

# AppAgent: Multimodal Agents as Smartphone Users

Chi Zhang\* †, Zhao Yang\*, Jiaxuan Liu\*, Yuchen Han, Xin Chen,  
Zebiao Huang, Bin Fu, Gang Yu♦,

Tencent

(\* Equal contributions, † Project Leader, ♦ Corresponding Author)

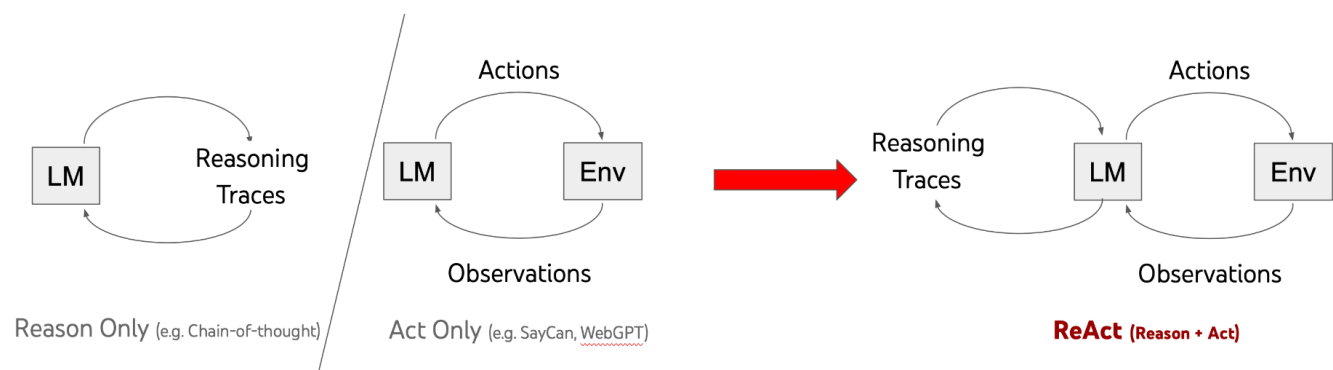
Presented by:  
John Tan Chong Min

# Motivation

- Create a smart AI assistant to help to navigate various apps
- Graphical User Interface (GUI)-based, no need for backend integration and likely more secure
- GUI-based interface control can possibly generalise better across apps, because the way of processing information is largely similar regardless of the backend used by app provider

# Recap: ReAct (Reasoning and Acting) Framework

- LLMs do not have access to the environment - Using **Observations**, align LLMs to the environment
- LLMs may not be good at one-step reasoning – Using **Thoughts**, generate action plan



**ReAct**

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.  
**Act 1:** `Search[Apple Remote]`  
**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

**Thought 2:** Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.  
**Act 2:** `Search[Front Row]`  
**Obs 2:** Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

**Thought 3:** Front Row is not found. I need to search Front Row (software) .  
**Act 3:** `Search[Front Row (software)]`  
**Obs 3:** Front Row is a discontinued media center software ...

**Thought 4:** Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.  
**Act 4:** `Finish[keyboard function keys]` ✓

ReAct: Synergizing Reasoning and Acting in Language Models. Yao et al. 2022

# ReAct across different tasks

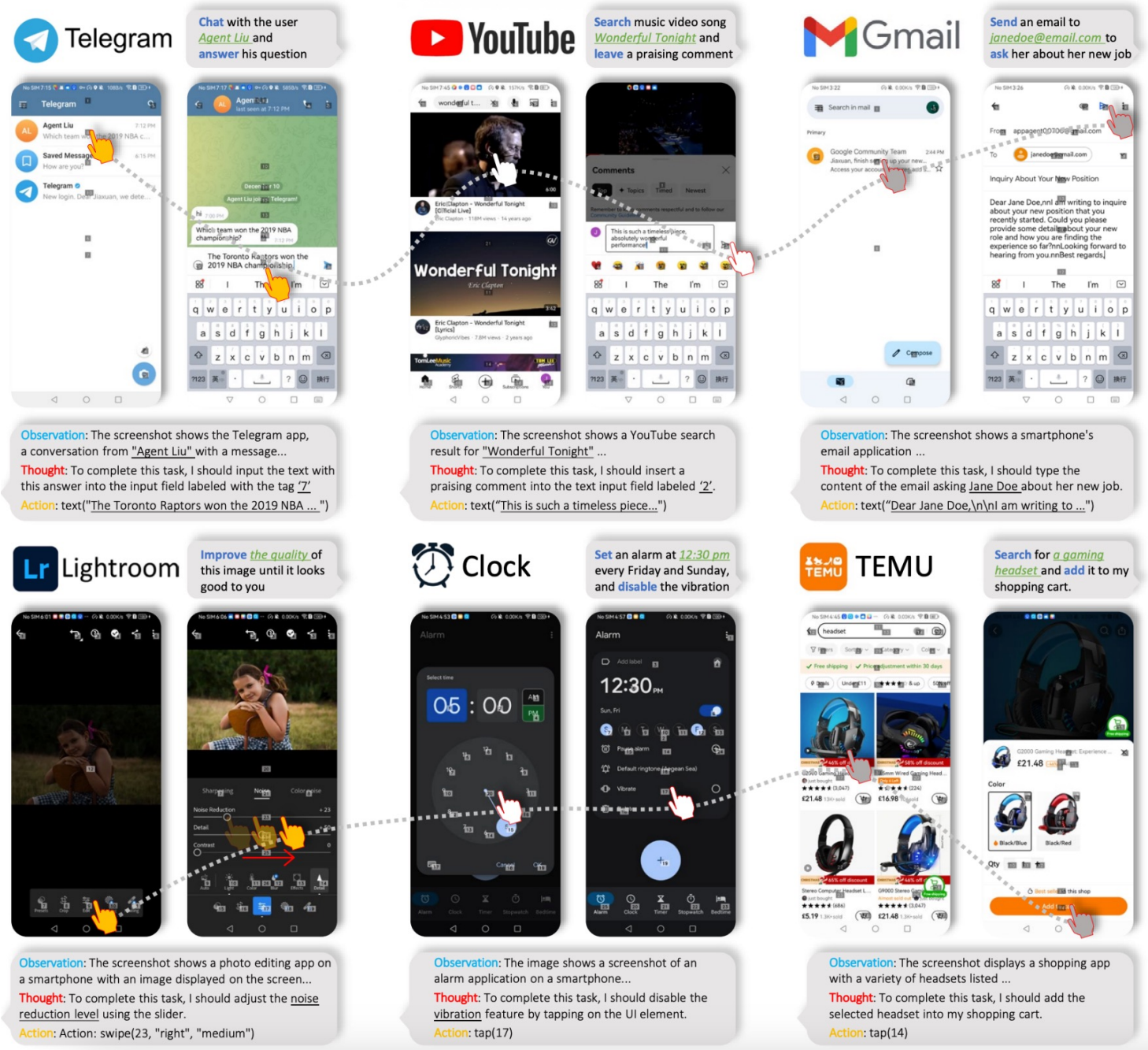
- Over **50 tasks in 10 different applications**, including social media, email, maps, shopping, and sophisticated image editing tools.

- **ReAct:**

Observation

Thought

Action





# How it works

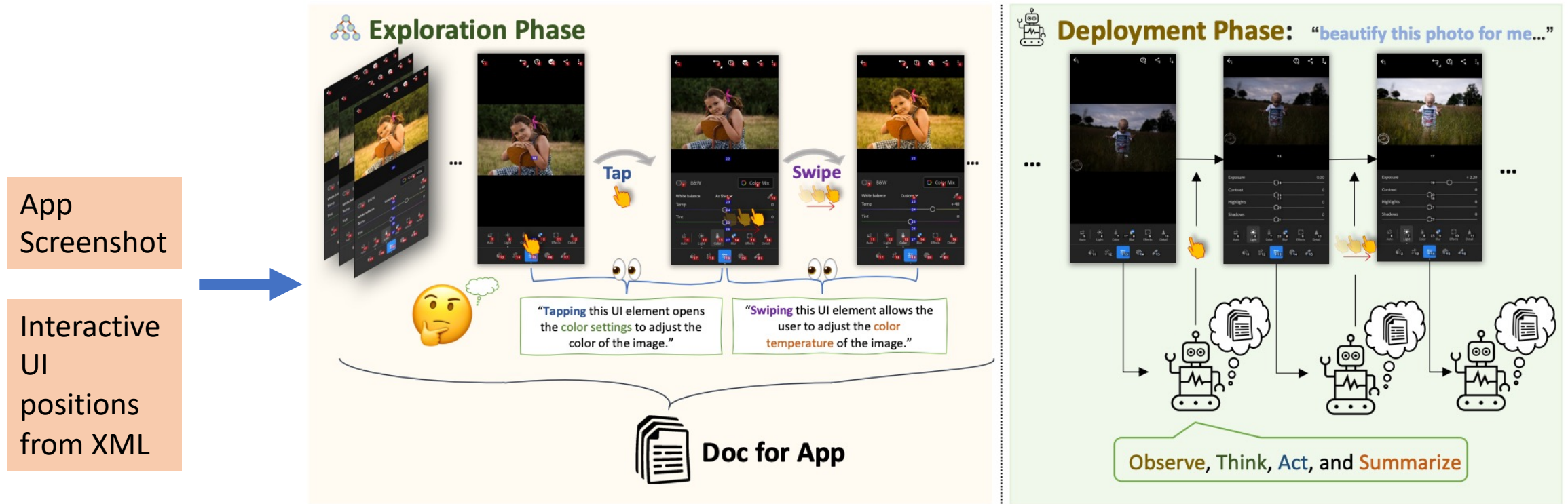


Figure 2: **Overview of our multimodal agent framework designed to operate smartphone applications.** The figure illustrates the two-phase approach of our framework. In the exploration phase, the agent interacts with a smartphone application and learns from their outcomes to create a comprehensive reference document. In the deployment phase, the agent utilizes the information compiled in this document to operate and navigate the apps effectively.

# Core Concepts for Processing

- Multimodal Query (GPT-4V) – Text + Image
- Processing of UI Elements
  - XML scraping of interactive elements
  - Overlay Unique Identifier Numbers (UIDs) on screenshot
- Simplified Action Space - Domain Specific Language
  - GPT-4V is not good at getting **pixel locations** of UI
  - Best to go by actions of interactive elements, with **exact positions from XML**

# Multimodal Query (GPT-4V)

- Input text prompt plus before image of UI to choose an action from
- Used to query action to do next, then resultant action will be fed into Android Emulator to carry out the action

```
content = [  
    {  
        "type": "text",  
        "text": prompt  
    },  
    {  
        "type": "image_url",  
        "image_url": {  
            "url": f"data:image/jpeg;base64,{base64_img_before}"  
        }  
    }  
]
```

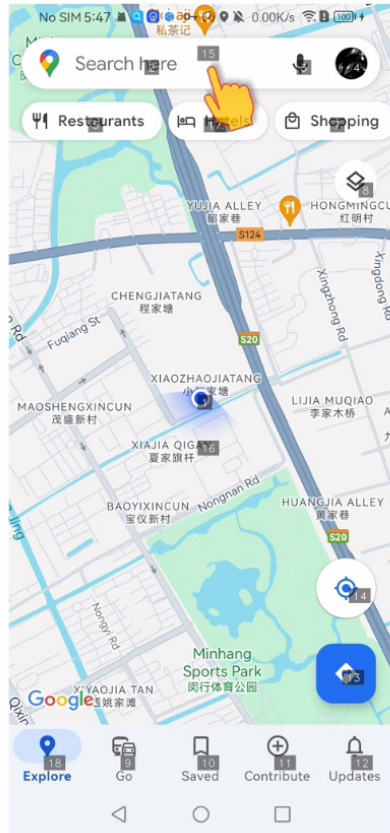
# Multimodal Query (GPT-4V)

- Input text prompt plus before and after images of UI interaction
- Used for reflection to find out what the UI does

```
content = [  
    {  
        "type": "text",  
        "text": prompt  
    },  
    {  
        "type": "image_url",  
        "image_url": {  
            "url": f"data:image/jpeg;base64,{base64_img_before}"  
        }  
    },  
    {  
        "type": "image_url",  
        "image_url": {  
            "url": f"data:image/jpeg;base64,{base64_img_after}"  
        }  
    }  
]
```



# Let's try it out!



## Observation

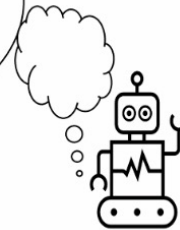
There is a UI element at the top labeled with the **number 15** indicating a **search field** where a user can input a search query or destination.

## Thought

To complete the given task, I need to **tap on the search field** to start inputting the destination "Tencent Shanghai Branch".

## Action

tap(15)



## 2 Views for AppAgent

**View 1:** A real-time screenshot showing the app's interface (image)

**View 2:** An XML file detailing the interactive elements

Interactive Elements UID are overlaid on the screenshot for better precision

# A very simplified example (GPT-4V Query)



How do I search for "Tencent Shanghai Branch". The element UID numbers are overlaid on the picture. Answer in Observation, Thought, Action format.

Action space includes Tap(element UID), Text(element UID, text).

```
content = [
  {
    "type": "text",
    "text": prompt
  },
  {
    "type": "image_url",
    "image_url": {
      "url": f"data:image/jpeg;base64,{base64_img_before}"
    }
  }
]
```

Diagram illustrating the JSON structure for the GPT-4V query. A blue arrow points from the "text" field to the prompt text "How do I search for 'Tencent Shanghai Branch'...". Another blue arrow points from the "image\_url" field to the map image on the left.

# A very simplified example (GPT-4V output)



## ChatGPT

### Observation:

The image shows a screenshot of a map application interface on a mobile device. There is a search bar at the top with the placeholder text "Search here". The interface elements are marked with unique identifier numbers (UIDs) for interaction.

### Thought:

To search for "Tencent Shanghai Branch," I would need to use the search bar at the top of the screen. The appropriate action would be to tap on the search bar to bring up the keyboard and then enter the desired text.

### Action:

#### Verdict:

- Wrong UID, but correct action
- GPT-4V OCR for small numbers may not be that great yet

1. Tap(1) - This would select the search bar.
2. Text(1, "Tencent Shanghai Branch") - This would enter the text "Tencent Shanghai Branch" into the search bar.



# Using Overlay without referencing UIIDs



Describe the image and how I would go about to search for "Tencent Shanghai Branch"



## ChatGPT

The image appears to be a screenshot of a digital map application interface, possibly Google Maps, showing an urban area with various roads and points of interest labeled in both English and Chinese characters. The interface includes various elements such as a search bar, buttons for "Restaurants," "Hotels," and "Shopping," and icons for different functionalities such as "Explore," "Go," "Saved," and others.

If you wanted to search for "Tencent Shanghai Branch" using this map application, you would click on the search bar at the top where it says "Search here" and then type in "Tencent Shanghai Branch." The application should then process your search query and provide you with the location and information regarding Tencent's Shanghai Branch, if available in its database. The screenshot shows that the map is centered around an area with street names and other locations in Chinese characters, suggesting that it is indeed in a Chinese-speaking region, which would be consistent with the search for a Shanghai location.

## Verdict:

- GPT-4V is already quite good at processing generic app screenshots
- Key challenge is how to navigate the UI

# Solution: Using Overlay + UID Information



How do I search for "Tencent Shanghai Branch". The element UID numbers are overlaid on the picture. Answer in Thought, Observation, Action format. Action space includes Tap(element UID), Text(element UID, text).

UID 15 is the search bar, UID 18 is the explore button.



## ChatGPT

Thought:

The user needs to search for "Tencent Shanghai Branch" on a map application. The search bar is typically used for entering search queries.

Observation:

There is a search bar present with the UID 15. This is likely where text needs to be entered to perform the search.

Verdict:

- Need to provide description of overlay UID for better results

Action:

To conduct the search, one would tap the search bar and enter the text of the search query.

Tap(UID 15), Text(UID 15, "Tencent Shanghai Branch").

# Can Text-only Interface work?

- Quite possible to do everything as text, if we can describe the image in text
- May be better than just using the image as GPT-4V may be able to distinguish fine details yet
- But image as separate abstraction space might unlock more information!



**You**

How do I search for "Tencent Shanghai Branch" on Google Maps? Answer in Observation, Thought, Action format.

Action space includes Tap(element UID), Text(element UID, text).

UID 15 is the search bar, UID 18 is the explore button.



**ChatGPT**

**Observation:**

You want to find the location of "Tencent Shanghai Branch" on Google Maps.

**Thought:**

To achieve this, you should enter the query into the search bar of Google Maps and then initiate the search. Since you've identified the relevant UI elements (UID 15 for the search bar and UID 18 for the explore button), these will be the focus of the action steps.

**Action:**

1. **Text(UID 15, "Tencent Shanghai Branch"):** This action inputs the text "Tencent Shanghai Branch" into the search bar (UID 15).
2. **Tap(UID 18):** After entering the search query, this action taps the explore button (UID 18) to initiate the search.

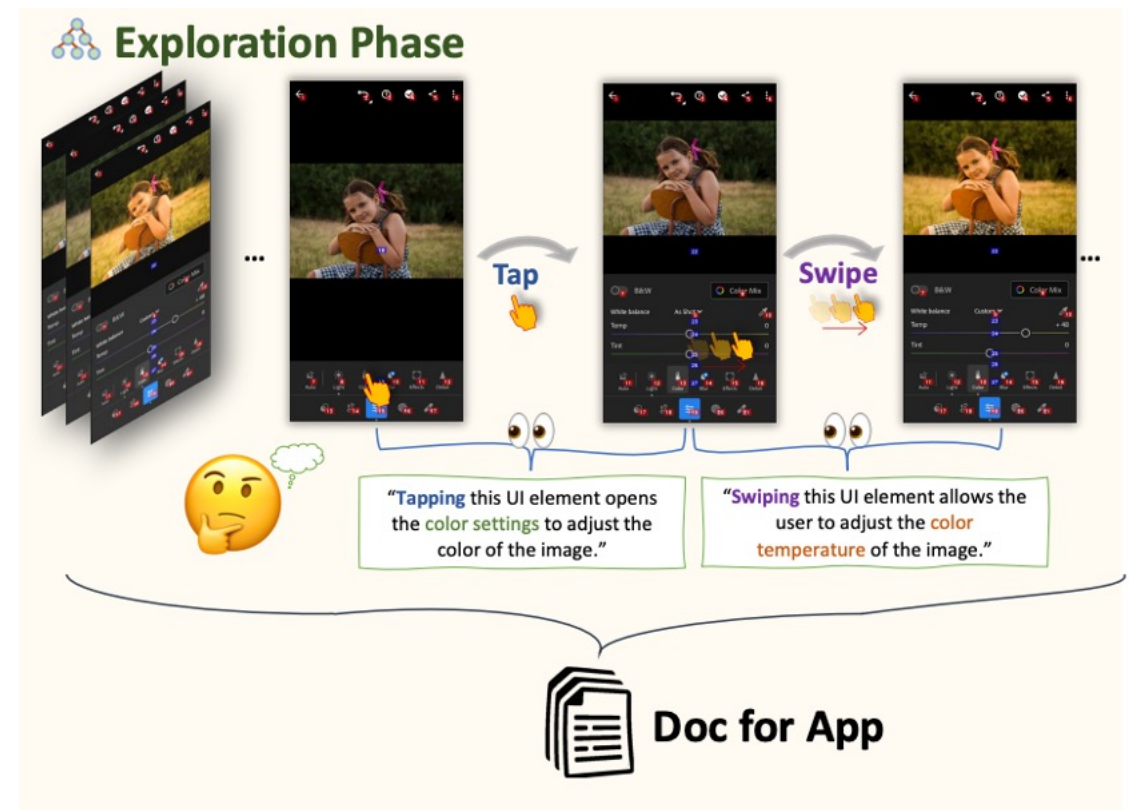


# Simplified Domain Specific Language (DSL)

- Rather than doing direct cursor space (x, y) manipulation, we do actions by UI elements
- **Limits the action space** and leads to better outcomes!
- **Actions:**
  - **Tap (UID):** Tap on UI element
  - **Long press (UID):** Long press on UI element
  - **Swipe (UID, direction, distance):** Mimics a finger swipe on UI element towards direction with distance
  - **Text (text):** Inputs text into a UI element
  - **Back ():** Returns to previous screen; Essentially the Android phone Back button
  - **Exit ():** Ends process when task is done or maximum steps reached

# Simplified Domain Specific Language (DSL)

- Rather than doing direct cursor space (x, y) manipulation, we do actions by UI elements
- Limits the action space and leads to better outcomes!
- `Tap(element : int)`: This function simulates a tap on the UI element numbered on the screen. For example, `tap(5)` would tap the element labeled '5'.
- `Long_press(element : int)`: This function emulates a long press (for 1 second) on a UI element.
- `Swipe ( element : int, direction : str, dist : str)`: It allows the agent to swipe on an element in a specified direction (up, down, left, right) and distance (short, medium, long). For instance, `swipe(21, "up", "medium")` would swipe up on element '21' for a medium distance.
- `Text(text : str)`: To bypass inefficient virtual keyboard typing, this function inputs text directly into an input field when a virtual keyboard is visible. For example, `text("Hello, world!")` inputs the string "Hello, world!".
- `Back()`: A system-level function that helps the agent return to the previous UI page, especially useful for exiting irrelevant pages.
- `Exit()`: A specialized function is employed to conclude processes, typically invoked upon successful task completion.



# Exploration Phase

Goal-Oriented Self-Exploration, or from Human Demonstration

Creates documentation for UI, adds existing documentation as context when generating new one.

# Take an Action

- Either:
  - **Self-Explore:** Same as Deployment Agent Template (covered later)
    - Goal-Oriented Self-Exploration
  - **Guided:** Actions will be given by human demonstrator

# Prompt for Reflection after Action

- **Given:** Task, Action, Before Image, After Image
- **Output:** Goal-Oriented Decision
  - BACK – Action leads to an incorrect path, go back one step
  - INEFFECTIVE – Action changed nothing on the screen
  - CONTINUE – Action changed something on the screen, but not useful
  - SUCCESS – Action changed something on the screen, and is part of task
- **Output:** Documentation of UI for all Decisions other than INEFFECTIVE
- BACK and INEFFECTIVE helps to truncate the search space by not performing similar actions in the future

```
self_explore_reflect_template = """I will give you screenshots of a mobile app before and after <action> the UI element labeled with the number '<ui_element>' on the first screenshot. The numeric tag of each element is located at the center of the element. The action of <action> this UI element was described as follows:
```

```
<last_act>
```

```
The action was also an attempt to proceed with a larger task, which is to <task_desc>. Your job is to carefully analyze the difference between the two screenshots to determine if the action is in accord with the description above and at the same time effectively moved the task forward. Your output should be determined based on the following situations:
```

# Documentation

- Serves as memory to inform agent how a UI element behaves!

```
Documentations retrieved for the current interface:  
Documentation of UI element labeled with the numeric tag '1':  
This UI element is clickable. The UI element activates the text input field, enabling the user to type in their search query.  
  
Documentation of UI element labeled with the numeric tag '24':  
This UI element is clickable. The UI element functions as a search icon, which when tapped, opens the search interface allowing users to search for specific contents or users.
```



# Prompt for Documentation Learning (aka Memory + Reflection)

- **Given:** Task, Action, Before Image, After Image, Previous Documentation (if any)
- **Output:** Documentation for functionality of UI
- Prompted to give generic documentation instead of focusing on the specifics, e.g. person
  - Better abstraction space for generalisation!

```
tap_doc_template = """I will give you the screenshot of a mobile app before and after tapping the UI element labeled with the number <ui_element> on the screen. The numeric tag of each element is located at the center of the element. Tapping this UI element is a necessary part of proceeding with a larger task, which is to <task_desc>. Your task is to describe the functionality of the UI element concisely in one or two sentences. Notice that your description of the UI element should focus on the general function. For example, if the UI element is used to navigate to the chat window with John, your description should not include the name of the specific person. Just say: "Tapping this area will navigate the user to the chat window". Never include the numeric tag of the UI element in your description. You can use pronouns such as "the UI element" to refer to the element."""
```

# Prompt to redefine from Previous Documentation

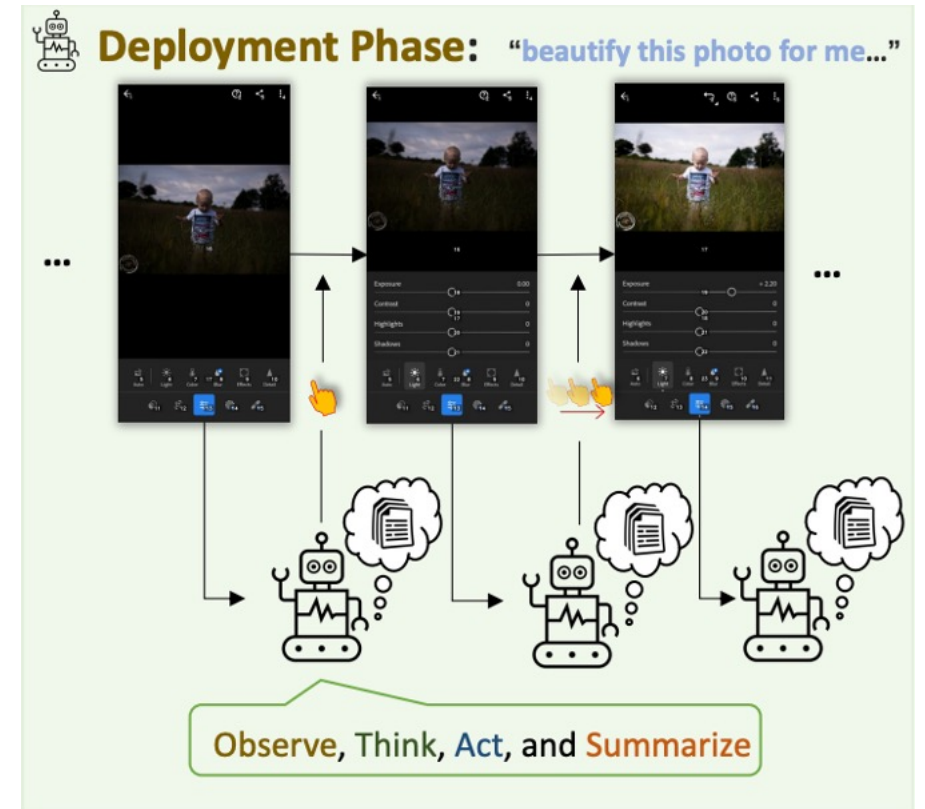
- Serves as a way to learn dynamically from changing environments
- Combine all kinds of uses of a particular UI
- Similar to how memories of the brain shift over time to suit the new context

```
refine_doc_suffix = """"\nA documentation of this UI element generated from previous demos is shown below. Your generated description should be based on this previous doc and optimize it. Notice that it is possible that your understanding of the function of the UI element derived from the given screenshots conflicts with the previous doc, because the function of a UI element can be flexible. In this case, your generated description should combine both. Old documentation of this UI element: <old_doc>"""
```

# Deployment Phase

Agent receives screenshot + documentation + function list + action history

Chooses next function



# Deployment Agent Template

- Given: Task, Before Image, Documentation, Function List (DSL), Past action history
- Output: Function for next action, Summary of past actions
- Each function contains (similar to Visual ChatGPT):
  - Input parameters
  - Input parameter type
  - Use case description
  - Example use case

# Deployment Agent Prompt (Part 1/2)

```
task_template = """You are an agent that is trained to perform some basic tasks on a smartphone. You will be given a  
smartphone screenshot. The interactive UI elements on the screenshot are labeled with numeric tags starting from 1. The  
numeric tag of each interactive element is located in the center of the element.
```

You can call the following functions to control the smartphone:

## 1. tap(element: int)

This function is used to tap an UI element shown on the smartphone screen.

"element" is a numeric tag assigned to an UI element shown on the smartphone screen.

A simple use case can be tap(5), which taps the UI element labeled with the number 5.

## 2. text(text\_input: str)

This function is used to insert text input in an input field/box. text\_input is the string you want to insert and must be wrapped with double quotation marks. A simple use case can be text("Hello, world!"), which inserts the string "Hello, world!" into the input area on the smartphone screen. This function is usually callable when you see a keyboard showing in the lower half of the screen.

## 3. long\_press(element: int)

This function is used to long press an UI element shown on the smartphone screen.

"element" is a numeric tag assigned to an UI element shown on the smartphone screen.

A simple use case can be long\_press(5), which long presses the UI element labeled with the number 5.

# Deployment Agent Prompt (Part 2/2)

4. `swipe(element: int, direction: str, dist: str)`

This function is used to swipe an UI element shown on the smartphone screen, usually a scroll view or a slide bar.

"element" is a numeric tag assigned to an UI element shown on the smartphone screen. "direction" is a string that represents one of the four directions: up, down, left, right. "direction" must be wrapped with double quotation marks. "dist" determines the distance of the swipe and can be one of the three options: short, medium, long. You should choose the appropriate distance option according to your need.

A simple use case can be `swipe(21, "up", "medium")`, which swipes up the UI element labeled with the number 21 for a medium distance.

The task you need to complete is to `<task_description>`. Your past actions to proceed with this task are summarized as follows: `<last_act>`

Now, given the following labeled screenshot, you need to think and call the function needed to proceed with the task. Your output should include three parts in the given format:

Observation: `<Describe what you observe in the image>`

Thought: `<To complete the given task, what is the next step I should do>`

Action: `<The function call with the correct parameters to proceed with the task. If you believe the task is completed or there is nothing to be done, you should output FINISH. You cannot output anything else except a function call or FINISH in this field.>`

Summary: `<Summarize your past actions along with your latest action in one or two sentences. Do not include the numeric tag in your summary>`

You can only take one action at a time, so please directly call the function.""""



# My Critique

- Prompts are too verbose
  - Can be made more concise and structured
- Output format is too free-form
  - Can be constrained better with JSON output, e.g. StrictJSON

# Results

- DSL is very important (2.2% to 48.9% Success Rate using GPT4)!
- Documentation helps LLM not repeat errors and focus more on the correct steps it has done in the past
- Auto Exploration using goal-directed learning may not get everything, may need to augment with Expert Demos!

Method	Document	Action Space	SR $\uparrow$	Reward $\uparrow$	Avg. Steps
GPT4 (Baseline)	None	Raw	2.2%	0.6	4.0
	None	<b>Ours</b>	48.9%	3.5	6.9
<b>AppAgent</b>	<b>Auto. Exploration</b>	<b>Ours</b>	73.3%	5.1	4.4
	<b>Watching Demos</b>	<b>Ours</b>	84.4%	4.7	5.1
	<b>Manually Crafted</b>	<b>Ours</b>	95.6%	5.5	5.5

Succeed  
within 10  
steps

# Questions to Ponder

- Goal-Oriented Self-Exploration using LLM is great, but it can fail to get to the goal all the time as it is dependent on the in-built knowledge of LLM, and may need human demonstrations to boost knowledge
  - Random exploration? Multi-agent exploration?
- Using XML to obtain all the UI elements may not get everything that is needed (i.e. some interactions have no UI elements involved like scrolling the map on Google Maps. Can GPT-4V or another system output the UI element positions, or could there be another action space for scrolling?
- Could the overlay of UID block some critical information of the app?
- How else to incorporate multiple abstraction spaces to an agent?