Smart Menu

////Preliminary Draft

Smart\_Menu is a lightweight and simple Internet-of-Things interface that connects a restaurant menu database directly to user preferences. Combining the functions of a menu database service from food apps (e.g Yelp) and diet trackers (e.g. [insert app here]), Smart users are able to preemptively preview what dishes served in the restaurants fit their dietary profile, making dietary tracking more convenient.

An Overview

A simple summary of the Smart\_Menu pipeline across the three platforms (desktop, web, m5 Stack) is as follows:

a. Desktop: This is the restaurants' end. Restaurants register with the service and upload their menu to the web server.

b. Web: This is the developer/service provider end. The web service registers restaurants (generates the unique restaurant ID) and stores the database to be accessed by users.

c. m5 Stack: This is the users' end. Users set their dietary preferences on the app, which then provides assistance when navigating the restaurants registered on the service as they plan their order (e.g. by highlighting dishes that exceed their intended calorie intake or dietary restrictions).

1. System Design [1000 words]:

a. Architecture of the entire system

b. Object-Oriented design of key sub-systems (e.g. Desktop Application, Web Application etc.) c. Requirements of key sub-systems (in the form of selected user stories) d. The evolution of UI wireframes for key sub-systems e. Details of the communication protocols in use (including a rational for your choice) f. Details of the data persistence mechanisms in use (including a rational for your choice) g. Details of web technologies in use (including a rational for your choice)

(b. Object-Oriented design of key sub-systems)

*M5 Object oriented design*

The smart menu design featured the List and menu classes. Since the menu must be able to display, select and flag an unknown number of restaurant names or menu items, object-oriented design seemed to be the most appropriate approach. The code has an array of list and menu once the items have been received via JSON. The classes allow variables associated with each menu item e.g. the name. These variables could then be manipulated for instance selecting an item and using a method to change the selection integer or changing the flag depending on Booleans for dietary requirements.

*DESKTOP Object oriented design*

This model is for a restaurent which could connect with web - sending its restaurant name, get its restaurant id and then send its menu items.

So there are at least three main object:

class Restaurant - which contains (String) restaurant\_name, (String) restaurant\_id;

class Food - which contains (String) food\_name, (int) food\_calories, (boolean) if\_vegetarian, (boolean) ifcontain\_gluten, (boolean) ifcontain\_nuts;

class Menu - which contains an arraylist, Food[] foods.

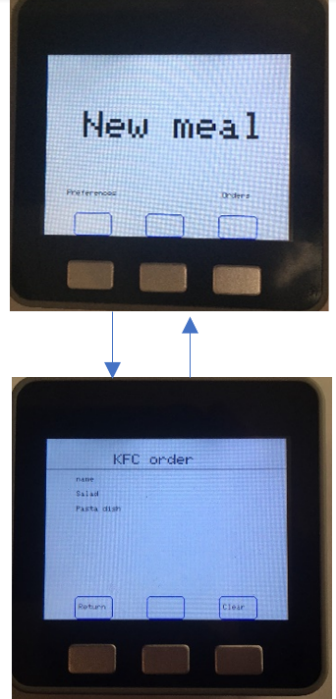
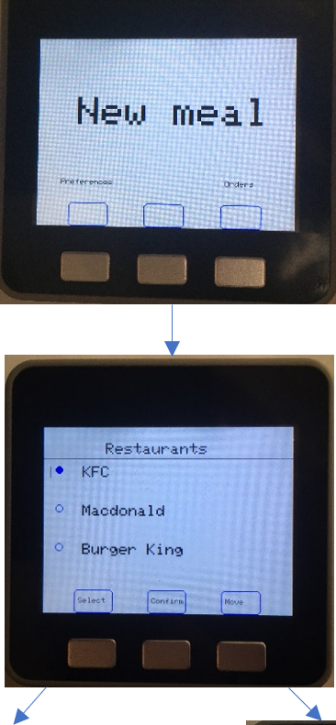
These functions make the system of a restaurant\_menu:

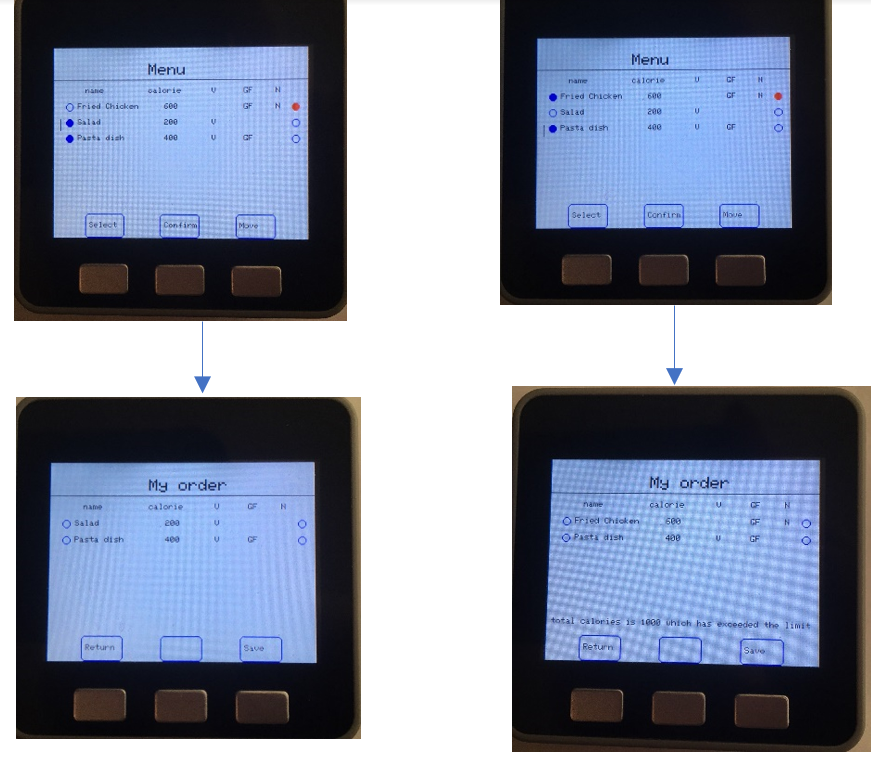
* adding some new food [add\_item(), doAdd() function]: pass name, calories, vegetarian, gluten, nuts as parameters and create a new food object;
* editing existing food [edit\_item(), doUpdata() function]: just create a new food object like add\_item, and then do some update, to send new items as json package;
* deleting existing food [delete(), doDelete() function]: we use menu object, and delete the selected index;
* doing registration [registration(), doRegister() function]: send menu item in the format of json Package.

(d. The evolution of UI wireframes for key sub-systems)

*DESKTOP UI wireframes*

* EDITION 1.0: have 2 pages for desktop app
  + A restaurant menu page: show the items of menu, and for registration
  + A registration page: the users can input restaurant name and choose 'register' and 'cancel'
* EDITION 2.0: have 2 pages, which including a restaurant menu page, a registration page
  + A restaurant menu page: show the items of menu, and the user can select the specific items of this menu, which helps user to delete. Another function in this page is to register.
  + A registration page
* EDITION 3.0: have 3 pages, which including a restaurant menu page, a registration page and a add/edit page
  + A restaurant menu page: the user can select the specific items not only for deleting items, but also for edit items. Totally there are 4 functions in this page: ADD, EDIT, DELETE, REGISTER\_NEW (means the regiatration for a new restaurant)
  + A add/edit page: the user could add the information of particular food item into the menu, which includs the name of food, food calories, and set if this food contains meat, nuts and gluten...the user could also edit those information in a page that has the memory of privious food information
  + A registration page





Evolution of UI M5

Due to the limited buttons we had to divide access to pages between 3 separate branches as seen from the diagrams above. There is a consistent layout for all pages to make the user experience more intuitive. This layout consisted of a title at the top and below displayed whatever functions the page served. At the bottom there are buttons squares with names attached describing the certain function of the button, whether it be moving to another page or saving an order. The buttons on screen were designed to line up with the buttons on the physical device so that the user knew which button did what. For the flagging of menu items red circles was chosen to be used. The M5 stack provides several ways to do this but it needs to be non-obtrusive and striking. The red colour draws the user’s attention and the small circle does not take up too much space.

Communication protocols

(e. Details of the communication protocols in use)

The JSON structure can be seen in the diagram above. The queryID JSON Number was used so that each system could determine whether they should ignore the JSON string or interpret the string. For instance, when the M5 stack sends a queryID = 10 to request a restaurant list, the desktop app will ignore the query but the web app will have an if statement to interpret the contents and send back a list of restaurants. The resList is an array of JSONObjects which stores the information about the restaurant such as name, ID and menu Array of JSONObject. The menu JSONObject stores information about the individual foods in the menu. These two arrays store information that will be useful for the user to inform them in their food choices. The restaurantSingle Object is identical to resList only it is not an array and only stores the information of one restaurant. This was done to make the code in the M5 stack have less nesting and easier to understand. The menuSize and listSize JSON integers were also used to make coding easier as when displaying the items in a list in a for loop. Lastly, the conID integer was a random digit from 1-1000. This was used to make sure when multiple m5 stack were contacting the MQTT server they would only M5 stack would interpret the JSON string.

As for the desktop app, the users usually send the data containing menu of food in the restaurant! We have a format for this transformation. When new users want to register this restaurant, it should send its reataurant name using this format to the Web and set the queryID is 30; and then the manager of web would return a new json package which contains a new restaurant id and using queryID 31; when this restaurant wants to send the new menu to web after adding, editing or deleting, it would use the same format but set the queryID to 40.

(f. Details of the data persistence mechanisms in use)

*M5 Data persistence*

The ardiuno-esp32 Preferences library allowed us to save the contents to flash memory on the M5 stack for permanent storage. This was necessary for the M5 stack to save the user dietary requirements, maximum calories per meal and the contents of the order they want to place. The preferences library had partitioning for the variables and each variable could be assessed using a key. The dietary requirements were saved as integers with 0(false) and 1(true). The max calories were saved as integer as well and the menu items and restaurant name were saved as a string.

*DESKTOP Data persistence*

* We have the need of persist data about menu and single restaurant in the daily life.
* We use json file (menu.json, restaurant.json) as API instead of real database.
* In processing desktop, we use functions such as loadMenu() and loadReataurant() to load the json file, and to use this database.
* When the user add/edit/delete the food, or register a new restaurant, the data would be changed in the saved json file!

ATTENTION: *Data Structure in json* would be uploaded in processing/data folder!

* menu.json: [ { "foodName": "salad", "containsGluten": false, "containsNuts": false, "calories": 500, "vegan": true } ]
* restaurant.json: { "name": "KFC", "id": "lawp" }

*Brief Overview of M5*

The M5 stack was designed to be a smart menu. There are three main task the menu had to accomplish. One was to record the dietary requirements of the user and set the maximum calories per meal. The other was get the restaurant list and menu items for that restaurant and flag any items that do not meet the dietary requirements and when the meal is over the calories limit. Lastly the menu must be able to save and display the last selected menu items.

*Brief Overview of DESKTOP*

This desktop app is for restaurant chef, which can implement a little bit functions: add new dishes, delete existed menu, send new items to web and register a new restaurant. It could use JSON Package to communicate with the web database, and send the information of dishes including food name, food calories, if it is vegetarian food(use boolean), if contains nuts, if contains gluten(use boolean) as well as restaurant name and ID.

1. System Implementation [1000 words]: a. Breakdown of project into sprints (showing the users stories implemented in each).

The project first began with work on the users' (m5) end. Once we settled on the general direction of our project (a combined menu browser + diet tracker) we started to outline what 'front-end' features such an interface would include. We initially envisioned a system that worked locally in the restaurant, where participating restaurants would encode the food date onto a unique identifier on each dish (conceptually, a QR code, or colored plate e.g. a sushi bar), to be scanned by restaurant patrons when ordering. This is captured by the captured footage of the paper prototype:

This idea was quickly scrapped, and we moved to a fully online system. Among other reasons, we considered the burden of implementation for restaurants to be far too high. We also realized that the physical step was unnecessary. Put simply, they would achieve the intended result of recording their meal by saving it from the menu on the app, without the need for local verification.

SPRINTS:

* ideation 1.30-2.3 We firstly got the idea that this M5 can be used to calculate calories!
* Prototype 2.4-2.10 We decided to use QR-CODE to help people to realize the calories in each food, and then, we plan to use M5 to calculate the total calories. We also have videos at this stage!
* play with processing 2.11-2.17
* play with M5 2.18-2.24
* consider with the whole system 2.15-3.4 This time we consider the 3 different parts of our system; at last we decide to use M5 for selecting menu and calculating calories, to use web to send menu to M5 stack, and to use desktop for single restaurant to add, edit and delete their menu. In addition, at about 3.2, we also have our GitHub firstly.
* data communication 3.5-3.15 Sam write json package for whole team at first
* the individual work 3.16-4.16 Have a meeting on 4.16 to test MQTT
* group testing 4.17-4.22 lots of meetings to ensure the communication
* write reports 5.1-5.8

[Something about the stages of designing the basic layout of the m5]

b. Details of how you evaluated your designs (techniques used & awareness of their limitations) c. Discussion of Social and Ethical implications of the work

1. Project Evaluation [800 words]: a. Reflective discussion of the success of the project b. Discussion of future work (in terms of design, development and evaluation) c. Reflect on the working practices of your group, how well they did or did not work, e.g, management of issues, communication, Agile (etc). d. This is a chance to reflect on how coronavirus has affected your project (remote working practices etc)

The project was a success, so far as achieving its intended goals. We successfully implemented all the primary intended features, most crucially in a manner that was conducive to future iteration and improvement.

Among considered directions for future work would be a persistent database for the web server. This would certainly be essential in a true final commercial product but was excluded in our project scope. This would likely be a sim. Another significant additional function would be a feedback/report system. There is currently no built-in form of communication between the intended user groups. While it was excluded on the grounds that it's not a core feature, this functionality is important in any commercial ready product to collect userbase feedback and communicate sentiment/service disruptions. Finally, significant additional UI/UX work could be made to make the product more visually appealing.

The group worked well to accomplish its goal despite numerous challenges. Our design sprints demonstrate that we practiced AGILE design principles from the very start, steadily building up our product in layers and organizing 'functionality test' meetings regularly.

Evaluation M5

All the main functions of the M5 were successfully implemented and during the testing of the system the M5 performed as intended. Since the M5 was the system that had to initialise the networking. When buttons were pressed, they had to produce and send a JSON String to the server. Once the response is received then the M5 would move onto the next page. This response was very rapid during testing worked without flaw. Though an issue may arise if the M5 does not receive a response back from the MQTT as the M5 does not notify the user when this has occurred. Whereas during the initial setup of the M5 there is message displaying connecting which will notify the user when M5 cannot connect to the WI-FI. One of the M5 issues was it is not suited to be a menu due to the size of the screen. We had to use the smallest text size to fit as much menu items in the screen as possible. But this may cause difficulty in reading for some users and there had to be a limit on the number of menu items. Another UI issue was the cursor had to be made from a line which once again might be difficult to see. Though overall the UI for the M5 is intuitive for the user due to the consistent layout (write about the layout in UI wireframe). The saving to permanent memory was successful for dietary requirement and max calories, for the menu it would have been better to save the menu as an array instead of multiple strings. For the structure of the code a switch was used to access each page, each of these had functions to display the UI. Some pages had reused functions such as draw\_buttons() and line\_cursor(). Of course each page had separate functions for specific tasks such JSON\_publish\_menu(), save\_order() and set\_cal(). So overall the code structure makes the system very maintainable for future updates.

Future work

The most obvious feature to improve for the M5 would be to increase the maximum menu items. Both saving menu as an array and implanting a scroll feature on listing pages would rectify this. The scroll feature could implement using the built-in gyroscopes on the M5. Another useful feature to implement would be to display the dietary requirements and max calories that have set in all pages so that user does not have to enter the preference branch to find out. Better error handling when the M5 has not received a JSON String. All pages have a use for each button except for my order page were only 2 buttons were used, it would have good to implement another feature. Obviously, this is a prototype in a commercial product with a touch screen interface it would be possible to have less of branch layout and some pages could be combined. Also, the menu could be incorporated into a better design like being incorporated into a table in a restaurant.

////Unedited Assignment Brief

1. System Design [40pts]: a. Architecture of the entire system b. Object-Oriented design of key sub-systems (e.g. Desktop Application, Web Application etc.) c. Requirements of key sub-systems (in the form of selected user stories) d. The evolution of UI wireframes for key sub-systems e. Details of the communication protocols in use (including a rational for your choice) f. Details of the data persistence mechanisms in use (including a rational for your choice) g. Details of web technologies in use (including a rational for your choice)
2. System Implementation [40pts]: a. Breakdown of project into sprints (showing the users stories implemented in each). b. Details of how you evaluated your designs (techniques used & awareness of their limitations) c. Discussion of Social and Ethical implications of the work
3. Project Evaluation [20pts]: a. Reflective discussion of the success of the project b. Discussion of future work (in terms of design, development and evaluation) c. Reflect on the working practices of your group, how well they did or did not work, e.g, management of issues, communication, Agile (etc). d. This is a chance to reflect on how coronavirus has affected your project (remote working practices etc)