

SAVAT

The cheapest prices only in SAVAT

Team Members:

Abdukhamidov Saidazim -

U1810210 Fozilov Bobur - U1810105

Akhmedova Ziyoda - U1810189

Kenjaeva Mekhriniso - U1810285

Tashpulatova Azizabonu - U1810286

GitHub repository:

<https://github.com/iuthub/ip-group-project-sophomores-online-store/tree/master/savat>

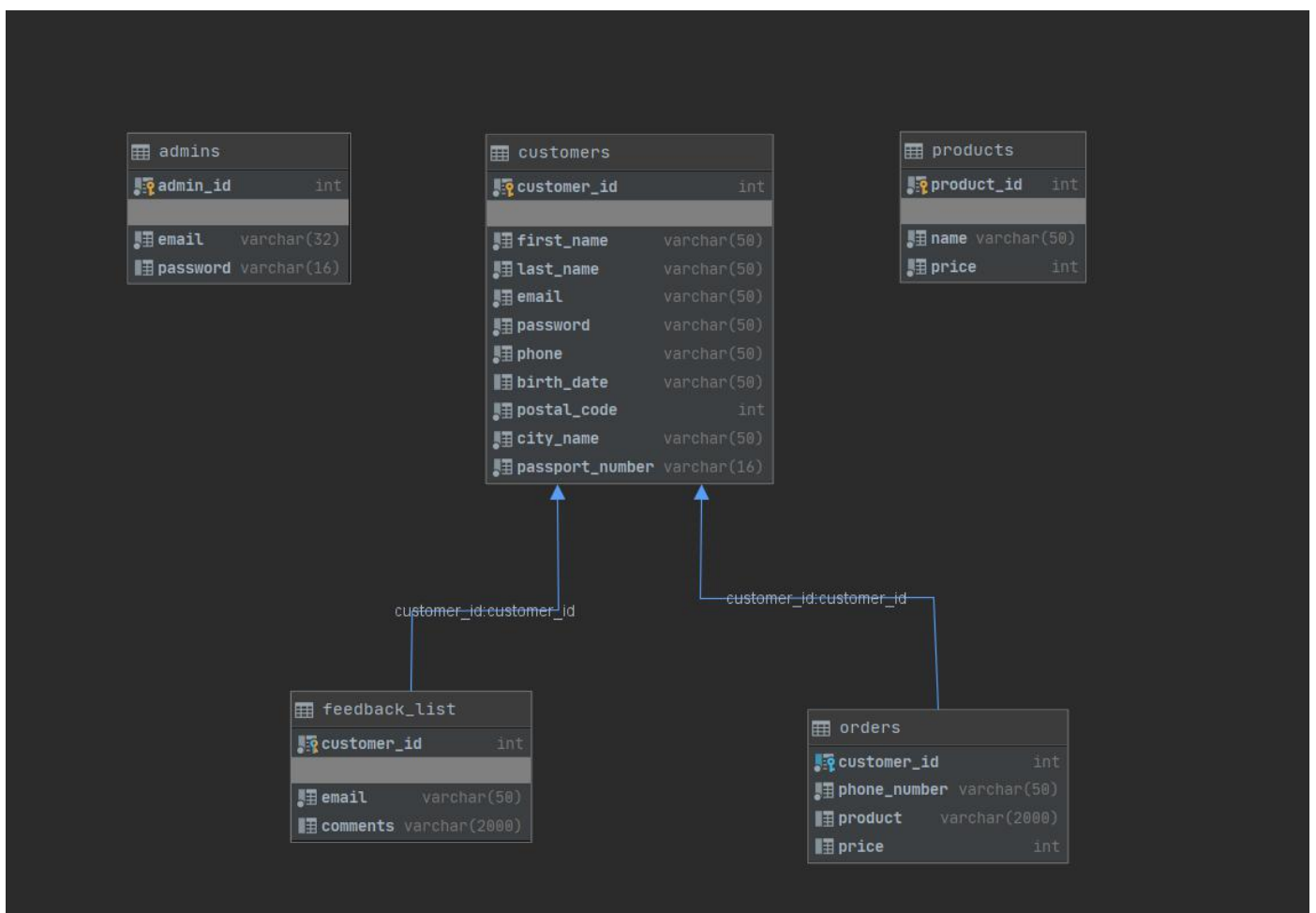
SAVAT is an online market for only eatables as drinks and vegetables. This web application can help people to find food and order them. In this application, we have 2 roles Admins and Customers. Admins can add **products**, manage **orders**, and **feedback**. Customers can search for **products** in our web application and **order** them by inserting just their phone number and send **feedback** to SAVAT managers.

We used **Angular** for making Single Page Application to make experience of user easier and faster. Also, we used **Angular Material** for styled components and **Bootstrap** for making our application responsive. **PHP** is used for only connecting to database and getting data using **REST APIs**.

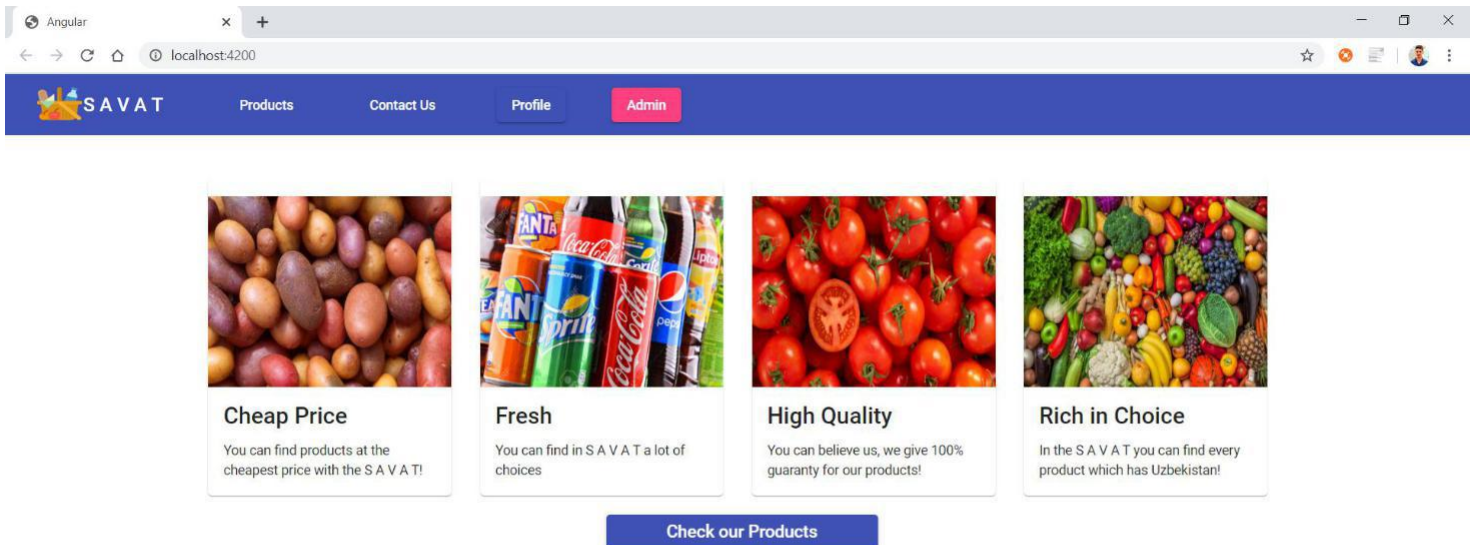
We created 5 tables :

Customers, admins, products, feedback_list and orders.

All registered customers are inserted into **customers** table. Admins are in the **admins** table. **Products** table is for saving products. **Feedback_list** and **orders** tables have relationship with **customers** table their keys are connected to customer_id in the **customers** table.



When user starts using this application, firstly landing page will be opened.



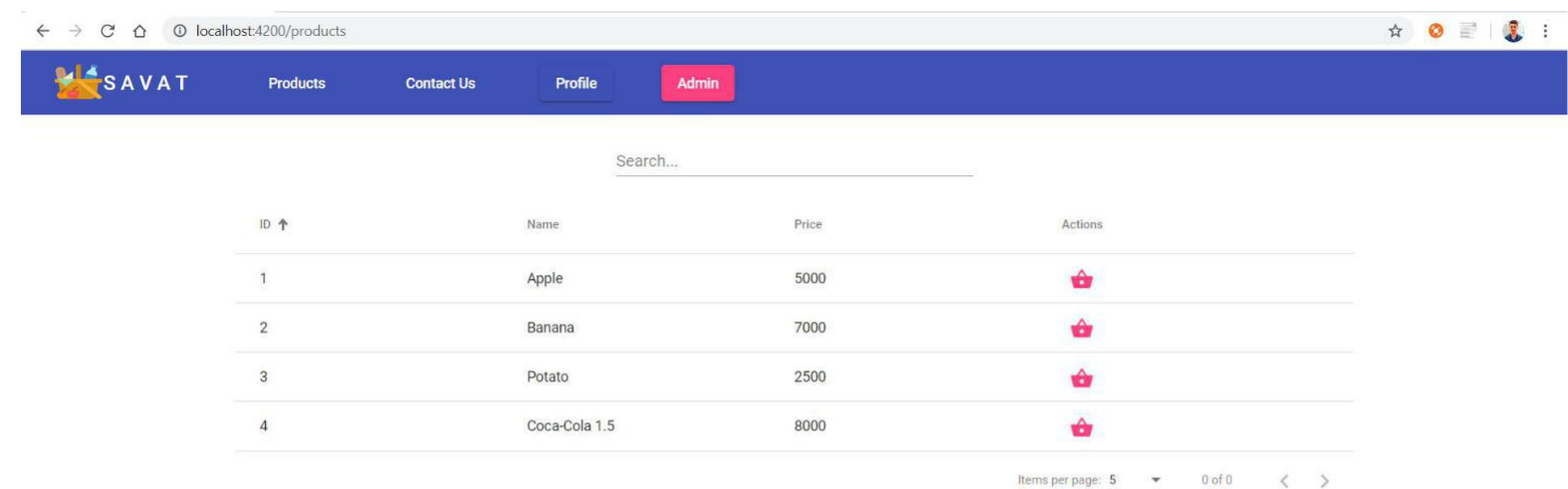
In *angular/src/app/components/home* directory there is home component for implementing this page.

```
home.component.html
1 <div class="container mt-5">
2   <div class="row">
3     <div class="col-lg-3">
4       <mat-card>
5         <mat-card-header>
6         </mat-card-header>
7         
8         <mat-card-title>Cheap Price</mat-card-title>
9         <mat-card-content>
10          <p>
11            You can find products at the cheapest price with the S A V A T!
12          </p>
13        </mat-card-content>
14      </mat-card>
15    </div>
16    <div class="col-lg-3">
17      <mat-card>
18        <mat-card-header>
19        </mat-card-header>
20        
21        <mat-card-title>Fresh</mat-card-title>
22        <mat-card-content>
23          <p>
24            You can find in S A V A T a lot of choices
25          </p>
26        </mat-card-content>
27      </mat-card>
28    </div>
29    <div class="col-lg-3">
30      <mat-card>
31        <mat-card-header>
32        </mat-card-header>
33        
34        <mat-card-title>High Quality</mat-card-title>
35        <mat-card-content>
36          <p>
37            You can believe us, we give 100% guaranty for our products!
38          </p>
39        </mat-card-content>
40      </mat-card>
41    </div>
42    <div class="col-lg-3">
43      <mat-card>
44        <mat-card-header>
45        </mat-card-header>
46        
47        <mat-card-title>Rich in Choice</mat-card-title>
48        <mat-card-content>
49          <p>
50            In the S A V A T you can find every product which has Uzbekistan!
51          </p>
52        </mat-card-content>
53      </mat-card>
54    </div>
55  </div>
56  <button class="m-b col-lg-3" routerLink="products"
57    mat-raised-button color="primary">Check our Products</button>
58 </div>
```

This is our *app-routing.module* All our routerLinks implemented in this file.

```
app-routing.module.ts x
13 const routes: Routes = [
14   {
15     path: '',
16     component: HomeComponent
17   },
18   {
19     path: 'products',
20     component: ProductsComponent
21   },
22   {
23     path: 'contacts',
24     component: ContactsComponent
25   },
26   {
27     path: 'sign-in',
28     component: SignInComponent
29   },
30   {
31     path: 'sign-up',
32     component: SignUpComponent
33   },
34   {
35     path: 'add-product',
36     component: AddProductComponent
37   },
38   {
39     path: 'orders',
40     component: OrdersComponent
41   },
42   {
43     path: 'feedback-list',
44     component: FeedbackListComponent
45   },
46   {
47     path: 'customers',
48     component: CustomersComponent
49   }
50 ];
51
52 @NgModule({
53   imports: [RouterModule.forRoot(routes)],
54   exports: [RouterModule]
55 })
56 export class AppRoutingModule {
57 }
58
59
```

If user navigate to Products, products list will be opened



We used Table of Angular Material, in this table we can do **searching, sorting, paging**.

Implementation of this page in:
app/components/products/products.ts

This is template for Products page

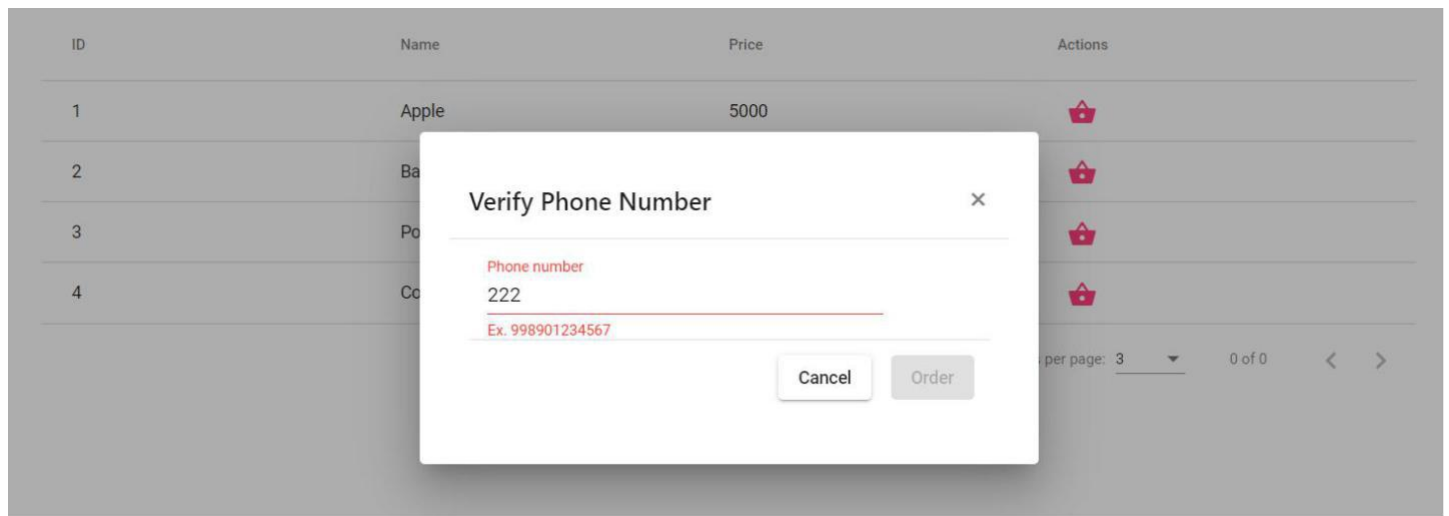
```
14 | template:
15 |     <div class="mt-3 d-flex justify-content-center">
16 |         <mat-form-field class="col-3">
17 |             <input matInput type="text" (keyup)="doFilter($event.target.value)" placeholder="Search...">
18 |         </mat-form-field>
19 |     </div>
20 |
21 |     <div class="container">
22 |         <mat-table [dataSource]="dataSource" matSort matSortStart="asc">
23 |
24 |             <ng-container matColumnDef="product_id">
25 |                 <mat-header-cell *matHeaderCellDef mat-sort-header>ID</mat-header-cell>
26 |                 <mat-cell *matCellDef="let products"> {{products.product_id}} </mat-cell>
27 |             </ng-container>
28 |
29 |             <ng-container matColumnDef="name">
30 |                 <mat-header-cell *matHeaderCellDef mat-sort-header>Name</mat-header-cell>
31 |                 <mat-cell *matCellDef="let products"> {{products.name}} </mat-cell>
32 |             </ng-container>
33 |
34 |             <ng-container matColumnDef="price">
35 |                 <mat-header-cell *matHeaderCellDef mat-sort-header>Price</mat-header-cell>
36 |                 <mat-cell *matCellDef="let products"> {{products.price}} </mat-cell>
37 |             </ng-container>
38 |
39 |             <ng-container matColumnDef="actions">
40 |                 <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
41 |                 <mat-cell *matCellDef="let products">
42 |
43 |                     <button mat-icon-button color="accent" matTooltip="Order This Product"
44 |                         (click)="openOrderModal()">
45 |                         <mat-icon>shopping_basket</mat-icon>
46 |                     </button>
47 |
48 |                 </mat-cell>
49 |             </ng-container>
50 |
51 |             <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
52 |             <mat-row *matRowDef="let row; columns: displayedColumns; let i = index"></mat-row>
53 |         </mat-table>
54 |         <mat-paginator [pageSize]="3" [pageSizeOptions]="[3, 5, 10, 15]">
55 |     </mat-paginator>
56 | </div>
```

This is logic of **Products** page. In the AdminService there are all functions for manipulationg with **REST API** using **HttpClient**.

```
63 export class ProductsComponent implements OnInit, AfterViewInit {
64     displayedColumns = ['product_id', 'name', 'price', 'actions'];
65     products: Products[];
66     dataSource = new MatTableDataSource<Products>();
67     product_id: number; //for sending id of product to Modal
68
69     @ViewChild(MatSort) sort: MatSort;
70     @ViewChild(MatPaginator) paginator: MatPaginator;
71
72     constructor(private adminService: AdminService,
73                 private modal: MatDialog,) {
74
75     }
76
77     ngOnInit() {
78         this.getProducts();
79     }
80
81     openOrderModal() {
82         const modal = this.modal.open(ModalOrder, config: {
83             width: '500px',
84             data: {product_id: this.product_id} //getting id of product for modal
85         });
86
87         modal.afterClosed().subscribe( next: result => {
88             this.getProducts();
89         });
90     }
91
92     ngAfterViewInit() {
93         this.dataSource.sort = this.sort;
94         this.dataSource.paginator = this.paginator;
95     }
96
97     doFilter(value: string) {
98         this.dataSource.filter = value.trim().toLocaleLowerCase();
99     }
100
101     getProducts() {
102         this.adminService.getProductList().subscribe( next: data => {
103             this.dataSource.data = data as Products[];
104         });
105     }
106 }
```

openOrderModal() function is for opening modal for ordering product. That means:

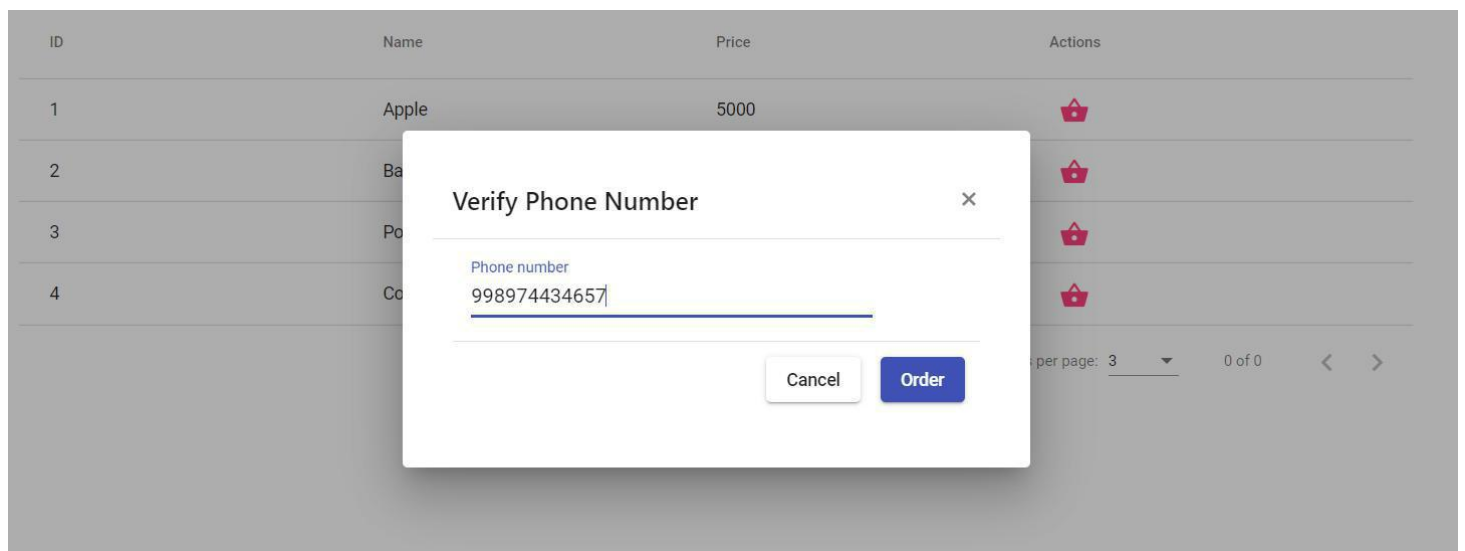
If we click to **basket** icon:



A screenshot of a web application showing a modal dialog titled "Verify Phone Number" with a close button (X). The modal contains a text input field with the label "Phone number" and a red error message "222" below it. A hint text "Ex. 998901234567" is visible below the input field. The modal has two buttons: "Cancel" and "Order". The "Order" button is disabled (grayed out). In the background, a table is visible with columns "ID", "Name", "Price", and "Actions". The table has four rows with IDs 1, 2, 3, and 4. The "Actions" column contains a red shopping basket icon for each row. The table also shows a "per page: 3" dropdown and "0 of 0" items.

User should be insert own phone number and we used **validation** for escaping mistakes. If there is mistake, **Order** button will be **disabled**.

After writing correctly:



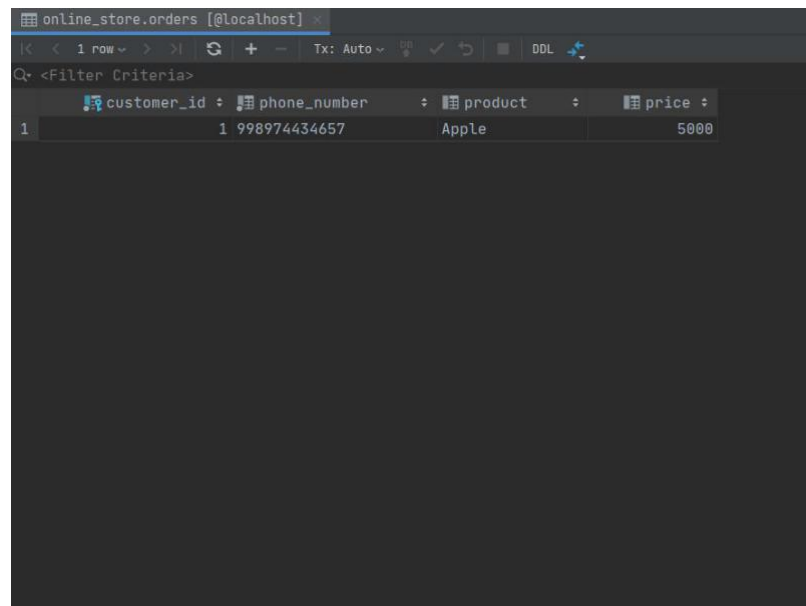
A screenshot of the same web application showing the "Verify Phone Number" modal dialog. The text input field now contains the correct phone number "998974434657". The error message has disappeared, and the "Order" button is now active (blue). The "Cancel" button remains gray. The background table and pagination controls are the same as in the previous screenshot.

Errors will disappear and **Order** button will be usable.

We used Angular's **FormBuilder** validator **patterns** for validate that input.

```
products.component.ts x
144     </div>
145   </div>
146   ,
147   })
148   export class ModalOrder {
149     order: Orders = new Orders();
150     submitted = false;
151     validate = new FormGroup( controls: {});
152
153     constructor(private adminService: AdminService,
154                 private customerService: CustomerService,
155                 private fb: FormBuilder,
156                 private modal: MatDialogRef<ModalOrder>,
157                 @Inject(MAT_DIALOG_DATA) public data: Products) {
158
159       this.validate = fb.group( controlsConfig: {
160         'phone_number': ['', [Validators.pattern( pattern: '[6-9]\\d{11}')] ],
161       })
162     }
163
164     orderProduct() {
165       this.submitted = true;
166       this.customerService.orderProduct(this.order)
167         .subscribe( next: data => console.log(data),
168                   error: error => console.log(error));
169       this.order = new Orders();
170     }
171
172     close() {
173       this.modal.close();
174     }
175   }
176 }
```

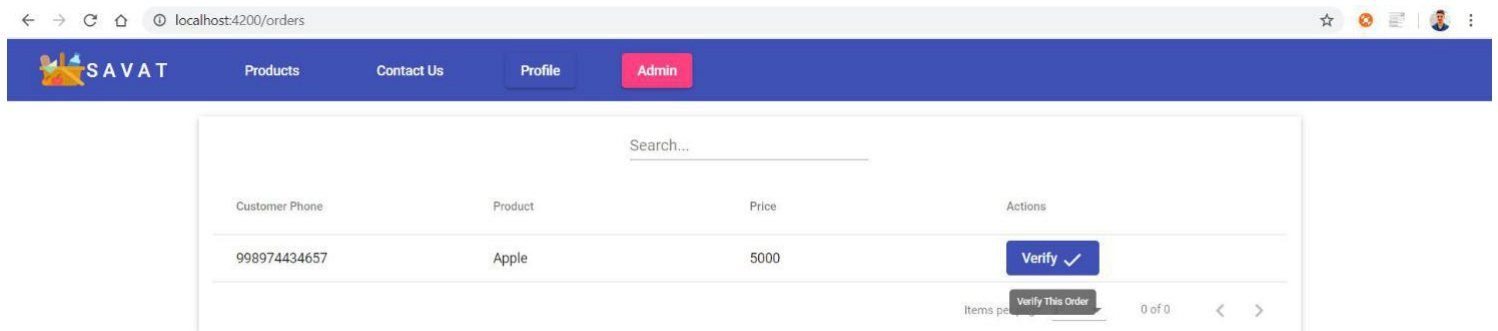
After ordering correctly, order will appear in **orders** table:



The screenshot shows a database client interface with the 'online_store.orders' table selected. The table has four columns: 'customer_id', 'phone_number', 'product', and 'price'. A single row is displayed with the following values: 1, 998974434657, Apple, and 5000.

customer_id	phone_number	product	price
1	998974434657	Apple	5000

Admin can see and **verify** that order in **Orders List**.

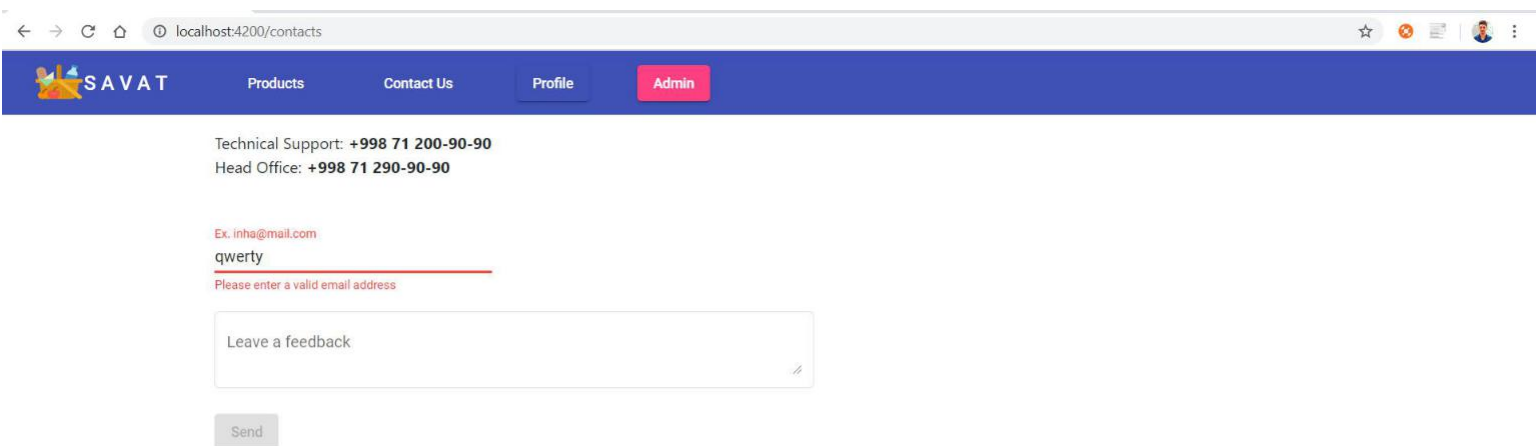


The screenshot shows the 'Orders List' page in the SAVAT application. The page has a navigation bar with 'Products', 'Contact Us', 'Profile', and 'Admin' (highlighted in red). Below the navigation bar is a search bar and a table with the following columns: 'Customer Phone', 'Product', 'Price', and 'Actions'. A single row is displayed with the following values: 998974434657, Apple, 5000, and a 'Verify' button with a checkmark icon. A tooltip 'Verify This Order' is visible over the button.

Customer Phone	Product	Price	Actions
998974434657	Apple	5000	<button>Verify ✓</button>

Also, users can send **feedback** to managers using **Contact Us** link:

If user insert invalid email address error appears instantly, and **Send** button will be disabled

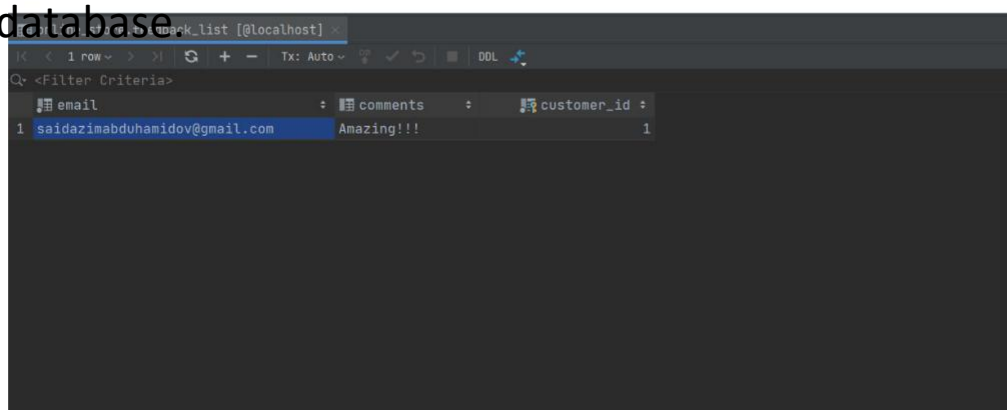


The screenshot shows a web browser at localhost:4200/contacts. The header has a blue bar with the SAVAT logo and navigation links: Products, Contact Us, Profile, and Admin. Below the header, contact information is displayed: Technical Support: +998 71 200-90-90 and Head Office: +998 71 290-90-90. The form section has a placeholder email 'Ex. inha@mail.com' and a text input field containing 'qwerty'. A red error message 'Please enter a valid email address' is shown below the input. The 'Leave a feedback' text is inside the input field. The 'Send' button is disabled (greyed out).



The screenshot shows the same web browser at localhost:4200/contacts. The header and contact information are the same. The form section now has a placeholder email 'Ex. inha@mail.com' and a text input field containing 'saidazimabduhamidov@gmail.com'. The 'Leave a feedback' text is inside the input field. The 'Send' button is now active (blue).

After inserting values correctly, **feedback** will be inserted to the **feedback_list** table in database



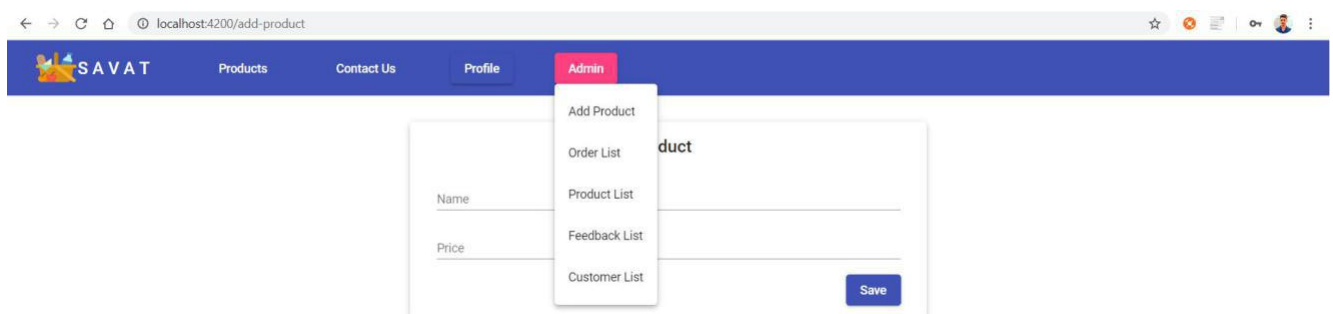
The screenshot shows a database client window titled 'feedback_list [localhost]'. The table structure is defined as:

email	comments	customer_id
saidazimabduhamidov@gmail.com	Amazing!!!	1

In **sendFeedback()** function we used **sendFeedback** function of **CustomerService** for **POST** HTTP request

```
contacts.component.ts
1 template: `
2   <div class="container">
3     <div class="col mt-3">
4       <div class="row">
5         Technical Support: <span> +998 71 200-90-90</span>
6       </div>
7       <div class="row">
8         Head Office: <span> +998 71 290-90-90</span>
9       </div>
10    </div>
11    <form (ngSubmit)="sendFeedback()">
12      <mat-form-field class="mt-5 w-25">
13        <input matInput type="email"
14          [formControl]="emailFormControl"
15          placeholder="Ex. inha@mail.com">
16        <mat-error *ngIf="emailFormControl.hasError('email') && !emailFormControl.hasError('required')">
17          Please enter a valid email address
18        </mat-error>
19        <mat-error *ngIf="emailFormControl.hasError('required')">
20          Email is <strong>required</strong>
21        </mat-error>
22      </mat-form-field>
23
24      <div class="mt-3">
25        <mat-form-field appearance="outline" style="width: 100%;">
26          <mat-label>Leave a feedback</mat-label>
27          <textarea matInput></textarea>
28        </mat-form-field>
29      </div>
30
31      <button [disabled]="emailFormControl.hasError('email')*"
32        mat-raised-button
33        color="primary">Send
34    </button>
35  </form>
36 </div>
37 `
38
39 styles: [
40   span {
41     font-weight: bold;
42     margin-left: 5px;
43   }
44
45   button {
46     outline: none;
47   }
48 ]
49
50 export class ContactsComponent {
51   emailFormControl = new FormControl( '', { validatorOrOpts: [
52     Validators.required,
53     Validators.email,
54   ]});
55
56   feedback: Feedback_list = new Feedback_list();
57
58   constructor(private customerService: CustomerService) {
59   }
60
61   sendFeedback() {
62     this.customerService.sendFeedback(this.feedback)
63       .subscribe( (next: data => console.log(data),
64         error: error => console.log(error));
65     this.feedback = new Feedback_list();
66   }
67 }
```

After putting feedback in database, Admins can see **feedback** in **Feedback List** link:



Customer Email	Comments
saidazimabduhamidov@gmail.com	Amazing!!!
Items per page: 3 0 of 0 < >	
5	
10	
15	

If user click **Add Product** button in the menu of **Admin** button, user can add product by this form

Add Product

Name

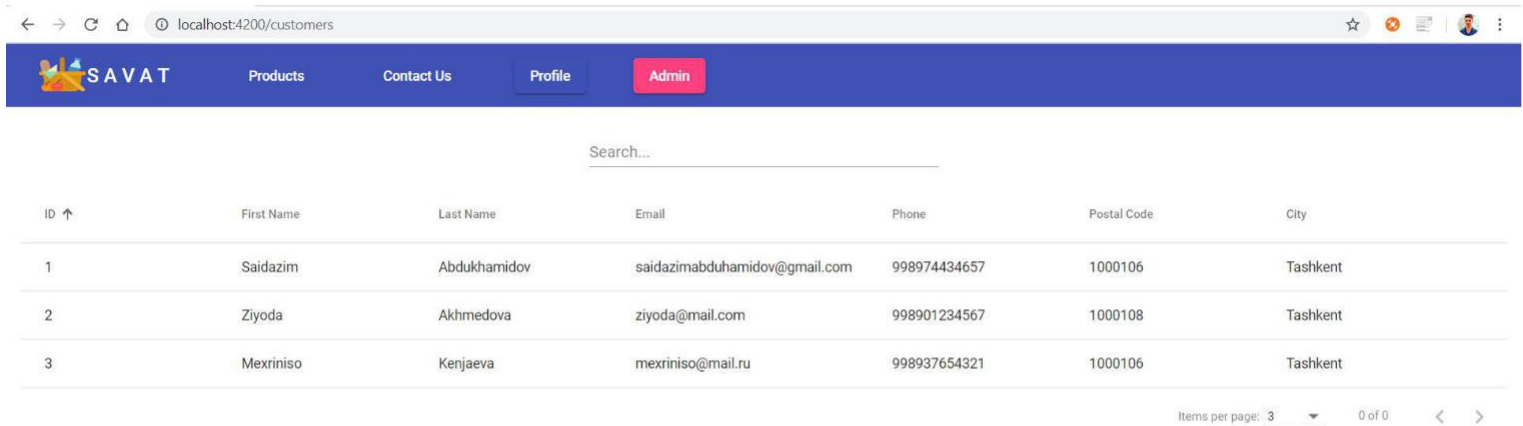
Coca-Cola 1.5

Price

8000

Save

And **Admin** can see list of registered customers:

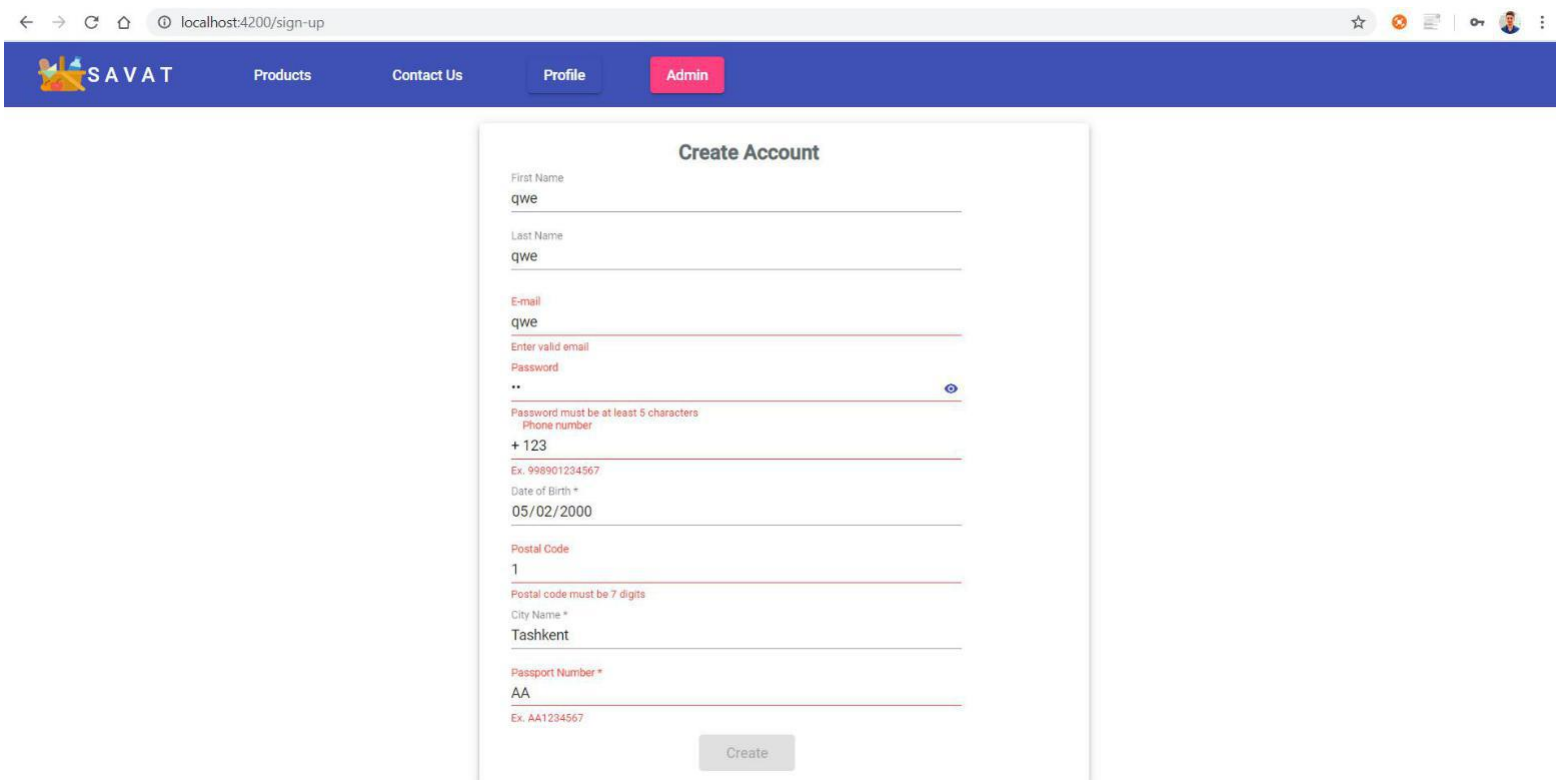


ID ↑	First Name	Last Name	Email	Phone	Postal Code	City
1	Saidazim	Abdukhamidov	saidazimabduhamidov@gmail.com	998974434657	1000106	Tashkent
2	Ziyoda	Akhmedova	ziyoda@mail.com	998901234567	1000108	Tashkent
3	Mexriniso	Kenjaeva	mexriniso@mail.ru	998937654321	1000106	Tashkent

Items per page: 3 0 of 0

Finally, **Login** and **Registration** pages. Users can Login or make Registration using **Profile** button in the **nav-bar**

In the registration page, we put validations for important values. For example:



Create Account

First Name
qwe

Last Name
qwe

E-mail
qwe
Enter valid email

Password
..
Password must be at least 5 characters

Phone number
+ 123
Ex. 998901234567

Date of Birth *
05/02/2000

Postal Code
1
Postal code must be 7 digits

City Name *
Tashkent

Passport Number *
AA
Ex. AA1234567

Create

Email is our login for **Entering** and it should be **at least 5 characters as Password**, Other input forms also have validations. With **errors** Create button **does not** work.

localhost:4200/sign-up

SAVAT Products Contact Us Profile Admin

Create Account

First Name
Saidazim

Last Name
Abdukhmidov

E-mail
saidazimabduhamidov@gmail.com

Password
a1234567 8/16

Phone number
+ 998974434657

Date of Birth *
05/02/2000

Postal Code
1000106
Ex. 100100

City Name *
Tashkent

Passport Number *
AA1234567
AA1234567

Create

```
sign-up.component.ts
6  @Component({
7    selector: 'app-sign-up',
8    templateUrl: './sign-up.component.html',
9    styleUrls: ['./sign-up.component.css']
10 })
11 export class SignUpComponent implements OnInit {
12   hide = true;
13   registration = new FormGroup( controls: {});
14   customers: Customers = new Customers();
15
16   constructor(private customerService: CustomerService,
17               private fb: FormBuilder) {
18
19     this.registration = fb.group( controlsConfig: {
20       'login': ['', [Validators.min( min: 5), Validators.max( max: 16)]],
21       'password': ['', [Validators.min( min: 5), Validators.max( max: 16)]],
22       'email': ['', [Validators.email]],
23       'phone_number': ['', [Validators.pattern( pattern: '[6-9]\\d{11}')]],
24       'postal_code': ['', [Validators.pattern( pattern: '[0-9]{7}')]],
25       'passport_number': ['', [Validators.pattern( pattern: '[A-Z]{2}[0-9]{7}')]]
26     })
27   }
28
29   ngOnInit() {
30   }
31
32   signUp() {
33     this.customerService.register(this.customers)
34       .subscribe( next: data => console.log(data),
35                  error: error => console.log(error));
36     this.customers = new Customers();
37   }
38 }
39
```

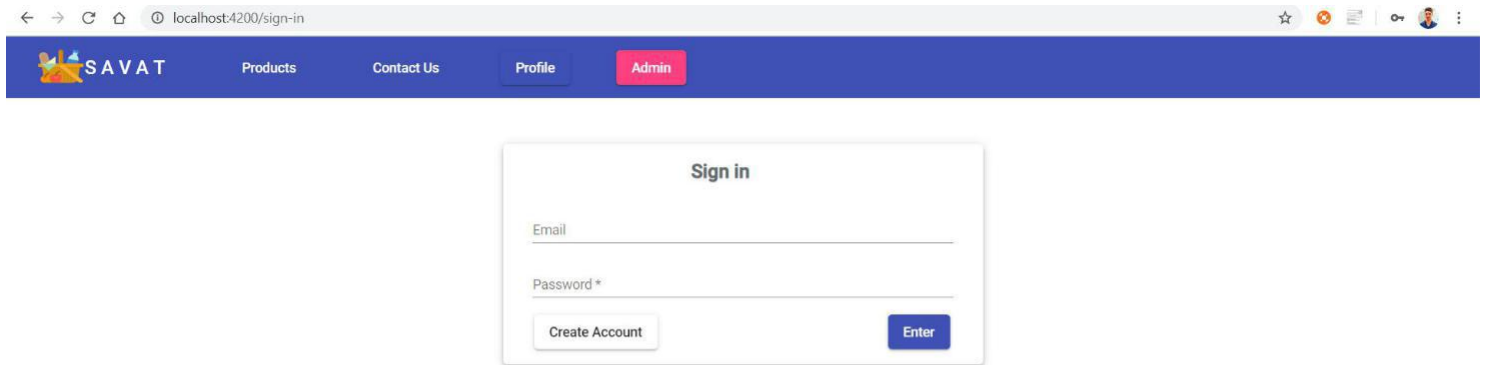

There are our services:

admin.service.ts

```
admin.service.ts x
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class AdminService {
9    private url = 'http://127.0.0.1:8080';
10
11    constructor(private http: HttpClient) {
12    }
13
14    addProduct(product: Object): Observable<Object> {
15      return this.http.post( url: `${this.url}/api/createProducts.php`, product);
16    }
17
18    getProductList(): Observable<any> {
19      return this.http.get( url: `${this.url}/api/readProducts.php`);
20    }
21
22    getCustomerList(): Observable<any> {
23      return this.http.get( url: `${this.url}/api/readCustomers.php`);
24    }
25
26    getOrderList(): Observable<any> {
27      return this.http.get( url: `${this.url}/api/readOrderItems.php`);
28    }
29
30    getFeedbackList(): Observable<any> {
31      return this.http.get( url: `${this.url}/api/readFeedback.php`);
32    }
33
34    verifyOrder(id: number): Observable<any> {
35      return this.http.delete( url: `${this.url}/api/verifyOrder.php`);
36    }
37  }
38  |
```

```
admin.service.ts x customer.service.ts x
1  import ...
2
3
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class CustomerService {
9    private url = 'http://127.0.0.1:8080';
10
11    constructor(private http: HttpClient) {
12    }
13
14    orderProduct(product: Object): Observable<Object> {
15      return this.http.post( url: `${this.url}/api/createOrderItems`, product);
16    }
17
18    sendFeedback(feedback: Object): Observable<Object> {
19      return this.http.post( url: `${this.url}/api/createFeedback_list.php`, feedback);
20    }
21
22    register(customer: Object): Observable<Object> {
23      return this.http.post( url: `${this.url}/api/createCustomers.php`, customer);
24    }
25  }
26
```

In the *savat/api* folder there are logic of our **PHP**, connecting to database, getting and setting data. For this reason next to our **URL** we add path of that files



The screenshot displays a web browser window with the address bar showing `localhost:4200/sign-in`. The page features a blue header with the SAVAT logo and navigation links: [Products](#), [Contact Us](#), [Profile](#), and [Admin](#). The main content area contains a 'Sign in' form with the following elements:

- Sign in** title
- Email
- Password *
-
-

Unfortunately, we could not make role-based authorization. For this reason, Login page does not work, but users can use this application without entering to own profile.