

Assignment 2

CSE-5306-004-DISTRIBUTED SYSTEMS

Project Assignment 2: gRPC-Backed Virtual Pet Adoption System

Group 20:

Srinivas Sudheer Reddy Buchipalli – 1002149811

Ali Ashan

Q1:

Node.js (Hello World)

Clone the repository to get the example code

```
$ git clone -b @grpc/grpc-js@1.9.0 --depth 1 --shallow-submodules  
https://github.com/grpc/grpc-node
```

Navigate to the node example

```
$ cd grpc-node/examples
```

Install the example's dependencies

```
$ npm install
```

Navigate to the dynamic codegen "hello, world" Node example:

```
$ cd helloworld/dynamic_codegen
```


The image shows a Visual Studio Code editor window. The left sidebar contains a file explorer with files like README.md, run-tests.bat, and run-tests.sh. The main editor area is split into two panes. The top pane shows a terminal window with the following commands and output:

```
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node> cd .\examples\  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples> cd .\helloworld\dynamic_codegen\  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ec  
ho "$(Get-Date) $(whoami)"; node greeter_client.js  
10/02/2024 22:47:24 srinivas\bssud  
Greeting: Hello world  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen>
```

The bottom pane shows the file explorer with files like README.md, run-tests.bat, and run-tests.sh. The status bar at the bottom indicates the current file is master, with 52 lines and 58 columns, using UTF-8 encoding and CRLF line endings.

The image shows a Visual Studio Code editor window. The left sidebar contains a file explorer with files like gulpfile.ts, LICENSE, MAINTAINERS.md, merge_kokoro_logs.js, PACKAGE-COMPARISO..., package.json, README.md, run-tests.bat, run-tests.sh, SECURITY.md, setup_interop_purejs.sh, setup_interop.sh, setup.sh, TROUBLESHOOTING.md, tsconfig.json, and util.js. The main editor area is split into two panes. The top pane shows a terminal window with the following commands and output:

```
7 packages are looking for funding  
run `npm fund` for details  
  
found 0 vulnerabilities  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples> cd helloworld\dynamic_codegen^C  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ^C  
  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ec  
ho "$(Get-Date) $(whoami)"; node greeter_server.js  
10/02/2024 22:44:12 srinivas\bssud  
gRPC listening on 50051  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ^C  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ec  
ho "$(Get-Date) $(whoami)"; node greeter_server.js  
10/02/2024 23:59:46 srinivas\bssud  
(node:31028) DeprecationWarning: Calling start() is no longer necessary. It can be safely omitted.  
(Use `node --trace-deprecation ...` to show where the warning was created)  
[]
```

The bottom pane shows the file explorer with files like gulpfile.ts, LICENSE, MAINTAINERS.md, merge_kokoro_logs.js, PACKAGE-COMPARISO..., package.json, README.md, run-tests.bat, run-tests.sh, SECURITY.md, setup_interop_purejs.sh, setup_interop.sh, setup.sh, TROUBLESHOOTING.md, tsconfig.json, and util.js. The status bar at the bottom indicates the current file is master, with 52 lines and 58 columns, using UTF-8 encoding and CRLF line endings.

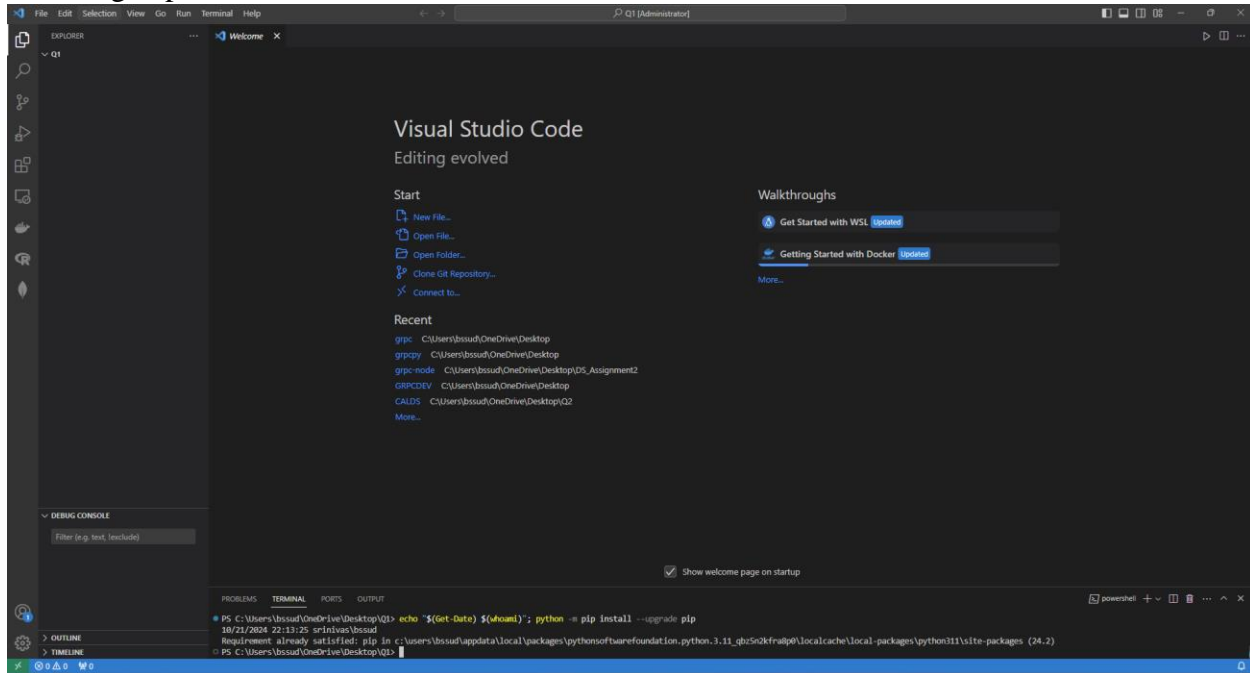
The image shows a Visual Studio Code editor window. The left sidebar contains a file explorer with files like README.md, run-tests.bat, and run-tests.sh. The main editor area is split into two panes. The top pane shows a terminal window with the following commands and output:

```
}  
}  
Greeting: Hello world  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ^C  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> ec  
ho "$(Get-Date) $(whoami)"; node greeter_client.js  
10/02/2024 23:59:53 srinivas\bssud  
Greeting: Hello world  
Greeting: Hello again, you  
PS C:\Users\bssud\OneDrive\Desktop\DS_Assignment2\grpc-node\examples\helloworld\dynamic_codegen> []
```

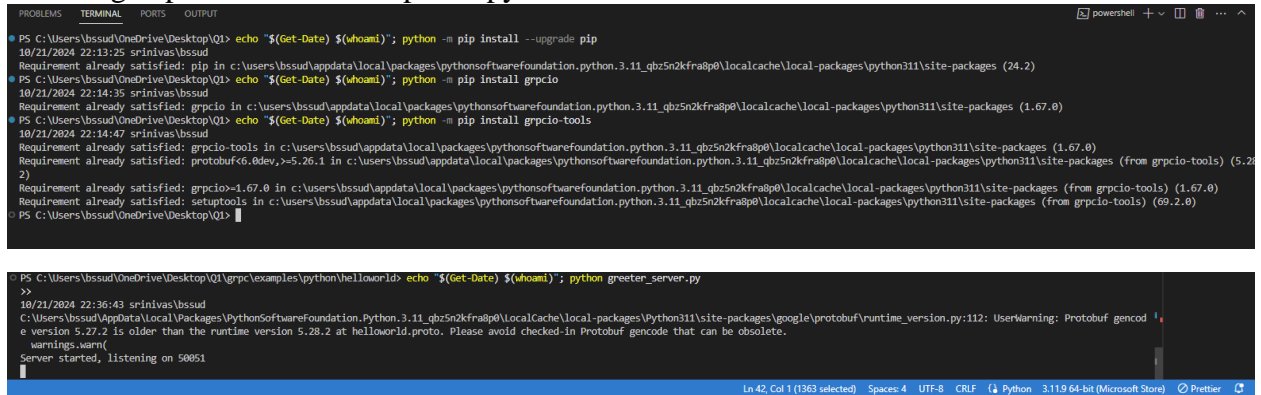
The bottom pane shows the file explorer with files like README.md, run-tests.bat, and run-tests.sh. The status bar at the bottom indicates the current file is master, with 52 lines and 58 columns, using UTF-8 encoding and CRLF line endings.

Part 2 (Q1) : Using Python:

Installing dependencies



Installing dependencies and Grpc for python:



The screenshot displays a Windows IDE environment for developing a gRPC server. The Explorer pane on the left shows a project structure with files like `flow_control`, `health_checking`, `helloworld.proto`, and various `.py` files. The main editor shows the `helloworld.proto` file with its content and comments. The bottom pane shows the terminal output of the gRPC server running on port 50051, displaying the 'hello, hi!' message.

Post Updating the changes and Printing Hello World Again.

The screenshot displays a Windows development environment with the following components:

- File Explorer (Left):** Shows a project structure with files like `flow_control.py`, `health_checking.py`, `helloworldstreamingworld.py`, `helloworld.py`, `pycache_`, `gRPCpython`, and various `async_*` files.
- Main Editor:** Displays the `helloworld.py` file, which defines a `Greeter` gRPC service. The service has two methods: `SayHello` and `SayHelloAgain`. The `SayHello` method takes a `request` and returns a `reply`. The `SayHelloAgain` method takes a `request` and returns a `reply`.
- Output Pane (Bottom):** Shows the command `python -m grpc_python_example --helloworld.proto` being executed. The output shows the server starting on port 50051 and the client sending a `SayHello` request.

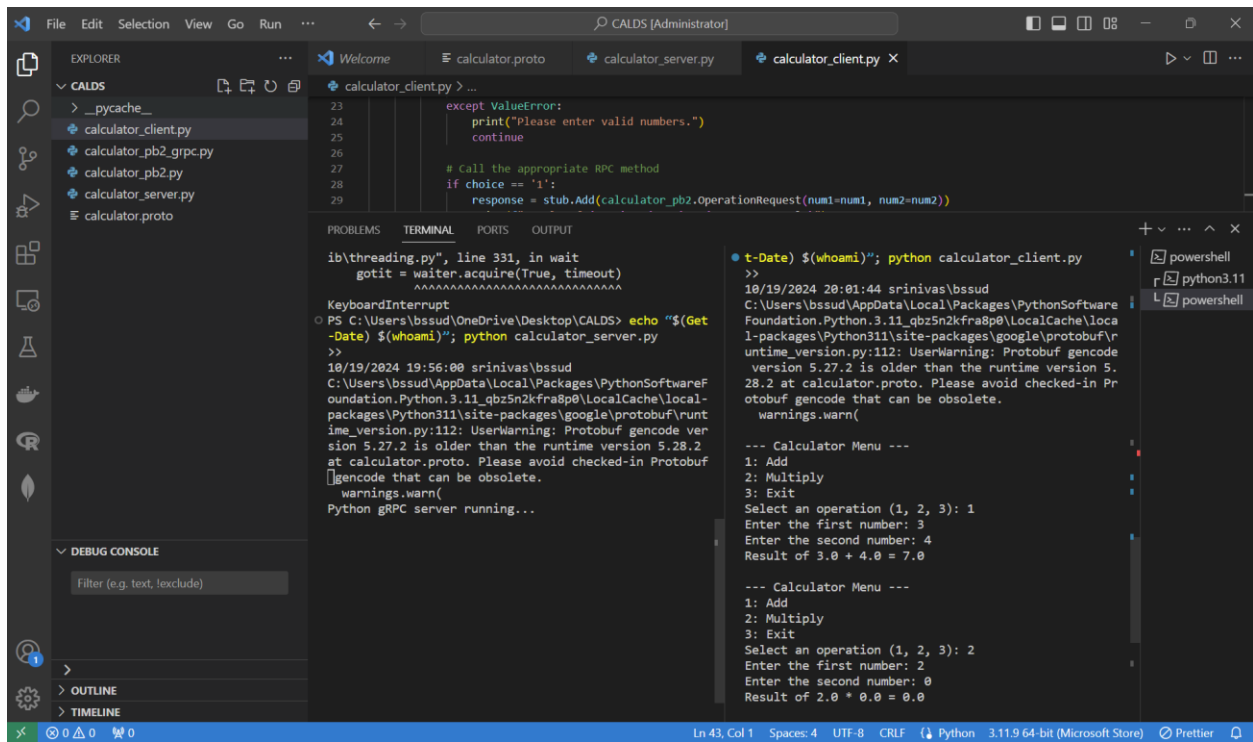
Q2:

A. Simple Calculator using Python

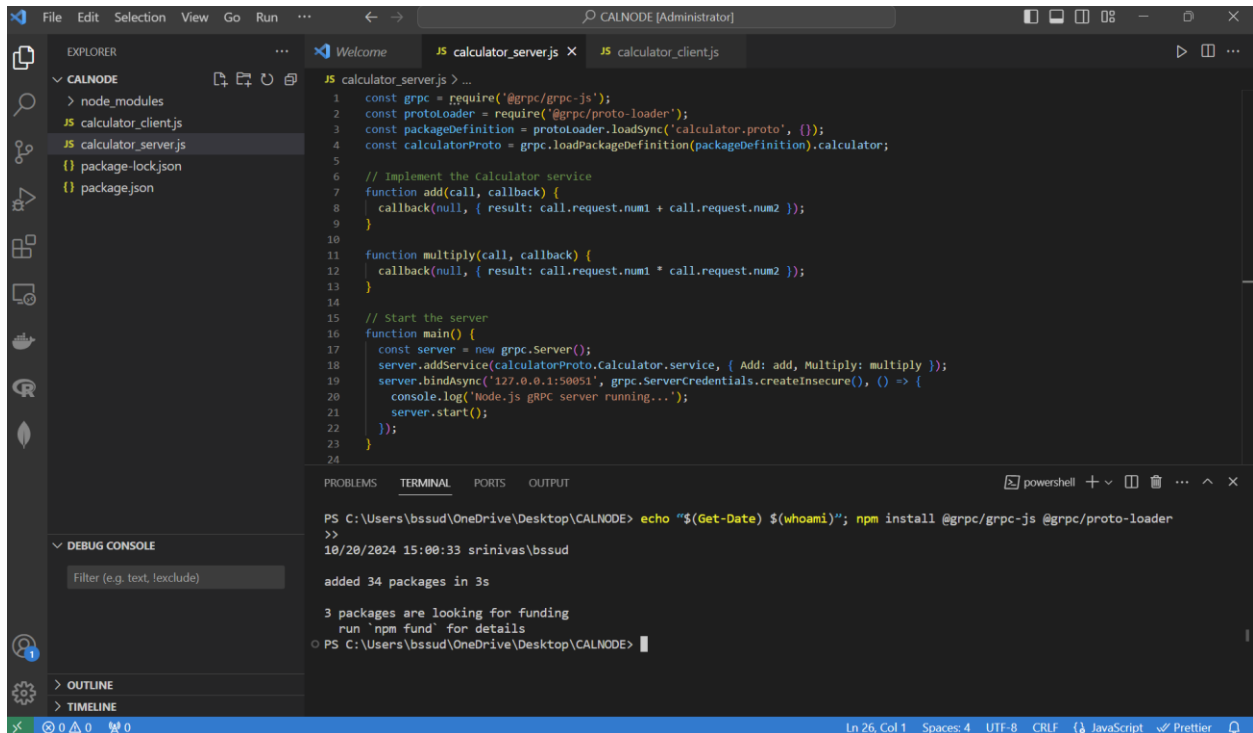
```
File Edit Selection View Go Run ... CALDS [Administrator]
EXPLORER
  CALDS
    calculator_client.py 2
    calculator_server.py 2
    calculator.pb2
    calculator.proto
calculator_client.py > ...
1 import grpc
2 import calculator_pb2
3 import calculator_pb2_grpc
4
5 # Connect to the gRPC server
6 channel = grpc.insecure_channel('localhost:50051')
7 stub = calculator_pb2_grpc.CalculatorStub(channel)
8
9 # Call Add RPC
10 response = stub.Add(calculator_pb2.OperationRequest(num1=5, num2=3))
11 print("Add: 5 + 3 = ", response.result)
12
13 # Call Multiply RPC
14 response = stub.Multiply(calculator_pb2.OperationRequest(num1=5, num2=3))
15 print("Multiply: 5 * 3 = ", response.result)
16
TERMINAL
PS C:\Users\bssud\OneDrive\Desktop\CALDS> echo "$(Get-Date) $(whoami)"; pip install grpcio grpcio-tools
>>
10/19/2024 19:43:07 srinivas\bssud
Requirement already satisfied: grpcio in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (1.66.2)
Requirement already satisfied: grpcio-tools in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (1.66.2)
Requirement already satisfied: protobuf<6.0dev,>=5.26.1 in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from grpcio-tools) (5.28.2)
Requirement already satisfied: setuptools in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from grpcio-tools) (69.2.0)

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: C:\Users\bssud\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
PS C:\Users\bssud\OneDrive\Desktop\CALDS>
```

```
File Edit Selection View Go Run ... CALDS [Administrator]
EXPLORER
  CALDS
    calculator_client.py
    calculator_pb2_grpc.py
    calculator_pb2.py
    calculator_server.py
    calculator.proto
calculator_client.py > ...
1 import grpc
2 import calculator_pb2
3 import calculator_pb2_grpc
4
5 # Connect to the gRPC server
6 channel = grpc.insecure_channel('localhost:50051')
7 stub = calculator_pb2_grpc.CalculatorStub(channel)
8
9 # Call Add RPC
10 response = stub.Add(calculator_pb2.OperationRequest(num1=5, num2=3))
11 print("Add: 5 + 3 = ", response.result)
12
13 # Call Multiply RPC
14 response = stub.Multiply(calculator_pb2.OperationRequest(num1=5, num2=3))
15 print("Multiply: 5 * 3 = ", response.result)
16
TERMINAL
10/19/2024 19:49:05 srinivas\bssud
--proto_path passed empty directory name. (Use "." for current directory.)
PS C:\Users\bssud\OneDrive\Desktop\CALDS> echo "$(Get-Date) $(whoami)"; python -m grpc_tools.protoc -I=C:\Users\bssud\OneDrive\Desktop\CALDS --python_out=. --grpc_python_out=. calculator.proto
>>
10/19/2024 19:49:51 srinivas\bssud
PS C:\Users\bssud\OneDrive\Desktop\CALDS>
```



B. Node (Simple Calculator)




```
File Edit Selection View Go Run ... CALNODE [Administrator]
EXPLORER
  CALNODE
    > node_modules
    JS calculator_client.js
    JS calculator_server.js
    {} package-lock.json
    {} package.json
  DEBUG CONSOLE
    Filter (e.g. text, exclude)
  OUTLINE
  TIMELINE

JS calculator_server.js > ...
1  const grpc = require('@grpc/grpc-js');
2  const protoloader = require('@grpc/proto-loader');
3  const packageDefinition = protoloader.loadSync('calculator.proto', {});
4  const calculatorProto = grpc.loadPackageDefinition(packageDefinition).calculator;
5
6  // Implement the calculator service
7  function add(call, callback) {
8    callback(null, { result: call.request.num1 + call.request.num2 });
9  }
10
11 function multiply(call, callback) {
12   callback(null, { result: call.request.num1 * call.request.num2 });
13 }
14
15 // Start the server
16 function main() {
17   const server = new grpc.Server();
18   server.addService(calculatorProto.calculator.service, { Add: add, Multiply: multiply });
19   server.bindAsync('127.0.0.1:50051', grpc.ServerCredentials.createInsecure(), () => {
20     console.log('Node.js gRPC server running...');
21     server.start();
22   });
23 }
24

TERMINAL
3 packages are looking for funding
run 'npm fund' for details
PS C:\Users\bssud\OneDrive\Desktop\CALNODE> echo "$(Get-Date) $(whoami)"; npm i
10/20/2024 15:01:02 srinivas\bssud

up to date, audited 35 packages in 812ms
3 packages are looking for funding
run 'npm fund' for details
found 0 vulnerabilities
PS C:\Users\bssud\OneDrive\Desktop\CALNODE>
```

```
File Edit Selection View Go Run ... CALNODE [Administrator]
JS calculator_server.js calculator.proto JS calculator_client.js > ...
51
52
53
54
55
56
57
58
59
60
61
62
63
64

if (error) {
  console.log('Result of ${num1} * ${num2} = ${response.result}');
} else {
  console.error("Error: ", error.message);
}
askOperation(); // Continue prompting after result
});
});
} else if (choice === '3') {
  console.log("Exiting...");
  rl.close(); // Exit the program
} else {

TERMINAL
PS C:\Users\bssud\OneDrive\Desktop\CALNODE> echo "$(Get-Date) $(whoami)"; node calculator_server.js
10/20/2024 15:02:39 srinivas\bssud
Node.js gRPC server running...
(node:55716) DeprecationWarning: Calling start() is no longer necessary. It can be safely omitted.
(Use 'node --trace-deprecation ...' to show where the warning was created)

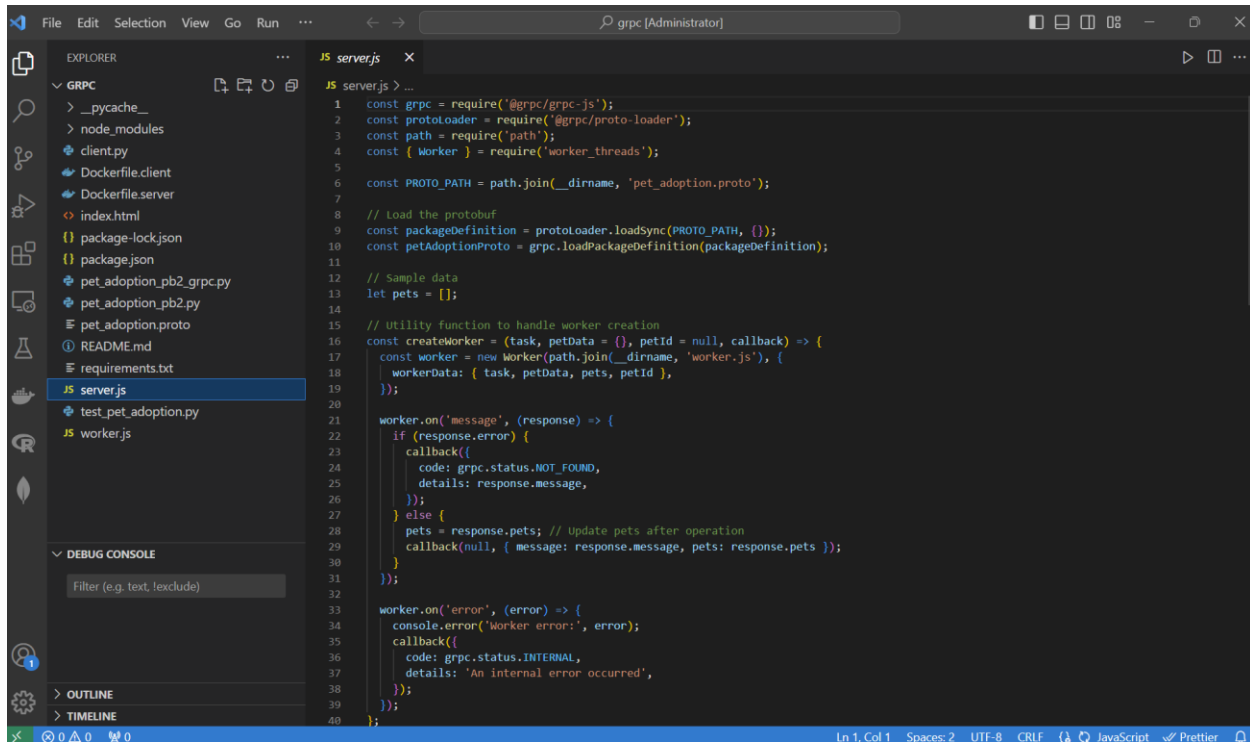
PS C:\Users\bssud\OneDrive\Desktop\CALNODE> ^C
PS C:\Users\bssud\OneDrive\Desktop\CALNODE> ^C
PS C:\Users\bssud\OneDrive\Desktop\CALNODE> echo "$(Get-Date) $(whoami)"; node calculator_client.js
10/20/2024 15:04:29 srinivas\bssud

--- Calculator Menu ---
1: Add
2: Multiply
3: Exit
Select an operation (1, 2, 3): 1
Enter the first number: 3
Enter the second number: 3
Result of 3 + 3 = 6

--- Calculator Menu ---
1: Add
2: Multiply
3: Exit
Select an operation (1, 2, 3):
```

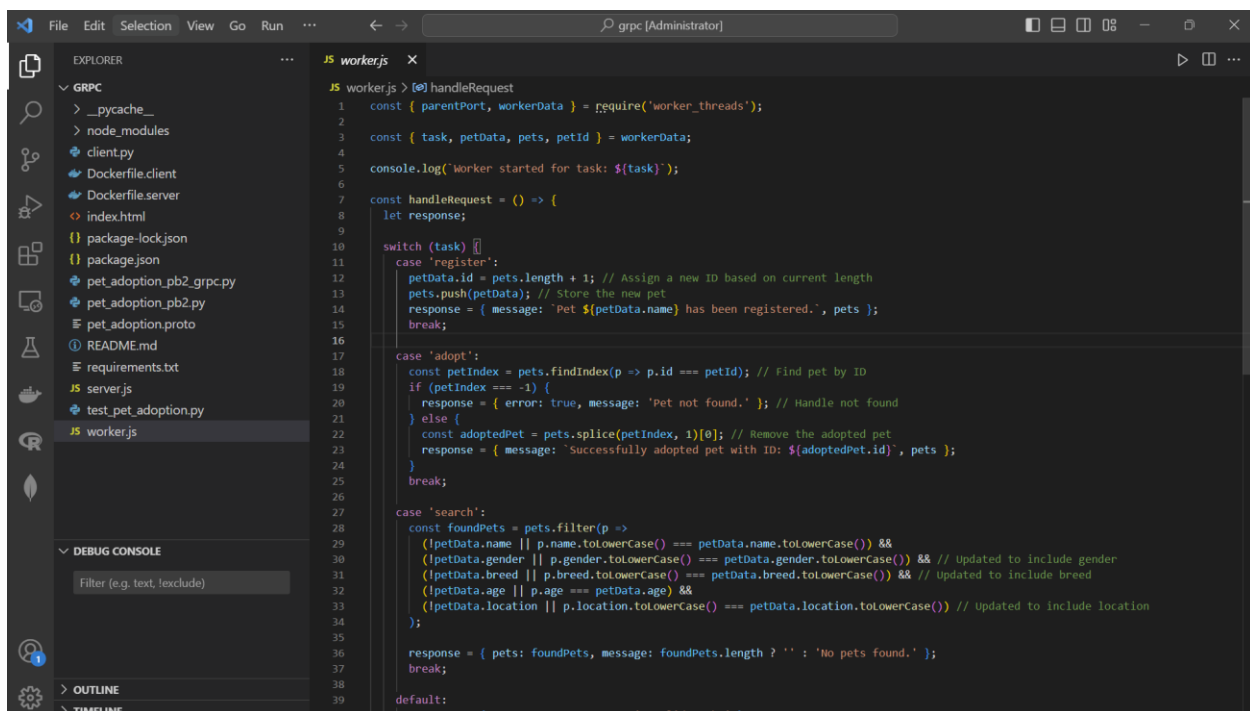

Q3:

Server code:



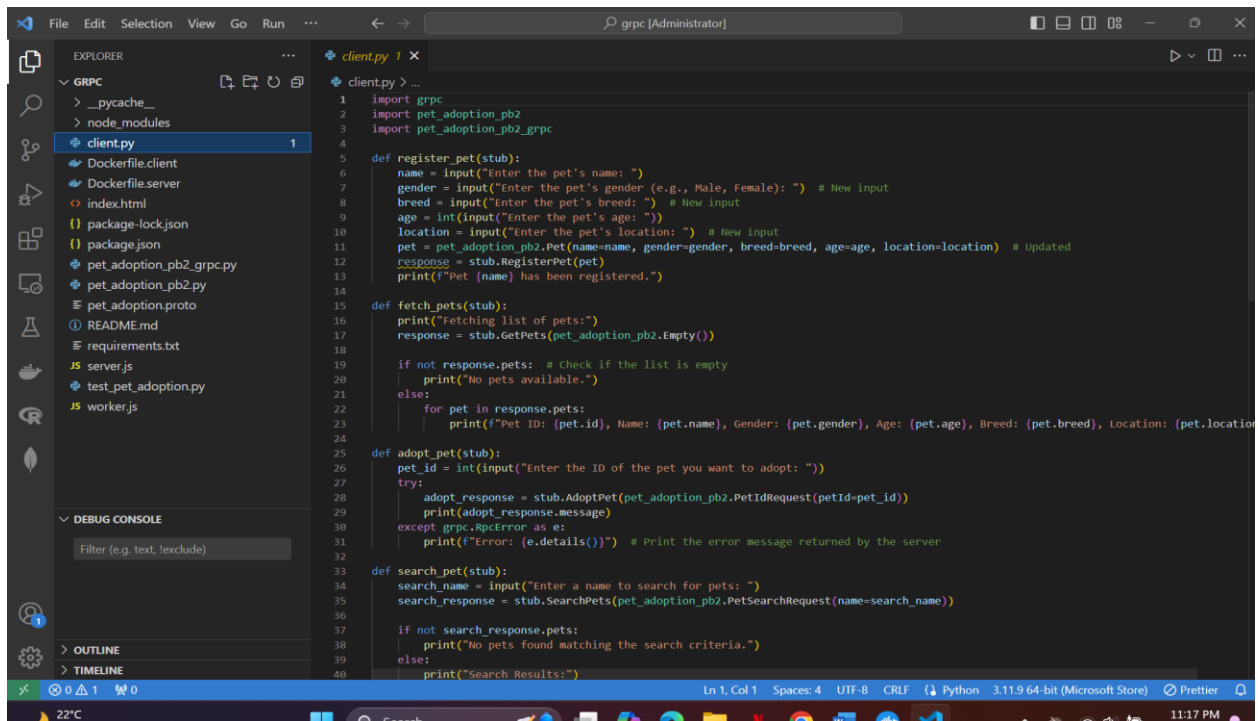
```
1 const grpc = require('@grpc/grpc-js');
2 const protoLoader = require('@grpc/proto-loader');
3 const path = require('path');
4 const { Worker } = require('worker_threads');
5
6 const PROTO_PATH = path.join(__dirname, 'pet_adoption.proto');
7
8 // Load the protobuf
9 const packageDefinition = protoLoader.loadSync(PROTO_PATH, {});
10 const petAdoptionProto = grpc.loadPackageDefinition(packageDefinition);
11
12 // Sample data
13 let pets = [];
14
15 // Utility function to handle worker creation
16 const createWorker = (task, petData = {}, petId = null, callback) => {
17   const worker = new Worker(path.join(__dirname, 'worker.js'), {
18     workerData: { task, petData, pets, petId },
19   });
20
21   worker.on('message', (response) => {
22     if (response.error) {
23       callback({
24         code: grpc.status.NOT_FOUND,
25         details: response.message,
26       });
27     } else {
28       pets = response.pets; // Update pets after operation
29       callback(null, { message: response.message, pets: response.pets });
30     }
31   });
32
33   worker.on('error', (error) => {
34     console.error('Worker error:', error);
35     callback({
36       code: grpc.status.INTERNAL,
37       details: 'An internal error occurred',
38     });
39   });
40 }
```

Worker.js



```
1 const { parentPort, workerData } = require('worker_threads');
2
3 const { task, petData, pets, petId } = workerData;
4
5 console.log('Worker started for task: ${task}');
6
7 const handleRequest = () => {
8   let response;
9
10  switch (task) {
11    case 'register':
12      petData.id = pets.length + 1; // Assign a new ID based on current length
13      pets.push(petData); // Store the new pet
14      response = { message: 'Pet ${petData.name} has been registered.', pets };
15      break;
16
17    case 'adopt':
18      const petIndex = pets.findIndex(p => p.id === petId); // Find pet by ID
19      if (petIndex === -1) {
20        response = { error: true, message: 'Pet not found.' }; // Handle not found
21      } else {
22        const adoptedPet = pets.splice(petIndex, 1)[0]; // Remove the adopted pet
23        response = { message: 'Successfully adopted pet with ID: ${adoptedPet.id}', pets };
24      }
25      break;
26
27    case 'search':
28      const foundPets = pets.filter(p =>
29        (petData.name || p.name.toLowerCase() === petData.name.toLowerCase()) &&
30        (petData.gender || p.gender.toLowerCase() === petData.gender.toLowerCase()) && // Updated to include gender
31        (petData.breed || p.breed.toLowerCase() === petData.breed.toLowerCase()) && // Updated to include breed
32        (petData.age || p.age === petData.age) &&
33        (petData.location || p.location.toLowerCase() === petData.location.toLowerCase()) // Updated to include location
34      );
35
36      response = { pets: foundPets, message: foundPets.length ? '' : 'No pets found.' };
37      break;
38
39    default:
40      response = { error: true, message: 'Invalid task.' };
41  }
42 }
```

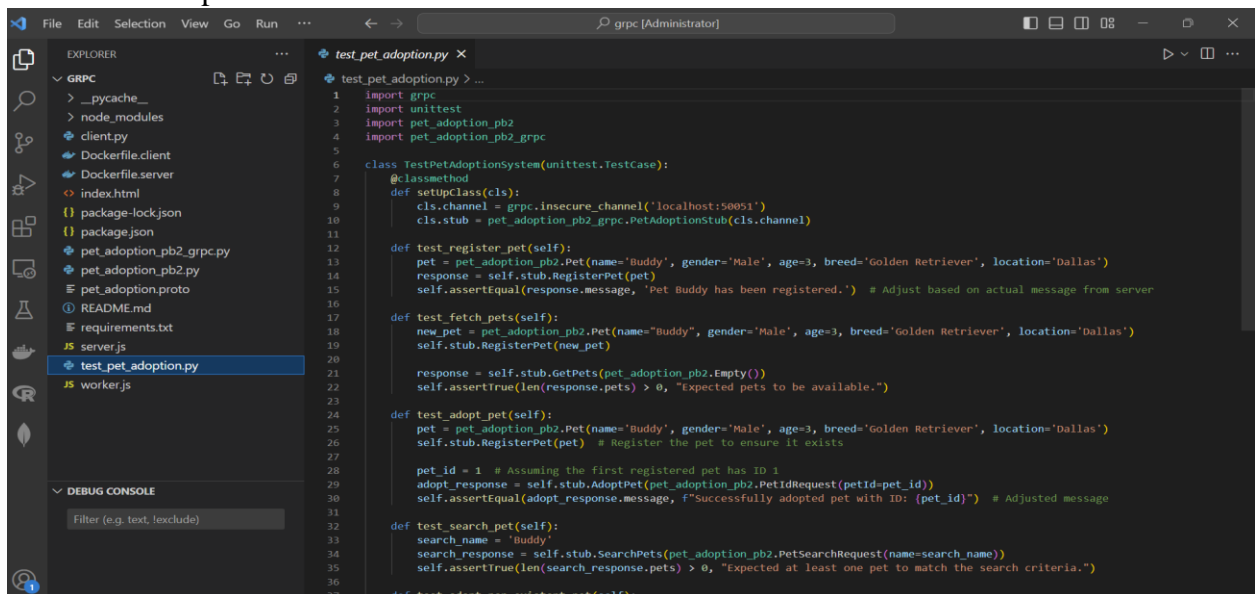
Client Script:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure for a gRPC client. The main editor window shows the `client.py` file. The code defines a `client` module with three functions: `register_pet`, `fetch_pets`, and `adopt_pet`. Each function uses the `grpc` library to interact with a `PetAdoptionService` stub. The `register_pet` function takes user input for pet details and registers a new pet. The `fetch_pets` function retrieves a list of available pets. The `adopt_pet` function allows a user to adopt a specific pet by ID.

```
1 import grpc
2 import pet_adoption_pb2
3 import pet_adoption_pb2_grpc
4
5 def register_pet(stub):
6     name = input("Enter the pet's name: ")
7     gender = input("Enter the pet's gender (e.g., Male, Female): ") # New input
8     breed = input("Enter the pet's breed: ") # New input
9     age = int(input("Enter the pet's age: "))
10    location = input("Enter the pet's location: ") # New input
11    pet = pet_adoption_pb2.Pet(name=name, gender=gender, breed=breed, age=age, location=location) # Updated
12    response = stub.RegisterPet(pet)
13    print(f"Pet {name} has been registered.")
14
15 def fetch_pets(stub):
16    print("Fetching list of pets:")
17    response = stub.GetPets(pet_adoption_pb2.Empty())
18
19    if not response.pets: # Check if the list is empty
20        print("No pets available.")
21    else:
22        for pet in response.pets:
23            print(f"Pet ID: {pet.id}, Name: {pet.name}, Gender: {pet.gender}, Age: {pet.age}, Breed: {pet.breed}, Location: {pet.location}")
24
25 def adopt_pet(stub):
26    pet_id = int(input("Enter the ID of the pet you want to adopt: "))
27    try:
28        adopt_response = stub.AdoptPet(pet_adoption_pb2.PetIdRequest(pet_id=pet_id))
29        print(adopt_response.message)
30    except grpc.RpcError as e:
31        print(f"Error: {e.details()}") # Print the error message returned by the server
32
33 def search_pet(stub):
34    search_name = input("Enter a name to search for pets: ")
35    search_response = stub.SearchPets(pet_adoption_pb2.PetSearchRequest(name=search_name))
36
37    if not search_response.pets:
38        print("No pets found matching the search criteria.")
39    else:
40        print("Search Results:")
```

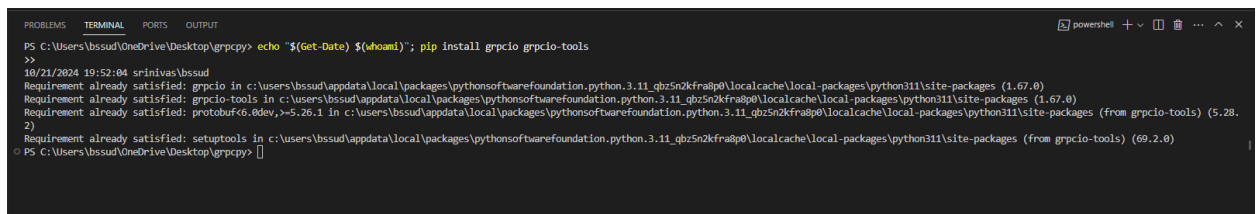
Test Cases Script:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `test_pet_adoption.py` file. The code defines a `TestPetAdoptionSystem` class that inherits from `unittest.TestCase`. It includes setup methods for the test environment and several test functions: `test_register_pet`, `test_fetch_pets`, `test_adopt_pet`, and `test_search_pet`. Each test function uses the `grpc` library to interact with a `PetAdoptionService` stub and asserts the expected results.

```
1 import grpc
2 import unittest
3 import pet_adoption_pb2
4 import pet_adoption_pb2_grpc
5
6 class TestPetAdoptionSystem(unittest.TestCase):
7     @classmethod
8     def setUpClass(cls):
9         cls.channel = grpc.insecure_channel('localhost:50051')
10        cls.stub = pet_adoption_pb2_grpc.PetAdoptionStub(cls.channel)
11
12    def test_register_pet(self):
13        pet = pet_adoption_pb2.Pet(name='Buddy', gender='Male', age=3, breed='Golden Retriever', location='Dallas')
14        response = self.stub.RegisterPet(pet)
15        self.assertEqual(response.message, 'Pet Buddy has been registered.') # Adjust based on actual message from server
16
17    def test_fetch_pets(self):
18        new_pet = pet_adoption_pb2.Pet(name='Buddy', gender='Male', age=3, breed='Golden Retriever', location='Dallas')
19        self.stub.RegisterPet(new_pet)
20
21        response = self.stub.GetPets(pet_adoption_pb2.Empty())
22        self.assertTrue(len(response.pets) > 0, "Expected pets to be available.")
23
24    def test_adopt_pet(self):
25        pet = pet_adoption_pb2.Pet(name='Buddy', gender='Male', age=3, breed='Golden Retriever', location='Dallas')
26        self.stub.RegisterPet(pet) # Register the pet to ensure it exists
27
28        pet_id = 1 # Assuming the first registered pet has ID 1
29        adopt_response = self.stub.AdoptPet(pet_adoption_pb2.PetIdRequest(pet_id=pet_id))
30        self.assertEqual(adopt_response.message, f'Successfully adopted pet with ID: {pet_id}') # Adjusted message
31
32    def test_search_pet(self):
33        search_name = 'Buddy'
34        search_response = self.stub.SearchPets(pet_adoption_pb2.PetSearchRequest(name=search_name))
35        self.assertTrue(len(search_response.pets) > 0, "Expected at least one pet to match the search criteria.")
36
37    def test_adopt_non_existent_pet(self):
```

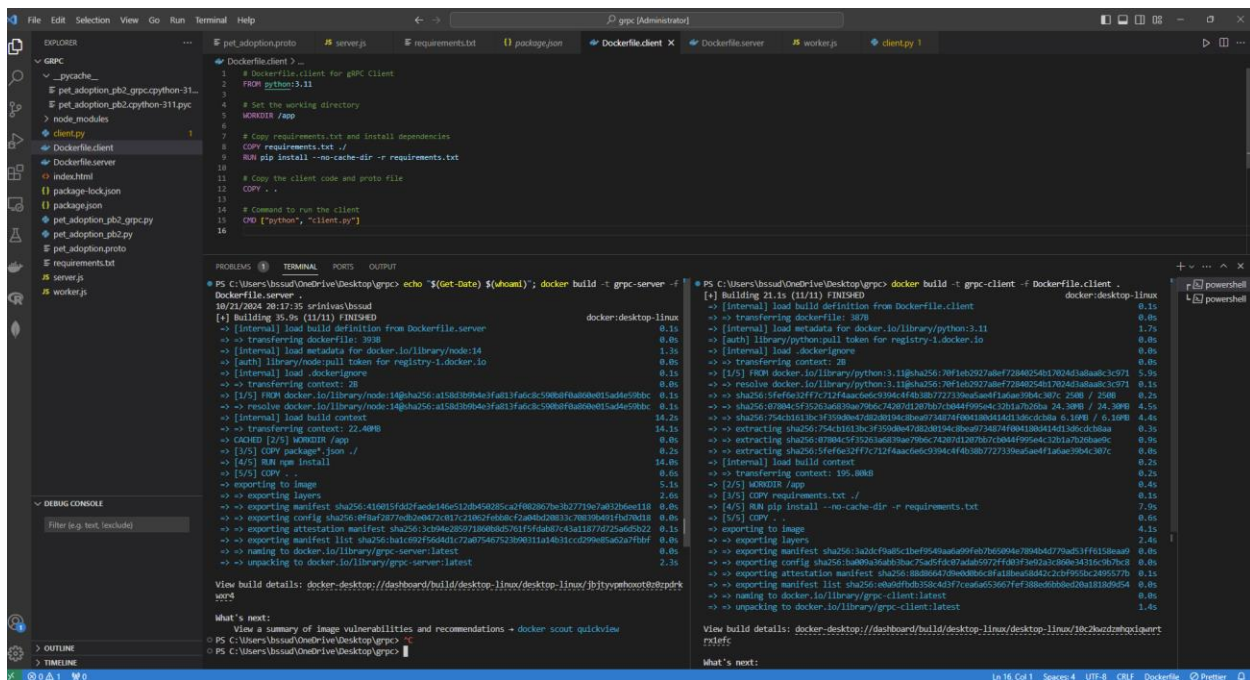
Packages needed:



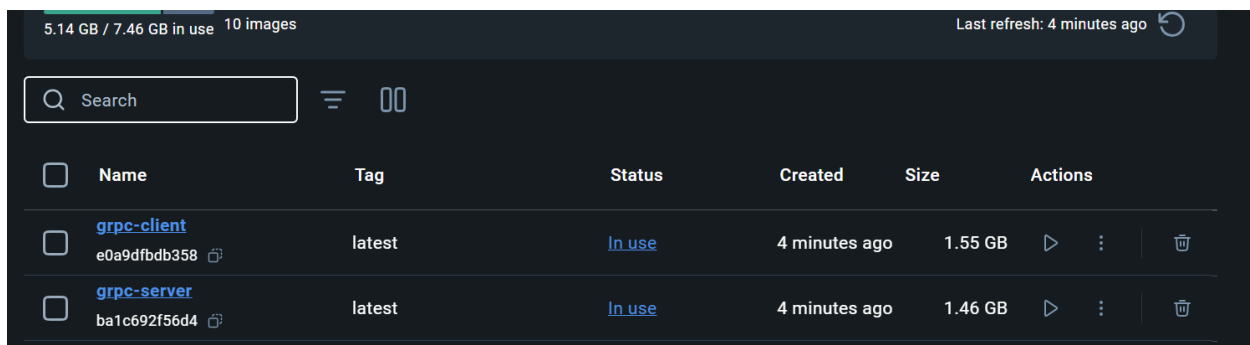
The screenshot shows a PowerShell terminal window with the command `pip install grpcio grpcio-tools` being executed. The output shows the installation progress and the paths where the packages were installed.

```
PS C:\Users\bssud\OneDrive\Desktop\grpcpy> echo $(Get-Date) $(whoami); pip install grpcio grpcio-tools
4/22/2024 19:52:04 ssinvs\bssud
Requirement already satisfied: grpcio in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (1.67.0)
Requirement already satisfied: grpcio-tools in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (1.67.0)
Requirement already satisfied: protobuf<6.0dev,>=5.26.1 in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from grpcio-tools) (5.28.2)
Requirement already satisfied: setuptools in c:\users\bssud\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from grpcio-tools) (69.2.0)
PS C:\Users\bssud\OneDrive\Desktop\grpcpy>
```

```
PS C:\Users\bssud\OneDrive\Desktop\grpcpy> echo "$(Get-Date) $(whoami)"; python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. pet.proto
>>
18/21/2024 19:54:31 srinivas\bssud
PS C:\Users\bssud\OneDrive\Desktop\grpcpy>
```



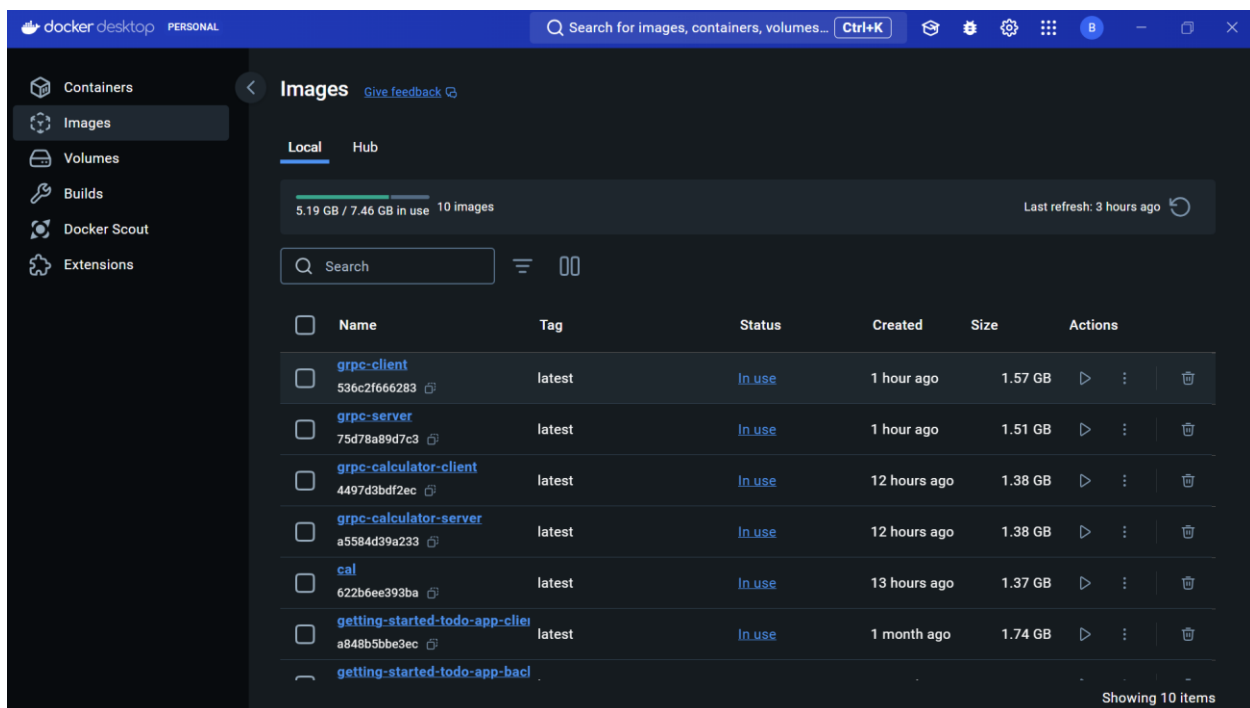
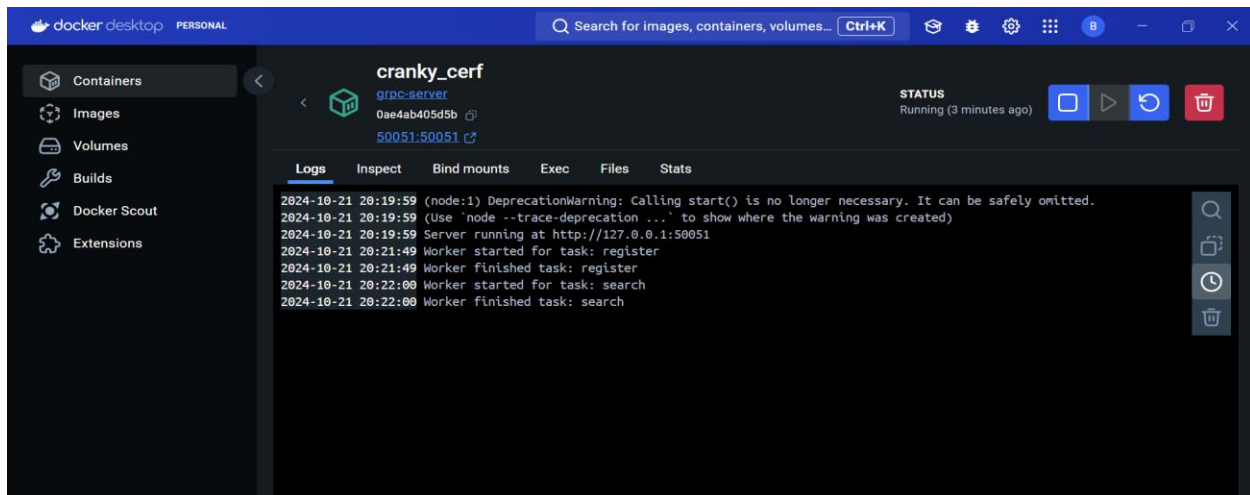
Docker Image:



To manually run in the IDE we can pull image and run this cmds:

```
docker run -p 50051:50051 grpc-server
```

```
docker run -it --network="host" grpc-client
```



Running Test Cases:

