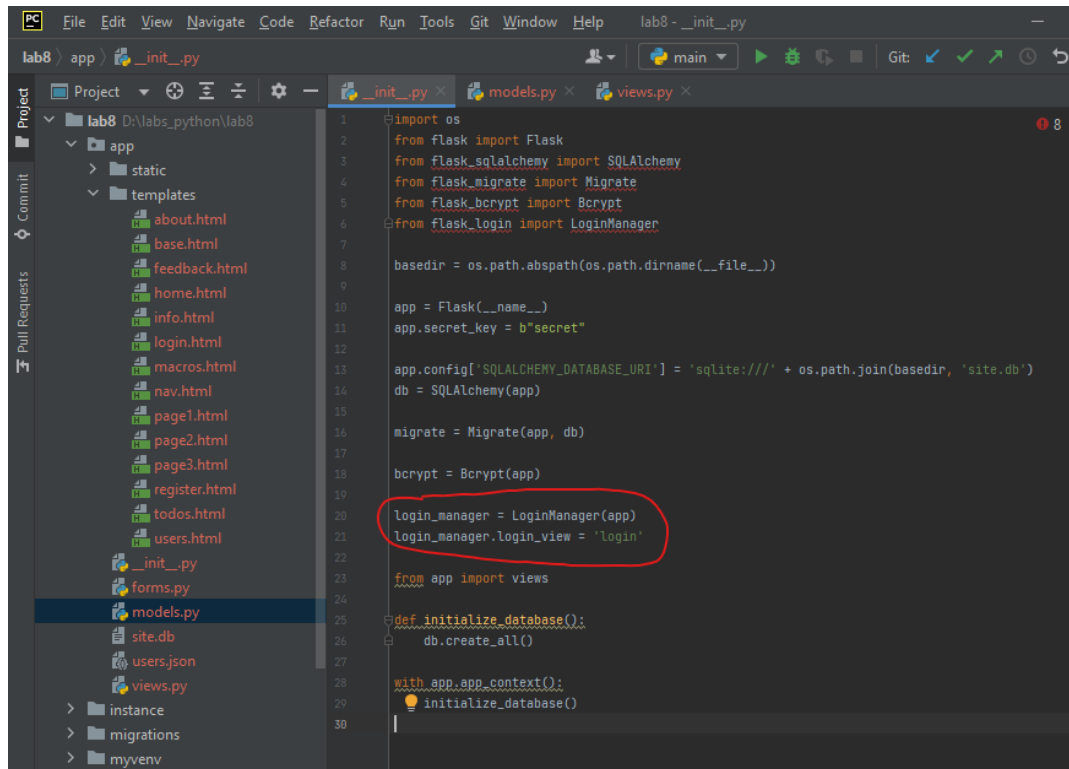


Лабораторна робота 8

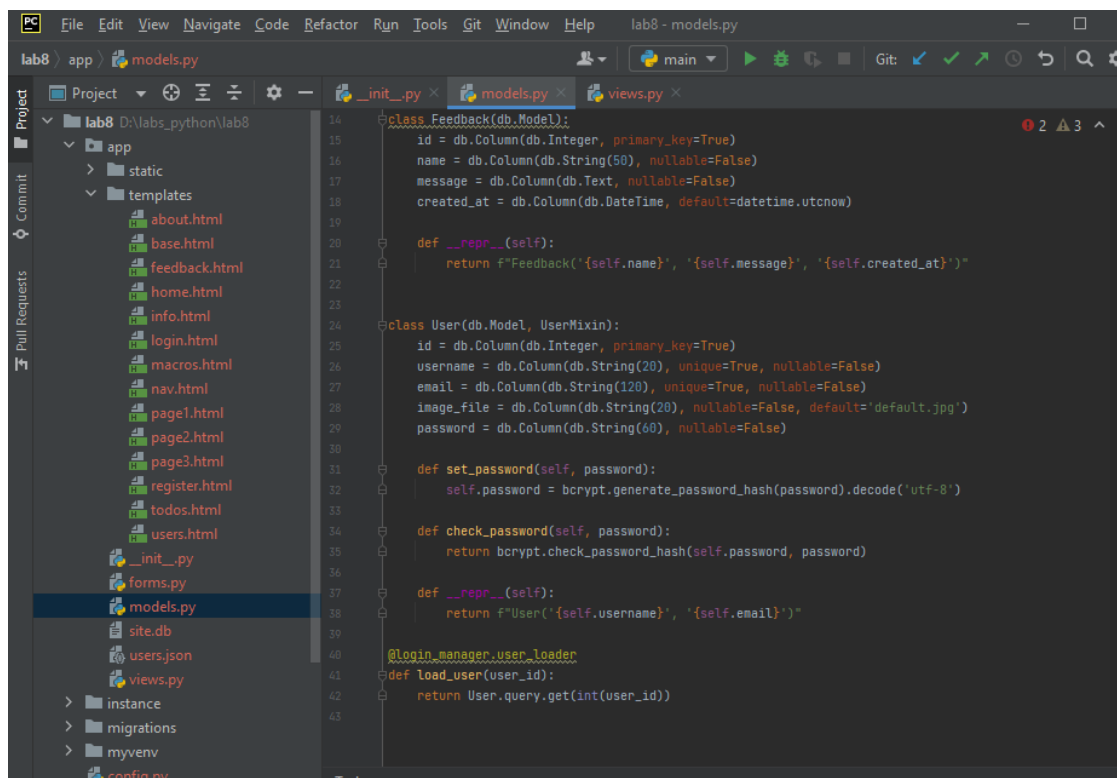
Прунька Романа

1. Ініціалізувати login_manager в __init__.py:



```
1 import os
2 from flask import Flask
3 from flask_sqlalchemy import SQLAlchemy
4 from flask_migrate import Migrate
5 from flask_bcrypt import Bcrypt
6 from flask_login import LoginManager
7
8 basedir = os.path.abspath(os.path.dirname(__file__))
9
10 app = Flask(__name__)
11 app.secret_key = b"secret"
12
13 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:/// ' + os.path.join(basedir, 'site.db')
14 db = SQLAlchemy(app)
15
16 migrate = Migrate(app, db)
17
18 bcrypt = Bcrypt(app)
19
20 login_manager = LoginManager(app)
21 login_manager.login_view = 'login'
22
23 from app import views
24
25 def initialize_database():
26     db.create_all()
27
28 with app.app_context():
29     initialize_database()
30
```

2. Імплементувати у модель User наступні чотири методи, Ці властивості та методи можуть бути реалізовані безпосередньо в класі моделі, але є і простіша альтернатива - Flask-Login надає клас UserMixin, який має реалізації за замовчуванням, які підходять для більшості випадків:



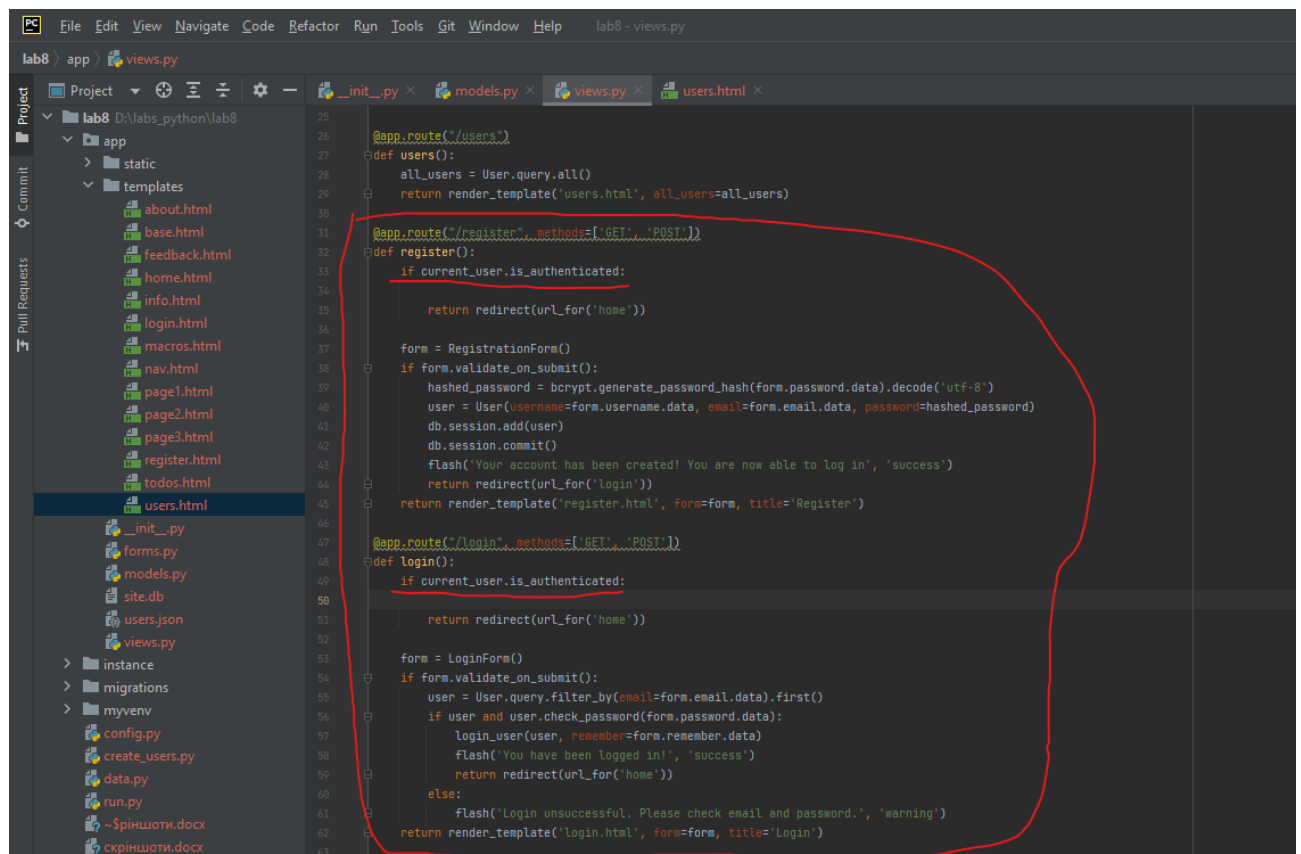
```
14 class Feedback(db.Model):
15     id = db.Column(db.Integer, primary_key=True)
16     name = db.Column(db.String(50), nullable=False)
17     message = db.Column(db.Text, nullable=False)
18     created_at = db.Column(db.DateTime, default=datetime.utcnow)
19
20     def __repr__(self):
21         return f"Feedback('{self.name}', '{self.message}', '{self.created_at}')"
22
23 class User(db.Model, UserMixin):
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 set_password(self, password):
31         self.password = bcrypt.generate_password_hash(password).decode('utf-8')
32
33     def check_password(self, password):
34         return bcrypt.check_password_hash(self.password, password)
35
36     def __repr__(self):
37         return f"User('{self.username}', '{self.email}')"
38
39 @login_manager.user_loader
40 def load_user(user_id):
41     return User.query.get(int(user_id))
42
43
```

3. Реєстрація сеансу функцією login_user () у views.py

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and user.check_password(form.password.data):
            login_user(user, remember=form.remember.data)
            flash('You have been logged in!', 'success')
            return redirect(url_for('home'))
        else:
            flash('Login unsuccessful. Please check email and password.', 'warning')
    return render_template("login.html", form=form, title='Login')
```

4. Об'єкт поточного користувача current_user

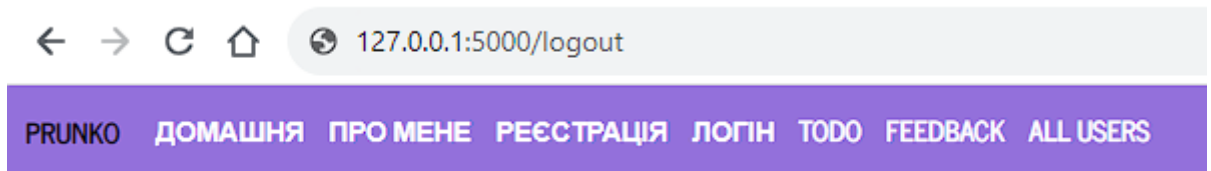
Зробіть імпорт у views.py та додайте частини коду для перевірки чи є поточний користувач аутентифікованим:



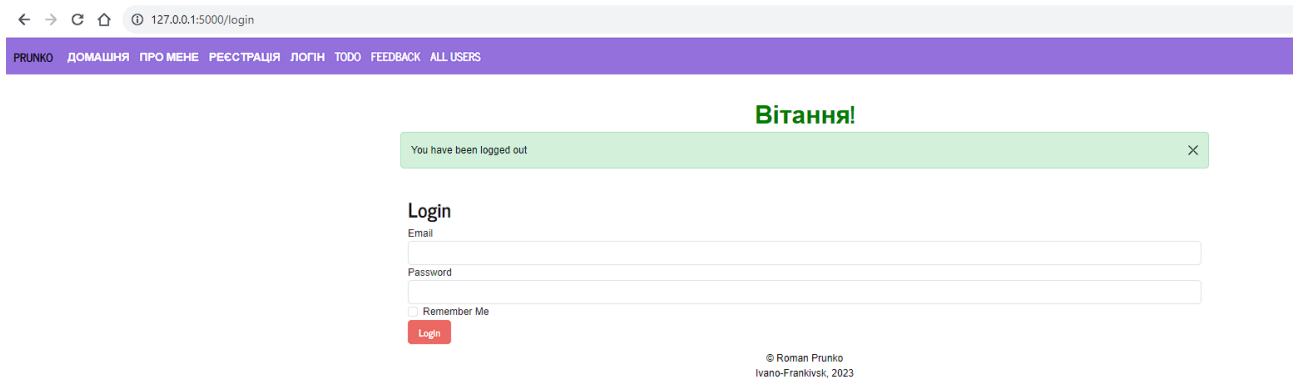
5. Реалізуйте роут для виходу з сеансу користувача за допомогою функції logout_user

```
@app.route("/logout")
def logout():
    logout_user()
    flash('You have been logged out', 'success')
    return redirect(url_for('login'))
```

Вводжу logout

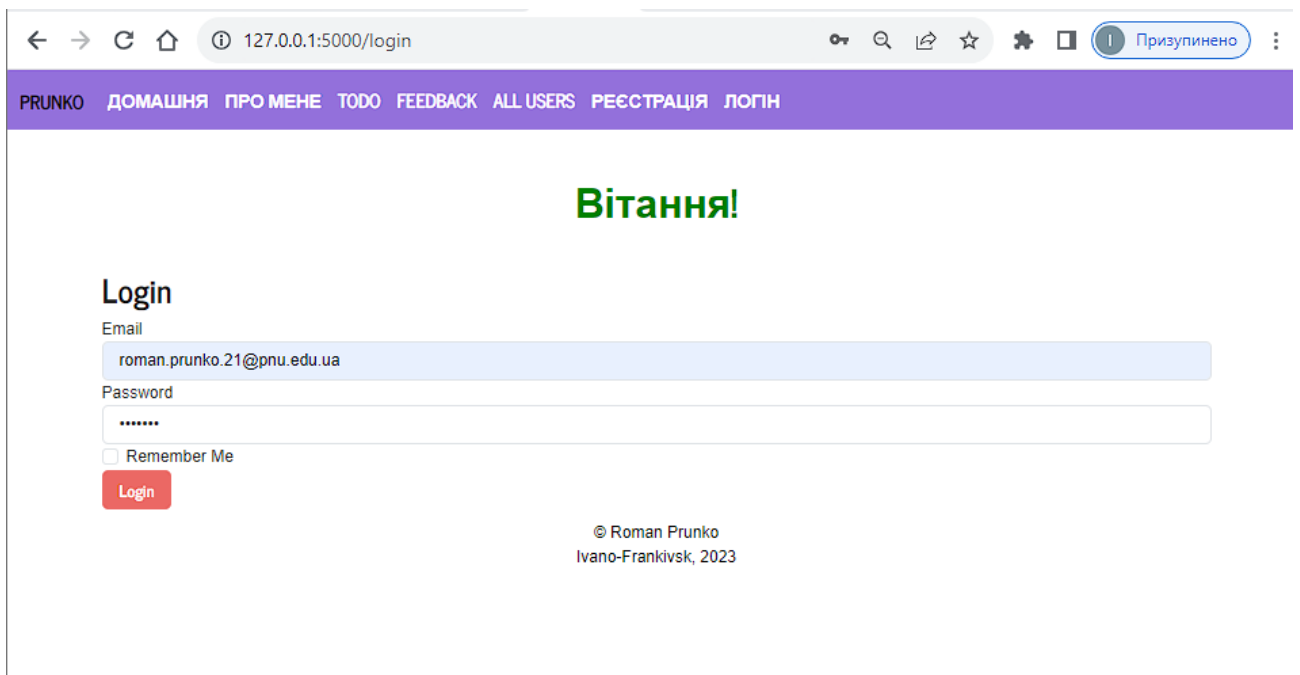


і можу даліше реєструватись, логінитись



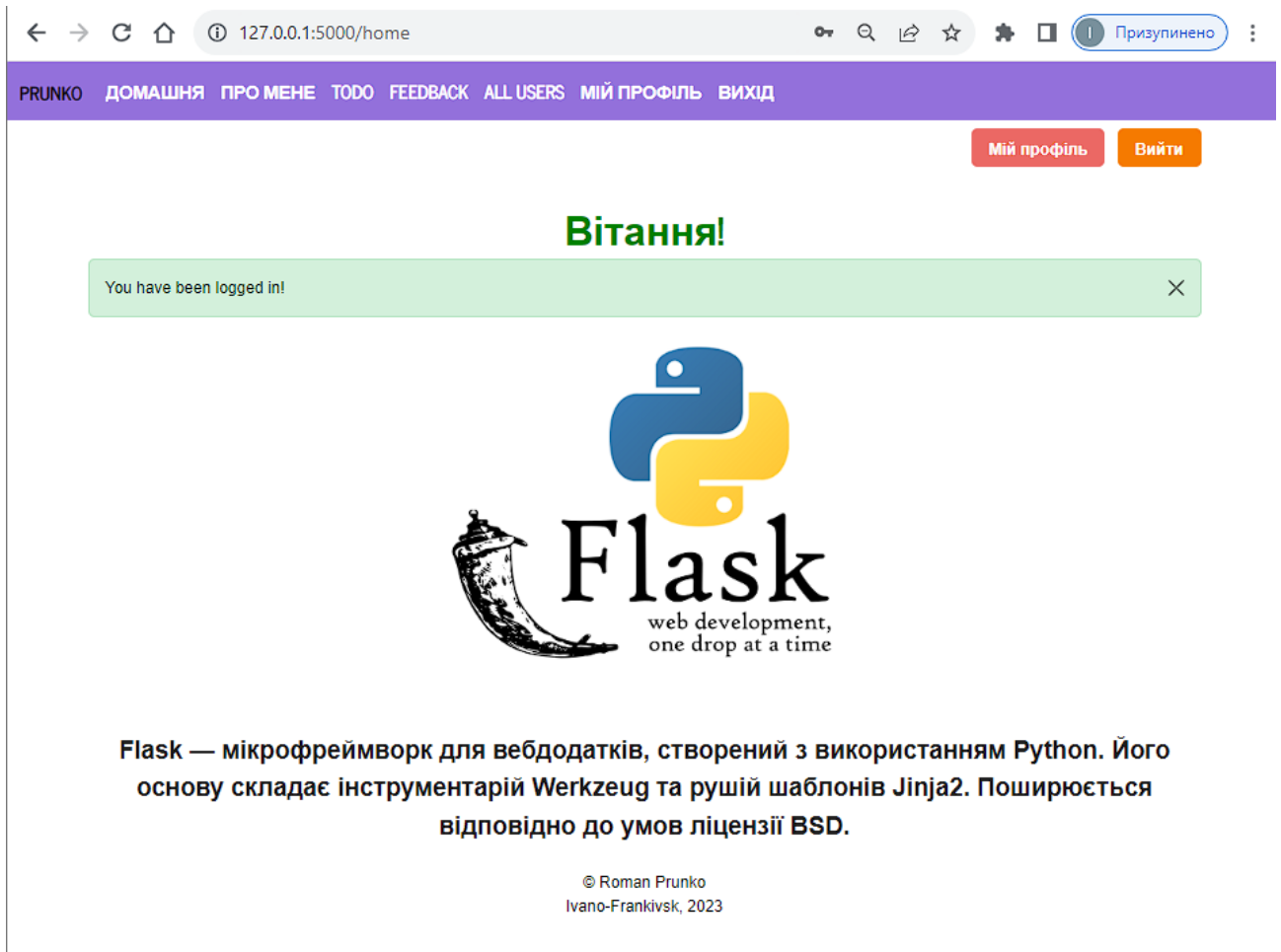
6. Реалізуйте наступну функціональність: при успішному вході користувачу на сайті не показується пункти меню реєстрації і входу, натомість з'являються два меню – Мій профіль та Вихід

Логінимось:

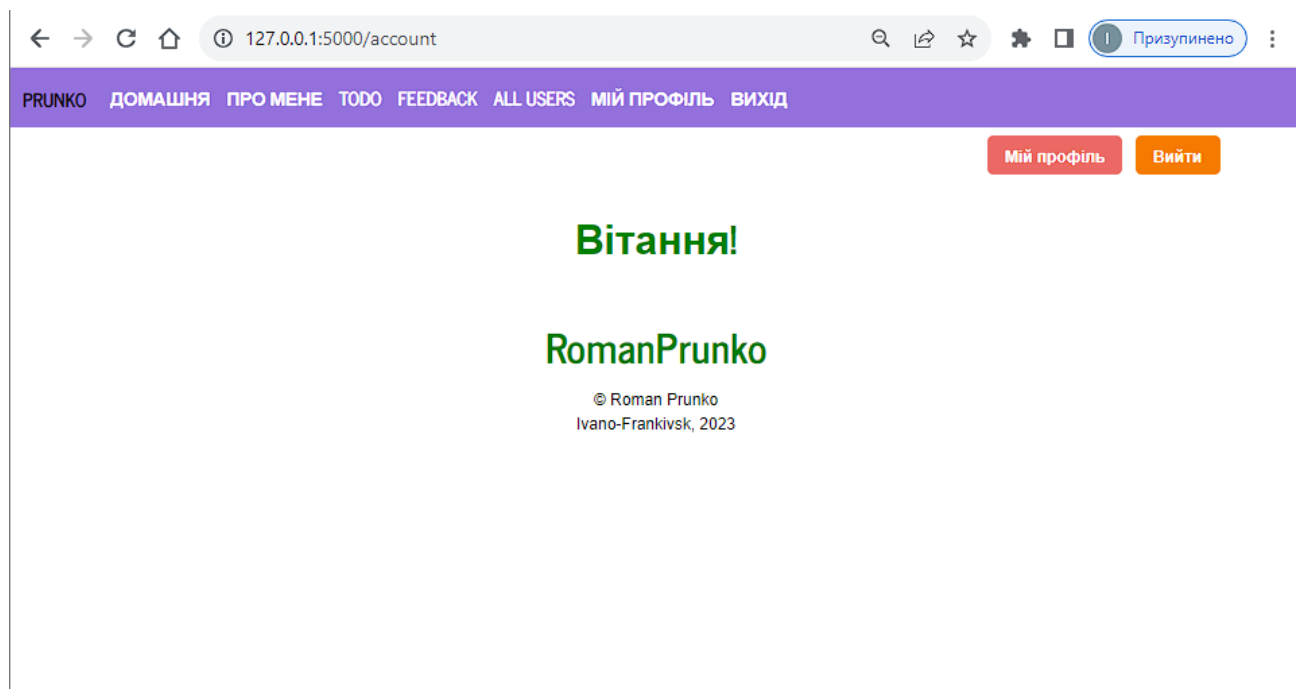


В меню(nav) – зникає реєстрація та логін, з'являється – **мій профіль та вихід**

Нас перекидає на home – де появляється – **мій профіль та вийти**.



7. Додайте маршрут “/account” і відповідну функцію представлення, а також шаблон account.html наступного вигляду



8. Декоратор @login_required

```
@app.route("/account")
@login_required
def account():
    return render_template('account.html', title='Account')
```

9. Атрибут login_view об'єкта LoginManager встановлює end-point для сторінки входу. Flask-Login перенаправить на сторінку входу, коли анонімний користувач намагається отримати доступ до захищеної сторінки.

```
__init__.py x  models.py x  views.py x  account.html x  base.html x  nav.html x
1  import os
2      from flask import Flask
3      from flask_sqlalchemy import SQLAlchemy
4      from flask_migrate import Migrate
5      from flask_bcrypt import Bcrypt
6      from flask_login import LoginManager
7
8      basedir = os.path.abspath(os.path.dirname(__file__))
9
10     app = Flask(__name__)
11     app.secret_key = b"secret"
12
13     app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:/// ' + os.path.join(basedir, 'site.db')
14     db = SQLAlchemy(app)
15
16     migrate = Migrate(app, db)
17
18     bcrypt = Bcrypt(app)
19
20     login_manager = LoginManager(app)
21     login_manager.login_view = 'login'
22     login_manager.login_message_category = 'info'
23
24     from app import views, models
25
26     @login_manager.user_loader
27     def load_user(user_id):
28         return models.User.query.get(int(user_id))
29
30     def initialize_database():
31         db.create_all()
32
33     with app.app_context():
34         initialize_database()
```