# Functional Specification

Allert - Allergen Notification Tool

Conor Clerkin & David Moore

18773059 & 18722869

Completed 19/11/2021

# 0. Table of contents

# 1. Introduction

## 1.1 Overview

For people who suffer from various food allergies keeping track of which items, products or brands may contain allergens can be a difficult task. Depending on the severity of a person's allergies, cross contamination between products can be a serious concern, and challenging to keep track of.

Allert is intended to provide a user with various tools to keep track of their allergen data, by allowing them to scan product ingredient listings with their smartphones. Allert will automatically detect any allergens found in the listings and notify a user if it conflicts with their own history.

## 1.2 Glossary

| | |
|---|---|
| **AWS** | Amazon Web Services, a collection of over 200 cloud computing web services including computing, storage, networking, database etc. |
| **AWS Lambda** | An event-driven, serverless computing platform provided by AWS. |
| **OCR** | Optical Character Recognition, practice of converting images of text into machine-encoded text. |
| **PostgreSQL** | An open source database management system. |
| **React** | A Javascript library for developing cross-platform user interfaces and applications. |
| **Terraform** | Infrastructure as code tool allowing users to declaratively create cloud computing resources e.g. AWS Lambdas |

# 2. General Description

## 2.1 Product / System Functions

Allert will be developed as a mobile application for Android and iOS, making use of Electron for a cross-platform front-end, along with an AWS hosted back-end for user account data, making use of the AWS Lambda service for image processing. Users will be able to create an account and input their own allergen history and data. Using the app, they will be able to scan ingredient listings of various products; Allert will flag any known allergens if they conflict with the user's own history.

## 2.2 User Characteristics and Objectives

The user base of Allert will be made up of people from a wide range of backgrounds. Since allergies are not exclusive to any one age or gender the app must be intuitive to a diverse group of users. When considering this group it is important to remember that there will also be varying levels of expected expertise. The vast majority of users will be familiar with mobile devices and applications and will be capable of competently navigating through the application. Others however may not be as comfortable with such applications and it is therefore important to make Allert accessible also to these users.

Upon first installation of the application, the sign-in/sign-up screen should be familiar to those with prior experience. The user should be greeted with an option to sign-in or register a new account, with it being abundantly clear that registration is intended only for first time users.
The actual registration process should be as concise as possible. Users should be able to quickly input their details and select their relevant allergens. These allergens could be accompanied by a short list of foods containing the allergen or an image. This would be a design choice based on the varying levels of expected expertise.

Once sign-up is complete or a returning user has signed in, the application itself should be easy to navigate. Above all else the allergen detection feature should always be prominent and visible to the user. Given that this is the main feature of the app and in many cases the user will need to access this feature on short notice, it should be available from the majority of screens within the app. There should be minimal delay between selecting the feature and the camera opening and preparing to scan.

With the scanning functionality itself there are two priorities for the user; reliability and speed. Reliability is paramount within the context of allergen detection. The user should be able to trust that the scanner will correctly identify any allergens relevant to them every time. Speed is also an important characteristic of this feature. Users would like to receive the result in a timely manner in order to make a decision on whether or not to buy or consume the product in question. While it is important for the user to receive this result quickly to improve their experience with the app, accuracy and reliability should always take precedence.

Beyond this functionality the user should also be able to track any allergic events and the allergen in question. Pinpointing the date and cause of the reaction should be a process which is simple and repeatable given that many first-time users may want to input their previous events.

Users should be able to update their details, especially their personal allergens. It is possible to both develop new allergies and grow out of others as your immune system changes and this should be reflected in the Allert application. Users should be able to update any of their details after creating an account. With regards to the allergic events tracking, these events should persist within the user's account even after removing an allergen from their profile. Again with a focus on reliability, if a user adds a new allergen, their profile should be updated immediately to reflect this change. The scanner should therefore be able to recognise this new allergen and alert the user.

## 2.3 Operational Scenarios

### 2.3.1 Use Case 1 - Creating an Account

| | |
|---|---|
| **Goal** | The user creates a new user account. |
| **Preconditions** | The user does not have an existing account. |
| **Success Condition** | The user successfully registers an account, and can log in securely using their credentials. |
| **Failure Condition** | User fails to register an account. |
| **Description** | |
| **Step 1.** | The user opens the application. |
| **Step 2.** | The user is prompted with a login screen, and presented with the option to register a new account. |
| **Step 3.** | The user chooses to create a new account. |
| **Step 4.** | The user is prompted to provide their authentication details for a new account. |
| **Branching Actions** | |
| **Step 5a.** | The user receives feedback that they have successfully registered an account and returned to the login screen. |
| **Step 5b.** | The user receives feedback that their credentials are invalid and is returned to step 4. |

### 2.3.2 Use Case 2 - User Login

| | |
|---|---|
| **Goal** | The user logs in to the application. |
| **Preconditions** | The user has a preexisting account. |
| **Success Condition** | The user successfully logs in and gains access to their account. |
| **Failure Condition** | The user’s credentials are invalid and the user is not logged in. |
| **Description** | |
| **Step 1.** | The user opens the application. |
| **Step 2.** | The user is prompted with a login screen. |

| | |
|---|---|
| **Step 3.** | The user provides their account credentials. |
| **Branching Actions** | |
| **Step 4a.** | The user receives feedback that they have successfully logged in to their account. |
| **Step 4b.** | The user receives feedback that their credentials are invalid, or the account does not exist, and is returned to step 2. |

### 2.3.3 Use Case 3 - Providing Allergen Details

| | |
|---|---|
| **Goal** | The user selects the allergens they would like to be flagged when scanning ingredient listings. |
| **Preconditions** | The user has created an account and successfully logged in to the application. |
| **Success Condition** | The user account successfully registers allergen information to the database. |
| **Failure Condition** | The user account fails to register allergen information to the database. |
| **Description** | |
| **Step 1.** | The user selects the account settings option from the application settings menu. |
| **Step 2.** | The user selects the allergen flags option from the account settings menu. |
| **Step 3.** | The user is prompted with a checkbox list of 14 allergens. |
| **Step 4.** | The user selects the allergens from the list that they would like to be alerted of when scanning ingredient listings. |
| **Step 5.** | The user is prompted to save their changes upon exiting the allergen flags menu. |
| **Branching Actions** | |
| **Step 5a.** | The user chooses to save their changes, and their new details are saved to the database. |
| **Step 5b.** | The user chooses to discard their changes, and is returned to the main application menu. |

### 2.3.4 Use Case 4 - Scanning Ingredient Listings

| | |
|---|---|
| **Goal** | The user uses their smartphone camera to scan ingredient listings for allergens. |
| **Preconditions** | The user has successfully created an account and logged in to the application. |
| **Success Condition** | The user successfully takes a picture with their smartphone camera of an ingredients listing and is provided with information about the ingredients. |
| **Failure Conditions** | The user fails to take a picture with their smartphone camera. |
| | The application is unable to find ingredient information in the image the user has provided. |
| **Description** | |
| **Step 1.** | The user selects the scan ingredients option from the main application menu. |
| **Step 2.** | The user is prompted for the application to access their smartphone camera. |
| **Step 3.** | The user takes a photograph of the ingredients listing of a product. |
| **Step 4.** | The user is provided with information about the ingredients that the application has detected in the photograph, and notified of any allergens from the listing that are flagged in the user's account settings. |
| **Branching Actions** | |
| **Step 2a.** | The user declines the application's access to their smartphone camera and is returned to the main application menu. |

### 2.3.5 Use Case 5 - Receiving Allergen Alerts

| | |
|---|---|
| **Goal** | The user receives a notification that their scanned photograph contains listings of allergens that they have flagged in their account settings. |
| **Preconditions** | The user has successfully created an account, logged in to the application and taken a photograph of a product's ingredient listings. |
| **Success Condition** | The user successfully receives a notification that their scanned photograph does or does not contain any allergens flagged in their account settings. |

| | |
|---|---|
| **Failure Condition** | The user fails to receive any notifications. |
| **Description** | |
| **Step 1.** | The user takes a photograph of a product's ingredient listing using their smartphone camera. |
| **Step 2.** | The user is prompted with information of scanned ingredients and notified of any allergens listed that have been flagged in the user's account settings. |

## 2.4 Constraints

### 2.4.1 Time Constraints

The application will have to be developed and tested appropriately before the project deadline of 17:00 on Friday 22nd April 2022.

### 2.4.2 Hardware Requirements

The application is intended to be cross-platform and capable of being used on both iOS and Android smartphones. Development and testing of the application will have to account for both of these platforms by either making use of an iOS and Android phone for testing, or emulating these platforms.

The application will make use of a cloud microservice for performing optical character recognition on images the user has taken with their smartphone camera. For this reason the application will require an internet connection in order to be used.

### 2.4.3 Security

The application will make use of a user account system in order to store information about the allergens users would like to be notified of, along with a history of ingredients the user has scanned. This user account system will make use of the third party authentication tool "Auth0", allowing for the use of multiple secure sign-in options.

# 3. Functional Requirements

## 3.1 Optical Character Recognition

| | |
|---|---|
| **Description** | Users should be able to take a photograph of an ingredient listing on a product's packaging using a smartphone camera. Allert will perform optical character recognition on the image to determine if the ingredient listing contains any known allergens. |
| **Criticality** | This requirement is part of the core functionality of the application. |

| | |
|---|---|
| **Technical Issues** | Optical character recognition can be difficult to implement across different devices. As this is intended to be used with smartphone cameras, there may be inconsistencies between different camera qualities. |
| **Dependencies** | OpenCV optical character recognition library. |

## 3.2 User Account Creation

| | |
|---|---|
| **Description** | Users should be able to create an account within the application using an email and password login system. This account will be used to store user data, allergen information and product scanning history. |
| **Criticality** | This requirement is necessary for a number of key features, such as flagging known allergens conflicting with a user's data when scanning ingredient listings. |
| **Technical Issues** | Secure login systems can be difficult and time-consuming to implement correctly, and will most likely require a third party authentication system. |
| **Dependencies** | Third party authentication with "auth0". |

## 3.3 User Login

| | |
|---|---|
| **Description** | Users should be able to log in to a previously created account securely using their email and password combination. This account should store persistent information for the user, such as their allergen history and product scanning history. |
| **Criticality** | This requirement is necessary to make full use of other core features of the application. Allert will require user allergen history in order to flag known allergens in scanned ingredient listings. |
| **Technical Issues** | |
| **Dependencies** | User Account Creation; "auth0" authentication. |

## 3.4 Allergen Database

| | |
|---|---|
| **Description** | Both user and allergen data will need to be stored in a database in order to alert users according to the allergens they have flagged in their account settings. |
| **Criticality** | This requirement is necessary for one of the core functions of the application. A database is required to store user data in order to flag known allergens for users from their scanned ingredient listings. |

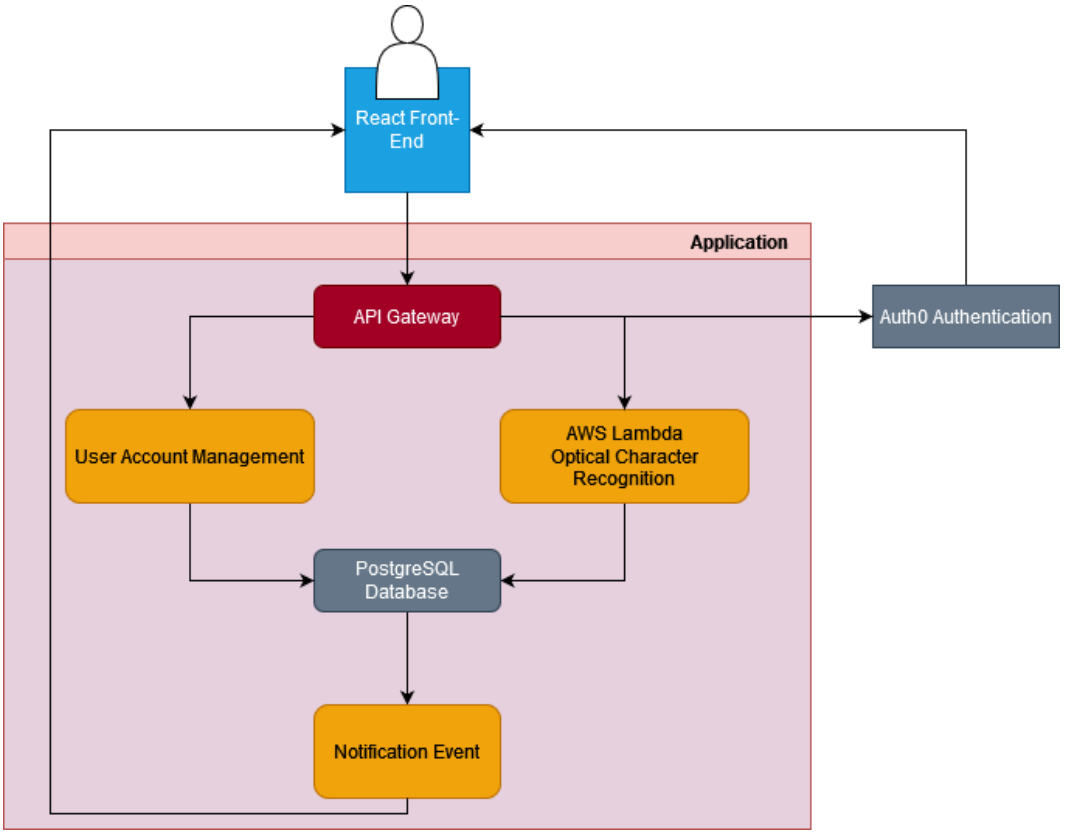
| | |
|---|---|
| **Technical Issues** | Creating and optimising this database will require storing allergen, product and user data. |
| **Dependencies** | User accounts. |

## 3.5 Cross-platform User Interface

| | |
|---|---|
| **Description** | The application should be accessible for both iOS and Android smartphones. This can be accomplished using cross-platform development tools such as React, which will allow us to create a single user interface to be used on either platform. |
| **Criticality** | This requirement is necessary for users to access the application. |
| **Technical Issues** | Cross-platform tools can speed up the development process of applications across different operating systems, however they can be inconsistent. Aiming for cross-platform accessibility can also add to testing requirements. We will have to be careful to ensure all of the application's features work correctly on all platforms. |
| **Dependencies** | Optical Character Recognition |

# 4. System Architecture

## 4.1 Basic Architecture Diagram



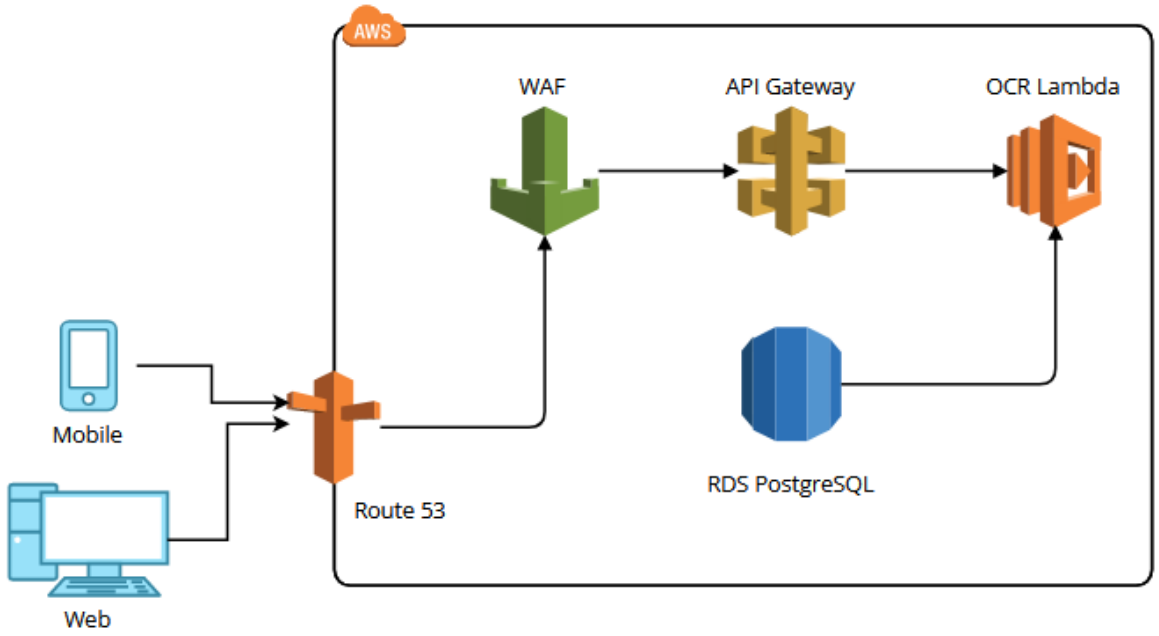
The application is intended to be a serverless solution, making use of the AWS Lambda service to perform optical character recognition on a user's images. Users will be able to access the service from a cross platform React client. User account authentication will be handled using Auth0.

Once successfully logged in, a user will be able to take a photograph using their smartphone camera, and trigger a request to the AWS Lambda service to run an optical character recognition function. The result of this will be written to our database as part of the user's scanning history. The results of up to 10 previous scans will be stored for the user.

Changes to the database will trigger a notification event. In the case that the scanning history contains any allergens that the user has flagged in their account settings, they will receive a notification in the front-end application.

By storing the user's scanning history we can also inform the user of any allergens within their history as part of the same notification event, should the user ever change their allergen flags in their account settings.

## 4.2 AWS Architecture Diagram




Following on from the previous section, this diagram details the setup of the AWS portion of the system architecture. As mentioned, AWS will handle the Optical Character Recognition functionality.

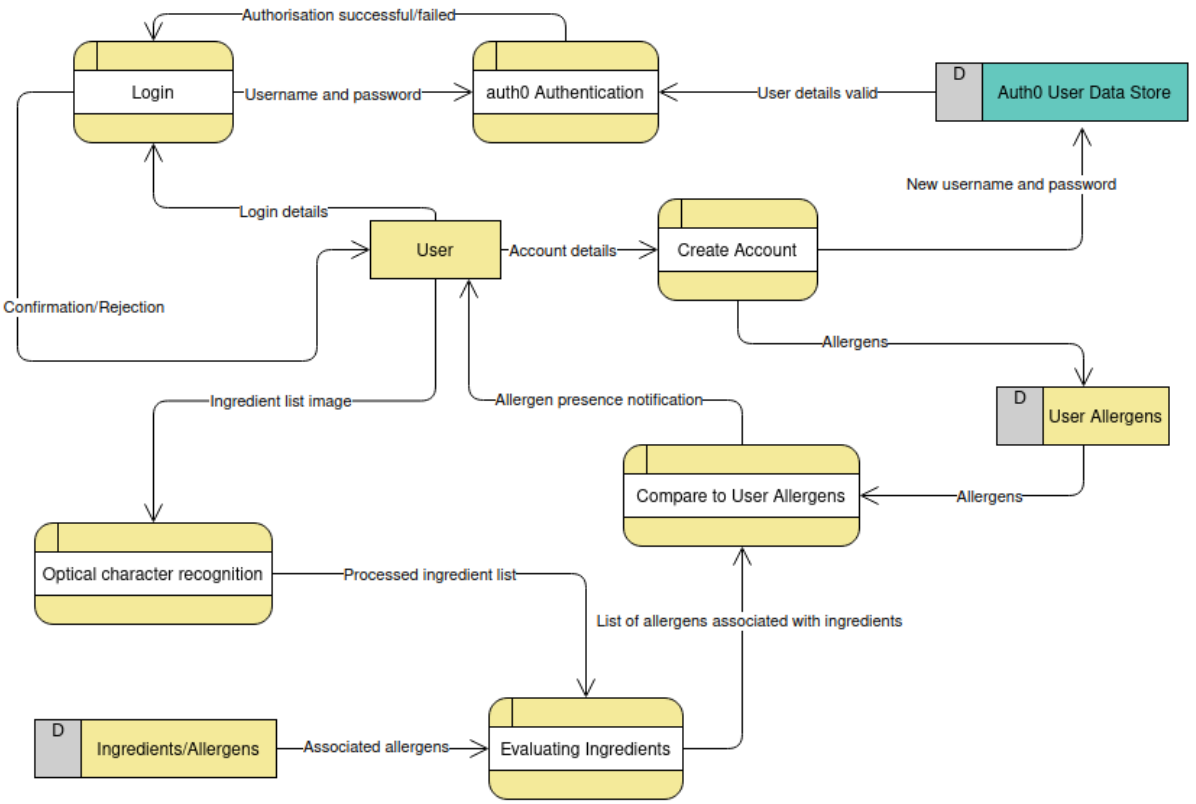
To achieve this, Route 53 leads to WAF which will provide a firewall in order to improve security. There is then the possibility of either using API gateway or moving straight to our

Lambda function. API gateway is industry standard however routing directly to Lambda and skipping this step works just as well but does not allow the service to be called as a REST API. Regardless, the lambda will take care of OCR, identifying allergens related to these ingredients and then matching these allergens to user allergies. The RDS PostgreSQL database will contain user, product and allergen information which will be used by the Lambda function in its various tasks.

This infrastructure will all be managed with Terraform, an infrastructure as code tool which will allow us more visibility as we create our infrastructure. We will be able to push this code to our project Git repository and this will add to our documentation about our AWS environment.

# 5. High-Level Design

## 5.1 Data Flow Diagram

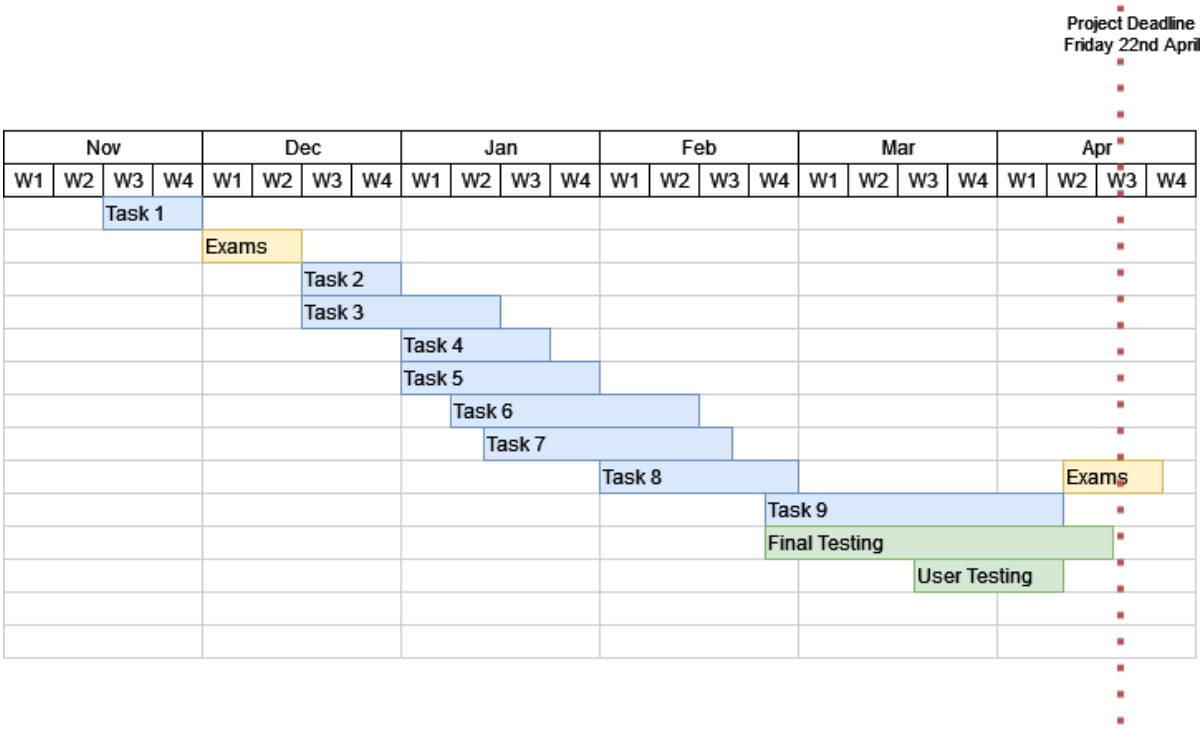



Above is a Data Flow Diagram detailing the flow of data from the user logging in or creating an account to performing an ingredient scan. Login details are passed to the login process which in turn passes these on to the Auth0 authentication process. This process checks an external data store hosted in the cloud by Auth0 for these user details. The Auth0 authentication process then returns the result of authentication to the login process which in turn notifies the user of confirmation or rejection of their login details.

If the user chooses to create an account they send their details to the create account process. From here the details are sent to two different data stores. The username and password is sent to the previously mentioned Auth0 user store while the user's allergens are sent to a data store containing the user's allergens.

The user can send data in the form of an image of an ingredient list to the OCR process which then outputs the text of the processed image to a process which evaluates the ingredients. This process queries each individual ingredient against the data store of allergens associated with ingredients before sending the final list of allergens contained within the product to another process which compares this list with the list of the users allergens. This data is received from the data store created upon account creation. Once this process has concluded a notification will be sent to the user notifying them if any matches have been found.

# 6. Preliminary Schedule




| | |
|---|---|
| **Task 1** | Researching and finalising plans for the application; which tools will be used for development and what will need to be set up before development can begin. |
| **Task 2** | Initial React application set up. |
| **Task 3** | Initial setup for optical character recognition with Python OpenCV |
| **Task 4** | PostgreSQL database setup. |
| **Task 5** | Integration for unit tests of cross-platform builds. |

| | |
|---|---|
| **Task 6** | User account and authentication implementation. |
| **Task 7** | Integration of optical character recognition with AWS lambda |
| **Task 8** | Integration of React front-end with AWS microservices |
| **Task 9** | Finalise setup for cross-platform builds of the application. |

# 7. Appendices

1. **React - https://reactjs.org/**
2. **PostgreSQL - https://www.postgresql.org/**
3. **OpenCV - https://opencv.org/**
4. **AWS Lambda - https://aws.amazon.com/lambda/**