Лабораторна робота 11.

Ionic Framework. Додавання сторінок та аутентифікація користувачів.

https://ionicframework.com/blog/navigating-the-change-with-ionic-4-and-angular-router/

Продовжимо роботу над проектом «Студентські групи» та додамо ще одну сторінку login, яка буде використовуватись для входу користувачів до застосунку (рис. 1).

```
›>ng g page login
```

Рис. 1. Додавання нової сторінки.

Відкриємо представлення сторінки login.page.html та розмістимо там поле вводу для імені користувача та кнопку увійти. По натисканню на кнопку виконаємо метод, що відповідатиме за перехід на головну сторінку (рис. 2).

```
<ion-header>
  <ion-toolbar>
    <ion-title>Студентські групи</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row color="primary" justify-content-center>
      <ion-col align-self-center size-md="6" size-lg="5" size-xs="12">
        <div text-center>
          <h3>ВХІД</h3>
        </div>

        <ion-item>
          <ion-label position="floating">Ім'я користувача</ion-label>
          <ion-input type="text" name="username"></ion-input>
        </ion-item>
        <ion-button color="primary"  class="ion-float-right" (click)="login()">
          <ion-icon slot="start" name="log-in"></ion-icon> Увійти
        </ion-button>

      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

Рис. 2. Файл login.page.html.

У класі LoginPage підключимо @angular/router та у методі login виконаємо перехід на домашню сторінку home (рис. 3).

```
import { Component, OnInit } from '@angular/core';
import {Router} from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
})
export class LoginPage implements OnInit {
  constructor(private router: Router) { }

  ngOnInit() {
  }

  login() {
    this.router.navigate( commands: ['/home']);
  }
}
```

Рис. 3. Файл login.page.ts.

Також додамо трохи власної стилізації для сторінки у login.page.scss (рис. 4).

```
ion-header {
  margin-bottom: 25px;
}

ion-item {
  margin-bottom: 10px;
}
```

Рис. 4. Додавання стилів для login page.

Для выдображення сторінки логіну при старті застосунку виконаємо зміну у масиві маршрутів у app-routing.module.ts, а саме – властивість redirectTo для корньового роуту ‘’ (рис. 5).

```
const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch: 'full' },
  { path: 'home', loadChildren: () => import('./home/home.module').then(
  onfulfilled: m => m.HomePageModule)},
  {
    path: 'login',
    loadChildren: () => import('./login/login.module').then( onfulfilled: m =>
m.LoginPageModule)
  },
];
```

Рис. 5. Зміни у app-routing.module.ts.

У представленні головної сторінки home.page.html до панелі інструментів додаємо кнопку logout, що повертатиме нас до логін-сторінки (рис. 6).

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Студентські групи
    </ion-title>
    <ion-buttons slot="secondary">
      <ion-button routerLink="/login" routerDirection="root">
          <ion-icon slot="icon-only" name="log-out"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
```

Рис. 6. Зміни у home.page.html.

Перевіримо роботу логін-форми, запустимо застосунок та натиснемо кнопку «Увійти» (рис. 7).

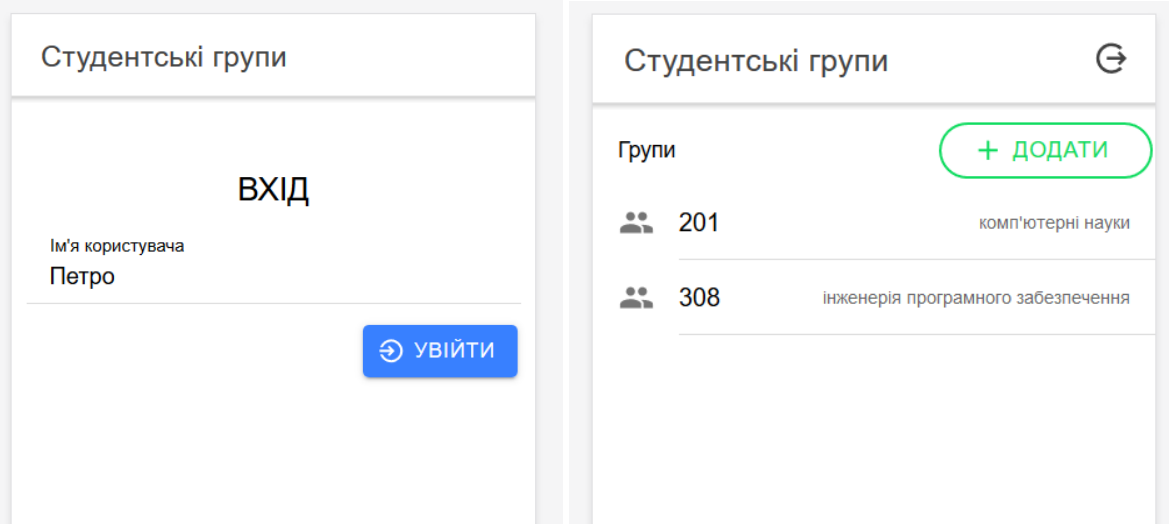

Рис. 7. Вхід до застосунку.

Зараз наша логін-форма не враховує, чи є такий користувач і взагалі дозволяє нічого не вводити у поле вводу, а просто натиснути «Увійти». Виправимо це, розширимо наш сервіс data-getter, додамо до нього масив користувачів, змінну для поточного користувача та методи для роботи із ними (рис. 8).

```
  private userName = '';

  private users = [
    'Василь', 'Петро', 'Олена'
  ];

  getUser() {
    return this.userName;
  }

  setUser(name: string) {
    this.userName = name;
  }

  userExists(name: string): boolean {
    return this.users.indexOf(name) !== -1;
  }

  constructor() { }
```

Рис. 8. Зміни у сервісі data-getter.service.ts.

Використаємо доданий до сервісу функціонал при виконанні входу – підключимо сервіс data-getter та будемо перевіряти, чи існує користувач, введений у поле вводу. Якщо користувача немає, виводитемо повідомлення (рис. 9).

```
import { Component, OnInit } from '@angular/core';
import {Router} from '@angular/router';
import {DataGetterService} from '../service/data-getter.service';
import {AlertController} from '@ionic/angular';

@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
})
export class LoginPage implements OnInit {
  userName: string;

  constructor(
      private router: Router,
      private dataGetter: DataGetterService,
      public alertController: AlertController) {}
```

```
18
19    ngOnInit() {
20    }
21
22    login() {
23      if (this.dataGetter.userExists(this.userName)) {
24        this.dataGetter.setUser(this.userName);
25        this.router.navigate( commands: ['/home']);
26      } else {
27        this.userNotExistAlert();
28      }
29    }
30
31    async userNotExistAlert() {
32      const alert = await this.alertController.create({
33        header: 'Увага !',
34        subHeader: 'Помилка аутентифікації',
35        message: `Користувача ${this.userName} на знайдено. Невірне ім'я
    користувача.`,
36        buttons: ['OK']
37      });
38
39      await alert.present();
40    }
41  }
```

Рис. 9. Зміни у login.page.ts.

Також на головній сторінці виведемо ім’я поточного користувача (рис. 10).

```
6       <ion-buttons slot="secondary">
7         ({{userName}})
8         <ion-button routerLink="/login" routerDirection="root">
9             <ion-icon slot="icon-only" name="log-out"></ion-icon>
10        </ion-button>
11      </ion-buttons>
```

Рис. 10. Додавання імені користувача до представлення home.page.html.

Перевіримо результат (рис. 11).

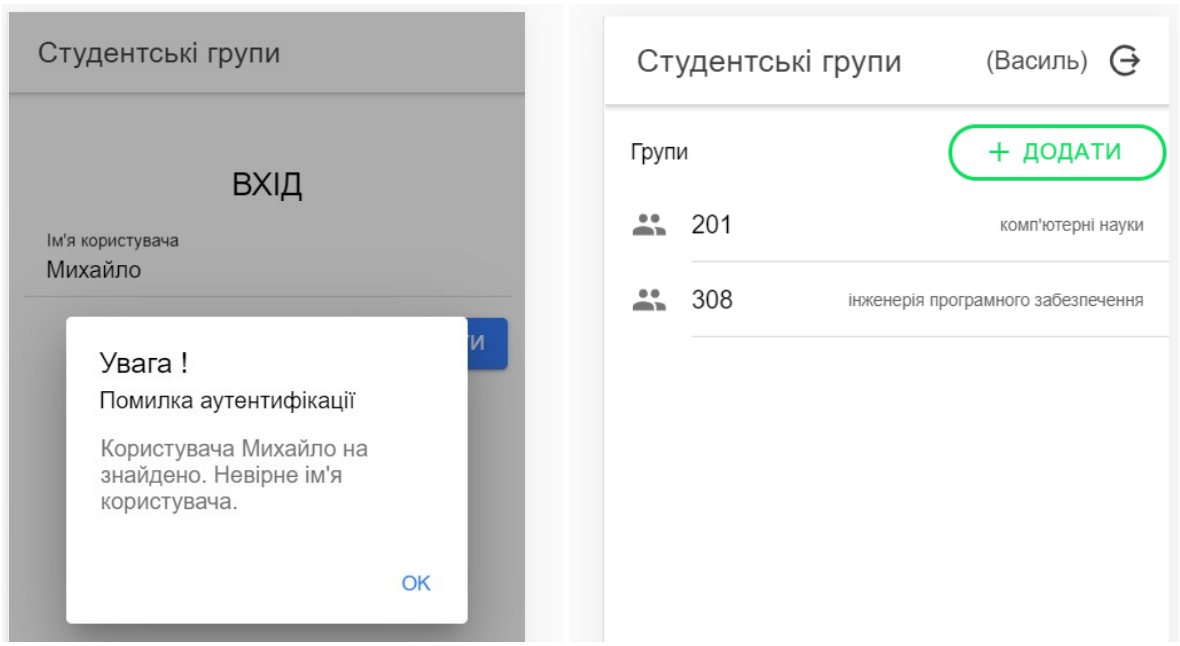

Рис. 11. Перевірка логін-форми.

Залишається наступна проблема – при переході за посиланням у адресному рядку до http://localhost:4200/home, ми можемо оминути процедуру аутентифікації (рис. 12).

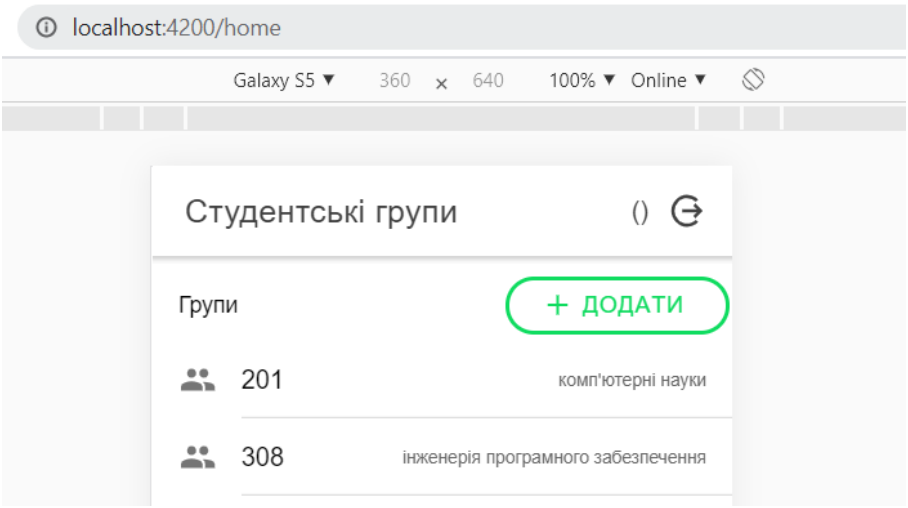

Рис. 12. Перехід до головної сторінки в обхід сторінки аутентифікації.

Виправимо дану проблему, додавши до проекту guard (ng generate guard guards/auth). Інтерфейс обираємо CanActivate. Далі наведемо код класу AuthGuard (рис. 13).

```
import { Injectable } from '@angular/core';
import {ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot}
 from '@angular/router';
import {DataGetterService} from '../service/data-getter.service';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  constructor(private dataGetter: DataGetterService,
              private router: Router) {}

  canActivate(
      next: ActivatedRouteSnapshot,
      state: RouterStateSnapshot): boolean {

    const isLoggenIn = this.dataGetter.getUser() !== '';
    if (!isLoggenIn) {
      this.router.navigateByUrl( url: '/login');
    }
    return isLoggenIn;
  }
}
```

Рис. 13. Guard Auth.

У файлі маршрутизації підключимо створений guard до маршруту home (рис. 14).

```
const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch: 'full' },
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then(
        onfulfilled: m => m.HomePageModule
    ),
    canActivate: [AuthGuard]
  },
```

Рис. 14. Зміни у app-routing.module.ts.

Тепер при спробі переходу до http://localhost:4200/home без проходження аутентифікації будемо перенаправлятися до сторінки логіну.

Завдання для самостійного виконання.

Реалізувати у застосунку згідно власного варіанту індивідуального завдання логін-форму та аутентифікацію користувачів.