ДОДАТОК А

Вихідні дані з тестових середовищ

Лістинг А.1 – Приклад тестового сценарія у середовищі WebArena


```
{
  "sites": [
    "shopping_admin"
  ],
  "start_url": "localhost:7780/women/tops-women/jackets-
women.html",
  "geolocation": null,
  "intent_template": "What is the top-{{n}} best-selling
product in {{year}}",
  "instantiation_dict": {
    "n": 1,
    "year": 2022
  },
  "intent": "What is the top-1 best-selling product in 2022",
  "require_reset": false,
  "eval": {
    "eval_types": [
      "string_match"
    ],
    "reference_answers": {
      "exact_match": "Quest Lumaflex\u2122 Band"
    },
    "reference_url": "",
    "program_html": [],
    "string_note": "",
    "reference_answer_raw_annotation": "Quest Lumaflex\u2122
Band"
  },
}
```

Лістинг А.2 – Результати тесту у середовищі WebArena

```
New Task: How many reviews does Josie Yoga Jacket have? And what is their summary.
{
  "result": [
    "Navigate to the Women's section.",
    "Search for 'Josie Yoga Jacket' in the search bar.",
    "Select the Josie Yoga Jacket product from the search results.",
    "Locate the reviews section on the product page.",
    "Retrieve the number of reviews and their summary."
  ]
}
{
  "task_complete": "False",
  "result": "",
  "image_observation": "Homepage with navigation menu visible.",
  "action_effect": "No action taken yet.",
  "hint_to_actor": "Click on the 'Women' section in the navigation menu."
}
{
  "reasoning": "The reviewer hint suggests clicking on the 'Women' section, which is identified by ID 177 in the annotation.",
  "action": "click [177]"
}
Trajectory: [{'iteration': 1, 'image_observation': 'Homepage with navigation menu visible.', 'new_action': 'click [177]', 'new_action_reason': "The reviewer hint suggests clicking on the 'Women' section, which is identified by ID 177 in the annotation."}]
{
  "task_complete": "False",
  "result": "",
```

"image_observation": "Women's section page is visible with categories listed.",
"action_effect": "Navigated to the Women's section.",
"hint_to_actor": "Use the search bar to search for 'Josie Yoga Jacket'."
}
{
"reasoning": "The search bar is available with ID 965. To search for 'Josie Yoga Jacket', typing in the search bar is necessary.",
"action": "type [965] [Josie Yoga Jacket] [1]"
}
Trajectory: [{'iteration': 1, 'image_observation': 'Homepage with navigation menu visible.', 'new_action': 'click [177]', 'new_action_reason': "The reviewer hint suggests clicking on the 'Women' section, which is identified by ID 177 in the annotation."}, {'iteration': 2, 'image_observation': "Women's section page is visible with categories listed.", 'new_action': 'type [965] [Josie Yoga Jacket] [1]', 'new_action_reason': "The search bar is available with ID 965. To search for 'Josie Yoga Jacket', typing in the search bar is necessary."}]
{
"task_complete": "False",
"result": "",
"image_observation": "Search results for 'Josie Yoga Jacket' are visible with product listings.",
"action_effect": "Search results displayed, showing multiple products including Josie Yoga Jacket.",
"hint_to_actor": "Click on the 'Josie Yoga Jacket' product to view its details."
}
{
"reasoning": "The reviewer hint suggests clicking on the 'Josie Yoga Jacket' to view details. The annotation shows the link with ID [2288].",
"action": "click [2288]"

}
Trajectory: [{'iteration': 1, 'image_observation': 'Homepage with navigation menu visible.', 'new_action': 'click [177]', 'new_action_reason': "The reviewer hint suggests clicking on the 'Women' section, which is identified by ID 177 in the annotation."}, {'iteration': 2, 'image_observation': "Women's section page is visible with categories listed.", 'new_action': 'type [965] [Josie Yoga Jacket] [1]', 'new_action_reason': "The search bar is available with ID 965. To search for 'Josie Yoga Jacket', typing in the search bar is necessary."}, {'iteration': 3, 'image_observation': "Search results for 'Josie Yoga Jacket' are visible with product listings.", 'new_action': 'click [2288]', 'new_action_reason': "The reviewer hint suggests clicking on the 'Josie Yoga Jacket' to view details. The annotation shows the link with ID [2288]."}]
{
  "task_complete": "False",
  "result": "",
  "image_observation": "Product page for Josie Yoga Jacket is visible with 4 reviews link.",
  "action_effect": "Navigated to the product page, reviews link is visible.",
  "hint_to_actor": "Click on the '4 Reviews' link to view the reviews section."
}
{
  "reasoning": "The '4 Reviews' link is visible and has the ID 3519, which matches the reviewer's hint to click it.",
  "action": "click [3519]"
}
Trajectory: [{'iteration': 1, 'image_observation': 'Homepage with navigation menu visible.', 'new_action': 'click [177]', 'new_action_reason': "The reviewer hint suggests clicking on the 'Women' section, which is identified by ID 177 in the annotation."}, {'iteration': 2, 'image_observation': "Women's section page is visible with categories listed.", 'new_action':

'type [965] [Josie Yoga Jacket] [1]', 'new_action_reason': "The search bar is available with ID 965. To search for 'Josie Yoga Jacket', typing in the search bar is necessary."}, {'iteration': 3, 'image_observation': "Search results for 'Josie Yoga Jacket' are visible with product listings.", 'new_action': 'click [2288]', 'new_action_reason': "The reviewer hint suggests clicking on the 'Josie Yoga Jacket' to view details. The annotation shows the link with ID [2288]."}, {'iteration': 4, 'image_observation': 'Product page for Josie Yoga Jacket is visible with 4 reviews link.', 'new_action': 'click [3519]', 'new_action_reason': "The '4 Reviews' link is visible and has the ID 3519, which matches the reviewer's hint to click it."}]
{
  "task_complete": "True",
  "result": "The Josie Yoga Jacket has 4 reviews. Summary: 1) 'Fitted, awesome' - 5 stars, 2) 'Shrinks a lot' - 2 stars, 3) 'Shrunk right away!' - 2 stars, 4) 'my new fave zip up' - not fully visible.",
  "image_observation": "Customer reviews section is visible with 4 reviews listed.",
  "action_effect": "The reviews section was accessed, displaying the reviews.",
  "hint_to_actor": ""
}
Result: The Josie Yoga Jacket has 4 reviews. Summary: 1) 'Fitted, awesome' - 5 stars, 2) 'Shrinks a lot' - 2 stars, 3) 'Shrunk right away!' - 2 stars, 4) 'my new fave zip up' - not fully visible.
Knowledge Manager Response: {
  "save_knowledge": "False",
  "knowledge_title": "",
  "knowledge_type": "OBSERVATIONS",
  "body": ""
}

Process finished with exit code 0

Лістинг А.3 – Результати тесту у середовищі MiniWobPlusPlus

```
============================================================
==========================
MiniWoB++ Startup
============================================================
==========================
Registering MiniWoB++ environments...
Batch episode: 2/10
Global Planner enabled: False
Reflection Agent enabled: False
Knowledge Review enabled: False
Using experiment: miniwob/drag-shapes-v1
Serving MiniWoB++ HTML from: http://127.0.0.1:57156/miniwob/
HTTP server request logs are hidden. Set MINIWOB_HTTP_LOGS=1 to
show them.
Creating MiniWoB++ environment with render_mode='human'...
MiniWoB episode time limit set to 3600000 ms.
MiniWoB click visualization overlays are hidden. Set
MINIWOB_CLICK_VISUALS=1 to show them.

============================================================
==========================
Episode
============================================================
==========================
Task: Drag all circles into the black box.

 Browser UI State ------------------------------------------
------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

 Global Planner --------------------------------------------
------------------------
```

```
Skipped. Baseline local-action mode uses the raw task as the
only plan.

------------------------------------------------------------
--------------------------
Step 1
------------------------------------------------------------
--------------------------
Reflecting and asking Action Agent...

 Browser UI State Before Action ------------------------------
--------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

 Local Action Context ----------------------------------------
--------------------------
Reflection Agent skipped. Action Agent will use raw task, DOM,
browser state, and previous effect.

------------------------------------------------------------
--------------------------
Action Agent Response
------------------------------------------------------------
--------------------------
{
  "reasoning": "Need to start dragging the only visible circle
(ref=7) towards the black box area (ref=9). Drag action not
available by instructions, so first click the circle to select
it.",
  "action": "CLICK_ELEMENT",
  "ref": 7,
  "text": null,
  "coords": null,
  "field": null,
  "done": false,
```

```
  "final_answer": ""
}
```

Executable Action ----------

action: CLICK_ELEMENT
reasoning: Need to start dragging the only visible circle (ref=7) towards the black box area (ref=9). Drag action not available by instructions, so first click the circle to select it.
kwargs: {'ref': 7}

Environment Effect ----------

CONTINUE: the episode is still active.
Environment returned reward=0.0, status=continuing, info={'done': False, 'env_reward': 0, 'raw_reward': 0, 'reason': None, 'elapsed': 5.739115953445435, 'root_dom': [1] body @ (0, 0) classes=[] children=1}.

Browser UI State After Action ----------

url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

Step 2

Reflecting and asking Action Agent...

Browser UI State Before Action ----------

url: http://127.0.0.1:57156/miniwob/drag-shapes.html

```
query: Drag all circles into the black box.

 Local Action Context ----------------------------------------
-------------------------
Reflection Agent skipped. Action Agent will use raw task, DOM,
browser state, and previous effect.

------------------------------------------------------------
-------------------------
Action Agent Response
------------------------------------------------------------
-------------------------
{
  "reasoning": "Drag the circle (ref=7) into the black box
(ref=9) by clicking at coordinates starting from the circle's
position to the black box location.",
  "action": "CLICK_COORDS",
  "ref": null,
  "text": null,
  "coords": [110, 100],
  "field": null,
  "done": false,
  "final_answer": ""
}
------------------------------------------------------------
-------------------------

 Executable Action ----------------------------------------
-------------------------
action: CLICK_COORDS
reasoning: Drag the circle (ref=7) into the black box (ref=9) by
clicking at coordinates starting from the circle's position to
the black box location.
kwargs: {'coords': [110, 100]}
```

```
 Environment Effect ------------------------------------------
-------------------------
CONTINUE: the episode is still active.
Environment returned reward=0.0, status=continuing,
info={'done': False, 'env_reward': 0, 'raw_reward': 0, 'reason':
None, 'elapsed': 13.377855777740479, 'root_dom': [1] body @ (0,
0) classes=[] children=1}.

 Browser UI State After Action -------------------------------
-------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

------------------------------------------------------------
-------------------------
Step 3
------------------------------------------------------------
-------------------------
Reflecting and asking Action Agent...

 Browser UI State Before Action ------------------------------
-------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

 Local Action Context ----------------------------------------
-------------------------
Reflection Agent skipped. Action Agent will use raw task, DOM,
browser state, and previous effect.

------------------------------------------------------------
-------------------------
Action Agent Response
------------------------------------------------------------
-------------------------
{
```

```
  "reasoning": "Click the circle (ref=7) to start dragging it into the black box (ref=9). Only one circle is present to drag into the box.",
  "action": "CLICK_ELEMENT",
  "ref": 7,
  "text": null,
  "coords": null,
  "field": null,
  "done": false,
  "final_answer": ""
}
```

Executable Action ----------

action: CLICK_ELEMENT
reasoning: Click the circle (ref=7) to start dragging it into the black box (ref=9). Only one circle is present to drag into the box.
kwargs: {'ref': 7}

Environment Effect ----------

CONTINUE: the episode is still active.
Environment returned reward=0.0, status=continuing, info={'done': False, 'env_reward': 0, 'raw_reward': 0, 'reason': None, 'elapsed': 18.554938554763794, 'root_dom': [1] body @ (0, 0) classes=[] children=1}.

Browser UI State After Action ----------

url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

```
------------------------------------------------------------
-------------------------
Step 4
------------------------------------------------------------
-------------------------
Reflecting and asking Action Agent...

 Browser UI State Before Action ------------------------------
-------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

 Local Action Context ----------------------------------------
-------------------------
Reflection Agent skipped. Action Agent will use raw task, DOM,
browser state, and previous effect.

------------------------------------------------------------
-------------------------
Action Agent Response
------------------------------------------------------------
-------------------------
{
  "reasoning": "Start dragging the circle (ref=7) by mouse down
at its position to initiate the drag into the black box
(ref=9).",
  "action": "MOUSEDOWN_COORDS",
  "ref": null,
  "text": null,
  "coords": [110, 100],
  "field": null,
  "done": false,
  "final_answer": ""
}
------------------------------------------------------------
-------------------------
```

```
 Executable Action ------------------------------------------
------------------------
action: MOUSEDOWN_COORDS
reasoning: Start dragging the circle (ref=7) by mouse down at
its position to initiate the drag into the black box (ref=9).
kwargs: {'coords': [110, 100]}

 Environment Effect -----------------------------------------
------------------------
CONTINUE: the episode is still active.
Environment returned reward=0.0, status=continuing,
info={'done': False, 'env_reward': 0, 'raw_reward': 0, 'reason':
None, 'elapsed': 23.440483331680298, 'root_dom': [1] body @ (0,
0) classes=[] children=1}.

 Browser UI State After Action ------------------------------
------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

------------------------------------------------------------
------------------------
Step 5
------------------------------------------------------------
------------------------
Reflecting and asking Action Agent...

 Browser UI State Before Action ------------------------------
------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

 Local Action Context ---------------------------------------
------------------------
```

```
Reflection Agent skipped. Action Agent will use raw task, DOM,
browser state, and previous effect.

------------------------------------------------------------
-------------------------
Action Agent Response
------------------------------------------------------------
-------------------------
{"reasoning":"Release the circle (drag) into the black box area
by mouse up at black box center coordinates [55, 95]. This
completes dragging the circle into the black
box.","action":"MOUSEUP_COORDS","ref":null,"text":null,"coords
":[55,95],"field":null,"done":false,"final_answer":""}
------------------------------------------------------------
-------------------------

 Executable Action ---------------------------------------
-------------------------
action: MOUSEUP_COORDS
reasoning: Release the circle (drag) into the black box area by
mouse up at black box center coordinates [55, 95]. This completes
dragging the circle into the black box.
kwargs: {'coords': [55, 95]}

 Environment Effect --------------------------------------
-------------------------
CONTINUE: the episode is still active.
Environment    returned    reward=0.0,    status=continuing,
info={'done': False, 'env_reward': 0, 'raw_reward': 0, 'reason':
None, 'elapsed': 27.903130769729614, 'root_dom': [1] body @ (0,
0) classes=[] children=1}.

 Browser UI State After Action ---------------------------
-------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.
```

```
------------------------------------------------------------
-------------------------
Step 6
------------------------------------------------------------
-------------------------
Reflecting and asking Action Agent...

 Browser UI State Before Action ----------------------------
-------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

 Local Action Context --------------------------------------
-------------------------
Reflection Agent skipped. Action Agent will use raw task, DOM,
browser state, and previous effect.

------------------------------------------------------------
-------------------------
Action Agent Response
------------------------------------------------------------
-------------------------
{
  "reasoning": "Only one circle (ref=7) to drag; already
attempted drag and drop using mouse down and up. Now submit to
complete as circle is in black box.",
  "action": "CLICK_ELEMENT",
  "ref": 10,
  "text": null,
  "coords": null,
  "field": null,
  "done": false,
  "final_answer": ""
}
```

```
------------------------------------------------------------
-------------------------

 Executable Action ------------------------------------------
-------------------------
action: CLICK_ELEMENT
reasoning: Only one circle (ref=7) to drag; already attempted
drag and drop using mouse down and up. Now submit to complete as
circle is in black box.
kwargs: {'ref': 10}

 Environment Effect -----------------------------------------
-------------------------
SUCCESS: MiniWoB accepted the final state.
Environment         returned         reward=0.9908030555555556,
status=terminated,     info={'done':     True,     'env_reward':
0.9908030555555556, 'raw_reward': 1, 'reason': None, 'elapsed':
33.09810733795166}.

 Browser UI State After Action ------------------------------
-------------------------
url: http://127.0.0.1:57156/miniwob/drag-shapes.html
query: Drag all circles into the black box.

============================================================
=========================
Knowledge Review
============================================================
=========================
Skipped. Baseline local-action mode measures Action Agent usage
only.

============================================================
=========================
Final Result
```

```
================================================================
==========================
Success!!!

================================================================
==========================
Episode Metrics
================================================================
==========================
experiment: miniwob/drag-shapes-v1
success: True
final_status: success
trajectory_length: 6
tokens_total: 16365
trajectory_stability: stability_score=0.833, loop_ratio=0.167,
loop_steps=1/6, repeated_action_steps=1,
loop_counts_by_size={'1': 0, '2': 1, '3': 0}
```

ДОДАТОК Б

Текстові промпти агентів

Лістинг Б.1 – Текстовий промпт агента виконання дій

```
LOCAL_PLANNER_PROMPT = f"""
You are a Agent tasked with navigating a web page.
Choose exactly one executable browser action using the user task, screenshot, annotation, and reviewer feedback.

Rules:
- Treat reviewer hint as the main short-term objective unless it contradicts the screenshot or annotation.
- Use only one action.
- For click, hover, and type, element_id must be a numeric ID copied exactly from the annotation, for example click [123].
- Before returning, double-check that every ID used in action appears exactly in Annotation.
- If the needed target has no visible numeric ID in the annotation, use scroll, keyboard, or another valid action instead of inventing an ID.
- If the needed target is not visible, use scroll, go_back, or another action likely to reveal it.
- If the user task is already answered on the page, use stop [answer].
- reasoning must briefly explain why the selected action and ID are valid.

Valid action formats. Do not remove [ or ] characters, they are mandatory.
- click [element_id]
- hover [element_id]
- type [element_id] [<text>] [enter_flag] / [0] or [1]
- press [key_combination]
- scroll [up|down]
- goto [url]
```

```
    - new_tab
    - go_back
    - go_forward
    - tab_focus [page_number]
    - close_tab
    - stop [answer]

    Key combinations:
    - Enter
    - ArrowDown
    - ArrowUp

    Do not wrap output with ```json
    Return only raw text with valid JSON.
    Do not return Markdown.
    Do not include explanations outside JSON.
    Do not wrap the JSON in a code block.
    The first character must be {{ and the last character must
be }}.

    The JSON must have exactly this structure:
    {{
      "reasoning": "",
      "action": ""
    }}

    If the task is complete, still return the same JSON structure
and put the final answer inside the action:
    {{
      "reasoning": "The requested name and price are visible, so
the task can be completed.",
       "action": "stop [The cheapest women's jacket is Example
Jacket at $12.99.]"
    }}

    Reasoning size limit: {REASONING_SIZE} characters
```

```
    User task:
    {user_task}

    Reviewer hint:
    {reasoner_hint}

    Reviewer image observation:
    {image_observation}

    Reviewer previous action effect:
    {action_effect}

    Annotation:
    {website_annotation}
    """
```

Лістинг Б.2 – Текстовий промпт агента спостереження та рефлексії

```
    # Role
    You are a Reviewer Agent for a web navigation system.
    Analyze screenshot and trajectory in order to determine
progress.
    Use precise minimal language.

    # Goal
    Evaluate the current page against the user task and full
global plan.
    Produce the next atomic hint for the Local Planner.

    # Knowledge
    Available knowledge titles:
    {knowledge_titles}

    Loaded knowledge:
    {loaded_knowledge_text}
```

```
    Always decide whether existing knowledge is needed.
    If needed, return only this JSON and no other fields:
    {{
      "knowledge_request": {{
        "knowledge_title":
"title_from_available_knowledge_titles"
      }}
    }}

    # Review Rules
    - Take the global plan as a hint for how to complete the
user task.
    - hint_to_actor must be one atomic next instruction; no vague
feedback or multi-step hints.
    - If no action has been taken yet, create the first atomic
hint for the earliest incomplete global plan item.
    - If the latest action did not help or repeated a loop, give
a specific different next action.
    - Set task_complete=True only when the full user task is
visibly complete; then put the final answer in result.
    - If task_complete=False, result must be "".
    - image_observation must describe the relevant visible state
after the previous action.
    - action_effect must summarize how the latest action
affected progress.
    - hint_to_actor must be based on image_observation and
action_effect.
    - If required page content is not visible, hint one scroll
action.

    # Output
    Return only raw valid JSON.
    Do not return Markdown or code fences.
    Do not include explanations outside JSON.
```

```
    The first character must be {{ and the last character must
be }}.

    Normal JSON structure:
    {{
      "task_complete": "False",
      "result": "",
      "image_observation": "",
      "action_effect": "",
      "hint_to_actor": ""
    }}

    User task:
    {user_task}

    Full global plan:
    {global_plan}

    Trajectory of previous states:
    {last_actions}
    """
```

Лістинг Б.3 – Текстовий промпт менеджера знань

```
    KNOWLEDGE_MANAGER_PROMPT = f"""
    # Role
    You are a Knowledge Manager Agent for a web navigation
system.
    Your only job is to decide whether the finished trajectory
contains reusable knowledge.

    # Existing Knowledge
    Consider existing knowledge to avoid duplicates:
    {existing_knowledge}

    # Save Policy
```

Save new knowledge only when the trajectory reveals reusable behavior that can help future executions.

Save knowledge when there are loops, local planner mistakes, UI quirks, environment limitations, or useful observations.

Do not save obvious one-off task details or duplicate existing knowledge.

Knowledge Types

Use exactly one:

- INFORMATIONAL: general reusable information.
- ERORS: bugs, loops, failed actions, local planner mistakes, or execution errors.
- OBSERVATIONS: useful notes from page behavior or trajectory.

Output

Return only raw valid JSON.

Do not return Markdown or code fences.

Do not include explanations outside JSON.

JSON structure:

```
{{
  "save_knowledge": "False",
  "knowledge_title": "",
  "knowledge_type": "OBSERVATIONS",
  "body": ""
}}
```

Execution Summary

User task:

{user_task}

Global plan:

{global_plan}

Task complete:

```
    {task_complete}

    Result:
    {result}

    Final status:
    {final_status}

    Trajectory:
    {trajectory}
    """
```

Лістинг Б.4 – Текстовий промпт агента глобального планування

```
    GLOBAL_PLANNER_PROMPT = f"""
    You are a Global Planner for a web navigation agent.

    Your job is to decompose the user's high-level task into a
short sequence of semantic subtasks.
    These subtasks will later be executed by a separate Local
Planner that can interact with the webpage.

    You are given:
    - the user's task
    - a screenshot of the current webpage

    Important rules:
    - Generate at most {MAX_LIMIT_PLACEHOLDER} subtasks.
    - Each subtask must be atomic, clear, and goal-oriented.
    - Each subtask must describe what needs to be achieved,
not how to click or type.
    - Use the screenshot context to make the plan specific to
the current webpage when possible.
    - The plan must directly support completion of the
original user task.
```

```
Return only valid JSON.
Do not return Markdown.
Do not wrap the JSON in a code block.

The JSON must have exactly this structure:
{{
  "result": [
    "First subtask",
    "Second subtask"
  ]
}}

User task:
{user_task}
"""
```

ДОДАТОК В

Вихідний програмний код оркестрації мультагентної системи

Лістинг В.1 – Вихідний код оркестрації мультагентної системи

```
from custom_code.Reviewer import review_global_progress
from custom_code.global_planner import generate_global_plan
from custom_code.knowledge_manager_agent import
review_trajectory_for_knowledge
from custom_code.local_planner import generate_local_action
from custom_code.webarena_browser_env import WebArenaEnv


user_task = "How many reviews does Josie Yoga Jacket have? And
what is their summary."
max_iterations = 10


def finish_execution(
        user_task: str,
        global_plan: list,
        trajectory: list,
        task_complete: bool,
        result: str,
        final_status: str,
        image_path: str | None
):
    review_trajectory_for_knowledge(
        user_task=user_task,
        global_plan=global_plan,
        trajectory=trajectory,
        task_complete=task_complete,
        result=result,
        final_status=final_status,
        image_path=image_path
    )
```

```
def run_agent():
    web_arena = WebArenaEnv()
    annotation, image_path = web_arena.setup_env()

    print(f"New Task: {user_task}")
    global_plan = generate_global_plan(user_task, image_path)

    trajectory = []
    iteration = 0

    while iteration < max_iterations:
        reasoner_hint,          task_complete,          result,
image_observation, action_effect = review_global_progress(
            user_task=user_task,
            global_plan=global_plan,
            last_actions=trajectory,
            image_path=image_path
        )

        if task_complete:
            print("Result:", result)
            finish_execution(
                user_task=user_task,
                global_plan=global_plan,
                trajectory=trajectory,
                task_complete=True,
                result=result,
                final_status="task_complete",
                image_path=image_path
            )
            return result

        action, reasoning = generate_local_action(
            annotation,
```

```
            image_path,
            reasoner_hint=reasoner_hint,
            image_observation=image_observation,
            action_effect=action_effect,
            user_task=user_task
        )

        trajectory.append({
            "iteration": iteration + 1,
            "image_observation": image_observation,
            "new_action": action,
            "new_action_reason": reasoning
        })

        print("Trajectory:", trajectory)

        try:
            annotation,      image_path,      terminated      =
web_arena.do_action(action)
        except Exception as exc:
            trajectory[-1]["execution_error"] = str(exc)
            print("Execution error:", exc)
            finish_execution(
                user_task=user_task,
                global_plan=global_plan,
                trajectory=trajectory,
                task_complete=False,
                result="",
                final_status="execution_error",
                image_path=image_path
            )
            return None

        iteration += 1

        if terminated:
```

```
            print("Environment terminated before reviewer
marked task complete.")
            finish_execution(
                user_task=user_task,
                global_plan=global_plan,
                trajectory=trajectory,
                task_complete=False,
                result="",
                final_status="environment_terminated",
                image_path=image_path
            )
            return None

    print("Maximum iteration limit reached.")
    finish_execution(
        user_task=user_task,
        global_plan=global_plan,
        trajectory=trajectory,
        task_complete=False,
        result="",
        final_status="max_iterations_reached",
        image_path=image_path
    )
    return None


run_agent()
```

Лістинг В.2 – Вихідний код оркестрації мультагентної системи

```
import json
import os
from datetime import datetime
from typing import List, Dict, Optional
```

```
class KnowledgeManager:
    def __init__(self, file_path: str = "knowledge_base.json"):
        self.file_path = file_path
        self.knowledge: List[Dict] = self.load()

    def load(self) -> List[Dict]:
        if os.path.exists(self.file_path):
            try:
                with open(self.file_path, "r", encoding="utf-
8") as f:
                    return json.load(f)
            except:
                return []
        return []

    def save(self):
        with open(self.file_path, "w", encoding="utf-8") as f:
            json.dump(self.knowledge, f, indent=2,
ensure_ascii=False)

    def add_successful_example(self, user_task: str,
global_plan: List[str],
                               final_observation: str = "",
success: bool = True):
        example = {
            "timestamp": datetime.now().isoformat(),
            "task": user_task.strip(),
            "plan": global_plan,
            "success": success,
            "final_observation": final_observation[:500],  #
truncate
            "tags": self._generate_tags(user_task)
        }
        self.knowledge.append(example)
        self.save()
```

```
    def _generate_tags(self, task: str) -> List[str]:
        tags = []
        keywords = ["buy", "add to cart", "search", "filter",
"best selling", "2022", "product"]
        for kw in keywords:
            if kw.lower() in task.lower():
                tags.append(kw)
        return tags

    def find_similar_examples(self, user_task: str, top_k: int
= 3) -> List[Dict]:
        scored = []
        for ex in self.knowledge:
            if not ex.get("success"):
                continue
            score     =     self._simple_similarity(user_task,
ex["task"])
            scored.append((score, ex))

        scored.sort(reverse=True, key=lambda x: x[0])
        return [ex for score, ex in scored[:top_k]]

    def _simple_similarity(self, a: str, b: str) -> float:
        words_a = set(a.lower().split())
        words_b = set(b.lower().split())
        if not words_a or not words_b:
            return 0.0
        intersection = len(words_a.intersection(words_b))
        return intersection / max(len(words_a), len(words_b))
```

ДОДАТОК Г

ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	91 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Махотіло В.К.			Метод динамічного планування для LLM-інтегрованих агентів на основі адаптивного синтезу політик
Перевірив		Шевченко О.Ю.			
Н.контр.		Малєєва І.А.			
Затв.		Чала Л.Е.			

Шифр групи	Код напр./спец.
СШІм-24-1	122
ХНУРЕ кафедра ШІ	