Mockplus:

Mockplus is an app that helps app developers to layout what their app will look like. Planning out how an app looks and functions prior to the actual coding of it allows developers to plan accordingly and see what needs to be done. Furthermore, this acts like a blueprint for what needs to be done as Mockplus allows users to create buttons and pages that will look somewhat similar to its end product. It also provides very detailed items, such as sliders, drop-down boxes, and as well as a scrolling function.

Dataset:

The most important dataset that was used in the development of the app was the USDA ingredient database, which provided the information on allergens, specifically their categories as well as some of the most prevalent allergens. This was incorporated into setting a user's allergen profile where users are able to select whether they are able to consume it, to limit it, or to avoid it. Furthermore, the dataset provided us with in-depth information about subcategories for larger brands of allergens, such as nuts (which contain multiple other allergens like almonds, walnuts, etc.). This was taken into consideration as some users may be allergic to a specific type of nut, but are able to eat other types of nuts. Most importantly, it also stores over 240,000 barcode information, which will be recognized when a user tries to scan a barcode. The barcode in the database contains detailed information about the ingredients that were used, all collected by the USDA.

The Code:

In this study, all of the backend and frontend components of the application were coded in Xcode with Swift, an intuitive programming for iOS, iPadOS, macOS, tvOS, and watchOS. Swift is heavily supported and created by Apple, which has provided detailed information on all the necessary tips and tricks – as well as support – an app developer could ask for. Most importantly, any references could be found in Xcode's built-in library which contains a variety of solutions to a problem. This library is also full of Xcode's default images which have been used as icons or buttons (such as the camera, barcode and person).

The first part of the coding was to get the back-end of the code finished. The front-end of the app, primarily aesthetics, were done last. The three primary structures of the app are the

GUI, Barcode Scanner, and OCR scanner. The GUI was left very simple, with the first page when opening the app composed of the name of the app (AllergyAssist), a picture of food items, and three buttons (User Account, Scan Barcode, Scan Label). These buttons would lead to a UITabBarController, which essentially leaves all three of those buttons on the button of the phone's screen so that it is always accessible. The UITabBarController would also be

The barcode scanner was created by using the built-in library of AVFoundation. The class is paired with the AVCaptureMetadataOutputObjectsDelegate and by this a textField is created which will bring back a barcode and a button that will lead us to our BarcodeScanner class.

The OCR scanner was mostly made with the Vision kit, another useful built-in library within Xcode. The first step in the creation of an OCR scanner was to convert the image into a cgImage (Bitmap image). The next step was to create the request with cgImage using the VNImageRequestHandler, which would be connected to the Vision provided VNRecognizeTextRequest.

All of the code was separated into many different files, with each file constituting its own ViewController (VC) or two VCs. Each VC was a separate function of the app, for example, the ScanLabel file consists of all the components for the OCR Scanner, whereas the ScanBarcode file consists of all the components for the barcode scanner. Separating all these files creates a neater and more organized working environment. Furthermore, all the separate ViewControllers were written while following the layout for the Mockplus. Buttons, labels, and other front-end components were combined with the main functionality of that page, meaning that the button to scan a barcode would be within the BarcodeScanner ViewController.