Second week implementation

João Passos 1210646
Pedro Vicente 1180558

EARS requirements organization:

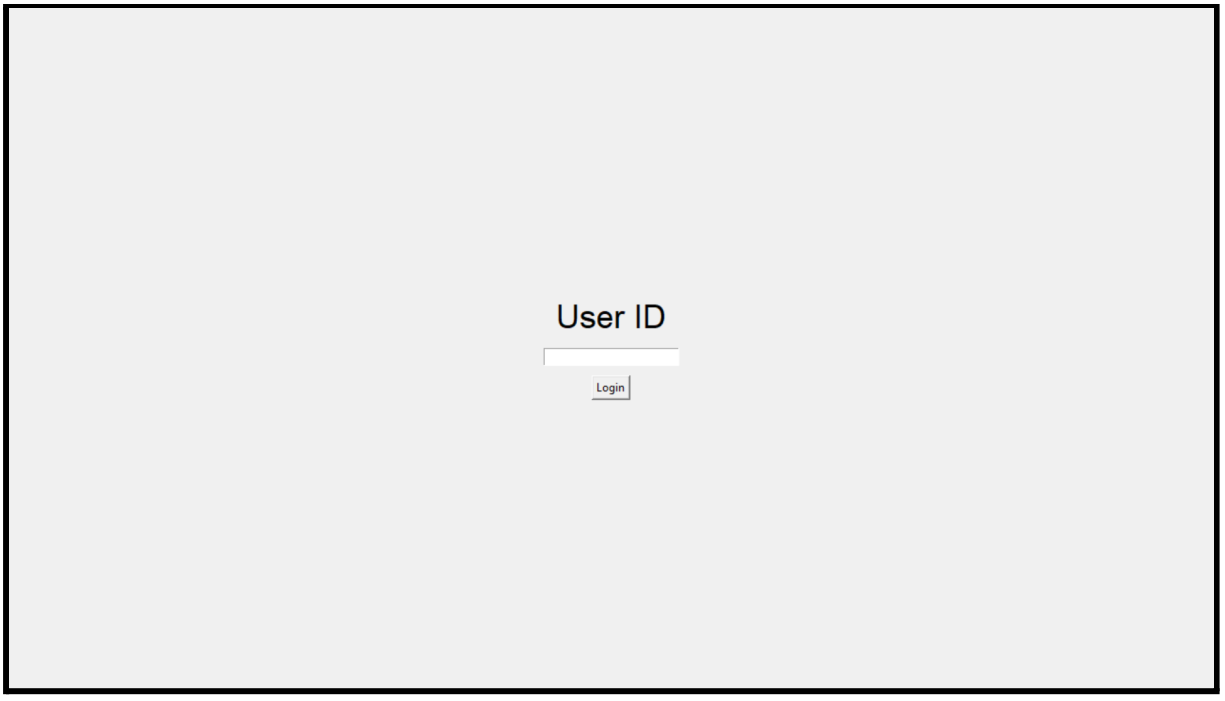
The name of the User Interface subsystem was changed to Capsule selection subsystem
The User Interface subsystem now involves the Authentication subsystem and the Capsule selection subsystem.
The requirements between these two subsystems were better distributed.
The Authentication subsystem conducts the user from the Login page to the Capsule selection page (Login process).
The Capsule selection subsystem conducts the user from the Capsule selection page till he logs out (Capsule selection process mainly).

Updated organization:

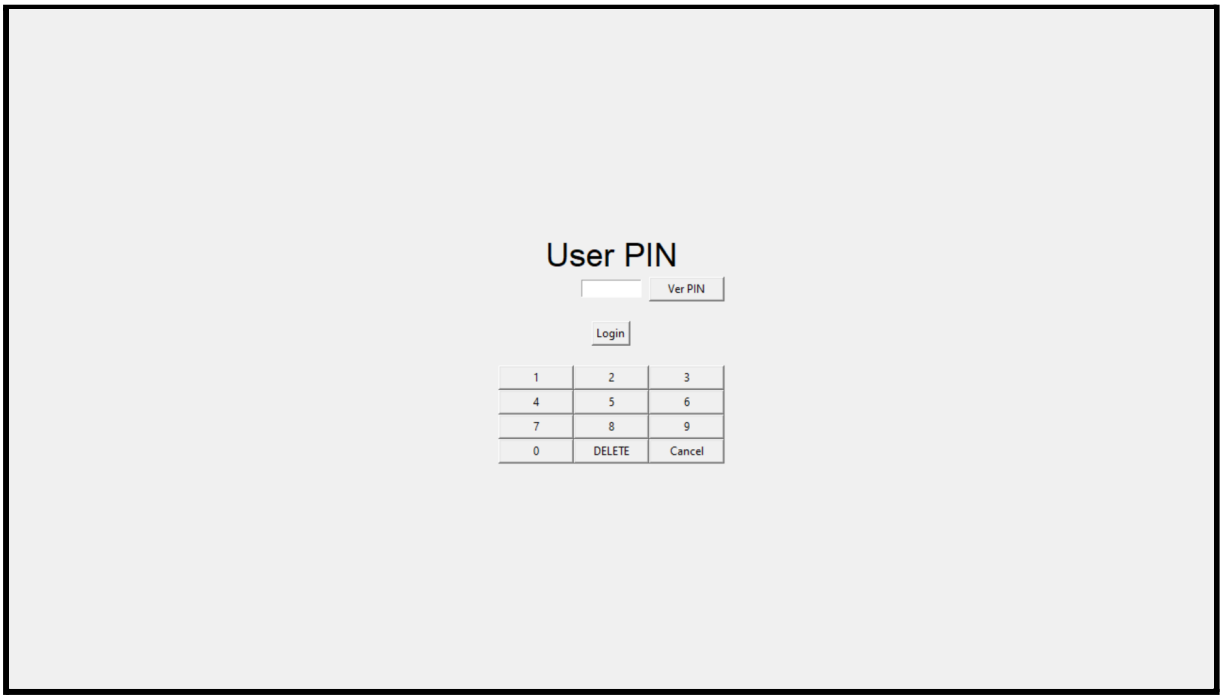
- User Interface:
  Authentication subsystem
  Capsule selection subsystem
- Dispensing subsystem
- Storage subsystem

User Interface:

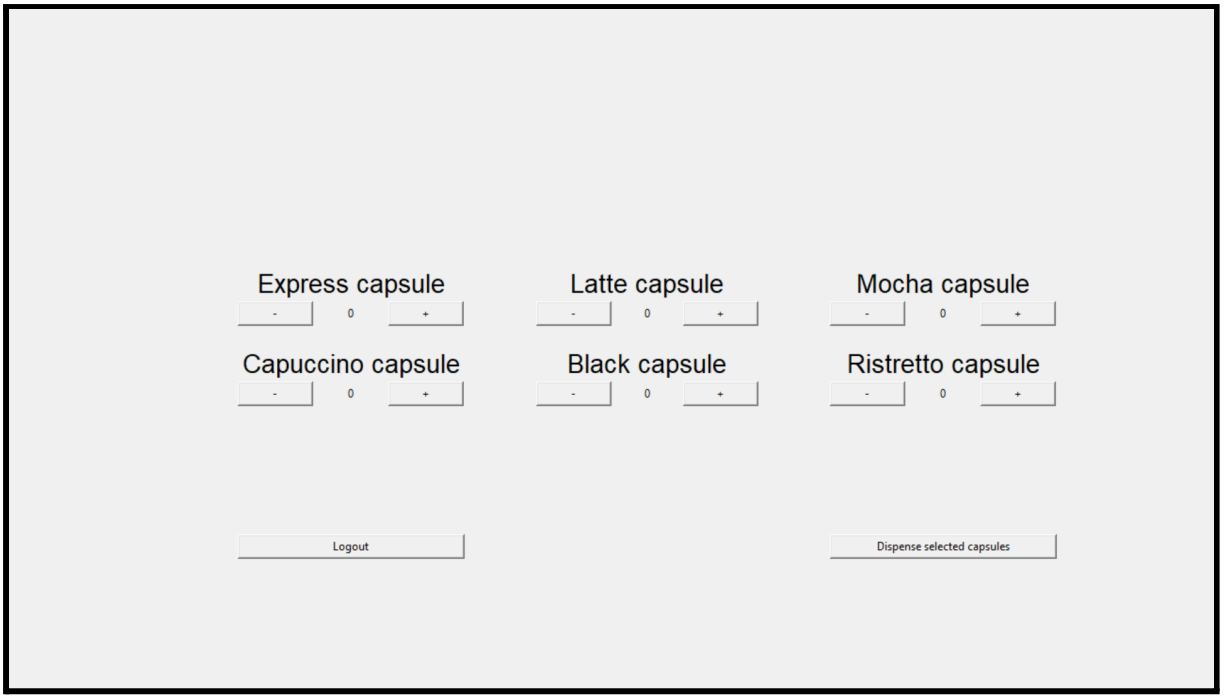
The User Interface was developed in Python with the use of the tkinter library
The Interface has a Login page (as show bellow).



For now the user must insert his ID and click the button “Login”.
After clicking “Login” the page were the user will insert his PIN will be loaded (image bellow)



The PIN is hidden by default but the press of the “Ver PIN” button shows the digits inserted so far and the second press will hide them again. The available numbers go from 0 to 9, the user can return to the Login page through the button “Cancel” and can delete the last inserted digit through the button “DELETE”.
Once the “Login” button is clicked the Capsule selection page will be loaded (as shown below)



From this menu the user can select the number of capsules he wants from the displayed drinks.
The “Logout” button will load the Login page and forget all the selected capsules selected (if any).
The “Dispense selected capsules” button will first verify if any capsule is selected (if there’s no capsule selected the page will display an error message about this problem and doesn’t load the next page) and then the Dispensing capsules page will be loaded (figure bellow)

This page displays the capsules selected by the user.

Note: The other page was deleted so the user can´t select any more capsules or interact with the system while the selected capsules are being dispensed.
The progress bar in the figure is a simulation (not actual progress), but can be further updated through real progress (motors position, etc).
After the progress bar is at 100% the page waits 2 seconds and returns to the Login page resetting the system.

MQTT server

The MQTT server is hosted in the main C program in a different thread or process. The MQTT clients are: the main C program, the Python interface (and other Smart Coffee Capsules Dispenser). All the mentioned clients will be publishers and subscribers since all need to receive and send information between each other.

- The Python Interface during his execution will read information inserted from the user and send it to the main program to verify it or store it. If the main program has to verify the information it will also need to publish a verification message so that the Python Interface knows how to proceed. The use of the MQTT server isn't the best way to correctly execute data transfer between C and Python programs but will be the method to use since the understanding and implementation of the bridge between C and Python code would be very time-consuming.
- Since all the Smart Coffee Capsules Dispenser machines would know how many capsules would be in all the other machines, periodically the main program would be responsible to publish his coffee capsules information to the other machines.

(The MQTT communication between the Python and the C program wasn't tested yet)

Concurrency and Real Time scheduling aspects:

The Smart Coffee Capsule Dispenser is a system whose behavior is very sequential and controlled. Therefore, the scheduling aspects of this particular system are not very complex and don't need that much attention.

From the first moment programs are started, the main C program is responsible to read the RFID tag (when detected), to identify the tag (when requested), to read the infrared sensors (when requested) and to activate the motors (when requested). The only activity the main C program has to be permanently verifying is the MQTT server which can be run in a different process or in a thread. This server is responsible to verify and store information from the Python Interface to the local database and to store information about the coffee capsules available in other machines.