Додаток А. Лістинг коду проекту

diploma/webapp

__init__.py

```
from flask import Flask, render_template

from flask_login import LoginManager

from flask_migrate import Migrate


from webapp.admin.views import blueprint as admin_blueprint

from webapp.news.views import blueprint as news_blueprint

from webapp.db import db

from webapp.user.models import User

from webapp.user.views import blueprint as user_blueprint


def create_app():

app = Flask(__name__)

app.config.from_pyfile('config.py')

db.init_app(app)

migrate = Migrate(app,db)


login_manger = LoginManager()

login_manger.init_app(app)

login_manger.login_view = 'user.login'

app.register_blueprint(admin_blueprint)

app.register_blueprint(user_blueprint)

app.register_blueprint(news_blueprint)


@login_manger.user_loader

def load_user(user_id):

return User.query.get(user_id)


return app
```

```
#Функція create_app() створює Flask додаток з конфігурацією з файлу config.py,

# підключує базу даних SQLAlchemy та міграції з Flask-Migrate.

# Далі, ініціалізує об'єкт LoginManager для авторизації користувачів та зареєстровує блюпрінти,

# які містять маршрути та логіку відповідної частини додатку. Також визначає функцію load_user(),

# яка повертає об'єкт користувача на основі ідентифікатора користувача.
```

diploma/

webapp/news/parsers/

dou.py

```
from datetime import datetime

from webapp.db import db


from webapp.news.models import News

from bs4 import BeautifulSoup

from webapp.news.parsers.utils import get_html, save_news


def get_news_snippets():

html = get_html("https://dou.ua/lenta/tags/python/?lang=en")

if html:

soup = BeautifulSoup(html, 'html.parser')

all_news = soup.find('div', class_='b-lenta').findAll('article', class_='b-postcard')

result_news = []

for news in all_news:

find_title = news.find('h2', class_='title').find('a')

title = find_title.text.strip()
```

```
url          =          news.find('h2',
class_='title').find('a')['href']

published      =      news.find('time',
class_='date').text


try:

published = datetime.strptime(published,
'%d                                %B,
%H:%M').replace(year=datetime.now().yea
r)

except ValueError:

published = datetime.strptime(published,
'%d %B %Y, %H:%M')

save_news(title, url, published)



def get_news_content():

news_without_text                        =
News.query.filter(News.text.is_(None))

for news in news_without_text:

html = get_html(news.url)

if html:

soup        =        BeautifulSoup(html,
'html.parser')

news_text     =     soup.find('article',
class_='b-typo                          b-
typo_post').decode_contents()

if news_text:

news.text = news_text

db.session.add(news)

db.session.commit()
```

webapp/static

style.css

```
.news-content img {

max-width: 50%;

}

.tg-promo img{

max-width: 5%;

}

input[type="submit"] {

margin-top: 10px;

}
```

webapp/templates/

templates/user

registration.html

```
{% extends 'base.html' %}



{% block content %}

<h1><div       class="mx-auto       p-2"
style="width:    300px;">{{    page_title
}}</div></h1>

{% include('messages.html') %}

<div class="row" >

<div  class="mx-auto  p-2"  style="width:
300px;">

<form                          action="{{
url_for('user.process_reg')          }}"
method="POST">

{{ form.hidden_tag() }}

<div class="mb-3">

{{ form.username.label }}

{{ form.username() }}

</div>

<div class="mb-3">

{{ form.email.label }}

{{ form.email() }}

</div>

<div class="mb-3">

{{ form.password.label }}

{{ form.password() }}

</div>

<div class="mb-3">

{{ form.password2.label }}

{{ form.password2() }}
```

```
</div>

{{ form.submit() }}

</form>

</div>

{% endblock %}
```

webapp/user

views.py

```
from webapp.user.forms import LoginForm, RegistrationForm

from webapp.user.models import User

from webapp.db import db


from flask import Blueprint, flash, redirect, url_for, render_template

from flask_login import current_user, login_user, logout_user

from webapp.utils import get_redirect_target


blueprint= Blueprint('user', __name__, url_prefix='/users')


@blueprint.route('/login')

def login():

if current_user.is_authenticated:

return redirect(get_redirect_target())

title = 'Authorization'

login_form = LoginForm()

return render_template('user/login.html', page_title = title, form = login_form)


@blueprint.route('/process-login', methods=['POST'])

def process_login():

form = LoginForm()

if form.validate_on_submit():

user = User.query.filter(User.username == form.username.data).first()

if user and user.check_password(form.password.data):

login_user(user, remember=form.remember_me.data)

flash("You are logged in")

return redirect(url_for('news.index'))


flash('Mistake in username or password')

return redirect(url_for('user.login'))


@blueprint.route('/logout')

def logout():

logout_user()

flash('Logged out')

return redirect(url_for('news.index'))


@blueprint.route('/register')

def register():

if current_user.is_authenticated:

return redirect(url_for('news.index'))

title = 'Registration'

form = RegistrationForm()

return render_template('user/registration.html', page_title = title, form = form)


@blueprint.route('/process-reg', methods=['POST'])

def process_reg():

form = RegistrationForm()

if form.validate_on_submit():

news_user = User(username=form.username.data, email=form.email.data, role='user')
```

```
news_user.set_password(form.password.da
ta)

db.session.add(news_user)

db.session.commit()

flash('You are passed a registration')

return redirect(url_for('user.login'))

else:

for field, errors in
form.errors.items():

for error in errors:

flash('Problem in {} : {}'.format(

getattr(form,field).label.text,

error

))

flash('Please correct the errors in the
form')

return
redirect(url_for('user.register'))
```

#код для Flask-блупрінту, який обробляє сторінки веб-додатку пов'язані з аутентифікацією та реєстрацією користувачів.

```
#Перші дві функції - це сторінки для
входу користувача.

# Функція login показує сторінку входу,
а функція process_login перевіряє, чи
вірні введені ім'я користувача та пароль.

# Якщо вони вірні, вона залогінює
користувача та перенаправить його на
сторінку з новинами.

# В іншому випадку вона поверне його на
сторінку входу з помилкою.

#Функція logout від'єднує користувача
від системи та перенаправляє його на
головну сторінку з повідомленням про
вихід.

#Функції register та process_reg
відповідають за реєстрацію користувача.
Функція register показує сторінку
реєстрації,

# а функція process_reg перевіряє
правильність введених даних та додає
нового користувача в базу даних,

# якщо вони вірні. Якщо ж дані не вірні,

# вона повертає користувача на сторінку
реєстрації з відповідною помилкою.

#В цьому коді також використовується
шаблонизатор Jinja2 для рендерингу HTML-
сторінок.
```