Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка Web-системи для оцінювання і перевірки знань студентів»

Студента 4 курсу, 9 групи
спеціальності
122 «Комп'ютерні науки»
спеціалізації
«Комп'ютерні науки»

_____
підпис студента

Гречишніков Павло Олексійович

Науковий керівник
кандидат педагогічних наук,
доцент

_____
підпис керівника

Дивак Володимир Валерійович

Гарант освітньої програми
доктор фізико-математичних наук,
професор

_____
підпис керівника

Демідов Павло Георгійович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»
Спеціалізація «Комп'ютерні науки»

Затверджую
Зав. кафедри _____ Пурський О.І.
« » 2020 р.

Завдання
на випускну кваліфікаційну роботу студенту

Гречишнікову Павлу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи

«Розробка Web-системи для оцінювання і перевірки знань студентів»
Затверджена наказом ректора від *«15» грудня 2020 р. № 3780*

2. Строк здачі студентом закінченої роботи *05 листопада 2020 року*

3. Цільова установка та вихідні дані до роботи
Мета роботи: розробити Web-систему для оцінювання і перевірки знань студентів.
Об'єкт дослідження: розробка Web-системи для оцінювання і перевірки знань студентів.
Предмет дослідження: методи і технологія Web-системи для оцінки і перевірки знань студентів мовою програмування Python.

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Дивак В.В.		
2	Дивак В.В.		
3	Дивак В.В.		

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ WEB-СИСТЕМИ ОЦІНЮВАННЯ І ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ

1.1. Поняття і зміст web-системи оцінювання і перевірки знань студентів

1.2. Особливості розробки web-системи оцінювання і перевірки знань студентів

РОЗДІЛ 2. МОДЕЛЬ РОЗРОБКИ WEB-СИСТЕМИ ОЦІНЮВАННЯ І ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ

2.1. Методи і технологія розробки web-системи оцінювання і перевірки знань студентів

2.2. Модель розробки web-системи оцінювання і перевірки знань студентів

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОВЕДЕННЯ ОНЛАЙНОВОГО АНКЕТУВАННЯ

3.1. Інформаційно-логічна модель проведення онлайнового анкетування

3.2. Програмна реалізація проведення онлайнового анкетування

ВИСНОВКИ

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи поправити до внесених змін попередньої сторінки

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*		
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*		
3	*Вступ*		
4	*РОЗДІЛ 1. Теоретичні аспекти розробки web-системи оцінювання і перевірки знань студентів*		

5	*РОЗДІЛ 2. Модель розробки web-системи оцінювання і перевірки знань студентів*		
6	*Підготовка статті у збірник наукових статей магістрів*		
7	*РОЗДІЛ 3. Розробка програмного забезпечення для проведення онлайнового анкетування*		
8	*Висновки*		
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*		
10	*Попередній захист випускної кваліфікаційної роботи*		
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*		
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*		
13	*Публічний захист випускної кваліфікаційної роботи*		

8. Дата видачі завдання « » 2020 р.

9. Керівник випускної кваліфікаційної роботи

Дивак В.В.

(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов А.Г.

(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник

Гречишніков П.О.

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Гречишнікова П.О._____

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Демідов П.Г.____

(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.____

(підпис, прізвище, ініціали)

«_____»_____2020 р.

ВСТУП

Актуальність дослідження зумовлена стрімким розвитком використання засобів автоматизації, таких як електронні журнали, що допомагають автоматизувати підрахунок оцінок, або системи оцінювання знань, де особливої ваги набувають алгоритми підрахунку оцінок студентів, механізми автоматичного збереження і сортування бланків відповідей.

Процес оцінки і перевірки знань студентів є дуже трудомістким і вимагає велику кількість часу викладача поза часом занять, адже потрібно не тільки провести заняття з перевірки знань, а й перевірити самі роботи, виставити оцінки, тощо.

В даному випадку, система автоматизації оцінки і перевірки знань студентів є не тільки гарним впровадженням, а й абсолютною необхідністю, яка в декілька разів спростить цей процес.

Мета роботи: розробити Web-систему для оцінювання і перевірки знань студентів.

Об’єкт дослідження: розробка Web-системи для оцінювання і перевірки знань студентів.

Предмет дослідження: методи і технологія Web-системи для оцінки і перевірки знань студентів мовою програмування Python.

Відповідно до предмету та мети дослідження визначено наступні **завдання**:

- Проаналізувати теоретичні підходи розробки Web-системи для оцінювання і перевірки знань студентів
- Розробити модель Web-системи для оцінювання і перевірки знань студентів

- З’ясувати особливості використання програмно-технічних засобів Web-системи для оцінювання і перевірки знань студентів
- Розробити і перевірити Web-систему для оцінювання і перевірки знань студентів мовою програмування Python

Методи дослідження: теоретичний аналіз наукової літератури, порівняння, систематизація, класифікація дали змогу дослідити і узагальнити матеріали з питань розробки Web-системи для оцінювання і перевірки знань студентів, розробити хід дослідження, окрім цього були використані пошуковий, аналітичний, статистичний, емпіричний, метод пошукового аналізу, метод порівняння, а також метод стратегічних пріоритетів. У ході дослідження було використано прикладні методи дослідження із застосуванням програмних систем для реалізації та розробки web-додатку.

Практичне значення полягає у розробці та впровадженні методів і технологій Web-системи для оцінювання і перевірки знань студентів мовою програмування Python.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ WEB-СИСТЕМИ ОЦІНЮВАННЯ І ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ

1.1. Поняття і зміст web-системи оцінювання і перевірки знань студентів

Класифікація в освіті - це спроба застосувати стандартизовані вимірювання різного рівня успішності в курсі. Оцінки можуть бути призначені як літери (наприклад, від A до F), як діапазон (наприклад, від 1 до 6), у відсотках або як число з можливих підсумків (наприклад, із 100).

У деяких країнах оцінки усереднюються, щоб створити середній бал (GPA). Середній бал розраховується за кількістю балів, які студент заробляє за певний проміжок часу. Середній бал часто розраховується для студентів середніх шкіл, студентів та аспірантів і може бути використаний потенційними роботодавцями чи навчальними закладами для оцінки та порівняння заявників. Сукупний середній бал (CGPA), який іноді називають просто середнім балом, є показником успішності студента на всіх його курсах.

Історик Єльського університету Джордж У. Пірсон пише: "За традицією перші класи, видані в Єльському університеті (і, можливо, перші в країні), були видані в 1785 році, коли президент Езра Стайлз після обстеження 58 людей похилого віку записав у свій щоденник що існували „Двадцять Оптімі, шістнадцять других Оптімі, дванадцять Інферіорес (Боні), десять Пежорів" . Боб Марлін стверджує, що концепція оцінки роботи студентів кількісно була розроблена вихователем на ім’я Вільям Фаршиш і вперше реалізована Кембриджський університет у 1792 р. Це твердження поставив під сумнів Крістофер Стрей, який вважає, що докази Фаріша як винахідника числової позначки є непереконливими. У статті Стрея також пояснюється складний взаємозв'язок між способом іспиту (усним чи письмовим) та різними

філософіями освіти, які ці способи передбачають як для вчителя, так і для студента. Як технологія, класифікація обох форм і відображає багато фундаментальних областей теорії та практики освіти.

Критикують, що оцінки - це лише короткочасні знімки того, наскільки студент навчився за певний проміжок часу, що лише частково відображає фактичну успішність і не враховує в достатній мірі індивідуальний розвиток учнів. Подібним чином погані оцінки протягом більш тривалого періоду часу створюють у студентів враження, що вони навчаться дуже мало або нічого, що ставить під загрозу вроджену внутрішню мотивацію кожної дитини вчитися. Діти, які вже втратили бажання вчитися і навчаються лише за свої оцінки, не мають підстав продовжувати навчання після того, як вони набрали найкращу оцінку. Крім того, погані оцінки є деструктивним зворотним зв’язком для студентів, оскільки вони не надають ніякої конструктивної допомоги, а лише абсолютні ключові показники. Також критикують, що спосіб мислення, який часто можна простежити за системою оцінювання, що погані оцінки призводять до поганих перспектив у майбутньому, призводить до збентеження, тиску та стресу серед батьків та дітей.

Критикується те, що студенти часто не вчаться для свого майбутнього життя або через інтерес до матеріалу, а лише за оцінки та пов'язаний з ними статус, що сприяє булімічному навчанню.

Німецький філософ і публіцист Річард Давід Прехт критикує систему шкільних оцінок у своїй книзі Anna, die Schule und der liebe Gott: Der Verrat des Bildungssystems an unseren Kindern. Він вважає, що цифри від 1 до 6 не виправдовують особистості дітей. На його думку, оцінки не є ні значущими, ні диференційованими, а отже, не корисними. Наприклад, на запитання, чи став студент більш мотивованим, зацікавився якоюсь темою, навчився краще справлятися з невдачами та чи не виробив він нових ідей, відповіді оцінками не даються. Натомість Прехт пропонує диференційоване письмове оцінювання

шляху навчання та розвитку учнів. На його думку, система оцінювання походить з неінформованої психологічно та педагогічно епохи і не належить до 21 століття.

Веб-програма - це прикладне програмне забезпечення, яке працює на веб-сервері, на відміну від комп'ютерних програм, що запускаються локально в операційній системі (ОС) пристрою. Доступ до веб-програм користувач здійснює через веб-браузер з активним підключенням до мережі. Ці програми програмуються за допомогою змодельованої структури клієнт-сервер - користувачеві ("клієнту") надаються послуги через зовнішній сервер, розміщений сторонньою стороною. Приклади часто використовуваних веб-програм включають: веб-пошту, роздрібні продажі в Інтернеті, Інтернет-банкінг та Інтернет-аукціони.

Загальна різниця між будь-якою динамічною веб-сторінкою та "веб-програмою" незрозуміла. Веб-сайти, які найімовірніше називатимуть "веб-додатками", - це веб-сайти, які мають подібну функціональність до настільного програмного забезпечення чи мобільного додатка. HTML5 представив явну підтримку мови для створення програм, які завантажуються як веб-сторінки, але можуть зберігати дані локально та продовжувати функціонувати в автономному режимі.

Односторінкові програми більш схожі на програми, оскільки вони відкидають більш типову веб-парадигму переміщення між різними сторінками з різними URL-адресами. Односторінкові фреймворки можуть бути використані для пришвидшення розробки такого веб-додатку для мобільної платформи.

У попередніх обчислювальних моделях, таких як клієнт-сервер, навантаження на обробку програми розподілялося між кодом на сервері та кодом, встановленим на кожному клієнті локально. Іншими словами, програма мала власну заздалегідь скомпільовану клієнтську програму, яка служила в якості її користувальницького інтерфейсу і мала встановлюватися окремо на

персональному комп'ютері кожного користувача. Оновлення серверного коду програми, як правило, також вимагає оновлення коду на стороні клієнта, встановленого на кожній робочій станції користувача, що збільшує вартість підтримки та зменшує продуктивність. Крім того, як клієнтський, так і серверний компоненти програми, як правило, були тісно пов'язані з певною архітектурою комп'ютера та операційною системою, і перенесення їх на інші часто було надмірно дорогим для всіх, крім найбільших додатків (Сьогодні рідні програми для мобільних пристроїв також мають деякими або всіма з попередніх питань).

На відміну від них, веб-програми використовують веб-документи, написані у стандартному форматі, наприклад HTML та JavaScript, які підтримуються різноманітними веб-браузерами. Веб-програми можна розглядати як конкретний варіант програмного забезпечення клієнт-сервер, де клієнтське програмне забезпечення завантажується на клієнтську машину під час відвідування відповідної веб-сторінки, використовуючи стандартні процедури, такі як HTTP. Оновлення програмного забезпечення клієнта може відбуватися кожного разу, коли відвідується веб-сторінка. Під час сеансу веб-браузер інтерпретує та відображає сторінки та виступає універсальним клієнтом для будь-якої веб-програми.

На початку Інтернету кожна окрема веб-сторінка доставлялася клієнтові як статичний документ, але послідовність сторінок все одно могла забезпечити інтерактивний досвід, оскільки введення користувачем поверталося через елементи веб-форми, вбудовані в розмітку сторінки. Однак кожна суттєва зміна веб-сторінки вимагала повернення назад на сервер для оновлення всієї сторінки.

У 1995 році Netscape представив мову сценаріїв на стороні клієнта під назвою JavaScript, що дозволяє програмістам додавати деякі динамічні елементи до користувацького інтерфейсу, що працює на стороні клієнта. Отже, замість надсилання даних на сервер для створення цілої веб-сторінки, вбудовані

сценарії завантаженої сторінки можуть виконувати різні завдання, такі як перевірка вводу або показ / приховування частин сторінки.

У 1996 році Macromedia представив Flash, програвач векторних анімацій, який можна було додати до браузерів як плагін для вбудовування анімації на веб-сторінки. Це дозволило використовувати мову сценаріїв для програмування взаємодії на стороні клієнта без необхідності спілкування з сервером.

У 1999 році концепція "веб-програми" була введена мовою Java у версії 2.2 специфікації сервлетів. На той час і JavaScript, і XML вже були розроблені, але Ajax ще не було створено, і об'єкт XMLHttpRequest був нещодавно представлений в Internet Explorer 5 як об'єкт ActiveX.

У 2005 році був введений термін Ajax, і такі програми, як Gmail, почали робити свої клієнтські сторони дедалі більш інтерактивними. Сценарій веб-сторінки може зв’язатись із сервером для зберігання / отримання даних без завантаження цілої веб-сторінки.

У 2007 році Стів Джобс оголосив, що веб-програми, розроблені в HTML5 з використанням архітектури AJAX, будуть стандартним форматом для програм для iPhone. Набір програмного забезпечення для розробки програмного забезпечення (SDK) не потрібен, і програми будуть повністю інтегровані в пристрій за допомогою механізму браузера Safari. Пізніше цю модель було замінено на App Store, як засіб запобігання джейлбрейкам та заспокоєння розчарованих розробників.

У 2014 році було доопрацьовано HTML5, який надає графічні та мультимедійні можливості без необхідності плагінів на стороні клієнта. HTML5 також збагатив семантичний зміст документів. API та об'єктна модель документа (DOM) більше не є задумом, а є основними частинами специфікації HTML5. API WebGL відкрив шлях для вдосконаленої 3D-графіки на основі полотна HTML5 та мови JavaScript. Вони мають важливе значення у створенні справді незалежних від платформи та браузера багатих веб-додатків.

У 2016 році під час щорічної конференції Google IO Ерік Бідельман (старший інженер програм розробників персоналу) представив Progressive Web Apps (PWA) як новий стандарт у веб-розробці. Джефф Буртофт, головний менеджер програм у Microsoft, сказав: "Google просунувся вперед з прогресивними веб-програмами, і після тривалого процесу ми вирішили, що нам потрібно повністю його підтримати". Таким чином, Microsoft і Google підтримували стандарт PWA.

Завдяки Java, JavaScript, CSS, Flash, Silverlight та іншим технологіям можливі спеціальні методи, такі як малювання на екрані, відтворення звуку та доступ до клавіатури та миші. Багато служб працювали над тим, щоб поєднати все це в більш звичний інтерфейс, який приймає вигляд операційної системи. Ці технології також підтримують методи загального призначення, такі як перетягування та падіння. Веб-розробники часто використовують сценарії на стороні клієнта, щоб додати функціональність, особливо для створення інтерактивного інтерфейсу, який не вимагає перезавантаження сторінки. Нещодавно були розроблені технології для координації сценаріїв на стороні клієнта з такими серверними технологіями, як ASP.NET, J2EE, Perl / Plack та PHP.

Ajax, техніка веб-розробки, що використовує поєднання різних технологій, є прикладом технології, яка створює більш інтерактивний досвід.

Зазвичай програми розбиваються на логічні фрагменти, які називаються "рівнями", де кожному рівню присвоюється роль. Традиційні програми складаються лише з 1 рівня, який розміщений на клієнтській машині, але веб-програми за своєю природою піддаються багаторівневому підходу. Хоча можливих багато варіантів, найпоширенішою структурою є трирівнева програма. У своїй найпоширенішій формі три рівні називаються презентацією, застосуванням та зберіганням, у цьому порядку. Веб-браузер - це перший рівень (презентація), механізм, що використовує деякі динамічні технології веб-вмісту

(такі як ASP, CGI, ColdFusion, Dart, JSP / Java, Node.js, PHP, Python або Ruby on Rails) - середній рівень (логіка програми), а база даних - це третій рівень (сховище). Веб-браузер надсилає запити середньому рівню, який обслуговує їх, роблячи запити та оновлення щодо бази даних та генеруючи користувальницький інтерфейс.

Для більш складних додатків 3-рівневе рішення може бути недостатнім, і може бути корисним використовувати підхід з ярусами, де найбільшою вигодою є розбиття бізнес-логіки, яка знаходиться на рівні додатків, на більш дрібну модель. Ще однією перевагою може бути додавання рівня інтеграції, який відокремлює рівень даних від решти рівнів, надаючи простий у використанні інтерфейс для доступу до даних. Наприклад, доступ до даних клієнта здійснюється шляхом виклику функції "list_clients ()" замість того, щоб робити запит SQL безпосередньо до таблиці клієнта в базі даних. Це дозволяє замінити базову базу даних, не вносячи жодних змін в інші рівні.

Є деякі, хто розглядає веб-додаток як дворівневу архітектуру. Це може бути "розумний" клієнт, який виконує всю роботу і запитує "німий" сервер, або "німий" клієнт, який покладається на "розумний" сервер. Клієнт обробляв би рівень презентації, сервер мав би базу даних (рівень зберігання), а бізнес-логіка (рівень додатків) знаходився б на одному з них або на обох. Хоча це збільшує масштабованість додатків та відокремлює дисплей та базу даних, це все одно не дає можливості справжньої спеціалізації шарів, тому більшість програм переростають цю модель.

Новою стратегією для компаній, що займаються прикладним програмним забезпеченням, є надання веб-доступу до програмного забезпечення, яке раніше розповсюджувалось як локальні програми. Залежно від типу програми, може знадобитися розробка абсолютно іншого інтерфейсу на основі браузера або просто адаптація існуючого додатка до використання іншої технології презентації. Ці програми дозволяють користувачеві платити щомісячну або

щорічну плату за використання програмного забезпечення без необхідності встановлювати його на локальний жорсткий диск. Компанія, яка дотримується цієї стратегії, відома як постачальник послуг додатків (ASP), і в даний час ASP приділяють велику увагу в галузі програмного забезпечення.

Порушення безпеки в таких видах програм викликає велике занепокоєння, оскільки воно може включати як інформацію про підприємство, так і дані приватних клієнтів. Захист цих активів є важливою частиною будь-якого веб-додатку, і є деякі ключові сфери діяльності, які повинні бути включені в процес розробки. Сюди входять процеси аутентифікації, авторизації, обробки активів, введення даних, реєстрації та аудиту. Вбудова безпеки в програми з самого початку може бути більш ефективною та менш руйнівною в довгостроковій перспективі.

Веб-програми хмарних обчислень - це програмне забезпечення як послуга (SaaS). Існують бізнес-програми, що надаються як SaaS для підприємств за фіксовану плату або в залежності від використання. Інші веб-програми пропонуються безкоштовно, часто приносячи прибуток від реклами, що відображається в інтерфейсі веб-програми.

1.2. Особливості розробки web-системи оцінювання і перевірки знань студентів

Веб-сайт - це сукупність веб-сторінок та пов’язаного вмісту, які ідентифікуються загальним доменним ім’ям та публікуються принаймні на одному веб-сервері. Помітними прикладами є wikipedia.org, google.com та amazon.com.

Усі загальнодоступні веб-сайти в сукупності складають Всесвітню павутину. Існують також приватні веб-сайти, доступ до яких доступний лише в приватній мережі, наприклад, внутрішній веб-сайт компанії для її співробітників.

Веб-сайти, як правило, присвячені певній темі або меті, такі як новини, освіта, комерція, розваги чи соціальні мережі. Гіперпосилання між веб-сторінками спрямовує навігацію по сайту, яке часто починається з домашньої сторінки.

Користувачі можуть отримати доступ до веб-сайтів на різних пристроях, включаючи настільні, ноутбуки, планшети та смартфони. Програма, що використовується на цих пристроях, називається веб-браузером.

Всесвітня павутина (WWW) була створена в 1990 році британським фізиком ЦЕРН Тімом Бернерсом-Лі. 30 квітня 1993 р. ЦЕРН оголосив, що Всесвітня павутина буде безкоштовною для використання будь-ким. До впровадження протоколу передачі гіпертексту (HTTP) для отримання окремих файлів із сервера використовувались інші протоколи, такі як протокол передачі файлів та протокол gopher. Ці протоколи пропонують просту структуру каталогів, в якій користувач переходить і де він обирає файли для завантаження. Документи найчастіше подавалися у вигляді текстових файлів без форматування або кодувались у форматах текстового процесора.

Веб-сайти можуть використовуватися по-різному: персональний веб-сайт, корпоративний веб-сайт компанії, урядовий веб-сайт, веб-сайт організації тощо. Веб-сайти можуть бути роботою приватної особи, бізнесу чи іншої організації і, як правило, присвячені конкретна тема або мета. Будь-який веб-сайт може містити гіперпосилання на будь-який інший веб-сайт, тому різниця між окремими сайтами, як сприймається користувачем, може бути розмитою.

Деякі веб-сайти вимагають реєстрації користувачів або передплати для доступу до вмісту. Приклади веб-сайтів, на які здійснюється підписка, включають багато бізнес-сайтів, веб-сайти новин, веб-сайти академічних журналів, ігрові веб-сайти, веб-сайти спільного використання файлів, дошки оголошень, веб-адреси електронної пошти, веб-сайти соціальних мереж, веб-сайти, що надають дані про фондовий ринок у режимі реального часу, а також веб-сайти, що надають різні інші послуги.

Хоча "веб-сайт" був оригінальним написанням (іноді з великої літери "Веб-сайт", оскільки "Веб" є власним іменником, коли йдеться про Всесвітню павутину), цей варіант став рідкісним, а "веб-сайт" став стандартним написанням . Усі основні керівництва стилем, такі як Чиказький посібник стилю та AP Stylebook, відображають цю зміну.

Статичний веб-сайт - це веб-сторінки, які мають веб-сторінки, що зберігаються на сервері у форматі, який надсилається клієнтському веб-браузеру. В основному він кодується мовою розмітки гіпертексту (HTML); Каскадні таблиці стилів (CSS) використовуються для контролю зовнішнього вигляду за базовим HTML. Зображення зазвичай використовуються для отримання бажаного вигляду та як частина основного змісту. Аудіо чи відео також можна вважати "статичним" вмістом, якщо він відтворюється автоматично або, як правило, є неінтерактивним. Цей тип веб-сайтів зазвичай відображає однакову інформацію для всіх відвідувачів. Подібно до роздачі друкованої брошури клієнтам або клієнтам, статичний веб-сайт, як правило,

надає послідовну стандартну інформацію протягом тривалого періоду часу. Незважаючи на те, що власник веб-сайту може періодично робити оновлення, редагування тексту, фотографій та іншого вмісту здійснюється вручну, і може знадобитися базові навички дизайну веб-сайту та програмне забезпечення. Прості форми або маркетингові приклади веб-сайтів, такі як класичний веб-сайт, веб-сайт із п'ятьма сторінками або веб-сайт брошури, часто є статичними веб-сайтами, оскільки вони представляють користувачеві заздалегідь визначену, статичну інформацію. Це може включати інформацію про компанію та її продукти та послуги через текст, фотографії, анімацію, аудіо / відео та навігаційні меню.

Статичні веб-сайти все ще можуть використовувати серверні компоненти (SSI) як зручність редагування, наприклад, спільний доступ до загального рядка меню на багатьох сторінках. Оскільки поведінка сайту до читача все ще залишається статичним, це не вважається динамічним сайтом.

Динамічний веб-сайт - це веб-сайт, який часто і автоматично змінюється або налаштовується. Динамічні сторінки на сервері генеруються "на льоту" за допомогою комп'ютерного коду, який виробляє HTML (CSS відповідає за зовнішній вигляд і, отже, статичні файли). Існує широкий спектр програмних систем, таких як CGI, Java-сервлети та Java Server Pages (JSP), Active Server Pages і ColdFusion (CFML), які доступні для створення динамічних веб-систем та динамічних веб-сайтів. Різні фреймворки веб-програм та системи веб-шаблонів доступні для загальновживаних мов програмування, таких як Perl, PHP, Python та Ruby, щоб полегшити та спростити створення складних динамічних веб-сайтів.

Сайт може відображати поточний стан діалогу між користувачами, відстежувати мінливу ситуацію або надавати інформацію певним чином персоналізовану відповідно до вимог окремого користувача. Наприклад, коли

запитується головна сторінка сайту новин, код, що працює на веб-сервері, може поєднувати збережені фрагменти HTML із новинами, отриманими з бази даних або іншого веб-сайту за допомогою RSS, щоб створити сторінку, що включає найсвіжішу інформацію. Динамічні сайти можуть бути інтерактивними, використовуючи HTML-форми, зберігаючи та зчитуючи файли cookie браузера, або створюючи ряд сторінок, що відображають попередню історію кліків. Інший приклад динамічного вмісту - це коли роздрібний веб-сайт з базою даних медіа-продуктів дозволяє користувачеві вводити запит на пошук, наприклад за ключовим словом "Бітлз". У відповідь вміст веб-сторінки спонтанно змінить те, як він виглядав раніше, а потім відобразить список продуктів "Бітлз", таких як компакт-диски, DVD-диски та книги. Динамічний HTML використовує код JavaScript, щоб вказувати веб-браузеру, як інтерактивно змінювати вміст сторінки. Одним із способів моделювання певного типу динамічного веб-сайту, уникаючи втрати продуктивності ініціювання динамічного механізму для кожного користувача або підключення, є періодична автоматична регенерація великої серії статичних сторінок.

Ранні веб-сайти мали лише текст, а незабаром і зображення. Потім плагіни веб-браузера використовувались для додавання аудіо, відео та інтерактивності (наприклад, для багатофункціональної програми Інтернету, яка відображає складність настільної програми, як текстовий процесор). Прикладами таких плагінів є Microsoft Silverlight, Adobe Flash, Adobe Shockwave та аплети, написані на Java. HTML 5 містить положення щодо аудіо та відео без плагінів. JavaScript також вбудований у більшість сучасних веб-браузерів і дозволяє творцям веб-сайтів надсилати код веб-браузеру, який вказує йому, як інтерактивно змінювати вміст сторінки та спілкуватися з веб-сервером, якщо це необхідно. Внутрішнє представлення вмісту браузера відомо як об’єктна модель документа (DOM), а техніка - динамічний HTML.

WebGL (Web Graphics Library) - це сучасний API JavaScript для візуалізації інтерактивної 3D-графіки без використання плагінів. Це дозволяє інтерактивний контент, такий як 3D-анімація, візуалізація та відео-пояснення, представленим користувачам найбільш інтуїтивно зрозумілим способом.

Тенденція 2010 року на веб-сайтах під назвою "адаптивний дизайн" забезпечила найкращий досвід перегляду, оскільки надає користувачам макет на основі пристрою. Ці веб-сайти змінюють свій макет відповідно до пристрою або мобільної платформи, що забезпечує багатий досвід користувачів.

Веб-сайти можна розділити на дві великі категорії - статичні та інтерактивні. Інтерактивні сайти є частиною спільноти веб-сайтів Web 2.0 і дозволяють взаємодіяти між власником сайту та відвідувачами або користувачами сайту. Статичні сайти обслуговують або збирають інформацію, але не дозволяють взаємодіяти з аудиторією або користувачами безпосередньо. Деякі веб-сайти є інформаційними або виготовляються ентузіастами або для особистого користування чи розваги. Багато веб-сайтів мають на меті заробляти гроші, використовуючи одну або кілька бізнес-моделей, зокрема:

- Розміщення цікавого контенту та продаж контекстної реклами або шляхом прямих продажів, або через рекламну мережу.
- Електронна комерція: товари чи послуги купуються безпосередньо через веб-сайт
- Реклама товарів або послуг, доступних у цегельному та будівельному бізнесі
- Freemium: базовий вміст доступний безкоштовно, але преміум-вміст вимагає оплати (наприклад, веб-сайт WordPress, це платформа з відкритим кодом для створення блогу чи веб-сайту).

  Деякі веб-сайти можуть бути включені в одну або кілька з цих категорій. Наприклад, веб-сайт бізнесу може рекламувати продукцію бізнесу, але

також може розміщувати інформативні документи, такі як технічні документи. Існують також численні підкатегорії до перерахованих вище.

Наприклад, порносайт - це певний тип веб-сайту електронної комерції або бізнес-сайту (тобто він намагається продати членство для доступу до свого сайту) або має можливості соціальних мереж. Фан-сайт може бути посвятою власника певній знаменитості. Веб-сайти обмежені архітектурними обмеженнями (наприклад, обчислювальна потужність, присвячена веб-сайту). Дуже великі веб-сайти, такі як Facebook, Yahoo !, Microsoft та Google, використовують багато серверів та обладнання для балансування навантажень, наприклад, комутатори служб вмісту Cisco, для розподілу навантажень відвідувачів на декілька комп'ютерів у різних місцях. На початок 2011 року Facebook використовував 9 центрів обробки даних із приблизно 63000 серверами.

У лютому 2009 року компанія Netcraft, компанія з моніторингу Інтернету, яка відслідковувала зростання Інтернету з 1995 р., Повідомила, що в 2009 р. Існувало 215 675 903 веб-сайтів з доменними іменами та вмістом на них, порівняно з лише 19 732 веб-сайтами в серпні 1995 р.

Після досягнення 1 мільярда веб-сайтів у вересні 2014 року, етап, підтверджений NetCraft в огляді веб-серверів за жовтень 2014 року, і те, що Інтернет-статистика стала першим, хто повідомив про це, про що свідчить цей твіт від самого винахідника Всесвітньої мережі, Тіма Бернерса Лі - кількість веб-сайтів у світі згодом зменшилась, повернувшись до рівня нижче 1 мільярда. Це пов'язано з щомісячними коливаннями кількості неактивних веб-сайтів. Кількість веб-сайтів до березня 2016 р. Продовжувала зростати до понад 1 млрд. І з тих пір продовжує зростати.

Веб-розробка - це робота, пов'язана з розробкою веб-сайту для Інтернету (Всесвітня павутина) або інтрамережі (приватна мережа). Веб-розробка може варіюватися від розробки простої однієї статичної сторінки простого тексту до

складних Інтернет-програм (Інтернет-додатків), електронного бізнесу та послуг соціальних мереж. Більш повний перелік завдань, до яких зазвичай відноситься веб-розробка, може включати веб-інженерію, веб-дизайн, розробку веб-вмісту, зв'язок з клієнтом, сценарії на стороні клієнта / сервера, налаштування безпеки веб-сервера та мережі та розвиток електронної комерції.

Серед веб-спеціалістів "веб-розробка" зазвичай відноситься до основних недизайнерських аспектів побудови веб-сайтів: написання розмітки та кодування. Веб-розробка може використовувати системи управління вмістом (CMS), щоб зробити зміст вмістом простішим та доступнішим з базовими технічними навичками.

Для великих організацій та бізнесу групи веб-розробників можуть складатися з сотень людей (веб-розробників) і дотримуватися стандартних методів, таких як Agile, під час розробки веб-сайтів. Менші організації можуть вимагати лише одного постійного або підрядного розробника, або вторинне призначення на відповідні посади, такі як графічний дизайнер або технік інформаційних систем. Веб-розробка може бути спільним зусиллям між департаментами, а не областю призначеного департаменту. Існує три різновиди спеціалізації веб-розробників: розробник з інтерфейсу, розробник з інтерфейсу та розробник із повним стеком. Інтернетні розробники несуть відповідальність за поведінку та візуальні ефекти, які працюють у браузері користувача, тоді як інтерфейсні розробники мають справу з серверами.

З моменту комерціалізації Інтернету веб-розробка стала зростаючою галуззю. Зростання цієї галузі відбувається за рахунок підприємств, які бажають використовувати свій веб-сайт для реклами та продажу товарів та послуг споживачам.

РОЗДІЛ 2. МОДЕЛЬ РОЗРОБКИ WEB-СИСТЕМИ ОЦІНЮВАННЯ І ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ

2.1. Технологія розробки web-системи оцінювання і перевірки знань студентів

Існує багато інструментів з відкритим кодом для веб-розробки, таких як BerkeleyDB, GlassFish, стек LAMP (Linux, Apache, MySQL, PHP) та Perl / Plack. Це дозволило звести витрати на навчання веб-розробці до мінімуму. Ще одним фактором, що сприяє зростанню галузі, стало зростання простого у використанні програмного забезпечення для веб-розробки WYSIWYG, такого як Adobe Dreamweaver, BlueGriffon та Microsoft Visual Studio. Для використання такого програмного забезпечення все ще необхідні знання мови розмітки HyperText (HTML) або мов програмування, але основи можна швидко вивчити та впровадити.

Постійно зростаючий набір інструментів та технологій допоміг розробникам створювати більш динамічні та інтерактивні веб-сайти. Крім того, веб-розробники тепер допомагають надавати програми як веб-служби, які традиційно були доступні лише як програми на настільному комп'ютері. Це дало багато можливостей для децентралізації розповсюдження інформації та засобів масової інформації. Приклади можна побачити із зростанням хмарних сервісів, таких як Adobe Creative Cloud, Dropbox та Google Drive. Ці веб-служби дозволяють користувачам взаємодіяти з програмами з багатьох місць, замість того, щоб бути прив'язаними до певної робочої станції для свого середовища програм.

Прикладами кардинальних перетворень у спілкуванні та комерції на чолі з веб-розробкою є електронна комерція. Інтернет-сайти аукціонів, такі як eBay, змінили спосіб споживачів знаходити та купувати товари та послуги. Інтернет-

магазини, такі як Amazon.com та Buy.com (серед багатьох інших), змінили досвід покупок та торгівлі для багатьох споживачів. Іншим прикладом трансформаційного спілкування, керованого веб-розробкою, є блог. Веб-програми, такі як WordPress та Movable Type, створили середовища для ведення блогів для окремих веб-сайтів. Поширене використання систем управління вмістом з відкритим кодом та систем управління корпоративним контентом розширило вплив веб-розробки на онлайн-взаємодію та спілкування.

Розробка веб-сайтів також вплинула на особисті мережі та маркетинг. Веб-сайти вже не є просто інструментами для роботи чи комерції, а ширше служать для спілкування та соціальних мереж. Такі веб-сайти, як Facebook та Twitter, надають користувачам платформу для спілкування, а організації - більш особистий та інтерактивний спосіб залучення громадськості.

Веб-розробка враховує багато міркувань безпеки, таких як перевірка помилок при введенні даних через форми, фільтрація вихідних даних та шифрування. Шкідливі практики, такі як введення SQL, можуть виконуватися користувачами з ненавмисними намірами, але лише з примітивними знаннями про веб-розробку в цілому. Сценарії можна використовувати для використання веб-сайтів, надаючи несанкціонований доступ зловмисним користувачам, які намагаються збирати таку інформацію, як адреси електронної пошти, паролі та захищений вміст, наприклад номери кредитних карток.

Дещо з цього залежить від серверного середовища, на якому запущена мова сценаріїв, наприклад ASP, JSP, PHP, Python, Perl або Ruby, і тому не обов'язково підтримувати самого розробника. Однак рекомендується жорстке тестування веб-додатків перед публічним випуском, щоб запобігти подібним подвигам. Якщо на веб-сайті є якась контактна форма, вона повинна включати в неї поле captcha, яке не дозволяє комп’ютерним програмам автоматично заповнювати форми, а також розсилати пошту.

Захист веб-сервера від вторгнень часто називають зміцненням порту сервера. Багато технологій застосовуються для захисту інформації в Інтернеті, коли вона передається з одного місця в інше.

Наприклад, сертифікати TLS (або "сертифікати SSL") видаються органами сертифікації для запобігання шахрайству в Інтернеті.

Багато розробників часто використовують різні форми шифрування при передачі та зберіганні конфіденційної інформації. Базове розуміння питань безпеки інформаційних технологій часто є частиною знань веб-розробника.

Оскільки нові веб-програми виявляють нові діри в безпеці навіть після тестування та запуску, оновлення виправлень безпеки часто трапляються для широко використовуваних програм.

2.2. Модель розробки web-системи оцінювання і перевірки знань студентів

Класифікація тестів

тести можна класифікувати за різними ознаками:

- за програмними цілями - інформаційні, діагностичні, навчальні, мотиваційні, атестаційні;
- за процедурою створення - стандартизовані, що не стандартизовані;
- за способом формування завдань - детерміновані, стохастичні, динамічні;
- за технологією проведення - паперові, в тому числі паперові з використанням оптичного розпізнавання, натурні, з використанням спеціальної апаратури, комп'ютерні;
- за формою завдань - закритого типу, відкритого типу, встановлення відповідності, упорядкування послідовності;
- по наявності зворотного зв'язку - традиційні і адаптивні

Традиційний тест

Традиційний тест містить список питань і різні варіанти відповідей. Кожне питання оцінюється в певну кількість балів. Результат традиційного тесту залежить від кількості питань, на які було дано правильну відповідь. На думку Аванесова В. С. традиційний тест - система завдань, що пред'являється в порядку збільшення складності в один і той же час, з однаковою системою оцінювання для всіх тестованих .

Адаптивний тест

Особливий вид тесту, в якому кожне наступне завдання вибирається залежно від відповідей на попередні завдання. Послідовність завдань і їх

кількість в такому вигляді тесту визначається динамічно. Самими значущими перевагами комп'ютерного адаптивного тестування перед традиційним є:

- можливість адаптації під рівень знань тестованого (не доведеться відповідати на занадто складні або занадто прості питання);
- економія часу і сил за рахунок скорочення кількості завдань (довжина тесту може бути зменшена до 60%) без втрати рівня достовірності.

Розроблювана веб-система повинна підтримувати основні методики створення тестів і їх основні види, саме це є основною частиною моделі створення веб-системи перевірки і оцінювання знань студента.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОВЕДЕННЯ ОНЛАЙНОВОГО АНКЕТУВАННЯ

3.1. Інформаційно-логічна модель проведення онлайнового анкетування

Діаграма варіантів використання

Діаграма використання найпростіша - це представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів використання може допомогти забезпечити огляд системи на більш високому рівні. Раніше вже було сказано, що "схеми використання - це принципи вашої системи".

Через їх спрощений характер, схеми використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно, ніж діаграми класів".

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для

забезпечення повного функціонального та технічного уявлення про систему. Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Елементи:

- рамки системи (англ. system border) - прямокутник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про корисну інформацію,
- актор (англ. актор) - стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, зовнішня сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),
- прецедент - еліпс із надписом, що обозначує виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім’я прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюструючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

На рисунках 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

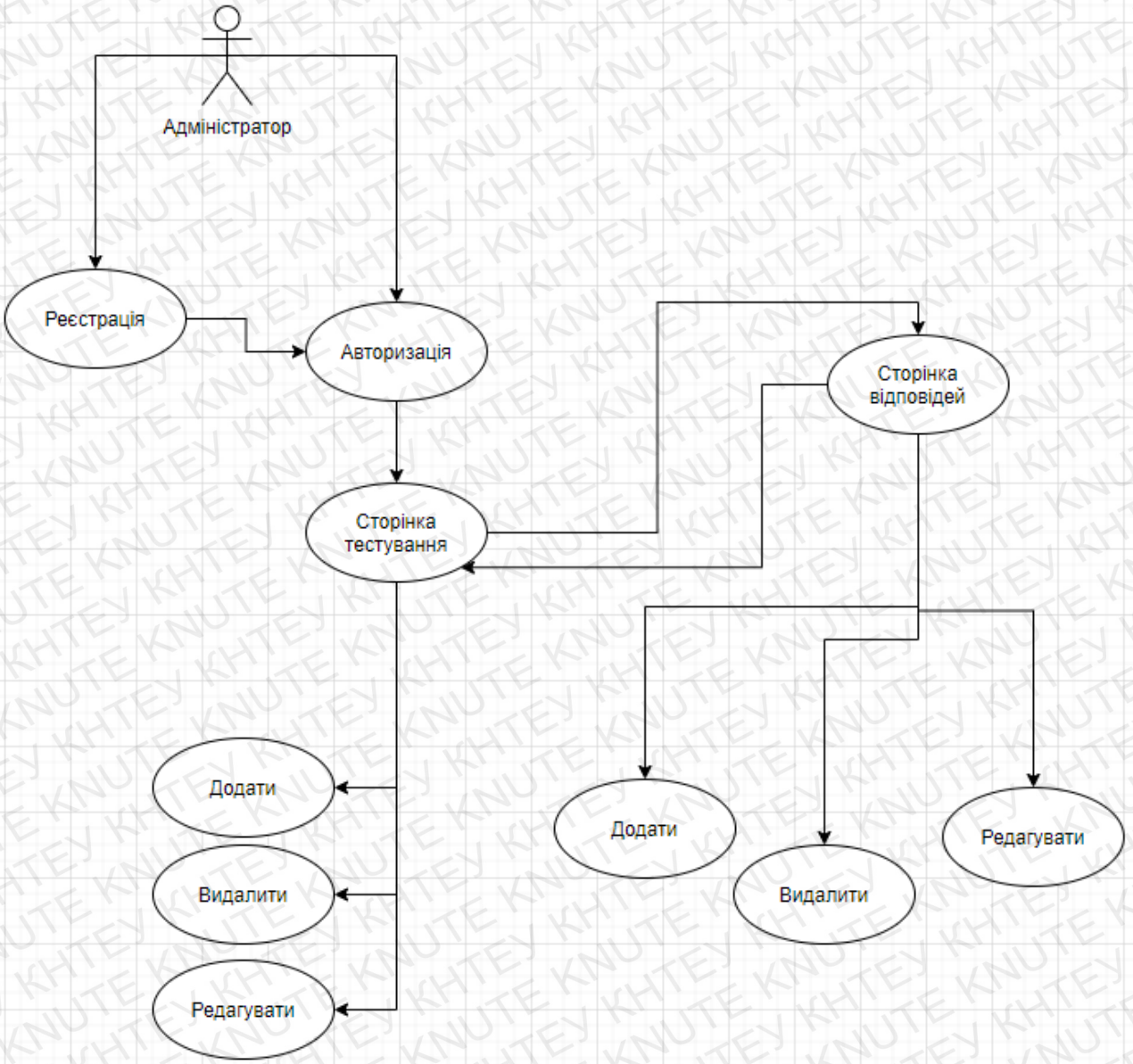

Рис. 3.1 — Діаграма варіантів використання для користувача

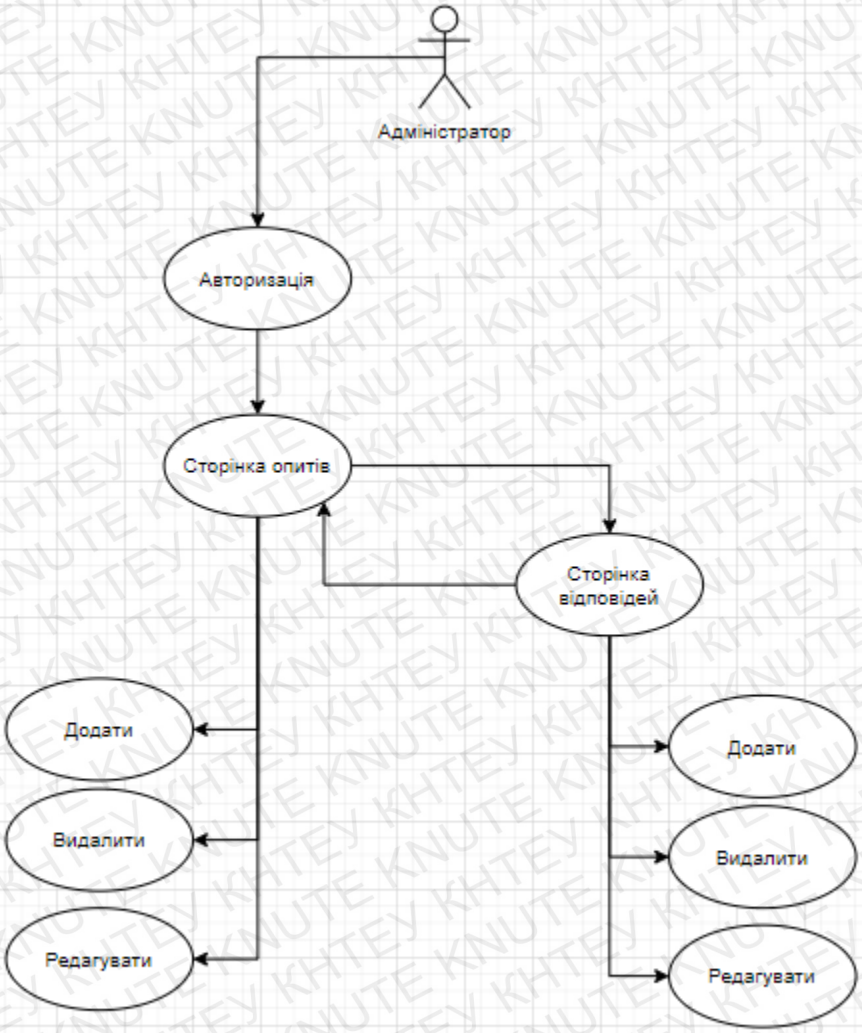

Рис. 3.2 — Діаграма варіантів використання для користувача

Діаграма класів

У програмній інженерії діаграма класів в Уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Діаграма класів є основним будівельним елементом об’єктно-орієнтованого моделювання. Він використовується для загального концептуального моделювання структури програми та для детального моделювання переведення моделей у програмовий код. Діаграми класів також можуть бути використані для моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в програмі, так і класи, що програмуються.

На схемі класи представлені вікнами, які містять три відділення:

- У верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами.
- Середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала.
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква - мала.

При проектуванні системи ряд класів ідентифікується та згруповується у схему класів, яка допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проекту часто поділяються на ряд підкласів.

Залежність - це семантичний зв’язок між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміни у визначенні одного елемента (сервера або цілі) можуть спричинити зміни для іншого (клієнта або джерела). Ця асоціація є односпрямованою. Залежність відображається у вигляді штрихової лінії з відкритою стрілкою, яка вказує від клієнта до постачальника.

Для подальшого опису поведінки систем ці діаграми класів можуть бути доповнені діаграмою стану або машиною стану UML.

Асоціація представляє родину посилань. Двійкова асоціація (з двома кінцями) зазвичай представляється у вигляді рядка. Асоціація може пов’язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною асоціацією. Асоціацію можна назвати, а кінці асоціації можна прикрасити іменами ролей, показниками власності, кратністю, видимістю та іншими властивостями.

Існує чотири різні типи асоціацій: двонаправлена, односпрямована, агрегаційна (включає агрегацію композиції) та рефлексивна. Двонаправлені та односпрямовані асоціації є найбільш поширеними.

Наприклад, клас польоту асоціюється з класом літака двонаправлено. Асоціація представляє статичне відношення, яке ділиться між об'єктами двох класів.

Агрегація є варіантом взаємозв'язку "має"; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частково цілі або часткові стосунки. Як показано на зображенні, професор "має" клас для викладання. Як тип асоціації, агрегація може бути названа та мати ті самі прикраси, що і асоціація. Однак агрегація не може включати більше двох класів; це має бути бінарна асоціація. Крім того, навряд чи існує різниця між агрегаціями та асоціаціями під час реалізації, і діаграма може взагалі пропустити відносини агрегування.

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але вміщені класи не мають сильної залежності життєвого циклу від контейнера. Вміст контейнера все ще існує, коли контейнер знищений.

В UML він графічно представлений у вигляді порожнистої форми ромба на вміщуючому класі одним рядком, що зв’язує його із вміщеним класом. Сукупність - це семантично розширений об’єкт, який у багатьох операціях трактується як одиниця, хоча фізично він складається з декількох менших об’єктів.

Приклад: Бібліотека та студенти. Тут студент може існувати без бібліотеки, зв'язок між студентом і бібліотекою є агрегацією.

Це вказує на те, що один із двох пов’язаних класів (підклас) вважається спеціалізованою формою іншого (супер тип), а суперклас - узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу є також екземпляром суперкласу. Зразкове дерево узагальнень цієї форми зустрічається в біологічній класифікації: людина - це підклас маймуни, який є підкласом ссавців тощо. Зв’язок найлегше зрозуміти за допомогою фрази „A - це B” (людина - це ссавець, ссавець - тварина).

Графічне представлення UML узагальнення - це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадщина або відносини "є".

Суперклас (базовий клас) у відносинах узагальнення також відомий як "батьківський", суперклас, базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

Зверніть увагу, що ці стосунки нічим не схожі на біологічні стосунки батьків та дітей: використання цих термінів надзвичайно поширене, але може ввести в оману.

A - це тип B

Наприклад, "дуб - це тип дерева", "автомобіль - це тип транспортного засобу"

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації - це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує

(реалізує або виконує) поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації - це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація - це взаємозв'язок між класами, інтерфейсами, компонентами та пакетами, що з'єднує елемент клієнта з елементом постачальника. Зв'язок реалізації між класами / компонентами та інтерфейсами показує, що клас / компонент реалізує операції, пропоновані інтерфейсом.

Залежність - це слабша форма зв'язку, яка вказує на те, що один клас залежить від іншого, оскільки він використовує його в певний момент часу. Один клас залежить від іншого, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між двома класами дуже слабкі. Вони взагалі не реалізовані зі змінними-членами. Швидше вони можуть бути реалізовані як аргументи функції-члена.

інший. Ці відносини зазвичай описуються як "А має В" (у матері-кота є кошенята, у кошенят - мати-кішка).

Представлення UML асоціації - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові позначення. Наприклад, ми можемо вказати, використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку відіграють елементи цього кінця, вказавши ім'я ролі та множинність

екземплярів цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутності моделюють довгоживучу інформацію, якою обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Як варіант, їх можна намалювати як звичайні класи із позначенням стереотипу «сутність» над назвою класу.

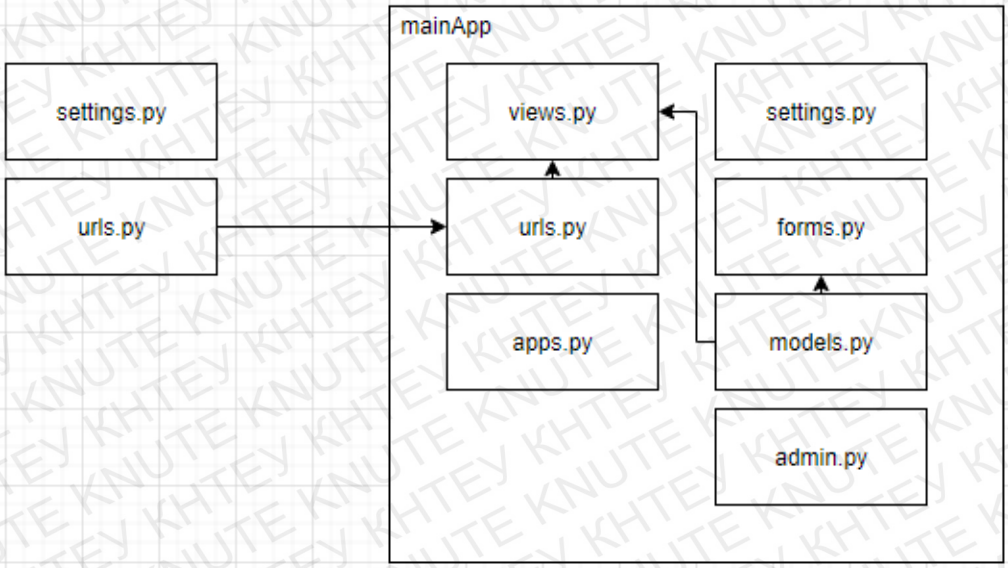

Рис. 3.2 — Діаграма класів

3.2. Програмна реалізація проведення онлайнового анкетування

Мова програмування Python

Python - інтерпретована мова програмування високого рівня та загального призначення. Філософія дизайну Python наголошує на читабельності коду завдяки помітному використанню значних відступів. Його мовні конструкції та об’єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та великих проектів.

Python динамічно набирається та збирається сміття. Він підтримує декілька парадигм програмування, включаючи структуроване (зокрема, процедурне), об'єктно-орієнтоване та функціональне програмування. Python часто описують як мову, що включає батареї, завдяки своїй повній стандартній бібліотеці.

Гідо ван Россум почав працювати над Python наприкінці 1980-х років, як наступник мови програмування ABC, і вперше випустив її в 1991 році під назвою Python 0.9.0. Python 2.0 був випущений в 2000 році і представив нові функції, такі як розуміння списків та система збору сміття з використанням підрахунку посилань, і був припинений з версією 2.7.18 у 2020 році. Python 3.0 був випущений в 2008 році і був серйозним переглядом мови, яка не є повністю сумісною із зворотною суттю, і більша частина коду Python 2 не працює незміненою на Python 3.

Python стабільно входить до числа найпопулярніших мов програмування.

Python був задуманий наприкінці 1980-х Гвідо ван Россумом з Centrum Wiskunde & Informatica (CWI) в Нідерландах як спадкоємець мови програмування ABC, натхненної SETL, здатною обробляти винятки та взаємодіяти з Операційна система Amoeba. Його реалізація розпочалась у грудні 1989 р. Ван Россум взяв на себе виключну відповідальність за проект, як провідний розробник, до 12 липня 2018 року, коли він оголосив про свої "постійні канікули", виконуючи обов'язки як "Доброзичливий диктатор за

життя" Пітона, титул, який йому надала спільнота Пітона, щоб відобразити його тривалий час. строкове зобов'язання як головний керівник проекту. Зараз він ділиться своїм керівництвом як член наглядової ради з п'яти осіб. У січні 2019 року активні розробники ядра Python обрали Бретта Кеннона, Ніка Коглана, Барі Варшаву, Керол Віллінг та Ван Россума членами керівної ради з п’яти членів. З тих пір Гвідо ван Россум відкликав свою кандидатуру на посаду Керівної ради 2020 року.

Python 2.0 був випущений 16 жовтня 2000 року з багатьма основними новими можливостями, включаючи збирач сміття, що виявляє цикл, та підтримку Unicode.

Python 3.0 був випущений 3 грудня 2008 року. Це був серйозний перегляд мови, який не є повністю сумісним із зворотною стороною. Багато його основних функцій були передані до версій Python 2.6.x та 2.7.x. Випуски Python 3 включають утиліту 2to3, яка автоматизує (принаймні частково) переклад коду Python 2 на Python 3.

Дата закінчення терміну служби Python 2.7 спочатку була встановлена на 2015 рік, а потім перенесена на 2020 рік через занепокоєння тим, що велика кількість існуючого коду не може бути легко перенесена на Python 3. Більше виправлень безпеки та інших удосконалень для нього випускати не буде. У кінці життя Python 2 підтримується лише Python 3.6.x і пізніші версії.

Python 3.9.2 та 3.8.8 були прискорені, оскільки всі версії Python мали проблеми із безпекою, що призвело до можливого віддаленого виконання коду та отруєння веб-кешу.

Python - мова програмування з багатьма парадигмами. Об'єктно-орієнтоване програмування та структуроване програмування повністю підтримуються, і багато його функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (в тому числі метапрограмуванням та метаоб'єктами (магічні методи)). Багато інших

парадигм підтримуються за допомогою розширень, включаючи дизайн за контрактом та логічне програмування.

Python використовує динамічне введення тексту та комбінацію підрахунку посилань та збирач сміття, що визначає цикл, для управління пам’яттю. Він також має динамічну роздільну здатність імен (пізнє прив'язування), яка зв'язує імена методів та змінних під час виконання програми.

Дизайн Python пропонує певну підтримку функціонального програмування за традицією Lisp. Він має функції фільтрування, картографування та зменшення; перелічити розуміння, словники, набори та вирази генераторів. Стандартна бібліотека має два модулі (itertools та functools), що реалізують функціональні інструменти, запозичені у Haskell та Standard ML.

Основна філософія мови узагальнена в документі Zen of Python (PEP 20), який включає такі афоризми, як:

- Красиве - краще, ніж потворне.
- Явне краще, ніж неявне.
- Просте - краще, ніж складне.
- Складний - це краще, ніж складний.
- Підрахунок читабельності.

Замість того, щоб усі його функціональні можливості були вбудовані в його ядро, Python був розроблений таким чином, щоб бути дуже розширюваним (з модулями). Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих додатків. Бачення Ван Россума щодо малої основної мови з великою стандартною бібліотекою та легко розширюваним перекладачем випливало з його розчарувань у ABC, який підтримував протилежний підхід. Python прагне до більш простого, менш захащеного синтаксису та граматики, одночасно надаючи розробникам можливість вибору методології кодування. На відміну від девізу Perl "існує

більше ніж один спосіб зробити це", Python охоплює "повинен бути один - і бажано лише один - очевидний спосіб зробити це" філософія дизайну. Алекс Мартеллі, співробітник Фонду програмного забезпечення Python і автор книги про Python, пише, що "Описувати щось як" розумне "не вважається компліментом у культурі Python" .

Розробники Python прагнуть уникнути передчасної оптимізації та відкидають виправлення для некритичних частин еталонної реалізації CPython, які пропонують незначне збільшення швидкості ціною ясності. Коли швидкість важлива, програміст Python може переміщати критично важливі для часу функції до модулів розширення, написаних мовами, такими як C, або використовувати PyPy, своєчасний компілятор. Також доступний Cython, який перекладає скрипт Python на C і здійснює прямі виклики API рівня C в інтерпретатор Python.

Важливою метою розробників Python є підтримка його задоволення від використання. Це відображається в назві мови - данина поваги британській гумористичній групі Монті Пайтон - і в іноді грайливих підходах до навчальних посібників та довідкових матеріалів, таких як приклади, що стосуються спаму та яєць (із відомого етюду Монті Пайтона) стандартних foo and bar.

Поширеним неологізмом у спільноті Python є пітонічний, який може мати широкий діапазон значень, пов’язаних зі стилем програми. Сказати, що код пітонічний, означає сказати, що він добре використовує ідіоми Python, що він природний або демонструє вільну мову, що відповідає мінімалістичній філософії Python та наголосу на читабельності. На відміну від цього, код, який важко зрозуміти або читається як груба транскрипція з іншої мови програмування, називається непітонічним.

Користувачів та шанувальників Python, особливо тих, хто вважається обізнаним чи досвідченим, часто називають Pythonistas. Python має бути легкочитабельною мовою. Його форматування візуально не захаращене, і в

ньому часто використовуються англійські ключові слова, де інші мови використовують розділові знаки. На відміну від багатьох інших мов, він не використовує фігурні дужки для розмежування блоків, а крапка з комою після операторів дозволена, але рідко, якщо взагалі використовується. Він має менше синтаксичних винятків та особливих випадків, ніж C або Pascal.

Python використовує пробіли, а не фігурні дужки або ключові слова, для відмежування блоків. Збільшення відступу відбувається після певних тверджень; зменшення відступу означає кінець поточного блоку. Таким чином, візуальна структура програми точно відображає семантичну структуру програми. Цю функцію іноді називають правилом "поза стороною", яке мають деякі інші мови, але в більшості мов відступ не має семантичного значення. Рекомендований розмір відступу - чотири пробіли.

Оператори Python включають (серед іншого):

- Оператор присвоєння, використовуючи один знак рівності =.
- Оператор if, який умовно виконує блок коду, разом з else та elif (скорочення else-if).
- Оператор for, який переглядає ітерабельний об’єкт, захоплюючи кожен елемент до локальної змінної для використання вкладеним блоком.
- Оператор while, який виконує блок коду, доки його умова відповідає істині.
- Оператор try, що дозволяє вилученням, що містяться у вкладеному блоці коду, ловити та обробляти за винятком речень; він також гарантує, що код очищення в блоці нарешті завжди буде виконуватися незалежно від того, як блок виходить.

- Оператор raise, що використовується для отримання вказаного винятку або повторного підняття спійманого винятку.
- Оператор class, який виконує блок коду та приєднує його локальний простір імен до класу для використання в об'єктно-орієнтованому програмуванні.
- Оператор def, що визначає функцію або метод.
- Оператор with з Python 2.5, випущений у вересні 2006 р. [82], який охоплює блок коду в контекстному менеджері (наприклад, отримання блокування перед запуском блоку коду та звільнення блокування після цього, або відкриття файлу, а потім закриваючи його), дозволяючи поведінку, схожу на отримання ресурсів - це ініціалізація (RAII), і замінює загальну ідіому try / нарешті.
- Оператор break, виходить із циклу.
- Оператор continue, пропускає цю ітерацію і продовжує наступний пункт.
- Оператор del видаляє змінну, що означає, що посилання з імені на значення видаляється, і спроба використання цієї змінної спричинить помилку. Видалену змінну можна перепризначити.
- Оператор pass, яка виконує функцію NOP. Це синтаксично потрібно для створення порожнього блоку коду.
- Оператор assert, який використовується під час налагодження для перевірки умов, які мають застосовуватися.
- Оператор yield, який повертає значення з функції генератора. З Python 2.5, yield також є оператором. Ця форма використовується для реалізації спільних програм.
- Оператор return, який використовується для повернення значення з функції.

- Оператор import, який використовується для імпорту модулів, функції чи змінні яких можна використовувати в поточній програмі. Є три способи використання імпорту: імпорт <ім'я модуля> [як <псевдонім>] або з <ім'я модуля> імпорт * або з <ім'я модуля> імпорт <визначення 1> [як <псевдонім 1>], <визначення 2> [як <псевдонім 2>], ...

Оператор присвоєння (=) діє шляхом прив'язки імені як посилання на окремий динамічно розподілений об'єкт. Потім змінні можуть бути відскочені в будь-який час до будь-якого об’єкта. У Python ім'я змінної є загальним власником посилання і не має фіксованого типу даних, пов’язаного з ним. Однак у даний момент часу змінна буде посилатися на якийсь об'єкт, який матиме тип. Це називається динамічним набором тексту і протиставляється статично набраним мовам програмування, де кожна змінна може містити лише значення певного типу.

Python не підтримує оптимізацію хвостових викликів або першокласні продовження, і, за словами Гвідо ван Россума, він ніколи не буде. Однак краща підтримка подібних до корутинів функціональних можливостей надається в 2.5, розширюючи генератори Python. До 2.5 генератори були ледачими ітераторами; інформація передавалась в односпрямованому напрямку з генератора. З Python 2.5 можна передавати інформацію назад у функцію генератора, а з Python 3.3 інформацію можна передавати через кілька рівнів стека.

Деякі вирази Python подібні до таких, що зустрічаються в таких мовах, як C та Java, а деякі - ні:

- Додавання, віднімання та множення однакові, але поведінка ділення відрізняється. У Python існує два типи поділів. Вони є розподілом підлоги (або цілочисельним поділом) // та плаваючою комою / поділом. Python також використовує оператор ** для піднесення до степені.

- З Python 3.5 був представлений новий оператор @ infix. Він призначений для використання бібліотеками, такими як NumPy, для множення матриць.
- З Python 3.8 був введений синтаксис: =, який називається «моржовим оператором». Він присвоює значення змінним як частину більшого виразу.
- У Python == порівнює за значенням проти Java, яка порівнює числові значення за значенням, а об'єкти за посиланням. (Порівняння значень в Java на об'єктах можна виконувати методом equals ().) Оператор Python is is може використовуватися для порівняння ідентифікацій об'єктів (порівняння за посиланням). У Python порівняння можуть бути ланцюговими, наприклад a <= b <= c.
- Python використовує слова та, або, не для своїх булевих операторів, а не для символічного &&, ||,! використовується в Java та C.
- Python має тип виразу, що називається розумінням списку, а також більш загальний вираз, який називається генераторським виразом.
- Анонімні функції реалізовані за допомогою лямбда-виразів; однак вони обмежені тим, що тіло може бути лише одним виразом.
- Умовні вирази в Python пишуться як x, якщо c else y (відрізняється за порядком операндів від оператора c? X: y, загального для багатьох інших мов).
- Python розрізняє списки та кортежі. Списки записуються як [1, 2, 3], змінюються та не можуть використовуватися як ключі словників (ключі словника повинні бути незмінними в Python). Кортежі записуються як (1, 2, 3), є незмінними і, отже, можуть використовуватися як ключі словників, за умови, що всі елементи кортежу є незмінними. Оператор + може бути використаний для

об'єднання двох кортежів, що безпосередньо не змінює їх вміст, а створює новий кортеж, що містить елементи обох передбачених кортежів. Таким чином, враховуючи змінну t, спочатку рівну (1, 2, 3), при виконанні t = t + (4, 5) спочатку обчислюється t + (4, 5), що дає (1, 2, 3, 4, 5) , який потім призначається назад до t, тим самим ефективно "модифікуючи вміст" t, одночасно відповідаючи незмінній природі об'єктів кортежу. Дужки не є обов’язковими для кортежів у однозначному контексті.

- Python має розпаковування послідовностей, при якому кілька виразів, кожен з яких обчислює будь-що, до чого можна присвоїти (змінну, властивість для запису тощо), пов’язані ідентично тому, що формує кортежні літерали, і, як ціле, розміщуються на ліва сторона знака рівності у твердженні про присвоєння. Оператор очікує ітерабельного об'єкта праворуч від знака рівності, який видає таку ж кількість значень, що і надані вираз для запису, коли його перебирають і буде перебирати його, присвоюючи кожне із створених значень відповідному виразу зліва .
- Python має оператор "формат рядка"%. Це функціонує аналогічно рядкам формату printf на C, наприклад "спам =% s яєць =% d"% ("бла", 2) оцінюється як "спам = бла-яйця = 2". У Python 3 та 2.6+ це було доповнено методом format () класу str, наприклад "спам = {0} яйця = {1}". формат ("бла", 2). Python 3.6 додав "f-рядки": blah = "blah"; яйця = 2; f'spam = {blah} яйця = {яйця} '.
- Рядки в Python можна об'єднати, "додавши" їх (той самий оператор, що і для додавання цілих чи значень з плаваючою точкою). Наприклад "спам" + "яйця" повертає "спамегги". Навіть якщо ваші рядки містять числа, вони все одно додаються як рядки, а не цілі числа. Наприклад "2" + "2" повертає "22".

- Python має різні типи рядкових літералів:
- Рядки, розділені одинарними або подвійними лапками. На відміну від оболонок Unix, мови Perl та Perl, що впливають на Perl, одинарні лапки та подвійні лапки функціонують однаково. Обидва типи рядків використовують зворотну скісну риску (\) як символ виходу. Інтерполяція рядків стала доступною в Python 3.6 як "відформатовані рядкові літерали".
    - Рядки з подвійними лапками, які починаються та закінчуються серією з трьох одинарних або подвійних лапок. Вони можуть охоплювати кілька рядків і функціонувати, як тут документи в оболонках, Perl і Ruby.
    - Сирі різновиди рядків, що позначаються префіксом рядкового літералу перед r. Послідовності втечі не інтерпретуються; отже, необроблені рядки корисні там, де буквенні зворотні слеші є загальними, такі як регулярні вирази та шляхи у стилі Windows. Порівняйте "@ -citation" у C #.
- Python має індекси масивів та вирази нарізування масивів у списках, що позначаються як [ключ], [старт: зупинка] або [старт: зупинка: крок]. Індекси базуються на нулі, а негативні індекси відносно кінця. Зрізи беруть елементи від початкового індексу до, але не включаючи, індексу зупинки. Третій параметр зрізу, який називається кроком або кроком, дозволяє пропускати та обертати елементи. Індекси зрізів можуть бути опущені, наприклад, [:] повертає копію всього списку. Кожен елемент зрізу є неглибокою копією.

У Python чітко дотримується різниці між виразами та висловлюваннями, на відміну від таких мов, як Common Lisp, Scheme або Ruby. Це призводить до дублювання деяких функціональних можливостей.

Методи на об'єкти - це функції, приєднані до класу об'єкта; синтаксис instance.method (аргумент) - це для звичайних методів та функцій синтаксичний цукор для Class.method (екземпляр, аргумент). Методи Python мають явний параметр self для доступу до даних екземпляра, на відміну від неявного self (або цього) в деяких інших об’єктно-орієнтованих мовах програмування (наприклад, C ++, Java, Objective-C або Ruby).

Python використовує набір качок і має набрані об'єкти, але нетипізовані імена змінних. Обмеження типу не перевіряються під час компіляції; швидше, операції з об'єктом можуть зазнати невдачі, що означає, що даний об'єкт не є відповідним типом. Не дивлячись на те, що Python набирається динамічно, забороняє операції, які не є чітко визначеними (наприклад, додавання числа до рядка), а не намагається їх тихо зрозуміти.

Python дозволяє програмістам визначати власні типи за допомогою класів, які найчастіше використовуються для об'єктно-орієнтованого програмування. Нові екземпляри класів будуються за допомогою виклику класу (наприклад, SpamClass () або EggsClass ()), а класи є екземплярами типу метакласу (сам екземпляр самого себе), що дозволяє метапрограмувати та відображати.

До версії 3.0 Python мав два типи класів: старий і новий. Синтаксис обох стилів однаковий, різниця полягає в тому, чи об’єкт класу успадковується, прямо чи опосередковано (усі класи нового стилю успадковуються від об’єкта і є екземплярами типу). У версіях Python 2, починаючи з Python 2.2, можна використовувати обидва типи класів. Класи старого стилю були ліквідовані в Python 3.0.

Довгостроковий план полягає в підтримці поступового набору тексту, а з Python 3.5 синтаксис мови дозволяє вказувати статичні типи, але вони не перевіряються в реалізації за замовчуванням, CPython. Експериментальна додаткова перевірка статичного типу з ім’ям муру підтримує перевірку типу під час компіляції.

CPython - це посилальна реалізація Python. Він написаний на мові C, відповідаючи стандарту C89 з декількома вибраними функціями C99 (з випуском пізніших версій C він вважається застарілим; CPython включає власні розширення C, але розширення сторонніх розробників не обмежуються старими C версії, наприклад, можуть бути реалізовані з C11 або C ++). Він компілює програми Python у проміжний байт-код, який потім виконується його віртуальною машиною. CPython поширюється з великою стандартною бібліотекою, написаною на суміші C та рідного Python. Він доступний для багатьох платформ, включаючи Windows (починаючи з Python 3.9, інсталятор Python навмисно не вдається встановити в Windows 7 і 8; Windows XP підтримувався до Python 3.5) та більшості сучасних Unix-подібних систем, включаючи macOS (та Apple M1 Macs, починаючи з Python 3.9.1, з експериментальним інсталятором) та неофіційну підтримку, наприклад VMS. Переносимість платформи була одним із її перших пріоритетів, протягом періодів Python 1 та 2 підтримувались навіть OS / 2 та Solaris; підтримка відтоді відмовилася для багатьох платформ.

Розробка Python здійснюється в основному за допомогою процесу Програми вдосконалення Python (PEP), основного механізму пропонування основних нових можливостей, збору вкладу спільноти з питань та документування рішень щодо проектування Python. Стиль кодування Python викладений у PEP 8. Видатні PEP переглядаються та коментуються спільнотою Python та керівною радою.

Покращення мови відповідає розробці посилальної реалізації CPython. Список розсилки python-dev є основним форумом для розвитку мови. Конкретні проблеми обговорюються у програмі відстеження помилок Roundup, розміщеній на bugs.python.org. Спочатку розробка здійснювалась у власному сховищі вихідного коду під управлінням Mercurial, поки Python не перейшов на GitHub у січні 2017 року.

Публічні випуски CPython бувають трьох типів, що відрізняються тим, яка частина номера версії збільшується:

Несумісні із зворотною мовою версії, де, як очікується, зламається код і його потрібно перенести вручну. Перша частина номера версії збільшується. Ці випуски трапляються рідко - версія 3.0 вийшла через 8 років після 2.0.

Основні або "функціональні" випуски відбувалися приблизно кожні 18 місяців, але з прийняттям щорічної каденції випуску, починаючи з Python 3.9, очікується, що це відбуватиметься раз на рік. Вони значною мірою сумісні, але вводять нові функції. Друга частина номера версії збільшується. Кожна основна версія підтримується виправленнями протягом декількох років після її випуску.

Випуски виправлень, які не вводять нових функцій, трапляються приблизно кожні 3 місяці і робляться, коли з останнього випуску виправлена достатня кількість помилок. Уразливості безпеки також виправлені у цих випусках. Третя і остання частина номера версії збільшується.

Багато альфа-, бета-версій та кандидатів на випуск також випускаються у вигляді попереднього перегляду та тестування перед остаточними випусками. Хоча існує приблизний графік кожного випуску, вони часто затримуються, якщо код не готовий. Команда розробників Python контролює стан коду, запускаючи великий пакет модульних тестів під час розробки.

Основною академічною конференцією з Python є PyCon. Існують також спеціальні програми наставництва Python, такі як Pyladies.

Pythons 3.10 припиняє wstr (буде видалено в Python 3.12; це означає, що розширення Python потрібно буде змінити до того часу), а також планує додати відповідність шаблону до мови.

Використання

З 2003 р. Python стабільно входить до десятки найпопулярніших мов програмування в Індексі програмного забезпечення TIOBE, де, станом на лютий 2021 р., Це третя за популярністю мова (після Java та C). Він був обраний

мовою програмування року (за "найвищий ріст рейтингу за рік") у 2007, 2010, 2018 та 2020 роках (єдина мова, яка зробила це чотири рази).

Емпіричне дослідження показало, що мови сценаріїв, такі як Python, є більш продуктивними, ніж звичайні мови, такі як C та Java, для проблем програмування, що включають маніпулювання рядками та пошук у словнику, і встановило, що споживання пам'яті часто "краще, ніж Java, а не набагато гірше, ніж C або C ++ ".

До великих організацій, які використовують Python, належать Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify та деякі менші організації, такі як ILM та ITA. Сайт соціальних новин Reddit був написаний переважно на Python.

Python може служити мовою сценаріїв для веб-додатків, наприклад, через mod_wsgi для веб-сервера Apache. Завдяки інтерфейсу шлюзу веб-сервера для полегшення роботи цих програм розроблено стандартний API. Веб-фреймворки, такі як Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle і Zope, підтримують розробників у розробці та обслуговуванні складних додатків. Pyjs та IronPython можуть бути використані для розробки клієнтської частини програм на базі Ajax. SQLAlchemy може використовуватися як перетворювач даних у реляційну базу даних. Twisted - це фреймворк для програмування зв’язку між комп’ютерами, який використовується (наприклад) Dropbox.

Такі бібліотеки, як NumPy, SciPy та Matplotlib, дозволяють ефективно використовувати Python у наукових обчисленнях, а спеціалізовані бібліотеки, такі як Biopython та Astropy, забезпечують функціональність, що залежить від домену. SageMath - математичне програмне забезпечення з інтерфейсом ноутбука, програмоване на Python: його бібліотека охоплює багато аспектів математики, включаючи алгебру, комбінаторику, числову математику, теорію

чисел і числення. OpenCV має палітурні прив'язки з багатим набором функцій для комп'ютерного зору та обробки зображень.

Python зазвичай використовується в проектах штучного інтелекту та машинному навчанні за допомогою таких бібліотек, як TensorFlow, Keras, Pytorch та Scikit-learn. Як мова сценаріїв з модульною архітектурою, простим синтаксисом та інструментами обробки багатого тексту, Python часто використовується для обробки природної мови. Python успішно вбудований у багато програмних продуктів як мову сценаріїв, включаючи програмне забезпечення методом скінченних елементів, таке як Abaqus, 3D-параметричний моделер, такий як FreeCAD, пакети 3D-анімації, такі як 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo , MotionBuilder, Softimage, композитор візуальних ефектів Nuke, програми для 2D-зображень, такі як GIMP, Inkscape, Scribus і Paint Shop Pro, та музичні нотатні програми, такі як автори та капели. Налагоджувач GNU використовує Python як гарний принтер для показу складних структур, таких як контейнери C ++. Esri пропагує Python як найкращий вибір для написання сценаріїв в ArcGIS. Він також використовувався в декількох відеоіграх і був прийнятий як перша з трьох доступних мов програмування в Google App Engine, інші дві - Java і Go.

Багато операційних систем включають Python як стандартний компонент. Він постачається з більшістю дистрибутивів Linux, AmigaOS 4 (з використанням Python 2.7), FreeBSD (як пакет), NetBSD, OpenBSD (як пакет) та macOS і може використовуватися з командного рядка (терміналу). Багато дистрибутивів Linux використовують інсталятори, написані на Python: Ubuntu використовує інсталятор Ubiquity, тоді як Red Hat Linux і Fedora використовують інсталятор Anaconda. Gentoo Linux використовує Python у своїй системі управління пакунками Portage.

Python широко використовується в галузі інформаційної безпеки, в тому числі в розробці експлойтів.

Більшість програм Sugar для одного ноутбука на дитину XO, які зараз розробляються в Sugar Labs, написані на Python. Проект одноплатної комп'ютерної програми Raspberry Pi прийняв Python як основну мову програмування користувачів.

LibreOffice включає Python і має намір замінити Java на Python. Його постачальник сценаріїв Python є основною функцією з версії 4.0 від 7 лютого 2013 року.

Вихоядчи зі сфери використання і величезного функціоналу мови Python, який був описаний вище, можна зробити висновок про його мультифунціональність. Саме через це дана мова програмування була обрана для виконання даної роботи.

Фреймворк Django

Django - це безкоштовний веб-фреймворк на основі Python, що відповідає архітектурному зразку model-template-views (MTV). Він підтримується Django Software Foundation (DSF), американською незалежною організацією, створеною як некомерційна організація 501 (c) (3).

Основна мета Django - полегшити створення складних веб-сайтів, керованих базами даних. Структура підкреслює багаторазовість та "підключеність" компонентів, менше коду, низьку зв'язок, швидкий розвиток та принцип "не повторюватися". Python використовується всюди, навіть для налаштувань, файлів та моделей даних. Django також надає додатковий адміністративний інтерфейс створення, читання, оновлення та видалення, який генерується динамічно за допомогою самоаналізу та налаштовується за допомогою моделей адміністратора.

Деякі добре відомі сайти, які використовують Django, включають PBS, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, і Nextdoor.

Розробка графічного інтерфейсу користувача

Графічний інтерфейс користувача складається з 4 основних сторінок, а саме:

- Сторінка авторизації (рис. 3.4)
- Сторінка реєстрації (рис. 3.5)
- Сторінка активних тестувань (рис. 3.6)
- Сторінка відповідей(рис. 3.7)

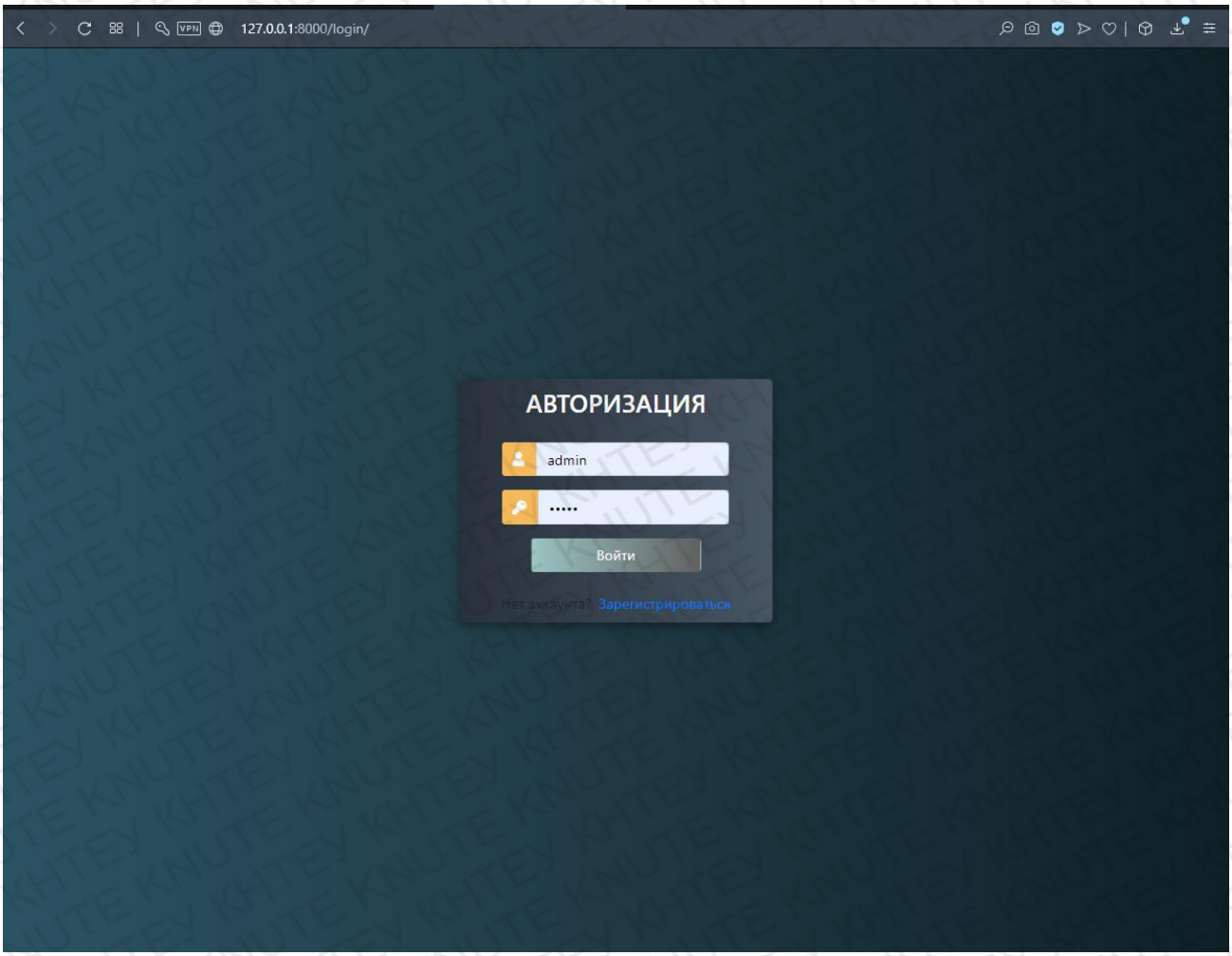

Рис. 3.4 — Сторінка авторизації

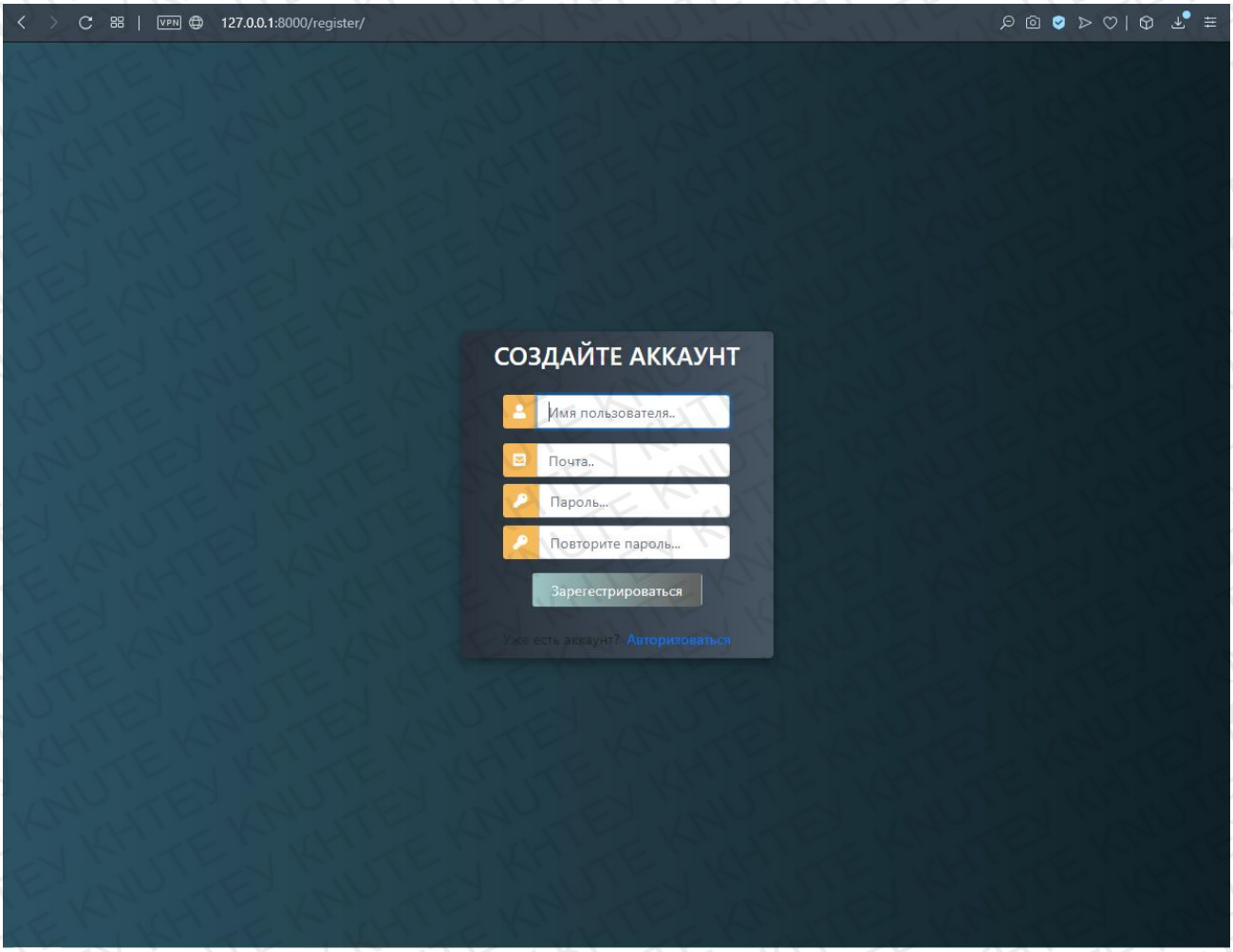

Рис. 3.5 — Сторінка реєстрації

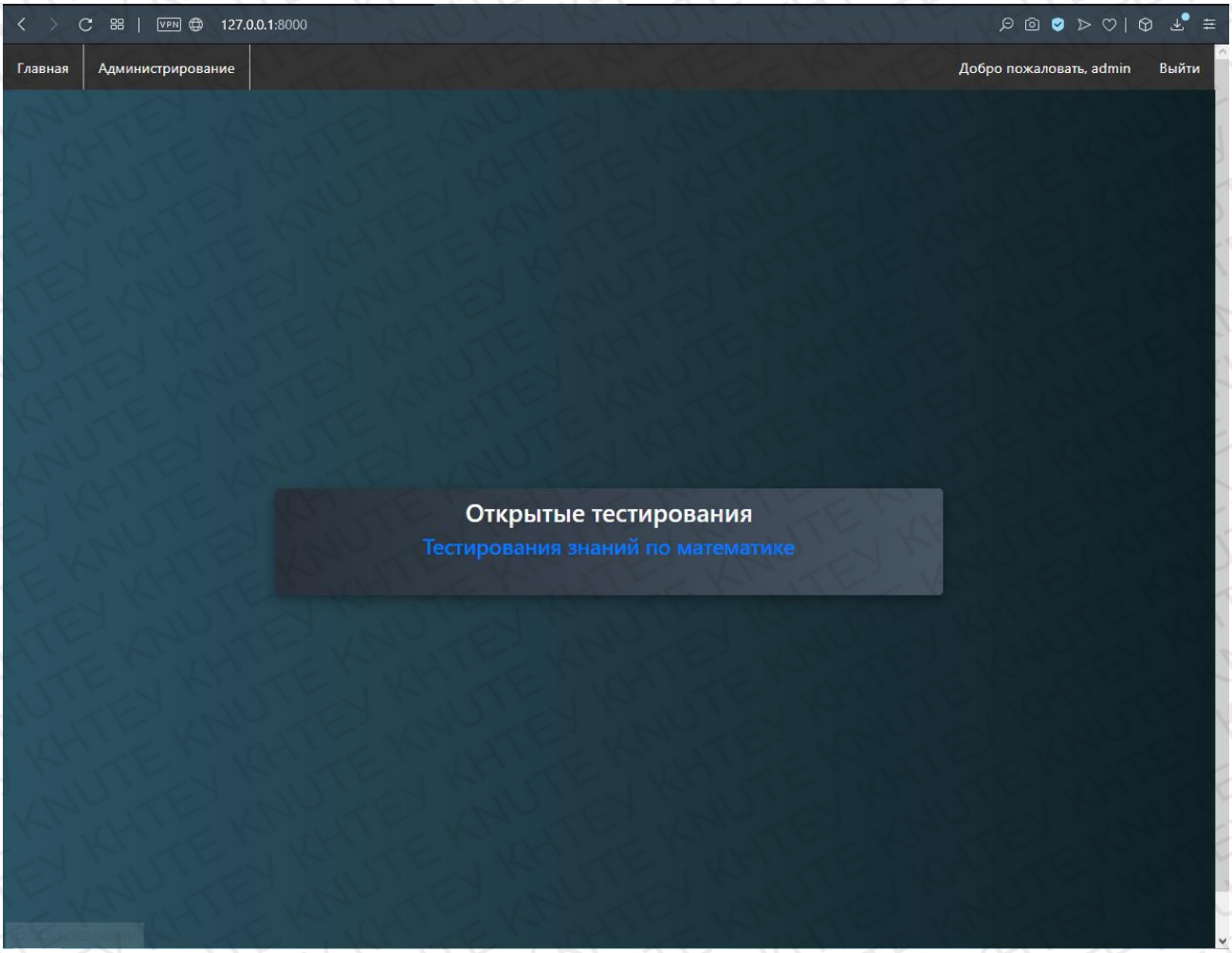

Рис. 3.6 — Сторінка активних опитів

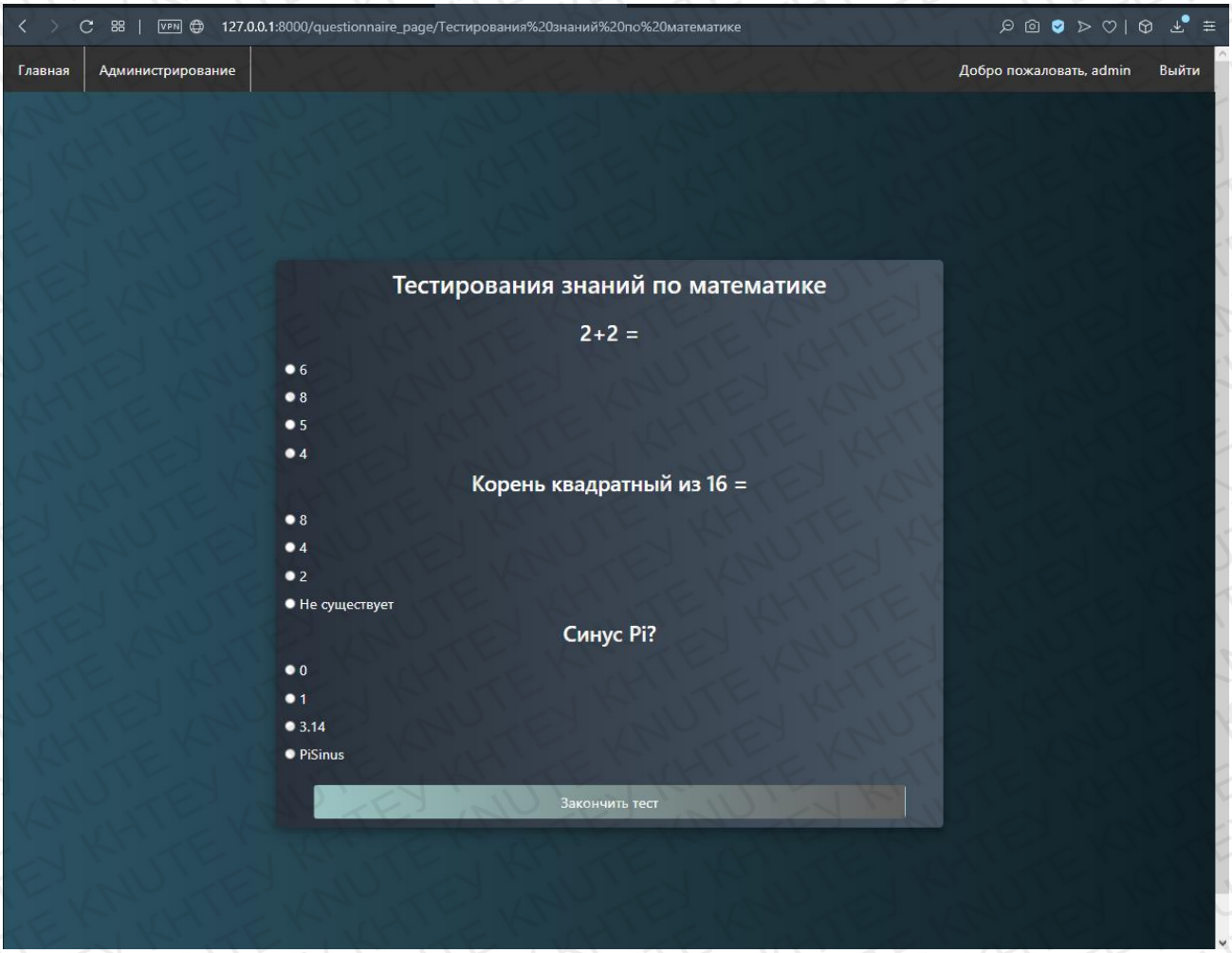

Рис. 3.7 — Сторінка опиту

ВИСНОВКИ І РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Узагальнення результатів теоретичної та практичної частини роботи дає підстави для таких висновків і результатів:

1.Проаналізовано теоретичні підходи та стан наукової розробленості питання розробки Web-системи для оцінювання і перевірки знань студентів, що свідчать недостатність розробленості питання та наявність великих пробілів в дослідженні.

2.Розроблено модель Web-системи для оцінювання і перевірки знань студентів яка складається з модулів публікації оцінювання, реєстрації та авторизації користувачів, проходження оцінювання, відправки бланку відповідей адміністратору.

3.З'ясовано особливості використання програмно-технічних засобів Web-системи для оцінювання і перевірки знань студентів, що являють собою чітку необхідність в постійні підтримці адекватного функціонування серверу веб-додатку з метою постійно працездатності всієї системи.

4.Розроблено Web-систему для оцінювання і перевірки знань студентів мовою програмування Python, що дало змогу забезпечити можливість навчальних закладів автоматизувати найбільш рутинну частину викладацької діяльності.

Завдяки виконанню завдань, поставлених на початку роботи було отримано повноцінну систему, що здатна виконувати закладений в неї функціонал та готова до використання в реальних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адаменко О.В., Духовна М.М., Панченко Л.Ф. та ін. Тестові завдання для контролю знань в курсі "Обчислювальна техніка і технічні засоби навчання": Навч.- метод, посібник / За ред. Т.О. Козлакової. — К.: ВІПОЛ, 1996.-84 с.
2. Алексейчук І.С. Про технологію створення системи тестування / І.С. Алексейчук // Нові технології навчання: Науково-методичний збірник. — К.: НМЦВД, 2000. — С.43-92.
3. Аткінсон Л., Сураскін З. PHP5. Бібліотека професіоналу. / Л. Аткінсон, З. Сураскін — М. : «Вільямс», 2006 — 543 с.
4. Биков В.Ю., Лапінський В.В. Методологічні та методичні основи створення і використовування електронних засобів навчального призначення // Комп'ютер у школі та сім'ї. — №3 . — 2012.
5. Болье А. — Learning SQL [Електронний ресурс]. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.
6. Використання ІКТ у навчально-виховному процесі студентів ВНЗ [Електронний ресурс]. - Режим доступу : http://olgayuzuk.do.am.
7. Гладка Л.І. Системний підхід до оцінки якості знань у формі комп'ютерного тестування [Електронний ресурс] / Л. І. Гладка. – Науковий вісник Донбасу. - 2014. - №2(26). - Режим доступу: http://nvd.luguniv.edu.ua/archiv/NN26/index.html.
8. Гуцало Е. У. Педагогічне тестування в системі контролю і оцінки якості навчання студентів (на базі дисциплін психолого-педагогічного циклупедагогічного університету) / Е. У. Гуцало. – Кіровоград : РВВ КДПУ ім. В. Винниченка, 2011. – 68 с.
9. Дмитрий Котеров, Алексей Костарев PHP. В подлиннике. / Д. Котеров, А. Костарев — Спб.: «БХВ-Петербург», 2005. — 1120 с.

10. Колисниченко Д. Н. Самоучитель PHP 5 / Д. Н. Колисниченко — СПб.: Наука и Техника, 2007. — 640 с.
11. Кондратенко Ю.П. Програмний комплекс для автоматизованого тестування знань студентів [Електронний ресурс] / Ю.П. Кондратенко, С.О. Волкова. - Режим доступу : http://svolkova.weebly.com.
12. Кравцов Г. М. Модель контроля знаний в системе дистанционного тестирования WEB-EXAMINER по стандарту IMS / Г. М. Кравцов, Д. Г. Кравцов // Proceedings ITEA-2007 (Second International Conference «New Information Technologies in Education for All: State of the Art and Prospects», Ukraine, IRTC, 21-23 November 2007). – Kiev, 2007. – P. 187-194.
13. Кузнецов Максим, Симдянов Игорь MySQL на примерах. / М. Кузнецов, И. Симдянов — Спб.: «БХВ-Петербург», 2008. — 952 с.
14. Кузнецов Максим, Симдянов Игорь PHP 5/6. / М. Кузнецов, И. Симдянов — Спб.: «БХВ-Петербург», 2009. — 1024 с.
15. Майоров А. Н. Теория и практика создания тестов для системы образования/ Майоров А. Н. — М.: Народное образование, 2000. — 352 с.
16. Мисник Л.Д. Методичні основи навчання та контролю знань студентів з теоретичної механіки з використанням тестових технологій / Л.Д. Мисник // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. – Х. : НТУ «ХПІ», 2013. - № 70 (1043). – С.122-126 .
17. Рашкевич Ю. Моделювання навчальних систем / Ю. Рашкевич, Д. Пелешко, М. Пасєка, А. Стецюк. – Технічні вісті. – 2002. – №1(14), 2(15). – С. 55-62.
18. Роберт Шелдон, Джоффрей Мойе MySQL: базовый курс Beginning MySQL./ Р. Шелдон, Д. Мойе — М.: «Диалектика» 2007. — 880 с.
19. Сальникова Н.А. Проведение аттестации знаний студентов с помощью компьютерного тестирования / Н.А. Сальникова, И.П. Михнев // Известия

волгоградского государственного технического университета. – 2007. – №4. – Т. 7. – С. 182-184.

20. Скотт Хокінс. Адміністрування веб-сервера Apache і керівництво по електронній комерції. / С. Хокінс. — М.: «Вільямс», 2001. — 336 с.
21. Ткаченко Л.П. Підходи до оцінювання знань в умовах застосування інноваційних технологій навчання / Л. П. Ткаченко // Сучасні педагогічні технології підготовки фахівців нового покоління : матеріали IV Міжнародної конференції. – Кривий Ріг, 2006. – С. 207–211.
22. Хокинс С. Администрирование Web-сервера Apache / С. Хокинс — М.: Вильямс, 2001. — 336 с.
23. Mogey N., Watt H. The use of computers in the assessment of student learning // LTDI: Implementing Learning Technology. – Edinburgh, 1996. – P. 50-51.

ДОДАТКИ

Додаток А. Лістинг програмного коду

```
from django.contrib import admin
from .models import questionnaireAnswers, questionnaireData
admin.site.register(questionnaireAnswers)
admin.site.register(questionnaireData)
from django.apps import AppConfig


class MainappConfig(AppConfig):
    name = 'mainApp'
    verbose_name = 'Опросник'
from django.forms import ModelForm
from django.contrib.auth.forms import UserCreationForm
from django import forms
from  django.contrib.auth.models import User

class CreateUserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
from django.db import models
from django.contrib.auth.models import User


class questionnaireData(models.Model):
```

```
        questionnaire_name = models.TextField('Название опросника')
        questions_text = models.TextField('Вопросы')
        questions_answers = models.TextField('Ответы')

        def __str__(self):
            return self.questionnaire_name

        class Meta:
            verbose_name = 'Опрос'
            verbose_name_plural = 'Опросы'


    class questionnaireAnswers(models.Model):
        questionnaire_name_fk = models.ForeignKey(questionnaireData,
on_delete=models.CASCADE)
        user = models.ForeignKey(User, on_delete=models.CASCADE)
        answers = models.TextField('Ответы')

        def __str__(self):
            return self.user.username + "/" +
self.questionnaire_name_fk.questionnaire_name

        class Meta:
            verbose_name = 'Ответ'
            verbose_name_plural = 'Ответы'
    from django.urls import path

    from . import views
```

```
urlpatterns = [
    path('', views.index, name='index'),
    path('register/', views.regist, name='register'),
    path('login/', views.loginPage, name='login'),
    path('logout/', views.logoutUser, name='logout'),
    path('questionnaire_page/<str:name>',                views.questionnaire_page,
name='questionnaire_page'),

]
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .forms import CreateUserForm
from .models import questionnaireAnswers, questionnaireData
from django.contrib.auth.models import User


def loginPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('index')
        else:
```

```
            messages.info(request, 'Имя пользователя или пароль введены неверно')
        context = {}
        return render(request, 'login.html', context)


    def regist(request):
        form = CreateUserForm
        if request.method == "POST":
            form = CreateUserForm(request.POST)
            if form.is_valid():
                form.save()
                return redirect('login')
        context = {'form': form}
        return render(request, 'register.html', context)


    def logoutUser(request):
        logout(request)
        return redirect('login')


    def index(request):
        if request.user.is_authenticated:
            questionnaire_list = questionnaireData.objects.all()
            return render(request, 'index.html', {'questionnaire_list': questionnaire_list})
        else:
```

```
            return redirect('login')


        def questionnaire_page(request, name):
            questionnaireCurr = questionnaireData.objects.get(questionnaire_name=name)
            questName = questionnaireCurr.questionnaire_name
            questions = questionnaireCurr.questions_text.split('/')
            tempAns = questionnaireCurr.questions_answers.split('/')
            tempQA = []
            for i in range(len(questions)):
                tempQA.append(questions[i] + '*' + tempAns[i])

            quest_ans = []
            for i in range(len(tempQA)):
                quest_ans.append(tempQA[i].split('*'))

            answers = []
            if request.method == 'POST':
                for i in range(1, len(quest_ans) + 1):
                    answers.append(request.POST.get('quest' + str(i)))
                addAnswersToDB(answers, questName, request)
                return redirect('index')
            return render(request, 'questionnaire_page.html', {'questName': questName, 'quest_ans': quest_ans})


        def addAnswersToDB(answers, name, req):
```

```
        quest = questionnaireData.objects.get(questionnaire_name=name)
        strAnsw = ""
        for str in answers:
            strAnsw += str + "\n"
        answ = questionnaireAnswers(questionnaire_name_fk=quest, user=req.user,
answers=strAnsw)
        answ.save()
    <!DOCTYPE html>
    <html lang = "ru">
     <head>
         <meta charset="utf-8">

        <title>
            Тестирования
        </title>


        <link                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknL
PMO" crossorigin="anonymous">
            <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
        <link                                                    rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.6.1/css/all.css"      integrity="sha384-
gfdkjb5BdAXd+lj+gudLWI+BXq4IuLW5IT+brZEZsLFm++aCMlF1V92rMkPaX4P
P" crossorigin="anonymous">
```

```
{% load static %}
    <link rel="stylesheet" type="text/css" href="{% static 'index/css/style.css' %}">
    <style>
        body,
        html {
            margin: 0;
            padding: 0;
            height: 100%;
            background: #0F2027;  /* fallback for old browsers */
        background: -webkit-linear-gradient(to right, #2C5364, #203A43, #0F2027); /* Chrome 10-25, Safari 5.1-6 */
        background: linear-gradient(to right, #2C5364, #203A43, #0F2027); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */

        }
        .user_card {
            width: 750px;
            margin-top: auto;
            margin-bottom: auto;
            background: #485563;  /* fallback for old browsers */
        background: -webkit-linear-gradient(to right, #29323c, #485563);  /* Chrome 10-25, Safari 5.1-6 */
        background: linear-gradient(to right, #29323c, #485563); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */

            position: relative;
```

```
            display: flex;
            justify-content: center;
            flex-direction: column;
            padding: 10px;
            box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0
rgba(0, 0, 0, 0.19);
            -webkit-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px
20px 0 rgba(0, 0, 0, 0.19);
            -moz-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px
20px 0 rgba(0, 0, 0, 0.19);
            border-radius: 5px;

        }

        .form_container {
            margin-top: 20px;
        }

        #form-title{
            color: #fff;

        }

        .login_btn {
            width: 100%;
            background: #33ccff !important;
            color: white !important;
        }
```

```
.login_btn:focus {
    box-shadow: none !important;
    outline: 0px !important;
}
.login_container {
    padding: 0 2rem;
}
.input-group-text {
    background: #f7ba5b !important;
    color: white !important;
    border: 0 !important;
    border-radius: 0.25rem 0 0 0.25rem !important;
}
.input_user,
.input_pass:focus {
    box-shadow: none !important;
    outline: 0px !important;
}

#messages{
    background-color: grey;
    color: #fff;
    padding: 10px;
    margin-top: 10px;
}

ul {
list-style-type: none;
```

```
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
  border-right:1px solid #bbb;
}

li:last-child {
  border-right: none;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #111;
}

.active {
```

```
 background-color: #4CAF50;
}
.usrName {
float:right;
vertical-align:5px;
color:white;
display: block;
text-align: center;
padding: 14px 16px;
}
        </style>

 </head>
 <body>
<ul>
 <li><a href="{% url 'index' %}">Главная</a></li>
  <li><a href="{% url 'admin:index' %}">Администрирование</a></li>
  <li  style="float:right" st><a href="{% url 'logout' %}">Выйти</a></li>
 <li class="usrName">Добро пожаловать, {{request.user}}</li>

</ul>

 <div class="container h-100">
                <div class="d-flex justify-content-center h-100">
                    <div class="user_card">
                            <div class="d-flex justify-content-center">
```

```
                    <h3 id="form-title">Открытые тестирования</h3>

                </div>

            {% if questionnaire_list %}

              {% for a in questionnaire_list %}

                <h4 align="center"> <a href="{% url 'questionnaire_page' a.questionnaire_name %}">{{a.questionnaire_name}}</a></h4><br>

              {% endfor %}

            {% endif %}

                </div>
              </div>
            </div>

            </body>
          </html>

          <!DOCTYPE html>
          <html lang = "ru">
            <head>
                <meta charset="utf-8">
```

```
<title>
    Продукты
</title>


<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.1/css/all.css" integrity="sha384-gfdkjb5BdAXd+lj+gudLWI+BXq4IuLW5IT+brZEZsLFm++aCMlF1V92rMkPaX4PP" crossorigin="anonymous">

    {% load static %}
    <link rel="stylesheet" type="text/css" href="{% static 'index/css/style.css' %}">
<style>
        body,
        html {
            margin: 0;
            padding: 0;
            height: 100%;
            background: #0F2027;  /* fallback for old browsers */
```

```
        background: -webkit-linear-gradient(to right, #2C5364, #203A43, #0F2027); /* Chrome 10-25, Safari 5.1-6 */
        background: linear-gradient(to right, #2C5364, #203A43, #0F2027); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */

    }
    .user_card {
        width: 750px;
        margin-top: auto;
        margin-bottom: auto;
        background: #485563;  /* fallback for old browsers */
        background: -webkit-linear-gradient(to right, #29323c, #485563);  /* Chrome 10-25, Safari 5.1-6 */
        background: linear-gradient(to right, #29323c, #485563); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */

        position: relative;
        display: flex;
        justify-content: center;
        flex-direction: column;
        padding: 10px;
        box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
        -webkit-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
        -moz-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
        border-radius: 5px;
```

```
}

.form_container {
    margin-top: 20px;
}

#form-title{
    color: #fff;

}

.login_btn {
    width: 100%;
    background: #616161;  /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, #9bc5c3, #616161);  /* Chrome 10-25, Safari 5.1-6 */
    background: linear-gradient(to right, #9bc5c3, #616161); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */

    color: white !important;
}
.login_btn:focus {
    box-shadow: none !important;
    outline: 0px !important;
}
.login_container {
    padding: 0 2rem;
```

```
        }
        .input-group-text {
            background: #f7ba5b !important;
            color: white !important;
            border: 0 !important;
            border-radius: 0.25rem 0 0 0.25rem !important;
        }
        .input_user,
        .input_pass:focus {
            box-shadow: none !important;
            outline: 0px !important;
        }

        #messages{
            background-color: grey;
            color: #fff;
            padding: 10px;
            margin-top: 10px;
        }

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}
```

```
li {
  float: left;
  border-right:1px solid #bbb;
}

li:last-child {
  border-right: none;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #111;
}

.active {
  background-color: #4CAF50;
}
.usrName {
float:right;
vertical-align:5px;
color:white;
```

```
display: block;
text-align: center;
padding: 14px 16px;
}

        </style>

 </head>
 <body>

<ul>
 <li><a href="{% url 'index' %}">Главная</a></li>
  <li><a href="{% url 'admin:index' %}">Администрирование</a></li>
  <li  style="float:right" st><a href="{% url 'logout' %}">Выйти</a></li>
 <li class="usrName">Добро пожаловать, {{request.user}}</li>

</ul>

 <div class="container h-100">
                <div class="d-flex justify-content-center h-100">
                        <div class="user_card">
                                <div class="d-flex justify-content-center">

 {% if questName %}
                                        <h3 id="form-title">{{questName}}</h3>
 {% endif %}
                                </div>
{% if quest_ans %}
```

```
                    <form method="POST" action="">
                {% csrf_token %}
                        <div class="input-group mb-3">

                        </div>

                    {% for a in quest_ans %}
                      {% for b in a %}
                <fieldset>
                    {% if forloop.first %}
                    <legend>  <h4  align="center"  style="color:
white;">{{b}}</h4> </legend>
                      {% else %}
                     <input                          type="radio"
id="quest{{forloop.parentloop.counter}}.ans{{forloop.counter}}"
name="quest{{forloop.parentloop.counter}}" value="{{b}}">
                     <label
for="quest{{forloop.parentloop.counter}}.ans{{forloop.counter}}"  style="color:
white;">{{b}}</label>
                      {% endif %}
                    {% endfor %}
                </fieldset>
                {% endfor %}
```

```
                                        <div class="d-flex justify-content-
center mt-3 login_container">
                                            <input class="btn login_btn"
type="submit" value="Закончить тест">
                                        </div>
                                    </form>


        {% endif %}

                </div>
            </div>
        </div>
        </body>
    </html>
```