

Computer Games Development

Project Report

Year IV

Sean Nash de Andrade

C00217019

29th April 2020

# 

# 

# 



**Faculty of Science**

**Open-Book and Remote Assessment Cover Page**

**Student Name: Seán A’bner Nash De Andrade**

**Student Number: C00217019**

**Lecturer Name: Martin Harrigan**

**Module: Project II (Fyp)**

**Stage/Year: Fourth Year**

**Date: 03/05/2020**

**Declaration**

This examination/assessment will be submitted using Turnitin as the online submission tool. By submitting my examination/assessment to Turnitin, I am declaring that this examination/assessment is my own work. I understand that I may be required to orally defend any of my answers, to the lecturer, at a given time after the examination/assessment has been completed, as outlined in the student regulations.

# **Contents**

[**Project Abstract**](#_nf7wt2rb9fxs) **2**

[**Project Introduction and Research Question**](#_rkmbhvndp7tm) **3**

[Introduction](#_6mmm2lh2uxgf) 3

[My Project within Erasmus+](#_9djpqq2km3db) 4

[Research Question](#_z0i9h7w5sz6l) 5

[**Background**](#_deyknrnkk07m) **5**

[**Literature Review**](#_jnsin1g32tyx) **5**

[Pytorch Documentation](#_h8gw7zoz533q) 5

[React Documentation](#_a7phuy259gco) 6

[Micheal Bridgette - 2018 Report](#_jtgn56o41iyf) 7

[Study](#_qjj7wjzdg6wp) 8

[**Project Description**](#_g7l5l9us4a5m) **8**

[React Front End App](#_ijfs8htig2js) 9

[Celiapp Integration](#_ovfjj6z2iuq3) 14

[Conformance to Specification and Design](#_u4h32cc0ilb1) 16

[Learning Outcomes](#_tlpn0p867izv) 17

[**Project Milestones**](#_zgukgkvd1qzw) **18**

[**Results and Discussion**](#_m55vpmx3872q) **18**

[**Project Review and Conclusions**](#_eb6a3znawdbk) **19**

[**References**](#_xrtpnete1kdu) **20**

# **Project Abstract**

This project is part of the Erasmus+ Programme. It uses the following Technology and code: Pytorch, Python and React. I will be creating a React Native front end application to allow a user to capture the results on a GIP Test Strick (Gluten Immunogenic Peptides). This stick indicates if a person has consumed gluten and if it is present in their system. This image will then be sent to a server backend where a photo recognition model will return a result based on what result it believes the picture to show. This result will then be displayed to the user in an clear and concise fashion in the UI.The date and time are displayed as is the result and accuracy of said result. The user is able to go through their entire history of tests with minimal effort.

# **Project Introduction and Research Question**

## ***Introduction***

This project is part of the Erasmus+ programme and its title is DESQOL which is a pseudo-acronym for “Teaching Interdisciplinary Human Centred Design to Improve Patient Quality of Life”. This programme has participants from the following:

1. IT Carlow

2. FHV, Vorarlberg University of Applied Sciences

3. HvA, Amsterdam

4. St. Luke's General Hospital Carlow-Kilkenny, HSE

5. Grupo IHP, Andalucia, Spain

Research has shown that teenagers with Coeliac’s Disease tend to struggle to comply with their dietary requirements. To aid them in meeting their gluten free diet requirement the above partners are working to create an application that will help comply with their diet and make their lives easier. The target age group is the 13 - 18 range. Compliant people tend to maintain their gluten free diets but Non-compliant people tend to fail to avoid Gluten constantly, as this behaviour can lead to type-1 Diabetes in the long run methods to help avoid this are needed.  
  
A meeting in Carlow had a few ideas being discussed, ranging from a VR game to teach Celiac patients how to safely prepare food without contamination or which foods are safe to eat to GIP Sticks. A GIP stick kit costs about 12 Euro, Urine can be tested half an hour after ingestion to check for the presence of Gluten. However, results disappear after 20 hours. Faeces tests are done one or two days after ingestion and its results will last for four days before disappearing.

These sticks give either positive or negative results. Lab equipment is more accurate but the purpose of these tests is not to check for Gluten but to monitor intake and compliance. To give certainty to the patients that they are dealing with things properly. The test takes between ten to fifteen minutes depending on what matter is being tested.

## My Project within Erasmus+

My project is to create a phone app that will allow a patient to easily take a picture of their tests. This picture will then be sent to a photo recognition model backend on a server and then the results will be displayed to them while also allowing them to look over previous test results.

# 

My main focus was creating a usable app front end. This is because no matter how accurate or useful the systems in place could be, it would never be used if the interface for it was too complicated or unclear. To this end the app should be easy to understand and use and have as few barriers between the user and their goal as possible. The user should be able to see their results or picture a test result within two button presses worth of complexity. As my focus was on the front end of the app I decided to use my predessor’s backend photo recognition model, integrating it into my front end to create two usable halves needed for a complete application.

As part of this I also integrated my work with React Native into the Erasmus+ Celiapp that went on to be used for the 21 day challenge where Celiac patients test the app.The 21 day challenge is a period where volunteer Celiac patients test out various applications and products to see at the end if their habits have improved or if things have been made easier due to regular use of the products. It is very useful for feedback and opinions due to the large number of people who can be reached for opinions and testing. Adjusting things to fit the design and use of the rest of the app as needed.

I also began exploring technology that would be useful to this goal. Such as the benefits of using Pytorch in comparison to Tensorflow. To this end I trained a Pytorch object detection model to try and identify test strips.

Following this the next part was to have this model set up on a server running on a machine that could be accessed by a phone app I would be making. This server would take in the data of the codified string of the picture the phone app had taken and feed it into the model it was hosting before sending back the result.

This project interested me because I was interested in working with machine learning. It’s a field I personally find fascinating and the addition to possibly having some beneficial impact on the quality of the life of some Celiac patients was quite encouraging.

## Research Question

Can we create a user-friendly mobile app that can capture GIP Stick results and display them to users in a simple and readable fashion? Specifically can we integrate a front end React application with the server back end of the photo recognition model made with Tensorflow. Allowing the user to capture images of their tests and read their results from the server in an easy manner.

# **Background**

I have no real experience with machine learning other than a few dabbling attempts to create a neural network AI.

I researched methods to identify objects and the two most well received systems was google’s TensorFlow and Facebook’s PyTorch. There were various discussions about the better option for which situation, to help solve this problem I decided to use PyTorch to compare my results to previous attempts of people working with Tensorflow.

# **Literature Review**

## Pytorch Documentation

The PyTorch documentation explains some of the benefits for using PyTorch .   
  
Pytorch is a Python-based scientific computing package that uses the power of graphics processing units. It is one of the preferred deep learning research platforms that’s built more for flexibility and speed. It provides among other things two high level features; Tensor computations with strong GPU acceleration support and building deep neural networks on a tape based autograd systems.

The main benefits of Pytorch include:

* A simple interface: it has an easy to use API. This makes it easier to operate and run like Python.
* Pythonic in nature: The library smoothly integrates with Python and thus it can leverage all the services and functionalities offered by the Python environment.
* Computational graphs: And finally Pytorch provides a platform which offers dynamic computational graphs allowing them to be altered during runtime. This is useful when it is not known how much memory will be required for creating a neural network model.

One of the main benefits Pytorch has over TensorFlow is in it’s dynamic versus static graphs. In the majority of programming environments adding two variables x and y representing numbers produces a value containing the result of that addition. E.g  
 X =1  
 Y = 2  
 Z = X+ Y  
 print(z) = 3

This is not the case in TensorFlow. X and Y would not be numbers but would instead be handles to graph nodes *representing* those values, rather than explicitly containing them. Furthermore, and more importantly, adding x and y would not produce the value of the sum of these numbers, but would instead be a handle to a computation graph, which, when executed, produces that value.

So working with TensorFlow is more like metaprogramming than actually programming. Pytorch works more like the latter of the two and is easy for developers to work out the logic of.

## React Documentation

The React Documentation also explains some of the benefits of using React.   
  
Namely that it’s a very friendly to use an open-source Javascript library. It’s very good a building user interfaces for single page applications such as Twitter. It’s used for handling view layer for web and mobile apps and allows the creation of reusable UI components.

It allows the creation of large web applications which can change data without reloading the page. E.g Twitter’s Live #Trending feed. The main purpose of React is to be fast, scalable and simple. It works only on user interfaces in application.

It can also be used to create mobile applications with React Native. This provides React architecture to native applications like IOS and Android. React native is a mobile apps building framework using only Javascript. It uses the same design as React, letting you utilize/include a mobile UI library/declarative components. The most useful part of using react is it allows for components written in objective-c, java or swift.

This all in total means that React is very good to quickly set up a useful and reactive front end the user can use. I plan to use React to handle the front end of my App.

## Micheal Bridgette - 2018 Report

My predecessor also worked on this app in a different manner, from his research report I’ve found a few things to benefit from.  
  
Namely that while TensorFlow is the older library and thus has a more established community and papers written about it. It is much harder to make use of, it offers almost assembly level commands and depends on its community to build up the required API’s and tools. These tools were indeed built, concurrently. As a result there are multiple popular frameworks to use with TensorFlow and making use of it can require learning multiple of them.

TensorFlow is also a “Define before run” system, it’s graphs it allows the user to make are static and there is some difficulty making use of GPU resources. TensorFlow does however allow for abstraction, it takes over the minute details and allows developers to abstract details while they focus on the overall logic as well as some visualization tools available in the TensorBoard visualization suite.

Also from what was explained in Bridgette(2018) his training of the Tensorflow model is well documented and the same process will be undertaken in training a Pytorch model. That being said I should keep an eye out for some of the issues that tripped up his training process such as a smaller sample size of pictures to train his models with leading to him having to create his own results on the strips using markers to get some tests to train with. Another issue being that some of the tests had notes on them, such as dates and batch numbers written on them by the doctors in Grupo IHP in spain as part of their normal documentation processes. But since he didn’t train his models with these in mind they threw off the results to an extent. So when training models there should be a mix of plain and marked tests to ensure that the model is able to get a proper grasp of the results it’s looking for.

## Study

My aim is to produce a front end that is easy to understand within seconds and clear to use by the target audience of teenagers and young adults. To this end I will be making paper mock ups and models to test my classmates and volunteers to see how they react depending on different layouts.

My main aim is to see if they require any information to accomplish their goals on the app or if they are able to achieve them without any help. The main measure of success is time and the amount of help needed to be given. If the user is able to complete their task without needing any help then the layout is clear enough to lead the user down a logical path to their goal.

To this end I also seek to take note of what they do first when they see the app, to record what people are most likely to be drawn to and to make use of this to improve the UI layout for the frontend of the app.

# 

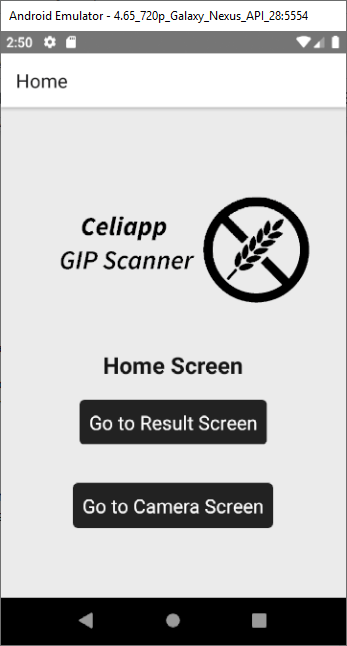
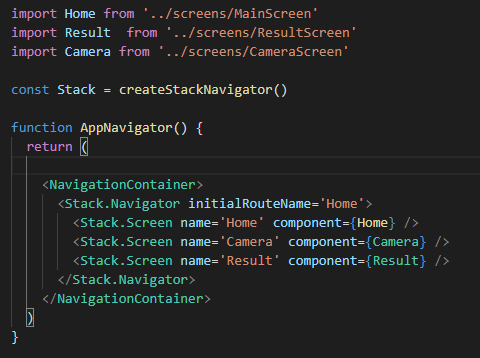
# **Project Description**

My finished product is a React Front End app that is prepared and set up to hook into any back end that a developer can set up for any back end application for the GIP stick in the future. In addition I implemented my work in the Celiapp which went on to be used in a wider testing period called the 21 day challenge where Celiac patients are asked to use and test for the given 21 days and afterwards are asked if the aids helped form any healthy habits along with any feedback for improvement.

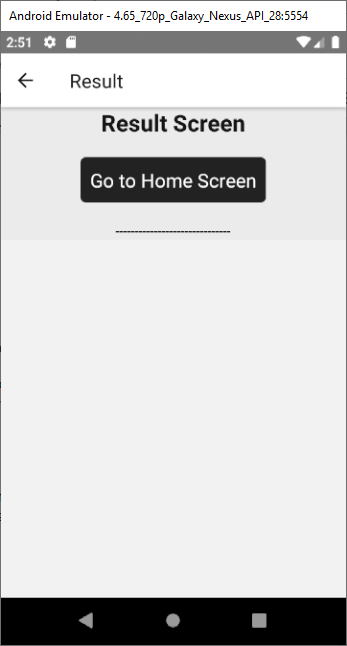
## React Front End App

The React Front end was created using React Native and Javascript. It has a few screens but focuses on allowing the user to quickly access and make use of the App’s functionality.

The App creates and is contained within a container allowing for the entire packaged to be bundled and imported into other applications or projects. The stack navigator created controls the transition between the available screens.



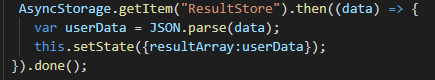
The home screen has few elements in focus of allowing the user to quickly access the result screens or to enter the camera screen. Selecting either of the buttons navigates to the indicated screen but keeps the previous screen in the stack allowing the user to quickly go back to their last location with little latency.



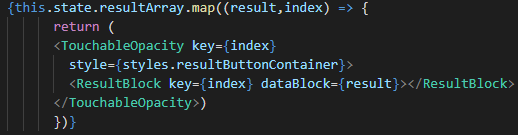
The Result screen is initially empty but displays all the user’s results as they are added. It has an array in its state that is used to hold all the results to display and updates them as needed.

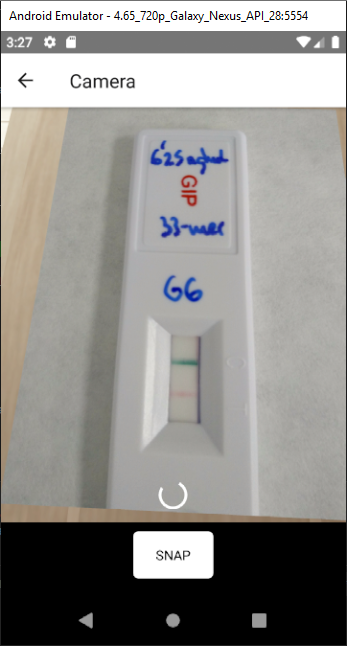
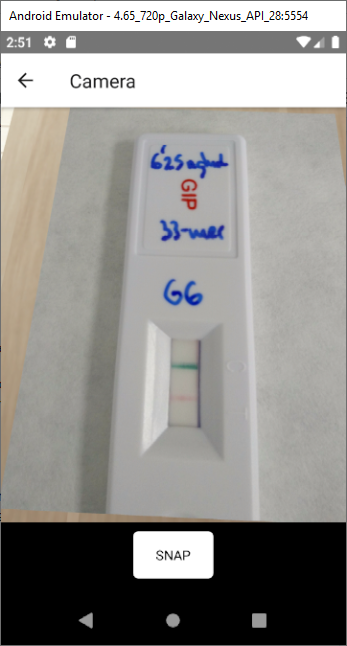


It gets its results from locally saved data on the AsyncStorage of the phone. This is data saved in the app data. This allows the phone’s results to still be browsed even offline and to be cleared as needed.



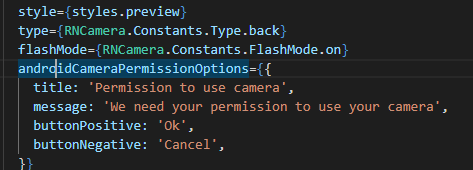
The data is pulled from the memory and parsed from JSON into an array of objects which is then used to update the state.



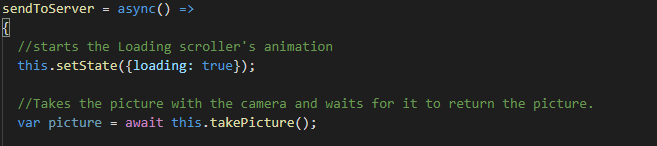
From there when rendering it iterates through the state’s array and creates a ResultBlock widget for each result to display. The ResultBlock is a react component I created for this project to display the results of the GIP tests. It expects to take in a prop to tell it what the result’s data looks like then adjusts itself as needed.  
  
  


The camera screen first requests permission from the user to access the camera systems of the phone. Then it renders what the Back camera is viewing. When the user selects to take the picture there is a loading period while the picture is taken, the picture should be sent to a server and then the results from that server are saved to the AsyncStorage to be used in the result screen which it then navigates too.

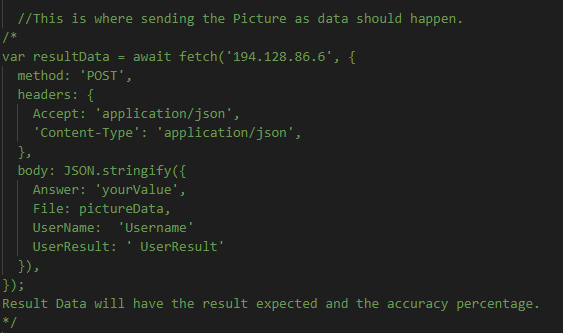
Asking for permission:



Once the button is pressed it calls the sendToServer asynchronous function. This starts the loading widget and sets the camera to take the picture. The await promise ensures that the picture will be taken before continuing.



Below it is where the picture should be sent to the server. Here the picture and required form data is sent to the server as needed before waiting for a response from the server. This section is commented out due to the server backend I was integrating not being set up to send back information. This does however allow for some flexibility for adjusting for different backend systems as the required code is adjusted and slotted into place here to work with little change to the rest of the product.

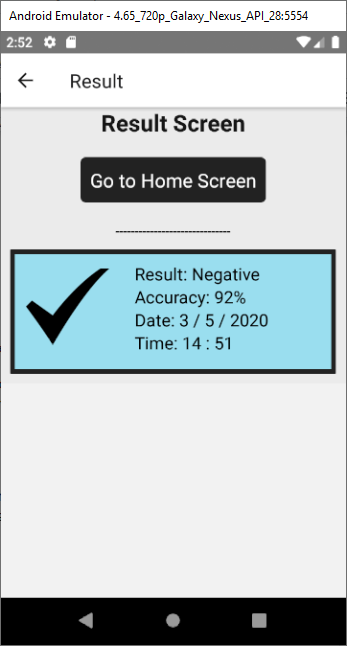


Once a result has been returned from the server it is converted into a format that the ResultBlock components can use. It consists of a float denoting what percentage the photo recognition model thinks it’s result is accurate too. Between 0 and 100%. The current timestamp of the phone and what result the model thinks the picture displays. Negative or positive.

Then the ResultStore saved data is accessed. If it is an empty array then the first element is added. If it is not empty then the new result is appended to the array after being converted to the JSON format. Then the system navigates to the Result screen automatically to display to the user their result.



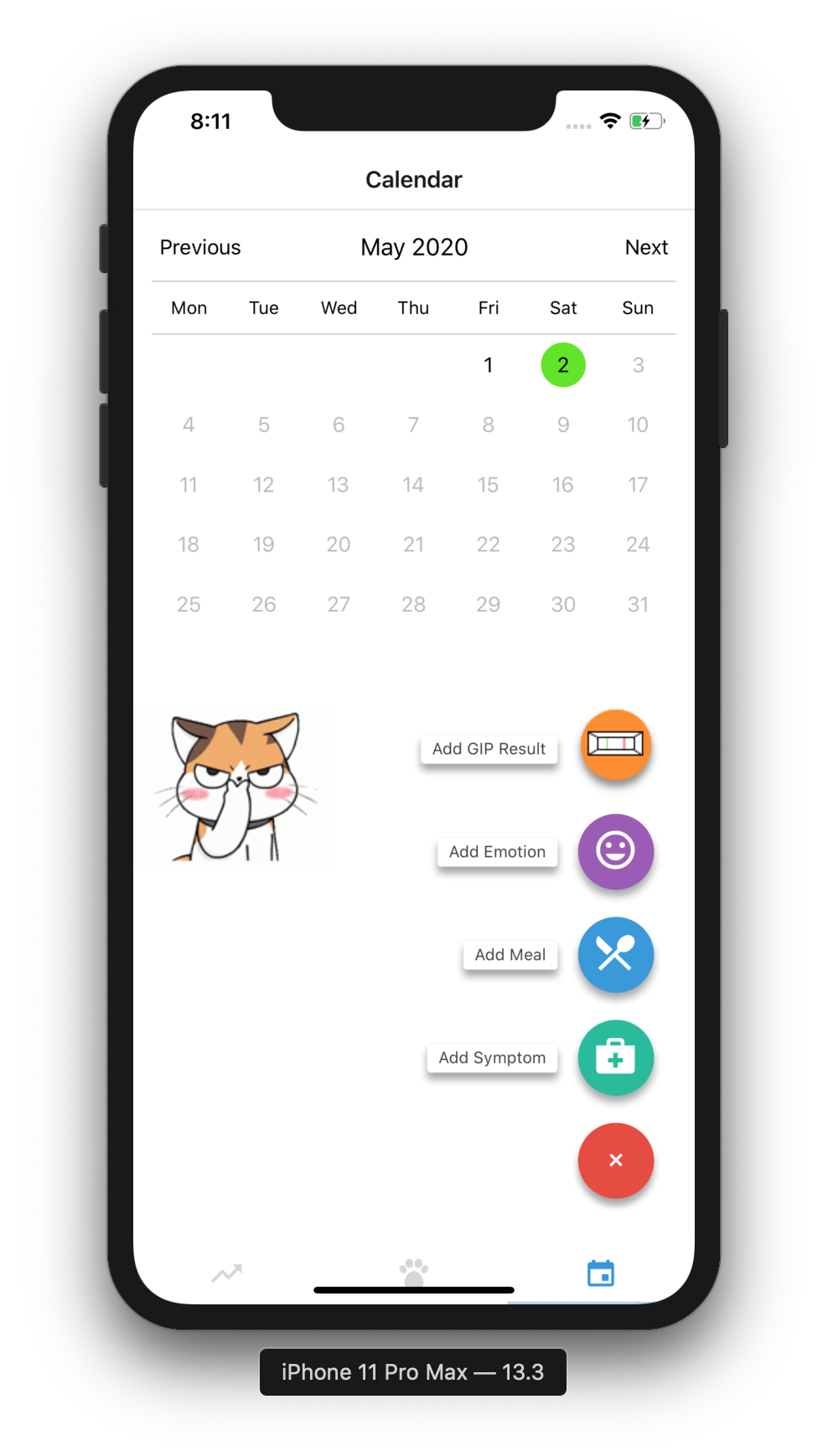
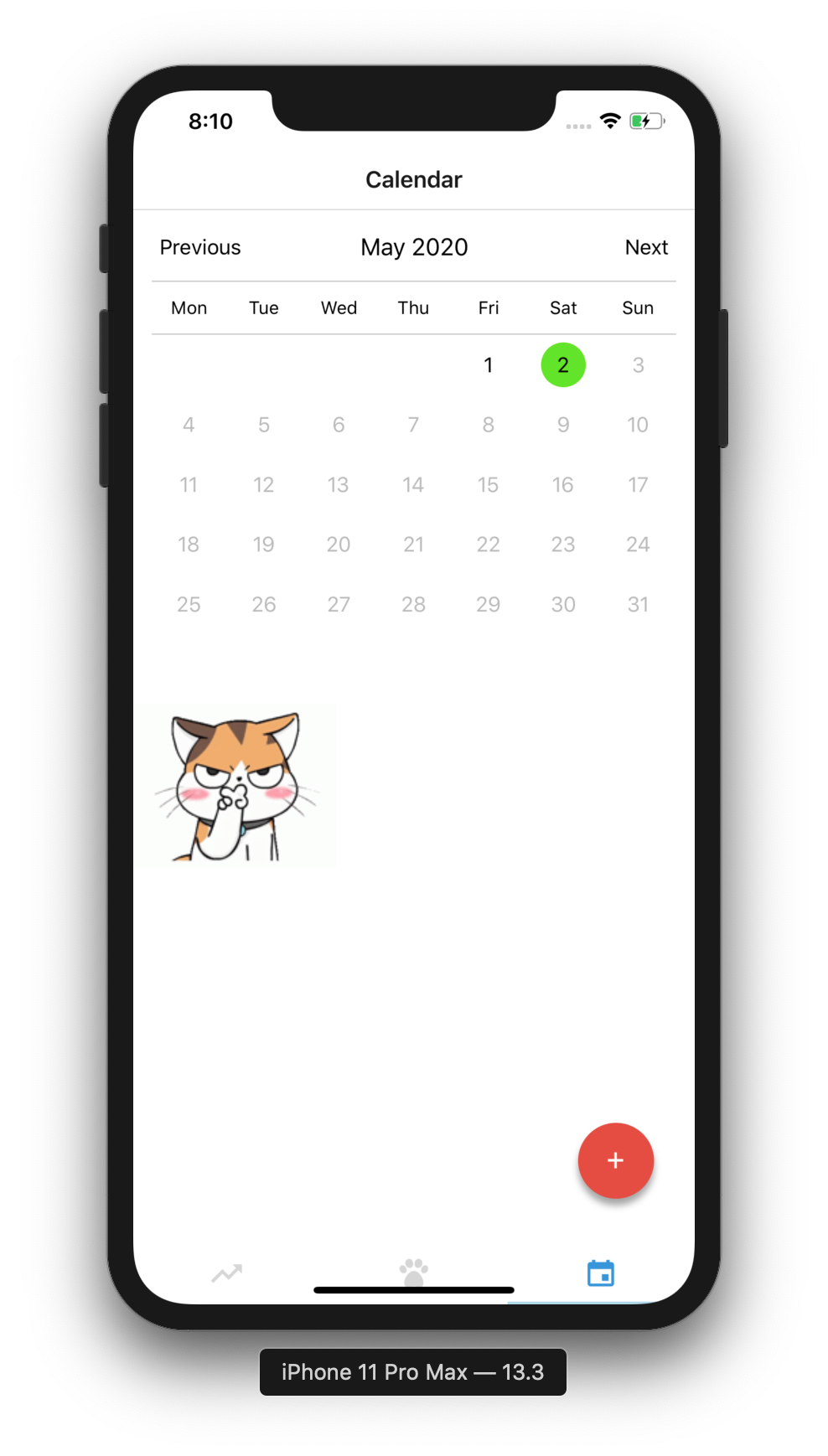
The result screen automatically adds the new result and displays it. It can handle any number of results, once the number of results grows past the amount that can be displayed on a screen it allows the user to scroll through the list of results.

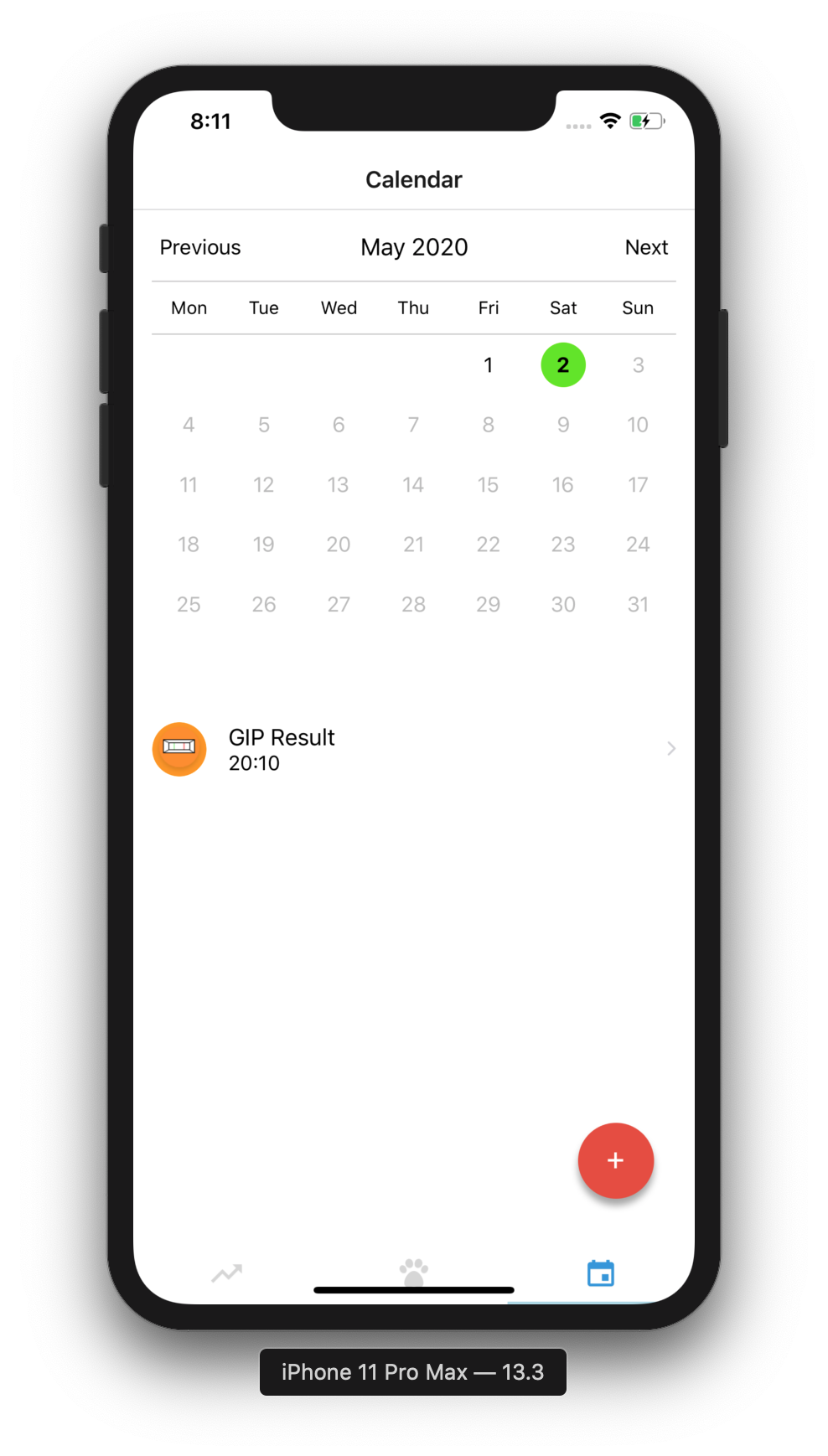


## Celiapp Integration

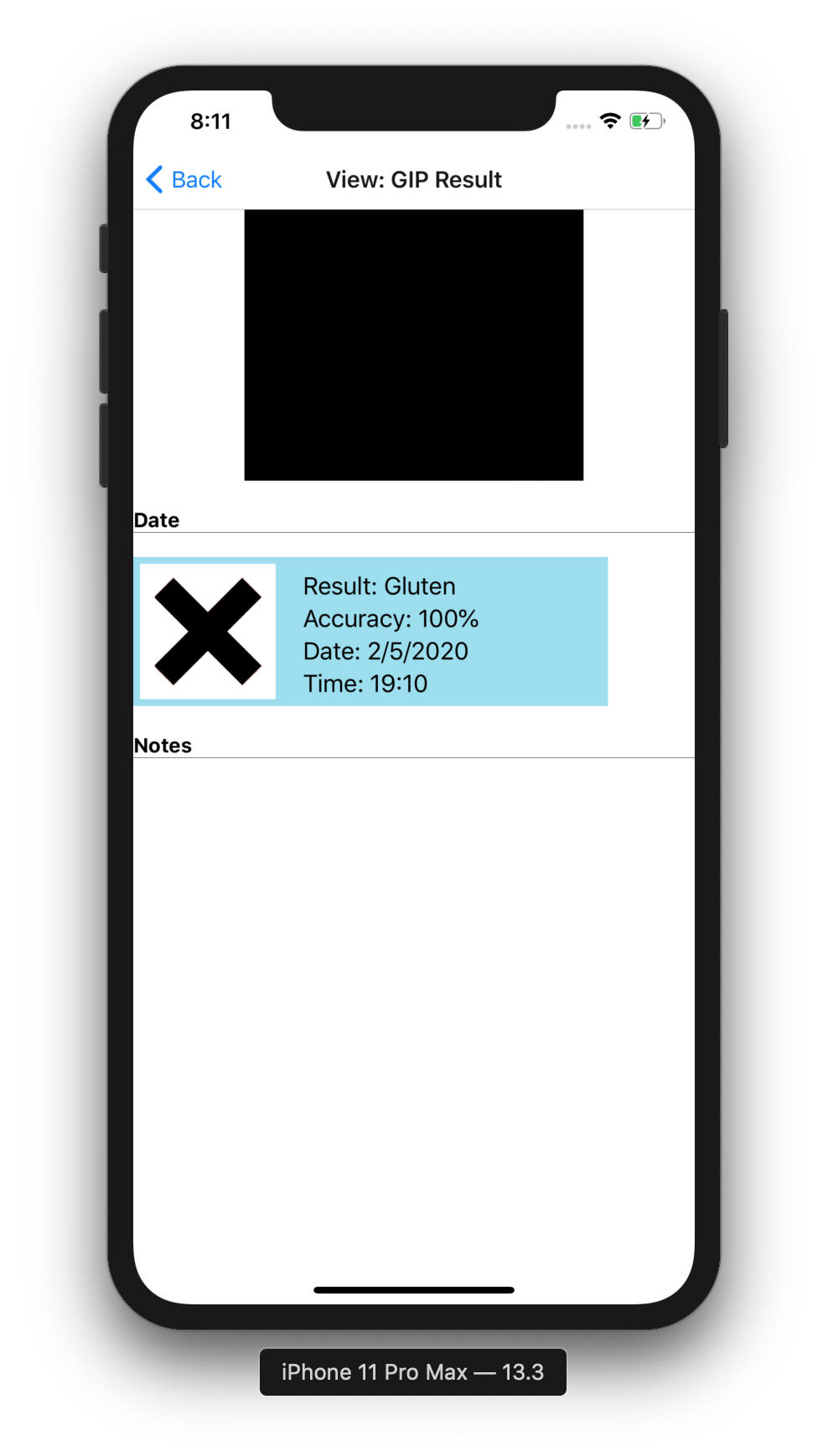
As part of my product working the Erasmus+ I also integrated some of my work and components into the Celiapp which was used during the 21 day challenge. The app was a collection of useful tools for Celiac patients to help them manage their health and diet. Some of the tools included quizzes to help teach which foods contain gluten and which did not. Helpful tips and a symptom tracker that allowed patients to add their symptoms and for easy view of the results.

In the list of available tools the GIP stick result display is the one I was working on. The home screen allowed the users to view their various recorded events and marked them on the calendar.



The user can record the result of a GIP stick and see it in their history.

Once they select into seeing their test result, the result is displayed within a ResultBlock component. Along with all the required information of the test in a concise form.



## Conformance to Specification and Design

The submitted product frontend does match with what I was hoping to achieve. An interface that is uncluttered and allows the user to easily access the functionality they want. However I was not able to implement the frontend into connecting with a server running the backend designed by my predecessor. The main issue I had with this was that the web app that was being hosted was not set up to return it’s result to the user.

The issue however is not overwhelming. With the benefit of hindsight I can see that a way to improve and access the model would be to remove the front end application of the webapp and have it directly respond to requests as this was causing conflicts that made it unable to be used.



## Learning Outcomes

In my work with the project I’ve learned a great deal about react through experience and research during the year. I do find that it’s a rather impressively flexibly API to use and I am confident that I would be able to use it to create an application or web app in the future for a different project.

Creating the front end of the application and working on integrating with the Celiapp taught me quite a bit about working within the bounds of older code and what has been written by people I can’t ask for explanations. There was some trial and error in fixing issues and adjusting code to match the rest of the app. This experience will be rather useful in the future with any project I find myself coming into partway through development.

I learned a lot about object detection during this project. Between research, testing the benefits of PyTorch and attempting to integrate the photo recognition backend into my frontend. I am now quite familiar with what steps are needed to create an object detection model with PyTorch and I am sure that I would be able to recreate the process on a different project.

But working on this project I find that I know much more about Coeliac Disease through working on this Erasmus + project than I did before. About the foods they can/cannot eat and many of the challenges they or the parents of coeliac patients face.

# 

# **Project Milestones**

1. **PyTorch Photo Recognition** (November 2019) - I wanted to have a PyTorch model that can recognise people and basic shapes to compare with Tensorflow.
2. **Early Layout** ( December 2019) - I wanted to have a mock up design for the front end of the app made for the Erasmus+ Meeting to allow for feedback.
3. **Layout Feedback and Improvements** (Early January 2020) - During the Erasmus+ meeting in Amsterdam I had time to get feedback on the UI layout and steps from graphic students along with help working React from more experienced developers.
4. **Integrated into Celiapp** (Early February 2020) - Added my components and work into the Celiapp for the Erasmus+ 21 day challenge.
5. **Basic Screens and Layout** (Late February 2020) - The initial react frontend completed. All screens set up, navigator set up. Camera screen accesses camera.
6. **Screens Completed in Detail** (Late March 2020) - After adjusting to the working remotely due to colleges being shut down due to quarantine I improved the React front end. All screens implemented and improvements were added. The result screen allowed for scrolling, the camera screen allowed for picture capturing and navigated to the result screen. New results were added automatically to the state.
7. **Final React Frontend** (April 2020) - Final improvements and adjustments based on feedback and fixes. Added a loading widget on the Camera screen while waiting for the picture to capture along with some cleaning up and adjustment of the code.

# **Results and Discussion**

My initial work into the designing the frontend ran into some questions namely how would the user control the interface. My initial plan of using swipes to control the interface had confusing results when I created paper mock ups and tested random volunteers. Some immediately began doing so but most were unclear as to what they could do. This was solved with the use of buttons which are definite in how they can be used. This changed how I planned to design the system.  
  
I decided to design the UI to be entirely operable using only one finger to touch the screen without moving afterwards in an aim to keep it as simple as possible. So transitioning from the home screen is a single button press, taking a picture of the GIP test is a single press and the user is automatically brought to the result screen.

This simple method worked but it was limiting in that the best option available in most cases was simply a button with a label. While this is enough for most purposes it is somewhat limiting.

Still the final result I received from people who tested the app was generally encouraging. All those who tested the app were able to complete the tasks asked of them and there was a marked improvement in speed in how things were accomplished.There was also a notable decrease in questions asked as most of the testers were able to work out the answers on their own. I believe this fits the target of easy, simple to use and user friendly UI. Assuming a working backend then the app is a success as it is very easy to go from start up to capturing test results successfully and easily.

Ways this could be improved involve minor animations and design assets. Colours on screen elements to group them together logically in the user’s mind and minor animations to draw attention to appropriate parts of the screen as required. Such as a flashing text input bar and so on.

# 

# **Project Review and Conclusions**

Overall I felt the project went well but could have gone better. The front end fits the vision I had for it but improvements could be made in appearance and features. React however proved to be a very useful tool for developing an Application that works on many different kinds of devices which I feel helped greatly in the 21 day challenge aspect of the project.

One of the main goals that I tried to keep in mind while working on the project is that the target audience would be teenagers and young adults. People who would be rather experienced with technology. That they would rather have an app’s functionality easy to use and quick to access the main function of an app than go through several layers to access the correct tool. So I tried to have the UI maintain a simple and clear appearance to bring the user to where they wanted to go quickly.

The project could have gone better in that it’s integration with the backend could have gone better. I have very little experience with networking, but with hindsight I can already think of ways to improve the issues I dealt with. I did set up my project to be flexible and easy to change with the intention that the backend was somewhat independent from the main interface so that models could be swapped out and improved without affecting the main function. The ideal scenario being that a future developer takes my front end and could then connect iit to their own server model to handle displaying data to users in a clear and simple fashion.

All that is required is that the model sends part of its results back to the client to have the correct results automatically displaying themselves.

I did quite a bit of research into the benefits of TensorFlow and PyTorch in the earlier portions of my project and I’m of the option that the better of the two depends on the nature of the project. Simpler systems or tasks which have been done before and don’t break too much new ground tend to work better with PyTorch. It’s easier to use with in-built useful APIs and more understandable logic. But TensorFlow while older does have an older and larger userbase, allowing for better documentation and experience to point in the right direction for projects that are new ground so to speak. The fact both are Python based does make integration into applications easier and less likely to cause conflicts.

Still following on from this I believe that given myself and my predecessor have created both a front end and a backend that any student following on from us might find it useful to take both parts and combine them into a whole basepoint from which to improve things.

# **References**

|  |  |  |
| --- | --- | --- |
| **Referenced Publication** | **Citation** | **Reference** |
| Report | Using Object Recognition To Read The Results Of Gluten Test Strips 2018 | Michael Bridgette, Using Object Recognition To Read The Results Of Gluten Test Strips, Final Year Research Report at IT Carlow |
| Website | (Torch Contributors 2019)    (Facebook 2019) | Torch Contributors. (2019). PyTorch.org. [Online]. (URL https://pytorch.org/docs/stable/index.html#). (Accessed November 2019).  Facebook. (2019). ReactJs.org . [Online]. (URL https://reactjs.org/docs/getting-started.html). (Accessed November 2019). |