Documentation

Overall Purpose of the Project

To build a full-stack task execution system that:

- Allows users to **create, manage, and run tasks** (defined by shell commands).
- **Tracks** each task's execution history (start time, end time, output).
- Provides a **user-friendly frontend** to interact with tasks.
- Uses a **MongoDB backend** to store task definitions and execution logs.
- Automates command-line operations via a web interface, enabling technical users (like DevOps or engineers) to **execute scripts without needing terminal access**.

Functional Goals by Layer

Frontend (React + Ant Design)

- UI for creating, listing, and deleting tasks.
- Search feature to find tasks by name.
- "Run" button to execute a task.
- Modal popup to view execution logs.
- Makes HTTP requests to the backend using `axios`.

Backend (Spring Boot + MongoDB)

- REST API to handle CRUD operations for tasks.
- Executes shell commands when triggered.
- Stores the output, start/end time in MongoDB.
- Handles CORS to support frontend requests.
- Uses TaskController, Task, and TaskExecution models for logical separation.

Real-World Use Cases

- **DevOps Automation:** Run deployment, backup, or cleanup scripts from a web interface.

- **Admin Tooling:** Give operations teams the ability to execute predefined tasks without needing direct terminal access.
- **Remote Execution Dashboard:** Useful in cases where tasks must be managed or run remotely (e.g., servers, CI/CD scripts).
- **Education or Demonstration:** Useful for teaching how command-line tasks can be integrated into modern web systems.

Task-1

Backend Configuration

Task Management Backend – Overview

Purpose

The backend provides a RESTful API for managing shell-based tasks. It allows users to:

- Create and store task definitions
- Execute shell commands via tasks
- Track execution history
- Retrieve, search, and delete tasks

Architecture

- **Framework**: Spring Boot (Java)
- **Type**: RESTful Web Service
- **Data Access**: Spring Data JPA
- **Database**: MongoDB
- **Execution Logic**: Shell command execution tracked with timestamps

Core Components

1. Model: Task

Represents a task with:

- id (String): Unique identifier
- name (String): Task name
- owner (String): Creator
- command (String): Shell command to be executed
- taskExecutions (List): History of runs

2. Model: TaskExecution

Represents a single execution of a task:

- id: UUID
- startTime: Execution start timestamp
- endTime: Execution end timestamp
- output: Output of the shell command

3. Controller: TaskController

Handles REST API endpoints:

- GET /tasks – List all tasks
- GET /tasks/{id} – Get task by ID
- GET /tasks/search?name= – Search by name
- PUT /tasks – Create/update task
- DELETE /tasks/{id} – Delete task
- PUT /tasks/{id}/execute – Run task and log execution

4. Repository: TaskRepository

Handles data persistence using JPA:

- Standard CRUD methods
- Custom search by name using:
  List<Task> findByNameContainingIgnoreCase(String name);

CORS Handling

- @CrossOrigin(origins = "http://localhost:3000") allows requests from the React frontend running on a different port.

Execution Workflow

1. **User creates a task** with command and metadata.
2. **Task is saved** to the database.
3. On execution:
   - Backend runs the shell command using Java.
   - Captures start time, end time, and output.
   - Saves a new `TaskExecution` entry under the task.

✅ Features Summary

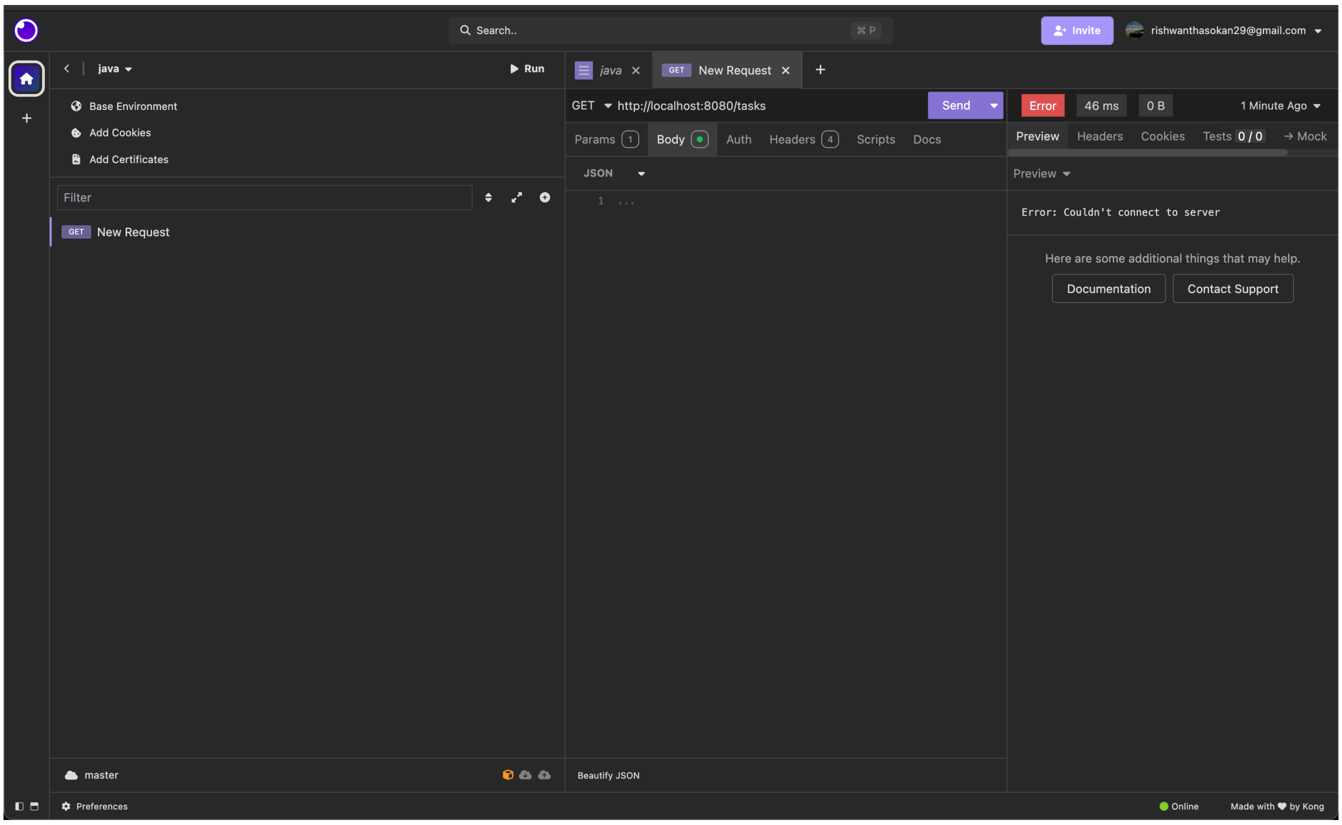
Feature	Status
Create Task	✅
Search Task by Name	✅
Delete Task	✅
Execute Task Command	✅
Track Execution History	✅
REST API w/ JSON	✅
CORS Support	✅

Example JSON Response

```
{
 "id": "task01",
 "name": "Backup",
 "owner": "Admin",
 "command": "tar -czf backup.tar.gz /home/data",
 "taskExecutions": [
   {
    "startTime": "2025-06-11T13:30:00",
    "endTime": "2025-06-11T13:30:05",
    "output": "Backup completed"
   }
 ]
}
```

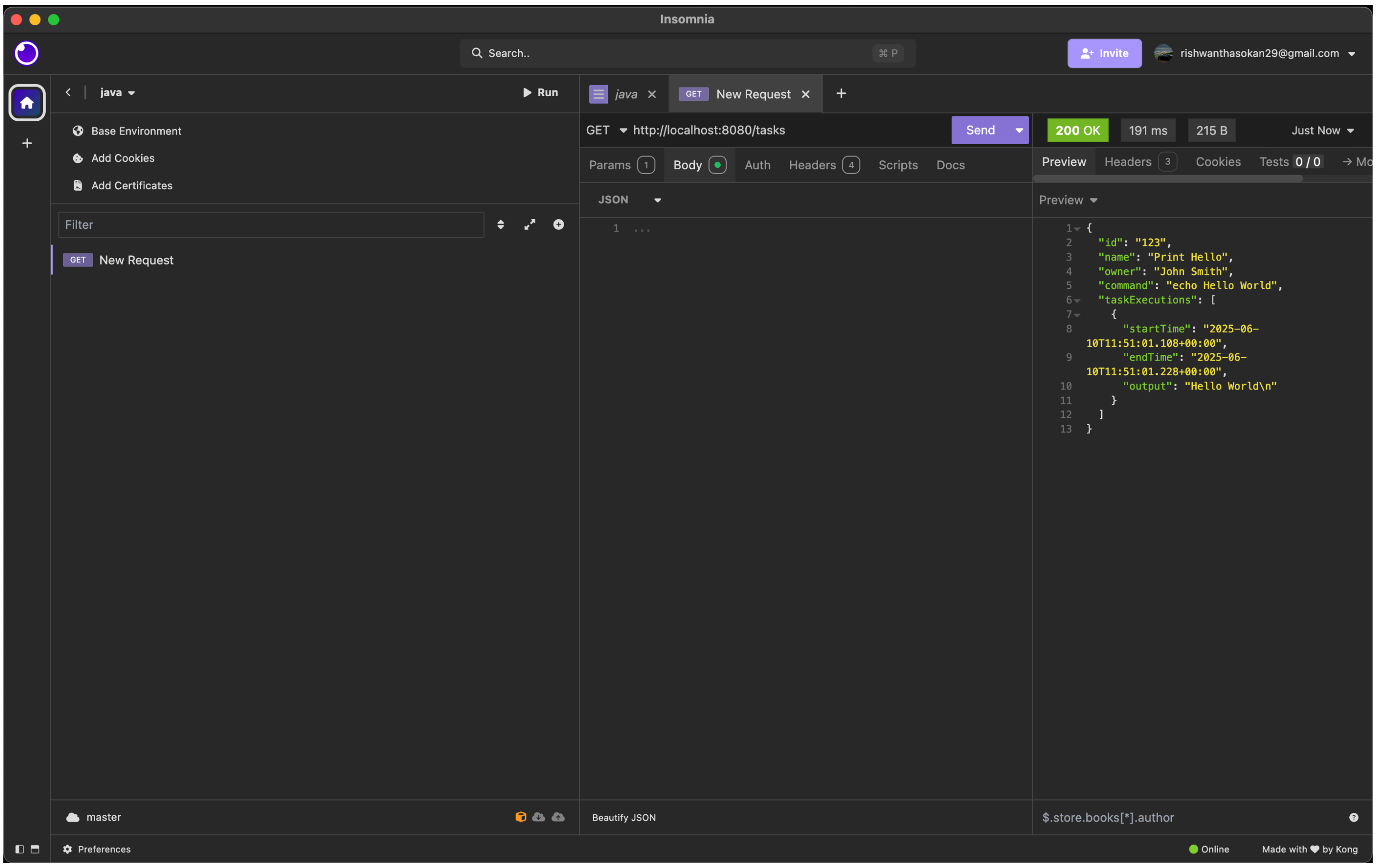
Api test Images - Insomnia

1, **Api**: http://localhost:8080/tasks
Method: Get
Description: Before Clicking Send Button



```
Error: Couldn't connect to server
```

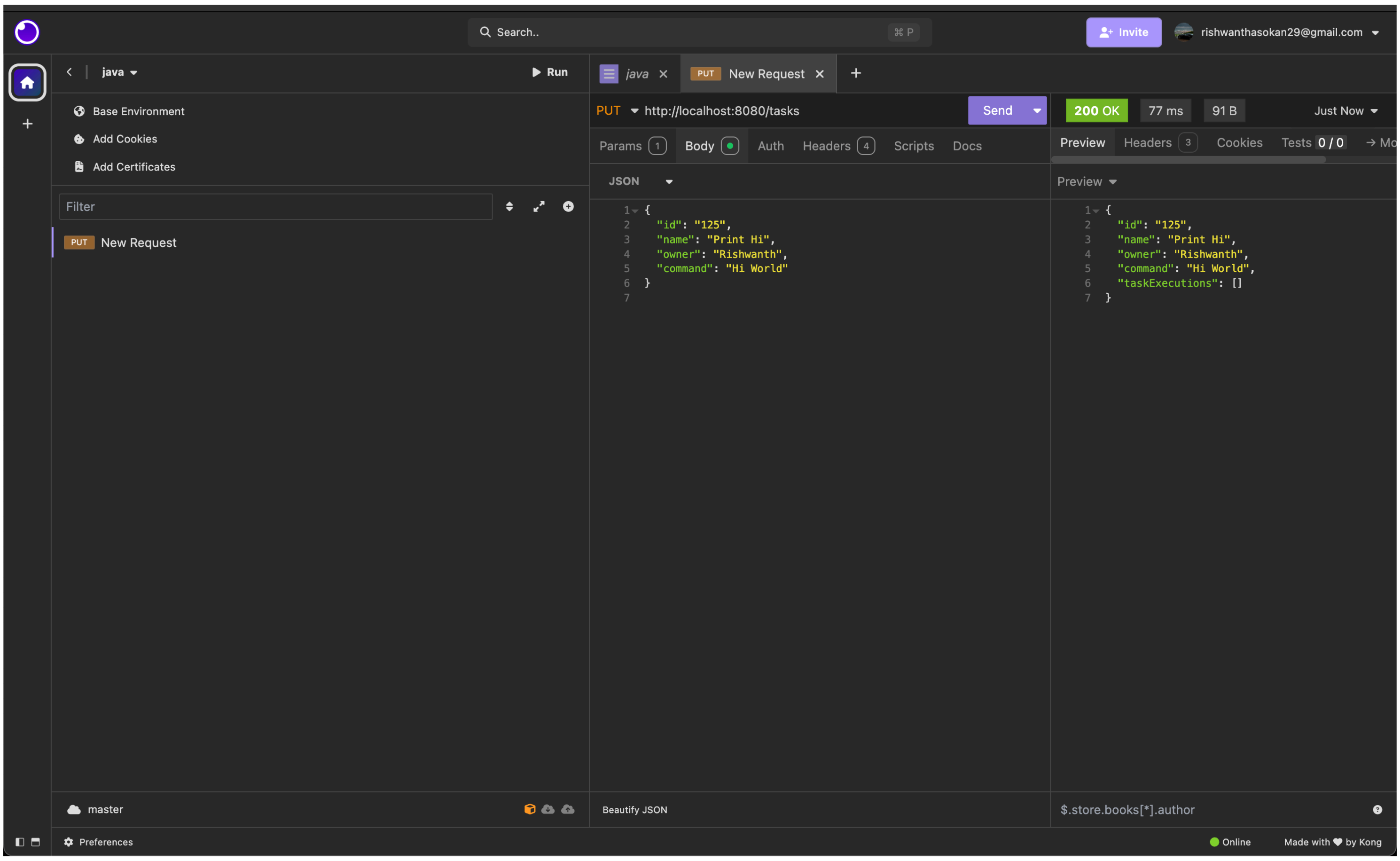
Description: After Clicking Send Button



```
GET http://localhost:8080/tasks
```

```
{
  "id": "123",
  "name": "Print Hello",
  "owner": "John Smith",
  "command": "echo Hello World",
  "taskExecutions": [
    {
      "startTime": "2025-06-
10T11:51:01.108+00:00",
      "endTime": "2025-06-
10T11:51:01.228+00:00",
      "output": "Hello World\n"
    }
  ]
}
```

2, **Api**: http://localhost:8080/tasks
Method: Put
Description: Creating new Data

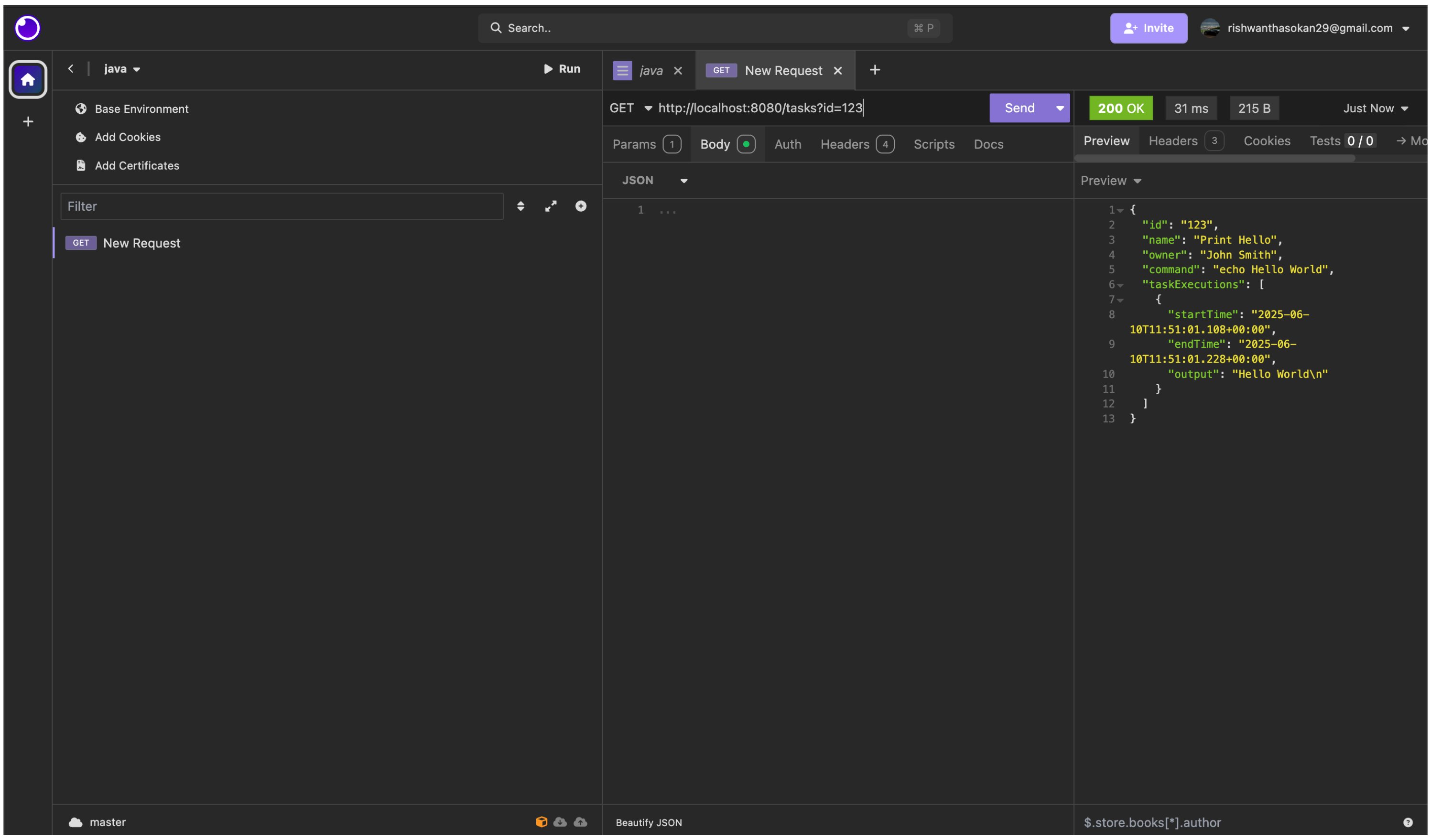


PUT http://localhost:8080/tasks

```
{
  "id": "125",
  "name": "Print Hi",
  "owner": "Rishwanth",
  "command": "Hi World"
}
```

```
{
  "id": "125",
  "name": "Print Hi",
  "owner": "Rishwanth",
  "command": "Hi World",
  "taskExecutions": []
}
```

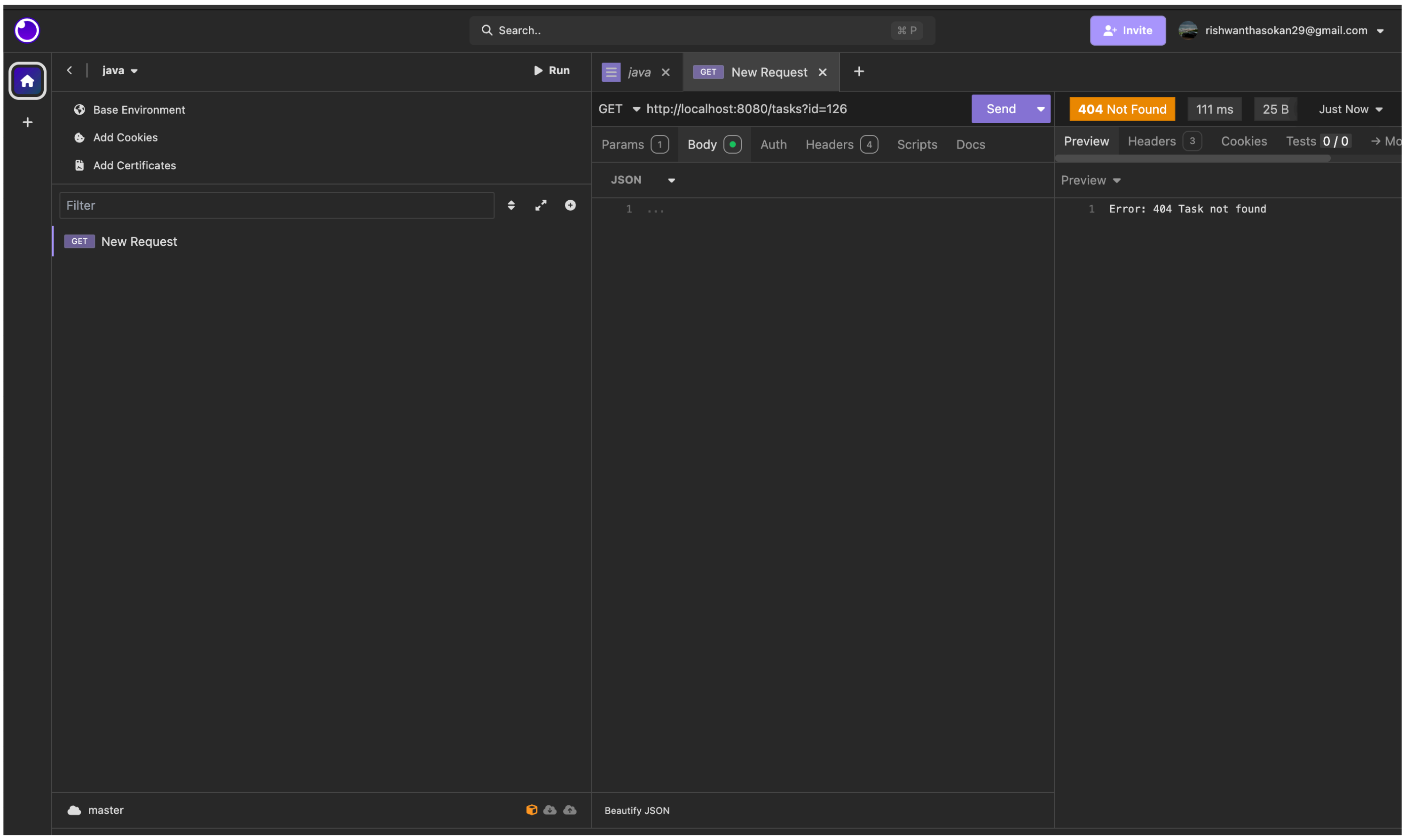
3, **Api**: http://localhost:8080/tasks?id=123
Method: Get
Description: Getting task by id



```
GET http://localhost:8080/tasks?id=123
```

```
{
  "id": "123",
  "name": "Print Hello",
  "owner": "John Smith",
  "command": "echo Hello World",
  "taskExecutions": [
    {
      "startTime": "2025-06-10T11:51:01.108+00:00",
      "endTime": "2025-06-10T11:51:01.228+00:00",
      "output": "Hello World\n"
    }
  ]
}
```

4, **Api**: http://localhost:8080/tasks?id=126
Method: Get
Description: Getting task by id, if no task found through error message



GET http://localhost:8080/tasks?id=126

404 Not Found 111 ms 25 B

```
Error: 404 Task not found
```

Task-3

Frontend

Task Management API Documentation

Base URL

http://localhost:8080/tasks

Endpoints

1. Get All Tasks

GET /tasks

Description: Retrieve a list of all tasks.

Response:

```
[
  {
    "id": "task1",
    "name": "Build Project",
    "owner": "Alice",
    "command": "mvn clean install",
    "taskExecutions": []
  }
]
```

2. Get Task by ID

GET /tasks/{id}

Description: Fetch a specific task by its ID.

Example Request:
GET /tasks/task1

Response:

```
{
  "id": "task1",
  "name": "Build Project",
  "owner": "Alice",
  "command": "mvn clean install",
  "taskExecutions": []
}
```

3. Search Tasks by Name

GET /tasks/search?name={query}

Description: Search for tasks containing the given query in their name.

Example Request:
GET /tasks/search?name=Build

Response:

```
[
  {
    "id": "task1",
    "name": "Build Project",
    "owner": "Alice",
    "command": "mvn clean install"
  }
]
```

4. Create or Update Task

PUT /tasks

Description: Create a new task or update an existing task.

Request Body:

```
{
  "id": "task2",
  "name": "Run Tests",
  "owner": "Bob",
  "command": "npm test"
}
```

Response:

```
{
  "id": "task2",
  "name": "Run Tests",
  "owner": "Bob",
  "command": "npm test"
}
```

! Ensure that the controller method is using @PutMapping and accepts @RequestBody.

5. Delete Task

DELETE /tasks/{id}

Description: Delete a task by its ID.

Example Request:
DELETE /tasks/task2

Response:

```
{
  "message": "Task deleted successfully"
}
```

6. Execute Task

PUT /tasks/{id}/execute

Description: Triggers the execution of a task. Stores execution history with timestamp and output.

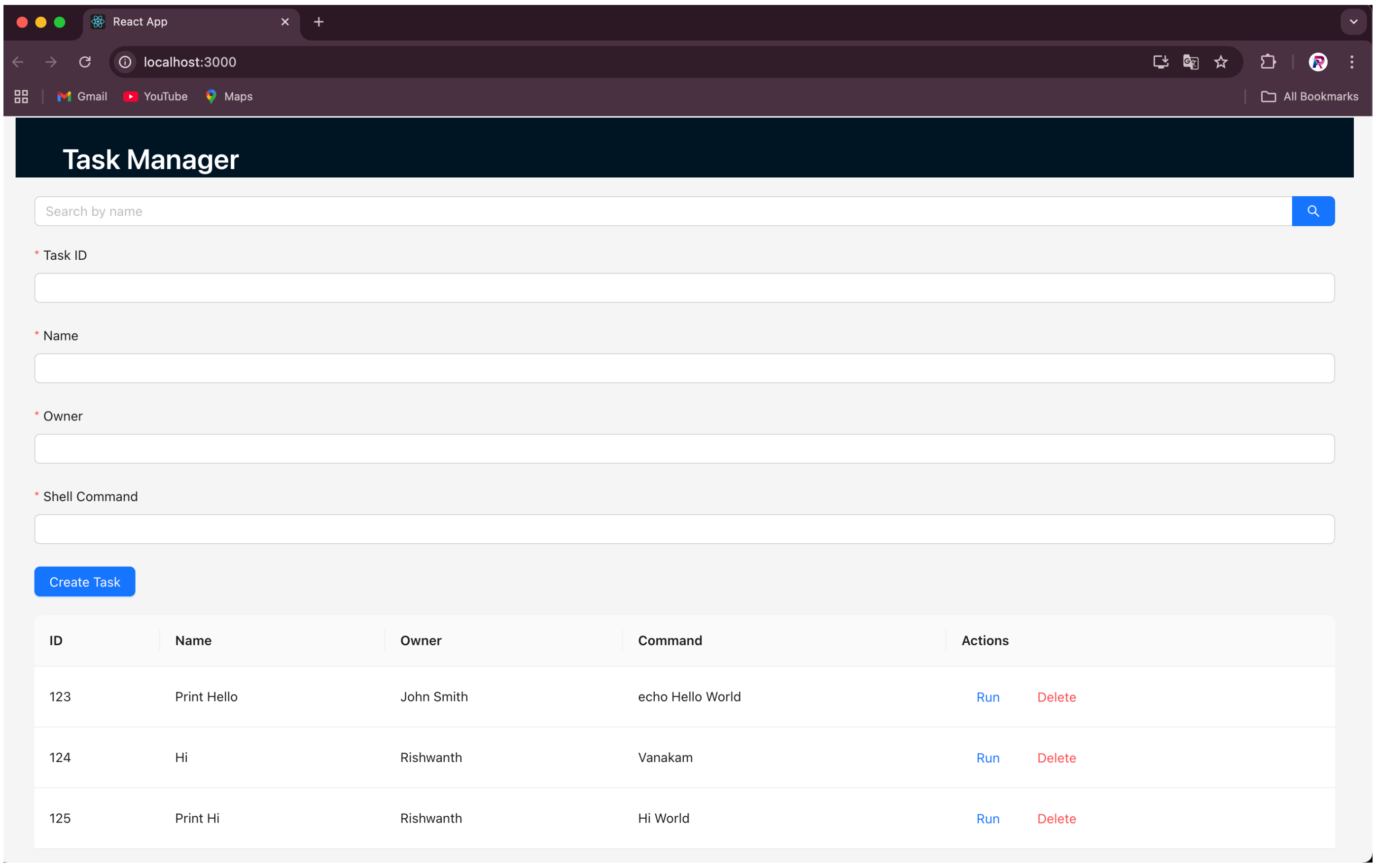
Example Request:
PUT /tasks/task1/execute

Response:

```
{
  "startTime": "2025-06-11T13:34:00",
  "endTime": "2025-06-11T13:34:02",
  "output": "BUILD SUCCESS"
}
```

Images:

1. Web page



Task Manager

Search by name

* Task ID

* Name

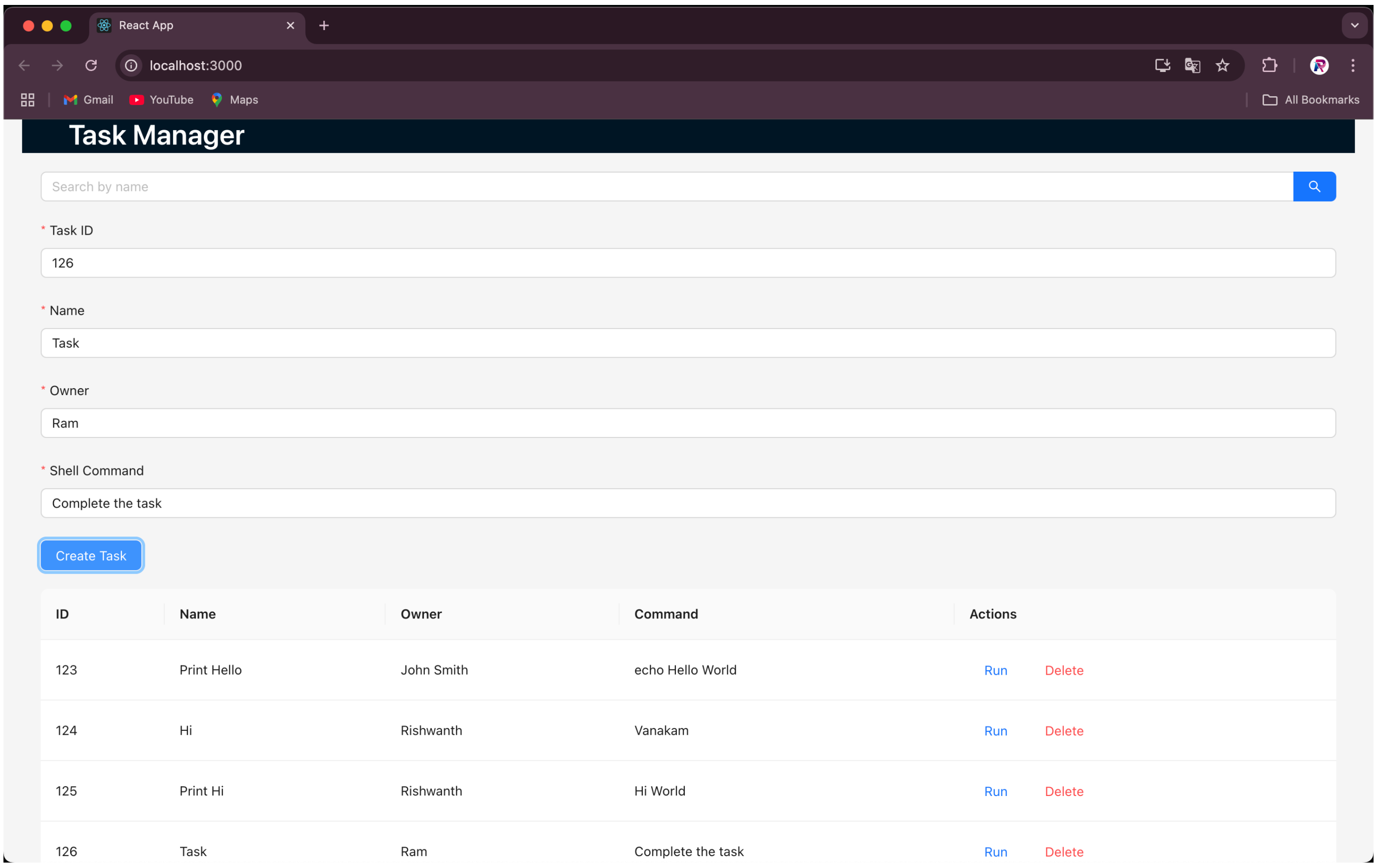
* Owner

* Shell Command

Create Task

ID	Name	Owner	Command	Actions
123	Print Hello	John Smith	echo Hello World	Run Delete
124	Hi	Rishwanth	Vanakam	Run Delete
125	Print Hi	Rishwanth	Hi World	Run Delete

2. Creating a new task



Task Manager

Search by name

* Task ID

126

* Name

Task

* Owner

Ram

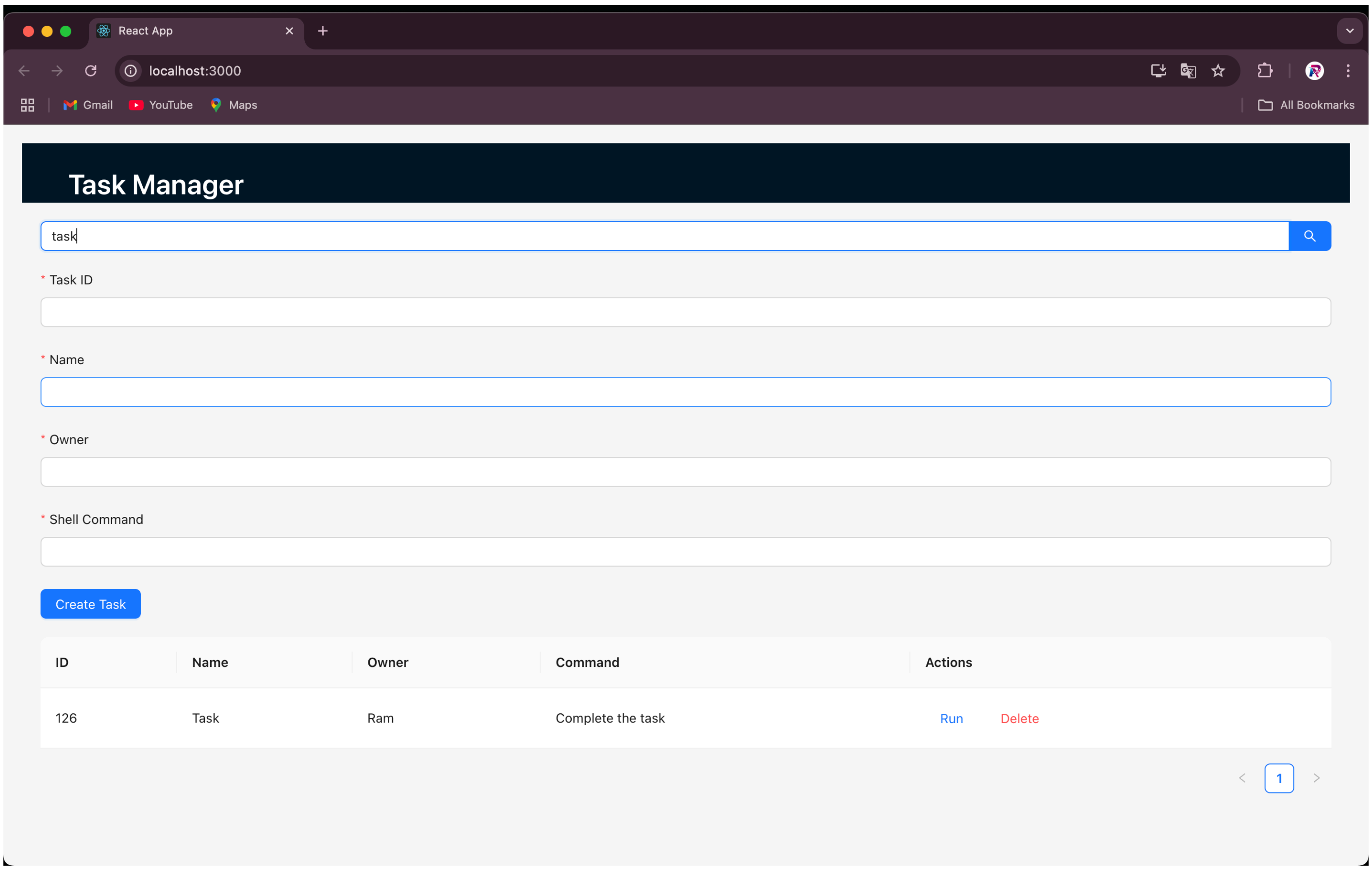
* Shell Command

Complete the task

Create Task

ID	Name	Owner	Command	Actions
123	Print Hello	John Smith	echo Hello World	Run Delete
124	Hi	Rishwanth	Vanakam	Run Delete
125	Print Hi	Rishwanth	Hi World	Run Delete
126	Task	Ram	Complete the task	Run Delete

3. Searching for a Task by name



Task Manager

task

* Task ID

* Name

* Owner

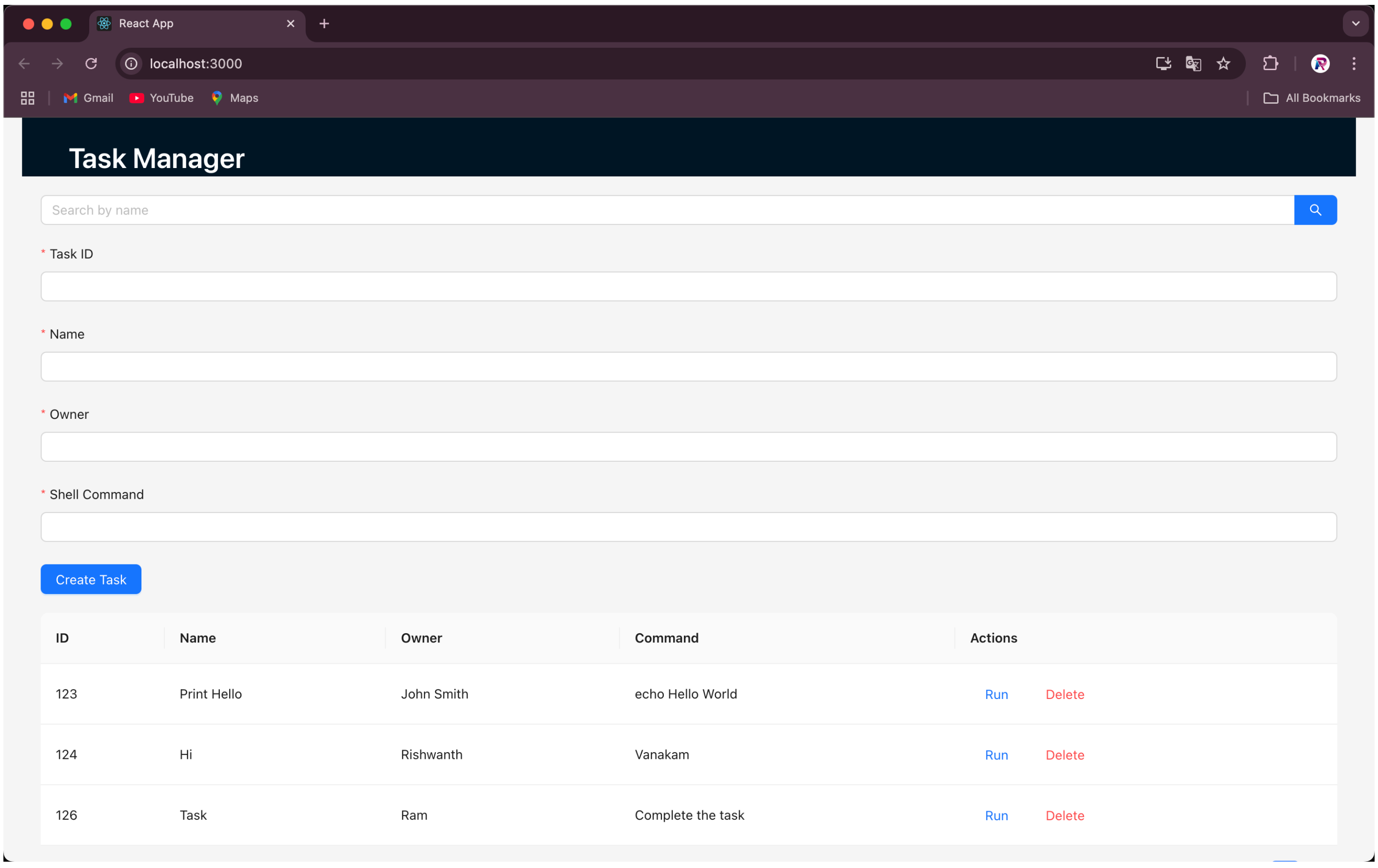
* Shell Command

Create Task

ID	Name	Owner	Command	Actions
126	Task	Ram	Complete the task	Run Delete

1

4. Deleting a Task Id- 125



Task Manager

Search by name

* Task ID

* Name

* Owner

* Shell Command

Create Task

ID	Name	Owner	Command	Actions
123	Print Hello	John Smith	echo Hello World	Run Delete
124	Hi	Rishwanth	Vanakam	Run Delete
126	Task	Ram	Complete the task	Run Delete

NPM/Yarn Dependencies

1. **React and React DOM.**
   These are essential for any React project.
   ```
   npm install react react-dom
   ```

2. **Ant Design**
   is used for UI components (Form, Input, Layout, Table, etc.).
   ```
   npm install antd
   ```

3. **dayjs**
   Used for formatting date and time in the execution history.
   ```
   npm install dayjs
   ```

4. **Axios**
   Needed for API communication (`getTasks`, `createTask`, etc. seem to use Axios).
   ```
   npm install axios
   ```

Optional (if not already configured)

5. **React Scripts (for Create React App)**
   If you're using Create React App:
   ```
   npm install react-scripts
   ```

6. **TypeScript**
   If you’re using TypeScript instead of JS (not required for the file you uploaded):
   ```
   npm install typescript @types/react @types/react-dom
   ```