**IOT CA1**

**The Smart Coaster**



Jamie Lawlor

Caitlin Maguire

Luke Hilliard

Shahzad Shabeer

Version 4

27 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**
   1. Mock-up UI ……………………………………………………………………………...………..
   2. User Connectivity………………………………………………………………………………….
   3. Testing the UI ……………………………………………………………………………...………

**Hardware – Luke**

A table with numbers and symbols

Description automatically generated

**Key Components:**

Light Dependent Resistor (LDR) & Electrolytic Capacitor

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 generatedA close-up of a capacitor

Description automatically generated

This type of input is considered analogueand with this we have a range of voltages from 0V to 3.3V. The Raspberry Pi 5 works with digital input so it is unable to detect what the exact voltage is.

By placing a capacitor in series with the resistor, the speed at which the capacitor charges can be speed up or slowed down. As the resistance changes, the capacitor charges at different rates which can be measured and translated into a digital signal the Pi can understand that reflect the analogue.

Tactile Button Switch

Positioned on the side of the coaster, the button acts as a backup if the LDR is unable to accurately detect a cup on the coaster, giving the user a simple override.

A group of small black and silver square objects

Description automatically generated

Temperature Sensor

Like the LDR, our temperature sensor will be integrated into the surface of the coaster to gather real time temperature data from a cup. From our findings, coffee stays hotter towards the bottom of the cup. This would give us more consistent data overall.

A close-up of a transistor

Description automatically generated

Circuit Diagram using Fritzing

**A circuit board with wires connected to it

Description automatically generated**

This diagram shows how we plan on using our Raspberry Pi 5 as the brain of our system. The LDR will detect changes in light and communicate this to the Pi, telling our system if a cup is placed on the coaster. A button is connected to override this feature if for some reason it fails.

The temperature sensor will collect real time data from a coffee cup and relay this back to the Pi. This data will be displayed on the application. The temperature data allows us to perform a ‘too hot’ or ‘too cold’ calculation, the result of which will turn on a corresponding red or blue LED.

**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.

**2.2 Data Storage**

For this project we will be using MongoDB as our schemaless database. First we will connect to the database using localhost to work with the database locally. A sample drink dataset will be loaded into the database that will provide us with different drink types. We can drop all redundant columns from this dataset for our project. The drink names will be stored in a “Drink Name column”. We will create a minimum temperature and a maximum temperature column and store the individual values in here. We will have a drink-type column that will store whether the drink is tea or coffee. Using MongoDB Compass, we can filter for duplicates in coffee types and remove them. We will need an alert column that will store values of the different alerts the server can call from the database to output to the user on the front end. Our final column for the database will be a temperature type column where we store whether the drink is a hot or an iced drink. To connect the database to the server we will utilise flask with MongoDB Atlas using the pymongo library. We can then run queries in our flask app such as sending a notification to the user when their drink is getting too cold.

**Security and Privacy – Shahzad**

**1.Physical Security**:

The Smart Coaster will be housed in a hard shell like enclosure to protect against accidental spills and to protect the internal sensors and components. This enclosure will have tamper evident seals(like in the smartphones) to indicate unauthorized access. the internal components will be secured with screws to prevent easy disassembly

#### **2 .Data Security**

**Data Encryption**:

All data transmitted between the Smart Coaster and the Flask web application and the server will be encrypted using TLS (Transport Layer Security) with at least 256bit encryption. This will safeguard data in transit against eavesdropping and tampering. Sensitive data stored in the database will also be encrypted using AES256 encryption.

**Flask Application Security**:

The Flask application will include proper input validation and sanitation to protect against common vulnerabilities, such as SQL injection and cross-site scripting (XSS). Flask will be configured to handle errors securely without exposing sensitive information.

**Database Security**:

The connection between the Flask application and the database will use secure credentials. Database access will be restricted to the IP address of the AWS server minimizing the risk of unauthorized access.

#### **3.Data Privacy**

**Data Minimization**:

The Smart Coaster will only collect data necessary for its functionality including temperature readings and usage data. No personally identifiable information will be collected. Given that our application does not require login or sign-up ensuring compliance with privacy regulations such as GDPR.

**Anonymization Techniques**:

Data aggregated for analysis will be anonymized to ensure user actions cannot be traced back to individual users. This will help protect user privacy

**User Consent and Transparency**:

When users first interact with the application they will be presented with a clear consent notice detailing data collection practices and usage. Users will have the option to opt-in or opt-out of non-essential data collection

**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:

A screen shot of a phone

Description automatically generated**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, and the current temperature of the drink. If the cup is lifted off the coaster the application will go back to the loading screen. The current temperature of the drink is shown just below the notifications section, this will be updated in real time when the temperature of the drink changes.

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. 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 current temperature of the drink is shown just below the notifications section, this will be updated in real time when the temperature of the drink changes.

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.

**A screenshot of a phone

Description automatically generatedScreen 6: Barista Controls**

This screen is almost like an admin dashboard for the barista, they have access to adding, editing or removing drinks, the database will be updated accordingly. When the user clicks the add, edit, remove or show all buttons it will take them to the corresponding screen.

**A screenshot of a menu

Description automatically generatedScreen 7: Add a new Drink**

The barista can add a new drink, there is input boxes for the coffee name, drink type and minimum and maximum temperatures. The drink type options are coffee or tea, this allows the drink data to be separated based on the type. The data entered will create a new drink which will be saved to the database, the “Show All” screen will also be updated and will display the new drink along with the list of other drinks in the database. The changes will be made when the user clicks the add button.

**A screenshot of a menu

Description automatically generatedScreen 8: Edit a Drink**

The user can edit a drink that is currently in the database, the list of drinks will appear in the drop-down list for selecting a drink. The user can change the drink type, the options are Coffee or Tea. They can also edit the name for example changing the name of “Flat White” to “Caramel Flat White”. The minimum and maximum temperatures can also be updated. The data that the user updates in this screen will be pulled into the database. The change will be made when the user clicks the edit button.

**A screenshot of a phone

Description automatically generatedScreen 9: Remove a Drink**

In this screen the user can delete a drink, this will also be deleted from the database and the show all screen. All the drinks that are in the database will appear in the drop-down list, the user can select the drink such as “Gingerbread Latte”, then click the “Delete” button to complete the task.

A screenshot of a menu

Description automatically generated**Screen 10: Show All Drinks**

In this screen the barista can view all available drinks, this data will be brought in from the database and displayed nicely. The data includes the drink name, we will have different coffee and teas in the data. A minimum temperature and maximum temperature will be saved for each drink in the database. The optimal temperature will be within the range of min and max. While in barista mode the user can view the data for each drink.

The “Back” button will take the user to the barista controls screen.

**User Connectivity**

It is quite a simple process for the user, the steps to connect include:

* Set the cup on the coaster
* Open our Sipify Smart Coaster website
* Press the “Continue” button
* Select the mode, the options include “Barista” or “Individual”
* View the current temperature of drink

**Data**

Both user types can view the current temperature of the drink. Each user can view alerts or notifications on screen, the alerts will appear accordingly with the temperature range for example if a Cappuccino is at 65 degrees “coffee is ready to go!”.

**Individual Mode**

In individual mode the user only has access to current temperature in terms of data, this changes in real time when the temperature of the drink on the coaster changes.

**Barista Mode**

The barista user has access to the list of tea and coffee drinks, this includes the drink name, minimum and maximum temperature. When the user decides to add or edit a drink, the database will be updated, and the new data will be stored.

**Testing the UI**

**User Types**

**Office/Desk bound workers:**

This user is sitting at a desk all day, drinking caffeine to remain motivated is essential. They may forget about their coffee while doing their work.

**Barista:**

Busy baristas working in a fast placed environment, need to work quickly on multiple coffee orders. The smart coaster can track drink temperatures to ensure coffee is given out to customers at the perfect temperature.

**Coffee Enthusiasts:**

There is a large culture of “coffee nerds” who would enjoy learning any additional information about their coffee, such as tracking the temperature and discovering the perfect drinking temperature.

**Testing**

To track project success, we will carry out methods of user testing with real life users. We will have someone roleplay as an office worker and another as a barista user.

We will give each of the user’s specific tasks, for the office worker they will use individual mode and track the current temperature of their coffee.

For the barista, they will use barista mode and tasks include tracking temperature, add, edit, delete and show all drinks. We can record the time taken for the users to complete each task and observe how they interact with the user interface.

After completing the roleplay, I will ask users to answer some questions using Google Forms, this will include questions like “How was your experience with Sipify?” and “Do you have any feedback?”, and other questions to get a feel for the user’s experience.

The project will be deemed successful if both users feel they had a positive experience interacting with our user interface and if they can complete the tasks at ease. The main goal of our project is to allow users to track the temperature of their tea or coffee, if they have difficulty carrying out this task it is essential that we as a team take the steps to uncover why and how we can overcome this.

In the case that the project is deemed unsuccessful, we will take onboard user feedback and adjust project features accordingly to ultimately improve user experience and to fulfil our project objectives.