VIT Vidyalankar Institute of Technology Accredited A+ by NAAC	**DEPARTMENT OF COMPUTER ENGINEERING**

Experiment No. 03

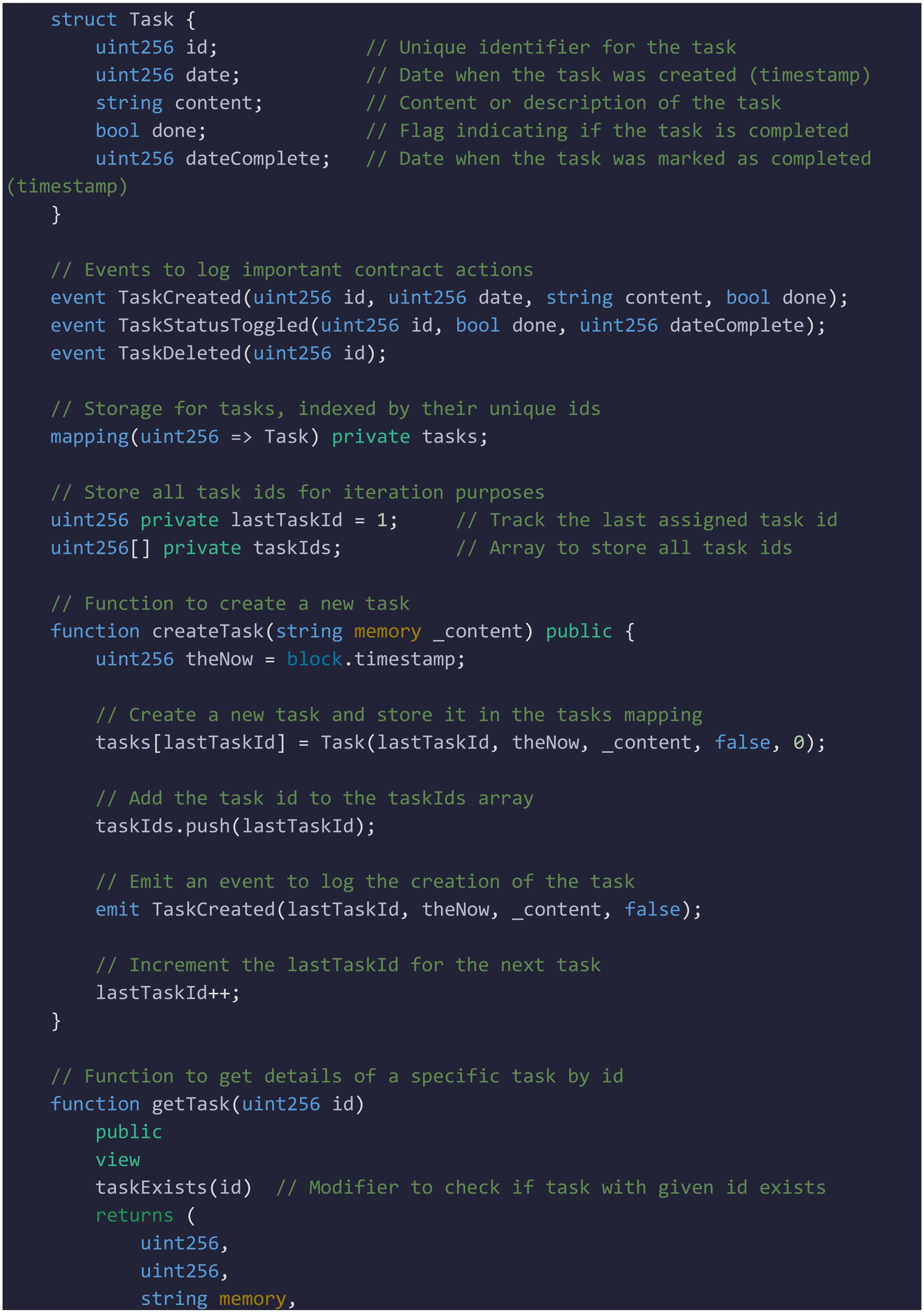
Semester	B.E. Semester VII – Computer Engineering
Subject	Blockchain Lab (CSDL7022)
Subject Professor In-charge	Prof. Swapnil S. Sonawane
Academic Year	2024-25

Student Name	Deep Salunkhe
Roll Number	21102A0014

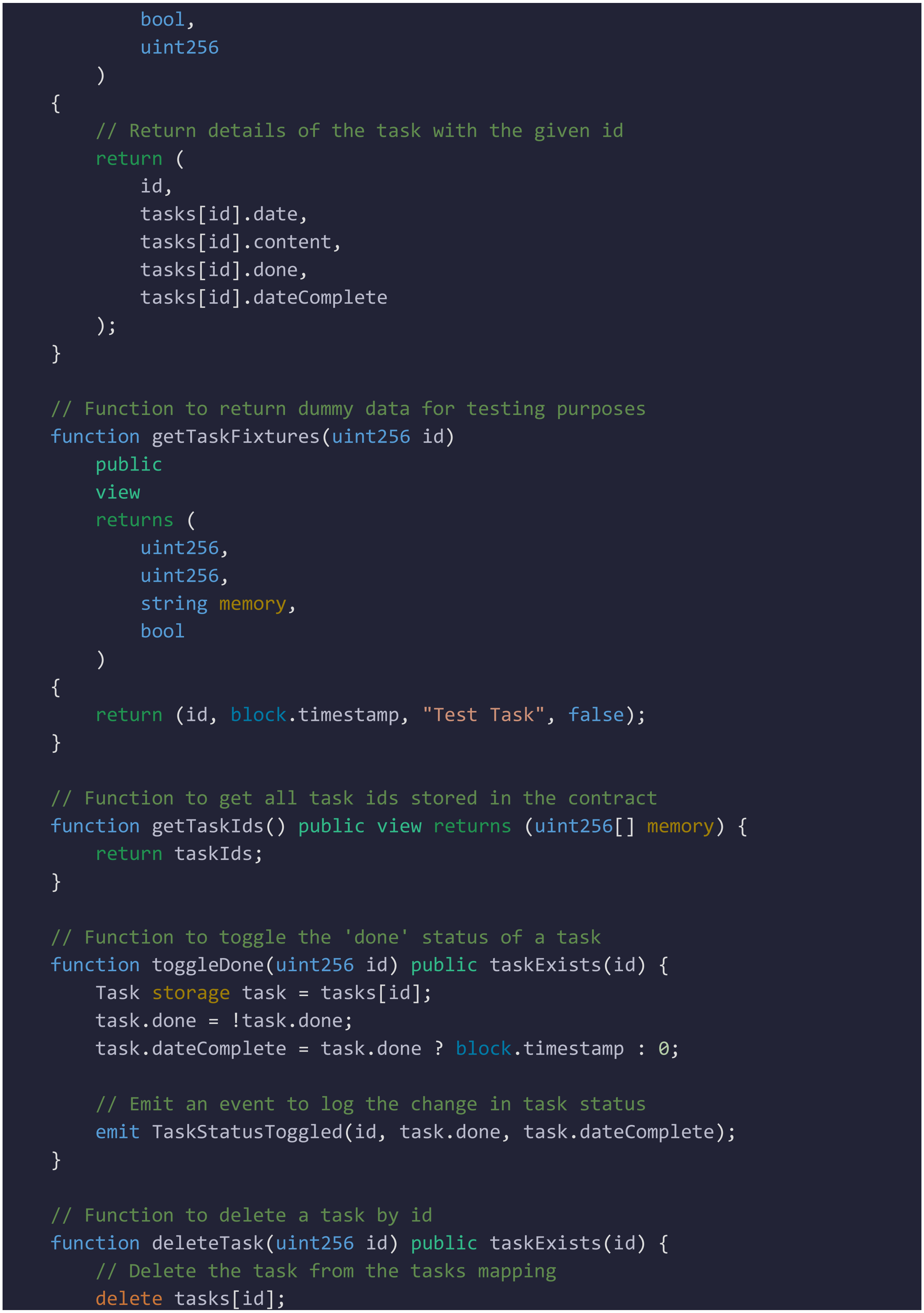
Title: Interacting with deployed smart contracts.

Program Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract todo {


     // Receive function to receive Ether
    receive() external payable {
        // Optionally, you can log the received Ether amount or perform other
actions
        // For example, emit an event to log the received Ether amount
        emit ReceivedEther(msg.sender, msg.value);
    }

    // Fallback function to receive Ether and handle any other calls
    fallback() external payable {
        // Optional: log the received Ether amount or perform other actions
        // This function is called when no other function matches the function
signature
        emit FallbackCalled(msg.sender, msg.value);
    }

    event ReceivedEther(address indexed sender, uint256 amount);
    event FallbackCalled(address indexed sender, uint256 amount);
    // Define a struct to represent a Task
```

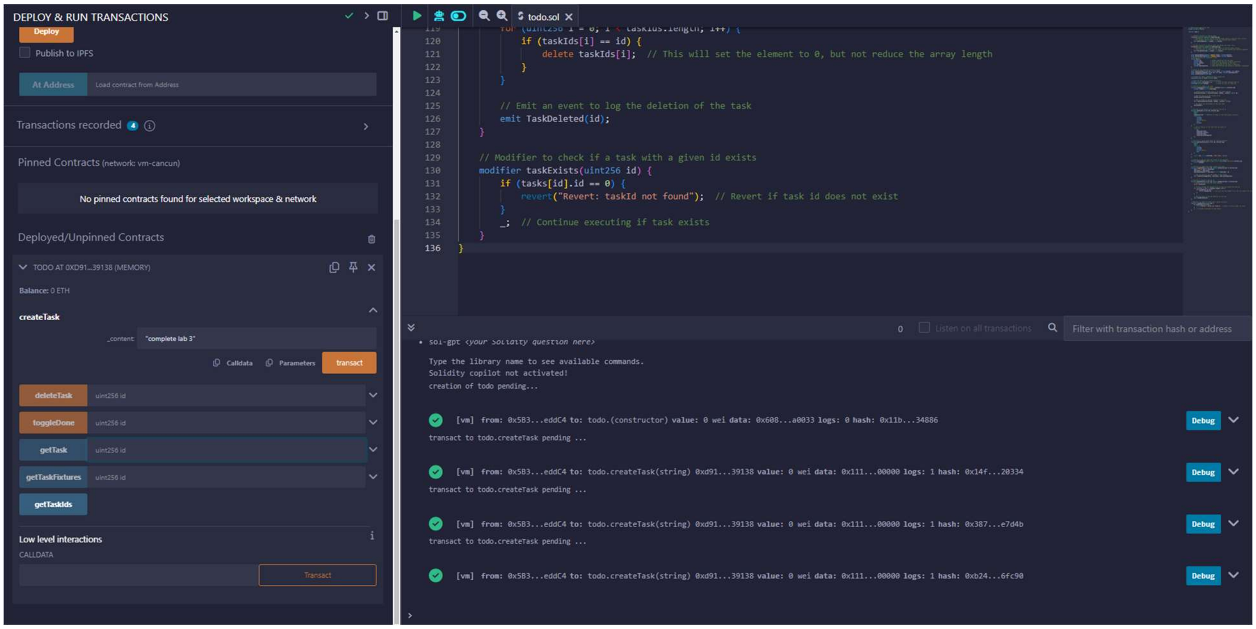


```
    struct Task {
        uint256 id;             // Unique identifier for the task
        uint256 date;           // Date when the task was created (timestamp)
        string content;         // Content or description of the task
        bool done;              // Flag indicating if the task is completed
        uint256 dateComplete;   // Date when the task was marked as completed
(timestamp)
    }

    // Events to log important contract actions
    event TaskCreated(uint256 id, uint256 date, string content, bool done);
    event TaskStatusToggled(uint256 id, bool done, uint256 dateComplete);
    event TaskDeleted(uint256 id);

    // Storage for tasks, indexed by their unique ids
    mapping(uint256 => Task) private tasks;

    // Store all task ids for iteration purposes
    uint256 private lastTaskId = 1;     // Track the last assigned task id
    uint256[] private taskIds;          // Array to store all task ids

    // Function to create a new task
    function createTask(string memory _content) public {
        uint256 theNow = block.timestamp;

        // Create a new task and store it in the tasks mapping
        tasks[lastTaskId] = Task(lastTaskId, theNow, _content, false, 0);

        // Add the task id to the taskIds array
        taskIds.push(lastTaskId);

        // Emit an event to log the creation of the task
        emit TaskCreated(lastTaskId, theNow, _content, false);

        // Increment the lastTaskId for the next task
        lastTaskId++;
    }

    // Function to get details of a specific task by id
    function getTask(uint256 id)
        public
        view
        taskExists(id)  // Modifier to check if task with given id exists
        returns (
            uint256,
            uint256,
            string memory,
```



```
            bool,
            uint256
        )
    {
        // Return details of the task with the given id
        return (
            id,
            tasks[id].date,
            tasks[id].content,
            tasks[id].done,
            tasks[id].dateComplete
        );
    }

    // Function to return dummy data for testing purposes
    function getTaskFixtures(uint256 id)
        public
        view
        returns (
            uint256,
            uint256,
            string memory,
            bool
        )
    {
        return (id, block.timestamp, "Test Task", false);
    }

    // Function to get all task ids stored in the contract
    function getTaskIds() public view returns (uint256[] memory) {
        return taskIds;
    }

    // Function to toggle the 'done' status of a task
    function toggleDone(uint256 id) public taskExists(id) {
        Task storage task = tasks[id];
        task.done = !task.done;
        task.dateComplete = task.done ? block.timestamp : 0;

        // Emit an event to log the change in task status
        emit TaskStatusToggled(id, task.done, task.dateComplete);
    }

    // Function to delete a task by id
    function deleteTask(uint256 id) public taskExists(id) {
        // Delete the task from the tasks mapping
        delete tasks[id];
        // Iterate through the taskIds array to find and remove the task id
        for (uint256 i = 0; i < taskIds.length; i++) {
            if (taskIds[i] == id) {
                delete taskIds[i];  // This will set the element to 0, but not reduce the array length
            }
        }

        // Emit an event to log the deletion of the task
        emit TaskDeleted(id);
    }

    // Modifier to check if a task with a given id exists
    modifier taskExists(uint256 id) {
        if (tasks[id].id == 0) {
            revert("Revert: taskId not found");  // Revert if task id does not exist
        }
        _;  // Continue executing if task exists
    }
}
```

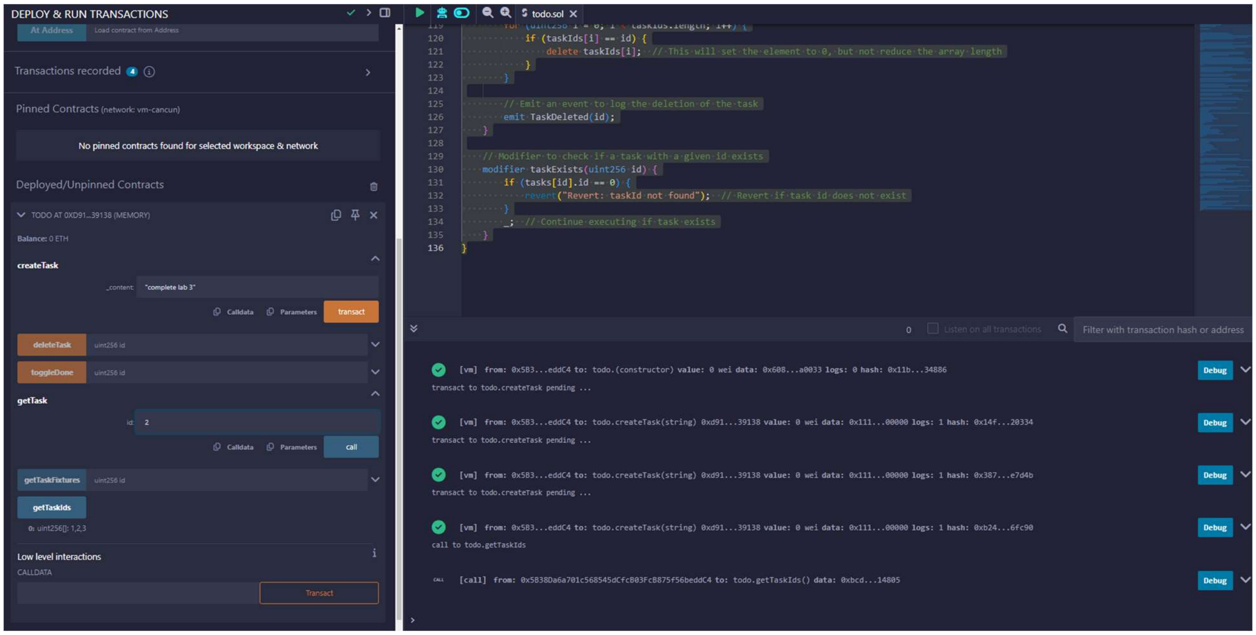
Output:


DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 4
Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
TODO AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
createTask
_content:
Calldata
Parameters
transact
deleteTask
uint256 id
toggleDone
uint256 id
getTask
uint256 id
getTaskFixtures
uint256 id
getTaskIds
Low level interactions
CALLDATA
Transact
todo.sol
0
Listen on all transactions
Filter with transaction hash or address
Debug

_content: "complete lab 3"

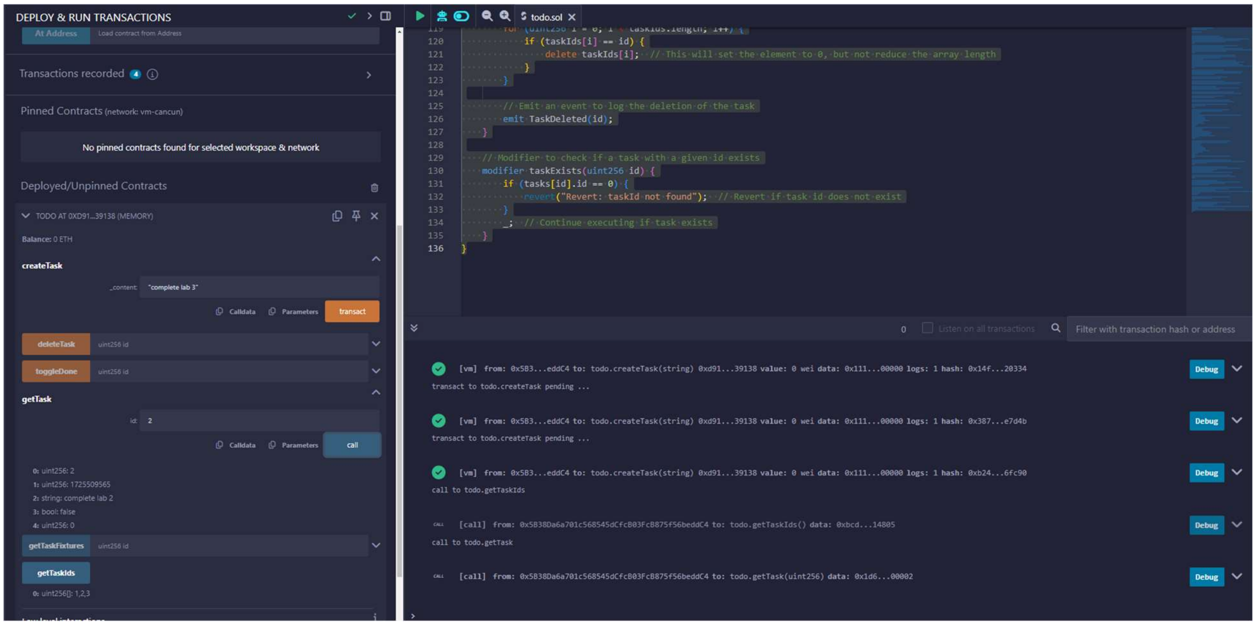
```
            for (uint256 i = 0; i < taskIds.length; i++) {
                if (taskIds[i] == id) {
                    delete taskIds[i];  // This will set the element to 0, but not reduce the array length
                }
            }

            // Emit an event to log the deletion of the task
            emit TaskDeleted(id);
        }

        // Modifier to check if a task with a given id exists
        modifier taskExists(uint256 id) {
            if (tasks[id].id == 0) {
                revert("Revert: taskId not found");  // Revert if task id does not exist
            }
            _;  // Continue executing if task exists
        }
    }
```

```
• sol-gpt <your Solidity question here>

Type the library name to see available commands.
Solidity copilot not activated!
creation of todo pending...

[vm] from: 0x5B3...eddC4 to: todo.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x11b...34886
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0x14f...20334
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0x387...e7d4b
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0xb24...6fc90
```


DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 4
Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
TODO AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
createTask
_content: "complete lab 3"
Calldata
Parameters
transact
deleteTask
uint256 id
toggleDone
uint256 id
getTask
id: 2
Calldata
Parameters
call
getTaskFixtures
uint256 id
getTaskIds
0: uint256[]: 1,2,3
Low level interactions
CALLDATA
Transact
todo.sol
0
Listen on all transactions
Filter with transaction hash or address
Debug

```
        for (uint256 i = 0; i < taskIds.length; i++) {
            if (taskIds[i] == id) {
                delete taskIds[i];  // This will set the element to 0, but not reduce the array length
            }
        }

        // Emit an event to log the deletion of the task
        emit TaskDeleted(id);
    }

    // Modifier to check if a task with a given id exists
    modifier taskExists(uint256 id) {
        if (tasks[id].id == 0) {
            revert("Revert: taskId not found");  // Revert if task id does not exist
        }
        _;  // Continue executing if task exists
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: todo.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x11b...34886
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0x14f...20334
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0x387...e7d4b
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0xb24...6fc90
call to todo.getTaskIds

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: todo.getTaskIds() data: 0xbcd...14805
```

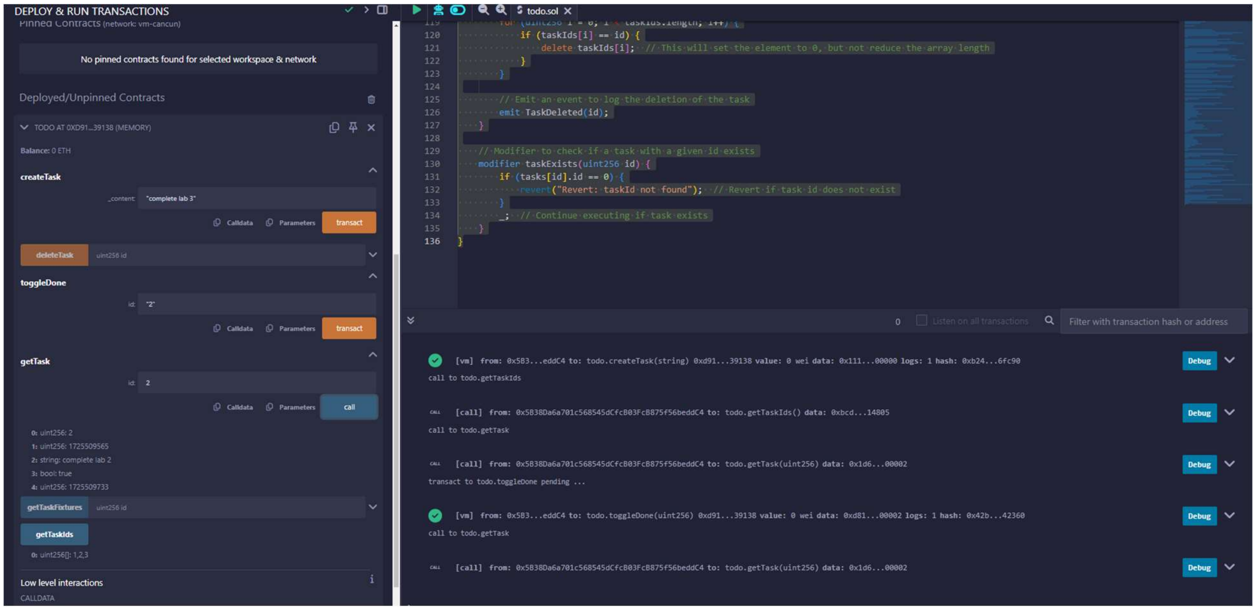

DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 4
Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
TODO AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
createTask
_content: "complete lab 3"
Calldata
Parameters
transact
deleteTask
uint256 id
toggleDone
uint256 id
getTask
id: 2
Calldata
Parameters
call
getTaskFixtures
uint256 id
getTaskIds
todo.sol
0
Listen on all transactions
Filter with transaction hash or address
Debug

```
0: uint256: 2
1: uint256: 1725509565
2: string: complete lab 2
3: bool: false
4: uint256: 0
```

```
0: uint256[]: 1,2,3
```

```
        for (uint256 i = 0; i < taskIds.length; i++) {
            if (taskIds[i] == id) {
                delete taskIds[i];  // This will set the element to 0, but not reduce the array length
            }
        }

        // Emit an event to log the deletion of the task
        emit TaskDeleted(id);
    }

    // Modifier to check if a task with a given id exists
    modifier taskExists(uint256 id) {
        if (tasks[id].id == 0) {
            revert("Revert: taskId not found");  // Revert if task id does not exist
        }
        _;  // Continue executing if task exists
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0x14f...20334
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0x387...e7d4b
transact to todo.createTask pending ...

[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0xb24...6fc90
call to todo.getTaskIds

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: todo.getTaskIds() data: 0xbcd...14805
call to todo.getTask

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: todo.getTask(uint256) data: 0x1d6...00002
```


DEPLOY & RUN TRANSACTIONS
Pinned Contracts (network: vm-cancun)
Deployed/Unpinned Contracts
Calldata
Parameters
transact
deleteTask
uint256 id
call
getTaskFixtures
getTaskIds
Low level interactions
CALLDATA
todo.sol
Listen on all transactions
Filter with transaction hash or address
Debug

No pinned contracts found for selected workspace & network

TODO AT 0XD91...39138 (MEMORY)

Balance: 0 ETH

createTask

_content: "complete lab 3"

toggleDone

id: "2"

getTask

id: 2

```
0: uint256: 2
1: uint256: 1725509565
2: string: complete lab 2
3: bool: true
4: uint256: 1725509733
```

getTaskIds

```
0: uint256[]: 1,2,3
```

```
        for (uint256 i = 0; i < taskIds.length; i++) {
            if (taskIds[i] == id) {
                delete taskIds[i];  // This will set the element to 0, but not reduce the array length
            }
        }

        // Emit an event to log the deletion of the task
        emit TaskDeleted(id);
    }

    // Modifier to check if a task with a given id exists
    modifier taskExists(uint256 id) {
        if (tasks[id].id == 0) {
            revert("Revert: taskId not found");  // Revert if task id does not exist
        }
        _;  // Continue executing if task exists
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: todo.createTask(string) 0xd91...39138 value: 0 wei data: 0x111...00000 logs: 1 hash: 0xb24...6fc90
call to todo.getTaskIds

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: todo.getTaskIds() data: 0xbcd...14805
call to todo.getTask

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: todo.getTask(uint256) data: 0x1d6...00002
transact to todo.toggleDone pending ...

[vm] from: 0x5B3...eddC4 to: todo.toggleDone(uint256) 0xd91...39138 value: 0 wei data: 0xd81...00002 logs: 1 hash: 0x42b...42360
call to todo.getTask

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: todo.getTask(uint256) data: 0x1d6...00002
```