User Study Participant Report Card

Participant Key: fe6c
Participant Group: B

Overview of Performance

Task 1: 13 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **51 / 84** (Scaled*) Correct: **61%** In Top **0,58** Percentile
Task 2: **0 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

> *The goal is to enhance the `finance_tracker` project by adding income tracking functionality to complement the existing expense, budget, user, and reporting features.*

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.*

Points: **3 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

Gather relevant information', 'summarize requirements', 'code implementation and show the diff', 'run tests and solve runtime errors'

Expected Answer:

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

Points: **1 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

run tests and modify the code to solve errors.

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **0 / 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".*

Recorded Answer:

Yes, maybe the step where it try to solve the same runtime errors again and again.

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.

Points: **1 / 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.

Recorded Answer:

--old_str 'from finance_tracker.reports import FinancialReport' --new_str 'from finance_tracker.reports import FinancialReport
from finance_tracker.income import IncomeTracker'

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **1 / 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.

Recorded Answer:

finance_tracker/cli.py, finance_tracker/reports.py

Expected Answer:

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

Points: **1 / 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.

Recorded Answer:

ModuleNotFoundError: No module named 'pandas'. The agent installs the package, and allows the execution go with no reference errors.

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'*

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas

Expected Answer:

pandas, matplotlib

Points: **0 / 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.

Recorded 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

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

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

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'

Points: **0 / 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?

Recorded Answer:

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.

Points: **0 / 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.

Recorded Answer:

Agent's logic: failed to change to another method to fix the issue. Implementation: the line break conditional statements seems was incorrect at some point.

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.

Points: **0 / 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.

Recorded Answer:

Agent's decision making: the failed command outputs mention that more commands could be run in a docker container, but the agent fails to adjust the next steps to adapt it. Logic: allows the failed steps repeat multiple times.

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.

Points: **0 / 1**

User Study Participant Report Card

Participant Key: 67c3
Participant Group: B

Overview of Performance

Task 1: 2 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **12 / 84** (Scaled*) Correct: **14%** In Top **0,92** Percentile
Task 2: **0 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

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.

Points: **0 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

Execute solutions and take over corrections

Expected Answer:

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

Points: **0 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

Ask and add error fixes

Expected Answer:

Update file

Points: **0 / 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.

Recorded Answer:

Missing libraries, non-working feature

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **0 / 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".*

Recorded Answer:

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.

Points: **0 / 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.

Recorded Answer:

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **0 / 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.

Recorded Answer:

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

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'

Points: **0 / 2**

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

List each one.

Recorded Answer:

Expected Answer:

pandas, matplotlib

Points: **0 / 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.

Recorded Answer:

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
```

Points: **0 / 1**

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

Provide the name of the file.

Recorded Answer:

financetracher/cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

I think so

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'

Points: **0 / 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?

Recorded Answer:

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.

Points: **0 / 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.

Recorded Answer:

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.

Points: **0 / 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.

Recorded Answer:

No access to the environment and therefore can't install or update anything, missing tools that are supposed to be used

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.

Points: **0 / 1**

User Study Participant Report Card

Participant Key: 0078
Participant Group: A

Overview of Performance

Task 1: 13 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **55 / 84** (Scaled*) Correct: **65%** In Top **0,50** Percentile
Task 2: **0 / 1**
Task 3: **1 / 1**

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."

Recorded Answer:

The agents goal is to implement an income tracking feature to a python project with cli functionality and integrating it into the rest of the program.

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.

Points: **2 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

Yes, the task was completed successfully.

Expected Answer:

No, the task was not completed.

Points: **0 / 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.

Recorded Answer:

Adding functionality of income tracking and integrating it into the CLI

Expected Answer:

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

Points: **0 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

adding the income report generation to the cli

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

The agent seems to have troubles with python syntax, i.e., the indentation (IndentationError)

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

Yes, it does. It requests multiple times to "Create the 'IncomeTracker' class ... to satisfy the PR description", the commit titles also indicate this with multiple "Add income report generation to CLI"

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.

Points: **1 / 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.

Recorded Answer:

```
import cmd
2
import argparse
3
from finance_tracker.expenses import ExpenseTracker
4
from finance_tracker.budgets import BudgetManager
5
from finance_tracker.users import UserManager
6
from finance_tracker.reports import FinancialReport
7
from finance_tracker.income import IncomeTracker
8
from datetime import datetime
9

10
```

```
class FinanceTrackerCLI(cmd.Cmd):
11
   """Command-line interface for the finance tracker."""
12
   prompt = "FinanceTracker> "
13
   intro = "Welcome to the Personal Finance Tracker. Type 'help' for commands."
14

15
   def __init__(self):
16
      super().__init__()
17
      self.user_manager = UserManager()
18
      self.current_user = None
19
      self.expense_tracker = None
20
      self.budget_manager = None
21
      self.report_generator = None
22

23
   def do_login(self, arg):
24
      """Login to the system: login <username> <password>"""
25
      args = arg.split()
26
      if len(args) != 2:
27
         print("Usage: login <username> <password>")
28
         return
29
      username, password = args
30
      if self.user_manager.authenticate_user(username, password):
31
         self.current_user = username
32
         self.expense_tracker = ExpenseTracker(username)
33
         self.budget_manager = BudgetManager(username)
34
         self.report_generator = FinancialReport(username, self.expense_tracker,
self.budget_manager)
35
```

```
            print(f"Logged in as {username}")
36
        else:
37
            print("Invalid username or password")
38

39
    def do_register(self, arg):
40
        """Register a new user: register <username> <password> <email>"""
41
        args = arg.split()
42
        if len(args) != 3:
43
            print("Usage: register <username> <password> <email>")
44
            return
45
        try:
46
            self.user_manager.register_user(*args)
47
            print(f"User {args[0]} registered successfully")
48
        except ValueError as e:
49
            print(f"Error: {e}")
50

51
    def do_add_expense(self, arg):
52
        """Add an expense: add_expense <amount> <category> <description> [tags] [recurring]
[period]"""
53
        if not self.current_user:
54
            print("Please login first")
55
            return
56
        args = arg.split()
57
        if len(args) < 3:
58
            print("Usage: add_expense <amount> <category> <description> [tags] [recurring] [period]")
59
            return
60
```

```
        amount, category, description = args[:3]
61
        tags = args[3].split(",") if len(args) > 3 else []
62
        is_recurring = args[4].lower() == "true" if len(args) > 4 else False
63
        period = args[5] if len(args) > 5 else None
64
        try:
65
            expense_id = self.expense_tracker.add_expense(float(amount), category, description,
66
                                        tags=tags, is_recurring=is_recurring,
67
                                        recurrence_period=period)
68
            print(f"Expense added with ID: {expense_id}")
69
        except ValueError as e:
70
            print(f"Error: {e}")
71

72
    def do_set_budget(self, arg):
73
        """Set a budget: set_budget <category> <amount> [period] [alert_threshold]"""
74
        if not self.current_user:
75
            print("Please login first")
76
            return
77
        args = arg.split()
78
        if len(args) < 2:
79
            print("Usage: set_budget <category> <amount> [period] [alert_threshold]")
80
            return
81
        category, amount = args[:2]
82
        period = args[2] if len(args) > 2 else "monthly"
83
        alert_threshold = float(args[3]) if len(args) > 3 else 0.8
84
        try:
85
            self.budget_manager.set_budget(category, float(amount), period, alert_threshold)
```

```
86
            print(f"Budget set for {category} ({period})")
87
        except ValueError as e:
88
            print(f"Error: {e}")
89

90
    def do_generate_report(self, arg):
91
        """Generate a report: generate_report <type> [start_date] [end_date]"""
92
        if not self.current_user:
93
            print("Please login first")
94
            return
95
        args = arg.split()
96
        if len(args) < 1:
97
            print("Usage: generate_report <type> [start_date] [end_date]")
98
            return
99
        report_type = args[0]
100
        start_date = args[1] if len(args) > 1 else None
101
        end_date = args[2] if len(args) > 2 else None
102
        try:
103
            report_file = self.report_generator.generate_report_pdf(report_type, start_date, end_date)
104
            print(f"Report generated: {report_file}")
105
        except Exception as e:
106
            print(f"Error: {e}")
107

108
    def do_exit(self, arg):
109
        """Exit the CLI."""
110
        print("Goodbye!")
111
```

```
        return True
112
113
    def preloop(self):
114
        """Initialize CLI state."""
115
        print("Personal Finance Tracker CLI. Type 'register' or 'login' to begin.")
```

Expected Answer:

```
7 + from finance_tracker.income import IncomeTracker
```

Points: **1 / 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.

Recorded Answer:

cli.py, report.py, income.py

Expected Answer:

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

Points: **2 / 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.

Recorded Answer:

Error (ModuleNotFoundError: No module named 'pandas'); next the agent asks the LLM what to do, the LLM suggests to install pandas with pip

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'*

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas and matplotlib

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded Answer:

An indentation error

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

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'

Points: **0 / 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?

Recorded Answer:

first test, then error so next "updated logic"

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.

Points: **0 / 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.

Recorded Answer:

The LLM suggests to update the read method to account for both \r and \n characters, but the agent misinterprets it and removes the \n (which was correct), instead of adding a new if-case for \r.

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.

Points: **0 / 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.

Recorded Answer:

The agent tries to use forbidden commands (for which it has no permission) multiple times. It also tries to chain commands, which also does not seem to work.
The LLM suggests to write a dockerfile, but the agent runs into problems.
The agent and LLM do not seem to recognize that they are running in circles.
It seems like after "The agent acknowledges that the LLM's training data is current only until October 2023.", the agent loses its "history" and tries commands again which have not worked before.

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.

Points: **1 / 1**

User Study Participant Report Card

Participant Key: 1d20
Participant Group: A

Overview of Performance

Task 1: 16 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **71 / 84** (Scaled*) Correct: **85%** In Top **0,08** Percentile
Task 2: **1 / 1**
Task 3: **1 / 1**

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."

Recorded Answer:

> *The goal is to implement a new feature as described by a PR description*

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.*

Points: **1 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

Listing contents of a directory, reading a file writing to a file, creating a new file, modifying a file via diff syntax, running bash commands

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

Modifying the contents of a file

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

The agent has trouble fixing a syntax error (the indentation error)

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

The agent modified a part of the code that was not the erroneous part of the code. It tried multiple times to fix the indentation error by modifying a different part of the code.

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.

Points: **1 / 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.

Recorded Answer:

7 + from finance_tracker.income import IncomeTracker

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **1 / 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.

Recorded Answer:

income.py, expenses.py, reports.py, cli.py

Expected Answer:

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

Points: **1 / 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.

Recorded Answer:

It gets the missing module error, and it tries to install the missing library

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'*

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas, matplotlib

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded Answer:

It encounters the indentation error

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded 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'

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'

Points: **1 / 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?

Recorded Answer:

They are all modifying the same location of the code, which is not the location that needs a fix. Their purpose is to fix the indentation error, but the edits are the same.

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.

Points: **1 / 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.

Recorded Answer:

It seems that the tool implementation is buggy and when the agent tries to modify the file, the process does not work correctly. From my inspections the agent tries to modify something, but the resulting file contents are not as expected.

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.

Points: **1 / 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.

Recorded Answer:

The main issue is that the agent was never able to create a working dockerfile to try different approaches to resolve the problem. The main problem I see in this trajectory is that the error message from one write to a dockerfile was not returned to the agent. The other logic problem that could exist is that the agent seems to not have a good instruction on that it has to first create a dockerfile to be able to run all bash commands.

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.

Points: **1 / 1**

User Study Participant Report Card

Participant Key: 33c6
Participant Group: A

Overview of Performance

Task 1: 17 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **74 / 84** (Scaled*) Correct: **88%** In Top **0,00** Percentile
Task 2: **1 / 1**
Task 3: **1 / 1**

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."

Recorded Answer:

Implementing a feature as suggested by a pull request

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.

Points: **1 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

listing files, showing file content, make modifications to files, create files, execute tests, install dependencies

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

editing files

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

Trying to add code to generate an income report. After multiple tries of making this change the tests always fail with an IndentationError.

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

The agent adds the code for generating an income report 6 times in a row before execution ends. It is always the same code change.

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.

Points: **2 / 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.

Recorded Answer:

object
.....
.....{2}
tool
.....
:.....str_replace_editor str_replace /finance_tracker/finance_tracker/cli.py --old_str 'from finance_tracker.reports import FinancialReport' --new_str 'from finance_tracker.reports import FinancialReport\nfrom finance_tracker.income import IncomeTracker'
args
.....
:..... null

Expected Answer:

```
7 + from finance_tracker.income import IncomeTracker
```

Points: **1 / 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.

Recorded Answer:

cli.py, reports.py, expenses.py, income.py

Expected Answer:

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

Points: **1 / 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.

Recorded Answer:

```
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'

The agent then installs the pandas dependency using pip
```

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'
```

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas, matplotlib

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded 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

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

Yes. Tool call 1: object
.....
.....{2}
tool
.....
:.....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'
args
.....
:..... null

Tool call 2:
.....ob

ect
.....
.....{2}
tool
.....
:.....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'
args
.....
:..... null

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'

Points: **1 / 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?

Recorded Answer:

They are all making the same change to cli.py.

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.

Points: **1 / 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.

Recorded Answer:

The LLM calls the write_fix tool with a meaningful fix trying to modify the line 58 to also handle other line endings. There seems to however be a bug in the implementation of applying the proposed fix to the repository. The file is not correctly modified but the line to modify is only removed.

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.

Points: **1 / 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.

Recorded Answer:

At the beginning the LLM tries to run a bash command to setup openjdk eventhough it is not allowed to do this at that state. In the second tool call it tries to run an unallowed command. The third LLM response is empty as the LLM rejects the prompt. Similar problems occur multiple times. The agent continues to fail to create a script for seting up the docker environment. After a while the LLM is able to successfully call the tool to write a docker file, but the agent fails to start the docker image. It tries this multiple more times but it is not clear why it fails.

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.

Points: **1 / 1**

User Study Participant Report Card

Participant Key: 49f7
Participant Group: B

Overview of Performance

Task 1: 16 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **67 / 84** (Scaled*) Correct: **80%** In Top **0,17** Percentile
Task 2: **0 / 1**
Task 3: **1 / 1**

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."

Recorded Answer:

> *The agents goal is to add an income tracking feature to a personal finance tracker written in python*

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.*

Points: **2 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

Listing directories (ls), printing file contents (cat), writing file contents (str_replace_editor), running program (python3)

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

str_replace_editor (replacing parts of a file)

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

It is unable to indent properly (IndentationError)

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

Yes, it keeps "replacing" the line ' def do_generate_income_report(self, arg):' in /finance_tracker/cli.py with the same value, not knowing that the actual problem is that 'def __init__(self):' has no proper body.

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.

Points: **2 / 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.

Recorded Answer:

```
--- a/finance_tracker/cli.py
+++ b/finance_tracker/cli.py
@@ -4,6 +4,7 @@ from finance_tracker.expenses import ExpenseTracker
 from finance_tracker.budgets import BudgetManager
 from finance_tracker.users import UserManager
 from finance_tracker.reports import FinancialReport
+from finance_tracker.income import IncomeTracker
 from datetime import datetime

 class FinanceTrackerCLI(cmd.Cmd):


(str_replace_editor str_replace /finance_tracker/finance_tracker/cli.py  --old_str 'from finance_tracker.reports import FinancialReport' --new_str 'from finance_tracker.reports import FinancialReport)
```

Expected Answer:

```
7 + from finance_tracker.income import IncomeTracker
```

Points: **0 / 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.

Recorded Answer:

cli.py, README.md

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

```
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'

it then proceeds to installing pandas using 'pip install pandas'
```

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'
```

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas, matplotlib

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded 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

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded 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):
        """Generate an income report: generate_income_report [start_date] [end_date]"""
        if not self.current_user:
            print("Please login first")
            return
        args = arg.split()
        start_date = args[0] if len(args) > 0 else None
        end_date = args[1] if len(args) > 1 else None
        try:
            report_data = self.income_tracker.generate_income_summary(start_date, end_date)
            print(json.dumps(report_data, indent=2))
        except Exception as e:
            print(f"Error: {e}")
'
It does this repeatedly without changing anything, re-runs the test and repeats.
```

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'

Points: **1 / 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?

Recorded Answer:

The agent does the same changes five times, sometimes running the test in between and sometimes not

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.

Points: **1 / 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.

Recorded Answer:

Test 'org.apache.commons.csv.CSVParserTest::testGetLineNumberWithCR' fails. However, the agent only is shown code from code from 'org.apache.commons.csv.CSVParserTest::testGetLineNumberWithCRLF'

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.

Points: **0 / 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.

Recorded Answer:

Docker commands are not allowed on the system. The agent is told upfront that this is not possible, but it does it anyways. At first, it correctly uses write_to_file instead, but when the docker container fails to start for an unspecified reason, the agent tries to "escape" into a new shell that does not have this restriction or issues with the docker command. That is prohibited as well. ...

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.

Points: **1 / 1**

User Study Participant Report Card

Participant Key: 536e
Participant Group: A

Overview of Performance

Task 1: 6 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **22 / 84** (Scaled*) Correct: **26%** In Top **0,75** Percentile
Task 2: **0 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

income tracking feature as detailed in the PR description for the finance tracker project

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.

Points: **2 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

Yes, the task was completed successfully.

Expected Answer:

No, the task was not completed.

Points: **0 / 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.

Recorded Answer:

execute bash commands, tool call, file reading, update function

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

tool call

Expected Answer:

Update file

Points: **0 / 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.

Recorded Answer:

IndentationError, indicating a syntax error

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

Yes, he wanted to create a IncomeTracker class with different descriptions

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.

Points: **0 / 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.

Recorded Answer:

replace a string in the File, altering the import statement for FinancialReport and adding IncomeTracker

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **1 / 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.

Recorded Answer:

report.py

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

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'*

Points: **0 / 2**

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

List each one.

Recorded Answer:

Expected Answer:

pandas, matplotlib

Points: **0 / 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.

Recorded Answer:

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
```

Points: **0 / 1**

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

Provide the name of the file.

Recorded Answer:

Expected Answer:

cli.py

Points: **0 / 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.

Recorded Answer:

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'

Points: **0 / 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?

Recorded Answer:

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.

Points: **0 / 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.

Recorded Answer:

The fix description from the agent isnt good enough for the LLM.

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.

Points: **0 / 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.

Recorded Answer:

The command line could not handle the agents input. It repeats it self.

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.

Points: **0 / 1**

User Study Participant Report Card

Participant Key: 5694
Participant Group: B

Overview of Performance

Task 1: 9 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **36 / 84** (Scaled*) Correct: **43%** In Top **0,67** Percentile
Task 2: **0 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

> *Enhance the `finance_tracker` project by adding income tracking functionality to complement the existing expense, budget, user, and reporting features. The feature allows users to record income sources, track income by category and date, calculate net balance, and integrate with reporting and CLI.*

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.*

Points: **3 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

Observations, replacements, package installation

Expected Answer:

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

Points: **1 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

Observations

Expected Answer:

Update file

Points: **0 / 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.

Recorded Answer:

FunctionCallingFormatError

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **0 / 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".*

Recorded Answer:

The Agent requeries the model 3 times in a row after a FunctionCallingFormatError

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.

Points: **0 / 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.

Recorded Answer:

- `add_income <amount> <category> <description> [tags] [recurring] [period]`
- `generate_income_report [start_date] [end_date]`

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **0 / 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.

Recorded Answer:

cli.py, README.md

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

ModuleNotFoundError> No module named 'pandas'

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'*

Points: **1 / 2**

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

List each one.

Recorded Answer:

matplotlib, pandas

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded Answer:

IndentationError

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

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'

Points: **0 / 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?

Recorded Answer:

The Agent compares two different versions of the cli.py file

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.

Points: **0 / 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.

Recorded Answer:

it proposes the same fix over and over again.

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.

Points: **0 / 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.

Recorded Answer:

Problem: it seems that the files which should be read have invalid syntax. The Agent is does not seem to be able to fix it...
Potential Bug: The Bug seems to be related with java, maybe the installation failed, or java is not yet a environment variable.

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.

Points: **0 / 1**

User Study Participant Report Card

Participant Key: bdb6
Participant Group: B

Overview of Performance

Task 1: 14 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **63 / 84** (Scaled*) Correct: **75%** In Top **0,25** Percentile
Task 2: **0 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

> *It's a helpful assistant that can interact with a computer to solve tasks.*

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.*

Points: **0 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

ls, cat, str_replace_editor, python3

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

cat

Expected Answer:

Update file

Points: **0 / 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.

Recorded Answer:

IndentationError: expected an indented block after function definition on line 40

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

The agent calls ls /finance_tracker/finance_tracker three times in succession, resulting in WARNING

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.*

Points: **2 / 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.

Recorded Answer:

str_replace_editor str_replace /finance_tracker/finance_tracker/cli.py --old_str 'from finance_tracker.reports import FinancialReport' --new_str 'from finance_tracker.reports import FinancialReport

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **1 / 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.

Recorded Answer:

/finance_tracker/finance_tracker/cli.py

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

```
OBSERVATION:
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'

pip install pandas
```

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'

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas, matplotlib,

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded Answer:

OBSERVATION: 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

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*

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

/finance_tracker/finance_tracker/cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded 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):
        """Generate an income report: generate_income_report [start_date] [end_date]"""
        if not self.current_user:
            print("Please login first")
            return
        args = arg.split()
        start_date = args[0] if len(args) > 0 else None
        end_date = args[1] if len(args) > 1 else None
        try:
            report_data = self.income_tracker.generate_income_summary(start_date, end_date)
            print(json.dumps(report_data, indent=2))
        except Exception as e:
            print(f"Error: {e}")
'
```

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'

Points: **1 / 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?

Recorded Answer:

They are the exact same change

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.

Points: **1 / 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.

Recorded Answer:

I do not see an fatal bug but continued warning of repetition detected! with repeated write_fix seems concerning. The core should give more drastic guidance or the warning is not effective.

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.

Points: **0 / 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.

Recorded Answer:

I don't see any obvious problem with agent's logic.

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.

Points: **0 / 1**

User Study Participant Report Card

Participant Key: bdf7
Participant Group: A

Overview of Performance

Task 1: 4 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **18 / 84** (Scaled*) Correct: **21%** In Top **0,83** Percentile
Task 2: **1 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

> *The agent's goal is to implement an finance tracker application by creating and testing the required functionalities.*

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.*

Points: **0 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

Read content, Implement code, Test code, and fix code

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

Fix code

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

The agent has trouble implementing a correct version of the income tracker

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **0 / 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".*

Recorded Answer:

Create the `IncomeTracker` class in `income.py` and update related files in the `finance_tracker` directory to incorporate income tracking features as outlined in the PR description.

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.

Points: **0 / 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.

Recorded Answer:

The file /finance_tracker/finance_tracker/cli.py has been edited. Here's the result of running `cat -n` on a snippet of /finance_tracker/finance_tracker/cli.py:
The file /finance_tracker/finance_tracker/reports.py has been edited. Here's the result of running `cat -n` on a snippet of /finance_tracker/finance_tracker/reports.py:

```
 1 .....import cmd
32 .....  "period": f"{start_date or 'all'} to {end_date or 'all'}",
 2 ..... import argparse
33 .....  "category_totals": categories,
 3 .....from finance_tracker.expenses import ExpenseTracker
34 .....  "total": sum(categories.values())
 4 .....from finance_tracker.budgets import BudgetManager
35 ..... }
 5 .....from finance_tracker.users import UserManager
36 .....
 6 .....from finance_tracker.reports import FinancialReport
37 ... def generate_income_summary(self, start_date: str = None, end_date: str = None) -> Dict:
 7 ..... from finance_tracker.income import IncomeTracker
38 ..... """Generate a summary of income by category."""
```

```
 8 ..... from datetime import datetime
39 ..... incomes = self.income_tracker.incomes
 9 .....
40 ..... if start_date and end_date:
10 ..... class FinanceTrackerCLI(cmd.Cmd):
41 ..... incomes = [inc for inc in incomes if start_date <= inc.date <= end_date]
42 ..... categories = {}
43 ..... for inc in incomes:
44 ..... cat = inc.category
45 ..... categories[cat] = categories.get(cat, 0.0) + inc.amount
46 ..... return {
47 ..... "user_id": self.user_id,
48 ..... "period": f"{start_date or 'all'} to {end_date or 'all'}",
49 ..... "category_totals": categories,
50 ..... "total": sum(categories.values())
51 ..... }
52 .....
53 ..... def generate_budget_comparison(self, period: str = "monthly") -> List[Dict]:
54 ..... """Compare spending against budgets."""
55 ..... budgets = self.budget_manager.get_all_budgets()
56 ..... comparison = []
Review the changes and make sure they are as expected. Edit the file again if necessary.
Review the changes and make sure they are as expected. Edit the file again if necessary.
```

Expected Answer:

```
7 + from finance_tracker.income import IncomeTracker
```

Points: **0 / 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.

Recorded Answer:

cli.py

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

IndentationError

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'*

Points: **0 / 2**

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

List each one.

Recorded Answer:

matplotlib

Expected Answer:

pandas, matplotlib

Points: **0 / 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.

Recorded Answer:

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
```

Points: **0 / 1**

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

Provide the name of the file.

Recorded Answer:

Expected Answer:

cli.py

Points: **0 / 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.

Recorded Answer:

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'

Points: **0 / 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?

Recorded Answer:

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.

Points: **0 / 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.

Recorded Answer:

The agent calls the write_fix tool to fix the problem but nothing is changed

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.

Points: **1 / 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.

Recorded Answer:

The agent executes commands that are not available or allowed

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.

Points: **0 / 1**

User Study Participant Report Card

Participant Key: 18e7
Participant Group: A

Overview of Performance

Task 1: 15 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **58 / 84** (Scaled*) Correct: **69%** In Top **0,33** Percentile
Task 2:**#N/A / 1**
Task 3:**#N/A / 1**

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."

Recorded Answer:

The agent goal is to implement new feature or new function-ality- that was described in the PR specifically the income tracking feature for a finance tracker project.

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.

Points: **2 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

Yes, the task was completed successfully.

Expected Answer:

No, the task was not completed.

Points: **0 / 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.

Recorded Answer:

File editing, File listing, Executing commands such as execute script and install package

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

File editing

Expected Answer:

Update file

Points: **1 / 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.

Recorded Answer:

Correctly editing the file? as there are many indentation errors…

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

Yes there was some repetition, e.g, repeated listing of files at the start

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.

Points: **2 / 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.

Recorded Answer:

```
1
import cmd
2
import argparse
3
from finance_tracker.expenses import ExpenseTracker
4
from finance_tracker.budgets import BudgetManager
5
from finance_tracker.users import UserManager
6
from finance_tracker.reports import FinancialReport
7
from finance_tracker.income import IncomeTracker
8
from datetime import datetime
9
```

```
10
class FinanceTrackerCLI(cmd.Cmd):
11
    """Command-line interface for the finance tracker."""
12
    prompt = "FinanceTracker> "
13
    intro = "Welcome to the Personal Finance Tracker. Type 'help' for commands."
14

15
    def __init__(self):
16
        super().__init__()
17
        self.user_manager = UserManager()
18
        self.current_user = None
19
        self.expense_tracker = None
20
        self.budget_manager = None
21
        self.report_generator = None
22

23
    def do_login(self, arg):
24
        """Login to the system: login <username> <password>"""
25
        args = arg.split()
26
        if len(args) != 2:
27
            print("Usage: login <username> <password>")
28
            return
29
        username, password = args
30
        if self.user_manager.authenticate_user(username, password):
31
            self.current_user = username
32
            self.expense_tracker = ExpenseTracker(username)
33
            self.budget_manager = BudgetManager(username)
34
            self.report_generator = FinancialReport(username, self.expense_tracker,
self.budget_manager)
```

```
35
            print(f"Logged in as {username}")
36
        else:
37
            print("Invalid username or password")
38

39
    def do_register(self, arg):
40
        """Register a new user: register <username> <password> <email>"""
41
        args = arg.split()
42
        if len(args) != 3:
43
            print("Usage: register <username> <password> <email>")
44
            return
45
        try:
46
            self.user_manager.register_user(*args)
47
            print(f"User {args[0]} registered successfully")
48
        except ValueError as e:
49
            print(f"Error: {e}")
50

51
    def do_add_expense(self, arg):
52
        """Add an expense: add_expense <amount> <category> <description> [tags] [recurring]
[period]"""
53
        if not self.current_user:
54
            print("Please login first")
55
            return
56
        args = arg.split()
57
        if len(args) < 3:
58
            print("Usage: add_expense <amount> <category> <description> [tags] [recurring] [period]")
59
            return
```

```
60
        amount, category, description = args[:3]
61
        tags = args[3].split(",") if len(args) > 3 else []
62
        is_recurring = args[4].lower() == "true" if len(args) > 4 else False
63
        period = args[5] if len(args) > 5 else None
64
        try:
65
            expense_id = self.expense_tracker.add_expense(float(amount), category, description,
66
                                                          tags=tags, is_recurring=is_recurring,
67
                                                          recurrence_period=period)
68
            print(f"Expense added with ID: {expense_id}")
69
        except ValueError as e:
70
            print(f"Error: {e}")
71

72
    def do_set_budget(self, arg):
73
        """Set a budget: set_budget <category> <amount> [period] [alert_threshold]"""
74
        if not self.current_user:
75
            print("Please login first")
76
            return
77
        args = arg.split()
78
        if len(args) < 2:
79
            print("Usage: set_budget <category> <amount> [period] [alert_threshold]")
80
            return
81
        category, amount = args[:2]
82
        period = args[2] if len(args) > 2 else "monthly"
83
        alert_threshold = float(args[3]) if len(args) > 3 else 0.8
84
        try:
85
```

```
            self.budget_manager.set_budget(category, float(amount), period, alert_threshold)
86
            print(f"Budget set for {category} ({period})")
87
        except ValueError as e:
88
            print(f"Error: {e}")
89

90
    def do_generate_report(self, arg):
91
        """Generate a report: generate_report <type> [start_date] [end_date]"""
92
        if not self.current_user:
93
            print("Please login first")
94
            return
95
        args = arg.split()
96
        if len(args) < 1:
97
            print("Usage: generate_report <type> [start_date] [end_date]")
98
            return
99
        report_type = args[0]
100
        start_date = args[1] if len(args) > 1 else None
101
        end_date = args[2] if len(args) > 2 else None
102
        try:
103
            report_file = self.report_generator.generate_report_pdf(report_type, start_date, end_date)
104
            print(f"Report generated: {report_file}")
105
        except Exception as e:
106
            print(f"Error: {e}")
107

108
    def do_exit(self, arg):
109
        """Exit the CLI."""
110
        print("Goodbye!")
```

```
111
        return True
112

113
    def preloop(self):
114
        """Initialize CLI state."""
115
        print("Personal Finance Tracker CLI. Type 'register' or 'login' to begin.")
```

Expected Answer:

```
7 + from finance_tracker.income import IncomeTracker
```

Points: **0 / 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.

Recorded Answer:

income.py, expenses.py, reports.py, cli.py, README,

Expected Answer:

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

Points: **1 / 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.

Recorded Answer:

Missing module pandas, the agent fixes it by installing the package

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'*

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas, matplotlib

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded Answer:

Error indicating indentation problems

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

There is a replace old income report and old method generation before running tests but not sure if they count as same change twice. Otherwise No.

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'

Points: **0 / 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?

Recorded Answer:

From the diff comparison they look similar

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.

Points: **1 / 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.

Recorded Answer:

#N/A

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.

Points: **#N/A...../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.

Recorded Answer:

#N/A

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.

Points: **#N/A...../1**

User Study Participant Report Card

Participant Key: e43c
Participant Group: B

Overview of Performance

Task 1: 13 / 21 (Unscaled) **: Points are scaled to ensure equal weight of each question.*
Task 1: **56 / 84** (Scaled*) Correct: **67%** In Top **0,42** Percentile
Task 2: **1 / 1**
Task 3: **0 / 1**

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."

Recorded Answer:

> *The agent goal is to implement necessary changes following the requirements in a pull requests. It will need to first analyses this task step by step and provide candidate solutions. Then, they need to run tools to validate the correctness of candidates solutions. Finally, they will submit the solutions if they pass the validation.*

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.*

Points: **1 / 3**

Question 2: Did the agent complete its task?

Recorded Answer:

No, the task was not completed.

Expected Answer:

No, the task was not completed.

Points: **1 / 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.

Recorded Answer:

run tools, generate solutions, fix solutions, submit solutions

Expected Answer:

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

Points: **2 / 2**

Question 4: What is the most frequently performed action?

Recorded Answer:

run tools

Expected Answer:

Update file

Points: **0 / 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.

Recorded Answer:

The agent has trouble on generating compilable code. Some errors include "ModuleNotFoundError", indicating they fail to setup environment correctly, and "IndentationError", indicating they fails to generate syntactically correct code.

Expected Answer:

The test script keeps failing with an IndentationError.

Points: **1 / 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".*

Recorded Answer:

Yes, the agent run programs, found some face indentation errors and then fixing these errors 4 times in a row before stopping.

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.

Points: **1 / 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.

Recorded Answer:

+from finance_tracker.income import IncomeTracker

Expected Answer:

7 + from finance_tracker.income import IncomeTracker

Points: **1 / 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.

Recorded Answer:

/finance_tracker/finance_tracker/cli.py

Expected Answer:

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

Points: **0 / 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.

Recorded Answer:

The agent get a feedback that a dependency, i.e., pandas, was missing. "ModuleNotFoundError: No module named 'pandas'" Based on this feedback the agent installed this module in next action.

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'
```

Points: **2 / 2**

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

List each one.

Recorded Answer:

pandas, matplotlib

Expected Answer:

pandas, matplotlib

Points: **1 / 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.

Recorded 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

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
```

Points: **1 / 1**

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

Provide the name of the file.

Recorded Answer:

/finance_tracker/finance_tracker/cli.py

Expected Answer:

cli.py

Points: **1 / 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.

Recorded Answer:

Yes "Found submission: diff --git a/finance_tracker/cli.py b/finance_tracker/cli.py"

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'

Points: **0 / 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?

Recorded Answer:

They tried to fix same error and return same fix for all turns.

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.

Points: **1 / 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.

Recorded Answer:

The write_fix() tool should generate the patch and apply this patch to current repository by modifying the read() method. However, I guess it only generate the fix and do not apply into current repository. That is reason why the test results are always failures.

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.

Points: **1 / 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.

Recorded Answer:

I think the error lies on tool design to execute command. This based on the observation that this error message "SYSTEM: Unable to execute command" repeat 39 times in the logs. Without executing the command, the agent cannot define the next steps, leading to failures.

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.

Points: **0 / 1**