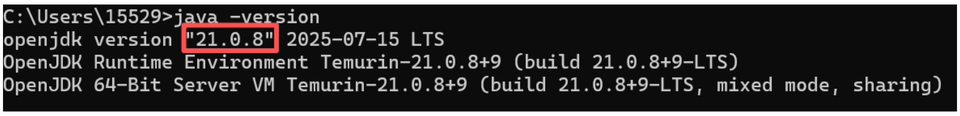
“Buggy Task Manager – Project Delivery Report” Prepared by Li Zi, October 2025

> **Thank you for the opportunity to work on these tasks — it was a valuable experience.**

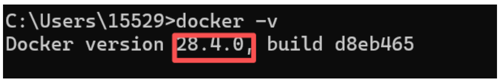
Environment

Component	Version
JDK	21
Payara Server	6.2024.5
Jakarta EE API	10.0.0
MySQL Connector	9.0.0
Maven Compiler Plugin	3.13.0
k6 (Load Testing)	

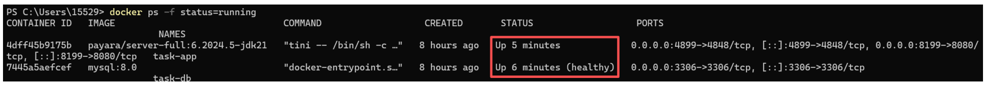
Task 1: Project Setup(DONE)

- Java 21(**installed**)
  - 

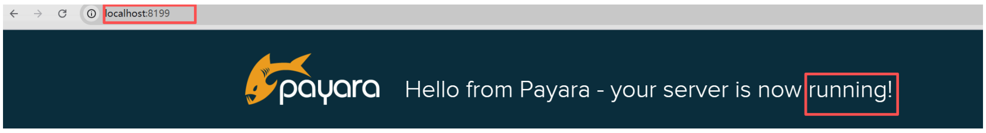
```
C:\Users\15529>java -version
openjdk version "21.0.8" 2025-07-15 LTS
OpenJDK Runtime Environment Temurin-21.0.8+9 (build 21.0.8+9-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.8+9 (build 21.0.8+9-LTS, mixed mode, sharing)
```

- Maven(**installed**)
  - 

```
C:\Users\15529>docker -v
Docker version 28.4.0, build d8eb465
```

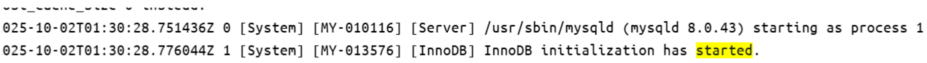
- Docker(**running**)
  - 

```
PS C:\Users\15529> docker ps -f status=running
CONTAINER ID   IMAGE                                 COMMAND                  CREATED         STATUS                     PORTS
                         NAMES
4dff45b9175b   payara/server-full:6.2024.5-jdk21    "tini -- /bin/sh -c …"   8 hours ago   Up 5 minutes               0.0.0.0:4899->4848/tcp, [::]:4899->4848/tcp, 0.0.0.0:8199->8080/
tcp, [::]:8199->8080/tcp   task-app
7445a5aefcef   mysql:8.0                             "docker-entrypoint.s…"   8 hours ago   Up 6 minutes (healthy)     0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp
                        task-db
```

- Payara(**running**)
  - 

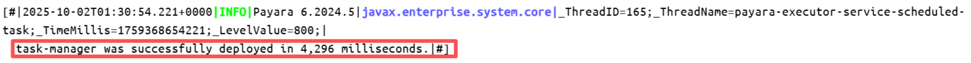
localhost:8199

Hello from Payara - your server is now running!

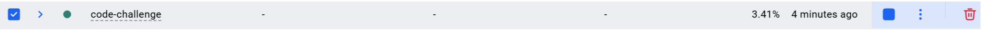
- MySQL(**running**)
  - 

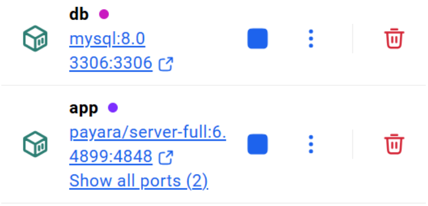
```
025-10-02T01:30:28.751436Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.43) starting as process 1
025-10-02T01:30:28.776044Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
```

- project(**deployed**)

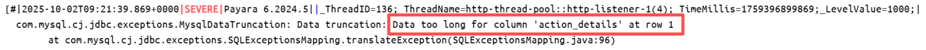


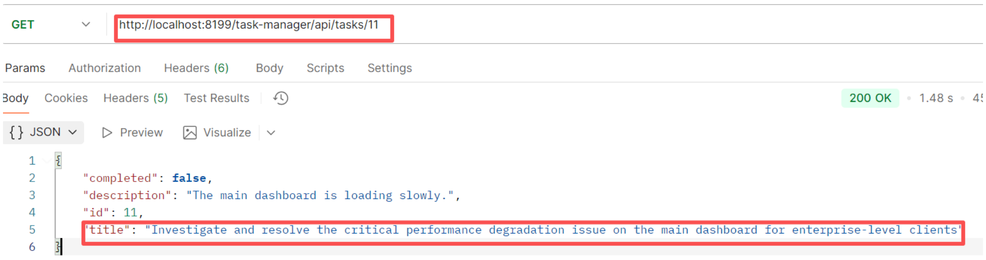
```
[#|2025-10-02T01:30:54.221+0000|INFO|Payara 6.2024.5|javax.enterprise.system.core|_ThreadID=165;_ThreadName=payara-executor-service-scheduled-task;_TimeMillis=1759368654221;_LevelValue=800;|
 task-manager was successfully deployed in 4,296 milliseconds.|#]
```



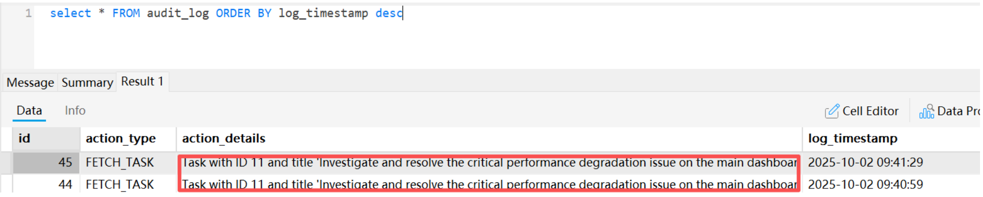


Task 2: Silent Failure Bug (solved)

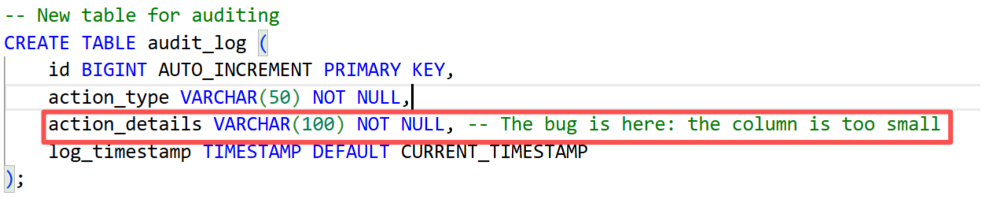
- scenario
  - Audit log creation on task view is implemented, but the `audit_log` table **remains empty**
    - 

      ```
      [#|2025-10-02T09:21:39.869+0000|SEVERE|Payara 6.2024.5||_ThreadID=136;_ThreadName=http-thread-pool::http-listener-1(4);_TimeMillis=1759396899869;_LevelValue=1000;|
        com.mysql.cj.jdbc.exceptions.MysqlDataTruncation: Data truncation: Data too long for column 'action_details' at row 1
              at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:96)
      ```
  - related interface
    - http://localhost:8199/task-manager/api/tasks/11 (**Get**)
- result(solved)
  - 

    http://localhost:8199/task-manager/api/tasks/11

    ```json
    {
      "completed": false,
      "description": "The main dashboard is loading slowly.",
      "id": 11,
      "title": "Investigate and resolve the critical performance degradation issue on the main dashboard for enterprise-level clients"
    }
    ```
  - 

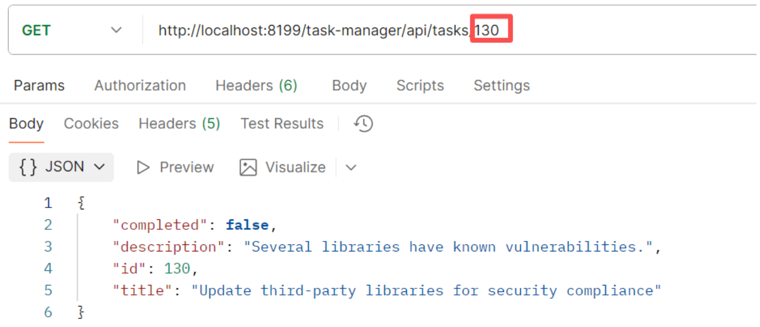
    ```sql
    select * FROM audit_log ORDER BY log_timestamp desc
    ```

id	action_type	action_details	log_timestamp
45	FETCH_TASK	Task with ID 11 and title 'Investigate and resolve the critical performance degradation issue on the main dashboar	2025-10-02 09:41:29
44	FETCH_TASK	Task with ID 11 and title 'Investigate and resolve the critical performance degradation issue on the main dashboar	2025-10-02 09:40:59
- root cause
  - 

    ```sql
    -- New table for auditing
    CREATE TABLE audit_log (
        id BIGINT AUTO_INCREMENT PRIMARY KEY,
        action_type VARCHAR(50) NOT NULL,
        action_details VARCHAR(100) NOT NULL, -- The bug is here: the column is too small
        log_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );
    ```
- solution

    1. Increase the column size（**recommend based on business**）
        - Pros
            - Simple change, **minimal migration** risk
            - **Better performance** for shorter data compared to TEXT
        - Cons
            - Still has a **hard upper** limit (e.g., VARCHAR(1000)), risk of hitting it again
            - Larger size may **waste storage** if most values are short
    2. change type from VARCHAR to TEXT
        - Pros
            - Can store **very large strings** (up to 64KB for TEXT)
        - Cons
            - **worse performance** than VARCHAR for small/medium datas
            - **Indexing is limited**
- change place
    - **in the initial sql about the audit_log filed action_details**

Task 3 : Incorrect Data Bug(solved)

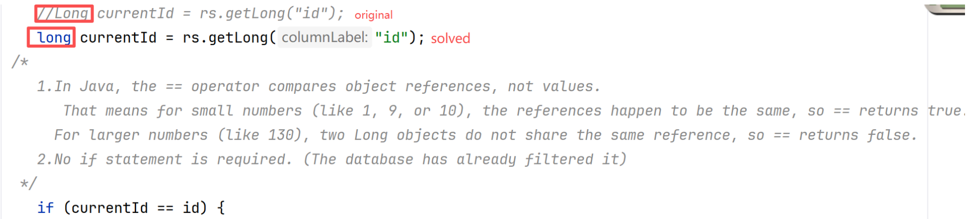
- scenario
    - when request a task with a **larger ID** the API returns a **404 Not Found** error,the task exists in the database.
    - **related interface**
        - http://localhost:8199/task-manager/api/tasks/130 （**Get**）
- result(solved)
    - 

http://localhost:8199/task-manager/api/tasks/130

```
{
    "completed": false,
    "description": "Several libraries have known vulnerabilities.",
    "id": 130,
    "title": "Update third-party libraries for security compliance"
}
```

- root cause

- `rs.getLong("id")` returns a **primitive long**, which is then **auto-boxed into a Long object** the **== operator** compares **object references**, not numerical values

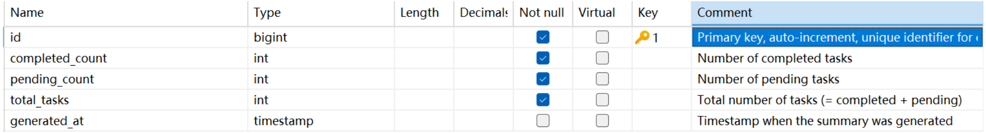
- 

```
  //Long currentId = rs.getLong("id"); original
 long currentId = rs.getLong( columnLabel: "id"); solved
/*
  1.In Java, the == operator compares object references, not values.
     That means for small numbers (like 1, 9, or 10), the references happen to be the same, so == returns true.
    For larger numbers (like 130), two Long objects do not share the same reference, so == returns false.
  2.No if statement is required. (The database has already filtered it)
 */
  if (currentId == id) {
```

- solution
  - Use **Long.equals()** Compare object values instead of references
  - Use **primitive long**，Avoid boxing/unboxing issues by working with primitives
  - (**recommend**)Remove the **redundant check** the **SQL query already filters** by `id` , no manual comparison is needed
- change place
  - ie.dpd.repository.TaskRepository#findById

Task 4: Implement a New Feature(DONE)

- Background
  - Managers need quick statistics:
    - How many tasks are completed?
    - How many are pending?
    - How many in total?
  - We need a new **Task Summary** feature to calculate and store these stats for future trend analysis.
- Design
  - **Database**: Add `task_summaries` table

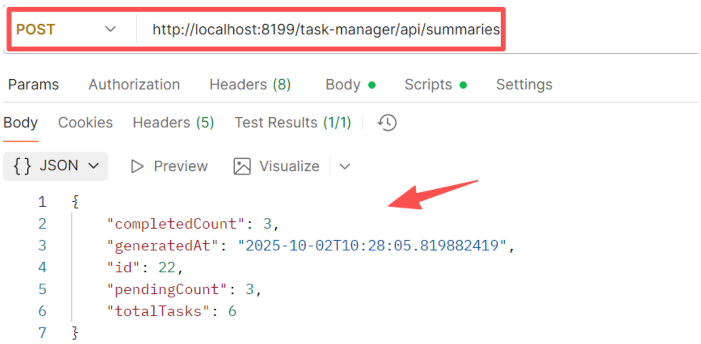


Name	Type	Length	Decimals	Not null	Virtual	Key	Comment
id	bigint			☑	□	1	Primary key, auto-increment, unique identifier for
completed_count	int			☑	□		Number of completed tasks
pending_count	int			☑	□		Number of pending tasks
total_tasks	int			☑	□		Total number of tasks (= completed + pending)
generated_at	timestamp			□	□		Timestamp when the summary was generated

  - **Repository**: `SummaryRepository` to
    - Generate a new summary from `tasks` and insert into `task_summaries`
    - Query all historical summaries(**in case manager just want to check and also prevent the data redundant**).

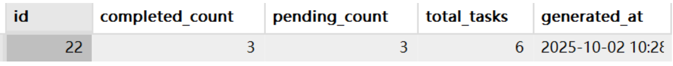
- **API**: `SummaryResource` with
  - **POST** **`/summaries`** → create new summary, return `201 Created`
  - **GET** **`/summaries`** → list all summaries.
- **Implementation**
  - Update `init.sql` with new table
  - Implement `SummaryRepository` with JDBC + `DataSource`
  - Implement `SummaryResource` REST endpoints
  - Test with Postman/curl and add to `k6` performance script

  1. For scalability, I added a **BaseRepository** for shared DB logic（only handles DB connections but can be extended with common reusable features later.）
  2. Created **SummaryRepository** for summary data and **SummaryResource** for API. This follows the **Single Responsibility Principle**, making the code easier to extend and maintain.
- **Result**
  - http://localhost:8199/task-manager/api/summaries (Post)



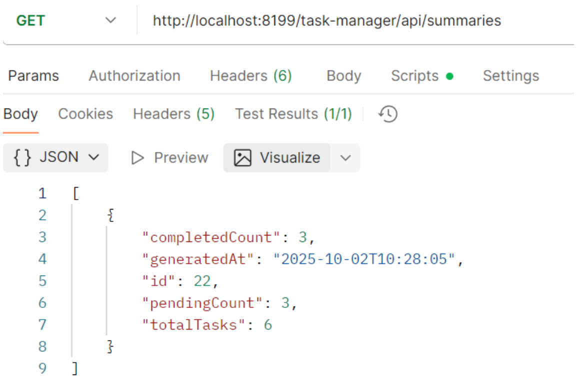
POST http://localhost:8199/task-manager/api/summaries

```
{
    "completedCount": 3,
    "generatedAt": "2025-10-02T10:28:05.819882419",
    "id": 22,
    "pendingCount": 3,
    "totalTasks": 6
}
```



id	completed_count	pending_count	total_tasks	generated_at
22	3	3	6	2025-10-02 10:2

  - http://localhost:8199/task-manager/api/summaries (Get)



GET http://localhost:8199/task-manager/api/summaries

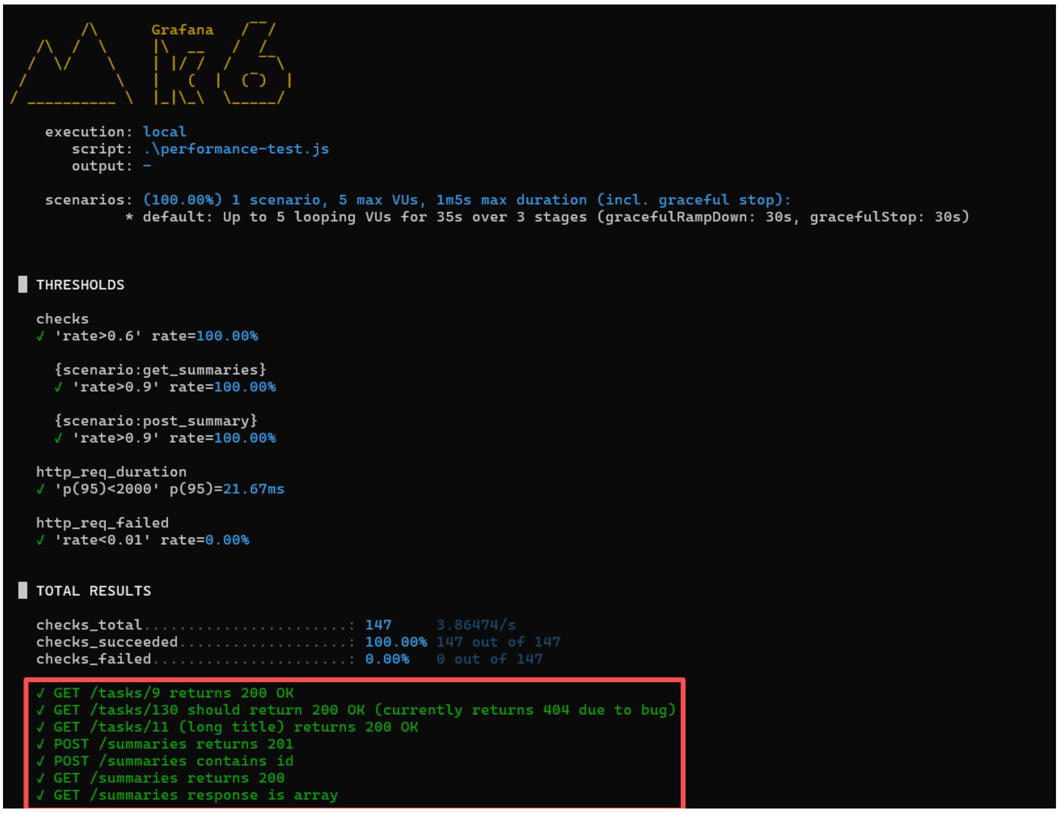
```
[
    {
        "completedCount": 3,
        "generatedAt": "2025-10-02T10:28:05",
        "id": 22,
        "pendingCount": 3,
        "totalTasks": 6
    }
]
```

- **code place**
  - ie.dpd.repository.SummaryRepository
  - ie.dpd.resource.SummaryResource
  - ie.dpd.repository.common.util.BaseRepository

Task 5: Performance and Load Testing(DONE)

Scenario ID	API Endpoint	Purpose	Expected Behavior	Thresholds
1	**GET /tasks**	Verify that the task list API works correctly.	Returns **200 OK** with an array of tasks.	`checks > 0.6` (≥60% pass)
2	**GET /tasks/9**	Verify fetching a task with a small ID works.	Returns **200 OK** with the task (ID=9).	`checks > 0.6`
3	**GET /tasks/130**	Reproduce the *equals vs == bug* (large IDs fail).	**Expected:** 200 OK with the task.**Current behavior (bug):** 404 Not Found.	`checks > 0.6`
4	**GET /tasks/11**	Trigger the *Silent Audit Failure bug* (long title → audit_log insert fails).	Returns **200 OK**. Should also insert into `audit_log` (currently fails silently).	`checks > 0.6`
5	**GET /tasks/1**	Simulate frequent queries to test DB connection pool stability.	Returns **200 OK** consistently, no connection errors.	`checks > 0.6`

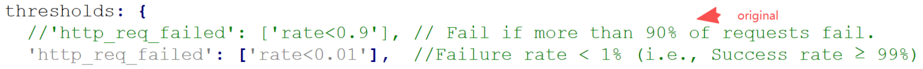
Scenario ID	API Endpoint	Purpose	Expected Behavior	Thresholds
6	POST /summaries	Test the new feature: generate a task summary report.	Returns **201 Created**, JSON body contains `id` and task counts.	`checks{scenario:post_summary} > 0.9` (≥90% pass)
7	GET /summaries	Verify retrieval of stored summaries (historical tracking).	Returns **200 OK**, response is an array of summary records.	`checks{scenario:get_summaries} > 0.9` (≥90% pass)



```
         /\      Grafana   /‾‾/
    /\  /  \     |\  __   /  /
   /  \/    \    | |/ /  /   ‾‾\
  /          \   |   (  |  (‾)  |
 / __________ \  |_|\_\  \_____/

     execution: local
        script: .\performance-test.js
        output: -

     scenarios: (100.00%) 1 scenario, 5 max VUs, 1m5s max duration (incl. graceful stop):
              * default: Up to 5 looping VUs for 35s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)



█ THRESHOLDS

    checks
    ✓ 'rate>0.6' rate=100.00%

      {scenario:get_summaries}
      ✓ 'rate>0.9' rate=100.00%

      {scenario:post_summary}
      ✓ 'rate>0.9' rate=100.00%

    http_req_duration
    ✓ 'p(95)<2000' p(95)=21.67ms

    http_req_failed
    ✓ 'rate<0.01' rate=0.00%


█ TOTAL RESULTS

    checks_total.......................: 147     3.86474/s
    checks_succeeded...................: 100.00% 147 out of 147
    checks_failed......................: 0.00%   0 out of 147

    ✓ GET /tasks/9 returns 200 OK
    ✓ GET /tasks/130 should return 200 OK (currently returns 404 due to bug)
    ✓ GET /tasks/11 (long title) returns 200 OK
    ✓ POST /summaries returns 201
    ✓ POST /summaries contains id
    ✓ GET /summaries returns 200
    ✓ GET /summaries response is array
```

Potential bug

- db
    - Normally, the id in the **tasks table is auto-incremented.** Here, IDs like 11 and 130 are manually inserted to reproduce known bugs (long title → audit log failure, large ID → equals vs == bug).

- performance test
  - **Bug:** The original threshold (`rate<0.9`) was misconfigured, allowing very poor success rates.
  - **Fix:** Use `rate<0.01` so the system must maintain ≥**99% success rate** to pass the test.
  - 

```
thresholds: {
  //'http_req_failed': ['rate<0.9'], // Fail if more than 90% of requests fail.
  'http_req_failed': ['rate<0.01'],  //Failure rate < 1% (i.e., Success rate ≥ 99%)
```

original

- code
  - The original SQL statement mistakenly used FROM tasks = ? (**invalid syntax**) instead of a proper WHERE clause. This caused queries with larger IDs (e.g., 130) to fail. The corrected version uses WHERE id = ?, which properly filters by the task's primary key and fixes the bug
  - **code place:** ie/dpd/repository/TaskRepository.java:52

Problem confront

1. Deployment Issue (Docker volume mapping error)

- **Problem:** Because of a wrong volume mount path in Docker, an empty WAR file was deployed to Payara. The Payara console misleadingly showed "deployment successful," which caused confusion. I suspected Payara's loading path, `payara-web.xml` , and even Jakarta API version issues.
- **Root Cause:** Incorrect volume mapping between host and container.
- **Solution:** By inspecting Docker's mounted directories, I found the mapping error and fixed it.

2. Summary Report Implementation Concern

- **Problem:** If every time `/summaries` is accessed we insert a new row into the database, it could cause issues such as:
  - Potential for malicious abuse (spamming requests).
  - Data redundancy in the `task_summaries` table.
- **Decision:** Due to time constraints, I added a **second endpoint** for fetching summaries without inserting new records. This allows managers to query historical summaries without creating unnecessary data.
- **Consideration:** Directly querying the `tasks` table with aggregation functions could be costly if the dataset grows large. Querying from the `task_summaries` table is more efficient, but if querying the raw `tasks` table is required, proper SQL optimization should be applied.

Relevant script used during solving

```
# Start with rebuild
docker compose -f code-challenge/docker-compose.yaml up --build
# Stop and clean
docker compose -f code-challenge/docker-compose.yaml down -v
# Check WAR in deployments
docker exec -it task-app ls -l /opt/payara/deployments/
# View logs
docker logs -f task-app
# Quick WAR check
docker exec -it task-app ls /opt/payara/deployments
# List deployed apps
docker exec -it task-app /opt/payara/appserver/bin/asadmin list-applications
# Stop all containers
docker ps -aq | % { docker rm -f $_ }
# Remove all images
docker images -aq | % { docker rmi -f $_ }
# Remove all volumes
docker volume ls -q | % { docker volume rm $_ }
```

Summary

All assigned tasks have been **completed successfully, including bug fixes, new feature development, database enhancements, and performance testing**. The project is working as expected, with a clearer structure for future improvements.

- **Name**: Li Zi
- **Phone**: 089-451-9683
- **Email**: liziglen7@gmail.com

Task	Issue	Cause	Fix	Code Change
Audit Log Bug	`audit_log` always empty	Column too small / wrong type	Increase size or use `TEXT`	`init.sql`
Large ID Bug	Task with ID 130 → 404	Used `==` on `Long`, SQL typo	Use `equals()` / `long`; fix `WHERE id = ?`	`TaskRepository.java`
Task Summary	NA	NA	Add `task_summaries`, new endpoints	`SummaryRepository`, `SummaryResource`
Performance Test	Threshold too loose	Wrong config (`<0.9`)	Require ≥99% success rate	`performance-test.js`

Task	Issue	Cause	Fix	Code Change
Deployment Issue	Empty WAR deployed	Wrong Docker volume mapping	Fix volume path	`docker-compose.yaml`

the code also maintain by **git** code-challenge/buggy-task-manager at master · GlenZiLi/code-challenge · GitHub