Max Marks: 100

Golang Training Assessment

Q1. Create a package user. (30 Marks)

- In user package create a map which will store name as key and address as value.
- Add two dummy users data directly while map creation.
- Create a function GetAddress(name) which accepts name as argument and return corresponding address if present, and return an error if not present.
    - Hint: If address not found in map then map will return " " empty string.
    - **NOTE:** Declare errors at global scope and store them in variables.
      For e.g var AddressNotFound = errors.New("msg")
- Please test GetAddress function. Use table testing and Subtests. Create test file in models package.
- Please add at least two test cases for GetAddress function.
- **Note:-** No Need of Http programming and main file. Just create a package and test it.

Q2. Create a middleware that log data in standard output. Please log following things: (25 Marks)

- URL For E.g., /home
- Request Body E.g. {“name” : “Ana”}
- Header data Hint: Check request object
- Request Method E.g. GET
- **Hint**: Check r *request object. We have methods already defined there to get relevant data.

So, create an HTTP Server that accepts request on one endpoint and log the request data using middleware as mentioned above, and after logging data, route the request to the actual handler function where you can show or print some success message to the user.

Please Note :- Send JSON body from the postman and log that using middleware.

Q3. Make 15 concurrent request to **https://loripsum.net/api** and print the data received, wait for maximum 2 seconds for go routines to finish the task. If time runs out cancel the request using context. (25 Marks)

Q4. Make a struct with following fields: Seats , RWMutex, Waitgroup. Create two methods over struct. First method returns number of seats available. Second method books a seat. If seat is equal to zero no one can book it.
You can allow simultaneous reading of seats data but make sure only one go routine can book a seat at a time.
In main function set available seats field of struct to 4 and run 10 go routines to book a seat and 15 go routines which will try to read available seats.
Use wait groups as necessary. (20 Marks)

Note: Create a folder with your name and inside it create 4 folders for solution of each question, and then zip or rar it.