Net-centric Programming

Lab: Simple RESTful API service

In this lab, we will develop a simple user management system that allows to create, manage user information. The system provides a RESTful API for interfacing with the frontend app and a database will be used to store user information.

Features:

1. Our system can use simple sql database, e.g. sqlite3. Sample go code to use sqlite3:

```
package main


import (
	"database/sql"
	_ "github.com/mattn/go-sqlite3"
)


func main() {
	// Open SQLite database connection
	db, err := sql.Open("sqlite3", "./user_management.db")
	if err != nil {
		fmt.Println("Error opening database:", err)
		return
	}
	defer db.Close()


	// Create 'users' table
	_, err = db.Exec(`
		CREATE TABLE IF NOT EXISTS users (
			id INTEGER PRIMARY KEY,
			username TEXT NOT NULL,
```

```
                firstname TEXT NOT NULL

            )

      `)

      }
```

2. REST API support: we can use go default net/http service or any 3rd party library, e.g: gin-gonic framework: https://github.com/gin-gonic/gin . Sample code:

```
package main


import (

      "github.com/gin-gonic/gin"

      "net/http"

)


func main() {

      // Create a new Gin router

      router := gin.Default()

      // Define a route and its handler

      router.GET("/hello", func(c *gin.Context) {

            c.JSON(http.StatusOK, gin.H{"message": "Hello",})

      })

      // Run the server

      router.Run(":8080")

}
```

3. CRUD operations: create, read, update and delete user. Each user should have a least the following information: first name, last name, user name, email, avatar, phone number, date of birth, address (country, city, street name, street address). When create new user:
    a. API allows passing user information to store to database

   b. If user information is empty, the backend will generate random user information by calling the random-data-api.com services. Checkout https://random-data-api.com/ for more information

4. Filtering and Sorting: provide API to return users based on different attributes such as username, first name or last name. Enable sorting based on these attributes as well.

5. Error handling and validation: you should consider implementing some basic error handling and validation to ensure data integrity.

6. Since we will not implement the frontend app, to test your API, use hoppscotch or postman. For hoppscotch, we need to install the browser extension (https://github.com/hoppscotch/hoppscotch-extension) and setup your server endpoint (https://github.com/hoppscotch/hoppscotch/discussions/2051#discussioncomment-1887545)