

Assignment 2: Customizing the Go API Codebase

Environment Setup

Item	Details
Operating System	Linux
Programming Lang	Go
Git Repo	https://github.com/Muditha-Kumara/Go/tree/task2
Commit	b8eb45c

Project Structure:

```
API 0.1/  
  cmd/api/main.go  
  internal/api/handlers/  
  internal/api/middleware/  
  internal/api/repository/  
  internal/api/server/  
  internal/api/service/
```

Introduction

This report documents the process of customizing the provided Go backend codebase as per the assignment requirements. The main objectives were to modify the Data entity, update the database schema, adjust handlers, services, validators, authentication, and ensure all changes are tested and documented.

1. Codebase Exploration

The project follows a clear Handler-Service-Repository-Model pattern:

- **Handler Layer:** Handles HTTP requests and responses (`internal/api/handlers/data/`).
- **Service Layer:** Contains business logic and validation (`internal/api/service/data/`).
- **Repository Layer:** Manages database access and queries (`internal/api/repository/DAL/SQLite/`).
- **Model Layer:** Defines data structures (`internal/api/repository/models/`).

The flow starts from `main.go` , which initializes the server, sets up middleware, and routes requests through the above layers.

2. Modifying the Data Entity

Original Data Entity:

```
type Data struct {
    ID          int
    DeviceID    string
    DeviceName  string
    Value       float64
    Type        string
    DateTime    string
    Description string
}
```

Testing Modification:

Suppose I add a new field `Location string` and remove `Description` .

```

2nd_Assignment > API 0.1 > internal > api > repository > models > -data.go > ...
1  package models
2
3  import "context"
4
5  type Data struct {
6      ID          int      `json:"id"`
7      DeviceID     string   `json:"device_id"`
8      DeviceName   string   `json:"device_name"`
9      Value        float64  `json:"value"`
10     Type          string   `json:"type"`
11     DateTime      string   `json:"date_time"`
12     Location      string   `json:"location"`
13 }
14
15 type DataRepository interface {
16     Create(Data *Data, ctx context.Context) error
17     ReadOne(id int, ctx context.Context) (*Data, error)
18     ReadMany(page int, rowsPerPage int, ctx context.Context) ([]*Data, error)
19     Update(data *Data, ctx context.Context) (int64, error)
20     Delete(data *Data, ctx context.Context) (int64, error)
21 }
22

```

3. Updating the Database Table

- The SQLite table schema was updated to match the new Data entity.
- The `production.db` file was deleted to allow the application to auto-generate a new schema on startup.

SQL Table Changed:

```

CREATE TABLE IF NOT EXISTS data (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    device_id VARCHAR(50) NOT NULL,
    device_name VARCHAR(50),
    value FLOAT,
    data_type VARCHAR(20),
    date_time TIMESTAMP,
    location VARCHAR(100)
);

```

2nd_Assignment > API 0.1 > cmd > api > production.db

Filter Rows: 1 Filter 1 rows... Upgrade to PRO

TABLES		value	data_type	date_time	location
data					
ROWID	1	42	temp	2025-12-13T12:0...	lab
id	+	2			
device...					
device...					
value					
data_...					
date_...					
location					
sqlite_sequence					

4. Customizing Handlers

- All handlers in `internal/api/handlers/data/` were updated to parse and return the new Data structure.
- Request and response payloads were adjusted to include/exclude the modified fields.

```
2nd_Assignment > API 0.1 > internal > api > repository > models > data.go > ...
1  package models
2
3  import "context"
4
5  type Data struct {
6      ID          int      `json:"id"`
7      DeviceID    string   `json:"device_id"`
8      DeviceName  string   `json:"device_name"`
9      Value       float64  `json:"value"`
10     Type        string   `json:"type"`
11     DateTime    string   `json:"date_time"`
12     Location    string   `json:"location"`
13 }
14
15 type DataRepository interface {
16     Create(Data *Data, ctx context.Context) error
17     ReadOne(id int, ctx context.Context) (*Data, error)
18     ReadMany(page int, rowsPerPage int, ctx context.Context) ([]*Data, error)
19     Update(data *Data, ctx context.Context) (int64, error)
20     Delete(data *Data, ctx context.Context) (int64, error)
21 }
22
```

5. Adjusting Services and Validators

- The service layer's validation logic was updated to check the new/removed fields.

- Any business rules related to the new entity were implemented.

```

2nd_Assignment > API 0.1 > internal > api > service > data > mocks.go > (*MockDataServiceSuccessful).ReadOne
1  package data
2
3  import (
4      "context"
5      "goapi/internal/api/repository/models"
6  )
7
8  // * Mock implementation of DataService for testing purposes, always returns a successful response and Data object(s) *
9  type MockDataServiceSuccessful struct{}
10
11 func (m *MockDataServiceSuccessful) ReadMany(page int, rowsPerPage int, ctx context.Context) ([]*models.Data, error) {
12     return []*models.Data{
13         {
14             ID:          1,
15             DeviceID:    "device1",
16             DeviceName:   "device1",
17             Value:        1.0,
18             Type:         "type1",
19             DateTime:     "2021-01-01 00:00:00",
20             Location:     "location",
21         },
22         {
23             ID:          2,
24             DeviceID:    "device2",
25             DeviceName:   "device2",
26             Value:        2.0,
27             Type:         "type2",
28             DateTime:     "2021-01-01 00:00:00",
29             Location:     "location",
30         },
31     }, nil
32 }
33
34 func (m *MockDataServiceSuccessful) ReadOne(id int, ctx context.Context) (*models.Data, error) {
35     return &models.Data{
36         ID:          1,
37         DeviceID:    "device1",
38         DeviceName:   "device1",
39         Value:        1.0,
40         Type:         "type1",
41         DateTime:     "2021-01-01 00:00:00",
42         Location:     "location",
43     }, nil
44 }
45

```

6. Authentication Update

- The authentication method uses a static username and password, which are set in the authentication middleware (internal/api/middleware/basic_authentication.go).
- The username and password were changed from the default values to new credentials as required by the assignment.

```

func validateUser(username, password string) bool {
    // ! This is a dummy implementation, replace this with real authentication logic
    return username == "admin123" && password == "Testing@123"
}

```

7. Testing

- All unit tests were updated to reflect the new Data entity and logic.

- Tests were run using `go test` to ensure correctness.
- The API was tested with the new entity and credentials using Thunder Client/Postman.

```

muditha@lap:~/Go/2nd_Assignment/API 0.1$ curl -X POST http://127.0.0.1:8080/data -u admin123:Testing@123 -H "Content-Type: application/json" -d '{"device_id":"dev1","device name":"Sensor","value":42.0,"type":"temp","date_time":"2025-12-13T12:00:00Z","location":"lab"}'
{"id":1,"device_id":"dev1","device name":"Sensor","value":42,"type":"temp","date_time":"2025-12-13T12:00:00Z","location":"lab"}
muditha@lap:~/Go/2nd_Assignment/API 0.1$ curl -X GET http://127.0.0.1:8080/data -u admin123:Testing@123 -H "Content-Type: application/json"
[{"id":1,"device_id":"dev1","device name":"Sensor","value":42,"type":"temp","date_time":"2025-12-13T12:00:00Z","location":"lab"}]
muditha@lap:~/Go/2nd_Assignment/API 0.1$ go test ./...
?      goapi/cmd/api      [no test files]
?      goapi/internal/api/repository/DAL      [no test files]
?      goapi/internal/api/repository/DAL/SQLite [no test files]
ok     goapi/internal/api/handlers/data      (cached)
?      goapi/internal/api/service      [no test files]
?      goapi/internal/api/service/data [no test files]
?      goapi/internal/api/repository/models [no test files]
?      goapi/internal/api/server      [no test files]
ok     goapi/internal/api/middleware      (cached)
muditha@lap:~/Go/2nd_Assignment/API 0.1$

```



Conclusion

All assignment requirements were met:

- The Data entity and all related layers were successfully customized.
- The database schema was updated.
- Handlers, services, and validators were adjusted.
- Authentication was changed.
- All changes were tested and verified.