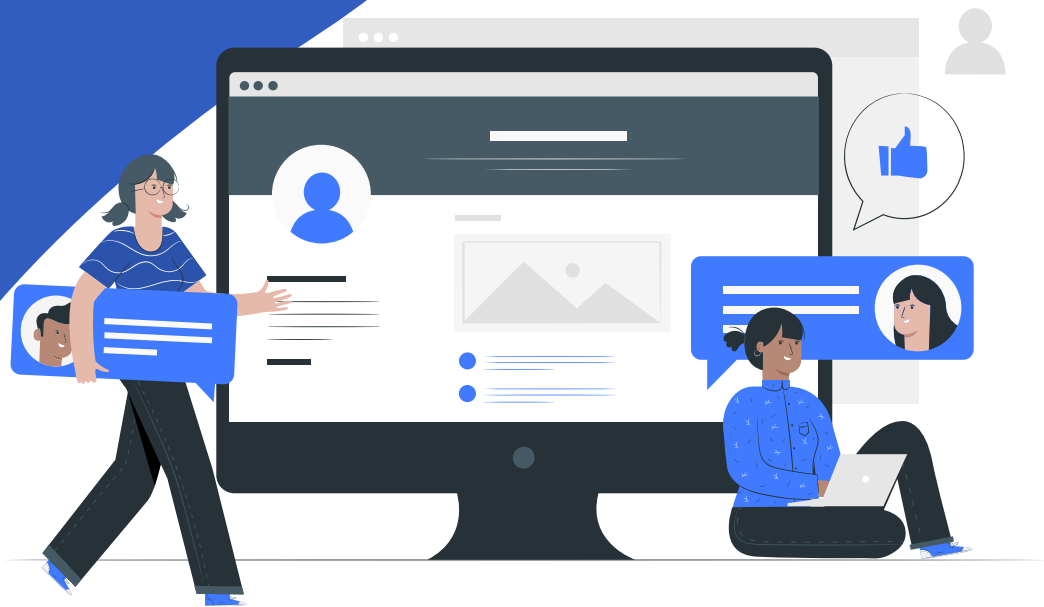


# Midterm

## SDU-Store

Beksultan Yerzhanov 200103513

Darkhan Tashim 200103208



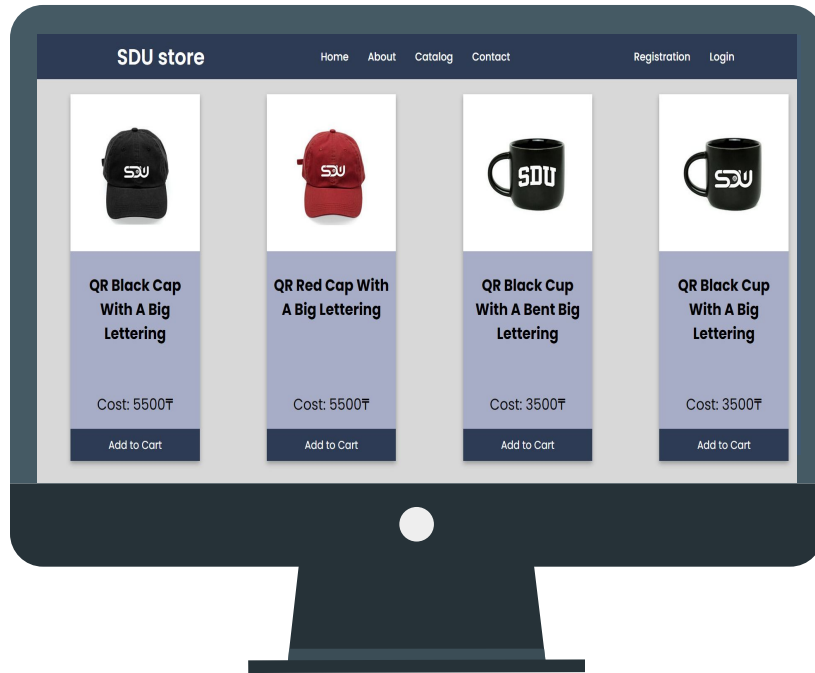


## **Introduction to the service**

**Our idea was  
to create a  
website for our  
university clothing  
store**

**It will be much  
easier with  
our website to  
buy and sell  
exclusive  
clothes**





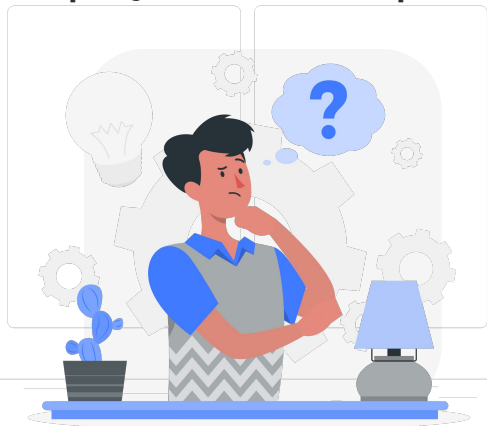
# What do we have now

In the future, we will continue to improve our site, but for now we have a registration and authorization system and a product catalog with a search.

# About our team

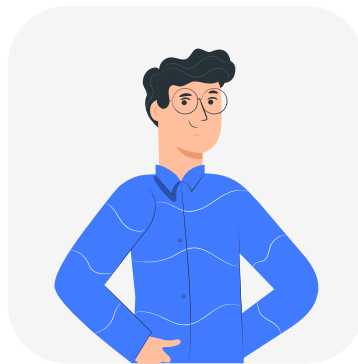
We have 2 persons in our team:

Beksultan and Darkhan. We are 3rd-year students. We divided this midterm project into two parts.



**Beksultan**

Search logic and connecting to the database.



**Darkhan**

Registration and authorization of the site.

# How to run our code?

01

## Install MySQL

You can see [this](#) tutorial to install MySQL to your computer

02

## Clone from Github

By this [link](#) you can clone our project and open by your IDE

03

## Connect database

By this [link](#) you can see how to connect database to the project

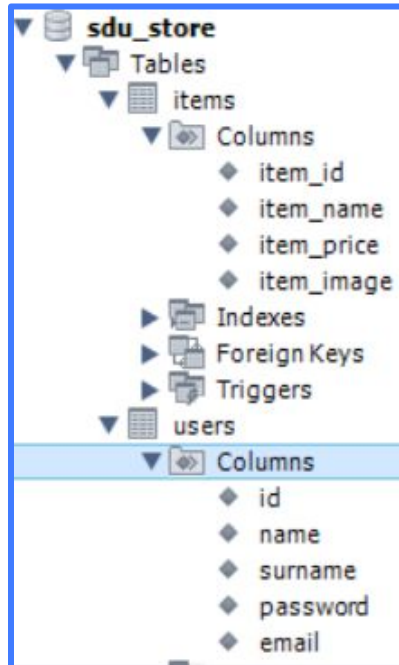
04

## Run the code

If you have any problems, you can contact us

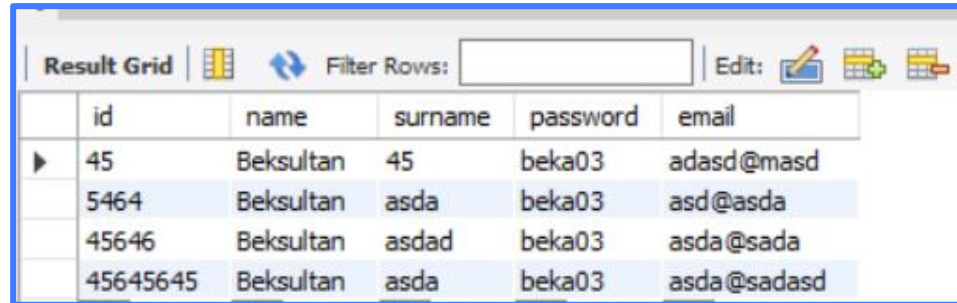
# Database

First of all, for our purposes we have 2 tables in our database:



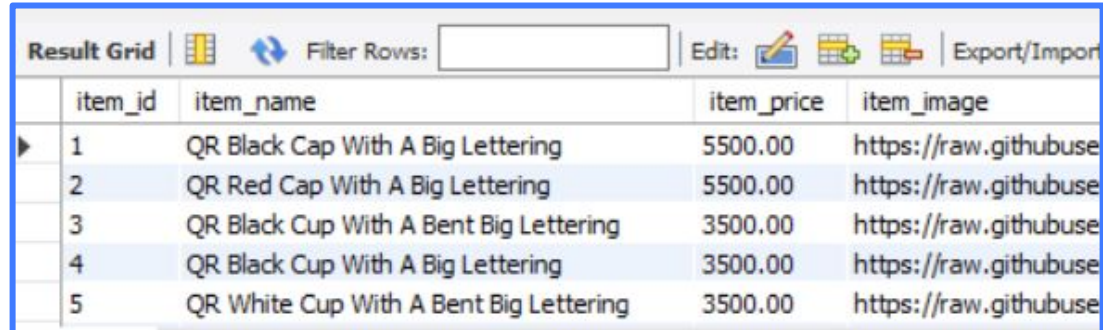
**sdu\_store**

- Tables
  - items
    - Columns
      - item\_id
      - item\_name
      - item\_price
      - item\_image
    - Indexes
    - Foreign Keys
    - Triggers
  - users
    - Columns
      - id
      - name
      - surname
      - password
      - email



Result Grid | Filter Rows: | Edit:

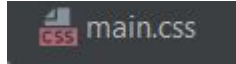
	id	name	surname	password	email
▶	45	Beksultan	45	beka03	adasd@masd
	5464	Beksultan	asda	beka03	asd@asda
	45646	Beksultan	asdad	beka03	asda@sada
	45645645	Beksultan	asda	beka03	asda@sadasd



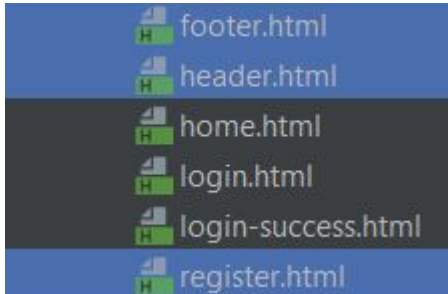
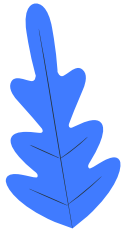
Result Grid | Filter Rows: | Edit: | Export/Import

	item_id	item_name	item_price	item_image
▶	1	QR Black Cap With A Big Lettering	5500.00	https://raw.githubusercontent.com
	2	QR Red Cap With A Big Lettering	5500.00	https://raw.githubusercontent.com
	3	QR Black Cup With A Bent Big Lettering	3500.00	https://raw.githubusercontent.com
	4	QR Black Cup With A Big Lettering	3500.00	https://raw.githubusercontent.com
	5	QR White Cup With A Bent Big Lettering	3500.00	https://raw.githubusercontent.com

# Registration (Frontend) (1)



+

A mockup of the "SDU store" registration page. The page has a dark blue header with the store name and navigation links: Home, About, Catalog, Contact, and Registration. The main content area is light gray and features a registration form with the following fields: ID, Username, Surname, Password, and Email. Each field has a corresponding label and a text input box. Below the fields is a green "Register" button. The footer of the page is dark blue and contains the text "SDU Store. All rights reserved.".



# Registration (Frontend) (2)

## main.css file

```
Main.go x main.css x header.html x register.ht
4 .register-success {
5     width: 1000px;
6     min-height: 100%;
7     padding: 25vh;
8     text-align: center;
9     margin: 0 auto auto auto;
10 }
11
12 .register-form {
13     min-height: 100%;
14     padding-top: 12vh;
15     width: 450px;
16     align-content: center;
17     margin: 0 auto auto auto;
18 }
```

## register.html file

```
register.html x
1  {{ define "register" }}
2  {{ template "header" }}
3
4  <div class="register-form" id="register-form">
5      <form action="/register" method="post">
6          <label for="id">ID:</label>
7          <input type="number" id="id" name="id" required><br>
8
9          <label for="username">Username:</label>
10         <input type="text" id="username" name="username" required><br>
11
12         <label for="surname">Surname:</label>
13         <input type="text" id="surname" name="surname" required><br>
14
15         <label for="password">Password:</label>
16         <input type="password" id="password" name="password" required><br>
17
18         <label for="email">Email:</label>
19         <input type="email" id="email" name="email" required><br>
20
21         <input type="submit" value="Register">
22     </form>
23 </div>
24
25 {{ template "footer" }}
26 {{ end }}
```

# Registration (Backend)

## registerHandler function

## handler


```
func main() {  Darkhan Tashim +1
    store = sessions.NewCookieStore([]byte(os.Getenv("secret")))
    db, err = sql.Open("driverName: "mysql", "dataSourceName")
    if err != nil { err.Error() }
    http.HandleFunc("/home", homeHandler)
    http.HandleFunc("/register", registerHandler)
    http.HandleFunc("/login", loginHandler)
```

```
func registerHandler(w http.ResponseWriter, r *http.Request) { 1 usage  Darkhan Tashim +1
    session, _ := store.Get(r, "session")

    if session.Values["entered"] != nil && session.Values["entered"].(bool) {
        http.Redirect(w, r, "/home", http.StatusSeeOther)
        return
    }

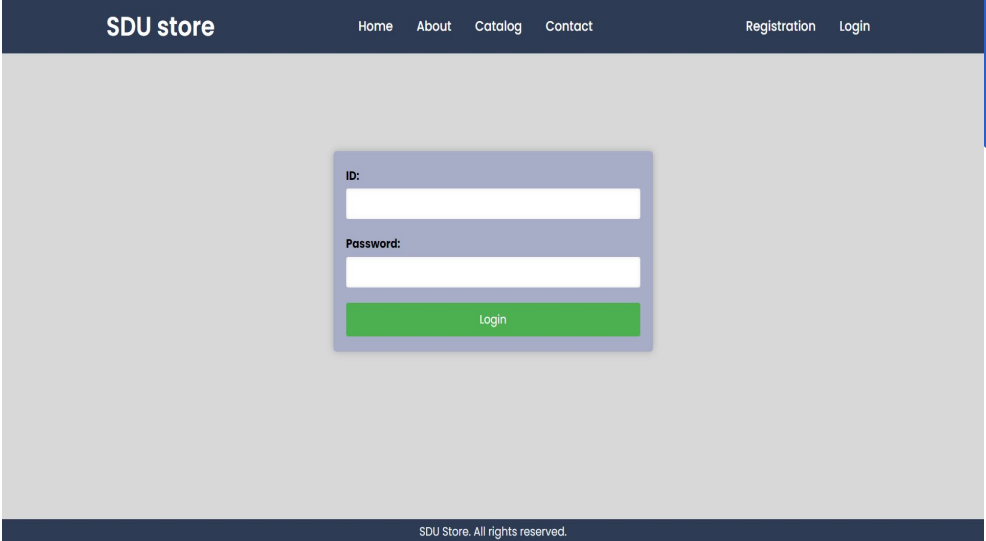
    if r.Method == "GET" {
        t, err := template.ParseFiles("templates/register.html", "templates/header.html", "templates/footer.html")
        if err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
            return
        }
        t.ExecuteTemplate(w, "register", data: nil)
    } else if r.Method == "POST" {
        id, _ := strconv.Atoi(r.FormValue("id"))
        name := r.FormValue("username")
        surname := r.FormValue("surname")
        password := r.FormValue("password")
        email := r.FormValue("email")
        var user User = User{Id: id, Username: name, Surname: surname, Password: password, Email: email}
        query := "INSERT INTO users VALUES (" + strconv.Itoa(user.Id) + ", '" + user.Username +
            "', '" + user.Surname + "', '" + user.Password + "', '" + user.Email + "');"
        insert, err := db.Query(query)
        if err != nil {
            fmt.Println(query)
            panic(err.Error())
        }
    }
}
```

# Authorization (Frontend) (1)

 main.css

+

`<> footer.html`  
`<> header.html`  
`<> home.html`  
`<> login.html`

A mockup of the SDU store website. The header is dark blue with the text 'SDU store' on the left and navigation links 'Home', 'About', 'Catalog', 'Contact', 'Registration', and 'Login' on the right. The main content area is light gray and contains a login form. The form has two input fields labeled 'ID:' and 'Password:', a green 'Login' button, and a footer with the text 'SDU Store. All rights reserved.'

# Authorization (Frontend) (2)

main.css file

```
Main.go x main.css x <> about.ht
170 .login-success {
171     width: 1000px;
172     min-height: 100%;
173     padding: 25vh;
174     text-align: center;
175     margin: 0 auto auto auto;
176 }
177
178 .login-form {
179     padding-top: 200px;
180     min-height: 100%;
181     width: 500px;
182     align-content: center;
183     margin: 0 auto 0 auto;
184 }
```

register.html file

```
Main.go <> login.html x main.css <> about.html <> contact.html <> d
1 {{ define "login" }}
2 {{ template "header" }}
3
4 <div class="login-form">
5     <form action="/login" method="post">
6         <label for="id">ID:</label>
7         <input type="number" id="id" name="id" required><br>
8
9         <label for="password">Password:</label>
10        <input type="password" id="password" name="password" required><br>
11
12        <input type="submit" value="Login">
13    </form>
14 </div>
15
16 {{ template "footer" }}
17 {{ end }}
```

# Authorization (Backend)

## registerHandler function

## handler

```
func main() {  Darkhan Tashim +1
    store = sessions.NewCookieStore([]byte(os.Getenv("SESSION_KEY")))
    db, err = sql.Open("driverName: "mysql", "dataSource: "mysql")
    if err != nil { err.Error() }
    http.HandleFunc("/home", homeHandler)
    http.HandleFunc("/register", registerHandler)
    http.HandleFunc("/login", loginHandler)
    http.HandleFunc("/catalog", catalogHandler)
}
```

```
func loginHandler(w http.ResponseWriter, r *http.Request) { 1 usage  Darkhan Tashim +1
    session, _ := store.Get(r, "session")

    if session.Values["entered"] != nil && session.Values["entered"].(bool) {
        http.Redirect(w, r, "/home", http.StatusSeeOther)
        return
    }

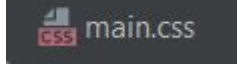
    if r.Method == "GET" {
        t, err := template.ParseFiles("templates/login.html", "templates/header.html", "templates/footer.html")
        if err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
            return
        }
        t.ExecuteTemplate(w, "login", data: nil)
    } else if r.Method == "POST" {
        id, _ := strconv.Atoi(r.FormValue("id"))
        password := r.FormValue("password")

        row := db.QueryRow("SELECT * FROM users WHERE id = ? AND password = ?", id, password)

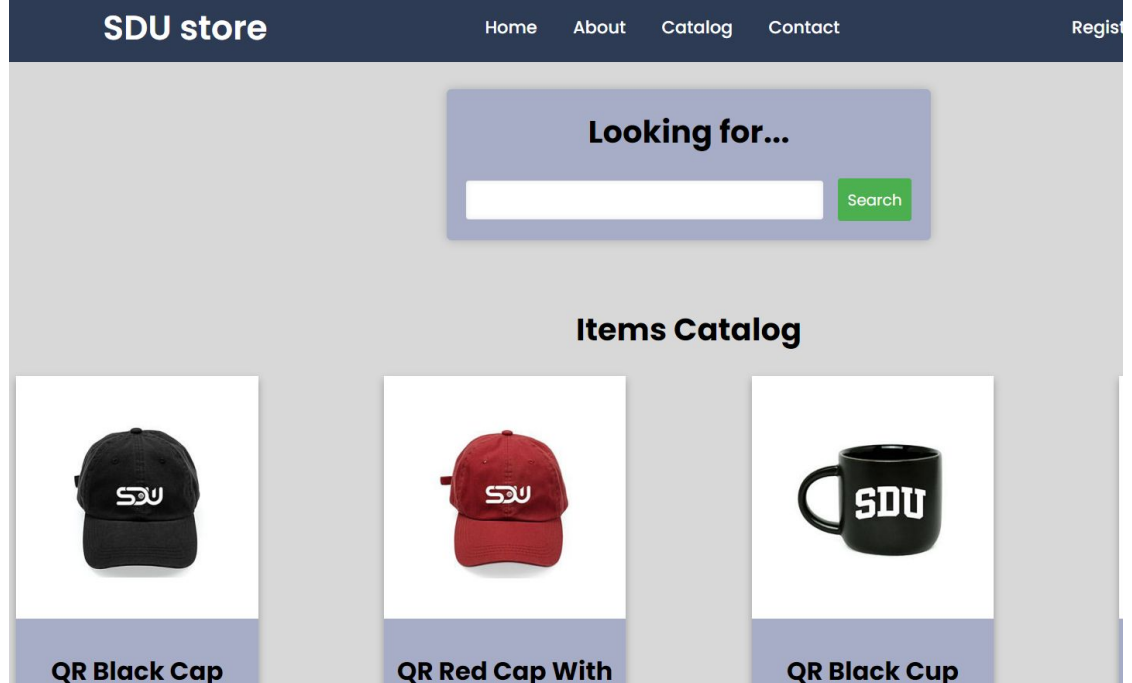
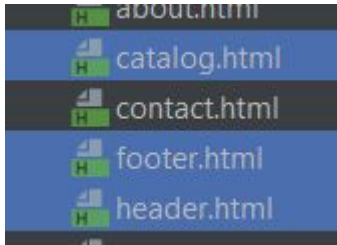
        var userid uint
        var name string
        var surname string
        var userEmail string
        var userPassword string
        err = row.Scan(&userid, &name, &surname, &userPassword, &userEmail)
        if err == sql.ErrNoRows {
            fmt.Println("No user found with the given email and password.")
        } else if err != nil { err } else {

```

# Catalog Page (Frontend) (1)



+





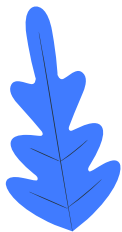
# Catalog Page (Frontend) (2)

main.css file

register.html file

```
register.html x main.css x catalog.html x Main.go x
81
82
83 .serv ul {
84     display: flex;
85     flex-wrap: wrap;
86     padding-left: 0;
87 }
88
89 .serv ul li {
90     list-style: none;
91     flex: 0 0 25%;
92     margin-bottom: 50px;
93 }
94
95 .item-text {
96
97 }
```

```
ster.html x main.css x catalog.html x Main.go x
<br>
<br>
<h1>Items Catalog</h1>
<br>
<div class="serv">
  <ul>
    {{range .}}
    <li>
      <div class="card">
        
        <div class="item-text">
          <h1 class="item-name">{{.ItemName}}</h1>
        </div>
        <br>
        <p class="price">Cost: {{.ItemPrice}}T</p>
        <br>
        <p>
          <button>Add to Cart</button>
        </p>
      </div>
    </li>
    {{end}}
  </ul>
</div>
```



# Catalog Page (Backend)

```
func main() {  Darkhan Tashim +1
    store = sessions.NewCookieStore([]byte(os.Getenv("SESSION_KEY")))
    db, err = sql.Open("driverName: "mysql", "dataSourceName")
    if err != nil { err.Error() }
    http.HandleFunc("/home", homeHandler)
    http.HandleFunc("/register", registerHandler)
    http.HandleFunc("/login", loginHandler)
    http.HandleFunc("/catalog", catalogHandler)
    http.HandleFunc("/about", aboutHandler)
```

```
func catalogHandler(w http.ResponseWriter, r *http.Request) { 1 usage  Yerzhanov Beksu
    if r.Method == "GET" {
        name := r.FormValue("key: "search")
        rows, err := db.Query("SELECT * FROM items WHERE item_name LIKE ?", name+"%")
        if err != nil {
            log.Fatal(err)
        }
        defer rows.Close()

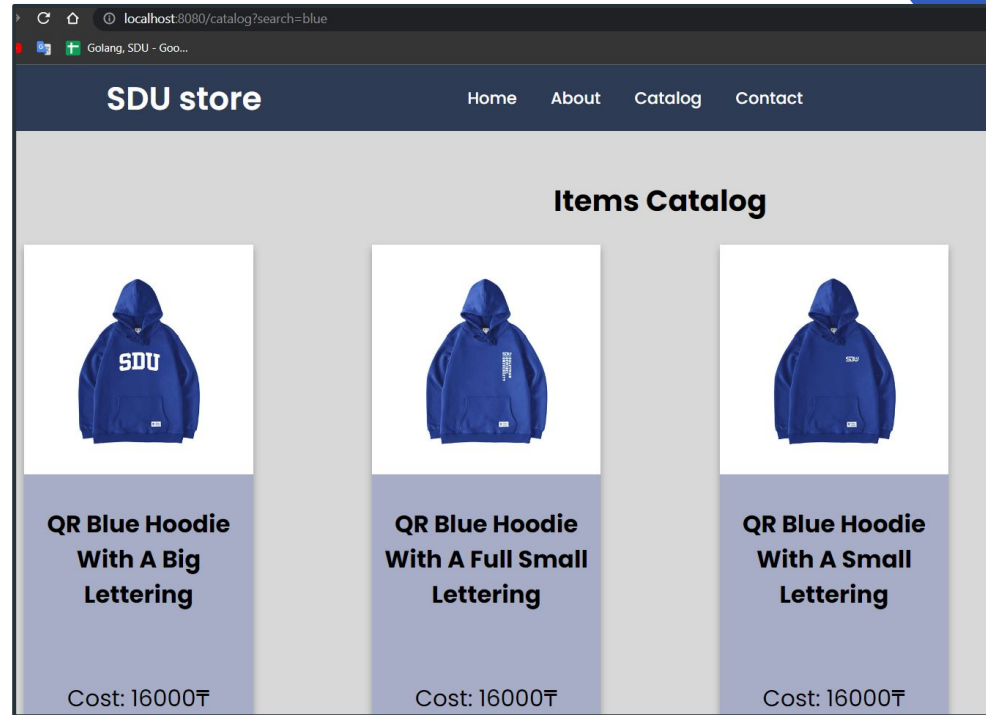
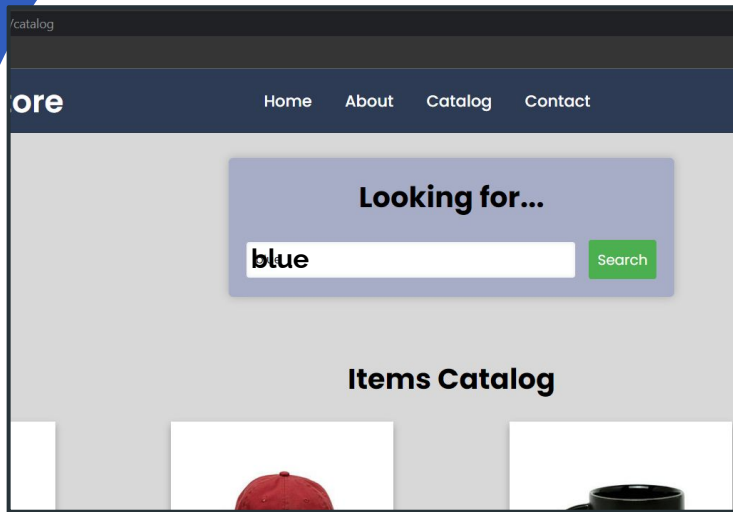
        items := make([]Item, 0)

        for rows.Next() {
            var item Item
            err := rows.Scan(&item.ItemId, &item.ItemName, &item.ItemPrice, &item.ItemImage)
            if err != nil {
                log.Fatal(err)
            }
            items = append(items, item)
        }

        tpl := template.Must(template.ParseFiles("templates/catalog.html"))
        err = tpl.ExecuteTemplate(w, "catalog", items)
        if err != nil {
            log.Fatal(err)
        }
    } else {
    }
```



# Search example



# Search logic in code

```
no catalogHandler(w http.ResponseWriter, r *http.Request) { // usage: go run main.go
    if r.Method == "GET" {
        name := r.FormValue( key: "search")
        rows, err := db.Query( query: "SELECT * FROM items WHERE item_name LIKE ?", args...: "%" + name + "%")
        if err != nil {
            log.Fatal(err)
        }
        defer rows.Close()

        items := make([]Item, 0)

        for rows.Next() {
            var item Item
            err := rows.Scan(&item.ItemId, &item.ItemName, &item.ItemPrice, &item.ItemImage)
            if err != nil {
                log.Fatal(err)
            }
            items = append(items, item)
        }

        tmpl := template.Must(template.ParseFiles( filenames...: "templates/catalog.html", "templates/head
        err = tmpl.ExecuteTemplate(w, name: "catalog", items)
        if err != nil {
            log.Fatal(err)
        }
    }
}
```

request

- Query