Αλέξης Φιλιππακόπουλος 3190212

Δημήτρης Καββαδάς 3190064

Λευτέρης Γεωργιάδης 3190031

Human - Computer Interaction

(https://drive.google.com/drive/folders/1b2HiS_iyb9dVdbBbAVluIvV9xTRkcEJ8?usp=sharing)

EasyWash

EasyWash is a user interface for washing machines designed to ameliorate the user experience of visually impaired people. The interface leverages techniques of computer vision, speech recognition and natural language processing to deliver a user-friendly experience for the specified target group.

1st Cycle

During this cycle the initial idea was to find which household appliance interface we should tackle. For the time being, the target group was solely people with limited vision and mostly elderly people. The reason was that rarely this target group is taken into consideration when designing most household appliances' interface. We decided to talk to some visually impaired elderly people since they are also not too keen on the latest technologies and most of them agreed that the newer washing machines are challenging because:

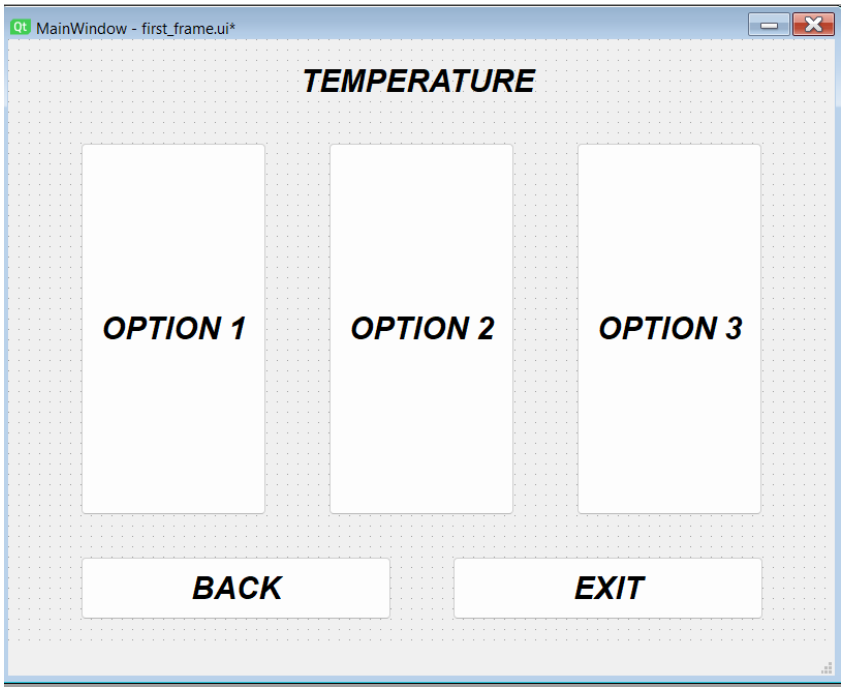
1. Most of the buttons-dials are not self-explanatory.
2. The font is too small.

Thus, the goal of this cycle was to design an interface centered around transparency, user-friendly and self-explanatory terminology, as well as having the system coordinate the interaction in order to make the experience more intuition-based and less initiative-based. Moreover, we aim for simplicity in the design pattern and thus opted for two-three large buttons

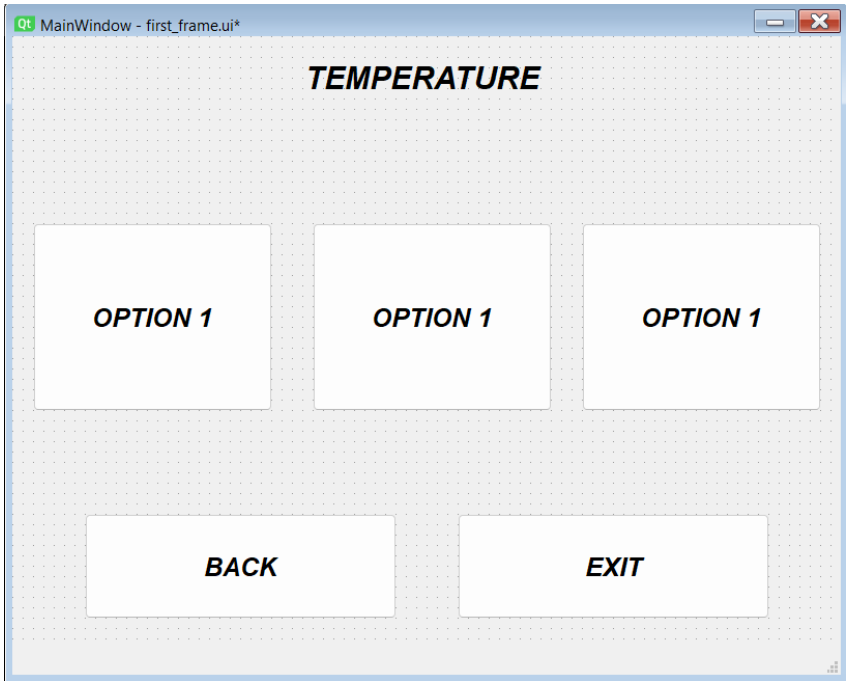
with large fonts that cover most of the screen to minimize the -possible- actions for the user and hence adapting the design for a more straightforward and conspicuous one to aid with the visual challenges of the targeted subjects. The application's main metaphor is to mimic a catalogue and only requiring the user to press one button each time while being provided with the possible answers, in an attempt to decrease the memory load and take away any initiative from the user. By designing the application as such we also achieve a continuous consistency of actions-objects within the designed model, further simplifying the process. We implemented three screens one for the temperature, one for the duration and one for the cycle type.

Here are some of the prototypes we created:

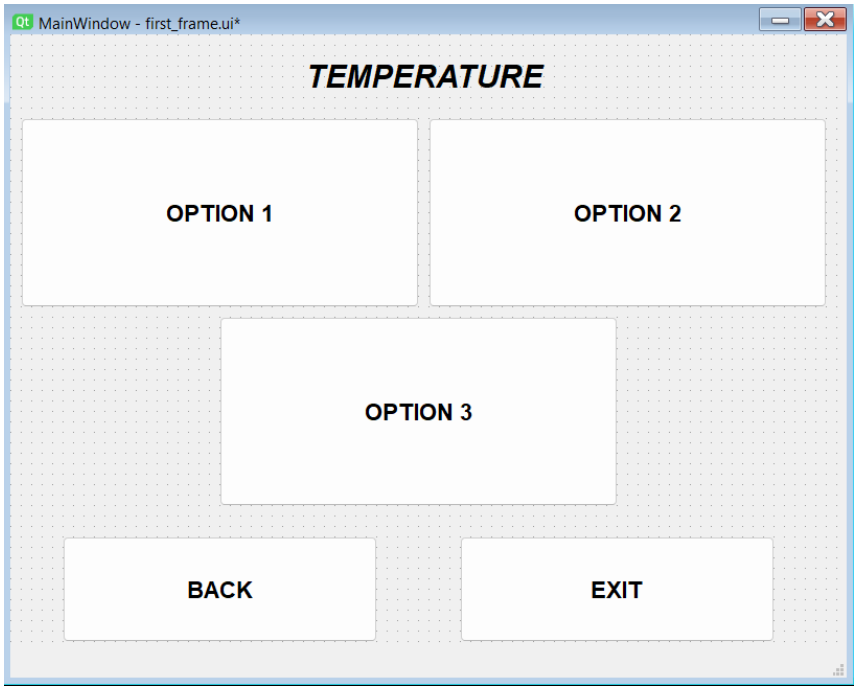
Design 1



Design 2

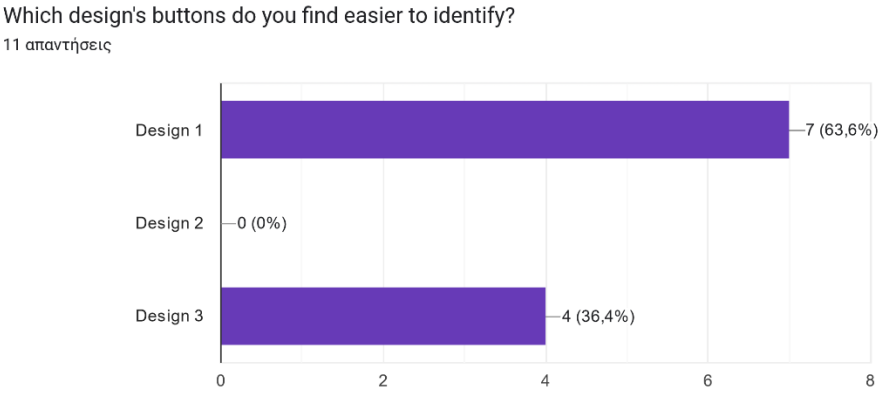


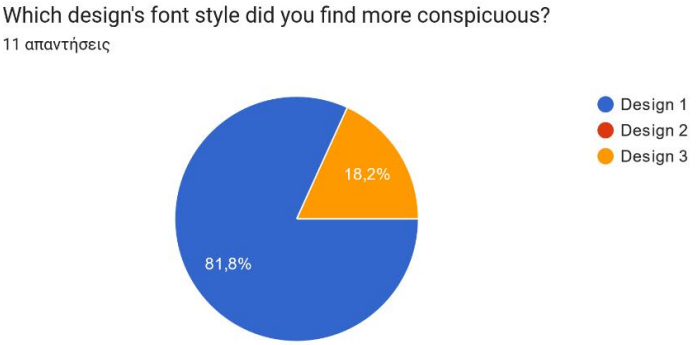
Design 3

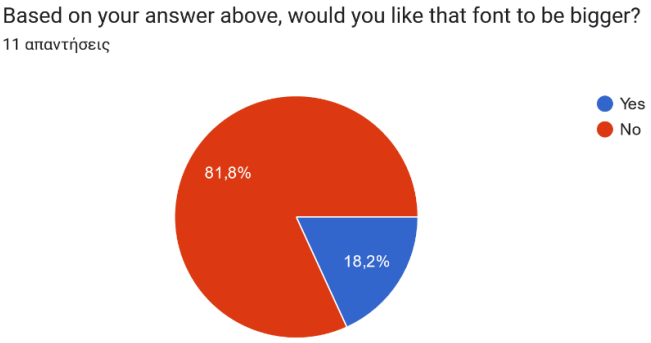


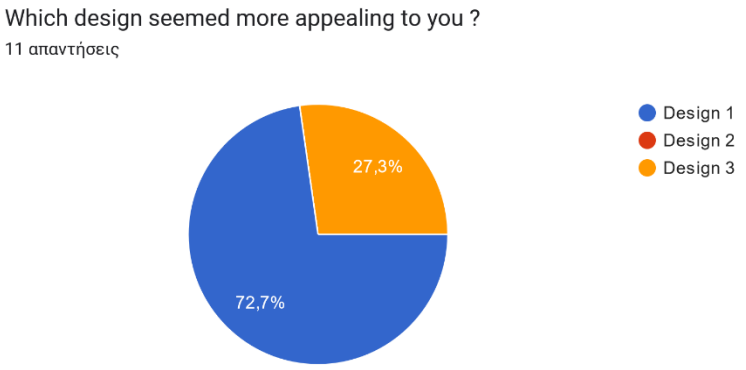
Evaluation

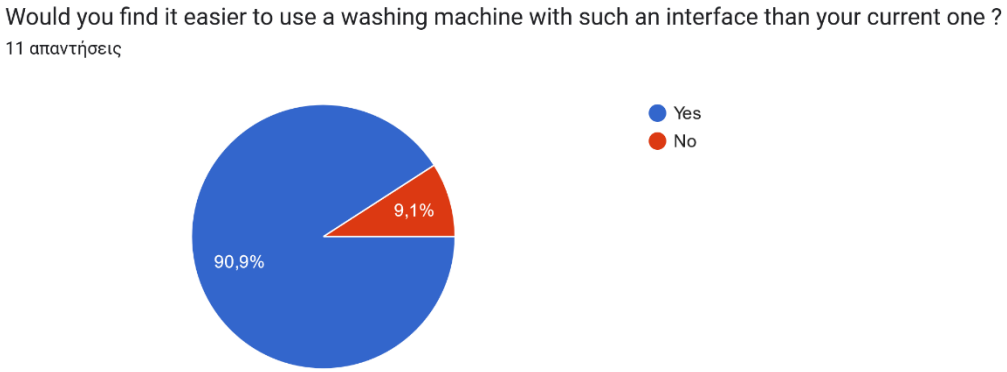
To evaluate our designs, we created a questionnaire. The questions were aimed at the design aspects of the interface and the subjects were the visually impaired elderly people that prompted us with the idea of a washing machine interface. Here are the results:











Analyzing the questionnaire, we decided on increasing the font size and using the first design as your main one.

As a second means of evaluation, we conducted a usability inspection by walking-through the application and timing how long it took to complete the task, which was to successfully start a washing cycle. Each one of us estimated how long it would take and attempted to walk-through the application while timing themselves and acting as a novice user. Here are the results:

Subject	Time Estimation	Actual Time
Alex	10s	8s
Dimitris	15s	7s
Lefteris	10s	9s

Average Time : 8s

We were content with the actual time it took to complete the process and thus decided to go with the first design and focus on improving the experience and applying it to a larger target group.

2nd Cycle

Using the feedback from the questionnaire we decided that the main design outline would be the first design with its font scaled up slightly and with the addition of a soft blue background for some color contrast and two distinct, to the eye, icons for the back and exit buttons. During the second cycle we decided to include people with zero visual capabilities in our target group because we figured that by including speech recognition and a voice assistant to coordinate the process, we could appeal to that group as well. Therefore, the goal was to introduce a dialogue system in the application.

Capturing Speech

We use python's pyaudio library to establish a microphone connection and capture the audio. We record the audio in 1024-byte chunks represented by 16-bit integer values (paInt16). We capture 44.100 samples per second and open the microphone for 3.5 seconds each time. In addition, we record in mono and not stereo (CHANNELS=1).

```
CHUNK_SIZE = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
SECONDS = 3.5
FILEPATH = 'data/input_rec.wav'
```

We decided to record audio only after each question to reduce bandwidth and unnecessary computational workload since the interface is designed to run as an IoT device. This implementation detail creates the need to coordinate when the subject is supposed to prompt the

assistant with its input. To tackle this, we drew inspiration from the ‘smart’ design of traffic lights that emit sounds at different rates to signal whether the light is green or red. The main principle of this design pattern is to utilize sound stimuli to signal a certain event. Therefore, we added two beep sounds, one to signal that the assistant is expecting vocal input, in other words an answer, and one to signal that no further input is being recorded. Upon testing we decided that 3.5 seconds was enough to answer each question without feeling that it is taking too long.

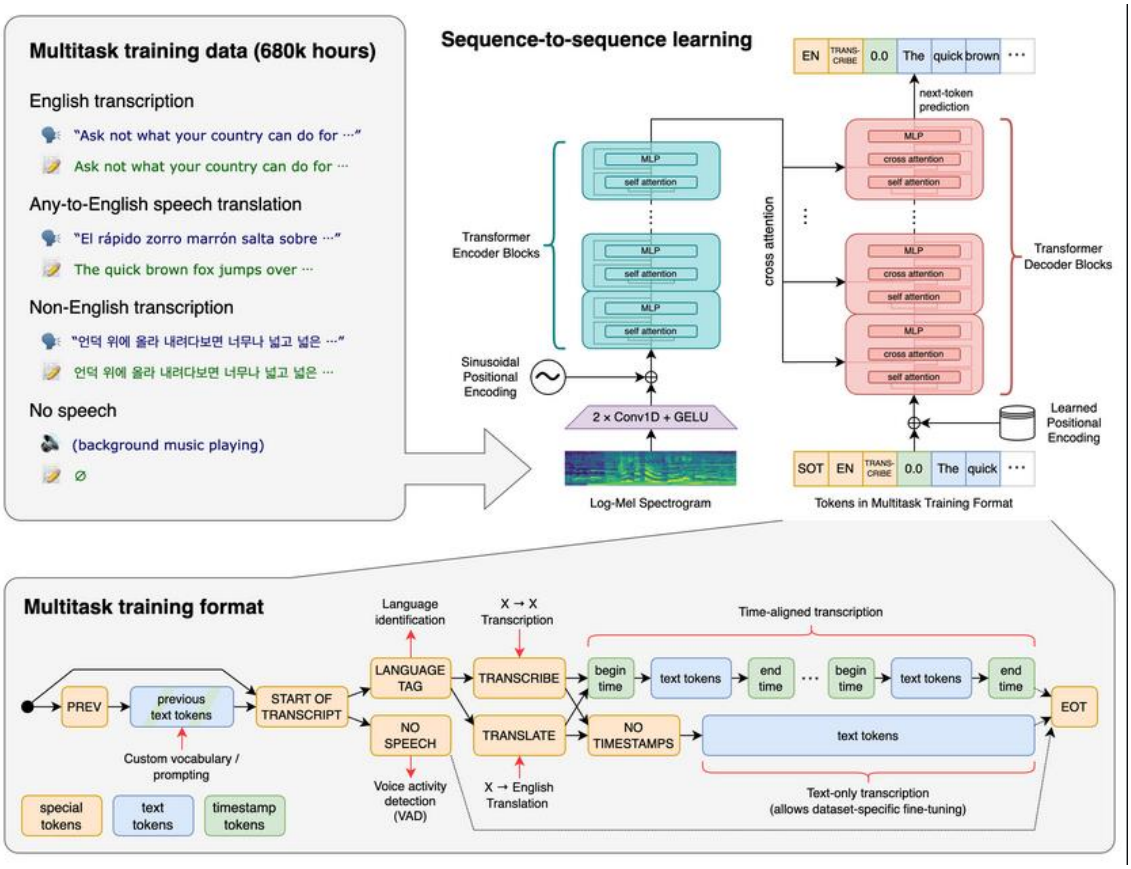
```
mic = pyaudio.PyAudio()
stream = mic.open(format=FORMAT, channels=CHANNELS, rate=RATE,
input=True,  output_device_index=0, frames_per_buffer=CHUNK_SIZE)
frames = []
print('recorded started')
winsound.Beep(300, 300)
for i in range(0, int(RATE / CHUNK_SIZE * SECONDS)):
    data = stream.read(CHUNK_SIZE)
    frames.append(data)
winsound.Beep(300, 200)
print('recorded ended')
stream.stop_stream()
stream.close()
mic.terminate()
threading.Thread(target=self.store_input, args=(frames, mic)).start()
```

Following this we use the wave library to concatenate all the recorded samples and store them as a single wave (.wav) file locally. Afterwards this becomes the inputs to our speech recognition model to decode the vocal input to natural language.

```
def store_input(self, frames, mic):
    file = wave.open(FILEPATH, 'wb')
    file.setnchannels(CHANNELS)
    file.setsampwidth(mic.get_sample_size(FORMAT))
    file.setframerate(RATE)
    file.writeframes(b''.join(frames))
    file.close()
    print('recording stored')
    file_event.set()
    threading.Thread(target=self.decode_input, args=()).start()
```

Speech Recognition

As a summary we decided on using OpenAI's Whisper model, since it is by far the best model currently available. More specifically we used the English-only base model comprised of 74 million parameters (approximately 1GB). Whisper utilizes an encoder – decoder transformer architecture. It resamples the audio input to 16.000 Hz and an 80-channel log-magnitude Mel spectrogram (visual representation of sound waves on the Mel scale) is computed on 25-millisecond windows with a stride of 10-milliseconds. Afterwards, all the input values are scaled to be in the [-1, 1] range and forward propagated through the first two convolutional-layers with a filter of width 3, a GELU activation function, and a stride of 2 on the second layer. Then Sinusoidal position embeddings are added, and the forward propagation continues through the encoder and the decoder.



Therefore, we load the wave file containing the subject's verbal input and preprocess it, we pad the 3.5 second recording with zeroes to extend it to the desired 30 second format, compute the Mel spectrogram of the input, set the decoding options and obtain the transcript of the input audio via forward propagation.

```
def decode_input(self):
    file_event.wait()
    file_event.clear()
    input = whisper.load_audio(file=FILEPATH)
    input = whisper.pad_or_trim(input)
    mel = whisper.log_mel_spectrogram(input).to(self.model.device)
    options = whisper.DecodingOptions(fp16=False, language='en')
    result = whisper.decode(model=self.model, mel=mel, options=options)
    self.speech_signal.emit()
    self.result_queue.put(result.text)
    return
```

Voice Assistant

For the voice assistant we decided on using a text-to-speech model rather than a large language model (LLM) as we wanted to have full control over the assistant's output and to not add additional memory needs to an interface designed as an IoT device. The whole process is designed based on a question – answer motif in an attempt to produce an intuitive feel when interacting with the interface and increase its transparency. We used python's pyttsx3 library and created a text file (.txt) in a dictionary form where each key is our evaluation of the user's input for a specific question (all possible answers) and the corresponding value is what the model's output should be for said answer. We account for exit and back (undo) functionalities as well as a message for when the system cannot infer on the user's input while also providing a possible solution for it. Additionally, the user is guided by sound stimuli in order to know when an answer is expected. We opted for a voice assistant as a technique of anthropomorphism, trying to make the experience more natural while providing guidance to the subject. Another detail is that

after each answer we repeat what the system ‘understood’ in order to notify the subject of any mismatch between its choice and system’s interpretation of it.

Input Evaluation

The interface is designed in such a way that each frame corresponds to a different ‘question’ / ‘page of catalogue’ (number of degrees, duration and cycle type) and each ‘question’ has 2 – 6 unique and straight-forward answers that are preset, as well as back and exit options. Since the inputs are short, from one word to one sentence, we perform simple keyword-occurrence classification to evaluate each frame’s possible options. We define unique dictionaries where the keys are the possible (preset) answers and the values the keywords associated with said category. We preprocess whisper’s output by removing the punctuation and white space to evaluate it accordingly. When no keywords are detected, the assistant is prompted to ask the subject to repeat its answer.

```
# Dictionaries used to evaluate client's verbal input
self.assist_frame_eval_dict = {'yes' : ['yes', 'yea', 'ye', 'sure', 'help', 'assist'], 'no' : ['no', 'nope', 'not', "n't", "don't"]}
self.first_frame_eval_dict = {'yes': ['yes', 'recommend', 'sure', 'yeah', 'yea', 'propose'], 'no' : ['no', 'nope', 'not', "n't", "don't", 'o
self.second_frame_eval_dict = {0: ['first', 'one','sixty', '1', '60', 'less', 'hour'], 1: ['two', '2', 'second', 'less', 'hours'], 2: ['thir
self.third_frame_eval_dict = {'light': ['light', 'white', 'gray', 'soft'], 'dark': ['black', 'dark', 'heavy'], 'mixed': ['mixed', 'both']}
self.fourth_frame_eval_dict = {'sensitive': ['light', 'sensitive'], 'normal': ['clothes', 'normal', 'plain', 'cotton'], 'heavy': ['heavy', '
self.fifth_frame_eval_dict = {'small': ['small', 'little'], 'medium': ['medium'], 'large': ['large', 'lot']}
self.mycycle_hour_eval_dict= {'30': ['thirty', 'thirtyminutes', 'half', 'halfanhour', '30'], '45': ['fourty', 'five', 'minutes', 'fourtyfive
self.mycycle_temp_eval_dict = {'30': ['thirty', '30'], '40': ['fourty', '40'], '60': ['sixty', '60', '16']}
self.back_exit_xpln_eval_dict = {'back': ['goback', 'back', 'previous', 'last', 'previousquestion', 'lastquestion'], 'exit': ['start', 'over
self.mycycle_type_eval_dict = {'sensitive': 'sensitive', 'normal': 'normal', 'heavy': 'heavy'}
# Dictionaries used to associate current frame with appropriate evaluation
self.frame_to_eval_dict = {0: self.back_exit_xpln_eval_dict, 1 : self.assist_frame_eval_dict, 2 : self.first_frame_eval_dict, 3: self.second
self.frame_to_option_eval_dicts = {1 : self.assistant_frame_option_eval, 2 : self.first_frame_option_eval, 3: self.second_frame_option_eval,
```

```
    """
    Takes the client's verbal response as a string.
    Uses a dictionary where the keys are the question's options and the values keywords that hint towards it.
    Returns one option based on the highest number of keywords present in the response
    """
    matches_per_option = {}
    for key, value in self.frame_to_eval_dict[self.current_widget_index].items():
        matched_words = 0
        for v in value:
            matched_words += 1 if response.__contains__(v) else 0
        matches_per_option[key] = matched_words
        print(f'Key: {key} Matched Words: {matched_words}')
    return 'None' if all(value == 0 for value in matches_per_option.values()) else max(matches_per_option, key=matches_per_option.get)
```

Evaluation

The evaluation of this cycle was centered on the voice assisted aspect of the process.

1) Performance Measurement:

   We assembled 5 subjects and gave them a goal that they must start a washing cycle using the voice assistant. We did not inform them on how the process is carried out or what to expect, in order to simulate a scenario of a novice user that has never interacted with the interface. We gathered the following data after observing the subjects.

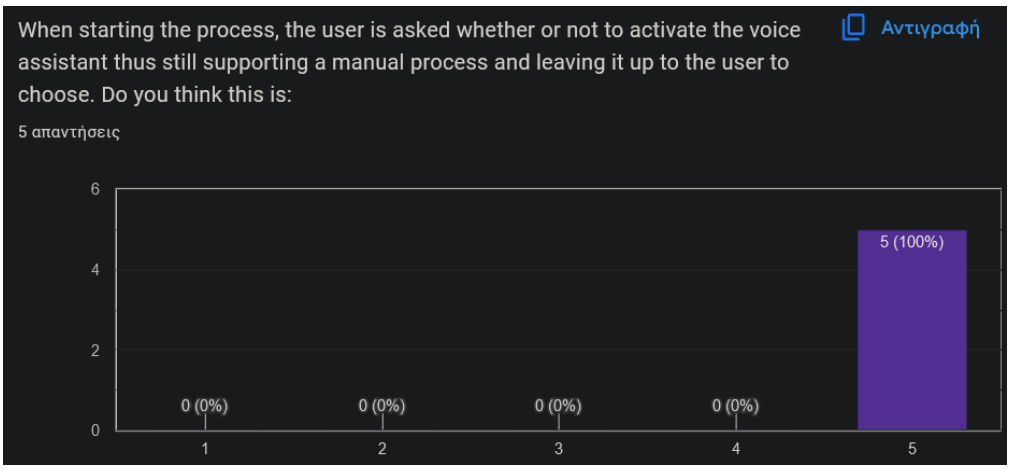
Subject ID	0	1	2	3	4
Times an action had to be repeated	0	0	1	0	1
Times subject did not act within the designated timeframe	0	0	0	0	0
Time to complete the process	1’10”	1’11”	1’23”	1’12”	1’20”

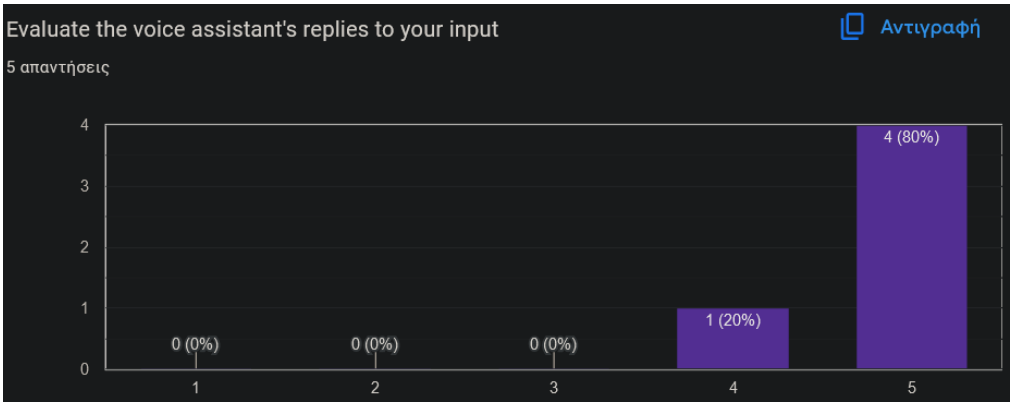
Average Duration: 1’15’’ (rounded to the closest second)

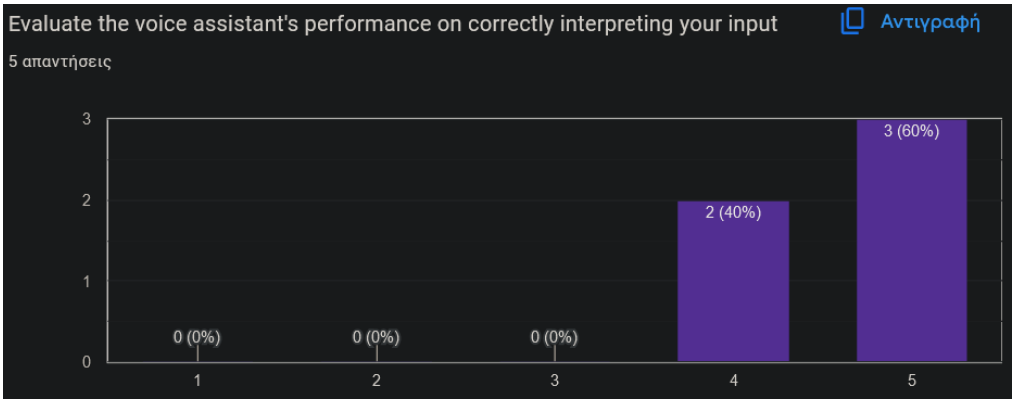
Average Actions Taken: 3,4 (Keep in mind the minimum is 3)

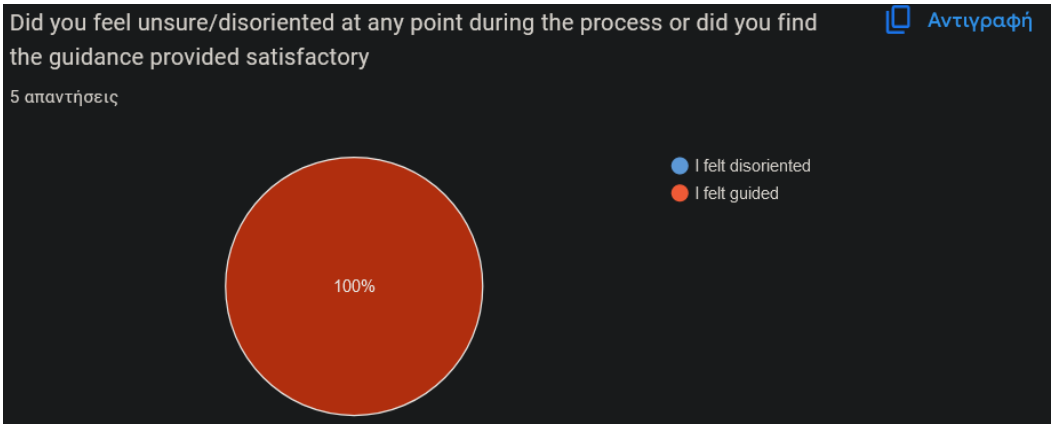
We see that the times do not differ significantly as the process is catered to be of a standard time frame, unless you must repeat your answer as shown by the data gathered in this stage. We were very content with the results as we wanted to keep the procedure close to a minute and under two. Furthermore, we were reassured that the system is robust to the input interpretation as shown by the number of times an action had to be retaken.

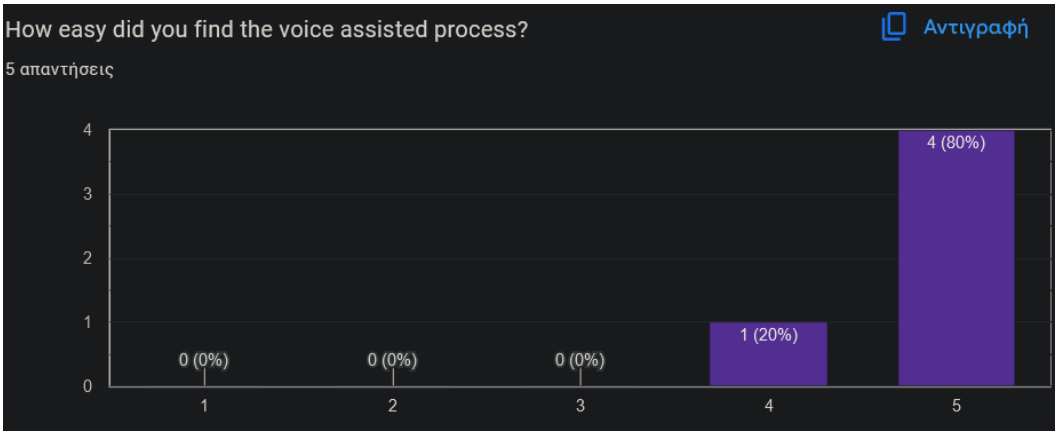
2) After the process the subjects were given the following questionnaire:

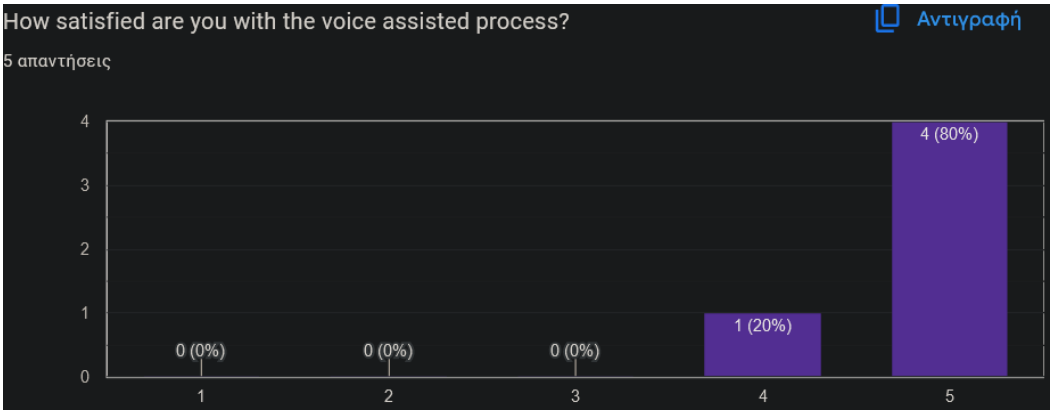


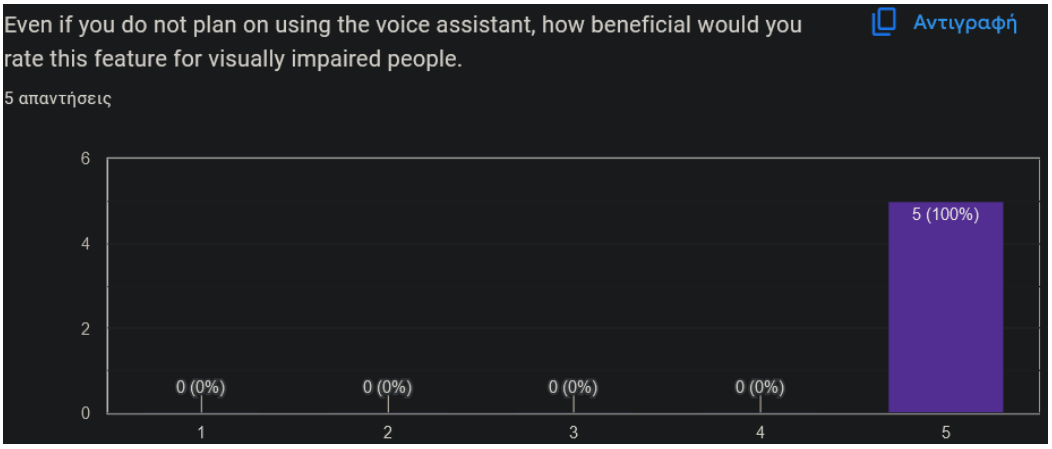





Did you feel unsure/disoriented at any point during the process or did you find the guidance provided satisfactory
Αντιγραφή
5 απαντήσεις
I felt disoriented
I felt guided
100%


How easy did you find the voice assisted process?
Αντιγραφή
5 απαντήσεις
4
3
2
1
0
0 (0%)
0 (0%)
0 (0%)
1 (20%)
4 (80%)
1
2
3
4
5


How satisfied are you with the voice assisted process?
Αντιγραφή
5 απαντήσεις
4
3
2
1
0
0 (0%)
0 (0%)
0 (0%)
1 (20%)
4 (80%)
1
2
3
4
5



3) Cognitive Walkthrough

We performed a cognitive walkthrough of the application based on a scenario where we are a novice user, the aim is to successfully start a washing cycle using the voice assistant. We accessed that the system provided investigative learning since it does not allow for the user to initiate actions but rather to pick from a set of possible-given answers and then instantly being redirected to the next ‘question’, thus eliminating any memory load on the subject as well as automating the process to always induce a feeling of guidance while interacting with the interface. Inherently, this minimizes both the memory load and the actions needed from the subject. In addition, we support undo and exit functionalities thus enabling a change of answer or an abortion of the process given an according command. At the beginning the user is informed on how the process is structured (e.g., when an answer is expected, how to navigate using voice commands etc.) while also being given sufficient guidance, according to us and the questionnaire, on when the system cannot interpret its input – how to overcome this and when the input is expected. Concluding, we deduced that there should be neither an estimation nor a execution gap. We were content with the evaluation of the second cycle and for the third cycle we were given a very interesting suggestion to incorporate a recommendation system for people

who may not be very well acquainted with washing cycles and how to choose one that fits their laundry.

3rd Cycle

During this cycle, we first decided on incorporating a recommendation system in the application. After discussing with our friends and siblings, we concluded that some users may not be fully acquainted with washing cycle and thus being baffled with choosing the correct cycle. Additionally, a very interesting thought we had was to further leverage sound stimuli to account for people who live in residences where the washing machines are shared and located in one room. The problem that we detected was that if a visually impaired person entered such a room, it would be difficult to exactly pinpoint where each washing machine is without trying to feel for it. To solve this, we decided to have some different music playing to signal whether the washing machine is in use or not and an alarming sound for when the cycle has finished. Lastly, we incorporated computer vision to recognize when a subject is about to use the washing machine so that the music can stop, and either start the application or stop the alarm if a cycle has ended. This makes the experience handsfree and results in a truly transparent interface.

Face Detection

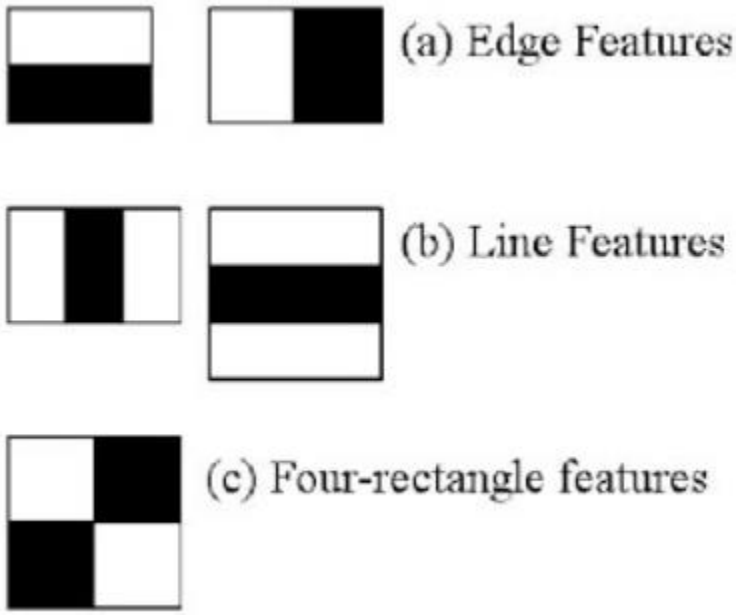
When implementing the face detection part described above, we concluded that this is not a classification task (e.g., infer if someone is in the frame) but rather an object detection task since we need to work with multiple frames and continuous movement of the subject. Moreover, we did not want to simply detect if a subject is within the frame since that would cause false positives when a subject is within the frame but is not actively engaging with the washing machine but rather doing something else in that room. To combat this, we wanted our criteria to be the following:

1. Subject is close to the camera

2. Subject is facing the camera directly

3. The above criteria are met for a continuous number of frames.

To implement this, we decided on using two pretrained Haar feature-based cascade classifiers from the library opencv. These classifiers are pretrained on large amounts of data where they extract Haar-like features using the following kernels and all their possible sizes and positions within the image.



We incorporated two classifiers, one used to detect a facial figure (e.g., a head) and one used to detect the eyes. We define a ‘person’ as 2 eyes and one facial figure and we keep track of how many continuous frames do we observe this. We set a threshold of 30 frames, therefore if these conditions are met for 30 sequential frames, we open the interface and greet the subject or stop the alarm and terminate the interface.

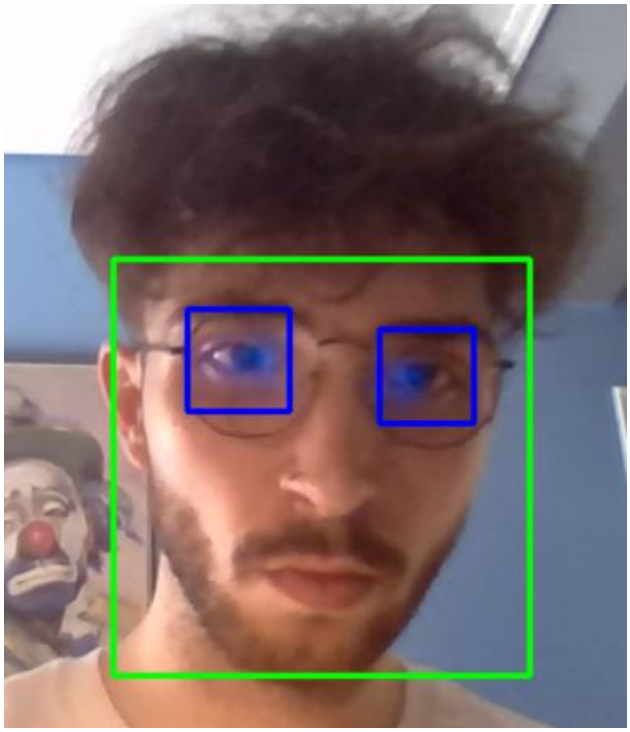
```
def run(self):
    while not self.detection_event.is_set():
        ret, frame = self.cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = self.face_cascade.detectMultiScale(gray, 1.3, 5)
        for (fx, fy, fw, fh) in faces:
            eyes = self.eye_cascade.detectMultiScale(gray[fy:fy + fw, fx:fx + fw], 1.3, 5)
        if len(faces) >= 1 and len(eyes) >= 2:
            self.detected_counter += 1
            if self.detected_counter >= 30:
                self.detection_signal.emit()
                self.detected_counter = 0
                break
```

Using these criteria, we found that we can make our classifier more robust by eliminating most false positives (detecting a subject when not engaging with the washing machine).

Here are some frames indicative of how and when a face is detected:

1) The subject is close to the camera and not looking at it directly → Not Detected.

2) The subject is close to the camera and looking directly at it → Detected.



Recommendation System

For the recommendation system we wanted to keep it as simple as possible. After researching what are the determinants of choosing a washing cycle, we narrowed them down to just 4 categories. Subject's available time (< 1h, < 2h, > 3h), prominent color of laundry (light, dark, mixed), prominent garment type (sensitive, plain, heavy), size of laundry (small, medium, large). We will use these 4 questions are our predictors. At first, we were thinking of constructing a decision tree to implement the recommender, however since we only have 4 features with 3 possible answers each, we decided to implement a rule-based approach. Thus, we created a prolog knowledge base by defining rules for each possible set of answers and an according recommendation. We used python's pyswip library, a python prolog wrapper, to establish the knowledge base and query it to get the recommendations. Furthermore, as previously mentioned one of our aims for this interface was to make it self-explanatory and

intuitive to interact with. To achieve this, the voice assistant is designed to have explanatory prompts for each question and possible answer, essentially helping the subject make its choice when prompted. Lastly the subject is informed of this capability as well as the more general rules of how to interact and use the interface when greeted by the assistant at the beginning.

```
def create_rules(self):
    """
    Insert rules into knowledge base
    """
    # If time available is less than an hour
    self.knowledge_base.assertz("recommend(0, _, sensitive, small, '30_30_sensitive')")
    self.knowledge_base.assertz("recommend(0, _, sensitive, medium, '30_30_sensitive')")
    self.knowledge_base.assertz("recommend(0, _, sensitive, large, '45_30_sensitive')")
    self.knowledge_base.assertz("recommend(0, _, normal, small, '30_40_normal')")
    self.knowledge_base.assertz("recommend(0, _, normal, medium, '30_40_normal')")
    self.knowledge_base.assertz("recommend(0, _, normal, large, '45_40_normal')")
    self.knowledge_base.assertz("recommend(0, _, heavy, small, '30_60_heavy')")
    self.knowledge_base.assertz("recommend(0, _, heavy, medium, '30_60_heavy')")
    self.knowledge_base.assertz("recommend(0, _, heavy, large, '45_60_heavy')")
    # If time available is less than two hours
    self.knowledge_base.assertz("recommend(1, _, sensitive, small, '60_30_sensitive')")
    self.knowledge_base.assertz("recommend(1, _, sensitive, medium, '60_30_sensitive')")
    self.knowledge_base.assertz("recommend(1, _, sensitive, large, '90_30_sensitive')")
    self.knowledge_base.assertz("recommend(1, _, normal, small, '60_40_normal')")
    self.knowledge_base.assertz("recommend(1, _, normal, medium, '60_40_normal')")
    self.knowledge_base.assertz("recommend(1, _, normal, large, '90_40_normal')")
    self.knowledge_base.assertz("recommend(1, _, heavy, small, '60_60_heavy')")
    self.knowledge_base.assertz("recommend(1, _, heavy, medium, '60_60_heavy')")
    self.knowledge_base.assertz("recommend(1, _, heavy, large, '90_60_heavy')")
    # If time available is three or more hours
    self.knowledge_base.assertz("recommend(2, _, sensitive, small, '120_30_sensitive')")
    self.knowledge_base.assertz("recommend(2, _, sensitive, medium, '120_30_sensitive')")
    self.knowledge_base.assertz("recommend(2, _, sensitive, large, '180_30_sensitive')")
    self.knowledge_base.assertz("recommend(2, _, normal, small, '120_40_normal')")
    self.knowledge_base.assertz("recommend(2, _, normal, medium, '120_40_normal')")
    self.knowledge_base.assertz("recommend(2, _, normal, large, '180_40_normal')")
    self.knowledge_base.assertz("recommend(2, _, heavy, small, '120_60_heavy')")
    self.knowledge_base.assertz("recommend(2, _, heavy, medium, '120_60_heavy')")
    self.knowledge_base.assertz("recommend(2, _, heavy, large, '180_60_heavy')")
```

Function to query the knowledge base and get a recommendation based on the subject's choices.

```
"""
Returns a recommendation based on the rules of the knowledge base, when presented with an appropriate query
"""
for recommendation in self.knowledge_base.query(query):
    return recommendation['Recommendation'].split('_')
```

Sound Stimuli

We opted for some jazz music for when the machine is idle (i.e. ready to be used), some more jazz, but a different song, for when the cycle is taking place and a smooth sound resembling an alarm due to its periodicity to signal that the cycle has finished and the laundry is ready to be taken out.

Evaluation

The evaluation of this cycle was centered around the recommendation system being tested both verbally and physically.

1) Performance Measurement

   We again assembled 5 subjects that were given a goal of using the recommendation feature of the application and start a washing cycle. No further instructions were given to simulate novice users that have never interacted with the interface.

Using the voice assistant:

Subject ID	0	1	2	3	4
Times an action had to be repeated	1	0	0	0	0
Times subject did not act within the designated timeframe	0	0	0	0	0
Time to complete the process	1'44"	1'35"	1'33"	1'29"	1'31"

Using the buttons:

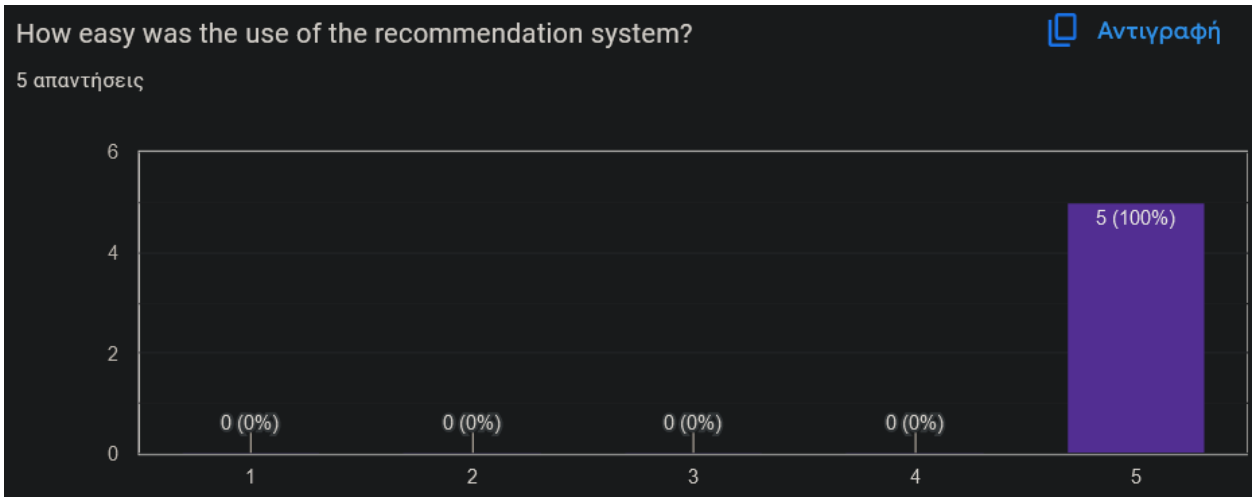
Subject ID	Time
0	12s
1	13s
2	14s
3	15s
4	14s

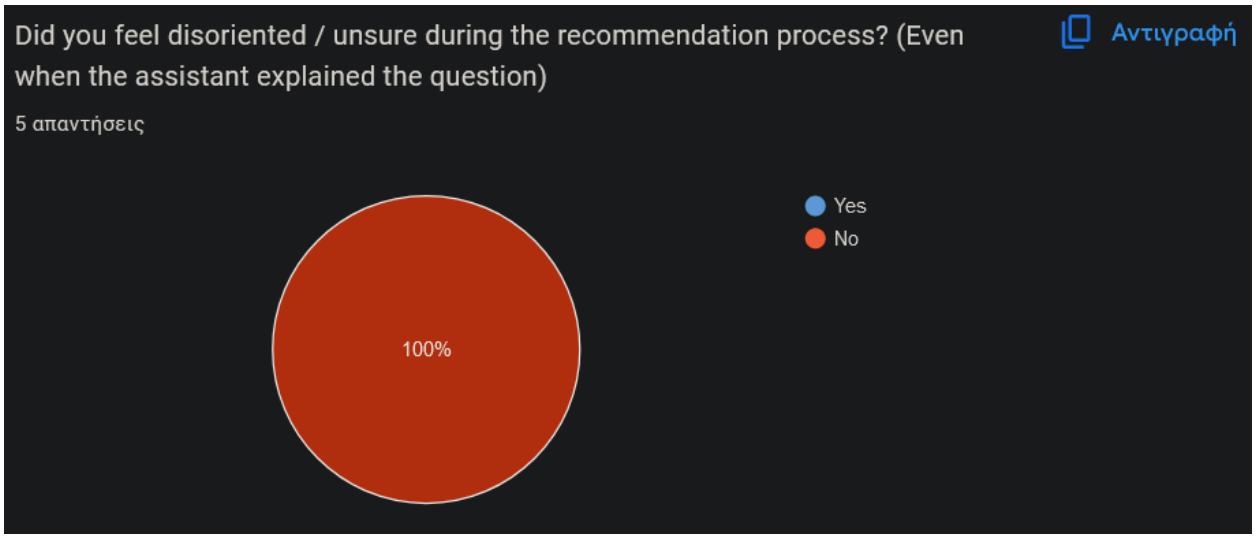
Average Time: 13,6s

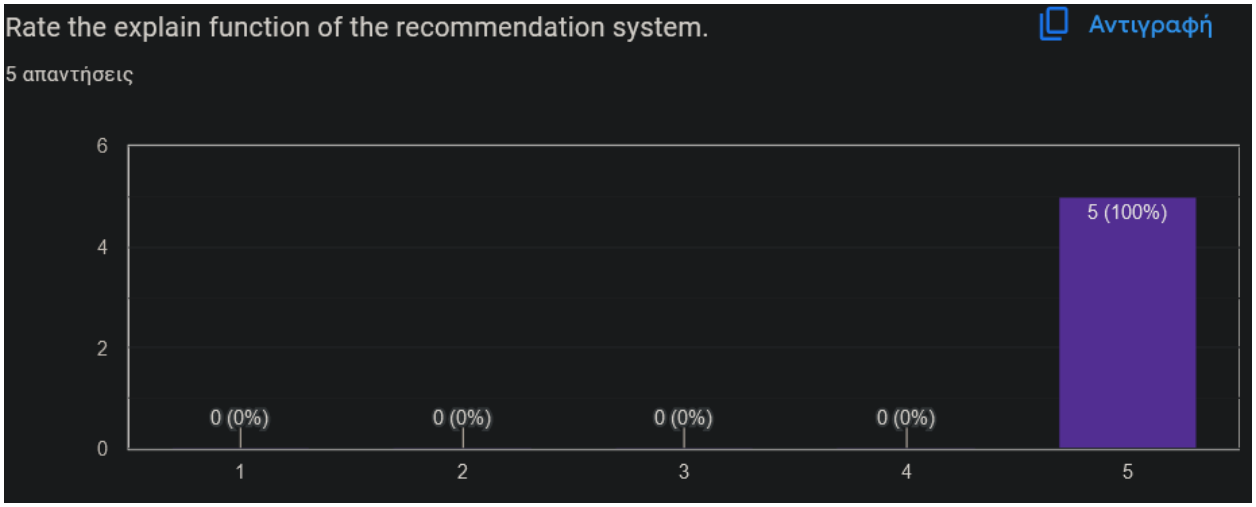
We were content with the results. Again, the times are similar because the process is catered to be driven by the system not the user. The only major time differences would occur if a subject would go back to change a previous answer, which was not the case here.

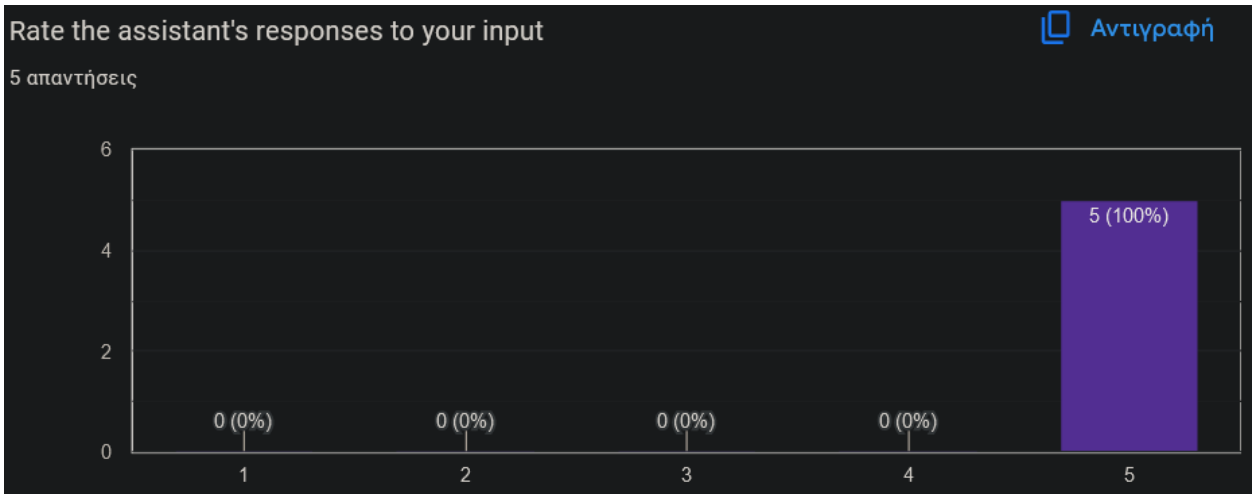
2) Following the experiment, the subjects were given a questionnaire to answer based on their experience.

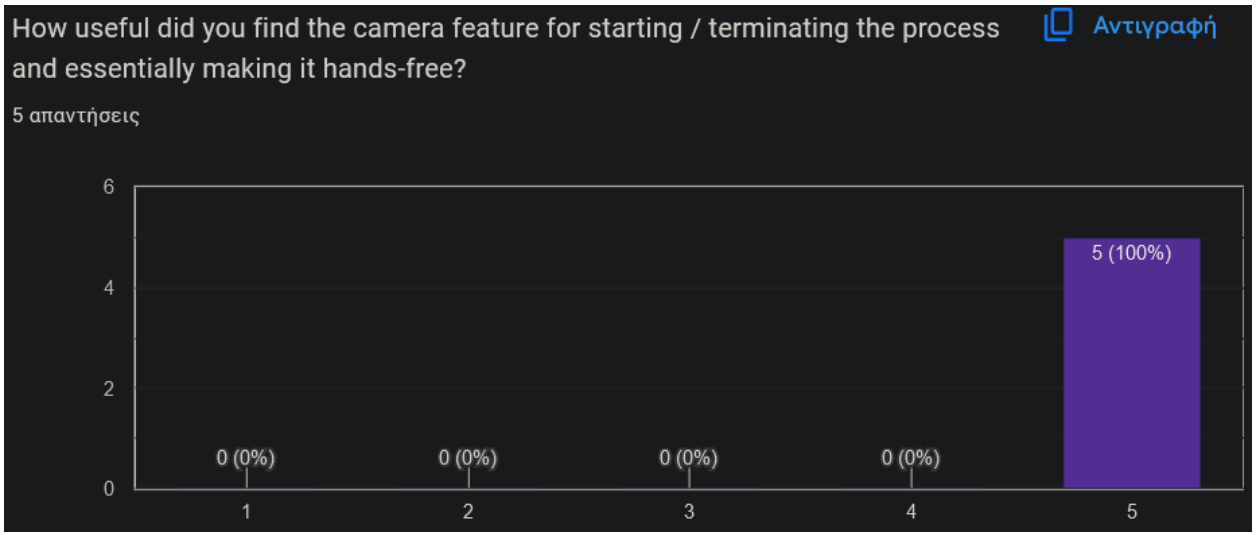
   Here are the results:

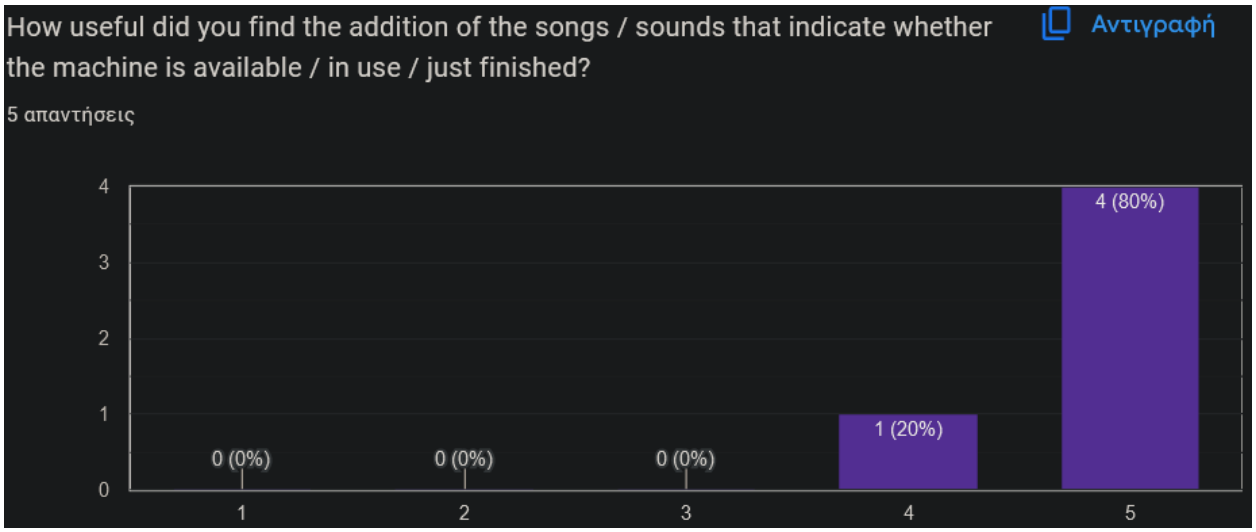


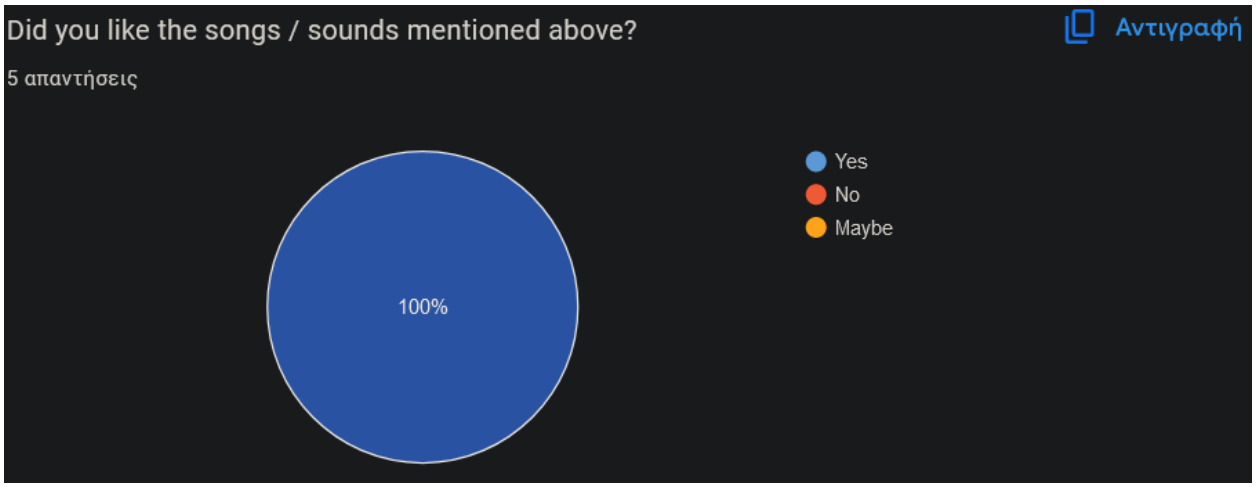

Did you feel disoriented / unsure during the recommendation process? (Even when the assistant explained the question)
Αντιγραφή
5 απαντήσεις
Yes
No
100%

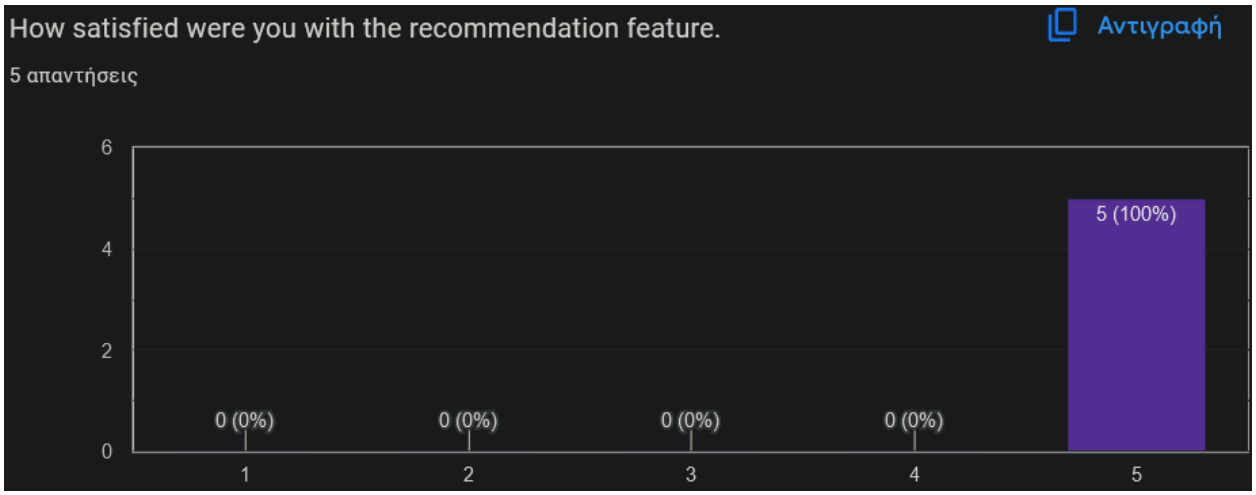

Rate the explain function of the recommendation system.
Αντιγραφή
5 απαντήσεις
6
4
2
0
0 (0%)
0 (0%)
0 (0%)
0 (0%)
5 (100%)
1
2
3
4
5

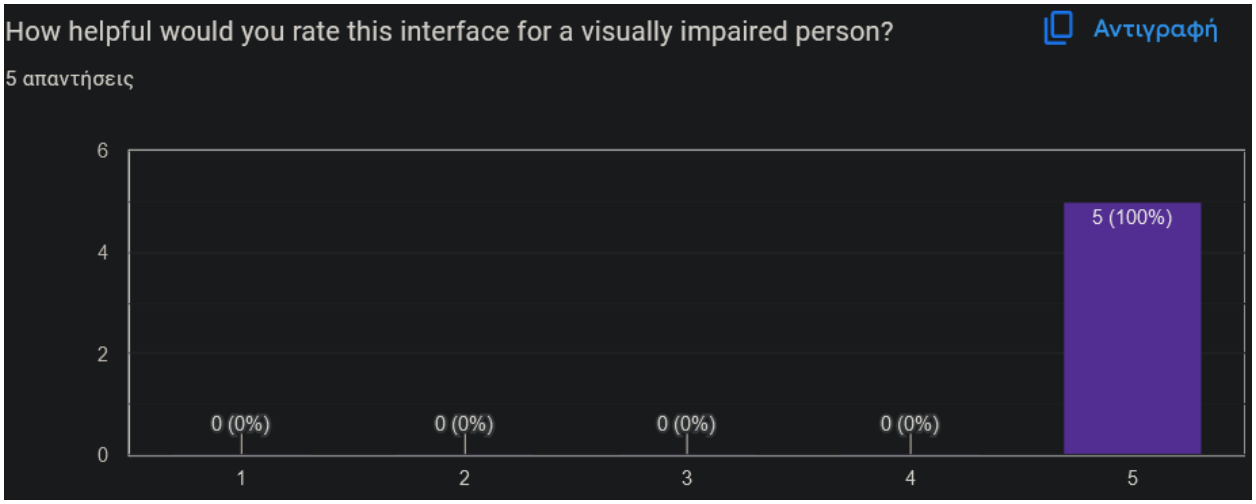

Rate the assistant's responses to your input
Αντιγραφή
5 απαντήσεις
6
4
2
0
0 (0%)
0 (0%)
0 (0%)
0 (0%)
5 (100%)
1
2
3
4
5

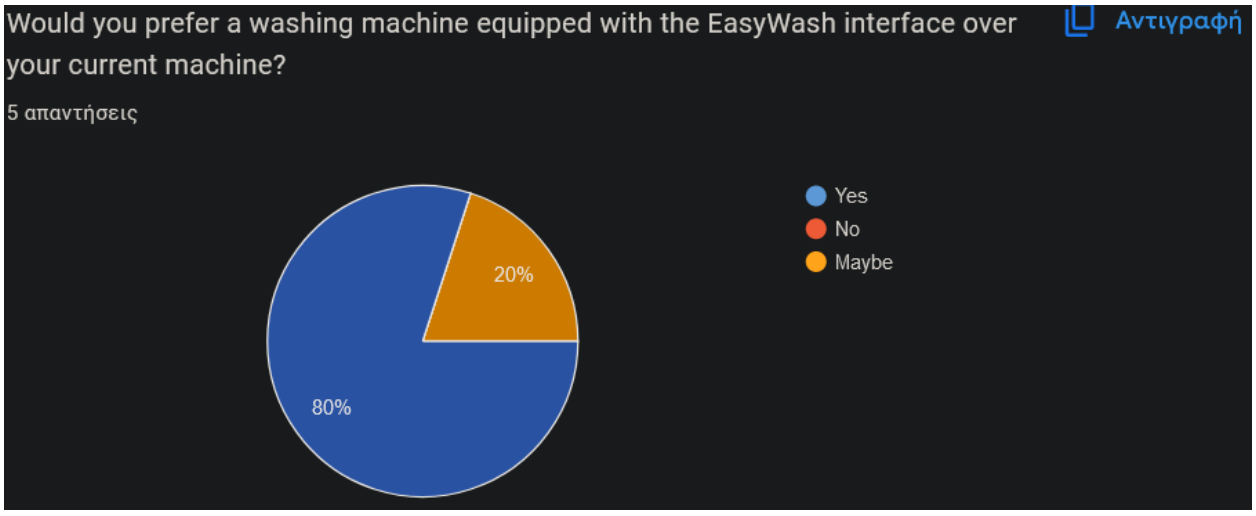

How useful did you find the camera feature for starting / terminating the process and essentially making it hands-free?
Αντιγραφή
5 απαντήσεις
6
4
2
0
0 (0%)
0 (0%)
0 (0%)
0 (0%)
5 (100%)
1
2
3
4
5


How useful did you find the addition of the songs / sounds that indicate whether the machine is available / in use / just finished?
Αντιγραφή
5 απαντήσεις
4
3
2
1
0
0 (0%)
0 (0%)
0 (0%)
1 (20%)
4 (80%)
1
2
3
4
5


Did you like the songs / sounds mentioned above?
Αντιγραφή
5 απαντήσεις
100%
Yes
No
Maybe







We were very content with the results. As illustrated above all 5 subjects found the interface very satisfactory and easy to use and would choose it over their current one.

3) Cognitive Walkthrough

We performed a cognitive walkthrough of the application focusing on the recommendation system. The scenario was to pretend to be a novice user with a goal of utilizing the recommender to start a washing cycle. Similarly with the previous cycles we deduced that the interface is transparent and supports investigative learning as it feels very intuitive to use, thanks to the consistent guidance provided by the assistant and the ‘menu’ metaphor eliminating any memory load and minimizing the necessary actions to complete the task. One minor issue we encountered was that the music was a slightly too loud for our liking and thus we lower its volume. All in all, we were very content.

4) Speaking Agents

As a final evaluation method we gave a new subject, who has never interacted with the interface during its development, the goal of starting a washing cycle. The main difference from the other 5 subjects, except from not having interacted with the interface, was that we introduced him to main aim of this interface and the features provided without any more details on how these features work. Afterwards, the subject was instructed to utilize both the recommendation and own cycle features using the assistant while having its eyes closed. Apart from simulating blindness, we wanted to truly remove all visual stimuli that might hint the functionality of each step and see how supportive of investigative learning our interface truly is. After conducting the experiments the subject was asked the following questions:

1) Q: Rate the interface based on your experience from 1 to 5, with 1 being very poor and 5 being very good.

   A: 5

2) Q: Rate the voice assistant's guidance from 1 to 5, with 1 being very poor and 5 being very good.

   A: 5

3) Q: Rate the voice assistant's guidance from 1 to 5, with 1 being very poor and 5 being very good.

   A: 5

4) Q: Rate the voice assistant's responses from 1 to 5, with 1 being very poor and 5 being very good.

   A: 5

5) Q: Rate the voice assistant's interpretation of your inputs from 1 to 5, with 1 being very poor and 5 being very good.

   A: 5

6) Q: Rate the voice assistant's guidance from 1 to 5, with 1 being very poor and 5 being very good.

   A: 5

7) Q: Did you feel disoriented at any point during the experiments?

   A: No

8) Q: Would you prefer this interface over the one your current machine has?

   A: Yes

9) Q: How easy was the completion of your task (start a cycle) from 1 to 5, with 1 being very difficult and 5 being very easy.

   A: 4

Conclusion

We were very content with the final version of our interface. The spiral model development process was very helpful as it allowed us to gather lots of feedback after each cycle. Therefore, each cycle's evolutionary prototypes were guided from that feedback, allowing us to design and add features while always having the user as the center piece. We utilized all the knowledge we gathered from the course and we are hoping that the interface truly conforms with the usability standards we set from the beginning.