User Study Task Grading Scheme

Task 1 (25 Minutes)

Question 1: What is the primary goal of the agent?

For example, if analyzing a Pizza Preparing Agent, you might say: "The agent's goal is to prepare a pepperoni pizza by selecting the correct ingredients and baking it according to the recipe."

Expected Answer:

The agent is supposed to implement a new "income tracking feature" to a personal finance tracker python project. It's supposed to first read the code, execute tests, and edit the source code until the tests are satisfied.

Grading Scheme:

• *1 point for fully generic description (e.g. "implement feature as described in PR")* • *2 points for very high-level answer (e.g. "implement an income tracking feature in project")* • *3 points for full answer*

Points: **3**

Question 2: Did the agent complete its task?

Expected Answer:

No, the task was not completed.

Grading Scheme:

• *1 point for right answer*

Points: **1**

Question 3: What are the main action categories performed by the agent?

For a Pizza Preparing Agent, you might list actions like "Select ingredients," "Mix dough," and "Bake pizza," focusing on action types, not specific details.

Expected Answer:

List directory, Read file, Update file, Execute test script, Install dependencies

Grading Scheme:

• *0 points for BS (e.g. "adding functionality of income tracking and integrating it into the CLI")* • *1 point for very high-level categories (e.g. "observations, replacements, package installations")* • *2 points for specific action categories as noted in expected answer*

Points: **2**

Question 4: What is the most frequently performed action?

Expected Answer:

Update file

Grading Scheme:

• *1 point for right answer*

Points: **1**

Question 5: What is the key challenge or issue the agent encounters during the task?

*For a Pizza Preparing Agent, you might say "The agent has trouble correctly setting the temperature of the oven. The oven keeps returning a **OvenNotAvailable** error, indicating that it is not set up correctly." **Make sure to name the specific reoccurring error.***

Expected Answer:

The test script keeps failing with an IndentationError.

Grading Scheme:

• *1 point for IndentationError*

Points: **1**

Question 6: Does the agent exhibit any repetitive behavior? If so, describe one example.

*Describe an instance if there is one, where the repetitive behavior persists over at least **three** query-tool call cycles. E.g. "The agent moves the dough from the workspace to the oven, and back to the workspace 3 times in a row before turning on the oven".*

Expected Answer:

The test script fails with an IndentationError, the agent makes some changes to cli.py, and reruns the tests which return the same error. ***OR*** *The agent lists the contents of the /finance_tracker/finance_tracker directory repeatedly three times in a row at the beginning of execution.*

Grading Scheme:

• *1 point for correctly identifying repetition without description, or generic description* • *2 points for correct identification with description*

Points: **2**

Question 7: What specific changes does the agent make to the **cli.py** file when it **first** modifies it?

Copy the diff of the change, or the tool call that makes the change.

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Grading Scheme:

• *1 point for right answer (also accept full file if provided as long as it contains the change)*

Points: **1**

Question 8: Which specific files does the agent modify to implement the **income report generation** feature?

List all files that are modified to support this feature.

Expected Answer:

cli.py, reports.py, income.py:

Grading Scheme:

• *1 point if answer contains at least two of the files above* • *2 points if the answer is exactly as above*

Points: **2**

Question 9: What feedback does the agent receive after its **first** attempt to run the test script for the finance tracker, and how does it influence the next action?

Explain, or copy and paste the results of the test script execution. Explain what the agent does next.

Expected Answer:

```
The agent installs the missing dependency next.
Traceback (most recent call last):
  File "/finance_tracker/finance_tracker/tests/test_income.py", line 5, in <module>
    from finance_tracker.reports import FinancialReport
  File "/finance_tracker/finance_tracker/reports.py", line 5, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
```

Grading Scheme:

- *1 point for only identifying the right error*
- *2 points for identifying the right error and next action*

Points: **2**

Question 10: Which Python packages does the agent install over the course of the run?

List each one.

Expected Answer:

pandas, matplotlib

Grading Scheme:

- *1 point for listing exactly both*

Points: **1**

Question 11: What feedback does the agent receive from running the test script for the finance tracker **after it has solved all import issues**?

Explain, or copy and paste the results of the test script execution.

Expected Answer:

```
Traceback (most recent call last):
  File "/finance_tracker/finance_tracker/tests/test_income.py", line 7, in <module>
    from finance_tracker.cli import FinanceTrackerCLI
  File "/finance_tracker/finance_tracker/cli.py", line 41
    def do_generate_income_report(self, arg):
    ^
IndentationError: expected an indented block after function definition on line 40
```

Grading Scheme:

- *1 point for correctly identifying the error*

Points: **1**

Question 12: Which file is the agent primarily focused on modifying **towards the end** of the run?

Provide the name of the file.

Expected Answer:

cli.py

Grading Scheme:

- *1 point for correct answer*

Points: **1**

Question 13: Does the agent ever attempt to make the same change **twice** before reevaluating the tests? If so, describe one instance.

Copy the tool calls from this instance, if there is any.

Expected Answer:

```
str_replace_editor str_replace /finance_tracker/finance_tracker/cli.py --old_str ' def do_generate_income_report(self, arg):' --new_str ' def do_generate_income_report(self, arg):\n \"\"\"Generate an income report: generate_income_report [start_date] [end_date]\"\"\"\n if not self.current_user:\n print(\"Please login first\")\n return\n args = arg.split()\n start_date = args[0] if len(args) > 0 else None\n end_date = args[1] if len(args) > 1 else None\n try:\n report_data = self.income_tracker.generate_income_summary(start_date, end_date)\n print(json.dumps(report_data, indent=2))\n except Exception as e:\n print(f\"Error: {e}\")\n'
```

Grading Scheme:

- *1 point for correct identification with pasted tool call*

Points: **1**

Question 14: Analyze the final four changes the agent makes to the repository. How do these changes relate to each other?

Describe any similarities or differences in these changes. Do they all serve the same purpose, or are they trying to accomplish different things?

Expected Answer:

The last four changes appear to be completely identical, however they don't overwrite each other. Instead they are appended to each other.

Grading Scheme:

- *1 point for identifying identical changes*
- *2 points for identifying that they are appended and do not overwrite each other*

Points: **2**

Task 2 (12 Minutes)

Agent Bug Localization Task: RepairAgent

Analyze the provided trajectory to determine why RepairAgent failed to fix the CSV-75 bug in the Apache Commons CSV project. Focus on potential issues in the agent's logic, such as misinterpretations of the LLM's outputs, incorrect logic in the agent's decision-making, or other implementation errors.

Expected Answer:

There appears to be an issue with the `write_fix` action, because after the first attempted fix, the supposed fixes are not written to the file. Consequently, there are no changes made to the buggy code and the tests keep failing.

Actual Bug & Fix:

The `write_fix` tool call's implementation contained a line, which discarded any changes that were deemed to be nearly identical to the file's original contents. Removing this line forces the tool to apply the changes, solving the problem.

Grading Scheme:

- *1 point for mentioning the tool call that writes the fix to the file as an issue. Disregard additional entries.*

Points: **1**

Task 3 (12 Minutes)

Agent Bug Localization Task: ExecutionAgent

Analyze the provided trajectory to determine why ExecutionAgent failed to set up and execute the test suite of the Gson project. Focus on potential issues in the agent's logic, such as misinterpretations of the LLM's outputs, incorrect logic in the agent's decision-making, or other implementation errors.

Expected Answer:

There appears to be an issue with the Docker integration of the agent. It seems that whenever the agent creates a Dockerfile, the container is built and started, however the tool call's return claims that the process failed.

Additional Info:
Having no access to the environment and being unable to execute commands in the native terminal is not the problem. This is by design. The agent is directed to set up a Docker container to be able to access more commands. See tool response: "'java' is not permitted. Allowed commands at this point are: cat, find, grep, head, less, ls, more, tail, tree. You would have access to more commands once you have written a Dockerfile which would automatically instantiate a docker container in which you can run more commands."

Actual Bug & Fix:

Due to an issue with the agent's Docker integration, the agent failed to connect to the newly built and started container. Reinstalling Docker solved this issue.

Grading Scheme:

- *1 point for mentioning an issue with Docker environment or the Dockerfile as an issue. Disregard additional entries.*

Points: **1**