Report Midterm

Link to the Video: https: //drive.google.com/drive/folders/1o7AeGhum9-M_6xFueu6T6kLERXhpTHz_?usp=share_link

Team Members:

1.Aslan Mergenov 200103255 Connecting a database, setting up a function, everything else is possible

2.Diana Bisseitova 200103277 Responsible for working with templates and design

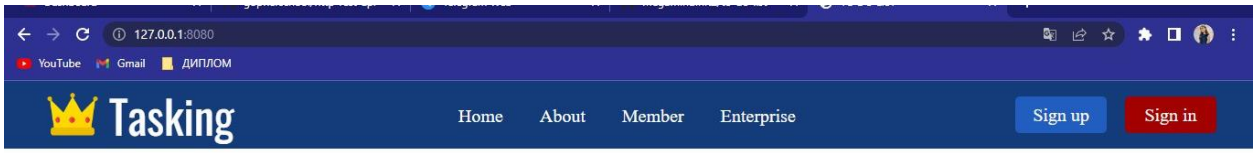
3.Maulen Myrzakhmet 200103458 Developed the function

What We Do

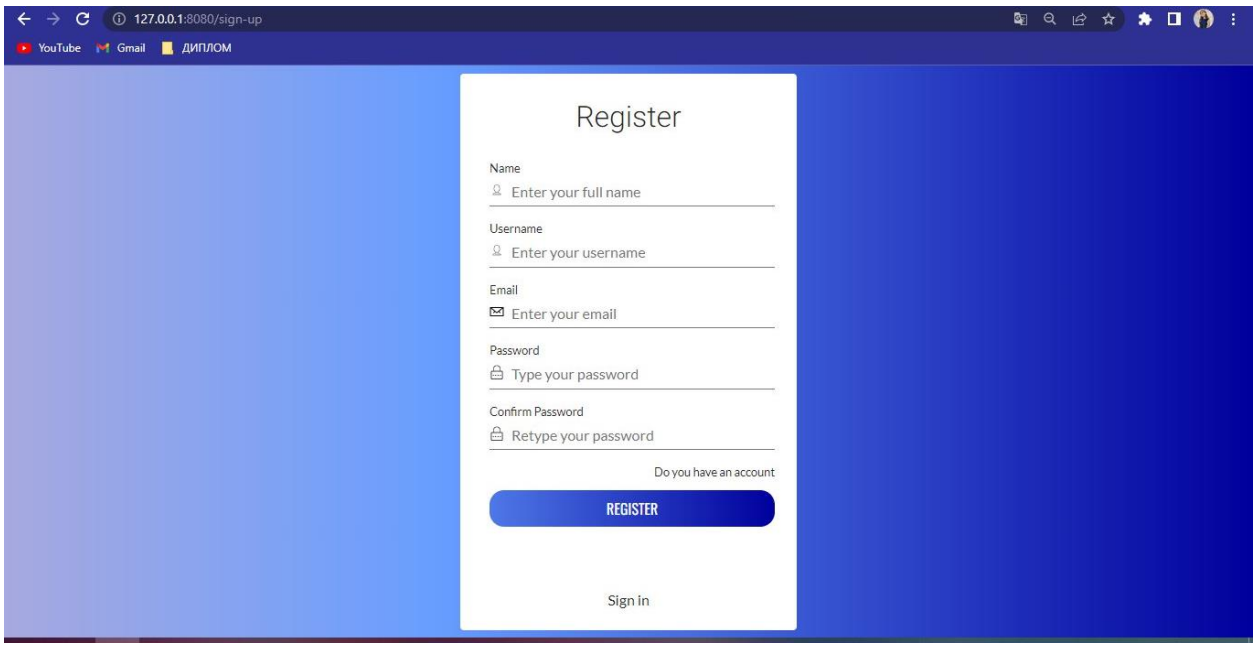
Made Registration with Authorization to the site.

The site itself has the ability to add files and data directly from the site. An ordinary site with an emphasis on cooking, do not pay attention to the site template until you have decided on it ....

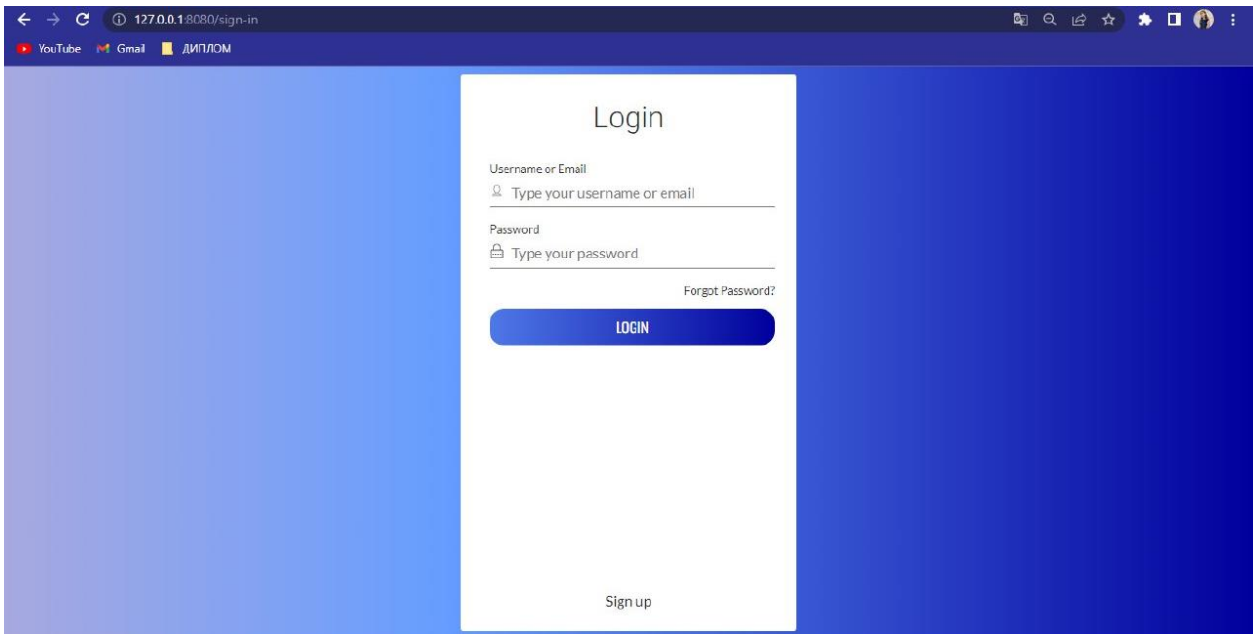
1. The primary template that meets when we go to the server, that is, welcome page. Here we see the title of the site and the registration and authorization functions themselves.



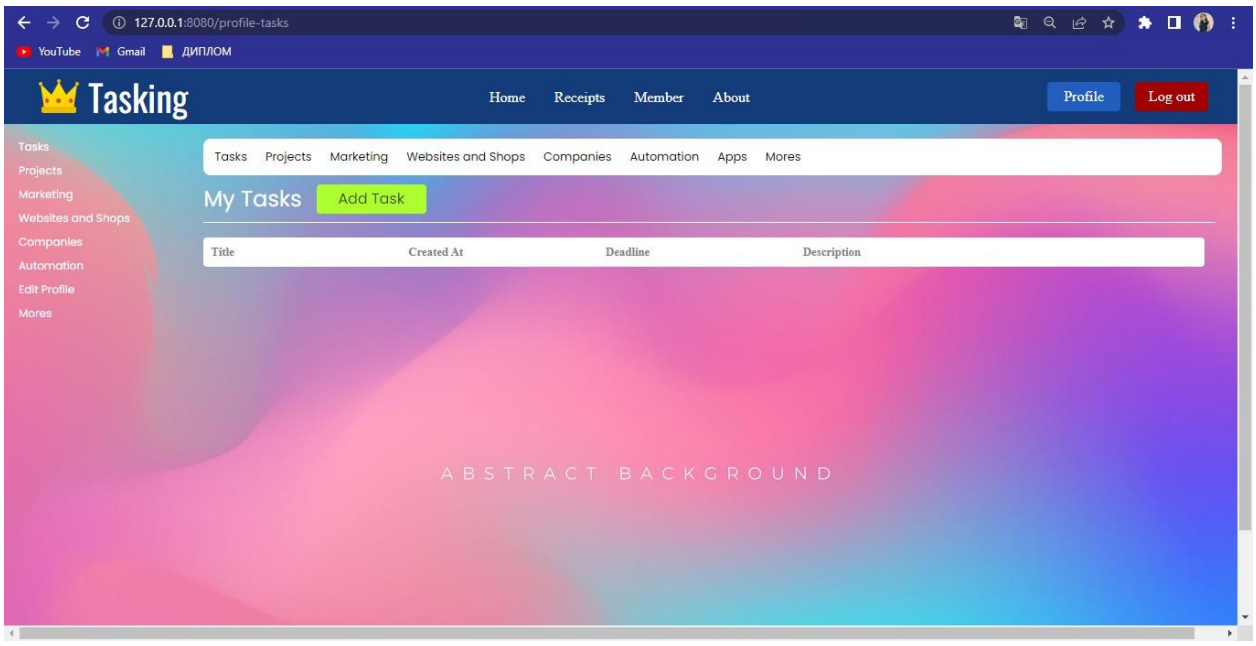
2. There is already a view of the registration template itself, with the usual registration data requests



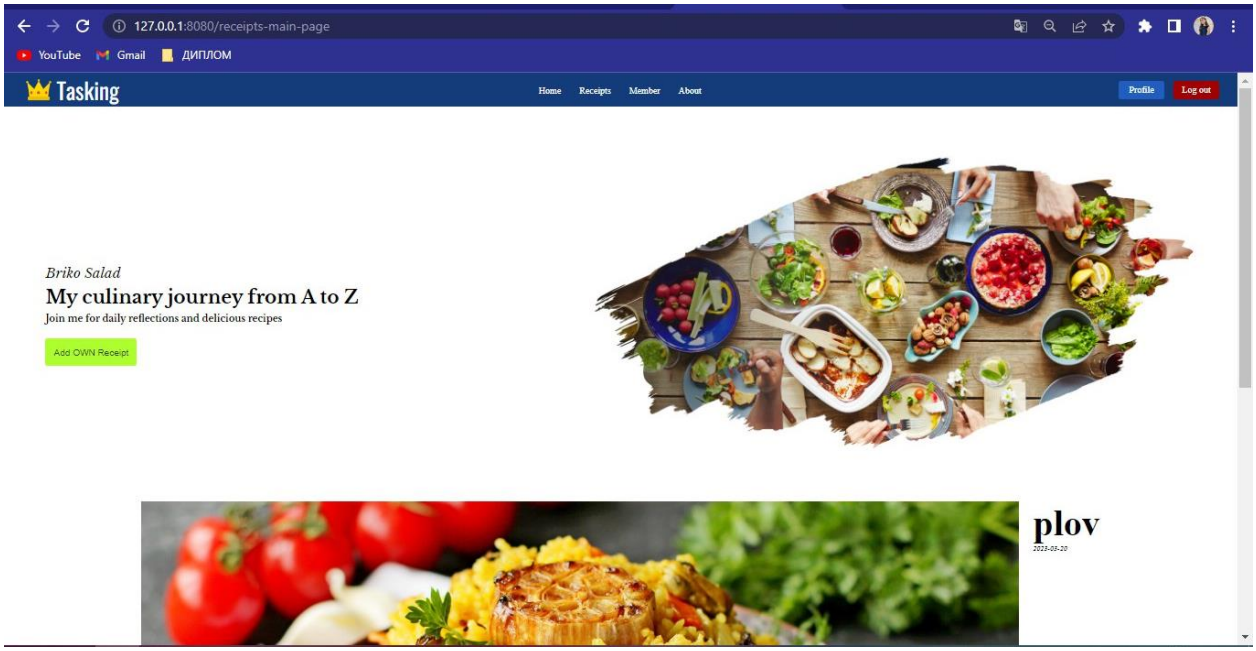
3. Here we see an authorization template, with basic requests



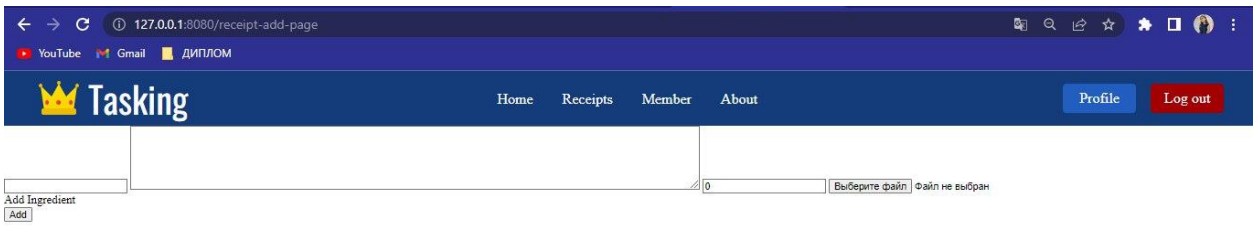
4. The view is already on the authorization itself, when the user logged into his account. There are working functions for adding tasks and changing the profile, as well as a recipe function, followed by a template and a function for adding data.



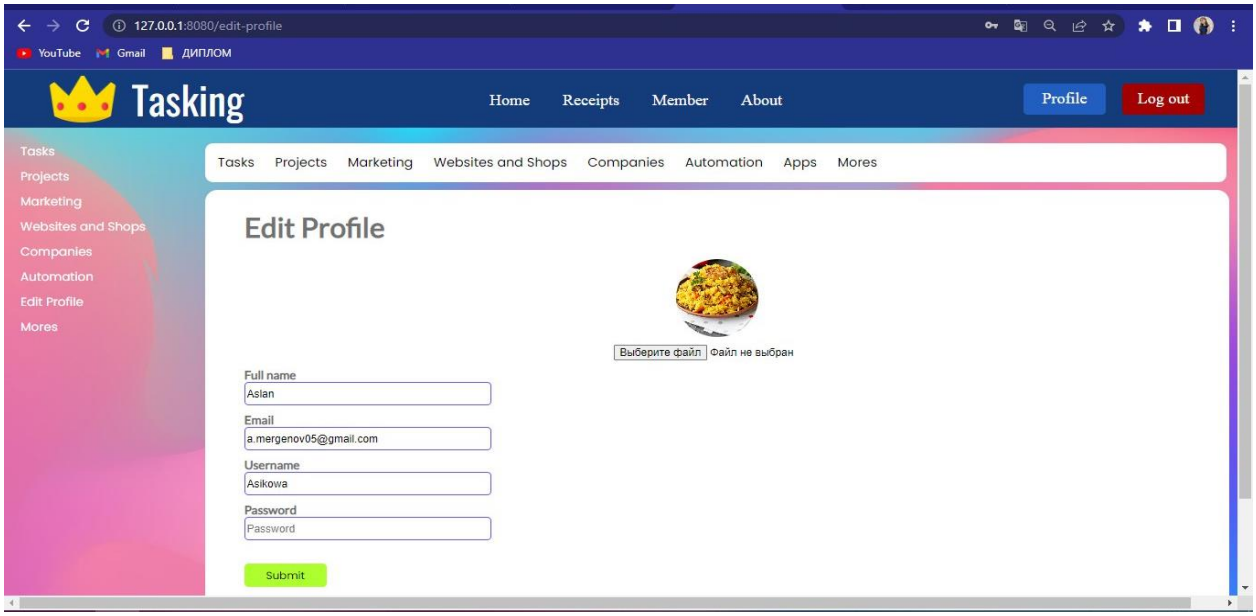
5. Template recipes, here we see recipes that we ourselves can add if desired



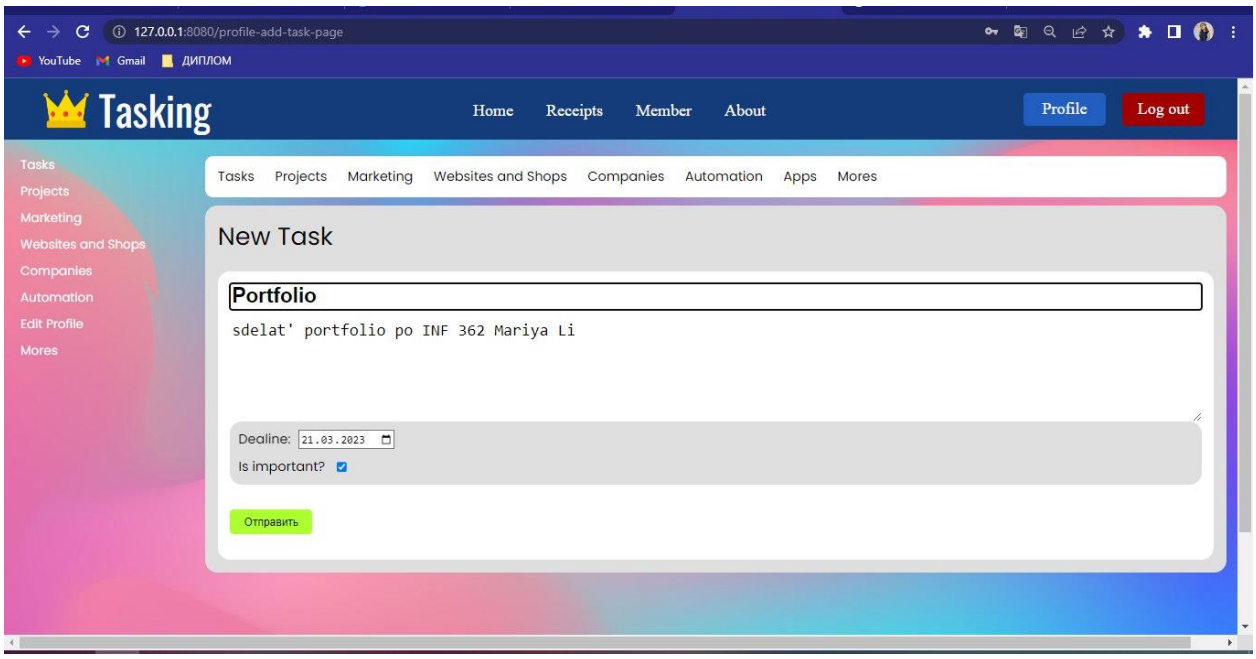
6. This is the function of adding recipes, the template itself.



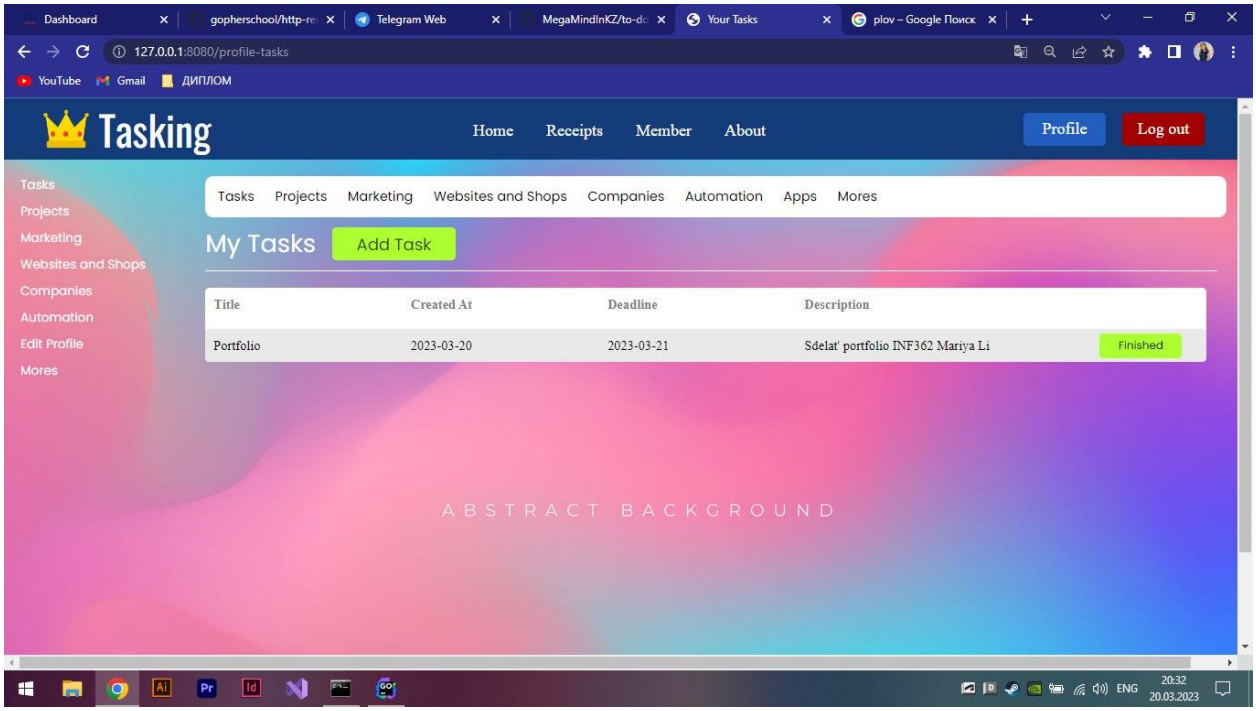
7. The function of changing the profile, here you can change the profile photo and information about yourself, changes are strictly through writing a password.



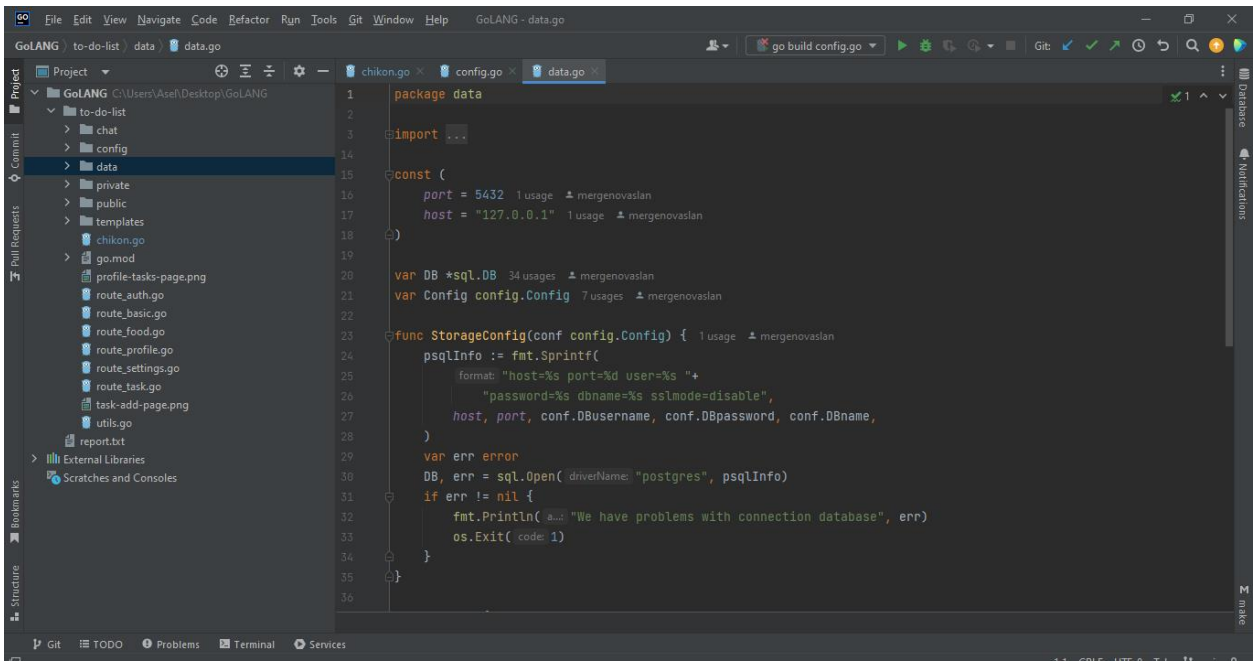
8. Function of adding tasks.



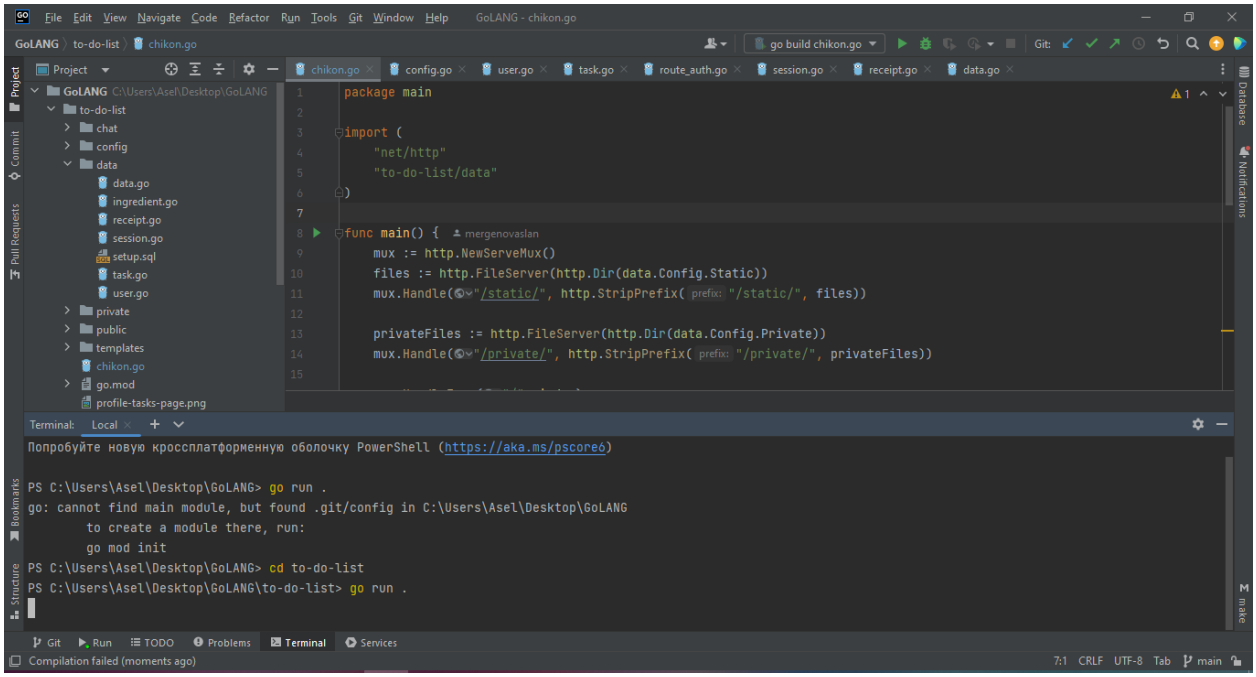
9. The view itself after adding a task or note, so to speak, the task disappears after pressing the finish button.



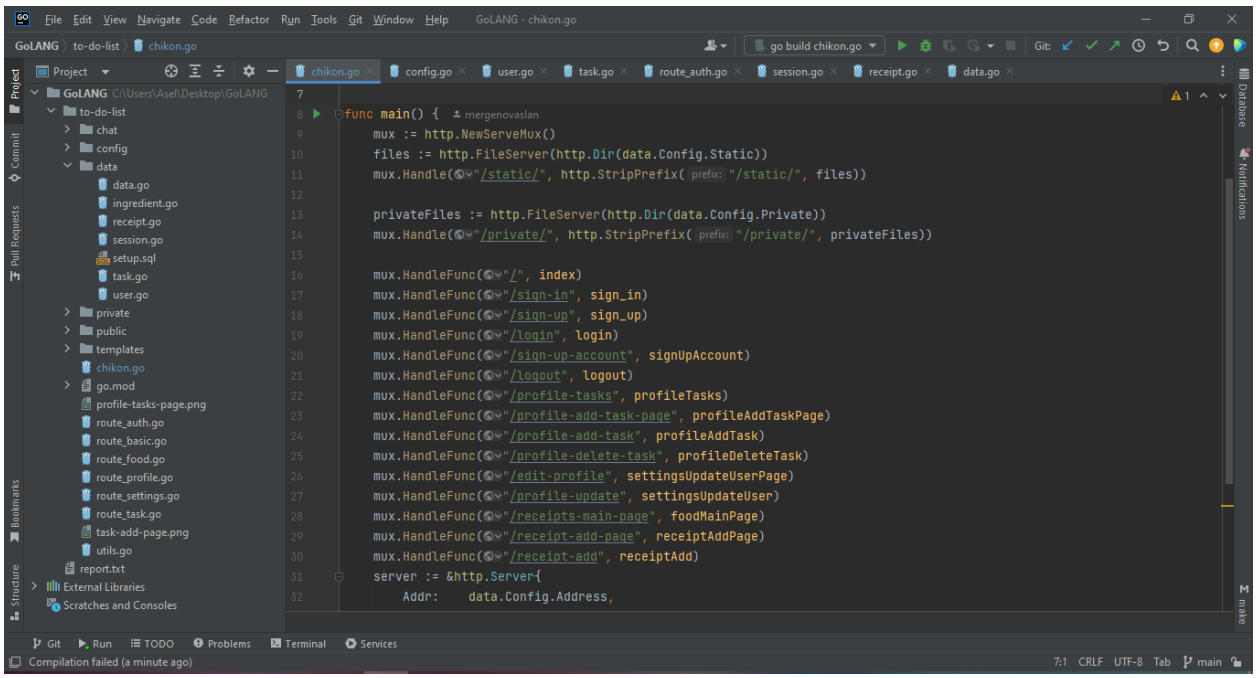
10. I connected the PostgreSQl database, registered the necessary data for further connection, this is our IP address and host



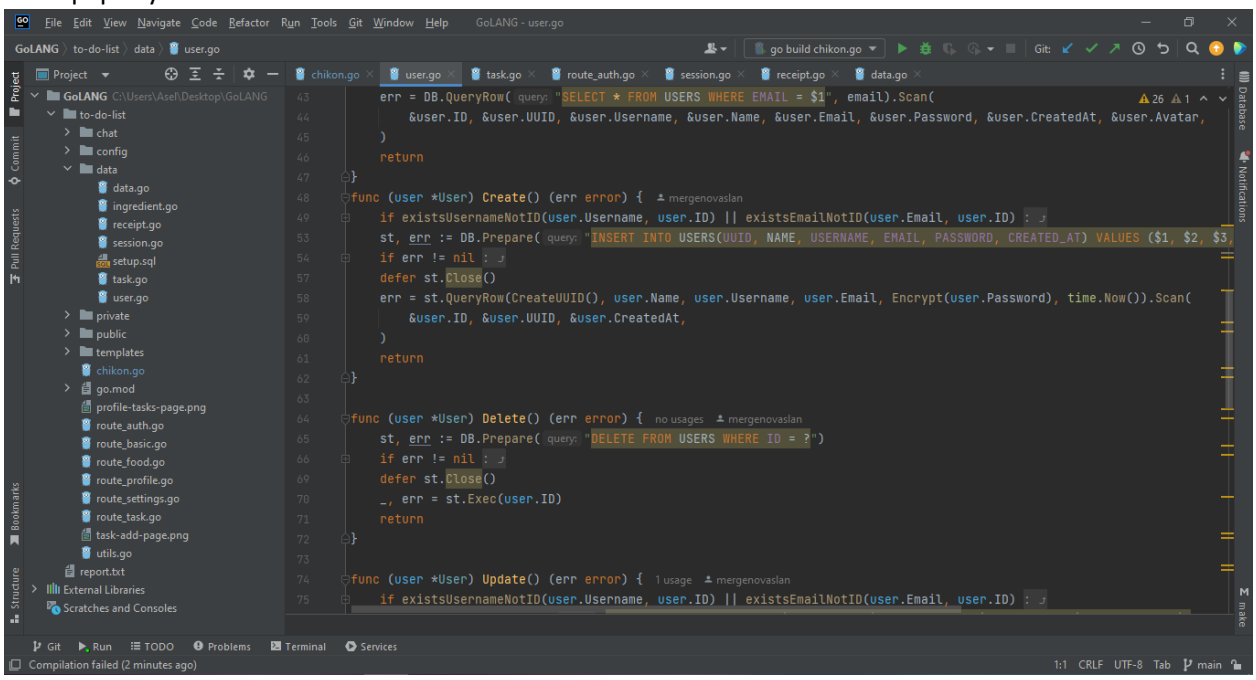
11. Run through the terminal by typing go run . then just go to the site at the address and check



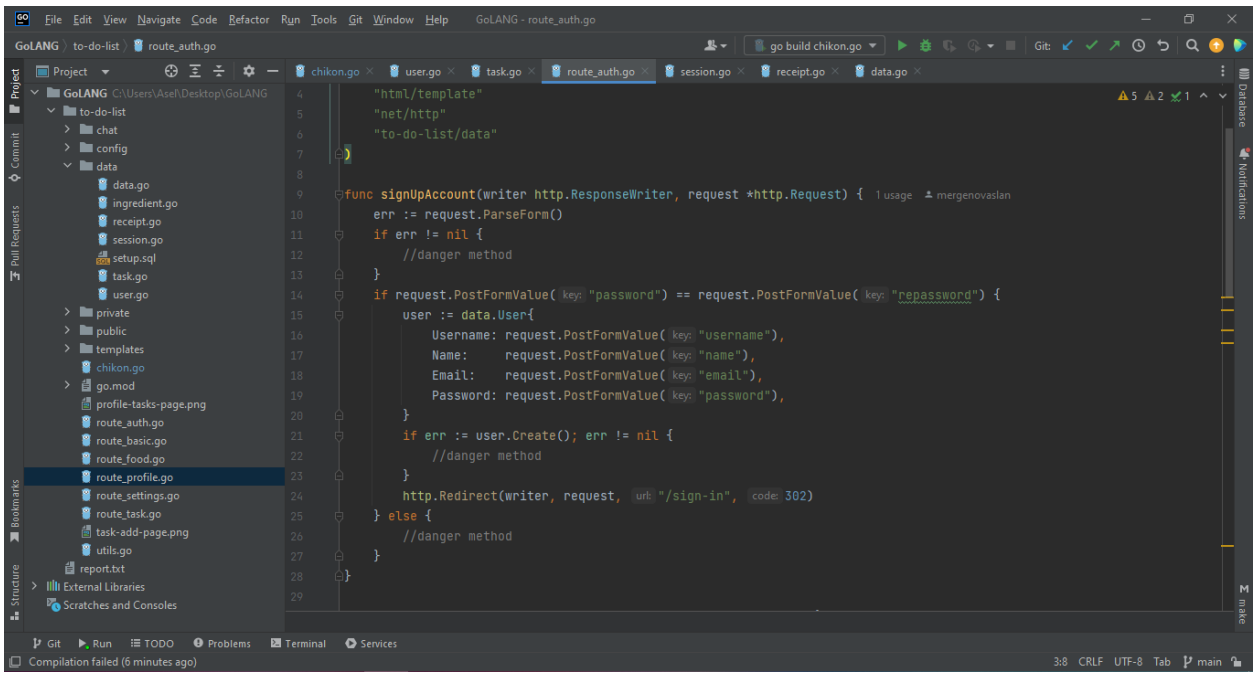
12. Handler functions written in the main file



13. The function to create with the necessary queries that fills in the data in the database according to the sql query insert into

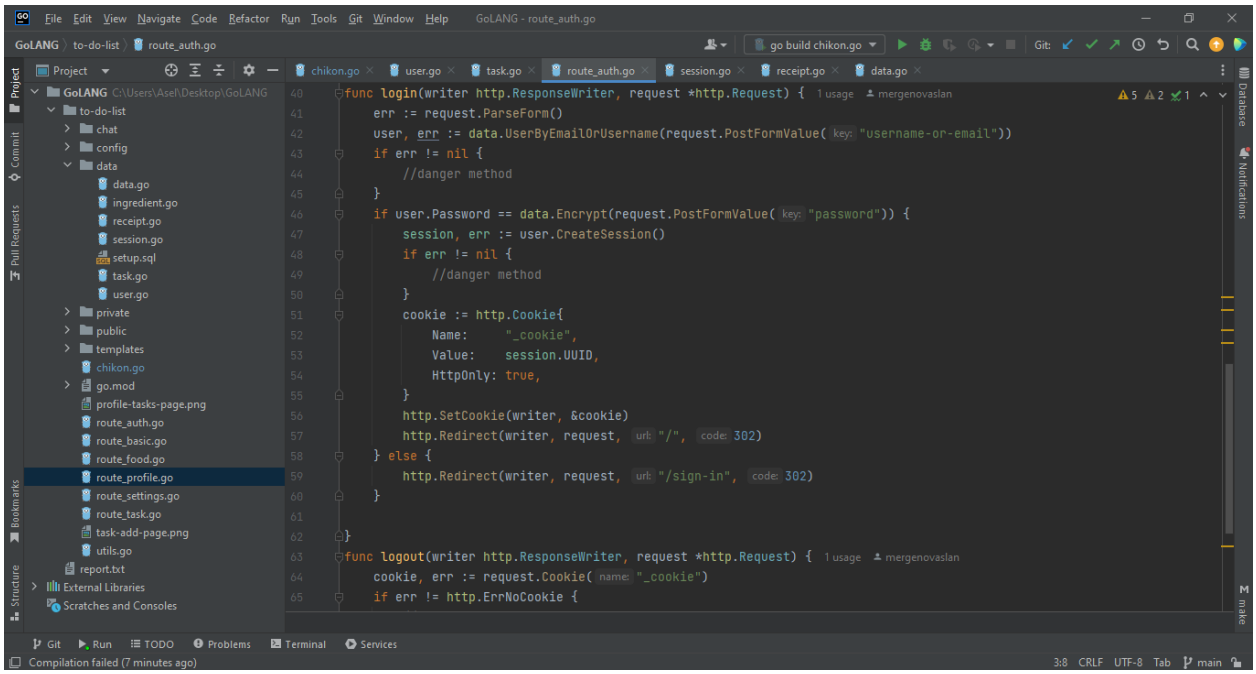


14.. When the user submits the registration form, we receive a POST request containing his information. We check the email or nickname by password,



```
    "html/template"
    "net/http"
    "to-do-list/data"
)

func signUpAccount(writer http.ResponseWriter, request *http.Request) {  1 usage  mergenovaslan
    err := request.ParseForm()
    if err != nil {
        //danger method
    }
    if request.PostFormValue( key: "password") == request.PostFormValue( key: "repassword") {
        user := data.User{
            Username: request.PostFormValue( key: "username"),
            Name:     request.PostFormValue( key: "name"),
            Email:    request.PostFormValue( key: "email"),
            Password: request.PostFormValue( key: "password"),
        }
        if err := user.Create(); err != nil {
            //danger method
        }
        http.Redirect(writer, request,  url: "/sign-in",  code: 302)
    } else {
        //danger method
    }
}
```



```
func login(writer http.ResponseWriter, request *http.Request) {  1 usage  mergenovaslan
    err := request.ParseForm()
    user, err := data.UserByEmailOrUsername(request.PostFormValue( key: "username-or-email"))
    if err != nil {
        //danger method
    }
    if user.Password == data.Encrypt(request.PostFormValue( key: "password")) {
        session, err := user.CreateSession()
        if err != nil {
            //danger method
        }
        cookie := http.Cookie{
            Name:     "_cookie",
            Value:    session.UUID,
            HttpOnly: true,
        }
        http.SetCookie(writer, &cookie)
        http.Redirect(writer, request,  url: "/",  code: 302)
    } else {
        http.Redirect(writer, request,  url: "/sign-in",  code: 302)
    }

}
func logout(writer http.ResponseWriter, request *http.Request) {  1 usage  mergenovaslan
    cookie, err := request.Cookie( name: "_cookie")
    if err != http.ErrNoCookie {
```