GOLANG Midterm1 report

Ayaulym Parmash	200103265
Myrzaliyeva Zhibek	200103281
Amangeldi Diana	200103455
Zhanna Kanal	200103157

INTRODUCTION TO SERVICE

We created a website for the NIKE store. It contains html, css of registration, authorization, main page, product page by providing the required images, information about the product.We tried to make a convenient user interface with eye-catching design, readable data. Also, a database was created with different tables such as Product, customer, admin, basket, category, comment, product, rating. The website contains functionalities like registration, authorization and searching for the items. In the registration page a customer can create his own account for the future purposes and in the login page he can sign in and open his page. It is useful because this is a store's website where people can buy and order something. He can add a product to his basket and at any time order it. Along with this, searching page makes it easier to search for needed products just by the name or even by the one word of the name.

TEAM MEMBERS (DESCRIPTION/CONTRIBUTION)

Ayaulym Parmash (200103265) :

GO -> establishing Net HTTP framework

GO -> connecting the database MySql to GoLang

GO -> making the authorization part

GO -> making the search process

GO -> managing the GitHub

Zhibek Myrzaliyeva (200103281) :

Database -> creating the database by providing different tables such as customer , product , basket , etc.

Database -> filling the tables with necessary product's data (information) like price , image , description , name , etc.

ERD -> creating the schema of database in order to model the data stored in database

GO -> making the registration part

Diana Amangeldi (200103455) :

Figma -> creating the design of login , registration pages by adding different attributes

HTML -> converting a prepared main page design into html , css

ERD -> creating the schema of database in order to model the data stored in database

GO -> filling the google form

GO -> connecting the HTML/CSS to GoLang

Zhanna Kanal (200103157) :

Figma -> creating the design of main page by adding different attributes

HTML -> converting a prepared login , registration pages design into html , css

GO -> writing the reports

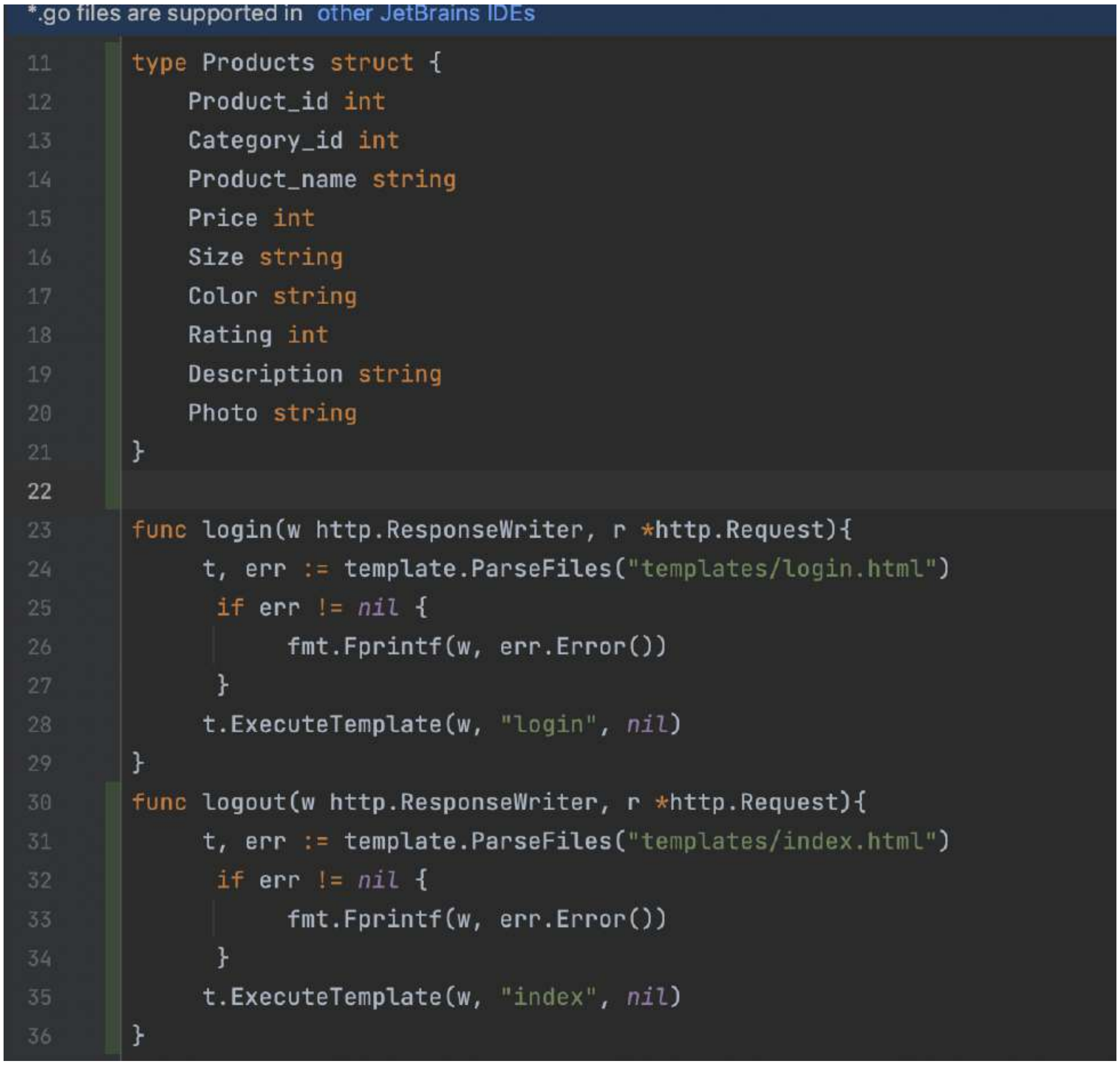
GO -> making the search process

HOW TO RUN

To run the project you need to open in the git module_go/src and type "go run any_source.go".

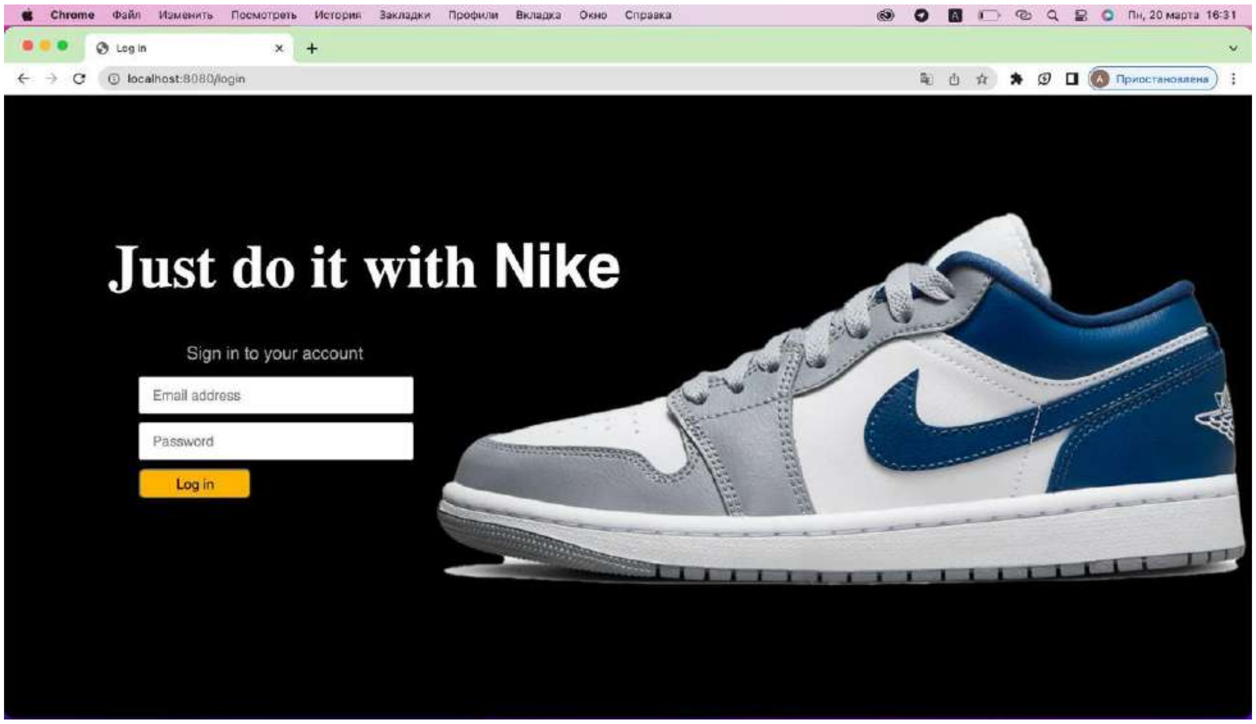
EXPLANATION

Here the code starts. We have a struct named Products for all our products in the database and it contains variables as in the database. The next function is the login function which is used to open the “login.html” template file. The function ParseFiles opens this file. And the function logout does the same thing. But it opens the main page if a user wants to logout.

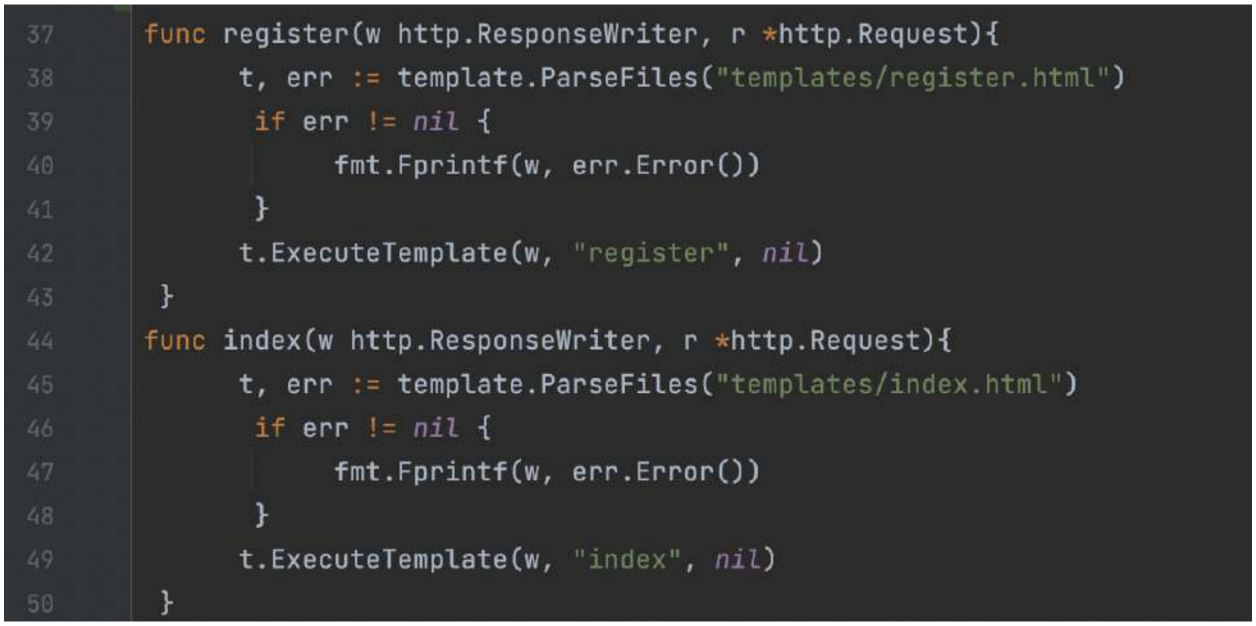


```go
type Products struct {
    Product_id int
    Category_id int
    Product_name string
    Price int
    Size string
    Color string
    Rating int
    Description string
    Photo string
}

func login(w http.ResponseWriter, r *http.Request){
     t, err := template.ParseFiles("templates/login.html")
      if err != nil {
           fmt.Fprintf(w, err.Error())
      }
     t.ExecuteTemplate(w, "login", nil)
}
func logout(w http.ResponseWriter, r *http.Request){
     t, err := template.ParseFiles("templates/index.html")
      if err != nil {
           fmt.Fprintf(w, err.Error())
      }
     t.ExecuteTemplate(w, "index", nil)
}
```

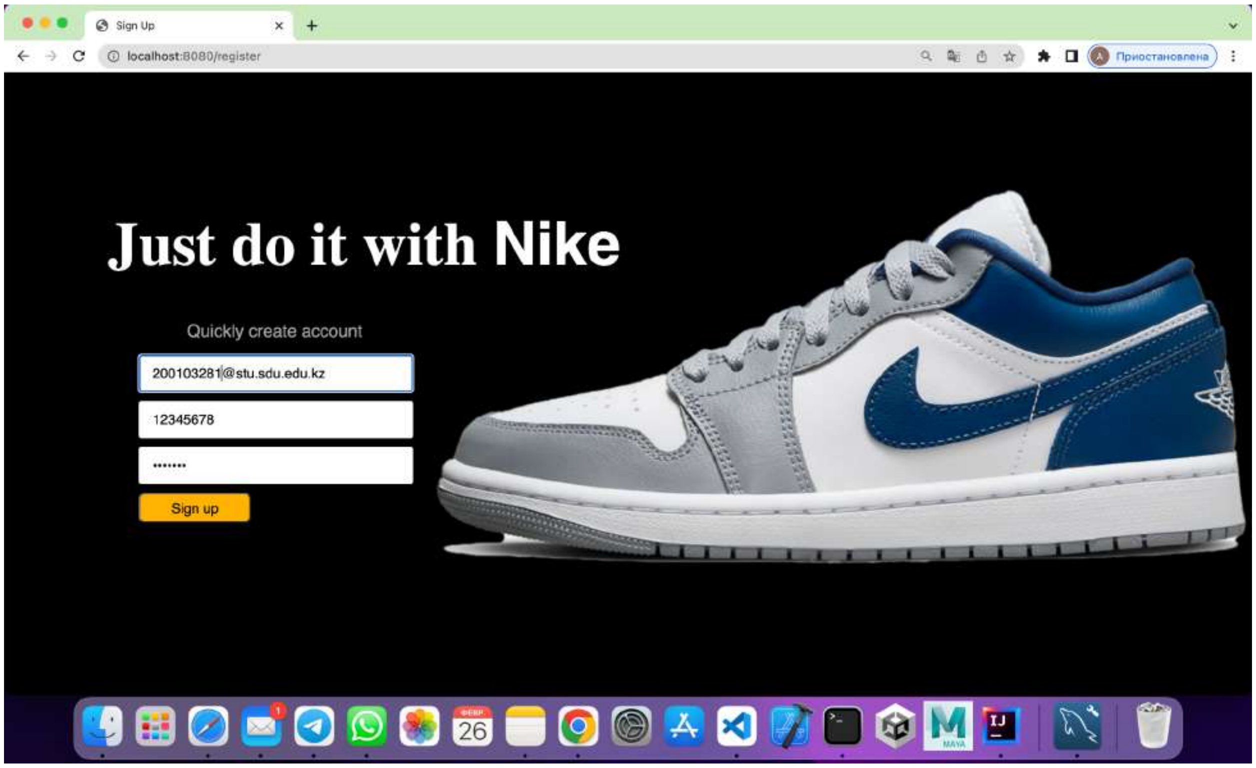
M



The next functions are register and index which open the register template and main page templates respectively with the ParseFiles method.



```go
func register(w http.ResponseWriter, r *http.Request){
	t, err := template.ParseFiles("templates/register.html")
	 if err != nil {
		fmt.Fprintf(w, err.Error())
	 }
	t.ExecuteTemplate(w, "register", nil)
 }
func index(w http.ResponseWriter, r *http.Request){
	t, err := template.ParseFiles("templates/index.html")
	 if err != nil {
		fmt.Fprintf(w, err.Error())
	 }
	t.ExecuteTemplate(w, "index", nil)
 }
```

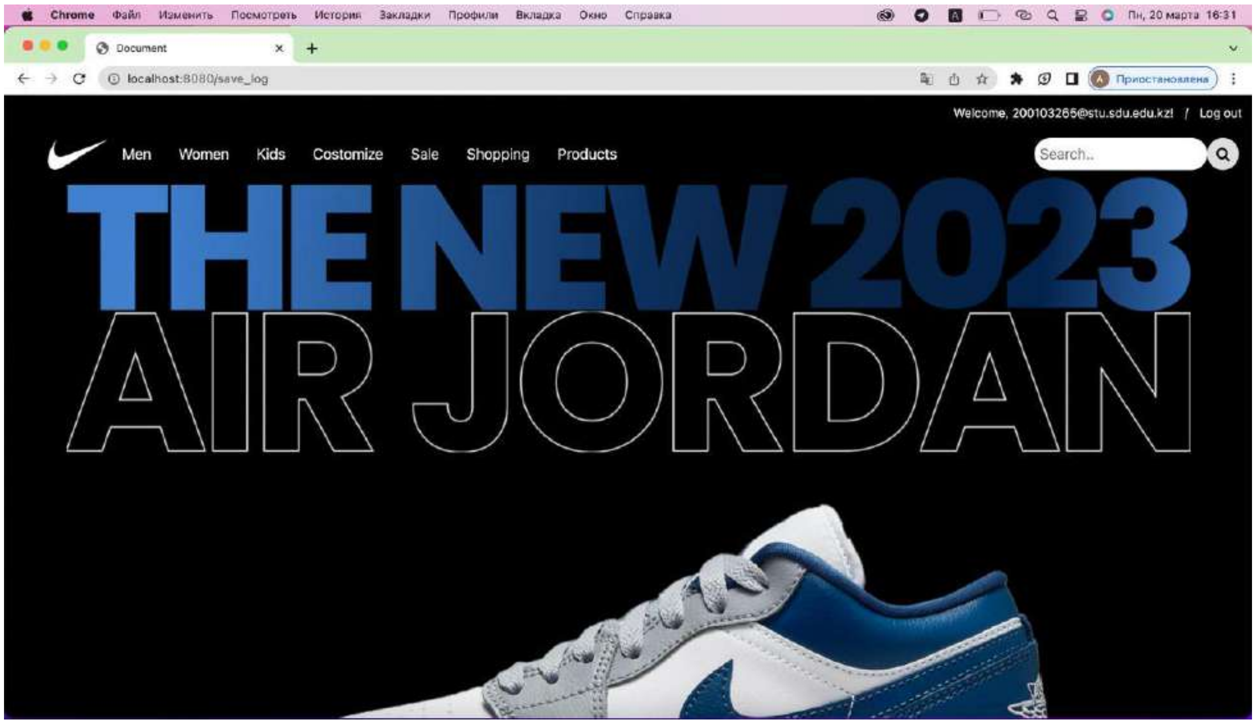


Quickly create account

200103281@stu.sdu.edu.kz

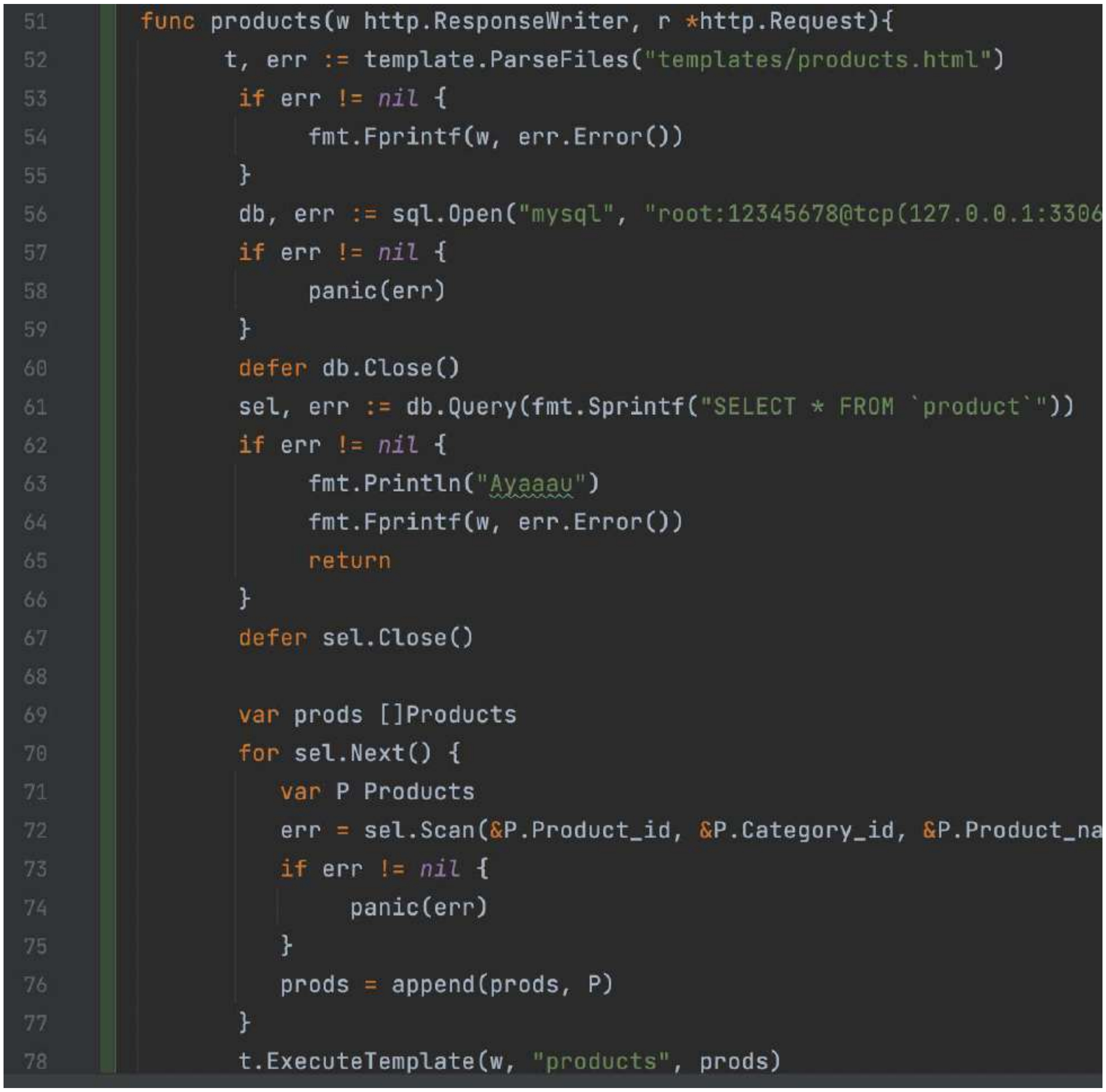
12345678

•••••••

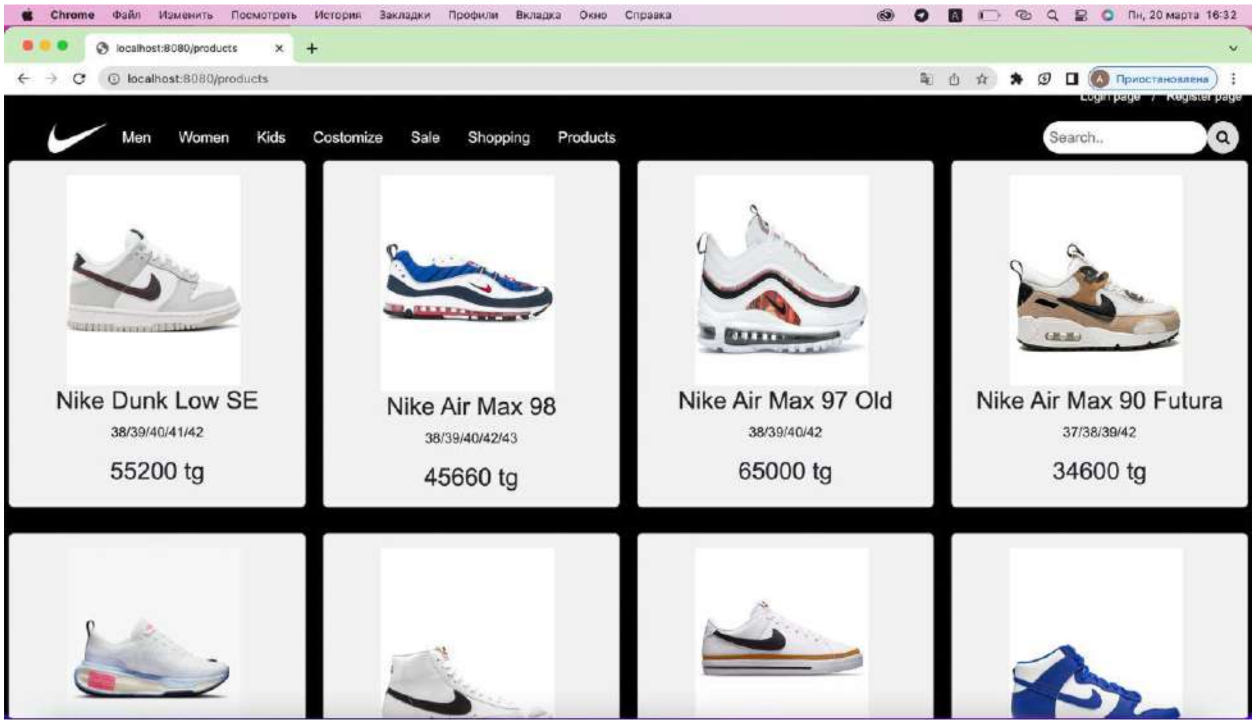


Welcome, 200103265@stu.sdu.edu.kz! / Log out

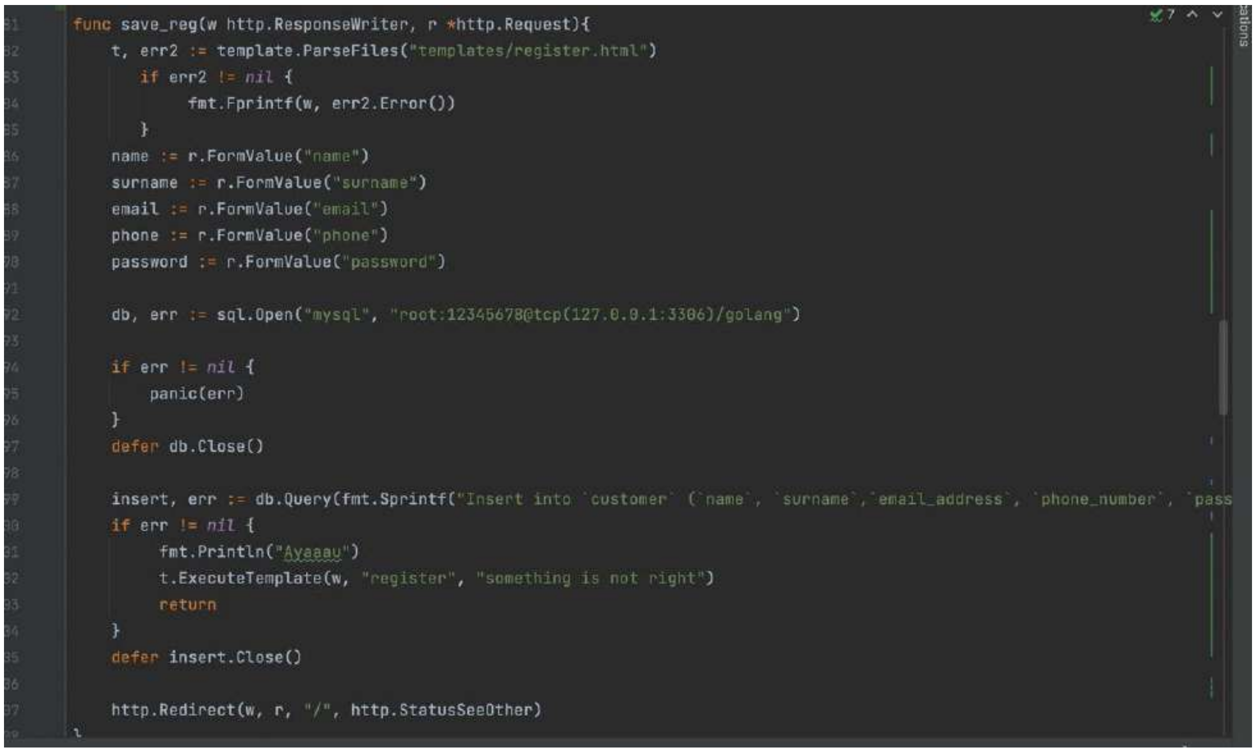
The piece of code below opens a products template and makes a connection with the database. After establishing a connection with the help of query "SELECT * FROM `product`" it sends this query to the database and saves all the returned data to sel var. After that we opened an array of Products where every product and its information will be saved separately. If sel has one more row of data it will be processed and scanned every column which will be appended to the array we opened before. Then, t.ExecuteTemplate(w, "products", prods) code will send this array to the template where it will be shown.



```
func products(w http.ResponseWriter, r *http.Request){
      t, err := template.ParseFiles("templates/products.html")
       if err != nil {
            fmt.Fprintf(w, err.Error())
       }
       db, err := sql.Open("mysql", "root:12345678@tcp(127.0.0.1:3306
       if err != nil {
            panic(err)
       }
       defer db.Close()
       sel, err := db.Query(fmt.Sprintf("SELECT * FROM `product`"))
       if err != nil {
            fmt.Println("Ayaaau")
            fmt.Fprintf(w, err.Error())
            return
       }
       defer sel.Close()

       var prods []Products
       for sel.Next() {
          var P Products
          err = sel.Scan(&P.Product_id, &P.Category_id, &P.Product_na
          if err != nil {
               panic(err)
          }
          prods = append(prods, P)
       }
       t.ExecuteTemplate(w, "products", prods)
```

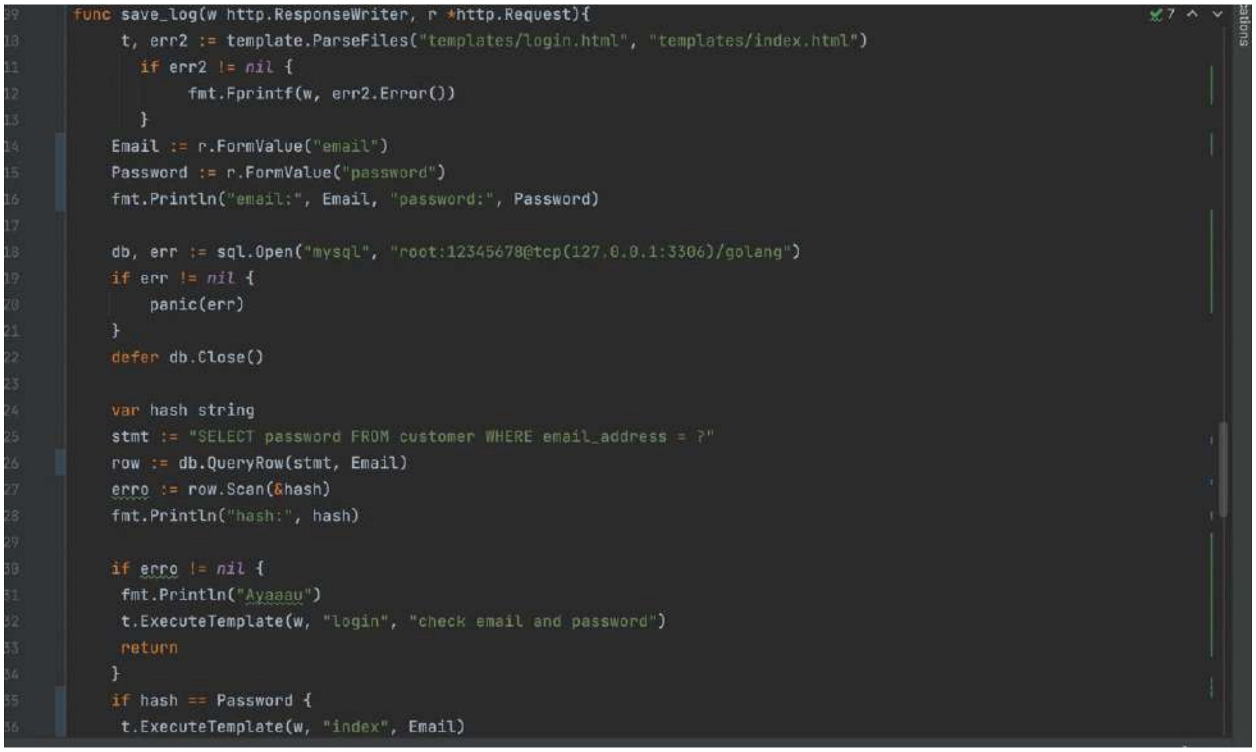


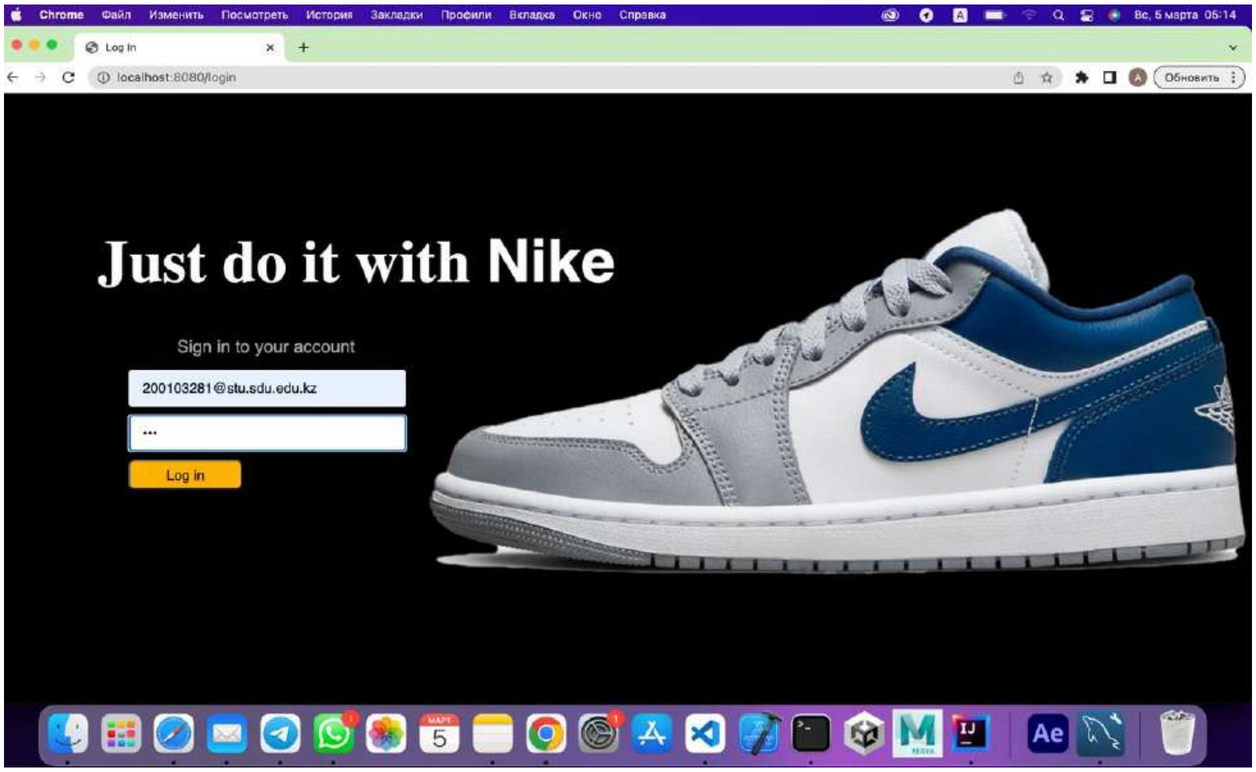
A save_reg function will connect to the template of registration and save all the form values from the template. Then, it will connect to the database and by the query Insert into `customer` (`name`, `surname`,`email_address`, `phone_number`, `password`) Values ('%s', '%s', '%s', '%s', '%s')", name, surname, email, phone, password. It will save all form values to the database. And if registration passes clearly, the main page will be opened. And one point here is that registration will not be passed if there exists the same email address. It will show an error message that email is the same. And we did it by the Unique function in the db.



```
func save_reg(w http.ResponseWriter, r *http.Request){
    t, err2 := template.ParseFiles("templates/register.html")
      if err2 != nil {
          fmt.Fprintf(w, err2.Error())
      }
    name := r.FormValue("name")
    surname := r.FormValue("surname")
    email := r.FormValue("email")
    phone := r.FormValue("phone")
    password := r.FormValue("password")

    db, err := sql.Open("mysql", "root:12345678@tcp(127.0.0.1:3306)/golang")

    if err != nil {
        panic(err)
    }
    defer db.Close()

    insert, err := db.Query(fmt.Sprintf("Insert into `customer` (`name`, `surname`,`email_address`, `phone_number`, `pass
    if err != nil {
         fmt.Println("Ayaaau")
         t.ExecuteTemplate(w, "register", "something is not right")
         return
    }
    defer insert.Close()

    http.Redirect(w, r, "/", http.StatusSeeOther)
}
```

Save_log function takes all form values and by SELECT password FROM customer WHERE email_address = ? query selects all data from the database where an email_address of the customer is equal to the form’s value. Then, it scans the rows where email_address of the customer is equal to the searched and if the Password form value is equal to the value of scanned password from the database it sends the email to the main page and a user will not see register or login buttons, otherwise if the login is not passed then it will show an error message.


```
func save_log(w http.ResponseWriter, r *http.Request){
    t, err2 := template.ParseFiles("templates/login.html", "templates/index.html")
      if err2 != nil {
           fmt.Fprintf(w, err2.Error())
      }
   Email := r.FormValue("email")
   Password := r.FormValue("password")
   fmt.Println("email:", Email, "password:", Password)

   db, err := sql.Open("mysql", "root:12345678@tcp(127.0.0.1:3306)/golang")
   if err != nil {
       panic(err)
   }
   defer db.Close()

   var hash string
   stmt := "SELECT password FROM customer WHERE email_address = ?"
   row := db.QueryRow(stmt, Email)
   erro := row.Scan(&hash)
   fmt.Println("hash:", hash)

   if erro != nil {
    fmt.Println("Ayaaau")
    t.ExecuteTemplate(w, "login", "check email and password")
    return
   }
   if hash == Password {
    t.ExecuteTemplate(w, "index", Email)
```


Chrome Файл Изменить Посмотреть История Закладки Профили Вкладка Окно Справка
Вс, 5 марта 05:14
Log In
localhost:8080/login
Обновить

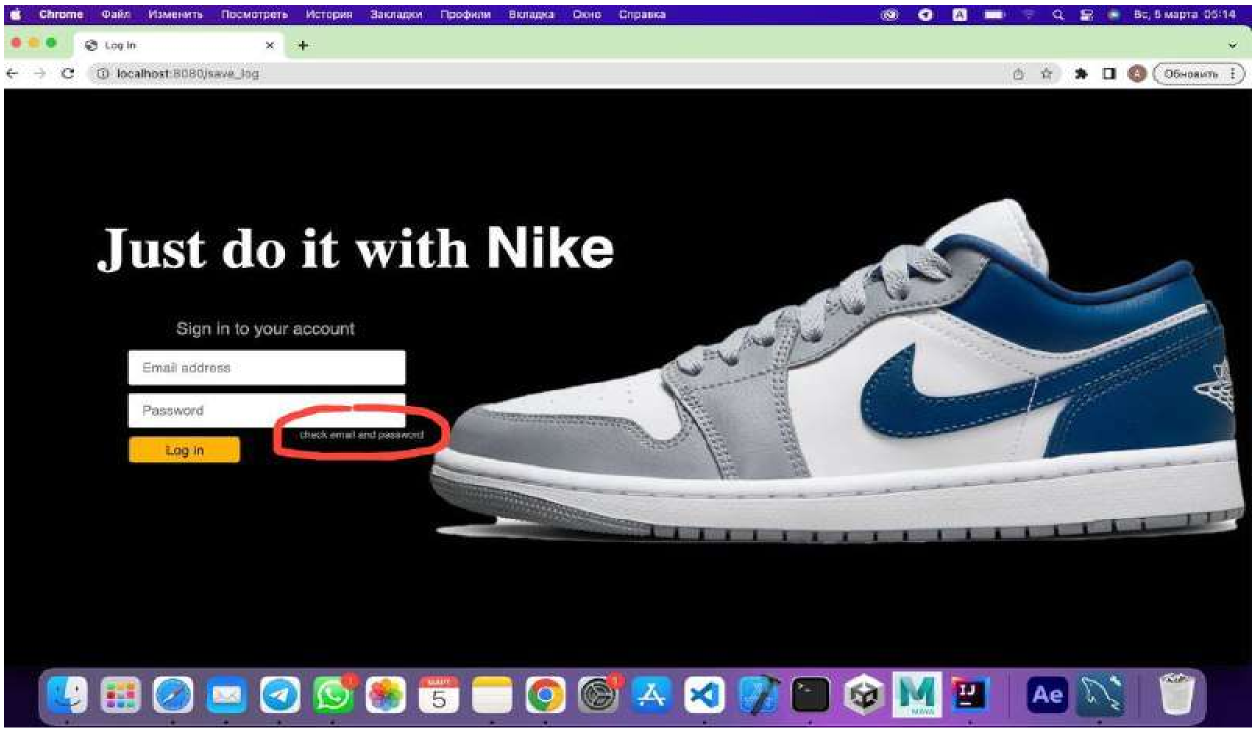
Just do it with Nike

Sign in to your account

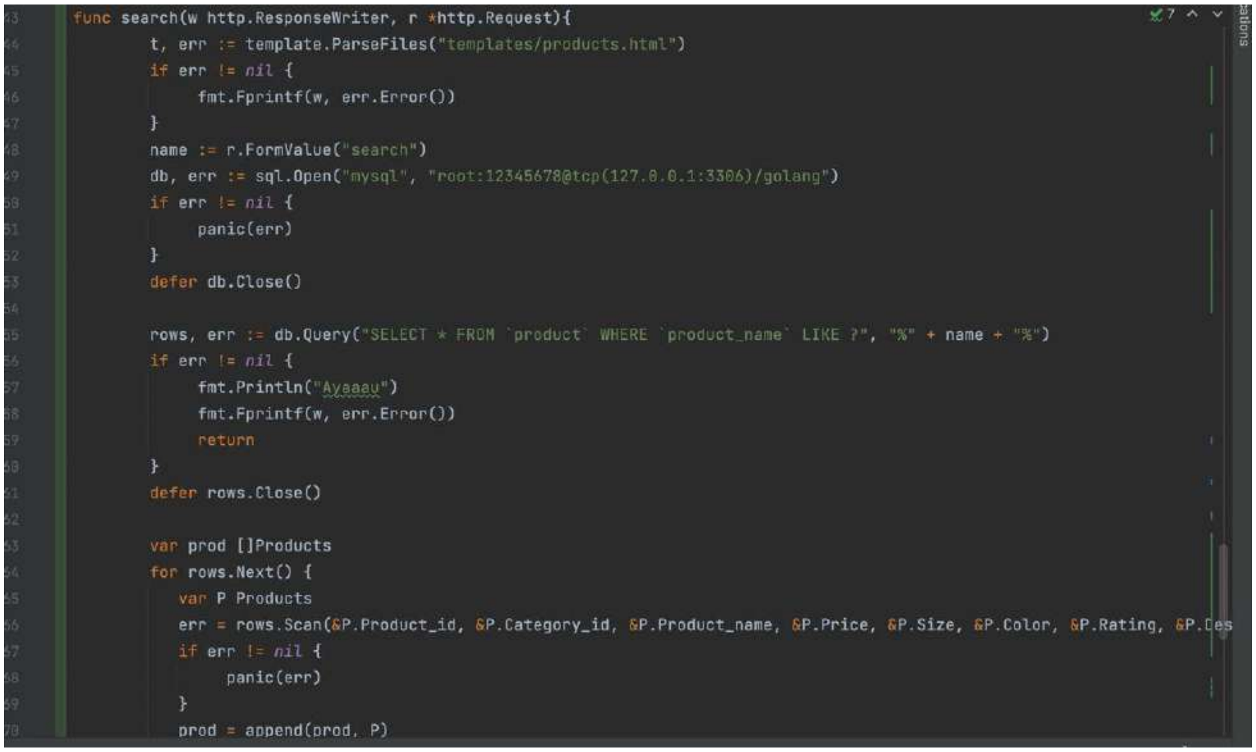
200103281@stu.sdu.edu.kz

•••

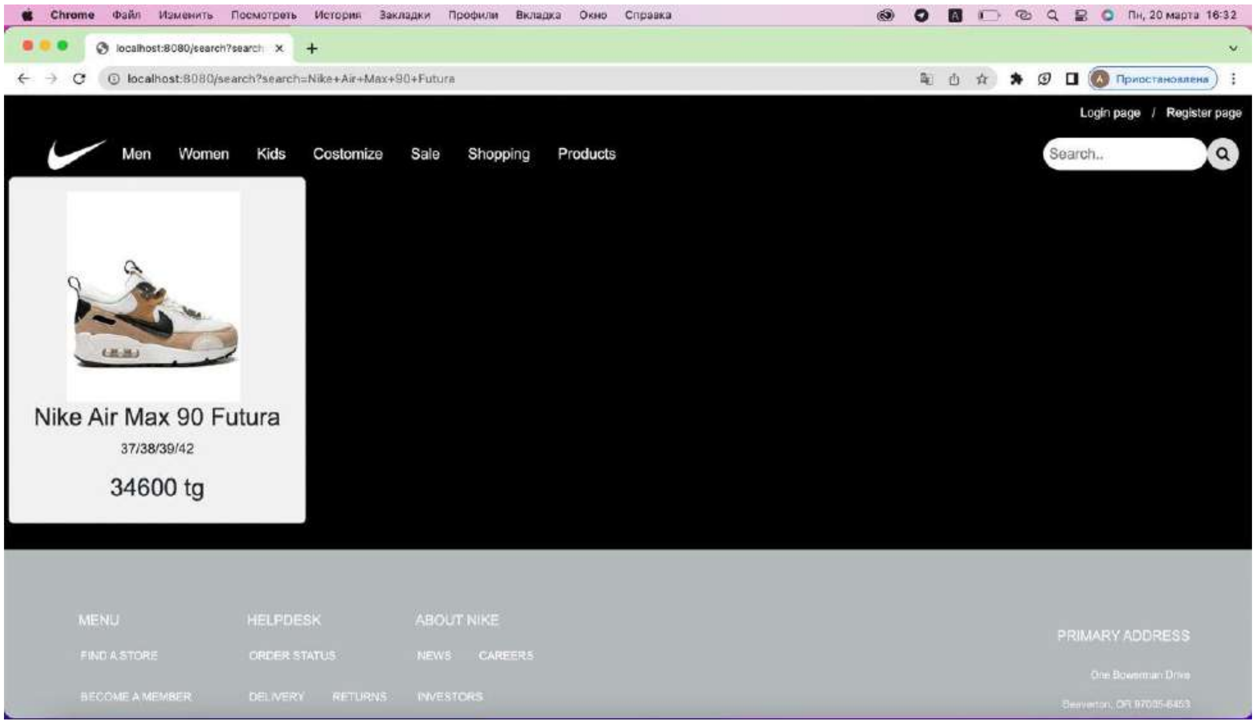
Log in

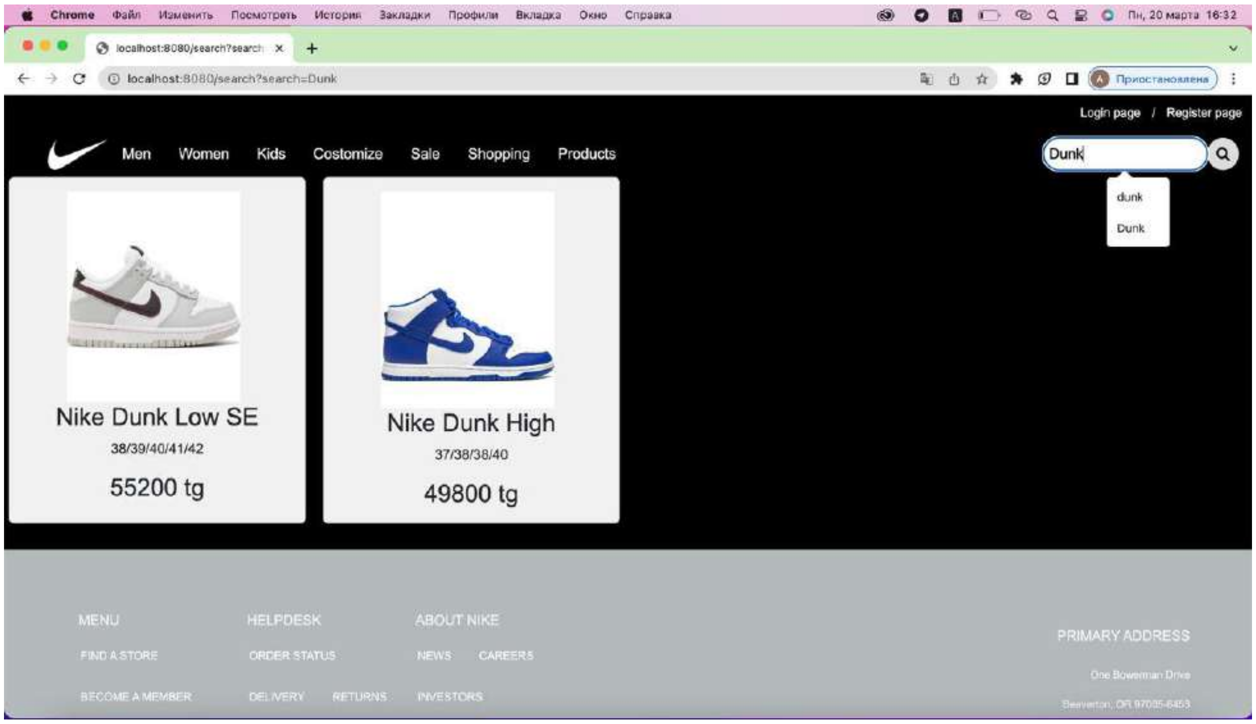


The last function is search. Here it takes a value of the form and by adding it to the SELECT * FROM `product` WHERE `product_name` LIKE ?", "%" + name + "%" query in the place of ? it will select just those rows where the name contains the searched value. And, here is the point that you don't need to search the exact name of a product, just searching by a word it contains is enough.

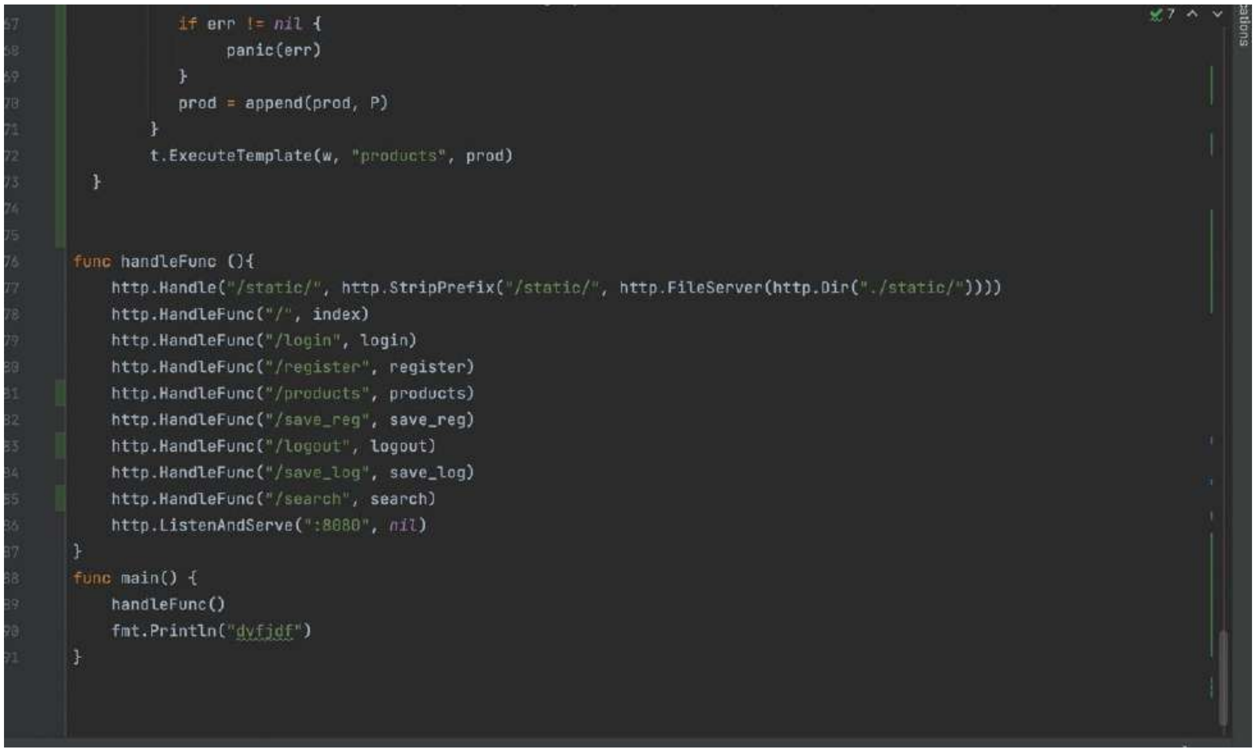


```go
func search(w http.ResponseWriter, r *http.Request){
        t, err := template.ParseFiles("templates/products.html")
        if err != nil {
            fmt.Fprintf(w, err.Error())
        }
        name := r.FormValue("search")
        db, err := sql.Open("mysql", "root:12345678@tcp(127.0.0.1:3306)/golang")
        if err != nil {
            panic(err)
        }
        defer db.Close()

        rows, err := db.Query("SELECT * FROM `product` WHERE `product_name` LIKE ?", "%" + name + "%")
        if err != nil {
            fmt.Println("Ayaaau")
            fmt.Fprintf(w, err.Error())
            return
        }
        defer rows.Close()

        var prod []Products
        for rows.Next() {
           var P Products
           err = rows.Scan(&P.Product_id, &P.Category_id, &P.Product_name, &P.Price, &P.Size, &P.Color, &P.Rating, &P.Des
           if err != nil {
               panic(err)
           }
           prod = append(prod, P)
```





The main part of the code is this. Here it bundles all the http functions and routes to the templates. HTTP.ListenAndServe method is needed to run all the code in the web and connect to the web.



```
            if err != nil {
                panic(err)
            }
            prod = append(prod, P)
        }
        t.ExecuteTemplate(w, "products", prod)
    }


func handleFunc (){
    http.Handle("/static/", http.StripPrefix("/static/", http.FileServer(http.Dir("./static/"))))
    http.HandleFunc("/", index)
    http.HandleFunc("/login", login)
    http.HandleFunc("/register", register)
    http.HandleFunc("/products", products)
    http.HandleFunc("/save_reg", save_reg)
    http.HandleFunc("/logout", logout)
    http.HandleFunc("/save_log", save_log)
    http.HandleFunc("/search", search)
    http.ListenAndServe(":8080", nil)
}
func main() {
    handleFunc()
    fmt.Println("dvfjdf")
}
```

REFERENCES

GitHub

https://github.com/Ayaulym2003/module_go.git

Figma

https://www.figma.com/file/YhZjlOdDR6O4gBcac3ZsjT/Untitled?node-id=0%3A1&t=KjqTEdoJwlm2hhBi-1

ERD

https://drive.google.com/file/d/1iXQ6nC5RKluFXOrShZJRbNJAeN8vXxGp/view?usp=sharing

+++++