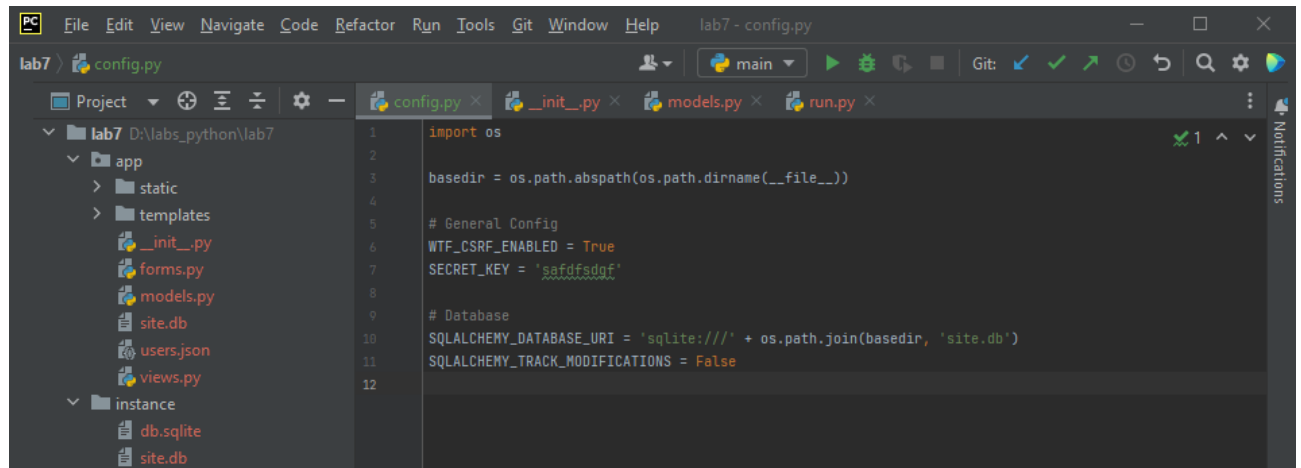


# Лабораторна 7

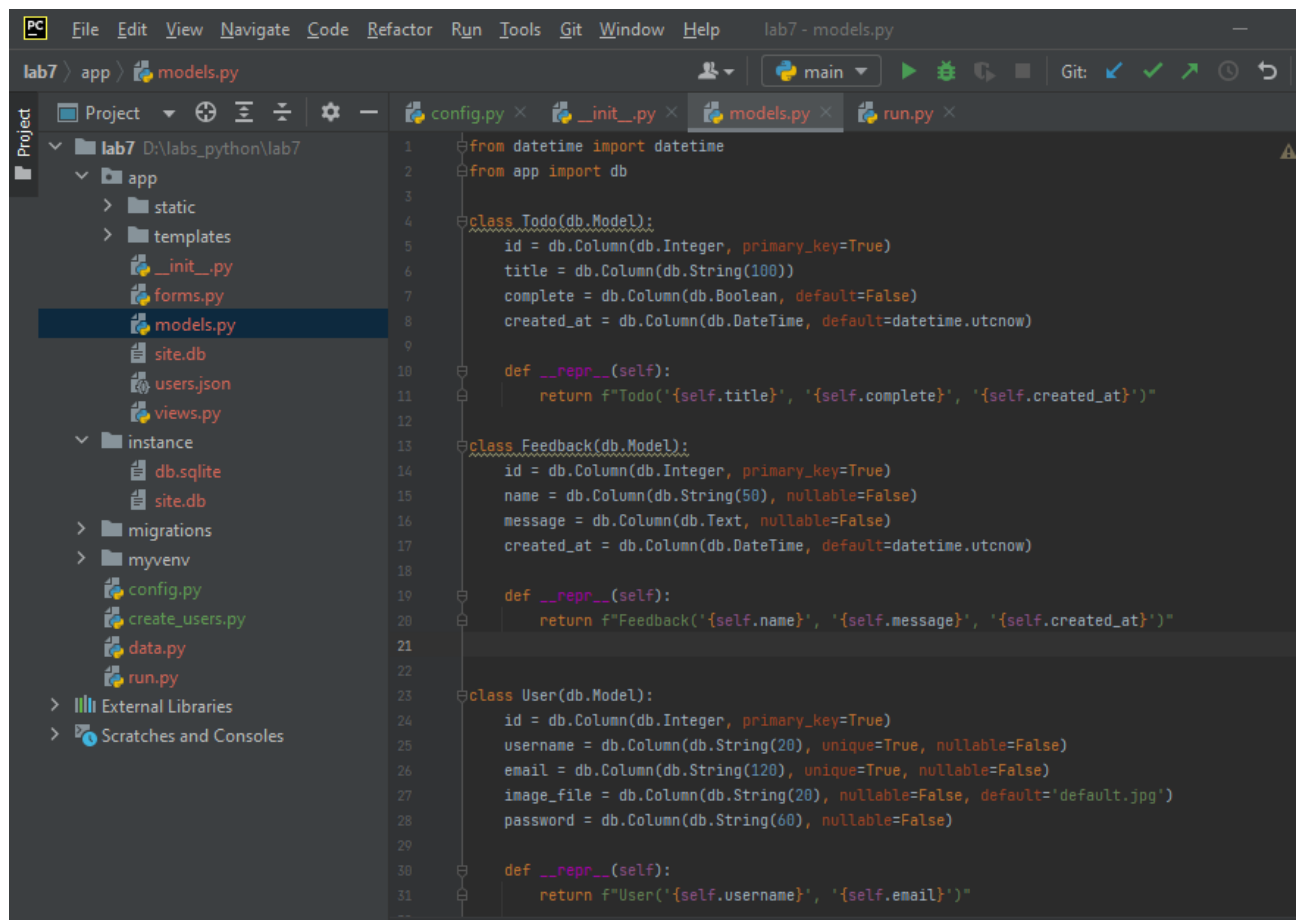
## Прунька Романа

1. Вказати у файлі config.py налаштування майбутньої бази даних (якщо нема)



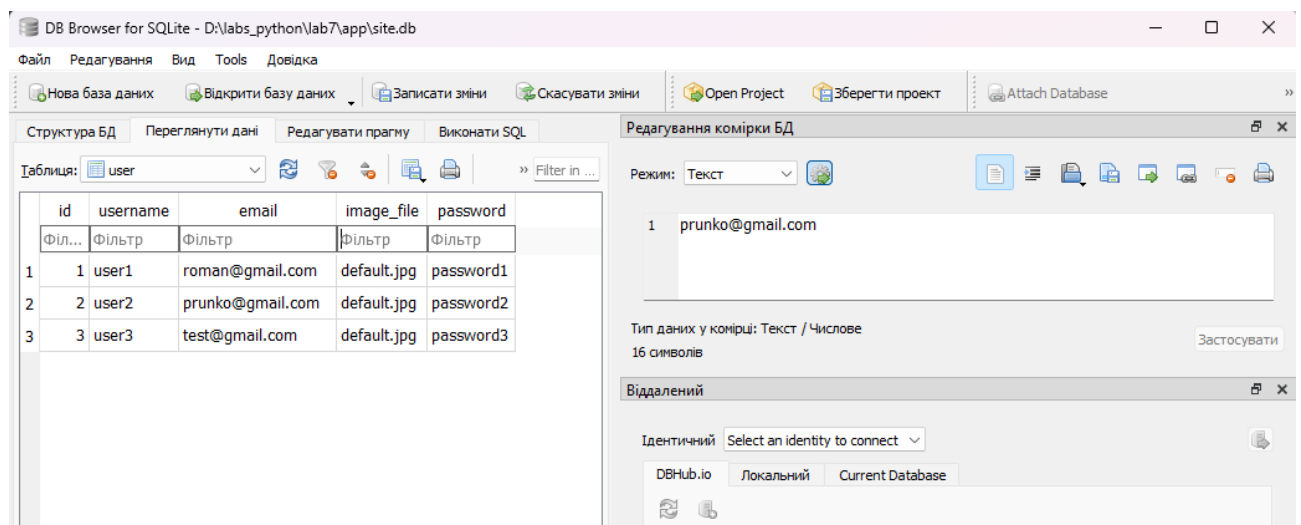
```
1 import os
2
3 basedir = os.path.abspath(os.path.dirname(__file__))
4
5 # General Config
6 WTF_CSRF_ENABLED = True
7 SECRET_KEY = 'safdfsdgdf'
8
9 # Database
10 SQLALCHEMY_DATABASE_URI = 'sqlite:/// ' + os.path.join(basedir, 'site.db')
11 SQLALCHEMY_TRACK_MODIFICATIONS = False
12
```

3. Створити модель користувача User у модулі models.py, імпортувавши відповідні залежності

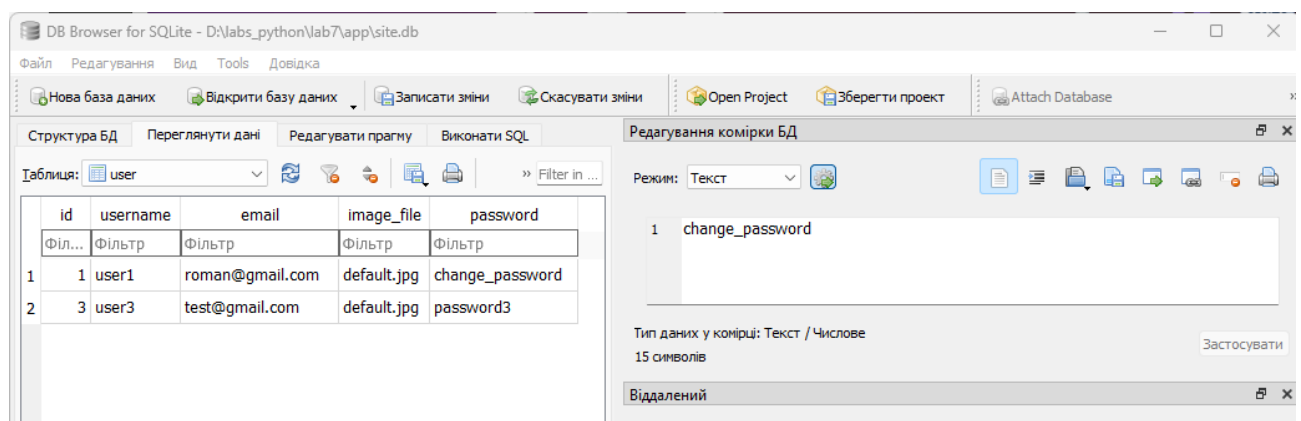


```
1 from datetime import datetime
2 from app import db
3
4 class Todo(db.Model):
5     id = db.Column(db.Integer, primary_key=True)
6     title = db.Column(db.String(100))
7     complete = db.Column(db.Boolean, default=False)
8     created_at = db.Column(db.DateTime, default=datetime.utcnow)
9
10     def __repr__(self):
11         return f'Todo('{self.title}', '{self.complete}', '{self.created_at}'))'
12
13 class Feedback(db.Model):
14     id = db.Column(db.Integer, primary_key=True)
15     name = db.Column(db.String(50), nullable=False)
16     message = db.Column(db.Text, nullable=False)
17     created_at = db.Column(db.DateTime, default=datetime.utcnow)
18
19     def __repr__(self):
20         return f'Feedback('{self.name}', '{self.message}', '{self.created_at}'))'
21
22
23 class User(db.Model):
24     id = db.Column(db.Integer, primary_key=True)
25     username = db.Column(db.String(20), unique=True, nullable=False)
26     email = db.Column(db.String(120), unique=True, nullable=False)
27     image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
28     password = db.Column(db.String(60), nullable=False)
29
30     def __repr__(self):
31         return f'User('{self.username}', '{self.email}'))'
```

4. Виконати міграцію БД. За допомогою командного рядка CLI або flask shell переконались, що в таблиці «user» нема даних (використати, наприклад програму SQLite Database Browser або аналог). Створити трьох користувачів і записати їх у БД, перевіривши це у SQLite Database Browser. Далі видалити другого користувача, а у першого змінити пароль. Результати виконання закріпити.

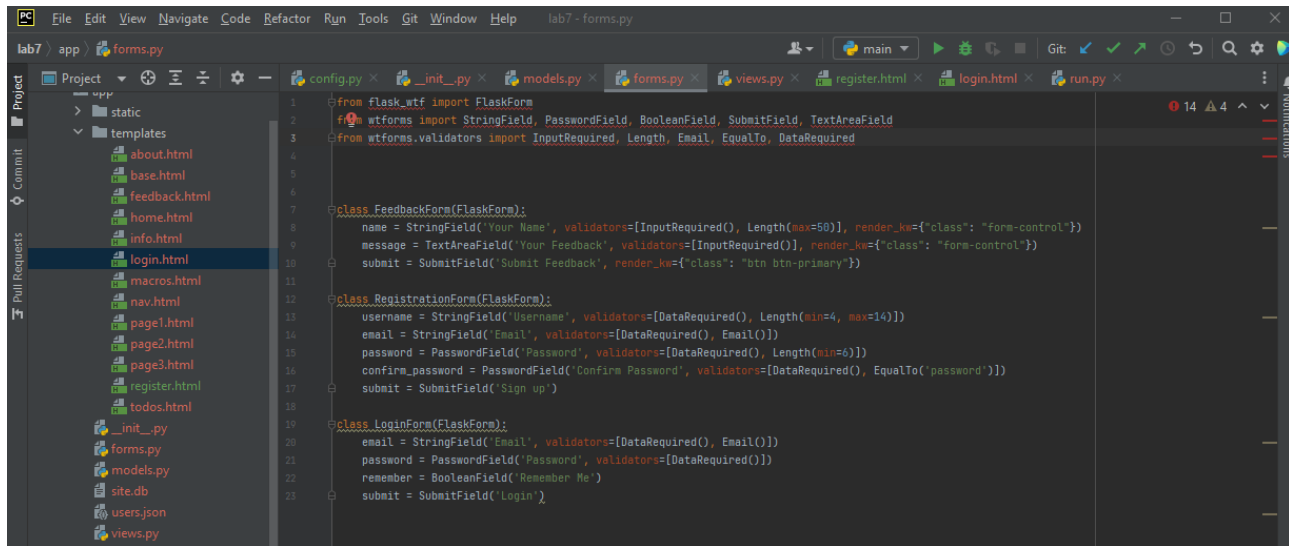


Видалив другого користувача та змінив пароль у першого.



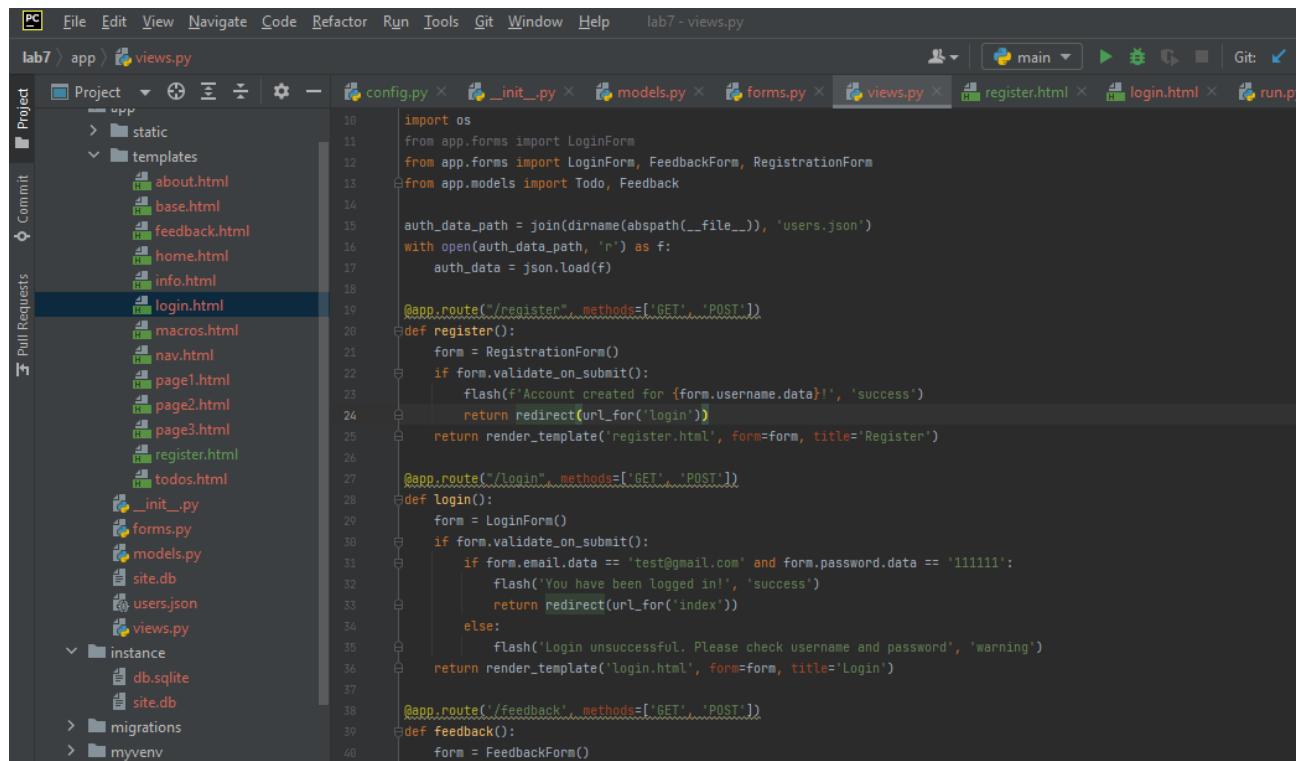
## 5. Створити класи для форм реєстрації і входу, імпортувавши потрібні поля і валідатори.

forms.py:



```
1 from flask_wtf import FlaskForm
2 from wtforms import StringField, PasswordField, BooleanField, SubmitField, TextAreaField
3 from wtforms.validators import InputRequired, Length, Email, EqualTo, DataRequired
4
5
6
7
8 class FeedbackForm(FlaskForm):
9     name = StringField('Your Name', validators=[InputRequired(), Length(max=50)], render_kw={'class': 'form-control'})
10    message = TextAreaField('Your Feedback', validators=[InputRequired()], render_kw={'class': 'form-control'})
11    submit = SubmitField('Submit Feedback', render_kw={'class': 'btn btn-primary'})
12
13 class RegistrationForm(FlaskForm):
14     username = StringField('Username', validators=[DataRequired(), Length(min=4, max=14)])
15     email = StringField('Email', validators=[DataRequired(), Email()])
16     password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])
17     confirm_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])
18     submit = SubmitField('Sign up')
19
20 class LoginForm(FlaskForm):
21     email = StringField('Email', validators=[DataRequired(), Email()])
22     password = PasswordField('Password', validators=[DataRequired()])
23     remember = BooleanField('Remember Me')
24     submit = SubmitField('Login')
```

## 6. Створимо маршрути і відповідні функції представлення register та login для початкового налаштування у файлі views.py



```
10 import os
11 from app.forms import LoginForm
12 from app.forms import LoginForm, FeedbackForm, RegistrationForm
13 from app.models import Todo, Feedback
14
15 auth_data_path = join(dirname(abspath(__file__)), 'users.json')
16 with open(auth_data_path, 'r') as f:
17     auth_data = json.load(f)
18
19 @app.route("/register", methods=['GET', 'POST'])
20 def register():
21     form = RegistrationForm()
22     if form.validate_on_submit():
23         flash(f'Account created for {form.username.data}!', 'success')
24         return redirect(url_for('login'))
25     return render_template('register.html', form=form, title='Register')
26
27 @app.route("/login", methods=['GET', 'POST'])
28 def login():
29     form = LoginForm()
30     if form.validate_on_submit():
31         if form.email.data == 'test@gmail.com' and form.password.data == '111111':
32             flash('You have been logged in!', 'success')
33             return redirect(url_for('index'))
34         else:
35             flash('Login unsuccessful. Please check username and password', 'warning')
36     return render_template('login.html', form=form, title='Login')
37
38 @app.route("/feedback", methods=['GET', 'POST'])
39 def feedback():
40     form = FeedbackForm()
```

7. Створити html-шаблони register.html та login.html для форм реєстрації та входу (використати базовий шаблон та розширити його, обов'язково застосувати стилі Bootstrap або власні, щоб форми гарно виглядали), додати у меню сайту відповідні пункти.

The image displays two screenshots of a web application interface. Both screenshots show a purple navigation bar at the top with the following menu items: PRUNKO, ДОМАШНЯ, ПРО МЕНЕ, РЕЄСТРАЦІЯ, ЛОГІН, TODO, FEEDBACK. The browser's address bar shows the URL 127.0.0.1:5000/register for the first screenshot and 127.0.0.1:5000/login for the second.

The first screenshot shows the 'Register' form. It features a green heading 'Вітання!' (Greetings!) above the form title 'Register'. The form contains four input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. A red 'Sign up' button is positioned below the 'Confirm Password' field. At the bottom right, there is a copyright notice: '© Roman Prunko Ivano-Frankivsk, 2023'.

The second screenshot shows the 'Login' form. It also features a green heading 'Вітання!' (Greetings!) above the form title 'Login'. The form contains two input fields: 'Email' and 'Password'. A 'Remember Me' checkbox is located below the 'Password' field. A red 'Login' button is positioned below the 'Remember Me' checkbox. At the bottom right, there is a copyright notice: '© Roman Prunko Ivano-Frankivsk, 2023'.

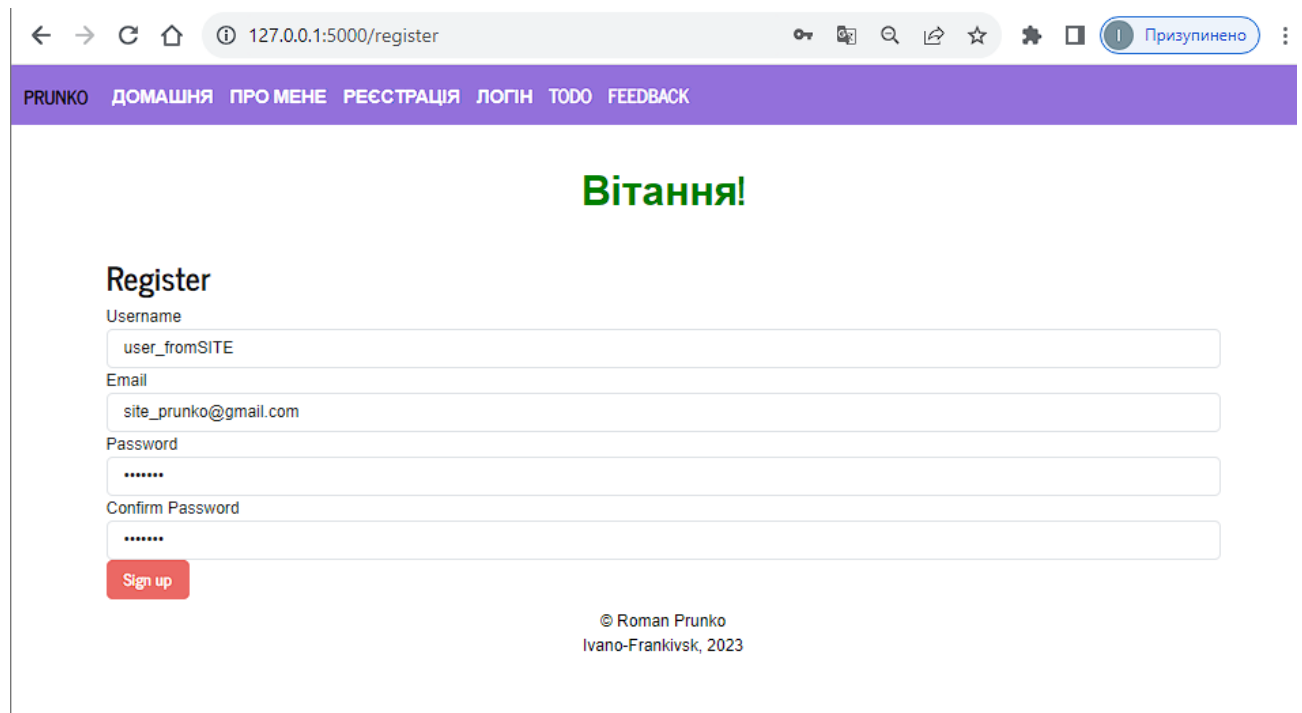
9. Модифікувати функції представлення на можливість додавання нових користувачів у базу даних та імплементувати можливість авторизації зареєстрованих користувачів з бази даних за поштою і паролем (без хешування). Використати flash-повідомлення про успішні чи неуспішні дії.

Якщо непроходить якась із валідацій, наприклад, ім'я користувача чи пошта можуть бути недоступні

при реєстрації або пароль може бути неправильним при вході, то в цьому випадку робиться

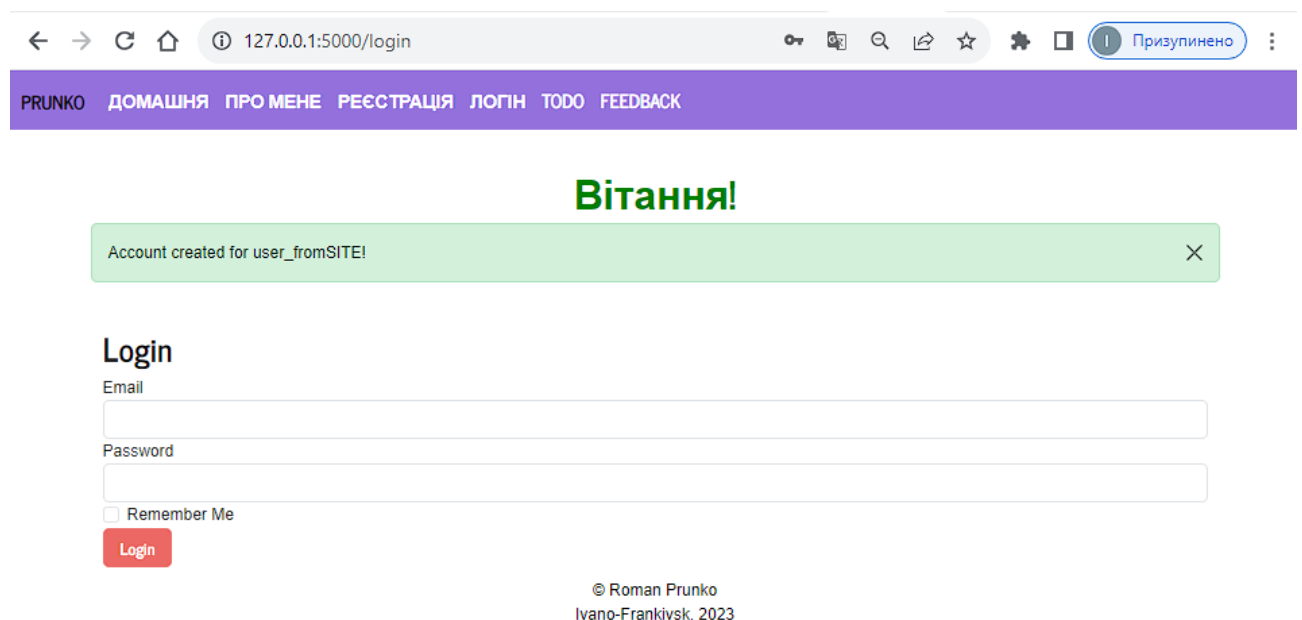
перенаправлення відповідно на register або login.

## Створюємо аккаунт



The screenshot shows a web browser at the address 127.0.0.1:5000/register. The page has a purple navigation bar with links: PRUNKO, ДОМАШНЯ, ПРО МЕНЕ, РЕЄСТРАЦІЯ, ЛОГІН, TODO, and FEEDBACK. The main content area features a green heading "Вітання!". Below it is a "Register" form with fields for Username (filled with "user\_fromSITE"), Email (filled with "site\_prunko@gmail.com"), Password (filled with "\*\*\*\*\*"), and Confirm Password (filled with "\*\*\*\*\*"). A red "Sign up" button is at the bottom of the form. The footer contains the text "© Roman Prunko Ivano-Frankivsk, 2023".

## Аккаунт створився



The screenshot shows the same website at the address 127.0.0.1:5000/login. A green success message box at the top states "Account created for user\_fromSITE!". Below this is a "Login" form with fields for Email and Password. There is a "Remember Me" checkbox and a red "Login" button. The footer text "© Roman Prunko Ivano-Frankivsk, 2023" is visible at the bottom.

В базу все прийшло:

DB Browser for SQLite - D:\labs\_python\lab7\app\site.db

Файл Редагування Вид Tools Довідка

Нова база даних Відкрити базу даних Записати зміни Скасувати зміни Open Project Зберегти проект Attach Database

Структура БД Переглянути дані Редагувати прагму Виконати SQL

Таблиця: user

	id	username	email	image_file	password
Філ...	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр
1	1	user1	roman@gmail.com	default.jpg	change_password
2	3	user3	test@gmail.com	default.jpg	password3
3	4	user_fromSITE	site_prunko@gmail....	default.jpg	test123

Редагування копії БД

Режим: Текст

1

Тип даних у копії: Текст / Числове  
1 символ

Віддалений

Ідентичний Select an identity to connect

Входимо в аккаунт, після успішного входу має перевести нас, на сторінку: «ДОМАШНЯ/ГОЛОВНА»

← → ↻ 🏠 ⓘ 127.0.0.1:5000/login 🔑 🔍 🔗 ☆ ⚙️ 📱 1 Призупинено ⋮

PRUNKO ДОМАШНЯ ПРО МЕНЕ РЕЄСТРАЦІЯ ЛОГІН TODO FEEDBACK

**Вітання!**

Account created for user\_fromSITE! ✕

**Login**

Email

site\_prunko@gmail.com

Password

.....

☐ Remember Me

Login



© Roman Prunko  
Ivano-Frankivsk, 2023

← → ↻ 🏠 ⓘ 127.0.0.1:5000/home 🔑 🔍 🔗 ☆ ⚙️ 📱 1 Призупинено ⋮

PRUNKO ДОМАШНЯ ПРО МЕНЕ РЕЄСТРАЦІЯ ЛОГІН TODO FEEDBACK

**Вітання!**

You have been logged in! ✕

**Flask**  
web development,  
one drop at a time

Flask — мікрофреймворк для вебдодатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Поширюється відповідно до умов ліцензії BSD.

© Roman Prunko  
Ivano-Frankivsk, 2023

10. Модифікуйте код так, щоб у БД замість паролів зберігались відповідні їх хеші, а при вході на сайт перевірка на рівність паролів проводилась як рівність по хешу.

Використати на вибір один із модулів: Flask-Bcrypt, werkzeug.security, passlib

Я використав модуль **Flask-Bcrypt**:

models.py:

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    password = db.Column(db.String(60), nullable=False)

    def set_password(self, password):
        self.password = bcrypt.generate_password_hash(password).decode('utf-8')

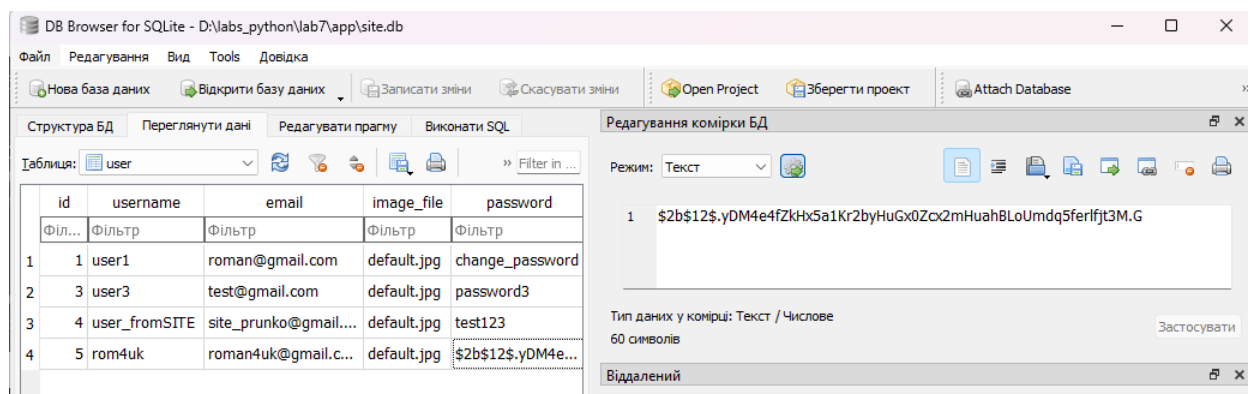
    def check_password(self, password):
        return bcrypt.check_password_hash(self.password, password)

    def __repr__(self):
        return f'User('{self.username}', '{self.email}')
```

views.py

```
@app.route("/register", methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data, password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created!', 'success')
        return redirect(url_for('login'))
    return render_template("register.html", title='Register', form=form)
```

Створив новий аккаунт:



The screenshot shows the DB Browser for SQLite interface. The main window displays a table named 'user' with the following data:

	id	username	email	image_file	password
1	1	user1	roman@gmail.com	default.jpg	change_password
2	3	user3	test@gmail.com	default.jpg	password3
3	4	user_fromSITE	site_prunko@gmail....	default.jpg	test123
4	5	rom4uk	roman4uk@gmail.c...	default.jpg	\$2b\$12\$.yDM4e...

The right-hand pane shows the raw text of the password hash: \$2b\$12\$.yDM4e4fZkHx5a1Kr2byHuGx0Zcx2mHuahBL0Umdq5ferfjt3M.G

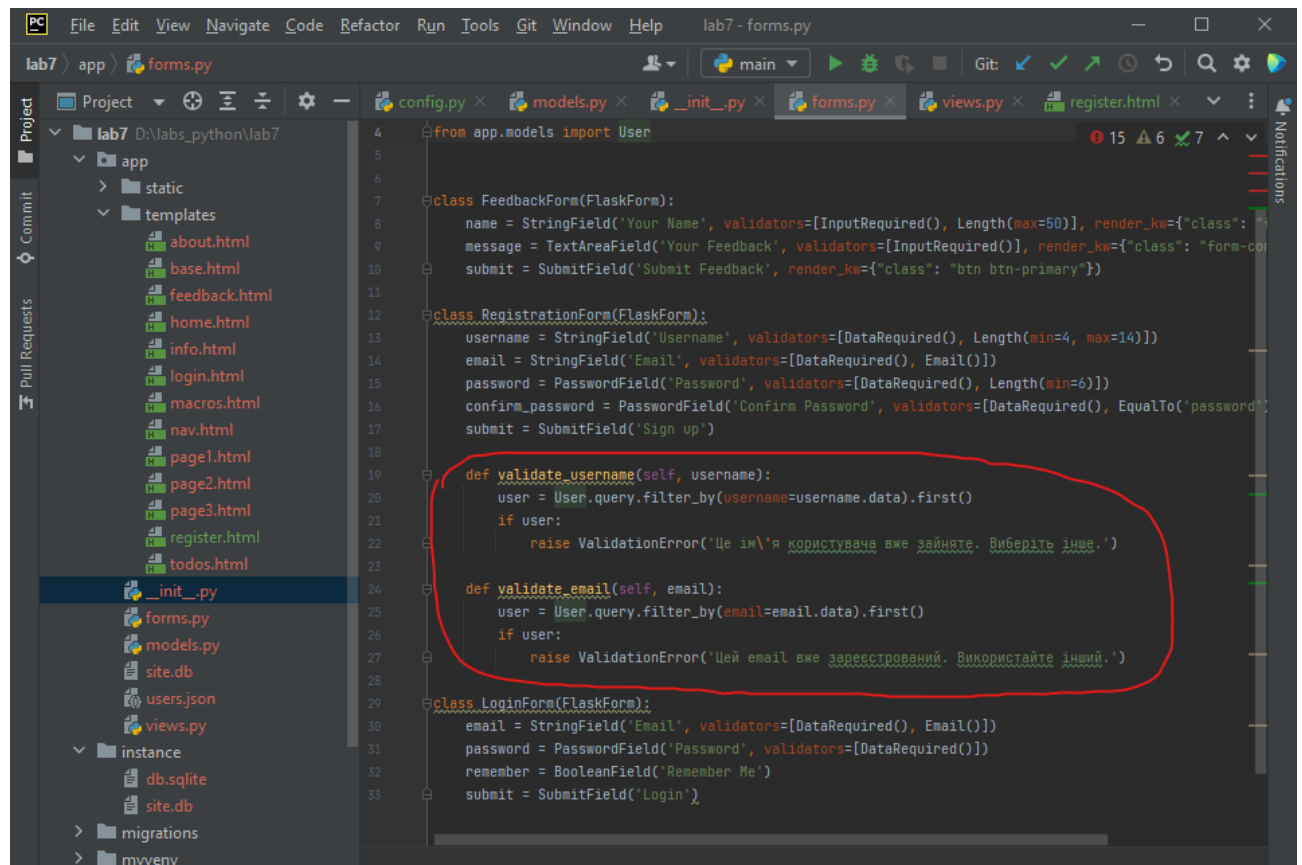
11. Додати до класу форми RegistrationForm два методи: validate\_username() та validate\_email().

WTForms прийме їх як користувацькі спеціальні валідатори та доповнює стандартні валідатори. У

цьому випадку спеціальні валідатори електронної пошти та імені користувача гарантують, що вказані

значення не є дублікатами у базі даних. Вони вказують на помилку перевірки, викликаючи виняток

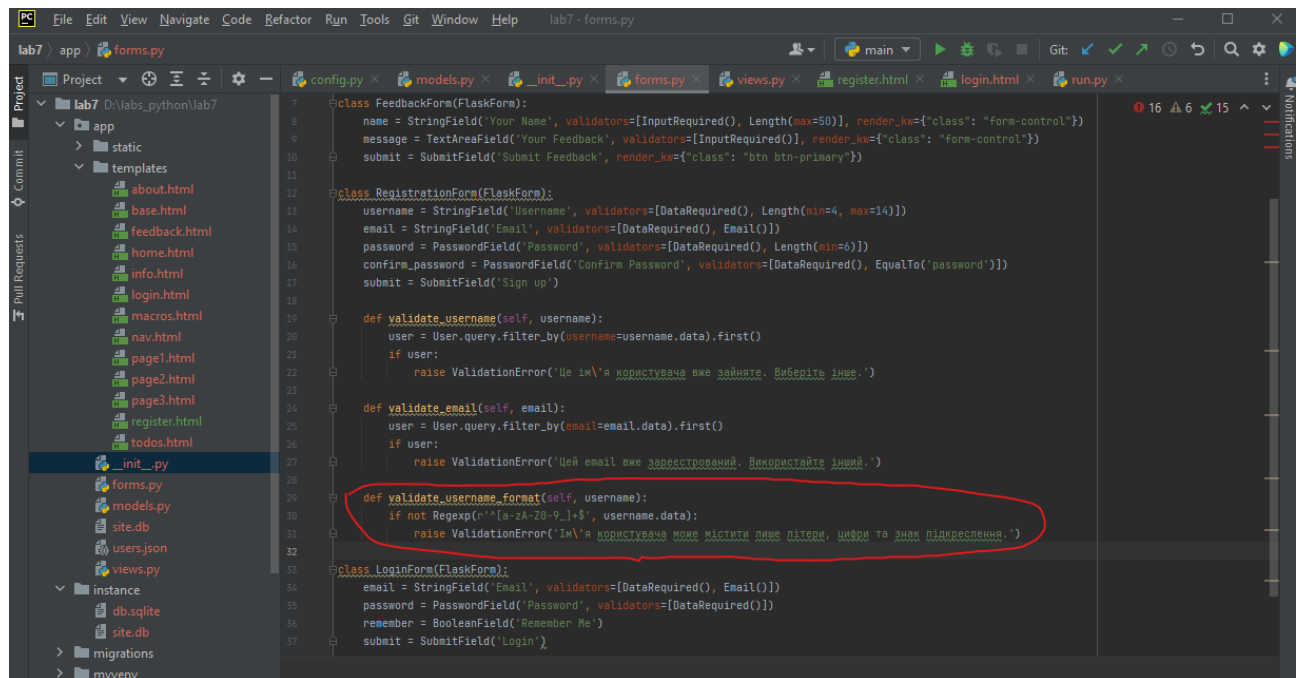
ValidationError з текстом повідомлення про помилку як аргументом.



```
4 from app.models import User
5
6
7 class FeedbackForm(FlaskForm):
8     name = StringField('Your Name', validators=[InputRequired(), Length(max=50)], render_kw={'class': 'form-control'})
9     message = TextAreaField('Your Feedback', validators=[InputRequired()], render_kw={'class': 'form-control'})
10    submit = SubmitField('Submit Feedback', render_kw={'class': 'btn btn-primary'})
11
12 class RegistrationForm(FlaskForm):
13     username = StringField('Username', validators=[DataRequired(), Length(min=4, max=14)])
14     email = StringField('Email', validators=[DataRequired(), Email()])
15     password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])
16     confirm_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])
17     submit = SubmitField('Sign up')
18
19     def validate_username(self, username):
20         user = User.query.filter_by(username=username.data).first()
21         if user:
22             raise ValidationError('Це ім'я користувача вже зайняте. Виберіть інше.')
23
24     def validate_email(self, email):
25         user = User.query.filter_by(email=email.data).first()
26         if user:
27             raise ValidationError('Цей email вже зареєстрований. Використайте інший.')
28
29 class LoginForm(FlaskForm):
30     email = StringField('Email', validators=[DataRequired(), Email()])
31     password = PasswordField('Password', validators=[DataRequired()])
32     remember = BooleanField('Remember Me')
33     submit = SubmitField('Login')
```

12. Встановіть додатковий валідатор для username за допомогою Regexp





```
class FeedbackForm(FlaskForm):
    name = StringField('Your Name', validators=[InputRequired(), Length(max=50)], render_kw={'class': 'form-control'})
    message = TextAreaField('Your Feedback', validators=[InputRequired()], render_kw={'class': 'form-control'})
    submit = SubmitField('Submit Feedback', render_kw={'class': 'btn btn-primary'})

class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=4, max=14)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Sign up')

    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('Цей ім'я користувача вже зайняте. Використайте інше.')

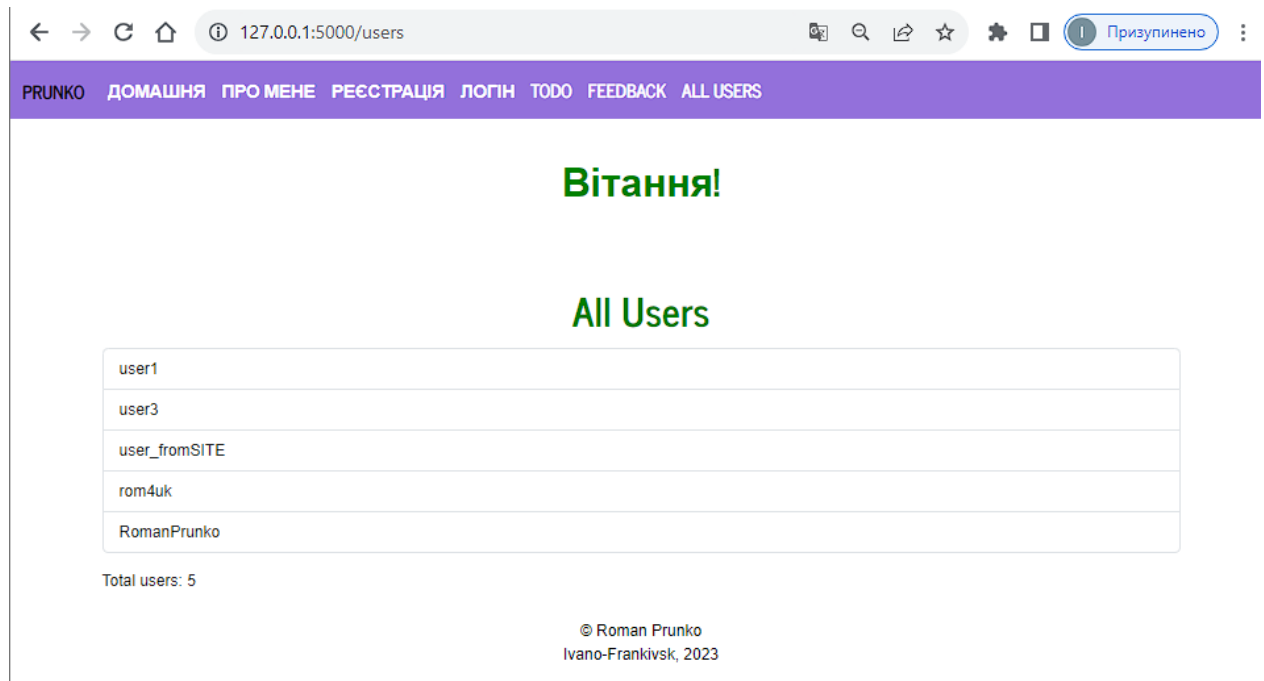
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('Цей email вже зареєстрований. Використайте інший.')

    def validate_username_format(self, username):
        if not Regexp(r'^[a-zA-Z0-9_]+$', username.data):
            raise ValidationError('Ім'я користувача може містити лише літери, цифри та знак підкреслення.')

class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')
```

14. Вивести на окремій сторінці всіх зареєстрованих користувачів та їх загальну кількість на сайті. Якщо нема зареєстрованих користувачів, то вивести відповідне повідомлення.

також добавив в меню – **ALL USERS**



Дані з БД:

