<u>Dulanji Cooray -Full Stack Engineer Take Home Assessment Answers</u>

System Architecture Overview

Your application will consist of three main components running in separate Docker containers:

Component	Technology Choice	Responsibilities
Database	PostgreSQL	Persist task data in a relational table
Backend API	Java with Spring Boot	Provide REST API endpoints for task operations
Frontend UI	HTML, CSS, Vanilla JavaScript	Single-Page Application for user interactions

The communication flow is as follows:

1. **User** interacts with the Frontend UI (clicks, forms)
2. **Frontend** sends HTTP requests to the Backend API (fetch, XMLHttpRequest)
3. **Backend** processes requests and interacts with the Database (JDBC, JPA)
4. **Database** stores and returns persistent task data

Database Design

You'll need a single task table. Here is the recommended schema:

Column Name	Data Type	Constraints	Description
id	BIGSERIAL or SERIAL	PRIMARY KEY	Unique auto-incrementing task identifier
title	VARCHAR(255)	NOT NULL	The main title/headline of the task
description	TEXT	-	Detailed description of the task (optional)
completed	BOOLEAN	NOT NULL DEFAULT FALSE	Completion status (TRUE if done)

created_at	TIMESTAMP	NOT NULL DEFAULT NOW()	Auto-recorded creation timestamp

This design supports all user requirements: storing title/description, tracking completion status, and using created_at to fetch the most recent tasks.

Backend API Design (Java Spring Boot)

The backend will be a RESTful API with the following endpoints:

HTTP Method	Endpoint	Request Body	Description
GET	/api/tasks	-	Fetches the 5 most recent **incomplete** tasks
POST	/api/tasks	{"title": "string", "description": "string"}	Creates a new task
PUT	/api/tasks/{id}/complete	-	Marks a specific task as completed

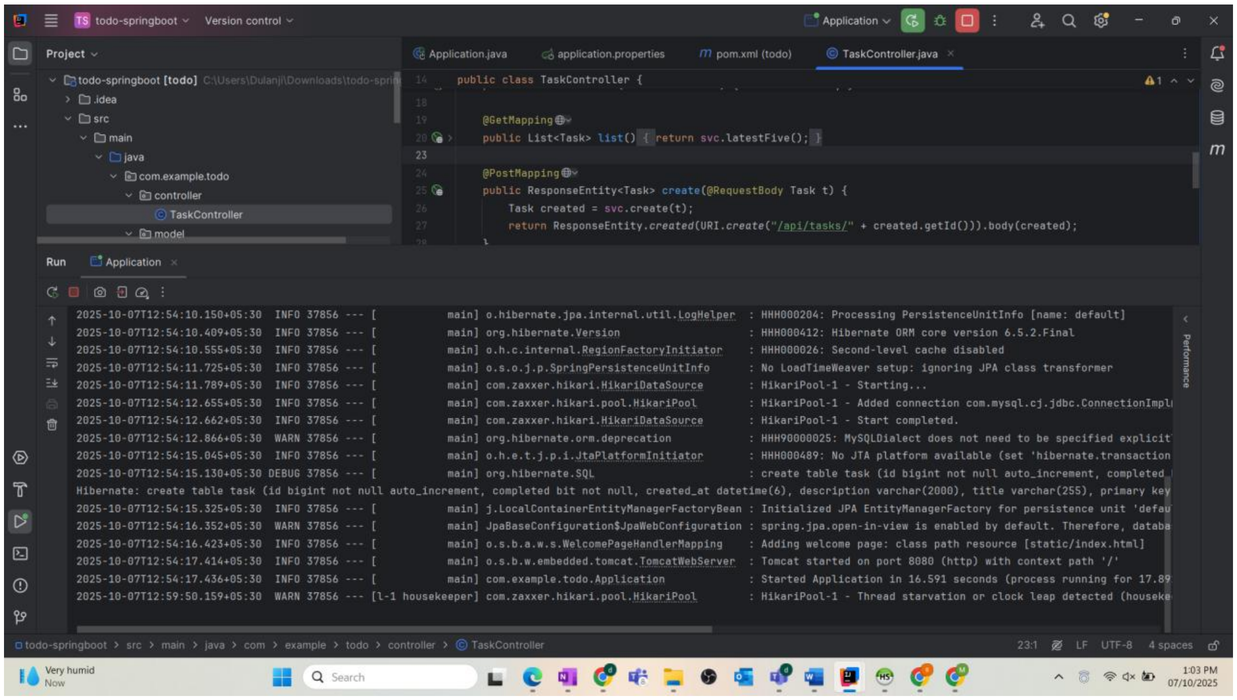
Applying SOLID Principles in Your Backend Code

Following SOLID principles is a key evaluation criterion. Here's how to apply them in your Java code:

- **Single Responsibility Principle (SRP)**: Each class has one reason to change.
  - TaskController: Handles HTTP requests and responses.
  - TaskService: Contains the core business logic (e.g., ensuring only 5 tasks are shown).
  - TaskRepository: Manages all data access and interaction with the database.
- **Dependency Inversion Principle (DIP)**: Depend on abstractions, not concrete implementations.
  - **Create a** TaskService **interface**: Define methods like List<Task> getRecentIncompleteTasks(), Task createTask(Task task), etc.

- **Create a** TaskServiceImpl **class**: The concrete implementation of the interface.
- **Inject the interface**: Your TaskController should depend on the TaskService interface, not the TaskServiceImpl class. This makes your code more flexible and easier to test.

Project Run and Screenshots

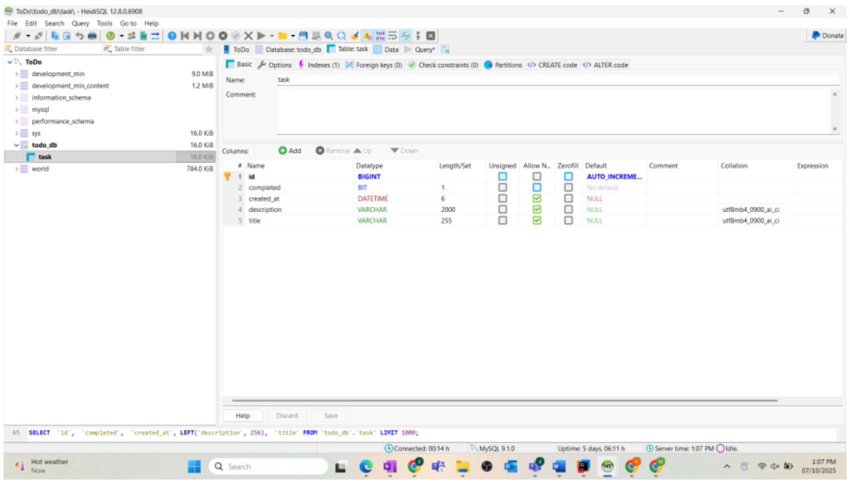


```java
public class TaskController {

    @GetMapping
    public List<Task> list() { return svc.latestFive(); }

    @PostMapping
    public ResponseEntity<Task> create(@RequestBody Task t) {
        Task created = svc.create(t);
        return ResponseEntity.created(URI.create("/api/tasks/" + created.getId())).body(created);
```

```
2025-10-07T12:54:10.150+05:30  INFO 37856 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-10-07T12:54:10.409+05:30  INFO 37856 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.5.2.Final
2025-10-07T12:54:10.555+05:30  INFO 37856 --- [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-10-07T12:54:11.725+05:30  INFO 37856 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-10-07T12:54:11.789+05:30  INFO 37856 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-10-07T12:54:12.655+05:30  INFO 37856 --- [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl
2025-10-07T12:54:12.662+05:30  INFO 37856 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-10-07T12:54:12.866+05:30  WARN 37856 --- [           main] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not need to be specified explicit
2025-10-07T12:54:15.045+05:30  INFO 37856 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction
2025-10-07T12:54:15.130+05:30 DEBUG 37856 --- [           main] org.hibernate.SQL                        : create table task (id bigint not null auto_increment, completed
Hibernate: create table task (id bigint not null auto_increment, completed bit not null, created_at datetime(6), description varchar(2000), title varchar(255), primary key
2025-10-07T12:54:15.325+05:30  INFO 37856 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'defau
2025-10-07T12:54:16.352+05:30  WARN 37856 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, databa
2025-10-07T12:54:16.423+05:30  INFO 37856 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2025-10-07T12:54:17.414+05:30  INFO 37856 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-10-07T12:54:17.436+05:30  INFO 37856 --- [           main] com.example.todo.Application             : Started Application in 16.591 seconds (process running for 17.89
2025-10-07T12:59:50.159+05:30  WARN 37856 --- [l-1 housekeeper] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Thread starvation or clock leap detected (houseke
```



Name: task

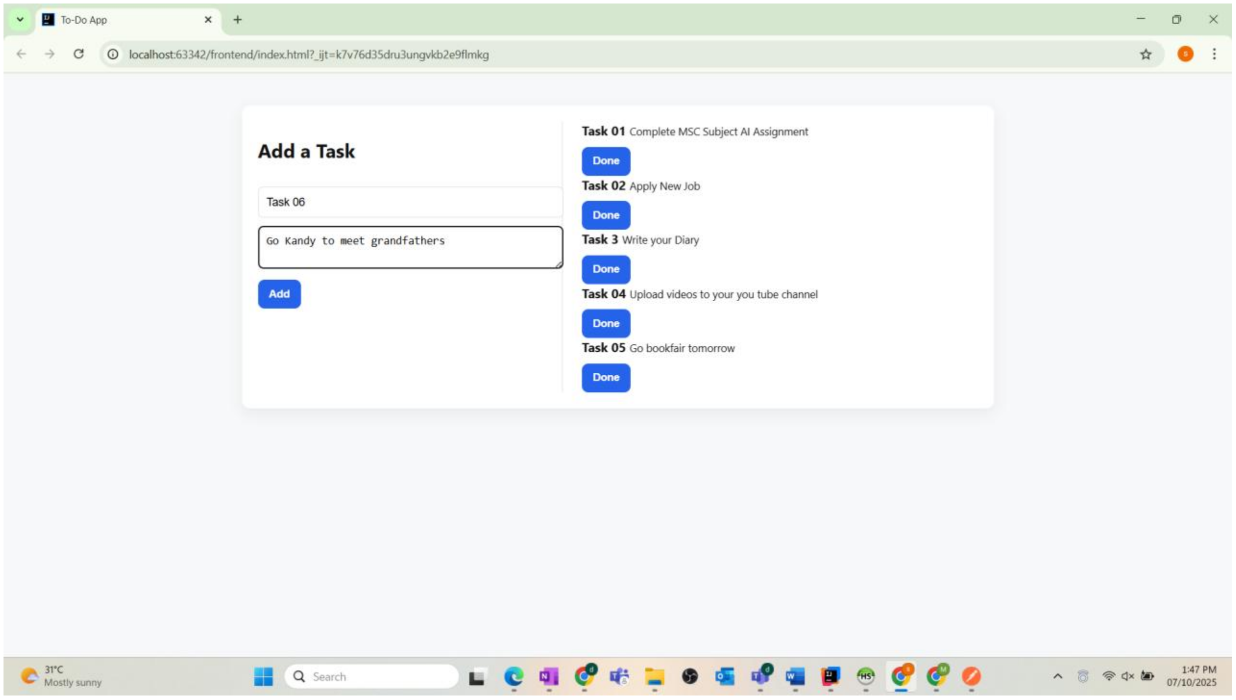
Comment:

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comment	Collation	Expression
1	id	BIGINT		☐	☐	☐	AUTO_INCREME...			
2	completed	BIT	1	☐	☐	☐	No default			
3	created_at	DATETIME	6	☐	✓	☐	NULL			
4	description	VARCHAR	2000	☐	✓	☐	NULL		utf8mb4_0900_ai_ci	
5	title	VARCHAR	255	☐	✓	☐	NULL		utf8mb4_0900_ai_ci	

```sql
SELECT `id`, `completed`, `created_at`, LEFT(`description`, 256), `title` FROM `todo_db`.`task` LIMIT 1000;
```

Connected: 00:14 h   MySQL 9.1.0   Uptime: 5 days, 06:11 h   Server time: 1:07 PM   Idle.

<u>My Project Prive After Run</u>


To-Do App

localhost:63342/frontend/index.html?_ijt=k7v76d35dru3ungvkb2e9flmkg

Add a Task

Task 06

Go Kandy to meet grandfathers

Add

Task 01 Complete MSC Subject AI Assignment

Done

Task 02 Apply New Job

Done

Task 3 Write your Diary

Done

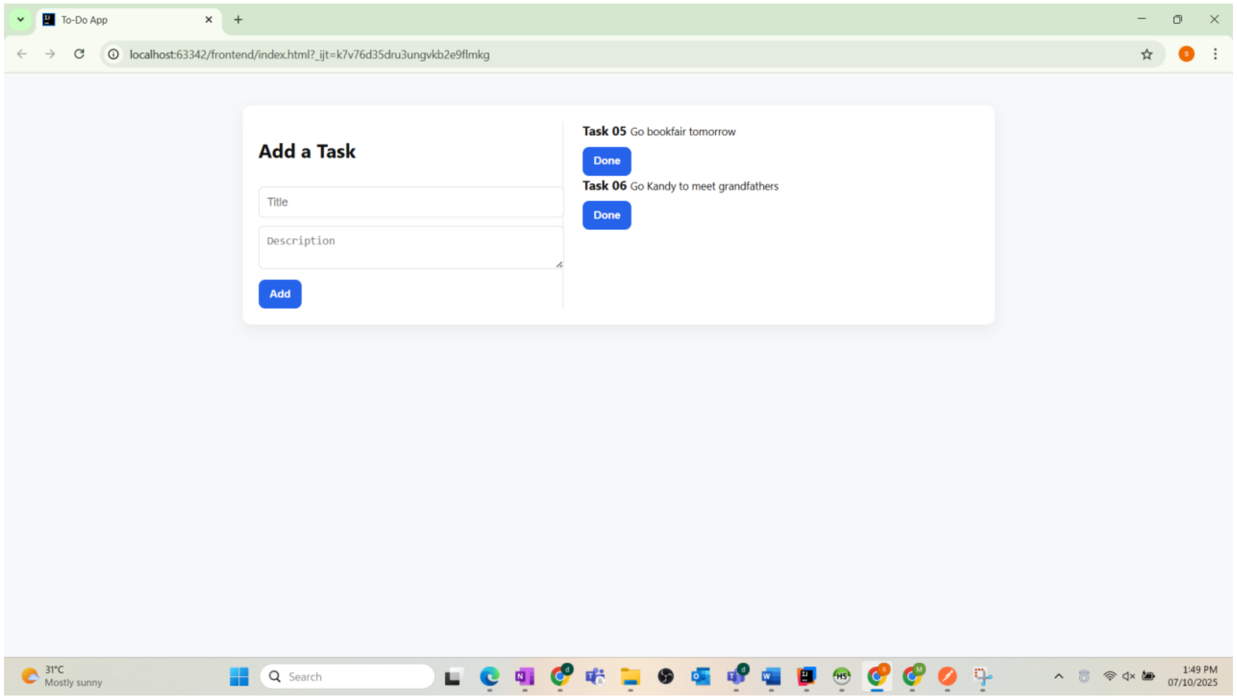
Task 04 Upload videos to your you tube channel

Done

Task 05 Go bookfair tomorrow

Done

31°C Mostly sunny 1:47 PM 07/10/2025


To-Do App

localhost:63342/frontend/index.html?_ijt=k7v76d35dru3ungvkb2e9flmkg

Add a Task

Title

Description

Add

Task 05 Go bookfair tomorrow

Done

Task 06 Go Kandy to meet grandfathers

Done

31°C Mostly sunny 1:49 PM 07/10/2025

curl Commands for Your Task API

1. GET All Tasks

bash

```
curl -X GET http://localhost:8080/api/tasks \
 -H "Content-Type: application/json"
```

2. Create a New Task

bash

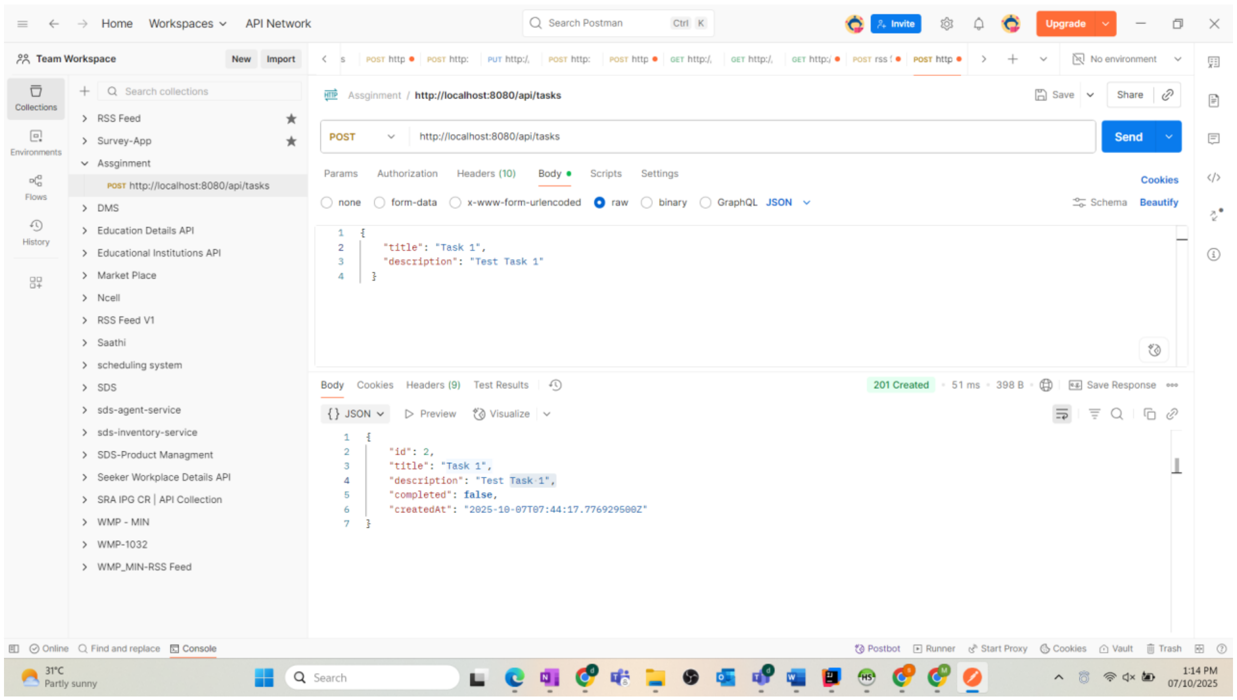
```
curl -X POST http://localhost:8080/api/tasks \
 -H "Content-Type: application/json" \
 -d '{
   "title": "Learn cURL testing",
   "description": "Test the POST endpoint using cURL"
 }'
```

3. Mark Task as Done

bash

```
# Replace {id} with actual task ID from the create response
curl -X POST http://localhost:8080/api/tasks/1/done \
 -H "Content-Type: application/json"
```

Create Task



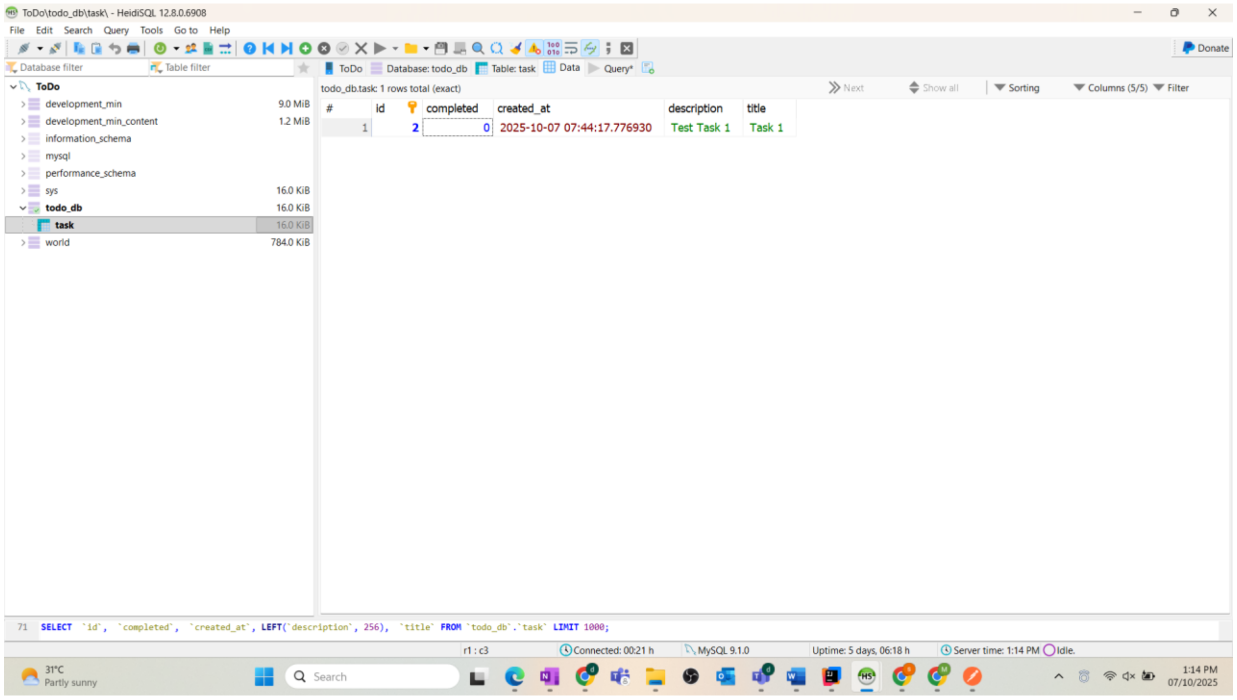
POST http://localhost:8080/api/tasks

Body (raw, JSON):

```
{
    "title": "Task 1",
    "description": "Test Task 1"
}
```

Response: 201 Created · 51 ms · 398 B

```
{
    "id": 2,
    "title": "Task 1",
    "description": "Test Task 1",
    "completed": false,
    "createdAt": "2025-10-07T07:44:17.776929500Z"
}
```



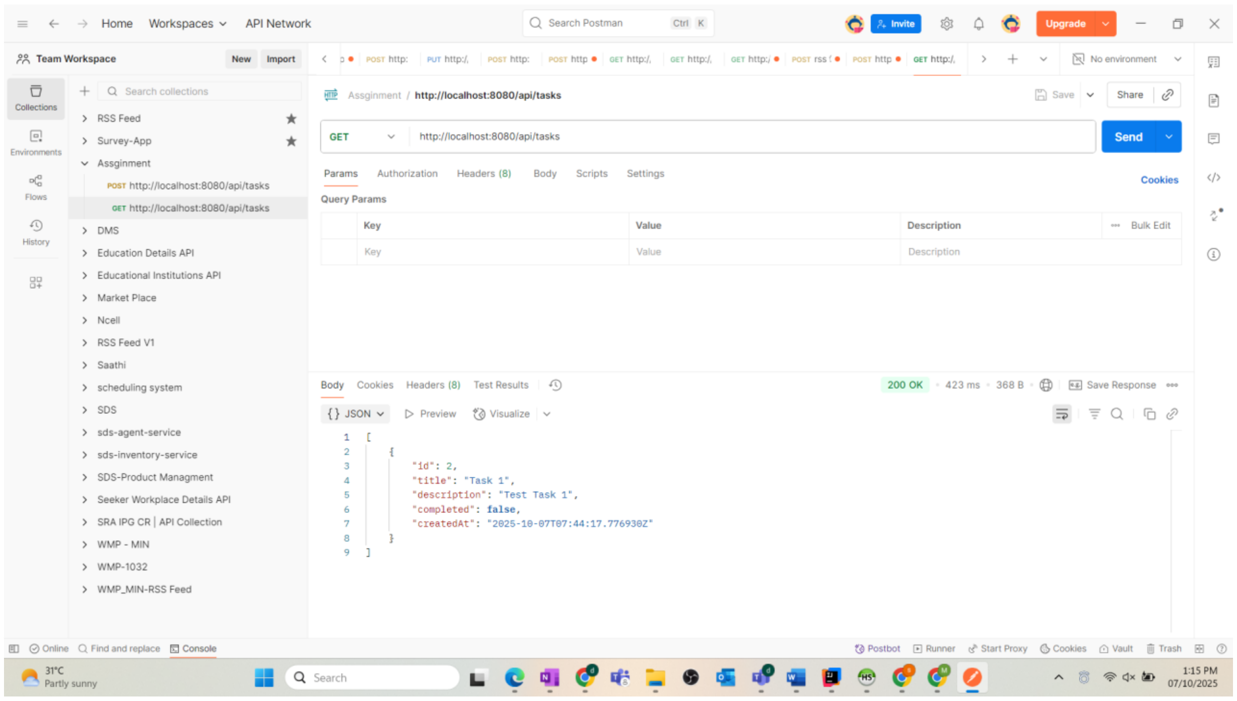
ToDo\todo_db\task\ - HeidiSQL 12.8.0.6908

todo_db.task: 1 rows total (exact)

#	id	completed	created_at	description	title
1	2	0	2025-10-07 07:44:17.776930	Test Task 1	Task 1

```
SELECT `id`, `completed`, `created_at`, LEFT(`description`, 256), `title` FROM `todo_db`.`task` LIMIT 1000;
```

Get Tasks list



Assignment / http://localhost:8080/api/tasks

GET http://localhost:8080/api/tasks

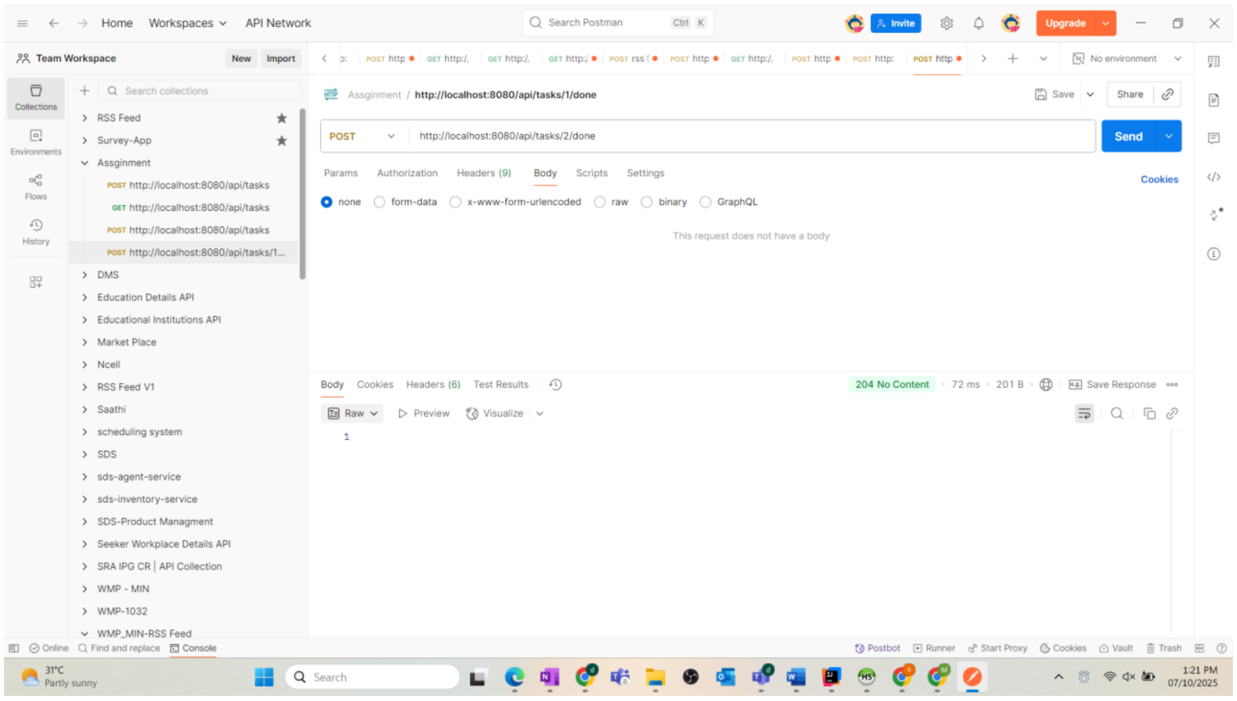
Query Params

Key	Value	Description
Key	Value	Description

200 OK · 423 ms · 368 B

```
[
    {
        "id": 2,
        "title": "Task 1",
        "description": "Test Task 1",
        "completed": false,
        "createdAt": "2025-10-07T07:44:17.776930Z"
    }
]
```

Mark Task as Done



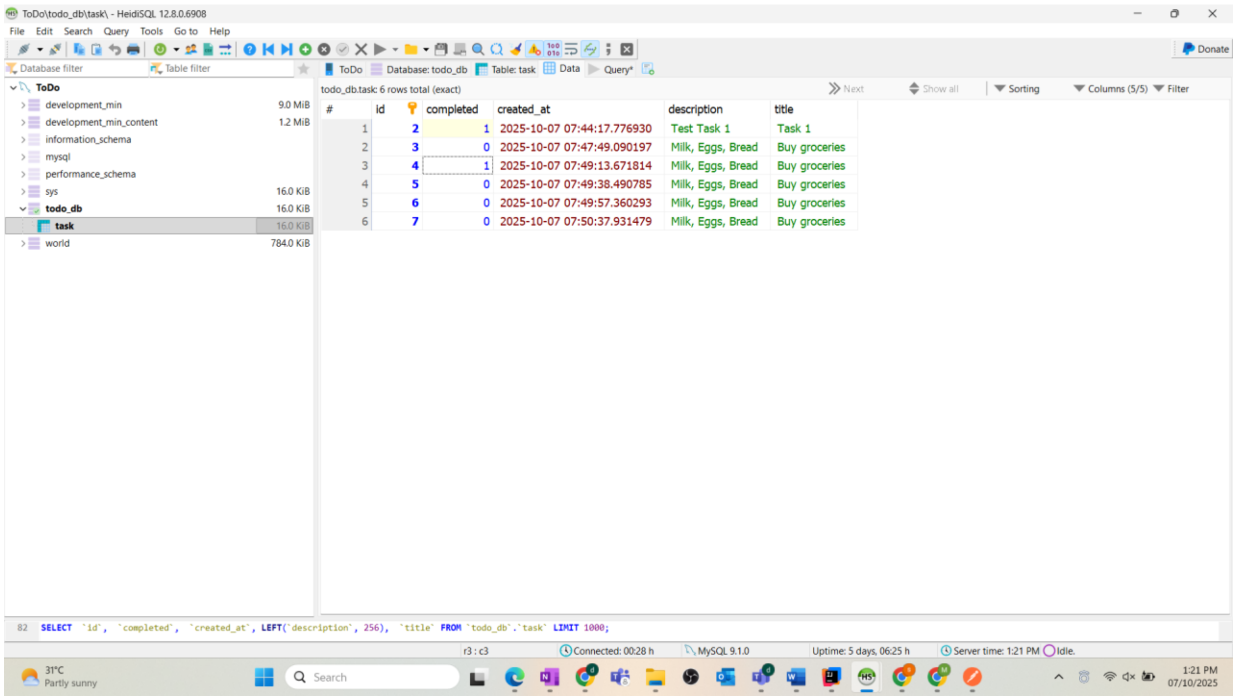
Assignment / http://localhost:8080/api/tasks/1/done

POST http://localhost:8080/api/tasks/2/done

Body: [✓ none] form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

204 No Content · 72 ms · 201 B



ToDo	
development_min	9.0 MiB
development_min_content	1.2 MiB
information_schema	
mysql	
performance_schema	
sys	16.0 KiB
todo_db	16.0 KiB
task	16.0 KiB
world	784.0 KiB

todo_db.task: 6 rows total (exact)

#	id	completed	created_at	description	title
1	2	1	2025-10-07 07:44:17.776930	Test Task 1	Task 1
2	3	0	2025-10-07 07:47:49.090197	Milk, Eggs, Bread	Buy groceries
3	4	1	2025-10-07 07:49:13.671814	Milk, Eggs, Bread	Buy groceries
4	5	0	2025-10-07 07:49:38.490785	Milk, Eggs, Bread	Buy groceries
5	6	0	2025-10-07 07:49:57.360293	Milk, Eggs, Bread	Buy groceries
6	7	0	2025-10-07 07:50:37.931479	Milk, Eggs, Bread	Buy groceries

```
82 SELECT `id`, `completed`, `created_at`, LEFT(`description`, 256), `title` FROM `todo_db`.`task` LIMIT 1000;
```

```
r3 : c3    Connected: 00:28 h    MySQL 9.1.0    Uptime: 5 days, 06:25 h    Server time: 1:21 PM    Idle.
```