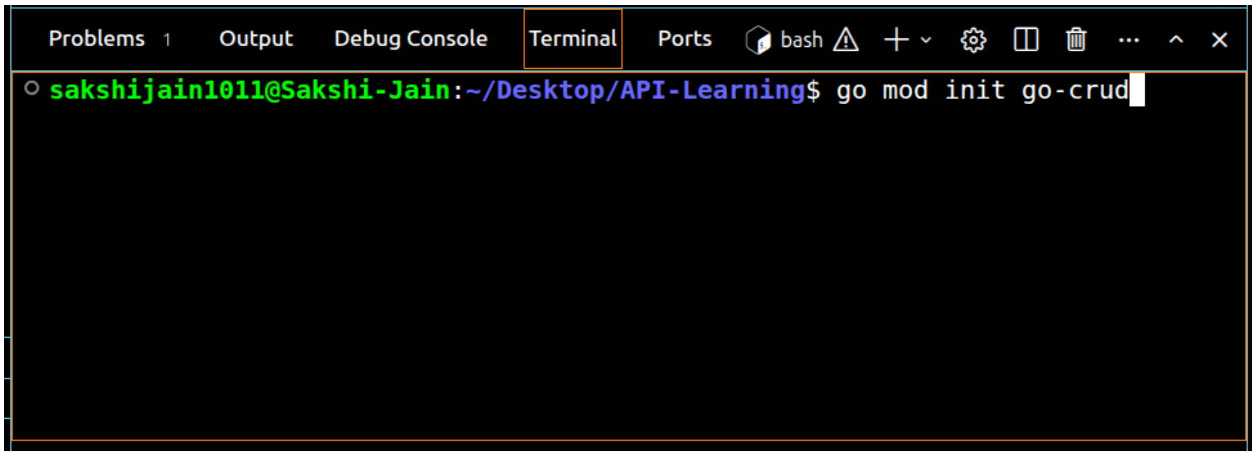
1. **Choose any backend stack you know (Rails, Express.js, Django, Phoenix, etc.).**

I choose Golang language as backend stack

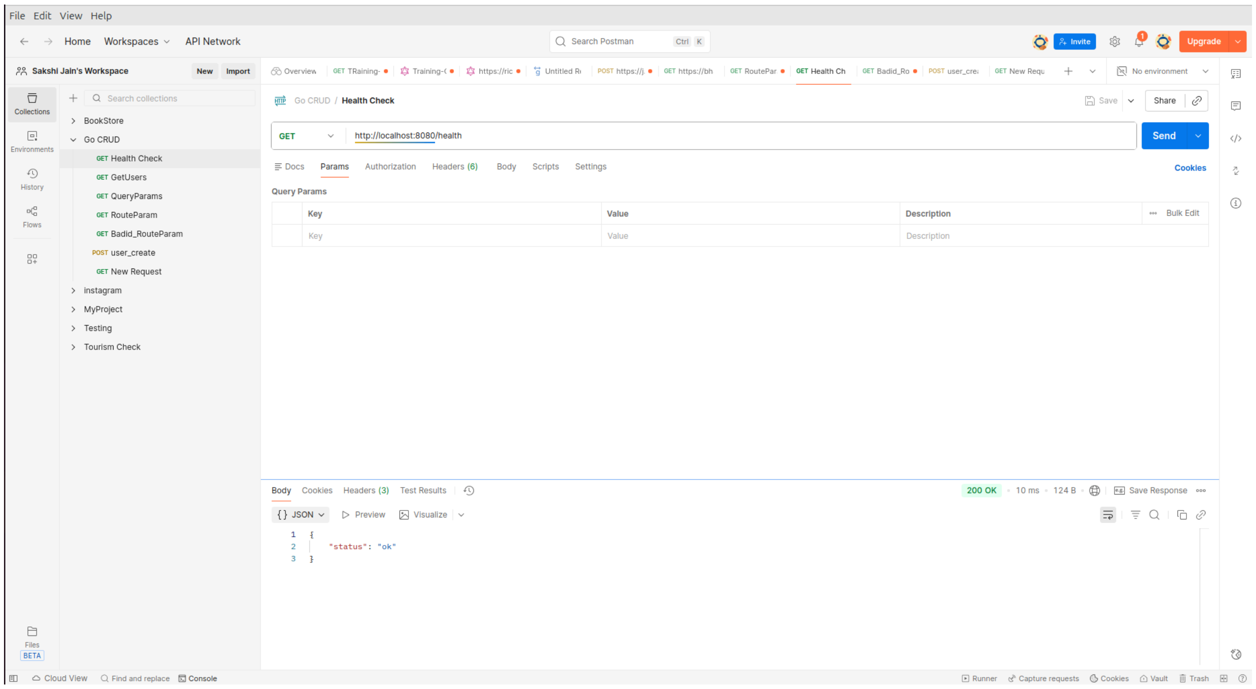
2. **Create a new API project.**

Initialised Project using go mod init filename



```
sakshijain1011@Sakshi-Jain:~/Desktop/API-Learning$ go mod init go-crud
```

3. **Implement endpoints:**
    - **GET /health → returns JSON: { "status": "ok" }**

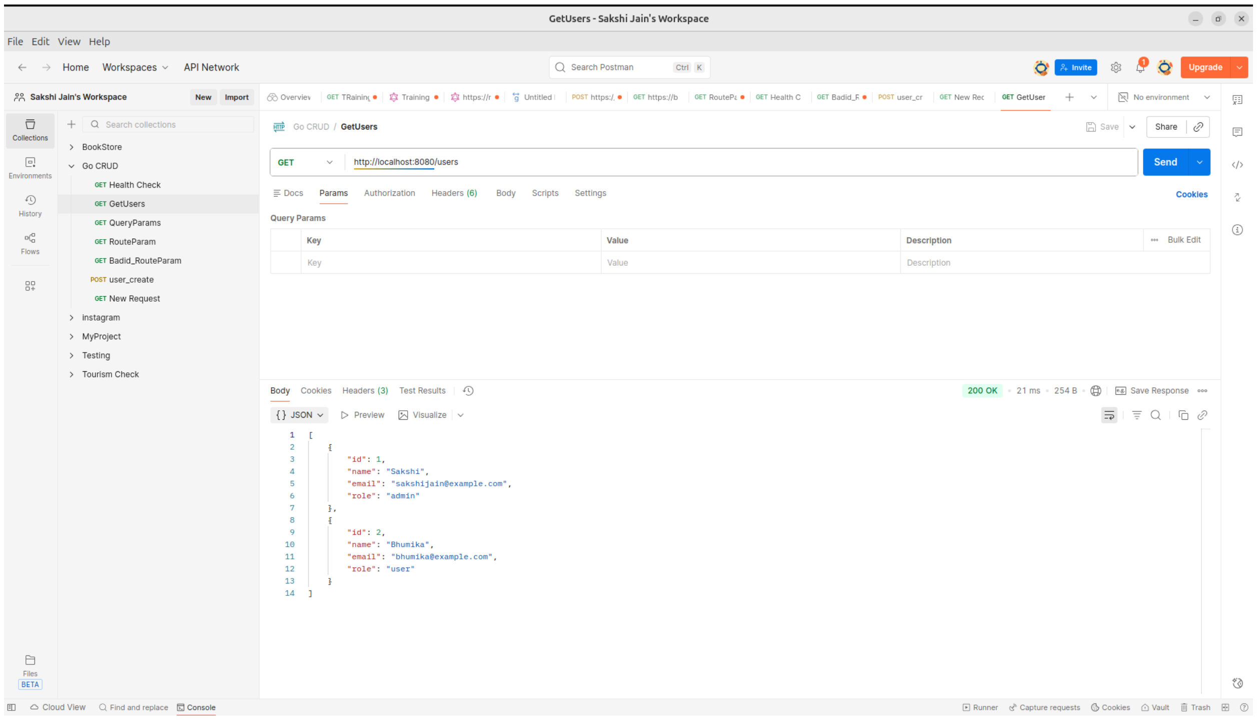


GET http://localhost:8080/health

200 OK · 10 ms · 124 B

```
{
    "status": "ok"
}
```

- GET /users → returns a list of users (can be static/in-memory).



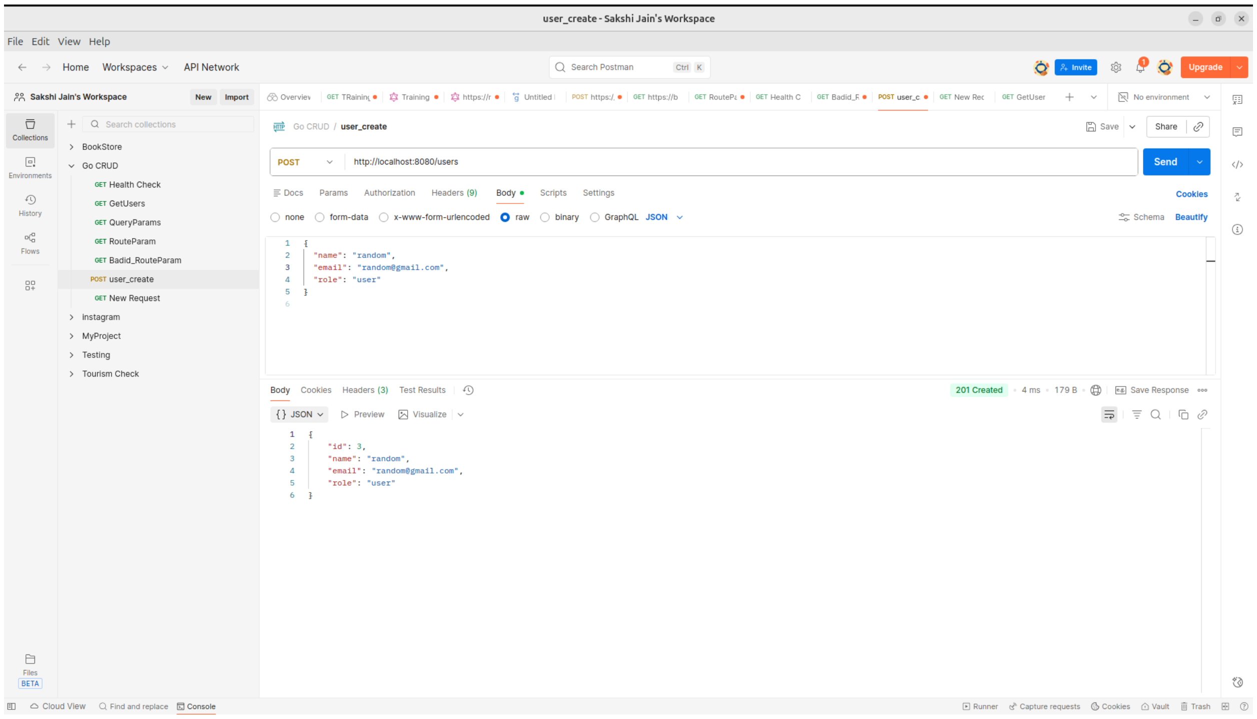
GetUsers - Sakshi Jain's Workspace

GET http://localhost:8080/users

200 OK · 21 ms · 254 B

```
[
    {
        "id": 1,
        "name": "Sakshi",
        "email": "sakshijain@example.com",
        "role": "admin"
    },
    {
        "id": 2,
        "name": "Bhumika",
        "email": "bhumika@example.com",
        "role": "user"
    }
]
```

- POST /users → accepts JSON body to create a user with at least name & email.


user_create - Sakshi Jain's Workspace

Go CRUD / user_create

POST http://localhost:8080/users

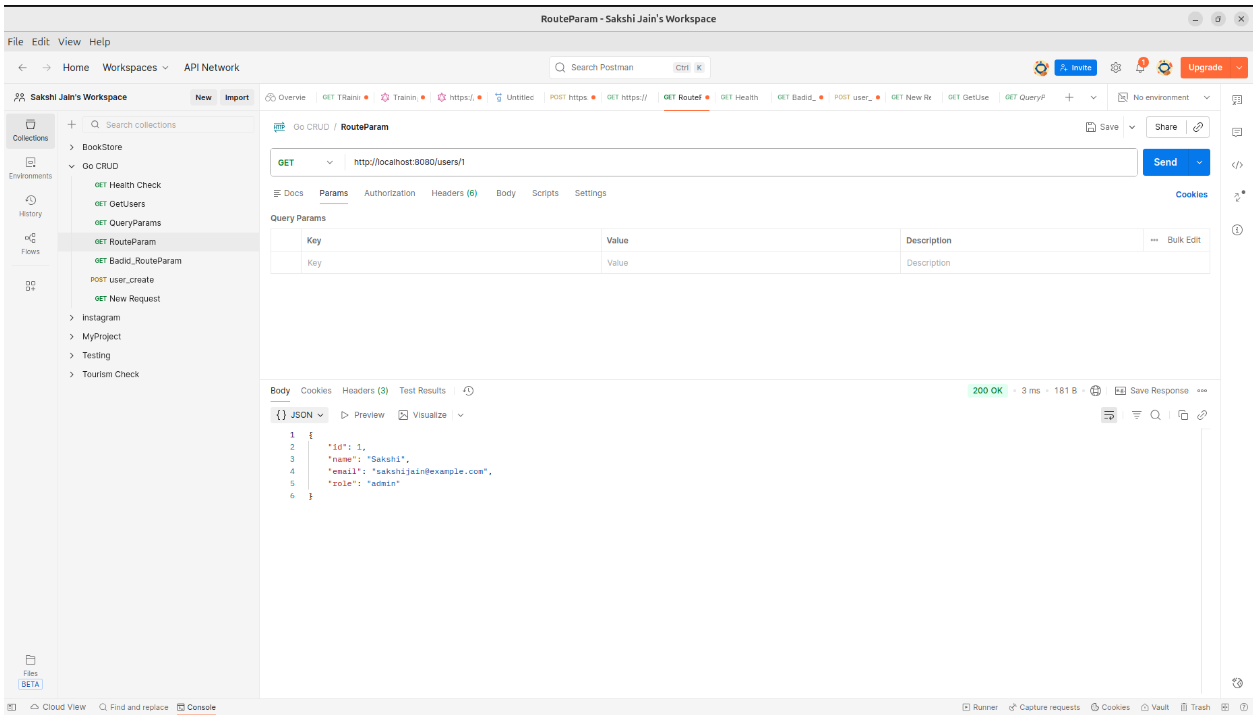
```
{
  "name": "random",
  "email": "random@gmail.com",
  "role": "user"
}
```

201 Created · 4 ms · 179 B

```
{
    "id": 3,
    "name": "random",
    "email": "random@gmail.com",
    "role": "user"
}
```

4. Support:

- Route params (if implementing GET /users/:id)


RouteParam - Sakshi Jain's Workspace

Go CRUD / RouteParam

GET http://localhost:8080/users/1

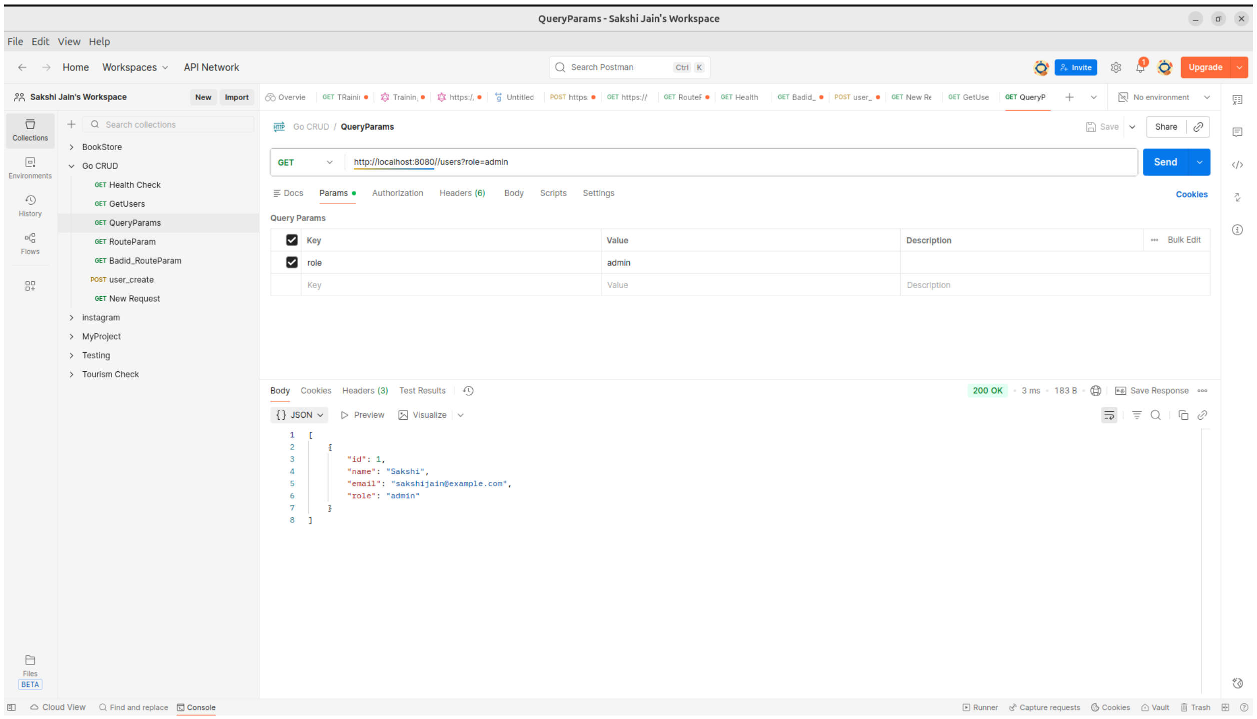
Query Params

Key	Value	Description
Key	Value	Description

200 OK · 3 ms · 181 B

```
{
    "id": 1,
    "name": "Sakshi",
    "email": "sakshijain@example.com",
    "role": "admin"
}
```

○ **Query params (e.g., GET /users?role=admin)**



GET http://localhost:8080//users?role=admin

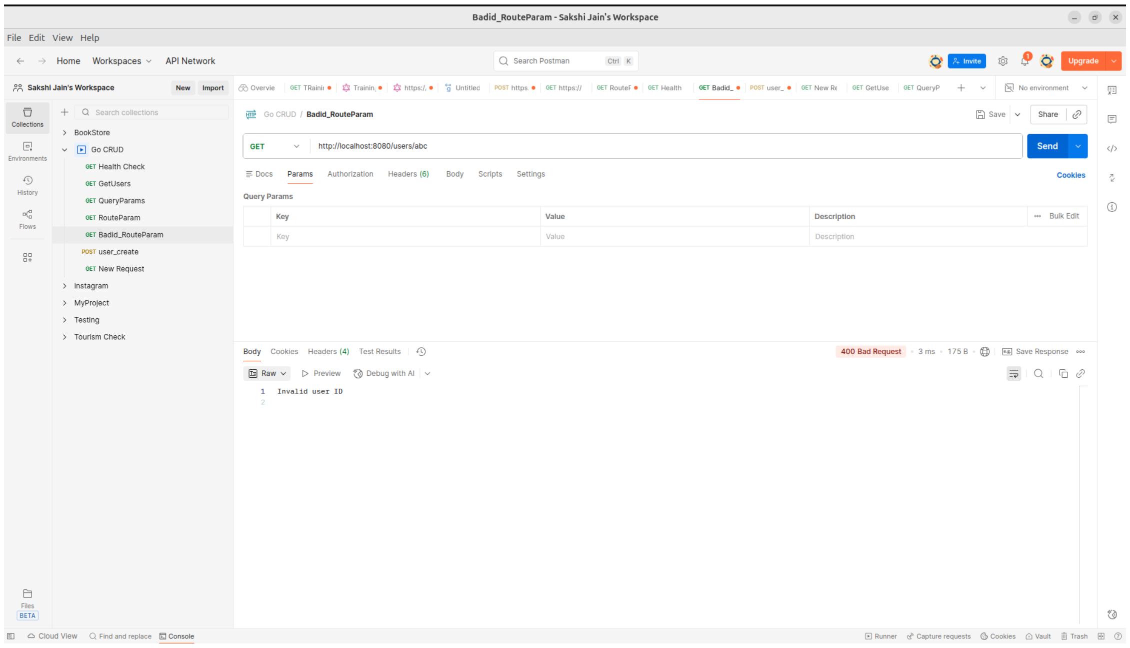
Query Params

	Key	Value	Description
✓	role	admin	

200 OK · 3 ms · 183 B

```
[
    {
        "id": 1,
        "name": "Sakshi",
        "email": "sakshijain@example.com",
        "role": "admin"
    }
]
```

5. **Ensure correct HTTP Status codes (200, 201, 400).**
   i. **Works for correct request**
   ii. **For Bad Request**



GET http://localhost:8080/users/abc

400 Bad Request · 3 ms · 175 B

```
Invalid user ID
```

6. **Submit: routes file, controller code, sample Postman collection export.**

Repository Link:
https://github.com/sakshijain-josh/Training/tree/main/Backend_Training/go-crud

Postman Collection of API:
https://api.postman.com/collections/43513762-47bffec8-96f6-43ab-a5e6-7c365be4e23a?access_key=PMAT-01KEH025T20BHF7CZSKEGFZ5GM