МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПЗ «ТРЕНАЖЕР ГРАМАТИКИ УКРАЇНСЬКОЇ МОВИ» ДЛЯ УЧНІВ З ВИКОРИСТАННЯМ WEB-ТЕХНОЛОГІЙ»**

Виконала: студентка 2 курсу, групи 8.1228-з
Спеціальності 122 комп'ютерні науки
(шифр і назв1а спеціальності)
освітньої програми комп'ютерні наук
А.В. Мацієвська
(ініціали та прізвище)
Керівник завідувач кафедри комп'ютерних наук, доцент, к.т.н., Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент заступник завідуючого кафедри фізики Національного університету «Запорізька політехніка», доцент, к. ф – м. н., Правда М. І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Математичний

Кафедра комп'ютерних наук

Рівень вищої освіти Магістр

Спеціальність 122 комп'ютерні науки

Освітня програма комп'ютерні науки

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерних наук,
к.т.н., доцент

_____ Борю С.Ю.
(підпис)

« 29 » травня 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Мацієвській Анні Вікторовні

(прізвище, ім'я та по батькові)

1. Тема роботи Розробка ПЗ «Тренажер граматики української мови» для учнів з використанням Web-технологій

керівник роботи Борю Сергій Юрійович, к.т.н., доцент
(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » 05 2019 року № 812-с

2. Строк подання студентом роботи 26.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Програмна реалізація
4. Тестування програмного продукту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 02.09.2019

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	1.06.2019	
2.	Збір вихідних даних.	28.05.2019	
3.	Обробка методичних та теоретичних	12.09.2019	
	джерел.		
4.	Розробка першого та другого розділу.	27.09.2019	
5.	Розробка третього розділу.	12.10.2019	
6.	Оформлення та нормоконтроль	20.12.2019	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	08.01.2020	

Студент _____ А.В. Мацієвська
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка ПЗ «Тренажер граматики української мови» для учнів з використанням Web-технологій»: 53 с., 15 рис., 1 табл., 12 джерел, 6 додатків.

ВЕБ-ДОДАТОК, КОМП'ЮТЕРНИЙ ТРЕНАЖЕР, ФРЕЙМВОРК, HTML, CSS, DJANGO, PYTHON

Об'єкт дослідження – комп'ютерне тренування навичок з граматики української мови.

Мета роботи: розробити комп'ютерний тренажер з граматики української мови для учнів початкових класів та реалізувати його у якості веб-додатку.

Метод дослідження – аналітичний.

У першому розділі кваліфікаційної роботи проведено аналіз особливостей побудови та використання веб-тренажерів та проведено огляд, існуючих програмних засобів для тренування навичок з української граматики.

У другому розділі розроблено вимоги до програмного продукту з тренування граматики української мови, сформульовано задач, які має вирішувати тренажер, розроблено його архітектуру, інформаційної модель та алгоритм реалізації.

Третій розділ присвячено вибору програмного інструментарію для розробки веб-сайту тренажеру, реалізації програмного коду та інструкції з впровадженню та експлуатації.

SUMMARY

Master’s Qualification Thesis «Development of «Ukrainian GrammarSimulator» Software for Studying Students Using Web-technologies»: 53 pages, 15 figures, 1 tables, 12 references, 6 supplements.

WEB APP, COMPUTER TRAINER, FRAMEWORK, HTML, CSS, DJANGO, PYTHON

The object of the study is a computer skills training in Ukrainian grammar.

The aim of the study is to develop a computer trainer for Ukrainian grammar for elementary students and to implement it as a web application.

The method of research is analytical.

In the first section of the qualification work analyzes the features of construction and use of web simulators and provides an overview of existing software for training Ukrainian grammar skills.

In the second section the requirements for the Ukrainian grammar training software are developed, the tasks that the simulator should solve are formulated, its architecture, information model and implementation algorithm are developed.

The third section is dedicated to the selection of software tools for the development of the simulator website, the implementation of the code and the instructions for implementation and operation.

ЗМІСТ

Завдання на кваліфікаційну роботу……..…….....2
Реферат…….4
Summary……5
Зміст 6
Вступ 7
1 Особливості побудови та використання веб-тренажерів…….9
1.1 Особливості методики вивчення граматики української мови школярами 9
1.2 Особливості проектування та побудови веб-тренажерів 12
1.3 Огляд існуючих рішень 14
1.4 Постановка задачі 16
1.5 Висновки до розділу 1 17
2 Проектування веб-тренажеру для школярів……18
2.1 Розробка формальних вимог 18
2.2 Формалізація задач тренажеру 19
2.3 Розробка архітектури веб – тренажеру 21
2.4 Розробка інформаційної моделі веб – тренажера 23
3 Розробка веб-ресурсу 27
3.1 Вибір мовного інструментарію для реалізації веб-додатку 27
3.2 Розробка програмного коду 28
3.3 Розробка інструкцій з використання веб – тренажеру 29
Висновки 34
Перелік посилань 35
Додаток А UML діаграма асоціацій 36
Додаток Б Міграції проекту 37
Додаток В Моделі проекту 44
Додаток Г Forms 47
Додаток Д Decorators 50
Додаток Е Urls 52

ВСТУП

Сучасний світ неможливо уявити без інформаційних технологій та Інтернету, який охоплює всі сфери нашого життя. В Інтернет переноситься пошта, телефонія, бізнес.

Веб-сфера стрімко розвивається в наші дні. На сьогоднішній день розроблено велика кількість яскравих сайтів з найрізноманітнішими призначеннями: візитки, торгові, інформаційні, розважальні, соціальні мережі, особисті та корпоративні блоги.

Використання веб-технологій помітно зростає в усіх сферах людського життя, зокрема це стосується сфери освіти. Однією з головних причин посиленої уваги педагогів до проблеми упровадження веб-технологій є зручність та простота використання наявних інструментів для пошуку, створення та використання освітніх веб-ресурсів. Використовуючи їх, можна суттєво підвищити ефективність навчального процесу, активізувати навчально-пізнавальну та самостійну діяльність учнів. У процесі своєї діяльності учителю доводиться вирішувати завдання, пов’язані з пошуком наявних освітніх ресурсів, аналізом їх на доцільність використання у навчальному процесі та створення власних.

Впровадження веб-технологій у навчальний процес та підготовка учителя до їх використання розглядається у ряді праць українських та закордонних авторів. Ці питання є предметом серйозного обговорення на науково-практичних та науково-методичних конференціях, а також на сторінках періодичних видань: журналів «Комп’ютер у школі та сім’ї», «Інформатика та інформаційні технології в навчальних закладах»; газет «Освіта України», «Педагогічна газета», «Інформатика», «Информатика и образование» та ін.

Аналіз науково-педагогічної літератури дає змогу зробити висновок, що використання сучасних веб-технологій та розроблених на їх основі веб ресурсів є важливою передумовою успішності навчальної діяльності.

У даній роботі поставлена задача розробити тренажер граматики української мови для школярів з використанням web-технологій. Отриманий у результаті тренажер поєднає теоретичні положення й практичні завдання, що сприяють засвоєнню граматики української мови і вдосконаленню правописних навичок.

1 ОСОБЛИВОСТІ ПОБУДОВИ ТА ВИКОРИСТАННЯ ВЕБ-ТРЕНАЖЕРІВ

1.1 Особливості методики вивчення граматики української мови школярами

Методика викладання мови — це педагогічна наука, адже предметом її дослідження є процес оволодіння українською мовою в умовах навчання дітей у початковій школі. Методика викладання української мови у школі визначає теоретичні аспекти навчання мови учнів і рекомендує ефективні методи і прийоми вдосконалення мовленнєвого розвитку школярів, формування в них засобами мови і літератури наукового уявлення про мову як знакову систему. Джерела збагачення методики:

– практичний досвід, узагальнення передового педагогічного досвіду;

– розвиток мовознавства та літературознавства як основних базових наук;

– розвиток суміжних наук — психології, дидактики;

– новітні дослідження в галузі теорії навчання мови;

– методичний експеримент;

– створення нових програм, підручників, нових практичних систем навчання, конструювання нових типів уроків тощо.

Дві взаємопов’язані частини, теоретична і практична, є складовими методики викладання мови:

Таблиця 1.1 – Складові методики викладання мови

Теоретична	Практична
1. Наукові основи методики викладання мови.	1. Методи та прийоми навчання мови.
2. Історія виникнення та розвиток методів навчання мови.	2.«Механізм» упровадження конкретних методів та прийомів у процес вивчення мови.

Продовження таблиці 1.1

3. Структура і форми організації навчального процесу. 4. Наукові вимоги до підручників та посібників з мови. Мета: дослідити процес оволодіння знаннями і вміннями, його закономірності; визначити принципи навчання; обґрунтувати методи; створити наукові основи конструювання уроків, їх циклів	Мета: озброїти вчителя та учнів системою методів та прийомів, спрямованою на оволодіння мовою

У формуванні теоретичних основ методика мови спирається на психолого-педагогічні науки, які є важливими не лише для уточнення змісту і методів навчання мови, адже глибокий аналіз причин труднощів, які супроводжують навчання дітей, індивідуальний підхід до учнів, знання загальних закономірностей розвитку мозку дитини забезпечать застосування адекватних (відповідних) прийомів навчання, сприятимуть засвоєнню програмового матеріалу. Останнім часом методика мови взяла на озброєння провідні положення молодої науки психолінгвістики, яка трактує мову як діяльність. Як педагогічна наука методика мови керується настановами педагогіки, зокрема дидактики — розділу педагогіки, який визначає загальну теорію освіти і навчання та теорії виховання у процесі навчання мови і читання. В основу методики мови покладено дидактичні принципи, які є підґрунтям навчального процесу. Методика мови тісно пов’язана з мовознавством. Дані мовознавчих наук становлять лінгвістичну основу викладання рідної мови. Методика мови будує свою теорію, функціонує і розвивається також у зв’язку з іншими науками.

У методиці використовуються методи як дидактичних класифікацій (словесні, наочні, практичні), так і суто методичних. Наприклад, у навчанні грамоти — звуковий аналітико-синтетичний метод, на уроках читання — метод читання, перечитування і бесіди за змістом тексту. Методи навчання — це

дидактична категорія. Методика української мови використовує їх з урахуванням особливостей цієї навчальної дисципліни, а також розробляє та апробує свої власні методи, обумовлені специфікою дисципліни. Методи навчання передбачають:

– визначення мети навчання як передумови взаємодії вчителя та учня;

– визначення способу подання навчального матеріалу;

– визначення способу взаємодії вчителя та учнів;

– визначення характеру оцінювання засвоєння.

Метод навчання — спосіб упорядкованої взаємної діяльності вчителя й учнів, спрямований на розв’язання навчально-виховних завдань. Методи навчання становлять систему способів діяльності суб’єктів навчального процесу, сукупність прийомів, спрямованих на досягнення певної мети засвоєння мовного матеріалу. Класифікація методів навчання мови можлива за такими ознаками:

– за характером керівництва розумовою діяльністю учнів: пояснювально - ілюстративний, репродуктивний, проблемний, частково пошуковий, дослідницький;

– за джерелом знань: наочні, словесні, практичні;

– за напрямом діяльності: практичні, практико-теоретичні, теоретичні.

Окремий аспект на сучасному етапі розвитку методики становлять інтерактивні методи навчання. Теорія методів навчання впорядковує діяльність учителя та учня, створює основу для створення технології навчання. Головна стратегічна лінія розвитку освітньої системи нині полягає в її адаптації до змінних соціально-економічних умов суспільства. Це націлює педагогів зміщувати акцент у методах навчання на пошук, «продукування» нових індивідуальних способів навчальної діяльності. Інтерактивні методи навчання спрацьовують через осмислення кожним учнем своєї діяльності, що звернена до здатності дитини перебудовувати свої дії, досвід, мотиви і потреби. Інтерактивне навчання допомагає сформувати в особистості вміння, навички

самостійно вивчати певні явища, процеси, користуючись при цьому інформацією [1].

1.2 Особливості проектування та побудови веб-тренажерів

Розвиток сучасного суспільства характеризується процесом інформатизації і, у першу чергу, інформатизації освіти. У цих умовах наука й освіта повинні постійно вдосконалюватись. Найбільш перспективним напрямком інтенсифікації навчального процесу у навчальних закладах є його комп'ютеризація.

Динамізм розвитку обчислювальної техніки, технологій програмування і телекомунікацій визначили значний прорив у сфері розробки різних комп'ютерних систем навчання, у тому числі комп'ютерних тренажерів, та їх застосування у підготовці школярів.

Комп'ютерний тренажер – автоматизований апаратно-програмний функціонально орієнтований комплекс для навчання людини та відпрацювання певних навичок і вмінь.

Основний принцип, що повинен бути покладений в основу створення перспективних комп'ютерних тренажерів − принцип відповідності засобів навчання змісту навчання і дидактичним завданням. Іншими принципами, які складають основу створення перспективних комп'ютерних тренажерів, є:

– принцип раціональності – раціонального розподілу завдань навчання за створюваними засобами навчання, тобто складність (не завжди технічна) розроблюваних засобів повинна відповідати значущості вирішуваних на них завдань підготовки;

– принцип узгодженості – усі засоби навчання повинні бути пов'язані між собою загальними й єдиними для цієї системи нормами оцінок, тобто перехід учня з одного технічного засобу навчання на інший повинен здійснюватися лише на основі оцінки його діяльності на попередньому засобі (засобах);

– принцип достатності – кожний технічний засіб системи має вирішувати певне коло завдань навчання, що не дублюють завдання, які вирішуються на інших засобах;

– принцип безперервності – кількість і технічні характеристики засобів, що складають систему засобів навчання, повинні вибиратися таким чином, щоб забезпечити безперервний процес навчання при повному завантаженні засобів, що його реалізують. Цей принцип вимагає врахування специфіки діяльності фахівця в процесі навчання. При цьому найбільш складні для засвоєння елементи діяльності (дії, операції), які вимагають великих витрат часу, повинні відпрацьовуватися на пристроях з більшою пропускною здатністю;

– принцип функціональної гнучкості – елементи тренажеру повинні бути спроектовані з урахуванням можливих змін характеру та змісту професійної діяльності, що відбуваються впродовж терміну експлуатації системи навчання;

– принцип наступності – при розробці кожного нового засобу, заміни існуючого необхідно враховувати досвід проектування й експлуатації створеного раніше. Новий засіб має без значних змін вписуватися в існуючу систему навчання як методично, так і конструктивно [2].

Проведення аналізу та виявлення особливостей розробки веб – сервісів показує, що в загальному випадку веб – додатки можна розглядати як додатки з клієнт – серверною архітектурою [3]. Взаємодія з користувачем відбувається через сторінки та форми, з яких користувач отримує необхідну для нього інформацію, а також відправляє зворотню інформацію. Загальна структура веб – додатку представляє собою логіку веб – додатку, яка розподілена між сервером та клієнтом, зберігання та обробку даних, яка здійснюється, переважно, на сервері, її представлення в зручному для користувача вигляді – у браузері.

1.3 Огляд існуючих рішень

Один зі шляхів підвищення культури мовлення – мовна самоосвіта.

Тренажер із правопису української мови – цікавий ресурс для вивчення української мови. На тренажері української мови всі охочі можуть перевірити свої мовні знання, вивчити чи повторити правила, перевірити на правильність тексти, підняти свій освітній рівень загалом.

Онлайн тренажер з правопису української мови [0] поєднує теоретичні положення й практичні завдання, що сприяє засвоєнню граматики української мови і вдосконаленню правописних навичок. Навчальні матеріали додатково містять різноманітну пізнавальну інформацію.

Регулярно займаючись на тренажері можна:

– перевірити свої знання української мови;

– вивчити й повторити правила;

– удосконалити свої навички спілкування;

– підняти освітній рівень.

Нижче приведено декілька існуючих ресурсів для перевірки своїх знань з української мови та літератури.

Онлайн-тест словникового запасу (рис.1.1). Доволі простий тренажер, який нещодавно був дуже популярним серед користувачів українського Facebook. Як серйозне джерело підготовки його навряд чи можна розцінювати, але приблизно оцінити багатство свого словника – це завжди цікаво.

Комплексний тренажер з української мови [5] (рис.1.2). Поєднує компактний виклад теоретичного матеріалу із великим обсягом практичних завдань. Ви зможете перевірити свої знання із фонетики, лексики, орфографії та культури мова. Чудовий додатковий засіб перевірки й засвоєння знань у підготовці до ЗНО з української мови.

Рисунок 1.1 – Онлайн-тест словникового запасу

Рисунок 1.2 – Комплексний тренажер з української мови

Зручний і красивий ресурс із великою базою вправ, головний акцент яких – допомогти розібратись зі «складними випадками» (рис.1.3).

Інтерактивний додаток з української літератури [6], який допоможе вам краще засвоїти відповідності між письменниками та їх псевдонімами (рис.1.4).

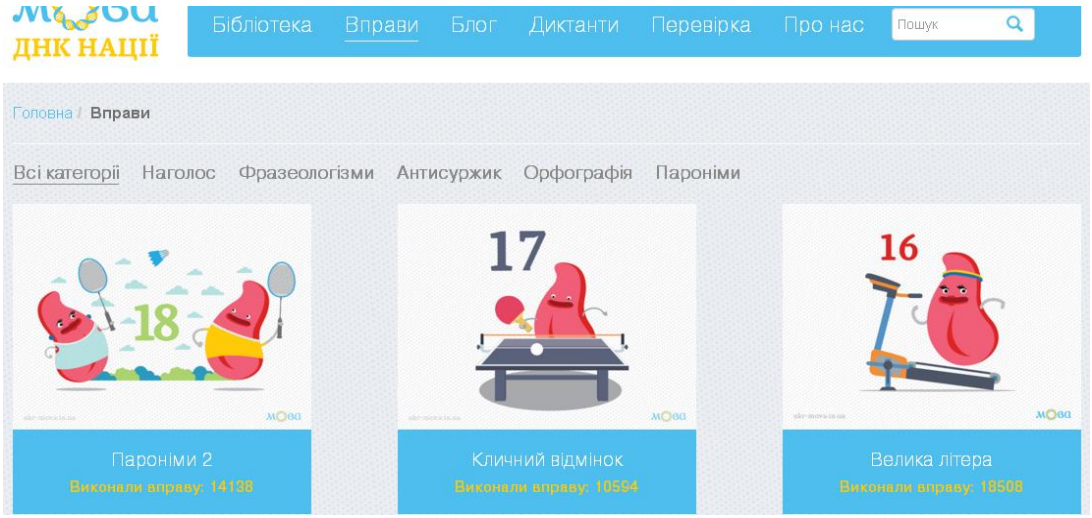


Рисунок 1.3 – Ресурс із великою базою вправ

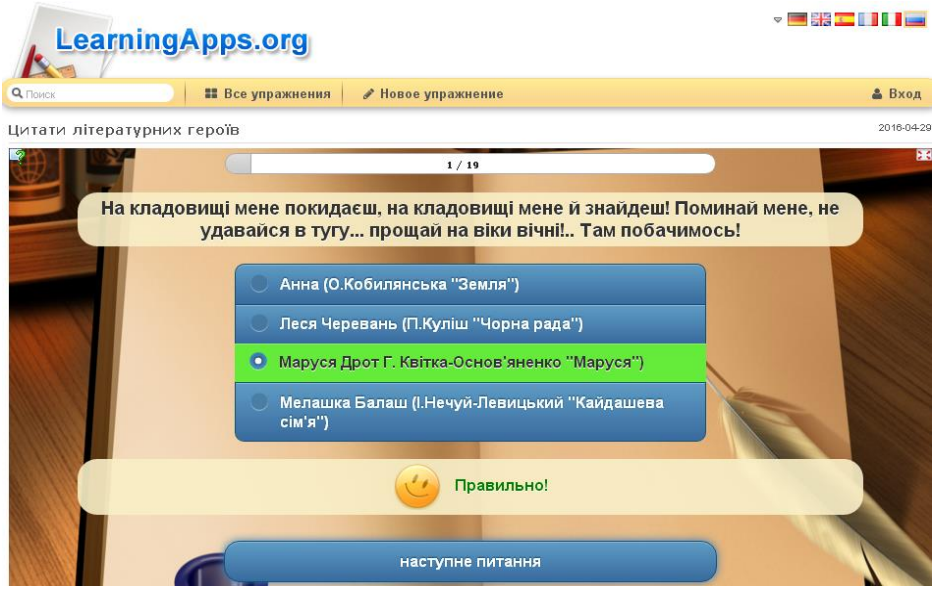


Рисунок 1.4 – Інтерактивний додаток з української літератури

1.4 Постановка задачі

Метою кваліфікаційної роботи є розробка програмного засобу «Тренажер граматики української мови» для школярів з використанням web-технологій».

Для досягнення поставленої мети необхідно розв'язуються наступні задачі:

– аналіз існуючих програмних засобів з вивчення граматики з української мови;

– розробка вимог до програмного продукту з тренування граматики української мови;

– формалізація задач, які вирішує тренажер;

– розробка архітектури веб-тренажера;

– розробка інформаційної моделі ВТ;

– розробка алгоритмів задач ВТ;

– розробка схеми зберігання контенту;

– вибір програмного інструментарію для розробки веб-сайту тренажеру граматики української мови;

– реалізація таблиць бази даних;

– реалізація програмного коду;

– розробка інструкцій по впровадженню і експлуатації;

– висновки.

1.5 Висновки до розділу 1

Використання у навчальному процесі інформаційно-комунікаційних технологій значно посилює його ефективність. Найбільшого ефекту можна досягнути лише при комплексному використанні засобів інформаційно-комунікаційних технологій на різних видах занять в процесі експериментально-дослідницької, інформаційно-пошукової, самостійної навчальної діяльності. Використання освітніх веб-ресурсів є доцільним при вивченні будь-яких дисциплін.

2 ПРОЕКТУВАННЯ ВЕБ-ТРЕНАЖЕРУ ДЛЯ ШКОЛЯРІВ

2.1 Розробка формальних вимог

Оскільки розроблюваний Інтернет ресурс орієнтований насамперед на школярів молодших класів необхідно враховувати вимоги до програмного забезпечення для дітей від 6 до 10 років, а саме:

– інтуїтивно зрозумілий інтерфейс з мінімальною кількістю об’єктів на одній сторінці;

– навігаційна структура; кожна сторінка інтернет ресурсу має містити навігаційні елементи, інформуючі, куди треба йти далі, де користувач знаходиться зараз та де він був перед цим;

– адаптований дизайн сайту, який дасть можливість працювати з ресурсом на пристроях з різними форматами моніторів;

– єдність графічного дизайну сайту на всіх сторінках;

– зрозумілість інтерфейсу веб-додатку;

– інтерактивність; вважаючи гіперактивність сучасних юних здобувачів знань, програмний продукт має швидко реагувати на дії учнів та видавати нові завдання.

Функціональними вимогами до тренажеру є наявність:

– форми реєстрації учнів і вчителя;

– функції завантаження вчителем тестів, розподілених за класами та темами;

– можливості збереження балів, набраних учнями під час виконання завдань тренажеру, для перегляду їх у подальшому батьками або вчителем;

– теоретичних відомостей з тем української граматики;

– можливості зміни класу учня при переході до наступного класу.

2.2 Формалізація задач тренажеру

Тренажер, що проектується повинен вирішувати низку задач, пов’язаних із навчальним процесом під час вивчення української мови у початковій школі. Насамперед він є допоміжним засобом оволодіння навичок з граматики української мови. Практично всі теми з української мови, які проходять в початковій школі можна і потрібно закріплювати за допомогою тренажерів, так як матеріалу підручників недостатньо.

Тренажер буде використовуватись для відпрацювання і закріплення різних навичок і вмінь, так і для контролю. Тому в ньому повинен бути закладений лічильник правильних і неправильних відповідей.

Для того, щоб питання не повторювались при повторному їх вирішенні, тренажеру необхідна велика база питань, з якої система буде обирати завдання довільним чином.

Вчителеві система має надавати зручний інтерфейс введення нових питань для оновлення бази тестів.

Визначимось із категоріями користувачів та їх ролями у системі. Виходячи з функціональної моделі додатку маємо наступні категорії користувачів та прецедентів:

а) системне оточення (actors):
   1) вчитель;
   2) учень;

б) системні прецеденти (use case):
   1) реєстрація;
   2) завантаження тексту;
   3) вивчення теорії;
   4) проходження тесту;
   5) ознайомлення з результатами тесту.

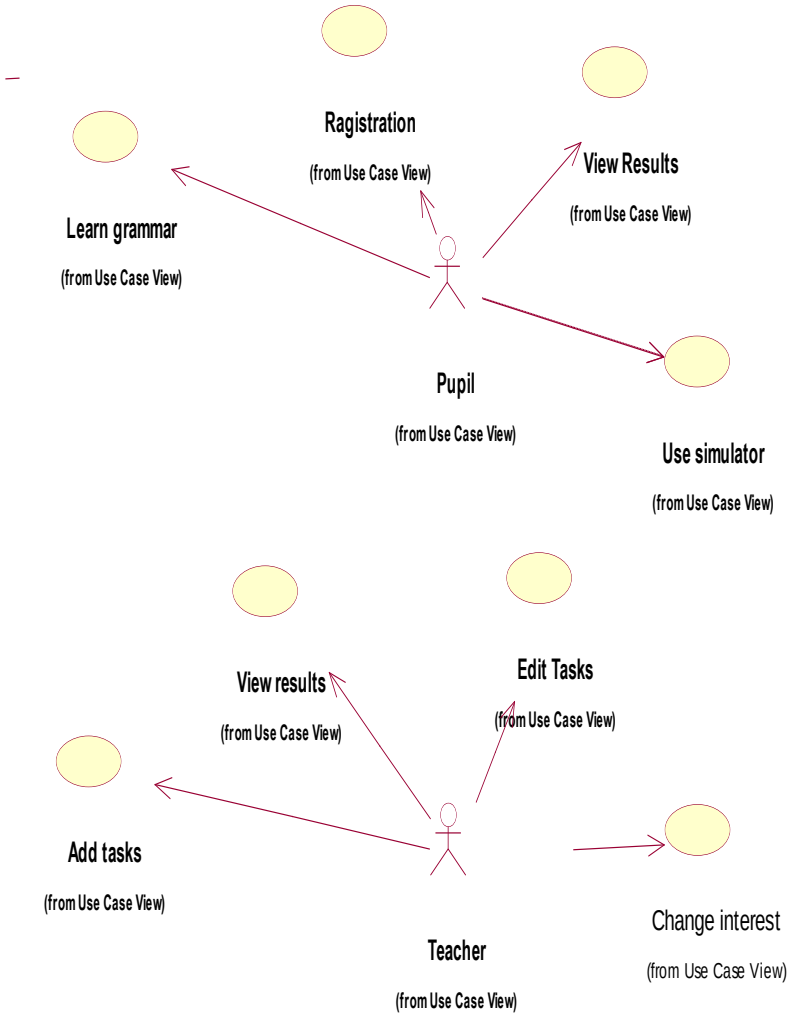


Рисунок 2.1 – UML-діаграма прецедентів

Розподіл прецедентів між користувачами системи схематично показано на рис. 2.1. Вчитель реєструється, а в подальшому авторизується у системі та може додавати тести за темами, редагувати їх, продивлятись списки зареєстрованих учнів та результати їх тестування. Учні також можуть реєструватись та авторизуватись у системі, але їх права обмежені можливістю їх акаунту, тобто. проходити тест, продивлятись свої набрані бали з тестів, ознайомлюватись з теорією та переходити до іншого класу.

Перелічимо усі задачі, які вирішує програмний тренажер:

– реєстрація двох категорій користувачів: вчителя та учня;

– вчитель додає тести тренажеру та продивляється результати проходження тестів учнями,

– вчитель змінює номер класу учня, коли той переходить у наступний клас;

– учні бачать тести з тем свого класу;

– учні проходять тести з тем;

– учні, авторизувавшись під своїм паролем продивляються свої результати тесту.

2.3 Розробка архітектури веб – тренажеру

Розроблюваний тренажер реалізується у якості веб-додатку, який складається з двох частин: клієнту та серверу. У якості клієнта виступає стандартний браузер (Google Chrome, Opera, Mozilla Firefox та інші), а серверу - веб-сервер. Логіка веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі.

Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є кросплатформенними сервісами.

У якості програмного інструментарію для реалізації веб-тренажеру з граматики української мови було обрано фреймворк Django. Django (Джанго, ['dʒæŋgoʊ] [7]) – вільний фреймворк для веб-додатків на мові Python.

Архітектура Django схожа на модель MVC (Model – View – Controller) ([**Ошибка! Источник ссылки не найден.**]) – схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Model – частина доступу до даних, обслуговується системою баз даних Django.

View – частина, яка займається вибором того, що саме і як саме відображати, в Django цим займаються представлення (views) і шаблони (templates).

Controller – частина, яка визначає представлення в залежності від вказівок користувача, цим займається сам фреймворк, слідуючи налаштуванням URLconf і викликаючи відповідну функцію Python для заданого URL-а.

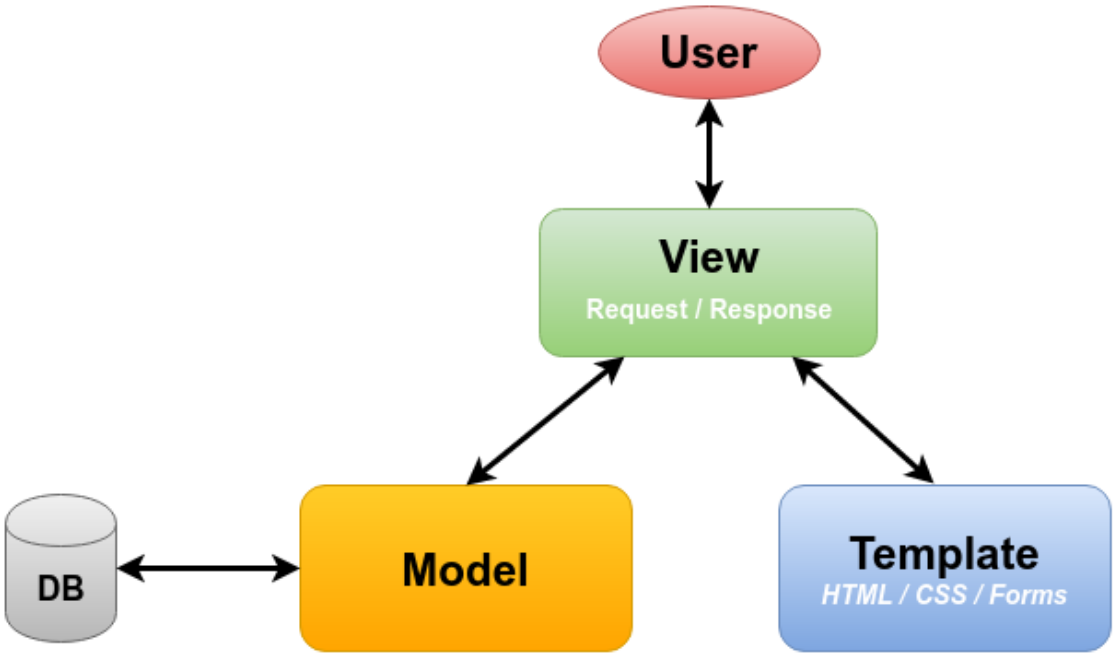


Рисунок 2.2 – Архітектура MTV

Контролер класичної моделі MVC приблизно відповідає рівню, який в Django називається Представлення (англ. View), а презентаційна логіка Представлення реалізується в Django рівнем Шаблонів (англ. Template). Через це рівневу архітектуру Django часто називають «Модель-Шаблон-Представлення» (MTV).

Таким чином, враховуючи усе вищесказане, можемо зробити висновок, що для розроблюваного додатку було обрано архітектуру «клієнт-сервер».

2.4 Розробка інформаційної моделі веб – тренажера

Інформаційна модель сайту має ієрархічну структуру у вигляді дерева, коренем якої виступає головна сторінка. На головній сторінці розміщуються посилання на сторінки другого рівня, а саме:

– інструкція з використання;

– правила з граматики;

– завдання тренажеру для різних класів;

– цікавинки.

Сторінками третього рівня виступають сторінки з теоретичним матеріалом з граматики та завданнями тренажеру.

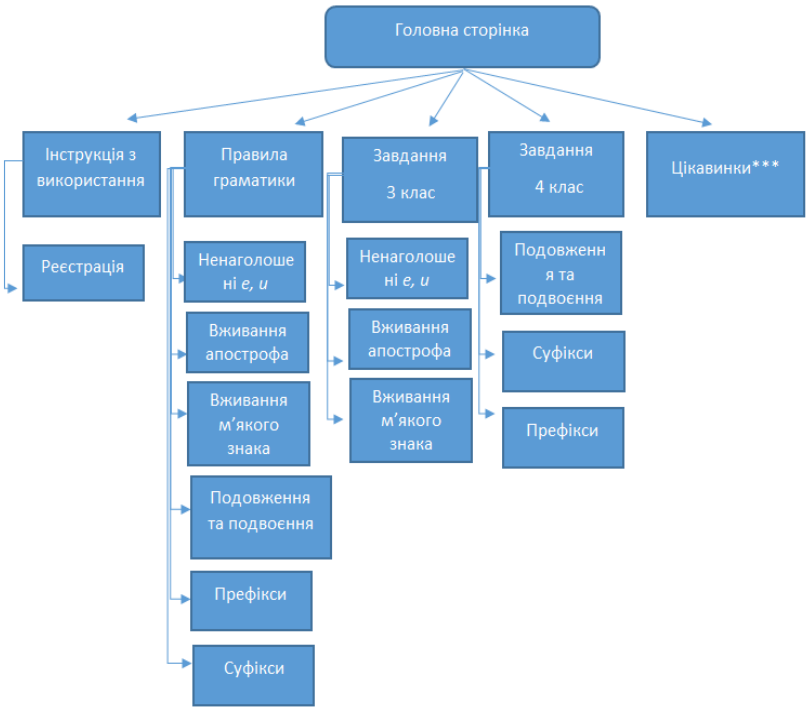


Рисунок 2.3 – Модель взаємодії структурних елементів сайту

Взаємодію елементів сайту показано на рисунку 2.3.

Збереження контенту сайту реалізується у базі даних системи керування даними SQLite. Таблиці бази даних – це об'єкти, які задані класами на мові програмування Python. Це еквівалент SQL запиту CREATE TABLE, але записаний в Python замість SQL і включає в себе більше, ніж просто опис

колонок таблиці. Django використовує моделі для виконання SQL запитів на фоні роботи програми, і повертає дані, що представляють таблиці бази даних, але у вигляді, який може бути оброблений Python.

Така технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних» називається ORM (англ. Object-Relational Mapping, укр. об'єктно-реляційне відображення, або перетворення) [9].

В Django це реалізовано саме таким чином з кількох причин. Перевірка даних вимагає додаткових витрат ресурсів і не може бути досконалою. Для того, що б надати зручний API для доступу до баз даних Django потрібно якось з'ясувати її формат, і щоб зробити це існує два способи. Перший – описати дані безпосередньо в Python, а другий – виконувати таку перевірку під час роботи програми

Другий спосіб простіший, тому що метадані про таблиці розташовані в одному місці, проте він може викликати кілька складнощів. По-перше – перевірка бази даних під час роботи програми вимагає додаткової витрати ресурсів. Якщо фреймворку доведеться виконувати таку перевірку при обробці кожного запиту, або при старті веб-сервера – це буде невиправдано високе навантаження (хоча деякі вважають її допустимою - розробники Django намагаються зменшити навантаження настільки, наскільки це можливо). По-друге – деякі сервери баз даних, особливо старі версії MySQL, не зберігають достатню кількість метаданих для повної перевірки.

Зберігання моделей даних в коді замість бази даних робить більш простим реалізацію контролю версій. Існує можливість відстежувати змінення в моделях даних.

SQL підтримує тільки кілька типів метаданих в моделі даних. Більшість баз даних, наприклад, не пропонують спеціального типу сховищ для подання адрес електронної пошти або URL-ів, тоді як в Django це реалізовано. Переваги високорівневих моделей даних роблять роботу коду більш продуктивною.

SQL часто несумісний на різних платформах. Для розповсюдження веб-додатку набагато зручніше надавати модуль на Python, який би описував модель даних, ніж окремі набори запитів типу CREATE TABLE для MySQL, PostgreSQL і SQLite.

Недолік такого підходу – це вірогідність "розсинхронізації" коду на Python з реальною схемою бази даних. Якщо робляться зміни в моделі Django необхідно так само вносити зміни в саму базу даних, щоб вона відповідала моделі.

Таким чином, проектування схеми збереження даних починається з визначення моделей. На діаграмі показано моделі, визначені для системи тренажерів, а саме:

– користувач (User);

– дозвіл (Permission);

– категорія користувача (Group);

– учень (Student);

– тест (Quize);

– питання (Question);

– відповідь (Answer);

– тема (Subject);

На діаграмі у додатку А показані залежності між моделями. Модель ористувач (абстрактний клас Users) є базовим класом для категорій користувачів, а саме: власники тестів (вчителі) та студенти, які можуть пройти тест. Об’єкт Student знаходиться у відношенні TakenQuize об’єктом Quize, а з об’єктом Answer – StudentAnswer. Назви властивостей моделей також наведено на діаграмі.

Спроектовані моделі у фреймворку Django описують структуру використовуваних даних. Використовувані в програмі дані зберігаються в базі даних, і за допомогою моделей здійснюється взаємодія з базою даних. За замовчуванням Django в якості бази даних використовує SQLite. Вона дуже

проста у використанні і не вимагає запущеного сервера. Всі файли бази даних можуть легко переноситися з одного комп'ютера на інший.

Кожна модель співставляється з певною таблицею в базі даних. Django використовує міграції для перенесення змін в моделях (додавання поля, видалення моделі і т.д.) на структуру бази даних.

Міграції створюються командою:

python manage.py makemigrations.

Після цього в папці проекту migrations створюється файл з міграціями, вміст якого наведено у додатку Б.

Для виконання міграцій використовується команда:

python manage.py migrate.

Після цього у системі керування базами даних SQLite буде створено таблиці бази даних проекту.

3 РОЗРОБКА ВЕБ-РЕСУРСУ

3.1 Вибір мовного інструментарію для реалізації веб-додатку

Клієнтська частина розроблюваного веб-додатку, т.т. його графічний інтерфейс, реалізується засобами мови гіпертекстової розмітки HTML (HyperText Markup Language). Оформлення веб-сторінок, їх стиль і колірна схема описуються в таблицях стилів – CSS.

Як вже зазначалось раніше, для реалізації серверної частини було обрано мову програмування Python [11] – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробки і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, в тому числі структурний, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних. Код в Python організовується у функції та класи, які можуть об'єднуватися в модулі (вони в свою чергу можуть бути об'єднані в пакети).

Інтерпретатор Python реалізовано практично на всіх платформах і операційних системах. Першою такою мовою була C, проте її типи даних на різних машинах могли займати різну кількість пам'яті і це служило деякою перешкодою при написанні програми. Python ж таким недоліком не володіє.

Наступна важлива риса Python – розширюваність мови. Це означає, що є можливість вдосконалення мови усіма зацікавленими програмістами. Також наявне велике число модулів, що підключаються до програми та забезпечують різні додаткові можливості.

Для організації структури проекту на Python у якості програмної платформи було обрано фреймворк Django. Це дуже потужний багаторівневий фреймворк останнього покоління з відкритим вихідним кодом.

Клієнтська частина додатку реалізється засобами мови гіпертекстової розмітки HTML 5.0 та таблиць каскадних стилів CSS.

3.2 Розробка програмного коду

Розроблюваний веб-додаток у Django отримує доступ і керуватиме даними через об'єкти Python, які називаються моделями. Моделі визначають структуру даних, що зберігаються. Вони реалізуються як підкласи django.db.models.Model, і можуть включати поля, методи і метадані.

Модель може мати довільну кількість полів будь-якого типу – кожен представляє стовпець даних, який зберыгатиметься в одній з таблиць бази даних. Кожен запис (рядок) бази даних буде складатися з одного значення кожного поля.

Програмний код моделі тест (Quize):

```
class Quiz(models.Model):
owner = models.ForeignKey(User, on_delete=models.CASCADE, related_name='quizzes')
name = models.CharField(max_length=255)
subject = models.ForeignKey(Subject, on_delete=models.CASCADE, related_name='quizzes')
    def __str__(self): return self.name
```

Модель також має методи. В даній моделі визначено стандартний метод класу для Python __str __ (), щоб повернути рядок для об'єкту, який використовується для представлення окремих записів на сайті адміністрування Цей метод повертає поле імені з моделі.

Імена усіх інших класів системи:

– class User;

– class Subject;

– class Quiz;

– class Question;

– class Answer;

– class Student;

– class TakenQuiz;

– class StudentAnswer;

Програмний код цих моделей проекту представлено у додатку В.

3.3 Розробка інструкцій з використання веб – тренажеру

Робота з проектом починається з завантаження головної сторінки сайту (рисунок 3.1), на якому користувач може або зареєструватись, або авторизуватись, як вчитель, або як учень.

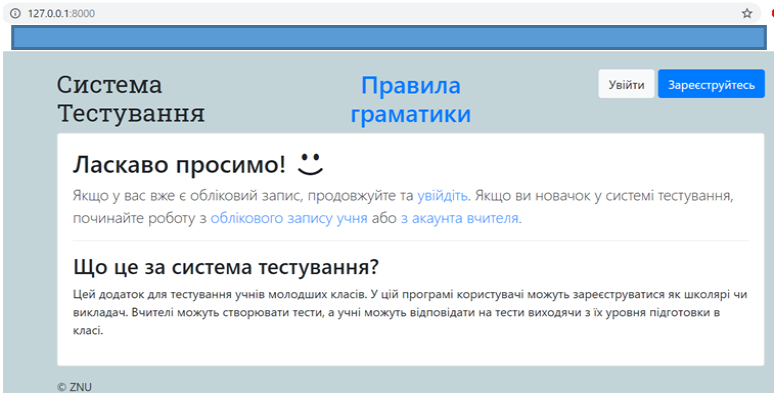


Рисунок 3.1 – Головна сторінка додатку

Форми реєстрації вчителів и учнів показано на рис.3.2, 3.3.

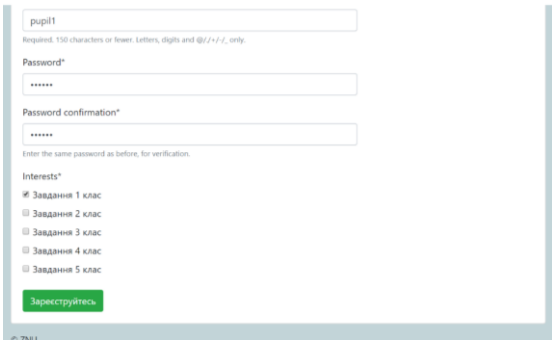


Рисунок 3.2 – Форма реєстрації учня

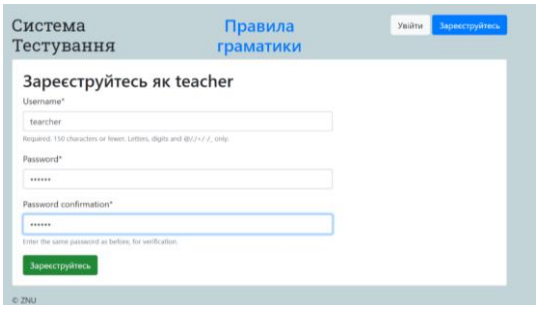


Рисунок 3.3 – Форма реєстрації вчителів

Вчитель, авторизувавшись під власними логіном та паролем, додає тему та вводить питання тесту, вказуючи вірні відповіді (рис.3.4, 3.5).

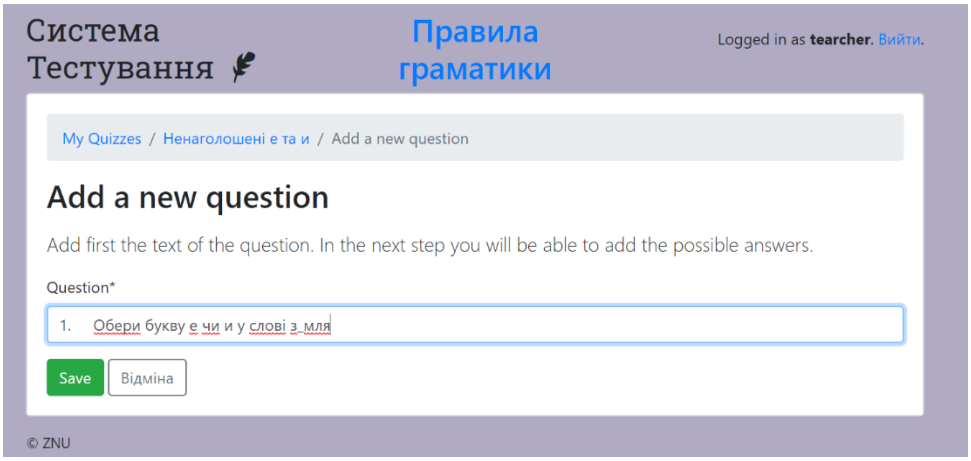


Рисунок 3.4 – Введення нового питання тесту

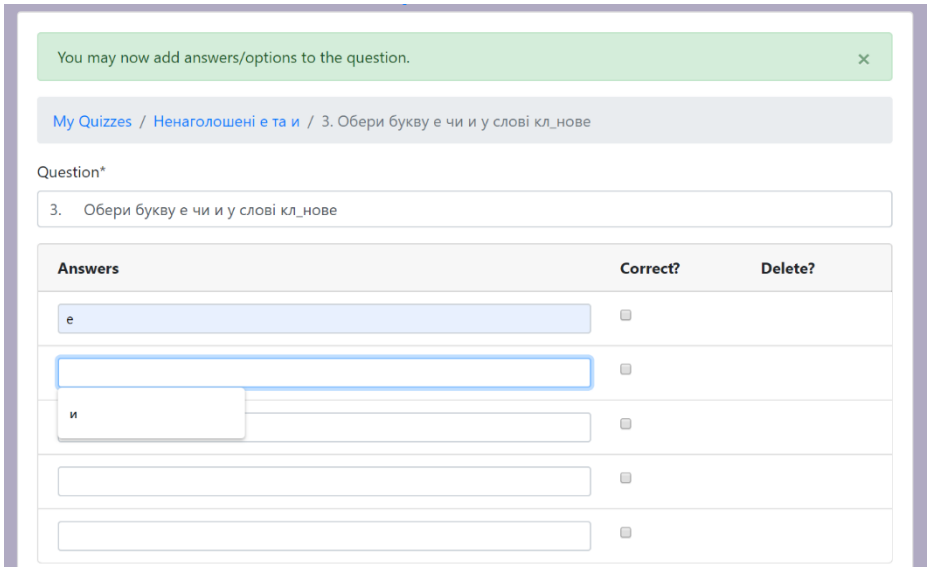


Рисунок 3.5 – Форма вводу варіантів відповіді

Учень, після авторизації на сайті, ознайомлюється з теоретичним матеріалом з граматики української мови (рис. 3.6) та проходить тесте, розташовані у вкладці «New», після проходження тестів учень ознайомлюється з результатами власного тестування, переходячи на вкладку «Taken» (рис.3.7, 3.9).

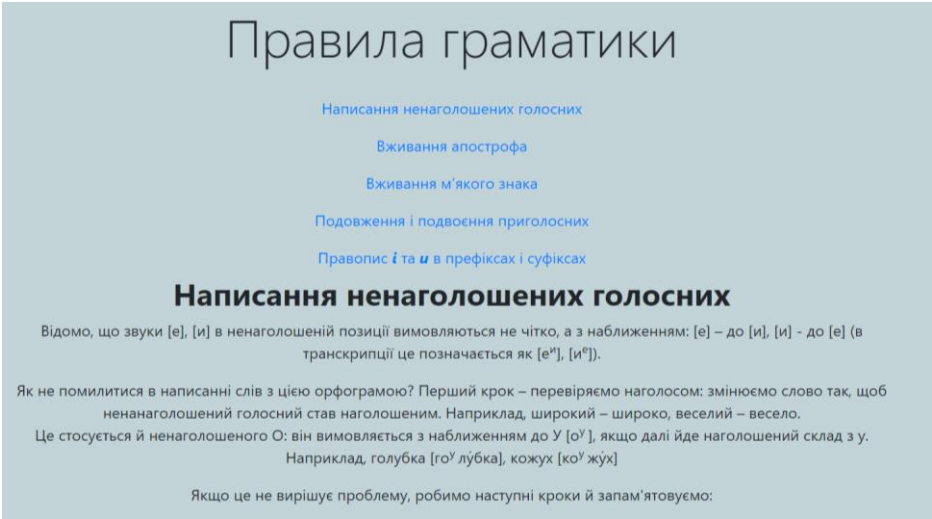


Рисунок 3.6 – Теоретичний матеріал з граматики

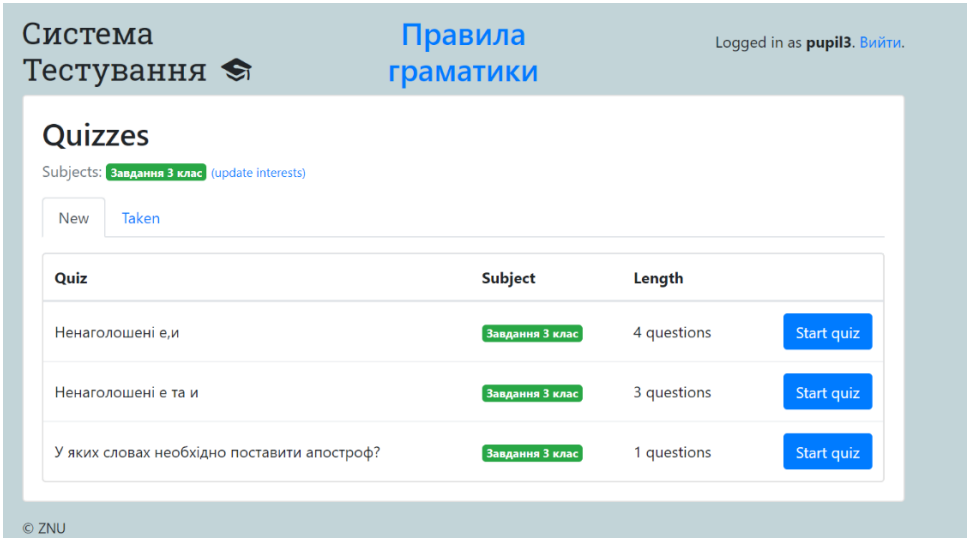


Рисунок 3.7 – Перелік тестів

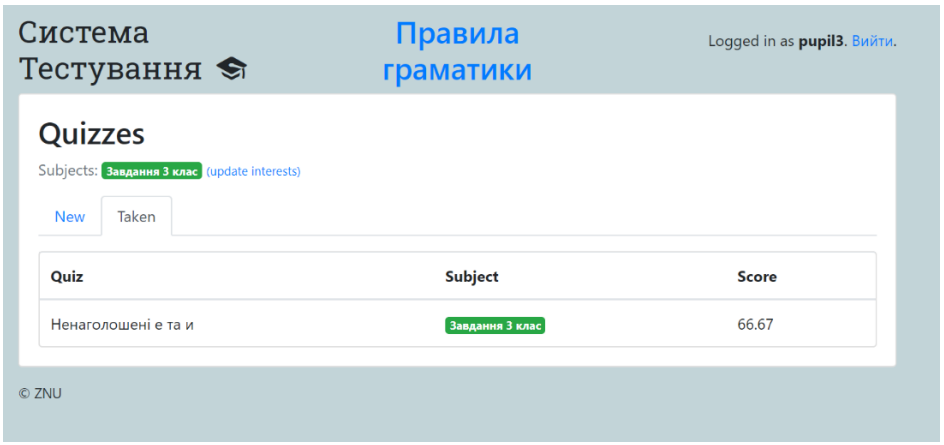


Рисунок 3.9 – Результати тестів

Після проходження тесту учнями вчитель може продивитись їх (рис.3.10).

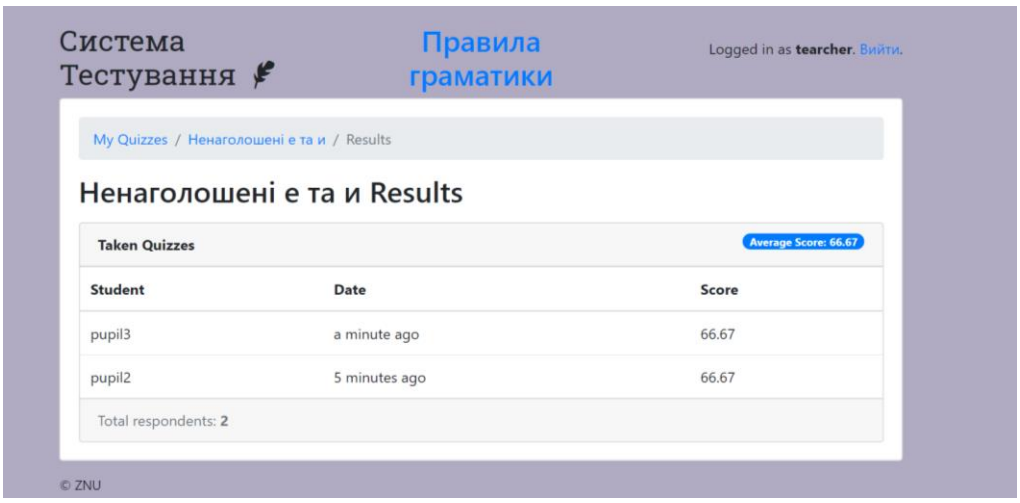


Рисунок 3.10 – Результати тесту

Представлені у інструкції форми мають тестовий характер, при використанні на практиці, учні и вчителя реєструються під власними реальними прізвищами, іменами та по батькові.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено тренажер для використання у якості допоміжного засобу при вивченні граматики української мови у початкових класах середньої школи.

Тренажер реалізовано у якості веб-додатку, який використовується через будь-який стандартний веб-браузер. Має зрозумілий графічний інтерфейс, тому може бути використаний учнями молодших класів самостійно без участі вчителя або батьків.

Розроблене програмне забезпечення впроваджуватиметься в експлуатацію учнями після зимових канікул. Очікується, що використання тренажеру значно підвищить запам’ятовування правил української граматики учнями початкової школи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вашуленко М. С. Методика навчання української мови в початковій школі : навчально – методичний посібник для студентів вищих навчальних закладів. Київ : Літера ЛТД, 2011. 364 с.

2. Андрощук А. С. Разработка компьютерных тренажеров для обучения персонала Государственной пограничной службы Украины // *Вісн. Національної академії Державної прикордонної служби України*. 2014. № 1. С.3.

3. И. Шапошников Web-сервисы Microsoft .NET Санкт-Петербург : БХВ 2002. С. 28-34.

4. Онлайн-тест словникового запасу. URL: http://n-slovo.com.ua/ (дата звернення: 5.11.2019)

5. Тренажер з правопису української мови. URL : https://webpen.com.ua/ (дата звернення: 5.11.2019)

6. Інтерактивний додаток з української літератури. URL: https://learningapps.org/ (дата звернення: 5.11.2019)

7. Bissex P., Wesley J Chun. Python Web Development with Django. Sebastopol : O'Reilly Media, 2009. 408p.

8. Glenn E. K, Stephen T. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80 // *Journal of Object-Oriented Programming.* V 1. Issue 3.1988. P. 26–48.

9. ORM. URL: https://ru.bmstu.wiki/ORM_(Object-Relational_Mapping) (дата звернення: 10.11.2019)

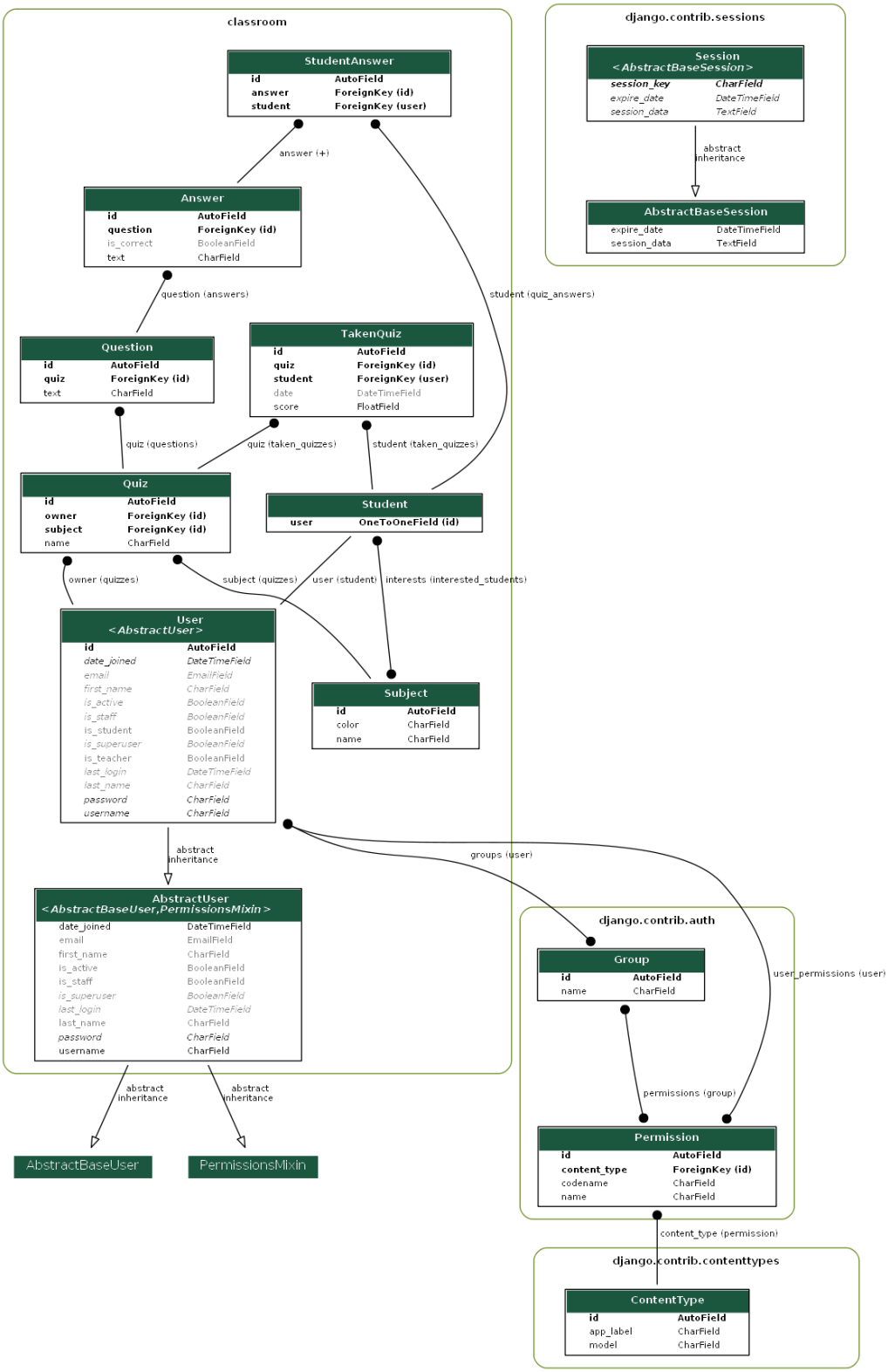
10. Harry J.W. Test-Driven Development with Python. Sebastopol : O'Reilly Media, 2014. 408p.

11. Марк Л. Программирование на Python. Санкт-Петербург : Символ-Плюс, 2011. 992 с.

12. Django Girls Tutorial.URL : https://tutorial.djangogirls.org/ru/django_orm/ (дата звернення : 24.11.2019)

ДОДАТОК А

UML діаграма асоціацій



ДОДАТОК Б

Міграції проекту

```
import django.contrib.auth.models
import django.contrib.auth.validators
import django.db.models.deletion
import django.utils.timezone
from django.conf import settings
from django.db import migrations, models
class Migration(migrations.Migration):
    initial = True
    dependencies = [
        ('auth', '0009_alter_user_last_name_max_length'),
    ]

    operations = [
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128,
verbose_name='password')),
                ('last_login', models.DateTimeField(blank=True, null=True,
verbose_name='last login')),
                ('is_superuser', models.BooleanField(default=False, help_text='Designates
that this user has all permissions without explicitly assigning them.',
verbose_name='superuser status')),
```

```
            ('username', models.CharField(error_messages={'unique': 'A user with that username already exists.'}, help_text='Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.', max_length=150, unique=True, validators=[django.contrib.auth.validators.UnicodeUsernameValidator()], verbose_name='username')),
            ('first_name', models.CharField(blank=True, max_length=30, verbose_name='first name')),
            ('last_name', models.CharField(blank=True, max_length=150, verbose_name='last name')),
            ('email', models.EmailField(blank=True, max_length=254, verbose_name='email address')),
            ('is_staff', models.BooleanField(default=False, help_text='Designates whether the user can log into this admin site.', verbose_name='staff status')),
            ('is_active', models.BooleanField(default=True, help_text='Designates whether this user should be treated as active. Unselect this instead of deleting accounts.', verbose_name='active')),
            ('date_joined', models.DateTimeField(default=django.utils.timezone.now, verbose_name='date joined')),
            ('is_student', models.BooleanField(default=False)),
            ('is_teacher', models.BooleanField(default=False)),
        ],
        options={
            'verbose_name': 'user',
            'verbose_name_plural': 'users',
            'abstract': False,
        },
        managers=[
            ('objects', django.contrib.auth.models.UserManager()),
        ],
    ),
```

```
        migrations.CreateModel(
            name='Answer',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('text', models.CharField(max_length=255, verbose_name='Answer')),
                ('is_correct', models.BooleanField(default=False, verbose_name='Correct
answer')),
            ],
        ),
        migrations.CreateModel(
            name='Question',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('text', models.CharField(max_length=255, verbose_name='Question')),
            ],
        ),
        migrations.CreateModel(
            name='Quiz',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=255)),
            ],
        ),
        migrations.CreateModel(
            name='StudentAnswer',
            fields=[
```

```
            ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
            ('answer',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='+', to='classroom.Answer')),
        ],
    ),
    migrations.CreateModel(
        name='Subject',
        fields=[
            ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
            ('name', models.CharField(max_length=30)),
            ('color', models.CharField(default='#007bff', max_length=7)),
        ],
    ),
    migrations.CreateModel(
        name='TakenQuiz',
        fields=[
            ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
            ('score', models.FloatField()),
            ('date', models.DateTimeField(auto_now_add=True)),
            ('quiz',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='taken_quizzes', to='classroom.Quiz')),
        ],
    ),
    migrations.CreateModel(
        name='Student',
```

```
            fields=[
                ('user',
models.OneToOneField(on_delete=django.db.models.deletion.CASCADE,
primary_key=True, serialize=False, to=settings.AUTH_USER_MODEL)),
                ('interests', models.ManyToManyField(related_name='interested_students',
to='classroom.Subject')),
            ],
        ),
        migrations.AddField(
            model_name='quiz',
            name='owner',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='quizzes', to=settings.AUTH_USER_MODEL),
        ),
        migrations.AddField(
            model_name='quiz',
            name='subject',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='quizzes', to='classroom.Subject'),
        ),
        migrations.AddField(
            model_name='question',
            name='quiz',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='questions', to='classroom.Quiz'),
        ),
        migrations.AddField(
            model_name='answer',
            name='question',
```

```
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='answers', to='classroom.Question'),
        ),
        migrations.AddField(
            model_name='user',
            name='groups',
            field=models.ManyToManyField(blank=True, help_text='The groups this
user belongs to. A user will get all permissions granted to each of their groups.',
related_name='user_set', related_query_name='user', to='auth.Group',
verbose_name='groups'),
        ),
        migrations.AddField(
            model_name='user',
            name='user_permissions',
            field=models.ManyToManyField(blank=True, help_text='Specific
permissions for this user.', related_name='user_set', related_query_name='user',
to='auth.Permission', verbose_name='user permissions'),
        ),
        migrations.AddField(
            model_name='takenquiz',
            name='student',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='taken_quizzes', to='classroom.Student'),
        ),
        migrations.AddField(
            model_name='studentanswer',
            name='student',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='quiz_answers', to='classroom.Student'),
        ),
```

```
        migrations.AddField(
            model_name='student',
            name='quizzes',
            field=models.ManyToManyField(through='classroom.TakenQuiz',
to='classroom.Quiz'),
        ),
    ]
```

ДОДАТОК В

Моделі проекту

```
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.utils.html import escape, mark_safe


class User(AbstractUser):
    is_student = models.BooleanField(default=False)
    is_teacher = models.BooleanField(default=False)


class Subject(models.Model):
    name = models.CharField(max_length=30)
    color = models.CharField(max_length=7, default='#007bff')
    def __str__(self):
        return self.name
    def get_html_badge(self):
        name = escape(self.name)
        color = escape(self.color)
        html = '<span class="badge badge-primary" style="background-color:
%s">%s</span>' % (color, name)
        return mark_safe(html)
class Quiz(models.Model):
    owner = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='quizzes')
    name = models.CharField(max_length=255)
```

```
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE,
related_name='quizzes')
    def __str__(self):
        return self.name
class Question(models.Model):
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE,
related_name='questions')
    text = models.CharField('Question', max_length=255)
    def __str__(self):
        return self.text
class Answer(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE,
related_name='answers')
    text = models.CharField('Answer', max_length=255)
    is_correct = models.BooleanField('Correct answer', default=False)
    def __str__(self):
        return self.text
class Student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
primary_key=True)
    quizzes = models.ManyToManyField(Quiz, through='TakenQuiz')
    interests = models.ManyToManyField(Subject,
related_name='interested_students')

    def get_unanswered_questions(self, quiz):
        answered_questions = self.quiz_answers \
            .filter(answer__question__quiz=quiz) \
            .values_list('answer__question__pk', flat=True)
        questions = quiz.questions.exclude(pk__in=answered_questions).order_by('text')
        return questions
```

```
    def __str__(self):
        return self.user.username


class TakenQuiz(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE,
related_name='taken_quizzes')
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE,
related_name='taken_quizzes')
    score = models.FloatField()
    date = models.DateTimeField(auto_now_add=True)


class StudentAnswer(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE,
related_name='quiz_answers')
    answer = models.ForeignKey(Answer, on_delete=models.CASCADE,
related_name='+')
```

ДОДАТОК Г

Forms

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.db import transaction
from django.forms.utils import ValidationError

from classroom.models import (Answer, Question, Student, StudentAnswer,
                              Subject, User)


class TeacherSignUpForm(UserCreationForm):
    class Meta(UserCreationForm.Meta):
        model = User

    def save(self, commit=True):
        user = super().save(commit=False)
        user.is_teacher = True
        if commit:
            user.save()
        return user


class StudentSignUpForm(UserCreationForm):
    interests = forms.ModelMultipleChoiceField(
        queryset=Subject.objects.all(),
        widget=forms.CheckboxSelectMultiple,
```

```
        required=True
    )

    class Meta(UserCreationForm.Meta):
        model = User

    @transaction.atomic
    def save(self):
        user = super().save(commit=False)
        user.is_student = True
        user.save()
        student = Student.objects.create(user=user)
        student.interests.add(*self.cleaned_data.get('interests'))
        return user


class StudentInterestsForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ('interests', )
        widgets = {
            'interests': forms.CheckboxSelectMultiple
        }


class QuestionForm(forms.ModelForm):
    class Meta:
        model = Question
        fields = ('text', )
```

```
class BaseAnswerInlineFormSet(forms.BaseInlineFormSet):
    def clean(self):
        super().clean()

        has_one_correct_answer = False
        for form in self.forms:
            if not form.cleaned_data.get('DELETE', False):
                if form.cleaned_data.get('is_correct', False):
                    has_one_correct_answer = True
                    break
        if not has_one_correct_answer:
            raise ValidationError('Mark at least one answer as correct.',
code='no_correct_answer')


class TakeQuizForm(forms.ModelForm):
    answer = forms.ModelChoiceField(
        queryset=Answer.objects.none(),
        widget=forms.RadioSelect(),
        required=True,
        empty_label=None)

    class Meta:
        model = StudentAnswer
        fields = ('answer', )
    def __init__(self, *args, **kwargs):
        question = kwargs.pop('question')
        super().__init__(*args, **kwargs)
        self.fields['answer'].queryset = question.answers.order_by('text')
```

ДОДАТОК Д

Decorators

```
from django.contrib.auth import REDIRECT_FIELD_NAME
from django.contrib.auth.decorators import user_passes_test


def student_required(function=None,
redirect_field_name=REDIRECT_FIELD_NAME, login_url='login'):
    '''
    Decorator for views that checks that the logged in user is a student,
    redirects to the log-in page if necessary.
    '''
    actual_decorator = user_passes_test(
        lambda u: u.is_active and u.is_student,
        login_url=login_url,
        redirect_field_name=redirect_field_name
    )
    if function:
        return actual_decorator(function)
    return actual_decorator


def teacher_required(function=None,
redirect_field_name=REDIRECT_FIELD_NAME, login_url='login'):
    '''
    Decorator for views that checks that the logged in user is a teacher,
    redirects to the log-in page if necessary.
```

```
'''
actual_decorator = user_passes_test(
    lambda u: u.is_active and u.is_teacher,
    login_url=login_url,
    redirect_field_name=redirect_field_name
)
if function:
    return actual_decorator(function)
return actual_decorator
```

ДОДАТОК Е

Urls

```
from django.urls import include, path
from django.views.generic import TemplateView
from .views import classroom, students, teachers

urlpatterns = [
    path('', classroom.home, name='home'),

    path('about',
         TemplateView.as_view(template_name='about_us.html'),
         name='about'),

    path('students/', include(([
        path('', students.QuizListView.as_view(), name='quiz_list'),
        path('about', TemplateView.as_view(template_name='about_us.html'),
name='about'),
        path('interests/', students.StudentInterestsView.as_view(),
name='student_interests'),
        path('taken/', students.TakenQuizListView.as_view(), name='taken_quiz_list'),
        path('quiz/<int:pk>/', students.take_quiz, name='take_quiz'),
    ], 'classroom'), namespace='students')),

    path('teachers/', include(([
        path('', teachers.QuizListView.as_view(), name='quiz_change_list'),
        path('quiz/add/', teachers.QuizCreateView.as_view(), name='quiz_add'),
```

```
    path('about', TemplateView.as_view(template_name='about_us.html'),
name='about'),
    path('quiz/<int:pk>/', teachers.QuizUpdateView.as_view(),
name='quiz_change'),
    path('quiz/<int:pk>/delete/', teachers.QuizDeleteView.as_view(),
name='quiz_delete'),
    path('quiz/<int:pk>/results/', teachers.QuizResultsView.as_view(),
name='quiz_results'),
    path('quiz/<int:pk>/question/add/', teachers.question_add, name='question_add'),
    path('quiz/<int:quiz_pk>/question/<int:question_pk>/',
teachers.question_change, name='question_change'),
    path('quiz/<int:quiz_pk>/question/<int:question_pk>/delete/',
teachers.QuestionDeleteView.as_view(), name='question_delete'),
  ], 'classroom'), namespace='teachers')),
]
```