

1. Home .....	2
1.1   Background Description .....	4
1.2   Personas .....	7
1.3   Requirements .....	10
1.4   Development Environment .....	12
1.4.1 Previous Projects - Setup & Tests .....	13
1.4.1.1 General Remarks .....	14
1.4.1.2 Bluering .....	15
1.4.1.3 Boxjelly .....	16
1.4.1.4 Redback .....	17
1.4.2   Coding Standards .....	18
1.4.3   Docker - Overview .....	19
1.4.4   Developer Tools .....	20
1.4.4.1 IDE .....	21
1.4.4.2 Develop inside Containers .....	22
1.4.4.3 Test & Debug .....	24
1.4.5   Installation Guide .....	25
1.4.5.1 Ubuntu .....	26
1.4.5.2 Choregraphe .....	27
1.4.5.3 Docker .....	28
1.4.5.4 Launch the Project .....	29
1.5   Development Features .....	31
1.5.1 Sprint 2 Features .....	32
1.5.1.1 AI-Generated Image .....	33
1.5.1.2 Filtering & Moderation .....	34
1.5.1.3 Migration: ROS1 to ROS2 .....	35
1.5.1.4 Struggle letter Module .....	38
1.5.2 Code Review .....	40
1.5.2.1 Sprint 2 .....	41
1.6   Sprint Ceremonies .....	42
1.6.1   Sprint Planning .....	43
1.6.1.1 Sprint1: Inception Planning .....	44
1.6.1.1.1 Assign the Story Points .....	46
1.6.1.2 Sprint2: Development .....	47
1.6.1.3 Sprint3: Development .....	50
1.6.2   Sprint Review .....	52
1.6.2.1 Sprint 1 Review .....	53
1.6.2.2 Sprint 2 Review .....	54
1.6.3   Sprint Retrospective .....	55
1.6.3.1 Sprint 1 Retrospective .....	56
1.6.3.2 Sprint 2 Retrospective .....	57
1.7   Ethical Considerations .....	59
1.8   Cyber Security Analysis .....	61
1.9   Appendix .....	63
1.10   Archives .....	64
1.10.1 Review previous projects .....	65

# Home

## Project Overview: ChatGPT and NAO Robot



Image source: [https://github.com/CHRI-Lab/cowriter\\_letter\\_learning](https://github.com/CHRI-Lab/cowriter_letter_learning)

Composed of a set of ROS nodes that facilitate the user interaction with a robot, the CoWriter project is designed for children to teach a social robot handwriting. The intuition behind this interaction is "learning by teaching" where children can have the acquisition of handwriting by teaching a robot to write better. Nao is an autonomous, programmable humanoid robot and has been used for demonstration purposes for this project. Last semester, three different teams from the Software Project subject (COMP90082) took over this project with goals to 1) update the CoWriter project from Python2 to Python3 and 2) integrate ChatGPT to enable NAO robot to have conversations with children. Currently, these projects are each deployed in different environments and have achieved slightly different outcomes.

This project aims to **1) merge the previous projects in one unified environment, 2) add further enhancements to the CoWriter project and NAO's conversation capabilities, and 3) implement a logging and annotation module for data management purposes.**

## Team NA-Redback

Name (Preferred Name)	Role	Contact	Bio
Wafa Johal	Client	wafa.johal@unimelb.edu.au	Dr Wafa Johal is a senior lecturer at the School of Computing & Information Systems, Faculty of Engineering and Information Technology, University of Melbourne. Her research focuses on human-robot/computer interaction and intelligent and autonomous systems.
Sebastian Bobadilla	Supervisor	bobadillacha@unimelb.edu.au	Sebastian is part of the teaching team for COMP90082 Software Project subject and is a supervisor for the NA-Redback team who oversees the progress of this project.
Eunji Kim (Rachel)	Product Owner, Dev Member	kimek@student.unimelb.edu.au	Rachel is a final-year master's student in IT specialising in AI at the University of Melbourne. Her interests lie in data analytics, natural language processing, and human-computer interaction.

## Recent space activity



**Chien-Pu LIN**

| Sprint Retrospective updated 3 minutes ago • [view change](#)



**Aurelien Plaire**

Sprint2: Development updated 12 minutes ago • [view change](#)



**Difan WU**

Sprint2: Development updated 20 minutes ago • [view change](#)

Sprint1: Inception Planning updated 20 minutes ago • [view change](#)

| Requirements updated 21 minutes ago • [view change](#)

## Space contributors

- [Chien-Pu LIN](#) (3 minutes ago)
- [Aurelien Plaire](#) (12 minutes ago)
- [Difan WU](#) (20 minutes ago)
- [EUNJI KIM](#) (37 minutes ago)
- [Yangchen Shen](#) (2 hours ago)
- ...

Difan Wu	Scrum Master, Dev Member	<a href="mailto:difanw@student.unimelb.edu.au">difanw@student.unimelb.edu.au</a>	Difan Wu is a Master of IT - Artificial Intelligence student at the University of Melbourne. Originally trained as an architect, his innate curiosity, determination and adaptability led him to embrace a new challenge in the world of IT. Currently, he is focusing on large data processing, machine learning and computer vision.
Aurelien Plaire	Tech Lead	<a href="mailto:aplaire@student.unimelb.edu.au">aplaire@student.unimelb.edu.au</a>	Aurélien is a former general engineering student at CentraleSupélec, completing a double degree at the University of Melbourne (Master of IT - Computing).  His interests lie in data processing and management, and software development.
Chien-Pu Lin (Jeff)	Test Lead	<a href="mailto:chienpu.lin@student.unimelb.edu.au">chienpu.lin@student.unimelb.edu.au</a>	Jeff is a Master of IT student, specializing in AI, at the University of Melbourne. His interests span AI, machine learning, cloud computing and robotics. He is eager to dive into cutting-edge technologies and innovations.
Yangchen Shen (Shen)	Quality Lead	<a href="mailto:yangchens2@student.unimelb.edu.au">yangchens2@student.unimelb.edu.au</a>	Shen is a Master of IT student at the University of Melbourne. His interests lie in natural language processing, data processing and management, and software development.

## Links to our Project

- [Trello board](#)
- [Github repository](#)

# | Background Description

## Project Background



Image source: [https://github.com/CHRI-Lab/cowriter\\_letter\\_learning](https://github.com/CHRI-Lab/cowriter_letter_learning)

Composed of a set of ROS nodes that facilitate the user interaction with a robot, the **CoWriter project** is designed for children to teach a social robot handwriting. The intuition behind this interaction is "learning by teaching" where children can have the acquisition of handwriting by teaching a robot to write better. When words are requested by displaying cards to the robot with **chilitags** on them, children then correct the robot's simulated handwriting by demonstrating on a tablet. This demonstration is subsequently used to improve the robot's shape learning algorithm. NAO is an autonomous, programmable humanoid robot and has been used for demonstration purposes for this project.

Last semester, three different teams from the Software Project subject (COMP90082) took over this project with goals to 1) update the CoWriter project from Python2 to Python3 and 2) integrate ChatGPT to enable NAO robot to have conversations with children. Currently, these projects are each deployed in different environments and have achieved slightly different outcomes.

References for the previous projects including code and documents can be found in the links below (please email the dev team for access):

- NAOHW-RedBack
- NAOHW-BlueRing
- NAOHW-Boxjelly

## Project Summary

This project, "ChatGPT and NAO Robot (Code: NA)", aims to **1) merge previous projects in one unified environment, 2) add further enhancements to the CoWriter project and NAO's conversation capabilities, and 3) implement a logging and annotation module for data management purposes.**

## Client Goals

Our client is **Dr Wafa Johal**, who is a senior lecturer at the School of Computing & Information Systems, Faculty of Engineering and Information Technology, University of Melbourne. Her research focuses on human-robot/computer interaction and intelligent and autonomous systems. Through the CoWriter Project, she aims to examine the application of the learning by teaching on motivating children to practice their handwriting skills accompanied with a humanoid robot (NAO).

The client goals can be largely divided into four:

1. Integration of previous projects in a single repository
2. Improvement of the CoWriter software
  - Optimise the learning process by being able to identify letters the children is struggling with and control the difficulty accordingly
  - Update UI:
    - Make the UI more user-friendly by integrating kids-friendly illustrations
    - Move to a web version (browser) from PyQt
    - Investigate possibility to have a cross platform UI such as iOS – apple pencil and Wacom (Out-of-scope)
3. Refinement of conversation using ChatGPT
  - Ensure that the conversation stays child-friendly
  - Enhance the interaction by incorporating motions/gestures to NAO robot during conversation (using the existing motion library)
  - Generate the code for new motion based on the text (Out-of-scope)
4. Upgrade ROSbag data management
  - Implement logging and annotation module

## Scope

### In Scope

- Integration of previous projects in a single repository
- Optimise the learning process by being able to identify letters the children is struggling with and control the difficulty accordingly
- Update the CoWriter UI by
  - moving to a web version (browser)
  - adding generated illustration
- Use ChatGPT to
  - maintain child-friendly and safe conversation between a robot and children
  - elicit motions/gestures to NAO robot during conversation (using the existing motion library)
- Implement logging and annotation module

### Out of Scope

- Investigate possibility to have a cross platform UI such as iOS – apple pencil and Wacom
- Use GPT to generate the code for new motion based on the text

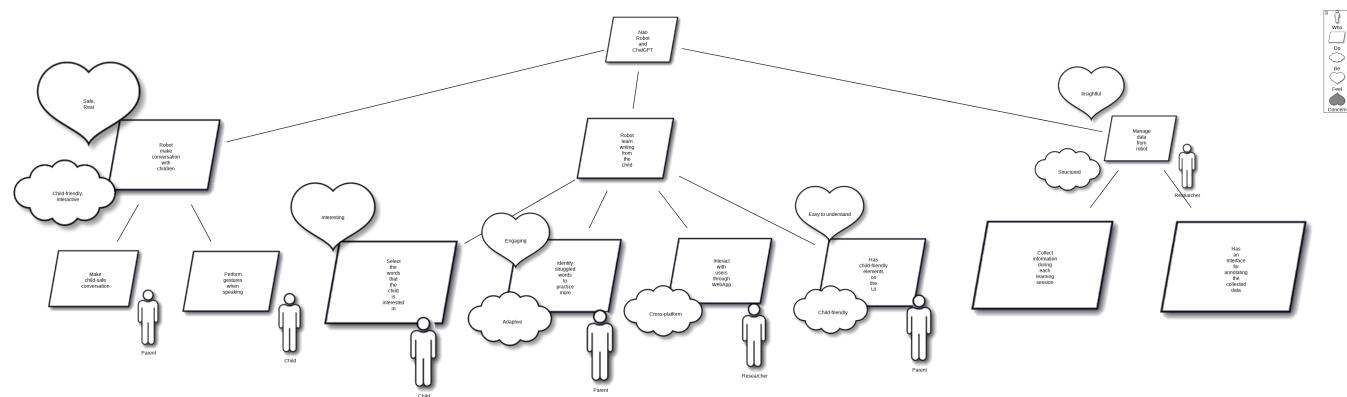
## Stakeholders

Role	Internal/External	Engagement	Interests
Client	Internal	High	High
Development Team	Internal	High	High
Children	External	High	Medium
Researchers	External	Medium	Medium
Parents	External	Low	High

## DO-BE-FEEL

WHO	DO	BE	FEEL
Child	Robot make conversation with children	Structured	Safe
Parent	Robot learn writing from the child	Child-friendly	Interesting
Researcher	Manage data from robot	Interactive	Engaging
	Make child-safe conversation	Adaptive	Real
	Perform gestures when speaking	Cross-platform	Easy to understand
	Select the words that the child is interested in		Insightful
	Identify struggled words to practice more		
	Has child-friendly elements on the UI		
	Interact with users through WebApp		
	Collect information during each learning session		
	Has an interface for annotating the collected data		

## Goal Model

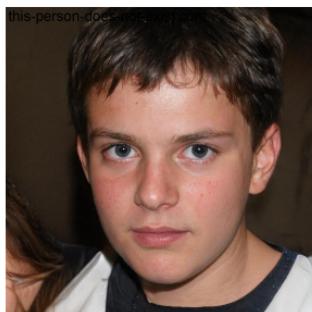


Link: <https://momo-comp90082.eresearch.unimelb.edu.au>

# Personas

Persona	Titles	Personality	Goals
Alex	Child	<ul style="list-style-type: none"> <li>• Introvert</li> <li>• Creative</li> <li>• Fickle</li> <li>• Curious</li> </ul>	<ul style="list-style-type: none"> <li>• Fun and Play</li> <li>• Exploration</li> <li>• Positive Reinforcement</li> </ul>
Lily	Child	<ul style="list-style-type: none"> <li>• Extrovert</li> <li>• Analytical</li> <li>• Fickle</li> <li>• Passive</li> </ul>	<ul style="list-style-type: none"> <li>• Variety and Flexibility</li> <li>• Positive Reinforcement</li> <li>• Independence</li> </ul>
Emily	Parent	<ul style="list-style-type: none"> <li>• Introvert</li> <li>• Thinking</li> <li>• Intuition</li> <li>• Perceiving</li> </ul>	<ul style="list-style-type: none"> <li>• Convenience</li> <li>• Engaging Activities</li> <li>• Quality Education</li> <li>• Technology Integration</li> </ul>
Olivia	Researcher	<ul style="list-style-type: none"> <li>• Extrovert</li> <li>• Creative</li> <li>• Loyal</li> <li>• Active</li> </ul>	<ul style="list-style-type: none"> <li>• Cutting-Edge Tools</li> <li>• Effective Learning Solutions</li> <li>• Data-Driven Insights</li> </ul>
Alan	Researcher	<ul style="list-style-type: none"> <li>• Extrovert</li> <li>• Creative</li> <li>• Easy</li> <li>• Active</li> </ul>	<ul style="list-style-type: none"> <li>• Real-World Application</li> <li>• Adaptive Learning</li> <li>• Data-Driven Insight</li> </ul>

## Alex



**"Explore, Play, and Write Your Way to Success with Our Robot Learning Companion!"**

Age: 6  
Work: Student  
Family: Father and Mother  
Location: Perth  
Character: Active

Organized      Practical  
Protective      Hardworking

### Bio

Alex is a 6-year-old boy with a curious and energetic personality. He is currently in kindergarten and is at the stage of learning to write his letters. Alex loves to explore the world around him and is fascinated by all things that move, make noise, and light up. He enjoys spending time with his friends, playing games, and drawing.

### Personality



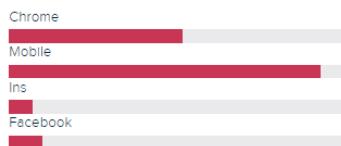
### Goals

- **Fun and Play:** Alex values activities that are fun, interactive, and capture his imagination.
- **Exploration:** He is naturally curious and loves to try new things, especially if they involve hands-on experiences.
- **Positive Reinforcement:** He thrives on positive feedback and encouragement, which motivates him to keep trying and learning.

### Needs & Pain Points

- **Engaging Learning:** Alex is looking for ways to learn that feel like play, where he can have fun while acquiring new skills.
- **Visual and Interactive:** He is drawn to activities that are visually appealing and involve interactive elements like touchscreens or buttons.
- **Encouragement:** He benefits from tools that provide positive reinforcement and celebrate his achievements.

### Preferred Channels



# Lily

Age: 8  
Work: Student  
Family: Parent  
Location: Seattle, WA

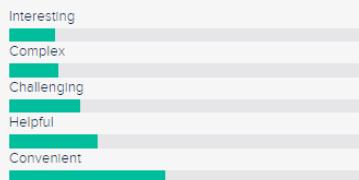
## Bio

Lily is a 8-year-old girl. She has a vibrant and imaginative personality and enjoys spending time exploring creative activities and hobbies. Lily is known for her artistic talents and has a natural ability to come up with unique and imaginative ideas. However, when it comes to traditional studies and academic subjects, Lily often finds it challenging to focus and maintain her interest.

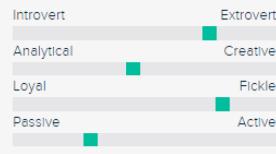
## Pain Points & Challenge

- Lack of Focus:** Lily often struggles to concentrate on one task for an extended period, leading to difficulty in completing assignments or studying for tests.
- Traditional Study Methods:** She becomes easily bored with traditional study methods that involve reading textbooks or memorizing information.
- Frustration with Routine:** Lily finds it frustrating to stick to rigid schedules or routines, as they can dampen her creative spirit and sense of exploration.
- Perceived Failure:** Lily's low tolerance for activities that she doesn't enjoy can lead to feelings of failure or inadequacy when it comes to academics.

## Motivation



## Personality



Naughty      Unlearning  
Independence



*"Unlock Your Creative Power: Learning Made Engaging, Personal, and Fun!"*

# Emily



Busy      Patient      High hopes for her child

## Bio

Emily is a 35-year-old parent of two children, aged 5 and 7. She works as a marketing manager for a tech company and is passionate about providing her children with the best learning experiences. Emily is dedicated to finding innovative and engaging ways to support her children's education and development.

*"Empowering my children's learning journey through interactive and personalized experiences is the key to unlocking their full potential."*

Name: **Emily**  
Gender: **Female**  
Age: **35**  
Work: **Marketing Manager**  
Family: **Married, kids**  
Location: **MEL, VIC**  
Income: **5000AUD per month**

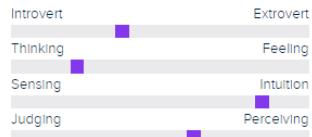
## Needs & Pain points

- Educational Support:** Emily is looking for effective ways to help her children learn essential skills, such as writing letters, outside of the traditional classroom environment.
- Engagement:** She wants her children to be excited about learning and is seeking activities that can capture their attention and maintain their interest.
- Customization:** Emily recognizes that each of her children has unique learning styles and needs, so she values tools that can adapt and provide personalized experiences.
- Transitioning from Play to Learning:** Emily faces the challenge of seamlessly transitioning her children from playtime to focused learning activities, and she seeks tools that can make this transition engaging and enjoyable for her kids.

## Goals

- Convenience:** Emily's busy work schedule makes it important for her to find efficient and convenient ways to support her children's learning outside of school.
- Engaging Activities:** She values activities that are not only educational but also enjoyable and interactive for her kids.
- Quality Education:** Emily believes in the importance of a strong educational foundation for her children and actively seeks out tools and resources to enhance their learning.
- Technology Integration:** As someone working in the tech industry, Emily is comfortable with technology and appreciates its potential to enhance her children's learning experiences.

## Personality



## Dr Olivia Bennett

Age: 50  
Work: Researcher  
Family: Single  
Location: San Jose, CA  
Character: The Computer Nerd



*"Empowering Education through Innovation: Nurturing Minds with Data-Driven Learning Solutions."*

Friendly Clever Go-Getter

### Goals

- Educational Excellence:** Dr. Bennett is deeply committed to advancing the field of education and ensuring that all children have access to high-quality learning experiences.
- Evidence-Based Practices:** She believes in the importance of using research and data to inform educational strategies and create effective learning environments.
- Holistic Learning:** Dr. Bennett values a holistic approach to education, emphasizing not only academic skills but also social, emotional, and cognitive development.

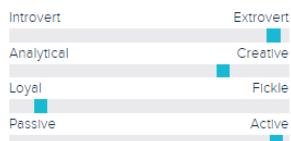
### Bio

Dr. Olivia Bennett is a 50-year-old educational psychologist and researcher with a distinguished career in child development. She holds a Ph.D. in Educational Psychology and has published numerous articles and books on effective teaching methods for young learners. Dr. Bennett is a professor at a renowned university and is dedicated to shaping the future of education through research and innovative practices.

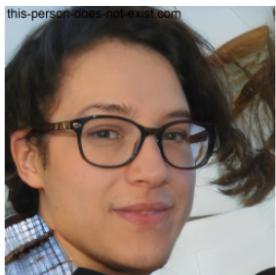
### Needs and Challenges

- Cutting-Edge Tools:** Dr. Bennett is always on the lookout for innovative tools and technologies that can enhance teaching and learning, aligning with her research-oriented mindset.
- Effective Learning Solutions:** She seeks resources that can address the diverse needs of students and provide differentiated instruction to support optimal learning outcomes.
- Data-Driven Insights:** Dr. Bennett needs tools that offer detailed data and insights into students' progress, helping her refine her teaching methods and interventions.

### Personality



## Dr. Alan Roberts



Age: 38  
Work: Researcher In AI  
Family: 3 roommates  
Location: Detroit, MI

*"Elevating Education with AI: Where Innovation and Learning Converge for a Brighter Future."*

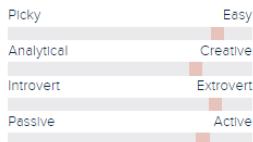
### Goals

- Advancing AI Education:** Dr. Roberts is passionate about pushing the boundaries of AI to enhance educational experiences and create new opportunities for learning.
- Research Excellence:** He is dedicated to conducting rigorous and impactful research that contributes to the advancement of AI and its applications in various domains.
- Interdisciplinary Collaboration:** Dr. Roberts values collaboration with experts from diverse fields, believing that combining AI with other disciplines can lead to transformative breakthroughs.
- Ethical Innovation:** He is committed to developing AI technologies that are ethically responsible and have a positive impact on society.

### Bio

Dr. Alan Roberts is a 38-year-old computer scientist and AI researcher. He holds a Ph.D. in Artificial Intelligence from a prestigious university and has a strong track record of groundbreaking research in machine learning and robotics. Dr. Roberts is known for his innovative thinking and his contributions to the development of cutting-edge AI technologies.

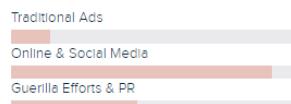
### Personality



### Frustrations

- Real-World Application:** He is interested in AI projects that have practical applications and can address real-world challenges, particularly in the field of education.
- Adaptive Learning:** He is intrigued by the potential of AI to provide personalized and adaptive learning experiences, catering to individual student needs and learning styles.
- Data-Driven Insights:** Dr. Roberts requires access to detailed data and analytics to analyze the effectiveness of AI-driven educational tools and refine their algorithms.

### Preferred Channels



# Requirements

## Prerequisites

Since it is a continued project from last semester and the client requested to review and merge the previous three projects to get started, comprehensive analysis and integration of past projects must be executed before implementing features as outlined in the User Stories below. We've taken into consideration the feedback from our supervisor that this integration process is not formally part of the user stories as it is not directly for the end-users. Accordingly, we have categorized it to prerequisites instead. This is a critical part of the project and can be further broken down into the following requirements:

Requirement	ID	Specification		Priority	Story Point (0-89)	Acceptance Criteria				ChatGPT Generated
						Scenario	Given	When	Then	
Integration of previous projects	CS1	All three projects should be able to run under one unified environment and have no compatibility issues between each chunk of code		Must	47	Client receives the final project	Given that three previous projects have been developed under the same environment/upgraded to the same version	Execute three previous project separately in terminal	The code can run successfully in the new unified environment.	No
	CS2	All three projects should be merged into a single repository with features integrated		Must	72	Client tests the final project	Given that three projects have been merged into one repository	Execute the code as a whole project in terminal	Can run successfully with no errors as a whole project.	No
	CS3	All the merging and integration process should be documented in a structured and organised way		Must	7	Client/New developer review the final project	Given that a clear document shown on Github and each code meet the coding standard and clearly commented	Other new developer review our project and code	They are able to understand the code/function and the project without much effort.	No

## User Stories

Note:

### 1) Priority is categorized as:

- **Must** - essential, non-negotiable features for the project's core functionality and success
- **Should** - not as critical as "Must", but important features that will significantly add up to the project's overall values
- **Could** - supplementary features that can enhance user experience, but can be omitted if sufficient time and resources are not allowed

### 2) Story points are allocated from the Fibonacci scale (0 - 89)

Epic	ID	User Story		Priority	Story Point (0-89)	Justification	Acceptance Criteria				ChatGPT Generated	
							Scenario	Given	When	Then		
Improvement of conversation using ChatGPT	U S1	A s a parent,	I want the robot to have a conversation with my kids in a child-friendly and safe manner	so that my child's interaction stays positive and free from inappropriate content.	Must	29	Ensuring a child-friendly and safe conversation is essential for the children's wellbeing, and keeping them away from potential harm and negative influence	Alex(Child) is communicating with the robot	Given that some prompts are set for the robot's conversation function	An improper topic appears during the conversation	The robot can detect the improper topic/words and start a new topic which is child-friendly.	No
	U S2	A s a child ,	I want the robot to associate a human gesture to its conversation	so that I would feel like talking to a real person.	Could	21	This is an optional feature that will make the interaction between kids and robot more fun and immersive	Alex(Child) want to communicate with robot in a lively way	Given that some new gestures are designed in ROS for the robot	After Alex talking to the robot, during the reply to Alex from the robot	The robot can find the gesture in the list corresponding to the emotion/keyword of the chatGPT-generated words and can move smoothly based on the command, which is designed in ROS.	No
Improvement of CoWriter	U S3	A s a parent,	I want my kids to be able to practice their struggling letters	so that the learning process can be more engaging.	Must	30	This will be a significant upgrade to the CoWriter software and will play a key role to make the writing practice more personalised and effective	Alex(Child) writes some letter badly	Given that some new machine learning methods are used in the robot and the robot is trained with enough data	After Alex writing a letter	The robot can find the bad letters written by Alex and generate some new words which contain those bad letters.	No
	U S4	A s a parent,	I want to add more child-friendly elements to UI such as illustrations	so it helps kids to easily understand and remember the words	Could	34	This is an extra feature that will potentially help with children's vocabulary retention. This is yet remains as a "Could" priority as it's only a supplementary feature and not an immediate critical need.	Alex(Child) is practicing some words and having hard time remembering the words	Given that a function is designed for the robot, which use openAI to generate the image based on the input word	After Alex and robot agree on the word to write	A image will be generated automatically on the child interface and can be updated with the new writing word.	No

	U S5	A s a re se ar ch er,	I want to move the UI from PyQt to a web version (browser)	so that I can access via various platforms and devices without the need for download /installation	S ho uid	35	This transition will provide end-users with increased accessibility, platform independence, and reduced download/installation barriers	Olivia(Researcher) want to access the project through Web, because the pad is not working	Given that all the backend software are connected to the UI in the web	After Olivia launching the project	He can access to the manager ui and child ui locally through the browser and all the function can be executed on browser successfully.	No
	U S6	A s a ch ild,	I want to be able to select the domain I'm interested in	so that the learning becomes more fun and engaging	C ou ld	34	This is another optional feature that can contribute to more motivating and personalised learning environment for children	Alex(Child) is bored with writing words in the list, he want to communicate with the robot in his interested domain and write the words which are related to his interests	Given that the robot will ask children's interest directly through conversation based on the chatGPT and find the children's information in the database	Alex starts to communicate with the robot	The robot can find the interest through the conversation with the children, communicate with Alex in his interested domain, and provide the writing words which are related to the domain.	No
<b>Upgrad e ROSba g data manag ement</b>	U S7	A s a re se ar ch er ,	I want to log relevant information during a session with a child	so that it helps me conduct autonomous assessment of interaction	S ho uid	27	This will be a valuable asset for driving the data-driven performance improvements to the robot's capabilities	Olivia(Researcher) wants to improve the Robot behaviour in communicating and interacting with the children	Given that the software will store the information and store every activity from the users	Olivia open the data file collected by the software	It can lists files which represent different activity and contain the data which collected by the software	No
	U S8	A s a re se ar ch er ,	I want to have a tool for ROSbag annotation	so that I can have a structured way to analyse and visualise data	S ho uid	26	This will be a valuable asset for driving the data-driven performance improvements to the robot's capabilities	Olivia(Researcher) is not expert in IT, and can not understand the raw data which collected by the robot.	Given that the software has some methods to analyse the data in different aspect and a window to illustrate these data analysis.	Olivia open the window for data visualisation.	Olivia can select the data for visualise and the interface can illustrate some diagrams of the data analysis.	No

# | Development Environment

## Software Processes Management

The following technologies will be used by the team to support the software development processes:

- **Trello** to track and manage development tasks ([our Trello board](#));
- **Confluence** to store all the useful resources linked with the project (meeting minutes, documentation ...);
- **Slack** as an external communication tool to interact with our supervisors and clients (the team decided to use **Discord** for internal communication) ;
- **Zoom** to host meetings with our supervisors and clients ;
- **GitHub** as a DevOps tool to update and manage the source code of the project ([our GitHub repository](#)).

# Previous Projects - Setup & Tests

The work we will conduct this semester constitutes the developments of three projects which were conducted last semester ([BlueRing](#), [BoxJelly](#) and [RedBck](#)). Following in from these three projects, we propose to use the same stack to develop our new source code on the top of the existing one. Indeed, the technologies used last semester are (almost) the same for each team.

The main part of the stack will be composed of:

- Linux (**Ubuntu**) for the operating system ;
- **ROS** (Robot Operating System) used for communication the robot and the Python controlling module ;
- **Python3** to develop the new functionalities for the robot.

Nevertheless, even though the objectives were similar for each of the previous teams, we noted differences in the versions of the softwares used and in the installation instructions. This is why **containerizing applications**, such as Docker, could represent a powerful tool to set a single and stable coding environment.

This approach will have two main advantages:

- No compatibility issues will be encountered by the team during the development phase (the code developed by one team member will be executable by any other) ;
- The environment will already be defined for the future teams involved on this project.

## Comparing Projects

Our first step will be to compare the three projects.

Last semester, each team have been either:

- updating existing modules (ex: upgrade Python's version)
- modifying existing modules so they could fit their source code
- developing new source code.

For the two first previous points, even if the teams started with the same source code, it is not said that they made the same changes/improvements. Two quick ways to compare files can be:

- if there is only one file to compare, VisualStudioCode ([| Developer Tools](#)) allows a user to compare two files (more information [here](#))
- if multiple files need to be compared, it might be wise to use Git: for example, create a first branch with the files from TeamA, an other one with the files from TeamB and create a pull request. The differences between the files will be shown.

# General Remarks

Overall, it was not an easy task to run the three previous projects, even with the documentation provided by each team. In most cases, issues we encountered were dealing with incomplete or missing installation instructions. Because of this, we spent a lot of time setting up the environment for each project.

To overcome these issues, we decided to create a specific Docker image for each project which contains all the installation instructions required to launch their respective source code. Even if the build time was really long (around an hour for Boxjelly and Bluering), it appeared to be the most effective method. Indeed, in (almost) all cases, if the encapsulated code worked on one of a team member's laptop it would work on the other machines too.

The "almost" in the previous sentence refers to two issues we encountered:

- with Docker Desktop (Ubuntu): we had issues with ports and permissions to access the virtual robot, but after uninstalling Docker Desktop and installing only Docker, the issue was resolved
- with the system architecture (described on [| Installation Guide](#)): some Python modules which require a lot of resources (like Tensorflow) result in code crash on Macs (error: Illegal instruction - core dumped).

# Bluering

## General Remarks

Probably the hardest project to run. We are still facing Python errors while running the source code.

The installation required a lot of rework and research to fix issues encountered due to missing or too generic instructions.

The issues we faced were the following:

- The ROS dependencies were only listed as bullet-points containing only a link to the GitHub source code of the modules. No instructions on how and where to include these modules in the source code were provided.
- For one of the ROS dependencies (gscam), one file has to be downloaded from GitHub and added in a specific directory to make the code run (no mention of this in the documentation).
- No list of the required Python modules was given (similar to Redback). To know which modules were missing, we just ran the code updating the module list, until it would run without any import error. For this point, we faced the architecture error (mentioned on [General Remarks](#)) after the installation and import of Python Tensorflow: the Mac users cannot run the code any further.
- Import and attribute errors for Python modules, which can be linked to the previous point (some specific version of modules might need to be installed).
- Declaration of environment variables (for example, which paths needs to be added to the .bashrc file). In addition, in order to build or run the project, a source command needs to be executed before calling ROS specific functions.
- We noticed that the exact same lines codes were commented in Boxjelly but not in Bluering. If they are not commented, the Bluering project cannot be ran (we need to investigate on this point because it might come from the fact we only tested the code with the simulated robot for now).

We provided the Dockerfile and the installation/run commands on one of our branches [here](#).

# Boxjelly

## General Remarks

For this project, the installation instructions were the most precise and complete ones compared to the two other teams. They even provided an auto-installation script which can run all the installation instructions.

But if we still faced some issues installing the environment manually, mostly because some required Python modules did not appear in the requirements file.

Moreover, since Bluering and Boxjelly share almost the same installing instructions, we were able to reuse some parts of the developed Dockerfiles, which sped up the environment setup.

We provided the Dockerfile and the installation/run commands on one of our branches [here](#).

# Redback

## General Remarks

This project was the easiest one to run, in particular for the GUI module for which the team created a Docker image (as soon as Docker is installed, one command is enough to run the code).

Nevertheless, we still encountered the following issues:

- missing installation instructions, especially the list of the required Python modules (listed in a requirements.txt file for example). To know which modules were missing, we just ran the code updating the module list, until it would run without any import error
- declaration of environment variables (for example, which paths needs to be added to the .bashrc file).

We provided the Dockerfile and the installation/run commands on one of our branches [here](#).

# | Coding Standards

## Branch Naming

The Git branching model used for this project will be [Gitflow Workflow](#).

The general idea of this workflow is to work with 2 principal branches:

- *main* which will host the tested and finalised source code ;
- *develop* which will serve as a feature integration branch to test new releases before merging them on main.

The other branches will be named according to the following convention:

- *feature*: used to develop new features or enhancements (created from develop)
- *release*: used to prepare for a new production release (created from develop)
- *hotfix*: used to quickly patch production releases (only branch type to be created from main)
- *(support)*: used for versions of software that are still supported but are not the latest version).

More information about this workflow is available [here](#).

## Code Formatting

Most of our code development will be Python, so the coding standards used for this language will be the following:

- [PEP8](#) for all the coding conventions (ex: syntax checking, best practices)
- [Black](#) for code formatting automation (ex: maximum character number per line).

To ensure that the code pushed is consistent with the chosen Python standards, a [GitHub Action](#) has been set on the [repository](#).

At each commit, a pipeline will be triggered. It will execute a command (module [Flake8](#)) to check if each specified Python file complies with PEP8's rules.

If the pipeline should fail, it would mean some changes have to be made to ensure the consistency of the source code.

# | Docker - Overview

In order to encapsulate the environment of our project, we chose to use [Docker](#).

Docker is platform that allows developers to *containerise* applications. It means that all the developed code, along with its dependencies, is executed in a closed and isolated environment called a "container".

Contrary to virtual machines, the containers are built on top of the operating system of the host, they are more light weighted.

## Docker Objects

Docker concepts mainly rely on two objects, **images** and **containers**.

### Docker Images

An image is a list of instructions that will help creating containers. Images are often built on the top of an existing one (the ROS Humble in our case), and developers will add their own instructions to adapt it to their project requirements.

Images are built from a `Dockerfile`. This file will contain the customisation instructions described above.

For example, if the project requirements specify to:

- use the Humble version of ROS
- install Python Tensorflow

the `Dockerfile` will be structured as follow:

```
FROM ros:humble # base image

# the image only contains the minimal requirements to run ROS, so we need to install pip3
RUN apt-get update && apt-get install -y python3-pip
RUN pip3 install tensorflow
```

Many other commands like `RUN` are available to build Docker images, this [page](#) list them all.

When the `Dockerfile` is ready, the image has to be built using the `docker build` command (documentation [here](#)). During the build, all the instructions listed in the `Dockerfile` will be executed and stored: the aggregation will compose the final image.

### Docker Containers

After building the image, containers (instances of the image) can be created and deployed.

As described above, a container hosts an environment that is constructed based on the sequence of commands specified in the `Dockerfile`.

Containers can be created and launched using the `docker run` command (documentation [here](#)).

They can also be executed (documentation [here](#)), if you want to log in the container and execute bash commands for example (useful for debugging).

## Installation

Installation instructions are listed in the [| Installation Guide](#).

(based on Docker's documentation, available [here](#) for more details)

# | Developer Tools

This section presents the recommended tools and commands each team member should use to ease the code development:

- the choice of the [IDE](#)
- how to [Develop inside Containers](#)
- how to [Test & Debug](#) the developed ROS code.

# IDE

We recommend to use [VisualStudioCode](#) as IDE (Integrated Development Environment). Indeed, it is possible to install extensions (for syntax highlighting for example) which will automatically format the code or warn the user if one of the chosen coding standards mentioned above are not respected ([Coding Standards](#)).



To install an extension, click on this button (it is located on the left navigation bar). Then, search the extension in the Marketplace and install it.

The following extensions should be installed (at least):

- [Black Formatter](#)
- [Flake8](#)
- [Pylance](#)
- [Python](#)

After installing these extensions, some configuration is needed.



In VisualStudioCode, go to the settings: Code > Settings > Settings (more information [here](#)). Then, click this button (it is located at the top right of the screen). That button will open a file (`settings.json`) which allows a user to edit the settings manually.

Add the following lines to that file:

```
// each time a Python file is saved, Black will format it
"[python]": {
    "editor.formatOnSave": true,
    "editor.defaultFormatter": "ms-python.black-formatter"
},
// how many characters per line in Python files
"black-formatter.args": [
    "--line-length",
    "80"
],
"flake8.args": [
    "--max-line-length",
    "80"
]
```

We also recommend to install the following extensions:

- [Docker extension](#): to easily manage Docker images and containers if you are not familiar with this development tool (or download [Docker Desktop](#))
- [Dev Containers](#): to enable the possibility of using containers as a development environment (view [Develop inside Containers](#) for more details)
- [Github Actions for VSCode](#): to manage GitHub workflows (CI/CD).

# Develop inside Containers

For this project, we chose to use Docker to encapsulate our environment (Python modules required, ROS software ...).

If you are not familiar with Docker and its concepts, we recommend you to read [| Docker - Overview first.](#)

Using the extension Dev Containers, it is possible to use a running container "as a full-featured development environment". It will open a VSCode window "remotely" so that the files located inside the container will be accessible and easily editable.

This development method has many advantages:

- Reproducibility and Consistency: the environment setup is consistent and does not differ from a developer to another. Therefore, if a developer requires to install a new Python module during his development phase for example, any other developer from the team can integrate his changes by re-building or pulling the new Docker image
- Onboarding: when a new developer joins the team, his onboarding would be faster than if he had to run all the environment setup instructions. He would only need to clone the Git repository and build or pull the associated Docker image
- All the VSCode features become available in the container environment (for example, code autocomplete using the Python extension).

## How to use the extension (video tutorial available [here](#)):

First, build the image containing your environment:

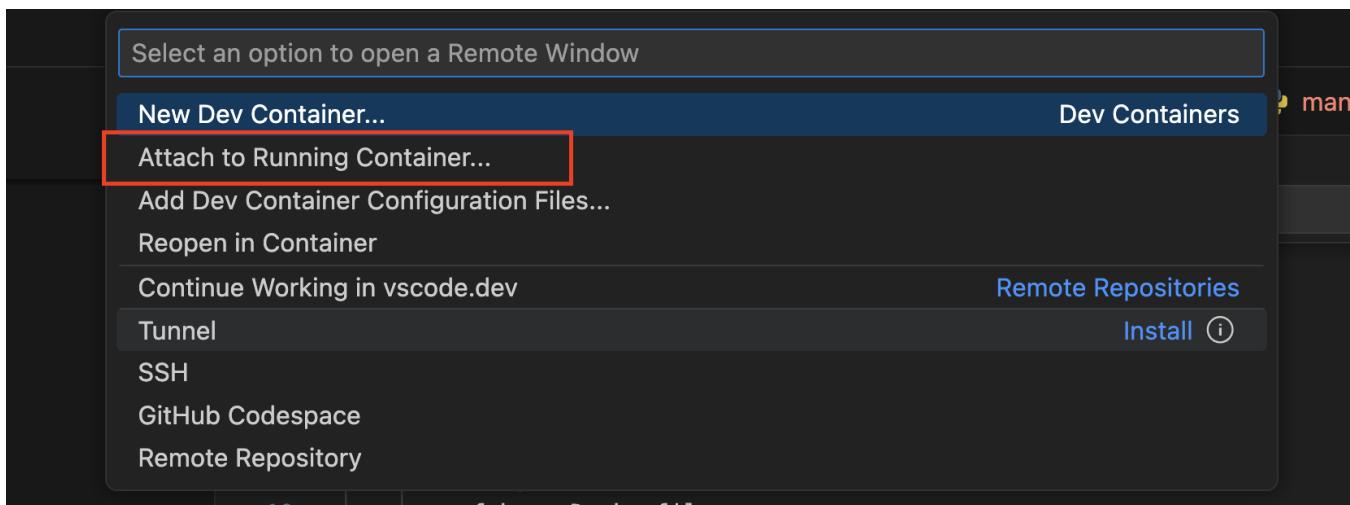
```
docker build --tag <your_image_name> --platform linux/amd64 -f <Dockerfile_name> .
```

Then, run a container which will host that environment:

```
# careful with which user you are using inside the container (option --user)
# in general, using root can be dangerous when volumes are used
docker run -it \
    --name <your_container_name> \
    --user <user> \
    --network=<docker_network> \
    -v <path_on_host>:<path_in_container> \
    <your_image_name> bash
```

**Note:** the v option (volume) will create a link between the selected folder of the host (for example, the src folder of your project) and the folder of the container. So every changes made inside the container using the extension will be transferred on the host and vice versa (find more information on Docker's documentation [here](#)).

Now that the container is running, we can attach a VSCode to it. Click the button  located at the extreme bottom left of your VSCode window. It will open a dialog window: select Attach To Running Container and click on the name of the controller launched during the previous step.

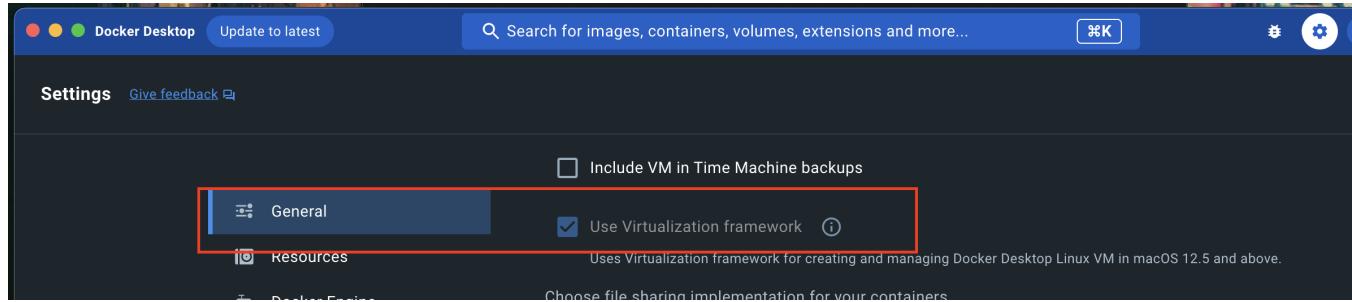


Finally, install the recommended extensions (see above) in your container.

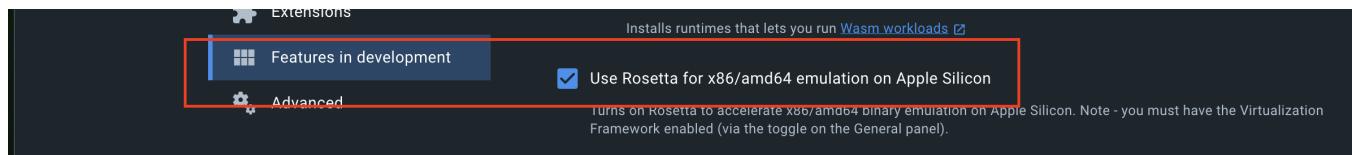
## Note for M1 and M2 Mac chips users:

By enabling:

- the "Virtualization Framework"



- the "Use Rosetta for x86/amd64 emulation on Apple Silicon" option in the "Features in development"



in Docker Desktop's settings (directly on your Mac, not in Ubuntu), it is possible to launch and build amd64 Docker images by adding a tag `--platform linux/amd64` for each Docker command ran (see the documentation [here](#)).

Thereby, the development environment will be accessible even without using Ubuntu. It will be useful to develop code and do a bit of debugging, but not to test it on the robot as mentioned in [| Installation Guide](#).

# Test & Debug

## Debugging/Testing code inside an ROS Docker environment (video tutorial available [here](#))

Once a container hosting the ROS environment is running, you can access the code inside it either by:

- opening a VSCode window using the Dev Container extension ([Develop inside Containers](#)). When the window is opened, open a terminal (ctrl+J or cmd+J), then on the prompt type `bash` and press enter
- executing the container (eg. log inside it) by running in a terminal window:

```
# note: before executing this command, the container must already be running
docker exec -it <the-name-of-the-container> bash
```

(All the following commands are meant to be ran inside the container, eg. in the terminal window opened by either of the two possible options presented above)

Ensure that the ROS environment variables are loaded:

```
source /opt/ros/humble/setup.bash
which ros2
# the output should be "/opt/ros/humble/bin/ros2"
```

Navigate to the same level as the `src` folder to build the desired ROS packages:

```
sudo colcon build --packages-select <the-package-name>
```

Add the package's path to the environment variables:

```
source ./install/setup.bash
```

The nodes / package can now be tested:

- to run a single node, run:

```
ros2 run <the-package-name> <the-name-of-the-node>
```

- to run all the nodes at the same time (if the package contains a `.launch.py` file:

```
ros2 launch <the-package-name> <the-name-of-the-launch-file>
```

If you modify the source code of the package, rebuild it and run the node / the package again. If the package is not rebuilt, the changes will not be taken into account.

# | Installation Guide

This section gathers all the installations and commands needed to launch the project. Hence, the pages must be consulted in the same order as they appear in Confluence's tree structure:

- [Ubuntu](#)
- [Choregraphe](#)
- [Docker](#)

to finally be able to run all the commands of [Launch the Project](#).

# Ubuntu

To run the software that enable communication with the NAO robot ([Choregraphe](#)), developers need to install a version of Ubuntu (20.04 or 22.04) on their laptop.

## For Windows users

A dual boot (complete installation of the Linux OS) might be the best option because virtual machines can be really slow, in particular when Choregraphe is running.

## For Mac users

First determine the computer's architecture by running the `uname -a` command in a terminal. Now there are two options:

- if the architecture is amd64 / x86 (mostly old models of Macs), then it is compatible with Choregraphe's required architecture. You should be able to install the desired version of Ubuntu using BootCamp
- if the architecture is arm64 (new versions of Macs, using M1 or M2 chips), then it is not compatible with the required architecture.

### Solutions for arm64 based Macs:

- VirtualBox, Parallels ... do not offer the possibility to emulate an amd64 architecture on arm64 based Macs, [UTM](#) does. But when Choregraphe and the programs are running, the virtual machine becomes really slow and laggy
- We also tried to only run Choregraphe in UTM and the code on the host. Choregraphe's robot port can be forwarded to the host and Docker Desktop offers the possibility to run amd64 based containers on arm64 architecture, but we never succeeded to reach that exposed port from the code running in a container
- Finally, we also considered to use GoogleCloudPlatform to host the amd64 based VM (tutorial [here](#)): by selecting the right amount of CPUs (at least 4), the result was more smooth compared to UTM. The only downside hosting the development environment on GCP is that we cannot connect to the real robot, the IP is unreachable from the VM. Moreover, running the VM generate costs (but usually, GCP offers some free credits to their new users that are sufficient to cover these costs).

In the end, UTM seems to be the unique viable solution if the tests have to be run with the real robot. If the simulated robot is enough, running the code on a GCP VM is a better (but costly) option.

If the developer only requires to do some code development or debugging that does not require any interaction with the robot (real or simulated), using Docker containers as a development environment can be enough (view [| Developer Tools](#)).

# Choregraphe

In order to communicate with the NAO robot, a software called [Choregraphe](#) has to be installed. It has two purposes:

- launch a robot simulation, to run code even if the physical robot is not available
- establish the communication with a real Nao robot.

Even if the installation page of Choregraphe offers installation files for Windows, Mac and Linux, but we recommend to use the Linux version.

## Installation on Ubuntu

1. Download the setup file from this [link](#).

2. Change the file permissions and execute it:

```
chmod +x choregraphe-suite-2.8.6.23-linux64-setup.run  
sudo ./choregraphe-suite-2.8.6.23-linux64-setup.run
```

3. Try to run:

```
"/opt/Softbank Robotics/Choregraphe Suite 2.8/bin/choregraphe_launcher"
```

4. In case of an error:

```
# error  
/opt/Softbank Robotics/Choregraphe Suite 2.8/bin/..../lib/..../lib/libz.so.1:  
version `ZLIB_1.2.9' not found (required by /usr/lib/x86_64-linux-gnu/libpng16.so.16)
```

Run the following commands:

```
cd "/opt/Softbank Robotics/Choregraphe Suite 2.8/lib/"  
sudo mv libz.so.1 libz.so.1.old  
sudo ln -s /lib/x86_64-linux-gnu/libz.so.1
```

5. Command specified in 3 should work, it will launch the application. It is also possible to launch Choregraphe by clicking directly on the application shortcut.

(Adapted from this [tutorial](#))

# Docker

To install Docker, run the following instructions in your Ubuntu instance.

## 1. Uninstall all conflicting packages

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

## 2. Add Docker's repository to your Apt repository

```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl gnupg  
sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
sudo chmod a+r /etc/apt/keyrings/docker.gpg  
  
# Add the repository to Apt sources:  
echo \  
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```

## 3. Install Docker packages

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

## 4. Check if the installation is successful

```
sudo docker run hello-world
```

(Docker's installation guide for Linux is available [here](#))

## Post installation steps

After installing Docker, we recommend to execute the following commands:

```
sudo groupadd docker  
sudo usermod -aG docker $USER
```

This will enable the current user to use Docker commands without calling `sudo`. To be effective, the user should restart his Ubuntu instance first.

In case the following error appears when executing a Docker command,

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?.  
See 'docker run --help'.
```

it means that Docker is not running. The service has to be started manually:

```
sudo systemctl start docker
```

# Launch the Project

## Requirements: Ubuntu with Docker and Choregraphe installed

To launch the project, run the following commands in the same order they appear:

### Clone the repository:

```
git clone git@github.com:COMP90082-2023-SM2/NA-Redback.git  
cd NA-Redback
```

### Check if the "make" command is available by running:

```
which make  
# if the output is "/usr/bin/make", then skip the following commands  
# if not, run  
sudo apt-get update  
sudo apt-get install -y make
```

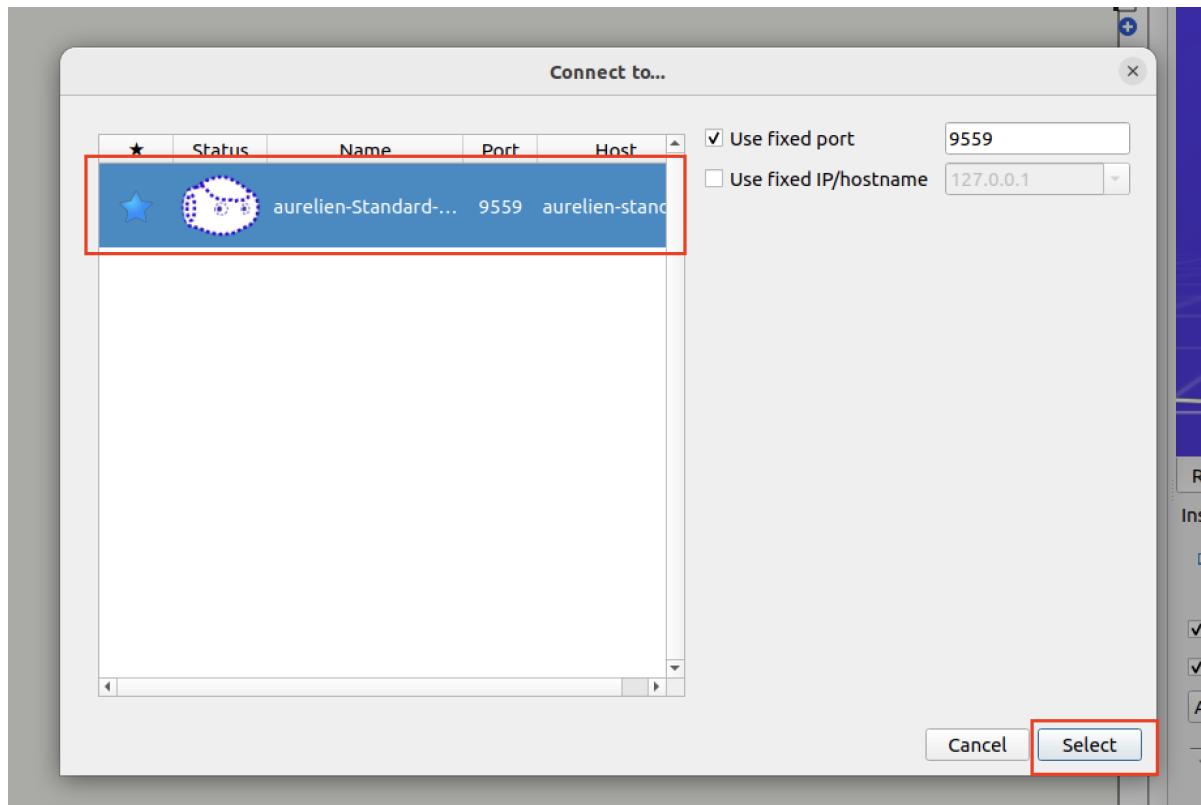
### Build the Docker images:

```
# you can run the commands in two separate terminals for efficiency  
# make sure to run them inside the project folder -> cd /path/to/NA-Redback  
make build-nodes  
make build-robot-controller
```

### Launch Choregraphe and start the virtual robot:

Once Choregraphe is opened, click on Connection > Connect to...

Select the robot by clicking on it, then press Select.



After the initialisation, you should be able to see a similar message as the one below, meaning that the virtual robot is ready to be used.



**Allow Docker to access Ubuntu's display:**

```
xhost +
```

**Launch the Docker containers:**

```
make run-robot-controller
```

That command will start a small Flask application (path between the ROS nodes and the NAOQi SDK).

Then, in an other terminal, run the following command to start the container hosting the ROS nodes:

```
make run-nodes
```

The container hosting the ROS nodes has now started. Now we need to launch each nodes.

Open 3 different terminals, and run one command per terminal:

```
# make sure to run them inside the project folder -> cd /path/to/NA-Redback
make start-cowriter
make start-trajectory-following
make start-adaptive-words
```

It can take some time to launch the ManagerUI (last command above). If the window is not responsive, wait a few seconds.

**If you updated the version of a Docker image, remove the existing containers before running the start commands again, otherwise the changes will not be taken into account:**

```
# make sure to run them inside the project folder -> cd /path/to/NA-Redback
make rm-nodes
make rm-robot-controller
```

Then:

```
# make sure to run them inside the project folder -> cd /path/to/NA-Redback
make run-nodes
# and/or
make run-robot-controller
```

Finally:

```
# make sure to run them inside the project folder -> cd /path/to/NA-Redback
make start-cowriter
make start-trajectory-following
make start-adaptive-words
```

## | Development Features

# Sprint 2 Features

## Demo Video

The video showcasing the features developed during this sprint can be found [here](#).

## Test Cases

Test cases were executed for user stories that were fully completed during Sprint 2. Those that are continuing to the next sprint (CS1, CS2, CS3, US3) will be tested during Sprint 3.

Epic	ID	User Story		Acceptance Criteria				Test Result	
				Scenario	Given	When	Then		
Improvement of conversation using ChatGPT	U S1	A s a parent,	I want the robot to have a conversation with my kids in a child-friendly and safe manner	so that my child's interaction stays positive and free from inappropriate content.	Alex(Child) is communicating with the robot	Given that some prompts are set for the robot's conversation function	An improper topic appears during the conversation	The robot can detect the improper topic/words and start a new topic which is child-friendly.	PASSED ■ refer to 0: 03: 49 in the demo video
Improvement of CoWriter	U S4	A s a parent,	I want to add more child-friendly elements to UI such as illustrations	so it helps kids to easily understand and remember the words	Alex(Child) is practicing some words and having hard time remembering the words	Given that a function is designed for the robot, which use openAI to generate the image based on the input word	After Alex and robot agree on the word to write	A image will be generated automatically on the child interface and can be updated with the new writing word.	PASSED ■ refer to 0: 02: 25 in the demo video

# AI-Generated Image

## OpenAI

OpenAI has pioneered advancements in artificial intelligence, particularly in natural language processing and computer vision. One of its standout features is the ability to generate high-quality images based on textual prompts, making it a valuable tool for developers seeking efficient and creative solutions for visual content.

## Set up

At the current stage, the image will be generated by using the writing word. When the user puts the word in the box and clicks the 'send word to write' in the 'manger\_ui' window, the OpenAI will generate the image based on that word and return the URL for the image. Then this image will be downloaded into the database(under the project folder) through the URL. If a new writing word has been entered, the image in the folder will be replaced.

When illustrating the AI-generated image on the "child\_ui" window, the image will be loaded and plotted on the window. A timer is used to keep refreshing the image and keep the image updated when there is a change in writing word.

## Prompt

According to the OpenAI, they've taken steps to limit DALL-E 3's ability to generate violent, adult, or hateful content. We believe that the image will be safe enough for the children. Therefore, only one prompt 'cartoon' is added to provide a childish AI-generated Image.

## Further Improvement

1. The method of downloading the image into the project folder for illustration is not efficient and waste of memory. Therefore, we will try to illustrate the image directly through the URL.
2. In order to realise improvement 1, the event for generating an image should be held in the 'child\_ui.py' instead of 'manager\_ui.py'. The ROS node for generating the image will be designed.
3. It will take some CPU when using the timer to keep the image updated. Therefore, we are thinking of a more efficient way to control the updated image.
4. After several experiments on generating the image, though OpenAI has safety control for generating the image, it is still found that some images will not be proper for the children. More prompts will be added to generate a better child-friendly image.

# Filtering & Moderation

NAO's conversation capabilities has been further improved by refined prompt engineering using **GPT-4**, OpenAI's latest and most advanced GPT model (as of September 2023), with its chat moderation supported by the **OpenAI's moderation API**. Using a moderation model along with fine-tuned prompt engineering was crucial in order to ensure that the conversation between the robot and children stay kids-friendly and safe.

## Prompt Engineering

- Use of [chat completion API](#) (currently supported in gpt-3.5-turbo or gpt-4 models) which allows for different roles within the chat including system, assistants, and user
- Set NAO's persona using the system role to control how NAO is expected to respond to further user messages
- Include the user information (e.g. children aged between 5-10) to ensure using of age-appropriate language throughout the chat

## Moderation Model

OpenAI's moderation API was used to filter and moderate input to ensure that it meets safety and content guidelines. Every input chat or audio from children will be passed to this moderation model and will receive an output like the example below. As you can see, the input will be evaluated based on multiple different categories such as hate, harassment, violence etc. We used the "category\_scores" result as it allows for more customised and refined filtering than the "flagged" result which is just a boolean value. We have set a relatively strict threshold (=0.05), and if the input scored any of these category scores higher than this threshold, it will be marked as flagged. In this case, the robot will redirect the conversation to a more child-friendly topic.

### Output Example

```
{  
  "id": "modr-XXXXX",  
  "model": "text-moderation-005",  
  "results": [  
    {  
      "flagged": true,  
      "categories": {  
        "sexual": false,  
        "hate": false,  
        "harassment": false,  
        "self-harm": false,  
        "sexual/minors": false,  
        "hate/threatening": false,  
        "violence/graphic": false,  
        "self-harm/intent": false,  
        "self-harm/instructions": false,  
        "harassment/threatening": true,  
        "violence": true,  
      },  
      "category_scores": {  
        "sexual": 1.2282071e-06,  
        "hate": 0.010696256,  
        "harassment": 0.29842457,  
        "self-harm": 1.5236925e-08,  
        "sexual/minors": 5.7246268e-08,  
        "hate/threatening": 0.0060676364,  
        "violence/graphic": 4.435014e-06,  
        "self-harm/intent": 8.098441e-10,  
        "self-harm/instructions": 2.8498655e-11,  
        "harassment/threatening": 0.63055265,  
        "violence": 0.99011886,  
      }  
    }  
  ]  
}
```

# Migration: ROS1 to ROS2

"The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. From drivers and state-of-the-art algorithms to powerful developer tools, ROS has the open source tools you need for your next robotics project." ([ROS Documentation](#))

The version we used for the upgrade is the **ROS2 Humble**, released in May 2023.

This page gathers the the most common updates that had to be made to the code to upgrade the ROS version, but also some general information about developing in the ROS development.

**Note:** this page will be focusing about Python ROS packages since it was our main coding language for this project.

## ROS2 Python Packages

### Organisation

To create a new Python ROS package, run the command

```
ros2 pkg create --build-type ament_python <name-of-the-package>
```

It will be composed of a least 5 main elements:

- a `setup.py` file containing the definition of the module and the dependencies specific to Python, but also the definition of the nodes (their name, the corresponding file and )
- a `package.xml` file which is the equivalent of `setup.py` but for the ROS dependencies
- a `setup.cfg` file to store options for the `setup.py` file
- a resource folder containing an empty file named after the package (essential when building the package)
- a source folder (named after the name of the package)which will contain the source code of the package.

Contrary to ROS1, pure Python ROS2 packages do not need to have a `CMakeLists.txt` file.

For more information about the files, see the documentation [here](#).

The architecture is the following:

```
<package-name>/  
  package.xml  
  setup.py  
  setup.cfg  
  resource/  
    <package-name> (the empty file)  
  <package-name>/  
    ... (Python source files)  
  test/  
    ... (test files)
```

Moreover, the nodes contained in the package can be launched at the same time using a `.launch.py` file. It should be placed in the resource folder of the package. For more information about launch files, see [here](#).

## Build and Run

To build a pure Python ROS2 package, run the following commands:

```
sudo colcon build --packages-select <package-name>  
# when it is done  
source ./install/setup.bash
```

To run one node:

```
ros2 run <package-name> <the-name-of-the-node>
```

To launch the package (if it contains a `.launch.py` file):

```
ros2 launch <the-package-name> <the-name-of-the-launch-file>
```

## Main changes in ROS2 (Python)

### Node implementation

The common practice to build a node in ROS2 is to create a class inheriting from the `Node` object of `rclpy`. Each node should have:

- its own file, containing a `main` function. This function will be declared in the `setup.py` file
- a name

For example (`example_node_file_name.py`):

```
import rclpy
from rclpy.node import Node

class ExampleNode(Node):
    def __init__(self):
        super().__init__("example_node_name") # name declaration
        ...

def main(args=None):
    rclpy.init(args=args)

    example_node = ExampleNode()

    rclpy.spin(example_node) # launch the node

    example_node.destroy_node() # destroy the node when the process is ended
    rclpy.shutdown()

if __name__ == "__main__":
    main()
```

To declare this node, the following line should be added in the `setup.py` file:

```
from setuptools import setup

package_name = "example_package_name"

setup(
    ...,
    entry_points={
        "console_scripts": [
            "example_node_name = example_package_name.example_node_file_name:main",
            "# ... other nodes
        ],
    },
)
```

## Subscriber / Publisher

To communicate, the nodes use a publisher/subscriber system. The publishers send messages to a specific topic. The subscribers listen to these topic messages and launch specific commands (callbacks) depending on what they receive.

The ROS2 documentation offers a really educational tutorial about this topic [here](#).

### Handling multithreading with ROS2 nodes

Once a node is launched using `rclpy.spin`, the `main` function will not execute any code declared after.

In order to execute a program in parallel of a node (a GUI for example), the programs can be launched in different threads. For example:

```

from threading import Thread
from rclpy.executors import MultiThreadedExecutor

...
### NOT WORKING
def main(args=None):
    rclpy.init()

    gui = GUI()
    node = GUINode(window)

    # here, the GUI starts first -> the node will not be spinned
    gui.start()
    rclpy.spin(node)

    # here, the node starts first -> the gui will not start
    rclpy.spin(node)
    gui.start()

### SOLUTION: create a distinct thread for the node
def main(args=None):
    rclpy.init()

    gui = GUI()
    node = GUINode(window)

    executor = MultiThreadedExecutor()
    executor.add_node(node)

    thread = Thread(target=executor.spin)
    thread.start()
    node.get_logger().info("node spin")

    try:
        gui.start()

    finally:
        node.get_logger().info("Shutting down ROS2 Node . . .")
        node.destroy_node()
        executor.shutdown()

```

## Using the NAOQi SDK with ROS2

The NAOQi SDK establish the communication between Python and the NAO robot (either virtual or real) using Choregraphe. This SDK has no existing distribution for ROS Humble and is running under Python2.7 (ROS2 is Python3).

In order to establish the communication between the ROS code and the SDK code, we need an interface: Python2.7 and Python3 cannot coexist in the same environment.

We chose to use Python Flask to create a small API, acting like a bridge between the two processes. It functions as follows:

- for example, a node will compute a new trajectory for the robot's arm
- the trajectory will be sent from the node to the API using one of its routes
- the API will execute the code linked to that route and inform the SDK about that trajectory
- the SDK will send the trajectory to Choregraphe.

To ease the development and the deployment of our source code, both environment have been containerised using Docker. Therefore, the two distinct Python environments are well isolated from each other.

# Struggle letter Module

Demo video here: <https://www.youtube.com/watch?v=RNiRpzP787M>

This folder consist of two method:

1. CNN based see in "CNNbased.py", in NAO writing project, we do not apply it to our struggled letter identify module, because it still in a low accuracy.
2. Mutil-attributes based see in "main.py", "robot\_write.py", and "ssim\_similarity.py".

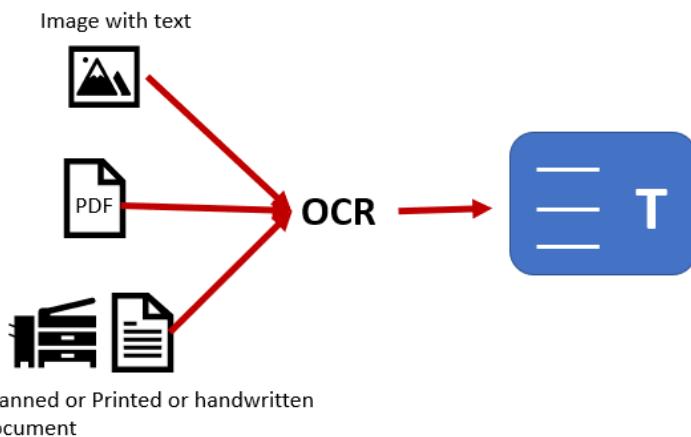
Intro of three file:

2.1 "robot\_write.py": this file only consist of a robot writing board. In this project, it will be pretended as a auto robot writing area.

2.2 "main.py": this file allow children type the letter they wanna practice and a area to write the letter. It also provide a button called "if a good writing" to generate the feedback for child writing.

2.3 "ssim\_similarity.py": this is the main function of how to rate the hand writing and it consists of OCR , MSE, SSIM, Hamming Distance to measure the handwriting.

**OCR (Optical Character Recognition):** Optical Character Recognition (OCR) is a technology that is used to convert printed or handwritten text into machine-encoded text. The primary purpose of OCR is to recognize and digitize text from physical documents, such as scanned paper documents, images, or even text in natural scenes, making it accessible for digital storage, editing, and analysis.



**MSE (Mean Squared Error):** It is a common metric used in statistics, machine learning, and image processing to measure the average squared difference between the values predicted by a model or system and the actual observed values. MSE is often used as a measure of the quality of a predictive model or as a cost function to be minimized during the training of machine learning algorithms.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

**SSIM (Structural Similarity Index):** It is a widely used image quality assessment metric that quantifies the structural similarity between two images. SSIM is designed to capture perceived changes in structural information, luminance, and contrast, making it a valuable tool for comparing the visual quality of images.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x + \sigma_y + c_2)}$$

**Hamming Distance:** Hamming Distance is a metric used to measure the difference between two equal-length binary strings or sequences. It calculates the minimum number of substitutions required to change one string into the other. Hamming Distance is often used in information theory, coding theory, error correction, and digital communications.

### Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

#### Overall method of measure handwriting:

weight sequence = [ ["OCR", Ture or False (0 or 1)], ["MSE", float from 0 to 1], ["SSIM", float from 0 to 1], ["Hamming Distance", float from 0 to 1] ]

Overall = weight[OCR] x OCR - weight[MSE] x MSE + weight[SSIM] x SSIM - weight[Hamming Distance] x Hamming Distance

NOTE: the number of these three attributes has been normalised into range[0,1] the weight for three attribute can be adjust manually, if you want, but you also need to adjust the threshold of struggled letter.

# **Code Review**

# Sprint 2

The code review conducted on GitHub is available [here](#).

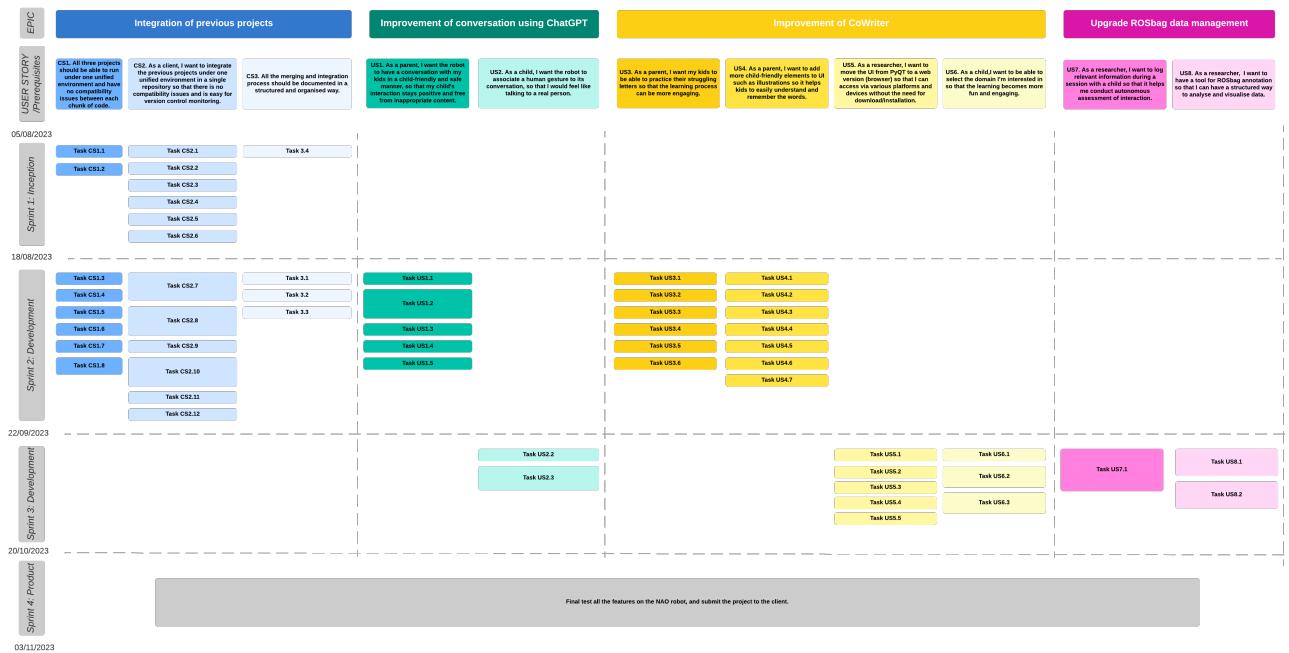
We could not run the GitHub action to use ChatGPT for the code review (the number of tokens requested was too high). Hence, we did it manually.

## General Remarks

- constants should be used more often and not be redefined twice. For example, if some constants are used in two different packages with different source code, a good idea could be to create a "global constants" module that could be imported in both packages. If one of the constant values had to be modified, the modification would only happen one time and be reflected wherever the value is used. Moreover, it reduces the chances of typo that could lead to bugs and/or crashes.
- respect the chosen coding standards (camel case to name classes or functions, use capital letters to define constants)
- tokens (such as OpenAI ones) must not be hardcoded and put in the source code. They must be declared in an environment file that must not be pushed to GitHub (use the `.gitignore` file). Then use a module like Python `os` to
- a lot of commented code is still present in the source code
- sometimes the files are too long and hard to read. For example, the ChatGPT related parts such as phrases, prompt declarations or image generation could be migrated into a distinct module for visibility. Currently, the AI related code is split in two different packages.

## | Sprint Ceremonies

# Sprint Planning



1. Task with larger square box means it has more points.
2. We still have problems with some user stories, so some of them won't have too much. We will update it after the next meeting with client.

Trello : <https://trello.com/invite/b/CPdSWap5/ATTIdcb481b112c05c9eb2833be7521b972a4EC07C65/comp90082-2023-sm2-na-redback>

# Sprint1: Inception Planning

## Sprint 1 Goal

For Sprint 1, our primary objective is to lay a solid foundation for the upcoming project by initiating productive interactions with our clients and comprehensively understanding their requirements. This involves engaging in an initial meeting with our clients to establish a clear scope and user stories for the project. The user stories and tasks should be arranged and distributed into sprints. In addition, we will delve into the past projects executed by our fellow teams to gain valuable insights and enhance our understanding of best practices.

## Key Milestone

- Client Interaction:** Arrange and conduct an initial meeting with the clients. During this session, our aim is to establish rapport, communicate our team's expertise, and create an environment conducive to open dialogue. This interaction will also serve as an opportunity to gather preliminary information regarding the project's objectives, desired outcomes, and any specific requirements or constraints.
- Requirements Identification:** Thoroughly document all the insights obtained from the client interaction. Extract and clarify the project's core requirements, functionalities, and potential challenges. Ensure that all stakeholders are on the same page regarding the project's scope and expectations.
- Project History Review:** Examine the previous projects undertaken by other teams within the organization. This review will provide valuable context, highlighting successful strategies, potential pitfalls, and opportunities for innovation. By understanding the evolution of similar projects, we can leverage lessons learned and align our approach accordingly.
- Resource Preparation:** Compile and organize the necessary resources for the project. This includes refining our understanding of the technologies, tools, and methodologies that will be employed. By familiarizing ourselves with the project's technical landscape, we can streamline our execution and minimize potential roadblocks.

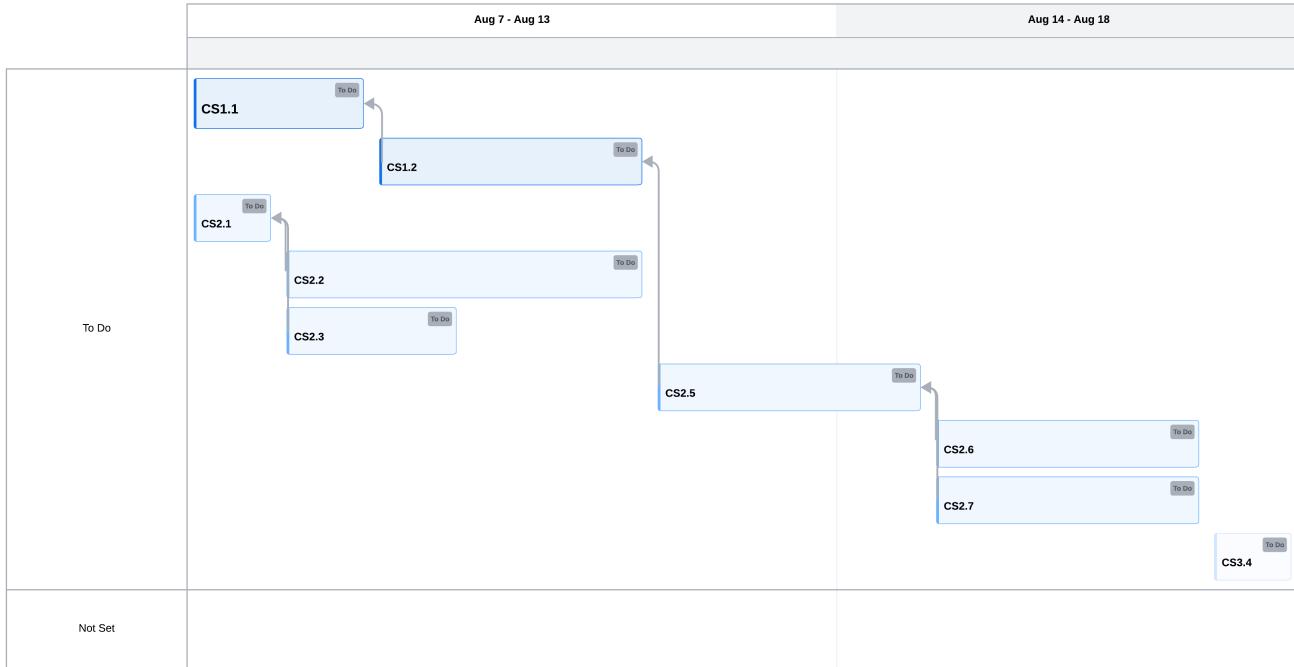
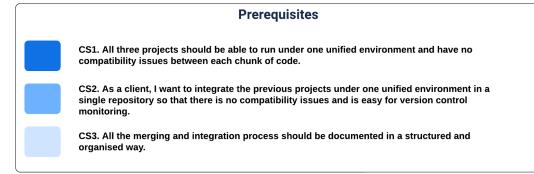
## Prerequisites

Note that "Estimated Workload" indicate the estimation for this sprint and does not necessarily indicate the entire story points for the given user story. For the continuing requirements and user stories, partial story points will be allocated for each sprint. "Estimated Story Points Left" is the estimated amount of the total story points left for this user story after this sprint.

Requirement	ID	Specification	Priority	Estimated Workload	Estimated Story Points Left
Integration of previous projects	CS1	All three projects should be able to run under one unified environment and have no compatibility issues between each chunk of code	Must	10	37
	CS2	All three projects should be merged into a single repository with features integrated	Must	12	60
	CS3	All the merging and integration process should be documented in a structured and organised way	Must	1	6
					<b>Total: 23</b>

## Sprint 1 Timeline

Sprint 1 Timeline  
Start Date: Aug 7  
End date: Aug 18



Link: [https://lucid.app/lucidspark/5bf7f508-d532-4314-a407-5c388416b59a/edit?viewport\\_loc=-1471%2C-684%2C4592%2C2244%2C0\\_0&invitationId=inv\\_b03ed2e3-9f2a-413c-9231-669dd6548ae2](https://lucid.app/lucidspark/5bf7f508-d532-4314-a407-5c388416b59a/edit?viewport_loc=-1471%2C-684%2C4592%2C2244%2C0_0&invitationId=inv_b03ed2e3-9f2a-413c-9231-669dd6548ae2)

## Expected Outcomes

By the end of Sprint 1, we aim to have a comprehensive understanding of the project's requirements, a clear overview of the client's expectations, and insights from previous projects that will guide our decision-making. This holistic preparation will set the stage for successful project execution in the upcoming sprints, ensuring that our team is well-equipped to deliver a solution that aligns with the client's needs and industry best practices.

- Background description
- DO-BE-FEEL list
- GOAL MODEL
- Personas
- User Stories
- Trello
- Development Environment
- Setting Github
- Plan diagram

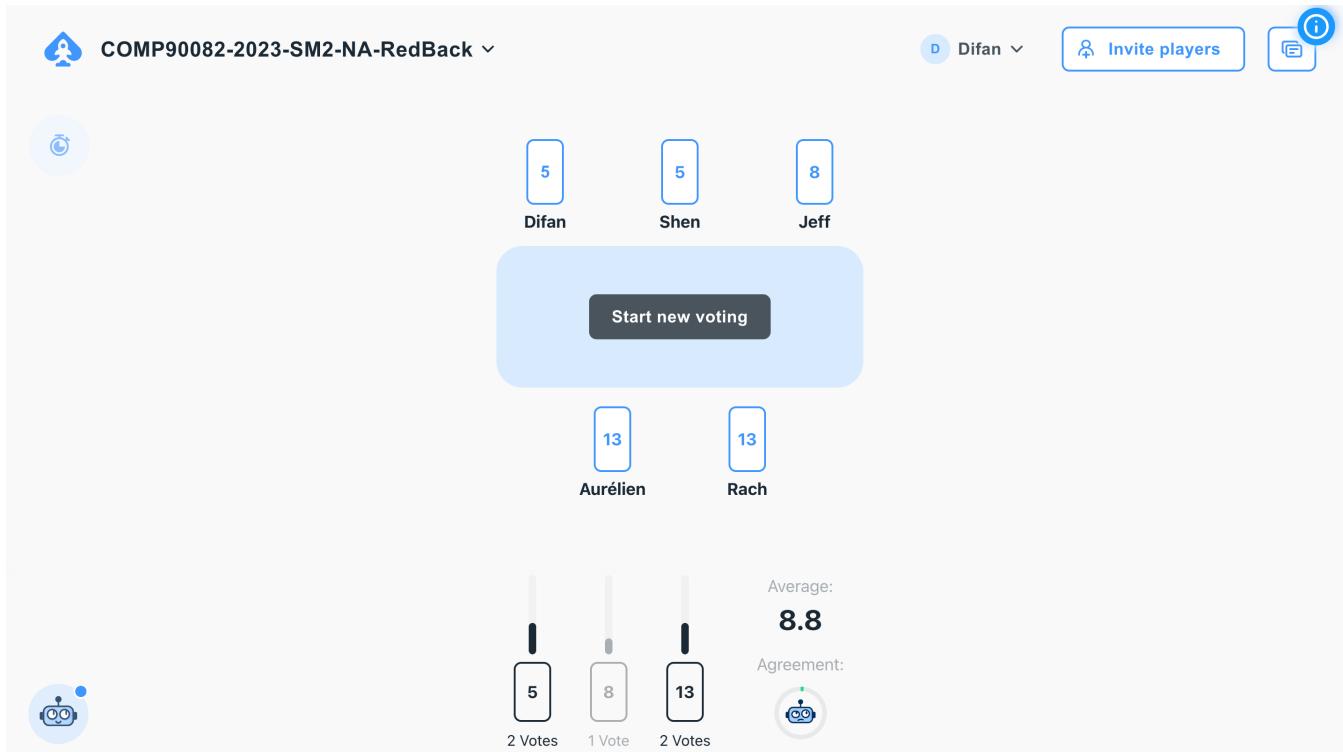
Trello : <https://trello.com/invite/b/CPdSWap5/ATTIdcb481b112c05c9eb2833be7521b972a4EC07C65/comp90082-2023-sm2-na-redback>

ChatGPT Generated: Yes

# Assign the Story Points

Our team employs a collaborative and dynamic approach to project planning through the use of Planning Poker. We gather around and assign story points using the Fibonacci sequence – a unique method that encourages thoughtful estimation. Each team member contributes their individual perspective by selecting a number based on their assessment of the complexity involved. This sparks productive discussions, where we openly share insights, leading to a well-rounded understanding of the task at hand. Through this process, we collectively arrive at a consensus or average for the final story point assignment. This not only fosters team cohesion but also ensures a comprehensive evaluation of each task's intricacies, promoting accuracy and alignment in our project planning efforts.

One screenshot of how we giving storing points is shown as below.



Source from: <https://planningpokeronline.com/>

# Sprint2: Development

## Sprint Goal:

In Sprint 2, our focus is on conducting a comprehensive review and seamless integration of the three previous projects. We will also start working on adding features to the product under the integrated environment, while incorporating feedback received from the Sprint 1 submission. Despite not being part of the user stories, integration of previous projects in one unified environment is a primary goal for this project and is a complex, yet critical task in order to ensure the quality and completeness of the end product.

## Sprint Backlog:

Note that "Estimated Workload" indicate the estimation for this sprint and does not necessarily indicate the entire story points for the given user story. For the continuing requirements and user stories, partial story points will be allocated for each sprint. "Estimated Story Points Left" is the estimated amount of the total story points left for this user story after this sprint.

### Prerequisites:

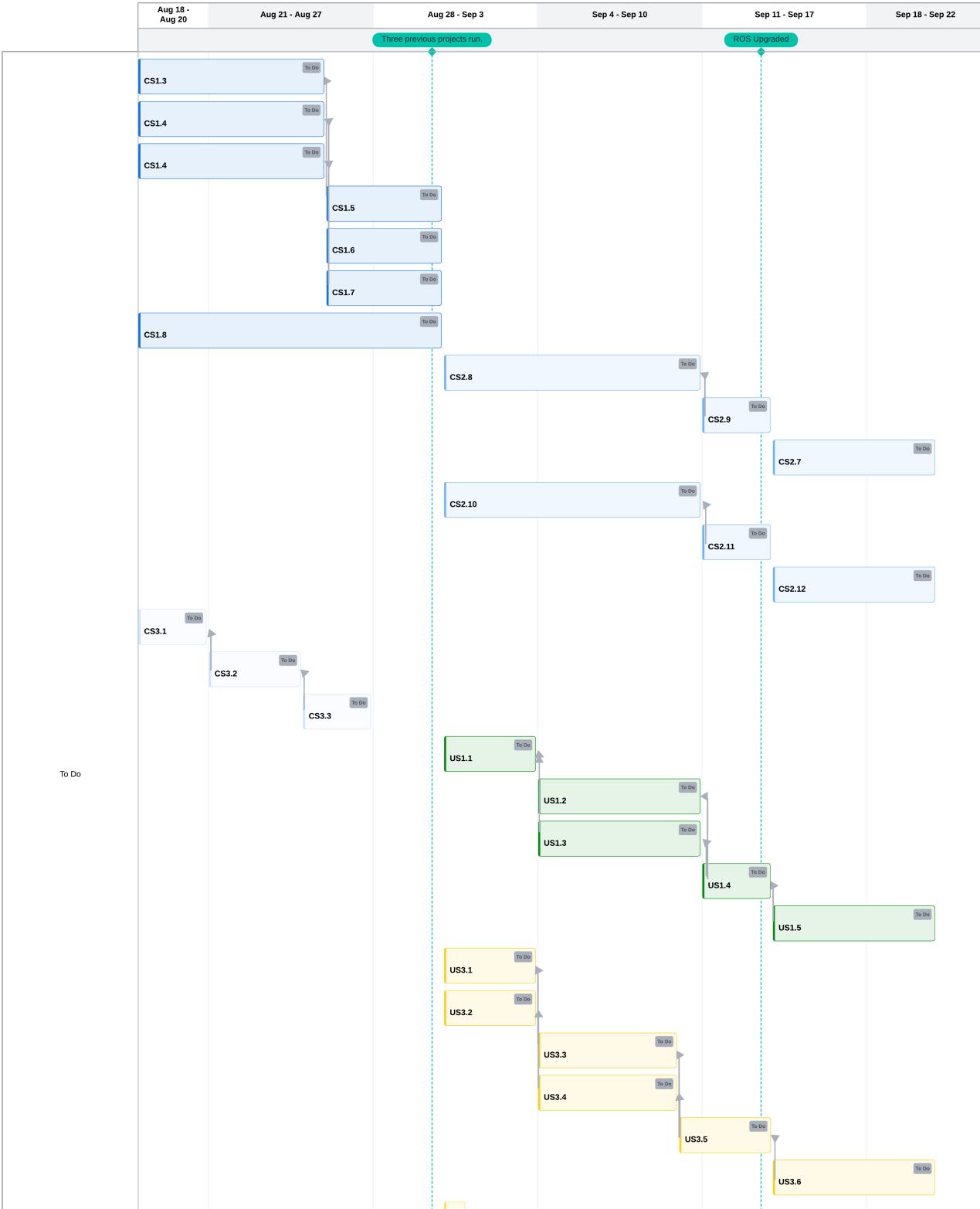
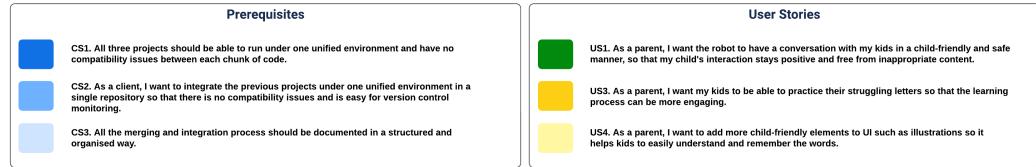
Requirement	ID	Specification	Priority	Estimated Workload	Estimated Story Points Left
Integration of previous projects	CS1	All three projects should be able to run under one unified environment and have no compatibility issues between each chunk of code	Must	37	0
	CS2	All three projects should be merged into a single repository with features integrated	Must	60	0
	CS3	All the merging and integration process should be documented in a structured and organised way	Must	6	0
					<b>Total: 103</b>

### User Stories:

Epic	ID	User Story			Priority	Estimated Workload	Estimated Story Points Left
Improvement of conversation using ChatGPT	U S1	As a parent ,	I want the robot to have a conversation with my kids in a child-friendly and safe manner	so that my child's interaction stays positive and free from inappropriate content.	Must	29	0
Improvement of CoWriter	U S3	As a parent ,	I want my kids to be able to practice their struggling letters	so that the learning process can be more engaging.	Must	30	0
	U S4	As a parent,	I want to add more child-friendly elements to UI such as illustrations	so it helps kids to easily understand and remember the words	Could	34	0
							<b>Total: 93</b>

## Sprint2 Timeline

Sprint 2 Timeline  
Start Date: Aug 19  
End date: Sep 21





Link: [https://lucid.app/lucidspark/9b773683-955a-48d7-9b52-0d7e11d673f4/edit?viewport\\_loc=-962%2C-3481%2C12000%2C6219%2C0\\_0&invitationId=inv\\_067fc950-5ba4-4633-8901-d076854732f7](https://lucid.app/lucidspark/9b773683-955a-48d7-9b52-0d7e11d673f4/edit?viewport_loc=-962%2C-3481%2C12000%2C6219%2C0_0&invitationId=inv_067fc950-5ba4-4633-8901-d076854732f7)

## Expected Outcomes

By the end of Sprint 2, we aim to integrate the previous projects together successfully under one unified environment and implement improved features to the product.

- Previous projects are merged together in one unified environment
- The code after integration can run smoothly on our environment
- Measures have been applied to ensure the kids-friendly and safe conversation between robot and children
- An algorithm to detect struggling letters have been implemented
- AI-generated illustrations have been added to the UI

# Sprint3: Development

## Sprint Goal:

In this sprint, our primary goal is to complete the continuing tasks from Sprint 2 as well as to work on 5 user stories as outlined below:

### 1. Continuing tasks from Sprint 2

- Prerequisites: Integration of previous projects
  - Merge Bluering and Boxjelly with Redback under ROS2 env
- US3: Finish integration of the struggling-letter algorithm

### 2. Implement more enhanced features to the product

- US2: Add gestures/motions to robot using ChatGPT
- US5: Move UI to a web version
- US6: Enhance a selection of words in line with a child's interest
- US7: Log session data
- US8: Annotate ROSbag

## Sprint Backlog:

Note that "Estimated Workload" indicate the estimation for this sprint and does not necessarily indicate the entire story points for the given user story. For the continuing requirements and user stories, partial story points will be allocated for each sprint. "Estimated Story Points Left" is the estimated amount of the total story points left for this user story after this sprint.

### 1. Continuing tasks from Sprint 2

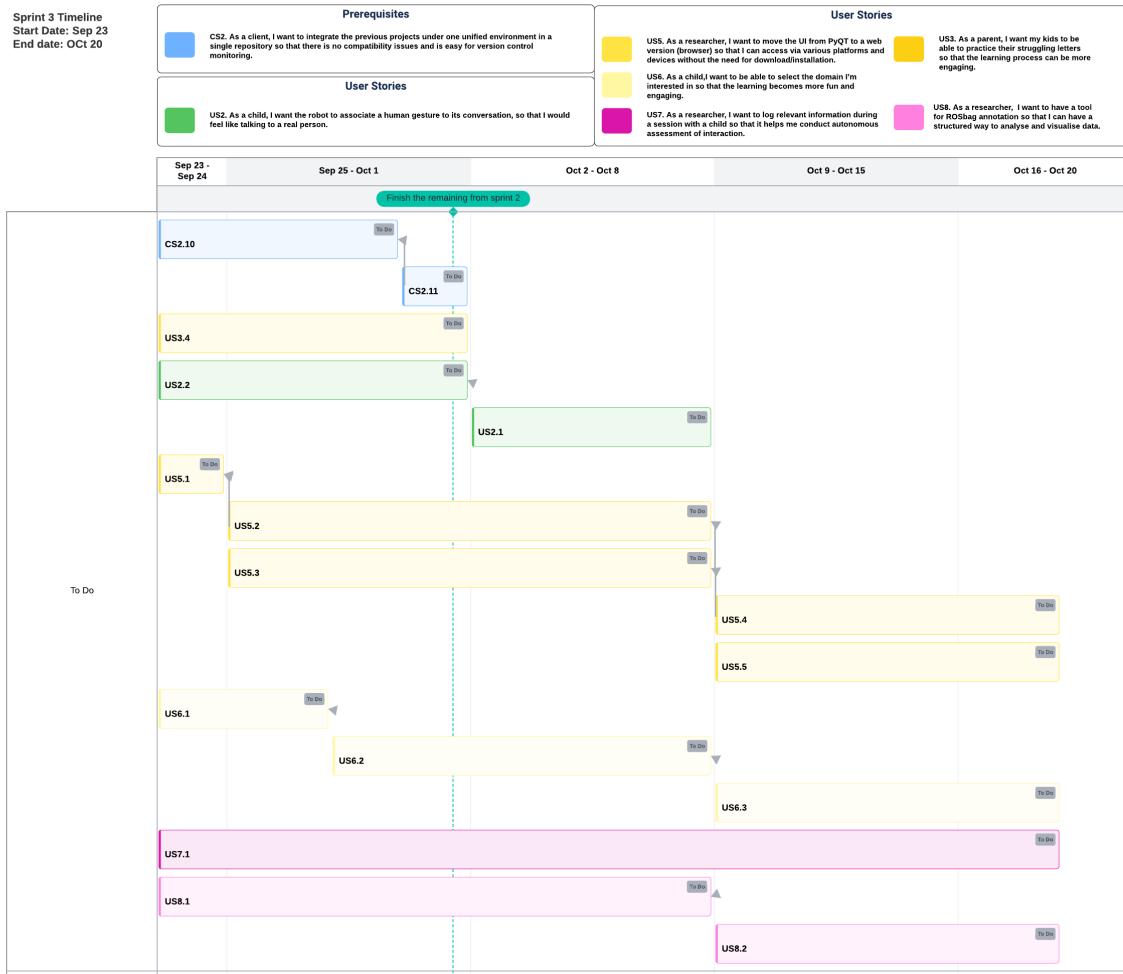
Requirement	ID	Specification	Priority	Estimated Workload	Estimated Story Points Left
Integration of previous projects	CS2	All three projects should be merged into a single repository with features integrated	Must	22	0
					Total: 22 0

Epic	ID	User Story	Priority	Estimated Workload	Estimated Story Points Left
Improvement of CoWriter	US3	As a parent, I want my kids to be able to practice their struggling letters so that the learning process can be more engaging.	Must	8	0
					Total: 8 0

### 2. User stories to complete

Epic	ID	User Story			Priority	Estimated Workload	Estimated Story Points Left
Improvement of conversation using ChatGPT	U S2	As a child,	I want the robot to associate a human gesture to its conversation,	so that I would feel like talking to a real person	Cou ld	21	0
Improvement of CoWriter	U S5	As a researcher,	I want to move the UI from PyQt to a web version (browser)	so that I can access via various platforms and devices without the need for download/installation	Sho uld	35	0
	U S6	As a child,	I want to be able to select the domain I'm interested in	so that the learning becomes more fun and engaging	Cou ld	34	0
Upgrade ROSbag data management	U S7	As a researcher,	I want to log relevant information during a session with a child	so that it helps me conduct autonomous assessment of interaction	Sho uld	27	0
	U S8	As a researcher,	I want to have a tool for ROSbag annotation	so that I can have a structured way to analyse and visualise data	Sho uld	26	0
					Total: 143		

## Sprint 3 Timeline



Link: [https://lucid.app/lucidspark/2ce2d1e7-c0da-4a5d-8e9b-8d7bd50b1105/edit?viewport\\_loc=-1507%2C-954%2C5554%2C2713%2C0\\_0&invitationId=inv\\_31a23a07-2f99-4fde-921b-40b8f5fb408](https://lucid.app/lucidspark/2ce2d1e7-c0da-4a5d-8e9b-8d7bd50b1105/edit?viewport_loc=-1507%2C-954%2C5554%2C2713%2C0_0&invitationId=inv_31a23a07-2f99-4fde-921b-40b8f5fb408)

## Expected Outcomes

By the end of Sprint 3, we aim to finish integrating the previous projects under a single repository and implement improved features to the product.

- All three previous projects are integrated together and can run under one unified environment
- The struggling-letter algorithm is further refined using trajectory instead of output and is successfully integrated
- NAO can make a variety of gestures/motion
- UI has been migrated to a web version
- NAO can suggest a selection of words in line with a child's interest
- Logging and annotation functionalities are enabled

# | Sprint Review

# Sprint 1 Review

Sprint 1 review meeting was held on 25 August between the client and dev team. The main agendas were to review the project scope, out-of-scope, and requirements (user stories) and to brief on the Sprint 2 plan. Overall, the client was happy with the current direction where the team is going and provided some tips and recommendations on implementing features for the next sprint.

Details on the discussion can be found below:

Discussion Item	Client Feedback
Scope	Client agrees with the current scope outlined
Requirements	Client agrees with the current requirements
Sprint 2 plan	<ul style="list-style-type: none"><li>• For US3, useful library called "fastdtw" can be used to compute the distance<ul style="list-style-type: none"><li>• way to measure how good of a child writing is based on the comparison between hand-writing and template (we should set a threshold for improving)</li></ul></li><li>• For US4, the illustration can be generated from AI</li></ul>
Questions & Feedback	<ul style="list-style-type: none"><li>• For US6, robot could suggest child to move to a domain that it thinks might be interesting for the child based on their conversation</li></ul>
Remarks from client	<ul style="list-style-type: none"><li>• Should be careful with the versions of previous projects since their environments may vary</li><li>• Conduct the lab induction and test on NAO robot</li><li>• The team is currently on the right direction with the project</li></ul>

The Sprint 1 meeting minutes and record can be found [HERE](#).

# Sprint 2 Review

Sprint 2 review meeting was held on 21 September between the client and dev team. The main agendas were to 1) **review the Sprint 2 achievement and demo video**, 2) **share the Sprint 3 plan**, and lastly 3) **clarify user stories related to ROSbag data annotation** which the team will be working on the next sprint. Implementation of the Sprint 2 dev tasks and features has been documented in details in the [Development Features page](#).

Overall, the client was satisfied with the team's achievement for this sprint and acknowledged the technical challenges of integrating previous projects.

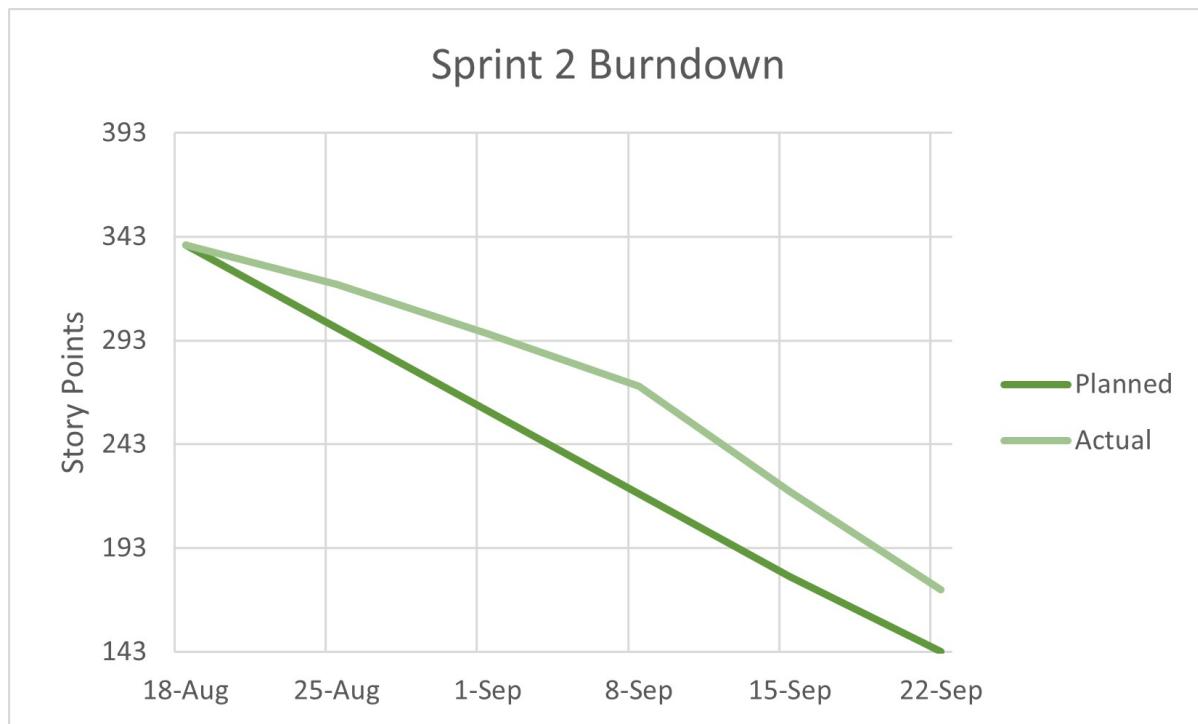
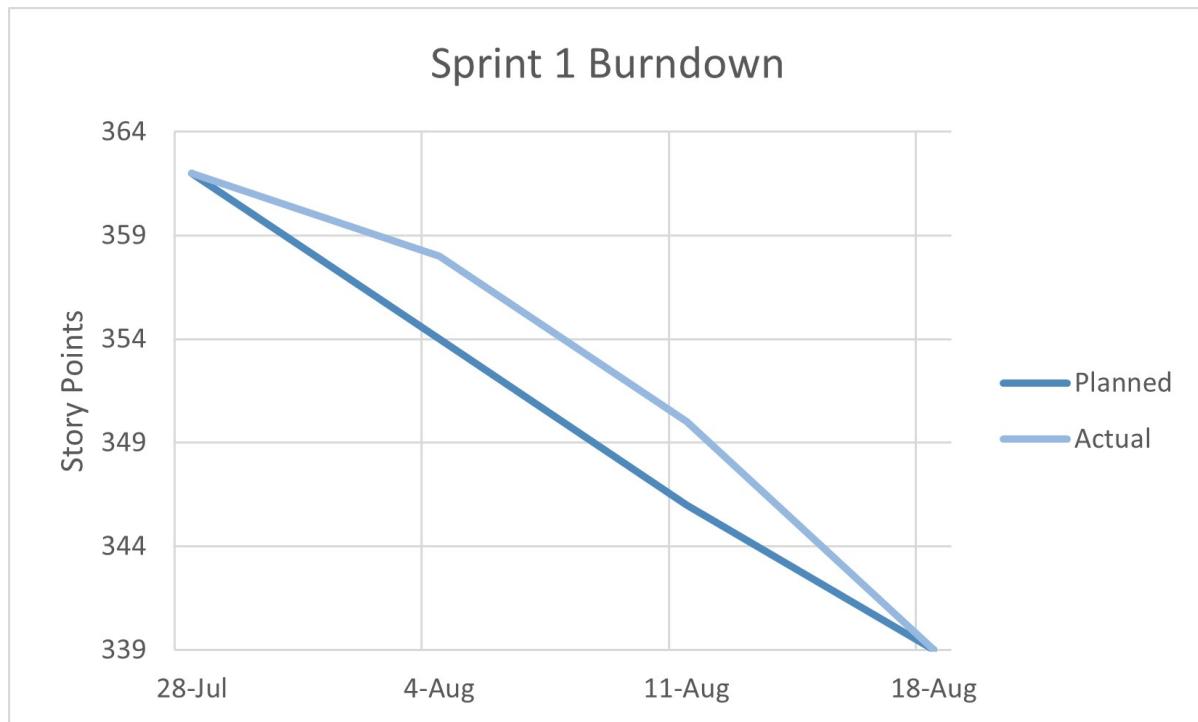
Details on the discussion can be found below:

Discussion Item	Client Feedback
Sprint 2 demo video	<p>Client is satisfied with the progress the team has made in this sprint and acknowledged the technical difficulties and complications involved with integration</p> <p>Few tips on improving US3</p> <ul style="list-style-type: none"><li>• try to use trajectory instead of the output</li><li>• can use "fastdtw" to measure distance</li><li>• setting a dynamic threshold is a tricky part</li></ul>
Sprint 3 plan	Client agrees with the plan proposed
Clarifying tasks related to ROSbag data annotation	<p>Two main tasks:</p> <ul style="list-style-type: none"><li>• logging information during the session with a kid (log activity, log video coming from the robot's camera, log audio which can be handled under the bag of data timestamp)</li><li>• annotating the ROSbag (build a system to detect failure during interaction, tag event)<ul style="list-style-type: none"><li>• can use an existing tool (<a href="https://github.com/chili-epfl/qml-rosbag-annotator">https://github.com/chili-epfl/qml-rosbag-annotator</a>) to log data from controller</li></ul></li></ul>

The Sprint 2 meeting minutes and record can be found [HERE](#).

# | Sprint Retrospective

Burndown Chart for each Sprint



# Sprint 1 Retrospective

## Meeting Outcome

Our team relies on TeamRetro to conduct our agile retrospectives seamlessly. This powerful tool has transformed the way we reflect on our sprints, enabling us to collaboratively analyze our successes and areas for improvement. With its anonymous interface, we can efficiently capture valuable feedback from every team member, ensuring that our retrospectives are not only productive but also engaging. The screenshot is shown below.

The screenshot shows the TeamRetro interface for the 'Sprint1 • AUGUST 25' retrospective. The top navigation bar includes 'BRAINSTORM', 'GROUP' (highlighted in blue), 'VOTE', 'DISCUSS', 'REVIEW', and 'CLOSE'. The current step is 'GROUP'. The sidebar on the right shows 'TEAM' (D, A, CH, R, YS), 'ACTIONS' (3), and 'AGREEMENTS'. The main area is divided into four sections: 'What went well?' (green), 'What went less well?' (red), 'What do we want to try next?' (blue), and 'What puzzles us?' (orange). Each section contains several items with upvote counts (e.g., ❤️, 🙌, 🎉, 🚀).

- What went well?**
  - everyone looks involved and motivated (2 upvotes)
  - assiduity for the meetings, everyone was always there (2 upvotes)
  - Finished Sprint 1 on time (3 upvotes)
  - We completed the sprint 1 submission on a timely manner (1 upvote)
  - Finished Sprint 1 on time (3 upvotes)
  - good communication (2 upvotes)
  - everyone looks involved and motivated (2 upvotes)
- What went less well?**
  - team meeting duration ? (a bit long) (2 upvotes)
  - The standup meeting last quite long, and the efficiency is not that much. (2 upvotes)
  - meeting efficiency need improved (1 upvote)
  - Some tasks were finished on the last day. (2 upvotes)
  - team meeting duration ? (a bit long) (2 upvotes)
  - meeting on time (1 upvote)
  - The standup meeting last quite long, and the efficiency is not that much. (2 upvotes)
- What do we want to try next?**
  - Set a fixed meeting time and date (twice a week?) (1 upvote)
  - type reports/meeting notes directly in confluence (2 upvotes)
  - We should add task timeline for management to keep the tasks are on track. (1 upvote)
  - Set a fixed meeting time and date (twice a week?) (1 upvote)
  - Add more standup meeting ( Less than 30 mins) (1 upvote)
  - maybe next time, work ahead of the meeting, and just share what we did no need to meet to work on our own ? (2 upvotes)
- What puzzles us?**
  - Estimating the time and effort needed and defining the project scope just from one meeting with client (1 upvote)
  - ROS (1 upvote)
  - Nao qi framework (1 upvote)

Link: <https://www.teamretro.com/>

From the feedback given by our teams, it can be categorized into following aspects.

What went well?	What went less well?	What do we want to try next?
Everyone get involved and motivated in Sprint 1.	The time for stand up meeting is quite long.	Set a fixed meeting duration and date.
Good communications between team members.	The meeting efficiency is not that high.	Document everything in confluence.
Finished Sprint 1 on time with a good quality.	Haven't agreed on a time that works out for everyone. (No well scheduled meeting.)	Add task timeline for management to ensure the tasks are on track. Add more stand up meeting.(At least twice a week) Set agenda for each meeting to make it more efficient.

When reviewing the progress of the first sprint, everyone agree that they were get involved and motivated in Sprint 1. According to the Trello board, the total workloads for the first sprint were properly assigned. It also shows that each team member is fully contributing to the project. For each meeting in the first sprint, the meeting time was well controlled to provide functional outcomes for monitoring the quality of the project. However, most of us pointed out that the efficiency for the meeting is not high enough. Therefore, for our next sprint, we should definitely organize the meeting in a good manner to make it more efficient.

Overall, the first sprint went smoothly and successfully as planned.

# Sprint 2 Retrospective

## Meeting Outcome

What went well?	What went less well?	What do we want to try next?	What puzzles us?
Meetings are held as planned. 👉 4 ❤️ 1	I missed a meeting sorry 😞 ☺ 3 👉 1 😢 1 😤 1	Give order to the task, so that if one task hasn't been finished, other people can start other task simultaneously, not waiting for finish. 👉 4	Which should we use for speech to text, open ai whisper or google cloud ☺ 1
Holding enough meetings. ☺ 1 👉 2 😊 1	The progress of the project went slowly. 👉 1 👉 2 😢 1	better dev environment (no need to rebuild the image/containers etc each time) ❤️ 3 👉 1	How to boost sprint productivity better ☺ 3 👉 1 😢 1
environment setup guide is better than all previous projects ❤️ 2 😊 2 👉 1	A little confused at the beginning, don't know what to do for each user story. 👉 4 ❤️ 1	Set dev tasks at a early stage so we exactly know what to do for each user story and plan in advance 👉 3 👉 1	Docker... ☺ 4 👉 1
Containerizing the project makes all team member have the same develop environment. 👉 3 👉 2	too late to find out we need some effort to upgrade ROS1 to ROS2 👉 4	Clear task distribution at the beginning of the next sprint. 👉 2 👉 1	there is a lot of stuff to do during sprint 3 👉 2 😊 1
I'm falling in love with the confluence ☺ 1 👉 2 😊 1 😢 1	Fail to estimate the workload in sprint 2. 👉 3 ❤️ 1 😢 1	Set a reasonable and detailed timeline for next sprint 👉 3	
Divided the dev tasks so everyone was working on something (last week) ☺ 1 👉 2 😊 1	Docker is a bit tricky to use, but it will be fine 😊 👉 2 😢 1 😤 1		
Last week is very productive ☺ 1 👉 2 😊 1	Task distribution is not clear at the beginning of this sprint. 👉 4 😢 1		
Documented the process on Confluence 👉 2 ❤️ 1 😊 1	The first two weeks of Sprint 2 was not so productive 👉 3 😢 2		
the last week 👉 2	Underestimated the workload of setting up the env 👉 4		
Managed to figure out environment set up at the end ☺ 1 👉 3 👉 1	Scrum master didn't lead the team well in this sprint. ☺ 1 👉 4		
the second client meeting (sprint review) 👉 3 ❤️ 1	environment setup at start, but in the end everyone could run the project 👉 2 😊 1		
	underestimate the effort of setting up environments 👉 5		

Link: <https://www.teamretro.com/>

From the feedback given by our teams, it can be categorized into following aspects.

What went well?	What went less well?	What do we want to try next?
Progress are documented in Confluence.	The progress of the project went slowly at the beginning.	Give order and timeline to the task. (Already been done)
Meeting is well planned and managed.	Underestimated the workload for this sprint.	Clearly distributed the task at the beginning of the sprint.
Take the meeting as planned.	Unable to complete all the task for this sprint.	A proper estimate on workload for next sprint.
A excellent developing environment has been set up.		

Task distribution is not clear at the beginning of the sprint.

After taking the sprint 1 retrospective, our team improved a lot in managing the meeting, which we performed less well in sprint 1. More meetings are held in sprint 2. However, the progress went slowly at the beginning of the sprint. Because most of our group members are inexperienced in handling real-world projects. It gets confused when setting tasks for user stories and distributing the tasks at the beginning. As it was the first time for us to engage in such a venture, there was a notable learning curve that needed to be navigated. Meanwhile, the lack of prior exposure results in overestimating the workload for some tasks. Even though we fixed the problem later and became more productive, our group was still unable to finish all the planned tasks on time.

Since now all of our group members are familiar with this project, a more clear distribution of the task will be given at the beginning of the next sprint. Meanwhile, a proper and reasonable estimate of workload will be given for the next sprint.

# | Ethical Considerations

## Intro

Due to the involvement of early childhood education in the project, addressing ethical issues is a critical task. This is to ensure that the design and execution of these product are carried out with extra care, prioritizing the well-being and development of children. Below are the key ethical factors that should be considered when creating and deploying a program that uses the NAO robot to teach children how to write letters, along with possible measures to take to minimise the impact.

## Ethical Principles

### 1. Privacy and Data Security

#### Data Collection and Storage

- Ensure that any data collected during interactions with the NAO robot, such as handwriting samples, audio recordings, or visual data, is securely stored and anonymized to protect the children's privacy.
- Obtain informed consent from parents or legal guardians regarding data collection and storage.

#### Consent and Transparency

- Clearly communicate to parents, guardians, and educators the types of data collected, how it will be used, and for what purposes.
- Allow parents and guardians to opt-out of data collection and usage if they have concerns about their child's privacy.

### 2. Child Safety

#### Physical Safety

- Implement safety features to ensure that the NAO robot does not pose any physical harm or danger to children.
- Regularly test and maintain the robot to prevent any malfunction that could lead to accidents.

#### Emotional Safety

- Monitor children's emotional reactions to the robot's teaching methods and content to ensure they are not experiencing distress while doing the testing process.
- Provide stop mechanisms for children if they encounter emotional challenges during the learning process.

#### Educational Safety

- Ensure that the robot are using right, healthy, and constructive conversation with children.
- Ensure that the robot's interactions are age-appropriate and align with educational goals.
- Avoid using crude language and obscene words while communication.

### 3. Equity and Inclusivity

#### Access and Opportunity

- Ensure that the program is accessible to a diverse range of children, regardless of their socioeconomic status, abilities, or geographic location.
- Take steps to bridge the digital divide and provide equal access to educational opportunities.

#### Avoiding Bias

- Carefully curate and design content to avoid reinforcing stereotypes, biases, or discriminatory ideas.
- Monitor and correct any unintended biases that may emerge during the interaction between the robot and children.

### 4. Educational Efficacy

#### Pedagogical Effectiveness

- Continuously evaluate and improve the program's effectiveness in teaching letter writing, focusing on evidence-based pedagogical methods.
- Involved with educationalist to setup and improve the education process.
- Ensure reasonable and effective cyclic learning, such as review all struggled letters.

#### Transparent Evaluation

- Share evaluation results with parents, guardians, educators, and relevant stakeholders to maintain transparency and accountability in the program's educational outcomes.
- Provide parents and children with meaningful and helpful learning feedback.

## Potential Ethical Problem

Creating a program that utilizes the NAO robot to teach children how to write letters must be done with a strong commitment to ethical considerations. We consider and conclude our current ethical consideration, but continuous monitoring, evaluation, and adaptation are key to upholding these ethical principles throughout the program's lifecycle. This means we will keep explore and consider the future potential ethical issues, and update this ethical part.

# | Cyber Security Analysis

## 1. Data Privacy and Protection:

The system collects user data, especially from children, it must adhere to data privacy regulations. Ensure that sensitive data is properly encrypted and stored securely. When data is transferred between the robot, laptops, and servers, it should be encrypted to prevent interception by malicious actors.

In terms of the logging and annotation module, consider what data will be logged, where it will be stored, and who will have access to it. Ensure that any logs containing sensitive information are properly protected.

- **Potential Attacks:**

Data breaches, unauthorized access, eavesdropping.

- **Consequences:**

Exposure of sensitive child and user data, legal and regulatory penalties, damage to reputation.

- **Possible Mitigations:**

- Encrypt sensitive data both at rest and in transit.
- Implement access controls and user authentication.
- Regularly audit data access and usage.
- Comply with data privacy regulations (e.g., GDPR, Privacy Act).
- Limit data collection to what is necessary for the project's functionality.

## 2. Child Safety:

Given that the project involves interactions with children, there's a responsibility to ensure that the system does not expose children to inappropriate content, including inappropriate responses from ChatGPT. When interacting with children, should be programmed to follow strict conversational guidelines to prevent any harmful or inappropriate content from being generated.

- **Potential Attacks:**

Exposure to inappropriate content, harmful interactions.

- **Consequences:**

Emotional harm to children, reputational damage.

- **Possible Mitigations:**

- Implement content filtering and moderation for ChatGPT.
- Supervise interactions with children.
- Educate children and parents about responsible use.

## 3. Web Security:

we are moving the UI to a web-based platform, so we need to ensure that the web application is designed with security best practices in mind. This includes protecting against common web vulnerabilities.

- **Potential Attacks:**

Cross-site scripting (XSS), SQL injection, cross-site request forgery (CSRF), session hijacking.

- **Consequences:**

Data theft, unauthorized actions, defacement of the web UI, loss of user trust.

- **Possible Mitigations:**

- Use a secure web framework with built-in protections.
- Implement input validation and output encoding.
- Use HTTPS to encrypt data in transit.
- Employ security headers (e.g., Content Security Policy) to mitigate XSS.
- Regularly update web components and libraries.

## 4. Third-party Libraries:

The project relies on third-party libraries and APIs (such as ChatGPT, Google Cloud, ROS). We need to make sure these components are up-to-date and secure. Vulnerabilities in third-party components can be exploited.

- **Potential Attacks:**

Supply chain attacks, vulnerabilities in third-party code.

- **Consequences:**  
Malicious code execution, data breaches, system compromise.
- **Possible Mitigations:**
  - Use reputable, well-maintained libraries.
  - Keep libraries up-to-date with security patches.
  - Conduct regular security assessments of third-party code.
  - Monitor for vulnerability disclosures in used libraries.

## 5. Secure Handling of API Keys and Tokens

OpenAI tokens, Google application credentials and other API keys, which are used for accessing third-party APIs, should be handled securely. Storing them directly in code (we found in one of the previous project), especially in public repositories, is a security risk. It's advisable to store API keys and tokens as environment variables or use a secure configuration management system to protect sensitive credentials.

- **Potential Attacks:**  
Exposure of sensitive tokens, API abuse.
- **Consequences:**  
Unauthorized data access, data leaks, unauthorized actions, API rate limiting.
- **Possible Mitigations:**
  - Store API keys and tokens as environment variables.
  - Use a secure secrets management solution.
  - Limit token privileges to the minimum necessary.
  - Rotate keys and tokens regularly.

## 6. Insecure Connections with the NAO Robot

The communication between the software components and the robot should be secured. If not properly configured, the connections could be vulnerable to eavesdropping or tampering by malicious actors. Implement encryption and authentication mechanisms to protect the integrity and confidentiality of data exchanged with the robot.

- **Potential Attacks:**  
Man-in-the-middle attacks, eavesdropping on robot commands, unauthorized control.
- **Consequences:**  
Unauthorized robot actions, privacy violations, potential physical harm.
- **Possible Mitigations:**
  - Keep the robot libraries up-to-date with security patches.
  - Use secure communication protocols between the software and robot.
  - Use strong authentication mechanisms.
  - Physically secure the robot to prevent tampering.

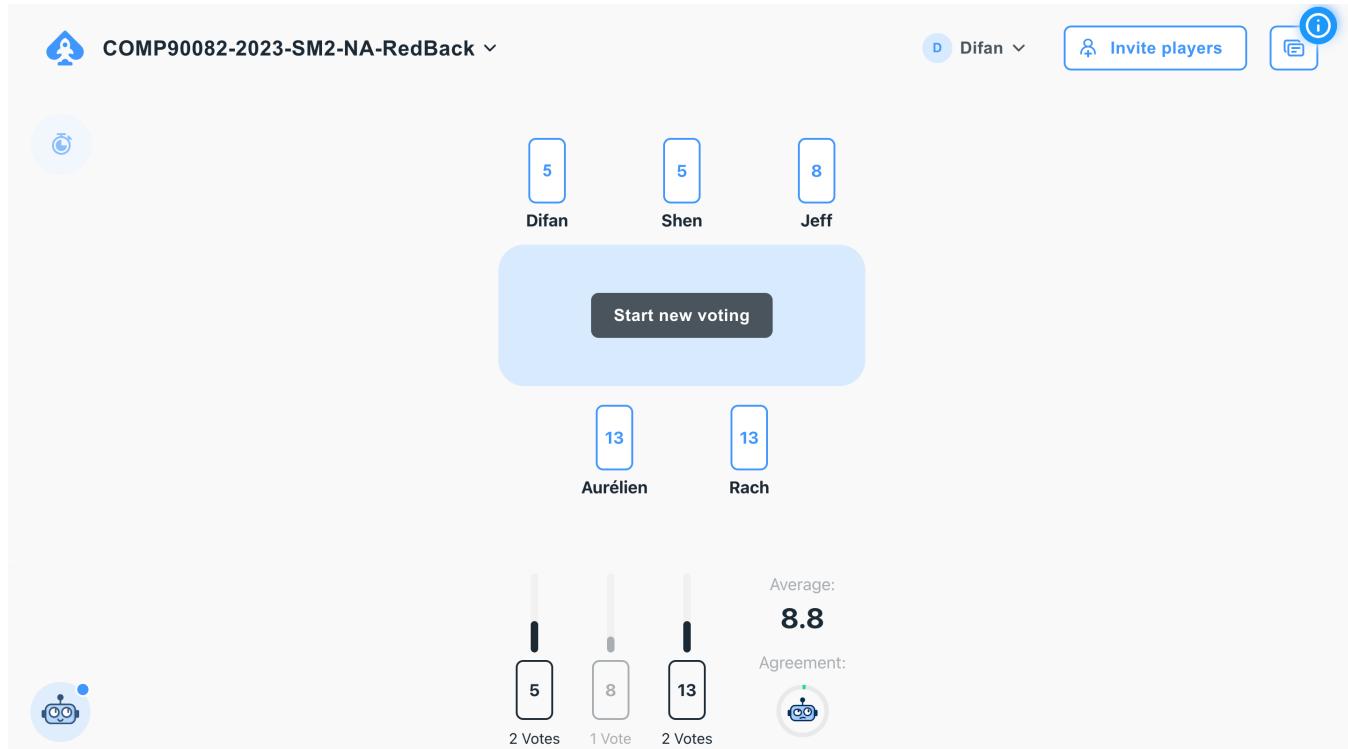
## 7. Container Security

Docker containers are used in the project, so there are security considerations to address. Exposing unnecessary ports within containers can create potential security vulnerabilities. Ensure that only essential ports are exposed, and follow best practices for securing Docker containers, including using official base images, updating containers regularly, and restricting container privileges.

- **Potential Attacks:**  
Container breakout, privilege escalation, unauthorized access.
- **Consequences:**  
Unauthorized access to host system, data leaks, container compromise.
- **Possible Mitigations:**
  - Follow container security best practices.
  - Minimize exposed ports and limit container privileges.
  - Use official base images and regularly update containers.
  - Monitor container activity for anomalies.

# Appendix

## Appendix A - Planning Poker



## Appendix B - Trello

The Trello board is organized into four columns: Questions, Backlogs, To Do, Doing, and Done. The Backlogs column contains several User Story cards for Sprint 2, including:

- User Story 1: 1.3. Test code on robot/ Simulate on computer (Due 19 Aug - 22 Sep, Points: 6, AP, CL, DW, EK, YS)
- User Story 2: 2.7. Encapsulate the environment in an image accordingly (Due 19 Aug - 22 Sep, Points: 3, AP, CL, DW, EK, YS)
- User Story 2: 2.8. Test the code to avoid any incompatibility issue. (Due 19 Aug - 22 Sep, Points: 3, AP, CL, DW, EK, YS)
- User Story 2: 2.9. Integrate them in this single repository. (Due 19 Aug - 22 Sep, Points: 31, AP, CL, DW, EK, YS)
- User Story 2: 2.10. Test the code to avoid any incompatibility issue. (Due 19 Aug - 22 Sep, Points: 31, AP, CL, DW, EK, YS)

The Done column contains cards for Sprints 1 and 2, including:

- Sprint 1: User Story 2 (2.4. Select one appropriate version of each software used in the different projects.) (Due 7 Aug - 18 Aug, Points: 2, AP, CL, DW, EK, YS)
- Sprint 1: User Story 2 (2.5. List all the commands required to create the developing environment) (Due 7 Aug - 18 Aug, Points: 5, AP, CL, DW, EK, YS)
- Sprint 1: User Story 2 (2.6. Grant necessary credentials (e.g. OpenAi)) (Due 7 Aug - 18 Aug, Points: 1, AP, CL, DW, EK, YS)
- Sprint 1: User Story 3 (3.4 Export our Confluence space to the Github repository.) (Due 7 Aug - 18 Aug, Points: 1, CL, DW, EK, YS)

## | Archives

Pages to store prev submissions in case we need to track changes and versions

# Review previous projects

<https://github.com/aplaire/compare-shape-learning/>

[https://github.com/aplaire/compare\\_cowriter](https://github.com/aplaire/compare_cowriter)

	Bluering	Redback	Boxjelly
What have they achieved (that differentiate from other teams)?	<ul style="list-style-type: none"> <li>Audio captured automatically without needing to press a button</li> <li>Bounding boxes customised for the shape of each letter to guide writing</li> </ul>	<ul style="list-style-type: none"> <li>Upgrade from ROS 1 to ROS 2</li> <li>Moved UI to the web browser (only child UI)</li> <li>Visualisation tool to track robot's writing in a plot</li> </ul>	
What have achieved, but need to be improved?		<ul style="list-style-type: none"> <li>Writing letters of the word in the same window instead of separate windows</li> <li>Need to press a button in order to capture audio</li> <li>Enabled the word selection for children's interest but needs further refinement and prompt engineering</li> </ul>	<ul style="list-style-type: none"> <li>More graceful termination of program (currently, ROS terminal will display some error msg)</li> <li>Ability for NAO to learn from full word demonstrations, rather than individual letters, but the learning algorithm is the same as the original one.</li> <li>Need to press a button to capture audio</li> <li>No bounding boxes</li> </ul>
What was their out-of-scope?	<ul style="list-style-type: none"> <li>Enabled the word selection in line with the child's domain of interest</li> <li>Add motions to the robot (&lt; need to double-check)</li> </ul>	<ul style="list-style-type: none"> <li>Implement a logging and annotation module using the ROS4HRI package</li> <li>Autonomous robot motion during conversation</li> </ul>	<ul style="list-style-type: none"> <li>Enabled the word selection in line with the child's domain of interest</li> </ul>
Environment?	<ul style="list-style-type: none"> <li>Ubuntu 20.04</li> <li>Python 3.8</li> <li>ROS 1 noetic</li> <li>libqi (C++) + libqi-python</li> </ul>	<ul style="list-style-type: none"> <li>Ubuntu 22.04</li> <li>Python 3.10 (src section) + Python 2.7 (test section)</li> <li>ROS 2 humble</li> <li>NAOqi SDK 2.8.6</li> </ul>	<ul style="list-style-type: none"> <li>Ubuntu 20.04</li> <li>Python 3.8</li> <li>ROS 1 noetic</li> <li>libqi-python</li> </ul>
Current issues?	<ul style="list-style-type: none"> <li>Incomplete setup instruction (missing dependencies, python modules)</li> <li>GUI not launching</li> </ul>	<ul style="list-style-type: none"> <li>Firefox crashing for some (Aurelien and Difan)</li> <li>Chat not available at the moment due to OpenAI credential</li> </ul>	

Note: Difan and Aurélien have computers with an ARM architecture, so they need to emulate an AMD architecture to install and launch Choregraphe. Thereby, it makes their VM really slow (not native architecture) and this is why they have issues like Firefox crashing or really slow code compilation.