**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**КУРСОВ ПРОЕКТ**

Предмет: Въведение в скриптовите езици

Тема: Сайт за новини

Ученик: Научен ръководител:

*Николай Белперчинов         Даниел Димитров*

СОФИЯ

2 0 2 3

-  **Увод**

По целия свят милиони хора разчитат на новинарски сайтове като източник на информация, включително по теми като: политика, икономика, наука и спорт. Тези онлайн източници информират хората за текущите събития и се превръщат във все по-важна част от нашето общество. За да бъдете информирани, бързият и точен достъп до новините е от съществено значение.

Моята дисертация за новинарски сайт има поредица от цели и задачи, които се надявам да постигна.

1. Осигуряването на удобна платформа за достъп до новини е една от моите крайни цел. Създаването на удобен за потребителя интерфейс, добре организирани страници и ясна навигация е това, което целя да постигна.
2. Да задоволим разнообразните интереси на нашите потребители, като събираме и предоставяме актуални новини от множество проверени и достоверни източници. В новините, които събираме, са обхванати различни области, за да предложа на нашите посетители богат избор.
3. В стремежа си към надеждни новини ние даваме приоритет на обективността, точността и справедливостта. Осигуряването на достоверност изисква старателна проверка на фактите.

Моите стремежи да изградя първокласен новинарски уебсайт се основават на постигане на цели, които предоставят актуализирано, изключително и ориентирано към потребителите съдържание, като същевременно отговарят на техните нужди и интереси. Нашата цел е да предоставим безпроблемен и приятен потребителски интерфейс.

-  **Първа глава**

Flask: Разширяване на знанията и

Python: Скриптов език

SQLAlchemy: предоставя ORM (Обектно-релационно съпоставяне) функционалност. Тя позволява удобно и ефективно взаимодействие с бази данни в Python. SQLAlchemy е изключително полезен инструмент за управление на данни в новинарски сайт.

Съчетаването на тези технологии - Flask, Python, SQLAlchemy, API, HTML и CSS, позволява създаването на функционален и интуитивен новинарски сайт. Flask и Python осигуряват основния фреймуърк и език за разработка на сайта, SQLAlchemy улеснява взаимодействието с базата данни, а API-тата позволяват достъп до външни източници на информация. HTML и CSS са използвани за създаване на структура и външен вид на уеб страниците.

-Втора Глава

1. Сто реда код на скриптов език.
2. Регистрация и вход в системата:

Проектът трябва да има функционалност за регистрация и вход на потребителите. На страницата за регистрация се изискват електронна поща, потребителско име и парола. На страницата за вход се изискват електронна поща и парола.

1. Информационна система за новини:

Проектът включва страница за новини, където потребителите могат да филтрират новините по държава и ключова дума. Това показва нуждата от наличието на база данни с новини и функционалност за извличане и показване на новините, които отговарят на избраните филтри.

1. Стилизация на интерфейса:

Изисква се стилизиран интерфейс, който да изглежда привлекателно и да осигури добро потребителско изживяване и лесна навигация през сайта. За тази цел се използват външни библиотеки за CSS и шрифтове.

1. Аутентикация и авторизация:

Системата трябва да осигури сигурен и надежден механизъм за аутентикация и авторизация на потребителите. Това може да бъде реализирано чрез функция в „auth.py“ и използва user модела на „user.py“ за удостоверяване на потребителите и ограничаване на достъпа до определени страници или функции.

2.1 Използвал съм python, защото е скриптов език и ми е лесно да го използвам.

Използвал съм „Flask“, защото лек фреймуърк на python и е лесен за употреба и разбиране. Не съм използвал Jinja2, защо ме дразни и съм употребил нормален html и CSS формат.

1. **Алгоритъма във „vews.py“** създава route за home page-а, създава route за news page-а, в който дефинираме следните „функции:”

**А) 1-ва функция:** index – в нея информацията въведена в html файла от потребителя бива наследена от променливи.

**Б) 2-ра функция**: get\_news - чрез request изпращаме запитване към API-а и правим for цикъл, чрез който достъпваме част от информацията въраната от API-а, която е запазена в друга променлива.

**В) 3-та функция**: get\_filtered\_news – филтрира информацията от втората функция, използвайки получените данни първата функция.

1. **Алгоритъма във „auth.py“** е отговорен за аутентикацията и регистрацията на потребителите във Flask уеб приложението, в който дефинираме следните функции:

**А)1-ва функция:** login - Проверява дали методът на заявката е POST.Извлича имейл и парола от формата в HTML страницата.

Използва User.query.filter\_by(email=email).first() за намиране на потребителя със съответния имейл в базата данни.

Проверява дали има намерен потребител и дали въведената парола съвпада с хешираната парола на потребителя. Ако паролата е вярна, се визуализира съобщение за успешно влизане в системата, се извиква функцията login\_user() за влизане на потребителя и се пренасочва към началната страница.

Ако паролата е грешна или имейлът не съществува, се визуализира съобщение за грешка.

**Б) 2-ра функция:** logout - Изпълнява функцията logout\_user() за излизане на текущия потребител от системата. Пренасочва към началната страница.

**В) 3-та функция**: sign\_up -Проверява дали методът на заявката е POST. Извлича имейл, потребителско име и две пароли от формата в HTML шаблона. Използва User.query.filter\_by(email=email).first() и User.query.filter\_by(first\_name=first\_name).first() за проверка за вече съществуващи потребители със съответния имейл и потребителско име. Проверява различни условия, като дължина на имейла и потребителското име, съществуването на вече регистриран имейл или потребителско име, съвпадение на паролите и дължина на паролата. Ако всички проверки преминат успешно, се създава нов потребителски обект, се добавя в базата данни и се визуализира съобщение за успешно създаден акаунт.

Потребителят се влиза в системата и се пренасочва към началната страница.

3. **Алгоритъмът в „\_\_init\_\_.py“** е отговорен за създаването и конфигурирането на Flask приложението, включително базата данни и аутентикационния мениджър. Ето съкратено обяснение на алгоритма:

**А) 1-ва функция**: create\_app - Създава Flask приложение и конфигурира неговите основни настройки. Инициализира SQLAlchemy обекта db и свързва го със създаденото приложение.

Регистрира два Blueprint-а: views и auth, които съответстват на разгледаните ранее файлове views.py и auth.py. Задава им префикси за URL пътища. Импортира модела User от файлът models.py. Създава базата данни, използвайки функцията db.create\_all() в контекста на приложението. Създава обект от LoginManager, който ще управлява аутентикацията на потребителите. Задава пътя към функцията за вход (login) на LoginManager. Инициализира LoginManager обекта със създаденото приложение. Дефинира функцията load\_user(), която ще се използва от LoginManager за зареждане на потребителя според идентификатора. Връща създаденото Flask приложение.

**Б) 2-ра функция:** create\_database - Проверява дали базата данни не съществува, като проверява дали файлът 'website/' + DB\_NAME не съществува на файловата система. Ако базата данни не съществува, извиква функцията db.create\_all() за създаване на нея. Извежда съобщение, че базата данни е създадена.

1. **Алгоритъмът в „main.py**“ е отговорен за стартирането на Flask уеб приложението. Ето кратко обяснение на алгоритъма:

**А)** Импортира функцията create\_app() от модула website.

Извиква create\_app() функцията, която създава и конфигурира Flask приложението. Присвоява създаденото приложение на променливата app. Проверява дали текущият модул е основният модул, който се изпълнява (а не импортиран от друг модул).

Ако условието е изпълнено, стартира приложението с app.run(debug=True), което пуска уеб сървър за обработка на заявки. Приложението ще се изпълнява в режим на отстраняване на грешки (debug mode), което означава, че при наличие на грешка ще бъде показвана подробна информация за грешката.

В home.html файла са включени различни външни библиотеки и стилове (Bootstrap, Font Awesome и Google Fonts). Страницата се състои от няколко секции, които са дефинирани чрез HTML елементи като <section>, <nav>, <div>, <h1>, <p>, и т.н.

В секцията <nav> се намира навигационно меню, което съдържа връзки към различни страници в зависимост от статуса на потребителя (влязъл или не). Секцията <div class="text-box"> съдържа заглавие, съобщение и бутон за вход в системата. Текстът се динамично променя в зависимост от статуса на потребителя.

В HTML кода се използват променливи и логически изрази с помощта на конструкции като {% if ... %} ... {% else %} ... {% endif %} и {{ ... }}. Тези променливи се попълват с данни и логика от Flask приложението. CSS файла "home.css" съдържа стилове за оформление на елементите в страницата. Използват се селектори, каскадни стилове и различни CSS свойства за постигане на желания дизайн и изглед на страницата.

login.html представлява страницата за вход в системата. Във файла има форма, където потребителят може да въведе своите данни за вход - имейл и парола. Формата е стилизирана с помощта на Bootstrap и CSS класове. Страницата показва съобщения за грешки или успешни операции, ако такива има. Има възможност за регистрация, като линкът "Sign up" води към страницата за регистрация. CSS стиловете задават външния вид на страницата - фоново изображение, цветове, размери и позиции на елементите.

Sing\_up.html страницата за регистрация. Кодът съдържа форма, където потребителят може да въведе своите данни за регистрация - имейл, потребителско име и парола. Формата е стилизирана с помощта на Bootstrap и CSS класове. Страницата показва съобщения за грешки или успешни операции, ако има такива. Има бутон "Sign up", който потребителят трябва да натисне, за да изпрати регистрационната форма. CSS стиловете задават външния вид на страницата - фоново изображение, цветове, размери и позиции на елементите.

News.html включва форма за филтриране на новините по държави и ключови думи, както и секция за показване на новините. Страницата има навигационно меню с връзки към "Logout" и "Home". Формата съдържа две секции - "Filter by countries" и "Filter by title or content". В първата секция има няколко отметки за избор на държави, които потребителят може да избере. Във втората секция има търсачка, където потребителят може да въведе ключова дума за филтриране на новините. Ако има филтрирани новини (променливата filtered\_news), те се показват в секцията "News Articles". Всяка новина се показва с заглавие, съдържание и линк към пълния текст на новината. Ако няма филтрирани новини, се показва съобщение "No news articles found." CSS стиловете задават външния вид на страницата - цветове, размери, позиции на елементите и фоново изображение.

В кода се използва Flask и SQLAlchemy за създаване на базата данни.

Структурата на базата данни включва една таблица - таблицата "User", която съхранява информация за потребителите на приложението.

Структурата на таблицата "User" включва следните колони:

id: уникален идентификатор за всеки потребител.

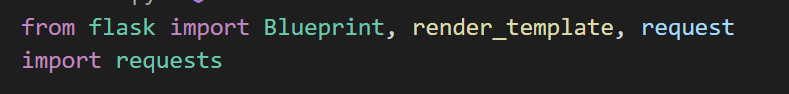
username: потребителско име на потребителя.

password: криптирана парола на потребителя.

Класът "User" е дефиниран във файлът "models.py" и е свързан с таблицата "User" в базата данни. Този клас съдържа методи и функционалности, свързани с потребителите, като вход в системата и проверка на идентичността. Функцията create\_app() създава и конфигурира Flask приложението. В нея се задава ключ за сесията (SECRET\_KEY), настройката за връзката със SQL базата данни (SQLALCHEMY\_DATABASE\_URI), инициализира се обектът на SQLAlchemy (db) и регистрират се двата Blueprint-a (views и auth) за обработка на URL адресите. Функцията create\_database(app) се използва за създаване на самата база данни. Ако файлът с базата данни не съществува, тя се създава чрез извикване на db.create\_all().

- **Трета глава**

Импортиране на нужните модули:



Blueprint се използва за дефиниране на секции в Flask приложението.

render\_template се използва за зареждане на HTML шаблони.

request се използва за получаване на информация от HTTP заявката.

requests се използва за извършване на HTTP заявки към NewsAPI.

Създаване на Blueprint:

Картина, която съдържа текст, Шрифт, екранна снимка

Описанието е генерирано автоматично

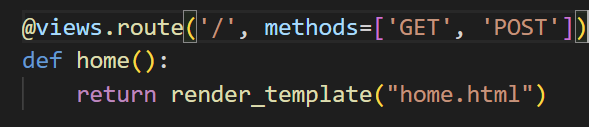
Създава се Blueprint с име 'views'.

Конфигуриране на NewsAPI:



Задава се променлива API\_KEY с ключа за NewsAPI.

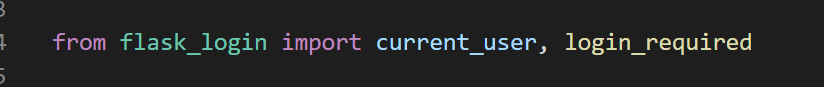
Дефиниране на пътя към началната страница:



Декораторът @views.route() определя пътя към страницата '/' (началната страница).

Функцията home() връща резултат, който включва зареждането на HTML шаблона "home.html".

Импортиране на Flask Login и защита на пътя за новини:



flask\_login се използва за управление на аутентикацията и защита на маршрути.

Дефиниране на пътя за новини:

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

Декораторът @views.route() определя пътя към страницата '/news'.

Декораторът @login\_required изисква потребителят да бъде влезнал в системата, за да достъпи този път.

Функцията index() се изпълнява при достъп до пътя /news.

Използва се request.form за получаване на данните от формата в HTML шаблона.

В зависимост от избраните страници и ключовата дума за търсене се извикват функциите get\_news() или get\_filtered\_news().

Резултатът се връща като HTML шаблон 'news.html', включващ filtered\_news и current\_user

Дефиниране на функцията get\_news()



Функцията get\_news() получава параметър selected\_countries, който има стойност по подразбиране 'us,gb,ca,de,ru,bg'.

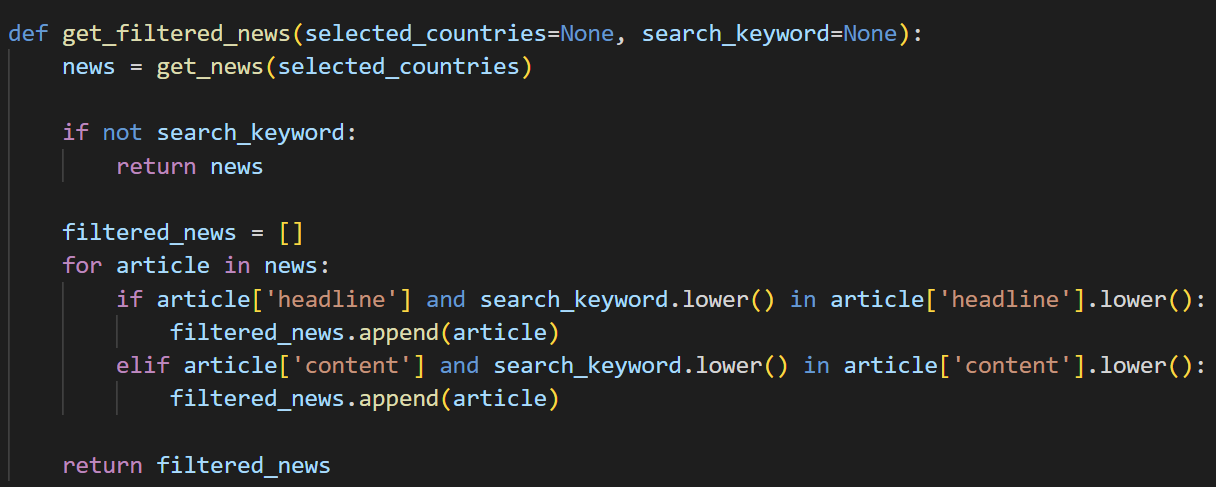
Определя се URL адресът и параметрите за NewsAPI заявката.

Заявката се изпраща с requests.get(), а резултатът се преобразува в JSON формат.

Ако заявката е успешна, новините се извличат от JSON резултата и се създава списък от новини със заглавие, съдържание и URL адрес.

Новините се връщат като резултат на функцията. Ако има грешка или няма налични новини, се връща празен списък.

Дефиниране на функцията get\_filtered\_news():



Функцията get\_filtered\_news() приема параметри selected\_countries и search\_keyword, които имат стойности по подразбиране None. Първо се извиква функцията get\_news() за да се получат всички новини. Ако няма ключова дума за търсене, се връщат всички новини. В противен случай се филтрират новините според ключовата дума за търсене, като се проверява дали тя се съдържа в заглавието или съдържанието на всяка новина. иФилтрираните новини се връщат като резултат.

Импортиране на db обекта от текущия модул (предполага се, че този модул е в същата директория като модела User).



Импортиране на UserMixin от библиотеката Flask-Login. UserMixin е клас, който предоставя реализация на някои от методите, необходими за управление на потребители във Flask-Login.



Импортиране на func от модула sqlalchemy.sql. func се използва за извикване на SQL функции при дефиниране на моделите.



Този ред дефинира класа User и наследява db.Model и UserMixin. db.Model е базовият клас за всички модели в SQLAlchemy и предоставя функционалността за взаимодействие с базата данни.

Картина, която съдържа Шрифт, текст, екранна снимка, Графика

Описанието е генерирано автоматично

Това дефинира колоната id в таблицата User. Тя е цялочислена (db.Integer) и е основен ключ (primary\_key=True), което означава, че тя е уникална и се използва за идентификация на потребителите.



Това дефинира колоната email в таблицата User. Тя е символен низ с максимална дължина 150 символа (db.String(150)) и е уникална (unique=True), което означава, че всеки потребител трябва да има уникален имейл адрес.



Това дефинира колоната password в таблицата User. Тя е символен низ с максимална дължина 150 символа и се използва за съхраняване на паролата на потребителя.



Това дефинира колоната first\_name в таблицата User. Тя е символен низ с максимална дължина 150 символа и се използва за съхраняване на името на потребителя



Тези дефинирани колони в модела User ще бъдат преобразувани в съответните колони в таблицата на базата данни, когато се извика db.create\_all().

Този код дефинира функцията login(), която се извиква при достъп до URL пътя /login. Той отговаря на HTTP заявките GET и POST. Когато методът на заявката е POST, функцията взима въведените от потребителя имейл и парола от формата и ги проверява в базата данни. Ако имейлът и паролата са валидни, потребителят се логва успешно, ако не, се извежда подходящо съобщение за грешка. Функцията използва функцията check\_password\_hash() от werkzeug.security, за да сравни хешираната парола от базата данни с въведената парола. След успешен вход в системата, се извежда съобщение за успех и потребителят се пренасочва към началната страница на views. Ако методът на заявката е GET, се показва шаблонът login.html, като текущият потребител се подава за контекст на шаблона.

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

Този код дефинира функцията logout(), която се извиква при достъп до URL пътя /logout. Това е маршрут за излизане от системата. Функцията извиква функцията logout\_user() от flask\_login, за да излезе текущият потребител от системата, и пренасочва потребителя към началната страница на views.

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

Този код дефинира функцията sign\_up(), която се извиква при достъп до URL пътя /sign-up. Функцията отговаря на HTTP заявките GET и POST. При GET заявка се показва шаблонът sign\_up.html, като текущият потребител се подава за контекст на шаблона. При POST заявка, функцията взима въведените от потребителя имейл, потребителско име и две пароли от формата и извършва проверки за валидност на входните данни. Ако въведените данни са валидни (например, няма потребител със същия имейл, паролите съвпадат и др.), новият потребител се създава и се записва в базата данни, след което потребителят се логва и се пренасочва към началната страница на views. Ако въведените данни не са валидни, се извежда подходящо съобщение за грешка и се показва отново шаблонът sign\_up.html.

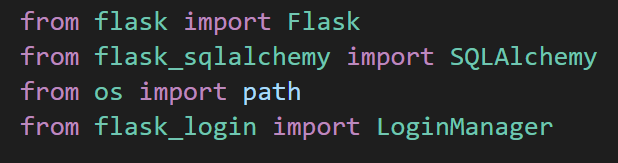
Картина, която съдържа текст, екранна снимка, софтуер

Описанието е генерирано автоматично

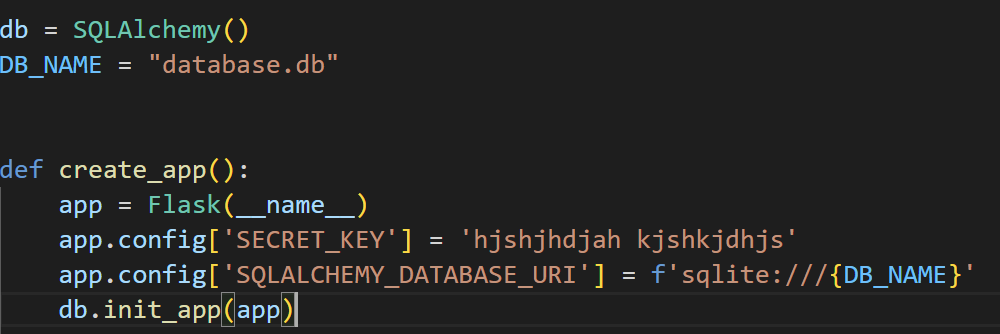
Flask се използва за създаване на Flask приложение.

SQLAlchemy се използва за взаимодействие с базата данни.

LoginManager се използва за управление на аутентикацията на потребителите.

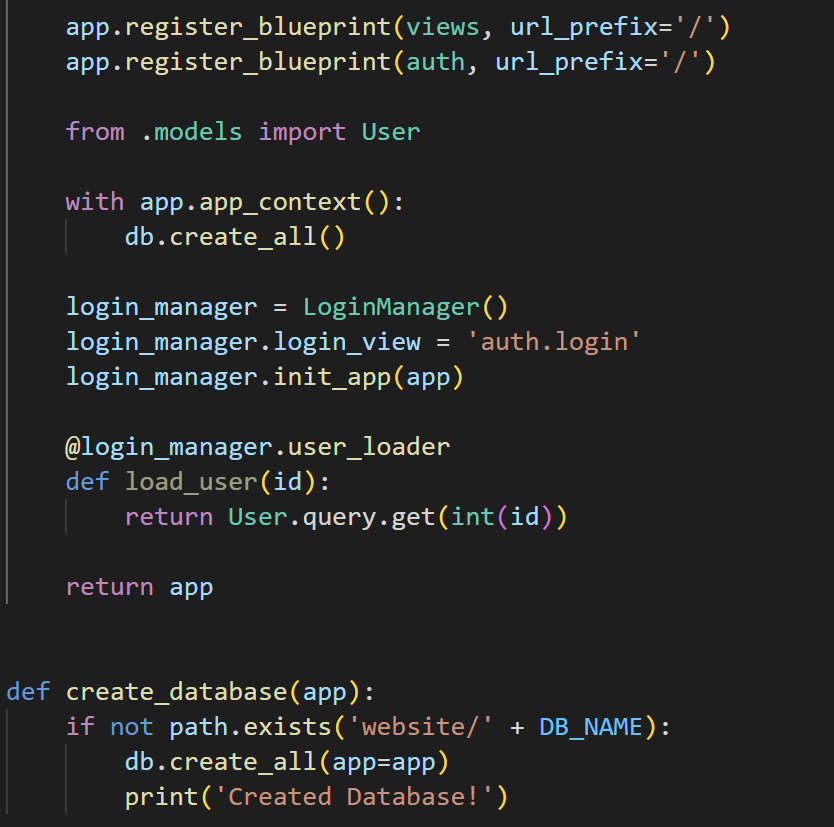


DB\_NAME представлява името на базата данни (в този случай "database.db"), а SECRET\_KEY представлява секретен ключ, използван за сигурността на сесиите и други цели в Flask приложението.



create\_app() функцията създава Flask приложение, конфигурира го и регистрира Blueprint-и и функционалности като Flask-Login. Също така се създава базата данни чрез db.create\_all() в контекста на приложението.

create\_database() функцията се извиква за да се създаде базата данни, ако все още не съществува. Тя проверява дали съществува файлът "database.db" в директорията "website" и, ако не съществува, използва db.create\_all() за да създаде всички таблици в базата данни.



Страницата "home.html" и свързания CSS файл "home.css". Ето обяснение на кода:

В home.html файла са включени различни външни библиотеки и стилове (Bootstrap, Font Awesome и Google Fonts). Страницата се състои от няколко секции, които са дефинирани чрез HTML елементи като <section>, <nav>, <div>, <h1>, <p>, и т.н.

В секцията <nav> се намира навигационно меню, което съдържа връзки към различни страници в зависимост от статуса на потребителя (влязъл или не). Секцията <div class="text-box"> съдържа заглавие, съобщение и бутон за вход в системата. Текстът се динамично променя в зависимост от статуса на потребителя.

В HTML шаблона се използват шаблонни променливи и логически изрази с помощта на конструкции като {% if ... %} ... {% else %} ... {% endif %} и {{ ... }}. Тези променливи се попълват с данни и логика от Flask приложението. CSS файла "home.css" съдържа стилове за оформление на елементите в страницата. Използват се селектори, каскадиращи стилове и различни CSS свойства за постигане на желания дизайн и изглед на страницата.

login.html представлява страницата за вход в системата. Във файла има форма, където потребителят може да въведе своите данни за вход - имейл и парола. Формата е стилизирана с помощта на Bootstrap и CSS класове. Страницата показва съобщения за грешки или успешни операции, ако такива има. Има възможност за регистрация, като линкът "Sign up" води към страницата за регистрация. CSS стиловете задават външния вид на страницата - фоново изображение, цветове, размери и позиции на елементите.

Sing\_up.html страницата за регистрация. Кодът съдържа форма, където потребителят може да въведе своите данни за регистрация - имейл, потребителско име и парола. Формата е стилизирана с помощта на Bootstrap и CSS класове. Страницата показва съобщения за грешки или успешни операции, ако такива има. Има бутон "Sign up", който потребителят трябва да натисне, за да изпрати регистрационната форма. CSS стиловете задават външния вид на страницата - фоново изображение, цветове, размери и позиции на елементите.

News.html включва форма за филтриране на новините по държави и ключови думи, както и секция за показване на новините. Страницата има навигационно меню с връзки към "Logout" и "Home". Формата съдържа две секции - "Filter by countries" и "Filter by title or content". В първата секция има няколко отметки за избор на държави, които потребителят може да избере. Във втората секция има търсачка, където потребителят може да въведе ключова дума за филтриране на новините. Ако има филтрирани новини (променливата filtered\_news), те се показват в секцията "News Articles". Всяка новина се показва с заглавие, съдържание и линк към пълния текст на новината. Ако няма филтрирани новини, се показва съобщение "No news articles found." CSS стиловете задават външния вид на страницата - цветове, размери, позиции на елементите и фоново изображение.

В кода се използва Flask и SQLAlchemy за създаване на базата данни.

Структурата на базата данни включва една таблица - таблицата "User", която съхранява информация за потребителите на приложението.

Структурата на таблицата "User" включва следните колони:

id: уникален идентификатор за всеки потребител.

username: потребителско име на потребителя.

password: криптирана парола на потребителя.

Класът "User" е дефиниран във файлът "models.py" и е свързан с таблицата "User" в базата данни. Този клас съдържа методи и функционалности, свързани с потребителите, като вход в системата и проверка на идентичността. Функцията create\_app() създава и конфигурира Flask приложението. В нея се задава ключ за сесията (SECRET\_KEY), настройката за връзката със SQL базата данни (SQLALCHEMY\_DATABASE\_URI), инициализира се обектът на SQLAlchemy (db) и регистрират се двата Blueprint-a (views и auth) за обработка на URL адресите. Функцията create\_database(app) се използва за създаване на самата база данни. Ако файлът с базата данни не съществува, тя се създава чрез извикване на db.create\_all().

* Четвърта глава:

1. На машината трябва да бъде изтеглен Python(<https://www.tutorialspoint.com/how-to-install-python-in-windows>)
2. На машината трябва да се изтегли Visual Studio Code (<https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/>)
3. Отваряте терминала на компютъра и изписвате следните команди:  
     
   Pip install flask

pip install flask-login

pip install flask-sqlalchemy

pip install request

1. Използвайки командата cd(change directory) до папката, в която желаете да поставите програмата
2. След това напишете в терминала „git clone<https://github.com/TheUnforgivenI/NewsWorld.git>“
3. Отваряте вече инсталираното Visual Studio Code и отваряте папката, в която се намира проекта
4. Проверете дали се намирате в папка „web“
5. В терминала, който е в долната част на Visual Studio Code напишете: „python3 main.py“
6. Копирайте линка, който ще се изпише в терминала и го поставете в нов прозорец на браузър (Пример за линк : <http://127.0.0.1:5000/>)
7. Тогава ще Ви се отвори нашето уеб приложение ще се отвори във вашия прозорец
8. Натиснете log in бутона горе в дясно
9. Ако нямате акаут натиснете синия бутон sign up, за да си създадете профил, а ако вече имате акаунт влезте в него
10. Сега натиснете бутона в средата „Click here to try !“ или бутона горе „News”
11. Кликнете върху една от кутийките до, която има желаната от вас абревиатура на държава и натиснете бутона „Search news“
12. Ще ви излязат новините от желаната от вас държава
13. Можете да натиснете бутона „read more“ под избраната от Вас новина, който ще Ви изпрати към официалния сайт на новината
14. Кликнете върху една от правоъгълника до, която има желаната от вас абревиатура на държава и натиснете бутона „Search news“
15. Можете да натиснете бутона „read more“ под избраната от Вас новина, който ще Ви изпрати към официалния сайт на новината

-  **Заключение**

В дипломната работа беше разработен код, който беше обект на изследване и разработка. Целта беше да се създаде функционално уеб приложение, което да предоставя възможности за регистрация, вход в системата, преглед на новини.

След анализ и тестване кода, може да се заключи, че беше постигната успешна имплементация на необходимата функционалност. Файловете signup.html, login.html и home.html съдържат съответния HTML за страниците за регистрация, вход и начална страница. Те се основават на добре структуриран HTML код и включват необходимите форми и елементи за въвеждане на информация от потребителите.

Файлът news.html представлява новинарско приложение, което включва навигационно меню и форма за филтриране на новините. Със сигурност тази функционалност би била полезна за потребителите, тъй като им дава възможност да фокусират търсенето си спрямо държава или ключова дума.

Подобренията, които бих предложил за допълнително усъвършенстване на разработката, са следните:

1. Добавяне на валидация на данните: В момента липсва валидация на входните данни от потребителите. Следва да се извърши валидация на въведената информация, преди тя да бъде обработена или запазена в базата данни.
2. Добавяне на възможност за възстановяване на парола: Текущата реализация не предоставя функционалност за възстановяване на забравена парола. Това може да бъде постигнато, чрез добавянето опция за възстановяване, например чрез изпращане на временна парола на потребителския имейл.
3. Усъвършенстване на дизайна и стиловете: Въпреки че текущите файлове включват CSS библиотеки за стилизиране, може да се извърши допълнителна работа по подобряване на дизайна и съобразяването му със съвременните стандарти за уеб дизайн.
4. С обогатяването и усъвършенстването на проекта може да се създаде по-пълноценно и използваемо уеб приложение, което да отговаря на нуждите на потребителите.

-  **Използвана литература**

1. <https://www.youtube.com/watch?v=dam0GPOAvVI&t=2729s>

2. <https://newsapi.org/>