College Appointment System

The college appointment system enables students and professors to manage their appointments, providing functionalities to:
1. View Appointments
2. Book/Cancel Appointments
3. Change slot availability

This project is built in Go, using Gin framework to handle server-side logic, we use the GORM library to interface with our sqlite database. The server uses cookies to store sessions of different users. Upon log-in, a user is given a cookie 🍪

Instructions to Run

The entry point of the program is `main.go` which will run a server on port `:8080`

```
go run main.go
```

To use the server reffer to the Postman Workspace: Link, ensure you set the environment to `ENV_VARIABLES`

Directory Structure

```
.
├── docs.md # my thought process while writing this codebase
├── go.mod
├── go.sum
├── handlers # handlers for different API endpoints
│   ├── auth.go
│   ├── prof.go
│   ├── structs.go
│   └── student.go
```

```
├── main.go # ENTRY POINT !
├── middleware # middleware for auth and control routes
│   └── middleware.go
├── models # datbase related files
│   ├── auth.go
│   ├── database.db
│   ├── schema.go
│   └── slots.go
├── routers # routing logic
│   ├── api
│   │   ├── auth.go
│   │   ├── prof.go
│   │   └── students.go
│   └── router.go
└── tests # test file
    └── integration_test.go
```

Handlers

The handler directory contains logic pertaining to how different endpoints should be handled. This involves using callback functions which are executed when a particular API end point is accessed. We have handlers for authorization and seperate logic for professors and students.

The expected JSON request format to avail these handlers are declared in `handlers/structs.go` which enforces JSON schema to incoming requests.

Middleware

We use middleware to authorize endpoints (except login and register) to logged in users. Further, using the ID, mail used while storing the session cookies, we can access the user type (which is either professor or student) and expose endpoints based on user type.

Models

This directory contains logic to communicate with the data, we are using `sqlite` database, and GORM library to interface with our database. This enables us to easily use structs as our objects which can easily by saved to our databases.

The functions in this directory are called primarily by `handlers` later on.

Routers

Handles the routing logic. Initializes a router and uses the handler and middleware to handle (or) expose different end points. This directory structure is based on usual standards. The code here also defines what API endpoints have which methods (PUT/POST/GET etc.) and the handler called when this endpoint is accessed.