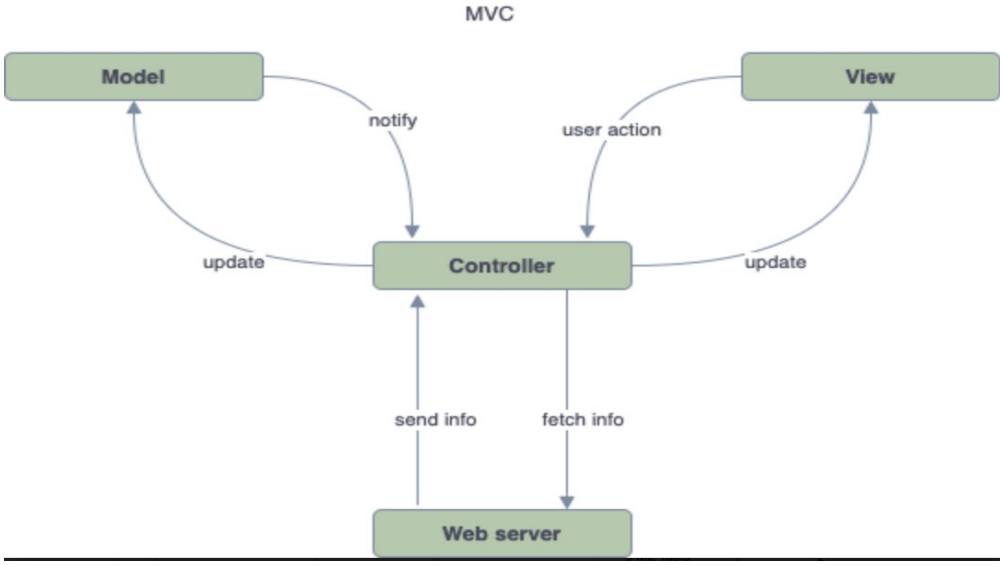
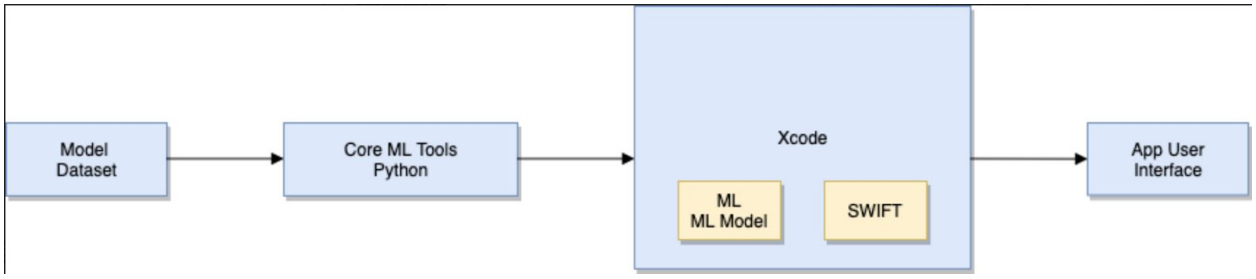
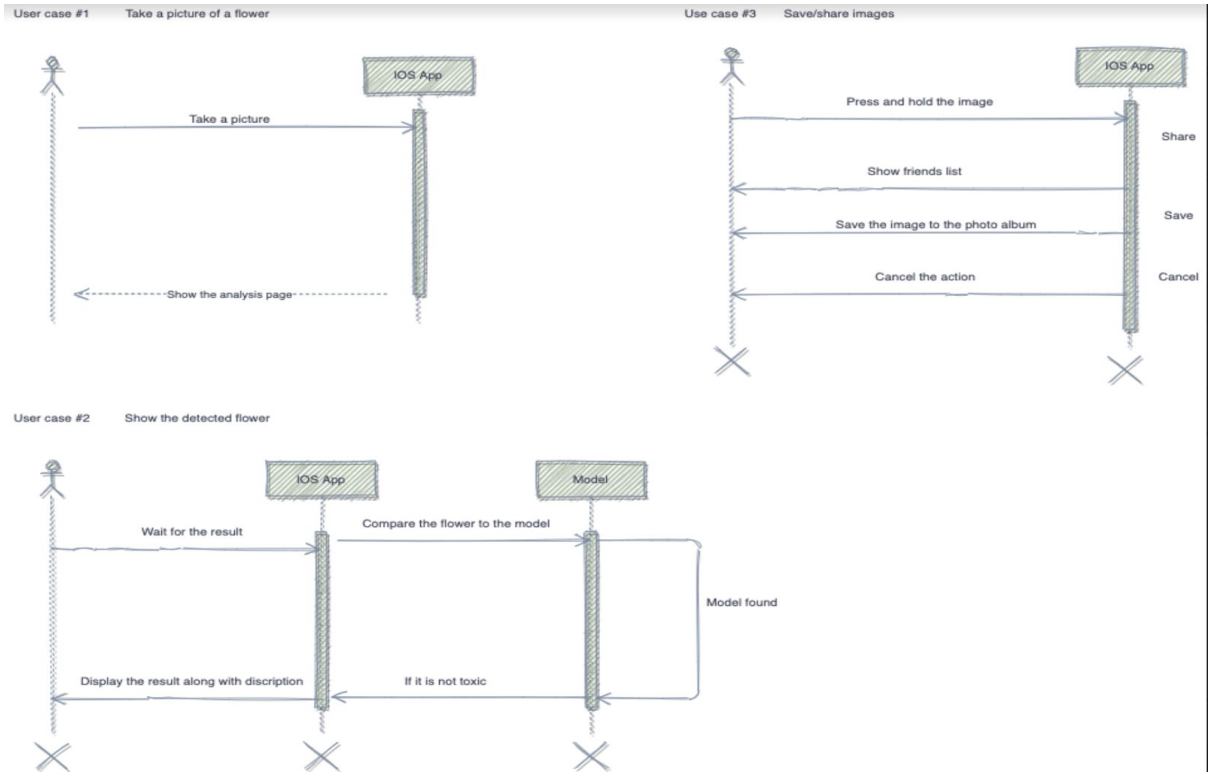
Final Report

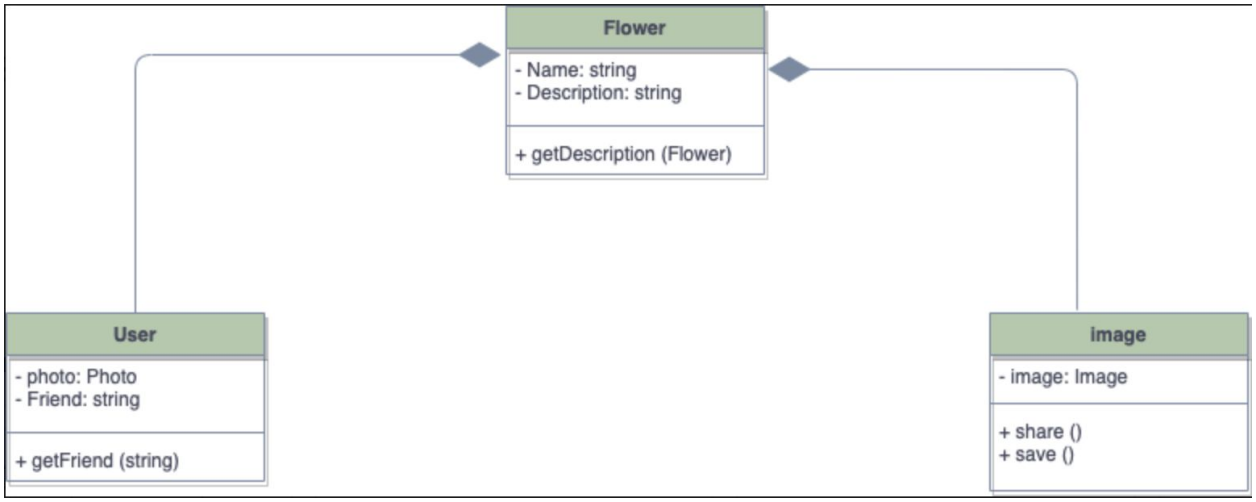
- Name of project and names of all team members:
    - iOS Flower Identifier App
    - Tongxin Zhu, Ruijiang Ma
- Final State of System Statement:
    - When the user sees an unknown flower, they can use this app to take a picture of the flower, and this iOS Flower Identifier App will tell the user which kind of flower it is. It will also fetch a description of the flower from the Wikipedia page.
- A paragraph on the final state of your system, what features were implemented, what features were not, and why, what changed from Project ⅘:
    - Implemented Features: The first feature is to identify the flower, Once the user has taken a picture of the flower, this app will use the pre-trained machine learning model to identify the flower and display the name on the screen. The second feature is: once the flower is identified, this app will go to the Wikipedia page of this flower and fetch the description as well as the picture of the flower and show it to the user.
    - Not Implemented Features: We originally wanted to build a homepage for this app that will automatically display the weather based on the user's current location, in addition to the main identification feature. But because of the complexity and the limitation of time, we end up not doing this part of the project.

- Final Class Diagram and Comparison Statement









- The key change of the system is that the application of MVC model. At the beginning of project 3, the structure of the project is purely based on the functionality of this app, that is, we didn't think much regarding UI back in the project 4. As more research has been done, we realized that in order for this app to work, we would need to use the MVC model since an iOS app is not just writing code, we need a model that can effectively and efficiently link the code and UI together.

• Third-Party code vs. Original code Statement

- Tutorial: iOS 11 & Swift 4 - The Complete iOS App Development Bootcamp https://www.udemy.com/course/ios11-app-development-bootcamp/

Stanford - Developing iOS 11 Apps with Swift [2017-18]:
https://www.youtube.com/watch?v=TZL5AmwuwlA&list=PL3d_SFOiG7_8ofjyKzX6Nl1wZehbdiZC_
The script file to transform the pre-trained caffe model to the .mlmodel was borrowed here:
https://apple.github.io/coremltools/generated/coremltools.converters.caffe.convert.html
• The Model class of our project also uses the example of the SeeFood-CoreML project from The Complete iOS App Development
https://github.com/appbrewery/SeeFood-iOS13-Completed/tree/master/SeeFood-CoreML.xcodeproj

• Statement on the OOAD process for your overall Semester Project
  • Model: We were not sure what methods the model class should contains initially. But after analyzing the features of the app, we knew that we at least need a function for deceting the images, and function to fetch information from the Internet.
  • View: Even though the user interface is relatively simple, we didn't know how should we link the UI components to the controller due to the independency of UI. We did know that we would need a listener to the view once the app is loaded, and we also needed a camera button that has a listener associated with it. Fortunately, these are all handle well by Xcode since it have API for these tasks.
  • Controller: This is probably the most challenging part of this project due to the nature of controller. We thought of the controller as the main class in Java since they both plays a role of initialization. However, as the development proceeds, we realized that controller actually plays a more crucial role in the MVC model. It acts as a bridge between the model and the view. So instead of just placing the initialization part of code inside the controller. We also included the imagePickerController method and the viewController method which allow the controller to does the job of coordinating between Model and View.