
    trip_url = 'https://egyptian-idco.com/ECHO/API/trip.php'
    new_data = {"response": "End_Trip"}
    trip_res = requests.post(url=trip_url, data=new_data)
```

Figure4-15: if distance less than 20 meter.

  - And if this condition true the action in Arduino will be like this .

```
case'T':
    frontled_ON();
    frontled_ON();
    Stop();
    break;
```

Figure4-16: action of arduino when the return distance less than 20 meter

  - If the different between to location more than 20 meter.

```
else :
    bus.write_byte(addr,'C')
    print ("continue")
```

Figure4-17: if distance more than 20 meter

- If this conditions true the action in Arduino will be .

```
case'C':
      frontled_ON();
      backled_OFF();
      forward(150);
      break;
```

Figure4-18: action of Arduino when the return distance more than 20 meter

- **Video Capture**
  - In this step we will stream the video from mobile camera using python programming language and I need to import cv2(openCV) library to do it like .
  - And we use the cap.read() function to return frames from video streaming .
  - The I use this frames to send it to the classify() function even allow the model of machine learning take this frames and predict what is it using classes and return the name of object in the frame .
  - Then I am using the result to make manipulating or do make the car take specific action Like STOP car or No fast …etc
  - And to see how to write this code look __fig. num__.

```
cap = cv2.VideoCapture("http://192.168.1.6:8080/video")
while(1):

    # Capture frame-by-frame
    ret, frame = cap.read()
    # Display the resulting frame
    cv2.imshow('frame',frame)
    print(frame.shape)
    sign = classify(frame)

    if sign == 'Stop' :
        bus.write_byte(addr,'ST')
```

Figure4-19: send capture and detect sign traffic stop

- And if the Model predicted that the object is STOP sign traffic the action in Arduino will be like this.

```
case"ST":
     frontled_ON();
     frontled_ON();
     Stop();
     break;
```

Figure4-20: action of Arduino when the model predicts STOP sign traffic

Mobile Application Experimental

Now it's time to know how the mobile application works and how we design it and develop it to fit the customer needs.

Login screen

The Coding behind this screen is when we press Register we POST to the API 1 parameter It's the email address and the button Register not going to be active if we didn't check the **I agree terms & Conditions**

How we going to communicate with API using Flutter/Dart

First, we need to import the HTTP library

```
import 'package:http/http.dart' as http;
```

And then we need to call the function that we already made to communicate with the API

```
Future<http.Response> Register_User(String Emailaddress) {
  return http.post(
    Uri.parse('http://LOCALHOST/ECHO/API/register.php'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, String>{
      'Email': Emailaddress,
    }),
  );
}
```

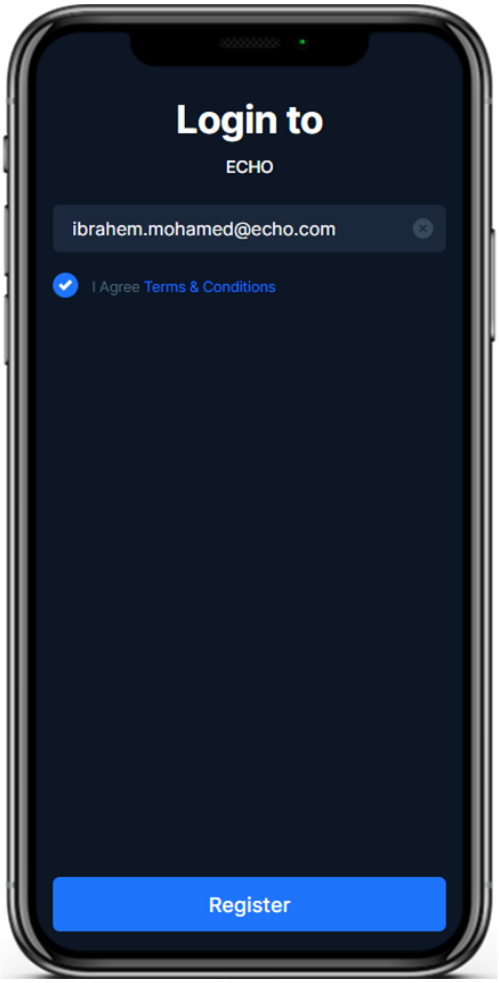


Figure4-21: Login Screen of ECHO App

So now we will call **API/Register.php** on the server and POST the email address to it

```php
<?
include "class.user.php"; // adding the class to this file
$user = new User(); // Making new object from user class
$Email = $_POST['Email']; // Detect the Email POSTing
$database_response = $user->Check_And_Register($Email);
$product_arr=new \stdClass();
// to store the response
if($database_response == 1){
    $security_code = $user->GetVerification($Email);
    $product_arr->response = "Success";
    $product_arr->verification_Code =  $security_code;
}else{
    $product_arr->response = "Fail";
}

echo json_encode($product_arr);
?>
```

Figure4-22: Register User API

Here it comes the API turn to check and register this user into the **MySQL** database

Detecting the Posted Email and pass it to the **Check_And_Register** Function as a parameter and this function will register the user if the email does not exist in the database and then insert it to the database, even if the email exists we will send a 6-digit verification code to verify his email address.

The result JSON code


```
{
    response: "Success",
    verification_Code: 145678
}
```


Figure4-23: Resultant JSON Code

User Table in the MySQL Database

	#	Name	Type	Collation	Attributes	Null	Default
☐	1	id 🔑	int(11)			No	*None*
☐	2	username	varchar(20)	utf8_general_ci		No	*None*
☐	3	email	varchar(100)	utf8_general_ci		No	*None*
☐	4	current_lat	decimal(11,9)			No	*None*
☐	5	current_long	decimal(11,9)			No	*None*
☐	6	activation_code	int(6)			No	*None*

Figure4-24: user table Mysql

Moving on to the next widget

Verification Screen

On this page, we Registered/Logged the user and the server sent the verification code to the user's Email Address

So the user inserts his v-code into this field and if the v-code inserted is equals to the one on the database so we confirm that this email is yours

How are we going to know if this is the right v-code or not?

From the JSON Response we have

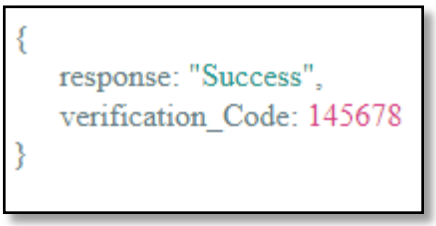


Figure4-27: API's Response of registered user

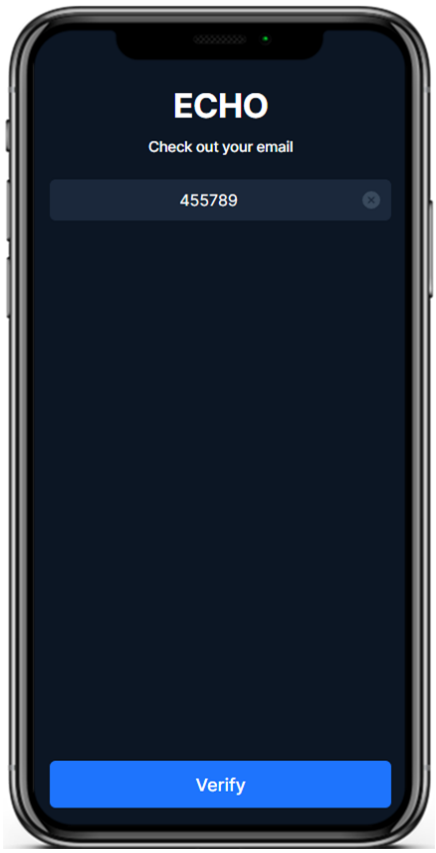


Figure4-25: Verification Screen of ECHO App

So we will match this v-code to the detected one while we register.

Then we continue filling some personal data like the location (Auto detected) and your **full Name**

When we press **Discover**

The app sends to the API Location and Full Name

```dart
Future<http.Response> Update_Name_Loc(String Name,String Location {
  return http.post(
    Uri.parse('http://LOCALHOST/ECHO/API/update_name.php'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, String>{
      'Name': Name,
      'Location': Location,
    }),
  );
}
```

Figure4-28: Dart Code that post data into API

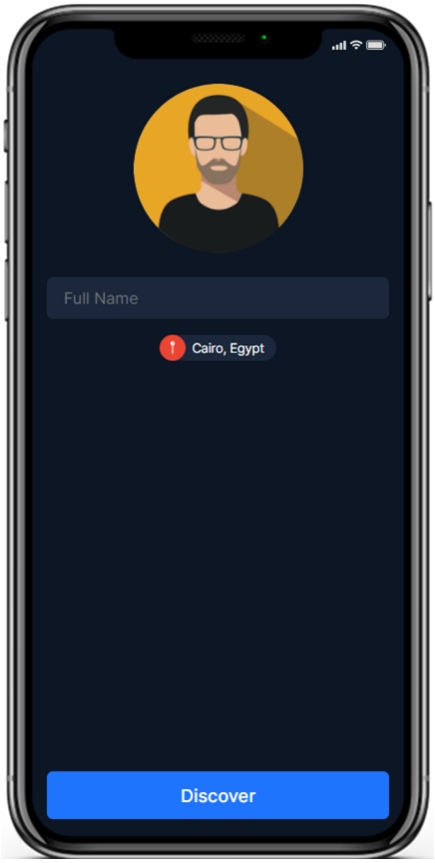


Figure4-26: filling user data Screen of ECHO App

```php
<?
include "class.user.php"; // adding the class to this file
$user = new User(); // Making new object from user class
$Name = $_POST['Name']; // Detect the Name post
$Location = $_POST['Location']; // Detect the Location post
$database_response = $user->Update_user_data($Name,$Location);
$product_arr=new \stdClass();
if($database_response == 1){
    $product_arr->response = "Success";
}else{
    $product_arr->response = "Fail";
}
echo json_encode($product_arr);
?>
```

Figure4-29: API code to update user's data in the database

This script updates user's data in the database to add the location and Username into users table.

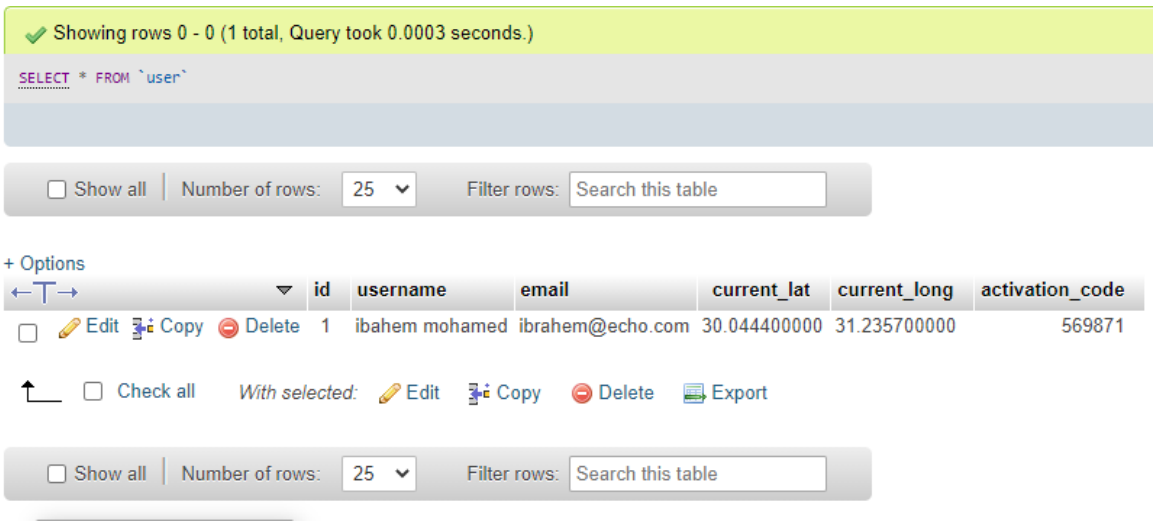


Figure4-30: Mysql Database User's table

Main Screen

This screen is so complex, we used so many libraries on this page, The most important library is the Google Maps API and how we used it.

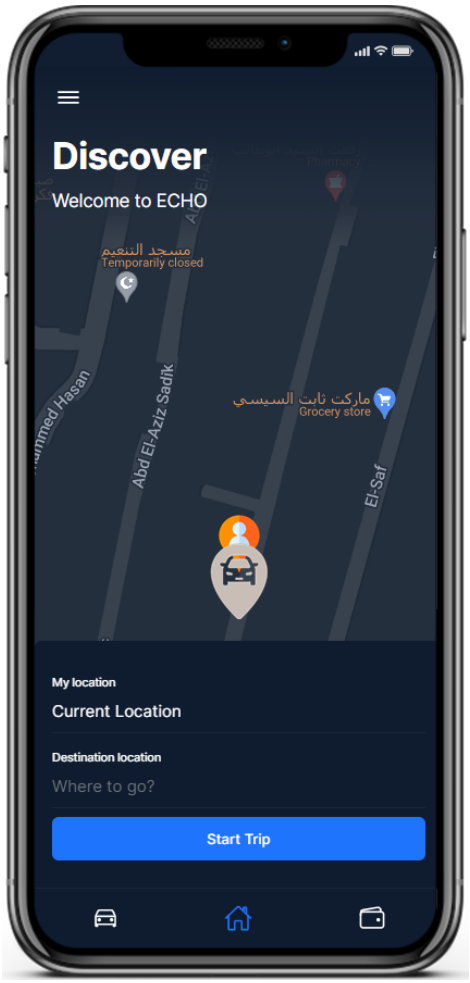


Figure4-31: Discover Screen of ECHO App

It's all about the API keys

To use Google Maps in your Flutter app, you need to configure an API project with the Google Maps Platform, following both the Maps SDK for Android's Get API key, and Maps SDK for iOS' Get API key processes. With API keys in hand.

Adding an API key for an iOS app

To add an API key to the iOS app, we edit the AppDelegate.swift file in iOS/Runner. Unlike Android, adding an API key on iOS requires changes to the source code of the Runner app. The AppDelegate is the core singleton that is part of the app initialization process.

```
import UIKit;
import Flutter;
import GoogleMaps;

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
  ) -> Bool {
    GeneratedPluginRegistrant.register(with: self)

    // TODO: Add your API key
    GMSServices.provideAPIKey("YOUR-API-KEY")

    return super.application(application, didFinishLaunchingWithOptions: launchOptions)
  }
}
```

Figure4-32: Initialize Google Maps API

Putting a map on the screen

Now it's time to get a map on the screen. Update lib/main.dart as follows:

```dart
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

class _MyAppState extends State<MyApp> {
    late GoogleMapController mapController;

    final LatLng _center = const LatLng(45.521563, -122.677433);

    void _onMapCreated(GoogleMapController controller) {
        mapController = controller;
    }
    @override
Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(
            title: Text('ECHO'),
            backgroundColor: Colors.Blue[700],
            ),
            body: GoogleMap(
            onMapCreated: _onMapCreated,
            initialCameraPosition: CameraPosition(
                target: _center,
                zoom: 11.0,
            ),
            ),
        ),
        );
}
```

Figure4-33: Add the map in the main screen Widget

So now we have the Google maps on our app we need now to put some pins on the map to show my location and the cars close to me etc with this function.

```dart
void _add_pin_to_map(Float lat,Float long ,String icon) {
    var markerIdVal = MyWayToGenerateId();
    final MarkerId markerId = MarkerId(markerIdVal);

    // creating a new MARKER
    final Marker marker = Marker(
      markerId: markerId,
      position: LatLng(
        center.latitude + sin(_markerIdCounter * pi / 6.0) / 20.0,
        center.longitude + cos(_markerIdCounter * pi / 6.0) / 20.0,
      ),
      infoWindow: InfoWindow(title: markerIdVal, snippet: '*'),
      onTap: () {
        _onMarkerTapped(markerId);
      },
    );

    setState(() {
      // adding a new marker to map
      markers[markerId] = marker;
    });
}
```

Figure4-34: Add pin to the map

So now we finished the most important parts we need now to call our API to track the near cars and my location and like that

So let's start with the API code and what is the algorithm

First, each car should update its location continuously so we made this API code to keep the cars up to date

```php
<?
include "class.car.php";
$car = new Car();
$car->update_location($_POST['lat'],$_POST['lon'],$_POST['sn']);
$product_arr=new \stdClass();
$product_arr->response = "Location Updated";
$product_arr->data = array("lat"=>$_POST['lat'],"lon"=>$_POST['lon'],"sn"=>$_POST['sn']);
echo json_encode($product_arr);
?>
```

Figure4-35: API Updating cars location

From Car Class, we have some methods one of them is the Update_Location()

This function accepts 3 parameters which is Latitude, Longitude, Serial Number

Each car in our system has a unique Serial Number in its hardware so we use it as a key for the car table in the database

And we keep updating Latitude and Longitude on the row each time the car calls the API

This figure shows the car table in MySQL Database..

#	Name	Type	Collation	Attributes	Null	Default
1	id	varchar(20)	utf8_general_ci		No	*None*
2	lat	decimal(11,9)			No	*None*
3	long	decimal(11,9)			No	*None*
4	status	enum('0', '1', '2', '3')	utf8_general_ci		No	*None*

Figure4-36: Car's Table MySQL database

So now we know how the cars update its location continuously, now it's time to know how we keep tracking the near cars

Tracking nearby cars

We have in car class Track_Near() function and this function accepts 2 parameters Longitude and latitude, then we calculate and select all the cars in certain diameter depend on these long and lat we feed into the function and return JSON array containing all car details, then we use mark update function on the app if the mark of a certain car already exists and if it doesn't exist we create a new marker with the function above.

```
<?
include "class.car.php";
$car = new Car();
$carlist = $car->Track_Near($_POST['lat'],$_POST['long']);
$product_arr=new \stdClass();
if (empty($carlist)) {
    $product_arr->response = "No Cars Behind";

}else{
    $product_arr->response = count($carlist)."Cars Detected";
    $product_arr->cars = $carlist;
}

echo json_encode($product_arr);

?>
```

Figure4-37: Tracking nearby cars API code

And the JSON Code from this function is

```
{
    response: "1 Cars Detected",
  - cars: [
    - {
            SN: "100000004153dc4c",
            Lat: 30.092730833,
            long: 31.205405333
        }
]
}
```

Figure4-38: Tracking nearby cars API Response

And this is how we Track/Update cars location continuously.

Start Trip

Moving around on our complex widget Main Screen into a Start Trip Part

As we see in this figure we have a card widget on the bottom of the page, consist of 2 inputs and one button to start the trip

When we change the value of any input we get autosuggestion of the locations on the map and if the user clicks any of these suggestions it takes him to the location and creates a pin at this place and this pin is free-to-move which you can change the road or something like that

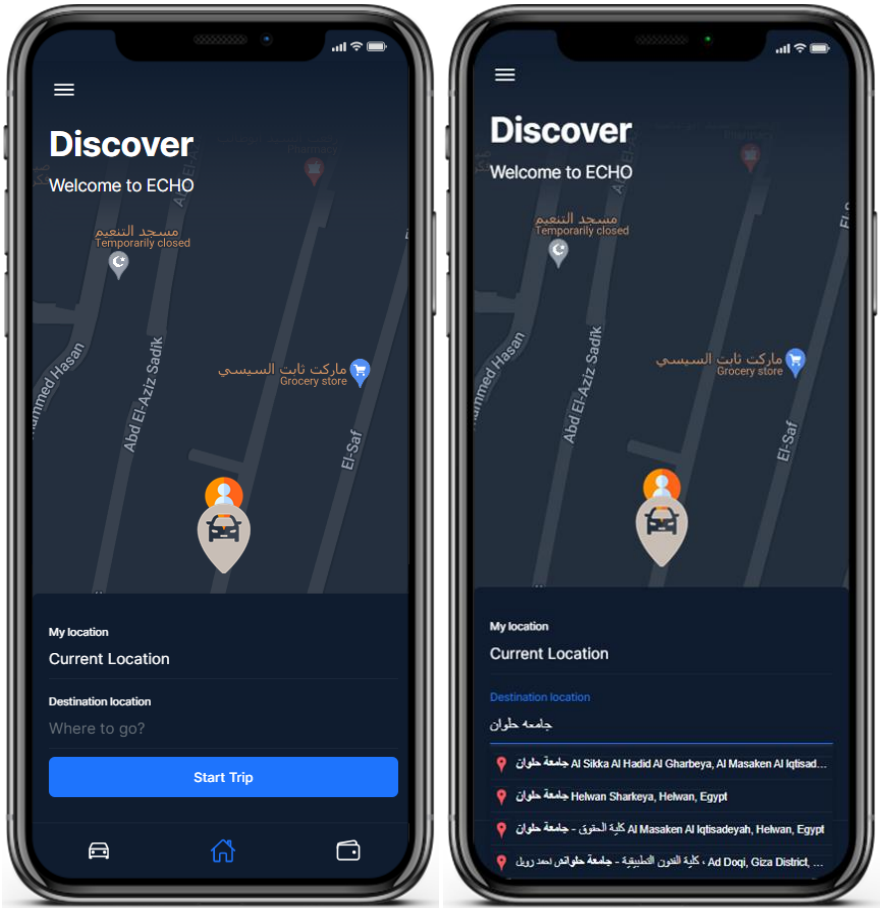


Figure4-39: Auto Suggestion on Discover Screen of ECHO App

But the science behind this part is pretty simple; the autosuggestion part is from Google maps API so I already talked about it enough so let's move to what will happen if the user clicked Start Trip.

Trip Table (MySQL)

At this table we insert the trip information so when we press start trip button we call the API to insert trip data

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**id**	int(11)			No	*None*		AUTO_INCREMENT
2	**car-sn**	varchar(50)	utf8_general_ci		No	*None*		
3	**user_id**	int(11)			No	*None*		
4	**pickup-lat**	decimal(11,9)			No	*None*		
5	**pickup-long**	decimal(11,9)			No	*None*		
6	**final-lat**	decimal(11,9)			No	*None*		
7	**final-long**	decimal(11,9)			No	*None*		
8	**status**	enum('0', '1')	utf8_general_ci		No	*None*	0=> Ongoing Trip 1=> Finished Trip	

Figure4-40: Trip Table Database

“id “belongs to the trip each trip has a unique id,

“car-sn” is a foreign key that aims for the serial number (id) in the car table

“user_id” is a foreign key that aims for the id in the user table

“pickup-lat” contains the latitude of the pickup location

“pickup-long” contains the longitude of the pickup location

“final-lat” contains the latitude of the final location

“final-long” contains the longitude of the final location

“Status” contains the state of the trip, ongoing or finished.

So we may see the relations of the 3 tables user, trip, car

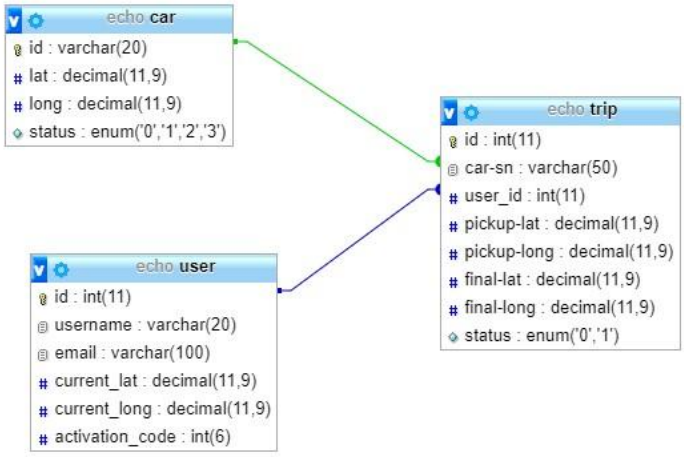


Figure4-41: Database Design

So by clicking start trip we call the API to insert into the database trip details

```php
<?
include "class.car.php";
$car = new Car();
$trip_response = $car->Start_Trip($_POST['Car_SN'],$_POST['p_lat'],$_POST['p_lon'],$_POST['f_lat'],$_POST['f_lon'],$_POST['u_id']);
$product_arr=new \stdClass();
if ($trip_response){
    $product_arr->response = "Trip Started";
}else{
    $product_arr->response = "No Cars Available";
}

echo json_encode($product_arr);
?>
```

Side Bar

We don't have much to say about side bars but we have in this side bar 3 parts.

Header widget having the name of the user and his location

List view widget having 3 items which is

Discover

Takes the user to the main screen

My trips

Takes the user to profile screen

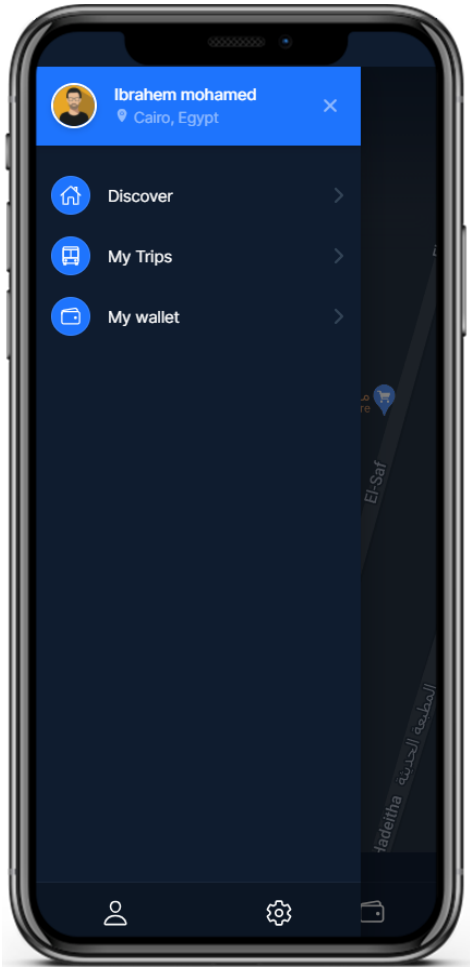


Figure4-42: Side bar Screen of ECHO App

My Wallet

Takes the user to my wallet screen

Bottom widget has 2 items

Profile

Takes the user to profile screen

Setting (underdevelopment)

Takes the user to Setting screen

Profile Screen

Contains user data, trips number and total hours traveled with this application and his funds on the wallet, then it contains a list of his trips

When the user clicks on any trip it shows the trip summary

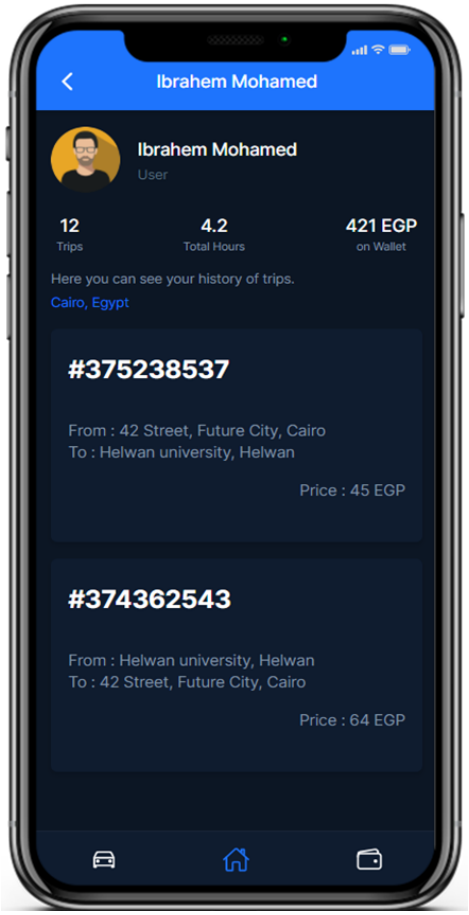


Figure4-43: Profile Screen of ECHO App

Conclusion

Finally, we can smart system to recognize sign traffic in self-driving system using computer vision techniques.

Architecture of our model:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        2432
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 22, 22, 32)        25632
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 32)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 11, 11, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 9, 9, 64)          18496
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               147712
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 43)                11051
=================================================================
Total params: 242,251
Trainable params: 242,251
Non-trainable params: 0
_________________________________________________________________
```

Figure5-1: Architecture of model

Confusion matrix of model:

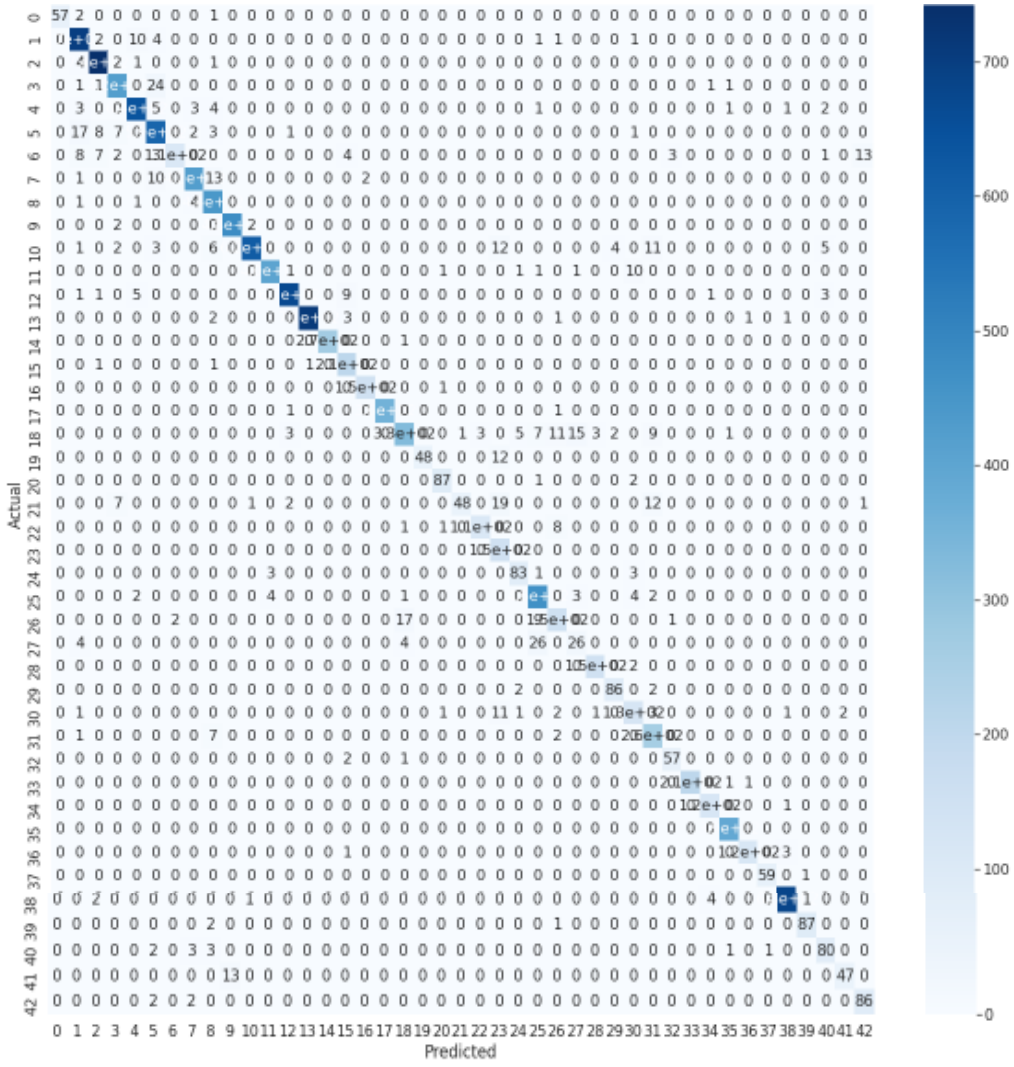


Figure5-2: Confusion matrix of model

Classification report of our model :

Classification Report is :

	precision	recall	f1-score	support
0	1.00	0.95	0.97	60
1	0.94	0.97	0.96	720
2	0.97	0.99	0.98	750
3	0.95	0.94	0.94	450
4	0.97	0.97	0.97	660
5	0.92	0.94	0.93	630
6	0.98	0.73	0.84	150
7	0.97	0.94	0.95	450
8	0.91	0.99	0.95	450
9	0.97	0.99	0.98	480
10	0.99	0.93	0.96	660
11	0.98	0.96	0.97	420
12	0.99	0.97	0.98	690
13	1.00	0.99	0.99	720
14	1.00	1.00	1.00	270
15	0.92	0.99	0.95	210
16	0.99	0.99	0.99	150
17	1.00	0.99	1.00	360
18	0.93	0.85	0.89	390
19	1.00	0.80	0.89	60
20	0.96	0.97	0.96	90
21	0.98	0.53	0.69	90
22	0.97	0.92	0.94	120
23	0.74	1.00	0.85	150
24	0.90	0.92	0.91	90
25	0.91	0.97	0.94	480
26	0.85	0.84	0.84	180
27	0.58	0.43	0.50	60
28	0.97	0.99	0.98	150
29	0.93	0.96	0.95	90
30	0.85	0.85	0.85	150
31	0.87	0.96	0.91	270
32	0.93	0.95	0.94	60
33	1.00	0.99	1.00	210
34	0.95	0.99	0.97	120
35	0.99	1.00	0.99	390
36	0.98	0.97	0.97	120
37	0.98	0.98	0.98	60
38	0.99	0.99	0.99	690
39	0.98	0.97	0.97	90
40	0.88	0.89	0.88	90
41	0.96	0.78	0.86	60
42	0.86	0.96	0.91	90
accuracy			0.95	12630
macro avg	0.94	0.92	0.93	12630
weighted avg	0.95	0.95	0.95	12630

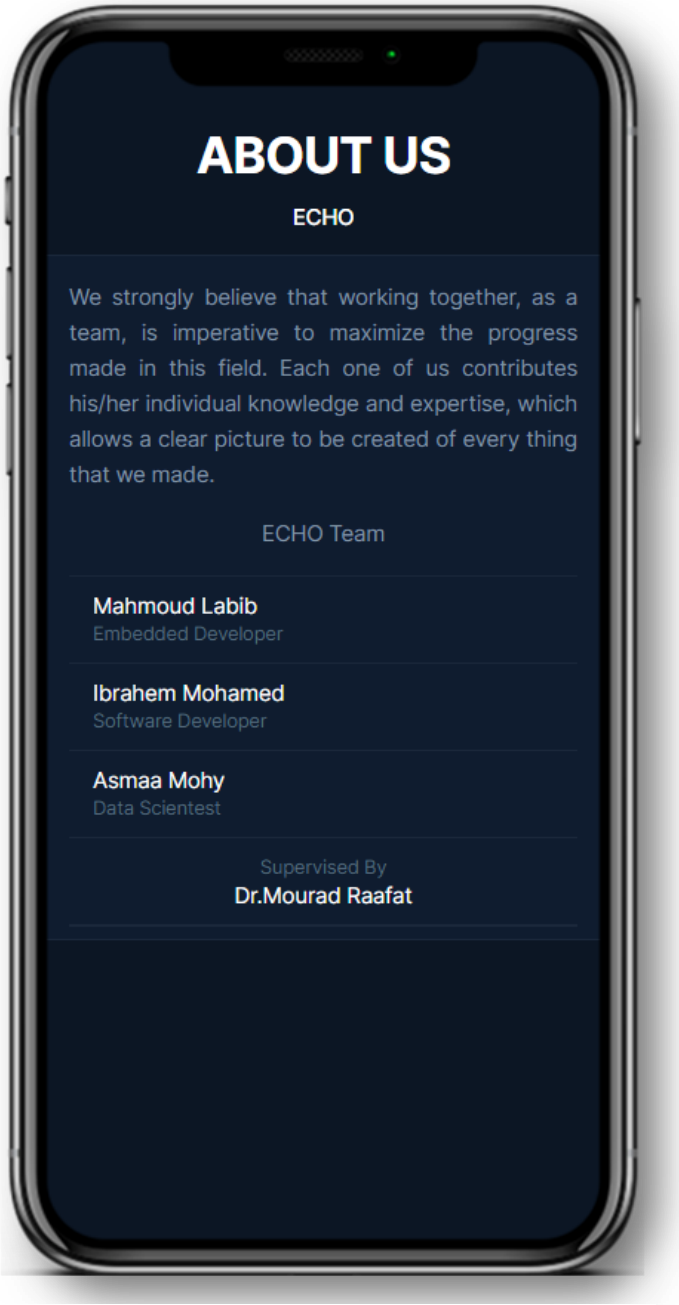
Figure5-3: Classification report of our model

Another ways for measurement:

```
Accuracy: 0.95

Micro Precision: 0.95
Micro Recall: 0.95
Micro F1-score: 0.95

Macro Precision: 0.94
Macro Recall: 0.92
Macro F1-score: 0.93

Weighted Precision: 0.95
Weighted Recall: 0.95
Weighted F1-score: 0.95
```

Figure5-4: Anther ways for measurement of model

Now we can control many autonomous vehicles and put them together in one system to form an interactive app that all people can use easily.



References

Sample correct formats for various types of references are as follows.
Websites:

1. GTSRB - German Traffic Sign Recognition Benchmark. (2021). Retrieved 11 July 2021, from https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?select=Meta.csv
2. GTSRB - German Traffic Sign Recognition Benchmark. (2021). Retrieved 11 July 2021, from https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?select=Meta.csv
3. The 6 Levels of Vehicle Autonomy Explained | Synopsys Automotive. (2021). Retrieved 11 July 2021, from https://www.synopsys.com/automotive/autonomous-driving-levels.html#:~:text=Level%205%20(Full%20Driving%20Automation,experienced%20human%20driver%20can%20do.
4. Team, K. (2021). Keras documentation: The Functional API. Retrieved 11 July 2021, from https://keras.io/guides/functional_api/
5. Confusion matrix — scikit-learn 0.24.2 documentation. (2021). Retrieved 11 July 2021, from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
6. Python Project on Traffic Signs Recognition with 95% Accuracy using CNN &Keras - DataFlair. (2021). Retrieved 11 July 2021, from https://data-flair.training/blogs/python-project-traffic-signs-recognition/
7. Mohamed Tarek. (2021). Retrieved 11 July 2021, from https://www.slideshare.net/mohamedtarek2013
8. Agile Methodology: What is Agile Software Development Model & Process in Testing?. (2021). Retrieved 11 July 2021, from https://www.guru99.com/agile-scrum-extreme-testing.html
9. Comprehensive Guide to the Agile Manifesto. (2021). Retrieved 11 July 2021, from https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto
10. Education, I. (2021). Introduction to Mobile Application Development. Retrieved 11 July 2021, from https://www.ibm.com/cloud/learn/mobile-application-development-explained
11. Introduction to Flutter: framework and its architecture - Axon. (2021). Retrieved 11 July 2021, from https://www.axon.dev/blog/flutter-a-full-introduction-to-the-framework
12. Introduction to web APIs - Learn web development | MDN. (2021). Retrieved 11 July 2021, from https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

13. Brala, B. (2021). Introduction to the JSON API. Retrieved 11 July 2021, from https://laravel-news.com/json-api-introduction
14. Brala, B. (2021). Introduction to the JSON API. Retrieved 11 July 2021, from https://laravel-news.com/json-api-introduction