Annex B: UAT Response & Fixes

Detailed Issue Analysis & Resolution

TC-014: Error Propagation - Wrong Exit Code

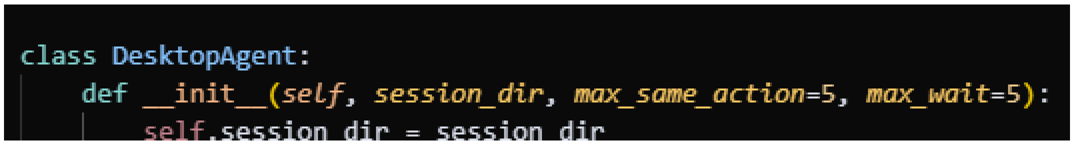
Issue Description:

- Wrong exit code was given (exited with code 1 when action failed multiple times, should be code 2)

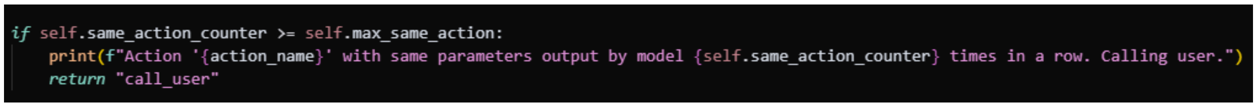
Justification:

- This is actually intended behaviour in the current system design
- When the agent reaches the maximum limit of repeated actions (5 times) or consecutive waits (5 times), it is designed to call for user intervention
- This is not a "failure" but rather a safety mechanism to prevent infinite loops
- The agent correctly returns "call_user" which should exit with code 1 (user intervention needed)

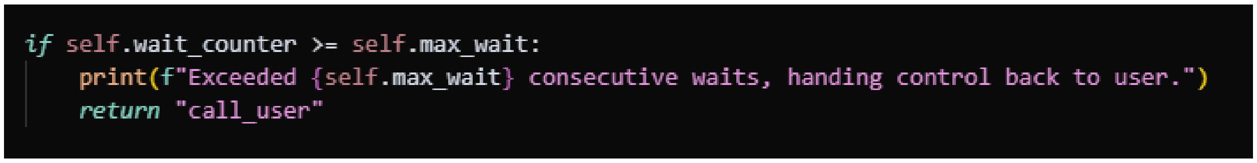
Code snippets in desktop_agent_core.py:



```python
class DesktopAgent:
    def __init__(self, session_dir, max_same_action=5, max_wait=5):
        self.session_dir = session_dir
```



```python
if self.same_action_counter >= self.max_same_action:
    print(f"Action '{action_name}' with same parameters output by model {self.same_action_counter} times in a row. Calling user.")
    return "call_user"
```



```python
if self.wait_counter >= self.max_wait:
    print(f"Exceeded {self.max_wait} consecutive waits, handing control back to user.")
    return "call_user"
```

Correct System Behaviour:

1. Exit Code 1: User intervention needed (including when agent hits retry limits)
2. Exit Code 2: Script errors, file issues, exceptions, validation failures
3. Exit Code 0: Successful completion

TC-015: Basic Instruction Transformation - Only First Instruction Extracted

Issue Description:

- Only the first instruction was extracted and recorded
- Model was not processing the complete input text

Root Cause Analysis:

- The system prompt in `prompt_optimiser.py` is not explicit enough to instruct the model to process the entire input text
- Model was stopping after processing the first instruction block

Fix Implemented:

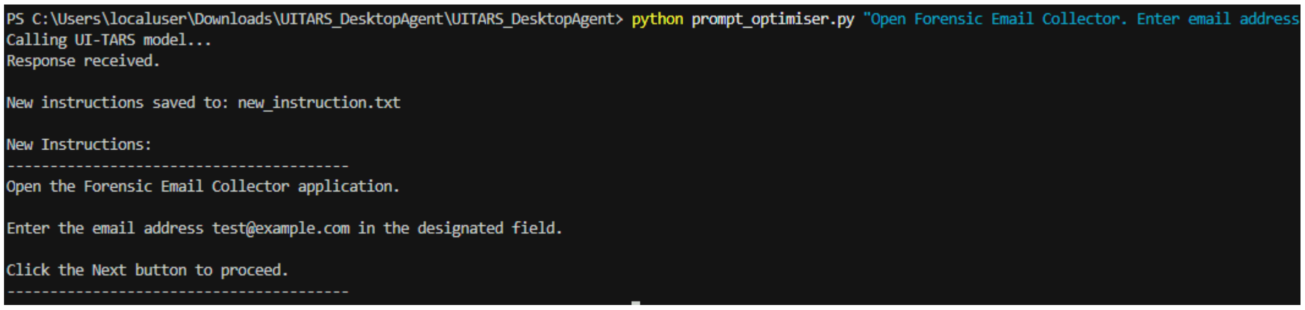
- Enhanced the system prompt to include explicit instruction and included more examples

Files Modified:

- `Prompt_optimiser.py`: Modified the system prompt to emphasis text preservation

Verification:

- Model now processes all instructions in the input text
- Complete instruction sets are extracted and transformed
- Output of TC-015:



```
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent> python prompt_optimiser.py "Open Forensic Email Collector. Enter email address
Calling UI-TARS model...
Response received.

New instructions saved to: new_instruction.txt

New Instructions:
----------------------------------------
Open the Forensic Email Collector application.

Enter the email address test@example.com in the designated field.

Click the Next button to proceed.
----------------------------------------
```

TC-016: Complex Multi-step Instruction - Missing Empty Line Separators

Issue Description:

- Steps were broken down but not separated with empty line separators
- Output format did not match expected structure

Root Cause Analysis:

- System prompt mentioned empty line separators but model was not consistently applying the empty line separator rule

Fix Implemented:

- Updated system prompt to clearly specify formatting requirements
- Added post-processing to ensure proper formatting if the model's output failed to do so

Files Modified:

- `prompt_optimiser.py`: Clarified system prompt with empty line separator requirement and added post process output method

Verification:

- Output now consistently includes empty lines between instruction steps
- Format matches expected structure for automation processing

TC-017: Special Character Handling - Password Incorrectly Extracted

Issue Description:

- Password 'P@$$w0rd!' was incorrectly extracted as 'Pw0rd'
- Special characters were being stripped from passwords and other sensitive data

Root Cause Analysis:

- The `sanitise_instruction()` function was removing special characters using regex pattern `[^\w\s.,;:()\-]`
- This was stripping important characters like @, $, ! from passwords and other data

Fix Implemented:

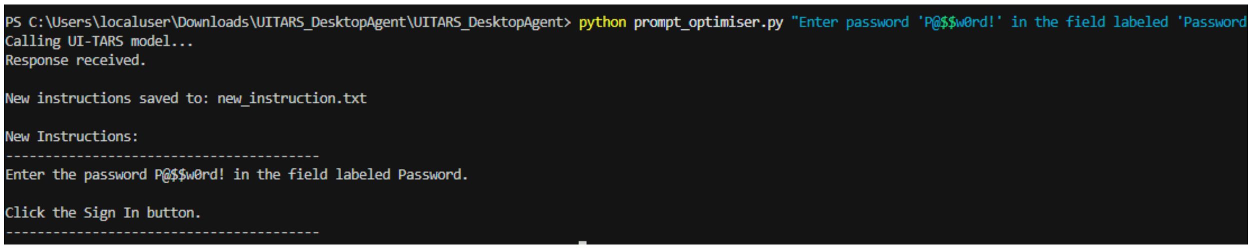
1. Rewrote `sanitise_instruction()` function to only remove quotation marks while preserving all other characters

Files Modified:

- `prompt_optimiser.py`: Enhanced prompt and sanitisation function

Verification:

- Special characters in passwords are now preserved exactly as provided
- TC-017 output:



```
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent> python prompt_optimiser.py "Enter password 'P@$$w0rd!' in the field labeled 'Password
Calling UI-TARS model...
Response received.

New instructions saved to: new_instruction.txt

New Instructions:
----------------------------------------
Enter the password P@$$w0rd! in the field labeled Password.

Click the Sign In button.
----------------------------------------
```

TC-019: Error Handling - Inconsistent Exit Codes

Issue Description:

- Gave exit code 1 when running first command (empty string), but exit code 2 when running second command (non-existent file)

- Inconsistent error handling across different validation scenarios

Root Cause Analysis:

- Different error conditions were handled by different parts of the code with inconsistent exit codes
- File reading errors used exit code 1, while argument parser errors used exit code 2

Fix Implemented:

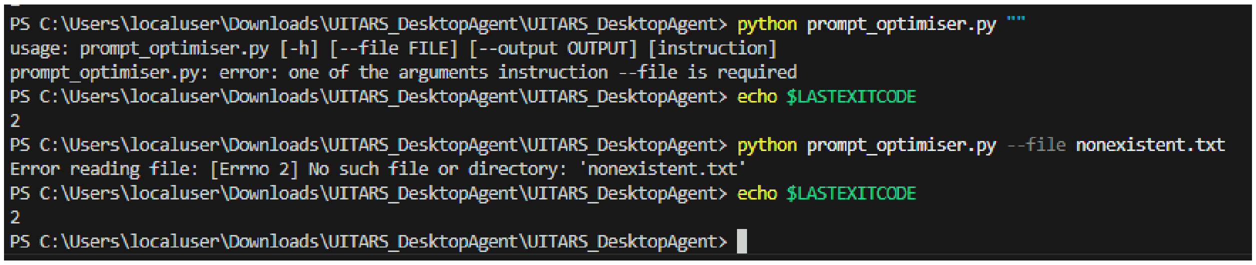
1. Standardised exit codes across all error conditions:
    - Exit code 2 for file errors and validation errors (empty instructions)
    - Exit code 1 for API/exception failures
2. Added explicit validation for empty instruction input
3. Ensured consistent error handling across all input methods

Files Modified:

- `prompt_optimiser.py`: Standardised exit codes and added validation

Verification:

- All validation errors now exit with code 2
- API failures exit with code 1
- Output of TC-019:



```
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent> python prompt_optimiser.py ""
usage: prompt_optimiser.py [-h] [--file FILE] [--output OUTPUT] [instruction]
prompt_optimiser.py: error: one of the arguments instruction --file is required
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent> echo $LASTEXITCODE
2
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent> python prompt_optimiser.py --file nonexistent.txt
Error reading file: [Errno 2] No such file or directory: 'nonexistent.txt'
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent> echo $LASTEXITCODE
2
PS C:\Users\localuser\Downloads\UITARS_DesktopAgent\UITARS_DesktopAgent>
```

TC-020: File Input Processing - Missing Empty Line Separators

Issue Description:

- Steps were broken down but not separated with empty line separators
- Same formatting issue as TC-016 but specifically for file input processing
- Worked when instructions were enclosed with double quotations

Root Cause Analysis:

- Same root cause as TC-016: insufficient emphasis on empty line separators in system prompt
- The fixes applied for TC-016 also resolve this issue since post-processing is added

Fix Implemented:

- Same fixes as TC-016 apply to this case
- Enhanced system prompt and improved instruction processing
- No additional changes needed as the core processing logic handles both input methods consistently

Files Modified:

- `prompt_optimiser.py`: Same modifications as TC-016

Verification:

- File input processing now produces properly formatted output with empty line separators
- Consistent behavior between direct input and file input methods

---

New requirements:

Requirement 1: Mobile Number Argument Support

Implementation details:

1. Added `--mobile` argument support to `run_agent_loop.py`
2. Implemented `$Mobile` placeholder replacement functionality
3. Added proper argument parsing and validation

Files Modified:

- `run_agent_loop.py`: Added mobile number argument support

Usage Example:

- Users can now provide both `--otp` and `--mobile` arguments

- `python run_agent_loop.py SecondIinstruction.txt --otp 123456 --mobile +1234567890`

Requirement 2: Session grouping under same UUID

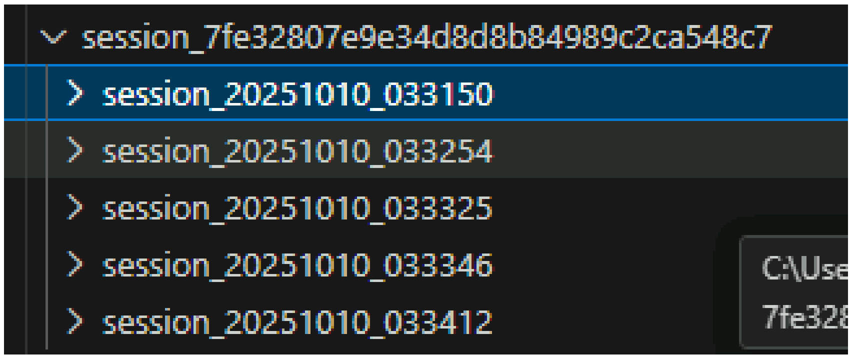
Implementation details:

1. Use a single parent session ID for all instructions in a batch
2. Each instruction now gets a timestamp-based subfolder under the same parent UUID

Files Modified:

- `run_agent_loop.py`: groups sessions together under same UUID

New session folder structure:



```
session_7fe32807e9e34d8d8b84989c2ca548c7
    session_20251010_033150
    session_20251010_033254
    session_20251010_033325
    session_20251010_033346
    session_20251010_033412
```

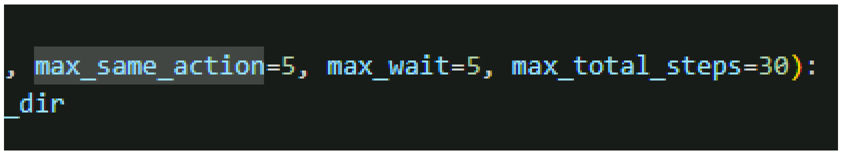
Requirement 3: Enhance Infinite Loop Detection

Implementation details:

1. Include a total step limit to maximum 30 steps per instruction to address previous limitation of tracking same actions only

Files Modified:

- `run_desktop_agent_core.py`: Include additional counter to keep track of number of actions carried out per instruction and return exit code 1 when it exceeds limits
- Please also note that the limit counters can be changed as according to your requirements and use cases freely:



```
, max_same_action=5, max_wait=5, max_total_steps=30):
_dir
```