|  |
| --- |
| Hampton School |
| Smart Chef |
| A2 – Computer Science Coursework |

|  |
| --- |
| Raaed Ali |

Contents

[Analysis 2](#_Toc36822299)

[Project Background Definition 2](#_Toc36822300)

[Prospective Users 4](#_Toc36822304)

[Interview with Primary Client – Antonio 4](#_Toc36822305)

[Acceptable Limitations 5](#_Toc36822306)

[Proposed Solution 5](#_Toc36822309)

[Objectives 6](#_Toc36822311)

[Data Dictionaries 7](#_Toc36822312)

[Data Source and Destinations 10](#_Toc36822316)

[Time Complexity 10](#_Toc36822317)

[Design 11](#_Toc36822318)

[Critical Path Diagram 11](#_Toc36822319)

[System Design 12](#_Toc36822320)

[Modular System Design 13](#_Toc36822321)

[Completed Dishes Popup 13](#_Toc36822322)

[Get Various Dish Information Popup 14](#_Toc36822323)

[Algorithms 15](#_Toc36822324)

[Definitions 19](#_Toc36822330)

[Implementation 21](#_Toc36822333)

[Main Window Screen 21](#_Toc36822334)

[Food Options Window 26](#_Toc36822337)

[Dish Information Menu 28](#_Toc36822340)

[Ingredients Menu 32](#_Toc36822343)

[Instructions Menu 34](#_Toc36822346)

[How to Run the Program 39](#_Toc36822352)

[Testing 39](#_Toc36822353)

[Tests 39](#_Toc36822354)

[Evidence 50](#_Toc36822355)

[System Tests 63](#_Toc36822381)

[Evaluation 68](#_Toc36822386)

[General Appraisal 68](#_Toc36822387)

[Meeting Objectives 68](#_Toc36822388)

[End-User Feedback 70](#_Toc36822389)

**Smart Chef**

# Analysis

**AIM:**

**“To provide an app for users to find recipes for dishes they want to make, use an in-built timer, and save/view the dishes they have previously completed”**

## Project Background Definition

The idea for this app was formed when I was looking online for a recipe to make a lasagne one Sunday morning. I spent 5 minutes to search the web online and eventually found a recipe that I wanted; however, it was obvious that this task could and should have been done in much less time. Therefore, I set out to try and make an app that I can use to find dishes and recipes with in almost half of the time.

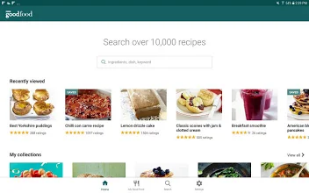
Searching for recipes online can be an incredibly difficult task, with often many people searching multiple websites to look for the perfect recipe, after which they have to follow the inconveniently worded and made instructions bit by bit, ensuring that the recipe is not only well rated, but also acceptable for different diets.

My app idea, Smart Chef, is a convenient way for users to find recipes for dishes. It will provide the user with the option to choose between two sources for recipes (BBC Good Food and Baking Mad Recipes), and then with a given query will search the respective source for various recipes of the dish the user desires, it will allow the user to choose from the many dishes images shown, and then provided a step by step guide for what ingredients the user needs and each and every step the user needs to take in order to create the dish, alongside a built in timer for when the user needs to wait for dishes to cook in the oven and all in the easy of an android app. The user can also search up ingredients in the query bar instead and the app will work to find the write dish for the given ingredient.

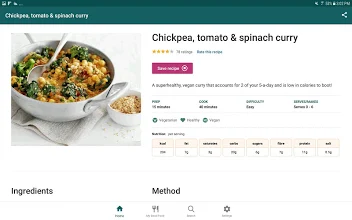
## Observation of existing system’s

Right now, there are two main apps that are currently doing similar functions to that off the app I intend to create. The first one is the BBC Good Food App, and the second one is the BuzzFeed Tasty App.

### BBC Good Food App

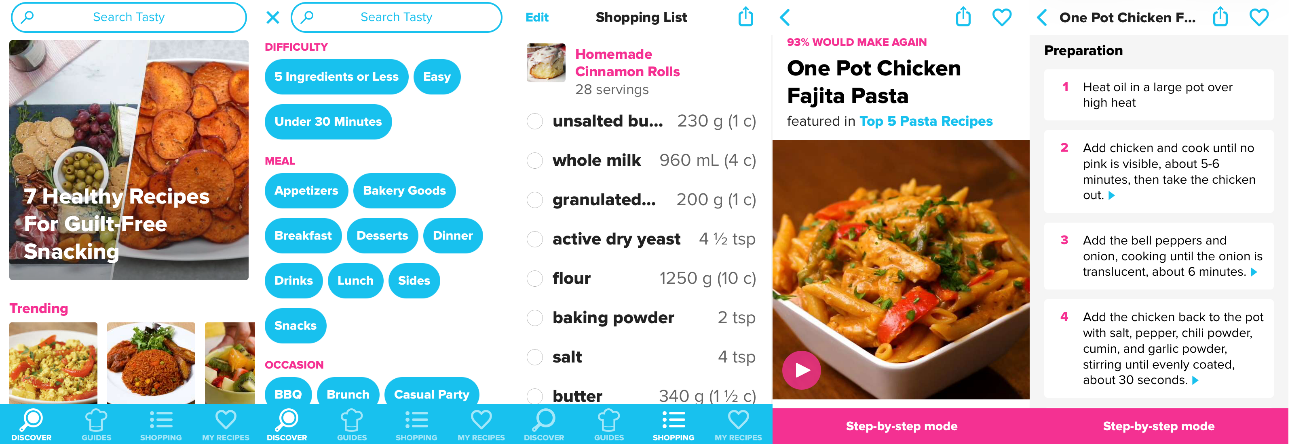
When the user enters the app, this is what they will see. The app has a nice interface with an obvious text bar and suggested dishes below.

This App is good in the ease and convenience of finding new dishes, as well as how the dish information is portrayed before the user needs to select the dish, and how the dishes each have individual ratings provided in a simplistic star layout. Furthermore, the ability to have a “difficulty rating” system within the app means that users can more accurately predict whether the dish is the right one for them to make.

However, the problems with the current system is firstly the lack of being able to use a timer within the app. This means that the timer will have to be accessed outside the app and therefore is inconvenient for the user, especially when this app is created with the purpose of convenience in mind. Secondly the app has no way of keeping count of the dishes the user has made, which could then be used in multiple areas, such as modifying the suggested dishes to have a little more relevance to what type of dishes the user likes to make, or just for letting the user remember what they have made beforehand.

Therefore, this app is different to the one I plan on making, as my app will have a built-in timer and will be able to store the previous dishes the user has made. Furthermore, the fact that I am using BBC Good Food as one of the sources of dishes means that my app will have more options available for the user to select dishes from, as it will incorporate every single BBC Good Food dish alongside another different source for dishes (Baking Mad Foods).

### BuzzFeed Tasty



The BuzzFeed tasty app is like the BBC Good Food app in the sense that both apps have similar limitations and positives. What makes the Tasty app stand out more however, (with almost 1 million downloads on the Google Play Store) is the ability to have personalised recommendations based on what sort of dishes the user likes. This means that the user is more enticed into using BuzzFeed Tasty if they are looking for dish choices as well as recipes for dishes.

However, the BuzzFeed Tasty App still lacks a timer function and has a lower search range for dishes than the app I will be designing is predicted to have.

## Prospective Users

This app is designed primarily for those who are looking for a simplistic app they could use instead of searching online for dishes, their recipes and the instructions. As such, the target audience could be anyone who has a passion for cooking or just wants simple instructions on how to make a simple dish. Furthermore, the simplistic design of the code should mean that most of the general public should be able to read / use the app with ease.

## Interview with Primary Client – Antonio

In order to understand what the typical user would want; I conducted an interview with an active cooking enthusiast. Where I asked many questions about what really stands out/ matters in a good app.

**Q1:** *What do you look for in a recipe app?*

**A1:** I am looking for a minimalist UI as simplicity is useful. Animations are necessary to make everything smoother, like transitions. Furthermore, I want stylishness, the UI needs a colour scheme/ icon that makes sense.

Analysis: Antonio has indicated that it would be better to prioritise minimalistic ideas to prevent clutter, as well as a smooth interface

**Q2:** *How often do you go onto a recipe app with the intention of making a random dish recommended to you by the app?*

**A2:** Never, I don’t think many people do this at all.

Analysis: There is little need for there to be a recommended dishes section to my app (although I will add it if there is time)

**Q3:** *Does the ability to store your previously saved dishes sound useful?*

**A3:** Yes, as I can easily find a recipe, I have already done.

Analysis: Storing previously saved dishes is useful and acts as a key differential factor for my app

**Q4:** *Do you prefer the instructions laid out on one page or multiple step-by-step pages with buttons between each page?*

**A4:** I think it would be easier with all the ingredients on one page, simply because changing pages with dirty hands while cooking is a nuisance.

Analysis: It is important to think about real-life applications of my app, which in this case would mean that it is more convenient to have all the instructions on one page

**Q5:** *What app do you use right now and are there any problems with it?*

**A5:** No app, which means that I have imprecise measurements and it just takes longer for me to find a dish he is looking for. Especially as most of those websites have these annoying essays about the dish before the actual useful information comes up.

Analysis: Antonio indicated that it is important for my app to be a lot easier to use and to prioritise the useful information when designing the app. This is in order to save time for the users and so that the users have an overall better experience.

**Q6:** *How important is the app’s interface if it gets the job done?*

**A6:** Interface is very important in an oversaturated market, in order to stand out I think the interface needs to look fresh and easy to use

Analysis: Antonio suggests that it is better to prioritise the interface when designing an app, but only as a point of competing against others.

To conclude, they key points I got from this interview that I need to write about are:

* Minimalistic Interface
* No need for recommended dishes section within the app
* Have an option to view/store previously completed dishes
* Have all the instructions shown on one page

## Acceptable Limitations

There are a few limitations that my code will have in comparison to what the code could have due to the following reasons:

### Time Limitation

In order to complete my code by around the time of the deadline, I may have to sacrifice certain elements such as interface aesthetics and looks of the overall app.

### Kivy Compatibility

I will be learning the kivy language from scratch, therefore I may be using techniques to make the app that are less efficient than techniques someone with more knowledge of the app could have used.

## Proposed Solution

Having analysed the users recommended requirements for what the app should have, I have decided that it would be best to use the kivy interface for python. Kivy is a python module which allows easy interface for android apps, while allowing for the code to be coded in python (usual Android apps are created in Java).

### Use of Kivy

In order to make my app, I think that research is the best for multiple reasons. Firstly, kivy is a free application which has IDLE integration, so the error checking is easy to understand and use in the IDLE editor.

Secondly, kivy has a built in app interface that allows users to import both App and Kivy into the python code, and then use the code that was made using the classes within App and kivy to then show the app on the android app store (given a little tinkering to the code). Which means that it is easier for me to pass the kivy code into an app.

Finally, Kivy comes with a built in Screen class which allows the designer to implement different screens easily within the code. The screen class also comes with a .jv page where the designer can separate the code from the layout, allowing the designer to change the layout independently from the code.

However, the kivy language also has a couple downsides to its use, such as the fact that ios interfacing with the kivy language is very difficult and hosting an app on ios costs anyways, therefore I will be unable to add support for Apple devices when making the app.

## Objectives

1. The app starts off with a main menu, where the user has a text input area and five buttons to choose from:
   1. The first button will be the start button
   2. The second button will be one of two toggle buttons, one which has the BBC Good Food toggle button, and will start off selected
   3. The third button will be the second of two toggle buttons, the other of which is the Baking Mad Toggle button
   4. The fourth button is a “show completed dishes” button
   5. The fifth button will be a quit button
2. If the “start button” is pressed and there is no input, or an input that returns no dishes, with the BBC Good Food toggle selected then a popup should appear saying that there are no dishes available, and a button that lets users return from the popup
3. If the “start button” is pressed and there is no input, or an input that returns no dishes, with the Baking Mad Foods toggle selected, then it should show a recommended page of dishes.
4. If the quit button is pressed, then the program ends
5. If the start button is pressed, and there is a valid input, then the screen is changed to one with a list of all the options
6. The options screen will be presented with a list of all the buttons of options, with a scroll bar on the side
7. The options screen also has a quit button at the bottom, which will return the user to the menu when pressed
8. If the user selects a dish option, then they are sent to a page with four titles of information, that being:
   1. Whether there is a vegetarian option
   2. The amount of people the dish serves
   3. The time it will take for the dish to cook
   4. The time the dish will take to prepare
9. This page will also have two buttons, a back button (that sends the user back to the options of dishes page) and a go forward button
10. When the go forward button is pressed, the user is sent to the ingredients page
11. The Ingredients page will show all the ingredients needed, and have two buttons (one of which will be a go back button and one of which is a continue button)
12. If the user clicks the go back button, they will go back
13. If the user clicks the continue button, then the user is sent to the next page (the instructions page)
14. If the user is on the instructions page, then the user is given a list of instructions needed to complete, alongside two buttons. A timer button and a finish button.
15. If the user presses the timer button, then the user is sent to a screen which has a selection of times in a scroll bar.
16. Once a time is selected a popup will start saying the timer has started and a timer will start for the given minutes, and then the popup will have an exit (the popup) button
17. If the user selects the finish button, then the user is sent to the main page to make another decision, and the dish the user made is stored in a text file.

## Data Dictionaries

Currently, the data being stored/taken is:

* The dish queries
* The completed dishes
* The various dish options
* The dish ingredients
* The dish instructions

### Level 0 Data Flow Diagram

A screenshot of a cell phone

Description automatically generated

### Level 1 Data Flow Diagram

A screenshot of a cell phone

Description automatically generated

### Overall Data Flow Diagram

A picture containing meter, parking, parked

Description automatically generated

As seen in the diagrams, the only way for the user to quit the program is in the main menu screen. Therefore, the user would need to return to the screen if wanting to quit. This is easy for the user to do however and there is always a back (or finish) button for the user to use on every single page for convenience.

## Data Source and Destinations

For my code to function, the app requires certain data:

* The user’s query
* The ingredients scraped from one of the two websites used (BBC Good Food/Baking Mad)
* The instructions required for each ingredient (from the same websites)
* The completed dishes

The ingredients and instructions will be screen scraped from whatever website the user selects, the choices being BBC Good Food or Baking Mad Dishes. However, this means that although the app will be compatible for the Google Play Store. I will not be able to upload it onto the play store as I will be scraping information from other websites.

## Time Complexity

This app does not involve much time complexity. However, scraping the information from the websites will be what will take the longest time, alongside uploading all the information onto the relevant screens. This is because the code for each screen (that being the screen with all the ingredients, or the screen with the instructions to make the dish) will only load once the user enters the screen. Therefore, upon entering each screen there will be a short time delay as the information is gathered for the specific screen. The information however is kept on the screen once it has loaded once, therefore the back buttons on each screen should take a lot less time to perform their actions as it would be going back to a pre-loaded screen (unless the dish query changes).

# Design

## Critical Path Diagram

A picture containing clock

Description automatically generated

This Critical Path suggests that I should be finished with my code in around 12 weeks. However, some tasks may take more/less time than expected, especially when trying to implement measures such as convenience alongside the fundamental aspects of what each page should do. An example of this is my idea to use multithreading when creating the timer, in order to allow the user to search continuously as the timer is ongoing.

I could decrease the time taken for the code by removing the optional features I have highlighted in red lines, that being the user-friendly interface and colour scheme; which isn’t mandatory to the apps functionality, but is recommended based on what the interviewee said earlier.

The text highlighted in red is non-critical and can therefore be ignored if time does not allow for it.

## System Design

In order to efficiently create my Smart Chef app. I should thoroughly design what the program will look like specifically and how it will function. This aids the object-oriented programming style I will be using when creating the app.

Firstly, the app will open on the main Menu. This main menu will have five button all lined up below a text input, and above the text input will be text telling the user to input a dish query they want, toggle the website they want to use, and some rules about how the app will handle the query. The five buttons below the text-input will be:

* The quit button, which will quit the program.
* The two toggle buttons, that the user will have to choose from (with the BBC Good Food being toggled on initially)
* The search button, for when the user has input a query and has chosen a query
* A search previously completed dish, for the user to use to view a pop up which will show all his previously completed dishes, alongside a button that once pressed will close the popup

If the user has pressed the search button and either has no query or the query returns no dishes, then a popup will be displayed telling the user that the query returned no dishes and a button to close the popup. However, if the user has input nothing and has selected the Baking Mad toggle, then the user will be sent to a page of recommended dishes to use.

Once the user has input a query that has dishes available and has selected one of the toggles, the user will be sent to the next page, which is a list of dishes that the user can make, based on the user’s input from the last screen. This page will have a list of dishes the user can make, alongside a go back button which will send the user back to screen one and a scrollable bar for the user to scroll if the list of dishes is too much to fit on the screen. Once the user selects a dish, they will be sent to the initial information page.

The initial information page will display the prep time, cook time, dietary information (vegetarian or not), and how many people this dish will serve (or the size of the dish in some scenarios). This initial information will be laid out in a grid layout and there will be a forwards and back button. If the back button is pressed, then the user is sent back to the page with all the dish options. If the forwards button is pressed, then the user is sent to the page will the ingredients required to make the dish.

The page with the ingredients required to make the dish will also display (alongside the ingredients required to make the dish) a back and forward button. If the back button is pressed then the user is sent back to the initial information page, whilst if the forwards button is pressed then the user is sent to a page with a list of instructions for how to make the dish.

The instructions page will display the instruction and will have a timer and finish button. If the finish button is pressed, then the program will store the completed dish and send the user back to the main screen, where they can decide to exit. If the timer button is pressed, then the user is sent to a screen which will display a selection of minutes (in intervals of 5) that being a scroll bar with several minutes that the user can select. Once the user selects the number of minutes, they want to the timer to run for, then a popup will appear telling the user that the number of minutes has been set and the user can select a button that will close the popup, and put the user back onto the instructions page.

## Modular System Design

This is a level 0 overall system design for the code

A screenshot of a social media post

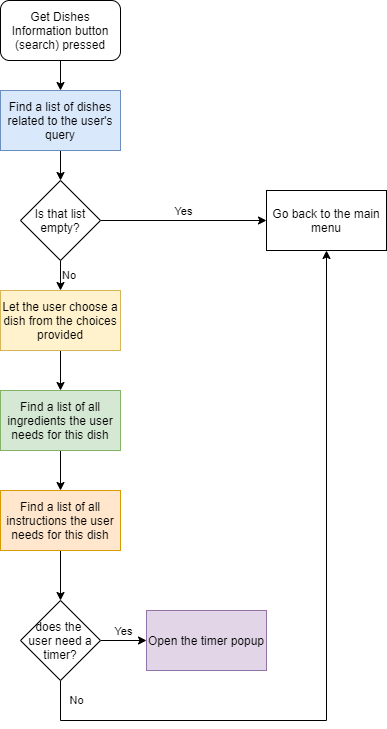
Description automatically generated

## Completed Dishes Popup

A close up of text on a black background

Description automatically generated

## Get Various Dish Information Popup



\*the colour of the box relates to the highlighted algorithm below

## Algorithms

### Find a List of dishes related to the user’s query

query = userinput

def list\_of\_dishes(self, query, url):

    page = urllib.request.urlopen(url)

    soup = BeautifulSoup(page, 'html.parser')

    content\_list = soup.get\_text()

    content\_list = soup.find\_all('h3')

    return content\_list

toggle = userinput

BBC = "https://www.bbc.co.uk/food/search?q=" + query

bakingMad = "https://www.bakingmad.com/search-results?searchtext=" + query

if toggle == "down":

     content\_list = self.list\_of\_dishes(query, BBC)

else:

     content\_list = self.list\_of\_dishes(query, bakingMad)

This algorithm, gets a query from the user (all user inputs have been highlighted in black) and then uses that query to screen scrape the “h3” tags from the page with the URL, which is defined by which ever button the user selects. The toggle label is defined by the user input, when they choose either BBC Dishes or Baking Mad, and if toggle is BBC Dishes then toggle is set to “down” otherwise toggle is set to “normal.”

The “h3” tags are used as both webpages have all of their dish names set to “h3” and therefore if I use a screen scraping module called Beautiful Soup to find only the h3 tags then it makes it easier for me to find all the dishes.

I have defined BBC and baking Mad as the website which is shown when the query is input into the query box that the individual websites have, for example if I searched up “cake” on BC Dishes website, I would be redirected to a list of cakes under the website name:

<https://www.bbc.co.uk/food/search?q=cake>

A screenshot of a social media post

Description automatically generated

### Let the user choose a dish from the provided choices

Choices = list of dish choices from the algorithm above

for txt in choices:

btn = Button(text=txt, size\_hint\_y=None, height=80, valign='middle', font\_size=12)

      btn.text\_size = (btn.size)

      btn.bind(on\_press = self.find\_dish)

      btn.bind(on\_release = popup.dismiss)

closebutton = Button(text='I want to make a different dish', size\_hint=(0.9,0.05))

closebutton.bind(on\_press= popup.dismiss)

closebutton.bind(on\_release = self.change\_page)

scrlv = ScrollView(size\_hint=(0.9,0.95))

slid = Slider(min=0, max=1, value=25, orientation='vertical', step=0.01, size\_hint=(0.1, 0.95)

The first for loop, goes through the list of choices created from the previous algorithm, and then for each choice creates a button, where the text of the button is the text of the dish, and the height and font size are set. Valign will ensure that the text is vertically aligned to the middle of the button, and size\_hint\_y will ensure that the y axis of the button can be changed depending on the size of the screen. Button text size works to ensure that the text of the button changes depending on the buttons size, which can be changed depending on the screen size. Furthermore the button is then binded such that when pressed, they run the find\_dish function which is set to get the button pressed, get its text input and set that to the dish being made, then the button changed to the next screen. And once the button is released the popup with all the dish options is closed.

The close button is made so that the user has the option to dismiss the popup with all of the dish options and then once the button is released, it runs the change\_page() function which send the user back to the menu screen. The scroll bar is implemented using ScrollView, which is a class inside of kivy. The size hint is the percentage of the screen which the scroll bar will use, with the two numbers being the x and y percentage and 1 being 100%, so in this case the scroll bar will use up 90% and 95% of the remaining x and y coordinated of the screen respectively. The slid variable then defines the functions of the scrollView by creating the physical scroller. Which then has definitions such as its orientation and what percentage of the screen is moved per step (the step=0.01 line). The value and min/max is a constant set to how many variables (in this case buttons) can fit on one screen at a time, and min/max is the percentage of the screen that can be covered in the buttons that are the dish options (this being 0% to 100%).

### Find a List of all ingredients needed by the user

1

ingredients\_list = self.ingredients()

ingredient\_string = ''

for ingredient in ingredients\_list:

      ingredient\_string += ingredient + '\n'

ingred\_list = Label(text=ingredient\_string)

2

def ingredients(self):

      url = full\_link

       r = requests.get(url)

      data = r.text

      soup = BeautifulSoup(data, “html.parser”)

      ingredient\_list = []

ingredient =  soup.find\_all(‘li’,attrs={“class”:”recipe ingredients\_\_list-item”})

      for a in ingredient[0:]:

          result = a.text.strip()

          ingredient\_list.append(result)

1

Firstly, the ingredients are got from the ingredients function (analysed below), however any label cannot use a list as its text, therefore I will append all the items of the list into a string, followed by a “\n” which starts a new line. Therefore, the label I add will have the text of all the ingredients added together in a neat row.

2

The ingredients function essentially gets the “full link) from another algorithm. The full link being the link of the that the dish is on (e.g. <https://www.bbc.co.uk/food/search?q=cake>) And then gets the html from the page using Beautiful soup. The html is then filtered to get the text within the “li” title (html lists) and then the “li” titles with the class name “recipe-ingredients\_\_list-item”. This class name was something I had to search for by inspecting the element of the page and then looking for what title the dishes were under. The text is then stripped from the html code and then appended to the list of ingredients, so that I can use that list to put into the label above.

A close up of a logo

Description automatically generated

The inspect element of “honey cake” on the BBC dishes page shows the “li” titles

### Find a List of all Instructions needed by the user

url = full\_link

r = requests.get(url)

data = r.text

soup = BeautifulSoup(data, "html.parser")

instructions\_list = []

if toggle == "down":

instruction = soup.find\_all('li',attrs={"class":"recipe-method\_\_list item"})

if toggle == "normal":

     instruction = soup.find\_all('div',attrs={"class":"method\_\_text"})

     for a in instruction[0:]:

           result = a.text.strip()

            result = str(result).replace(".","\n")

            instructions\_list.append(result)

This instructions string is very similar to the method used to get the ingredients from the same web page. However, I found out the different websites use different names to define the instructions lists that they have, therefore I made it such that if toggle == “down”, i.e. if the user selected BBC dishes earlier on in the code, then the instructions search the html for the list (“li” with the class name “recipe-method\_\_listitem”. Otherwise, if the user selected the BakingMad link, then the class name searched for is “method\_\_text”. Then for every item found in the list of html code. The text is stripped from the html that came alongside the “li” tag. Then in order to prevent the line within the instruction to overfill the width of the screen, I replaced all the full stops with a “\n” so the instructions go onto a new line instead of overfilling the screen and not being seen. Then the result is added to the list I have defined as “instructions\_list.”

A screenshot of a social media post

Description automatically generated

This is the “li” class name for the method on BBC Dishes, which is different on the Baking Mad website

A screenshot of a cell phone

Description automatically generated

### Timer Popup

thread = threading.Thread(target= self.threading\_timer)

        thread.start()

def threading\_timer(self):

        time.sleep(count\_num)

        popup = Popup(title="Countdown Finished!")

        lbl = Label(text="Your countdown is complete",font\_size=20)

        btn = Button(text="Close?")

        btn.bind(on\_release = popup.dismiss)

This is my simplistic idea behind the timer, I will have multithreading imported as a module in Python, and then using that multithreading I will have the timer set as solely a time.sleep method for the count\_num (count\_num being the minutes that the user will have set beforehand). Then once the time has ended a popup will open telling the user that the time has finished and then a button that will allow the user to press and then close the label that was opened. This is best as it allows the user to do virtually anything they would like to do with the app as the timer is running, and the timer will still run provided the app is open. For example, they could look at other dishes or even set another timer as the initial timer is running.

## Definitions

### Global Variables

Having researched kivy, It turns out that one cannot pass class variables from one class to another, due to the fact that the screens overlap and the entire app crashes if there is overlap in screens. However, I need different classes for different screens, but the screen will need information from one another, therefore I had to set a couple global variables. I have tried to minimise the number of global variables in the overall code however there are six

|  |  |  |
| --- | --- | --- |
| Class Name | Variable Name | Function |
| Global | query | The user input for what type of dish they want |
| toggle | Either “down” or “normal”, is “down” if the user selects the BBC Dishes toggle button, and “normal” otherwise |
| choices | The list of dish options depending on the query |
| dish\_choice | The users dish choice from the list of dishes |
| full\_link | The full link for the website + query the information is going to be accessed from |
| count\_num | The seconds equivalent of the minutes the user chose when creating a timer |

### Subroutines / Functions

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Subroutine** | **Function** |
| MainWindow | clean | Checks to see if there is a ‘>’ in the string for the content\_list |
|  | old\_dishes | Opens the popup with all the information regarding the previously completed dishes |
|  | list\_of\_dishes | Will scrape the websites given to create the list of dishes that the user can then select from |
|  | list\_of\_websites | Will scrape the websites given to create a list of links that correspond to the dishes found in the list of dishes (not used in the current version of the app, but kept fur future reference) |
|  | choices\_list | Cleans the html list of websites, and removes all the information that is not about the dishes |
|  | pressed | The function run when the start button is pressed, that checks if the list of dishes made is 0, and if it is it informs the user, else returns the dishes list of choices |
| OptionsWindow | on\_enter | Run when the user enters the OptionsWindow class and opens a popup with a list of buttons of all the options for dishes the user must choose from, and a back button to go to the main screen. |
|  | change\_page | Changes the current screen that user is viewing, back to the main one |
|  | scroll\_change | Updates the scroll value of the scroll tab |
|  | slider\_change | Avoids the maximum recursion depth error |
|  | find\_dish | Gets the user’s dich choice and changes to the next screen (DishInformtion Screen) |
| DishInformation | on\_enter | When the user enters this screen, then a popup open showing all the initial information of the screen |
|  | get\_websites\_mad | Used only if the user selects BakingMad toggle, this function finds the link for the dish the user has selected in the previous screen |
|  | get\_websites | This is a mirror of the above function^^ used only for the BBC Dishes function and returns a list of websites instead that accompanies the list of dishes |
|  | go\_back | Changes the screen to the previous screen (OptionsWindow) |
|  | list\_matcher | Gets the full link if the user selected the BBC toggle button |
|  | initial\_info | Gets all the html code from the website that is specified (depending on the toggle) So the information can be later filtered out of it |
|  | go\_forward | Changes the screen to the next one (Ingredients) |
| Ingredients | on\_enter | Creates a popup that will show the list of ingredients and a forwards/back button to change screen |
|  | back | Send the user back to the page with the initial information |
|  | advance | Send the user to the next screen (Instructions) |
|  | ingredients | Get the list of instructions from the webpage specified |
| **Class Name** | **Subroutine** | **Function** |
| Instructions | on\_enter | Opens a popup that has a forward and back button, and shows all the instructions to make the dish |
|  | finish | Saves the dish made by the user and then send the user back to the main menu screen |
|  | timer\_setup | Opens a popup that shows the user a list of minutes that they can select to start a timer for the minutes selected |
|  | get\_instructions | Gets all the instructions that the user wanted through html screen scraping and appends them into a list |
|  | timeset | Creates a popup that tells the user that their timer has been set |
|  | threading\_timer | The function is the actual timer that uses multithreading and will wait for the time the user has set before opening a popup that tells the user their time is up |
| MyApp | build | Build is the function that is run when someone runs the class (MyApp().run() runs the build function) |

# Implementation

## Main Window Screen

### Design

A screenshot of a cell phone

Description automatically generated

### Code

#The builder load string is essentailly a window where I can customise each screen of my app. Each screen being a different class

Builder.load\_string("""

<MainWindow>:

    GridLayout:

        rows: 3

        Label:

            text\_size: self.size

            font\_size: 25

text: "Please Input a Dish:\\nThen select a source for the recipies\\n(If none are selected it will automatically select BakingMad)\\nIf you are looking for ideas select BakingMad and select"

            color: 0.32,1,1,1

            halign: 'center'

            valign: 'middle'

        TextInput:

            id: dish

            multiline: False

            text\_size: self.size

            font\_size: 30

            halign: 'center'

            valign: 'middle'

        GridLayout:

            cols:5

# This is the quit button for when the user wants to quit the applicaiton

            Button:

                text: "Quit"

                text\_size: self.size

                font\_size: 20

                color: 0.32,1,1,1

                halign: 'center'

                valign: 'middle'

                on\_release:

                    app.stop()

            Button:

                text: "View Recently Completed Dishes"

                text\_size: self.size

                font\_size: 20

                color: 0.32,1,1,1

                halign: 'center'

                valign: 'middle'

                on\_release:

                    root.old\_dishes()

            ToggleButton:

                id: BBC

                text: "BBC Dishes"

                text\_size: self.size

                font\_size: 20

                color: 0.32,1,1,1

                halign: 'center'

                valign: 'middle'

                group: 'dish'

                state: 'down'

            ToggleButton:

                id: Bak

                text: "BakingMad Dishes"

                text\_size: self.size

                font\_size: 20

                color: 0.32,1,1,1

                halign: 'center'

                valign: 'middle'

                group: 'dish'

            Button:

                text: "Search"

                text\_size: self.size

                font\_size: 20

                color: 0.32,1,1,1

                halign: 'center'

                valign: 'middle'

                on\_release:

                    root.pressed()

                    app.root.current = "second"

                    root.manager.transition.direction = "left"

#Here I have defined all the other classes that are used below, however I have used an alternative method to laying the classes out,

#This is because the Builder\_load\_screen mothod is inefficient and restricts the amount of customisation that I needed for the next couple screens

<OptionsWindow>:

<DishInformation>:

<Ingredients>:

<Instructions>:

""")

#This closes the Builder string, which is an alternative to using a seperate .kv file

class MyDishes():

    pass

class MainWindow(Screen):

    def clean(self, raw\_line):

        if '>' in raw\_line:

            raw\_line.find('>')

            return True

    def old\_dishes(self):

        popup = Popup(title = "Previously Searched Dishes

        layout = GridLayout(rows = 2)

        old\_dishes = ''

        dish\_list = open('dishes.txt','r')

        dishes = dish\_list.readlines()

        for line in dishes:

            line = line.rstrip()

            old\_dishes += line + '\n'

        dish\_list.close()

        old\_dish\_label = Label(text=old\_dishes)

        cnt\_btn = Button(text="Continue")

        cnt\_btn.bind(on\_release = popup.dismiss)

        layout.add\_widget(old\_dish\_label)

        layout.add\_widget(cnt\_btn)

        popup.content = layout

        popup.open()

    def list\_of\_dishes(self, query, url):

        page = urllib.request.urlopen(url)

        soup = BeautifulSoup(page, 'html.parser')

        content\_list = soup.get\_text()

        content\_list = soup.find\_all('h3')

        return content\_list

    def list\_of\_websites(self, query, url): #

        link = urllib.request.urlopen(url)

        soup = BeautifulSoup(link, 'html.parser')

        website\_list = soup.get\_text()

        website\_list = soup.find\_all('a', href=True)

        return website\_list

    def choices\_list(self, content\_lis):

        options = []

        for i in range(0, len(content\_lis)):

            dish = content\_lis[i].get\_text() #

            options.append(dish)

            self.clean(str(content\_lis))

        if toggle != "down":

            if options[-3] == "Recipes":

                options = options[:-3]

        return options

    def pressed(self):

        global query

        query = self.ids.dish.text

        query = "+".join(query.split())

        global toggle

        toggle = self.ids.BBC.state

        BBC = "https://www.bbc.co.uk/food/search?q=" + query

bakingMad = "https://www.bakingmad.com/search- results?searchtext=" + query

        if toggle == "down":

            content\_list = self.list\_of\_dishes(query, BBC)

        else:

            content\_list = self.list\_of\_dishes(query, bakingMad)

        global choices

        choices = self.choices\_list(content\_list)

        if len(choices) == 0:

            popup\_alert = Popup(title = "No dishes")

            layout = GridLayout(rows = 2)

      label2 = Label(text="There are no dishes that match your criteria \n Please go back")

            btn2 = Button(text="Accept")

            btn2.bind(on\_release = popup\_alert.dismiss)

            layout.add\_widget(label2)

            layout.add\_widget(btn2)

            popup\_alert.content = layout

            popup\_alert.open()

        return choices

## Food Options Window

### Design

A screenshot of a cell phone

Description automatically generated

### Code

class OptionsWindow(Screen, App): #OptionsWindow is the second screen, which provides a list if all the options that the user has for as dishes

    def on\_enter(self):

        popup = Popup(title='Dish Options', size\_hint=(1,1))

        layout1 = StackLayout(orientation='lr-bt')

  closebutton = Button(text='I want to make a different dish', size\_hint=(0.9,0.05))

        closebutton.bind(on\_press= popup.dismiss)

        closebutton.bind(on\_release = self.change\_page)

        scrlv = ScrollView(size\_hint=(0.9,0.95))

  slid = Slider(min=0, max=1, value=25, orientation='vertical', step=0.01, size\_hint=(0.1, 0.95))

        scrlv.bind(scroll\_y=partial(self.slider\_change, slid))

        slid.bind(value=partial(self.scroll\_change, scrlv))

        layout2 = GridLayout(cols=4, size\_hint\_y=None)

        layout2.bind(minimum\_height=layout2.setter('height'))

        for txt in choices:

btn = Button(text=txt, size\_hint\_y=None, height=80, valign='middle', font\_size=12)

            btn.text\_size = (btn.size)

            btn.bind(on\_press = self.find\_dish)

            btn.bind(on\_release = popup.dismiss)

            layout2.add\_widget(btn)

        scrlv.add\_widget(layout2)

        layout1.add\_widget(closebutton) #Add the close button to the layout

        layout1.add\_widget(scrlv) # Add the scroll bar to the layout

        layout1.add\_widget(slid)

        popup.content = layout1

        popup.open()

        if len(choices) == 0:

            popup.dismiss()

            sm.current = "main"

    def change\_page(self, event):

        sm.current = "main"

    def scroll\_change(self, scrlv, instance, value):

        scrlv.scroll\_y = value

    def slider\_change(self, s, instance, value):

        if value >= 0:

            s.value=value

    def find\_dish(self, btn):

        global dish\_choice

        dish\_choice = "+".join(btn.text.split())

        sm.current = "third"

## Dish Information Menu

### Design

A screenshot of a cell phone

Description automatically generated

### Code

class DishInformation(Screen, App):

    def on\_enter(self):

        popup = Popup(title='Dish Information')

        if toggle == "down": #If the BBC toggle was selected earlier

            website\_list = self.get\_websites()

            matched\_link = self.list\_matcher(website\_list)

            soup = self.initial\_info(matched\_link)

            try:

cook\_time = soup.find('p',attrs={"class":"recipe-metadata\_\_cook-time"}).text

            except AttributeError:

                cook\_time = 'Not stated'

            try:

prep\_time = soup.find('p',attrs={"class":"recipe-metadata\_\_prep-time"}).text

            except AttributeError:

                prep\_time = 'Not stated'

            try:

dietary = soup.find('p',attrs={"class":"recipe-metadata\_\_dietary-vegetarian-text"}).text

            except AttributeError:

                dietary = 'Not Suitable for Vegetarians'

            try:

serves = soup.find('p',attrs={"class":"recipe-metadata\_\_serving"}).text

            except AttributeError:

                serves = 'Not stated'

            cook\_time = 'The time it will take to cook is: ' +'\n' + cook\_time

prep\_time = 'The time it will take to prepare is: ' +'\n' + prep time

            dietary = 'The dish is: '+'\n' + dietary

            if len(serves) > 25:

                serves = serves[:25] + ':\n'+ serves[25:]

            serves = 'The dish:' +'\n' + serves

        if toggle == "normal":

            website\_link = self.get\_websites\_mad()

            global full\_link

            full\_link = 'https://www.bakingmad.com'+ str(website\_link)

            soup = self.initial\_info(full\_link)

            try:

cook\_time = soup.find('div',attrs={"class":"recipe-info\_\_prep-time"}).text

            except AttributeError:

                cook\_time = 'Not stated'

            try:

prep\_time = soup.find('div',attrs={"class":"recipe-info\_\_total-time"}).text

            except AttributeError:

                prep\_time = 'Not stated'

            try:

dietary = soup.find('li',attrs={"class":"recipe-info\_\_diet has-child"}).text

            except AttributeError:

                dietary = 'Not Suitable for Vegetarians'

            try:

serves = soup.find('li',attrs={"class":"recipe-info\_\_yield"}).text

            except AttributeError:

                serves = 'Not stated'

            cook\_time = 'The time it will take to cook is: ' +'\n' + cook\_time

            prep\_time = " ".join(prep\_time.split())

            prep\_time = prep\_time[:11] + ':\n'+ prep\_time[11:]

        first\_layout = GridLayout(cols=2)

        prep\_info = Label(text =prep\_time,

                        font\_size=20,

                        halign="center", #halign is the horizontal align

                        valign = "bottom") #valign is the horizontal align

        cook\_info = Label(text =cook\_time,

                        font\_size=20,

                        halign="center",

                        valign = "bottom")

        dietary\_info = Label(text = dietary,

                        font\_size=20,

                        halign="center",

                        valign = "bottom")

        serve\_info = Label(text= serves,

                        font\_size=20,

                        halign="center",

                        valign = "bottom")

back\_btn = Button(text="Would you like to go back?", size\_hint = (0.8,0.2))

        back\_btn.bind(on\_press = self.go\_back)

        back\_btn.bind(on\_release = popup.dismiss)

        next\_btn = Button(text='Next page?',size\_hint = (0.8,0.2))

        next\_btn.bind(on\_press = self.go\_forward)

        next\_btn.bind(on\_release = popup.dismiss)

        first\_layout.add\_widget(prep\_info)

        first\_layout.add\_widget(cook\_info)

        first\_layout.add\_widget(dietary\_info)

        first\_layout.add\_widget(serve\_info)

        first\_layout.add\_widget(back\_btn)

        first\_layout.add\_widget(next\_btn)

        popup.content = first\_layout

        popup.open()

    def get\_websites\_mad(self):

url = "https://www.bakingmad.com/search- results?searchtext=" +query #Redefine the URL used for baking mad

        r = requests.get(url)

        data = r.text

        soup =BeautifulSoup(data, "html.parser")

        web\_list = []

        for link in soup.find\_all('a'):

            web\_list.append(link.get('href'))

        new\_web\_list = []

        for val in web\_list:

            if val not in new\_web\_list:

                new\_web\_list.append(val)

        new\_dish = dish\_choice.replace('&', '')

        newer\_dish = new\_dish.replace('+', '-')

        newest\_dish = newer\_dish.replace('--', '-')

        newest\_dish = newest\_dish[:46]

        other\_new\_dish = dish\_choice.replace('&', 'and')

        other\_newer\_dish = other\_new\_dish.replace('+', '-')

        other\_newest\_dish = other\_newer\_dish.replace('--', '-')

        other\_newest\_dish = other\_newest\_dish[:49]

        for val in new\_web\_list:

            if newest\_dish.lower() in str(val):

                link = val

            if other\_newest\_dish.lower() in str(val):

                link = val

        return link #Return the link for the queery that the user has selected

    def go\_back(self, event):

        sm.current = "second"

    def get\_websites(self):

        url = "https://www.bbc.co.uk/food/search?q=" + query

        r = requests.get(url)

        data = r.text

        soup = BeautifulSoup(data, "html.parser")

        web\_list = []

        for link in soup.find\_all('a'):

           web\_list.append(link.get('href'))

        pages = []

        for link2 in web\_list:

            if  "/food/search?q="+query+"&page=" in link2:

                pages.append(link2)

        for web in pages:

            if web in web\_list:

                web\_list.remove(web)

        for link3 in range(0, len(web\_list)):

            if web\_list[link3] == "/food/my/favourites":

                num1 = link3

            if web\_list[link3] == "/food/faqs":

                num2 = link3

        web\_list = web\_list[num1+1:num2] #Return the list of dishes

        return web\_list

    def list\_matcher(self, web\_list):

        dish\_choices = dish\_choice.replace('+', ' ')

        for dish in range(0, len(choices)):

            if dish\_choices == (choices[dish]).strip():

                num = int(dish)

        global full\_link

        full\_link = 'https://www.bbc.co.uk'+ web\_list[num]

        return full\_link

    def initial\_info(self, full\_link):

        url = full\_link

        r = requests.get(url)

        data = r.text

        soup = BeautifulSoup(data, "html.parser")

        return soup #Return all the HTML code

    def go\_forward(self, event):

        sm.current = "forth"

## Ingredients Menu

### Design

A screenshot of a cell phone

Description automatically generated

### Code

class Ingredients(Screen, App):

    def on\_enter(self):

        popup = Popup(title='Ingredients-List') #On entering this screen, a popup is made

        g\_layout = GridLayout(rows=2)

        g\_layout2 = GridLayout(cols=2, size\_hint=(1,0.3))

        cont\_btn = Button(text='Would you like to continue?',size\_hint\_y=None, font\_size=20)

        cont\_btn.bind(on\_press = popup.dismiss)

        cont\_btn.bind(on\_release = self.advance)

        back\_btn = Button(text='Would you like to go back?', size\_hint\_y=None, font\_size=20)

        back\_btn.bind(on\_press = self.back)

        back\_btn.bind(on\_release = popup.dismiss)

        ingredients\_list = self.ingredients()

        ingredient\_string = ''

        for ingredient in ingredients\_list:

            if len(ingredient) > 70:

                ingredient = ingredient[:70] + '\n'+ ingredient[70:]

            ingredient\_string += ingredient + '\n'

        ingred\_list = Label(text=ingredient\_string,

                                 halign="center",

                                 valign = "top")

        g\_layout2.add\_widget(back\_btn)

        g\_layout2.add\_widget(cont\_btn)

        g\_layout.add\_widget(ingred\_list)

        g\_layout.add\_widget(g\_layout2)

        popup.content = g\_layout

        popup.open()

    def back(self, event):

        sm.current = "second"

    def advance(self, event):

        sm.current = "fifth"

    def ingredients(self):

        url = full\_link

        r = requests.get(url)

        data = r.text

        soup = BeautifulSoup(data, "html.parser")

        ingredient\_list = []

        if toggle == "down": #

            ingredient =  soup.find\_all('li',attrs={"class":"recipe-ingredients\_\_list-item"})

            for a in ingredient[0:]:

                result = a.text.strip()

                ingredient\_list.append(result)

        if toggle == "normal": #If the user selects BBC GoodFood

     ingredient = soup.find\_all('li', attrs={"itemprop":"recipeIngredient"}) #Finds all the recipe ingredients with the html tag li

            for a in ingredient[0:]:

                result = a.text.strip()

                new\_result = result.replace("\n",' ')

                long\_int = new\_result.find("   ")

                if long\_int > 0:

                    new\_result  = new\_result[:int(long\_int)]

                ingredient\_list.append(new\_result)

        return ingredient\_list

## Instructions Menu

### Information Design

A screenshot of a cell phone

Description automatically generated

### Timer Design – 1

A screenshot of a computer

Description automatically generated

### Timer Design – 2

A screenshot of a cell phone

Description automatically generated

### Timer Design – 3

A screenshot of a cell phone

Description automatically generated

### Code

class Instructions(Screen, App):

    def on\_enter(self):

        popup = Popup(title="Instructions")

        instructions = self.get\_instructions()

        g\_layout = GridLayout(rows=2, spacing=10)

        g\_layout2 = GridLayout(cols=2, size\_hint=(1,0.25))

        fin\_btn = Button(text='Would you like to Finish?',size\_hint\_y=None, font\_size=20)

        fin\_btn.bind(on\_press = popup.dismiss)

        fin\_btn.bind(on\_release = self.finish)

        timer\_btn = Button(text='Would you like to use a timer?', size\_hint\_y=None, font\_size=20)

        timer\_btn.bind(on\_press = self.timer\_setup)

        instruction\_string = ''

        for i in instructions:

            instruction\_string += i + '\n'

      instruc\_lbl = Label(text=instruction\_string, text\_size=self.size,  font\_size=13, halign="center",valign="middle")

        g\_layout2.add\_widget(timer\_btn)

        g\_layout2.add\_widget(fin\_btn)

        g\_layout.add\_widget(instruc\_lbl)

        g\_layout.add\_widget(g\_layout2)

        popup.content = g\_layout

        popup.open()

    def finish(self, event):

        dish\_list = open("dishes.txt",'a')

        dish\_list.write(dish\_choice.replace('+',' ') + '\n')

        dish\_list.close()

        sm.current = "main"

    def timer\_setup(self, event):

        popup2 = Popup(title = "Interval time:")

        grid\_layout = GridLayout(cols=2)

        anchor = AnchorLayout(anchor\_x = 'center', anchor\_y='bottom')

input\_time = Label(text="Please input the time you \nwould like between each recipe instruction: ", font\_size=20,)

        dropdown = DropDown()

        for minutes in range(19):

btn = Button(text=(str(int(minutes+1)\*5)), size\_hint\_y=None, height = 44)

            btn.bind(on\_release = lambda btn: dropdown.select(btn.text))

            dropdown.add\_widget(btn)

        mainbutton = Button(text="Minutes", size\_hint\_y=None)

        mainbutton.bind(on\_release = dropdown.open)

dropdown.bind(on\_select = lambda instance, x: setattr(mainbutton, 'text', x))

        dropdown.bind(on\_select = self.timeset)

        dropdown.bind(on\_dismiss = popup2.dismiss)

        anchor.add\_widget(mainbutton)

        grid\_layout.add\_widget(input\_time)

        grid\_layout.add\_widget(anchor)

        popup2.content = grid\_layout

        popup2.open()

def get\_instructions(self):

        url = full\_link

        r = requests.get(url)

        data = r.text

        soup = BeautifulSoup(data, "html.parser")

        instructions\_list = []

        if toggle == "down": #If the user selects BBC GoodFood

  instruction =  soup.find\_all('li',attrs={"class":"recipe-method\_\_list-item"})

        if toggle == "normal":

            instruction = soup.find\_all('div',attrs={"class":"method\_\_text"})

        for a in instruction[0:]:

            result = a.text.strip()

            result = str(result).replace(".","\n")

            instructions\_list.append(result)

        return instructions\_list

    def timeset(self, instance, x):

        global count\_num

        count\_num = (int(x)\*60)

        popup4 = Popup(title = "Confirmation")

        layout = GridLayout(rows = 2)

        label2 = Label(text="Your timer has been set \n You do not need to set  another \n Unless you want to")

        btn2 = Button(text="Agreed")

        btn2.bind(on\_release = popup4.dismiss)

        layout.add\_widget(label2)

        layout.add\_widget(btn2)

        popup4.content = layout

        popup4.open()

        thread = threading.Thread(target= self.threading\_timer)

        thread.start()

    def threading\_timer(self):

        time.sleep(count\_num)

        layout = GridLayout(rows=2)

        popup = Popup(title="Countdown Finished!")

        lbl = Label(text="Your countdown is complete",font\_size=20)

        btn = Button(text="Close?")

        btn.bind(on\_release = popup.dismiss)

        layout.add\_widget(lbl)

        layout.add\_widget(btn)

        popup.content = layout

        popup.open()

## How to Run the Program

The program is automatically run when the program is run using the following lines at the end of the code:

sm = ScreenManager()

sm.add\_widget(MainWindow(name="main"))

sm.add\_widget(OptionsWindow(name="second"))

sm.add\_widget(DishInformation(name="third"))

sm.add\_widget(Ingredients(name="forth"))

sm.add\_widget(Instructions(name="fifth"))

class MyApp(App):

    def build(self):

        return sm

if \_\_name\_\_ == '\_\_main\_\_': #This intialitises the code

    MyApp().run()

This is because \_\_name\_\_ is always equal to ‘\_\_main\_\_’ and MyApp().run() is a command that runs the build function of MyApp, which then returns the ScreenManager, and therefore turns the “main” screen on. The widgets that are being added to the ScreenManager() are all the different classes and their information, and then I have defined each class in the screen manager as either “main” or the second, third… etc. screen. I can then refer to sm at any point in my code and change the sm.current to whatever screen I have defined. (e.g. in the code I use sm.current=“second” to go to the second screen)

# Testing

In this section I will be performing all my tests in order to ensure that the code I wrote functions as intended. As my code has no “boundary data” per say I will be attempting various inputs that I believe could potentially cause the code to crash. Furthermore, as there is little user input in my code, the testing section will involve various methods outside of the code itself which could also cause the code to crash. I will be analysing all types of data to see whether the code works as function, an example of this would be inputting numbers into a string input.

## Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Main Menu** | | | | | |
| **Test Number** | **Test Description** | **Test Data** | **Expected Output** | **Actual Output** | **Pass?** |
| 1 | Does the main menu open when the code is run? | **Normal Data**  Run the program | The main menu opens | The main menu opens | Yes |
| 2 | Will inputting a dish that should return a list of dishes with BBC toggle returns a list of cakes | **Normal Data**  Input “cake” and press search with BBC toggle | A list of cakes should be shown | A list of cakes is shown | Yes |
| 2b | Will inputting a dish that should return a list of dishes with Baking Mad toggle return a list of cakes | **Normal Data**  Input “cake” and press search with Baking Mad toggle | A list of cakes should be shown | A list of cakes is shown | Yes |
| 2c | Will inputting a dish that should return a list of dishes with no toggle return a list of cakes | **Normal Data**  Input “cake” and press search with no toggle | A list of cakes should be shown – same as BakingMad cakes search | A list of cakes is shown which is the same as BakingMad cake search | Yes |
| 3 | Will inputting a string dish that should not return an output with BBC toggle open a not found popup? | **Erroneous Data**  Input “dwaoinda” and press search with BBC toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 3b | Will inputting a string dish that should not return an output with Baking Mad toggle open a not found popup? | **Erroneous Data**  Input “dwaoinda” and press search with Baking Mad toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 3c | Will inputting a string dish that should not return an output with no toggle open a not found popup? | **Erroneous Data**  Input “dwaoinda” and press search with no toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 4 | Will inputting a numerical dish that should not return an output with BBC toggle open a not found popup? | **Erroneous Data**  Input “0122194” and press search with BBC toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 4b | Will inputting a numerical dish that should not return an output with Baking Mad toggle open a not found popup? | **Erroneous Data**  Input “0122194” and press search with Baking Mad toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 4c | Will inputting a numerical dish that should not return an output with no toggle open a not found popup? | **Erroneous Data**  Input “0122194” and press search with no toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 5 | Will inputting no dish open a nothing found popup in BBC toggle? | **Erroneous Data**  Input nothing and press search with BBC toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 5b | Will inputting no dish open a nothing found popup in no toggle? | **Erroneous Data**  Input nothing and press search with no toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 5c | Will inputting no dish open a list of recommended dishes using BakingMad toggle? | **Normal Data**  Input nothing and press search with BakingMad toggle | A list of recommended dishes is shown on the next screen | A list of recommended dishes is shown on the next screen | Yes |
| 6 | Will inputting a symbols dish that should not return an output with BBC toggle open a not found popup? | **Erroneous Data**  Input “!?%^&$” and press search with BBC toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 6b | Will inputting a symbols dish that should not return an output with Baking Mad toggle open a not found popup? | **Erroneous Data**  Input “!?%^&$” and press search with Baking Mad toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 6c | Will inputting a symbols dish that should not return an output with no toggle open a not found popup? | **Erroneous Data**  Input “!?%^&$” and press search with no toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 7 | Will inputting a dish with “.com” that should not return an output with BBC toggle open a not found popup? | **Boundary Data**  Input “cowfish.com” and press search with BBC toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 7b | Will inputting a dish with “.com” that should not return an output with Baking Mad toggle open a not found popup? | **Boundary Data**  Input “cowfish.com” and press search with Baking Mad toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 7c | Will inputting a dish with “.com” that should not return an output with no toggle open a not found popup? | **Boundary Data**  Input “cowfish.com” and press search with no toggle | A popup saying no dishes should be shown | A popup saying no dishes is shown | Yes |
| 8 | Does the quit button function when pressed? | **Normal Data**  Click on the quit button | The app should close | The app closes | Yes |
| 9 | Does the previously completed dishes work when the list is empty? | **Normal Data**  Make the list empty and select “show previously completed dishes | An empty list should be shown | An empty list is shown | Yes |
| **Dish Options Menu** | | | | | |
| **Test Number** | **Test Description** | **Test Data** | **Expected Output** | **Actual Output** | **Pass?** |
| 10 | Do the dish options buttons work when I put in a dish that returns dishes under the BBC dishes tab | **Normal Data**  Put in “fish” under the BBC toggle then select any dish at random | The user should be taken to the initial information screen for “fish stock” | The user is taken to the initial information screen for “fish stock” | Yes |
| 10b | Do the dish options buttons work when I put in a dish that returns dishes under the BakingMad dishes tab | **Normal Data**  Put in “fish” under the BakingMad toggle then select any dish at random | The user should be taken to the initial information screen for “fish pie with leek and chorizo” | The user is taken to the initial information screen for “fish stock” | Yes |
| 10c | Do the dish options buttons work when I put in a dish that returns dishes under no dishes tab | **Normal Data**  Put in “fish” under the BakingMad toggle then select any dish at random | The user should be taken to the initial information screen for “fish pie with leek and chorizo” | The user is taken to the initial information screen for “fish stock” | Yes |
| 11 | Does the “Different dish” button work in the dish options menu | **Normal Data**  Click the different dish button | The user should be sent back to the main menu screen | The user is sent back to the main menu screen | Yes |
| 12 | Does the scroll bar work in the dish options menu when there is a large amount of dish options? | **Normal Data**  Put in “cake” under the BBC toggle then make the screen size smaller and try scrolling | The scroll bar should allow the user to see all the options | The user can scroll to see all the available dishes | Yes |
| 13 | Does the scroll bar blue tab work in the dish options menu when there is a large amount of dish options? | **Normal Data**  Put in “cake” under the BBC toggle then make the screen size smaller and try scrolling | The blue tab in the scroll bar should be representative of the distance the user has made of the full options while scrolling | The scroll tab is accurate to the position the user is while scrolling | Yes |
| 14 | Is the blue tab invisible if the number of dishes shown is not enough to allow for scrolling? | **Normal Data**  Put in “fish” under the BakingMad dishes toggle (returns less than 4 options) | The blue tab should be invisible | The blue tab is invisible | Yes |
| 15 | Does the scroll bar work in the dish options menu when there is a small amount of dish options? | **Normal Data**  Put in “fish” under the Baking Mad toggle then make the screen size smaller and try scrolling (returns less than 3 dishes) | The scroll bar should work if the screen is made small enough | The scroll bar works when the screen is small enough for the user to need to scroll | Yes |
| 16 | Does selecting a dish close the Dish Options popup? | **Normal Data**  Select a random dish (In this case Strawberry Heart Cake in the cake options of BakingMad Dishes) | The popup should close and the next popup (initial information) should open | The popup closes and initial information is opened | Yes |
| **Dish Information Menu** | | | | | |
| **Test Number** | **Test Description** | **Test Data** | **Expected Output** | **Actual Output** | **Pass?** |
| 17 | Is the total time always shown when using a dish within the BBC dishes toggle? | **Normal Data**  Use a random query (banana) and choose a random dish (Banana bread loaf) and check the total time with the BBC toggle | The total time should be shown | The total time is shown (“less than 30 mins”) | Yes |
| 17b | Is the total time always shown when using a dish within the BakingMad toggle? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the total time with the BakingMad dishes toggle | The total time should be shown | The total time is shown (“1h 25m”) | Yes |
| 17c | Is the total time always shown when using a dish with no toggle? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the total time with no toggle | The total time should be shown | The total time is shown (“1h 25m”) | Yes |
| 18 | Is the time it will take to cook shown when using a dish within the BBC dishes toggle / Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Banana bread loaf) and check the time taken to cook with the BBC toggle on | The total time should be shown/ It should say that there is no time applicable | The total time is shown (“1 to 2 hours”) | Yes |
| 18b | Is the time it will take to cook shown when using a dish within the BakingMad toggle/ Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the time taken to cook with the BakingMad dishes toggle | The total time should be shown/ It should say that there is no time stated | It says that the total time is not stated | Yes |
| 18c | Is the time it will take to cook shown when using a dish with no toggle/ Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the time taken to cook with no toggle | The total time should be shown/ It should say that there is no time stated | It says that the total time is not stated | Yes |
| 19 | Is the Dietary Needs shown when using a dish within the BBC dishes toggle / Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Banana bread loaf) and check the dietary needs with the BBC toggle on | The dietary needs should be shown/ It should say that there are no needs applicable | The dietary needs are shown (Vegetarian) | Yes |
| 19b | Are the Dietary needs shown when using a dish within the BakingMad toggle/ Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the dietary needs with the BakingMad dishes toggle | The dietary needs should be shown/ It should say that there are no needs applicable | The dietary needs are shown (Vegetarian) | Yes |
| 19c | Are the dietary needs shown when using a dish with no toggle/ Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the dietary needs with no toggle | The dietary needs should be shown/ It should say that there are no needs applicable | The dietary needs are shown (Vegetarian) | Yes |
| 20 | Is the number of people the dish serves shown when using a dish within the BBC dishes toggle / Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Banana bread loaf) and check the serving size with the BBC toggle on | The number of people the dish serves should be shown/ The size of the dish/Information about the size of the dish/ It should say that there is no time applicable | The dish information about the serving size is shown (“Makes two loaves”) | Yes |
| 20b | Is the number of people the dish serves shown when using a dish within the BakingMad toggle/ Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the serving size with the BakingMad dishes toggle | The number of people the dish serves should be shown/ The size of the dish/Information about the size of the dish/ It should say that there is no time applicable | The dish information about the serving size is sown (“Serves 1”) | Yes |
| 20c | Is the number of people the dish serves shown when using a dish with no toggle/ Does it say that it cannot be shown? | **Normal Data**  Use a random query (banana) and choose a random dish (Best Banana Cake) and check the serving size with no toggle | The number of people the dish serves should be shown/ The size of the dish/Information about the size of the dish/ It should say that there is no time applicable | The dish information about the serving size is shown (“Serves 1”) | Yes |
| 21 | Does the back button work on the Dish Information screen? | **Normal Data**  Press the back button on the Dish information screen | It should send the user back to the Dish Options screen and the popup for dish information should close | The user is sent back to the Dish information Screen and the dish information popup is close | Yes |
| 22 | Does the forwards button work on the Dish Information screen? | **Normal Data**  Press the Next page button on the Dish Information screen | It should send the user to the next page (The Ingredients-List) | The user is sent to the next page (The ingredients list) | Yes |
| **Ingredients List** | | | | | |
| **Test Number** | **Test Description** | **Test Data** | **Expected Output** | **Actual Output** | **Pass?** |
| 23 | Are the ingredients shown for any random dish with the BBC toggle | **Normal Data**  Select a random query and a random dish from the query using the BBC toggle (“ice cream – query and then selecting Pistachio ice cream) | The list of ingredients should be shown | The list of ingredients is shown | Yes |
| 23b | Are the ingredients shown for any random dish with the BakingMad toggle | **Normal Data**  Select a random query and a random dish from the query using the Baking Mad toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of ingredients should be shown | The list of ingredients is shown | Yes |
| 23c | Are the ingredients shown for any random dish with no toggle | **Normal Data**  Select a random query and a random dish from the query using no toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of ingredients should be shown | The list of ingredients is shown | Yes |
| 24 | Are all the ingredients shown for any random dish with the BBC toggle | **Normal Data**  Select a random query and a random dish from the query using the BBC toggle (“ice cream – query and then selecting Pistachio ice cream) | The list of ingredients shown should be the same as that on the website of the dish | The list of ingredients is shown is the same as that on the website | Yes |
| 24b | Are all the ingredients shown for any random dish with the BakingMad toggle | **Normal Data**  Select a random query and a random dish from the query using the Baking Mad toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of ingredients shown should be the same as that on the website of the dish | The list of ingredients shown is the same as that on the website of the dish | Yes |
| 24c | Are all the ingredients shown for any random dish with the no toggle | **Normal Data**  Select a random query and a random dish from the query using no toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of ingredients shown should be the same as that on the website of the dish | The list of ingredients shown is the same as that on the website of the dish | Yes |
| 25 | Is the go back button working on the Ingredients List menu | **Normal Data**  Select the go back button on the ingredients list menu | The user should be sent back to the dish options menu | The user is sent back to the dish options menu | Yes |
| 26 | Is the continue button working on the Ingredients list | **Normal Data**  Select the continue button on the ingredient’s menu page | The user should be sent to the instruction’s menu page | The user is sent to the instruction’s menu page | Yes |
| **Instructions List** | | | | | |
| **Test Number** | **Test Description** | **Test Data** | **Expected Output** | **Actual Output** | **Pass?** |
| 27 | Are the instructions shown for any random dish with the BBC toggle | **Normal Data**  Select a random query and a random dish from the query using the BBC toggle (“ice cream – query and then selecting Pistachio ice cream) | The list of instructions should be shown | The list of instructions is shown | Yes |
| 27b | Are the instructions shown for any random dish with the BakingMad toggle | **Normal Data**  Select a random query and a random dish from the query using the Baking Mad toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of instructions should be shown | The list of instructions is shown | Yes |
| 27c | Are the instructions shown for any random dish with no toggle | **Normal Data**  Select a random query and a random dish from the query using no toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of instructions should be shown | The list of instructions is shown | Yes |
| 28 | Are all the instructions shown for any random dish with the BBC toggle | **Normal Data**  Select a random query and a random dish from the query using the BBC toggle (“ice cream – query and then selecting Pistachio ice cream) | The list of instructions shown should be the same as that on the website of the dish | The list of instructions is shown is the same as that on the website | Yes |
| 28b | Are all the instructions shown for any random dish with the BakingMad toggle | **Normal Data**  Select a random query and a random dish from the query using the Baking Mad toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of instructions shown should be the same as that on the website of the dish | The list of instructions shown is the same as that on the website of the dish | Yes |
| 28c | Are all the instructions shown for any random dish with the no toggle | **Normal Data**  Select a random query and a random dish from the query using no toggle (“ice cream – query and then selecting Raspberry ripple ice cream) | The list of instructions shown should be the same as that on the website of the dish | The list of instructions shown is the same as that on the website of the dish | Yes |
| 29 | Is the finish button working on the Instructions List menu | **Normal Data**  Select the finish button on the instructions list menu | The user should be sent back to the main menu | The user is sent back to the main menu | Yes |
| 30 | Is the timer button working on the Ingredients list | **Normal Data**  Select the timer button on the ingredient’s menu page | The user should be sent to the timer set page | The user is sent to the timer set menu page | Yes |
| **Timer Settings Page** | | | | | |
| **Test Number** | **Test Description** | **Test Data** | **Expected Output** | **Actual Output** | **Pass?** |
| 31 | Will the Interval timer button open a list of minute options for the user to choose from when clicked? | **Normal Data**  Click on the minutes button inside the Interval Time menu | A list of minutes should appear | A list of minutes appears | Yes |
| 32 | Are the minutes options scrollable, so the user can scroll down to up t0 95 minutes? | **Normal Data**  Click on the minutes button and then try to scroll down to the 95th minute | The user should be able to scroll down to the 95th minute option | The user can scroll down to the 95th minute option | Yes |
| 33 | Does the popup close when a minute is selected from the minutes scrollable button? | **Normal Data**  Click on the minutes button and select any number of minutes (5 in this case) | The popup for interval time should close | The popup for interval time closes | Yes |
| 34 | When the user selects a time, a new popup confirming the user has set a time is opened | **Normal Data**  Click on the minutes button and select any number of minutes (5 in this case) | The new popup saying that the time has been set should open | The new popup saying that the time has been set is opened | Yes |
| 35 | The new popup telling that user that they have selected has a button that once pressed closes the popup | **Normal Data**  Press the agreed button on the popup | The popup should close | The popup closes | Yes |
| 36 | After the time set by the user, a new popup should open telling the user that the time has been completed | Set a timer for 5 minutes and after 5 minutes check to see if a popup opens | A timer should open after 5 minutes | A timer opened after 5 minutes | Yes |

**Yellow Highlighting means that evidence for the tests will be shown below**

## Evidence

### Evidence 1 - Test 2

A screenshot of a cell phone

Description automatically generated

Here it shows that when the user inputs a dish such as cake and BBC Dishes is toggled, the list of cakes will be shown

### Evidence 2 - Test 3b

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

When the user inputs an input that doesn’t work (a list of strings) an error message is popped up

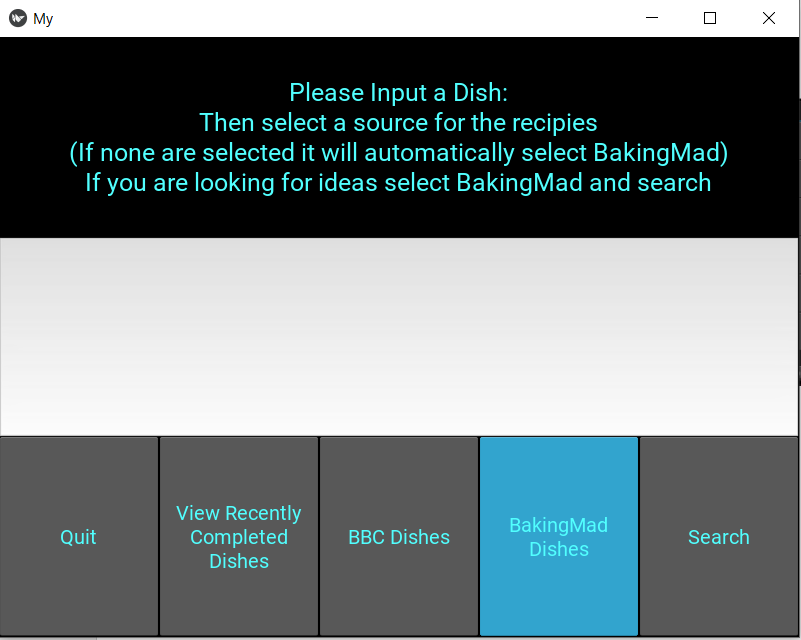
### A screenshot of a cell phone Description automatically generatedEvidence 3 - Test 4c

A screenshot of a cell phone

Description automatically generated

When the user inputs a list of numbers, with no toggle activated, then an error box is shown as follows

### Evidence 4 -Test 5c

A picture containing screenshot

Description automatically generated

When the user specifically inputs Baking Mad dishes toggle and no input as above:

This then shows a list of random dishes that the app recommends the user can make

### Evidence 5 - Test 6b

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

When the input into BBC dishes is a list of random symbols, then the output is a box saying that no dishes were found

### Evidence 6 - Test 7b

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

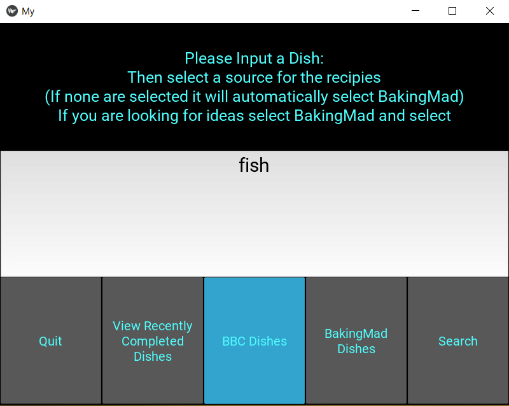
This however is interesting as the query is appended onto the end of a search url that is premade, therefore if I added .com to a url that already ends in .com there is a chance it would return an error within the code, however the code still treats the .com as just a string to append to the end of a list

When the user inputs a query with a .com inside of the query then the output is still a popup saying no dishes are available

### Evidence 7 - Test 10

A screenshot of a cell phone

Description automatically generated



When fish in inputted under the BBC Dishes toggle, then the list of dishes is opened, and when the user selects any dish (in this case the Fish stock) then the initial information is successfully displayed

### Evidence 8 - Test 11

A screenshot of a cell phone

Description automatically generatedA picture containing monitor, screenshot

Description automatically generated

When the user clicks on the Different dish button, they are sent back to the main menu screen

### Evidence 9 - Test 12

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A picture containing monitor, screenshot

Description automatically generated

When the window is made smaller, the scroll bar still functions as intended as seen above when scrolling through the list

As seen above, having searched up cake using the BBC Dishes toggle bring up a list of cakes

### Evidence 10 - Test 14

A screenshot of a cell phone

Description automatically generated

When the list of queries is not long enough for scrolling to be needed (In this example where fish is searched up using the Baking Mad toggle). Then the blue dot on the scroll bar is not shown until the screen is made small enough for scrolling to be necessary

### Evidence 11 - Test 15

A screenshot of a cell phone

Description automatically generated

In the same query as Test 14 (fish searched up with the Baking Mad toggle) If the screen is made small enough, then the blue tab appears, and the scrolling feature can function as intended

### Evidence 12 - Test 17b

A screenshot of a cell phone

Description automatically generatedA screenshot of a video game

Description automatically generated

When the user inputs banana under the Baking Mad toggle, then the total time is shown on the Dish Information screen (When best banana cake is chosen at random to represent all dishes under this circumstance)

A screenshot of a cell phone

Description automatically generated

### Evidence 13 - Test 18

A picture containing screenshot, monitor

Description automatically generated

When the user inputs Banana under the BBC toggle and selects Banana Bread Loaf (chosen at random to represent all dishes under this circumstance). Then the time is will take to cook is shown

### Evidence 14 - Test 19b

A screenshot of a cell phone

Description automatically generatedA screenshot of a video game

Description automatically generated

When the user inputs banana under the Baking Mad toggle, then the Dietary needs are shown on the Dish Information screen (When best banana cake is chosen at random to represent all dishes under this circumstance)

### Evidence 15 - Test 20c

A screenshot of a cell phone

Description automatically generatedA screenshot of a video game

Description automatically generated

When the user inputs banana under no toggle, then the serving information is shown on the Dish Information screen (When best banana cake is chosen at random to represent all dishes under this circumstance)

### A picture containing screenshot Description automatically generatedEvidence 16 - Test 23

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A list of ingredients is successfully shown when the user inputs Ice cream under a BBC toggle then selects pistachio ice cream

### Evidence 17 - Test 24b

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

When the user inputs ice cream under the Baking Mad toggle and then selects the Raspberry Ripple ice cream. The instructions that are shown are the exact same as those found online in the Baking Mad website

### Evidence 18 / 19 - Test 26 / Test 27b

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

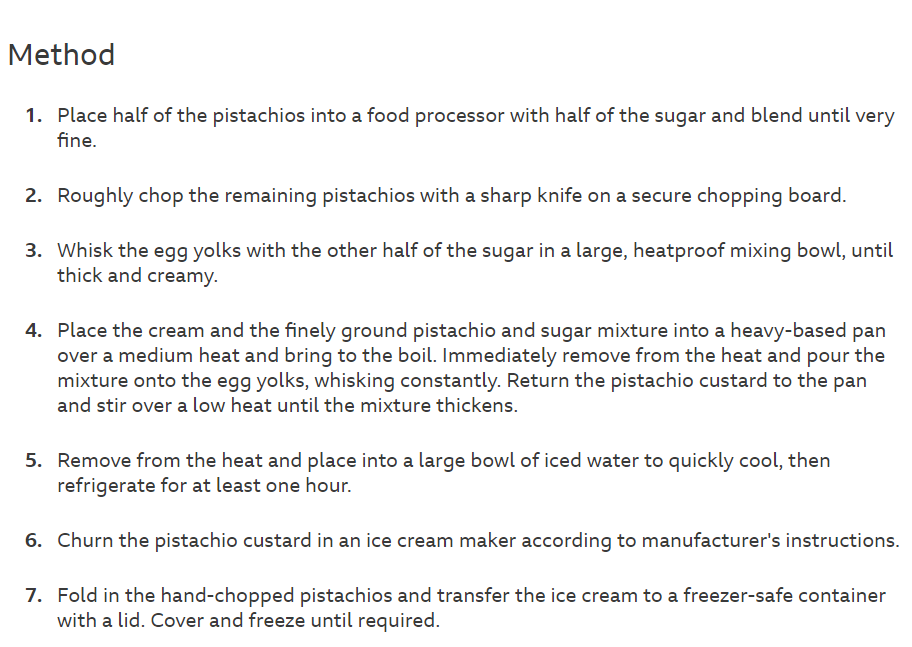
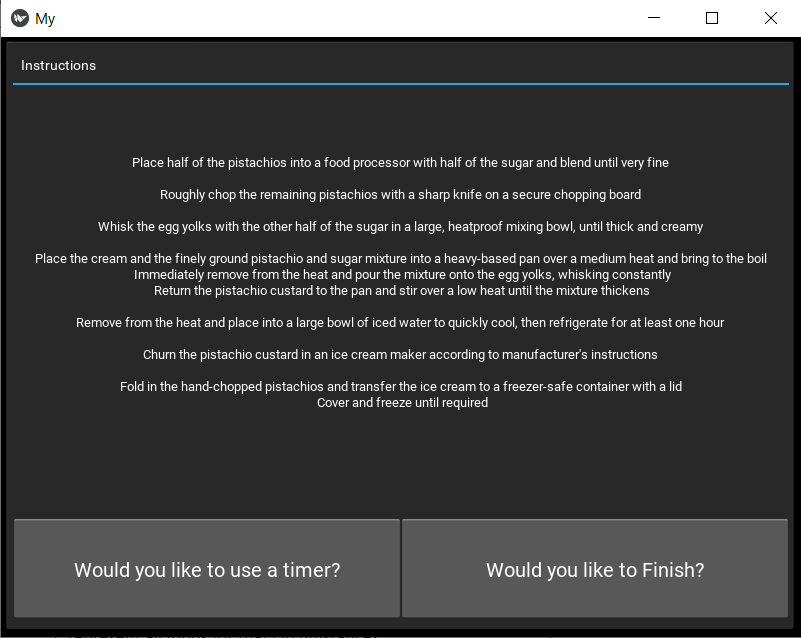
27b:

All the instructions are shown when the user enters the instructions page for raspberry ripple ice cream when ice cream is searched under the Baking Mad toggle

26:

When the continue button is pressed on the Ingredients List then the user is sent to the Instructions list on the next page

### Evidence 20 - Test 28



When the user inputs ice cream under the BBC dishes toggle then selects Pistachio Ice Cream. Then all the instructions shown are the same as that on the BBC website for Pistachio Ice Cream

### Evidence 21 - Test 29

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

When the user selects the Finish button on the instructions page they are sent back to the main menu, with the query still in the search box

### Evidence 22 - Test 30

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

When the user selects the timer button, they are successfully sent to the timer menu screen

### Evidence 23 - Test 31

A screenshot of a computer

Description automatically generatedA screenshot of a cell phone

Description automatically generated

When the user selects the Minutes button then a list of minutes appears in intervals of 5

### A screenshot of a computer Description automatically generatedEvidence 24 - Test 32

A screenshot of a computer

Description automatically generated

The user can scroll down the minute’s options displayed when the minutes button is clicked, all the way down to the 95th minute.

### Evidence 25 - Test 34

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

When the user selects a time from the time popup, a message appears letting the user know that the timer has been set, and a button appears that lets the user close the popup displaying the message

### A screenshot of a computer Description automatically generatedEvidence 26 - Test 36

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Once the user has selected a time, as soon as the popup confirming the time has been set. After whatever number of minutes that the user has set the time for, a new popup will be shown saying that the countdown is complete, and then a button that will allow the user to close the countdown

## System Tests

Now that all the Individual tests have been completed, I will be doing three full system tests, one for each toggle, with a random dish that will be used as a representative for every dish in the given circumstance (i.e. given toggle). This will ensure that the code runs smoothly. I will then run a system test on the timer function to ensure that it works as it is meant to.

### System Test 1 – BBC Toggle

A screenshot of a cell phone

Description automatically generated

The toggle is set to BBC dishes and a random dish that I know will return dish options is inputted (pie)

A screenshot of a cell phone

Description automatically generated

The pie input returned a list of pie dishes in the dish options menu, where I selected the top right dish randomly, pecan pie

A screenshot of a cell phone

Description automatically generated

Upon selecting the pecan pie I was shown all of the initial information

A screenshot of a cell phone

Description automatically generated

Then when I clicked the forwards button, I got the list of ingredients required for the dish

A screenshot of a cell phone

Description automatically generated

Upon clicking the continue button again I was shown a list of the instructions required to make the Pecan Pie

A screenshot of a cell phone

Description automatically generated

When I clicked the finish button on the instructions menu I was sent back to the main menu, where pie was still the search query

### System Test 2 – Baking Mad Toggle

A screenshot of a cell phone

Description automatically generated

I toggled the Baking Mad Dishes and input a random dish I know will return an output (salmon)

A screenshot of a cell phone

Description automatically generated

I was sent to a list of salmon dishes, and randomly selected the dish below the top right once (Smoked Salmon Blini)

A screenshot of a cell phone

Description automatically generated

I was then showed a list of initial information when I clicked on the dish

A screenshot of a cell phone

Description automatically generated

After clicking next page I was shown a list of ingredients

A screenshot of a cell phone

Description automatically generated

Upon clicking on the continue button I was shown a list of instructions required to make the dish

A screenshot of a cell phone

Description automatically generated

When I clicked on the finish button then I was sent back to the main menu with salmon as the query

### System Test 3 – No Toggle

A screenshot of a cell phone

Description automatically generated

I had no toggle set and then selected searched up mango dishes

A screenshot of a video game

Description automatically generated

I was shown a list of mango dishes and randomly selected the one in the top right (Mango Tart)

A screenshot of a cell phone

Description automatically generated

Then when I selected the dish, I was shown a list of initial information

A screenshot of a cell phone

Description automatically generated

Upon clicking next page, I was shown a list of ingredients required to make the dish

A screenshot of a cell phone

Description automatically generated

When I clicked on the continue button, I was shown a list of the instructions necessary to make the dish

A screenshot of a cell phone

Description automatically generated

When I clicked on the finish button I was sent back to the main menu and the query of mango was still in the input

### System Test 4 – Timer Function

A screenshot of a cell phone

Description automatically generated

If the user clicked on the timer button on the instruction’s menu, they are sent to this page where there is a button saying Minutes and a prompt asking the select a time

A screenshot of a computer

Description automatically generated

Once the user clicks on the minutes button ,they are shown a list of minutes that they can then choose from, the minutes options also being a scrollable list up to 95 minutes

A screenshot of a cell phone

Description automatically generated

Once the user has selected a time, then they are shown a popup that tells the user their time has been set and then they can agree to close the popup

A screenshot of a cell phone

Description automatically generated

Once the minutes set has been completed, they user is shown a popup telling them that the countdown is complete and then a button they can press to close the popup

# Evaluation

## General Appraisal

Overall, I believe that the project has went well. I have definitely gained a lot of experience in using the kivy module and if I had been given this task given the information I now have, I believe that I could have used a lot more efficient techniques to create the same code. However, this has been an extremely beneficial project as I can now create apps using a language that I am already aware of using, Python.

The Analysis section of this project allowed me to interview various people and therefore specify the project towards the requirements of the general user. Then the design section was extremely beneficial in helping me create the app as I had an overlay of what section of code should go where and how to organise my code into various functions and classes required. The Implementation of the code took the most time as I had to learn and use a module that I had never heard of before, however over time kivy became easier to understand and I felt more comfortable handling the module. Finally, the testing helped me cover any bugs that the code could have had by inputting boundary data and inputs that I would not have thought the user would have tried otherwise.

I would say the most challenging aspect of the code was attempting to pass variables through the classes without being able to call classes into one another. This was an extremely big challenge and did mean that I had to inevitably use global variables, however I did spend hours searching for alternative routes only to find out that they all had their own negatives that would have had disastrous consequences to the app overall.

## Meeting Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| **Num.** | **Objective** | **How well was it met?** | **Could it be improved?** |
| 1 | The app starts off with a main menu, where the user has a text input area and five buttons to choose from: | Great | No improvements |
| 1.a | The first button will be the start button | Great | No improvements |
| 1.b | The second button will be one of two toggle buttons, one which has the BBC Good Food toggle button, and will start off selected | Great | No improvements |
| 1.c | The third button will be the second of two toggle buttons, the other of which is the Baking Mad Toggle button | Great | No improvements |
| 1.d | The fourth button is a “show completed dishes” button | Great | No improvements |
| 1.e | The fifth button will be a quit button | Great | No improvements |
| 2 | If the “start button” is pressed and there is no input, or an input that returns no dishes, with the BBC Good Food toggle selected then a popup should appear saying that there are no dishes available, and a button that lets users return from the popup | Great | No improvements |
| 3 | If the “start button” is pressed and there is no input, or an input that returns no dishes, with the Baking Mad Foods toggle selected, then it should show a recommended page of dishes. | Great | No improvements |
| 4 | If the quit button is pressed, then the program ends | Great | No improvements |
| 5 | If the start button is pressed, and there is a valid input, then the screen is changed to one with a list of all the options | Great | No improvements |
| 6 | The options screen will be presented with a list of all the buttons of options, with a scroll bar on the side | Great | No improvements |
| 7 | The options screen also has a quit button at the bottom, which will return the user to the menu when pressed | Great | No improvements |
| 8 | If the user selects a dish option, then they are sent to a page with four titles of information, that being: | Great | No improvements |
| 8.a | Whether there is a vegetarian option | Great | Add a diagram |
| 8.b | The amount of people the dish serves | Great | Add a diagram |
| 8.c | The time it will take for the dish to cook | Great | Add a diagram |
| 8.d | The time the dish will take to prepare | Great | Add a diagram |
| 9 | This page will also have two buttons, a back button (that sends the user back to the options of dishes page) and a go forward button | Great | No improvements |
| 10 | When the go forward button is pressed, the user is sent to the ingredients page | Great | No improvements |
| 11 | The Ingredients page will show all the ingredients needed, and have two buttons (one of which will be a go back button and one of which is a continue button) | Great | No improvements |
| 12 | If the user clicks the go back button, they will go back | Great | No improvements |
| 13 | If the user clicks the continue button, then the user is sent to the next page (the instructions page) | Great | No improvements |
| 14 | If the user is on the instructions page, then the user is given a list of instructions needed to complete, alongside two buttons. A timer button and a finish button. | Great | No improvements |
| 15 | If the user presses the timer button, then the user is sent to a screen which has a selection of times in a scroll bar. | Great | No improvements |
| 16 | Once a time is selected a popup will start saying the timer has started and a timer will start for the given minutes, and then the popup will have an exit (the popup) button | Great | No improvements |
| 17 | If the user selects the finish button, then the user is sent to the main page to make another decision, and the dish the user made is stored in a text file. | Great | No improvements |

## End-User Feedback

Having completed the app, I showed it to the same user who I questioned earlier, Antonio. He then gave me the following feedback (which I have condensed into bullet points):

* The program meets all but one suggestion I made for you:
  + You should have emphasised more on the aesthetics and looks of the app
* The physical app does what it was meant to do, however the timer function could have a live stopwatch addition
* There needs to be more emphasis on the looks of the app, it looks “ugly”
* I like how I can view my previously completed dishes
* The addition of a recommended options for users is good, although it will not be used much
* I like the buttons; they are clear, and their functions are obvious
* The entire app could do with more colour and a colour theme

## Analysis of the End-User Feedback

Overall, the end user said that he was pleased with the app as it does what it needed to do, all in a convenient Android phone situation. The primary issue he had with the code was the lack of a colour scheme, which was an additional aspect I was going to add if timer permitted, however unfortunately time did not allow this to take place due to unforeseen circumstances. The colour scheme was important to him as it would make my app stand out in comparison to the other apps already on the app store. However, he was also aware that my app was one of the only ones with a built-in timer and the ability to see previously completed dishes therefore I already had a point of standing out. Furthermore, I was not allowed to upload the app onto the android app store due to the copyright issues that would pursue, as my app is scraping information from two websites, and I do not have permission from the websites to use that information in the form of an app.

I based the idea of having all of the instructions based on this user’s input, however it may be a good idea to add the feature to have the option to look at each instruction individually, as it makes for easier understanding for newer cooks.

The addition of the recommended options was solely so that I did not lack a feature that my competitors did, even though it is apparent that not many people truly use this feature when going onto a dish finding app, often times the users have somewhat idea of what kind of dish they would like to make, for example pasta.

## Suggested Improvements

I would like to include a live timer for the timer function, so that the user knows how many minutes have gone and how much time is remaining, however this would require a lot of processing power as it would involve a for loop with a time.sleep function called every second.

Furthermore I would like to dedicate more time later on for the aesthetic of the entire app and how it looks, as currently I do not like the colour scheme and would need suggestions from people who are a lot more artistically aware then I am, as for how the app should look.

Additionally, I would like to add images for all the dishes on the “dish options” screen later on as that would allow the user to make more of an educated guess, instead of simply just looking at the initial information for advice.