Software Design Document (SDD) with Security by Design (SBD)

Project Information

Project Name	**Team ID**	**Mentor(s)**
Pi Vision	31	Niek Aukes & Cristi Angheluş

Introduction

Our project is a Hands-Free Gesture Recognition system designed to control a computer. It uses a Raspberry Pi 5 as the main processor and a Logitech C270 HD Webcam to capture hand gestures, which are mapped to keystrokes or mouse movements. The system also features a login function that identifies users and saves their personalized gestures as keystroke mappings.

Revised Requirements

As for the various requirements of our project, we did not revisit any of them, neither functional nor non-functional nor security requirements. However, after reviewing the RAD, we decided to add other functional requirements:

- The system shall send the commands to the computer as if it were a keyboard or a mouse
  - ***Reason***: in the initial Requirement document, this information was a bit vague and therefore we have concluded it would be better to make a more clear specification about how the final implementation should be working
- The system should allow per-application gesture bindings (e.g., different gestures in Chrome vs. a game).
  - ***Reason***: it may be possible that users may have different preferences of input when (for example) playing videogames or browsing on the internet. It is therefore fundamental to offer the user the possibility to modify such inputs.
- The system should allow users to register a new account with unique gesture bindings.
  - ***Reason***: every user should have a personal account to access, and such an account should be bound to a password that the user will need to input in order to access their account with their personal gesture mapping.
- The system should store user-specific gesture mappings in persistent local storage.
  - ***Reason***: Again, every user has a unique/personal gesture mapping. A user's mapping can be different from other users and therefore it is important to save

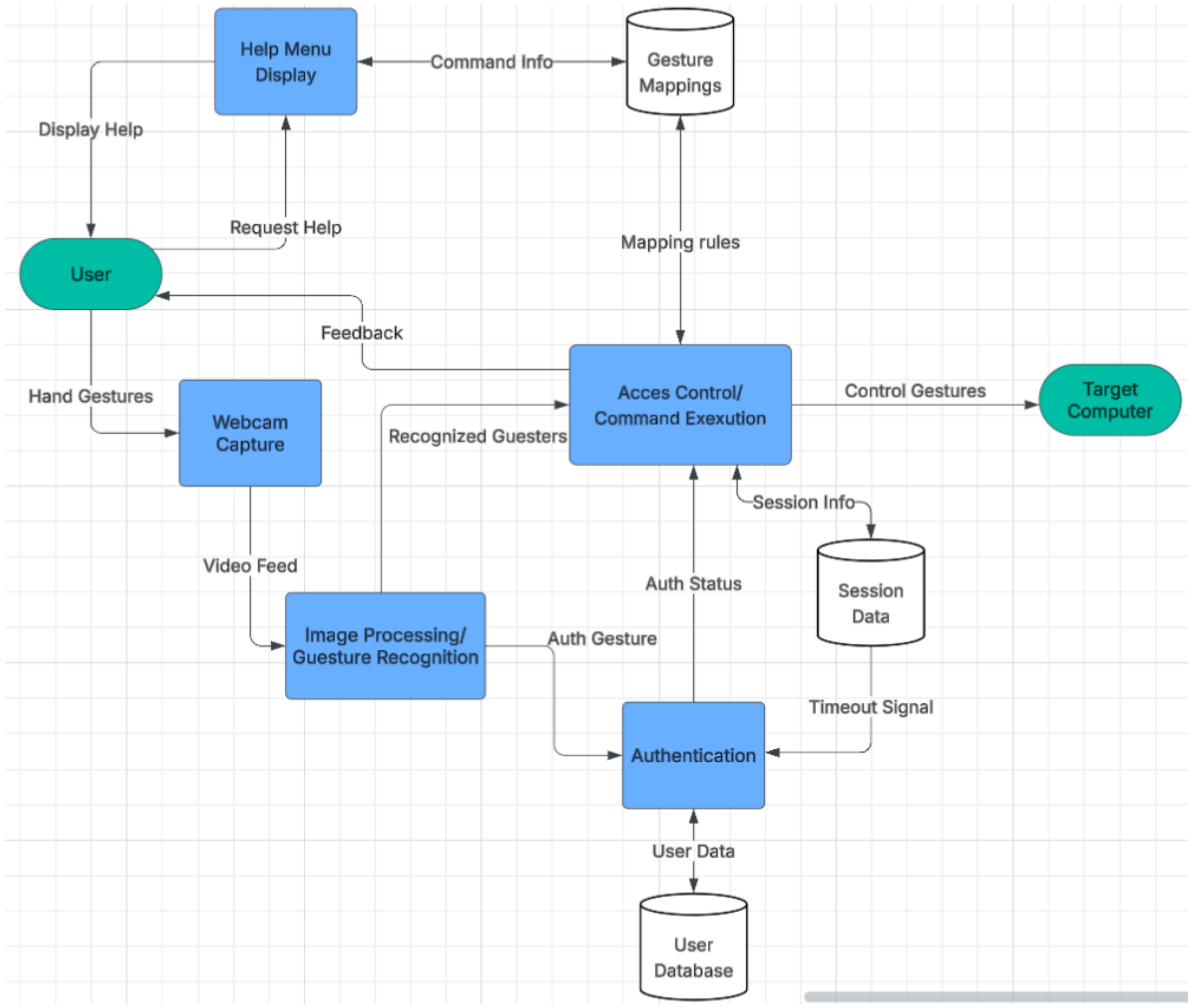
and store a specific user's mapping so that, once they log in, their preferences will be saved.

- The system should allow users to view and test recognized gestures in real time through a simple visual feedback interface
    - ***Reason***: With this, the user has a way of knowing what gesture is recognised. This feedback enhances user experience by adding a layer of convenience.

Architectural Design

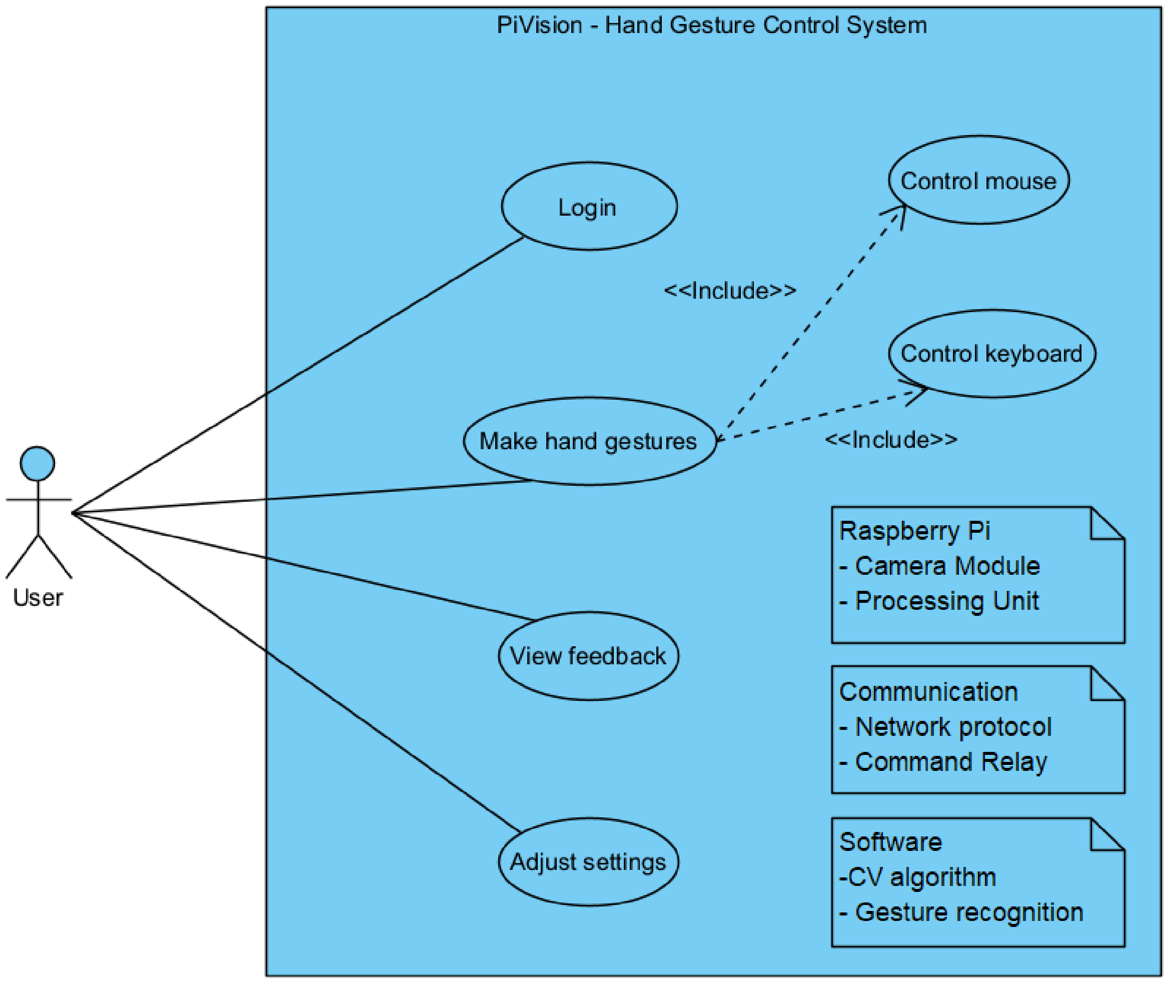
Data Flow Diagram

Below, you can see the way data will flow through our system. The Hand gestures from the user are picked up by the webcam. After a certain gesture is recognized by the system, the system will act accordingly. There is also an Authentication loop, which will handle everything regarding the authentication part of the system. It does this by regularly checking if there is a valid session (properly logged in, user is still active, no timeout signal). Finally, the gestures for controlling the computer system will be sent to the target computer.



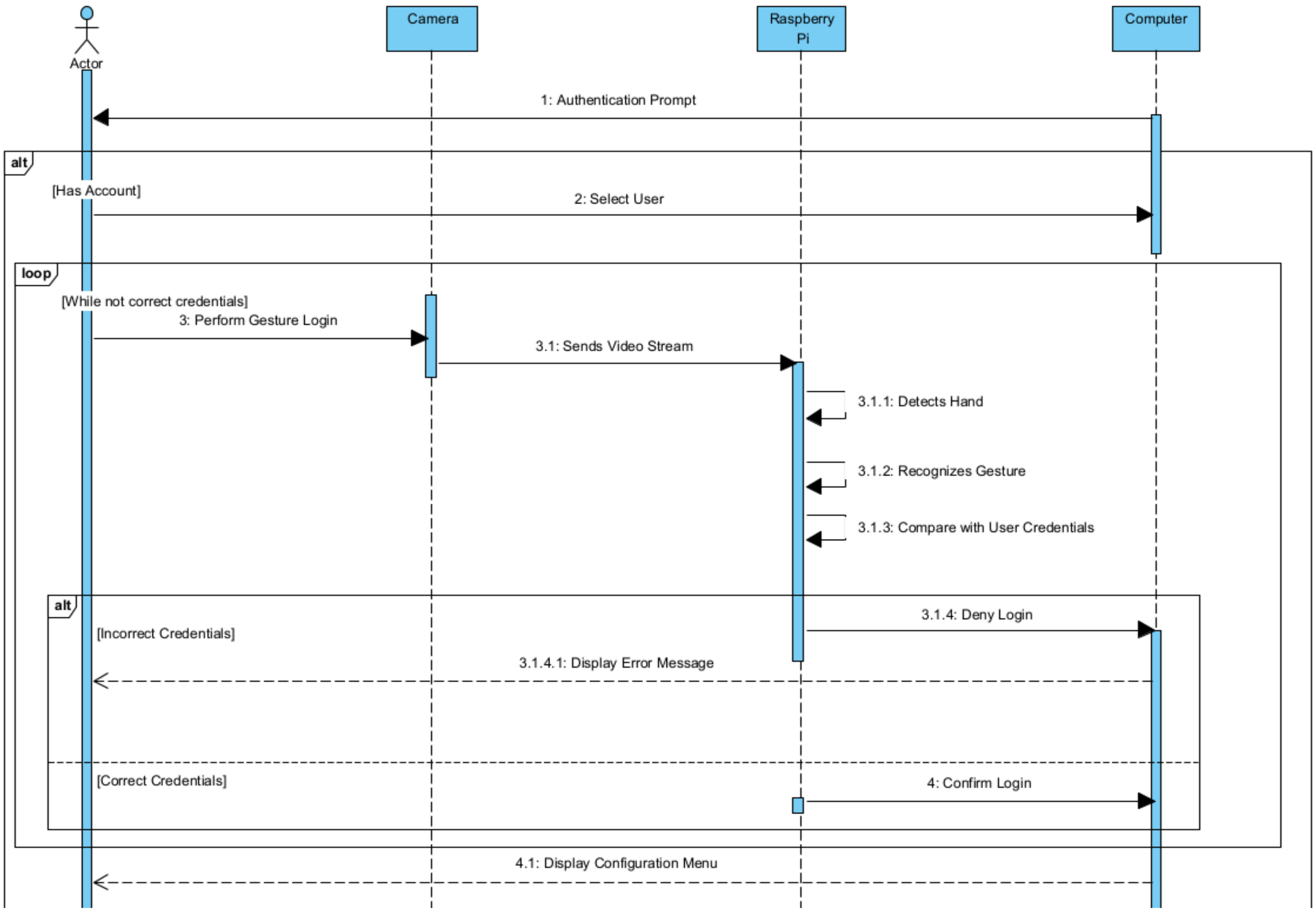
Use Case Diagram

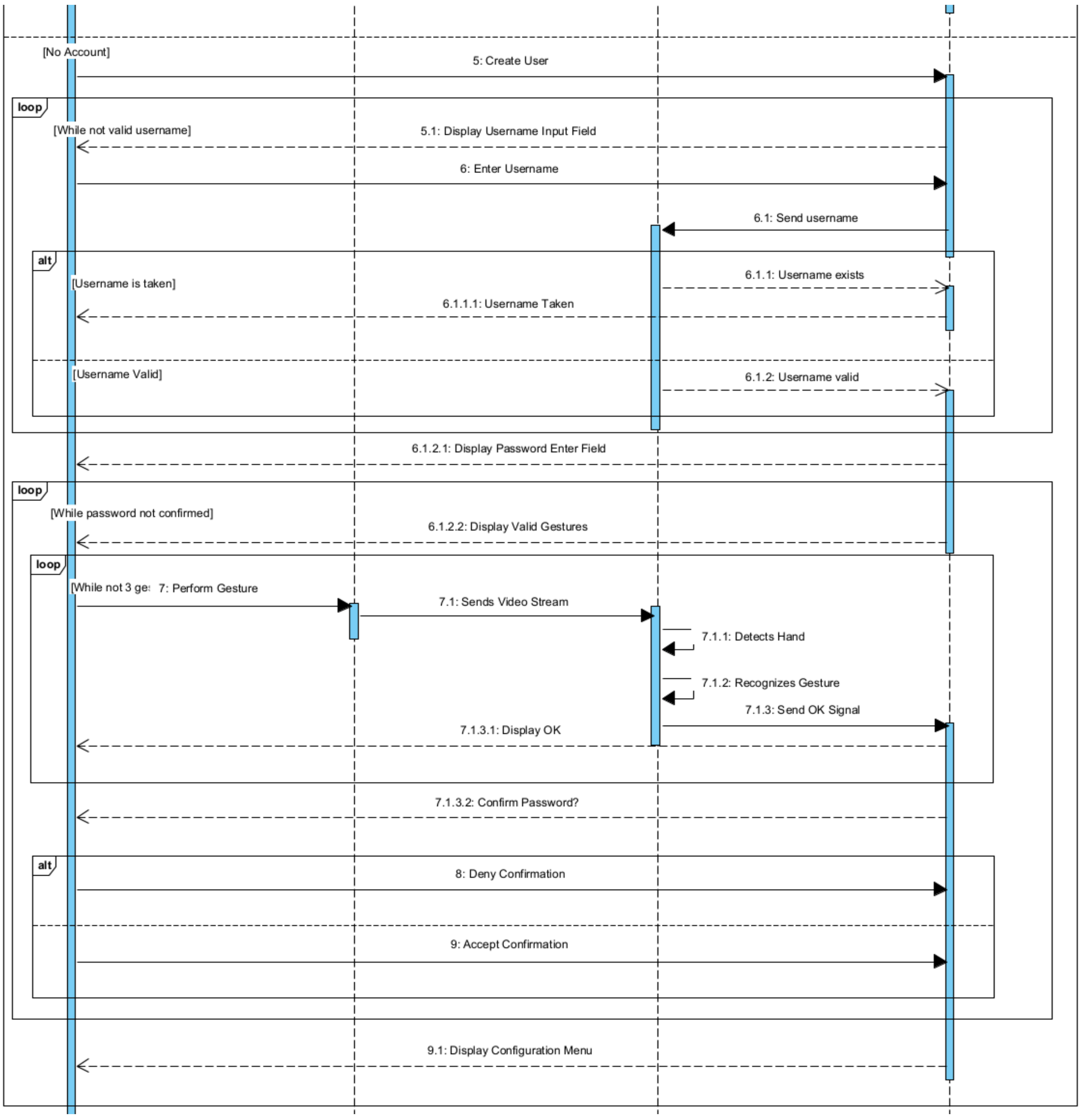
The system has only one actor, the User. First, this user has to authenticate in order to use the system. If that happens, the User can control the mouse, applications, and keyboard with the detection of the hand gestures of the User. The User can also see if the system is accurate and actually detects the correct hand gestures. There is also the option for the User to adjust settings so that certain gestures can be mapped to different actions.

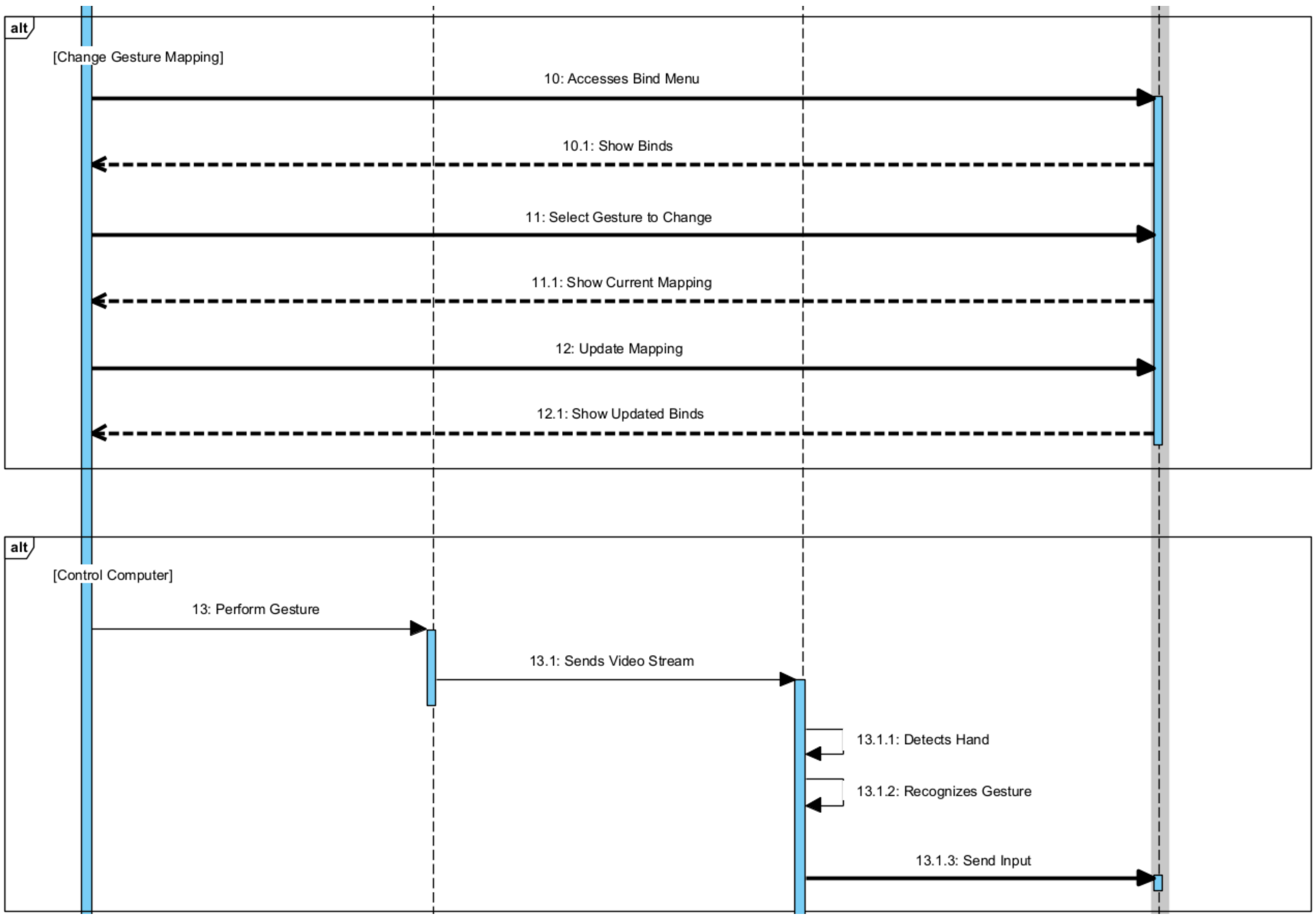


Sequence Diagram

The system uses a sequence diagram to show the order of the actions that a user performs in order to carry out a certain action. In our system, these actions include: logging in, creating the account, modifying the binding of gestures and entering input. To enhance the security aspects of the system, (assuming the user has an existing account), the user would be asked to introduce a password to access their account. If this matches the registered one, the user will have access and will be able to; otherwise, the program will block and the user will later be asked to try again to input the password.




[No Account]
5: Create User
loop
[While not valid username]
5.1: Display Username Input Field
6: Enter Username
6.1: Send username
alt
[Username is taken]
6.1.1: Username exists
6.1.1.1: Username Taken
[Username Valid]
6.1.2: Username valid
6.1.2.1: Display Password Enter Field
loop
[While password not confirmed]
6.1.2.2: Display Valid Gestures
loop
[While not 3 ge:
7: Perform Gesture
7.1: Sends Video Stream
7.1.1: Detects Hand
7.1.2: Recognizes Gesture
7.1.3: Send OK Signal
7.1.3.1: Display OK
7.1.3.2: Confirm Password?
alt
8: Deny Confirmation
9: Accept Confirmation
9.1: Display Configuration Menu



Product User Interface

The project includes a graphical user interface. The user interface will be a pop-up residing on the corner of the display. Its primary functions include: login/sign-up (authentication), help menu, change gesture mapping, and also displaying the translation of his gesture to the corresponding input (keystroke, mouse movement).

Prevention / Mitigation Criteria

Prevention

- Hash with a salt the user login data before storing.
- Disable unused USB ports to prevent malicious devices from being plugged in.
- Prevent uploading of other code to the Raspberry Pi once the product is finished, to prevent anything else from running.
- Require re-authentication before allowing gesture reconfiguration.
- Timeout for login if too many incorrect attempts

Mitigation

- If someone gains access to the Pi, they can only send allowed commands to the device.
- Gesture mappings can not be changed without authentication
- If the Raspberry Pi filesystem is accessed, sensitive data remains protected through encryption, limiting what an attacker can read or extract.

As much as we can prevent any possible problem or predict any possible malicious attempt, it is also necessary to say that it is possible that some problems may arise, and therefore, our teams have predicted them and prepared some mitigation strategies which may help to reduce the possible consequences. For example, someone got access to the Raspberry Pi? Not a problem! Our program does only allow for a specific set of instructions to be executed. In other words, there is a set of instructions, in our case gestures, and only that specific set can be executed. What if the gestures want to be mapped to other commands? Mapping is only allowed when a user is identified. Therefore, since we will avoid data breaching, it should not be a problem. Moreover, since we are hashing stored data, any filesystem accessed from the Raspberry Pi will not have major impact on elements such as user privacy

Cost Involved

Time costs

During the 5 sprints of 2 weeks each, the workload will be distributed equally across all 6 members. With an estimated average of 7 hours of work towards the project per person per week. This would accumulate to an average of 42 man-hours per week. And 420 man-hours in total (70 hours per person).

Realistically, about 80% of this time would be spent developing the product, 10% on testing the product and 10% on preparing the sprint reviews, presentations and rounds of feedback.

Financial costs

For this project, we chose the Raspberry Pi 5 with 4GB of RAM, together with the Logitech C270 HD Webcam.
These two products combined end up costing €91,79, which, distributed over the 6 people in our team, would be €15,30 each.

Our product will use all resources locally, meaning that we have no such costs as cloud fees or recurring payments.

Trade-offs

Unlike last year's Module 5, we had to get our own Raspberry Pi; this meant that we had to choose which Raspberry Pi to get.
There is the obvious “performance vs cost” debate. We figured that since the Raspberry Pi is our main component, and we could sell it after the project if we wanted to, that we would get the Pi 5 instead of the Pi 4. The performance boost would be significant enough for our goals.

Another trade-off we already made is that we chose not to implement multi-factor authentication. We won’t do this because this would be too time consuming. Our product is supposed to be an I/O device, and you don't use MFA for a keyboard or mouse. This would sacrifice a bit of the security aspect, but would enhance user experience significantly.

Conclusion

To summarize, the document focuses on the design choices the team made towards the realization of the project, together with some minor adjustments.

After discussing our requirements (Functional, Not Functional and Security) we concluded that we needed to revise some of our Functional Requirements as we though **.**

The team has worked towards the realization of the Architectural Design of the project, clarifying and specifying how the different components will work (Data Flow Diagram), the sequence of actions to be completed to obtain a certain outcome from the system (Sequence Diagram) and visualize the functional requirements of our system (Use Case Diagram). Building up a consistent and clear Architectural Design will serve the group in the near future not only to gain a clearer idea of the project, but also as a point of reference for two main reasons:

- focus solely and exclusively on the implementation of the fundamental functions of the project, thus avoiding going off track;
- clarify any ambiguities or more complicated structural elements, thereby facilitating implementation.

The next phase for our project would be the actual implementation, where we begin to familiarize ourselves with the libraries, start programming and try all our best to bring what now only appears as an idea into something tangible. Nevertheless, this might generate a possible challenge and concrete risk that can play a crucial role in the final outcome of the project:

- We have talked about how we will use the Google Library and will use its trained model in order to make our project work. However, it is also true that it is the first time we will use such a library, thus we have not yet an exact idea on how and if it will work as expected.
  - The risk therefore will be for us to train our own model and train it in such a way it will respect our project scope and what we want to implement. This is

of course not optimal as it takes time from the actual implementation of the system.

References

Gesture recognition task guide. (n.d.). Google AI For Developers. https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer

Hand landmarks detection guide. (n.d.). Google AI For Developers. https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

HaGRID Sample 30K 384p. (2022, 8 September). Kaggle. https://www.kaggle.com/datasets/innominate817/hagrid-sample-30k-384p

Hand Recognition and Finger Identification with Raspberry Pi and OpenCV. (n.d.). Core Electronics. https://core-electronics.com.au/guides/hand-identification-raspberry-pi/

Usage of AI Tools

During this sprint, our team used OpenAI's ChatGPT to brainstorm ideas and get suggestions for certain diagrams. Also, to refresh our knowledge of some diagrams' syntax.