The University of Ottawa

CEG 4912 - Computer Engineering Design Project I

https://github.com/SantiagoCely/robotDeliverySystem

Submitted by:

Amen Abshir (300035421)

Harouna Sylla (300086991)

Marc-Frédérick Tremblay (8603273)

Michelle Tang (300010861)

Santiago Cely (300038950)

Submitted to:

Emil M. Petriu, P.Eng.

Date of Submission: December 8, 2021

Table of Contents:

Table of Figures:

1 Project Charter

1.1 Project Description

Food delivery automation is slowly being introduced in the food service industry where it will only grow on a larger scale as time progresses. The main idea is to introduce robots as a way of creating a more efficient environment for restaurants. The CEO of Yum Brands believes that Artificial intelligence could replace humans in the food services industry by the mid-2020s. The main purpose of this project is to develop a robot that will perform various tasks within a restaurant to improve efficiency, lower possible costs that arise from hiring staff members, provide a better experience for customers and decrease the possibilities of human error that result from different tasks within a restaurant. Integrating robots in a restaurant allows businesses to have a more stable scheduling process as predictive scheduling eliminates unaccounted-for complications that may arise from manual processes and human emergencies. The robot to be created will be able to navigate through a crowded restaurant and narrow spaces, decreasing the chance of human accidents. Restaurant automation also has the chance of carrying out tasks that would rather be time-consuming to create a more reliable environment all around. Restaurant automation in this project will also benefit restaurants by being more secure, as features will be added to ensure that any transaction involved in a credit card fraud will be immediately reported to the authorities.

1.2 Project Rationale

Following the pandemic, many restaurant owners are still struggling financially. Inefficiencies related to labour reduce considerably the profitability of such businesses. Integrating automation in the food service industry will improve operating efficiency by reducing cost and error rate which will in turn improve the overall customer satisfaction while yielding higher profit margins for the owners. Furthermore, it would reduce the circulation of servers in the dining room thus helping owners respect social distancing rules during this period of pandemic.

1.3 Expectations

The goal to be achieved within this project is the successful development of a food delivery robot to be integrated within restaurants performing various tasks while decreasing person-to-person interactions which results in a lower chance of contracting COVID and providing a safer environment. This project allows restaurants to utilize a newer method to improve efficiency, decrease some costs that owners face every day while providing a unique and enjoyable experience for customers.

Being a part of an engineering program, we as students are always striving to learn and work with new technologies as well as acquire more experience in both the hardware and software side of engineering. This project allows us to strengthen our programming skills that were acquired in the past few years of our education along with familiarizing ourselves more with the fundamental hardware components and mechanical parts required to develop a functional robot that carries out the tasks it was set to perform.

Developing a robot is a concept that is new to the members of our team and is a great opportunity for us to learn and acquire new skills that can be applied later in our future careers as engineers. Robotics in the food industry is being utilized more as technology advances every day. We will be using different programming skills that we were taught during our time in the program to create an android application that will be available to use by the customer and staff members of the restaurant. Our programming skills will also be beneficial while working on connecting the hardware aspects of this project to the “brain” of the robot, as well as finding quicker and more efficient solutions to any software obstacles we may face during the project. The concept of robotics is a way for us to learn how to not only work on the software knowledge obtained through the program, but apply the theories learned about computer architecture to practical applications to become more familiar with them for future references.

Due to the fact that robots are a large-scale project and different sets of skills are necessary to successfully complete it, we as a team will be enhancing our communication skills as we are constantly required to consult with each other to ensure that we have a clear understanding of all the tasks that need to be completed, as well as provide help when further explanation is needed. Additionally, we will be enhancing our organizational skills through the scrum methodology we decided to employ as it allows us to have a guideline of the tasks to be completed at various dates. We hope to gain lots of experience and different skills through this project as well as create a great team dynamic to enjoy the process.

1.4 Market

1.5 Clients and Participants

Clients and Participants

The food delivery robot to be implemented targets one client, restaurant owners, and one participant, customers of restaurants. As new advances in technology are seen every day, businesses such as restaurants are given the opportunity to incorporate AI technologies in order to increase efficiency, reduce labor costs while maximizing profits and decreasing human error that result from regular tasks. Statistics show that 74% of business owners confirm that automation saves and 58% believe that automation increases financial opportunities. With the implementation of a food delivery robot in a restaurant environment, owners are also able to decrease stress levels as automation allows them to work with a more predictable employment schedule while avoiding any complications that arise from human scheduling such as sick days, emergency situations and vacation days. It also frees servers from unstimulating and repetitive tasks, giving them more free time. This rise in productivity will stimulate the economy, offsetting the loss of jobs that is caused by automation by creating more engaging jobs. A food delivery robot also interacts with customers in several ways. The robot will bring food to the customer which will put many customers at ease during the pandemic we are facing as they will not have to interact with people who may be carrying COVID while unaware of it. Customers are also able to create an account and look at reviews from previous customers to get more of an understanding of what menu items they might enjoy.

2.0 System Requirements Specifications

The following functional and non-functional requirements are essential to the successful development of our food delivery robot along with meeting all of the project's expectations.

2.1 Functional and Non-functional Requirements

- Robot must detect oncoming staff and customers and stop when necessary (no spillage from tray).
- Robot must recognize table numbers as well as bring/take food from table to kitchen and vice versa.
- Robot must be able to move with 360° motion. (Left, right, forward, backwards).
- Robot must have a retractable tray that allows for an easy removal and serving of dishes.
- Robot must alert authorities if there is any indication that a credit card used in a transaction is involved in a credit card case.
- Robot must accept different methods of payment from customers. (Cash or card).
- Robot must calculate the best path to reach a certain point.
- Robot must interact with the customer verbally.
- Robot must show customer the menu, review order and show employee the table order as well as allow them to make any menu modifications.
- Robot could easily create let client create an account and let him rate the food and/or the experience
- Robot could gather data about what people order and allow employees to see the data gathered so the restaurant can improve their menu
- Application could be straightforward, easy to use and has smooth transitions (for customer side as well as employee side)
- Robot could use a facial recognition algorithm to be able to recognize people that have been reported for not paying at restaurants

-

2.2 Constraints and Limitations

- Customers can easily create an account or continue as a guest while being able to rate menu items and add comments at the end of the dining experience.
- Staff mode will enable restaurant staff to easily modify menu items or prices as required.
- Staff mode allows restaurant staff to review data and comments by customers to further improve the customer experience in the future.

3.0 Detailed Design

3.1 Software Architecture

3.1.1 Software Design

3.1.1.1 Image Processing Unit

The Image Processing Unit is responsible for processing all data from the camera module and process this data. Once processed this data will be forwarded it to the High-Level Controller where it will be used to take the appropriate decision. The Image processing Unit consist mainly of an object detection and facial recognition model as well as some code to appropriately preprocess input data and label the predictions appropriately. The image processor is central to meeting the functional requirements of the robot, for example it has to send a signal to the High-Level controller when it predicts that a customer needs help then the appropriate command is taken. To implement the object detection module, we will be using a machine learning model, specifically a convolutional neural network (CNN). Instead of implementing our own CNN we opted to train our model on an existing architecture called You Only Look Once v4 (YOLOv4), an extremely fast and accurate object detection architecture. We chose to train our model using four classes: Person, Table, Plate, Food. The number of classes will expand over time, but these initial classes are the type of object that will be mostly encountered by the robot. The data is sampled from the MS COCO dataset, a large-scale object detection, segmentation, and captioning dataset published by Microsoft. Once trained we wrapped our model with TensorFlow and deployed it on the cloud where we continuously test our model.

3.1.1.2 Path Planner

The robot must forge different trajectories depending on the requests received from the high- or low-level controllers. This can range from going towards a customer needing help to going to the charging station where the chargers are located. The path planner is not hard-coded as it would be hard or close to impossible to consider all the different types of restaurant layout and obstacles the robot will face. Instead, we use a deep learning model that employs reinforcement learning. In this model, the agent (robot) takes an action in an environment, then this environment updates its states and computes a reward that will be fed to the agent. The agent takes another action that will maximize the reward. This feedback loop makes it so the agent will, after training, find the best set of actions that will maximize the reward.

To start training, our model we first need to create an environment. We used PyBullet, an easy-to-use Python module for physics simulation, robotics and deep reinforcement learning based on the Bullet Physics SDK, to simulate a restaurant environment. Once we had an adequate environment, we chose the states the robot can be in, the actions it can take and the rewards that will be computed. We then proceeded to train the agent using the TF-Agents library's Deep Q Network (DQN) implementation where a neural network model can learn to predict QValues (expected returns) for all actions, given an observation.

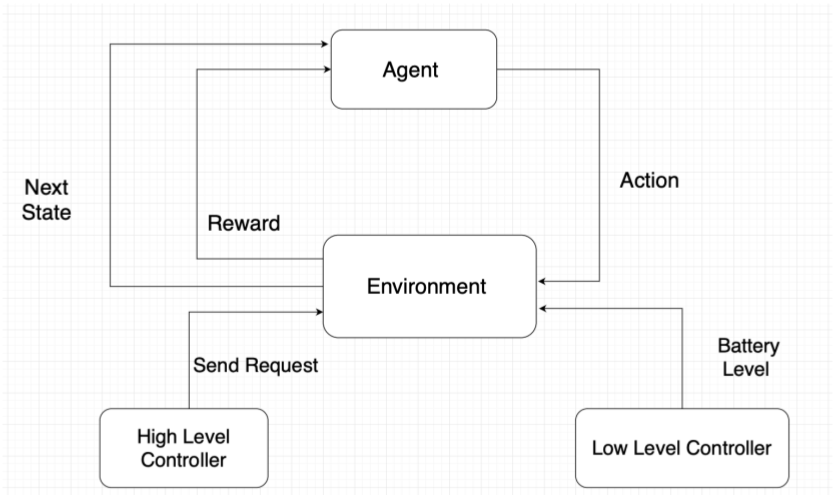


Figure 1: Agent-Environment interaction

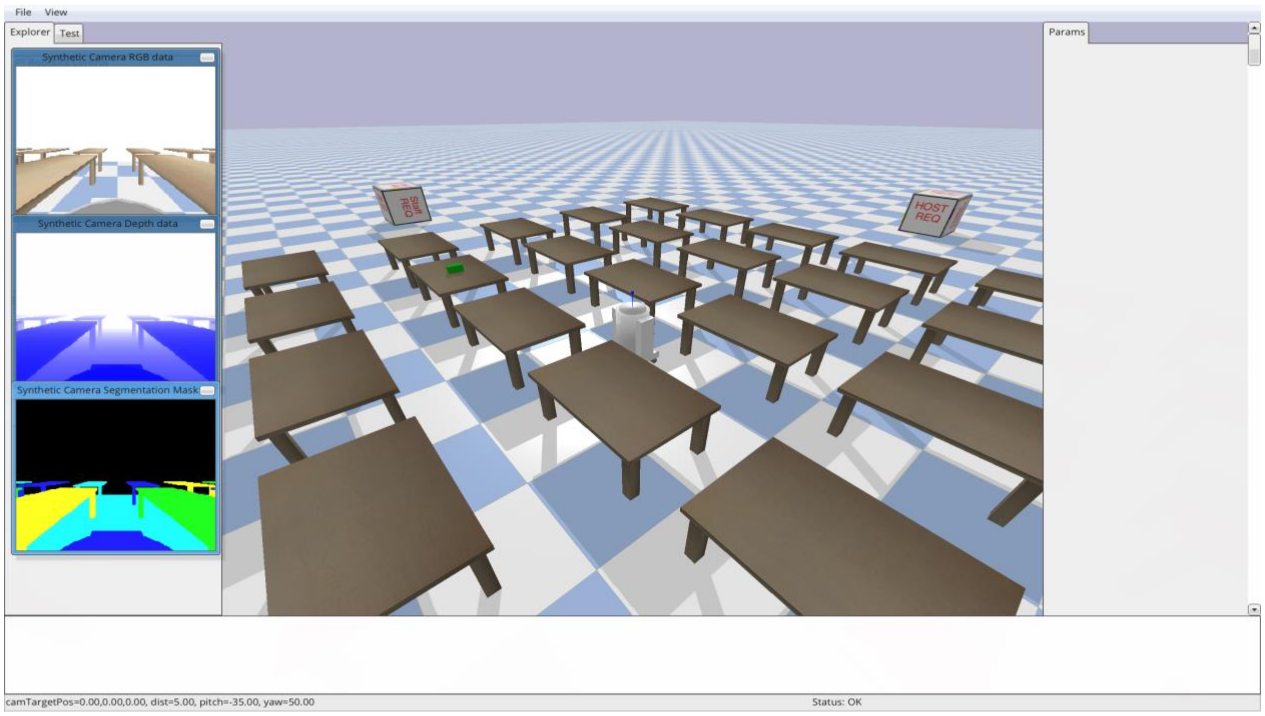

File
View
Explorer
Test
Params
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Staff REQ
HOST REQ
camTargetPos=0.00,0.00,0.00, dist=5.00, pitch=-35.00, yaw=50.00
Status: OK

Figure 2: 3D environment of a generic restaurant layout Simulated using PyBullet

3.1.1.3 Natural Language Processing

The natural language processing (nlp) is a functional requirement of our robot. It is the function that allows the robot to engage conversation with its entourage. So, that means the robot should understand user's message and be able to respond to them. Thus, that means we should use machine learning for the robot's nlp management. Thereby, we opted for rasa framework which already has machine learning functions built in. And by using these functions we can interpret user's message and respond appropriately to them.

Moreover, we chose to design a chatbot individually since the rasa doesn't include voice-based nlp design and that means the input and output of the nlp at this level should be text-based. But we designed another module aside that's called voicebot which will be responsible of collecting user's message in audio and convert it into text and send it to the chatbot through a local port. In the other hand it takes the chatbot's output in text and convert it into audio to be displayed through a speaker using python library *pyttsx3*.

The advantage of using the rasa chatbot is that it comes with another tool called rasa-x. This will allow us to fine tune the design of the nlp by testing it and at the same time collecting additional data from users. This is understandable especially if the effectiveness of a nlp is based on whether the type of data trained is appropriate for the context (restaurant here) or not and the best way to collect appropriate data is to share the chatbot to “real users”. Thus rasa-x offers functions that allows us to access all the conversation the chatbot had with users and we can review the behaviour of the chatbot. So, all the wrong interpretations of the user’s message and the wrong response of the chatbot can be corrected with a simple click then we retrain the model so next time it will respond appropriately to the same message. Thereby we will keep sharing the chatbot to users until we are ready to deploy all the robot components on the raspberry Pi.

On the other hand, the rasa-x requires rasa to be up and running continuously so that it can be shared and that’s why we deploy it into the google cloud platform. This will facilitate the deployment on the Raspberry Pi as well.

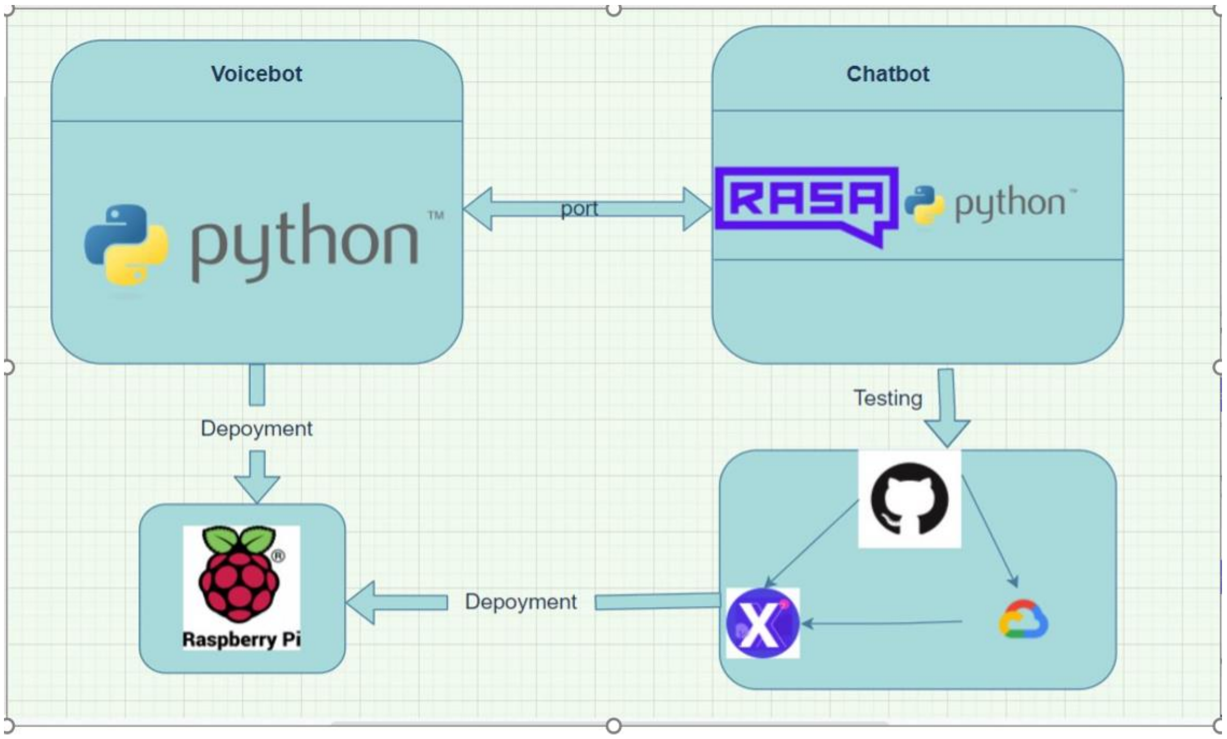
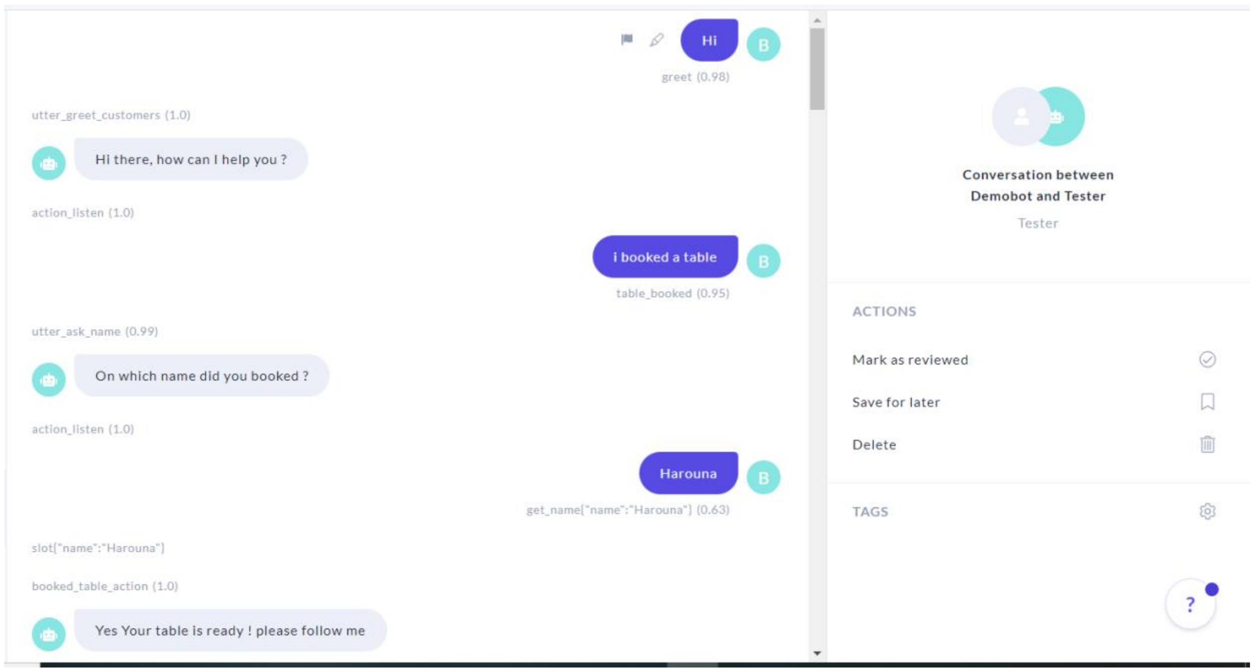


Figure 3: Design process of natural language processing



Hi
greet (0.98)
utter_greet_customers (1.0)
Hi there, how can I help you ?
action_listen (1.0)
i booked a table
table_booked (0.95)
utter_ask_name (0.99)
On which name did you booked ?
action_listen (1.0)
Harouna
get_name{"name":"Harouna"} (0.63)
slot{"name":"Harouna"}
booked_table_action (1.0)
Yes Your table is ready ! please follow me

Conversation between Demobot and Tester
Tester

Figure 4: rasa-x interface: conversation between the chatbot and a user

3.1.2 Front-End Design

The interface of the frontend side of the project will be a restaurant-ordering e-commerce Android application. The application will use the Hybrid App Development, with the Native Android as the wrapper and Ionic's Capacitor as the frontend interface framework. This application will use the MVC framework model since the user will be interacting with the application to obtain menu items, order statuses, etc.

Ionic's Capacitor allows for a faster development as the application essentially becomes a web application that is coded in Javascript, HTML and CSS, that will be encapsulated by a native (Android) shell. Using this technology will greatly reduce development time and increase productivity.

The frontend will be connected to the backend, so the system can send and receive all the information required to facilitate a food-ordering experience (e.g., viewing menu items and categories, viewing the cart, etc). It will also allow restaurant staff to modify the menu and restaurant layout at will and to view analytics concerning the menu items and customers.

The frontend application will have two user interfaces: Customer-mode and Staff-mode. The Customer-mode is default; administrators must login to access Staff-mode. Since the database does not store the datatypes as Objects, and returns them as Observables once queried, we will be defining Observable Object Interfaces for our datatypes. Services provide the modules with communication with other modules or the database. To help facilitate the development of the application, the application will be modularized. Each functionality of Customer mode and Staff mode will have its own module and thus page.

The following Object Interfaces are required:

- **CustomerAccount:** Where the information about customer accounts is saved. Used purely to increase customer experience by keeping track preferences, contact information and past orders, as well for analytics on administrator's side.
    - **customerId:** ID of customer

  - **name:** Name of customer
  - **email:** Email of customer account
  - **password:** Password of customer account
  - **preferences:** Dietary restrictions saved onto account (e.g., Nut-Free)
  - **pastOrders:** String array of IDs of past orders made by customer
  - **favourites:** Menu items favourited by customer

- **MenuItem:**
  - **itemId:** ID of the menu item
  - **name:** Name of the menu item
  - **categories:** Dietary filters which the menu item belongs to (e.g., Vegan, Gluten-Free, etc)
  - **price:** Price of the menu item
  - **type:** Category which the menu item belongs to (eg Appetizer, Burgers, Fish)

- **Order:**
  - **orderId:** ID of the order
  - **items:** String array of IDs of menu items ordered
  - **totalDue:** Total due of the order
  - **totalPaid:** Total amount paid by customer
  - **table:** Table number at which the customer sat

The following Services will be created:

- **CRUD Service:** This is the service used to create, read, update and delete documents in the database. It is called whenever access to the database is required.

- **Events Service:** This is the service that will be used to communicate between modules. It will allow modules to subscribe to and emit content.

The following modules will be created for the Customer interface:

- **browse-menu:** This module loads the menu from the database in live time and allows the customer to browse the menu, filter, and add items to their order. It must communicate with the view-order module to pass on items by using a Service to pass on the menu item ID. It has the following methods:

    - **constructor():** This initializes the services used by the module. The services include the **CRUD Service** and the **Events Service**.

    - **OnInit()**: This method is called upon initialization of the module when it is first loaded. It calls the **displayMenuItems()** function.

    - **displayMenuItems():** It will load all the menu items from the database and display them according to the "type" they are by calling the MenuItem Service.

    - **addToCart(String itemID):** This method is called when the Customer adds an item to the cart. It will communicate with the view-order module by using the Events Service.

    - **addFilter(String filter):** This method is called to when the Customer filters the menu items by the preferences (eg Vegan, Halal, etc). It will reload the menu items displayed by calling the MenuItem Service and query the database so only the items filtered will be displayed.

- **view-order:** This module communicates with the browse-menu module and takes the menu item IDs added to the cart. It then queries the database for the items with the matching IDs. It will display the items added to cart but not yet submitted on top, and the items that have been submitted (and thus, in progress).

  - **constructor():** This initializes the services used by the module. The services include the **CRUD Service** and the **Events Service**.

  - **OnInit():** This method is called upon initialization of the module. It will create a temporary Order Observable Object.

  - **createOrder():** This method adds one item to the Order object and uses the **CRUD service** to create a new **Order** object in the database. The Order ID is returned and is subsequently used for every **submit()** call to add the items to the order.

  - **submit():** This method is called when the Customer submits the items in the cart. If a order has not yet been created, it will call the **createOrder()** function to create a new order in the database for the customer. It will use the **CRUD service** to update the order in the database.

- **account:** This module allows Customers to login to view previous activity and edit profiles and can communicate with the browse-menu module. It has the following functions:

  - **constructor():** This initializes the services used by the module. The services include the **CRUD Service** and the **Events Service**.

  - **OnInit():** Load the preferences set by the Customer into the **browse-menu filters()** function, so the Customer will have a pre-saved **browse-menu** experience.

  - **login():** Allow user to login with SSL encryption.

  - **createAccount()**: Allow Customer to create an account.

  - **editProfile():** This will allow the user to update their contact information.

  - **viewPastOrders():** This will load a list of past orders saved on the Customer's account by calling the **CRUD Service**.

The Staff interface will have the following modules:

- **admin-login:** This module allows Staff to login and access Staff mode. It has the following methods:

  - **login():** Allow user to login with SSL encryption to access Staff Mode. Once logged in, it will automatically redirect to the module admin-nav.

- **admin-nav:** This module serves as a navigation page for the staff to redirect the user to different modules to access different functionalities. It will have the following methods:

  - **logout():** This terminates the user's session once the back button is pressed when on the admin navigation page.

- **view-current:** This module allows Staff to view current orders at each table and send out the menu items once they have been completed. It utilizes the **CRUD Service**. It has the following methods:

  - **constructor():** This initializes the services used by the module. The services include the **CRUD Service.**

  - **viewOrder(String orderID)**: This allows the user to view the current order at the specified table by using the CRUD Service to retrieve and display the order.

  - **sendItem(String itemID, String orderID):** This allows the user to send out the order to the customer.

- **edit-menu:** This module allows Staff to modify and add menu items. It utilizes the CRUD Service. It has the following methods:
    - **constructor():** This initializes the services used by the module. The services include the **CRUD Service.**
    - **addItem():** This method uses the CRUD Service to update the database with the newly created menu item.
    - **editItem(String itemID):** This method uses the CRUD Service to update the database with the updated menu item.
- **edit-layout:** This module allows Staff to modify the restaurant layout. It utilizes the CRUD Service. It contains the following methods:
    - **constructor():** This initializes the services used by the module. The services include the **CRUD Service.**
    - **moveObjectX(String itemID):** Move the selected object across the x-axis.
    - **moveObjectY(String itemID):** Move the selected object across the y-axis.
    - **changeXDim():** Change the width of the dining space.
    - **changeYDim():** Change the length of the dining space.
- **view-analytics:** This module allows Staff to view analytics about the restaurant and orders. It will have the following method:
    - **viewData(String data):** This method takes the data that the user wishes to see (eg most popular menu items) and queries Firebase Analytics to display the data.

The UI design of the Customer interface and the Staff interface were designed with user needs in mind and the principles of UI design. They differ in that the Staff interface will have bigger and more accessible buttons as we can safely assume that Staff will need more “direct” access and do not have as much time as a Customer. The Customer on the other hand, can spend much more time browsing and exploring the app. We plan on using familiar widgets such as buttons and modals to increase user adaptability and usability.

The UI design of the Customer interface is shown in the following Figures below:

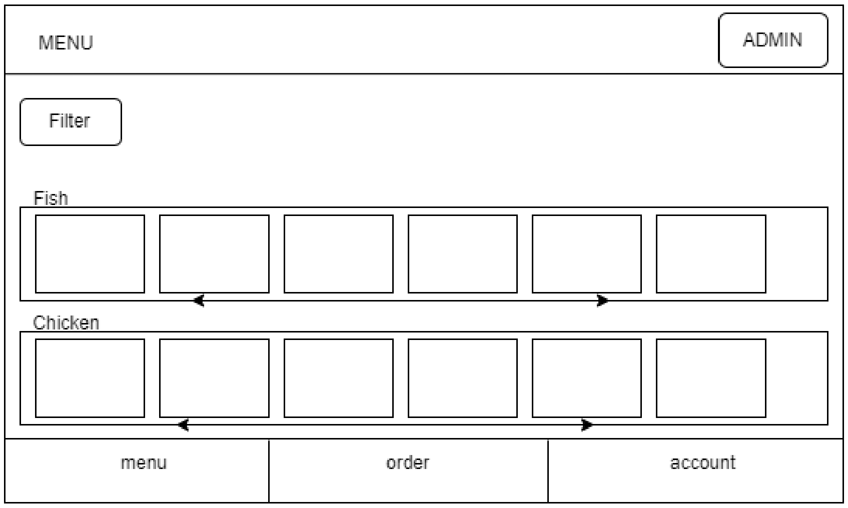


Figure 5: UI Design for Browsing Menu

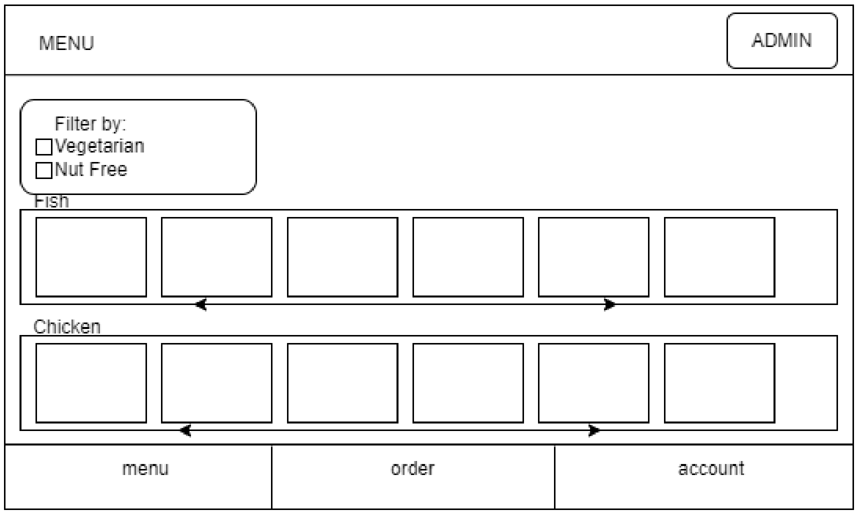


Figure 6: UI Design for Browsing Menu and Filtering Products

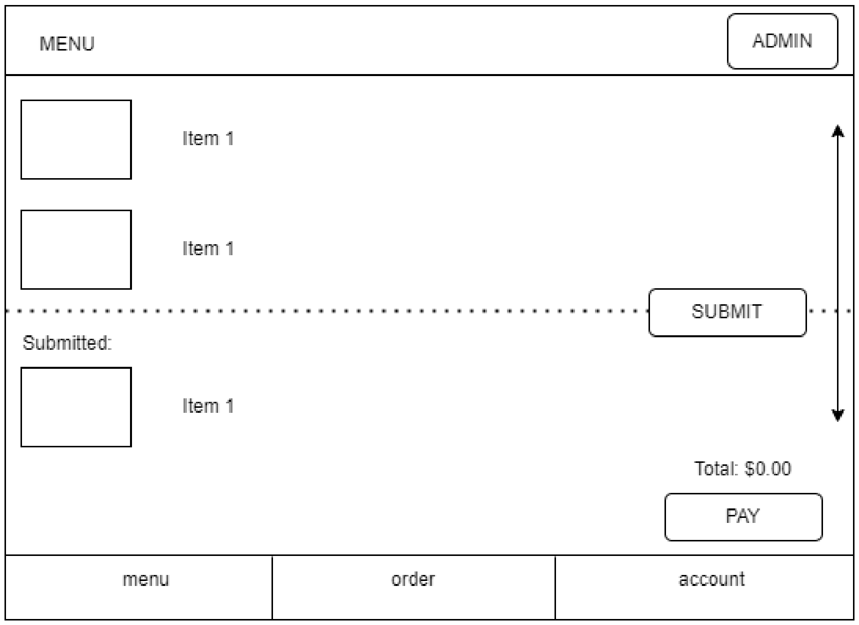


Figure 7: UI Design for Viewing Current Order and Submitting Items in Order

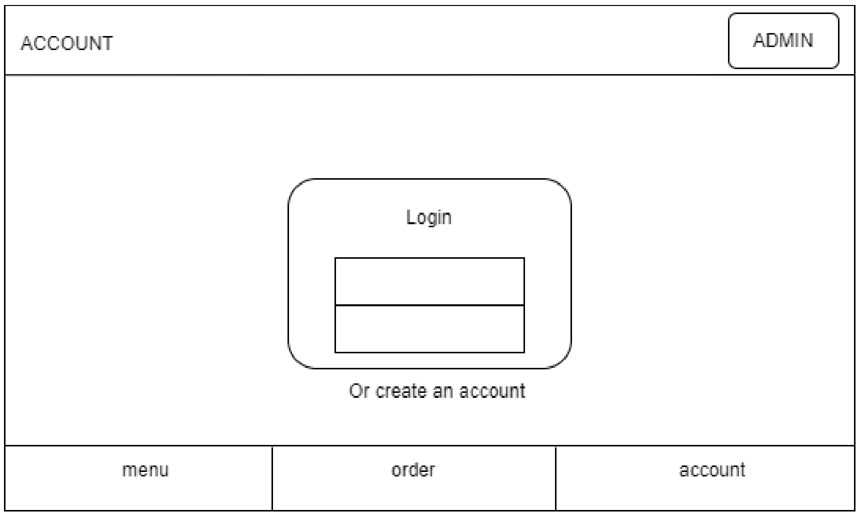


Figure 8: UI Design for Logging into User Account

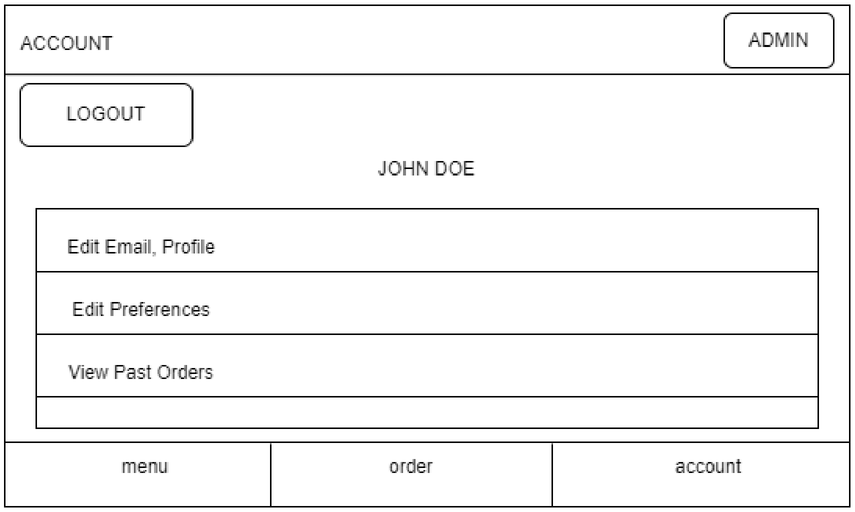


Figure 9: UI Design to View User Account

Note that staff are able to access staff mode by pressing on the “ADMIN” button at the upper left corner of the Customer Interface.

The UI design of the Staff interface is shown below:

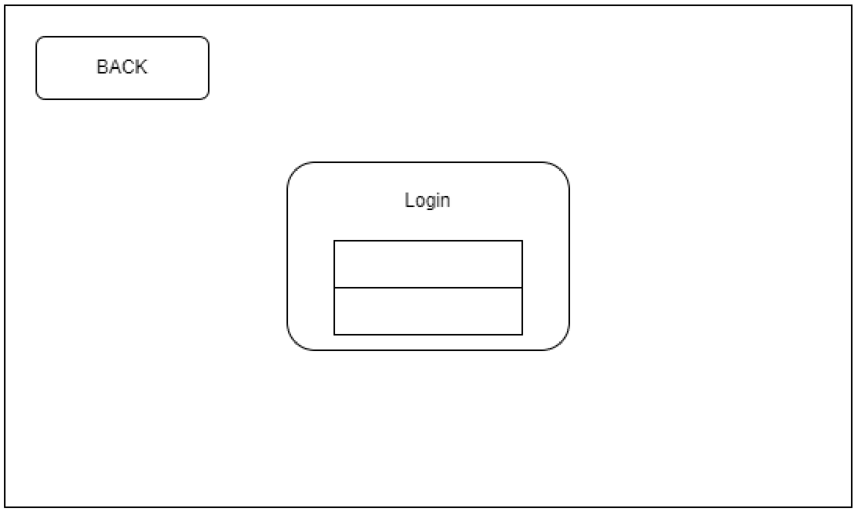


Figure 10: UI Design for Logging into Admin Mode

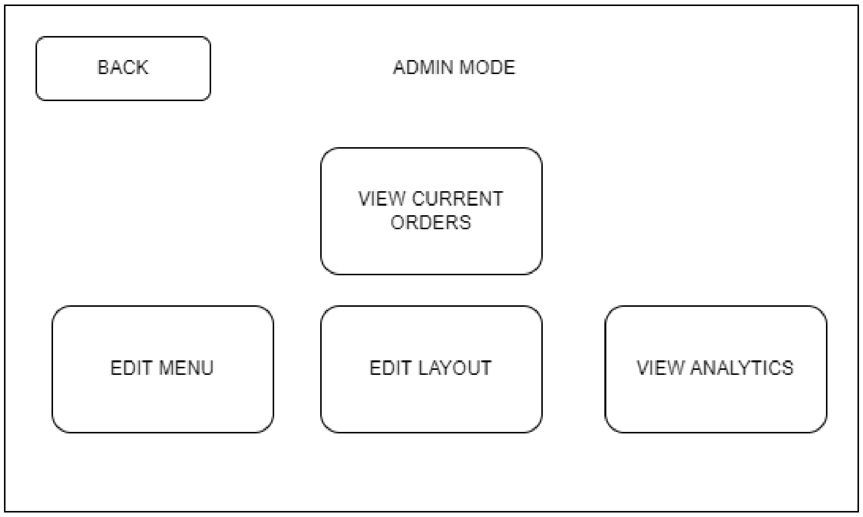


Figure 11: UI Design for Admin Navigation

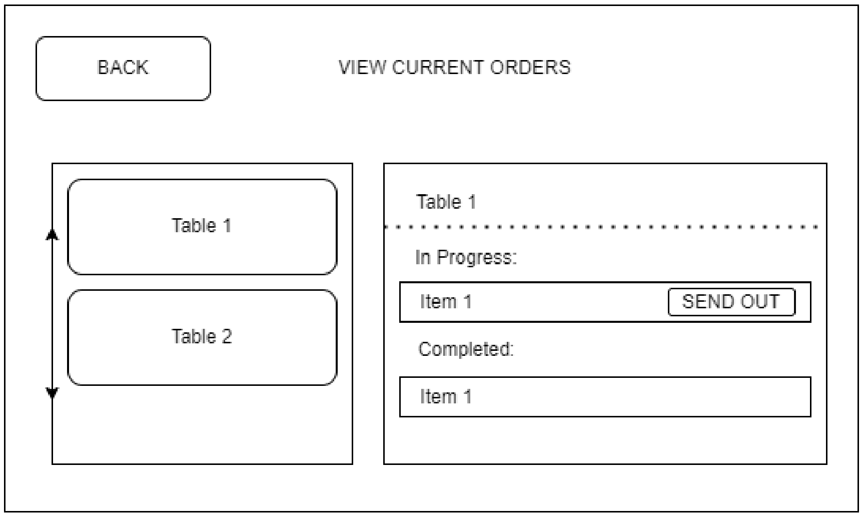


Figure 12: UI Design for Admin to View Current Orders and Send Items Out

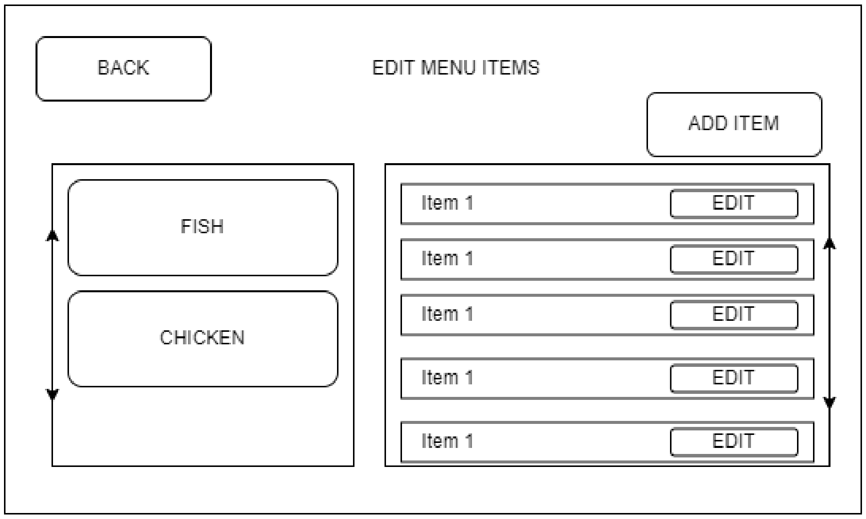


Figure 13: UI Design for Admin to Modify Menu

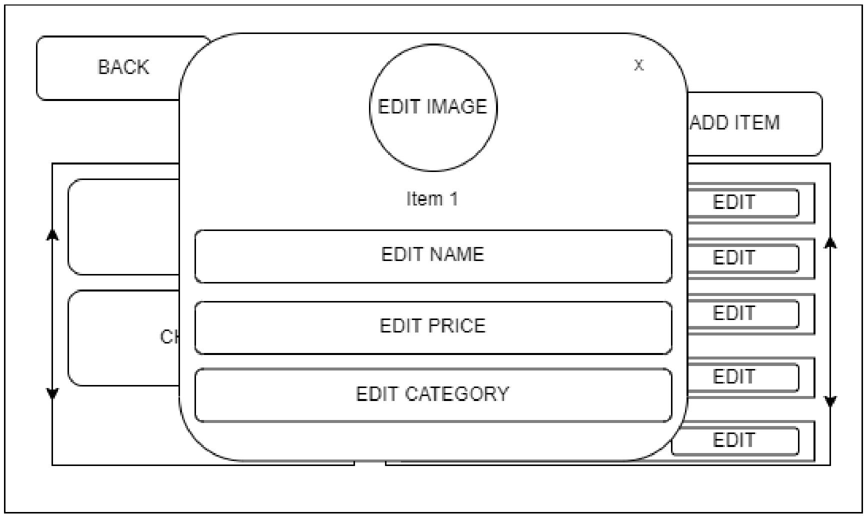


Figure 14: UI Design for Admin to Modify/Add Menu Item

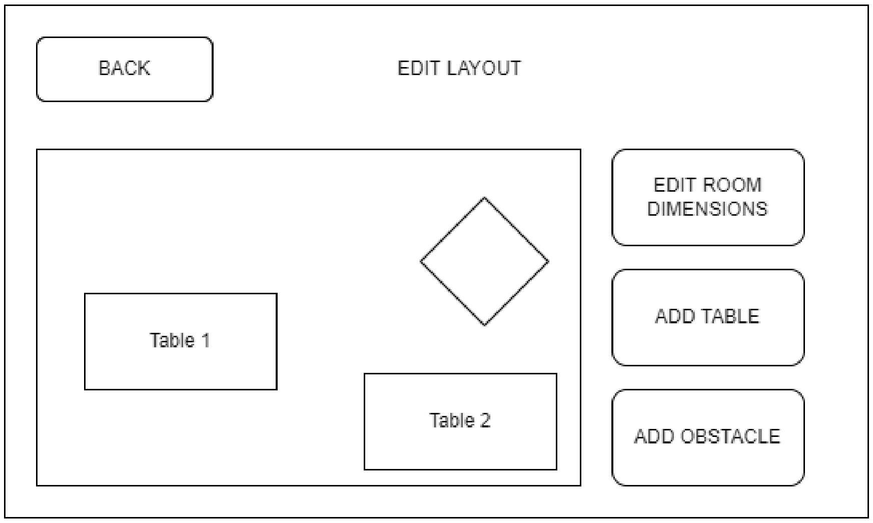


Figure 15: UI Design for Admin to Modify Restaurant Layout

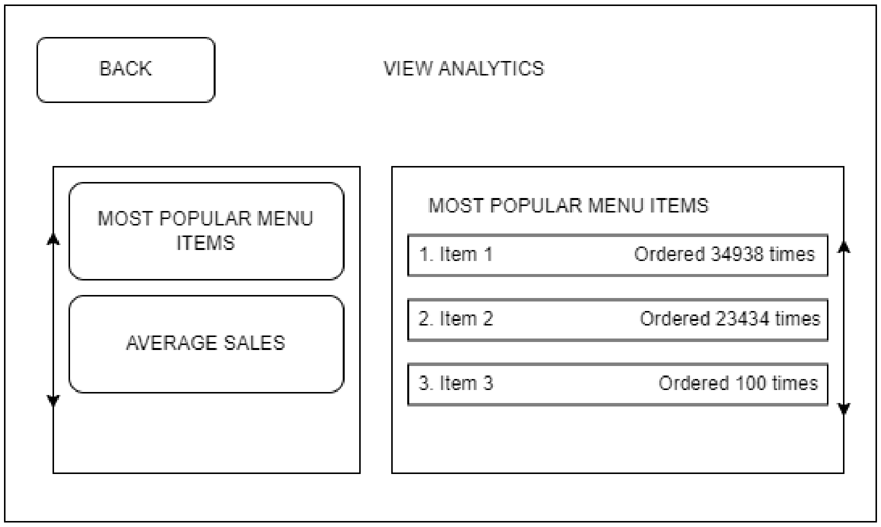
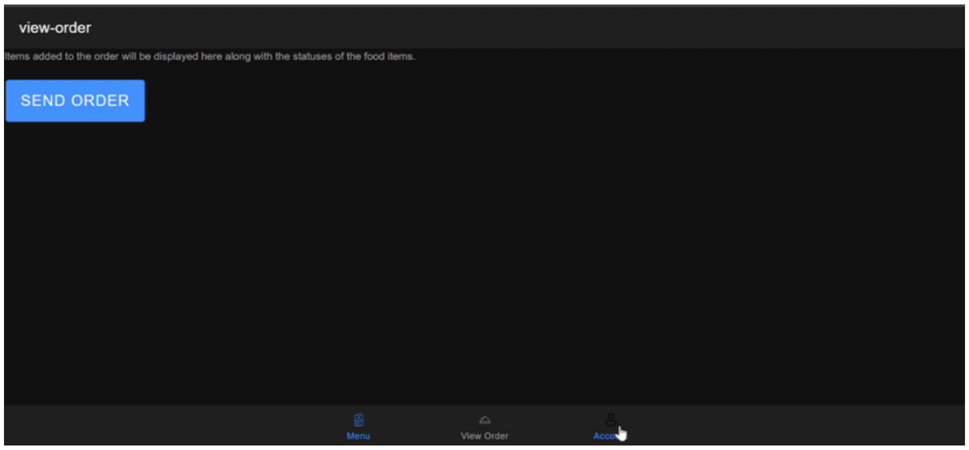
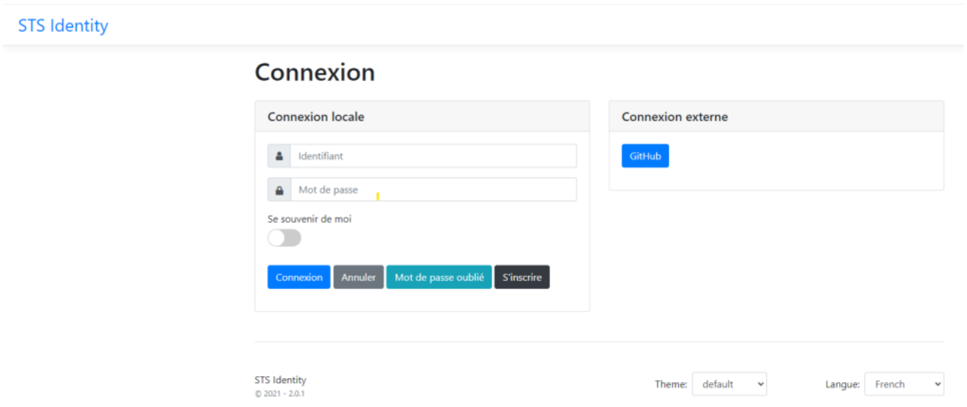

BACK
VIEW ANALYTICS
MOST POPULAR MENU ITEMS
AVERAGE SALES
MOST POPULAR MENU ITEMS
1. Item 1 Ordered 34938 times
2. Item 2 Ordered 23434 times
3. Item 3 Ordered 100 times

Figure 16: UI Design for Admin to View Analytics and Example Viewing Most Popular Items


view-order

Items added to the order will be displayed here along with the statuses of the food items.

Figure 17: Current Implementation of View Order (UI to be updated)



STS Identity
© 2021 - 2.0.1

Theme: default

Langue: French

Figure 18: Current Implementation of the Login Interface

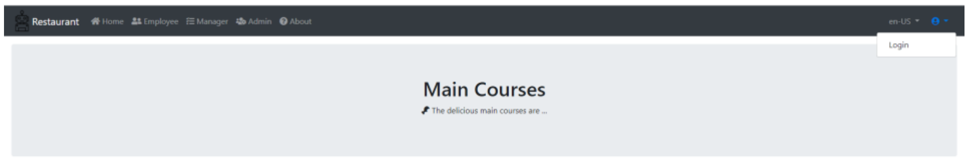


Figure 19: Current Implementation of new UI (Template to be updated to match design requirements)

3.1.3 Back-End

Since the application will be a web application, it will communicate with a Google Cloud Firestore database using a backend API. The database will use collections to store all the different data types (one data type per collection). The following describes the collections that will be stored:

- **Account:**
    - **StaffAccount:** Where the information about the restaurant is saved.
        - **adminId:** ID of the administrator
        - **email:** Email of the administrator account
        - **password:** Password of the administrator account
        - **layout:** Layout of the restaurant

    - **CustomerAccount:** Where the information about customer accounts is saved. Used purely to increase customer experience by keeping track preferences, contact information and past orders, as well for analytics on administrator's side.
        - **customerId:** ID of customer
        - **name:** Name of customer
        - **email:** Email of customer account
        - **password:** Password of customer account

        - **preferences:** Dietary restrictions saved onto account (e.g., Nut-Free)
        - **pastOrders:** Past orders made by customer
        - **favourites:** Menu items favourited by customer

- **MenuItem:** Where the information of all the menu items available in the front end is stored.
    - **itemId:** ID of the menu item
    - **name:** Name of the menu item
    - **categories:** Dietary filters which the menu item belongs to (e.g., Vegan, Gluten-Free, etc)
    - **price:** Price of the menu item
    - **type:** Category which the menu item belongs to (eg Appetizer, Burgers, Fish)

- **Order:** All orders made will be saved into the backend after order completion for analytics, as well as for the customer to review on their customer accounts.
    - **orderId:** ID of the order
    - **items:** Menu items ordered
    - **totalDue:** Total due of the order
    - **totalPaid:** Total amount paid by customer
    - **table:** Table number at which the customer sat
    - **comments:** Customer comments

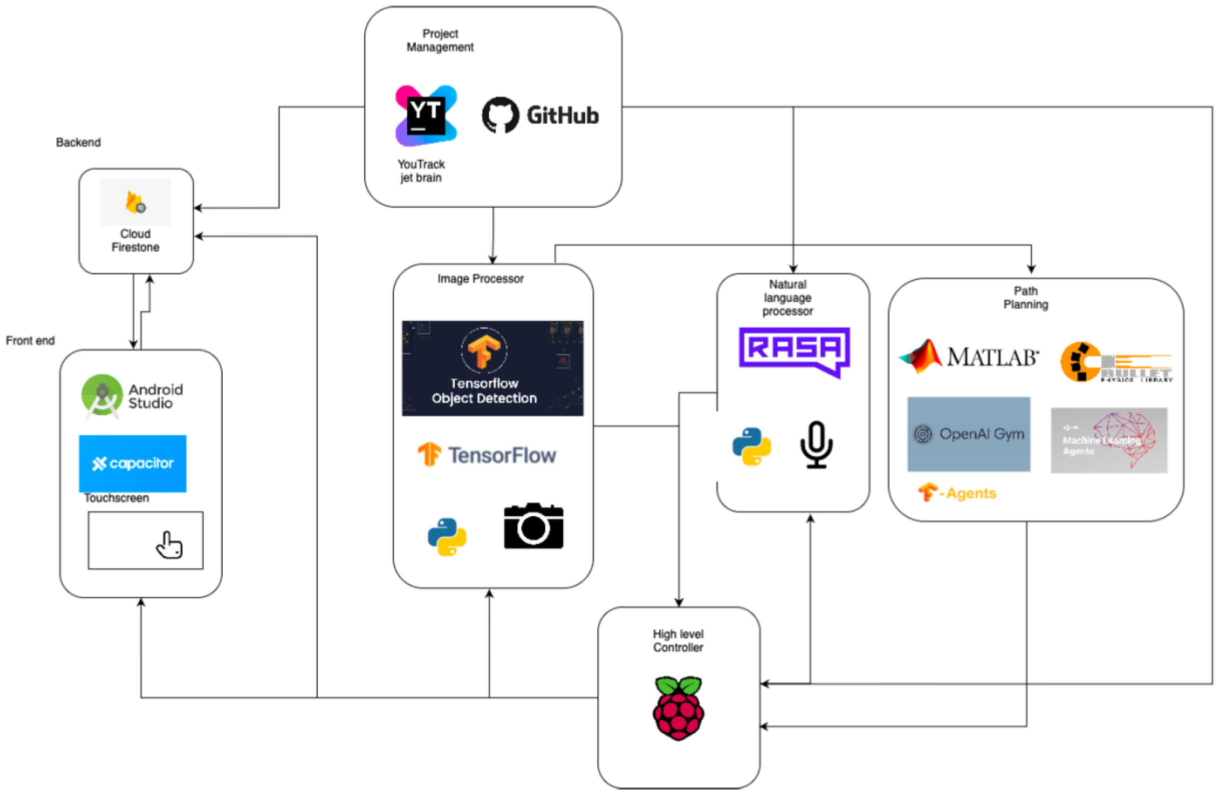


Figure 20: Software Architecture Conceptual Diagram

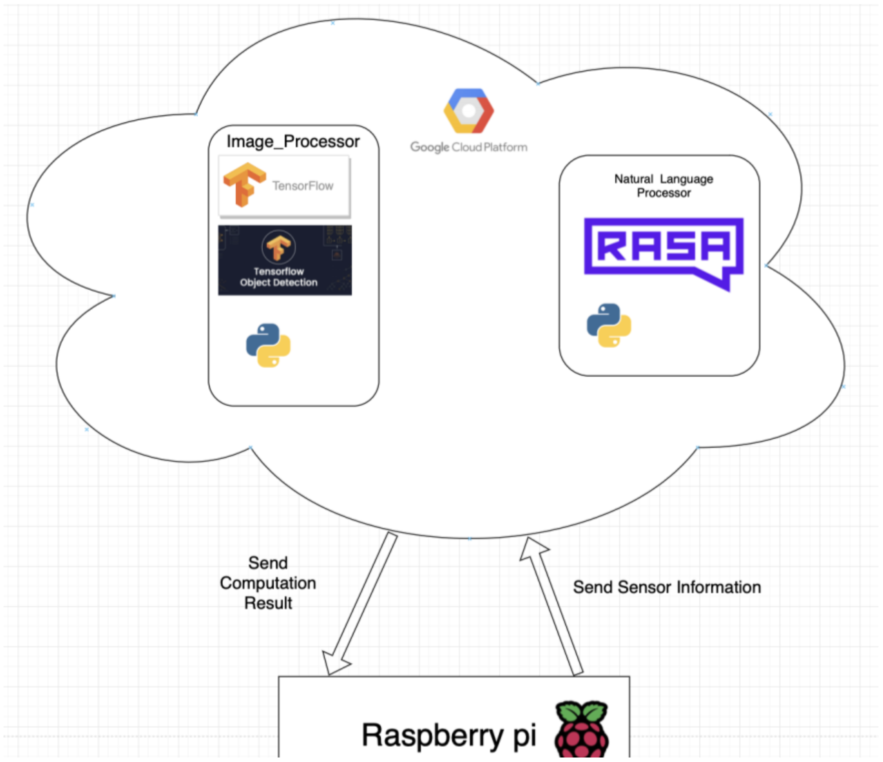


Figure 21: Cloud Deployment Diagram

3.1.5 Low-Level Controller

The low-level integrates into a single entity all the functions used to control or read from different components. Currently, the low-level controller is not entirely completed. However, you will find bellow a list of its current functions that can be called by the high-level controller:

get_battery_level(): This function calls the external function *voltage ()* which returns the instantaneous voltage reading of the Main Batteries. From there, the function uses a custom mathematical formula which we have developed to return the current battery level in the form of a percentage (%).

get_rpm(): This function calls the external function *get_rpm()* inside *motor_encoder.py* to get an instantaneous reading of the current RPM of the servo motors. The value for the reading is returned.

get_speed(): This function calls the external function *get_rpm()* inside *motor_encoder.py* to get an instantaneous reading of the current RPM of the servo motors. It then multiplies this reading by the circumference of the wheel in meters and divides by 60. The result from this mathematical operation is the current speed of the wheel in m/s. This value is returned.

get_sonar_distance(): This function calls the external function *ReadDistance(int)* inside *sonar.py*. The integer argument that must be given when calling this function is simply a minimum distance threshold from the nearest frontal obstacle. The external function that was called will then return to the calling function either *True* or *False* depending on if we have reached that minimum distance threshold or not. The calling function will return the boolean value.

stop (): This function stops the robot.

move_forward(): This function makes the robot move forward at maximum velocity.

move_backward(): This function makes the robot move backward at maximum velocity.

in_place_right(): This function makes the robot turn to the right in place (along the vertical axis of the robot). This is the fastest way to turn right however it will also cause higher centrifugal forces.

in_place_left(): This function makes the robot turn to the left in place (along the vertical axis of the robot). This is the fastest way to turn left however it will also cause higher centrifugal forces.

sweeping_right(): This function makes the robot turn right along the vertical axis of the left wheel (sweeping motion). This is the slowest way to turn right however it will also reduce the intensity of centrifugal forces. A secondary consequence of using this method to turn is that the robot will move forward by a distance equal to half the wheels trackwidth of the robot.

sweeping_left(): This function makes the robot turn left along the vertical axis of the right wheel (sweeping motion). This is the slowest way to turn left however it will also reduce the intensity of centrifugal forces. A secondary consequence of using this method to turn is that the robot will move forward by a distance equal to half the wheels trackwidth of the robot.

3.1.6 High-Level Controller

The high-level controller is where all the logic about the behaviour of the robot is. It receives input from the user application, the image processor, the natural language processor, the path planner and the low-level controller and it outputs an action to the robot through the low-level controller. The high-level controller will basically be a while loop constantly requesting input from all the components to choose the best course of action. The requests are made through getter methods implemented in each one of the modules, the decisions are made by if statements inside of the while loop and the actions are performed by setter methods implemented in the low-level controller. The implementation of threads will help to keep a live state of the robot up to date as well as the readings and alerts from all the modules. Especially, from the low-level controller if it encounters an obstacle while moving from point A to B. It is still early on the implementation of the high-level controller module but some of the functions that can be found inside this module are the following:

init_modules_connection(): open connection with the google cloud server to be able to run the path planner, the image recognition module and the natural language processor.

close_modules_connection(): close connection with the google cloud server to save bandwidth when the robot is charging or turned off.

3.2 Hardware Architecture

As a head start for the next semester, we decided to start building the hardware part of our robot in advance. This decision was motivated by the desire to test the software for our sensors, motors and camera in the real world and start optimizing our code. Furthermore, this allowed us to observe flaws in our software design in advance and change what was not working. Please note that this design is not final and may change as we continue testing next semester.

The hardware design for our project was done with two goals in mind. The first goal being that the design should be safe to avoid electrical fires or component damage. The second goal being to maximize the battery life of our robot.

To satisfy the need of safety, we have used a 5A fuse that is located between the main batteries and the rest of the circuit. In the event of a short circuit, the fuse will blow preventing an electrical fire and overheating damage to the circuit elements. Furthermore, we have installed one diode going from the charging source to the main batteries and one diode going from the main batteries to the main circuit to prevent current flowing in the wrong direction. This increases the reliability of our design and protects against user error by protecting our circuit from polarity inversion.

Secondly, since our robot will be used in the restauration business were depending on the time of day the demand can be very high for a prolonged period of time, we deemed that it was important for our robot to have a long battery life. Indeed, having a long battery life allows the robot to operate during those prolonged peak hours without needing to stop working to go recharge itself. To meet this design criteria, we have decided to use two separate battery modules. The first battery module labeled “Main Batteries” consists of two 6-cell batteries placed in parallel which is used to power the electric servo motors. The second battery module labeled “Secondary Battery” consists of a 10Ah/3.7V battery that is used to power the Raspberry Pi, Arduino, sensors and I/O devices. Not only is using a separate 3.7V battery providing us with more capacity but it is also avoiding the loss of power efficiency that would occur converting the 7.2V battery voltage to a 5V voltage required by our circuit.

The whole power system can be recharged using two Type A USBs which is more convenient than the specialized charger that was provided with the platform. Furthermore, in the future we plan on implementing wireless charging to avoid the need for human interaction in the charging process.

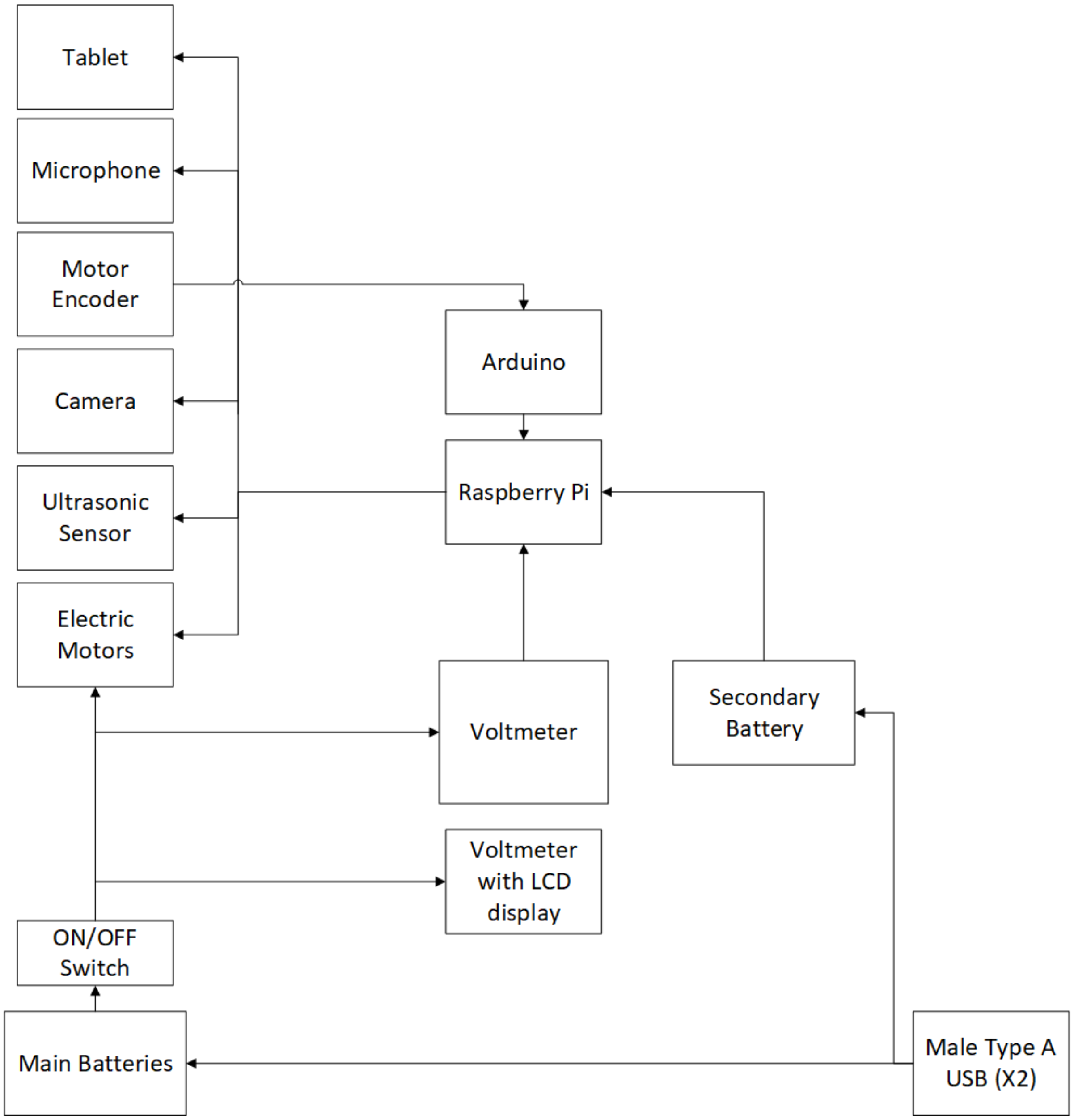


Figure 22: Hardware Architecture Conceptual Diagram

3.3 Frame Design

The frame design consists of the main platform containing the powertrain/electronics and a series of 3D printed parts attached to that platform.

In the figure bellow, the lower cylinder is the platform provided by the university. We decided to put the majority of our electronics in this section to lower the center of gravity and in this way improve both the stability of the platform and reduce the risk of tipping over.

The next rectangular shape located on top of that cylindrical platform is a 3D printed plate with 4 holes. These holes are used to bolt the remaining of the frame to the platform using metal bolts.

Above this plate we have attached a series of 3 hollow cylinders. The decision of making them hollow had to do with reduction of material cost, reduction of manufacturing time, reduction of weight, minimizing the height of the center of gravity, increasing the cooling of our electronics and allowing the tablet wires to pass through the structure. Furthermore, a cylindrical structure can support very well forces applied at different angles especially compressive forces (forces applied along the longitudinal axis of the cylinder). Please note that another valid design in terms of structural integrity would have been the use of triangles to form a truss. Although a truss would have been very solid structurally and would have reduced weight drastically over a cylindrical shape, it would have also increased the design complexity and would have been less pleasing aesthetically.

Finally, we have our 3D printed food plate that is attached to the top of the cylinders.

The total height of structure is roughly 0.75m with a maximum radius of 0.27cm.

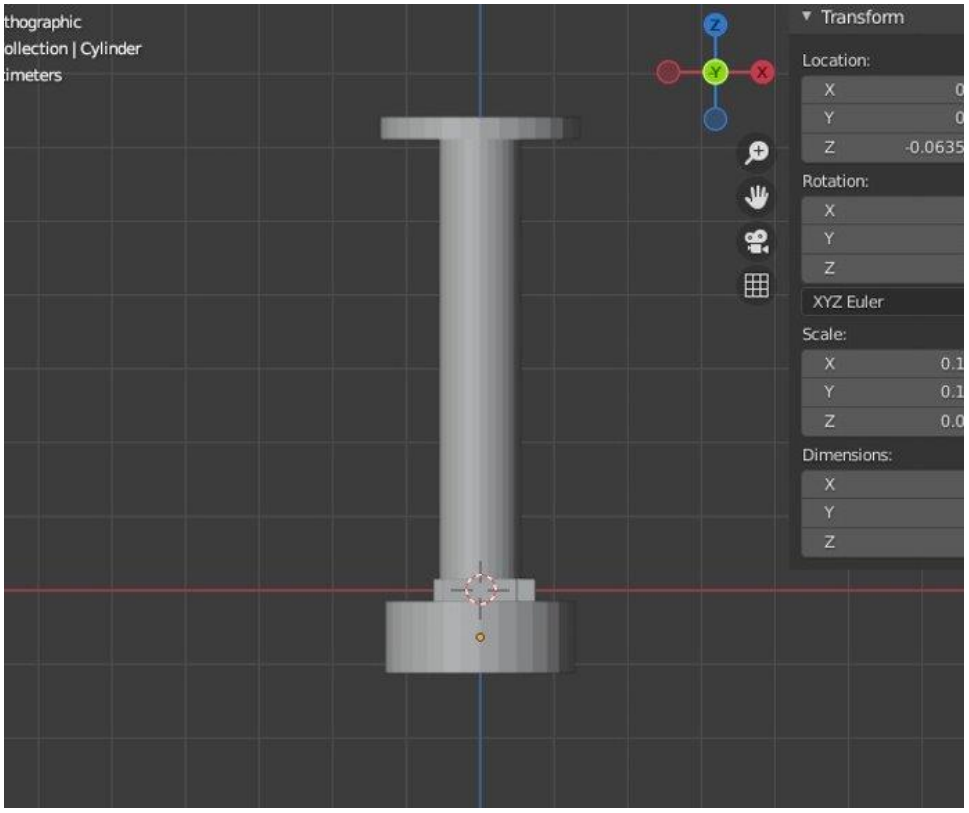
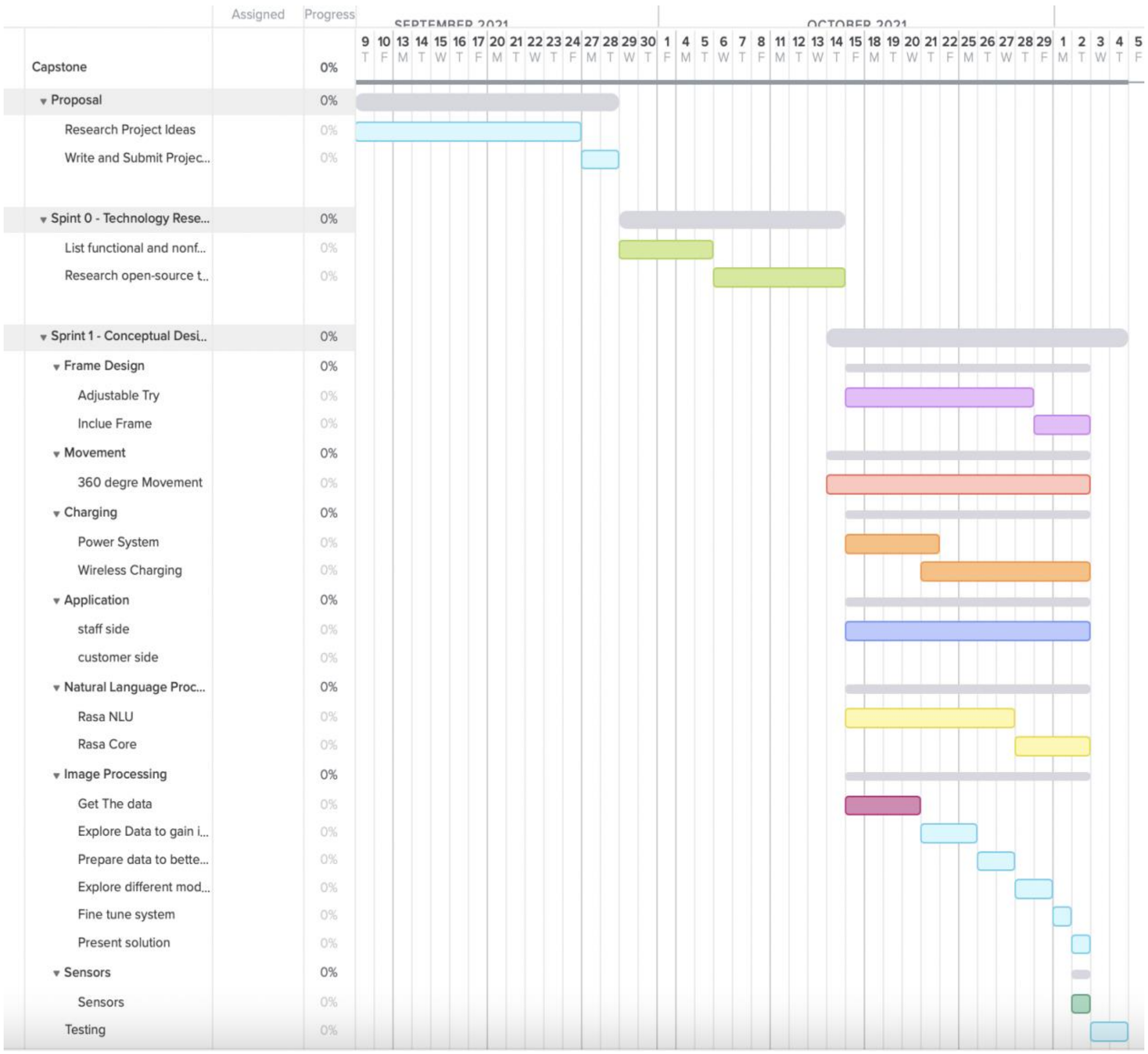


Figure 23: Frame Conceptual Design

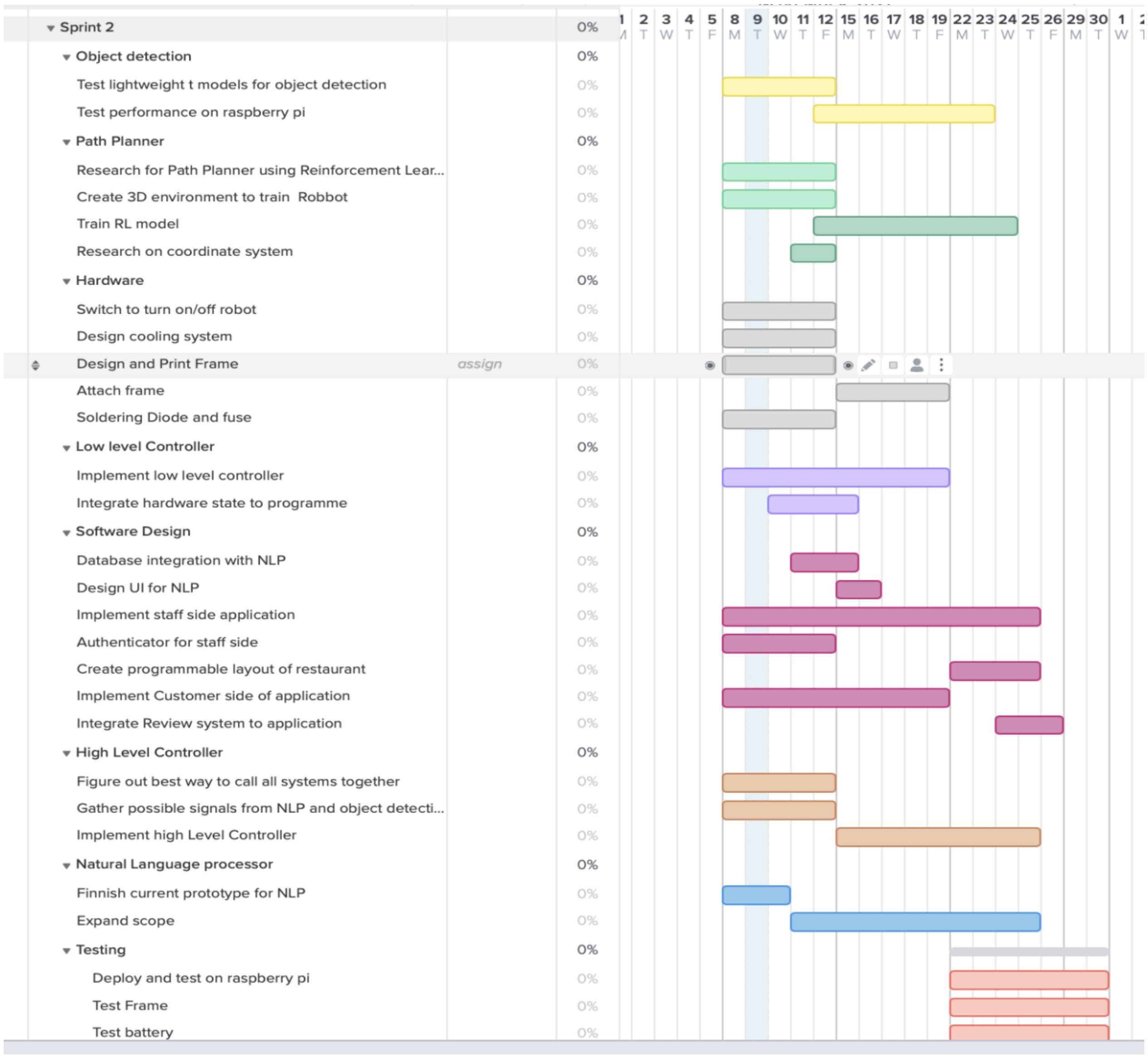
4.0 Project Management

4.1 Gantt Chart



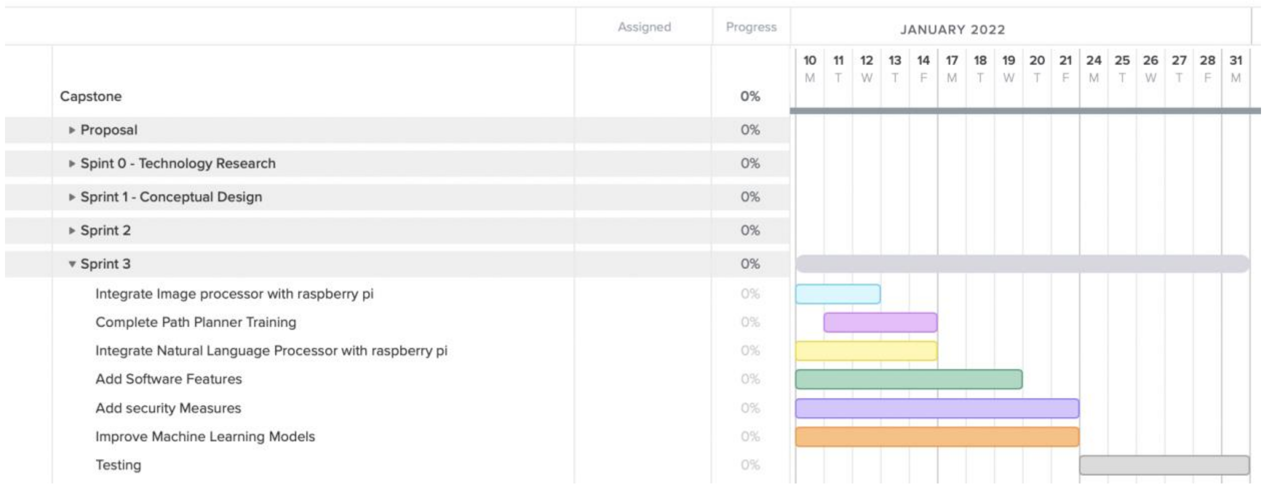
	Assigned	Progress
Capstone		0%
▾ Proposal		0%
Research Project Ideas		0%
Write and Submit Projec...		0%
▾ Spint 0 - Technology Rese...		0%
List functional and nonf...		0%
Research open-source t...		0%
▾ Sprint 1 - Conceptual Desi...		0%
▾ Frame Design		0%
Adjustable Try		0%
Inclue Frame		0%
▾ Movement		0%
360 degre Movement		0%
▾ Charging		0%
Power System		0%
Wireless Charging		0%
▾ Application		0%
staff side		0%
customer side		0%
▾ Natural Language Proc...		0%
Rasa NLU		0%
Rasa Core		0%
▾ Image Processing		0%
Get The data		0%
Explore Data to gain i...		0%
Prepare data to bette...		0%
Explore different mod...		0%
Fine tune system		0%
Present solution		0%
▾ Sensors		0%
Sensors		0%
Testing		0%

Figure 24: Gantt Chart Fall 2021 (Phase 1 - September 2021 to October 2021)



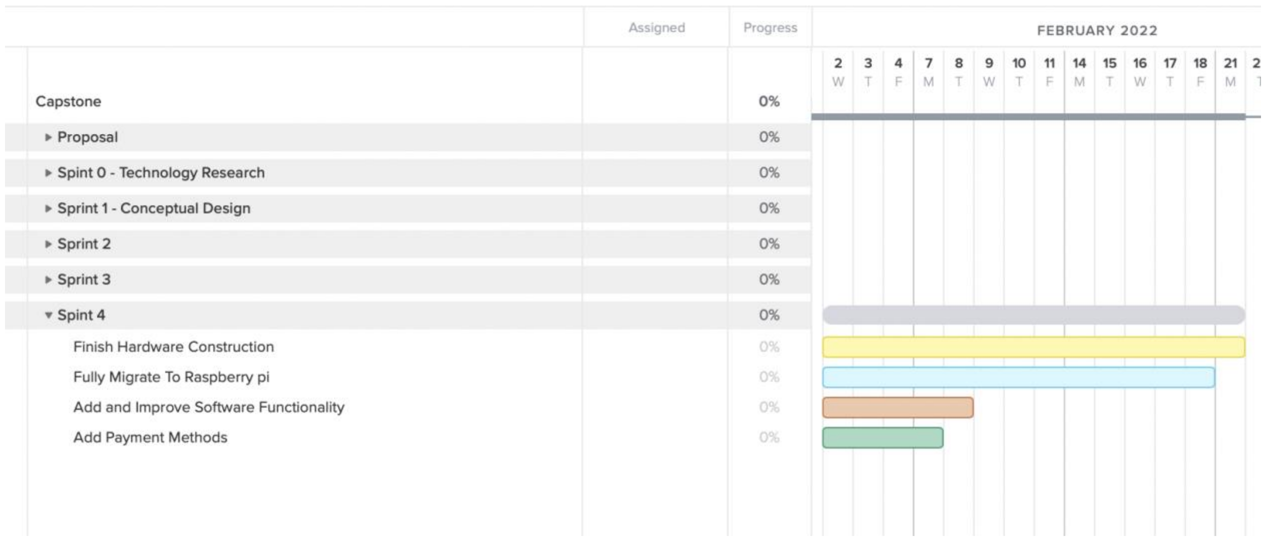
▾ Sprint 2		0%
▾ Object detection		0%
Test lightweight t models for object detection		0%
Test performance on raspberry pi		0%
▾ Path Planner		0%
Research for Path Planner using Reinforcement Lear...		0%
Create 3D environment to train Robbot		0%
Train RL model		0%
Research on coordinate system		0%
▾ Hardware		0%
Switch to turn on/off robot		0%
Design cooling system		0%
Design and Print Frame	assign	0%
Attach frame		0%
Soldering Diode and fuse		0%
▾ Low level Controller		0%
Implement low level controller		0%
Integrate hardware state to programme		0%
▾ Software Design		0%
Database integration with NLP		0%
Design UI for NLP		0%
Implement staff side application		0%
Authenticator for staff side		0%
Create programmable layout of restaurant		0%
Implement Customer side of application		0%
Integrate Review system to application		0%
▾ High Level Controller		0%
Figure out best way to call all systems together		0%
Gather possible signals from NLP and object detecti...		0%
Implement high Level Controller		0%
▾ Natural Language processor		0%
Finnish current prototype for NLP		0%
Expand scope		0%
▾ Testing		0%
Deploy and test on raspberry pi		0%
Test Frame		0%
Test battery		0%

Figure 25: Gantt Chart Fall 2021 (Phase 2 - November 2021 to December 2021)



	Progress
Capstone	0%
▸ Proposal	0%
▸ Sprint 0 - Technology Research	0%
▸ Sprint 1 - Conceptual Design	0%
▸ Sprint 2	0%
▾ Sprint 3	0%
Integrate Image processor with raspberry pi	0%
Complete Path Planner Training	0%
Integrate Natural Language Processor with raspberry pi	0%
Add Software Features	0%
Add security Measures	0%
Improve Machine Learning Models	0%
Testing	0%

Figure 26: Gantt Chart Winter 2022(Phase 3 - January 2021 to February 2021)



	Progress
Capstone	0%
▸ Proposal	0%
▸ Spint 0 - Technology Research	0%
▸ Sprint 1 - Conceptual Design	0%
▸ Sprint 2	0%
▸ Sprint 3	0%
▾ Spint 4	0%
Finish Hardware Construction	0%
Fully Migrate To Raspberry pi	0%
Add and Improve Software Functionality	0%
Add Payment Methods	0%

Figure 27: Gantt Chart Winter 2022 (Phase 4 - February 2021 to March 2021)

4.4 Budget

4.4.1 Software

No purchases were made or will be made because all software is implemented by the group by either doing it from scratch or simply using open-source code, like templates, as the base for our implementation.

4.4.2 Hardware

The tables below present the components we have used for this project. The first table represents all the items we bought which were all towards the hardware design. We decided we wanted to have a piece of hardware to test our code as we developed it, so we built a robot for a proof of concept. We believe by doing that, we improve the testing on a real-life environment because we're using the hardware capabilities of a Rasberry Pi which are not much compared to high end computers like most students have for university. The second table below represents the components that were provided to us by the university. Note that we were able to build a full functional robot and therefore the budget will most likely not change for the next semester.

By adding both totals of both tables we see that a full functional solution can be provided for only 568.58 CAD which is more than 5 times cheaper than other food delivery robots with similar capabilities.

Item	**Price (CAD)**
Welding Kit	10.32
Step Up Inverter	14.09
Voltage Meter	16.99

Fuse Holder + 5A fuse	13.99
1N5400 Diode (3A - 50V)	8.99
Cooling fan	17.99
Blank PCB Board	15.95
Blank PCB Board	17.99
Analogue Voltmeter	15.99
ON/OFF switch	7.99
Jumper wires	15.99
Fuse assortment	13.59
Male stacking headers	12.99
MCP3008-8-Channel 10-Bit ADC	12.78
Microphone	32.00
Arduino	45.00
TOTAL	269.54

Table 1. Items bought for project

Item	**Price (CAD)**
Rasberry Pi 4 4GB	55.00
Servo motors x2	76.10
Camera	32.99
Ultrasonic Sensor	34.95
Tablet	100.00
TOTAL	299.04

Table 2. Items provided by the university for project

Total Cost = Bought + Provided = 269.54 + 299.04 = 568.58 CAD

4.5 Risk Management

As the development of this project continues, we may fall into certain risks. The table below shows the risks that can occur, a brief description of them, and possible solutions for these risks.

Risk Item	Description	Resolution Approach	Who	Date
Machine Learning Models.	These risk items can arise: Incompatible data, insufficient data, Overfitting during training, distribution of test set. Hardware capabilities	Use of existing data sets, real time tests, use of various performance measure, error analysis and migration to cloud for optimal performance.	Amen A, and Harouna S.	During training and deployment.
Hardware	Electrical Fire or Component Damage due to short-circuits or polarity inversion	Use of a 5A fuse to protect against short-circuits. Use of two diodes to protect against polarity inversion.	Marc-Frédérick Tremblay	During hardware design and conception.

Software	Security and Privacy Breaches Slow performances due to connectivity	Use HTTPS when making calls to the database, writing secure rules for application access to the database Limit the number of calls made to the database	Santiago C, Michelle T.	During implementation.

Table 3. Risk Table

5.0 Testing

5.1 Test plan

Machine Learning models

Once training is done, the model is first tested using a test set that we split from the dataset. If the performance of the model exceeds a certain threshold, we deploy the models on the cloud where the raspberry pi can access its predictions or directly on the raspberry pi as is the case for the path planner. We then perform real time test with probable use cases. If at any point the models are not performing adequately, they are tweaked and retrained.

Hardware

During the hardware design and conception, we estimated the performance of our components. As we continue the testing of our software using the hardware, we might notice that the performance is not sufficient. In this case, we might have to add battery capacity or replace some components with a more performant alternative as required.

Frame

The frame we have designed is anticipated to exceed our minimum stability, rigidity and functional requirements. Of course, as we continue our testing, we might notice that there are design flaws which prevents our robot from achieving certain requirements in a consistent manner. In this case, we will have to revise the design to improve upon those flaws.

Application

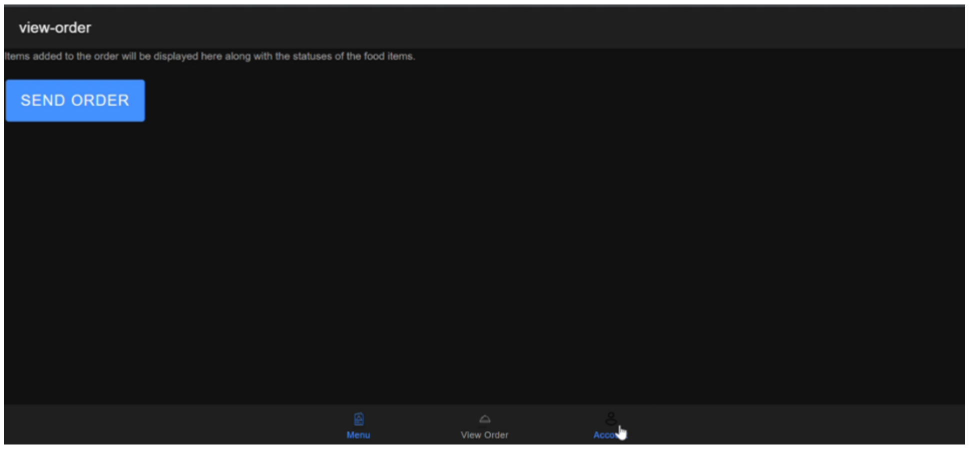
The application is first designed as web app using TypeScript, HTML, JavaScript and SCSS so it can be easily tested using the web browser. We can easily test functionality by using the console of the browser to print out results. We use later a wrapper to make it an Android application and therefore being able to deploy it on the tablet. We can safely say the behaviour seen using the web browser will be very similar to the one seen on the tablet. However, we count on deploying the application to the tablet during phase 4 so we can fix any UI issues we may encounter.

5.2 Test Cases and Results

Software

The functionality of browse-menu module, the view-order module as well as the Staff mode login were tested by running the application. We were successful at implementing the staff mode login and authentication, and somewhat successful at implementing the browse-menu module. The details of the tests are listed below in sections 6.2 and 6.3.

6.0 Proof of Concept



Items added to the order will be displayed here along with the statuses of the food items.

Figure 28: Current Implementation of View Order (UI to be updated)

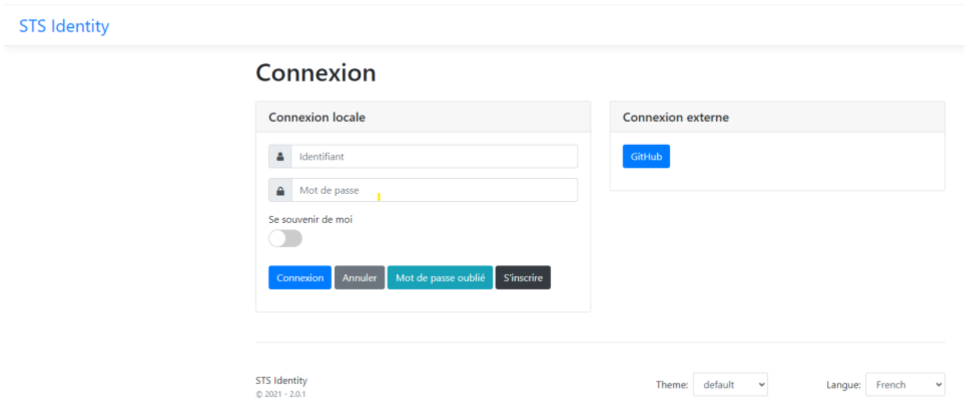


Figure 29: Current Implementation of the Login Interface

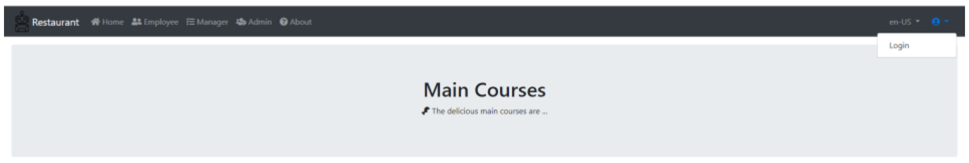


Figure 30: Current Implementation of new UI (Template to be updated to match design requirements)

6.1 Assumptions Made

Software

It is assumed that the consumer will have consistent Internet connection (WiFi to be specific) when using the application. We also assume that the user will have some prior experience in using mobile applications.

6.2 Results Obtained

6.2.1 Success Case

Object Detection

Once video data is fed into the model, it outputs, bounding boxes of the object detected with a number on top stating how certain it is. A successful object detection looks like the following:

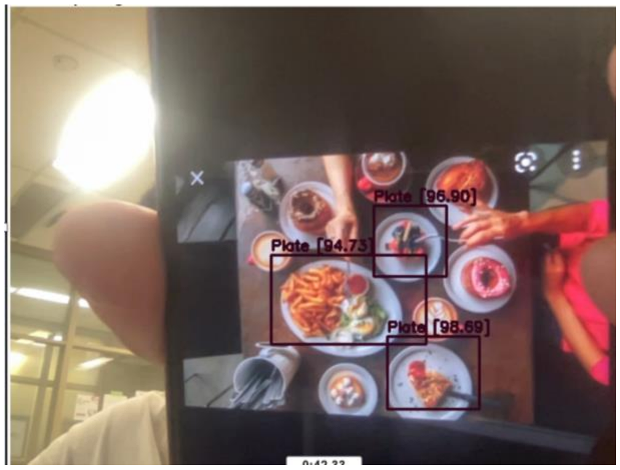


Figure 31: Result of successful live object detection

Path Planner

A success case for the path planner, would be the agent choosing the most efficient path towards its task, and having the least amount of collision. To arrive to a model that is performant, several weeks of training will be required. An unsuccessful object detection looks like the following:

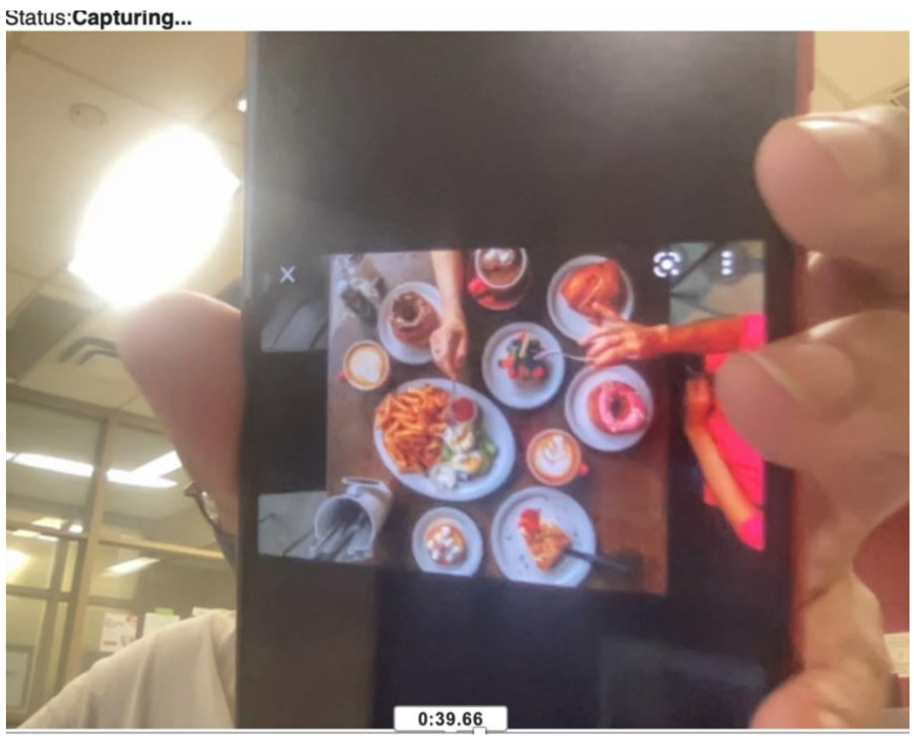


Figure 32: Image of an error case on the object detection model

Collision Avoidance

A success case for the collision avoidance, would be the successful avoidance of an obstacle while remaining on course. In order to have a performant collision avoidance algorithm, we would have to use the sonar sensor in pair with stereoscopic vision to successfully detect obstacles before a collision occurs. Furthermore, to stay on course during the collision avoidance, we would have to use the motor encoders and work in conjunction with the path planner algorithm to make sure that we do not deviate from our path. Finally, since motor encoders are subject to odometry errors, that is a deviance between the measured travelled distance and the true travelled distance; we will have to periodically re-calibrate the motor encoders measurements with those measured with stereoscopic vision.

Software Application:

A successful browse-menu module would correctly load all the menu items from the database and display them according to the type, as shown in the figure below.

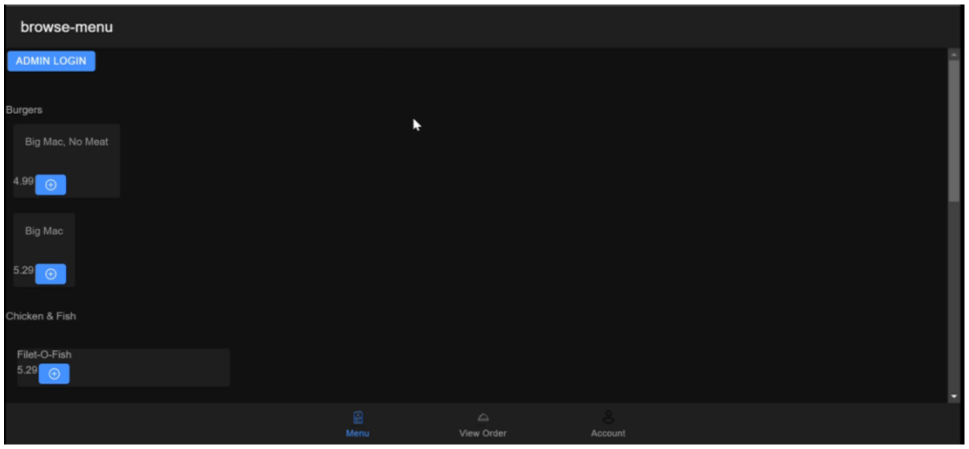


Figure 33: Current Implementation of Menu Browsing (UI to be updated)

A successful login interface exchanges the username and password in a secure way with the database and prompt for a fast authentication to the user. The following figure shows a successful implementation of it:

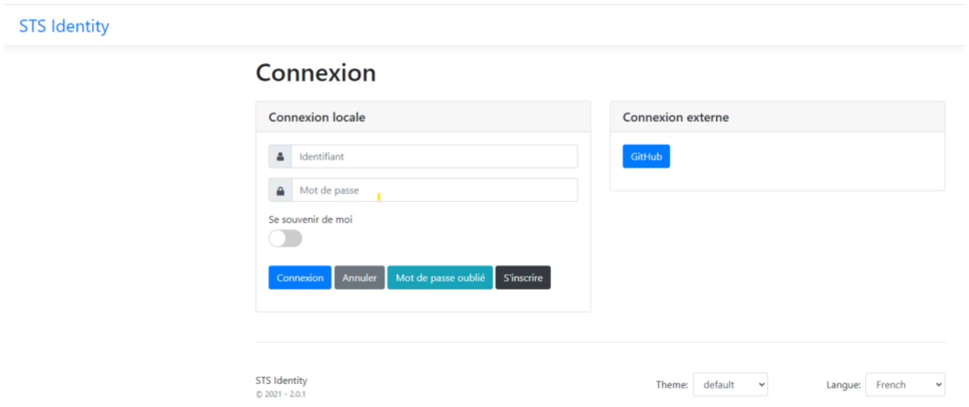


Figure 34: Current Implementation of the Login Interface

6.2.2 Error Case

Object Detection

False identification and lagging to make predictions are the most common error cases encountered. This can have serious ramifications, since the object detection is the primary source of data inputted to the high-level controller.

Path Planner

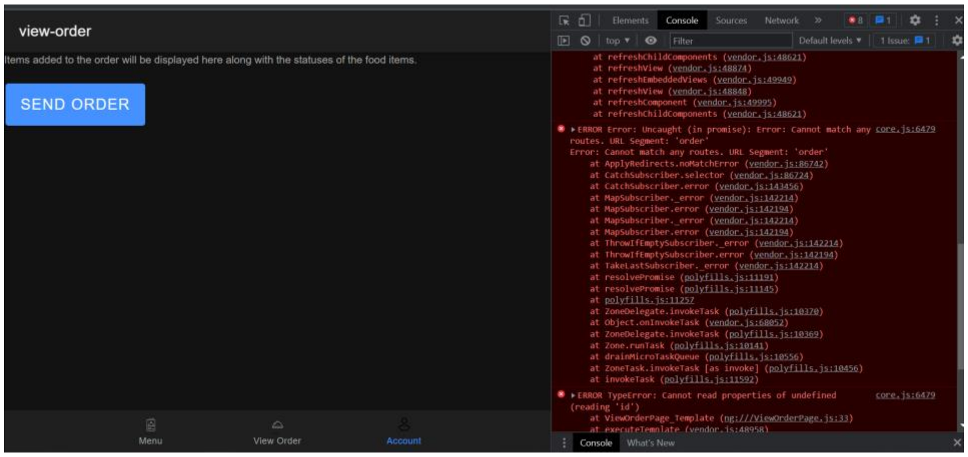
An error case for the path planner, would be the agent wandering off, an inefficient path chosen, too many collisions, generally choosing the sub-optimal action.

Collision Avoidance

An error case for the collision avoidance, would be a collision with an obstacle due to incorrect sonar readings combined with a failure or erroneous detection of matching points on an object during our passive triangulation computations. Another error case would be a deviation from our path that could be caused by odometry errors that was unsuccessfully corrected with stereoscopic vision; or due to an edge case that was not considered in our path planning algorithm.

Software Application:

Error cases for the software application include bugs in the software, as well as errors in functionality and communication (a specific example is described below), and usability for the average user. A successful implementation of the browse menu and view order functionalities would allow the user to add an item to their order. This would involve a correct implementation of the Events Service. Once an order is submitted, there should be a new order created in the database. Unfortunately, no successful implementation has been created, and the following is an image of an error case:



```
      at refreshChildComponents (vendor.js:48621)
      at refreshView (vendor.js:48874)
      at refreshEmbeddedViews (vendor.js:49949)
      at refreshView (vendor.js:48848)
      at refreshComponent (vendor.js:49995)
      at refreshChildComponents (vendor.js:48621)
▸ERROR Error: Uncaught (in promise): Error: Cannot match any core.js:6479
routes. URL Segment: 'order'
Error: Cannot match any routes. URL Segment: 'order'
     at ApplyRedirects.noMatchError (vendor.js:86742)
     at CatchSubscriber.selector (vendor.js:86724)
     at CatchSubscriber.error (vendor.js:143456)
     at MapSubscriber._error (vendor.js:142214)
     at MapSubscriber.error (vendor.js:142194)
     at MapSubscriber._error (vendor.js:142214)
     at MapSubscriber.error (vendor.js:142194)
     at ThrowIfEmptySubscriber._error (vendor.js:142214)
     at ThrowIfEmptySubscriber.error (vendor.js:142194)
     at TakeLastSubscriber._error (vendor.js:142214)
     at resolvePromise (polyfills.js:11191)
     at resolvePromise (polyfills.js:11145)
     at polyfills.js:11257
     at ZoneDelegate.invokeTask (polyfills.js:10370)
     at Object.onInvokeTask (vendor.js:68052)
     at ZoneDelegate.invokeTask (polyfills.js:10369)
     at Zone.runTask (polyfills.js:10141)
     at drainMicroTaskQueue (polyfills.js:10556)
     at ZoneTask.invokeTask [as invoke] (polyfills.js:10456)
     at invokeTask (polyfills.js:11592)
▸ERROR TypeError: Cannot read properties of undefined       core.js:6479
(reading 'id')
     at ViewOrderPage_Template (ng:///ViewOrderPage.js:33)
     at executeTemplate (vendor.js:48958)
```

Figure 35: Incorrect Implementation of View Order (UI to be updated)

6.3 Analysis and Justification

7.0 Contribution List

Amen Abshir

- Implementation of the path planner.
- Implementation of the object detection in the image processing unit
- Research on the implementation of the facial recognition feature that will help classify customers as existing or new and implement the security component of the robot.

Marc-Frédérick Tremblay

- Hardware design & conception.
- Coding for the sensors.
- Implementation of the low-level controller.
- Implementation of the hard-coded collision avoidance algorithm.
- Frame design & conception.
- Calculations of Center of Gravity (CoG), Static Stability Factor (SSF) and Rollover Risk to optimally distribute the weight in our robot.

Michelle Tang

- Software and Application UI Design

- Implementation of the Customer Interface and front end of the application to allow for menu browsing and order viewing
- Implementation of the database

Santiago Cely Bernal

- Implementation of the high-level controller.
- Implementation of the front-end of the application.
- Implementation of the login interface to allow customers and staff easily and securely access their account.
- Testing

Harouna Sylla

- Implementation of the rasa chatbot
- Implementation of the voicebot
- Testing with rasa-x

8.0 Post-Performance Analysis

Amen Abshir

While developing the object detection model, we encountered several, problems. First, the dataset has to satisfy several constraints. We first started by creating my own dataset, but this approach seems to be too hard for large datasets. My next approach was to find existing datasets. After some research, I landed on the COCO data set, as it was labeled and of very high quality. The next task was to choose the models architecture. After some, research the YOLOv4 model satisfied our project need the most, as it was very accurate and very fast.

Once training and testing was done, deployment on raspberry pi was very slow. So, we decided to migrate the model to the cloud. This gave us an optimal performance and allows us to easily update our model.

Development on the path planner started after the end of sprint1. The main challenge was developing an environment that can mirror the restaurant, creating a model for the robot, defining the states the bot will find itself in, the actions it can take, computing the best possible reward. Once the environment was defined and implemented using Python and the Pybullet environment, the training was straightforward thanks to existing libraries like tf-Agents.

Marc-Frédérick Tremblay

The collision avoidance algorithm was developed late in the semester. Due to time constraints, we could not use the path planner to avoid obstacles in an intelligent manner. Therefore, for this semester our collision avoidance algorithm is hard-coded. If a single obstacle is placed in front of the robot, the robot will avoid on the right and then go back to its original path. If an obstacle is placed in front of the robot and on the right side of the robot, the robot will avoid on the left and then go back to its original path. Finally, if an obstacle is placed in front, on the right and on the left of the robot simultaneously, the robot will stop as there is no reliable obstacle avoidance strategy that can be used without a path planning algorithm. Obviously, such an algorithm is low in performance and not reliable. Indeed, since the main obstacle detecting mechanism for this algorithm is a sonar, it is very easy to get a false reading or no reading at all if an obstacle has a weird shape or is positioned in an angle that prevents the sound waves from bouncing back to the sensor. Once the development and training of the path planner is complete, we will be able to avoid obstacles in a much more efficient and reliable manner.

Since the low-level controller is not fully implemented, a post-performance analysis cannot be done.

The coding for the different sensors and component has been developed relatively early during the semester. For the motor-encoder reading, the readings were compared with physical measurement of displacement of the robot. The readings were fairly accurate however after a long period of time it is expected that the readings from the encoder will slowly deviate from the true distance travelled due to odometry errors. The voltage reading function was compared with physical measurement of the battery voltage using a handheld voltmeter. The values measured by the function was equivalent to the readings on my handheld voltmeter. The functions for the movement of the robot were tested and are performing as expected. The function reading the sonar sensor data is working. However, due to the low-reliability nature of the sonar, the readings were sometimes incorrect.

The frame design was not tested yet since the robot is not fully functional yet. Therefore, a post-performance analysis is not possible.

The hardware design was tested in debth. I have recharged the batteries on multiple occasions. The batteries were not overheating during charging, and we could reach a full charge. I have also tested the maximum battery life when using both motors at maximum velocity. On average, I had a maximum battery life of 2 - 2.5 hours. The servo motors are working properly although the wheels are not properly centered and causes the robot to vibrate during operations. I tried aligning the wheels, but it seems to be a physical defect of the wheels. Stress tests have been performed and none of the circuit components overheat during usage. Polarity inversion protection was tested by disconnecting the circuit and using a multimeter to measure current flow. The diodes are working as planned and allows a negligible amount of current to flow in the opposite direction. The fuse was tested by disconnecting the circuit and carefully causing a short circuit in a controlled manner. The fuse blew as expected. Before replacing the bad fuse, I reconnected the circuit and tried charging the batteries and using different components on the robot. All the components were not working, and charging was not possible; proving that the circuit was well designed, and the fuse was indeed protecting all the components. Analogue to Digital circuit and the analogue voltage meter are both performing correctly.

Michelle Tang

While I had originally created a software design at the beginning of Sprint 1, I realized W developing the software application, that since the application is essentially a web app and we could not work with objects as we had known them. Database objects are read as Observables and cannot really be stored in objects since our application requires many API calls to the database. I had to redesign the architecture, so it was more suited towards web development and modularized the application into different modules for different functionalities. This also meant that I had to design the UI very early on in Sprint 1, so I had a better idea of which components belonging together. I also started populating the database with sample data. I realized that since the app involved many database operations, we would need to create a CRUD application.

In Sprint 2, I started web development and had little success outside of reading documents in the database (implementation of the CRUD service). I realized that in order to fulfill the “filtering preferences” functionality of browsing the menu, I would have to find some way of querying or “piping” the data from the database. My attempts were unsuccessful as I tried using RXJS operators, and I decided to switch to Angular piping. I also found out that in order to pass data between modules, I would need to implement an Events Service to inform subscribing modules of any changes and updates. Due to time constraints and researching implementations and solutions, I was unable to complete these two functionalities. In addition, I had started looking at the implementation of a storage bucket to add images and linking the references from the storage bucket to the database.

Harouna Sylla

At beginning of the sprint 0 we encountered lot of difficulties choosing an approach to follow for the implementation of the natural language processing. And at the start we wanted to design the nlp starting from the very low level like directly manipulating machine learning algorithms to create our own natural language understanding, natural language generator et the dialog management. All of these three modules would form the nlp we have now. But when we were doing research, we came across the rasa framework whose design is already based on the modules we wanted to implement and will allows to implement the nlp without having to deal directly with machine learning algorithms which would made our project more feasible so we opted for rasa.

But there were still some obstacles. Rasa is a fairly an instable framework that's always evolving so we had to learn how to use it and configure it with some difficulties. In addition, there was the fact that it only offers a text-based solution and that was a stopping point. So, we had to think how to implement an extra module which could interact with users in voice-based way. And then we found a way to connect this module to rasa chatbot via local port and this led us to our current prototype of the nlp.

Santiago Cely

During phase 1, I did all the research about the rasberryPi and did all the start up in it including the installation of the operating system, setting up a git repository and cloning it to the rasberry Pi, installing all the drivers necessary and generating test code to ensure the good functionality of the Ultrasonic sensor and the camera and finally I set up a VNC account for our group so anyone could access the rasberryPi remotely to test code. During this phase I encountered some problems related to the camera because the rasberryPi would not recognize it so after doing some debugging, I concluded that the camera provided was faulty in which case we asked for a different one. I also implemented some basic code to calculate the distance between the ultrasonic sensor and the object ahead. It consisted in measuring the time from where the wave was sent and where the last wave was received and then multiplying these values by the speed of sound to get a distance.

During phase 2, we switched our RasberryPi 3B+ by a RasberryPi 4B and got new servo motors. Then, I started the implementation of the high-level controller, but I ran into the problem that none of the other modules were ready yet. So, I helped with some debugging on the natural language module and then I started the implementation of the new UI as well as the login interface. The UI and the login interface took me about 3 weeks to complete because I had to do some research first on the web applications and then I ran into some compatibility problems between Ionic and Angular. All the issues were resolved by two weeks before the end of phase 2 and right now I'm working on adjusting the UI to the design requirements and developing the high-level controller logic as the image recognition module and the natural language module come to their final implementation.

9.0 References

1. Christiansen, B. L. (2021, March 5). *Clear Pros and Cons of an Automated Restaurant*. Hubworks. Retrieved October 23, 2021, from https://zipschedules.com/restaurant-management/automated-restaurant.html

2. Team, D. (2021, September 7). *Benefits of Artificial Intelligence in the Restaurant Industry*. Deputy. Retrieved October 23, 2021, from https://www.deputy.com/blog/benefits-of-artificial-intelligence-in-the-restaurant-industry

3. Justin Lokitz. *The future of work: How humans and machines are evolving to work together* https://www.businessmodelsinc.com/machines/
4. Aurélion Géron *Hands on Machine Learning with Sci-Kit learn, keras and TensorFlow.*
5. Dr. Pierre Payeur, *COMMANDE PAR ORDINATEUR EN ROBOTIQUE Chapitre 3, CEG 4558*
6. Ionicframework. (2021, September 9). *What is hybrid app development?* Ionic Article.

   Retrieved October 20, 2021, from

https://ionic.io/resources/articles/what-is-hybrid-app-development.

7. Google. (n.d.). *Use The Cloud Firestore Rest API | firebase documentation*. Google. Retrieved October 21, 2021, from https://firebase.google.com/docs/firestore/use-rest-api.