

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:

*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:

```
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: 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\"{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:

```
1 import cmd
2
3 import argparse
4
5 from finance_tracker.expenses import ExpenseTracker
6
7 from finance_tracker.budgets import BudgetManager
8
9 from finance_tracker.users import UserManager
10
11 from finance_tracker.reports import FinancialReport
12
13 from finance_tracker.income import IncomeTracker
14
15 from datetime import datetime
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
315
316
317
317
318
319
319
320
321
322
323
323
324
325
325
326
327
327
328
329
329
330
331
331
332
333
333
334
335
335
336
337
337
338
339
339
340
341
341
342
343
343
344
345
345
346
347
347
348
349
349
350
351
351
352
353
353
354
355
355
356
357
357
358
359
359
360
361
361
362
363
363
364
365
365
366
367
367
368
369
369
370
371
371
372
373
373
374
375
375
376
377
377
378
379
379
380
381
381
382
383
383
384
385
385
386
387
387
388
389
389
390
391
391
392
393
393
394
395
395
396
397
397
398
399
399
400
401
401
402
403
403
404
405
405
406
407
407
408
409
409
410
411
411
412
413
413
414
415
415
416
417
417
418
419
419
420
421
421
422
423
423
424
425
425
426
427
427
428
429
429
430
431
431
432
433
433
434
435
435
436
437
437
438
439
439
440
441
441
442
443
443
444
445
445
446
447
447
448
449
449
450
451
451
452
453
453
454
455
455
456
457
457
458
459
459
460
461
461
462
463
463
464
465
465
466
467
467
468
469
469
470
471
471
472
473
473
474
475
475
476
477
477
478
479
479
480
481
481
482
483
483
484
485
485
486
487
487
488
489
489
490
491
491
492
493
493
494
495
495
496
497
497
498
499
499
500
501
501
502
503
503
504
505
505
506
507
507
508
509
509
510
511
511
512
513
513
514
515
515
516
517
517
518
519
519
520
521
521
522
523
523
524
525
525
526
527
527
528
529
529
530
531
531
532
533
533
534
535
535
536
537
537
538
539
539
540
541
541
542
543
543
544
545
545
546
547
547
548
549
549
550
551
551
552
553
553
554
555
555
556
557
557
558
559
559
560
561
561
562
563
563
564
565
565
566
567
567
568
569
569
570
571
571
572
573
573
574
575
575
576
577
577
578
579
579
580
581
581
582
583
583
584
585
585
586
587
587
588
589
589
590
591
591
592
593
593
594
595
595
596
597
597
598
599
599
600
601
601
602
603
603
604
605
605
606
607
607
608
609
609
610
611
611
612
613
613
614
615
615
616
617
617
618
619
619
620
621
621
622
623
623
624
625
625
626
627
627
628
629
629
630
631
631
632
633
633
634
635
635
636
637
637
638
639
639
640
641
641
642
643
643
644
645
645
646
647
647
648
649
649
650
651
651
652
653
653
654
655
655
656
657
657
658
659
659
660
661
661
662
663
663
664
665
665
666
667
667
668
669
669
670
671
671
672
673
673
674
675
675
676
677
677
678
679
679
680
681
681
682
683
683
684
685
685
686
687
687
688
689
689
690
691
691
692
693
693
694
695
695
696
697
697
698
699
699
700
701
701
702
703
703
704
705
705
706
707
707
708
709
709
710
711
711
712
713
713
714
715
715
716
717
717
718
719
719
720
721
721
722
723
723
724
725
725
726
727
727
728
729
729
730
731
731
732
733
733
734
735
735
736
737
737
738
739
739
740
741
741
742
743
743
744
745
745
746
747
747
748
749
749
750
751
751
752
753
753
754
755
755
756
757
757
758
759
759
760
761
761
762
763
763
764
765
765
766
767
767
768
769
769
770
771
771
772
773
773
774
775
775
776
777
777
778
779
779
780
781
781
782
783
783
784
785
785
786
787
787
788
789
789
790
791
791
792
793
793
794
795
795
796
797
797
798
799
799
800
801
801
802
803
803
804
805
805
806
807
807
808
809
809
810
811
811
812
813
813
814
815
815
816
817
817
818
819
819
820
821
821
822
823
823
824
825
825
826
827
827
828
829
829
830
831
831
832
833
833
834
835
835
836
837
837
838
839
839
840
841
841
842
843
843
844
845
845
846
847
847
848
849
849
850
851
851
852
853
853
854
855
855
856
857
857
858
859
859
860
861
861
862
863
863
864
865
865
866
867
867
868
869
869
870
871
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
883
883
884
885
885
886
887
887
888
889
889
890
891
891
892
893
893
894
895
895
896
897
897
898
899
899
900
901
901
902
903
903
904
905
905
906
907
907
908
909
909
910
911
911
912
913
913
914
915
915
916
917
917
918
919
919
920
921
921
922
923
923
924
925
925
926
927
927
928
929
929
930
931
931
932
933
933
934
935
935
936
937
937
938
939
939
940
941
941
942
943
943
944
945
945
946
947
947
948
949
949
950
951
951
952
953
953
954
955
955
956
957
957
958
959
959
960
961
961
962
963
963
964
965
965
966
967
967
968
969
969
970
971
971
972
973
973
974
975
975
976
977
977
978
979
979
980
981
981
982
983
983
984
985
985
986
987
987
988
989
989
990
991
991
992
993
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
```

```
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
37     else:
38
39
40     def do_register(self, arg):
41         """Register a new user: register <username> <password> <email>"""
42
43         args = arg.split()
44
45         if len(args) != 3:
46
47             print("Usage: register <username> <password> <email>")
48
49             return
50
51
52         try:
53             self.user_manager.register_user(*args)
54
55             print(f"User {args[0]} registered successfully")
56
57         except ValueError as e:
58
59             print(f"Error: {e}")
60
61
62     def do_add_expense(self, arg):
63
64         """Add an expense: add_expense <amount> <category> <description> [tags] [recurring]
65         [period]"""
66
67         if not self.current_user:
68
69             print("Please login first")
70
71             return
72
73         args = arg.split()
74
75         if len(args) < 3:
76
77             print("Usage: add_expense <amount> <category> <description> [tags] [recurring] [period]")
78
79             return
80
81
```

```
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
88     except ValueError as e:
89
90     print(f"Error: {e}")
91
92
93     def do_generate_report(self, arg):
94
95         """Generate a report: generate_report <type> [start_date] [end_date]"""
96
97         if not self.current_user:
98
99             print("Please login first")
100
101         return
102
103         args = arg.split()
104
105         if len(args) < 1:
106
107             print("Usage: generate_report <type> [start_date] [end_date]")
108
109             return
110
111         report_type = args[0]
112
113         start_date = args[1] if len(args) > 1 else None
114
115         end_date = args[2] if len(args) > 2 else None
116
117         try:
118
119             report_file = self.report_generator.generate_report_pdf(report_type, start_date, end_date)
120
121             print(f"Report generated: {report_file}")
122
123             except Exception as e:
124
125                 print(f"Error: {e}")
126
127
128
129         def do_exit(self, arg):
130
131             """Exit the CLI."""
132
133             print("Goodbye!")
134
135
```

```
    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:

```
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: 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\"\"\"{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\"\"\"{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:
.....obj
```

```

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()\nstart_date = args[0] if len(args) > 0 else None\n    end_date = args[1] if len(args) > 1 else None\ntry:\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\"\"\"{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()\nstart_date = args[0] if len(args) > 0 else None\n    end_date = args[1] if len(args) > 1 else None\ntry:\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\"\"\"{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()\nstart_date = args[0] if len(args) > 0 else None\n      end_date = args[1] if len(args) > 1 else None\ntry:\n      report_data = self.income_tracker.generate_income_summary(start_date,
end_date)\n      print(json.dumps(report_data, indent=2))\nexcept 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\"{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 isn't 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:

```
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: 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):
11 .....         incomes = [inc for inc in incomes if start_date <= inc.date <= end_date]
12 ..... 
13 .....         categories = {}
14 ..... 
15 .....         for inc in incomes:
16 .....             cat = inc.category
17 ..... 
18 .....             categories[cat] = categories.get(cat, 0.0) + inc.amount
19 ..... 
20 .....         return {
21 .....             "user_id": self.user_id,
22 .....             "period": f'{start_date or "all"} to {end_date or "all"}',
23 .....             "category_totals": categories,
24 .....             "total": sum(categories.values())
25 .....         }
26 ..... 
27 .....     }
28 ..... 
29 .....     def generate_budget_comparison(self, period: str = "monthly") -> List[Dict]:
30 .....         """
31 .....             Compare spending against budgets.
32 .....         """
33 .....         budgets = self.budget_manager.get_all_budgets()
34 ..... 
35 .....         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
37     else:
38
39     def do_register(self, arg):
40
41         """Register a new user: register <username> <password> <email>"""
42
43         args = arg.split()
44
45         if len(args) != 3:
46
47             print("Usage: register <username> <password> <email>")
48
49             return
50
51
52         try:
53
54             self.user_manager.register_user(*args)
55
56             print(f"User {args[0]} registered successfully")
57
58         except ValueError as e:
59
60             print(f"Error: {e}")
61
62
63     def do_add_expense(self, arg):
64
65         """Add an expense: add_expense <amount> <category> <description> [tags] [recurring]
66         [period]"""
67
68         if not self.current_user:
69
70             print("Please login first")
71
72             return
73
74         args = arg.split()
75
76         if len(args) < 3:
77
78             print("Usage: add_expense <amount> <category> <description> [tags] [recurring] [period]")
79
80             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:
```

```
    self.budget_manager.set_budget(category, float(amount), period, alert_threshold)
86
87     print(f"Budget set for {category} ({period})")
88
89     except ValueError as e:
90
91         print(f"Error: {e}")
92
93
94
95     def do_generate_report(self, arg):
96
97         """Generate a report: generate_report <type> [start_date] [end_date]"""
98
99         if not self.current_user:
100
101             print("Please login first")
102
103             return
104
105         args = arg.split()
106
107         if len(args) < 1:
108
109             print("Usage: generate_report <type> [start_date] [end_date]")
110
111             return
112
113         report_type = args[0]
114
115         start_date = args[1] if len(args) > 1 else None
116
117         end_date = args[2] if len(args) > 2 else None
118
119         try:
120
121             report_file = self.report_generator.generate_report_pdf(report_type, start_date, end_date)
122
123             print(f"Report generated: {report_file}")
124
125             except Exception as e:
126
127                 print(f"Error: {e}")
128
129
130
131     def do_exit(self, arg):
132
133         """Exit the CLI."""
134
135         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\"{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

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\"{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
