Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра МБІС

**ЛАБОРАТОРНА РОБОТА №3**

з дисципліни «Засоби програмування та захисту веб-додатків»

на тему: «Розробка web-додатку на Flask»

Студента 2 курсу групи 3КІТС-23б

напряму підготовки 125

спеціальності «Кібербезпека та захист інформації»

Яцюка Дениса Володимировича

Перевірив: Присяжний Дмитро Петрович

Вінниця 2025

**Мета роботи:** *Створити повнофункціональний веб-додаток для ведення рейтингу хакерів з можливостями реєстрації, авторизації, управління профілями та адміністрування системи..*

**1 РОЗРОБКА СТРУКТУРИ ДОДАТКУ**

У процесі розробки було створено веб-додаток на основі фреймворку Flask Основний файл додатку app.py містить усю логіку контролерів та маршрутизацію. Для стилізації застосовано кібер-панк дизайн з використанням пікселізованого шрифту "Press Start 2P" та неонових кольорів.

Створено базовий шаблон base.html, який містить навігацію та загальну структуру сторінок:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Hacker Leaderboard</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

<link href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap" rel="stylesheet">

</head>

<body>

<header>

<nav>

<ul>

<li><a href="{{ url\_for('index') }}">Home</a></li>

{% if session.get('logged\_in') %}

<li><a href="{{ url\_for('edit\_profile') }}">Edit Profile</a></li>

<li><a href="{{ url\_for('logout') }}">Logout</a></li>

{% endif %}

</ul>

</nav>

</header>

...

</body>

</html>

Головна сторінка index.html відображає таблицю рейтингу з можливістю надавання поваги (respect) іншим користувачам:

<table>

<thead>

<tr>

<th>Username</th>

<th>Country</th>

<th>Rank</th>

<th>Challenges Completed</th>

<th>Respect</th>

</tr>

</thead>

<tbody>

{% for entry in leaderboard %}

<tr>

<td>{{ entry.username }}</td>

<td>{{ entry.country }}</td>

<td>{{ entry.rank }}</td>

...

</tr>

{% endfor %}

</tbody>

</table>

Система ранжування реалізована через функцію get\_rank(), яка визначає рівень користувача на основі кількості виконаних завдань:

def get\_rank(challenges\_completed):

challenges\_completed = int(challenges\_completed)

if challenges\_completed >= 30:

return "Legend"

elif challenges\_completed >= 20:

return "Guru"

elif challenges\_completed >= 12:

return "Elite Hacker"

elif challenges\_completed >= 6:

return "Pro Hacker"

elif challenges\_completed >= 3:

return "Hacker"

else:

return "Script Kiddie"

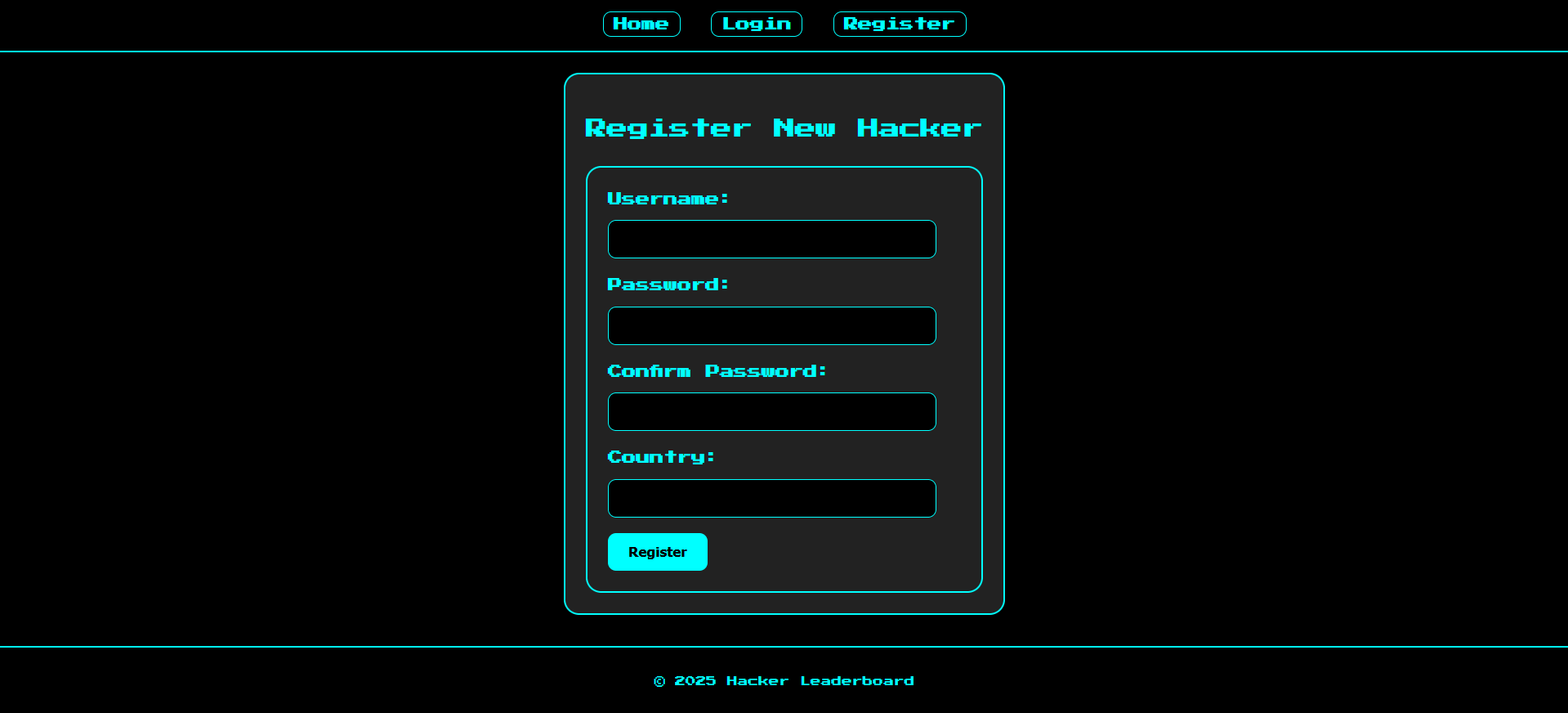


Рисунок 1.1 – Вікно авторизації

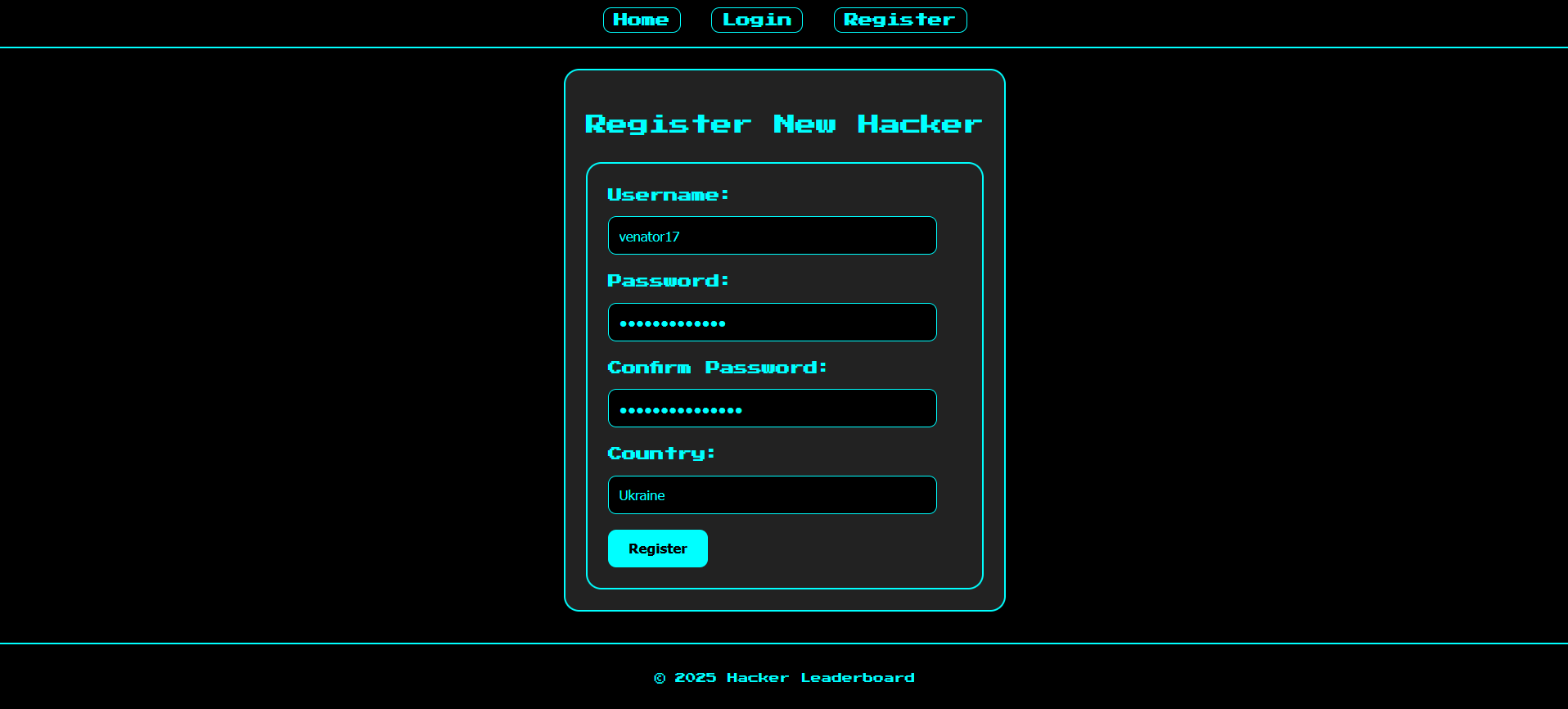


Рисунок 1.2 – Вікно реєстрації



Рисунок 1.3 – Головна сторінка

На рисунках 1.1 – 1.3 зображено головні сторінки вебдодатку, які повністю працезнадтні та виконують поставлені на них функції.

# 2 НАПОВНЕННЯ ДОДАТКУ

Для цього проєкту була використана БД SQLite, оскільки вона не потребує окремого сервера та ідеально підходить для навчальних проєктів. Створено базу даних із назвою hacker\_leaderboard.db. Таблиці бази даних повністю відповідають функціональним вимогам системи рейтингу хакерів.

Структура бази даних включає три основні моделі (users, leaderboard, likes):

db.execute('''

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT UNIQUE NOT NULL,

password TEXT NOT NULL,

is\_admin INTEGER DEFAULT 0

)

''')

db.execute('''

CREATE TABLE IF NOT EXISTS leaderboard (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT UNIQUE NOT NULL,

country TEXT NOT NULL,

challenges\_completed INTEGER DEFAULT 0,

respect INTEGER DEFAULT 0,

user\_id INTEGER,

FOREIGN KEY (user\_id) REFERENCES users (id)

)

''')

db.execute('''

CREATE TABLE IF NOT EXISTS likes (

id INTEGER PRIMARY KEY AUTOINCREMENT,

liker\_user\_id INTEGER NOT NULL,

liked\_leaderboard\_id INTEGER NOT NULL,

FOREIGN KEY (liker\_user\_id) REFERENCES users (id),

FOREIGN KEY (liked\_leaderboard\_id) REFERENCES leaderboard (id),

UNIQUE(liker\_user\_id, liked\_leaderboard\_id)

)

''')

Ініціалізація бази даних відбувається через функцію init\_db(), яка також створює адміністративний обліковий запис з логіном "admin" та паролем "super\_secure\_admin\_password". Для роботи з базою даних використовується контекстний менеджер Flask через функції get\_db() та close\_connection().створити, яке кодування символів використовувати, які таблиці створити та з якими полями.

def get\_db():

db = getattr(g, '\_database', None)

if db is None:

db = g.\_database = sqlite3.connect(DATABASE)

db.row\_factory = sqlite3.Row

return db

@app.teardown\_appcontext

def close\_connection(exception):

db = getattr(g, '\_database', None)

if db is not None:

db.close()

# 3 РОЗРОБКА ФУНКЦІОНАЛУ ДОДАТКУ

## 3.1 CRUD

Система CRUD операцій реалізована для всіх основних сутностей додатку. Головний контролер index() забезпечує відображення рейтингової таблиці з сортуванням за кількістю поваги та виконаних завдань:

@app.route('/')

def index():

db = get\_db()

leaderboard\_entries = db.execute('SELECT \* FROM leaderboard ORDER BY respect DESC, challenges\_completed DESC').fetchall()

leaderboard\_data = []

current\_user\_id = session.get('user\_id')

for entry in leaderboard\_entries:

entry\_dict = dict(entry)

entry\_dict['rank'] = get\_rank(entry\_dict['challenges\_completed'])

if current\_user\_id:

has\_liked = db.execute(

'SELECT 1 FROM likes WHERE liker\_user\_id = ? AND liked\_leaderboard\_id = ?',

(current\_user\_id, entry\_dict['id'])

).fetchone()

entry\_dict['has\_liked'] = bool(has\_liked)

leaderboard\_data.append(entry\_dict)

## 3.2 Авторизація та реєстрація

Форма реєстрації register.html включає валідацію паролів та створення записів у двох таблицях одночасно:

<form method="post">

<div>

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

</div>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

</div>

<div>

<label for="confirm\_password">Confirm Password:</label>

<input type="password" id="confirm\_password" name="confirm\_password" required>

</div>

<div>

<label for="country">Country:</label>

<input type="text" id="country" name="country" required>

</div>

<button type="submit">Register</button>

</form>

return render\_template('index.html', leaderboard=leaderboard\_data)

Контролер реєстрації включає валідацію та хешування паролів:

@app.route('/register', methods=['GET', 'POST'])

def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

confirm\_password = request.form['confirm\_password']

country = request.form['country']

if password != confirm\_password:

flash('Passwords do not match. Please try again.', 'danger')

return render\_template('register.html')

hashed\_password = hash\_password(password)

db = get\_db()

try:

cursor = db.execute("INSERT INTO users (username, password) VALUES (?, ?)",

(username, hashed\_password))

user\_id = cursor.lastrowid

db.execute("INSERT INTO leaderboard (username, country, challenges\_completed, respect, user\_id) VALUES (?, ?, ?, ?, ?)",

(username, country, 0, 0, user\_id))

db.commit()

flash('Registration successful! Please log in.', 'success')

return redirect(url\_for('login'))

except sqlite3.IntegrityError:

flash('Username already exists. Please choose a different one.', 'danger')

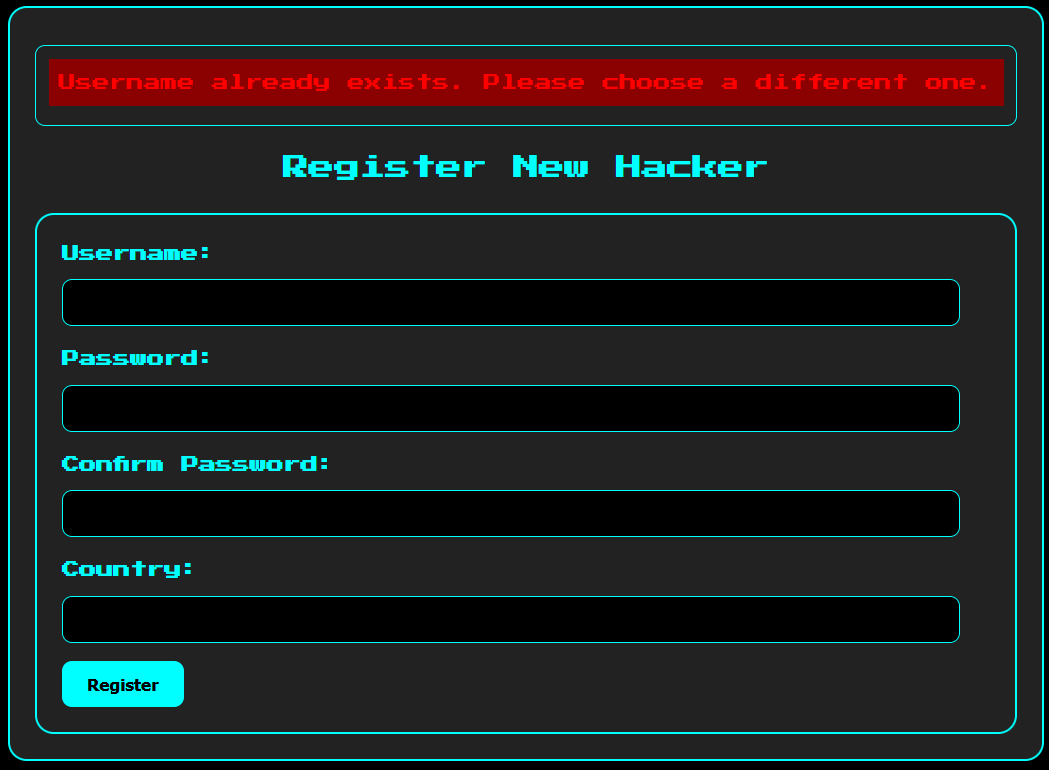


Рисунок 1.4 – Верифікація існуючого користувача

Авторизація реалізована через перевірку хешованих паролів та створення сесії:

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

db = get\_db()

user = db.execute('SELECT \* FROM users WHERE username = ?', (username,)).fetchone()

if user and check\_password(user['password'], password):

session['logged\_in'] = True

session['user\_id'] = user['id']

session['username'] = user['username']

session['is\_admin'] = bool(user['is\_admin'])

flash('Logged in successfully!', 'success')

return redirect(url\_for('index'))

else:

flash('Invalid username or password.', 'danger')

Функціональність надання поваги реалізована через систему лайків з унікальними обмеженнями:

@app.route('/respect/<int:leaderboard\_entry\_id>', methods=['POST'])

def respect\_entry(leaderboard\_entry\_id):

if not session.get('logged\_in'):

flash('You need to be logged in to give respect.', 'danger')

return redirect(url\_for('login'))

liker\_user\_id = session['user\_id']

db = get\_db()

existing\_like = db.execute(

'SELECT 1 FROM likes WHERE liker\_user\_id = ? AND liked\_leaderboard\_id = ?',

(liker\_user\_id, leaderboard\_entry\_id)

).fetchone()

if existing\_like:

flash('You have already given respect to this entry.', 'warning')

else:

db.execute('INSERT INTO likes (liker\_user\_id, liked\_leaderboard\_id) VALUES (?, ?)',

(liker\_user\_id, leaderboard\_entry\_id))

db.execute('UPDATE leaderboard SET respect = respect + 1 WHERE id = ?',

(leaderboard\_entry\_id,))

db.commit()

flash('Respect given!', 'success')

@Repository  
public interface EventRepository extends JpaRepository<Event, Long>   
 Optional<Event> findById(Long id);  
 @Query("SELECT e FROM Event e ORDER BY e.createdAt DESC")  
 List<Event> findAll();  
 List<Event> findAllByUser\_Id(Long userId);  
}



Рисунок 1.5 – Система Поваги

# 4 РОБОТА З UI КОМПОНЕНТАМИ

Адміністративна панель включає повний набір CRUD операцій для управління користувачами та записами рейтингу. Захист адмін-панелі реалізовано через декоратор @app.before\_request:

@app.before\_request

def check\_admin\_privileges():

admin\_endpoints = [

'admin\_panel', 'admin\_add\_leaderboard', 'admin\_edit\_leaderboard',

'admin\_delete\_leaderboard', 'admin\_add\_user', 'admin\_edit\_user', 'admin\_delete\_user'

]

if request.endpoint and request.endpoint in admin\_endpoints and not session.get('is\_admin'):

flash('Access denied. Admin privileges required.', 'danger')

return redirect(url\_for('index'))

Стилізація використовує кастомні CSS змінні для підтримки кібер-панк тематики:

:root {

--pixel-font: 'Press Start 2P', monospace;

--bg-black: #000000;

--main-cyan: #00FFFF;

--dark-cyan: #008080;

--medium-grey: #444444;

--dark-grey: #222222;

}

body {

font-family: var(--pixel-font);

background-color: var(--bg-black);

color: var(--main-cyan);

font-size: 0.85em;

line-height: 1.6;

}

Таблиці мають спеціальне оформлення з неоновими кольорами та ефектами наведення:

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

border: 2px solid var(--main-cyan);

overflow: hidden;

}

th {

background-color: var(--main-cyan);

font-weight: bold;

color: var(--bg-black);

padding: 12px 10px;

font-size: 0.9em;

}

tr:hover {

background-color: var(--medium-grey);

}

Кнопки поваги мають інтерактивну зміну стану з емодзі:

.respect-button {

background: none;

border: none;

color: var(--main-cyan);

cursor: pointer;

font-size: 1.8em;

padding: 0;

transition: color 0.2s ease;

}

.respect-button.respected {

color: #00FF00;

}

Інтеграція jQuery UI для роботи з датами та модальними вікнами:

$(function() {

$("#some\_date").datepicker({

dateFormat: "yy-mm-dd"

});

});

function openModal(modalId) {

document.getElementById(modalId).style.display = "block";

}

function closeModal(modalId) {

document.getElementById(modalId).style.display = "none";

}



Рисунок 1.6 – Панель Адміністратора

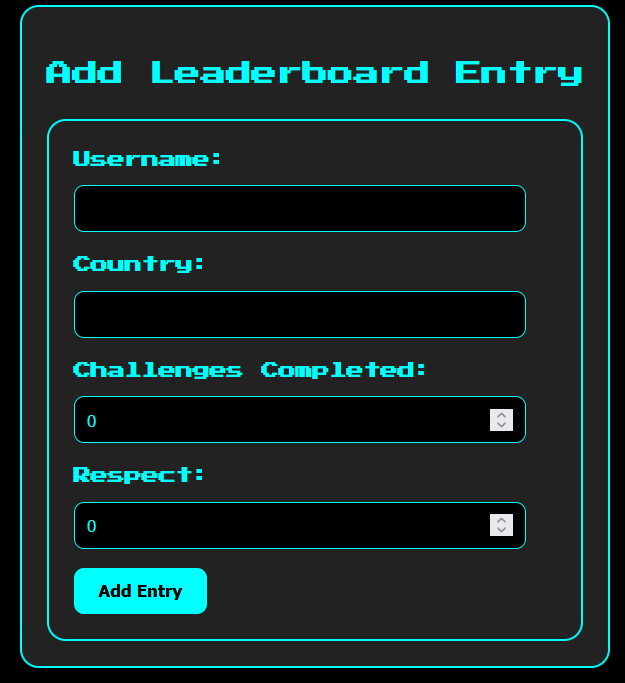


Рисунок 1.7 – Додання нової доски лідерів

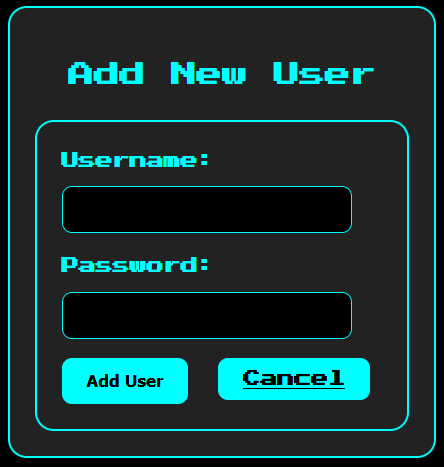


Рисунок 1.8 – Додавання нового користувача

# 5 ПРЕДСТАВЛЕННЯ РОЗРОБЛЕНОГО ПРОЕКТУ

## 5.1 Опис розробленого проєкту

Розроблено повнофункціональний веб-додаток системи рейтингу хакерів "Hacker Leaderboard" з використанням Python Flask, SQLite, HTML/CSS/JavaScript та jQuery UI. Додаток присвячений веденню рейтингу учасників хакерських змагань та CTF турнірів.

Система включає наступні сторінки: головна сторінка з рейтинговою таблицею, форми реєстрації та авторизації, сторінка редагування профілю, адміністративна панель з можливостями управління користувачами та записами рейтингу.

Головна функціональність включає систему ранжування на основі виконаних завдань (від "Script Kiddie" до "Legend"), можливість надавання поваги іншим учасникам, адміністративне управління всіма аспектами системи.

Дизайн виконано в стилі кібер-панк з використанням неонових кольорів (блакитний, зелений, червоний), пікселізованого шрифту "Press Start 2P" та тематичного оформлення елементів інтерфейсу.

База даних містить три взаємопов'язані таблиці з належними зовнішніми ключами та обмеженнями унікальності. Безпека забезпечується хешуванням паролів за допомогою SHA-256 та сесійною авторизацією.

## 5.2 Сторінки

Сторінка авторизації (рис. 1.1), сторінка реєстрації (рис. 1.2), головна сторінка (1.3), панель адміністратора (рис. 1.6).

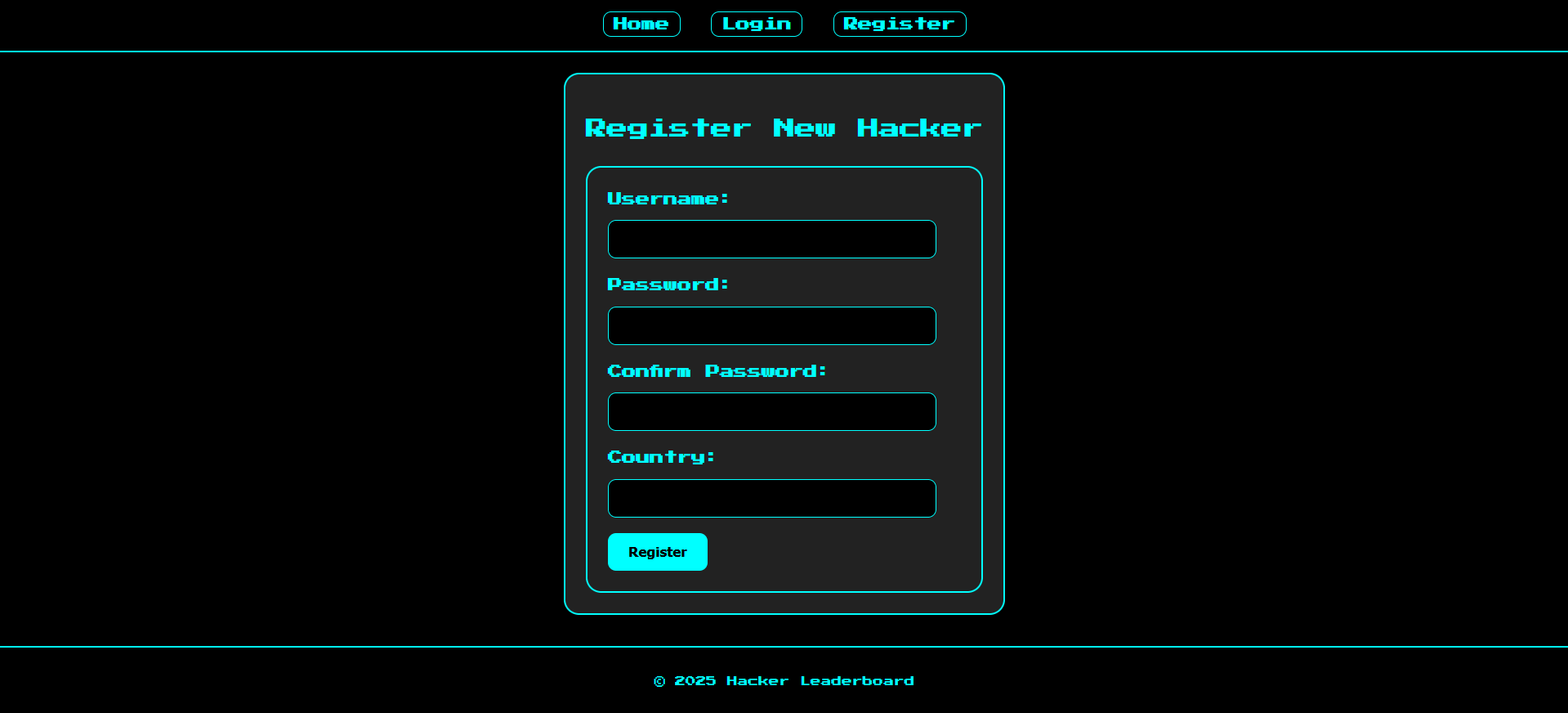


Рисунок 1.1 – Вікно реєстрації

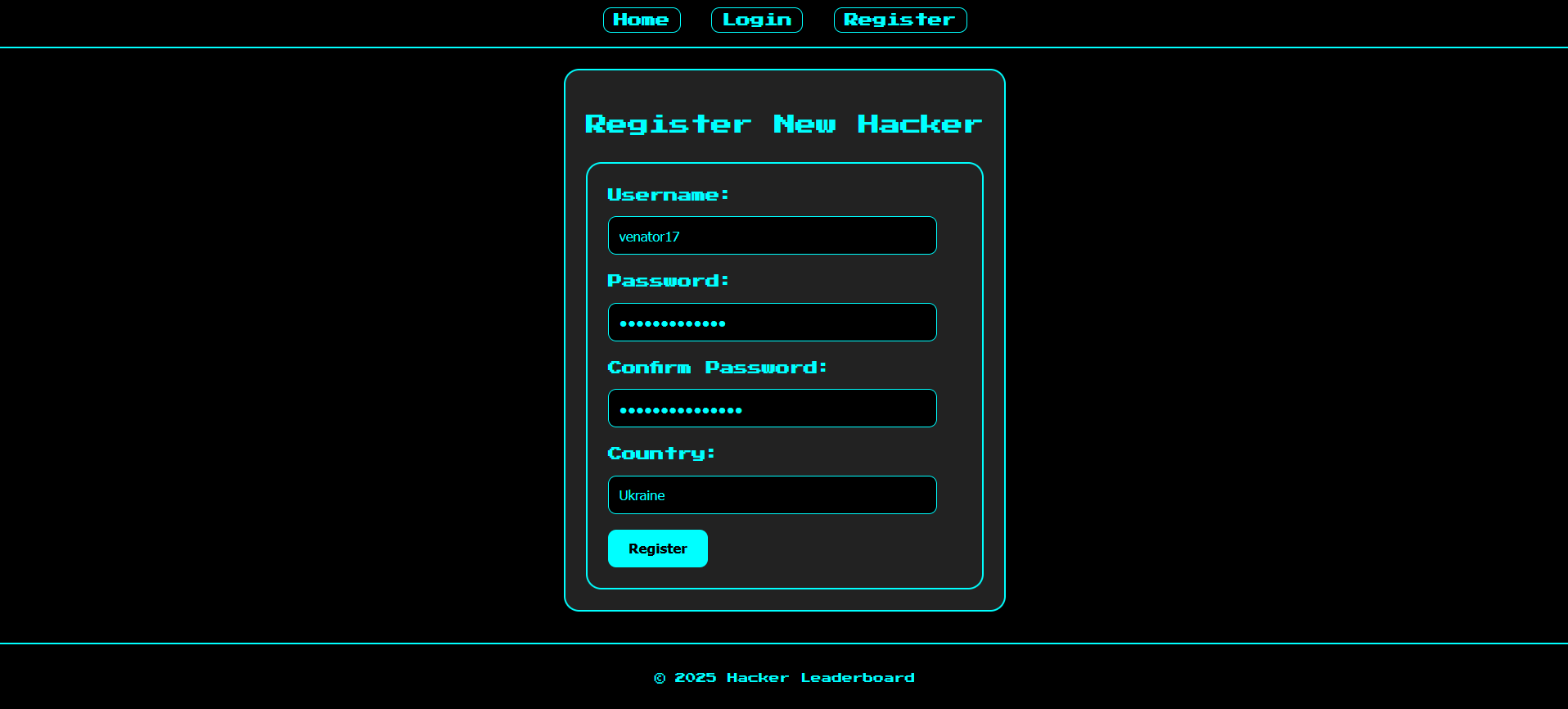


Рисунок 1.2 – Вікно авторизації



Рисунок 1.3 – Головна сторінка



Рисунок 1.6 – Панель адміністратора

**Висновки:** в результаті виконання даної роботи було розроблено повнофункціональний веб-додаток системи рейтингу хакерів з адміністративною панеллю, системою авторизації та інтерактивним користувацьким інтерфейсом у стилі кібер-панк.

Під час виконання роботи я навчився працювати з фреймворком Flask для створення веб-додатків, проектувати та реалізовувати реляційні бази даних SQLite, створювати системи авторизації з хешуванням паролів, розробляти адміністративні панелі з повним набором CRUD операцій, стилізувати веб-інтерфейси за допомогою сучасних CSS технологій та інтегрувати JavaScript бібліотеки для покращення користувацького досвіду.