Pepper Waiter HRI Project

Daniele Appetito 1916560
Salvatore Cognetta 1874383
Simone Rossetti 1900592

September 2021

All the students contributed equally to the project.

Contents

1 Introduction

Human Robot Interaction is the study of how one or more humans work with one robots in order to accomplish a goal. The Pepper Waiter project was devised to create such an interaction.

Pepper is a robot created by Softbank Robotics specifically to interact with people. It has tactile, visual, and audio sensors on its body to help a person convey information in a variety of ways. On top of that it is equipped with an android tablet on its "chest" that can be used for further interaction.

The aim for this project was to create a way for Pepper to interact (whether virtually or physically) with people in an environment of our choice.

In this project we decided to create a program that allowed to use the Pepper in a bar or restaurant environment to take an order instead of a human. This idea came to us by thinking of the current global situation and how the less human-to-human contact there is, the better. As such we decided to create a way to adhere to the social distancing norms by removing interaction with a human waiter.

2 Related works and solution

The main source of inspiration for the choice of this project was the currently ongoing COVID-19 pandemic, and the idea that we wanted to create a way to keep a somewhat social interaction while respecting social distancing norms. We did initially research other papers however, one of which was about the use of robots to mimic colloquial accents [1] in different languages to make a person feel more "at home", and ease the communication. A second paper we looked at involved the use of anthropomimetic robots [2] (i.e. robots that mimic not only th skeletal, but also the muscular structure of a human). A third paper we saw had to do with the portraying of emotions in a robot's face [3].

Some further research made us realise that we would need to have an exceedingly more complex robot in order to fully incorporate all the functions we studied above. As such we went over the more necessary functions that could be implemented and used by Pepper.

Because the aforementioned Softbank Robotics robot does not have any moving facial features (e.g. eyebrows, upper/lower lips, cheekbones, etc) we decided to ignore the facial features and instead incorporate a sort of emotion through words, such as using exclamation marks, ”please” and ”thank you”, etc. The same reasoning was done for the dialect and anthropomimetics, which cannot realistically be fully implemented on a Pepper robot, instead we decided to use multiple languages to help with communication. Seeing as Pepper has a somewhat similar physical structure to a human (humanoid head, humanoid hands, torso) we decided that it was a good enough attempt at mimicking human posture.

After a final research and consideration we decided that creating a robot waiter would be the best way to portray all of our ideas into a single project. In the problem we analysed earlier we concluded that we wanted a way to allow safe interactions in restaurants/bars that adhered to the social distancing norms imposed due to the current global pandemic. To solve this problem we decided to create a robot waiter that would help serve people in a restaurant. This solves the problem partially as it gets rid of a whole interaction step in the business by replacing it with a machine incapable of contracting diseases (given that it is kept clean and sanitised often).

3 Implementation

Different tools and API are used in order to create human interactions with a Pepper real robot or simulator, like:

- Softbank SDK;
- Naoqi softwares;
- MODIM - Multi-modalInteraction Manager.

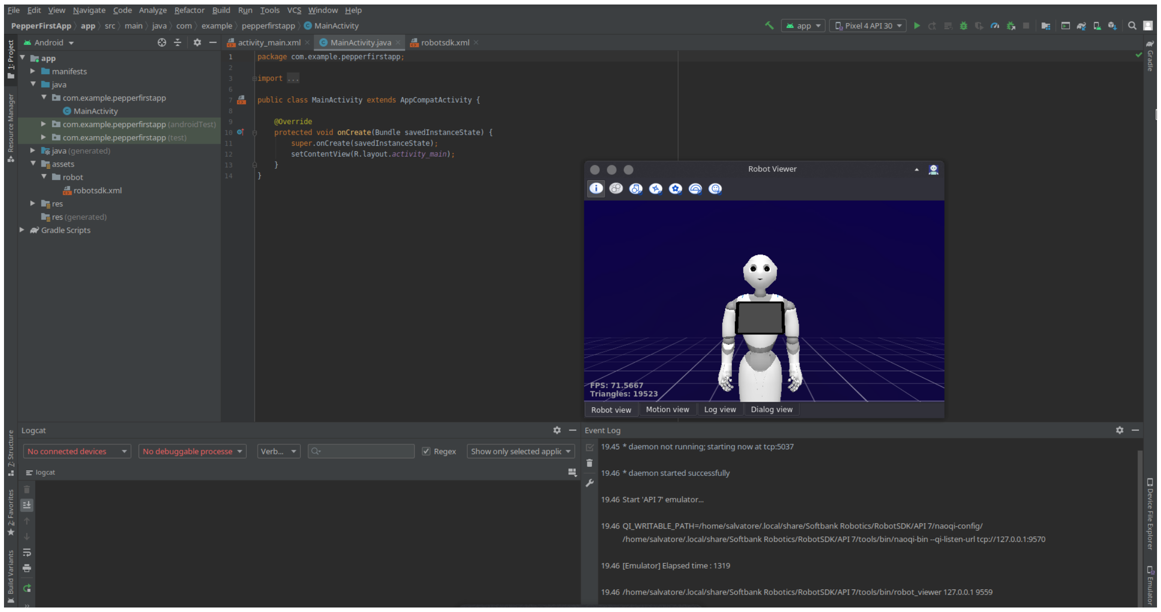
All of the tools are described in the following paragraphs.

3.1 Softbank SDK simulator

The Softbank company, which produces Pepper robots, has made avaible an SDK which allows to connect to a real robot or to create a simulation of the robot using Android Studio application (Softbank SDK). Because of Covid constraints, it was not possible to use directly the Pepper robot, placed in the laboratories of DIAG-University of La Sapienza: for this reason the simulatotion mode is was used.

This simulator gives the opportunity of visualize the movements, the dialogs and the interactions that the robot has with people, as can be seen in figure [1]

Therefore, to be able to interact and communicate with Pepper, this simulator is used as a client interface, which is connected to a docker image containing the `naoqi` Operative System, containing all the tools and softwares used to run and operate the robot itself.
In the simulator, as can be seen from the figure 1, there is also a tablet on the chest of the robot. This tablet can be virtualized through another library, modim, later detailed.



```java
package com.example.pepperfirstapp;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
19.45 * daemon not running; starting now at tcp:5037

19.46 * daemon started successfully

19.46 Start 'API 7' emulator...

19.46 QI_WRITABLE_PATH=/home/salvatore/.local/share/Softbank Robotics/RobotSDK/API 7/naoqi-config/
      /home/salvatore/.local/share/Softbank Robotics/RobotSDK/API 7/tools/bin/naoqi-bin --qi-listen-url tcp://127.0.0.1:9570

19.46 [Emulator] Elapsed time : 1319

19.46 /home/salvatore/.local/share/Softbank Robotics/RobotSDK/API 7/tools/bin/robot_viewer 127.0.0.1 9559
```

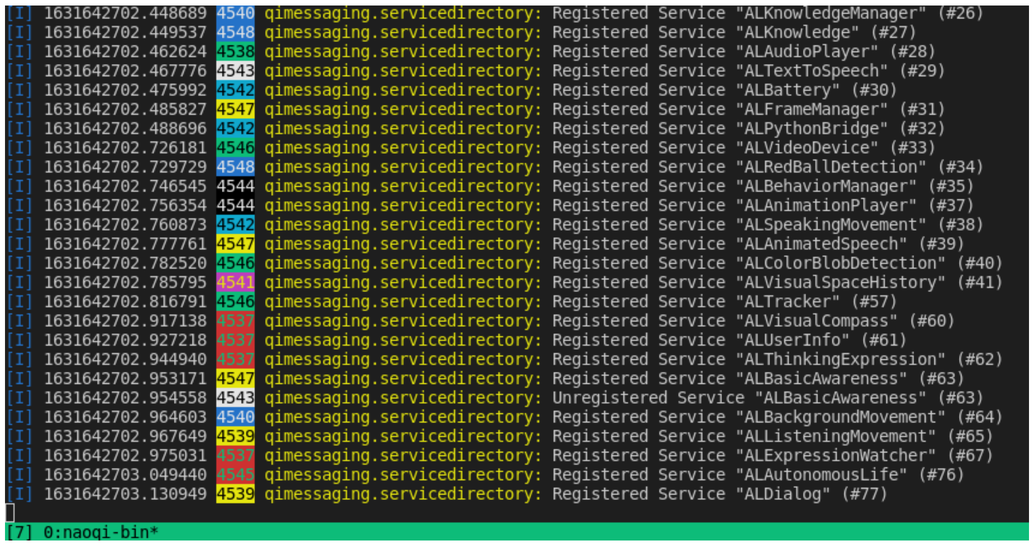
Figure 1: Pepper simulator in Android Studio.

3.2 Naoqi - Pepper tools

As said before, the OS of Pepper is virtualized thanks to a docker image that is running on our host systems (For further references see the following Git Repository). Once inside the docker image the `naoqi` software must be started in order to initialize and register all services of Pepper, like Dialogs, TextToSpeech, SpeechRecognitrion and more.

This gives us the possibility of command Pepper, indeed inside the docker image different tools are present, with wich one can set some robot parameters like joint angles and the posture, or simulate an input on a sensor like the approach of an human being or a person that talks with Pepper.

Beyond the already presented tools, obviously scripts created from us can be runned inside naoqi, in order to set the behaviour of the robot. These scripts can be put inside a shared falder between the host system and the docker image (the directory is `/home/pc_name/playground`). Inside this directory the Pepper Waiter softwares are positioned and developed .



```
[I] 1631642702.448689 4540 qimessaging.servicedirectory: Registered Service "ALKnowledgeManager" (#26)
[I] 1631642702.449537 4548 qimessaging.servicedirectory: Registered Service "ALKnowledge" (#27)
[I] 1631642702.462624 4538 qimessaging.servicedirectory: Registered Service "ALAudioPlayer" (#28)
[I] 1631642702.467776 4543 qimessaging.servicedirectory: Registered Service "ALTextToSpeech" (#29)
[I] 1631642702.475992 4542 qimessaging.servicedirectory: Registered Service "ALBattery" (#30)
[I] 1631642702.485827 4547 qimessaging.servicedirectory: Registered Service "ALFrameManager" (#31)
[I] 1631642702.488696 4542 qimessaging.servicedirectory: Registered Service "ALPythonBridge" (#32)
[I] 1631642702.726181 4546 qimessaging.servicedirectory: Registered Service "ALVideoDevice" (#33)
[I] 1631642702.729729 4548 qimessaging.servicedirectory: Registered Service "ALRedBallDetection" (#34)
[I] 1631642702.746545 4544 qimessaging.servicedirectory: Registered Service "ALBehaviorManager" (#35)
[I] 1631642702.756354 4544 qimessaging.servicedirectory: Registered Service "ALAnimationPlayer" (#37)
[I] 1631642702.760873 4542 qimessaging.servicedirectory: Registered Service "ALSpeakingMovement" (#38)
[I] 1631642702.777761 4547 qimessaging.servicedirectory: Registered Service "ALAnimatedSpeech" (#39)
[I] 1631642702.782520 4546 qimessaging.servicedirectory: Registered Service "ALColorBlobDetection" (#40)
[I] 1631642702.785795 4541 qimessaging.servicedirectory: Registered Service "ALVisualSpaceHistory" (#41)
[I] 1631642702.816791 4546 qimessaging.servicedirectory: Registered Service "ALTracker" (#57)
[I] 1631642702.917138 4537 qimessaging.servicedirectory: Registered Service "ALVisualCompass" (#60)
[I] 1631642702.927218 4537 qimessaging.servicedirectory: Registered Service "ALUserInfo" (#61)
[I] 1631642702.944940 4537 qimessaging.servicedirectory: Registered Service "ALThinkingExpression" (#62)
[I] 1631642702.953171 4547 qimessaging.servicedirectory: Registered Service "ALBasicAwareness" (#63)
[I] 1631642702.954558 4543 qimessaging.servicedirectory: Unregistered Service "ALBasicAwareness" (#63)
[I] 1631642702.964603 4540 qimessaging.servicedirectory: Registered Service "ALBackgroundMovement" (#64)
[I] 1631642702.967649 4539 qimessaging.servicedirectory: Registered Service "ALListeningMovement" (#65)
[I] 1631642702.975031 4537 qimessaging.servicedirectory: Registered Service "ALExpressionWatcher" (#67)
[I] 1631642703.049440 4545 qimessaging.servicedirectory: Registered Service "ALAutonomousLife" (#76)
[I] 1631642703.130949 4539 qimessaging.servicedirectory: Registered Service "ALDialog" (#77)

[7] 0:naoqi-bin*
```

Figure 2: Naoqi SDK: overview of service registered on start.

3.3 Modim

Like described before, the app running on Pepper's tablet can be virtualized through antoher library called Modim (For further references see the following Git Repository)

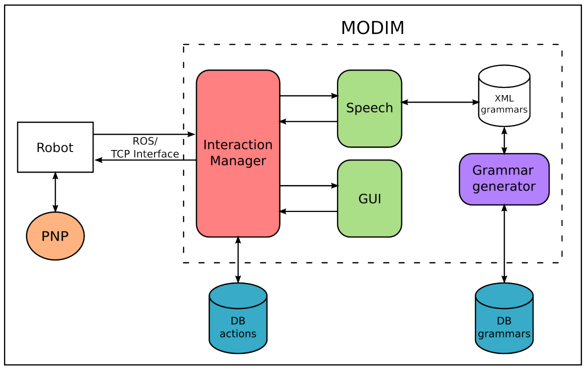
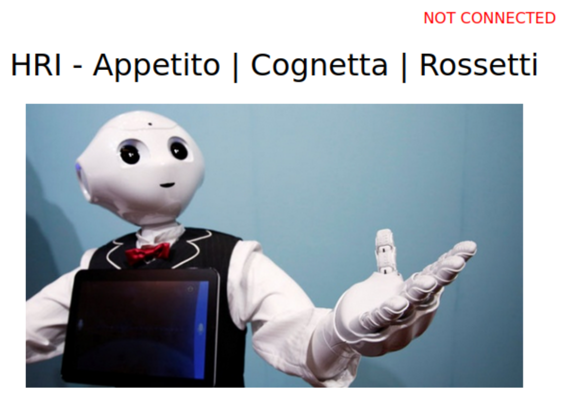


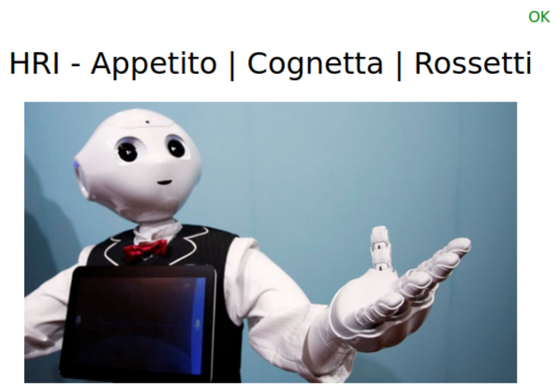
Figure 3: Multi-modal Interaction Manager software architecture.

The Multi-modalInteraction Manager (MODIM), the Graphical User Interface (GUI) and the speech system allow the use of different modalities for the output and input of information from the robot to the user and viceversa. The output modalities considered are the use of texts, images or videos using the GUI or by voice using the speech synthesis. Input modalities are the touch-screen (i.e., with the use of buttons on the GUI) or spoken inputs interpreted by the speech recognizer. The general architecture of this Manager is shown in figure [3]

Therefore, after connecting the Interaction Manager with Pepper, thanks to this library the app on the tablet can be simulated on a web browser; the user can interact with this one with buttons present on the GUI, both clicking on them through the web browser and giving spoken input (e.g. simulating them thanks to the `naoqi` software).



(a) Modim not connected to Pepper.



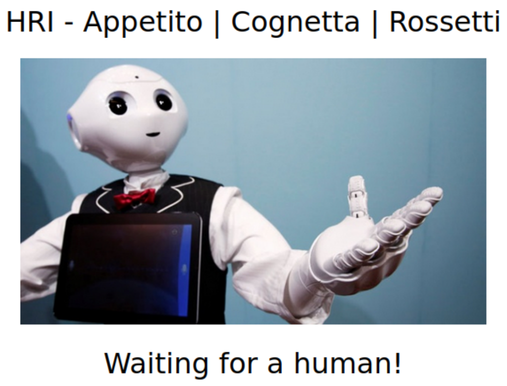
(b) Modim connected to Pepper.

Figure 4: Connection of MODIM to the robot

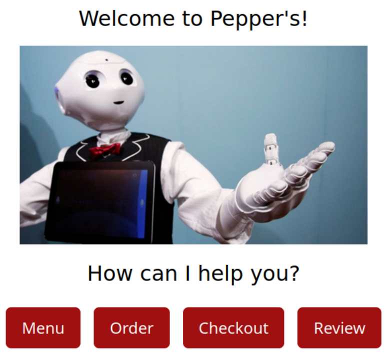
4 Results

Pepper Waiter works as a virtual (or physical, although we could only test it virtually) waiter that is designed to welcome customers to a restaurant, show the menu and take their order, as well as assist them in paying the bill and finally take a satisfactory questonnaire.

Initially the robot is waiting to interact with a person in front of him. When a person approaches Pepper, its sonar front sensors will detect the person and greet the customer with a welcome message both on its front tablet, and by voice. From there a customer can choose from a 4 options: ”Menu”, ”Order”, ”Checkout”, and ”Review”. All of these options being self explanatory and selecting each word will take the customer to a separate screen for the according action.



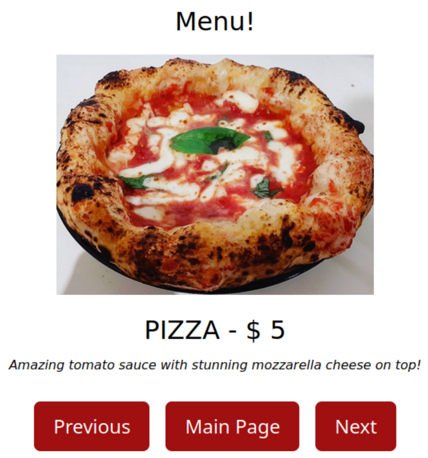
(a) Pepper is waiting for a person.



(b) Pepper detect a person in front of him.

Figure 5: Approach of a human to Pepper Waiter

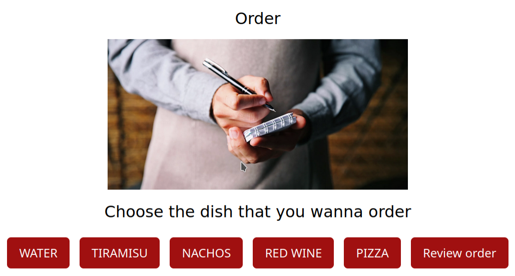
The ”Menu” screen consists of an individual page for each item of the menu where one can read the description, the ingredients, and the price as we can see from figure [6]. Pepper will also read the description out loud. The dishes present in the menu can be changed, addded or removed easily using a csv file, which for each dish contains a tuple of $(food\ name, path\ to\ image, price, description)$. This csv file is then imported inside the python script and a list of `Food` objects is dinamically created.



Amazing tomato sauce with stunning mozzarella cheese on top!

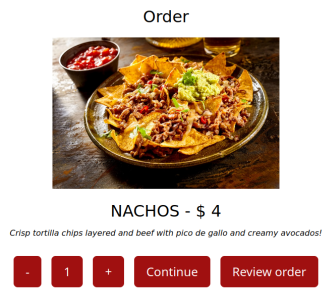
Figure 6: Example of the menu showing a photo, name, description and price of the dish.

After having decided on what to order from the menu the customer can the return to the main page described in the previous paragraph to continue on with the order. In the "Order" screen one can choose any of the available dishes (in this example pizza, nachos, tiramisu) and drinks (water, red wine), as shown in figure [7a]. After clicking on one of the aforementioned buttons the customer can then choose the quantity of the food/beverage that they want, adding or removing an item of that dish, as can be seen in figure [7b]. They can continue the order, going to the previous page, choosing another dish or review directly the order.



Choose the dish that you wanna order

(a) Ordination page.

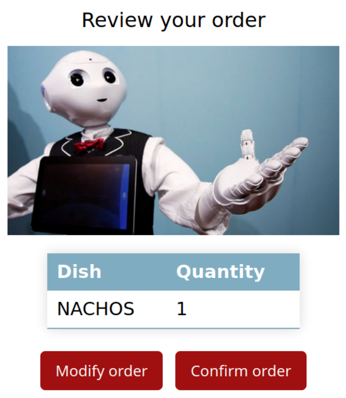


Crisp tortilla chips layered and beef with pico de gallo and creamy avocados!

(b) Choose the quantity.

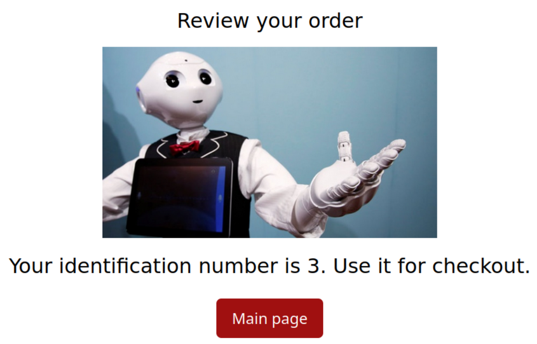
Figure 7: Ordination of a dish.

Once they are fully ready they can click on "Review Order" to review what they have ordered, in a table format shown in figure [8a]. From there the customer will have the possibility of altering the order one or confirming it. Once confirmed, the customer will receive an order number that will be then used to recognise the order at checkout. After this the robot will automatically go back to the main menu page, in order to prepare to serve someone else.


Review your order

Dish	Quantity
NACHOS	1

(a) Review of the ordination.


Review your order

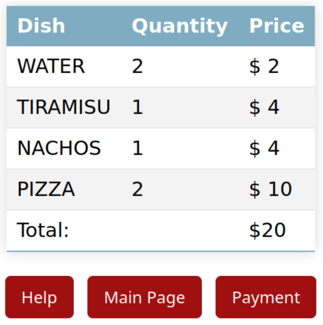
Your identification number is 3. Use it for checkout.

(b) Number used for checkout.

Figure 8: Confirmation of the order.

Once the customer is finished with the food they can then walk over to Pepper again and (after being greeted again) can press the "Checkout" button, where a list of ids are shown. After clicking the previously assigned id, a lists of the ordered items and the total price will be displayed, as can be seen in figure [9].

In this page one will be able to review their order after stating the order number assigned to them earlier.
After this a "fake" payment procedure, with a qr code, is then done (see figure [10]). This procedure will automatically prompt a succesfull payment landing page, but can be substituted with a real one, using the preferred API for payment. Once the payment is confirmed the customer can choose to leave the restaurant or leave a review of the service where they'll have to rate their experience with Pepper on a scale of 1 to 5.



Dish	Quantity	Price
WATER	2	$ 2
TIRAMISU	1	$ 4
NACHOS	1	$ 4
PIZZA	2	$ 10
Total:		$20

Figure 9: Checkout

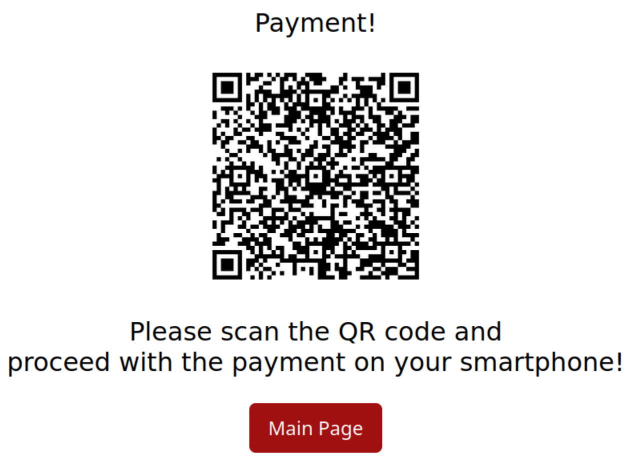
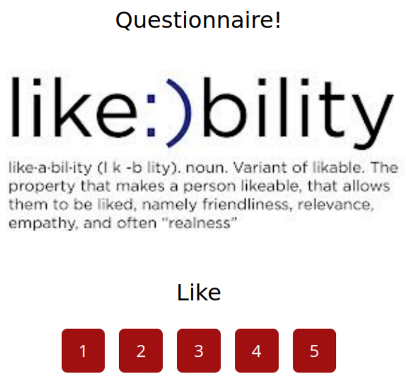
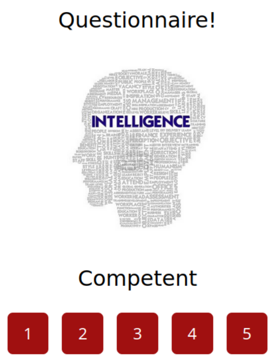

Payment!
Please scan the QR code and
proceed with the payment on your smartphone!
Main Page

Figure 10: Example of payment via qr code.

The questionnaire is subdivided into two categories:
i) intelligence perceived and *ii*) likelability.
In the first case the human can give a score on the following characteristics: 1. Knowledgeable; 2. Responsible; 3. Intelligent; 4. Sensible.
While in the second case the characteristics in question are: 1. Friendly; 2. Kind; 3. Pleasant; 4. Nice.
All the reviews are then locally stored logging into modim thanks to `im.logdata()` and can be later used to improve the behaviour of Pepper Waiter.


Questionnaire!
like:)bility
like-a-bil-ity (l k -b lity). noun. Variant of likable. The property that makes a person likeable, that allows them to be liked, namely friendliness, relevance, empathy, and often "realness"
Like
1
2
3
4
5

(a) Review of likeability.



(b) Review of intelligence.

Figure 11: Questionnaire

5 Conclusion

We created a program for the Pepper robot designed to interact with humans in a customer-waiter way. The project itself was fun and challenging. One of our greater challenges was the setup of all the interconnecting API and libraries as well as the simulation start on android studio. However, after the initial hickups on starting to use the program, the process was streamlined and entertaining. We learned a lot about human-robot interaction application creations, as well as what Pepper is capable of doing.

Something that would be nice to try in the future is to actually use the program on Pepper itself rather than a simulation. Due to the current pandemic we were not allowed physical access to the robot. I think seeing the program work on a fully functioning Pepper robot live would be the next step in looking at improving the project. One thing that could be done to further improve on this project is create a navigation program that would let Pepper guide you to the desired free table before assisting you with an order. Of course, the most ideal improvements to this experience would be using a robot with a more anthropomimetic figure (i.e. a full human body, which currently do not exist at this level of automation) and with a range of changing facial features, in order to express contempt/happiness. This would allow it to more accurately mimic a real human-to-human interaction.

References

[1] Marilyn A. Walker et al. "PARADISE: A Framework for Evaluating Spoken Dialogue Agents". In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics.* ACL '98/EACL '98. Madrid, Spain: Association for Computational Linguistics, 1997, 271–280. DOI: 10.3115/976909.979652. URL: https://doi.org/10.3115/976909.979652.

[2] Steffen Wittmeier et al. "Toward Anthropomimetic Robotics: Development, Simulation, and Control of a Musculoskeletal Torso". In: *Artificial Life* 19 (2013), pp. 171–193. URL: https://www.semanticscholar.org/paper/Toward-Anthropomimetic-Robotics%3A-Development%2C-and-a-Wittmeier-Alessandro/1ae4f8a739efbaf8609ab08f2c3e2606892a7899?p2df.

[3] Cynthia Breazeal. "Emotion and sociable humanoid robots". In: *International Journal of Human-Computer Studies* 59.1 (2003). Applications of Affective Computing in Human-Computer Interaction, pp. 119–155. ISSN: 1071-5819. DOI: https://doi.org/10.1016/S1071-5819(03)00018-1. URL: https://www.sciencedirect.com/science/article/pii/S1071581903000181.