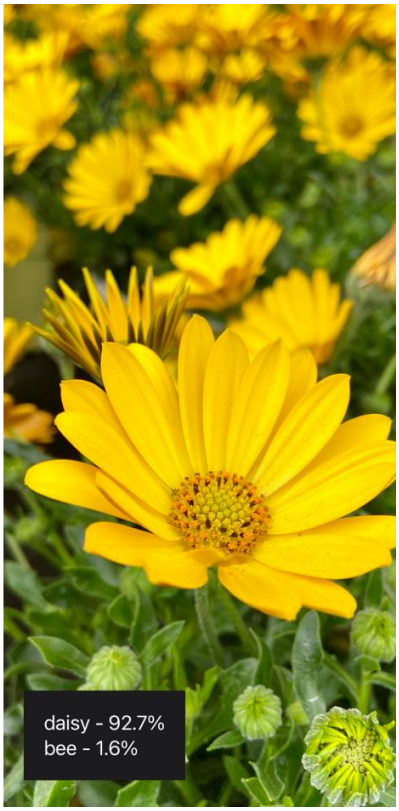
TABLE OF CONTENT

INTRODUCTION

1.1 Introduction-

The app in this sample identifies the most prominent object in an image by using MobileNet, an open source image classifier model that recognizes around 1,000 different categories.



Each time a user selects a photo from the library or takes a photo with a camera, the app passes it to a Vision image classification request.

Vision resizes and crops the photo to meet the MobileNet model's constraints for its image input, and then passes the photo to the model using the Core ML framework behind the scenes. Once the model generates a prediction, Vision relays it back to the app, which presents the results to the user.

The sample uses MobileNet as an example of how to use a third-party Core ML model. You can download open source models — including a newer version of MobileNet — on the Core ML model gallery.

Before you integrate a third-party model to solve a problem — which may increase the size of your app — consider using an API in the SDK.

For example, the Vision framework's VNClassifyImageRequest class offers the same functionality as MobileNet, but with potentially better performance and without increasing the size of your app (see Classifying Images for Categorization and Search).

Also you can make a custom image classifier that identifies your choice of object types with Create ML. See Creating an Image Classifier Model to learn how to create a custom image classifier that can replace the MobileNet model in this sample.

1.2 Problem Statement -

The project "Object Detection System using Machine Learning Technique" detects objects efficiently based on the algorithm on image data and video data to detect objects.

1.3 Need for the proper System -

Following are the requirement-

1. IOS 11 or later
2. MacOS 10.13 or later
3. Xcode latest version

4. To take photos within the app, run the sample on a device with a camera. Otherwise, you can select photos from the library in Simulator.

1.4 Objective -

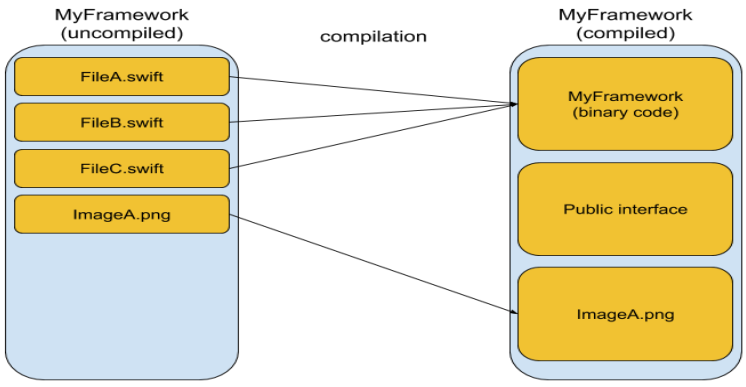
The Objective of the project is to Crop and scale photos using the Vision framework and classify them with a Core ML model and to take photos within the app, run the sample on a device with a camera. Otherwise, selected photos from the library in Simulator.

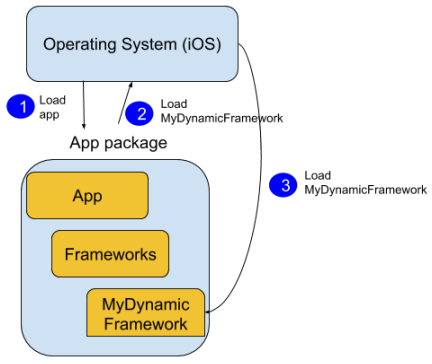
1.5 Modules of the system -

Modularisation of this IOS app is done using-

1. Dynamic frameworks

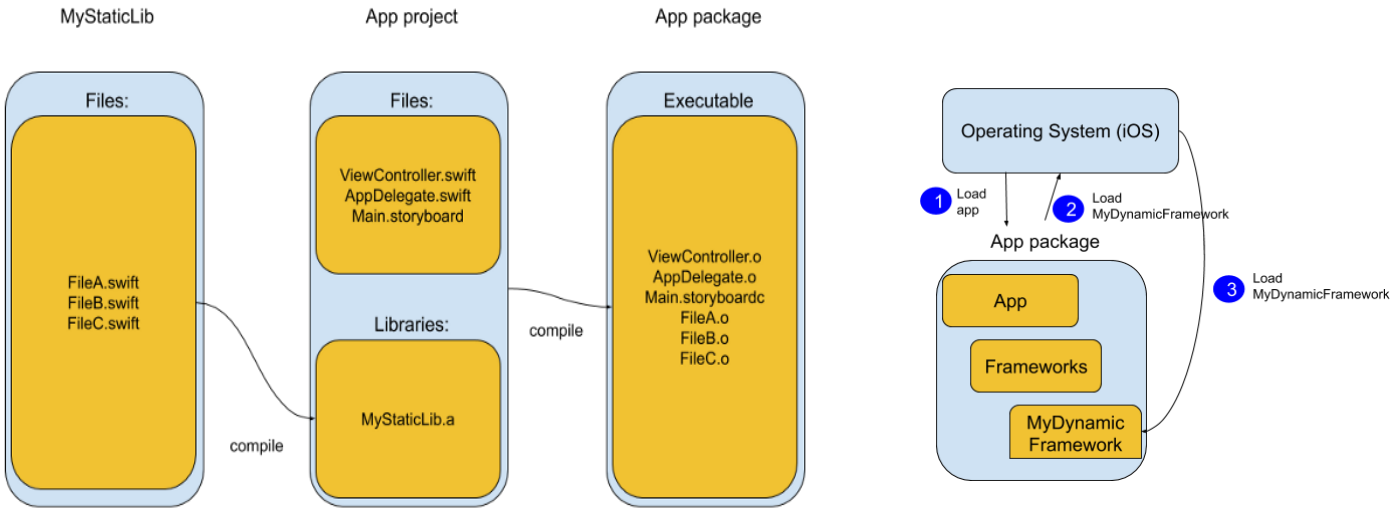
A framework is a structured in directory that can contain shared code and shared resources such images, nibs (compiled form of xibs and storyboards) and other assets.





2. Static libraries

A static library is a collection of compiled source code files. Let's say we have FileA.swift, FileB.swift and FileC.swift. With a static library we can compile these files and wrap them inside a single file containing all of these. The file has has an extension of .a; short for archive. Sort of like getting some pages and making a book out of it.



1.6 Scope –

Object detection is a vision technique that **allows us to identify and locate objects in an image or video**. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them.

LITERATURE SURVEY

2.1 Existing System -

The situation has been significantly changed with the advent of the operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc. Each of these "eternal" problems are solved by trial and error by the efforts of numerous groups of the engineers and scientists.

As modern technical solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems. Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image.
Object localization is referring to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term "object recognition", they often mean "object detection".

2.2 Proposed System -

With reduced complexity, faster computation, latest dynamic framework, and static libraries this system provides a means to crop and scale photos using the Vision framework and classify them with a Core ML model and to take photos within the app, run the sample on a device with a camera. Otherwise, selected photos from the library in Simulator.

2.3 Feasibility Study -

Overall App Feasibility-
• Considering the major features of the current scope (at a high level) and evaluating as they are feasible
• Benefits of the product or service
– Will earn money from downloads (APP STORE).
– Will move one step ahead in development like camera and use of AR, VR.
• Old concept: App in which most of the technical concepts has already been implemented earlier in our apps or other apps in market like Photo editor, Identify screen etc.

2.3.1 Technical Feasibility-

• Experimental features: Identified features in the design that seem experimental in nature, such as techniques, perspectives, or other unique ideas.

Machine Learning Image Analysis
Training a Create ML Model to Classify Flowers
Train a flower classifier using Create ML in Swift Playgrounds, and apply the resulting model to real-time image classification using Vision.
class VNCoreMLRequest
An image analysis request that uses a Core ML model to process images.
class VNClassificationObservation
An object that represents classification information that an image analysis request produces.
class VNPixelBufferObservation
An object that represents an image that an image analysis request produces.
class VNCoreMLFeatureValueObservation
An object that represents a collection of key-value information that a Core ML image analysis request produces.

• What kinds of technology will we need?

1. IOS 11 or later
2. MacOS 10.13 or later
3. Xcode latest version

4. To take photos within the app, run the sample on a device with a camera. Otherwise, you can select photos from the library in Simulator.

• Is relevant technology developed enough to be easily applied to our problem?
Yes who are you having ready when technology developed enough to be applied and get thorough solution but in the future scope we need to incorporate technologies which are developing to their fullest potential such as augmented reality and virtual reality.
• Do we currently possess the necessary technology?
- Yes, and it does have the capacity to handle the solutions around 1,000 different categories of open-source image classifier model.

2.3.2 Economic Feasibility-

• Resource cost is based on the estimated resources within the technical analysis.
• Employee costs should be based on salaries and overhead.
    - In the event of expansion as automated technology require this application to help automate tasks.
• Any hardware or software that you purchase should be listed as well.
    - Currently none as it uses free API and frameworks provided by Core ML(Machine Learning) from Apple Development.
• Additional costs (if any): This section is an assessment of additional costs incurred from licensing, contracting, out-sources testing, and so on.
    - None.
• Cost of maintenance of equipment.
    - None.

2.3.3 Operational Feasibility-

• Assessment of the overall appeal to the market for the APP
• Market timeliness: best suitable time for release
• Identification of the target audience
    - Cloud based platforms, Automation Technology etc.
• Comparing with other similar competing Apps.

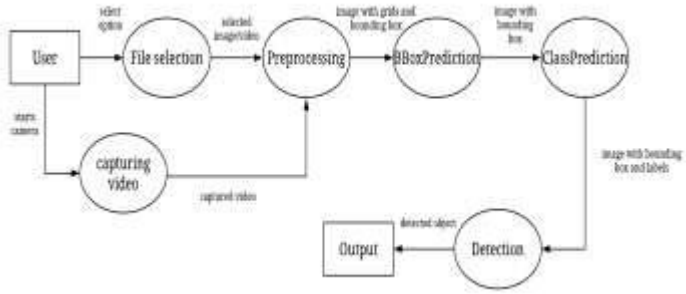
Finding out special features/differentiators-
1. Add better graphics
2. Better marketing (hit idea, Catchy name, keyword rich description, added girl for in app billing etc.)

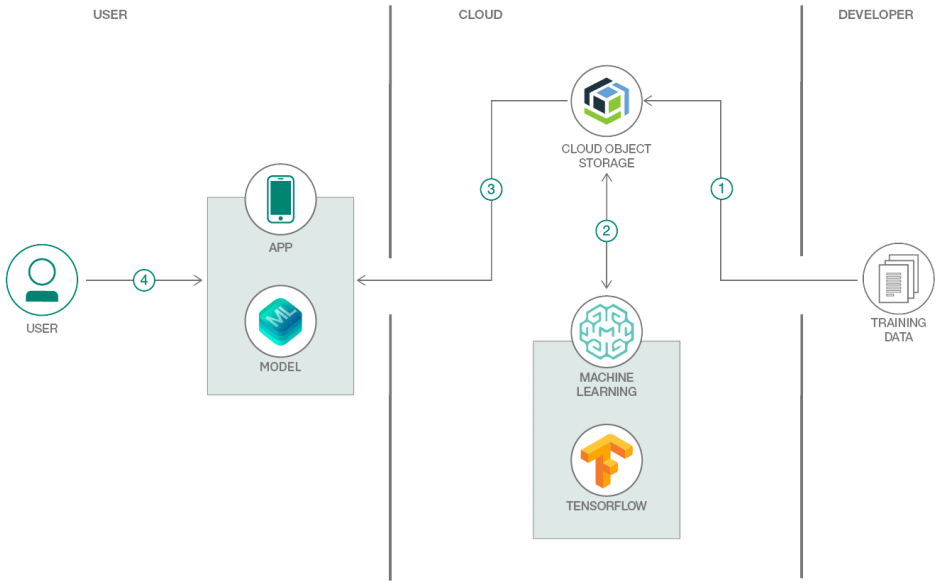
Schedule Feasibility-
• Do we have right estimation for this project?
• Can we complete the project on said date?
• Did we schedule the review period well in advance?
• Did we take into account the national holidays, team, individual leaves?
• Based on this parameters we calculate the score and check the schedule feasibility of the project.

REQUIREMENTS ANALYSIS

3.1 Method used for Requirement analysis -

Data flow diagram
This technique is used to visually represent systems and processes that are complex and difficult to describe in text. Data flow diagrams represent the flow of information through a process or a system. It also includes the data inputs and outputs, data stores, and the various subprocess through which the data moves. DFD describes various entities and their relationships with the help of standardized notations and symbols. By visualizing all the elements of the system, it is easier to identify any shortcomings. These shortcomings are then eliminated in a bid to create a robust solution.





3.2 Data Requirements -

The application uses camera to get it's data set Otherwise, Add your own photos to the photo library in Simulator by dragging photos onto its window by-

Creating an Image Classifier Instance-

At launch, the ImagePredictor class creates an image classifier singleton by calling its createImageClassifier() type method.

```
/// - Tag: name
static func createImageClassifier() -> VNCoreMLModel {
    // Use a default model configuration.
    let defaultConfig = MLModelConfiguration()

    // Create an instance of the image classifier's wrapper class.
    let imageClassifierWrapper = try? MobileNet(configuration: defaultConfig)

    guard let imageClassifier = imageClassifierWrapper else {
        fatalError("App failed to create an image classifier model instance.")
    }

    // Get the underlying model instance.
    let imageClassifierModel = imageClassifier.model

    // Create a Vision instance using the image classifier's model instance.
    guard let imageClassifierVisionModel = try? VNCoreMLModel(for:
imageClassifierModel) else {
        fatalError("App failed to create a `VNCoreMLModel` instance.")
    }

    return imageClassifierVisionModel
}
```

3.3 Functional Requirements –

Functions the product is required to perform-

the project needs to Crop and scale photos using the Vision framework and classify them with a Core ML model and to take photos within the app, run the sample on a device with a camera. Otherwise, selected photos from the library in Simulator.

The method creates a Core ML model instance for Vision by:

- Creating an instance of the model's wrapper class that Xcode auto-generates at compile time.
- Retrieving the wrapper class instance's underlying MLModel property.
- Passing the model instance to a VNCoreMLModel initializer.
- The Image Predictor class minimizes runtime by only creating a single instance it shares across the app.

3.4 Non-Functional Requirements -

1: Identify Key Stakeholders and End-Users

The first step of the requirements analysis process is to identify key stakeholders who are the main sponsors of the project. They will have the final say on what should be included in the scope of the project.

Next, identify the end-users of the product. Since the product is intended to satisfy their needs, their inputs are equally important.

2: Capture Requirements

Ask each of the stakeholders and end-users their requirements for the new product. Here are some requirements analysis techniques that you can use to capture requirements:

- Hold One-on-One Interviews

Interview each stakeholder and end-user individually. This technique will help you gather specific requirements.

- Use Focus Groups

Conduct group interviews or group workshops to understand the flow of information between different stakeholders and end-users. This technique will ensure that there will be no conflict of interest later on during the project.

-Utilize Use Cases

Use cases provide a walkthrough of the entire product through the eyes of the end-user. This technique will help visualize how the product will actually work.

- Build Prototypes

A prototype provides users a sample look and feel of the final product. This technique will help address feasibility issues and identify problems ahead of time.

3.4 System Specification –

3.5.1 Hardware specification-

This app targets iOS operating system. A list of all the current devices supporting this OS:
iPhone:11 or later
MacOS 10.13 or later
iPod: 5th generation.
Pad: 2, 3rd and 4th generation, Air.
The app has been tested on the above highlighted devices. However, all the performance metrics and exhaustive tests have been executed on the iPhone 13. This device presents the following technical specifications:
Processor 1.3 GHz dual-core Apple-designed ARMv7s (Apple A6). RAM 1GB LPDDR2 DRAM.

Graphics PowerVR SGX543MP3 (tri-core, 266 MHz) GPU. Storage 16, 32 or 64 GB.
Back Camera 8 MP photos, f/2.4, 1080p HD video (30 fps), Infrared cut- o↵ filter, Back-illuminated sensor, face detection, video stabilization, panorama and ability to take photos while shooting videos.
Front Camera 1.2 MP photos, 720p HD video (30 fps), Back-illuminated sensor.
For the preset we use, the camera uses images with a resolution of 460 ⇥ 380.

3.5.2 Software Specification-

- Xcode: Latest version.
- Vision Framework.
- IOS: 11 or later.
- Core ML Model.

DESIGN

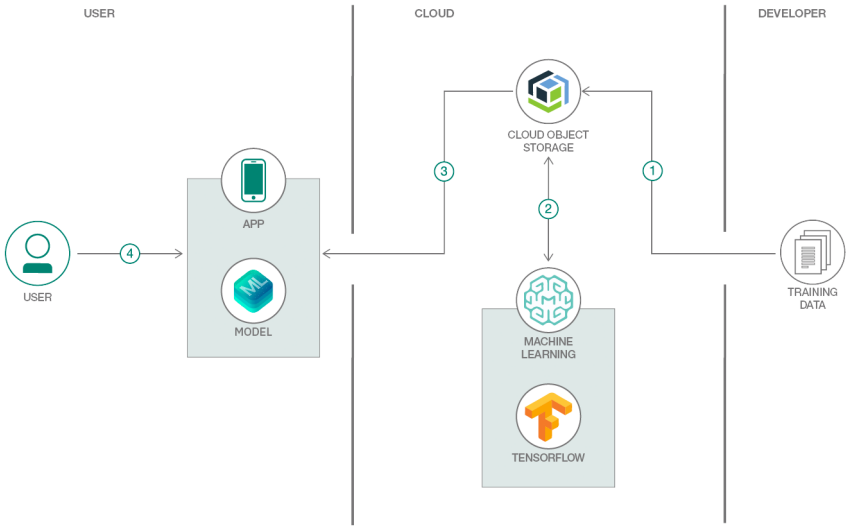
4.1 Software Requirements Specification –

- Xcode: Latest version.
- Vision Framework.
- IOS: 11 or later.
- Core ML Model.

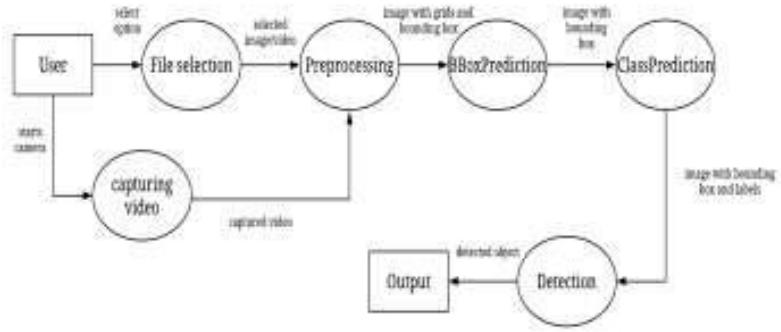
4.1.1 Supplementary Specifications -

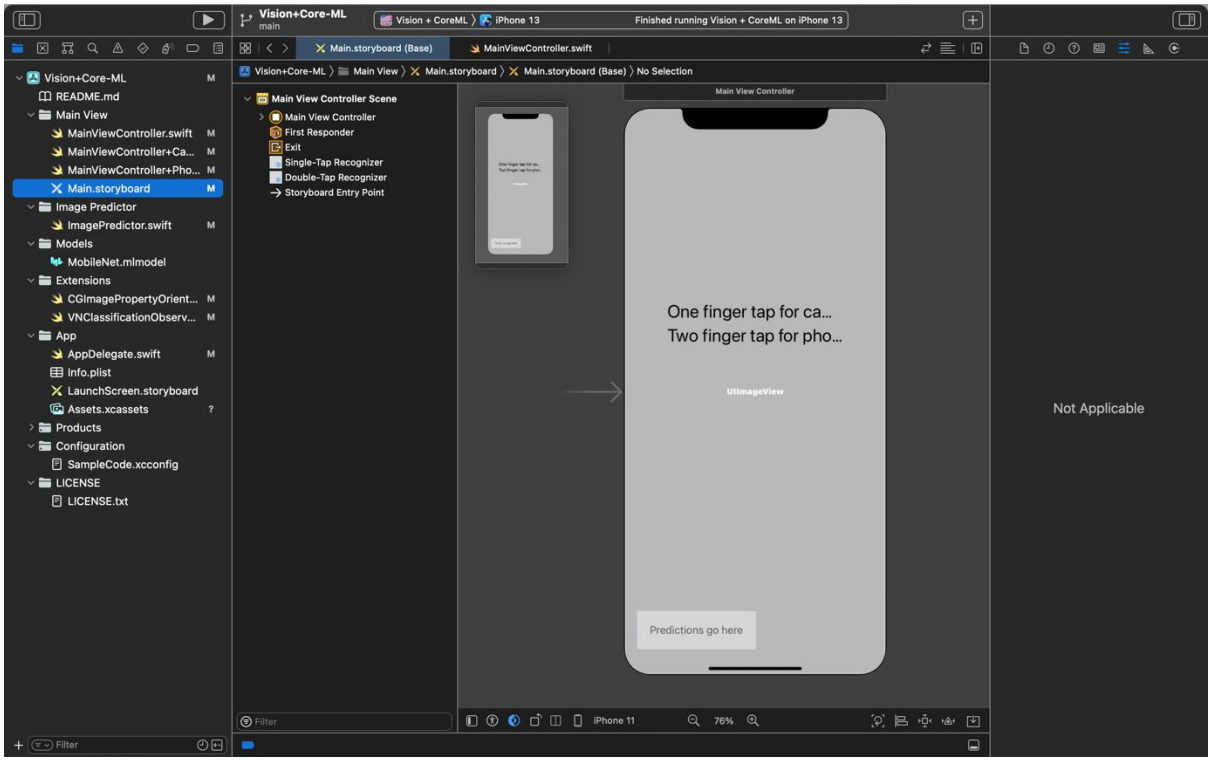
iPhone:11 or later
MacOS 10.13 or later
iPod: 5th generation.
Pad: 2, 3rd and 4th generation, Air.
The app has been tested on the above highlighted devices. However, all the performance metrics and exhaustive tests have been executed on the iPhone 13. This device presents the following technical specifications:
Processor 1.3 GHz dual-core Apple-designed ARMv7s (Apple A6). RAM 1GB LPDDR2 DRAM.
Graphics PowerVR SGX543MP3 (tri-core, 266 MHz) GPU. Storage 16, 32 or 64 GB.
Back Camera 8 MP photos, f/2.4, 1080p HD video (30 fps), Infrared cut- o↵ filter, Back-illuminated sensor, face detection, video stabilization, panorama and ability to take photos while shooting videos.
Front Camera 1.2 MP photos, 720p HD video (30 fps), Back-illuminated sensor.
For the preset we use, the camera uses images with a resolution of 460 ⇥ 380.

4.1.2 Use Case Model –



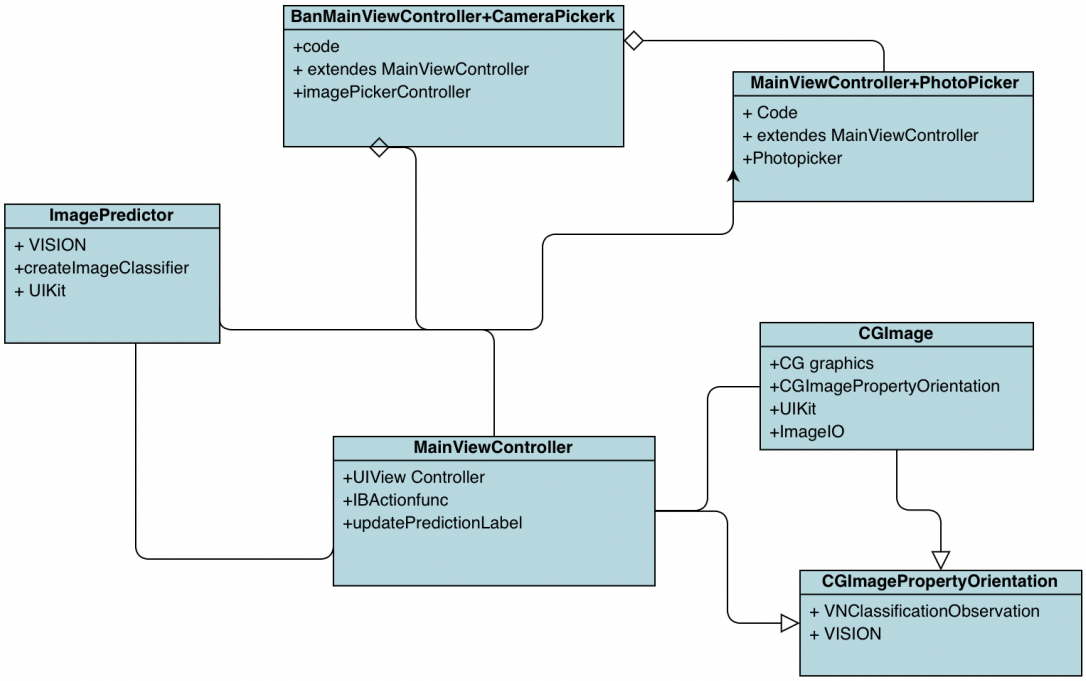
4.1.3 Data flow Diagram -



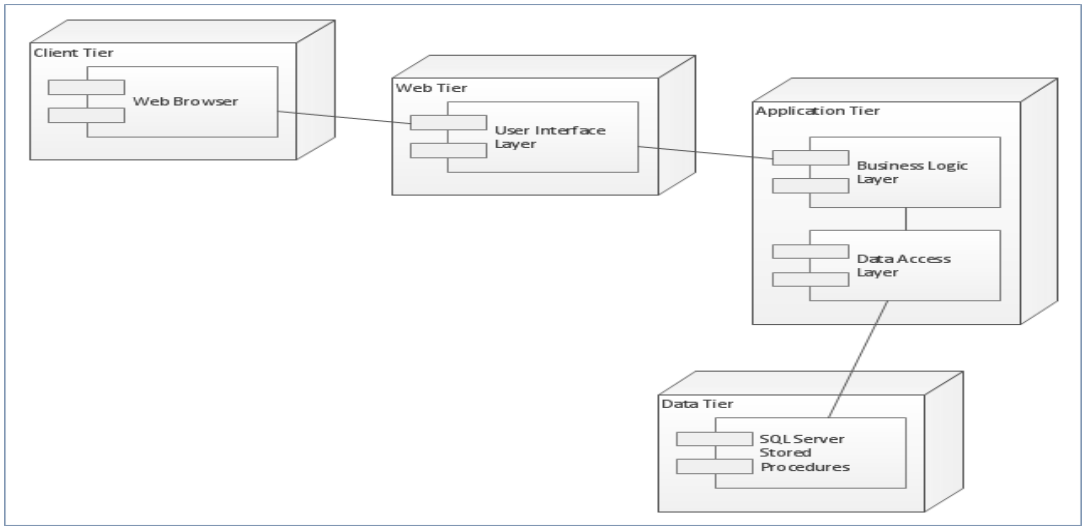


SYSTEM MODELING

5.1 Detailed Class Diagram-



5.1.1 Deployment Diagram-



5.2 Testing –

The Table-1 shows the different Test Cases.
Test Results –

TC1	When image is chosen as input	Image with bounding box around the objects and predicted class	SUCCESSFUL
TC2	When video is chosen as input	Video with bounding box around the objects and predicted class	SUCCESSFUL
TC3	When camera is chosen as input	Objects detected in the real time with bounding box, confidence score and predicted class	SUCCESSFUL
TC4	When black and white image is taken as input	Image with bounding box around the objects and predicted class	SUCCESSFUL
TC5	Image with far objects is taken as input	Image with detected objects	UNSUCCESSFUL
TC6	When image with overlapping objects is	Image with bounding box around the objects and	SUCCESSFUL
TC7	When image with far objects is taken as input	Image with detected objects	UNSUCCESSFUL

CONCLUSION & FUTURE WORK

6.1 Limitation of Project-

The only limitation this app contains is that it is only available for IOS Operating system. The project is developed with objective of detecting real time objects in image, video and camera. Bounding Boxes are drawn around the detected objects along with the label indicating the class to which the object belongs. We have used CPU for the processing in the project.

6.2 Future Scope of the Project –

- Future enhancements can be focused by implementing the project on the system having GPU for faster results and better accuracy.
- To add Other Important CoreML like object motion tracking, AI Visual Enhancement, AR, VR.

BIBLIOGRAPHY & REFERENCES

7.1 Reference Books, links, and docs -

7.1.1 BOOKS-

- The Swift Programming Language Edition - 5.3.
- App Development with Swift.

7.1.2 LINKS-

- https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml.
- https://developer.apple.com/documentation/vision.
- https://developer.apple.com/documentation/coreml.