НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування і спеціалізованих комп’ютерних систем**

**ЛАБОРАТОРНА РОБОТА № 1**

з дисципліни

**«** **Web-дизайн»**

**Тема:** Розробка статичного інтерфейсу Web-додатка

Виконала Лукащук Ю. А.

Студентка групи КВ-22

github: <https://github.com/yuliya2409/WebD>

**telegram: @sunshine\_o9**

**Мета роботи:**

Ознайомитись із засобами мови Javascript та навчитись їх застосовувати побудови Web-інтерфейсу користувача.

**Постановка задачі:**

1. Запропонувати тематику Web-додатку (див. рекомендований список).
2. Ознайомившись з переліком обов’язкових сторінок, виконати їх програмування:
3. Встановити середовище розробки (рекомендовано VS Code)
4. Ознайомитись з методичними вказівками щодо реалізації шаблону MVC
5. Виконати кодування додатку згідно із загальноприйнятими вимогами (див. https://github.com/airbnb/javascript).
6. Організувати код у вигляді модулів (класів) шаблону MVC (див. приклад додатку ToDo: https://github.com/Aroxed/js-todo-mvc).
7. Як сховище даних використовувати структури даних Javascript: масиви, об’єкти, рядки тощо або localStorage.
8. За бажанням студента доповнити сайт іншими сторінками.
9. Створити репозиторій та завантажити усі файли на github.com.

**Контакти:**

Github-репозиторій:

Github Pages:

Telegram:

**Опис додатка:**

Назва: Dictionary

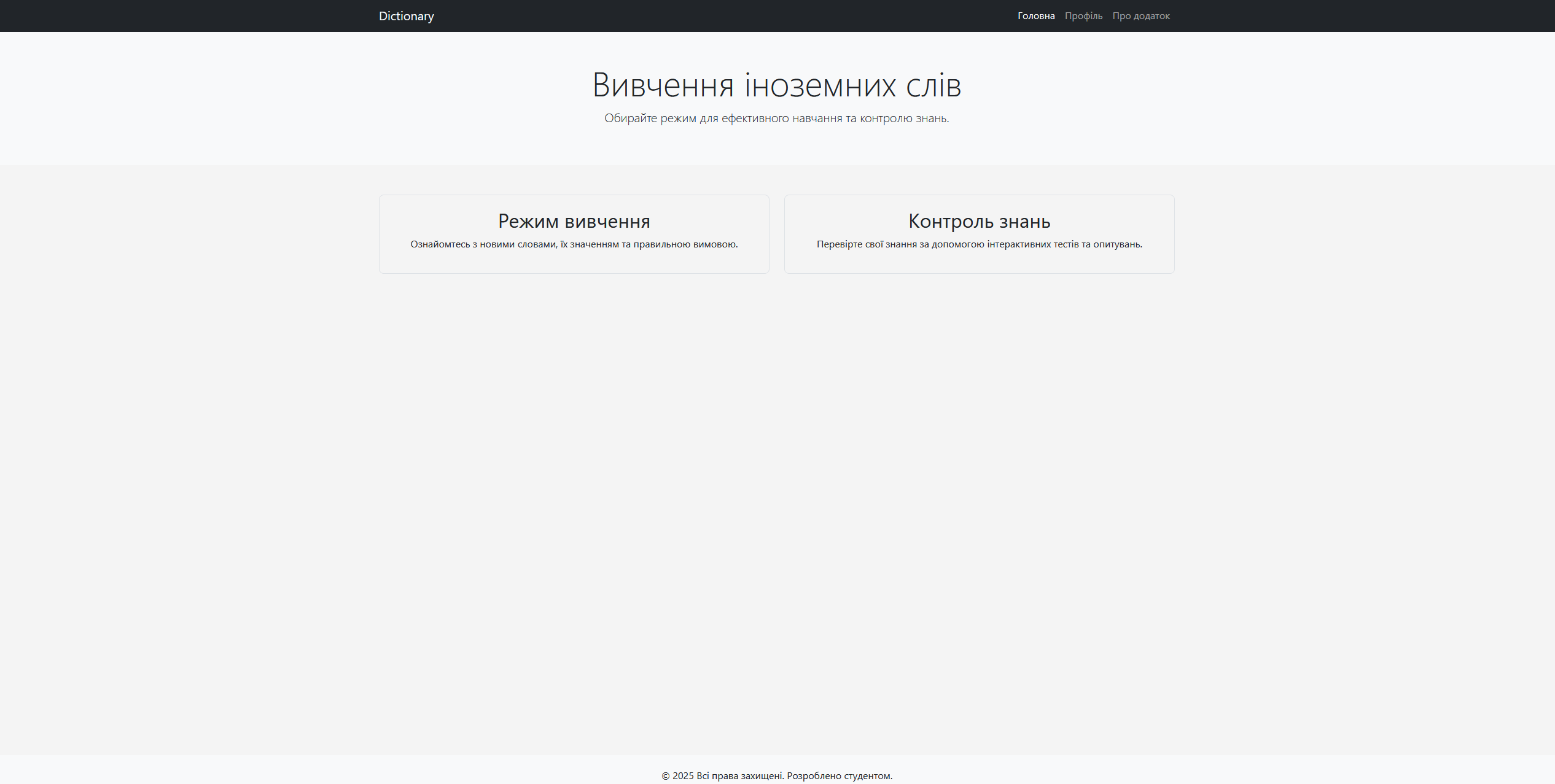
Додаток створений для полегшення вивчення іноземних мов. Можна ознайомитись з новими словами, практикувати їх за допомогою інтерактивних вправ, та контролювати знання за допомогою тестів.

**Схема переходу:**

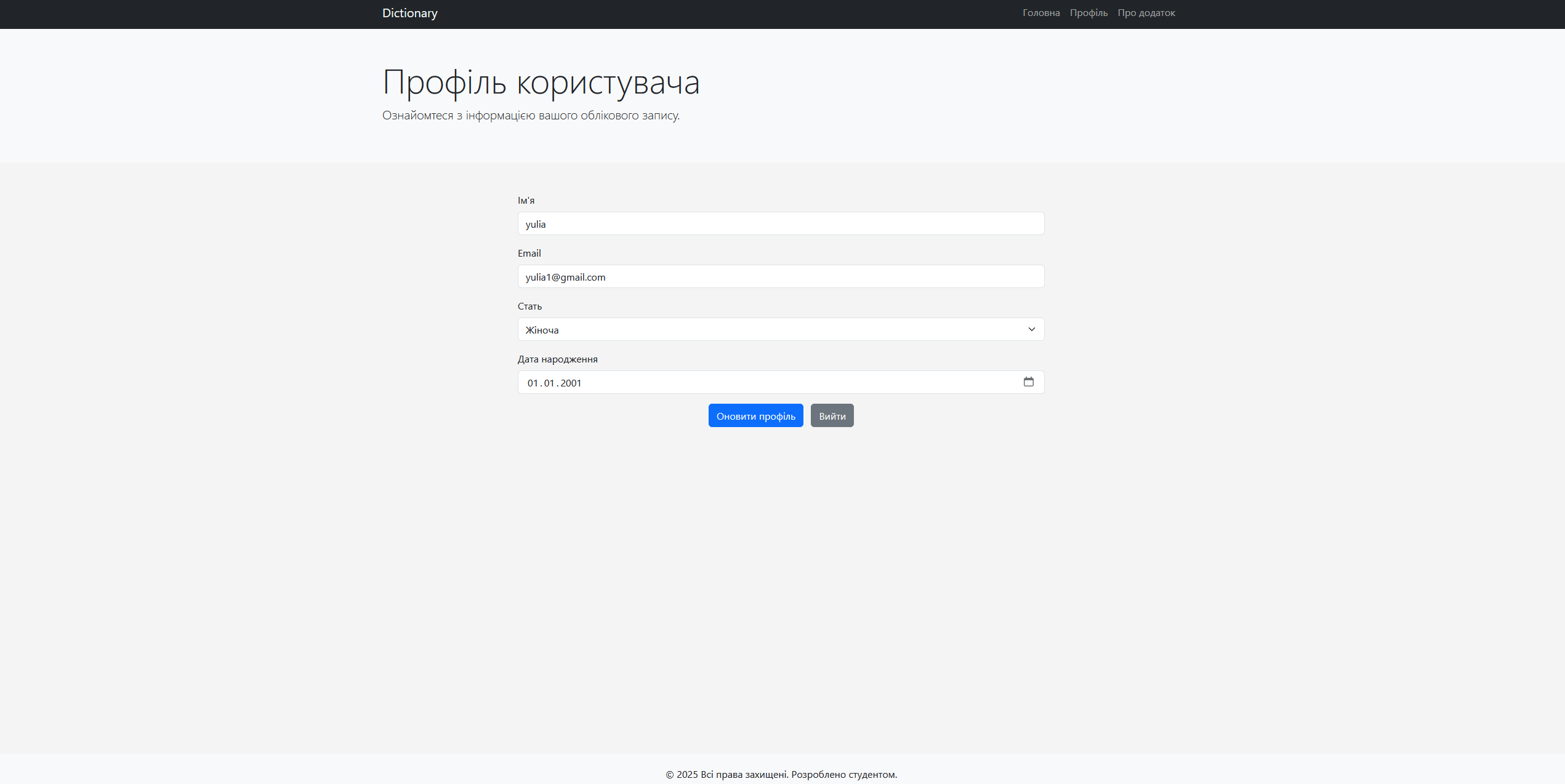
З будь-якої сторінки сайту можна перейти на будь-яку іншу сайту (тільки з головної можна потрапити на режим вивчення та контроль знань).

**Копії екрана усіх сторінок додатка:**

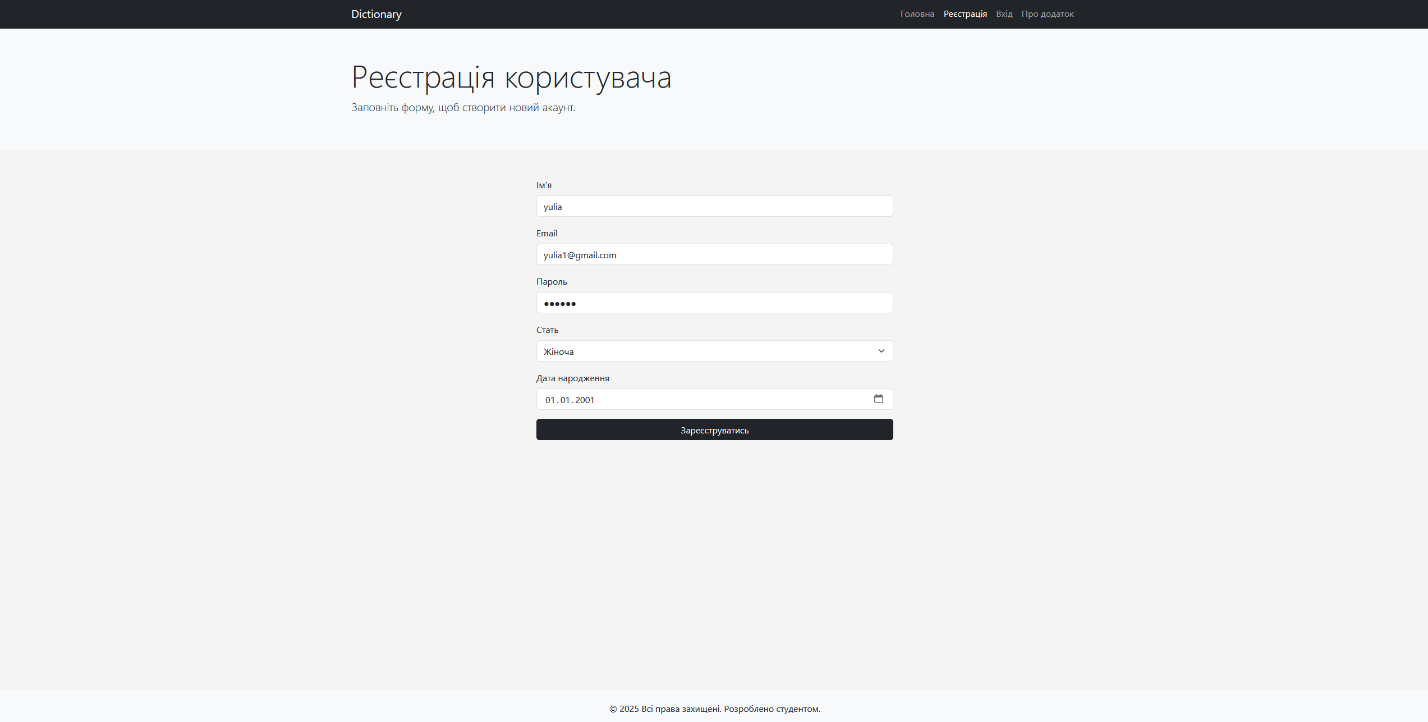
Робоча сторінка (головна):



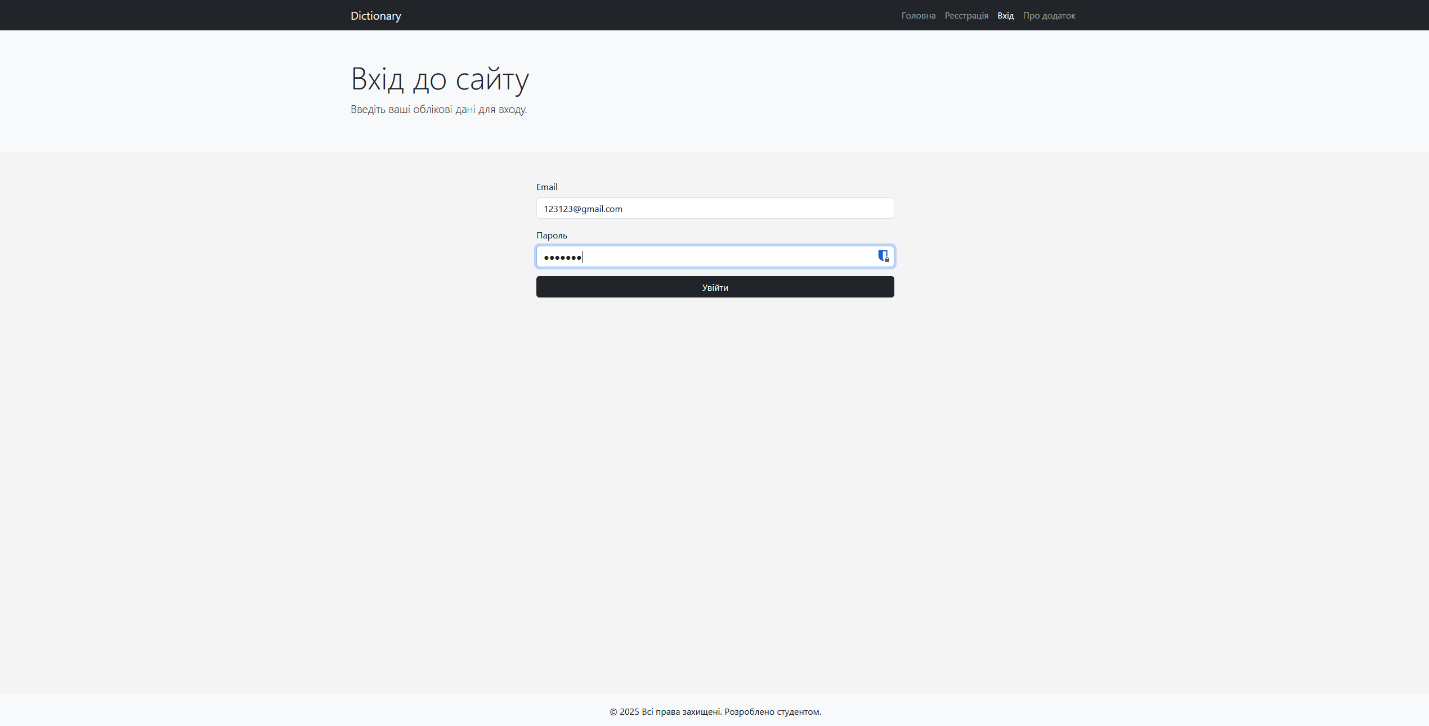
Профіль:



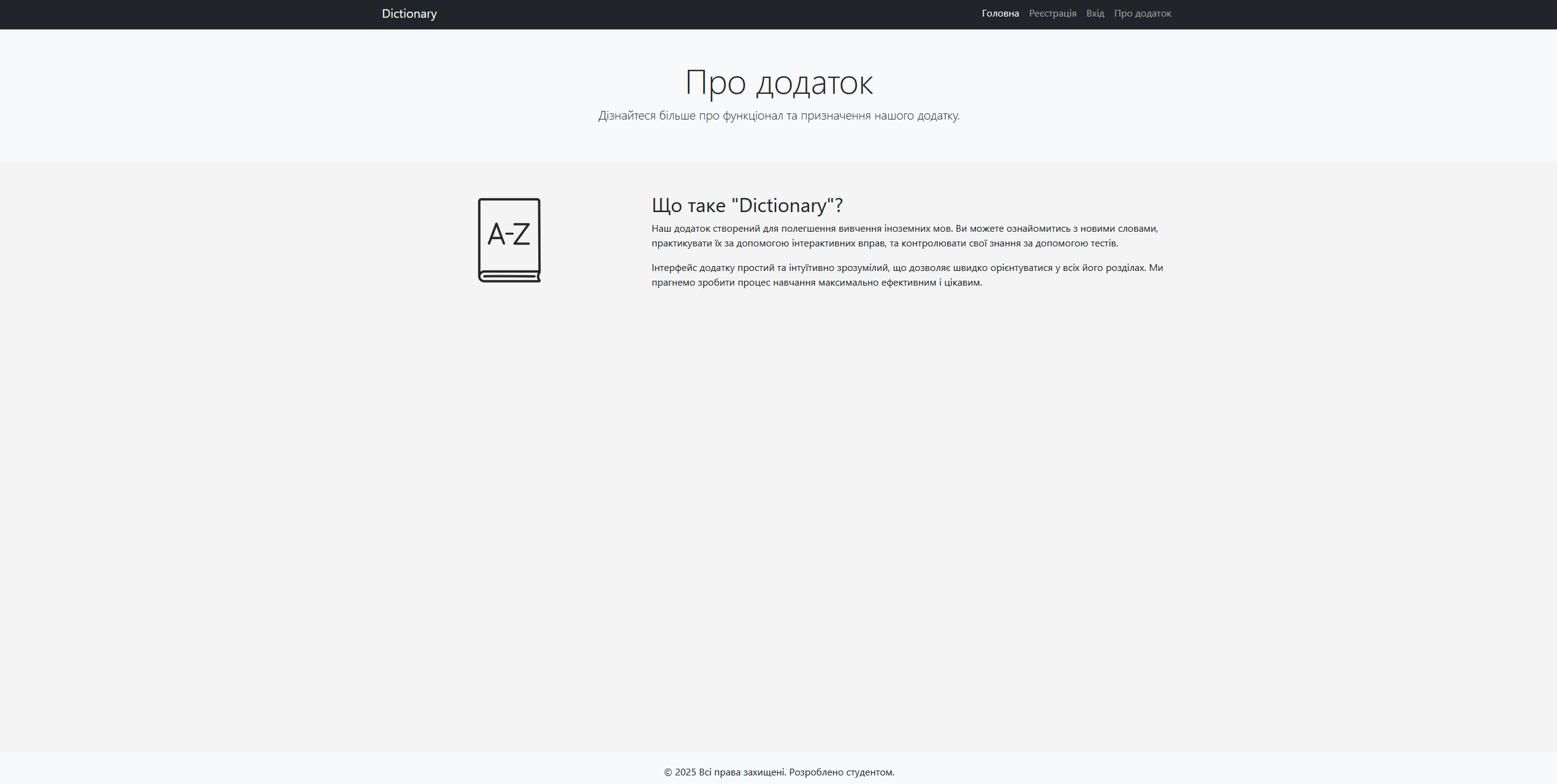
Реєстрація:



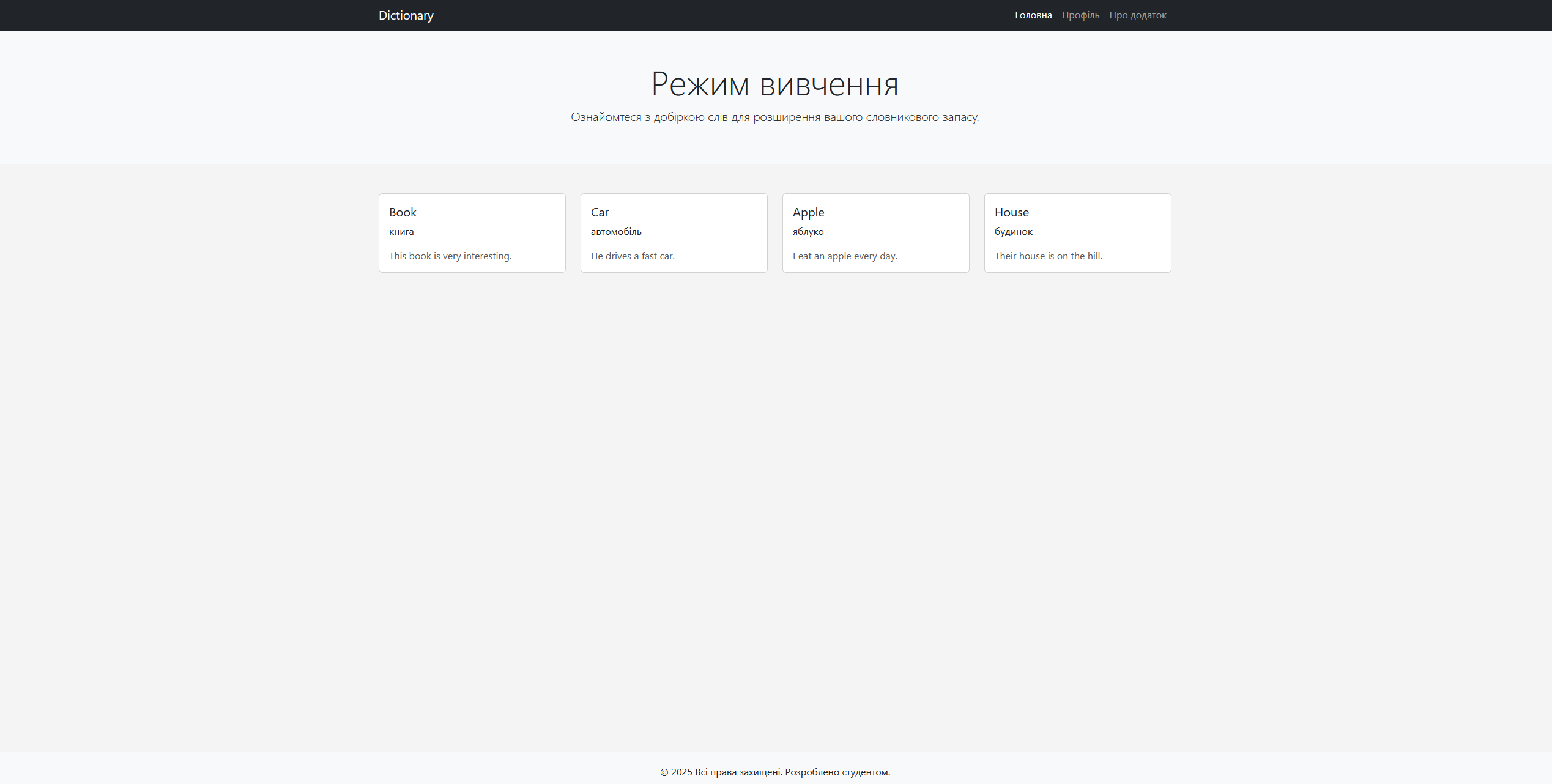
Вхід:



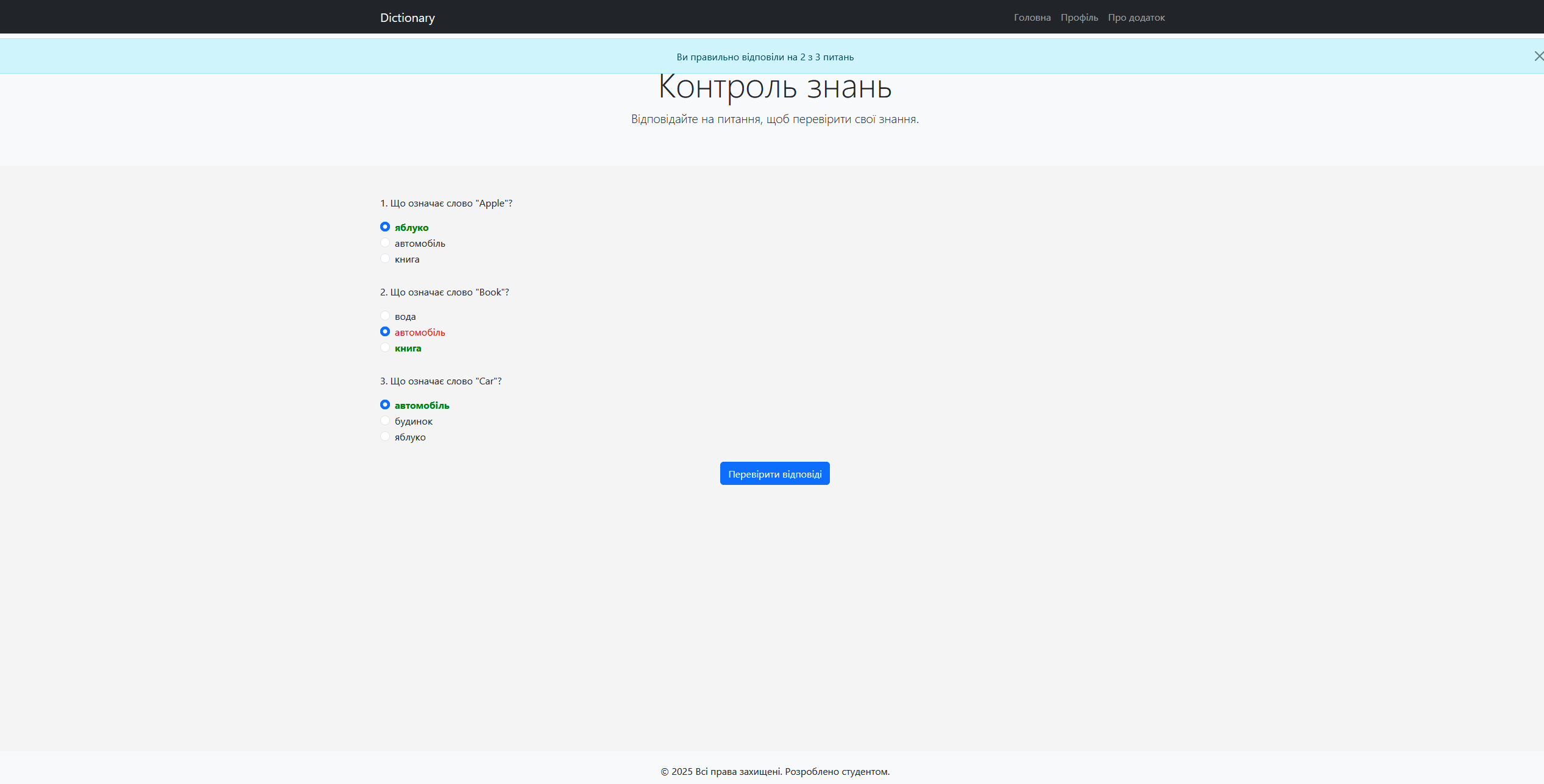
Про додаток:



Режим вивчення:



Контроль знань:



**Javascript-код:**

***app.js:***

import UserController from './controllers/UserController.js';

import WordController from './controllers/WordController.js';

import UserView from './views/UserView.js';

import WordView from './views/WordView.js';

class DictionaryApp {

    constructor() {

        this.initControllers();

        this.initViews();

        this.checkAuthStatus();

    }

    initControllers() {

        this.userController = UserController;

        this.wordController = WordController;

    }

    initViews() {

        this.userView = new UserView(this.userController);

        this.wordView = new WordView(this.wordController);

    }

    checkAuthStatus() {

        const currentUser = this.userController.getCurrentUser();

        const currentPath = window.location.pathname;

        const protectedRoutes = [

            '/profile.html',

            '/study.html',

            '/quiz.html'

        ];

        const guestRoutes = [

            '/login.html',

            '/registration.html'

        ];

        if (currentUser) {

            if (guestRoutes.some(route => currentPath.endsWith(route))) {

                window.location.href = '/main.html';

            }

        } else {

            if (protectedRoutes.some(route => currentPath.endsWith(route))) {

                window.location.href = '/login.html';

            }

        }

        this.updateNavbarVisibility(currentUser);

    }

    updateNavbarVisibility(currentUser) {

        const navItems = document.querySelectorAll('.navbar-nav .nav-item');

        navItems.forEach(item => {

            const link = item.querySelector('.nav-link');

            const href = link.getAttribute('href');

            if (currentUser) {

                if (href.includes('login.html') || href.includes('registration.html')) {

                    item.style.display = 'none';

                }

            }

            else {

                if (href.includes('profile.html')) {

                    item.style.display = 'none';

                }

            }

        });

    }

}

document.addEventListener('DOMContentLoaded', () => {

    window.app = new DictionaryApp();

});

export default DictionaryApp;

***UserModel.js & WordModel.js:***

class UserModel {

    constructor() {

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

    }

    register(userData) {

        const existingUser = this.users.find(user => user.email === userData.email);

        if (existingUser) {

            throw new Error('Користувач з такою електронною поштою вже існує');

        }

        this.users.push(userData);

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

        return userData;

    }

    login(email, password) {

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

        if (!user) {

            throw new Error('Невірний email або пароль');

        }

        return user;

    }

    getCurrentUser() {

        return JSON.parse(localStorage.getItem('currentUser')) || null;

    }

    setCurrentUser(user) {

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

    }

    logout() {

        localStorage.removeItem('currentUser');

    }

    updateProfile(updatedData) {

        const currentUser = this.getCurrentUser();

        if (!currentUser) {

            throw new Error('Користувач не авторизований');

        }

        const updatedUser = { ...currentUser, ...updatedData };

        this.setCurrentUser(updatedUser);

        const userIndex = this.users.findIndex(u => u.email === currentUser.email);

        if (userIndex !== -1) {

            this.users[userIndex] = updatedUser;

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

        }

        return updatedUser;

    }

}

export default new UserModel();

class WordModel {

    constructor() {

        this.words = [

            { english: 'Apple', ukrainian: 'яблуко', usage: 'I eat an apple every day.', translation: 'Я щодня їм яблуко.' },

            { english: 'Book', ukrainian: 'книга', usage: 'This book is very interesting.', translation: 'Ця книга дуже цікава.' },

            { english: 'House', ukrainian: 'будинок', usage: 'Their house is on the hill.', translation: 'Їхній будинок знаходиться на пагорбі.' },

            { english: 'Car', ukrainian: 'автомобіль', usage: 'He drives a fast car.', translation: 'Він водить швидкий автомобіль.' },

            { english: 'Water', ukrainian: 'вода', usage: 'Water is essential for life.', translation: 'Вода необхідна для життя.' },

            { english: 'Table', ukrainian: 'стіл', usage: 'The book is on the table.', translation: 'Книга на столі.' },

            { english: 'Chair', ukrainian: 'стілець', usage: 'He sat on the chair.', translation: 'Він сів на стілець.' },

            { english: 'Sun', ukrainian: 'сонце', usage: 'The sun is shining.', translation: 'Світить сонце.' }

        ];

    }

    getAllWords() {

        return this.words;

    }

    getRandomWords(count = 4) {

        return this.\_shuffleAndTake(this.words, count);

    }

    \_shuffleAndTake(array, count) {

        return [...array]

            .sort(() => 0.5 - Math.random())

            .slice(0, count);

    }

    generateOptions(correctWord) {

        const otherWords = this.words.filter(w => w.english !== correctWord.english);

        const wrongOptions = this.\_shuffleAndTake(otherWords, 2).map(w => w.ukrainian);

        return [correctWord.ukrainian, ...wrongOptions].sort(() => 0.5 - Math.random());

    }

    generateQuizQuestions(count = 3) {

        const quizWords = this.getRandomWords(count);

        return quizWords.map((word, idx) => ({

            id: idx + 1,

            question: `Що означає слово "${word.english}"?`,

            correctAnswer: word.ukrainian,

            options: this.generateOptions(word)

        }));

    }

}

export default new WordModel();

***UserView.js & WordView.js:***

class UserView {

    constructor(controller) {

        this.controller = controller;

        this.initEventListeners();

    }

    initEventListeners() {

        const registrationForm = document.querySelector('form[action="registration"]');

        if (registrationForm) {

            registrationForm.addEventListener('submit', this.handleRegistration.bind(this));

        }

        const loginForm = document.querySelector('form[action="login"]');

        if (loginForm) {

            loginForm.addEventListener('submit', this.handleLogin.bind(this));

        }

        const profileForm = document.querySelector('form[action="profile"]');

        if (profileForm) {

            this.populateProfileForm();

            profileForm.addEventListener('submit', this.handleProfileUpdate.bind(this));

        }

        const logoutBtn = document.getElementById('logout-btn');

        if (logoutBtn) {

            logoutBtn.addEventListener('click', this.handleLogout.bind(this));

        }

    }

    handleRegistration(event) {

        event.preventDefault();

        const formData = this.getFormData(event.target);

        try {

            this.controller.register(formData);

            this.showMessage('Реєстрація успішна! Ви можете увійти.', 'success');

            setTimeout(() => window.location.href = 'login.html', 2000);

        } catch (error) {

            this.showMessage(error.message, 'danger');

        }

    }

    handleLogin(event) {

        event.preventDefault();

        const email = document.getElementById('loginEmail').value;

        const password = document.getElementById('loginPassword').value;

        try {

            this.controller.login(email, password);

            this.showMessage('Вхід успішний!', 'success');

            setTimeout(() => window.location.href = 'profile.html', 2000);

        } catch (error) {

            this.showMessage(error.message, 'danger');

        }

    }

    handleProfileUpdate(event) {

        event.preventDefault();

        const formData = this.getFormData(event.target);

        try {

            this.controller.updateProfile(formData);

            this.showMessage('Профіль успішно оновлено!', 'success');

        } catch (error) {

            this.showMessage(error.message, 'danger');

        }

    }

    handleLogout() {

        this.controller.logout();

        window.location.href = 'login.html';

    }

    populateProfileForm() {

        const user = this.controller.getCurrentUser();

        if (user) {

            document.getElementById('name').value = user.name || '';

            document.getElementById('email').value = user.email || '';

            document.getElementById('gender').value = user.gender || '';

            document.getElementById('dob').value = user.dob || '';

        }

    }

    getFormData(form) {

        const formData = {};

        const inputs = form.querySelectorAll('input, select');

        inputs.forEach(input => {

            if (input.id) {

                formData[input.id] = input.value;

            }

        });

        return formData;

    }

    showMessage(message, type = 'info') {

        let messageContainer = document.getElementById('message-container');

        if (!messageContainer) {

            messageContainer = document.createElement('div');

            messageContainer.id = 'message-container';

            messageContainer.className = 'fixed-top text-center mt-5 pt-3';

            document.body.insertBefore(messageContainer, document.body.firstChild);

        }

        const messageElement = document.createElement('div');

        messageElement.className = `alert alert-${type} alert-dismissible fade show`;

        messageElement.innerHTML = `

            ${message}

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

        `;

        messageContainer.innerHTML = '';

        messageContainer.appendChild(messageElement);

    }

}

export default UserView;

class WordView {

    constructor(controller) {

        this.controller = controller;

        this.questions = [];

        this.initEventListeners();

    }

    initEventListeners() {

        if (window.location.pathname.endsWith('study.html')) {

            this.populateStudyWords();

        }

        if (window.location.pathname.endsWith('quiz.html')) {

            this.generateQuizQuestions();

            this.setupQuizSubmission();

        }

    }

    populateStudyWords() {

        const studyWordsContainer = document.querySelector('.study-words');

        if (!studyWordsContainer) return;

        try {

            const words = this.controller.getRandomWords();

            studyWordsContainer.innerHTML = words.map(word => `

                <div class="col-md-3">

                    <div class="card word-card">

                        <div class="card-body">

                            <h5 class="card-title">${word.english}</h5>

                            <p class="card-text">${word.ukrainian}</p>

                            <p class="card-text text-muted">${word.usage}</p>

                        </div>

                    </div>

                </div>

            `).join('');

        } catch (error) {

            console.error('Error populating study words:', error);

            this.showMessage('Не вдалося завантажити слова', 'danger');

        }

    }

    generateQuizQuestions() {

        const quizContainer = document.querySelector('.quiz-container');

        if (!quizContainer) return;

        try {

            // Генерируем и сохраняем вопросы

            this.questions = this.controller.generateQuizQuestions();

            // Рендерим

            quizContainer.innerHTML = this.questions.map((q, idx) => `

                <div class="quiz-question">

                    <p class="mb-3">${idx + 1}. ${q.question}</p>

                    ${q.options.map((opt, oi) => `

                        <div class="form-check">

                            <input class="form-check-input" type="radio"

                                   name="question${idx}"

                                   id="question${idx}\_option${oi}"

                                   value="${opt}">

                            <label class="form-check-label" for="question${idx}\_option${oi}">

                                ${opt}

                            </label>

                        </div>

                    `).join('')}

                </div>

            `).join('');

        } catch (err) {

            console.error('Error generating quiz questions:', err);

            this.showMessage('Не вдалося створити тест', 'danger');

        }

    }

    setupQuizSubmission() {

        const checkBtn = document.querySelector('.btn-primary');

        if (!checkBtn) return;

        checkBtn.addEventListener('click', () => {

            const questionElems = document.querySelectorAll('.quiz-question');

            let correctCount = 0;

            questionElems.forEach((elem, idx) => {

                const selected = elem.querySelector('input:checked');

                const labels   = elem.querySelectorAll('.form-check-label');

                const correct  = this.questions[idx].correctAnswer;

                // Сбрасываем стили

                labels.forEach(l => l.classList.remove('text-success','text-danger'));

                if (selected) {

                    const label = selected.nextElementSibling;

                    if (selected.value === correct) {

                        label.classList.add('text-success');

                        correctCount++;

                    } else {

                        label.classList.add('text-danger');

                        // подсветить правильный

                        labels.forEach(l => {

                            if (l.textContent.trim() === correct) {

                                l.classList.add('text-success');

                            }

                        });

                    }

                }

            });

            this.showMessage(`Ви правильно відповіли на ${correctCount} з ${this.questions.length} питань`, 'info');

        });

    }

    showMessage(message, type = 'info') {

        let messageContainer = document.getElementById('message-container');

        if (!messageContainer) {

            messageContainer = document.createElement('div');

            messageContainer.id = 'message-container';

            messageContainer.className = 'fixed-top text-center mt-5 pt-3';

            document.body.insertBefore(messageContainer, document.body.firstChild);

        }

        const messageElement = document.createElement('div');

        messageElement.className = `alert alert-${type} alert-dismissible fade show`;

        messageElement.innerHTML = `

            ${message}

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

        `;

        messageContainer.innerHTML = '';

        messageContainer.appendChild(messageElement);

    }

}

export default WordView;

***UserController.js & WordController.js:***

import UserModel from '../models/UserModel.js';

class UserController {

    constructor() {

        this.model = UserModel;

    }

    register(userData) {

        this.validateRegistrationData(userData);

        userData.password = this.hashPassword(userData.password);

        return this.model.register(userData);

    }

    login(email, password) {

        if (!email || !password) {

            throw new Error('Введіть email та пароль');

        }

        const hashedPassword = this.hashPassword(password);

        const user = this.model.login(email, hashedPassword);

        this.model.setCurrentUser(user);

        return user;

    }

    logout() {

        this.model.logout();

    }

    getCurrentUser() {

        return this.model.getCurrentUser();

    }

    updateProfile(userData) {

        this.validateProfileUpdate(userData);

        return this.model.updateProfile(userData);

    }

    validateRegistrationData(userData) {

        const requiredFields = ['name', 'email', 'password', 'gender', 'dob'];

        requiredFields.forEach(field => {

            if (!userData[field]) {

                throw new Error(`Поле ${field} є обов'язковим`);

            }

        });

        const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

        if (!emailRegex.test(userData.email)) {

            throw new Error('Невірний формат електронної пошти');

        }

        if (userData.password.length < 6) {

            throw new Error('Пароль повинен містити принаймні 6 символів');

        }

    }

    validateProfileUpdate(userData) {

        if (userData.email) {

            const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

            if (!emailRegex.test(userData.email)) {

                throw new Error('Невірний формат електронної пошти');

            }

        }

    }

    hashPassword(password) {

        let hash = 0;

        for (let i = 0; i < password.length; i++) {

            const char = password.charCodeAt(i);

            hash = ((hash << 5) - hash) + char;

            hash = hash & hash;

        }

        return hash.toString();

    }

}

export default new UserController();

import WordModel from '../models/WordModel.js';

class WordController {

    constructor() {

        this.model = WordModel;

    }

    getAllWords() {

        return this.model.getAllWords();

    }

    getRandomWords(count = 4) {

        return this.model.getRandomWords(count);

    }

    generateQuizQuestions(count = 3) {

        return this.model.generateQuizQuestions(count);

    }

    getQuizCorrectAnswer(questionIndex) {

        return this.model.getQuizCorrectAnswer(questionIndex);

    }

    addWord(wordData) {

        this.validateWordData(wordData);

        throw new Error('Додавання слів наразі не підтримується');

    }

    validateWordData(wordData) {

        const requiredFields = ['english', 'ukrainian', 'usage', 'translation'];

        requiredFields.forEach(field => {

            if (!wordData[field]) {

                throw new Error(`Поле ${field} є обов'язковим`);

            }

        });

        if (wordData.english.length < 1 || wordData.ukrainian.length < 1) {

            throw new Error('Слова не можуть бути порожніми');

        }

    }

}

export default new WordController();