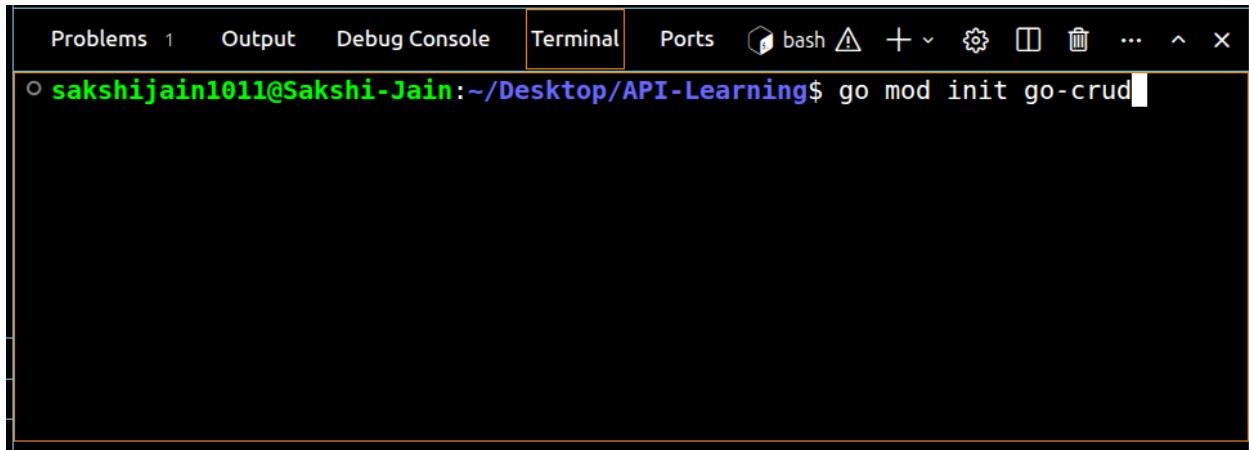


1. Choose any backend stack you know (Rails, Express.js, Django, Phoenix, etc.).

I choose Golang language as backend stack

2. Create a new API project.

Initialised Project using go mod init filename



A screenshot of a terminal window from a code editor. The terminal tab is selected. The command `sakshijain1011@Sakshi-Jain:~/Desktop/API-Learning$ go mod init go-crud` is visible in the terminal window.

3. Implement endpoints:

- o GET /health → returns JSON: { "status": "ok" }

The screenshot shows the Postman interface with a successful response to a GET request to `http://localhost:8080/health`. The response body is JSON, containing the key "status" with the value "ok".

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

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

The screenshot shows the Postman interface with a successful response to a GET request to `http://localhost:8080/users`. The response body is JSON, listing two users: Sakshi and Bhunika.

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

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

The screenshot shows the Postman interface with a collection named "Sakshi Jain's Workspace". A POST request is made to `http://localhost:8080/users` with the following JSON body:

```

1 {
2   "name": "random",
3   "email": "random@gmail.com",
4   "role": "user"
5 }

```

The response is a `201 Created` status with a response body identical to the request body.

4. Support:

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

The screenshot shows the Postman interface with a collection named "Sakshi Jain's Workspace". A GET request is made to `http://localhost:8080/users/1`. The "Query Params" section shows a key "Key" with a value "Value". The response is a `200 OK` status with a response body containing user details:

```

1 {
2   "id": 1,
3   "name": "Sakshi",
4   "email": "sakshijain@example.com",
5   "role": "admin"
6 }

```

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

The screenshot shows the Postman interface with the following details:

- Collection:** Sakshi Jain's Workspace
- Request Type:** GET
- URL:** <http://localhost:8080/users?role=admin>
- Query Params:**

| Key | Value | Description |
|------|-------|-------------|
| role | admin | |
- Body (JSON Preview):**

```
{
  "id": 1,
  "name": "Sakshi",
  "email": "sakshijain@example.com",
  "role": "admin"
}
```
- Status:** 200 OK
- Headers:** 3 ms, 183 B

5. Ensure correct HTTP Status codes (200, 201, 400).

- Works for correct request
- For Bad Request

The screenshot shows the Postman interface with the following details:

- Collection:** Sakshi Jain's Workspace
- Request Type:** GET
- URL:** <http://localhost:8080/users/abc>
- Query Params:**

| Key | Value | Description |
|-----|-------|-------------|
| Key | Value | Description |
- Body (Raw Preview):**

```
1 Invalid user ID
2
```
- Status:** 400 Bad Request
- Headers:** 3 ms, 175 B

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](https://api.postman.com/collections/43513762-47bffec8-96f6-43ab-a5e6-7c365be4e23a?access_key=PMAT-01KEH025T20BHF7CZSKEGFZ5GM)