**Міністерство освіти і науки України**

**Національний технічний університет України**

**“Київський політехнічний інститут”**

**Факультет прикладної математики**

**Кафедра спеціалізованих комп’ютерних систем**

**Лабораторна робота № 2**

з дисципліни «ВЕБ-дизайн»

«Розробка функціональності Web-додатка мовою Javascript»



Виконав:

студент групи КВ-01

**Шкільнюк Владислав**

Київ 2023

**Зміст**Мета: ознайомитись із засобами мови Javascript та навчитись їх застосовувати побудови Web-інтерфейсу користувача.

Загальне завдання: розробити функціональність для статичних сторінок Web-додатку першої лабораторної роботи із використанням шаблону MVC

Інструменти розробки: мови HTML5, CSS3, Javascript.

Зауваження: Web-фреймворки (ReactJS, AngularJS, VueJS) не використовувати!

**Завдання**

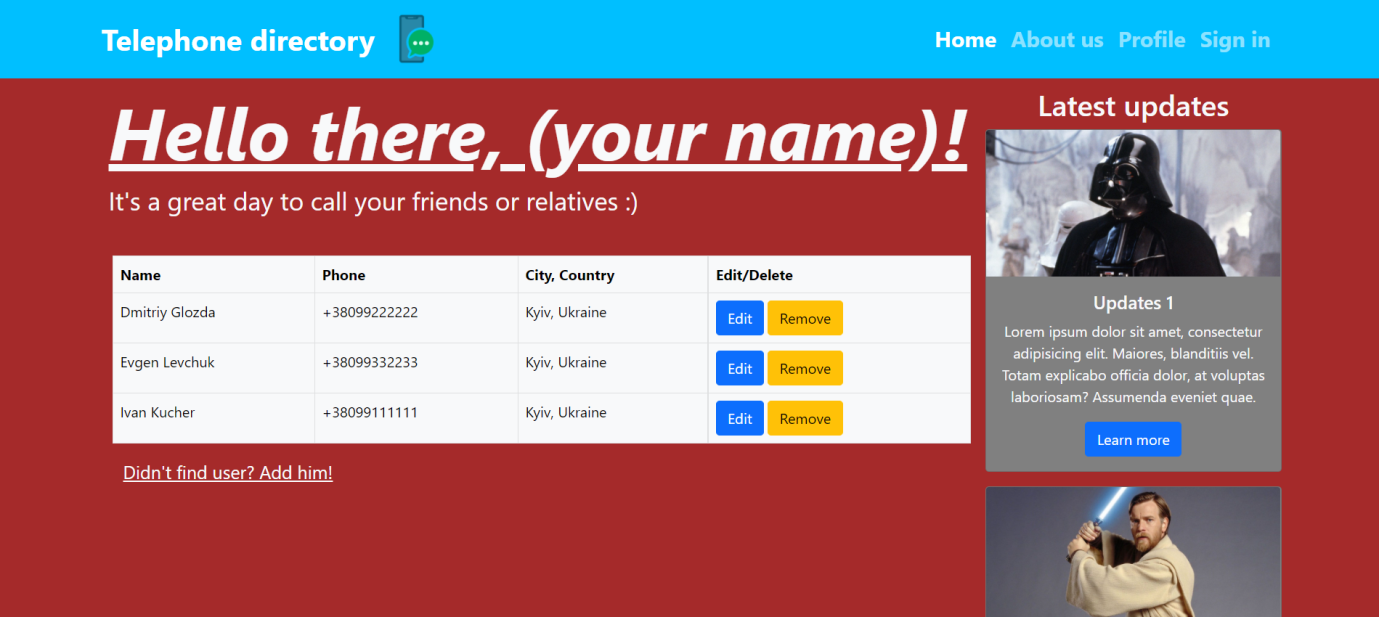
**Github-репозиторій з лабораторної:** https://github.com/vladikusen/web\_labs

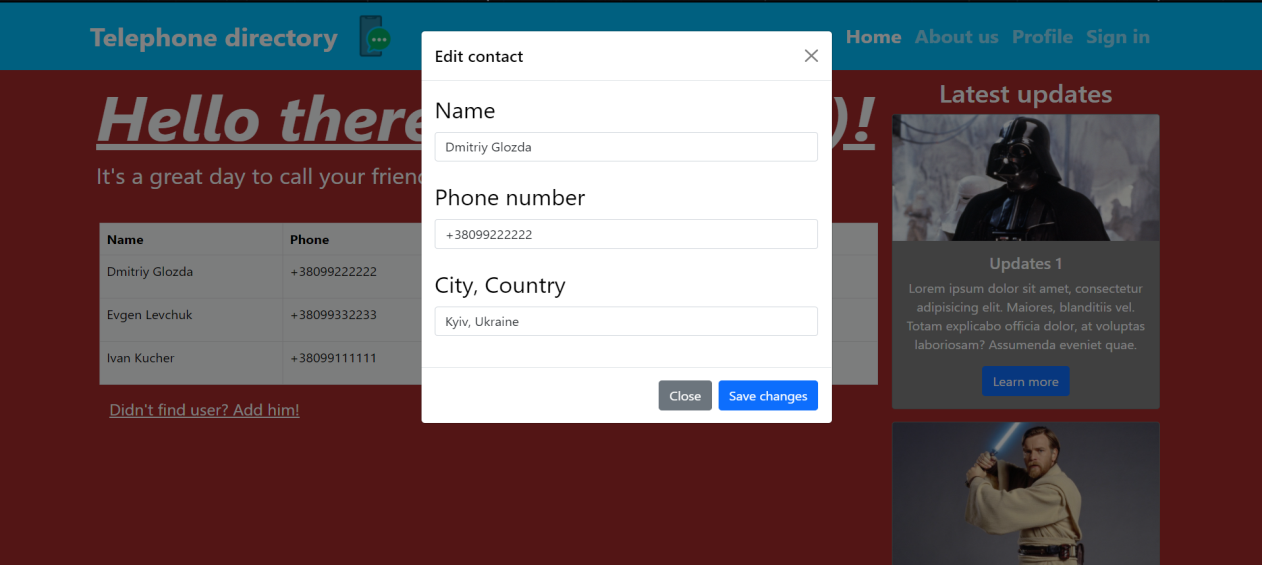
**Тема: “Телефонний довідник.**   
 У цій лабораторній роботі я розробив статичний Web-інтерфейс для додатку “Телефонний довідник”. Були реалізовані наступні веб-сторінки:

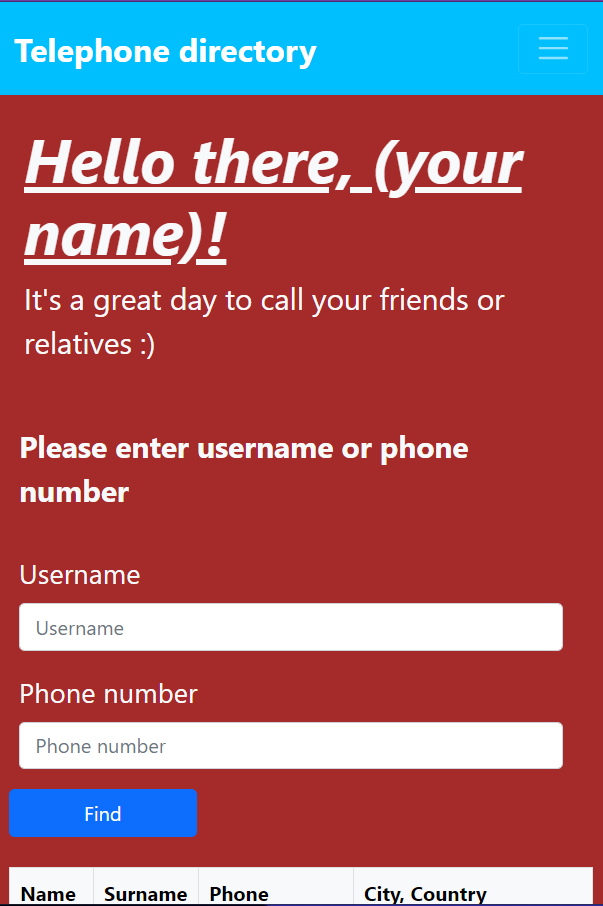
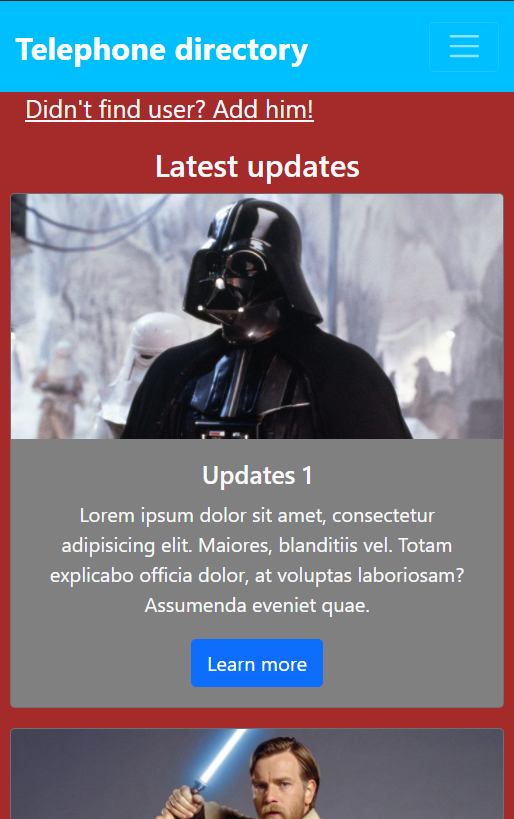
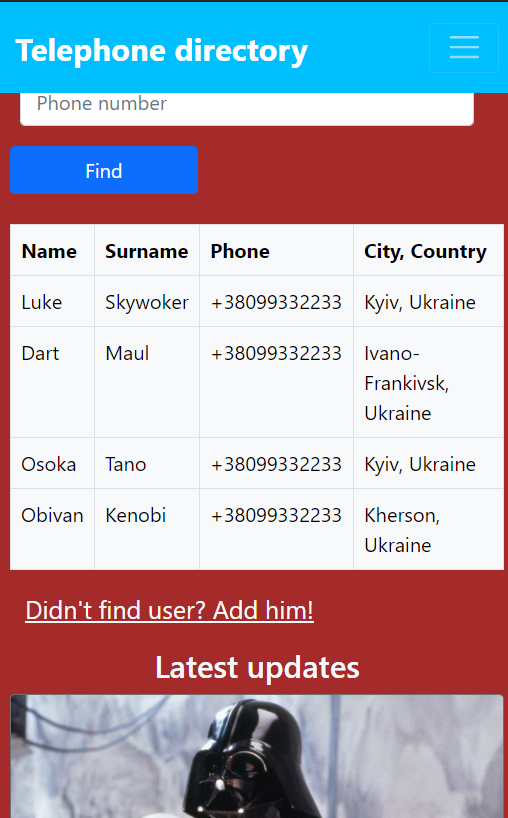
* index.html - головна сторінка. На ній користувач може знайти необхідний йому контакт за його ім’ям або/та номером телефону. Якщо контакт не було знайдено, користувач може створити його. Також на сторінці можно побачити пости з останніми оновленнями у додатку.
* new-contact.html - сторінка для створення нового контакту. На ній знаходиться звичайна форма з полями, необхідними для створення контакту.
* profile.html - сторінка профілю користувача.
* login.html - сторінка для логіну користувача.
* registration.html - сторінка для створення акаунта користувача.
* about.html - сторінка з детальною інформацією про додаток.

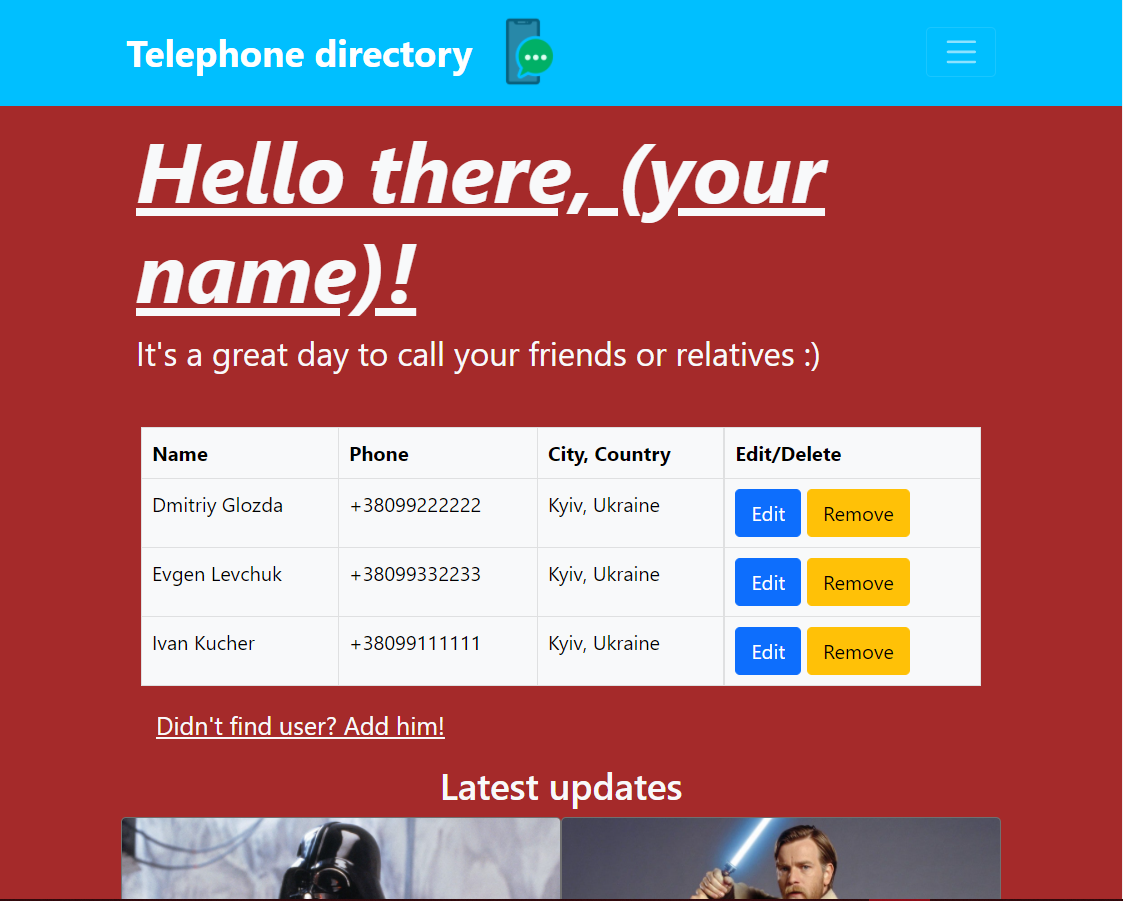
**index.html**

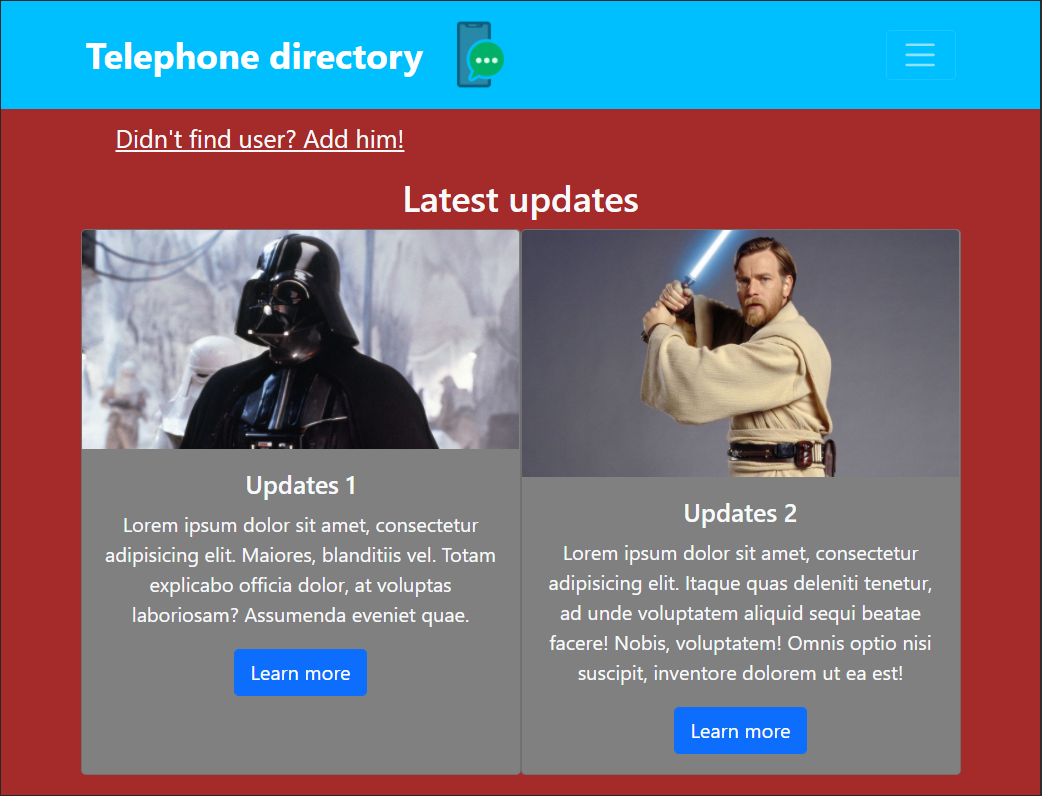
Для більш комфортної роботи з вже існуючими даними та їх відображенням, я вирішив зберегти дані про всі контакти у localStorage полі “contacts”.   
При першому завантаженні додатку початкові дані для додатку завантажуються з файлу contacts.js та записуються у localStorage .   
Контакти відображаються у алфавітному порядку за першою літерою “Name”.

****

При натисненні на кнопку “Remove” контакт буде видалено.   
При натисненні на кнопку “Edit” буде відкрите модальне вікно, з існуючими даними контакту, який користувач хоче змінити.   
  
Якщо користувач натисне на кнопки “Х”, “Close” або натисне на область біля модального вікна - змін не буде. Якщо буде натиснута кнопка “Save changes”, зміни будуть збережені.

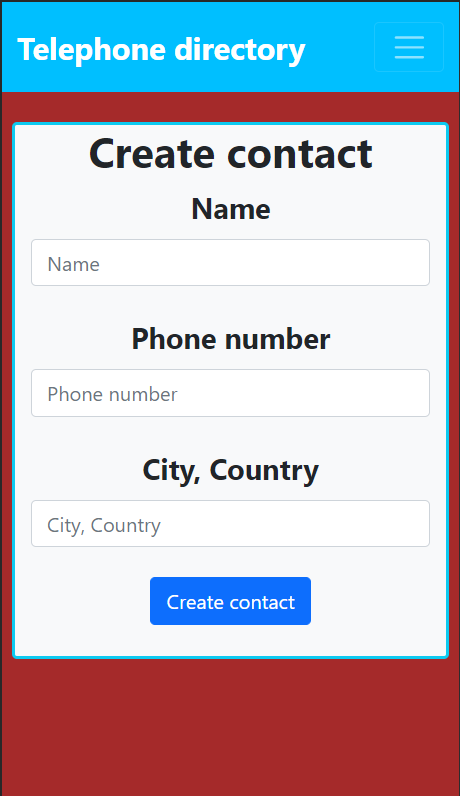
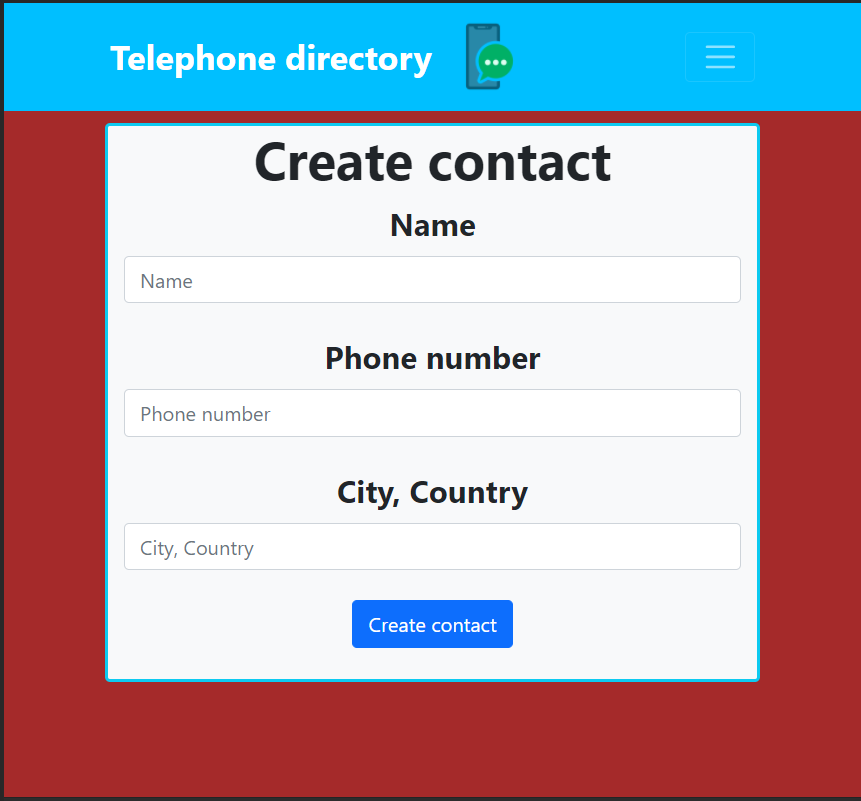
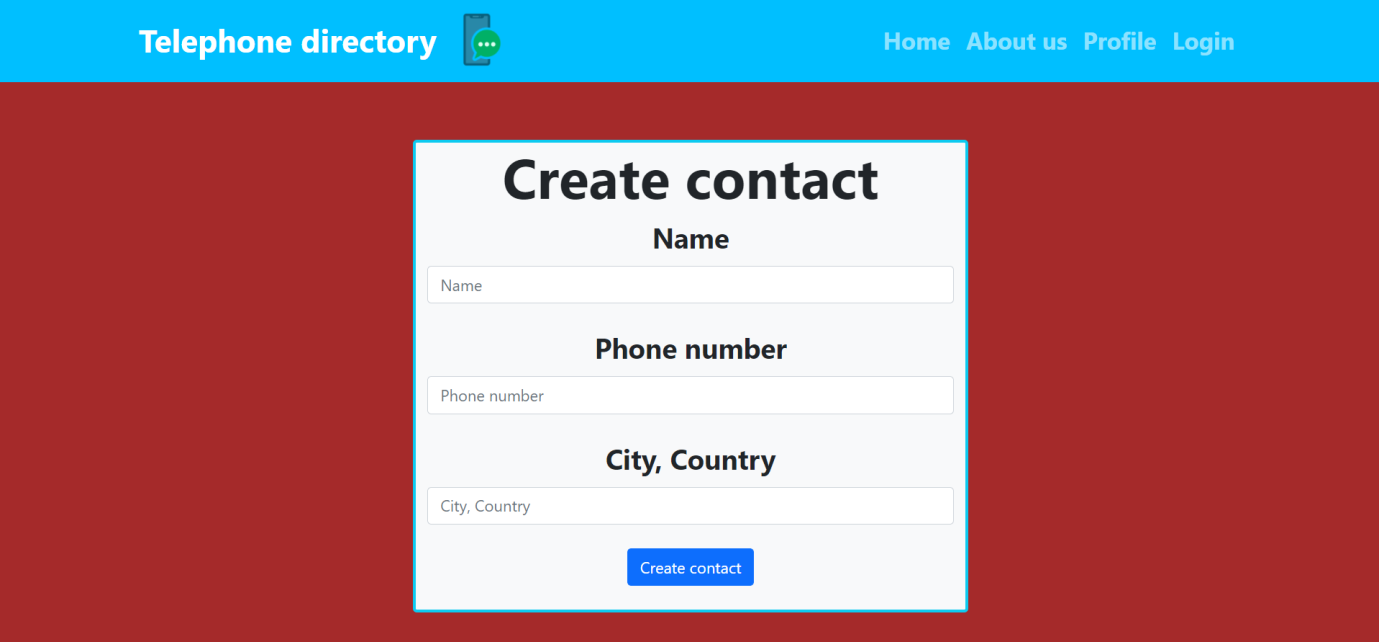
 



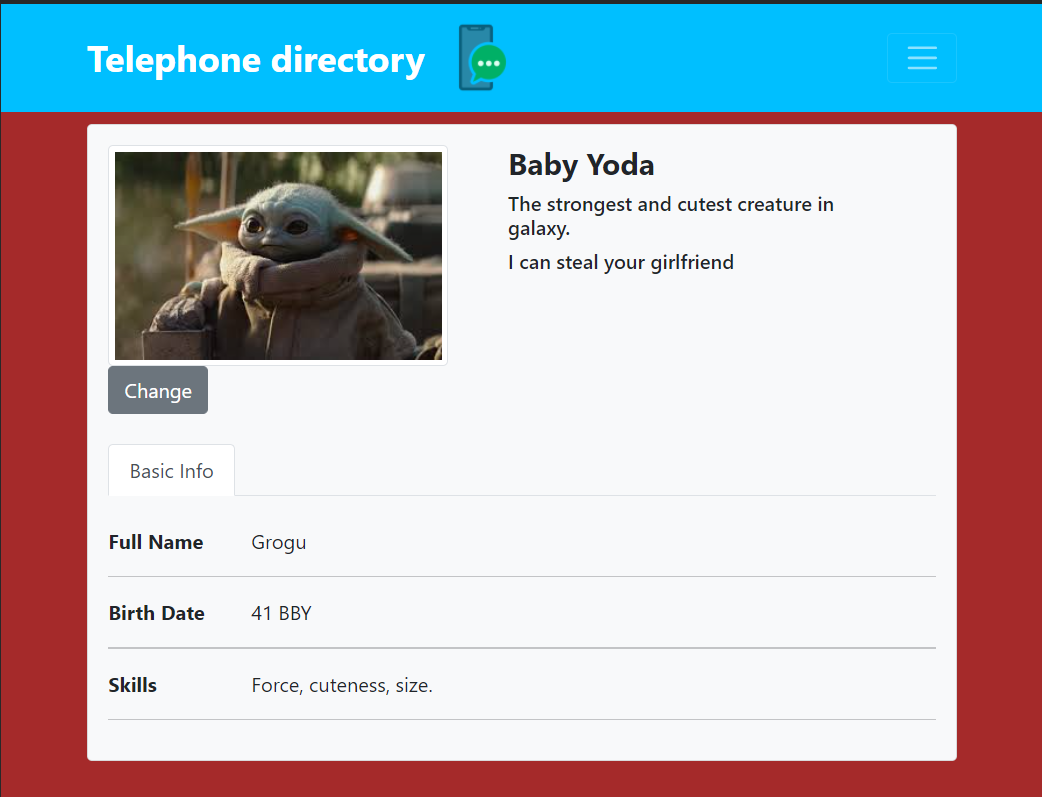
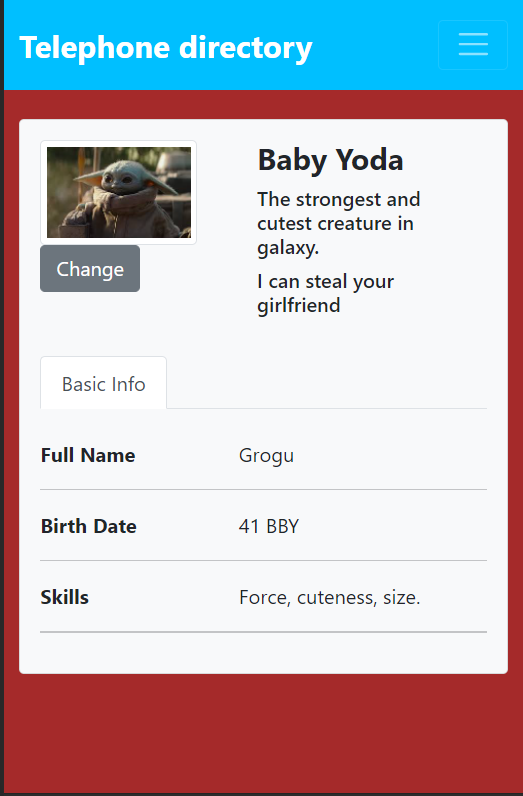
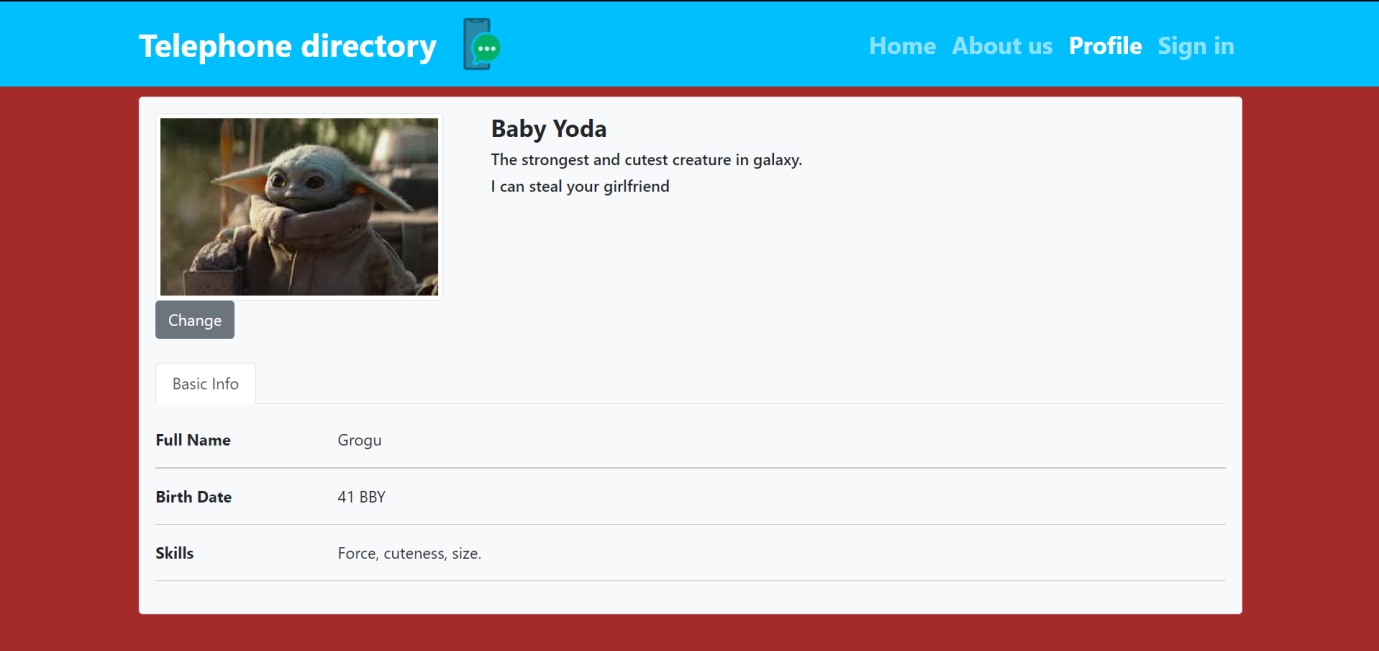


**new-contact.html**

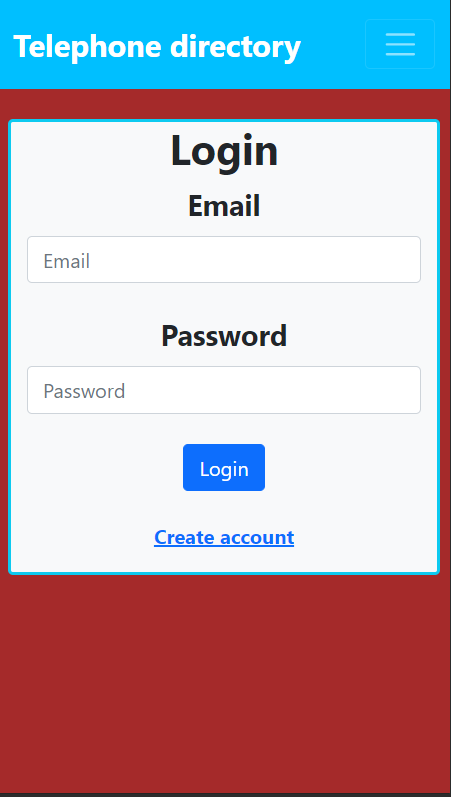
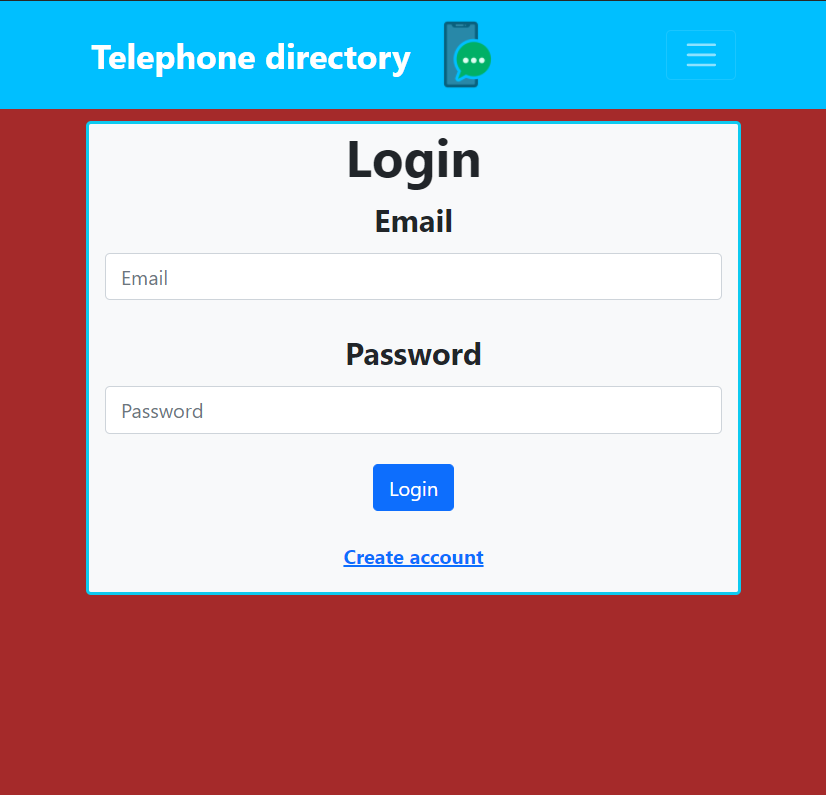
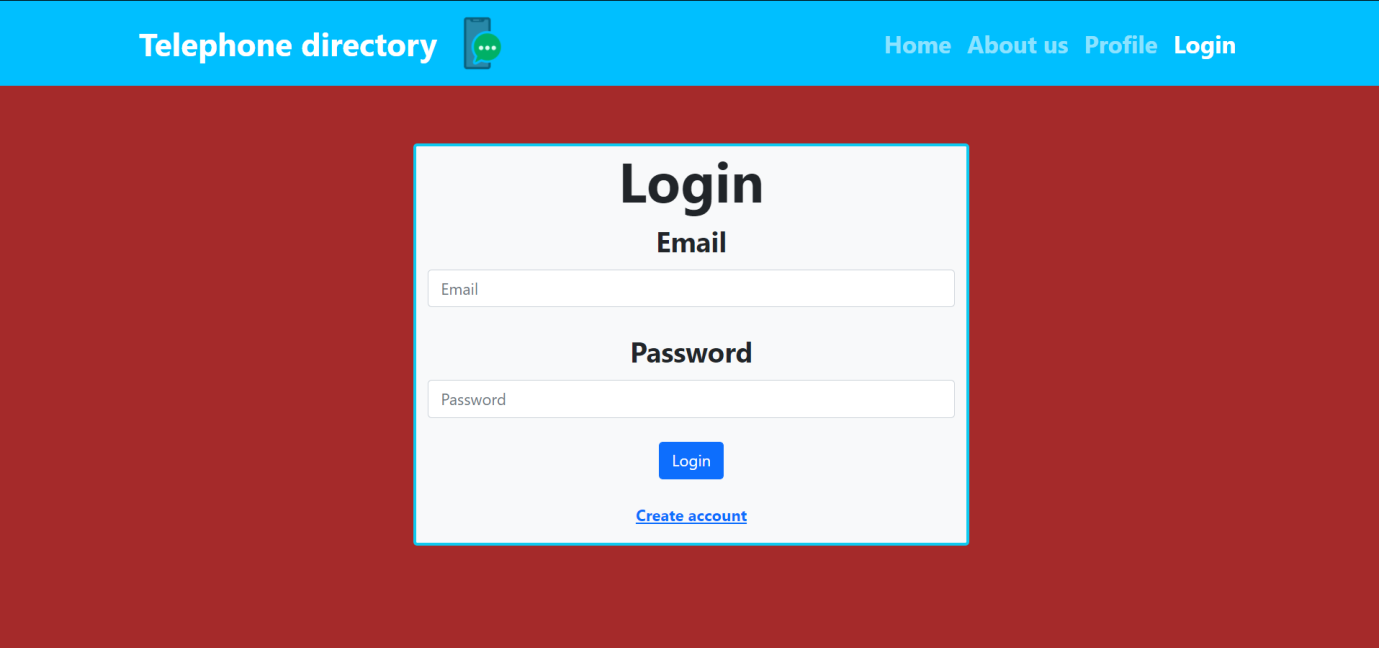
Була реалізована можливість створювати нові контакти. При натисненні на кнопку “Create contact”, контакт буде створено, а користувача буде перенаправлено до головної сторінки додатку.

****

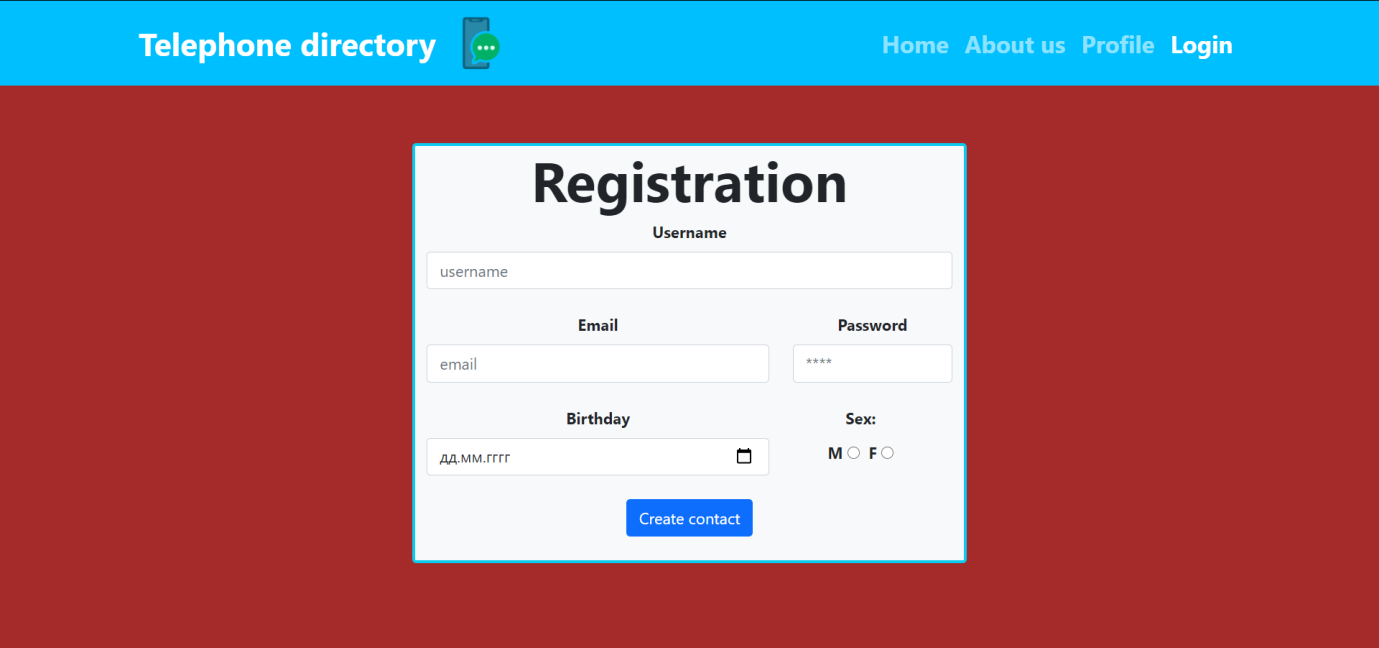
**profile.html**

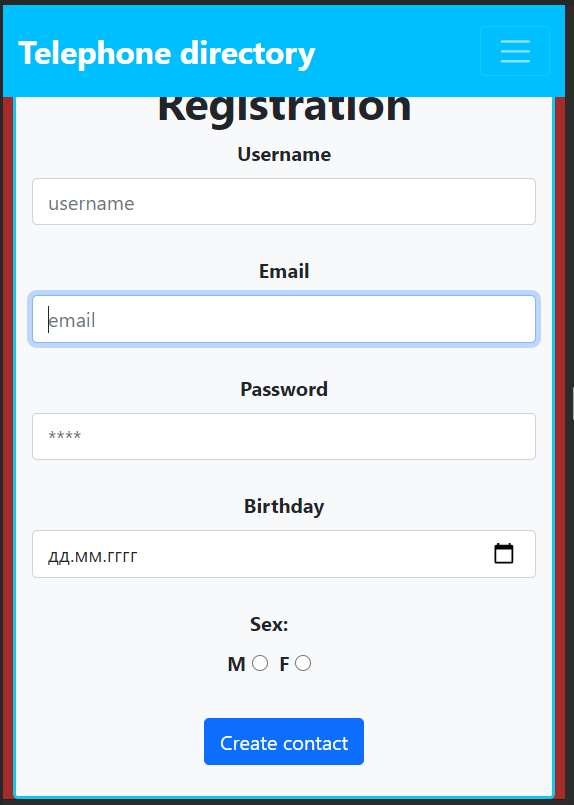
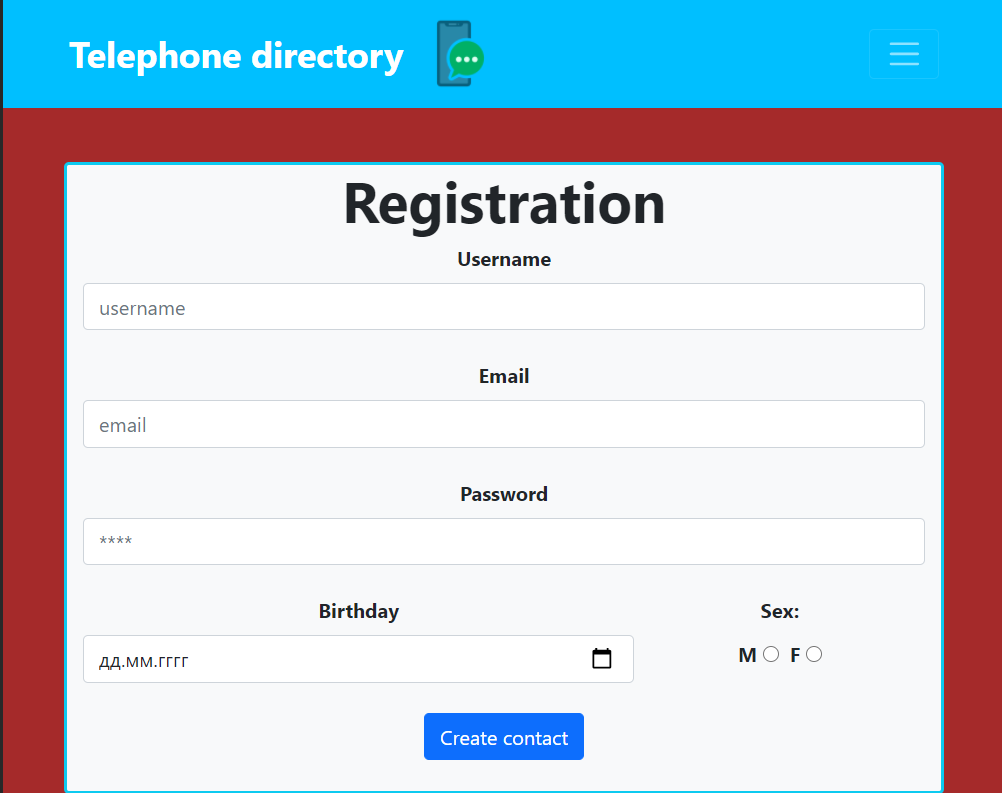
****

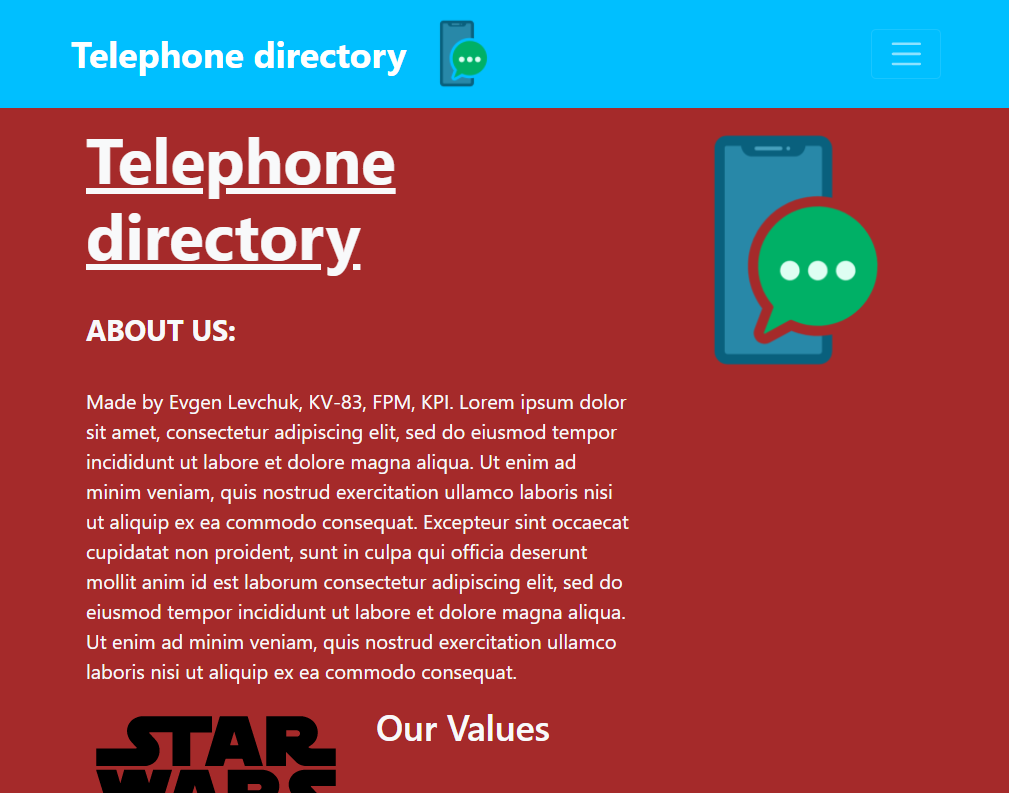
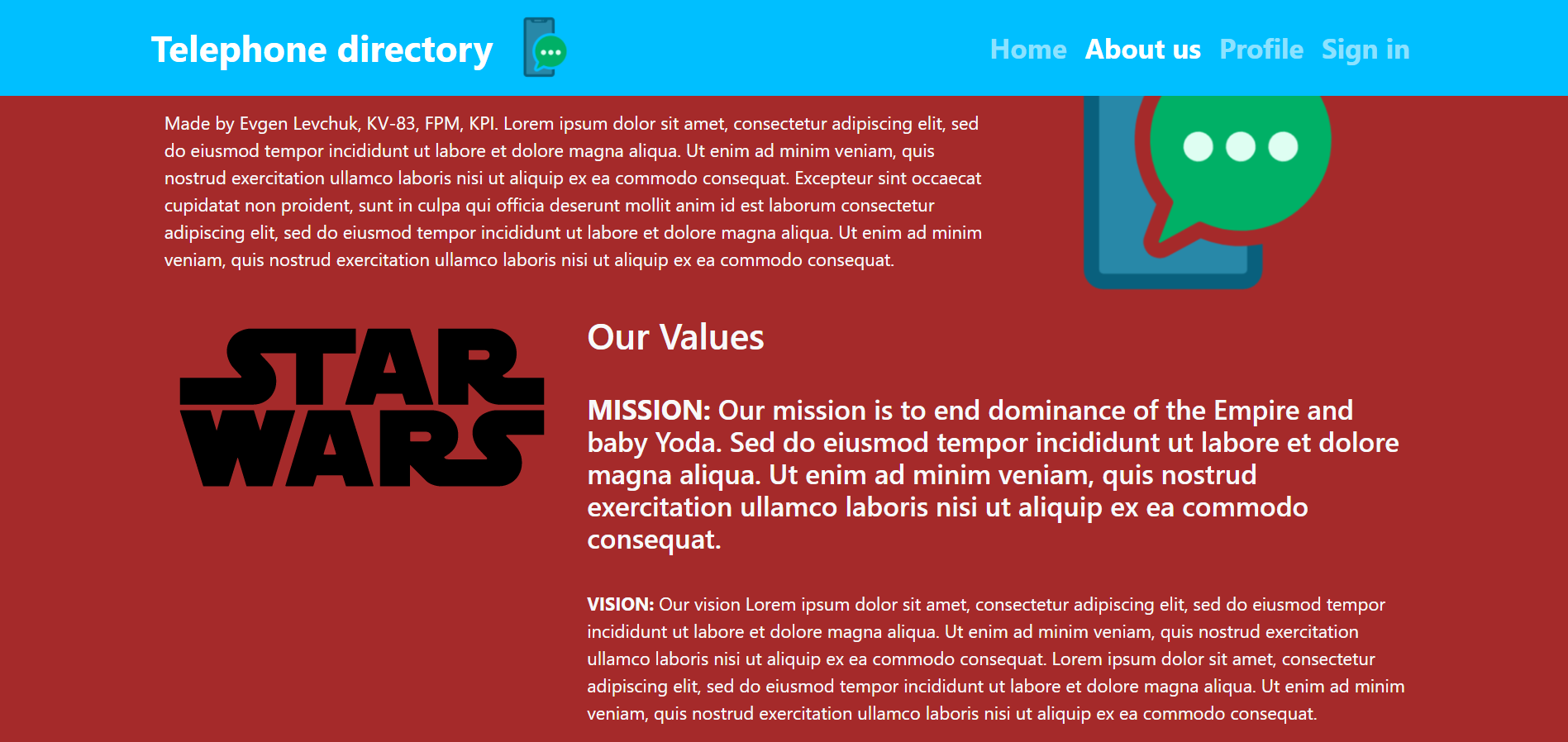
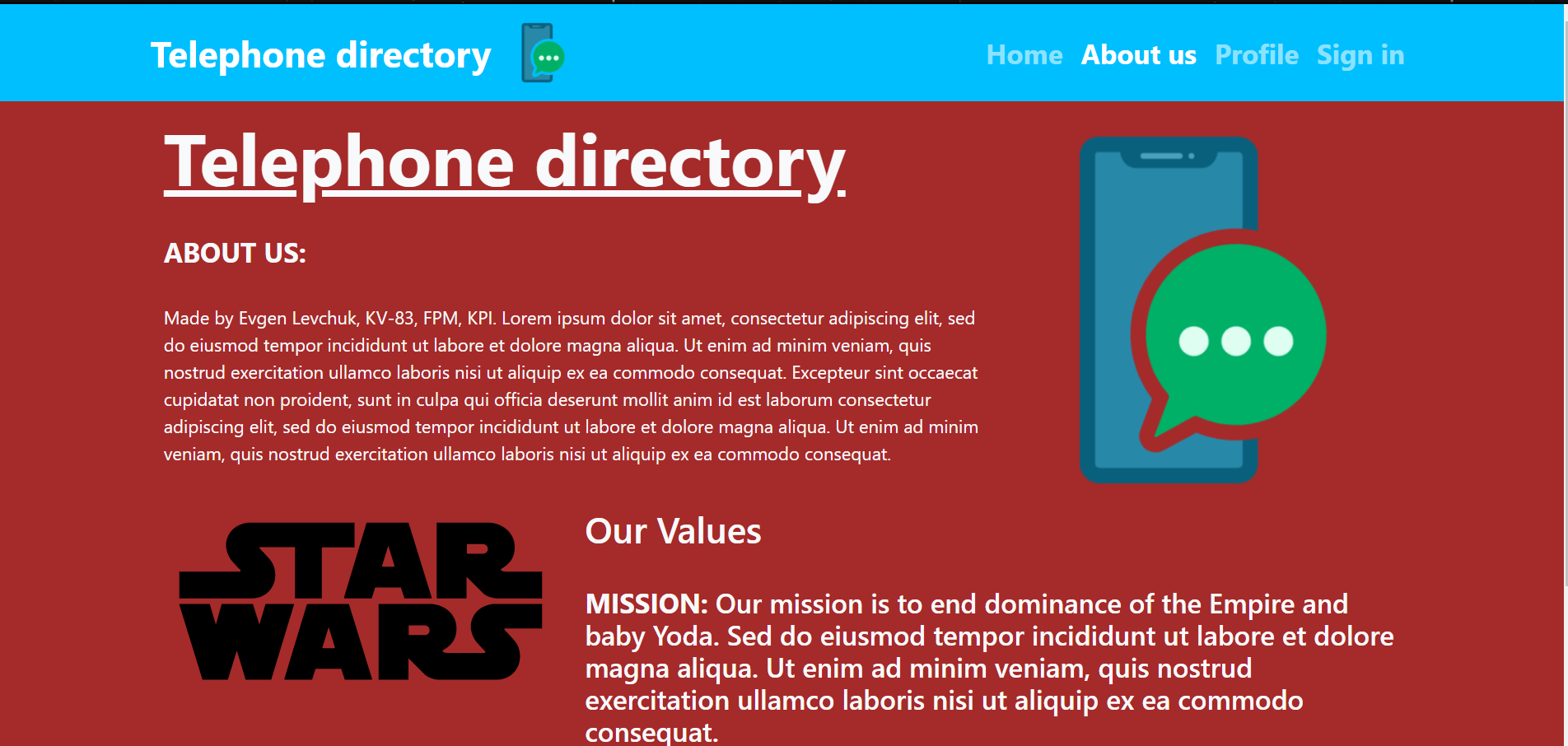
**login.html**

****

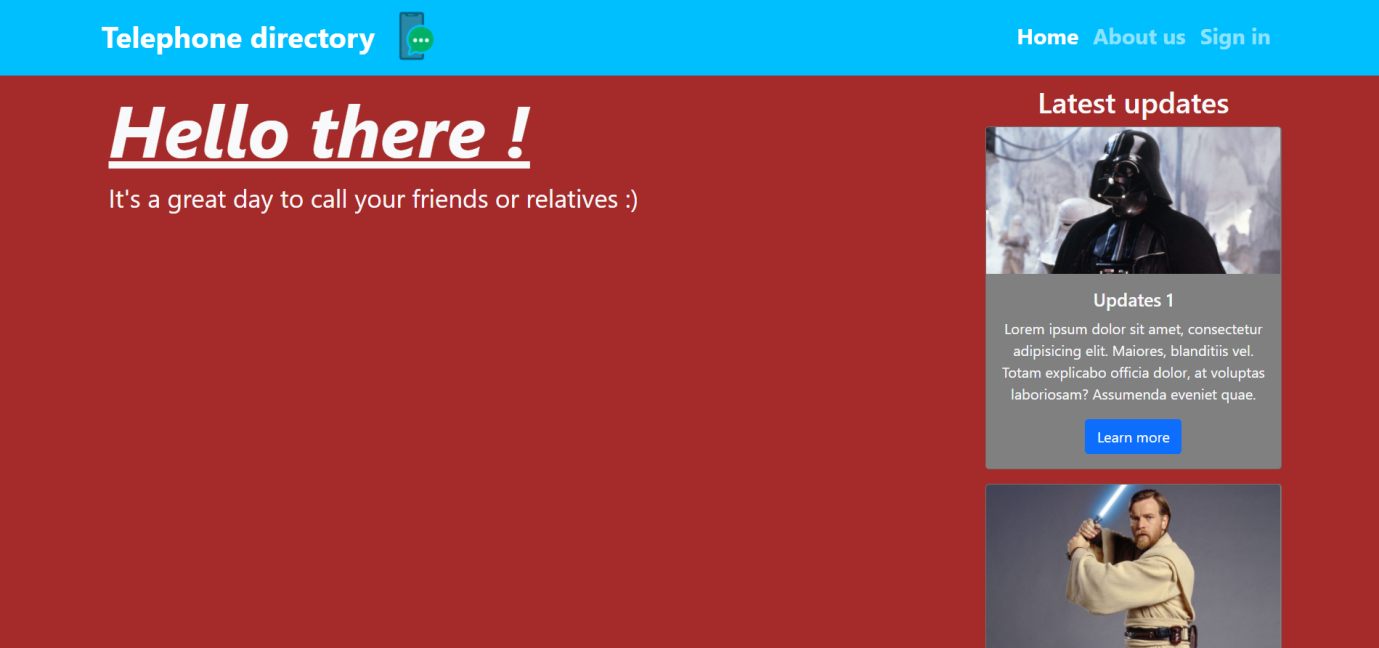
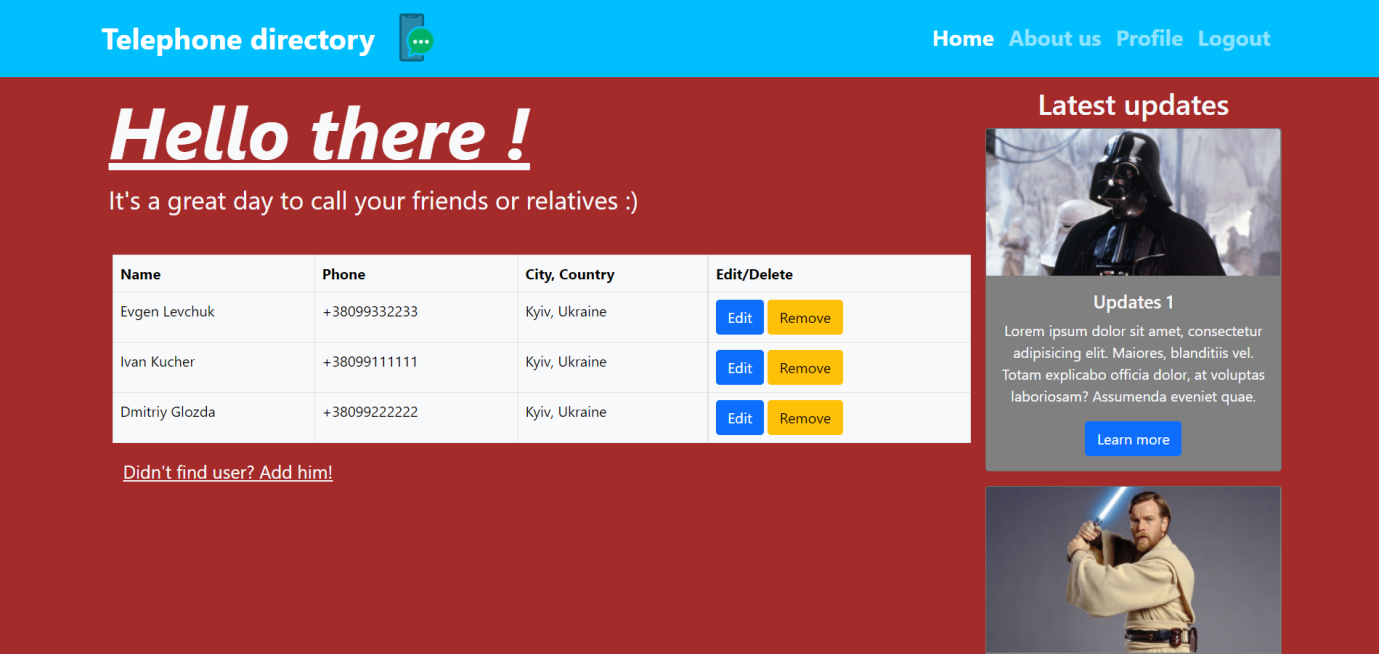
**registration.html**

****

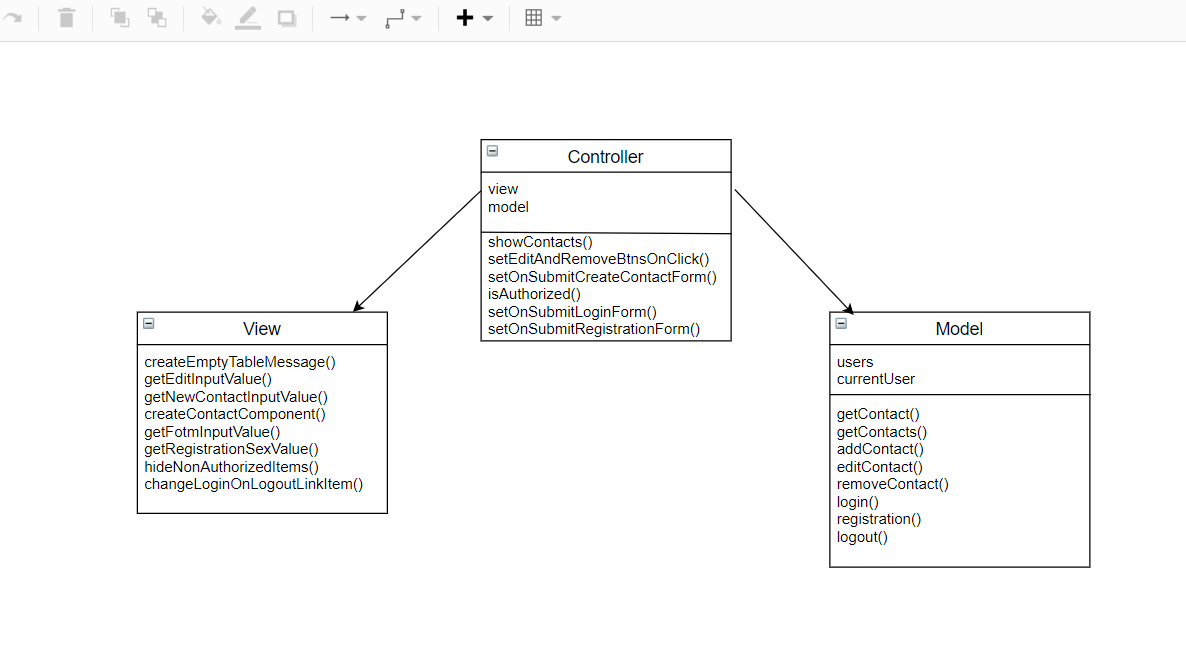
****

**about.html**

У лабораторній роботі було реалізовано авторизацію. У кожного користувача є свій username, password, email, sex та массив contacts з номерами контактної книжки, що належать користувачу.  
Якщо користувач не авторизований, йому не доступна сторінка перегляду профілю та можливість додавати.

Зображення неавторизованої головної сторінки.  
  
При авторизації напис “Sign In” змінюється на “Logout”. Вкладка тепер відповідає за вихід з профілю поточного користувача.  


**Діаграма класів**

Модуль controller.js зберігає у собі класс Controller, який зберігає методи для встановлення функціональності ВЕБ-сторінки. Для кожної сторінки був створений js-файл з викликом необхідного методу.  
  
Модуль model.js зберігає у собі класс Model, який зберігає дані та методи для роботи з ними.  
  
Модуль view.js зберігає у собі класс View, який зберігає в собі посилання та методи для отримання посилання на необхідні DOM-елементи, а також методи для створення нових DOM-елементів.   


**about.js**

App.IsAuthorized();

**Controller.js**

class Controller {

    constructor() {

        this.model = new Model();

        this.view = new View();

    }

    IsAuthorized = () => {

*if* (!this.model.currentUser) {

            this.view.hideNonAuthorizedItems();

*return* false;

        }

        const logoutBtn = this.view.changeLoginOnLogoutLinkItem();

        logoutBtn.onclick = () => {

            this.model.logout();

            location.replace('/index.html');

        };

        this.view.loginLogoutItem.innerHTML = '';

        this.view.loginLogoutItem.appendChild(logoutBtn);

*return* true;

    };

    showContacts = () => {

        const contacts = this.model.getContacts();

*if* (contacts.length) {

            contacts.map((contact, id) => {

                const contactComponent = this.view.createContactComponent({ id, ...contact });

                this.view.contactsTable.append(contactComponent);

            });

        } *else* {

            this.view.contactsTable.append(this.view.createEmptyTableMessage());

        }

    };

    setEditAndRemoveBtnsOnClick = () => {

        const editBtns = document.querySelectorAll('.edit');

        editBtns.forEach((btn, id) => {

            btn.onclick = () => {

                const name = this.view.getEditInputValue('name', id);

                const number = this.view.getEditInputValue('phone', id);

                const cityandCountry = this.view.getEditInputValue('city\_country', id);

                this.model.editContact(id, {

                    name,

                    number,

                    cityandCountry

                });

                location.reload();

            };

        });

        const removeBtns = document.querySelectorAll('.remove');

        removeBtns.forEach((btn, id) => {

            btn.onclick = () => {

                this.model.removeContact(id);

                location.reload();

            };

        });

    };

    setOnSubmitCreateContactForm = () => {

        this.view.createContactForm.addEventListener('submit', () => {

            const name = this.view.getFormInputValue('name');

            const number = this.view.getFormInputValue('phone');

            const cityandCountry = this.view.getFormInputValue('city\_country');

            this.model.addContact({ name, number, cityandCountry });

        });

    };

    setOnSubmitLoginForm = () => {

        this.view.loginForm.addEventListener('submit', () => {

            const email = this.view.getFormInputValue('email');

            const password = this.view.getFormInputValue('password');

            this.model.login(email, password);

        });

    };

    setOnSubmitRegistrateForm = () => {

        this.view.registrationForm.addEventListener('submit', e => {

            e.preventDefault();

            const username = this.view.getFormInputValue('username');

            const email = this.view.getFormInputValue('email');

            const sex = this.view.getRegistrationSexValue();

            const password = this.view.getFormInputValue('password');

            const birthday = this.view.getFormInputValue('birthday');

            console.log({ username, email, password, sex, birthday });

            this.model.registration(username, email, password, sex, birthday);

        });

    };

}

const App = new Controller();

**data.js**

const data = [

    {

        name: 'Evgen Levchuk',

        number: '+38099332233',

        cityandCountry: 'Kyiv, Ukraine'

    },

    {

        name: 'Ivan Kucher',

        number: '+38099111111',

        cityandCountry: 'Kyiv, Ukraine'

    },

    {

        name: 'Dmitriy Glozda',

        number: '+38099222222',

        cityandCountry: 'Kyiv, Ukraine'

    }

];

*if* (!localStorage.getItem('currentUser')) {

    localStorage.setItem('currentUser', null);

}

*if* (!localStorage.getItem('users')) {

    localStorage.setItem(

        'users',

        JSON.stringify([

            {

                username: 'test',

                email: 'test@test.com',

                password: '111',

                sex: 'female',

                birthday: '2000-11-11',

                contacts: data

            }

        ])

    );

}

**index.js**

*if* (App.IsAuthorized()) {

    App.showContacts();

    App.setEditAndRemoveBtnsOnClick();

}

**login.js**

App.IsAuthorized();

App.setOnSubmitLoginForm();

**Model.js**

class Model {

    constructor() {

        this.currentUser = JSON.parse(localStorage.getItem('currentUser'));

        this.users = JSON.parse(localStorage.getItem('users'));

    }

    getContact = id => {

*return* this.currentUser ? this.currentUser.contacts[id] : null;

    };

    getContacts = () => {

*return* this.currentUser ? this.currentUser.contacts : null;

    };

    addContact = contact => {

        this.currentUser.contacts.push(contact);

        localStorage.setItem('currentUser', JSON.stringify(this.currentUser));

    };

    editContact = (id, editedContact) => {

        this.currentUser.contacts[id] = editedContact;

        localStorage.setItem('currentUser', JSON.stringify(this.currentUser));

    };

    removeContact = id => {

        this.currentUser.contacts = this.contacts.filter((\_, idx) => idx !== id);

        localStorage.setItem('currentUser', JSON.stringify(this.currentUser));

    };

    login = (email, password) => {

        const currentUser = this.users.find(u => email === u.email && password === u.password);

*if* (currentUser) {

            this.currentUser = currentUser;

            localStorage.setItem('currentUser', JSON.stringify(this.currentUser));

*return* true;

        }

*return* false;

    };

    registration = (username, email, password, sex, birthday) => {

        const newUser = { username, email, password, sex, birthday, contacts: [] };

        this.users.push(newUser);

        this.currentUser = newUser;

        localStorage.setItem('currentUser', JSON.stringify(this.currentUser));

        localStorage.setItem('users', JSON.stringify(this.users));

    };

    logout = () => {

        this.currentUser = null;

        localStorage.setItem('currentUser', JSON.stringify(this.currentUser));

    };

}

**new-contact.js**

App.setOnSubmitCreateContactForm();

**register.js**

App.IsAuthorized();

App.setOnSubmitRegistrateForm();

**View.js**

class View {

    constructor() {

        this.contactsTable = document.querySelector('#contacts');

        this.createContactForm = document.querySelector('#createForm');

        this.loginForm = document.querySelector('#login');

        this.registrationForm = document.querySelector('#registration');

        this.loginLogoutItem = document.querySelector('#login-page');

    }

    getEditInputValue = (inputName, id) => document.querySelector(`#${inputName}${id}`).value;

    getFormInputValue = inputName => document.querySelector(`#${inputName}`).value;

    getRegistrationSexValue = () => [...document.querySelectorAll('.sex')].find(radio => radio.checked).value;

    hideNonAuthorizedItems = () => {

        document.querySelector('#profile').style.display = 'none';

*if* (document.querySelector('#add-contact')) {

            document.querySelector('#add-contact').style.display = 'none';

        }

    };

    createContactComponent = ({ id, name, number, cityandCountry }) => {

        const element = document.createElement('tr');

        element.innerHTML = `

                            <td>${name}</td>

                            <td>${number}</td>

                            <td>${cityandCountry}</td>

                            <td class="d-none d-md-block">

                            <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#modal${id}">

                            Edit

                            </button>

                            <button class="btn btn-warning remove">Remove</button>

                            </td>

<div class="modal fade" style="background: transparent;"  id="modal${id}" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content" >

      <div class="modal-header">

        <h5 class="modal-title" id="exampleModalLabel">Edit contact</h5>

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

      </div>

      <div class="modal-body">

        <form class="col-12 col-lg-6 rounded text-center fw-bold">

                    <div class="form-outline mb-4 fs-3">

                        <label class="form-label" for="name">Name </label>

                        <input required placeholder="Name" type="text" id="name${id}" class="form-control" value="${name}"/>

                    </div>

                    <div class="form-outline mb-4 fs-3">

                        <label class="form-label" for="phone">Phone number </label>

                        <input required placeholder="Phone number" type="tel" id="phone${id}" class="form-control" value="${number}"/>

                    </div>

                    <div class="form-outline mb-4 fs-3">

                        <label class="form-label" for="city\_country">City, Country </label>

                        <input required placeholder="City, Country" type="text" id="city\_country${id}"

                            class="form-control" value="${cityandCountry}"/>

                    </div>

                </form>

      </div>

      <div class="modal-footer">

        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

        <button type="button" class="btn btn-primary edit">Save changes</button>

      </div>

    </div>

  </div>

</div>

`;

*return* element;

    };

    createEmptyTableMessage = () => {

        const message = document.createElement('p');

        message.innerHTML = 'No contacts.';

        message.classList.add('display-5', 'fw-bold', 'text-light');

*return* message;

    };

    changeLoginOnLogoutLinkItem = () => {

        const elem = document.createElement('a');

        elem.classList.add('nav-link');

        elem.innerHTML = 'Logout';

*return* elem;

    };

}