**IOT CA1**

**The Smart Coaster**



Jamie Lawlor

Caitlin Maguire

Luke Hilliard

Shahzad Shabeer

Version 3

26 October 2024

**Table of contents**

1. [**HARDWARE………………………………………………………………………………………………..**.](#Hardware_Section) **3**
2. [**DATA, DATA STORAGE & DATA PROCESSING…………………………………………………….**](#Data_Section) **4**

[2.1 Datagathering](#Data_Section)……………………………………………………………………………...………4

1. [**SECURITY & PRIVACY………………………………………………………………….**](#Security_Section)**................... 5**
2. [**THE USER, UI & TESTING……………………………………………………………………………….**](#UI_Section) **6**

**Hardware – Luke**

A table with numbers and symbols

Description automatically generated

**Light Dependent Resistor (LDR)**

By integrating an LDR into the surface of the coaster this will allow the system to know when a cup is placed on top. The cup will create a dark environment around the resistor. This type of resistor will increase and decrease its resistance depending on the light within its environment.

A close-up of a red and white device

Description automatically generated

**Data, Data Storage and Data Processing – Jamie**

**2.1 Data gathering**

The temperature sensor will be used to measure the temperature of the coffee. After each minute, the sensor will retrieve the temperature again and update the previous data. Using PubNub the sensor reading will be sent to the server so that it may be processed, and code can be written to use this data reading. The Light Dependent Resistor will be used to track if a coffee cup has been set on top of the coaster. Once the LDR recognises that the lighting of the coaster has changed it will send the data to the server to record that a coffee cup has been set onto the coaster. The data will be stored on a schemaless database. For this project we will be using MongoDB. We will use a third-party dataset from Kaggle to fill our dataset with basic data such as the names of the common types of coffee and add and drop columns as needed.

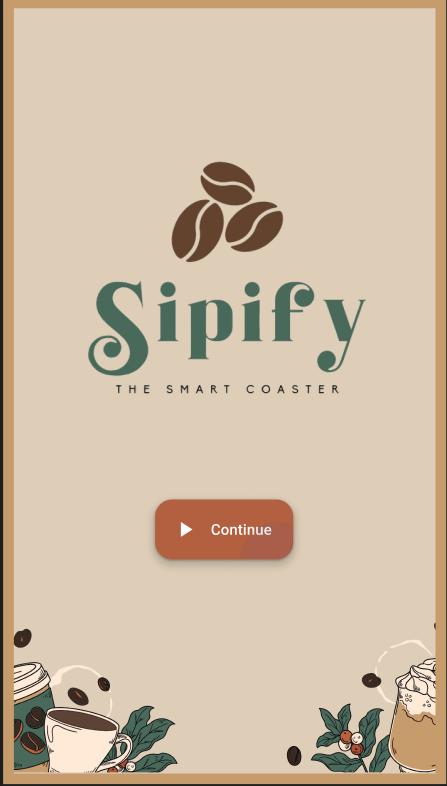
**Security and Privacy – Shahzad**

**The UI, User and Testing – Caitlin**

**Mock Up UI**

To create a mock-up of the UI for the smart coaster I used Figma. I have created various screens to visually show the different steps a user would take in the application.

Below I have included screenshots of the different screens in the prototype:

**Screen 1: Loading Screen**

The user will be faced with a loading screen when the application is run initially, once the “Continue” button is clicked, the user will be redirected to screen 2.

**Screen 2: Select Mode**

**A person making coffee in a coffee shop

Description automatically generated**

In the second screen the user can select a mode, the mode options are “Barista” and “Individual”. The Individual mode allows the user to complete the goal of the project which is merely to track the current temperature of their drink.

Whereas the Barista mode gives the user more control as they can perform different operations such as add, edit and delete drinks. Once the user has selected their chosen mode, they will carry on to the associated screen.

For example, if the individual mode is chosen the user will be redirected to screen 3. When the Barista mode is chosen the user will be redirected to screen 4.

**Screen 3: Individual Mode**

**A screenshot of a phone

Description automatically generated**

In individual mode whilst a cup is set on the smart coaster, this screen will show the user an alert or notification, a percentage of how full the cup is currently, and the temperature. If the cup is lifted off the coaster the application will go back to the loading screen.

The notifications which appear just under the cup image, will change accordingly with the temperature range.

For example, on average a cup of coffee is at the perfect drinking temperature within the range of 55 to 60 degrees. Within this range a notification “Drink while it’s hot!” will appear on screen, to alert the user of the coffees current state.

If the coffee reaches the range of 40 to 49 degrees, the notification may be “Consider heating up!” and so on for different temperature ranges.

The user can click the “Finish” button when they are finished their coffee, and this button will take them back to the loading screen.

**Screen 4: Select Coffee Type**

**A screenshot of a menu

Description automatically generated**

The barista can select the drink of the current order they would like to track. This data will be saved and displayed on screen in the next screen.

If the project was being created on a larger scope our idea would be to have the smart coaster connected to the café ordering system, this could send the data for the order over to our app and things like the drink type, order number and customer name would be displayed on screen. When the first order is complete, the next cup could be set on the coaster and the next order appears on the screen.

In terms of our UDP project this screen for manually selecting the drink type makes the most sense.

**A screenshot of a phone

Description automatically generatedScreen 5: Barista Mode**

In barista mode, the user has lots of options. Firstly, the barista has the same options as individual mode, I have not included the percentage of the amount in the cup as it doesn’t seem necessary in the case of a barista. It is assumed that drinks are freshly made and won’t have someone drinking out of the cup whilst it is set on the coaster.

Apart from this, the barista has alerts which will appear and change at different temperature ranges. The alerts may be a little different compared to the individual mode, for example “Ready to go!” seems more fitting for a busy barista aiming to get lots of drinks out to eager customers.

The difference between barista and individual mode, is the barista has more controls. The barista can select whether the drink is hot or iced in the controls on the left of the screen. This allows for the temperature to be read more accurately, for example if the barista has set an iced latte on the coaster, the application must show alerts that are related to cold drinks such as “Ice is starting to melt!” if the temperature has fallen below the ideal temperature for iced drinks. This can be applied for hot drinks also and ensures notifications on screen and temperatures tracked are read more accurately according to the drink type.

The barista can select the option of “Extra Hot”, allowing for special orders of extra hot drinks, in this case the alert can be updated to suit drinks which are above the ideal or optimal drinking temperature.

The barista user can view what drink they have selected, in a busy workplace like a café it can be easy for workers to forget what the current order is. Or for waiters coming to bring out drinks to customers, it is valuable for them to easily see on screen what coffee is currently on the coaster.

The barista can select the button on the top left corner, this button will take them to screen 6. Once a coffee has went out to the customer the barista can select the finish button to go back to the loading screen, until another coffee order comes in.

* Show screenshots from figma , maybe create a page for the add, edit and delete drinks options
* Explain what is happening in each screen , how does users connect, what happens when they pick up the cup for both individual and barista mode

**User Connectivity**

* What data is available to them – individual mode only has access to the current temperature – so not much data
* Whereas, for the barista mode – they have access to a list of coffee types , if a new drink is added it will be added into that list
* They can also view a list of min and max temperatures of each coffee type, this will be stored in the database.
* Different notifications will appear on screen , accordingly with the temperature so for example if a Cappuccino is at 65 degrees “coffee is ready to go!”

**Testing the UI**

* User types – briefly talk about the use cases
* How will we track project success – getting test users to roleplay as our user groups – give them a task and see how they interact with the user face , how long a task takes