Holy Spirit University of Kaslik

School of Engineering

**Department of Computer, Telecommunications, and
Electrical and Electronics Engineering**

Human Robot Interaction – GIN456

Final Project Report

Fall 2025 – 202510

Year 2024/2025

Presented to:

Br. Elie SAAD

Presented by:

Asif ALAM, 202201934

Antonio HADDAD, 202200238

Karl ROMANOS ABOU JAOUDE, 202200876

Elias-Charbel SALAMEH, 202201047

Wednesday, December 18th, 2024

List of Acronyms

SFTP: Secured File Transfer Protocol

UC: Use Case

List of Figures

List of tables

Acknowledgment

We would like to thank Br. Elie SAAD for his efforts and commitment to give this course. His knowledge and devotion to help us learn were essential to the completion of this project.

In addition, we would like to thank computer engineering student Yuri YAACOUB for his assistance in various aspects of the projects.

Abstract

Pepper, the social robot developed by SoftBank Robotics, is an advanced humanoid known for its ability to communicate with human beings. Pepper can recognize and answer to human emotions. Its non-verbal communication makes the exchange realistic and interactive.

This robot solution can be deployed in any field such as education, reception, hospitality, and much more.

It can be programmed to match the needs of the application at hand. This aspect makes the opportunities to use Pepper in our line of work unlimited.

In our case, we decided to explore the ability to use our social robot in a martial arts learning environment where the coach has many students from diverse backgrounds. While Pepper is very effective in the education sector, this project allows us to assist the instructor by allowing Pepper to teach lower-level students to improve the coach’s productivity.

Report structure

Table of Contents

Introduction

In this human robot interaction class, we aim to integrate Pepper in a Martial Arts Class to assist the coach in teaching students from various backgrounds. We will focus on Karate to start since other martial arts may emulate the process that we apply in this project.

Such a solution could encourage people to entertain the opportunity of learning without being judged, ignored or even mocked. They can easily reach amateur levels by learning the basics of Karate based on tailored tutorials within an interactive session.

Proposal

Project Description

A Martial Arts Instructor that can teach basic fighting stances and manners, assess someone's skill, teach interested students about the history of Martial Arts and teach some stances.

The deployment context

Any sort of Martial Arts class with room for an extra hand to assist the human sensei with teaching the younger and less experienced fighters.

The intended audience

Coaches who have mixed age/experience classes, better for children aid or any beginner level student.

The objectives

To help unexperienced students get familiar with the basics and to give the coach more freedom to give more attention to other more experienced students and make a report about individual levels of experience as well as a class report.

The tools needed

An external server/machine for all computations.

Computations related to the AI model and the Image detection, recognition, and segmentation models

Motivation behind selecting the project topic

The problem

Coaches being overwhelmed when faced with mixed aged or mixed experience level classes and thus not being able to give each group the attention they need to get better.

The added value

Provides a helping hand to the coach when needed and get tedious methods out their way.

Activities and tools needed

An external server for all computations

Notes

Potential addition of movement recognition on stance not on complex moves

Add image segmentation reference picture (points on joints to analyze the stance)

Material used

Applications

- VS code
- Choregraph

Languages and Frameworks

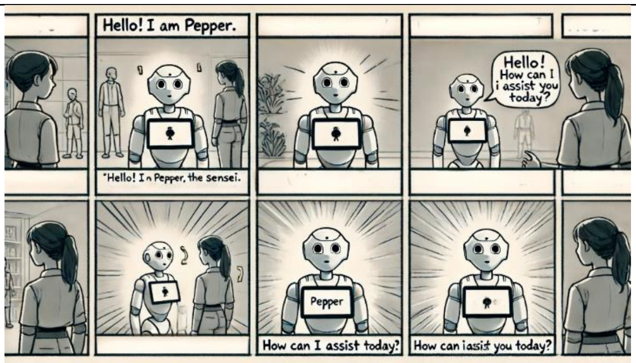
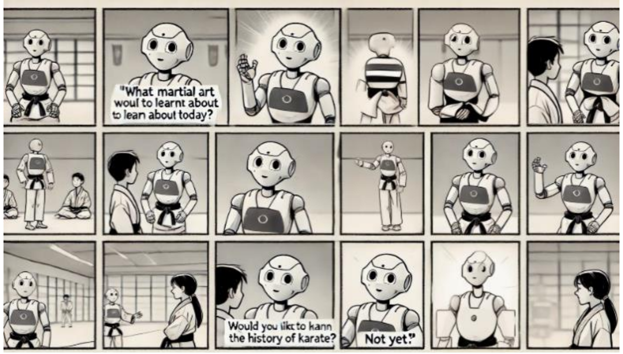
- Python 2.7
- Python 3.7 and its libraries: ultralytics, os, pysftp
- Css
- HTML
- NAO.qi 2.9

Use Cases

Use Case 1-2

Author(s):
Karl Romanos Abou Jaoudeh and Antonio Haddad

UC-ID and Name	**UC1-2 Initiate Interaction and Assess Skills**		
Created By	Group A	**Creation Date**	December 17, 2024
Actors	Pepper the robot Student		
Trigger	The student approaches Pepper		
Description (Objectives/Goals)	To commence the interaction between Pepper the Robot and a potential Student and assess their skill level.		
Preconditions	• Pepper is powered on and is operational • Pepper is in a visible and accessible location in the martial arts class		
Postconditions	• Pepper has greeted the student • The student is aware of Pepper's presence and level • Pepper has collected basic information about the student's interests and experiences • Pepper has determined the next appropriate action (teach history, or begin instruction)		
Action Sequence (Success Scenario)	1. Pepper detects a person approaching 2. Pepper turns to face the person 3. Pepper greets the person 4. Pepper waits for the student's response 5. Pepper prompts the user to fill in their level verbally or using the tablet		

	6. User fills in the answers 7. Based on the student's response, Pepper gets an understanding about the skill level 8. Pepper asks if the user is ready to learn the martial art
Extensions	
Requirements	- Pepper must have functional proximity sensors - Pepper must be able to rotate to face the approaching person - Pepper must have clear, audible speech output
Storyboards	 Hello! I am Pepper. "Hello! I n Pepper, the sensei. Hello! How can I i assist you today? Pepper How can I assist today? How can iassit you today? **Figure 1 – UC1 Storyboard**  "What martial art woul to learnt about to lean about today? Would you likc to kann the history of karate? Not yet?" **Figure 2 - UC2 Storyboard**
Priority	
Related Use Cases	UC3

Assumptions	
Open Issues	

Table 1 - UC1

Use Case 3

Author: Asif Alam

UC-ID and Name	**UC3 Interactive Martials Arts History Exploration**		
Created By	Group A	**Creation Date**	October 6, 2024
Actors	Pepper the Robot Student		
Trigger	UC1-2 is complete, and student expressed interest in learning about martial arts history		
Description (Objectives/Goals)	To provide the student with an engaging, interactive exploration of martial arts history, adapting to their interests and encouraging active participation.		
Preconditions	The student has expressed interest in learning about martial arts history Pepper has access to a diverse database of martial arts historical information		
Postconditions	-The student has gained insights into martial arts history through an interactive storytelling experience -The student has actively participated in the historical exploration by answering questions and making choices -Pepper has adapted the storytelling based on the student's responses and interests		
Action Sequence (Success Scenario)	1) Pepper initiates the history exploration and asks for interest between origins, famous masters or evolution of technique.		

2) Based on the student's choice, Pepper begins an interactive story: "Let's start our journey with [chosen aspect].”

3) Pepper presents a historical scenario and asks the student to make a choice: "We've arrived in ancient China. Do you want to [verb] [relevant idea or place] or [verb] [relevant idea or place]?"

4) Pepper adapts the story based on the student's choice, providing relevant historical information.

5) Throughout the story, Pepper asks engaging questions.

6) At key points, Pepper offers choices that affect the direction of the historical exploration

7) Pepper occasionally challenges the student with historical "what-if" scenarios based on previous ideas.

8) As the exploration concludes, Pepper summarizes the journey and key learnings:

9) Pepper gauges the student's interest for further exploration: "Would you like to dive deeper into any part of our journey, or are you ready to try some moves yourself?"

Alternative action sequence:

1) If the student shows particular interest in a specific area, Pepper can offer more detailed information on that topic.

2) If the student struggles with the quiz, Pepper can offer to review the key points again

	3) If the student expresses disinterest in history, Pepper can suggest moving to physical instruction sooner
Extensions	**3a. No matching historical information is found.** Pepper offers alternative related topics or asks the student to choose a different aspect of martial arts history. **5a. The student provides an unexpected or off-topic response:** Pepper acknowledges the student's input and gently guides the conversation back to martial arts history. **7a. The student repeatedly chooses to skip or fast-forward through historical periods:** Pepper inquires if the student would prefer to move on to learning physical techniques. **8a. The student struggles to engage with the "what-if" scenarios:** Pepper shifts to presenting straightforward historical facts with simple comprehension questions. **10a. The student expresses disinterest in further historical exploration:** 10a1. Pepper summarizes the key points covered and transitions to introducing basic martial arts movements.
Requirements	1. Pepper must have an extensive, interconnected database of martial arts history. 2. Pepper must be able to adapt its storytelling based on student choices and responses. 3. Pepper should use its tablet display to show relevant images or animations that complement the storytelling.

	4. Pepper must be able to track the flow of the interactive story to maintain coherence. 5. Pepper should have adaptive responses based on the student's performance and interest levels 6. Pepper must be able to recognize signs of engagement or disengagement from the student and adjust accordingly.
Storyboards	 Great! Let's sep the history of Karate together? oo ye aterming? Great! begoming? KOU KARATE KARATE? Do's find interesting? Wou-4 key poonts Dou yo sey ats Great! Let's explore, explore the history of Karate! Karate together? ·Would yo key port. Let's explore this history of Karate this points? Now, ke's seave ? Will winke mout this poirts? ow, we hike to t KARATE? Doi you history toge ? Now, let's you know mow nbout th part? Now, let's rave mow a quiz a quiz? I L'll ull have un quiz afterward! Now, I 'll hav part? **Figure 3 - UC3 Storyboard**
Priority	
Related Use Cases	UC1-2, UC4
Assumptions	
Open Issues	

Table 2 – UC3

Use Case 4

Author: Elias-Charbel Salameh

UC-ID and Name	**UC-04 Teach Martial Arts**		
Created By	Group A	**Creation Date**	October 6, 2024
Actors	Pepper the robot Student		
Trigger	UC3 must be complete or the user requests to move on from history earlier		
Description (Objectives/Goals)	To instruct the students in basic martial arts stances and movements		
Preconditions	1. The student has exhibited readiness to learn physical movements 2. Pepper has assessed the student’s skill level and physical condition		
Postconditions	1. The student has learned and practiced basic martial arts movements 2. Pepper has provided feedback on the student’s performance		
Action Sequence (Success Scenario)	1. Pepper starts with a brief warm-up: "Let's start with some stretching." 2. Pepper demonstrates and guides through stretching exercises 3. Pepper introduces the starting stance: "Now, let's learn the starting stance."		

	4. Pepper demonstrates the stance and instructs the student to mimic by displaying videos of a coach performing the movement 5. Pepper uses visual recognition to assess the student's stance and provides feedback 6. Pepper progresses to teaching basic moves, demonstrating each one 7. Pepper observes the student's attempts and provides feedback 8. After completing the session, Pepper concludes: "Great job! Keep practicing these moves." Alternative action sequence: 1. If the student struggles with a move, Pepper asks the user to repeat the action 2. If the student excels, Pepper can introduce more advanced variations
Extensions	
Requirements	1. Pepper must be capable of demonstrating martial arts movements 2. Pepper must have visual recognition capabilities to assess student performance 3. Pepper must be able to provide clear, step-by-step instructions 4. Pepper should have safety protocols to ensure students don’t overexert themselves

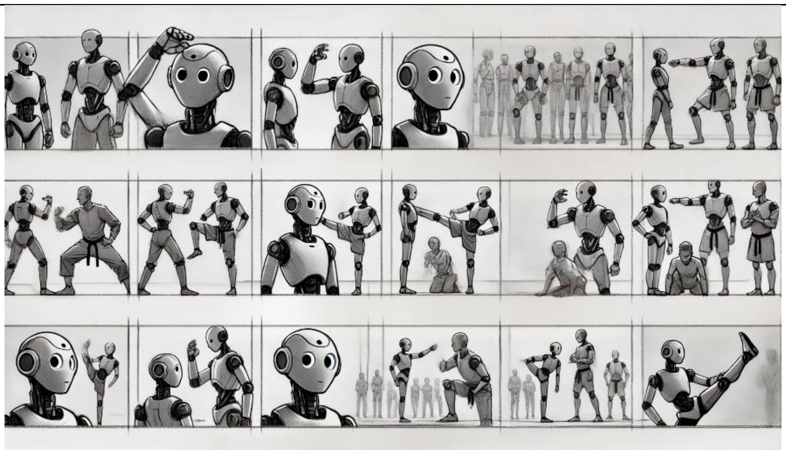
Storyboards	 **Figure 4 - UC4 Storyboard**
Priority	
Related Use Cases	UC3
Assumptions	
Open Issues	

Table 3 - UC4

Personas

Persona 1

Persona - "I want to be able to defend myself at school"

Demographics

- **Name**: Billy
- **Age**: 17
- **Experience Level**: Beginner
- **Classes Taken**: None (0 months)
- **Preferred Martial Art**: Karate

Motivation

- **Main Motive:** Is bullied at school
- **Hobbies**: Interested in action movies and martial arts video games.
- **Goals**: Wants to learn martial arts for self-defense and to build confidence.
- **Sports**: Not very active in other sports but motivated to improve physical fitness through martial arts.
- **Personal Challenges**:
  - Struggles with coordination and memorizing movements
  - needs personalized guidance to stay motivated
  - lacks confidence
- **School**:
  - Average performer

  - interested in physical activities and hands-on learning rather than academic subjects.
- **Emotional Needs**:
  - Billy is shy and lacks confidence in public, so practicing with a robot helps them feel more comfortable without judgment from others.
  - Constant bullying led him to this situation

<u>Social Environment</u>

- **Background**:
  - Comes from a middle-class family. Parents support extracurricular activities to help Billy develop physical and social skills.
- **Parents**:
  - Supportive but have busy work schedules, so they appreciate solutions that help Billy practice independently at home. They hope martial arts will boost their child's self-confidence.
  - Do not know of their kid's school problems

<u>Interaction with Social Robot</u>

- **Preferred Features**:
  - Visual and vocal feedback on correct stances.
  - Motivational prompts to keep practicing.
  - Encouragement through gamified progress tracking.
  - Regular challenges that encourage learning new moves.
  - Continuous improvement
  - Confidence development

- **Learning Style**:
    - Enjoys step-by-step instructions and positive reinforcement.
    - Prefers visual learning through demonstrations.
- **Expectations from the Robot**:
    - A patient and supportive virtual instructor that adjusts to Billy's pace, provides real-time feedback, and gradually increases difficulty to help Billy progress.

Persona 2

Persona - "I want to become a world champion"

Demographics

- **Name**: Alex
- **Age**: 11
- **Experience Level**: Beginner
- **Classes Taken**: None (0 months)
- **Preferred Martial Art**: Karate

Motivation

- **Main Motive:** wants to be a champion one day
- **Hobbies**: Interested in martial arts sports
- **Goals**: Wants to learn martial arts to compete on a high level
- **Sports**: In-shape, lacks the technique
- **Personal Challenges**:
    - Being left aside by the coaches since he is young

- **School**:
    - High performance in sports classes
    - Prioritizes activities over other academic subjects
- **Emotional Needs**:
    - Alex needs constant feedback on his performance to improve
    - Wants to avoid boredom

<u>Social Environment</u>

- **Background**:
    - Comes from a upper-class family. Parents love to invest in their boy to reach new highs.
- **Parents**:
    - Supportive but have busy work schedules, so they appreciate solutions that help Alex practice independently at home. They hope martial arts will boost their child's self-confidence.
    - progress.

Persona 3

Persona - "I want to improve productivity"

<u>Demographics</u>

- **Name**: Jack
- **Age**: 38
- **Experience Level**: Master
- **Students in his dojo**: 60

- **Different levels**: Beginner, amateur, intermediate, semi-professional, professional

Motivation

- **Main Motive:** wants to make the most money possible
- **Hobbies**: Interested in teaching martial arts sports
- **Goals**: Wants to improve productivity
- **Preferences**: Has been teaching for 10 years and wants to optimize the process.

  Annoyed of beginner levels students taking much of his time

  Wants to train the next generation of champions without having to lose on money

Implementation

We started by splitting the tasks such that UC1-2 are combined, their end leads to the start of UC3, and its respective end launches UC4.

Let's start with UC1-2. The purpose of this use case is to **Initiate the Interaction and Assess the user's level**. The scope of this section is to prepare the user for the following history story-telling part.

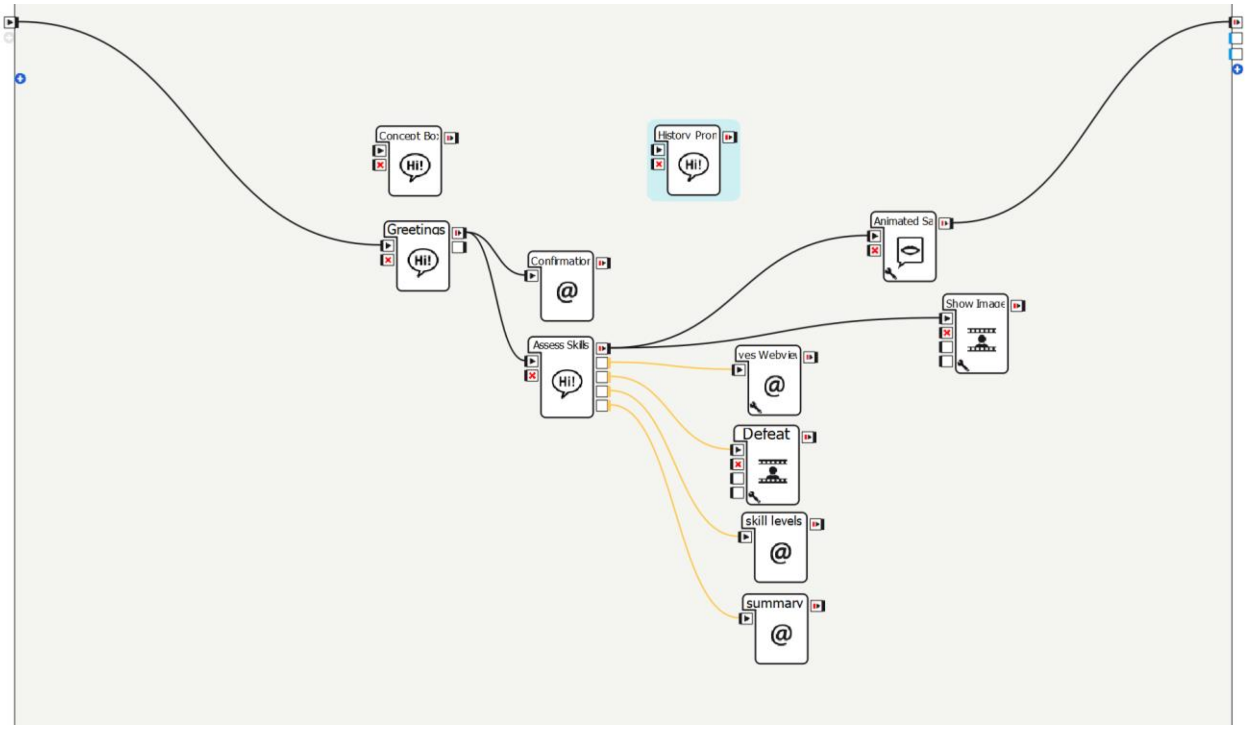


Figure 5 - UC1-2 behavior

In the above figure, we can see the flow of our design. We start with the **greetings** dialog box that starts the interaction between Pepper and the karate learner.

The **Greeting** dialog box in fig.6 includes the **Concepts**, fig.7, needed in this design.

On Start, the proposal with the tag "introduction" is accessed. Pepper greets the user with a random pre-defined expression set in the "greetings" concept.

The $\backslash pau = X \backslash$ notation allows Pepper to pause for X milliseconds.

After completing the proposal, we are officially in the scope of the introduction proposal. The user can either answer an affirmative answer from the “yes” concept or stop the process by using of the answers available in the “no” concept.

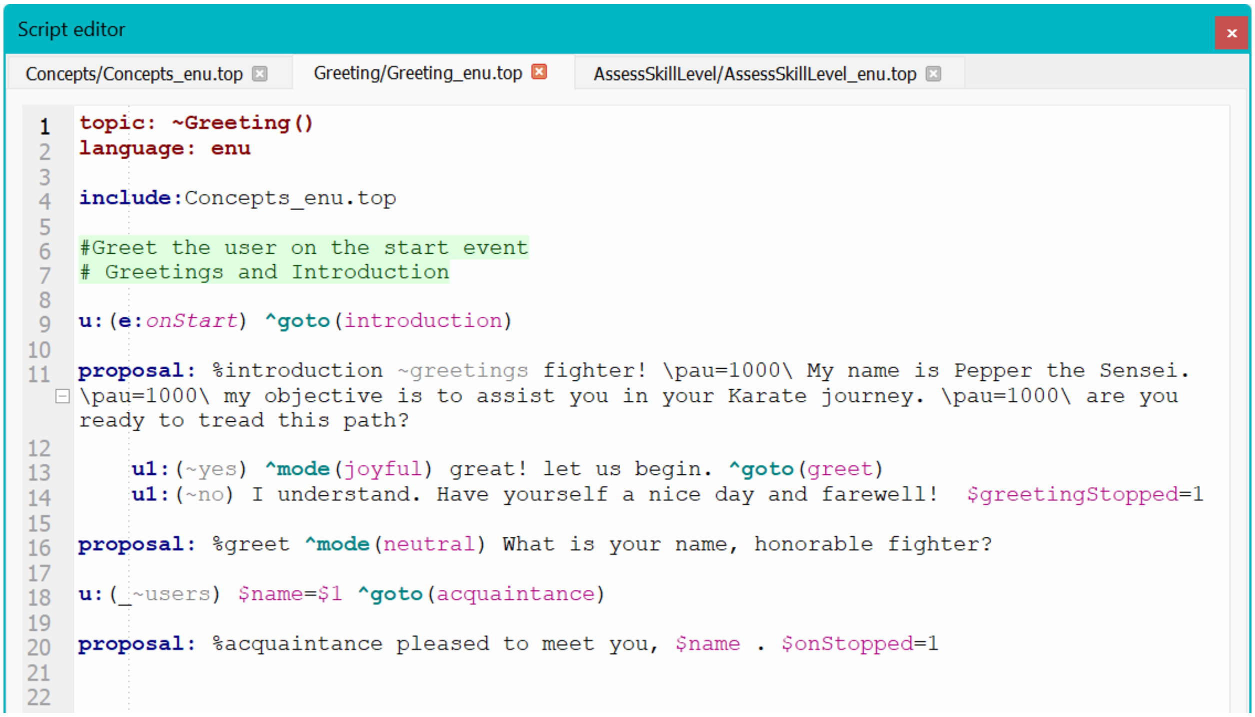
^mode(joyful) allows us to arrange the tonality of Pepper during the following messages until it is reset.

We use the **^goto(tagOfTheProposal)** to navigate and access the next proposal.

The tonality adjustment back to neutral allows us the reset Pepper’s enthusiasm.

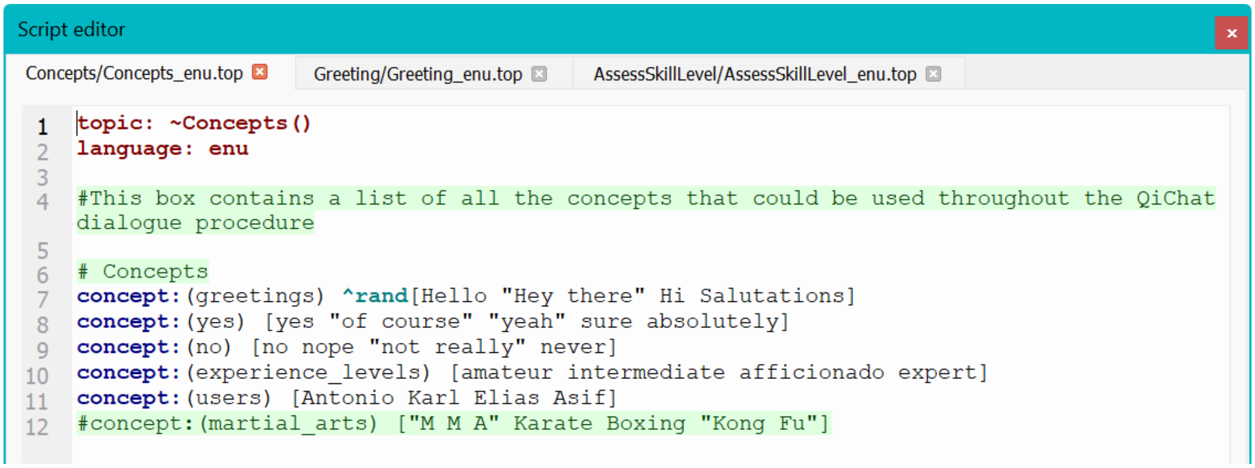
Then, we use the _ character to save the value after it. Here, Pepper expects one of the four names set in the “name” concept. It saves the string in the **$1** as the first entry from this prompt. We proceed to store the **$1** variable in the **$name** local variable.

By jumping to the last proposal, we finalize and stop this dialog box by raising the **onStopped** variable.


Script editor
×
Concepts/Concepts_enu.top
Greeting/Greeting_enu.top
AssessSkillLevel/AssessSkillLevel_enu.top

```
topic: ~Greeting()
language: enu

include:Concepts_enu.top

#Greet the user on the start event
# Greetings and Introduction

u:(e:onStart) ^goto(introduction)

proposal: %introduction ~greetings fighter! \pau=1000\ My name is Pepper the Sensei. \pau=1000\ my objective is to assist you in your Karate journey. \pau=1000\ are you ready to tread this path?

    u1:(~yes) ^mode(joyful) great! let us begin. ^goto(greet)
    u1:(~no) I understand. Have yourself a nice day and farewell!  $greetingStopped=1

proposal: %greet ^mode(neutral) What is your name, honorable fighter?

u:(_~users) $name=$1 ^goto(acquaintance)

proposal: %acquaintance pleased to meet you, $name . $onStopped=1


```

Figure 6 - Greeting Dialog Box

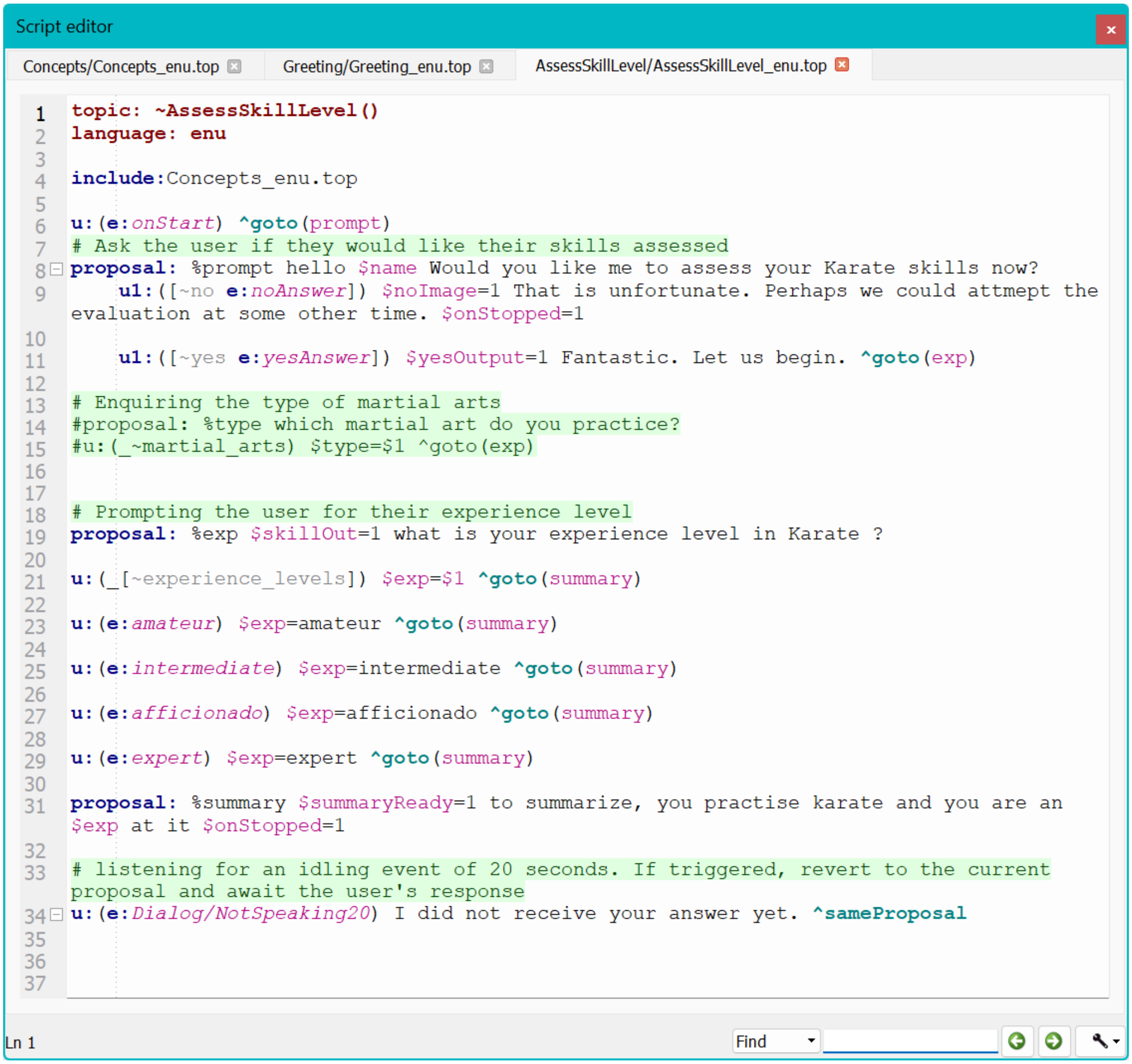

Script editor
×
Concepts/Concepts_enu.top
Greeting/Greeting_enu.top
AssessSkillLevel/AssessSkillLevel_enu.top

```
topic: ~Concepts()
language: enu

#This box contains a list of all the concepts that could be used throughout the QiChat dialogue procedure

# Concepts
concept:(greetings) ^rand[Hello "Hey there" Hi Salutations]
concept:(yes) [yes "of course" "yeah" sure absolutely]
concept:(no) [no nope "not really" never]
concept:(experience_levels) [amateur intermediate afficionado expert]
concept:(users) [Antonio Karl Elias Asif]
#concept:(martial_arts) ["M M A" Karate Boxing "Kong Fu"]
```

Figure 7 - Concepts Dialog Box

Fig.7 displays the needed concepts for UC1-2.

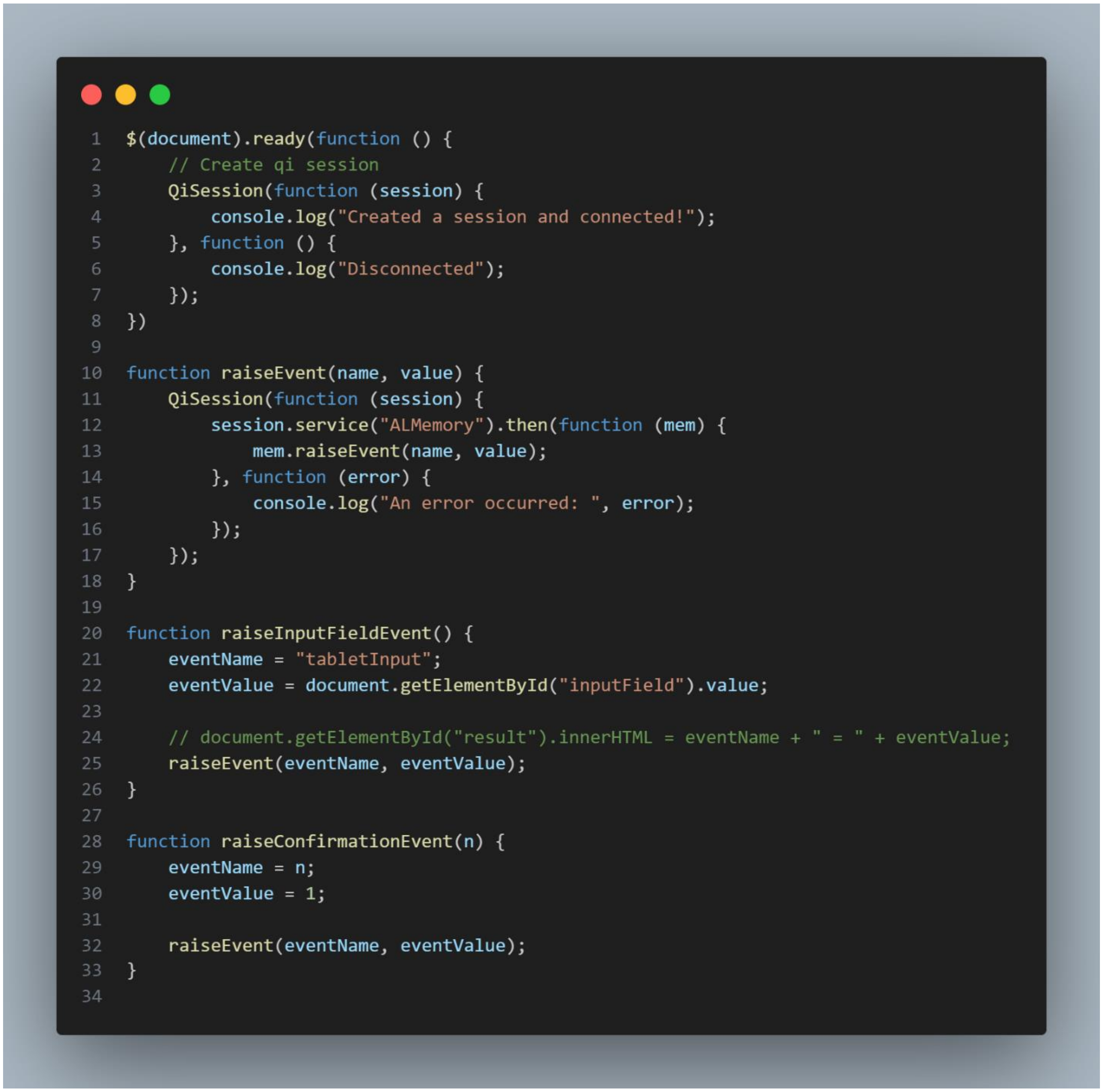
We go back fig.5 to proceed to the “Assess skills” dialog box available in fig.8. The On Stopped of the previous box calls the default confirmation webview to be displayed. No changes were made to the HTML and CSS files for this webview.


Script editor
Concepts/Concepts_enu.top
Greeting/Greeting_enu.top
AssessSkillLevel/AssessSkillLevel_enu.top
Ln 1
Find

```
topic: ~AssessSkillLevel()
language: enu

include:Concepts_enu.top

u:(e:onStart) ^goto(prompt)
# Ask the user if they would like their skills assessed
proposal: %prompt hello $name Would you like me to assess your Karate skills now?
    u1:([~no e:noAnswer]) $noImage=1 That is unfortunate. Perhaps we could attmept the evaluation at some other time. $onStopped=1

    u1:([~yes e:yesAnswer]) $yesOutput=1 Fantastic. Let us begin. ^goto(exp)

# Enquiring the type of martial arts
#proposal: %type which martial art do you practice?
#u:(_~martial_arts) $type=$1 ^goto(exp)


# Prompting the user for their experience level
proposal: %exp $skillOut=1 what is your experience level in Karate ?

u:(_[~experience_levels]) $exp=$1 ^goto(summary)

u:(e:amateur) $exp=amateur ^goto(summary)

u:(e:intermediate) $exp=intermediate ^goto(summary)

u:(e:afficionado) $exp=afficionado ^goto(summary)

u:(e:expert) $exp=expert ^goto(summary)

proposal: %summary $summaryReady=1 to summarize, you practise karate and you are an $exp at it $onStopped=1

# listening for an idling event of 20 seconds. If triggered, revert to the current proposal and await the user's response
u:(e:Dialog/NotSpeaking20) I did not receive your answer yet. ^sameProposal



```

Figure 8 - Assess User Skills Dialog Box

In fig.8, we assess the user's skills by prompting him about his level at practicing karate. The user has 4 options displayed on the screens as buttons. The events are raised from the javascript and qi session connection through the function that raises the event.



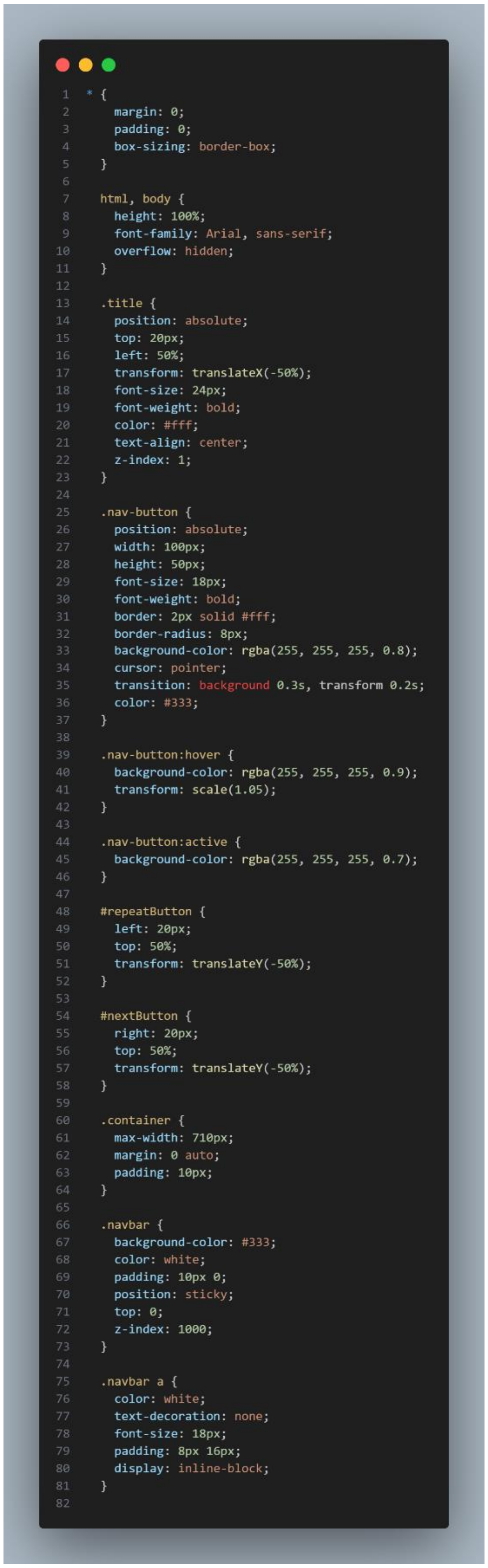
```
$(document).ready(function () {
    // Create qi session
    QiSession(function (session) {
        console.log("Created a session and connected!");
    }, function () {
        console.log("Disconnected");
    });
})

function raiseEvent(name, value) {
    QiSession(function (session) {
        session.service("ALMemory").then(function (mem) {
            mem.raiseEvent(name, value);
        }, function (error) {
            console.log("An error occurred: ", error);
        });
    });
}

function raiseInputFieldEvent() {
    eventName = "tabletInput";
    eventValue = document.getElementById("inputField").value;

    // document.getElementById("result").innerHTML = eventName + " = " + eventValue;
    raiseEvent(eventName, eventValue);
}

function raiseConfirmationEvent(n) {
    eventName = n;
    eventValue = 1;

    raiseEvent(eventName, eventValue);
}

```

Figure 9 - Javascript Raise Event Logic

This file in fig.9 is unchanged and upon being raised, its sets the event to a high value which drivers the user rule after the tablet input was delivered.

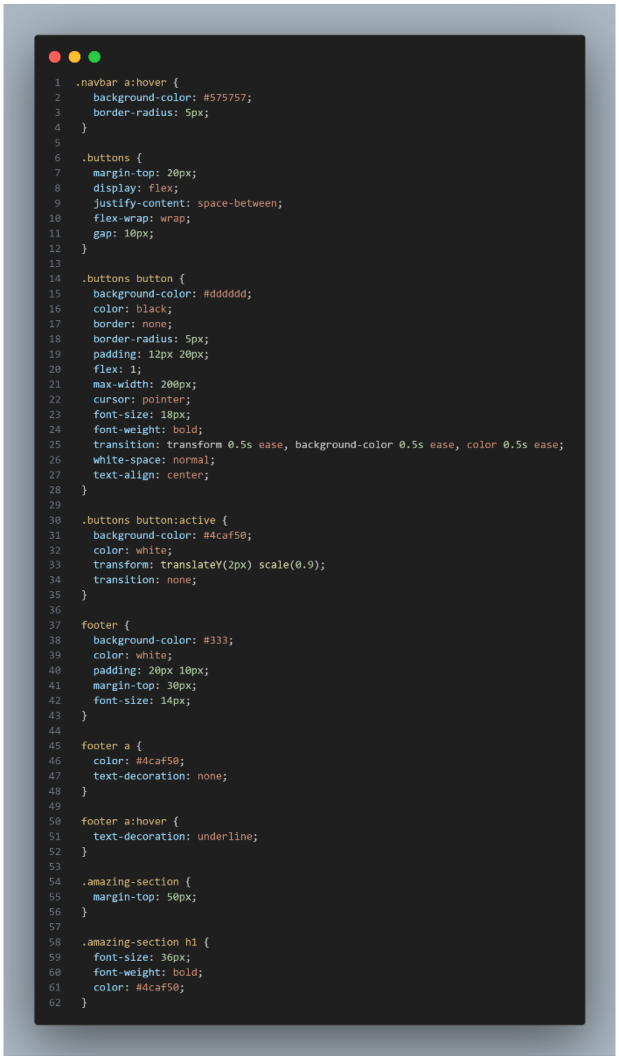
We also save the value of the level for Pepper to affirm it back to the user.

In the case where the user has been silent for more than 20 seconds, we prompt him to answer the question.



```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

html, body {
  height: 100%;
  font-family: Arial, sans-serif;
  overflow: hidden;
}

.title {
  position: absolute;
  top: 20px;
  left: 50%;
  transform: translateX(-50%);
  font-size: 24px;
  font-weight: bold;
  color: #fff;
  text-align: center;
  z-index: 1;
}

.nav-button {
  position: absolute;
  width: 100px;
  height: 50px;
  font-size: 18px;
  font-weight: bold;
  border: 2px solid #fff;
  border-radius: 8px;
  background-color: rgba(255, 255, 255, 0.8);
  cursor: pointer;
  transition: background 0.3s, transform 0.2s;
  color: #333;
}

.nav-button:hover {
  background-color: rgba(255, 255, 255, 0.9);
  transform: scale(1.05);
}

.nav-button:active {
  background-color: rgba(255, 255, 255, 0.7);
}

#repeatButton {
  left: 20px;
  top: 50%;
  transform: translateY(-50%);
}

#nextButton {
  right: 20px;
  top: 50%;
  transform: translateY(-50%);
}

.container {
  max-width: 710px;
  margin: 0 auto;
  padding: 10px;
}

.navbar {
  background-color: #333;
  color: white;
  padding: 10px 0;
  position: sticky;
  top: 0;
  z-index: 1000;
}

.navbar a {
  color: white;
  text-decoration: none;
  font-size: 18px;
  padding: 8px 16px;
  display: inline-block;
}

```

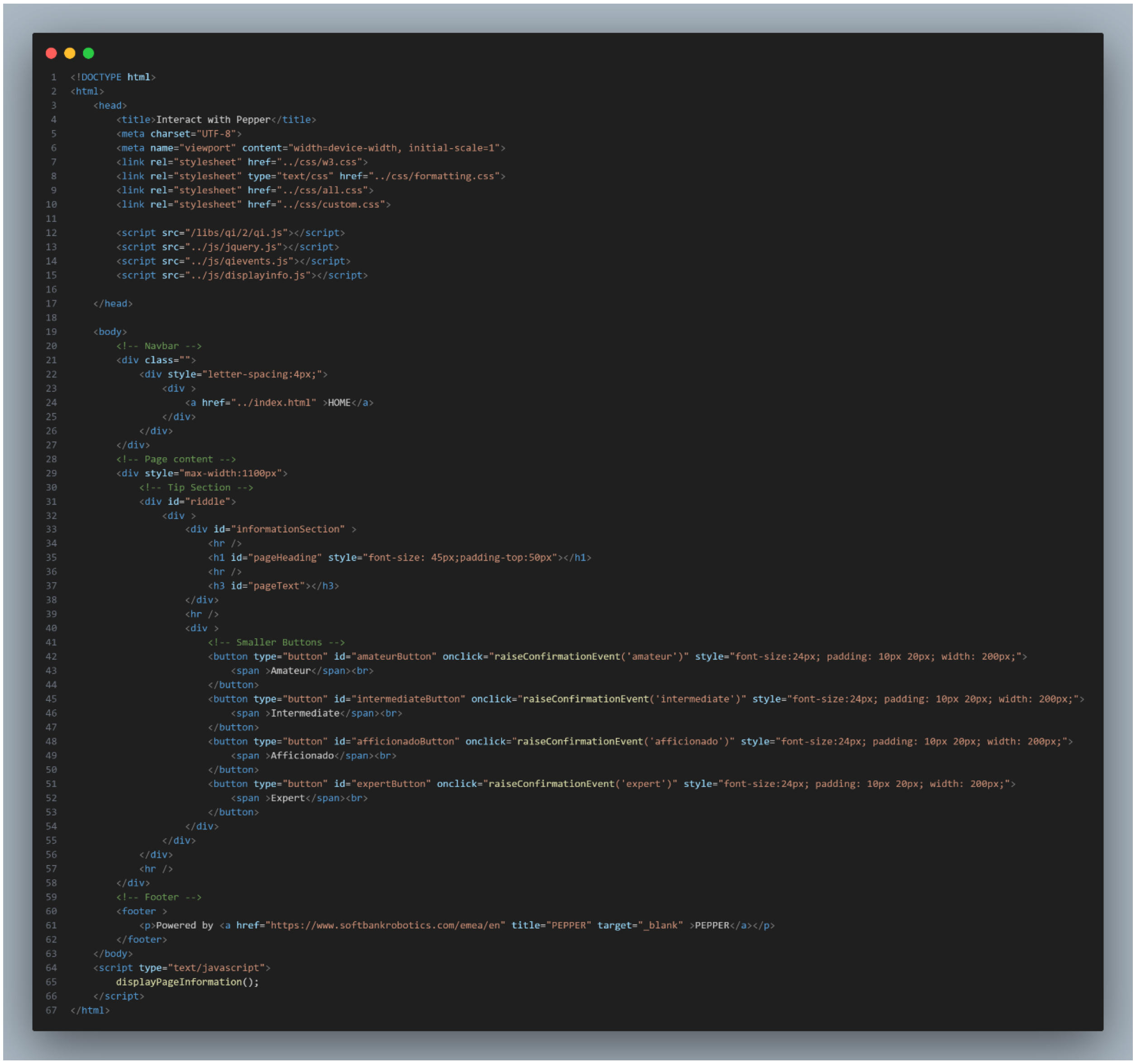
Figure 11 - Custom CSS file pt.1



```css
.navbar a:hover {
  background-color: #575757;
  border-radius: 5px;
}

.buttons {
  margin-top: 20px;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  gap: 10px;
}

.buttons button {
  background-color: #dddddd;
  color: black;
  border: none;
  border-radius: 5px;
  padding: 12px 20px;
  flex: 1;
  max-width: 200px;
  cursor: pointer;
  font-size: 18px;
  font-weight: bold;
  transition: transform 0.5s ease, background-color 0.5s ease, color 0.5s ease;
  white-space: normal;
  text-align: center;
}

.buttons button:active {
  background-color: #4caf50;
  color: white;
  transform: translateY(2px) scale(0.9);
  transition: none;
}

footer {
  background-color: #333;
  color: white;
  padding: 20px 10px;
  margin-top: 30px;
  font-size: 14px;
}

footer a {
  color: #4caf50;
  text-decoration: none;
}

footer a:hover {
  text-decoration: underline;
}

.amazing-section {
  margin-top: 50px;
}

.amazing-section h1 {
  font-size: 36px;
  font-weight: bold;
  color: #4caf50;
}
```

Figure 10 - Custom CSS file pt.2

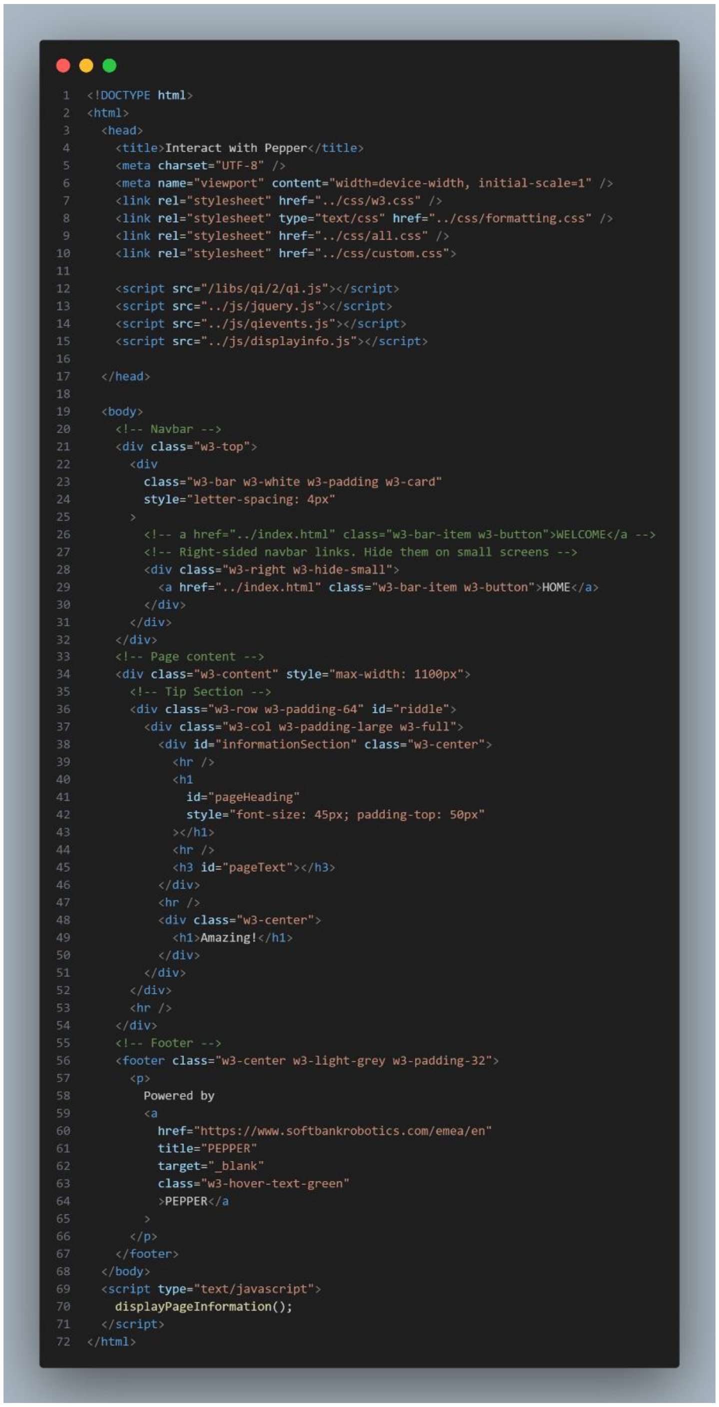
The Css file is a custom file designed to follow best practices of software engineering by linking a reference to the template that we are using in the needed files such as the “skills” and “summary” HTML files.



```html
<!DOCTYPE html>
<html>
    <head>
        <title>Interact with Pepper</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="../css/w3.css">
        <link rel="stylesheet" type="text/css" href="../css/formatting.css">
        <link rel="stylesheet" href="../css/all.css">
        <link rel="stylesheet" href="../css/custom.css">

        <script src="/libs/qi/2/qi.js"></script>
        <script src="../js/jquery.js"></script>
        <script src="../js/qievents.js"></script>
        <script src="../js/displayinfo.js"></script>

    </head>

    <body>
        <!-- Navbar -->
        <div class="">
            <div style="letter-spacing:4px;">
                <div >
                    <a href="../index.html" >HOME</a>
                </div>
            </div>
        </div>
        <!-- Page content -->
        <div style="max-width:1100px">
            <!-- Tip Section -->
            <div id="riddle">
                <div >
                    <div id="informationSection" >
                        <hr />
                        <h1 id="pageHeading" style="font-size: 45px;padding-top:50px"></h1>
                        <hr />
                        <h3 id="pageText"></h3>
                    </div>
                    <hr />
                    <div >
                        <!-- Smaller Buttons -->
                        <button type="button" id="amateurButton" onclick="raiseConfirmationEvent('amateur')" style="font-size:24px; padding: 10px 20px; width: 200px;">
                            <span >Amateur</span><br>
                        </button>
                        <button type="button" id="intermediateButton" onclick="raiseConfirmationEvent('intermediate')" style="font-size:24px; padding: 10px 20px; width: 200px;">
                            <span >Intermediate</span><br>
                        </button>
                        <button type="button" id="afficionadoButton" onclick="raiseConfirmationEvent('afficionado')" style="font-size:24px; padding: 10px 20px; width: 200px;">
                            <span >Afficionado</span><br>
                        </button>
                        <button type="button" id="expertButton" onclick="raiseConfirmationEvent('expert')" style="font-size:24px; padding: 10px 20px; width: 200px;">
                            <span >Expert</span><br>
                        </button>
                    </div>
                </div>
            </div>
            <hr />
        </div>
        <!-- Footer -->
        <footer >
            <p>Powered by <a href="https://www.softbankrobotics.com/emea/en" title="PEPPER" target="_blank" >PEPPER</a></p>
        </footer>
    </body>
    <script type="text/javascript">
        displayPageInformation();
    </script>
</html>
```

Figure 12 - Skills HTML page

Here are the four needed buttons with their ID and the event listener to raise the 4 level events and issue a user input.



```html
<!DOCTYPE html>
<html>
  <head>
    <title>Interact with Pepper</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="../css/w3.css" />
    <link rel="stylesheet" type="text/css" href="../css/formatting.css" />
    <link rel="stylesheet" href="../css/all.css" />
    <link rel="stylesheet" href="../css/custom.css">

    <script src="/libs/qi/2/qi.js"></script>
    <script src="../js/jquery.js"></script>
    <script src="../js/qievents.js"></script>
    <script src="../js/displayinfo.js"></script>

  </head>

  <body>
    <!-- Navbar -->
    <div class="w3-top">
      <div
        class="w3-bar w3-white w3-padding w3-card"
        style="letter-spacing: 4px"
      >
        <!-- a href="../index.html" class="w3-bar-item w3-button">WELCOME</a -->
        <!-- Right-sided navbar links. Hide them on small screens -->
        <div class="w3-right w3-hide-small">
          <a href="../index.html" class="w3-bar-item w3-button">HOME</a>
        </div>
      </div>
    </div>
    <!-- Page content -->
    <div class="w3-content" style="max-width: 1100px">
      <!-- Tip Section -->
      <div class="w3-row w3-padding-64" id="riddle">
        <div class="w3-col w3-padding-large w3-full">
          <div id="informationSection" class="w3-center">
            <hr />
            <h1
              id="pageHeading"
              style="font-size: 45px; padding-top: 50px"
            ></h1>
            <hr />
            <h3 id="pageText"></h3>
          </div>
          <hr />
          <div class="w3-center">
            <h1>Amazing!</h1>
          </div>
        </div>
      </div>
      <hr />
    </div>
    <!-- Footer -->
    <footer class="w3-center w3-light-grey w3-padding-32">
      <p>
        Powered by
        <a
          href="https://www.softbankrobotics.com/emea/en"
          title="PEPPER"
          target="_blank"
          class="w3-hover-text-green"
          >PEPPER</a
        >
      </p>
    </footer>
  </body>
  <script type="text/javascript">
    displayPageInformation();
  </script>
</html>
```

Figure 13 - Summary HTML

The **Summary** HTML file is an image of the display info html template.

The display image process is done without any change from the template since there were just two images to display.

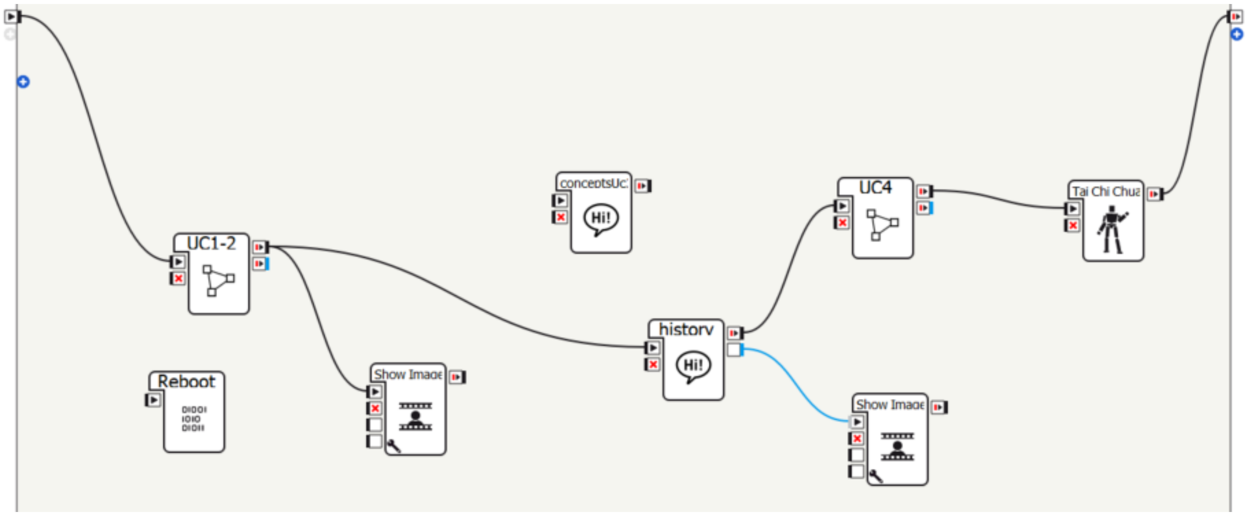
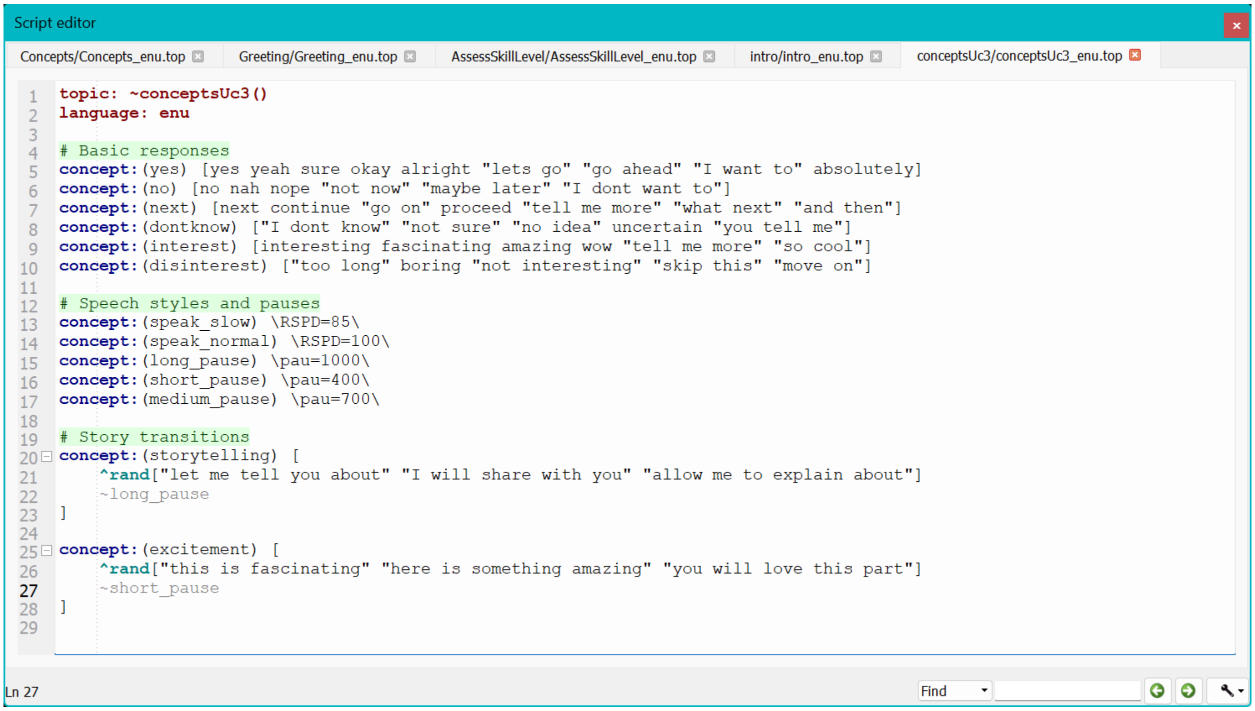


Figure 14 - UC3 Behavior

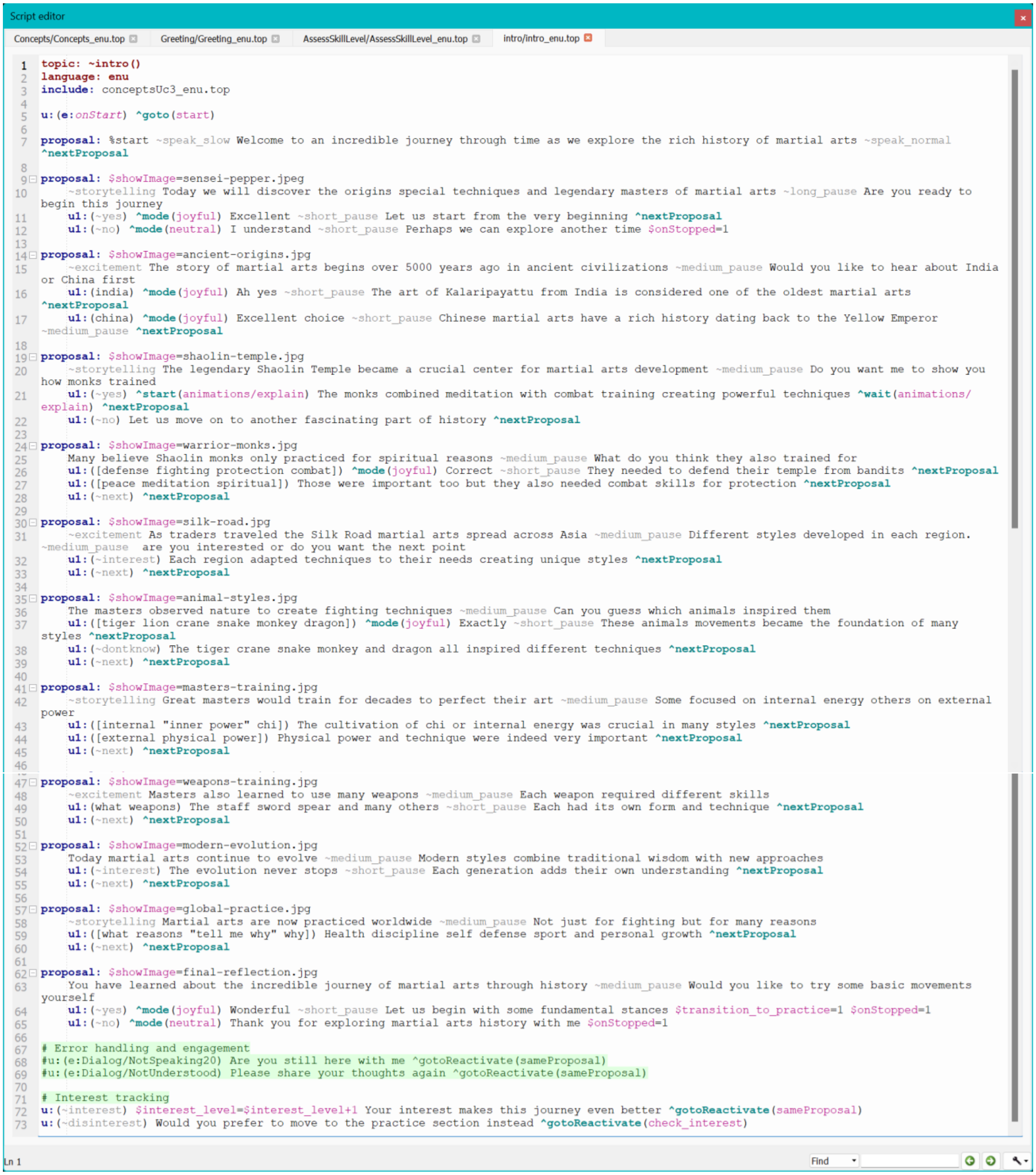
The third use case has a purpose to teach the history of Karate to the students. Some technical issues forced us to abort the usage of a **Run Behavior** box instead of this sub-optimal method. However, since we just need three blocks, we can afford to go for this alternate solution.



Script editor
Concepts/Concepts_enu.top
Greeting/Greeting_enu.top
AssessSkillLevel/AssessSkillLevel_enu.top
intro/intro_enu.top
conceptsUc3/conceptsUc3_enu.top
Ln 27
Find

```
topic: ~conceptsUc3()
language: enu

# Basic responses
concept:(yes) [yes yeah sure okay alright "lets go" "go ahead" "I want to" absolutely]
concept:(no) [no nah nope "not now" "maybe later" "I dont want to"]
concept:(next) [next continue "go on" proceed "tell me more" "what next" "and then"]
concept:(dontknow) ["I dont know" "not sure" "no idea" uncertain "you tell me"]
concept:(interest) [interesting fascinating amazing wow "tell me more" "so cool"]
concept:(disinterest) ["too long" boring "not interesting" "skip this" "move on"]

# Speech styles and pauses
concept:(speak_slow) \RSPD=85\
concept:(speak_normal) \RSPD=100\
concept:(long_pause) \pau=1000\
concept:(short_pause) \pau=400\
concept:(medium_pause) \pau=700\

# Story transitions
concept:(storytelling) [
    ^rand["let me tell you about" "I will share with you" "allow me to explain about"]
    ~long_pause
]

concept:(excitement) [
    ^rand["this is fascinating" "here is something amazing" "you will love this part"]
    ~short_pause
]

```

Figure 15 - ConceptsUc3 Dialog box

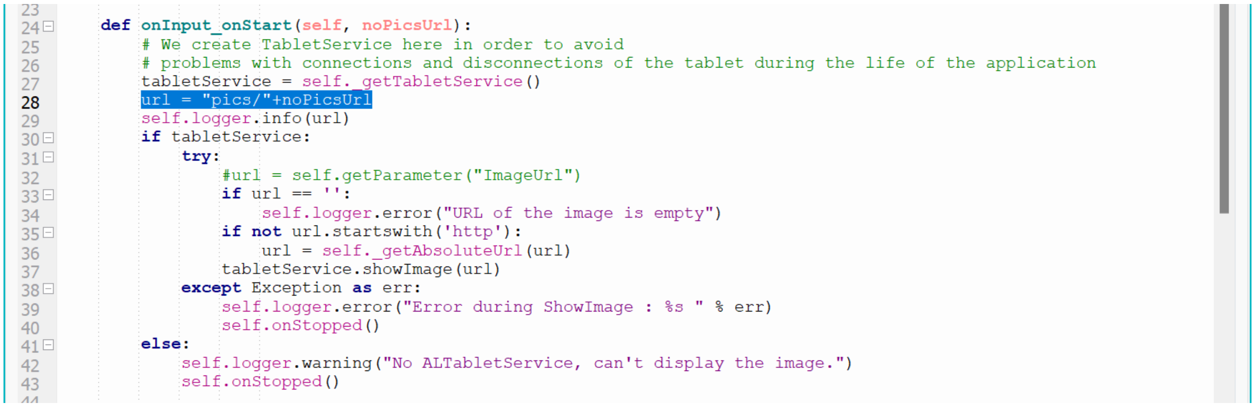

Script editor
Concepts/Concepts_enu.top
Greeting/Greeting_enu.top
AssessSkillLevel/AssessSkillLevel_enu.top
intro/intro_enu.top
Ln 1
Find

```
topic: ~intro()
language: enu
include: conceptsUc3_enu.top

u:(e:onStart) ^goto(start)

proposal: %start ~speak_slow Welcome to an incredible journey through time as we explore the rich history of martial arts ~speak_normal ^nextProposal

proposal: $showImage=sensei-pepper.jpeg
    ~storytelling Today we will discover the origins special techniques and legendary masters of martial arts ~long_pause Are you ready to begin this journey
    u1:(~yes) ^mode(joyful) Excellent ~short_pause Let us start from the very beginning ^nextProposal
    u1:(~no) ^mode(neutral) I understand ~short_pause Perhaps we can explore another time $onStopped=1

proposal: $showImage=ancient-origins.jpg
    ~excitement The story of martial arts begins over 5000 years ago in ancient civilizations ~medium_pause Would you like to hear about India or China first
    u1:(india) ^mode(joyful) Ah yes ~short_pause The art of Kalaripayattu from India is considered one of the oldest martial arts ^nextProposal
    u1:(china) ^mode(joyful) Excellent choice ~short_pause Chinese martial arts have a rich history dating back to the Yellow Emperor ~medium_pause ^nextProposal

proposal: $showImage=shaolin-temple.jpg
    ~storytelling The legendary Shaolin Temple became a crucial center for martial arts development ~medium_pause Do you want me to show you how monks trained
    u1:(~yes) ^start(animations/explain) The monks combined meditation with combat training creating powerful techniques ^wait(animations/explain) ^nextProposal
    u1:(~no) Let us move on to another fascinating part of history ^nextProposal

proposal: $showImage=warrior-monks.jpg
    Many believe Shaolin monks only practiced for spiritual reasons ~medium_pause What do you think they also trained for
    u1:([defense fighting protection combat]) ^mode(joyful) Correct ~short_pause They needed to defend their temple from bandits ^nextProposal
    u1:([peace meditation spiritual]) Those were important too but they also needed combat skills for protection ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=silk-road.jpg
    ~excitement As traders traveled the Silk Road martial arts spread across Asia ~medium_pause Different styles developed in each region. ~medium_pause  are you interested or do you want the next point
    u1:(~interest) Each region adapted techniques to their needs creating unique styles ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=animal-styles.jpg
    The masters observed nature to create fighting techniques ~medium_pause Can you guess which animals inspired them
    u1:([tiger lion crane snake monkey dragon]) ^mode(joyful) Exactly ~short_pause These animals movements became the foundation of many styles ^nextProposal
    u1:(~dontknow) The tiger crane snake monkey and dragon all inspired different techniques ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=masters-training.jpg
    ~storytelling Great masters would train for decades to perfect their art ~medium_pause Some focused on internal energy others on external power
    u1:([internal "inner power" chi]) The cultivation of chi or internal energy was crucial in many styles ^nextProposal
    u1:([external physical power]) Physical power and technique were indeed very important ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=weapons-training.jpg
    ~excitement Masters also learned to use many weapons ~medium_pause Each weapon required different skills
    u1:(what weapons) The staff sword spear and many others ~short_pause Each had its own form and technique ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=modern-evolution.jpg
    Today martial arts continue to evolve ~medium_pause Modern styles combine traditional wisdom with new approaches
    u1:(~interest) The evolution never stops ~short_pause Each generation adds their own understanding ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=global-practice.jpg
    ~storytelling Martial arts are now practiced worldwide ~medium_pause Not just for fighting but for many reasons
    u1:([what reasons "tell me why" why]) Health discipline self defense sport and personal growth ^nextProposal
    u1:(~next) ^nextProposal

proposal: $showImage=final-reflection.jpg
    You have learned about the incredible journey of martial arts through history ~medium_pause Would you like to try some basic movements yourself
    u1:(~yes) ^mode(joyful) Wonderful ~short_pause Let us begin with some fundamental stances $transition_to_practice=1 $onStopped=1
    u1:(~no) ^mode(neutral) Thank you for exploring martial arts history with me $onStopped=1

# Error handling and engagement
#u:(e:Dialog/NotSpeaking20) Are you still here with me ^gotoReactivate(sameProposal)
#u:(e:Dialog/NotUnderstood) Please share your thoughts again ^gotoReactivate(sameProposal)

# Interest tracking
u:(~interest) $interest_level=$interest_level+1 Your interest makes this journey even better ^gotoReactivate(sameProposal)
u:(~disinterest) Would you prefer to move to the practice section instead ^gotoReactivate(check_interest)
```

Figure 16 - History Dialog Box

In fig.15, similarly to UC1-2, we set the concepts. Though, this time, since we are in a storytelling situation, we change the speed of speech of pepper in addition to its tonality

The **Intro** dialog box that has the history of Karate content is a combination of proposal, subrules, tonality and speed of speech changes. An important thing to note is the dynamic image display using the image name as a parameter.



```
23
24      def onInput_onStart(self, noPicsUrl):
25          # We create TabletService here in order to avoid
26          # problems with connections and disconnections of the tablet during the life of the application
27          tabletService = self._getTabletService()
28          url = "pics/"+noPicsUrl
29          self.logger.info(url)
30          if tabletService:
31              try:
32                  #url = self.getParameter("ImageUrl")
33                  if url == '':
34                      self.logger.error("URL of the image is empty")
35                  if not url.startswith('http'):
36                      url = self._getAbsoluteUrl(url)
37                  tabletService.showImage(url)
38              except Exception as err:
39                  self.logger.error("Error during ShowImage : %s " % err)
40                  self.onStopped()
41          else:
42              self.logger.warning("No ALTabletService, can't display the image.")
43              self.onStopped()
```

Figure 17 - Dynamic Image Display

Line 28 pf fig.17 represents the change that we made to the function to make it dynamic.

We added a parameter to the function onInput_onStart called noPicsUrl.

We concatenate the string parameter passed in the Intro dialog box to the “pics/” string to have the relative path to reach the image since we have a subfolder now.

We also need to change the type of the onStart of the box to a string to reduce any error in the type-compatibility.

We reach the end of UC3.

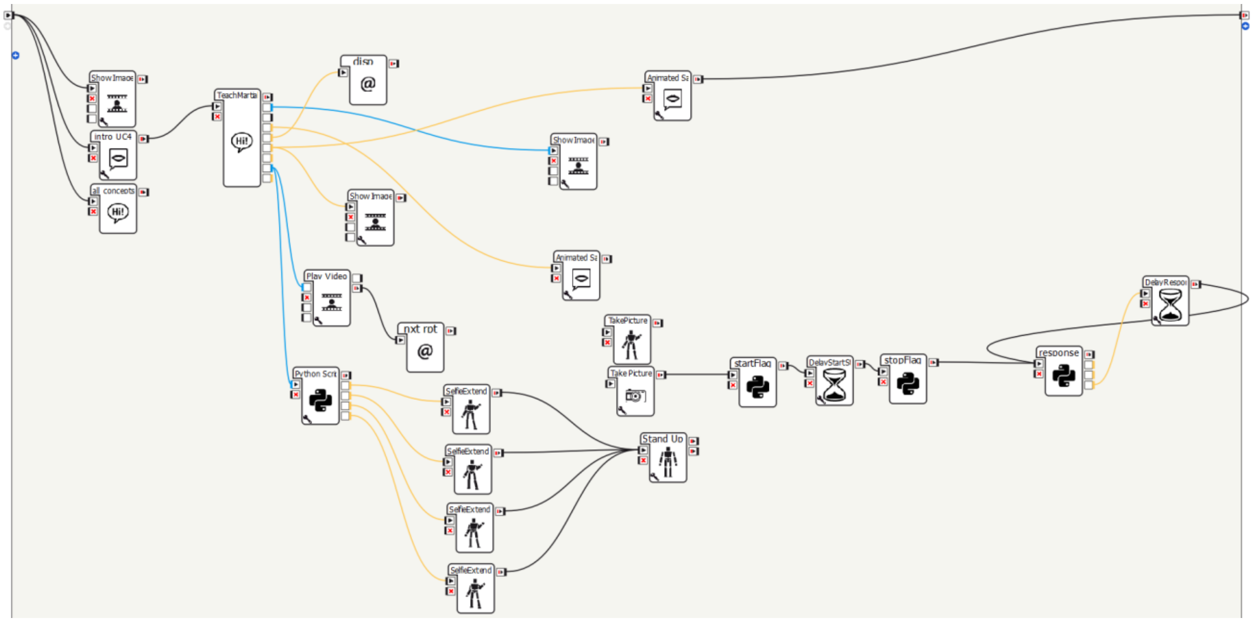
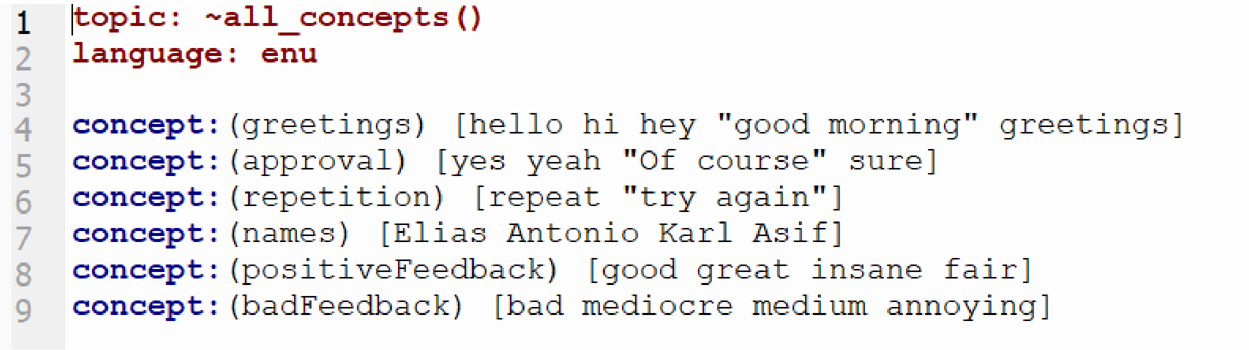


Figure 18 - UC4 Behavior



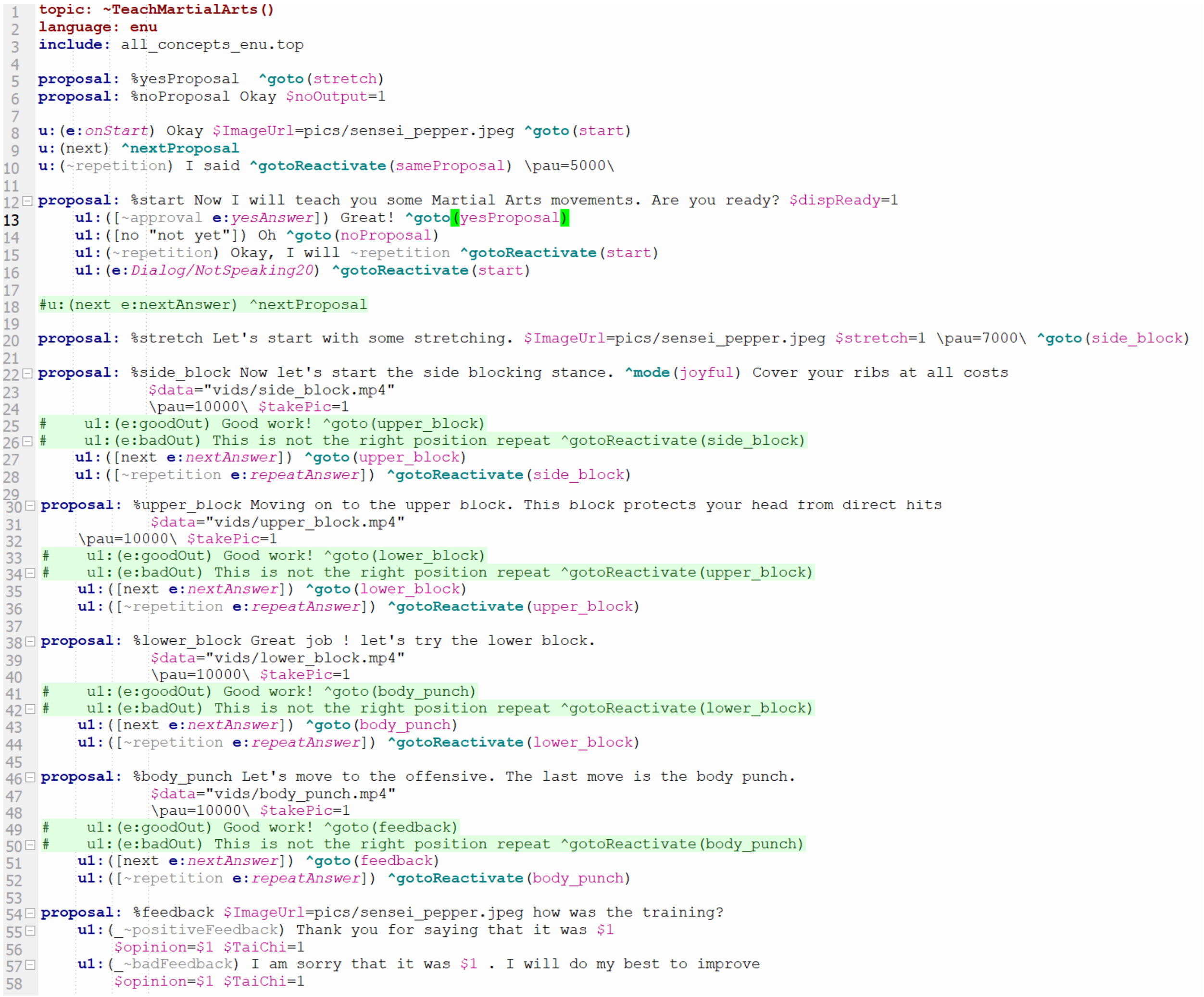
```
topic: ~all_concepts()
language: enu

concept:(greetings) [hello hi hey "good morning" greetings]
concept:(approval) [yes yeah "Of course" sure]
concept:(repetition) [repeat "try again"]
concept:(names) [Elias Antonio Karl Asif]
concept:(positiveFeedback) [good great insane fair]
concept:(badFeedback) [bad mediocre medium annoying]
```

Figure 19 - All Concepts Dialog Box

The all_concepts dialog box in fig.19 has the needed concepts for the last use case.

From the fig.18 behavior flow diagram, we notice that we only have 1 dialog box that manages all the process.

In this use case we display videos, and images dynamically based on their names and relative path. We edit the showImage and playVideo boxes by adding a parameter and editing the onStart type from bang to string.

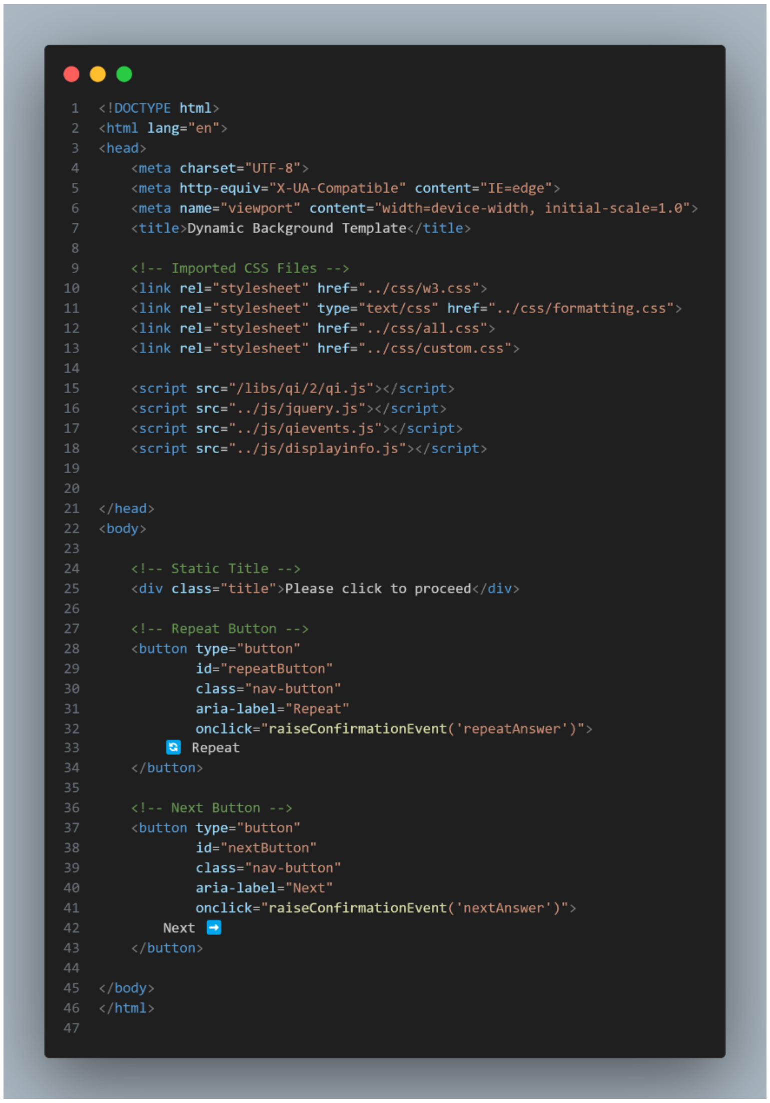


```
topic: ~TeachMartialArts()
language: enu
include: all_concepts_enu.top

proposal: %yesProposal  ^goto(stretch)
proposal: %noProposal Okay $noOutput=1

u:(e:onStart) Okay $ImageUrl=pics/sensei_pepper.jpeg ^goto(start)
u:(next) ^nextProposal
u:(~repetition) I said ^gotoReactivate(sameProposal) \pau=5000\

proposal: %start Now I will teach you some Martial Arts movements. Are you ready? $dispReady=1
    u1:([~approval e:yesAnswer]) Great! ^goto(yesProposal)
    u1:([no "not yet"]) Oh ^goto(noProposal)
    u1:(~repetition) Okay, I will ~repetition ^gotoReactivate(start)
    u1:(e:Dialog/NotSpeaking20) ^gotoReactivate(start)

#u:(next e:nextAnswer) ^nextProposal

proposal: %stretch Let's start with some stretching. $ImageUrl=pics/sensei_pepper.jpeg $stretch=1 \pau=7000\ ^goto(side_block)

proposal: %side_block Now let's start the side blocking stance. ^mode(joyful) Cover your ribs at all costs
            $data="vids/side_block.mp4"
            \pau=10000\ $takePic=1
#    u1:(e:goodOut) Good work! ^goto(upper_block)
#    u1:(e:badOut) This is not the right position repeat ^gotoReactivate(side_block)
    u1:([next e:nextAnswer]) ^goto(upper_block)
    u1:([~repetition e:repeatAnswer]) ^gotoReactivate(side_block)

proposal: %upper_block Moving on to the upper block. This block protects your head from direct hits
            $data="vids/upper_block.mp4"
    \pau=10000\ $takePic=1
#    u1:(e:goodOut) Good work! ^goto(lower_block)
#    u1:(e:badOut) This is not the right position repeat ^gotoReactivate(upper_block)
    u1:([next e:nextAnswer]) ^goto(lower_block)
    u1:([~repetition e:repeatAnswer]) ^gotoReactivate(upper_block)

proposal: %lower_block Great job ! let's try the lower block.
            $data="vids/lower_block.mp4"
            \pau=10000\ $takePic=1
#    u1:(e:goodOut) Good work! ^goto(body_punch)
#    u1:(e:badOut) This is not the right position repeat ^gotoReactivate(lower_block)
    u1:([next e:nextAnswer]) ^goto(body_punch)
    u1:([~repetition e:repeatAnswer]) ^gotoReactivate(lower_block)

proposal: %body_punch Let's move to the offensive. The last move is the body punch.
            $data="vids/body_punch.mp4"
            \pau=10000\ $takePic=1
#    u1:(e:goodOut) Good work! ^goto(feedback)
#    u1:(e:badOut) This is not the right position repeat ^gotoReactivate(body_punch)
    u1:([next e:nextAnswer]) ^goto(feedback)
    u1:([~repetition e:repeatAnswer]) ^gotoReactivate(body_punch)

proposal: %feedback $ImageUrl=pics/sensei_pepper.jpeg how was the training?
    u1:(_~positiveFeedback) Thank you for saying that it was $1
        $opinion=$1 $TaiChi=1
    u1:(_~badFeedback) I am sorry that it was $1 . I will do my best to improve
        $opinion=$1 $TaiChi=1
```

Figure 20 - Teach Martial Arts Dialog Box

Here is the script of teaching karate. We choose 4 stances to teach: body punch, lower block, upper block, and side block. After the Play video box ends, its triggers the webview **nextrepeat** that uses the custom css template. It has a dynamic background image as well as 2 buttons. The next and repeat buttons ensure that the user is guided through the process.

The new next repeat HTML page is available in fig.21.



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dynamic Background Template</title>

    <!-- Imported CSS Files -->
    <link rel="stylesheet" href="../css/w3.css">
    <link rel="stylesheet" type="text/css" href="../css/formatting.css">
    <link rel="stylesheet" href="../css/all.css">
    <link rel="stylesheet" href="../css/custom.css">

    <script src="/libs/qi/2/qi.js"></script>
    <script src="../js/jquery.js"></script>
    <script src="../js/qievents.js"></script>
    <script src="../js/displayinfo.js"></script>


</head>
<body>

    <!-- Static Title -->
    <div class="title">Please click to proceed</div>

    <!-- Repeat Button -->
    <button type="button"
            id="repeatButton"
            class="nav-button"
            aria-label="Repeat"
            onclick="raiseConfirmationEvent('repeatAnswer')">
        🔄 Repeat
    </button>

    <!-- Next Button -->
    <button type="button"
            id="nextButton"
            class="nav-button"
            aria-label="Next"
            onclick="raiseConfirmationEvent('nextAnswer')">
        Next ➡️
    </button>

</body>
</html>

```

Figure 21 - Next Repeat HTML page

We are using the qievents.js and the custom.css to build the style needed for our web-view.

In fig.18, the left-most python script tests the video name to tell the robot to move its joints in a specific way and fix them for better guidance.

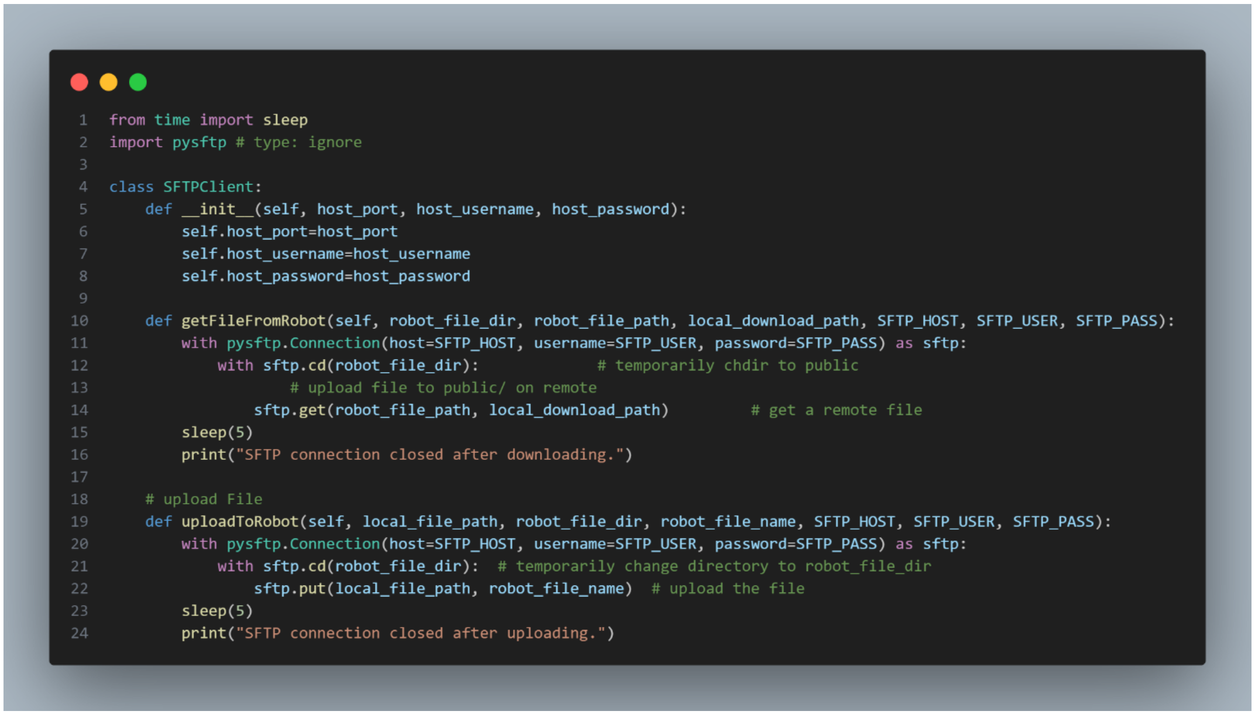
This feature was complete due to its importance relatively to the AI-feedback aspect of the project

An alternative way to proceed is to give feedback to the students to drive the teaching process instead of having the next repeat buttons.

This method is fully developed. Yet, we faced difficulties in setting the delay times and the synchronization of the Choregraph and the Python 3.7 script.

We shall see the details of this exchange shortly.

The Secured File Transfer Protocol allows to send and receive files through a protocol using port 22. This will allow us to send and receive files of type txt and jpg to complete out project



```python
from time import sleep
import pysftp # type: ignore

class SFTPClient:
    def __init__(self, host_port, host_username, host_password):
        self.host_port=host_port
        self.host_username=host_username
        self.host_password=host_password

    def getFileFromRobot(self, robot_file_dir, robot_file_path, local_download_path, SFTP_HOST, SFTP_USER, SFTP_PASS):
        with pysftp.Connection(host=SFTP_HOST, username=SFTP_USER, password=SFTP_PASS) as sftp:
            with sftp.cd(robot_file_dir):             # temporarily chdir to public
                    # upload file to public/ on remote
                sftp.get(robot_file_path, local_download_path)         # get a remote file
        sleep(5)
        print("SFTP connection closed after downloading.")

    # upload File
    def uploadToRobot(self, local_file_path, robot_file_dir, robot_file_name, SFTP_HOST, SFTP_USER, SFTP_PASS):
        with pysftp.Connection(host=SFTP_HOST, username=SFTP_USER, password=SFTP_PASS) as sftp:
            with sftp.cd(robot_file_dir):  # temporarily change directory to robot_file_dir
                sftp.put(local_file_path, robot_file_name)  # upload the file
        sleep(5)
        print("SFTP connection closed after uploading.")
```

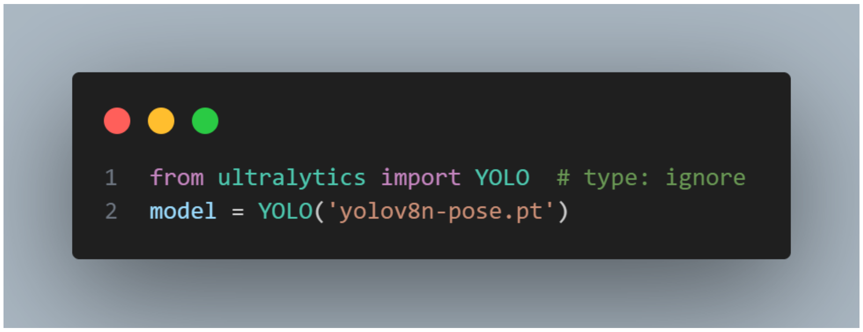
Figure 22 - SFTP Python Code

Using the pysftp and time libraries from python, we build a script to create a class called SFTP Client that has 3 attributes: the host port, the username, and the password.

Hence, we can call the functions getFileFromRobot and uploadToRobot function to write to the file or read the file.

A delay of 5 seconds is a default value for testing. That can be changed at all times to optimize the process.

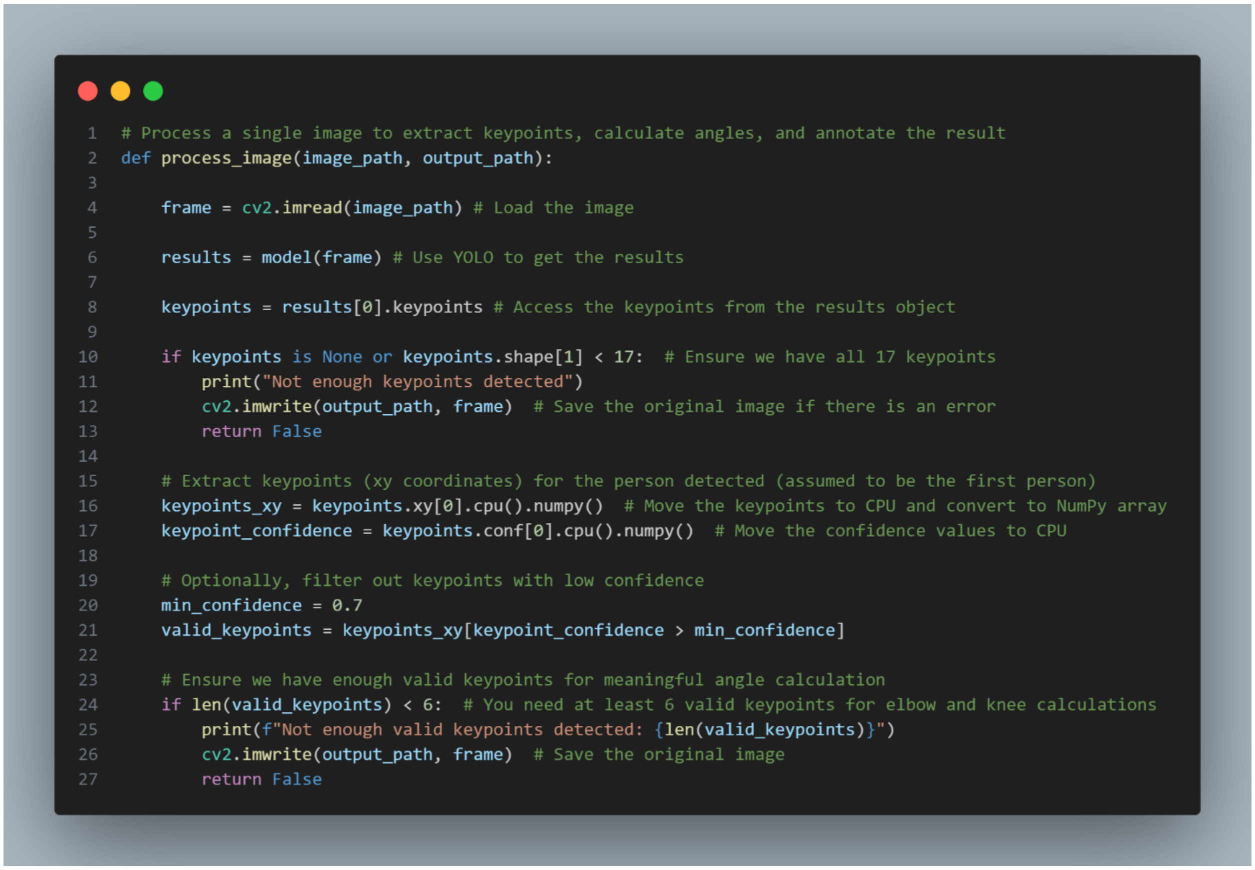
We need to pass to the robot the variable paths and directories that are parameters that change a lot and are not constant for the same robot. Thus, these won't fall into the attributes category.



```python
from ultralytics import YOLO  # type: ignore
model = YOLO('yolov8n-pose.pt')
```

Figure 23 - Model Loading

Moving on from the SFTP part, we focus on the AI computer vision part.

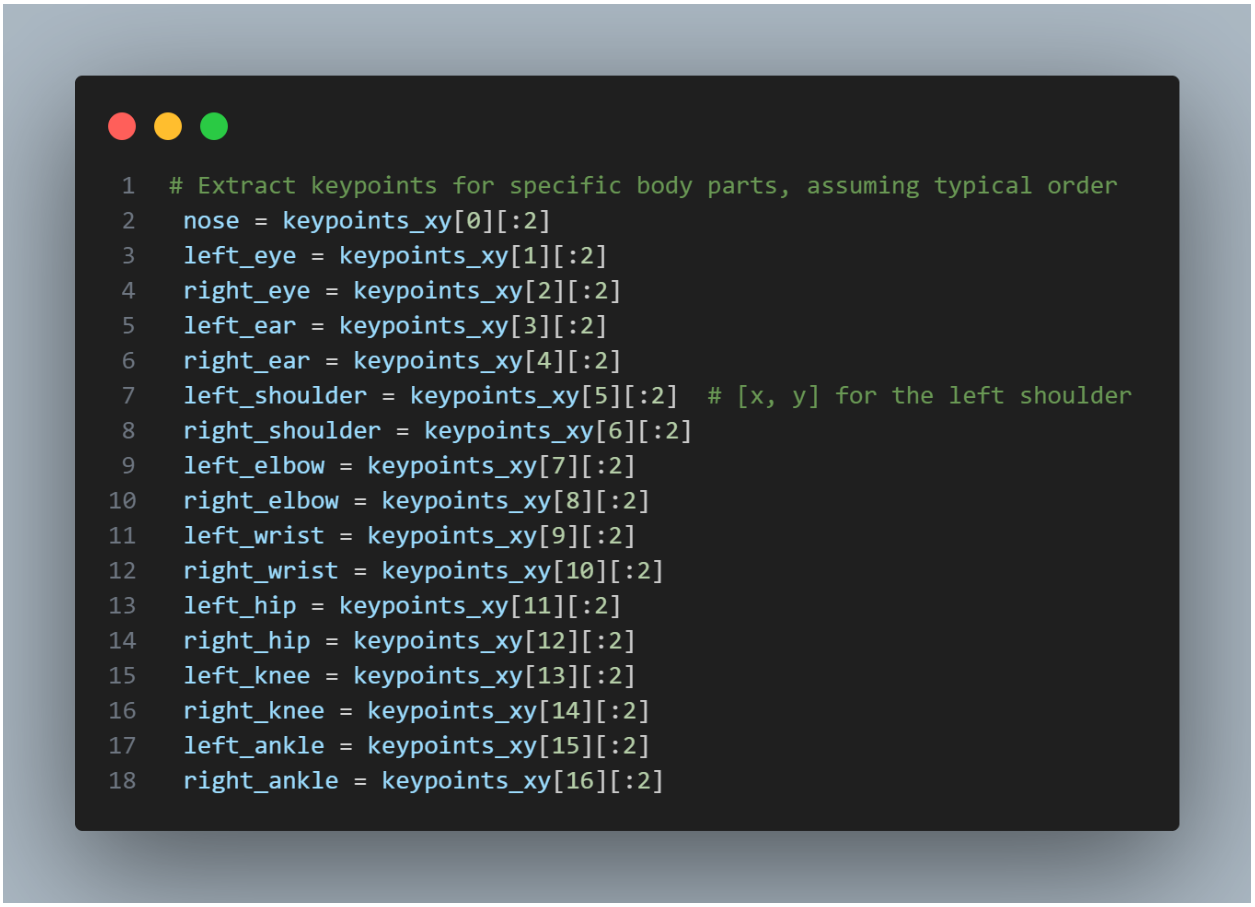


```python
# Process a single image to extract keypoints, calculate angles, and annotate the result
def process_image(image_path, output_path):

    frame = cv2.imread(image_path) # Load the image

    results = model(frame) # Use YOLO to get the results

    keypoints = results[0].keypoints # Access the keypoints from the results object

    if keypoints is None or keypoints.shape[1] < 17:  # Ensure we have all 17 keypoints
        print("Not enough keypoints detected")
        cv2.imwrite(output_path, frame)  # Save the original image if there is an error
        return False

    # Extract keypoints (xy coordinates) for the person detected (assumed to be the first person)
    keypoints_xy = keypoints.xy[0].cpu().numpy()  # Move the keypoints to CPU and convert to NumPy array
    keypoint_confidence = keypoints.conf[0].cpu().numpy()  # Move the confidence values to CPU

    # Optionally, filter out keypoints with low confidence
    min_confidence = 0.7
    valid_keypoints = keypoints_xy[keypoint_confidence > min_confidence]

    # Ensure we have enough valid keypoints for meaningful angle calculation
    if len(valid_keypoints) < 6:  # You need at least 6 valid keypoints for elbow and knee calculations
        print(f"Not enough valid keypoints detected: {len(valid_keypoints)}")
        cv2.imwrite(output_path, frame)  # Save the original image
        return False
```

Figure 24 - Process Image Python Function

Using the YOLOv8 model from Ultalytics, we build a code to load the model and process the image and predict the coordinates of the 17 key points from the feed of the camera on the head of pepper.

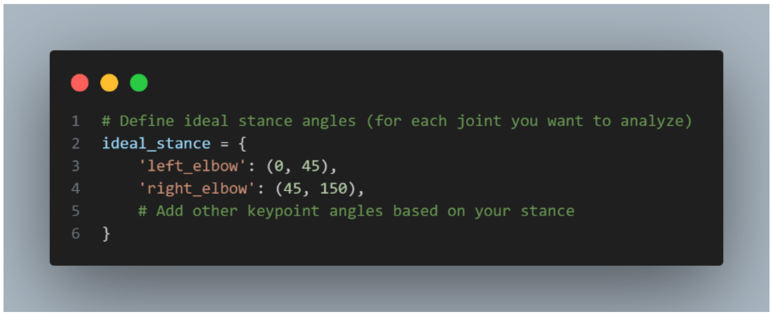
Here are the key points needed. After testing, we notice that each one is fixed to 1 setpoint of the human body.



```python
# Extract keypoints for specific body parts, assuming typical order
 nose = keypoints_xy[0][:2]
 left_eye = keypoints_xy[1][:2]
 right_eye = keypoints_xy[2][:2]
 left_ear = keypoints_xy[3][:2]
 right_ear = keypoints_xy[4][:2]
 left_shoulder = keypoints_xy[5][:2]  # [x, y] for the left shoulder
 right_shoulder = keypoints_xy[6][:2]
 left_elbow = keypoints_xy[7][:2]
 right_elbow = keypoints_xy[8][:2]
 left_wrist = keypoints_xy[9][:2]
 right_wrist = keypoints_xy[10][:2]
 left_hip = keypoints_xy[11][:2]
 right_hip = keypoints_xy[12][:2]
 left_knee = keypoints_xy[13][:2]
 right_knee = keypoints_xy[14][:2]
 left_ankle = keypoints_xy[15][:2]
 right_ankle = keypoints_xy[16][:2]
```

Figure 25 – Key points Extractions & Assignment

Using this information, we can set a setpoint to compare the angles to.

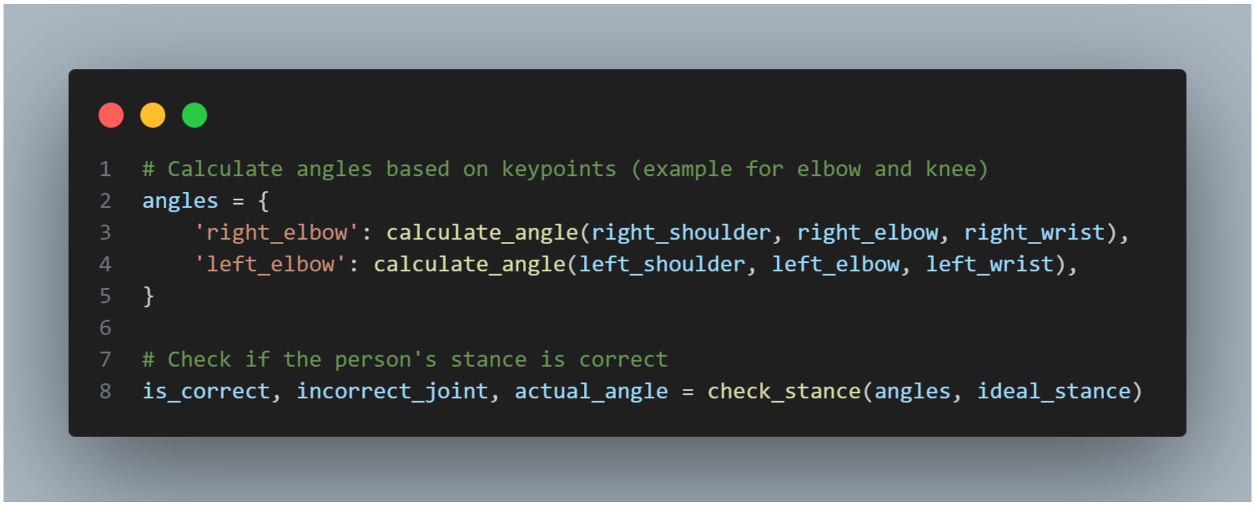


```python
# Define ideal stance angles (for each joint you want to analyze)
ideal_stance = {
    'left_elbow': (0, 45),
    'right_elbow': (45, 150),
    # Add other keypoint angles based on your stance
}
```

Figure 26 - Ideal Stance

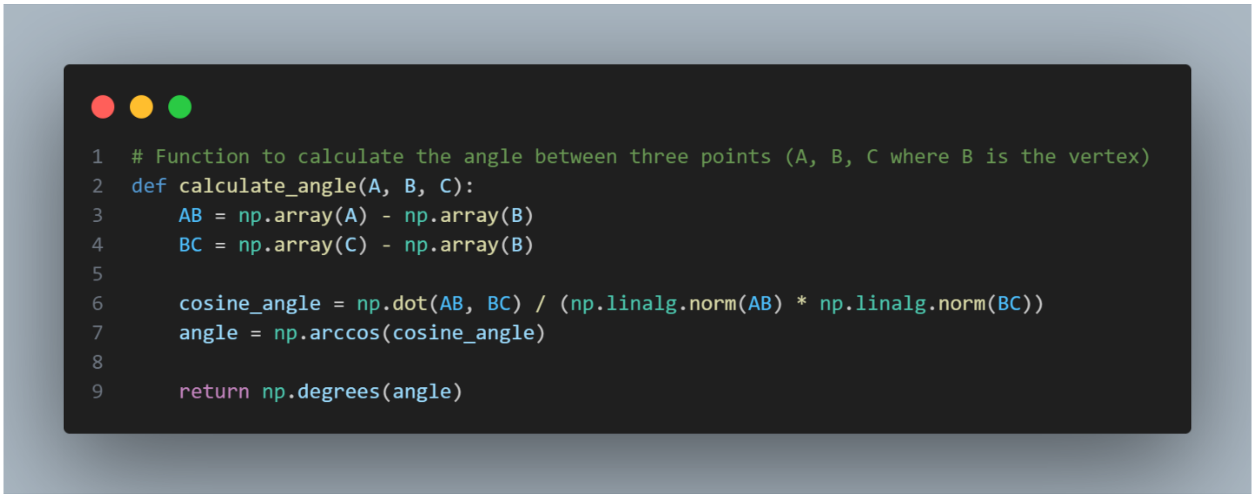
We set which joints we need to work on. Thus, we set the joints needed by attributing 3 adjacent key points to one angle.

The selected combinations are saved as “angles” and passed to the check_stance function to calculate the angle and compare it to the acceptable threshold in fig.26.



```python
# Calculate angles based on keypoints (example for elbow and knee)
angles = {
    'right_elbow': calculate_angle(right_shoulder, right_elbow, right_wrist),
    'left_elbow': calculate_angle(left_shoulder, left_elbow, left_wrist),
}

# Check if the person's stance is correct
is_correct, incorrect_joint, actual_angle = check_stance(angles, ideal_stance)
```

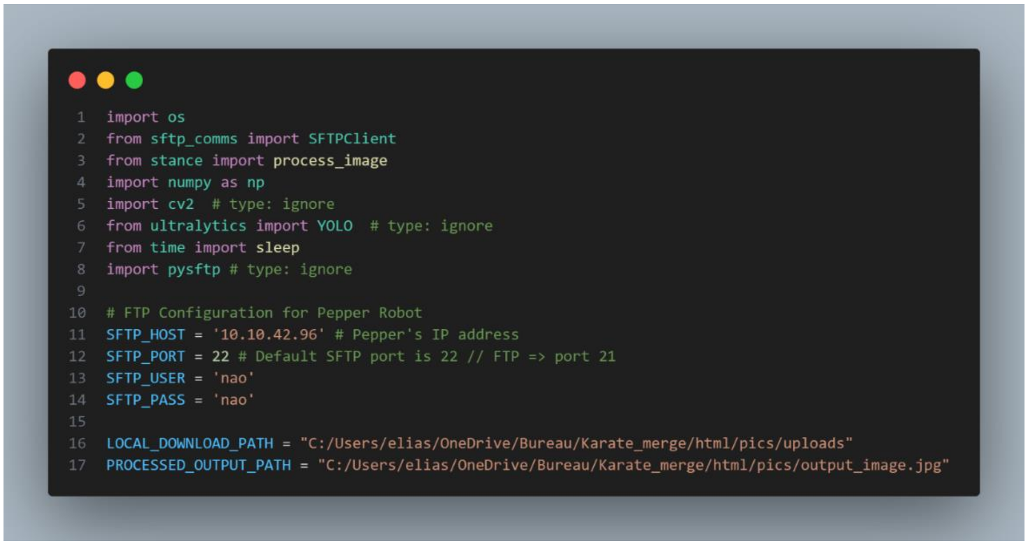
Figure 27 - Studied Joint



```python
# Function to calculate the angle between three points (A, B, C where B is the vertex)
def calculate_angle(A, B, C):
    AB = np.array(A) - np.array(B)
    BC = np.array(C) - np.array(B)

    cosine_angle = np.dot(AB, BC) / (np.linalg.norm(AB) * np.linalg.norm(BC))
    angle = np.arccos(cosine_angle)

    return np.degrees(angle)
```

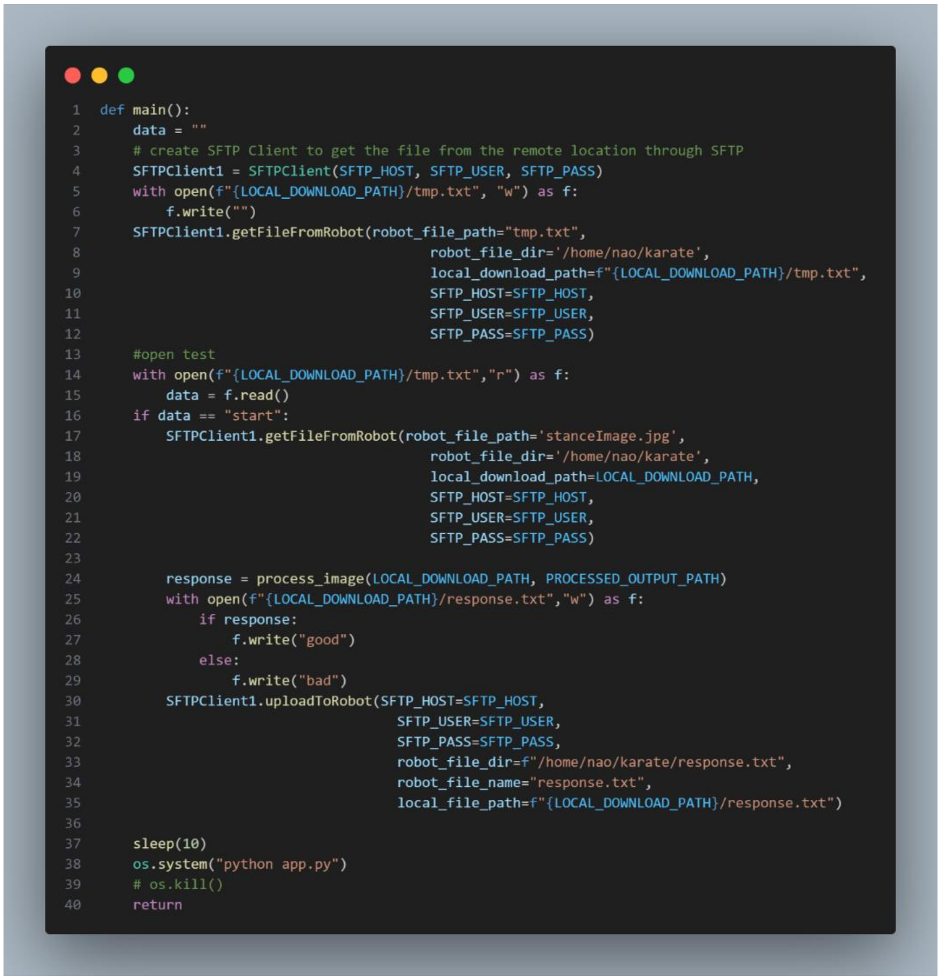
Figure 28 - Calculate Angle Python Function

The fig.28 function is responsible for the angle calculation.



```python
import os
from sftp_comms import SFTPClient
from stance import process_image
import numpy as np
import cv2  # type: ignore
from ultralytics import YOLO  # type: ignore
from time import sleep
import pysftp # type: ignore

# FTP Configuration for Pepper Robot
SFTP_HOST = '10.10.42.96' # Pepper's IP address
SFTP_PORT = 22 # Default SFTP port is 22 // FTP => port 21
SFTP_USER = 'nao'
SFTP_PASS = 'nao'

LOCAL_DOWNLOAD_PATH = "C:/Users/elias/OneDrive/Bureau/Karate_merge/html/pics/uploads"
PROCESSED_OUTPUT_PATH = "C:/Users/elias/OneDrive/Bureau/Karate_merge/html/pics/output_image.jpg"
```

Figure 29 - Python Main Script pt.1



```python
def main():
    data = ""
    # create SFTP Client to get the file from the remote location through SFTP
    SFTPClient1 = SFTPClient(SFTP_HOST, SFTP_USER, SFTP_PASS)
    with open(f"{LOCAL_DOWNLOAD_PATH}/tmp.txt", "w") as f:
        f.write("")
    SFTPClient1.getFileFromRobot(robot_file_path="tmp.txt",
                                        robot_file_dir='/home/nao/karate',
                                        local_download_path=f"{LOCAL_DOWNLOAD_PATH}/tmp.txt",
                                        SFTP_HOST=SFTP_HOST,
                                        SFTP_USER=SFTP_USER,
                                        SFTP_PASS=SFTP_PASS)
    #open test
    with open(f"{LOCAL_DOWNLOAD_PATH}/tmp.txt","r") as f:
        data = f.read()
    if data == "start":
        SFTPClient1.getFileFromRobot(robot_file_path='stanceImage.jpg',
                                        robot_file_dir='/home/nao/karate',
                                        local_download_path=LOCAL_DOWNLOAD_PATH,
                                        SFTP_HOST=SFTP_HOST,
                                        SFTP_USER=SFTP_USER,
                                        SFTP_PASS=SFTP_PASS)

        response = process_image(LOCAL_DOWNLOAD_PATH, PROCESSED_OUTPUT_PATH)
        with open(f"{LOCAL_DOWNLOAD_PATH}/response.txt","w") as f:
            if response:
                f.write("good")
            else:
                f.write("bad")
        SFTPClient1.uploadToRobot(SFTP_HOST=SFTP_HOST,
                                    SFTP_USER=SFTP_USER,
                                    SFTP_PASS=SFTP_PASS,
                                    robot_file_dir=f"/home/nao/karate/response.txt",
                                    robot_file_name="response.txt",
                                    local_file_path=f"{LOCAL_DOWNLOAD_PATH}/response.txt")

    sleep(10)
    os.system("python app.py")
    # os.kill()
    return
```

Figure 30 - Python Main Script pt.2

In fig. 29 and 30, we display the logic needed to complete this design.

We import the libraries needed, set the constants and proceed to the main function.

We initialize the data as an empty string and the tmp.txt file at each iteration.

We will explain the functionality of this code briefly here before diving in the flow chart that connects everything discussed.

The choregraph script has the start flag python script which writes to the tmp.txt file to start the process. It clears the value after a slight delay and looks for the response in response.txt (fig.18)

The code continuously reads from the file to look for the word start and clears the value instantly until its set again.

When it finds the word “start” it calls the function to get the image from the robot and process the image using the AI computer vision code discussed earlier. The process image returns a true or false value based on the stance being correct or wrong,

The returned value dictates what to write in the response.txt file that would be processed in the choregraph UC4 fig.18 behavior.

Here is a recap of the process.

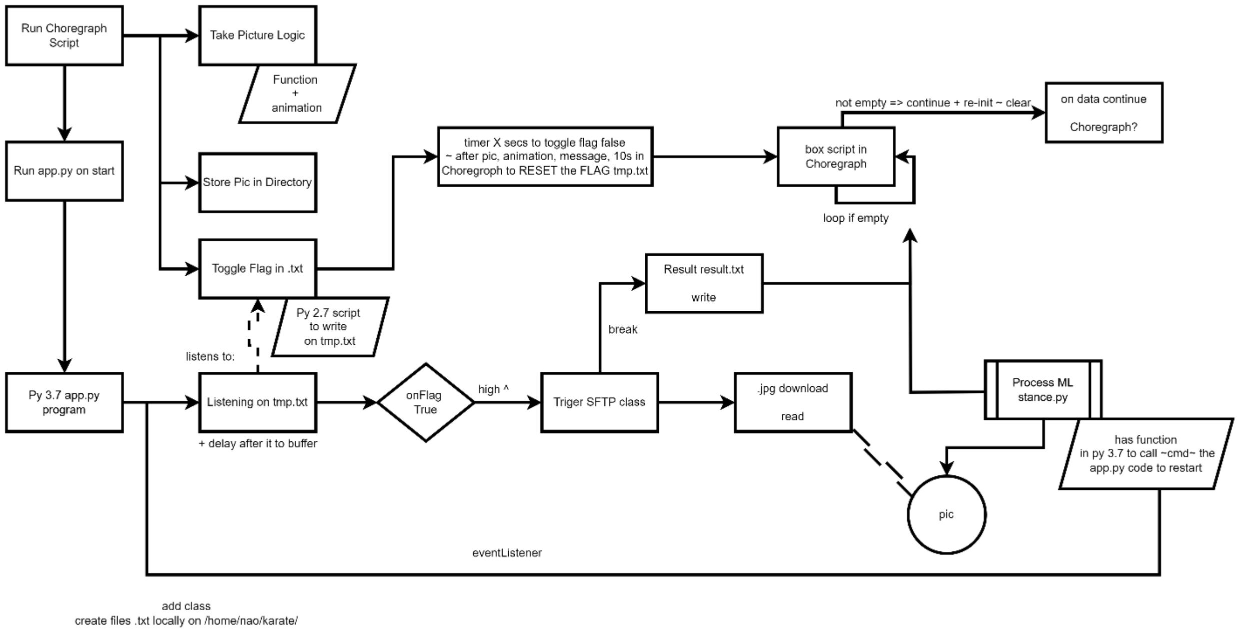

Run Choregraph Script
Take Picture Logic
Function + animation
Run app.py on start
Store Pic in Directory
Toggle Flag in .txt
Py 2.7 script to write on tmp.txt
listens to:
timer X secs to toggle flag false ~ after pic, animation, message, 10s in Choregroph to RESET the FLAG tmp.txt
box script in Choregraph
not empty => continue + re-init ~ clear
on data continue Choregraph?
loop if empty
Result result.txt write
break
Py 3.7 app.py program
Listening on tmp.txt
+ delay after it to buffer
onFlag True
high ^
Triger SFTP class
.jpg download read
Process ML stance.py
has function in py 3.7 to call ~cmd~ the app.py code to restart
pic
eventListener
add class
create files .txt locally on /home/nao/karate/

Figure 31 - System Flowchart

Unfortunately, this script could not be completed on time since the project has its challenges. The design is complete. The only missing step is the delay optimization, synchronization, and testing.

Challenges faced

Asif’s Challenges

I didn’t really face many challenges as my part was straight to the point and easy to implement, the lack of qi chat reliable documentation and code examples was frustrating at times, the app and pepper were buggy at times, and different versions of the codes worked and didn’t work due to lack of support too. major issues or tricks I faced while doing the use case would be to make history sections interactive and fun, and not only storytelling, adding emotions, speech tones and variations alongside different animations while keeping it simple and concise too.

Antonio’s Challenges

No significant challenges were encountered. Most obstructions could be classified as minor inconveniences. Accustoming myself to the Qi chat syntax was rather unnatural at first, but this issue was resolved with practice. Also, the task of designing web interfaces was cumbersome to me given that I have no concrete knowledge of web development languages like html and css. This was overcome smoothly with the help of my computer engineering colleagues. Another irritating inconvenience was the erratic behavior of the onStart event listener. The implementation of this functionality would throw random errors upon execution, which would cause me to spend considerable time debugging the program. It was only after multiple trials that I realized this behavior was simply inadvertent and that stopping the program and re-initiating it would resolve the issue. Overall, these challenges provide remarkable insight into the duality of robotics; specifically, it highlights the stark contrast between the enjoyable experience of interacting with the robot and the often tedious and lethargic process of designing it.

Karl’s Challenges

Most of the troubles I faced involved getting accustomed to using Choregraphe. The app would sometimes lag or bug out, but restarting the app or Pepper itself usually fixed the issue. Another challenge was learning QiChat syntax and conventions, as there isn’t much documentation available to rely on. Additionally, the microphones installed on Pepper caused some trouble during testing, but this was easily resolved by manually typing text input into Choregraphe. Overall, I would say I didn’t face many challenges while completing our project. I quickly adapted to the new environment and implemented correct logic where needed. Any inconveniences were minor.

Elias-Charbel’s Challenges

The first challenge I faced would be the integration of the codes together since we had to connect the behaviors using the “Run Behavior” Box. We were forced to use the onStopped variable instead of being able to send outputs.

This forced our code to have a sequential path on the micro and macro scales which means that we could not have the option to jump from the UC1-2 behavior to the UC4 if the user wanted to skip the history block.

I tried to load the onStopped variable as a String with a word that I would test in a python script. This did not work and the parameter passed to the onInput_onStart of the python script showed to have no type. I changed the type from bang to string to Boolean to integer. In all cases, its type could not be recognized.

Another issue would be the lack of compatibility of python 3 codes with pepper. This forced us to run the AI script on our local machine which, in turn, needed a protocol of communication between the 2 machines. Hence, we opt for the SFTP communication protocol.

In addition, I tried to build an html page with an option to display an image or video dynamically with the title to be displayed with 2 fixed buttons next and repeat. This page did not work since we can’t pass more than 1 parameter to the webview and debug messages displayed clearly than 1 parameter was expected instead of 4.

Conclusion

This experience has been a marvelous one. Working on this project and combining all these technologies helped us integrate many previously learned concepts.

The issue of time is an issue that we would always face in life. The project would have a better alternative result with the AI implementation.

That is why we set it as a future plan of ours to complete this AI – Pepper integration.

Links

GIN456_Final_Video.mp4

https://github.com/eliascharbelsalameh/GIN456-Martial_Arts

https://github.com/eliascharbelsalameh/GIN456-Martial_Arts-AI