**АННОТАЦІЯ**

Ця бакалаврська кваліфікаційна робота призначена для розробки серверної частини програмного забезпечення для організації комунікації викладача зі студентом на лекції зі застосування сучасних IT-технологій.

Під час роботи над бакалаврською кваліфікаційною роботою було досліджено різноманітні IT-засоби для організації комунікації студентів з викладачем на лекції, виявлено основні переваги і недоліки існуючих засобів, на основі яких було створено вимоги до програмного забезпечення, які було формалізовано у специфікації вимог. Також, для забезпечення належного і правильно функціонування, було проведено тестування розробленого програмного забезпечення за допомогою модульного тестування, а також статичного аналізу коду.

Крім цього, для оцінки економічної доцільності розробленого продукту, було проведено економічний аналіз, який показав, що займатися розробкою цього програмного забезпечення доцільно і допоміг вибрати стратегію для виходу на ринок.

У майбутньому планується створити відкритий API для того, щоб інші розробники могли створювати свої клієнти для даного серверного застосування, а також розширення функцій, наприклад, персоналізація кожного студента, створення електронного щоденника відвідування та інші функції, які будуть запропоновані користувачами даної системи.

Загальний обсяг роботи: 95 сторінок

**ABSTRACT**

This bachelor qualification work is intended for developing backend software that provide students and teacher communication on lecture using modern IT-technologies.

During working on this bachelor qualification work, was explored various of IT-tools for organizing students and teacher communication on lecture, revealed main advantages and disadvantages of existing solutions, based on them, was created requirements for software, that was formalized in requirements specification. Also, for providing adequate and properly functioning, was performed testing of developed software using unit-tests and static code analysis.

Furthermore, for estimation of economic feasibility of this software, was conducted economic analyses, that show that developing of this software is advisable and help to choose strategy for entering market.

In the future is planned to create open API for providing ability to other developers to create own client-side applications for this backend software, and also adding new features such as: identification for every student, creating the digital journal of attending or other features that will be offered by the software users.

The total amount of the work are 95 pages.

**ЗМІСТ**

[вступ 7](#_Toc483891696)

[Розділ 1. Огляд IT – засобів для організації комунікації студентів з викладачем на лекції 8](#_Toc483891697)

[1.1 Огляд предметної області 8](#_Toc483891698)

[1.2 Висновки до розділу аналізу існуючих засобів 12](#_Toc483891699)

[Розділ 2. Постановка завдання для розробки програмного забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина) 13](#_Toc483891700)

[2.1 Постановка завдання 13](#_Toc483891701)

[2.2 Специфікація вимог до програмного забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина) 14](#_Toc483891702)

[2.3 Висновки до розділу постановка задачі та специфікації вимог 21](#_Toc483891703)

[РОЗДІЛ 3. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ЗАСТОСУВАННЯ 22](#_Toc483891704)

[3.1 Проектування архітектури 22](#_Toc483891705)

[3.2 Вибір інструментальних засобів для реалізації веб-системи 23](#_Toc483891706)

[3.3 Проектування класів веб-системи 25](#_Toc483891707)

[3.4 Проектування бази даних 26](#_Toc483891708)

[3.3 Проектування графічного інтерфейсу користувача 27](#_Toc483891709)

[3.4 Висновки до розділу проектування та архітектури 29](#_Toc483891710)

[4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ КОМУНІКАЦІЇ СТУДЕНТІВ З ВИКЛАДАЧЕМ НА ЛЕКЦІЇ (СЕРВЕРНА ЧАСТИНА) 30](#_Toc483891711)

[4.1 Реалізація веб-системи 30](#_Toc483891712)

[4.2 Опис роботи з веб-системою 34](#_Toc483891713)

[4.3 Тестування веб-системи 39](#_Toc483891714)

[4.4 Висновки розділу реалізації і тестування веб-сервісу 42](#_Toc483891715)

[РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА 43](#_Toc483891716)

[5.1. Економічна характеристика проектного рішення (програмного продукту) 43](#_Toc483891717)

[5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару 44](#_Toc483891718)

[5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ 44](#_Toc483891719)

[5.4. Формування стратегічних альтернатив 46](#_Toc483891720)

[5.5. Бюджетування 48](#_Toc483891721)

[5.6. Вибір стратегії 53](#_Toc483891722)

[5.7 Висновки економічної оцінки веб-сервісу 54](#_Toc483891723)

[ВИСНОВКИ 55](#_Toc483891724)

[СПИСОК ОПРАЦЬОВАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 56](#_Toc483891725)

[ДОДАТОК А. ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ 57](#_Toc483891726)

[ДОДАТОК Б. ДІАГРАМА РОЗГОРТУВАННЯ 58](#_Toc483891727)

[ДОДАТОК В. ДІАГРАМА КЛАСІВ 59](#_Toc483891728)

[ДОДАТОК Д. ЗВІТ ПРО ТЕСТУВАННЯ 60](#_Toc483891729)

[Д.1 Вступ 60](#_Toc483891730)

[Д.2 Розробка Unit-тестів 60](#_Toc483891731)

[Д.3 Розробка тестових випадків 62](#_Toc483891732)

[Д.4 Результати тестування 68](#_Toc483891733)

[Д.5 Висновок 75](#_Toc483891734)

[ДОДАТОК Е. ІНСТРУКЦІЯ КОРИСТУВАЧА 76](#_Toc483891735)

[Е.1 Компоненти ПЗ 76](#_Toc483891736)

[Е.2 Встановлення ПЗ 76](#_Toc483891737)

[Е.3 Налаштування ПЗ 76](#_Toc483891738)

[Е.4 Базові функції ПЗ 77](#_Toc483891739)

[Е. 5 Обробка помилкових ситуацій 78](#_Toc483891740)

[ДОДАТОК Ж. ФРАГМЕНТИ КОДУ ПРОГРАМИ 79](#_Toc483891741)

# 

# вступ

Однією з проблем викладання лекцій є відсутність будь-якої комунікації студента з викладачем – зараз усе відбувається за моделлю пасивного слухання: викладач приходить на лекції, розповідає про матеріал, а студенти просто слухають і конспектують усе і зрідка ставлять питання викладачу. Тобто, відсутня будь-яка комунікація і взаємодія відбувається лише в одну сторону: від викладача до студента.

Недоліком такого підходу є те, що викладач не отримує жодного відгуку про власний лекційний матеріал, не може отримати думки студентів про ситуацію ведення лекції в цілому, тобто він не може дізнатися чи зрозуміли студенти викладений матеріал чи ні, які питання потребують більш широко висвітлення, які ж навпаки є представленні достатньо детально.

Для студента такий підхід також створює проблеми, адже він не моє змоги оцінити матеріал, показати, що йому зрозуміло, а що ні.

Отже, метою бакалаврської кваліфікаційної роботи є розроблення серверного програмного забезпечення, яке б вирішувало перелічені вище проблеми, а саме дозволяло викладачам створювати питання для студентів з різними варіантами відповіді, студентам голосувати за різні варіанти відповіді над викладацькими питаннями, викладачу переглядати графік відповідей студентів у реальному часі, а також можливість студентам самим створювати питання для викладача.

# Розділ 1. Огляд IT – засобів для організації комунікації студентів з викладачем на лекції

1.1 Огляд предметної області

Огляд предметної області можна почати з опису поточних засобів для організації комунікації студентів з викладачем на лекції.

У сфері освіти для комунікації студентів з викладачем все більш популярним стає використання різноманітних технологій реагування [1]. Прикладом застосування технологій реагування є пристрій клікерс, який забезпечує альтернативний метод залучення студентів до комунікації в класі.

Клікерс - (“Clickers”, від назви пультів, (рис. 4.1), на яких потрібно клікати студентам) - це система інтерактивної взаємодії між студентами та викладачем на лекції, яка здійснюється за допомогою пультів, які є в студентів, бази, яка отримує та обробляє інформацію він пультів і знаходиться у лекційній аудиторі та спеціального програмного забезпечення [2].



Рис. 4.1. Пульт-клікер.

Зазвичай, подібні системи володіють такими можливостями:

* Викладач показує питання з декількома варіантами відповідей студентам на великому екрані проектора
* Кожен студент надсилає відповідь на питання використовуючи спеціальний пульт (клікер), який надсилає відповідь студента для обробки на спеціальний пристрій - приймач.
* Викладач разом з студентами у реальному часі дивиться на кругову діаграму з відповідями, як студенти голосують на питання і одразу обговорює їх.

Подібні системи дозволяють не лише отримувати відгук і відповіді у студентів, а й мають можливість задавати багато різних типів питань, ось перелічені типи деяких з них:

* Питання-нагадування – у цьому типі питання перевіряються залишкові знання студентів деяких фактів, концепцій та іншої інформації, що була подана на попередніх лекціях або курсах. За допомогою цього типу питання можна перевірити чи студенти запам’ятовують найважливіші пункти та ключові моменти з попередньо опрацьованого матеріалу.
* Питання на розуміння концептуальних речей – це питання на розуміння концептуальних речей, які були подані під час лекції, вони застосовуються для перевірки рівня розуміння студентів поточної лекції.
* Питання на застосування опрацьованого матеріалу – цей тип питань застосовується, коли потрібно перевірити рівень практичного розуміння матеріалу студентами, тобто наскільки у практиці студенти можуть використовувати знання здобуті на лекції.
* Питання на критичне мислення – даний тип питань використовується для оцінки того, як студенти аналізують зв’язки між декількома концепціями, викладеними у матеріалі або створюють оцінки на основі якогось практичного критерію.
* Питання на досвід студентів – ці питання дозволяють оцінити рівень студентів у тих матеріалах, у яких вони могли мати якийсь попередній досвід.
* Питання на рівень розуміння матеріалу – цим типом опитування можна оцінити рівень розуміння матеріалу, як студенти оцінюють себе самі.
* Питання на моніторинг – цей тип питань дозволяє швидко дізнатися, скільки студентів присутніх на лекції, скільки студентів завершили виконання самостійної роботи, приблизно заміряти час необхідний студентам для виконання якоїсь роботи, адже суть цього типу питань полягає у тому, що студенти коли завершують якусь роботу або іншу дію, клікають і викладач бачить кількість студентів, що вже готові до виконання наступних дій.
* Питання для перевірки гіпотез і соцопитування – цей тип питань використовується, коли потрібно провести соцопитування серед студентів, наприклад, як студенти ставляться до якихось соціальний або політичних подій.

Крім різних типів питань, система клікерс дозволяє також використовувати її у різних видах діяльності, наприклад:

* Електронна перекличка – студенти за допомогою пультів відповідають на питання «Ви на лекції?» і не має ніякої потреби у проведенні переклички за допомогою списку студентів.
* Отримання відгуку від студентів по лекції – дозволяє викладачу у реальному часі отримувати рівень розуміння студентами викладеного матеріалу і, за потреби, висвітлювати додаткові місця або пропускати те, що вже є зрозумілим.
* Оцінювання рівня студентів – можна проводити оцінювання рівня студентів, виставляти кожному бали за правильні відповіді.
* Проведення дискусій – викладач під час лекцій може поділити аудиторію на декілька груп, ставити різні питання і отримувати думки вже не конкретних студентів, а груп і одразу проводити дискусії з ними.
* Проведення лекцій, побудованих на питаннях студентів – це підхід, у якому викладач подає лише базові матеріали на лекції або взагалі виносить їх на самостійне опрацювання, а основу лекції створюють самі студенти – вони ставлять питання, а викладач одразу відповідає на отриманні питання [3].

Основні переваги використання клікерсів для викладача полягають у тому, що вони можуть:

* Підтримувати увагу студентів – дослідження показуюють що увага розсіюється після 10-18 хвилин пасивного слухання, а клікерс дозволяє залучати студентів до навчання – вони ставлять питання і це сприяє увазі від студентів.
* Сприяє активному залученні студентів на лекції – відповіді студентів на питання дозволяє більш глибоко засвоїти матеріал, для них лекції тепер не просто пасивне прослуховування.
* Сприяє дискусії і співпраці між студентами на лекції – якщо студенти працюють у групах, їм треба співпрацювати і доходити консенсусу у викладацьких питаннях, адже у такому випадку працює лише один клікер на всю групу.
* Дозволяє залучити усіх студентів на лекції – коли студенти відповідають піднімаючи руки, то тоді з цілої аудиторію працюють лише найбільш активні і часто можна дізнатися лише їх думки, клікерс дозволяє відповідати усім і одночасно.
* Допомагає виразити думку сором’язливим і невпевненим студентам – часто студенти соромляться відкрито відповідати або невпевнені у своїх відповідях і тому мовчать, дана система дозволяє усунути цю проблему.
* Дозволяє перевірити розуміння студентами матеріалу під час лекції – викладач може одразу побачити наскільки студенти розуміють прочитаний матеріал.
* Дозволяє спрямувати лекцію у правильне русло – якщо викладач бачить, що на питання більшість студентів відповідають неправильно, він може ще раз пояснити дану тема і навпаки, якщо всім усе зрозуміло, то можна не зациклюватися на якісь темі.
* Дозволяє провести електронну перекличку і тестування на оцінку – викладачу не треба робити переклички і він може давати завдання на лекції на додаткові бали.
* Додає інтерактивності на лекції – студентам і викладачу цікаво бачити, який варіант на графіку набирає найбільше голосів [4].

Однак, не зважаючи на всі переваги використання цієї системи, вона має свої недоліки. До недоліків пристрою клікерс можна віднести такі:

* висока вартість комплексу;
* складність вставлення комплексу;
* необхідність кожному студенту купувати собі пристрій клікер;
* необхідність у спеціальному навчанні викладачів;
* обмеження на питання – в основному, клавіатура клікерсів складається з 4 кнопок, тобто у відповіді може бути тільки 4 варіанти відповідей.

Також потрібно врахувати, що усі ці вищезгадані недоліки належать до клікерсів, які представляють собою фізичні пристрою. Вони вже досить застарілі і їм на заміну приходять клікерси нового покоління – електронні, які представляють собою набір програмного забезпечення і не вимагають додаткової покупки баз, пультів чи розсортування якоїсь додаткової інфраструктури [5].

До основних переваг електронних клікерсів є те, що вони містять у собі усі позитивні характеристики традиційних клікерсів і вирішують основні їх недоліки. Наприклад, у електронних клікерсів відсутня складність встановлення комплексу – адже немає ніякої необхідності у якихось додаткових фізичних пристроях, прийом і обробка інформації здійснюється на спеціально виділеному сервері, який також необов’язково використовувати, а можна орендувати. Крім цього, студентам не потрібно купувати фізичні пульти, усе, що необхідно зробити для них це лише скачати і установити відповідну мобільну аплікацію у себе на смартфоні. І одне з найголовніших, фізичні пульти накладають суттєве обмеження – не можна створювати більше 4 варіантів відповіді на питання, це зумовлене присутністю лише 4 кнопок на пультах. Однак мобільна аплікація знімає будь-які обмеження на кількість питань, а також дозволяє відображати додаткову інтерактивну інформацію на дисплеї смартфонів студентів поряд з питанням – зображення, відеоматеріали, аудіо матеріали, різного роду документи та таблиці.

1.2 Висновки до розділу аналізу існуючих засобів

Завершуючи розділ, можна зробити висновки, що було проаналізовано та описано предметну область IT-засобів для організації комунікації студентів з викладачем на лекції, визначено поточний стан, цілі та існуючі рішення для організації подібної взаємодії, а також визначено основні переваги та недоліки, які можна відтворити та вирішити у новій веб-системі.

# 

# Розділ 2. Постановка завдання для розробки програмного забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина)

# 2.1 Постановка завдання

На основі опису предметної області, можна сформувати постановку завдання: необхідно створити веб-систему, яка буде надавати усі можливості і переваги традиційних клікерсів але водночас позбавить їх недоліків.

Тобто, потрібно створити серверне забезпечення, яке дозволить здійснювати такі дії, як: створення різноманітних опитувань викладачем, а також голосування студентів на різні питання. Це дві найголовніші функції, які потрібно реалізувати у системі клікерс.

Також, веб-система повинна мати додаткові функції, наприклад, функції для управління лекціями та дисциплінами для викладача, адже це дозволить більш ефективно готуватися викладачу до лекцій – він може наперед створити питання для різних предметів.

Крім того, слід забезпечити просте створення питань і прямо під час лекцій, щоб викладач міг оперативно реагувати на думки аудиторії. Також, слід додати можливість студентам самим створювати питання для викладача – це чудова перевага інтерактивної взаємодії викладача і студента, якої немає на фізичних клікерсах і їх там неможливо реалізувати в принципі, однак використання веб-системи дозволяє мати подібні функції.

Вхідні дані для роботи: документація по мові програмування Python, веб-фреймоворку Django [6, 7], бібліотеці Django REST Framework [8], фреймоворку AngularJS, специфікація вимог.

Вихідні дані: результуюче застосування, яке реалізовує поставлену задачу.

# 2.2 Специфікація вимог до програмного забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина)

**2.2.1 Вступ**

**2.2.1.1 Призначення, мета**

Метою даного проекту є створення веб-сервісу для організації комунікації студентів з викладачем на лекції. Веб-сервіс повинен мати API для можливості комунікації веб-сайту для викладача та мобільної аплікації.

**2.2.2 Загальний опис**

**2.2.2.1 Характеристики продукту**

На високому рівні абстракції можна виділити наступні основні характеристики:

* студенти можуть голосувати із допомогою мобільного застосування;
* викладач має змогу управляти своїми дисциплінами та лекціями;
* викладач має змогу створювати запитання як попередньо, так і під час проведення лекції;
* учасники можуть переглядати результати опитування в реальному часі;
* можливість сформувати статистику по окремих запитаннях;
* можливість обмеження доступу до лекцій за паролем;
* створення питань студентами для обговорення на лекції у режимі реального часу.

**2.2.2.2 Класи користувачів**

Всього у системи буде 2 класи користувачів: викладачі та студенти. Викладачі – це користувачі, які займаються читанням лекцій, їхня основна потреба у отриманні відгуків від студентів. Студенти – це користувачі, які перебувають і вчаться на лекціях, думки яких хоче дізнатися викладач.

**2.2.2.3 Середовище функціонування**

Клієнтська частина вимагає веб-браузер з повною підтримкою JavaScript, HTML5, CSS3.

Серверна частина повинна функціонувати на операційній системі Ubuntu Server, з встановленою мовою програмування Python, і відповідними модулями, такими як Djagno та Django REST Framework.

**2.2.3 Характеристики системи**

1. **Реєстрація у системі викладача**
   1. Опис і пріоритет

Пріоритет – високий. Можливість реєстрації викладачів у системі.

* 1. Послідовності дія-відгук

1. Користувач відкриває сторінку реєстрації.
2. Користувач заповнює необхідні поля.
3. Система перевіряє коректність введеної інформації.
4. Якщо введена інформація некоректна, система повідомляє про це.
5. Користувач підтверджує реєстрацію.
6. Система повідомляє про результат реєстрації.
   1. Функціональні вимоги

REQ 1. Усі необхідні поля мають бути перевірені на коректність.

REQ 2. Довжина паролю: від 8 символів.

REQ 3. Унеможливити реєстрацію користувача з адресою електронної пошти не у домені lpnu.

1. **Вхід у систему**
   1. Опис і пріоритет

Пріоритет – високий. Можливість входу викладача у систему під своїм обліковим записом.

* 1. Послідовності дія-відгук

1. Користувач відкриває сторінку входу.
2. Користувач заповнює поле «Електронна пошта», «Пароль».
3. Користувач підтверджує вхід.
4. Система повідомляє про результат входу.
5. Якщо результат – успішний, система переадресовує користувача на сторінку з навчальними предметами.
   1. Функціональні вимоги.
6. **Додавання нового навчального предмету**
   1. Опис і пріоритет

Пріоритет – високий. Можливість додати новий начальний предмет, щоб створювати до нього лекції.

* 1. Послідовності дія-відгук

1. Користувач відкриває сторінку з списком навчальних предметів.
2. Користувач натискає кнопку «Створити».
3. Користувач вводить назву предмета.
4. Користувач вводить навчальні групи, для яких ведеться цей предмет.
5. Користувач натискає кнопку «Створити».
6. Система переходить на сторінку .з списком предметів вже з створеним предметом.
   1. Функціональні вимоги

REQ 1. Усі необхідні поля мають бути перевірені на коректність.

REQ 2. Можливість очистити введену інформацію.

REQ 3. Можливість видалити навчальний предмет і автоматично видалити усі лекції цього предмету.

REQ 4. Можливість редагувати інформацію про вже створені предмети.

1. **Додавання нової лекції**
   1. Опис і пріоритет

Пріоритет – високий. Можливість додати нову навчальну лекцію, щоб мати можливість створювати питання і отримувати питання.

* 1. Послідовності дія-відгук

1. Користувач відкриває сторінку з списком лекцій.
2. Користувач натискає кнопку «Створити».
3. Користувач вводить назву лекції.
4. Користувач вибирає навчальний предмет зі списку, до якого відноситься дана лекція.
5. Користувач вводить пароль до лекції.
6. Користувач вводить місце проведення лекції.
7. Користувач вводить номер заняття.
8. Користувач вибирає дату проведення лекції.
9. Користувач натискає кнопку «Створити».
10. Система переходить на сторінку .з списком лекцій вже з створеною лекцію.
    1. Функціональні вимоги

REQ 1. Усі необхідні поля мають бути перевірені на коректність.

REQ 2. Можливість очистити введену інформацію.

REQ 3. Можливість видалити лекцію.

REQ 4. Можливість редагувати інформацію про вже створені лекції.

1. **Додавання питань для студентів**
   1. Опис і пріоритет

Пріоритет – високий. Викладач має можливість створити питання, на які будуть відповідати студенти.

* 1. Послідовності дія-відгук

1. Користувач відкриває список питань у потрібній лекції.
2. Користувач натискає кнопку «Створити».
3. Користувач вводить текст питання.
4. Користувач вибирає, чи студенти можуть відповідати на декілька варіантів відповіді.
5. Користувач створює щонайменше два варіанти відповіді на питання.
6. Користувач може створити додаткові варіанти відповіді.
7. Користувач натискає кнопку «Створити».
8. Система переходить на сторінку .з списком питань, які відносяться до якоїсь лекції.
   1. Функціональні вимоги

REQ 1. Усі необхідні поля мають бути перевірені на коректність.

REQ 2. Можливість видалити питання.

REQ 3. Можливість редагувати інформацію про вже створені питання.

REQ 4. Можливість обнулити статистику по студентських відповідях.

1. **Перегляд питань від студентів**
   1. Опис і пріоритет

Пріоритет – високий. Викладач має можливість переглянути питання, які створили студенти для нього.

* 1. Послідовності дія-відгук

1. Користувач відкриває список питань створених студентами у потрібній лекції.
2. Користувач переглядає їх і у реальному часі отримує нові питання.
   1. Функціональні вимоги

REQ 1. Система повинна відображати питання студентів, одразу, як вони створюють їх.

1. **Видалення питань від студентів**
   1. Опис і пріоритет

Пріоритет – середній. Викладач має можливість видаляти питання створенні студентами.

* 1. Послідовності дія-відгук

1. Користувач відкриває список питань створених студентами у потрібній лекції.
2. Користувач видаляє питання, які він вважає недоречними.
   1. Функціональні вимоги

REQ 1. Система повинна дозволяти видалити питання, створенні студентами.

1. **Перегляд графіків по питаннях**
   1. Опис і пріоритет

Пріоритет – середній. Користувач має можливість переглянути графіки, які відображають статистичну інформацію про відповіді студентів.

* 1. Послідовності дія-відгук

1. Користувач відкриває список питань у потрібній лекції.
2. Користувач натискає на кнопку «Графік».
3. Система відображає кругову діаграму, якщо кількість варіантів відповіді два, якщо більше, то стовпчасту.
   1. Функціональні вимоги
4. **Генерація QR-коду**
   1. Опис і пріоритет

Пріоритет – середній. Викладач має можливість відкрити QR-код на якому закодований пароль до лекції, щоб студенти могли швидко зчитати пароль на мобільній аплікацій.

* 1. Послідовності дія-відгук

1. Користувач відкриває список лекцій.
2. Користувач у потрібній лекції натискає на кнопку «QR-код».
3. Система відкриває нову вкладку веб-браузера, де відображає QR-код, який містить закодований пароль до лекції.
   1. Функціональні вимоги

REQ 1. Кодування паролю до лекції у QR-коді.

**2.2.4 Вимоги до зовнішніх інтерфейсів**

**2.2.4.1 Користувацькі інтерфейси**

Користувач може взаємодіяти з веб-системою за допомогою миші та клавіатури або сенсорного екрану.

**2.2.4.2 Апаратні інтерфейси**

Поточна веб-система не буде використовувати жодних апаратних інтерфейсів.

**2.2.4.3 Програмні інтерфейси**

Взаємодія з базою даних (PostgreSQL) буде відбуватися за допомогою бібліотеки Django ORM, яка використовує бібліотеку psycopg2.

**2.2.4.4 Комунікаційні інтерфейси**

Клієнтська частина системи буде здійснювати комунікацію через REST API з допомогою HTTP-методів протоколу HTTP/1.1.

**2.2.5 Інші нефункційні вимоги**

**2.2.5.1 Вимоги продуктивності**

Веб-сайт повинен миттєво відповідати на дію користувача (уся інформація повинна бути надана у межах 3 секунд, якщо це неможливо то повинен відображатися анімація завантаження)

**2.2.5.2 Вимоги безпеки**

Верифікація користувача відбувається за допомогою електронної пошти та паролю. Паролі не зберігаються у відритому вигляді, а захешовані. Усі дані, які створив користувач доступні лише йому одному.

**2.2.5.3 Атрибути якості програмного продукту**

* Переносимість.
* Надійність.
* Зручність використання.
* Зручність супроводу.

2.3 Висновки до розділу постановка задачі та специфікації вимог

Завершуючи розділ, можна зробити висновки, що було створено та формалізовано опис усіх функцій та вимог, було поставлено чітке завдання на розробку веб-системи, а також описані вхідні та вихідні дані для роботи над розробкою веб-системи.

РОЗДІЛ 3. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ЗАСТОСУВАННЯ

# 3.1 Проектування архітектури

Щодо архітектури, то найкращим варіантом є реалізація клієнт-серверної архітектури, адже цей тип архітектури є найбільш підходящим для збирання та обробки інформації від студентів. Тобто, можна мати один централізований сервер, на який будуть відправлятися та отримуватися усі дані від студентів та викладачів. Також його можна використовувати для інтелектуальної обробки інформації від студентів, а на клієнтах лише відображати різноманітну інформацію. Це значно спростить розробку клієнтів даної веб-системи на зменшить вимого щодо апаратного забезпечення клієнтів, тому, що усі ресурсоємні операції будуть відбуватися на сервері.

Найкращим інтерфейсним рішення є створення взаємодія за підходом RESTful API – цей спосіб побудови API є найбільш сучасним та виправданим у даному випадку, адже ресурси системи будуть змінювати нечасто і API буде досить коротким і лаконічних, а обмін даними можна здійснювати у форматі JSON, який є малим і тому швидким а це досить важливо для даної системи, адже вона вимагає отримання відповідей від студентів та відображення цих даних у реальному часі [9].

Щодо організації архітектури серверної частини, то доцільно використати архітектурний шаблон MVC (Model-View-Controller) – шаблон, у якому компоненти веб-системи поділяються на три частини:

* Model – частина, що відповідає за збереження і отримання даних, саме вона взаємодіє з базою даних. Найчастіше це здійснюється з допомогою ORM (Object-relational mapping) – це коли таблиці бази даних відображаються на класи мови програмування і з даними у базі даних можна працювати як зі звичайними об’єктами мови програмування.
* View – частина, що відповідає за представлення даних, усе для чого вона потрібна, то це відображення даних у зручному для користувача вигляді. Сучасною тенденцією для веб-систем є винесення компонент View цілком на клієнт використовуючи сучасні веб-фреймворки, такі як Angular чи React.
* Controller – частина, що відповідає за бізнес-логіку і зв’язування Model та View – вона отримує дані від Model, обробляє їх і відправляє у View для відображення користувачу, також користувач може змінювати дані, що є у View, у цьому випадку Controller отримує дані з View і надсилає у Model.

Детальну схему можна побачити на рис. 3.1 – тут відображено схема взаємодії усіх трьох компонент і користувача.



Рис. 3.1. Схема взаємодії компонент по шаблону MVC

Для кращого розуміння взаємодії користувачів з системою була створена діаграма випадків використання, яку можна побачити у додатку А.

# 3.2 Вибір інструментальних засобів для реалізації веб-системи

Програмні засоби, які будуть використанні для реалізації даної архітектури будуть такими, щодо серверної частини, то вона буде реалізована на мові програмування Python, з використанням веб-фреймовку Django, для побудови RESTful API буде використаний модуль Django REST Framework.

Django - це веб-фреймворк написаний на мові програмування Python і призначення для побудови веб-систем [6].

Архітектура побудована на основі моделі MVC (Model-View-Controller), однак у Django вона має деякі відмінності від цієї класичної моделі – тут використовується підхід MTV (Model-Template-View) [7], діаграму взаємодії компонентів цієї моделі з клієнтом можна на рис. 3.2.



Рис. 3.2. Взаємодія компонентів у моделі MTV.

Основна відмінність моделей MTV та MVC полягає у тому, що роль Controller з моделі MVC тут виконує View, а View – Template, тому що Django має власні шаблони, які дозволяють відповідати за відображення даних користувачу. Однак, враховуючи останні тенденції у сфері веб-розробки, компоненту Template виносять з роботи веб-фреймоворка на один з сучасних JavaScript фреймоворків Angular або React. Поняття Model і Model цілком збігаються [6]. Детальніше відмінності MVC та MTV можна побачити на рис. 3.2.

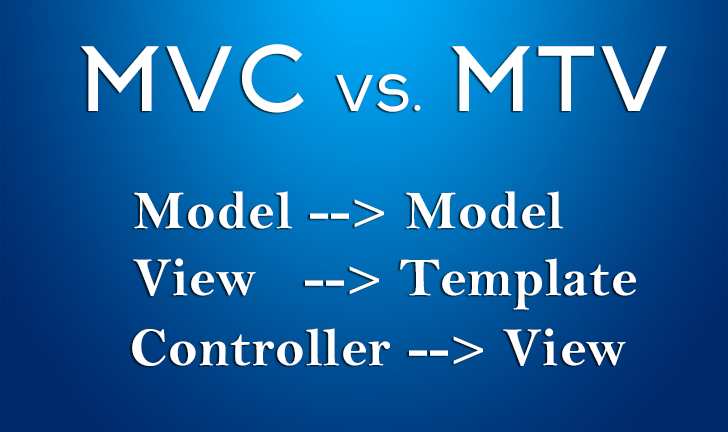


Рис. 3.2. Відмінності MVC та MTV

Також у цьому веб-фреймоворку реалізована одна з основних ідей філософії мови програмування Python – “Поставка з батарейками”, тобто веб-фреймоворк містить в собі усі необхідні модулі для повноцінної розробки, відлагодження та розгортування.

Для зберігання і опрацьовування даних буде використана система управління базами даних PostgreSQL, адже це вільне програмне забезпечення і веб-фрейморк Django був спроектований для роботи з нею.

Щодо клієнтської частини, яка буде написана для створення повноцінного кабінету викладача, то там буде використовуватися мова програмування JavaScript з допомогою фреймворку AngularJS.

Для зручного відображення взаємодії різних компонент веб-системи і щоб побачити де буде кожен з компонентів системи, була створена діаграма розгортування, яку можна побачити у додатку Б.

3.3 Проектування класів веб-системи

Діаграму класів можна побачити у додатку В. Опис головних класів веб-системи:

Subject – сутність, яка представляє собою навчальні дисципліни. Її поля:

* name – назва навчальної дисципліни;
* create\_date – дата створення навчальної дисципліни;
* update\_date – дата оновлення даних у навчальній дисципліні;
* groups – текст з інформацією про навчальні групи, які прослуховують навчальну дисципліну;
* lectures – список лекцій, які прив’язані до навчальної дисципліни.

Lecture – сутність, яка представляє собою навчальні лекції. Її поля:

* lecture\_name – назва лекції;
* secret\_key – пароль для доступу студентів до лекції;
* create\_date – дата створення лекції;
* update\_date – дата оновлення даних у лекції;
* place – текст з інформацією про аудиторію, де буде проводитися лекція;
* start\_time – час початку лекції;
* date – дата проведення лекції;
* lesson\_number – порядковий номер лекції у розкладі;
* questions – список викладацьких питань, які прив’язані до лекції;
* student\_ questions – список студентських питань, які прив’язані до лекції.

Subject – сутність, яка представляє собою навчальні дисципліни. Її поля:

* name – назва навчальної дисципліни;
* create\_date – дата створення навчальної дисципліни;
* update\_date – дата оновлення даних у навчальній дисципліні;
* groups – текст з інформацією про навчальні групи, які прослуховують навчальну дисципліну;
* lectures – список лекцій, які прив’язані до навчальної дисципліни.

Question – сутність, яка представляє собою питання, що створив викладач для студентів. Її поля:

* question\_text – текст питання;
* multiply\_choices – можливість студентам вибирати декілька варіантів відповіді;
* choices – список варіантів відповідей, які прив’язані до питання.

Choice – сутність, яка представляє собою варіант відпоіді. Її поля:

* choice\_text – текст питання;
* votes – кількість студентів, що проголосували за цей варіант відповіді.

Також тут присутня функція vote(), що потокобезпечно збільшує лічильник кількості студентів, що проголосували за варіант відповіді.

StudentQuestion – сутність, яка представляє собою питання від студентів. Її поля:

* question\_text – текст питання, створеного студентом.

3.4 Проектування бази даних

Як вже було сказано у вибір інструментальних засобів для реалізації веб-системи, оптимальною СУБД є використання PostgreSQL, тому що веб-фреймоворк Django був спроектований для роботи саме з нею. Завдяки цьому, можна не писати SQL-запити вручну, а використовувати більше сучасний спосіб роботи з базою даних, а саме працювати з нею через ORM і використовувати міграції від Django.

Завдяки роботі через ORM, таблиці бази даних будуть відображені у класи мови програмування Python і працювати з рядками таблиць можна через об’єкти цих класів. Це дозволяє відмовитися від написання SQL-запитів і від необхідності знання SQL взагалі.

Також використання міграцій Django дозволяє відслідковувати історію зміни структури бази даних, її розвиток, а також у разі потреби повертатися на попередню версію. Крім цього, міграції значно спрощують роботу у команді над базою даних.

Саме тому було прийнято рішення працювати через ORM і діаграму класів, які працюють через ORM можна побачити у додатку В.

# 3.3 Проектування графічного інтерфейсу користувача

Для побудови графічного інтерфейсу користувача використовувалися рекомендації по стилю Material Design [10], розроблений компанією Google. Особливістю цього стилю дизайну є те, що він значно полегшує сприйняття інформації, а також його можна використовувати як на мобільних додатках, так і на веб-сайтах, тобто легше досягнути уніфікованого дизайну.

Основна колірна схема була створена з допомогою веб-сайту Material Palette (materialpalette.com) і вона відповідає усім вимогам Material Design, щодо правил відповідності кольорів. Саму колірну схему можна побачити на рис. 3.3

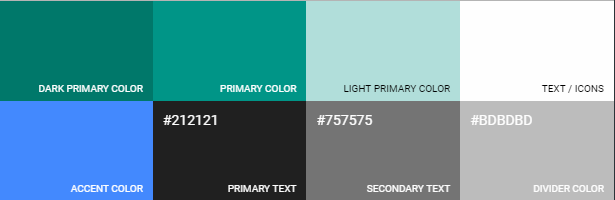


Рис. 3.3. Колірна схема

Для пришвидшення загальної розробки, прототип був створений з допомогою бібліотеки Angular Material – це дозволило створити прототип одразу згідно вимог Material Design і застосувати отриманий прототип у подальшій розробці веб-сервісу.

Однією з основ Material Design є карточка – коротка структурована інформація, яка виділяється від іншої (рис. 3.4), було прийнято рішення застосувати її для відображення усієї важливої інформації про навчальні предмети і лекції.

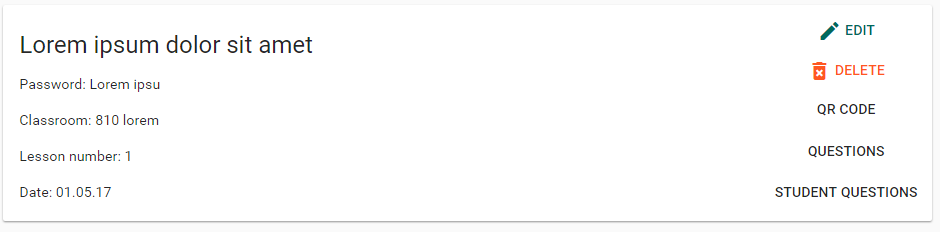


Рис. 3.4. Карточка яка відображає інформацію про лекцію

Також для керування інформацією про лекції чи предмет, на карточці було створений спеціальний блок з кнопками виконаними у стилі Material Design і використанні спеціальні іконки, які дозволяють спростити розуміння дії, яку виконує кнопка (наприклад, іконка олівця на кнопці “Edit” або іконка сміттєвого бачка на кнопці “Delete” ) (рис. 3.5).

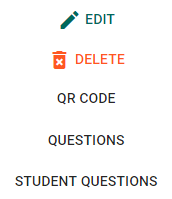


Рис. 3.5. Блок керування кожної картки створений кнопка у стилі Material Design

Також для створення чи редагування інформації про предмети чи лекції було використано карточку, форми вводу тексту відповідно до стандартів Material Design з вертикальними підписами призначення форми вводу, а також обмеженнями на максимальний розмір тексту у формі. Крім цього, тут були використанні дві кнопки у акцентуючому стилі, одна з яких використовує головний колір колірної схеми, а інші колір уваги (рис. 3.6).

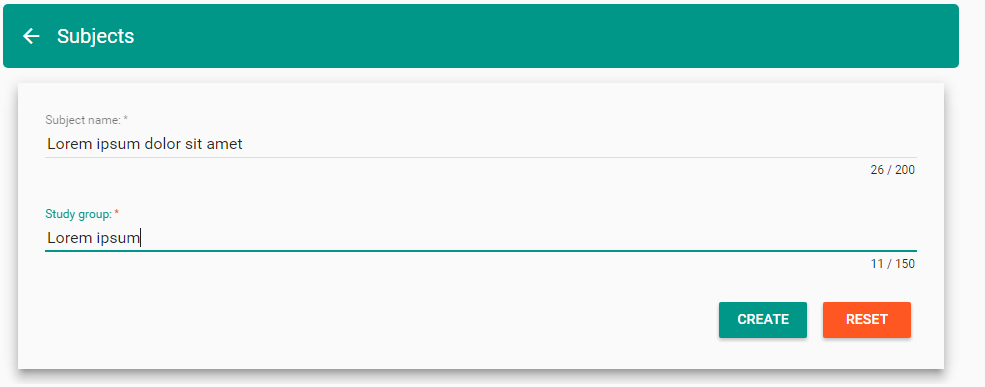


Рис. 3.6. Зображення картки для створення навчального предмету

3.4 Висновки до розділу проектування та архітектури

Завершуючи розділ, можна зробити висновки, що було проведено повноцінне попереднє проектування веб-системи для організації взаємодії студента з викладачем на лекції, був проведений огляд та вибір найкращої архітектури, проведено вибір найбільш підходящих засобів для створення цього програмного забезпечення, спроектовані класи та графічний інтерфейс користувача.

1. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ КОМУНІКАЦІЇ СТУДЕНТІВ З ВИКЛАДАЧЕМ НА ЛЕКЦІЇ (СЕРВЕРНА ЧАСТИНА)

4.1 Реалізація веб-системи

Для повноцінного використання веб-системи, вона має бути реалізована з декількох частин: частина, що відповідає за роботу з базою даних, валідацію даних, бізнес логіку, а також для створення RESTful API. Ця частина називається бек-ендом і відповідає за отримання, обробку і відправлення інформації іншим клієнтам. Друга частина, відповідає за створення графічного інтерфейсу користувача для роботи з системою і відображення даних, що можна дістати використовуючи першу частину. Ця частина називається фронт-ендом.

Враховуючий вищесказане, була обрана стандартна схема організації проекту для створення веб-системи з допомогою веб-фреймоворку Django, яку можна побачити на рис. 4.1.

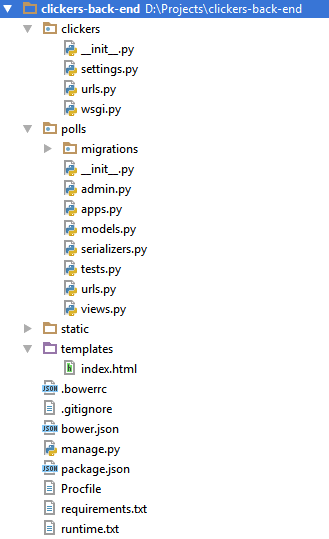


Рис. 4.1. Структура проекту

Структура проекту складається з 4 модулів: clickers, polls, static, templates та конфігураційних файлів необхідних для запуску веб-системи.

Модуль clickers – це головний модуль проекту, у ньому знаходиться файли:

* \_\_init\_\_.py – пустий файл, він вказує інтерпретатору Python, що ця папка – це модуль,
* settings.py – тут знаходяться різні налаштування для конфігурації, наприклад налаштування підключення до бази даних, способи підключення до електронної скриньки для надсилання електронних листів і т.д.,
* urls.py – це файл, який визначає як буде здійснюватися маршрутизація за різними url адресами між різними модулями,
* wsgi.py – налаштування для роботи з веб-сервером – nginx або власний сервер Django призначення для відлагодження.

Модуль polls – це модуль для реалізації усієї логіки для роботи з даними, їх збереження, отриманні, обробка та відправлення, у термінології Django цей модуль називається App.

Модуль migrations призначення для збереження міграцій – історії змін схеми бази даних, це дозволяє гнучко працювати з базою даних у команді, швидко розгортувати проект, відслідковувати еволюцію схеми бази даних, а також визначати хто саме які зміни вніс до неї.

Файл admin.py призначений для створення універсального інтерфейсу для роботи з даними у Django Admin, який генерується автоматично, він дозволяє легко і швидко працювати з даними у веб-інтерфейсі без необхідності доступу до бази даних.

Файл apps.py потрібний для конфігурації модуля App, у цьому випадку – polls, він дозволяє визначати ім’я модуля та інші його метадані.

Файл models.py використовується для створення моделей бази даних і роботи з нею через ORM. Також так описується логіка яка відноситься до роботи з об’єктами бази даних, наприклад, у цьому проекті там зберігається функція для збільшення кількості студентів, що проголосували за задане питання.

Файл serializers.py призначений для обміну та валідації даних між моделями описаними у models.py і RESTful API логіка якого розміщена у файлі views.py і реалізованого з допомогою бібліотеки Django REST Framework.

Файл test.py призначення для створення unit-тестів для тестування коректності роботи функцій у цьому модулі. Файл urls.py потрібний для маршрутизації обробки заданого url одним зі класів чи функцій у файлі views.py. Файл views.py призначення для створення бізнес-логіки, отримання інформації, обробки GET, POST, PUT, DELETE запитів, створення RESTful API.

Модуль Templates призначений для генерування шаблонів веб-сторінок але так як клієнтський веб-сайт буде реалізований за допомогою AngularJS, то у ньому немає потреби і тут зберігається лише один шаблон – index.htm, який служить вхідною точкою для роботи AngularJS.

Файли .bowerrc і bower.json – призначені для конфігурації та контролю залежностей проекту AngularJS з допомогою пакетного менеджера Bower – він дозволяє автоматично знаходити, зберігати, оновлювати і встановлювати необхідні пакети для проекту, побудованому на AngularJS.

Файл .gitignore – це службовий файл системи контролю Git необхідний для ігнорування деяких файлів при роботі з цією системою контролю версій.

Файл manage.py потрібний для роботи з веб-фреймоворком Django – він дозволяє створювати міграції, застосовувати їх, створювати адмінів, керувати статичними файлами, запускати веб-сервер у різних режимах та багато інших функцій.

Файли package.json, Procfile, runtime.txt – це конфігураційні файли призначенні для автоматизованого розгортування на сервері з використанням Gunicorn uwsgi-сервера.

У файлі reqirements.txt міститься інформація про необхідні зовнішні залежності, такі як бібліотеки та модулі. За допомогою утиліти pip можна автоматично встановлювати і оновлювати усі залежності проекту з цього файлу.

Щодо модуля static, то там розміщаються усі статичні фали (HTML, CSS, JavaScript), які відносяться до проекту. Також там розміщується проект побудований на AngluarJS, детальнішу структуру якого можна побачити на рис. 4.2.

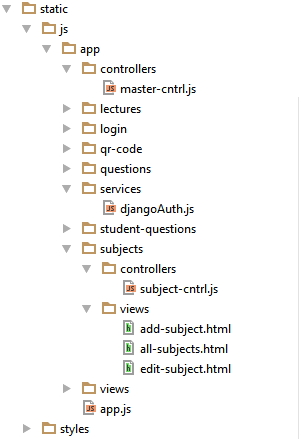


Рис. 4.2. Структура проекту AngularJS

Папка app – це головний модуль, який відноситься до проекту AngularJS. У середині нього є папки, які відповідають за організацію сторінок, кожна з яких має у собі папку controllers, у якій розміщуються js-код, який реалізує контролер поточної сторінки, а також папки views, де розміщуються html файли, які реалізують шаблони вигляду для кожної сторінки.

Окремо хочеться виділити файл app.js і папку services – app.js це головний файл проекту, там здійснюється власна маршрутизація між іншими сторінками, а також підключення усіх необхідних модулів.

Щодо папки services – там знаходяться файли, які реалізують сервіси на AngularJS – спеціальні функції для роботи з REST API сервера.

4.2 Опис роботи з веб-системою

Так як веб-система складається з двох частин: серверна частина, яка надає REST API і веб-сайта, який представляє собою кабінет для викладача, то спосіб роботи з ними буде відрізнятися.

Перша за все, щодо серверної частини, для можливості створення різних клієнтів для роботи з веб-системою (веб-сайти побудовані з допомогою веб-фреймоворків AngularJS або React, мобільні аплікації на платформах Android, iOS та Windows Phone) був створений платформо незалежний API, який був спроектований згідно підходу REST.

Для роботи з ним необхідно використовувати такі url-адреси:

/api/polls/ (методи GET і POST) – кінцева точка призначена для створення та отримання списку усіх питань, створених викладачем.

/api/polls/<id>/ (методи GET, PUT, PATCH, DELETE) – кінцева точка призначена для отримання, зміни або видалення питання, створеного викладачем за її id.

/api/choices/ (методи GET і POST) – кінцева точка призначена для створення та отримання списку усіх варіантів відповідей.

/api/choices/<id>/ (методи GET, PUT, PATCH, DELETE) – кінцева точка призначена для отримання, зміни або видалення варіанту відповіді за його id.

/api/choices/<id>/vote/ (метод POST) – кінцева точка призначена для голосування за варіанту відповіді по його id.

/api/lectures/ (методи GET і POST) – кінцева точка призначена для створення та отримання списку усіх лекцій.

/api/lectures/<id>/ (методи GET, PUT, PATCH, DELETE) – кінцева точка призначена для отримання, зміни або видалення лекції за її id.

/api/subjects/ (методи GET і POST) – кінцева точка призначена для створення та отримання списку усіх предметів.

/api/subjects/<id>/ (методи GET, PUT, PATCH, DELETE) – кінцева точка призначена для отримання, зміни або видалення предмету за його id.

/api/student-questions/ (методи GET і POST) – кінцева точка призначена для створення та отримання списку усіх питань від студентів.

/api/student-questions/<id>/ (метод DELETE) – кінцева точка призначена для видалення питання від студента за його id.

/api/rest-auth/password/reset/ (метод POST) – кінцева точка призначена для скидання пароля користувача.

/api/rest-auth/password/reset/confirm/ (метод POST) – кінцева точка призначена для встановлення нового пароля користувача після того, як секретне посилання, що на дійшов на поштову скриньку користувача було підтверджене.

/api/rest-auth/password/reset/confirm/ (метод POST) – кінцева точка призначена для встановлення нового пароля користувача після того, як секретне посилання, що на дійшов на поштову скриньку користувача було підтверджене.

/api/rest-auth/login/ (метод POST) – кінцева точка призначена для отримання токена призначеного для роботи з REST API.

/api/rest-auth/logout/ (метод POST) – кінцева точка призначена для скидання токена отриманого при логуванні для роботи з REST API.

/api/rest-auth/logout/ (метод POST) – кінцева точка призначена для скидання токена отриманого при логуванні для роботи з REST API.

/api/rest-auth/user/ (методи GET, PUT, PATCH) – кінцева точка призначена для отримання або зміни інформації про поточного користувача.

/api/rest-auth/password/change/ (метод POST) – кінцева точка призначена для зміни пароля користувача.

/api/rest-auth/registration/ (метод POST) – кінцева точка призначена для реєстрації нового користувача у веб-системі.

/api/rest-auth/registration/verify-email/ (метод POST) – кінцева точка призначена для підтвердження поштової скриньки при реєстрації нового користувача у веб-системі.

Увесь обмін даних відбувається у форматі JSON, для роботи з REST API потрібно при кожному запиті посилати спеціальний токен, отриманий після логування користувача у систему за допомогою кінцевої точки /api/rest-auth/login/ у заголовку у полі Authorization у вигляді “Token: xxx”, де xxx – це отриманий токен.

Також, користувач буде отримувати лишу ту інформацію, яка належить йому, наприклад при отриманні списку предметів, він отримає лише ті предмети, які створив він сам, а не всі, що є у системі, аналогічно для інших сутностей. Таким чином забезпечується безпека даних.

Для спрощення роботи з REST API була створена спеціальна сторінка /api/docs/ у якій міститься уся необхідна інформація для роботи з REST API основний вигляд сторінки можна побачити на рис. 4.3. Ця сторінка була згенерована за допомогою бібліотеки DRF Docs, яка дозволяє автоматично генерувати та оновлювати усю документацію з коментарів, які описують класи, які обробляються відповідні кінцеві точки.



Рис. 4.3. Вигляд згенерованої документації

Також ця сторінка дозволяє випробувати створений REST API – можна обрати потрібну кінцеву точку і надсилати різні запити, приклад для отримання лекцій користувача за його токеном показаний на рис. 4.4.

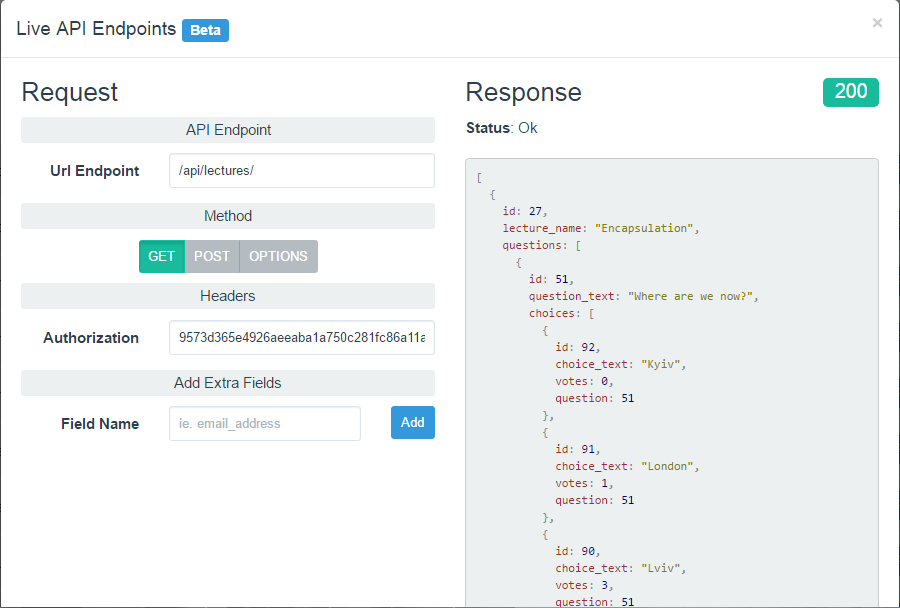


Рис. 4.4. Приклад сторінки з GET запитом для отримання лекцій і відповіддю від сервера

Щодо клієнтського сайту, то для початку необхідно перейти на основну сторінку сайту і увійти використовуючи форму на зображену на рис. 4.5, де у відповідні поля потрібно ввести логін або адресу електронної скриньки і пароль.

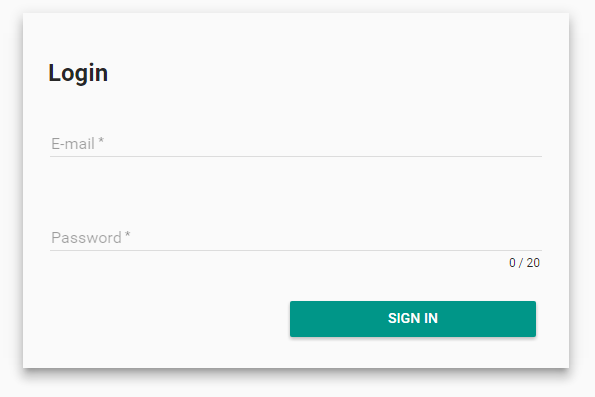


Рис. 4.5. Форма для входу у систему

Далі перед користувачем з’явиться головне вікно сайту, де він може працювати з навчальними дисциплінами або лекціями (рис. 4.6).

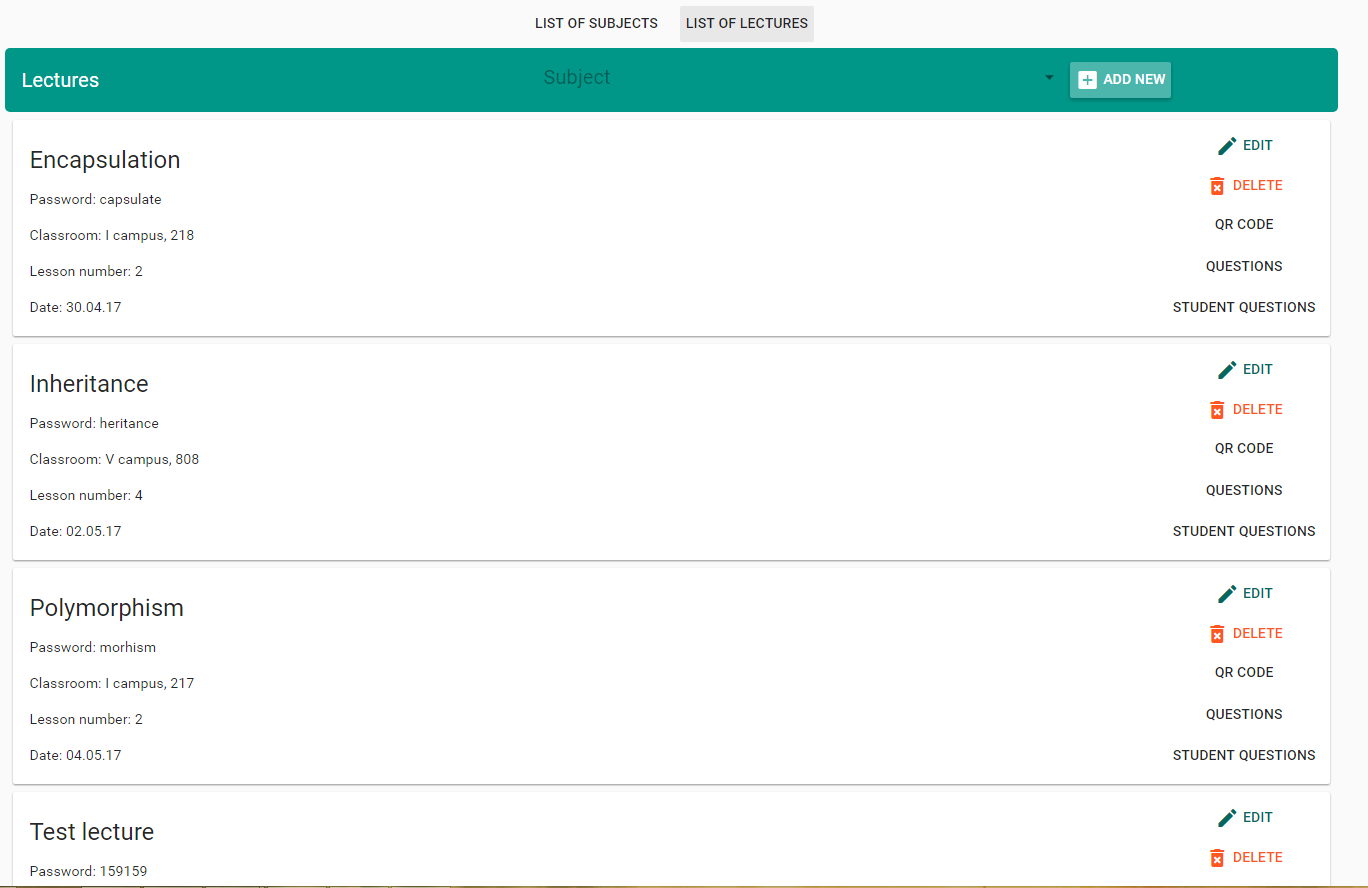


Рис. 4.6. Сторінка з списком лекцій

На сторінці зі списком лекцій (рис. 4.6) ми можемо редагувати їх, видаляти, генерувати QR-код для доступу студентів до лекції, перейти на сторінку з власними питаннями (рис. 4.7), а також переглядати питання від студентів (рис. 4.8).

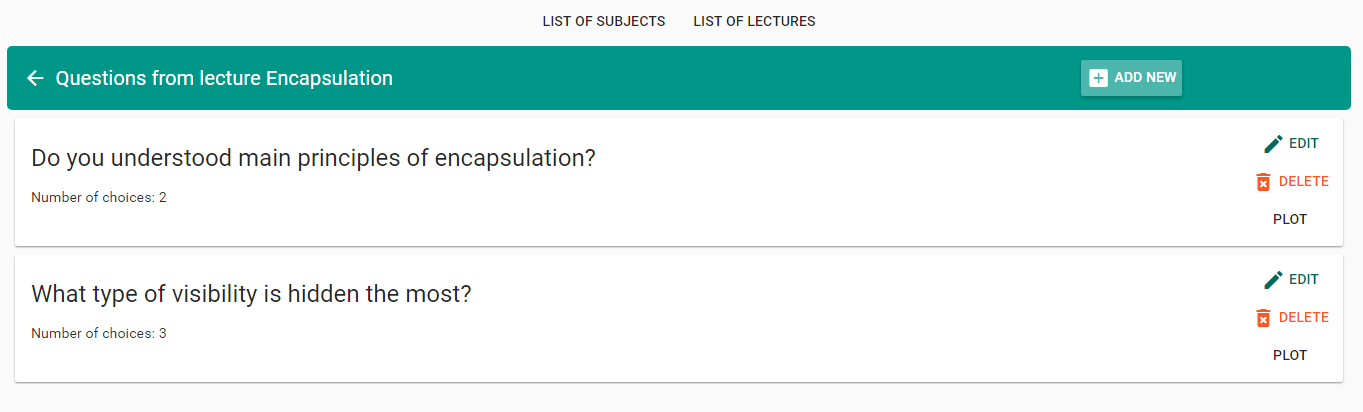


Рис. 4.7. Питання створенні викладачем до лекції “Encapsulation”

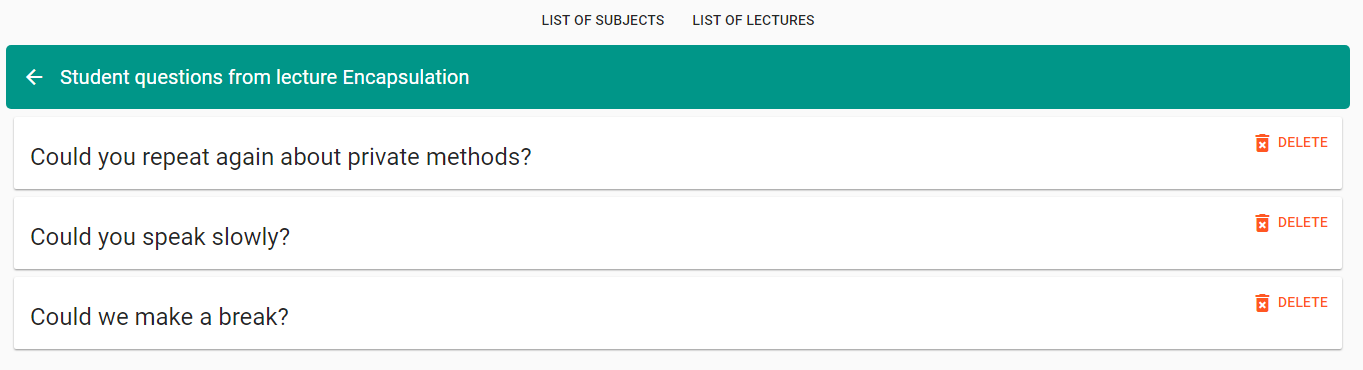


Рис. 4.8. Питання, які створили студенти на лекції “Encapsulation”

Також, як видно з рис. 4.7, викладач може створювати нові питання для студентів, редагувати і видаляти існуючі, а також переглядати як відповідали студенти на кожне питання (рис. 4.9). Щодо питань студентів (рис. 4.8), викладач може видаляти їх, якщо побачить якесь питання, яке не відповідає контексту лекції.

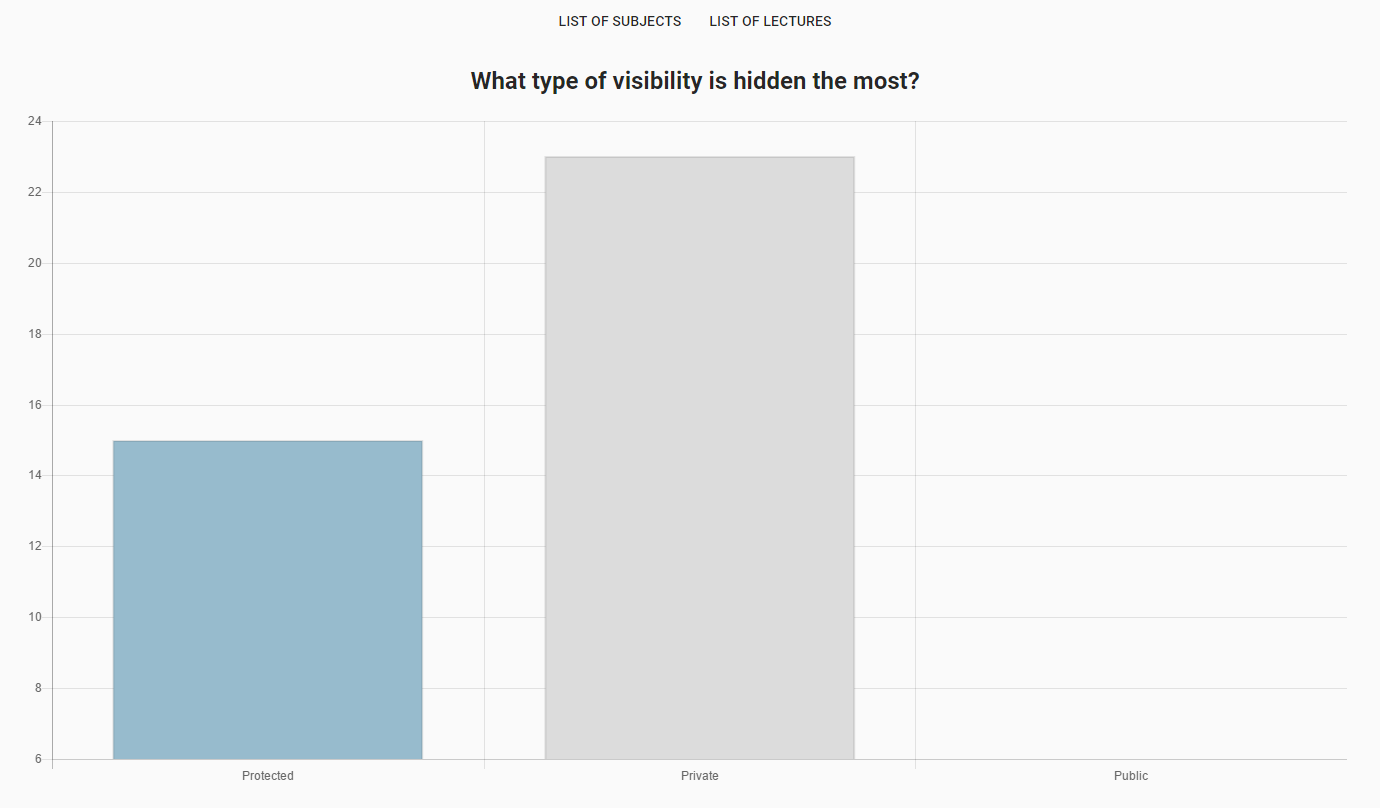


Рис. 4.9. Результати голосування студентів на питання

4.3 Тестування веб-системи

Для підвищення якості веб-системи, забезпечення правильності функціонування, а також, щоб під час розробки не “поламати” якихось функцій, було прийняти рішення створити Unit-тести, які будуть тестувати окремі функції та класи. Unit-тести – це вид модульного тестування, при якому здійснюється тестування окремих частинок системи, які є повністю ізольованими від решти програми. Для того, щоб це досягнути потрібно замінити інші частини системи (базу даних, мережеві запити) від яких залежить якась функція на власні, які будуть віддавати наперед відомий результат.

Для створення Unit-тестів для веб-фреймоворку Django є спеціальна бібліотека django.test, яка є унаслідується від стандартної бібліотеки для тестування у Python unittest, тобто у загальному побудова тестових класів відрізнятися не буде, додасться тільки специфічні для Django методи роботи.

Щодо тестування REST API, то для цього також існують спеціальна методи, які надаються бібліотекою Django REST Framework, це модуль rest\_framework.test. У цьому модулі є спеціальні класи, для різноманітної перевірки обробки даних через REST API:

* APIRequestFactory – клас призначений для зручного створення основних запитів до REST API: GET, POST, PATCH, PUT, DELETE.
* APIClient – клас призначений для отримання відповідей від REST API.
* RequestsClient – клас для тестування API з допомогою популярної бібліотеки для створення мережевих запитів requests.
* CoreAPIClient – базовий клас для роботи з API, від нього можна наслідуватися і створювати власні класи для тестування.
* APITestCase – клас для створення тестових випадків для роботи з API.

Враховуючи усе вищезгадане був створений набір Unit-тестів, які перевіряють коректність роботи REST API, результати проходження тестів можна побачити на рис. 4.10. Детальний опис кожного тесту знаходиться у звіті про тестування (див. додаток ).

Як видно з рис. 4.10, усі тести були пройдені і можна бути впевненим, що система функціонує коректно згідно заданих вимог. Також при подальшій розробці, якщо якісь зміни порушать коректність роботи поточного API це буде швидко виявлено, адже у цьому випадку тести проходитися не будуть.

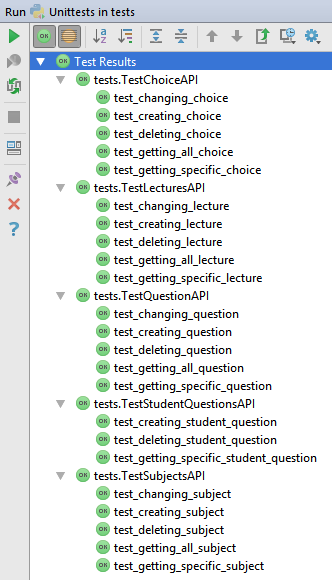


Рис. 4.10. Результати проходження тестів

Також для підвищення якості коду і виявлення семантичних та синтаксичних помилок був застосований статичний аналізатор коду PyLint. Цей аналізатор проводить перевірку на коректність коду без його запуску (саме тому він статичний), а також на відповідність коду стандарту PEP8 – це домовленість щодо стилю коду (найменування зміних, класів, функцій, обмеження на максимальну довжину рядку). Результати тестування можна побачити на рис. 4.11

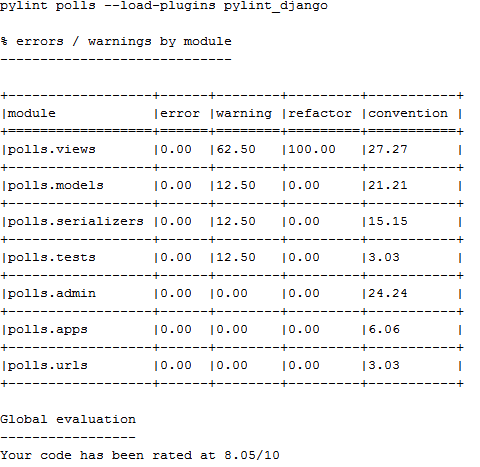


Рис. 4.11. Результати тестування статичним аналізатором PyLint

4.4 Висновки розділу реалізації і тестування веб-сервісу

Завершуючи розділ, можна зробити висновки, що було розроблено серверну частина програмного забезпечення для організації комунікації студента з викладачем на лекції, було створено повноцінний REST API для того, щоб з серверною частиною могла працювати будь-яка клієнтська частина, стокерний API повністю виконує функції, поставлені на нього у специфікації вимог, також було розроблено клієнтський сайт, який дозволяє працювати з веб-системою викладачеві, а для забезпечення коректності функціонування і високої якості коду, було проведено тестування системи з допомогою модульного тестування, а також статичного аналізу коду.

РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

5.1. Економічна характеристика проектного рішення (програмного продукту)

Метою програмного забезпечення для організації комунікації студентів з викладачем на лекції є впровадження інноваційних методів двосторонньої інтерактивної взаємодії студента і викладача на лекції з допомогою сучасних IT-засобів, а також заміна застарілих технологій для організації подібної взаємодії – клікерсів.

Веб-система повинна мати такі функції:

• студенти можуть голосувати із допомогою мобільного застосування;

• викладач має змогу управляти своїми дисциплінами та лекціями;

• викладач має змогу створювати запитання як попередньо, так і під час проведення лекції;

• учасники можуть переглядати результати опитування в реальному часі;

• можливість сформувати статистику по окремих запитаннях;

• можливість обмеження доступу до лекцій за паролем;

• створення питань студентами для обговорення на лекції у режимі реального часу.

Ця веб-система розроблена у вигляді сервера, який займається зберіганням і обробкою даних, а також клієнта – веб-сайту, який надає зручний графічний інтерфейс для користувача. Також до сервера підключається мобільна аплікація, яка дозволяє працювати з веб-системою за допомогою мобільних пристроїв. Завдяки цій системі легко організувати активну двосторонню взаємодію між студентом і викладачем на лекції.

5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару

В Україні зараз відбувається значні зміні у сфері освіти і вона стає все більше інноваційною і ближчою для студентів, саме тому виникає проблема у впровадженні нових способів введення лекцій.

Одним з популярних інноваційних методів введення лекцій є використання спеціальної системи – клікерсів. Клікерс (“Clickers”, від назви пультів, на яких потрібно клікати студентам) - це система інтерактивної взаємодії між студентами та викладачем на лекції, яка здійснюється за допомогою пультів, які є в студентів, бази, яка отримує та обробляє інформацію він пультів і знаходиться у лекційній аудиторі та спеціального програмного забезпечення.

Це система вже давно використовується в університетах США, однак взагалі не розповсюджена в Україні. Основними причинами не розповсюдження цих пристроїв є відсутність інтересу до покращення навчального процесу у вищих навчальних закладах України, а також висока вартість у використанні (кожний набір пультів-клікерсів у середньому коштує 700 доларів США, а їх треба закупити кожному студенту).

Враховуючи усе вище сказане, можна зробити висновок, що розробка програмного забезпечення для організації комунікації студентів з викладачем на лекції є дуже потрібна не тільки на ринку Україну, а й на ринку США, адже на ринку України не існує подібних систем і при виході на ринок України не буде конкуренції, тобто буде діяти стратегія «голубого океану», створення нової ніші, а щодо ринку США, то цей продукт буде мати усі шанси отримати велику ринкову долю, адже дозволить значно знизити витрати на подібні системи в університетах (не потрібно закупляти кожному студенту пульт, студентам досить встановити спеціальну мобільну аплікацію, яка є безкоштовною)

5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ

Передбачає оцінювання та аналіз факторів зовнішнього та внутрішнього середовищ групою експертів.

Фактори зовнішні оцінюються за шкалою [-5;5], при цьому межі шкали відображають максимальний негативний та позитивний вплив факторів на організацію, 0 демонструє, що фактор впливає на організацію нейтрально.

Фактори внутрішні оцінюються за шкалою [0;5], при цьому 0 демонструє нерозвинутість, відсутність чи катастрофічний стан фактора внутрішнього середовища, оцінка 5 демонструє високий рівень розвитку даного фактора.

Сума вагомостей усіх факторів становить одиницю, тобто рівень вагомості для кожного фактора визначається за допомогою коефіцієнтів. Зважений рівень впливу факторів розраховується як добуток впливу фактора у балах та рівня вагомості. Результати експертних оцінок впливу факторів зовнішнього середовища на організацію наведено у табл. 5.1.

*Таблиця 5.1*

**Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Фактори** | **Середня експертна оцінка, бали** | **Середня вагомість факторів** | | **Зважений рівень впливу, бали** |
| 1 | 2 | 3 | | 4 |
| *Фактори зовнішнього середовища* |  | | | |
| Споживачі | 5 | 0,11 | 0,55 | |
| Постачальники | 2 | 0,1 | 0,2 | |
| Конкуренти | 0 | 0,1 | 0 | |
| Державні органи влади | 0 | 0,05 | 0 | |
| Інфраструктура | 4 | 0,06 | 0,24 | |
| Законодавчі акти | -1 | 0,1 | -0,1 | |
| Профспілки, партії та інші громадські організації | 0 | 0,05 | 0 | |
| Система економічних відносин в державі | 1 | 0,06 | 0,06 | |
| Організації-сусіди | 3 | 0,01 | 0,03 | |
| Міжнародні події | 2 | 0,01 | 0,02 | |
| Міжнародне оточення | 3 | 0,03 | 0,03 | |

*Продовження табл. 5.1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Науково-технічний прогрес | 3 | 0,07 | | 0,21 |
| Політичні обставини | 0 | 0,06 | | 0 |
| Соціально-культурні обставини | 0 | 0,05 | | 0 |
| Рівень техніки та технологій | 4 | 0,04 | | 0,16 |
| Особливості міжнародних економічних відносин | 1 | 0,02 | | 0,02 |
| Стан економіки | 0 | 0,08 | | 0 |
| Загальна сума |  | 1 | | 1,42 |
| *Фактори внутрішнього середовища* |  | | | |
| Цілі | 5 | 0,11 | 0,55 | |
| Структура | 4 | 0,16 | 0,64 | |
| Завдання | 4 | 0,07 | 0,28 | |
| Технологія | 5 | 0,2 | 1 | |
| Працівники | 3 | 0,21 | 0,63 | |
| Ресурси | 5 | 0,25 | 1,25 | |
| Загальна сума |  | 1 | 4,35 | |

Отже, як видно з таблиці 5.1, на діяльність інформаційної системи найбільш позитивно впливають такі фактори зовнішнього середовища, як споживачі, інфраструктура, рівень техніки та технологій. При цьому необхідно зауважити, що найвагоміший вплив на діяльність системи здійснюють користувачі та інфраструктура (адже для використання побідної системи, університети повинні бути забезпеченні відповідною інфраструктурою – комп’ютери, проектори, покриття Wi-Fi), а найбільш негативно діючим чинникам є законодавчі акти. Найкраще розвинутим фактором внутрішнього середовища є цілі, технологія та ресурси, а найменш розвинутими є працівники.

5.4. Формування стратегічних альтернатив

Перша група стратегічних альтернатив. Критеріями поділу альтернативних стратегій розвитку є існуючий продукт (програмне забезпечення) та новий, а також супутні послуги (рис.5.1).

**Існуюче**

**Нове**

**Стратегія розроблення нового продуту**

**Стратегія нового продукту з супутніми послугами**

**Стратегія розвитку існуючого продуту**

**Стратегія розвитку існуючого продукту з супутніми послугами**

**Продукт (проектне рішення )**

**Наявні**

**Відсутні**

**Додаткові послуги**

*Рис. 5.1. Стратегічні альтернативи*

Для веб-системи по організації комунікації студентів з викладачем на лекції найбільше підходить стратегія розроблення нового продукту, адже після аналізу ситуації на ринку України, можна зробити висновок, що тут немає подібних систем.

Друга група стратегічних альтернатив. Критеріями поділу альтернативних стратегій розвитку є існуючий ринок та продукт, новий ринок та продукт (рис. 5.2.).

**Існуючий**

**Новий**

**Стратегія розвитку продукту**

**Диверсифікація**

**Стратегія глибокого проникнення продукту**

**Стратегія розвитку ринку**

**Продукт (проектне рішення )**

**Ринок**

**Новий**

**Існуючий**

*Рис. 5.2. Стратегічні альтернативи*

Для цієї веб-системи найбільше підходить стратегія диверсифікації, адже вона розробляється для нового сегменту. Оскільки цей продукт є нововведенням, це буде розширенням послуг.

5.5. Бюджетування

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, пов’язаних з виготовленням та реалізацією продукту, яка дає можливість здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва.

*Таблиця 5.2*

**Бюджет витрат матеріалів та комплектуючих виробів**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва матеріалів та комплектуючих** | **Марка, тип, модель** | **Фактична кількість, шт.** | **Ціна за одиницю, грн.** | **Разом, грн.** |
| Ноутбук | HP ProBook 450 G4 | 4 | 19 300 | 77 200 |
| Доменне ім’я | clickers-nulp**.**[com](https://www.ukrnames.com/domains/zone.jsp?name=com) | 1 | 402 | 402 |
| *Всього* | | | | 77 602 |

Бюджет витрат на оплату праці включає заробітну плату всіх працівників, які задіяні в розробці програмного продукту. Зарплата кожного працівника обчислюється на основі трудоємності відповідних робіт і людино-днях та середньої ЗВ відповідних категорій працівників.

Для розробки програмного продукту необхідні чотири спеціалісти розробники:

* Розробник програмного забезпечення для серверної частини (BD)
* Розробник програмного забезпечення для клієнтської частини (FD)
* Менеджер проекту (PM)
* Тестер (QA)

Місячний оклад розробника програмного забезпечення для серверної частини складає – 40 000 грн., для клієнтської частини – 30 000 грн., менеджера проекту – 28 000 грн., і тестера 15 000.

Обчислюємо денну зарплату розробників (22 робочі дні в місяці):

Для розрахунку витрати на оплату праці кожного учасника проекту проводимо за формулою:

(5.1.)

де – чисельність розробників проекту *i-*ої спеціальності, чол.;

*–* час, витрачений на розробку проекту працівником *i-*ої спеціальності, днів.;

*–* деннаЗП розробника *i-*ої спеціальності, грн.

Обчислюємо витрати на оплату праці для кожного працівника:

*Таблиця 5.3*

**Бюджет витрат на оплату праці**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Посада, спеціальність** | **Кількість працівників, осіб** | **Час роботи, дні** | **Денна заробітна плата працівників, грн.** | **Сума витрат на оплату праці, грн.** |
| *Основна заробітна плата* | | | | |
| Програміст серверної частини | 1 | 20 | 1818,18 | 36363,6 |
| Програміст клієнтської частини | 1 | 20 | 1363,63 | 27272,6 |
| Тестер | 1 | 10 | 681,82 | 10681,82 |
| Менеджер проекту | 1 | 22 | 1272,72 | 27999,84 |
| Разом: | | | | 102323,26 |

Згідно бюджету витрат на оплату праці працівників даного проекту, необхідно здійснити відрахування обов’язкових платежів та податку, зокрема, у 2017 році ставка єдиного соціального внеску складає 22%, ставка податку з доходів фізичних осіб – 18%, військовий збір – 1,5%. Таким чином, визначаємо суму відрахувань ЄСВ із заробітної плати працівників:

визначаємо суму податку з доходів фізичних осіб із заробітної плати працівників:

Визначаємо суму відрахувань на військовий збір із заробітної плати працівників:

*Таблиця 5.4*

**Бюджет загальновиробничих витрат**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| *Змінні загальновиробничі витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 7000 |
| - витрати на МШП; | 2724 |
| - витрати на електроенергію; | 5000 |
| - витрати на ремонт; | 1000 |
| - інші змінні витрати; | 3000 |
| Разом змінних витрат: | 18724 |
| *Постійні загальновиробничі витрати, у т.ч.:* |  |
| *-* заробітна плата допоміжного персоналу; | 11000 |
| - комунальні послуги; | 1000 |
| - витрати на оренду; | 5000 |
| - витрати на ремонт; | 2500 |
| - інші постійні витрати; | 5809,5 |
| Разом постійних витрат: | 25309,5 |
| *Разом загальновиробничих витрат:* | 44033,5 |

*Таблиця 5.5*

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| 1 | 2 |
| *Адміністративні витрати, у т.ч.:* |  |
| - заробітна плата адміністративного персоналу; | 3500 |
| - витрати на МШП; | 2100 |
| - витрати на відрядження; | 3000 |
| - витрати на ремонт; | 3000 |
| - витрати на паливно-мастильні матеріали; | 3575 |
| - витрати на сплату податків і зборів; | 3200 |
| - знос адміністративного обладнання; | 150,5 |
| - інші адміністративні витрати; | 1500 |
| Разом адміністративних витрат: | 20025,5 |
| Витрати на збут, у т.ч.: |  |
| - заробітна плата менеджерів зі збуту; | 2375 |
| - витрати на гарантійний ремонт; | 1975,5 |
| - витрати на відрядження; | 1900 |
| - витрати на гарантійне обслуговування; | 500 |
| - витрати на налагодження і експлуатацію; | 500 |
| - витрати на паливо-мастильні матеріали; | 1500 |
| - витрати на рекламу; | 2500 |
| - інші витрати на збут; | 600 |
| Разом витрат на збут: | 11850,5 |

*Таблиця 5.6*

**Зведений кошторис витрат на розробку проектного рішення (продукту)**

|  |  |
| --- | --- |
| **Статті витрат** | **Разом, грн.** |
| Сировина і матеріали | - |
| Купівельні напівфабрикати та комплектуючі вироби | 77602 |
| Зворотні відходи (вираховуються) | - |

*Продовження табл. 5.6*

|  |  |
| --- | --- |
| Паливо та електроенергія на технологічні цілі | 6000 |
| Основна заробітна плата | 102323,26 |
| Додаткова заробітна плата | - |
| Відрахування на соціальне страхування | 22511,12 |
| Витрати на утримання й експлуатацію устаткування | 1100 |
| Загальновиробничі витрати, у т.ч.: |  |
| - змінні; | 18724 |
| - постійні; | 25309,5 |
| Разом виробничих витрат: | 253570,37 |
| Адміністративні витрати | 20025,5 |
| Витрати на збут | 11850,5 |
| Інші операційні витрати | 2750 |
| Разом виробничих і операційних витрат: | 288196,37 |

Визначаємо вартість продукту, що розробляється за формулою:

*Ц = СБ \* Р+СБ* (5.2)

де *Ц* – ціна одинці продукту, грн.

*СБ* – собівартість продукту, грн.

*Р* – рентабельність виробництва, %.

*СБ* = 288196,37 грн.;

*Р* = задана рентабельність 40%;

Ц = 288 196,37 \* 0,4 + 288 196,37 = 403474,92 (грн.)

Оскільки, ПДВ для програмних продуктів 20%, кінцева ціна:

ЦзПДВ = Ц \* (1 + ПДВ) = 403474,92 \* (1 + 0,2) = 484169,90 (грн.)

*Таблиця 5.7*

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| **Показники** | **Сума, грн.** |
| 1 | 2 |
| Дохід від реалізації продукції | 484169,90 |
| Податок на додану вартість (20%) | 96833,98 |
| Чистий дохід від реалізації продукції | 387335,92 |
| Собівартість реалізованої продукції | 288196,37 |
| Валовий прибуток | 99139,55 |
| Операційні витрати: |  |
| - адміністративні витрати: | 20025,5 |
| - витрати на збут; | 11850,5 |
| Інші операційні витрати | 2750 |
| Фінансовий результат від операційної діяльності | 64513,55 |
| Податок на прибуток (18%) | 11612,44 |
| Чистий прибуток (збиток) | 52901,11 |

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Дане проектне рішення унікальне. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентоздатним.

5.6. Вибір стратегії

Основна мета програмного забезпечення – це надання користувачам можливості організації двосторонньої інтерактивної взаємодії студентів з викладачем на лекції.

З огляду на проведений аналіз українського ринку, можна зробити висновок, що даний програмний продукт є конкурентоспроможним, навіть при виході на ринок інших великих гравців, як Eleks чи SoftServe або виробників фізичних клікерсів (TurningPoint).

При розробці варто обрати стратегію розвитку нового продукту, оскільки на ринку немає подібного продукту, оскільки таким способом покращення навчального процесу тільки тепер зацікавилися університети в Україні. В якості ринкової стратегії варто обрати стратегію диверсифікації, оскільки система розробляється для нового сегменту.

Найвагоміший позитивний вплив на діяльність системи здійснюють користувачі та інфраструктура. Користувачі впливають найбільше бо саме вони визначають, які функції будуть у цього програмного забезпечення, а також, як саме воно буде використовуватися. Щодо інфраструктури то для використання подібної системи, університети повинні бути забезпеченні відповідною інфраструктурою – комп’ютери, проектори, покриття Wi-Fi.

Щодо найбільш негативно діючих чинників, то це є законодавчі акти, адже нестабільність у зміні законодавства щодо оподаткування, підприємницької діяльності та у сфері авторського права дуже негативно впливає на подальше планування та розвиток даного програмного продукту.

Серед внутрішніх факторів найбільш розвиненим є цілі, технологія та ресурси.

Найбільшу кількість витрат на виробництво становить основна зарплата, а також витрати матеріалів та комплектуючих виробів.

Собівартість продуктів становить 288196,37 грн. Отже, для коефіцієнту рентабельності в 40% ціна товару буде становити 403474,92 грн. Ціна товару з ПДВ – 484169,90 грн. Чистий дохід від реалізації продукції становить 52901,11 грн. Отже економічно доцільно розроблювати даний програмний продукт.

5.7 Висновки економічної оцінки веб-сервісу

Завершуючи розділ, можна зробити висновки, що був проведений економічний аналіз, який показав, що розробка подібного програмного забезпечення є доцільною, адже на ринку зараз існують лише застарілі та дорогі рішення, які значно поступаються системі, яка буде розроблена, тобто існують значні перспективи розвитку, а також великий попит, враховуючи низьку вартість порівняно з існуючими рішеннями.

ВИСНОВКИ

За результатами виконання бакалаврської кваліфікаційної роботи було проаналізовано сучасні IT-засоби для організації комунікації студента з викладачем на лекції, виходячи з якого можна зробити висновок, що існує потреба у створенні нового рішення, яке б дозволило вирішити основні недоліки існуючих продуктів (висока вартість, необхідність покупки для кожного студента спеціального пульта, фізичне обмеження на кількість кнопок і , відповідно, кількість можливих варіантів відповіді), і покращити вже існуючі функції.

Розроблене програмне забезпечення повністю відповідає вимогам вказаним у специфікації вимог, а також надає головні функції для організації комунікації студента з викладачем на лекції: створення питань для студентів викладачем, голосування за різні варіанти відповіді студентами, створення питань для викладача студентами.

Також для забезпечення коректності функціонування програмного забезпечення, а також перевірки повноти реалізації усіх функцій, дане програмне забезпечення було протестовано з допомогою створених модульних тестів, які створювалися відповідно до специфікації вимог, а також з допомогою статичного аналізу коду.

Перевагами розробленого програмного забезпечення є платформо незалежний і уніфікований REST API, який дозволяє використовувати серверну частину у будь-яких клієнтських технологіях (сучасні JavaScipt веб-фреймоворки, такі як Angular чи React, мобільні аплікації на платформах iOS, Android, Windows Phone, інше серверне забезпечення, яке може використовувати розроблену веб-систему у якості мікросервісу). Крім цього, у розробленому клієнтському сайті використовується Material Design, який забезпечує простий і зрозумілий графічний інтерфейс користувача.

У майбутньому планується створити відкритий API, а також розширення функцій, наприклад, персоналізація кожного студента, створення електронного щоденника відвідування та інші функції, які будуть запропоновані користувачами даної системи.

СПИСОК ОПРАЦЬОВАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. ClassroomResponse Systems ("Clickers"). [Електронний ресурс]. – Веб-доступ до сторінки: https://cft.vanderbilt.edu/guides-sub-pages/clickers/ (2017).
2. Classroom Response Systems [Електронний ресурс]. – Веб-доступ до сторінки: http://serc.carleton.edu/sp/library/classresponse/index.html (2017).
3. Clickerstechnologies [Електронний ресурс]. –Веб-доступ до сторінки: presswestern.uwo.ca/faculty.html#tech (2017).
4. Using Clickers in the Classroom Systems [Електронний ресурс]. – Веб-доступ до сторінки: https://www.youtube.com/watch?v=CnnP0uCqD4k (2017).
5. Тушницький Р. Б., Кулявець В. Р., Берегуляк Ю. Т. Програмне забезпечення для організації комунікації студентів з викладачем на лекції // Матеріали ІІ Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених “Інформаційні технології в моделюванні ІТМ-2017”. – Миколаїв, 2017. – С.181-182.
6. Django documentation [Електронний ресурс]. – Веб-доступ до сторінки: https://docs.djangoproject.com/en/ (2017).
7. The Django book [Електронний ресурс]. – Веб-доступ до сторінки: http://djangobook.com/ (2017).
8. Django REST Framework documentation [Електронний ресурс]. – Веб-доступ до сторінки: http://www.django-rest-framework.org/#api-guide (2017).
9. Representational state transfer [Електронний ресурс]. – Веб-доступ до сторінки: https://en.wikipedia.org/wiki/Representational\_state\_transfer (2017).
10. Material design [Електронний ресурс]. – Веб-доступ до сторінки: https://material.io/guidelines/material-design/introduction.html (2017).

ДОДАТОК А. ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Кулявець В. Р. |  |  | Діаграма варіантів використання | Сторінка |
| Перевірив | Тушницький Р. Б. |  |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-44 | | | |

ДОДАТОК Б. ДІАГРАМА РОЗГОРТУВАННЯ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Кулявець В. Р. |  |  | Діаграма розгортування | Сторінка |
| Перевірив | Тушницький Р. Б. |  |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-44 | | | |

ДОДАТОК В. ДІАГРАМА КЛАСІВ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав | Кулявець В. Р. |  |  | Діаграма класів | Сторінка |
| Перевірив | Тушницький Р. Б. |  |  | 1 з 1 |
| ІКНІ кафедра ПЗ ПІ-44 | | | |

ДОДАТОК Д. ЗВІТ ПРО ТЕСТУВАННЯ

Д.1 Вступ

Процес тестування є надзвичайно важливим під час створення програмного забезпечення адже завдяки ньому забезпечуються перевірка коректності роботи усіх функцій, забезпечується висока якість програмного забезпечення, а також перевірка роботи системи у штатних і нештатних випадках.

Для отримання вищезгаданих переваг, було обрано рішення про проведення трьох видів тестування:

* Модульного (unit-тести) – перевіряється коректність виконання різних функцій системи за повної ізоляції від інших частин системи.
* Статичний аналіз коду – перевіряється якість коду згідно з вибраними домовленостями щодо форматування коду, назв змінних, класів і т.д. (у випадку даного програмного забезпечення це PEP8). Підтримувати високу оцінку у даному виді тестування важливо для того, щоб код був легкий для читання і підтримки.
* Ручне тестування тестових випадків – це вий тестування буде застосований для перевірки виконання усіх вимог, вказаних у специфікації вимог, а також коректність їх функціонування.

Д.2 Розробка Unit-тестів

Для підвищення якості веб-системи, перевірки коректності роботи REST API і запобіганню поломки при подальших змінах, були створені такі unit-тести:

TestChoiceAPI – перевірка API для роботи з варіантами відповідей. Містить такі unit-тести:

* test\_changing\_choice – перевірка API на виконання запиту PUT і зміни об’єкта моделі Choice у базі даних.
* test\_creating\_choice – перевірка API на виконання запиту POST і створення нового об’єкта моделі Choice у базі даних.
* test\_deleting\_choice – перевірка API на виконання запиту DELETE і видалення об’єкта моделі Choice у базі даних.
* test\_getting\_all\_choice – перевірка API на виконання запиту GET і отримання усіх об’єктів моделі Choice, що містяться у базі даних.
* test\_getting\_specific\_choice – перевірка API на виконання запиту GET для конкретного id і отримання цього об’єктів моделі Choice, що містяться у базі даних.

TestLectureAPI – перевірка API для роботи з лекціями. Містить такі unit-тести:

* test\_changing\_lecture – перевірка API на виконання запиту PUT і зміни об’єкта моделі Lecture у базі даних.
* test\_creating\_lecture – перевірка API на виконання запиту POST і створення нового об’єкта моделі Lecture у базі даних.
* test\_deleting\_lecture – перевірка API на виконання запиту DELETE і видалення об’єкта моделі Lecture у базі даних.
* test\_getting\_all\_lecture – перевірка API на виконання запиту GET і отримання усіх об’єктів моделі Lecture, що містяться у базі даних.
* test\_getting\_specific\_lecture – перевірка API на виконання запиту GET для конкретного id і отримання цього об’єктів моделі Lecture, що містяться у базі даних.

TestQuestionAPI – перевірка API для роботи з питаннями. Містить такі unit-тести:

* test\_changing\_question – перевірка API на виконання запиту PUT і зміни об’єкта моделі Question у базі даних.
* test\_creating\_question – перевірка API на виконання запиту POST і створення нового об’єкта моделі Question у базі даних.
* test\_deleting\_question – перевірка API на виконання запиту DELETE і видалення об’єкта моделі Question у базі даних.
* test\_getting\_all\_question – перевірка API на виконання запиту GET і отримання усіх об’єктів моделі Question, що містяться у базі даних.
* test\_getting\_specific\_question – перевірка API на виконання запиту GET для конкретного id і отримання цього об’єктів моделі Question, що містяться у базі даних.

TestStudentQuestionAPI – перевірка API для роботи з студентськими питаннями. Містить такі unit-тести:

* test\_creating\_student\_question – перевірка API на виконання запиту POST і створення нового об’єкта моделі StudentQuestion у базі даних.
* test\_deleting\_student\_question – перевірка API на виконання запиту DELETE і видалення об’єкта моделі StudentQuestion у базі даних.
* test\_getting\_specific\_student\_question – перевірка API на виконання запиту GET для конкретного id і отримання цього об’єктів моделі StudentQuestion, що містяться у базі даних.

TestSubjectAPI – перевірка API для роботи з навчальними предметами. Містить такі unit-тести:

* test\_changing\_subject – перевірка API на виконання запиту PUT і зміни об’єкта моделі Subject у базі даних.
* test\_creating\_subject – перевірка API на виконання запиту POST і створення нового об’єкта моделі Subject у базі даних.
* test\_deleting\_subject – перевірка API на виконання запиту DELETE і видалення об’єкта моделі Subject у базі даних.
* test\_getting\_all\_subject – перевірка API на виконання запиту GET і отримання усіх об’єктів моделі Subject, що містяться у базі даних.
* test\_getting\_specific\_subject – перевірка API на виконання запиту GET для конкретного id і отримання цього об’єктів моделі Subject, що містяться у базі даних.

Д.3 Розробка тестових випадків

Для забезпечення виконання усіх вимог, вказаних у специфікації вимог, а також коректність їх функціонування були створенні такі тестові випадки:

ID 1 “Реєстрація”

|  |  |
| --- | --- |
| Короткий опис:реєстрація у системі користувача | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває сторінку реєстрації. 2. Користувач заповнює необхідні поля. 3. Користувач підтверджує реєстрацію. | 1. Відкривається сторінка з формою реєстрації. 2. Система перевіряє коректність введеної інформації (якщо інформація некоректна, система повідомляє про це). 3. Система повідомляє про результат реєстрації. |

ID 2 “Вхід у систему”

|  |  |
| --- | --- |
| Короткий опис: вхід у систему користувача | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває сторінку входу. 2. Користувач заповнює поле «Електронна пошта», «Пароль». 3. Користувач підтверджує вхід. | 1. Відкривається сторінка з формою входу. 2. Система перевіряє коректність введеної інформації (якщо інформація некоректна, система повідомляє про це). 3. Система повідомляє про результат входу: якщо дані для входу коректні, то система переадресовує користувача на сторінку з навчальними предметами, інакше відображається повідомлення про помилку. |

ID 3 “Створення предмету”

|  |  |
| --- | --- |
| Короткий опис:створення нового навчального предмету | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває сторінку з списком навчальних предметів. 2. Користувач натискає кнопку «Створити». 3. Користувач вводить назву предмета. 4. Користувач вводить навчальні групи, для яких ведеться цей предмет. 5. Користувач натискає кнопку «Створити». | 1. Відображається сторінка з списком навчальних предметів. 2. Відображається сторінка з формою створення нового предмету. 3. Назва предмета з’являється у відповідному полі і перевіряється на коректність. 4. Назва навчальних груп з’являються у відповідному полі і перевіряються на коректність. 5. Система переходить на сторінку .з списком предметів вже з створеним предметом. |

ID 4 “Створення лекцій”

|  |  |
| --- | --- |
| Короткий опис:створення нової лекції | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває сторінку з списком лекцій. 2. Користувач натискає кнопку «Створити». 3. Користувач вводить назву лекції. 4. Користувач вибирає навчальний предмет зі списку, до якого відноситься дана лекція. 5. Користувач вводить пароль до лекції. 6. Користувач вводить місце проведення лекції. 7. Користувач вводить номер заняття. 8. Користувач вибирає дату проведення лекції. 9. Користувач натискає кнопку «Створити». | 1. Відображається сторінка з списком лекцій. 2. Відображається сторінка з формою створення нової лекції. 3. Назва лекції з’являється у відповідному полі і перевіряється на коректність. 4. Вибраний навчальний список з’являється у відповідному полі. 5. Пароль у відкритому вигляді з’являється у відповідному полі і перевіряється на коректність. 6. Місце проведення лекції з’являється у відповідному полі і перевіряється на коректність. 7. Номер заняття з’являється у відповідному полі і перевіряється на коректність. 8. Дата проведення лекції з’являється у відповідному полі і перевіряється на коректність. 9. Система переходить на сторінку .з списком лекцій вже з створеною лекцію. |

ID 5 “Створення питань”

|  |  |
| --- | --- |
| Короткий опис:створення питань для студентів викладачем | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває список питань у потрібній лекції. 2. Користувач натискає кнопку «Створити». 3. Користувач вводить текст питання. 4. Користувач вибирає, чи студенти можуть відповідати на декілька варіантів відповіді. 5. Користувач створює два варіанти відповіді на питання. 6. Користувач натискає кнопку «Створити». | 1. Відображається сторінка з списком питань, які відносяться до вибраної лекції. 2. Відображається сторінка з формою створення нового питання. 3. Текст питання з’являється у відповідному полі і перевіряється на коректність. 4. У полі, яке відповідає чи можуть студенти голосувати за декілька варіантів відповіді з’являється помітка. 5. Текст з двома варіантами відповіді з’являється у відповідних полях і перевіряються на коректність. 6. Система переходить на сторінку .з списком питань, які відносяться до вибраної лекції вже з створеним питанням. |

ID 6 “Перегляд питань”

|  |  |
| --- | --- |
| Короткий опис: перегляд питань від студентів викладачем | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває список питань створених студентами у вибраній лекції. 2. Користувач переглядає їх. | 1. Відображається сторінка з списком питань, які створили студенти і вони відносяться до вибраної лекції. 2. Відображаються вже існуючі питання і у реальному часі з’являються нові питання |

ID 7 “Видалення питань від студентів”

|  |  |
| --- | --- |
| Короткий опис:видалення питань від студентів | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває список питань створених студентами у вибраній лекції. 2. Користувач видаляє питання, які він вважає недоречними. | 1. Відображається сторінка з списком питань, які створили студенти і вони відносяться до вибраної лекції. 2. Видалені питання зникають з списку. |

ID 8 “Перегляд графіків по питаннях”

|  |  |
| --- | --- |
| Короткий опис: перегляд графіків по відповідях студентів на вибране питання | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває список питань у потрібній лекції. 2. Користувач натискає на кнопку «Графік». | 1. Відображається сторінка з списком питань, які відносяться до вибраної лекції. 2. Відображається кругова діаграма, якщо кількість варіантів відповіді два, якщо більше, то стовпчаста. |

ID 9 “Генерація QR-коду”

|  |  |
| --- | --- |
| Короткий опис:генерація QR-коду для входу студентам у систему | |
| Кроки | Очікуваний результат |
| 1. Користувач відкриває список лекцій. 2. Користувач у потрібній лекції натискає на кнопку «QR-код». | 1. Відображається сторінка з списком лекцій. 2. Відкривається нова вкладка веб-браузера, де відображається QR-код, який містить закодований пароль до лекції. |

Д.4 Результати тестування

**Д.4.1 Результати тестування на Unit-тести**

Як можна побачити на рис. Д.1., усі модульні тести, пройшли перевірку, тому можна вважати, що веб-система функціонує коректно.

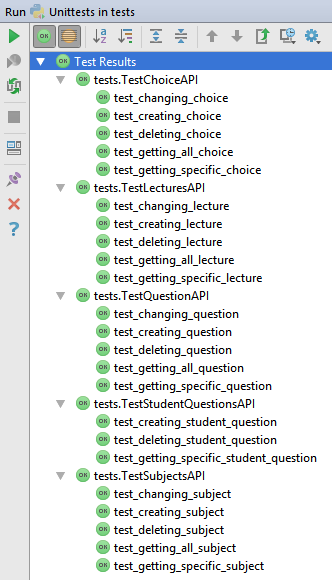


Рис. Д.1. Результати модульного тестування

**Д.4.2 Результати тестування на статичний аналіз коду**

Статичний аналіз коду здійснювався за допомогою спеціальної програми PyLint, яка перевіряє код згідно з домовленостями щодо форматування, назв змінних, класів та функцій, які прописані у спеціальному документів PEP8 (<https://www.python.org/dev/peps/pep-0008/>). Також, для адаптації перевірки до коду веб-фреймоврку Django був використаний пагін pylint\_django.

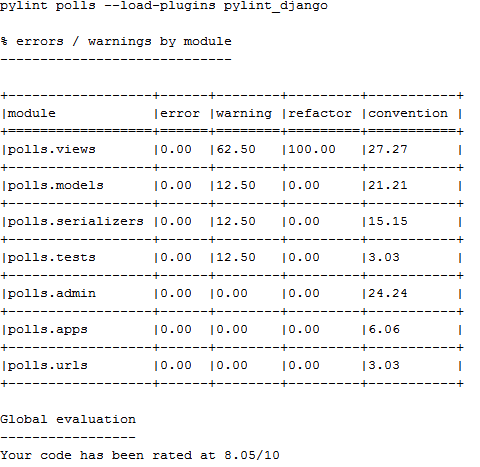


Рис. Д.2. Результати тестування на статичний аналіз коду

Результати статичного аналізу коду зображенні на рис. Д.2., і можна побачити оцінку 8.05 з 10 можливих, що говорить про високу якість написаного коду.

**Д.4.3 Результати тестування тестових випадків**

ID 1 “Реєстрація”

|  |  |  |
| --- | --- | --- |
| Короткий опис:реєстрація у системі користувача | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває сторінку реєстрації. 2. Користувач заповнює необхідні поля. 3. Користувач підтверджує реєстрацію. | 1. Відкривається сторінка з формою реєстрації. 2. Система перевіряє коректність введеної інформації (якщо інформація некоректна, система повідомляє про це). 3. Система повідомляє про результат реєстрації. | 1. Пройшов. 2. Пройшов. 3. Пройшов. |

ID 2 “Вхід у систему”

|  |  |  |
| --- | --- | --- |
| Короткий опис: вхід у систему користувача | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває сторінку входу. 2. Користувач заповнює поле «Електронна пошта», «Пароль». 3. Користувач підтверджує вхід. | 1. Відкривається сторінка з формою входу. 2. Система перевіряє коректність введеної інформації (якщо інформація некоректна, система повідомляє про це). 3. Система повідомляє про результат входу: якщо дані для входу коректні, то система переадресовує користувача на сторінку з навчальними предметами, інакше відображається повідомлення про помилку. | 1. Пройшов. 2. Пройшов. 3. Пройшов. |

ID 3 “Створення предмету”

|  |  |  |
| --- | --- | --- |
| Короткий опис:створення нового навчального предмету | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває сторінку з списком навчальних предметів. 2. Користувач натискає кнопку «Створити». 3. Користувач вводить назву предмета. 4. Користувач вводить навчальні групи, для яких ведеться цей предмет. 5. Користувач натискає кнопку «Створити». | 1. Відображається сторінка з списком навчальних предметів. 2. Відображається сторінка з формою створення нового предмету. 3. Назва предмета з’являється у відповідному полі і перевіряється на коректність. 4. Назва навчальних груп з’являються у відповідному полі і перевіряються на коректність. 5. Система переходить на сторінку .з списком предметів вже з створеним предметом. | 1. Пройшов. 2. Пройшов. 3. Пройшов. 4. Пройшов. 5. Пройшов. |

ID 4 “Створення лекцій”

|  |  |  |
| --- | --- | --- |
| Короткий опис:створення нової лекції | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває сторінку з списком лекцій. 2. Користувач натискає кнопку «Створити». 3. Користувач вводить назву лекції. 4. Користувач вибирає навчальний предмет зі списку, до якого відноситься дана лекція. 5. Користувач вводить пароль до лекції. 6. Користувач вводить місце проведення лекції. 7. Користувач вводить номер заняття. 8. Користувач вибирає дату проведення лекції. 9. Користувач натискає кнопку «Створити». | 1. Відображається сторінка з списком лекцій. 2. Відображається сторінка з формою створення нової лекції. 3. Назва лекції з’являється у відповідному полі і перевіряється на коректність. 4. Вибраний навчальний список з’являється у відповідному полі. 5. Пароль у відкритому вигляді з’являється у відповідному полі і перевіряється на коректність. 6. Місце проведення лекції з’являється у відповідному полі і перевіряється на коректність. 7. Номер заняття з’являється у відповідному полі і перевіряється на коректність. 8. Дата проведення лекції з’являється у відповідному полі і перевіряється на коректність. 9. Система переходить на сторінку .з списком лекцій вже з створеною лекцію. | 1. Пройшов. 2. Пройшов. 3. Пройшов. 4. Пройшов. 5. Пройшов. 6. Пройшов. 7. Пройшов. 8. Пройшов. 9. Пройшов. |

ID 5 “Створення питань”

|  |  |  |
| --- | --- | --- |
| Короткий опис:створення питань для студентів викладачем | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває список питань у потрібній лекції. 2. Користувач натискає кнопку «Створити». 3. Користувач вводить текст питання. 4. Користувач вибирає, чи студенти можуть відповідати на декілька варіантів відповіді. 5. Користувач створює два варіанти відповіді на питання. 6. Користувач натискає кнопку «Створити». | 1. Відображається сторінка з списком питань, які відносяться до вибраної лекції. 2. Відображається сторінка з формою створення нового питання. 3. Текст питання з’являється у відповідному полі і перевіряється на коректність. 4. У полі, яке відповідає чи можуть студенти голосувати за декілька варіантів відповіді з’являється помітка. 5. Текст з двома варіантами відповіді з’являється у відповідних полях і перевіряються на коректність. 6. Система переходить на сторінку .з списком питань, які відносяться до вибраної лекції вже з створеним питанням. | 1. Пройшов. 2. Пройшов. 3. Пройшов. 4. Пройшов. 5. Пройшов. 6. Пройшов. |

ID 6 “Перегляд питань”

|  |  |  |
| --- | --- | --- |
| Короткий опис: перегляд питань від студентів викладачем | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває список питань створених студентами у вибраній лекції. 2. Користувач переглядає їх. | 1. Відображається сторінка з списком питань, які створили студенти і вони відносяться до вибраної лекції. 2. Відображаються вже існуючі питання і у реальному часі з’являються нові питання | 1. Пройшов. 2. Пройшов. |

ID 7 “Видалення питань від студентів”

|  |  |  |
| --- | --- | --- |
| Короткий опис:видалення питань від студентів | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває список питань створених студентами у вибраній лекції. 2. Користувач видаляє питання, які він вважає недоречними. | 1. Відображається сторінка з списком питань, які створили студенти і вони відносяться до вибраної лекції. 2. Видалені питання зникають з списку. | 1. Пройшов. 2. Пройшов. |

ID 8 “Перегляд графіків по питаннях”

|  |  |  |
| --- | --- | --- |
| Короткий опис: перегляд графіків по відповідях студентів на вибране питання | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває список питань у потрібній лекції. 2. Користувач натискає на кнопку «Графік». | 1. Відображається сторінка з списком питань, які відносяться до вибраної лекції. 2. Відображається кругову діаграму, якщо кількість варіантів відповіді два, якщо більше, то стовпчасту. | 1. Пройшов. 2. Пройшов. |

ID 9 “Генерація QR-коду”

|  |  |  |
| --- | --- | --- |
| Короткий опис:генерація QR-коду для входу студентам у систему | | |
| Кроки | Очікуваний результат | Результат |
| 1. Користувач відкриває список лекцій. 2. Користувач у потрібній лекції натискає на кнопку «QR-код». | 1. Відображається сторінка з списком лекцій. 2. Відкривається нова вкладку веб-браузера, де відображається QR-код, який містить закодований пароль до лекції. | 1. Пройшов. 2. Пройшов. |

Д.5 Висновок

За результатами проведеного тестування, можна зробити висновок, що розроблене програмне забезпечення функціонує цілком коректно.

ДОДАТОК Е. ІНСТРУКЦІЯ КОРИСТУВАЧА

Е.1 Компоненти ПЗ

Веб-система була розроблена на мові програмування Python з допомогою веб-фреймовку Django та бібліотеки Django REST Framework, для розробки використовувалося IDE від JetBrains PyCharm.

Створена веб-система повністю функціонує на комп’ютерах і серверах з операційною системою Ubuntu. Для коректної роботи, середовище функціонування повинне відповідати таким параметрам:

* Операційна система Ubuntu версії не нижче 16.04.
* Встановлена мова програмування Python не нижче версії 2.7.11.
* Встановлена СУБД PostgreSQL не нижче версії 9.4.
* Процесор з тактовою частотою не нижче 2 ГГц.
* Об’єм оперативної пам’яті не нижче 6 Гб.

Е.2 Встановлення ПЗ

Для роботи веб-системи, необхідно, щоб на сервері були встановлені і налаштовані веб-сервер NGINX, інтерпретатор мови програмування Python 2.7, СУБД PostgreSQL, пакетний менеджер bower.

Для встановлення веб-системи, необхідно скопіювати папку з проектом на жорсткий диск системи, у терміналі перейти у дерикторію цієї папки і виконати команди:

* pip install –r requeirements.txt
* bower install
* python manage.py migrate
* python manage.py collectstatic

Після виконання цих команд веб-система повністю встановлена.

Е.3 Налаштування ПЗ

Після встановлення програмного забезпечення, необхідно налаштувати підключення до бази даних PostgreSQL, для цього потрібно у файлі settings.py, який знаходиться у папці clickers у змінній DATABASES прописати назву бази даних, адресу СУБД, назву користувача і пароль користувача.

Е.4 Базові функції ПЗ

Найголовнішими функціями веб-системи є створення питань для студентів та перегляд результатів відповідей студентів на питання.

**Е. 4.1. Вхід у веб-систему**

1. Перейти на головну сторінку веб-системи.
2. У формі входу ввести логін або електронну скриньку і пароль.
3. Натиснути кнопку “SIGN IN”.

**Е. 4.2. Реєстрація у веб-системі**

1. Перейти на головну сторінку веб-системи.
2. У панелі зверху у правому кутку натиснути на кнопку “Register”.
3. У формі реєстрації ввести електронну скриньку і пароль.
4. Натиснути кнопку “REGISTER”.

**Е. 4.3. Створення предметів**

1. Увійти у веб-систему.
2. Вибрати у верхньому меню пункт “LIST OF SUBJECTS”.
3. У панелі предметів зверху у правому кутку натиснути на кнопку “ADD NEW”.
4. У формі створення ввести усю необхідну інформацію, яка відображається на формі.
5. Натиснути кнопку “CREATE”.

**Е. 4.3. Створення лекцій**

1. Увійти у веб-систему.
2. Вибрати у верхньому меню пункт “LIST OF LECTURES”.
3. У панелі лекцій зверху у правому кутку натиснути на кнопку “ADD NEW”.
4. У формі створення ввести усю необхідну інформацію, яка відображається на формі.
5. Натиснути кнопку “CREATE”.

**Е. 4.3. Створення питань для студентів**

1. Увійти у веб-систему.
2. Вибрати у верхньому меню пункт “LIST OF LECTURES”.
3. У потрібній лекції у правій стороні картки натиснути на кнопку “QUESTIONS”.
4. У панелі лекцій зверху у правому кутку натиснути на кнопку “ADD NEW”.
5. У формі створення ввести усю необхідну інформацію, яка відображається на формі.
6. Натиснути кнопку “CREATE”.

**Е. 4.3. Перегляд результатів відповідей студентів на питання**

1. Увійти у веб-систему.
2. Вибрати у верхньому меню пункт “LIST OF LECTURES”.
3. У потрібній лекції у правій стороні картки натиснути на кнопку “QUESTIONS”.
4. У потрібному питанні у правій стороні картки натиснути на кнопку “PLOT”.
5. Переглянути сформований графік.

Е. 5 Обробка помилкових ситуацій

Усі операції які відбуваються у системі логуються, тому при виникненні будь-яких виключних ситуацій можна переглянути детальнішу інформацію про помилку у консолі браузера (якщо проблема відноситься до клієнтського сайту веб-системи) або у консолі сервера, де розгорнута і запущена веб-система (якщо проблема відноситься до серверної частини веб-системи).

ДОДАТОК Ж. ФРАГМЕНТИ КОДУ ПРОГРАМИ

Ж.1. Лістинг файлу models.py.

from \_\_future\_\_ import unicode\_literals

from django.db import models

from django.contrib.auth.models import User

from django.utils import timezone

class Subject(models.Model):

name = models.CharField(max\_length=200)

user = models.ForeignKey(User)

create\_date = models.DateTimeField(null=True, auto\_now\_add=True)

update\_date = models.DateTimeField(null=True, auto\_now=True)

groups = models.CharField(max\_length=200, null=True)

def \_\_unicode\_\_(self):

return self.name

class Lecture(models.Model):

subject = models.ForeignKey(Subject, related\_name='lectures')

user = models.ForeignKey(User)

lecture\_name = models.CharField(max\_length=200)

secret\_key = models.CharField(max\_length=200)

create\_date = models.DateTimeField(null=True, auto\_now\_add=True)

update\_date = models.DateTimeField(null=True, auto\_now=True)

place = models.CharField(max\_length=200, null=True)

start\_time = models.TimeField(null=True)

date = models.DateTimeField(null=True)

lesson\_number = models.IntegerField(null=True)

def \_\_unicode\_\_(self):

return self.lecture\_name

class Question(models.Model):

lecture = models.ForeignKey(Lecture, on\_delete=models.CASCADE, related\_name='questions')

question\_text = models.CharField(max\_length=200)

multiply\_choices = models.BooleanField(default=False)

def \_\_unicode\_\_(self):

return self.question\_text

class Choice(models.Model):

question = models.ForeignKey(Question, on\_delete=models.CASCADE, related\_name='choices')

choice\_text = models.CharField(max\_length=200)

votes = models.IntegerField(default=0)

def vote(self):

self.votes = models.F('votes') + 1 # avoiding race condition

self.save()

def \_\_unicode\_\_(self):

return self.choice\_text

class StudentQuestion(models.Model):

lecture = models.ForeignKey(Lecture, on\_delete=models.CASCADE, related\_name='student\_questions')

question\_text = models.CharField(max\_length=200)

def \_\_unicode\_\_(self):

return self.question\_text

Ж.2. Лістинг файлу serializers.py.

# -\*- coding: utf-8 -\*-

from rest\_framework import serializers

from polls.models import Question, Choice, Lecture, Subject, StudentQuestion

from datetime import datetime

class ChoiceSerializer(serializers.ModelSerializer):

"""

Serializer answer choices

"""

votes = serializers.IntegerField(default=0, required=False)

class Meta:

model = Choice

fields = ('id', 'choice\_text', 'votes', 'question')

def update(self, instance, validated\_data):

"""

Update and return an existing "Choice" instance, given the validated data.

"""

instance.choice\_text = validated\_data.get('choice\_text', instance.choice\_text)

instance.save()

return instance

class QuestionSerializer(serializers.ModelSerializer):

"""

Serializer for Question

"""

id = serializers.IntegerField(read\_only=True)

question\_text = serializers.CharField(max\_length=200)

choices = ChoiceSerializer(many=True, required=False)

class Meta:

model = Question

fields = ('id', 'question\_text', 'choices', 'lecture', 'multiply\_choices')

def create(self, validated\_data):

question = Question(\*\*validated\_data)

question.save()

return question

def update(self, instance, validated\_data):

"""

Update and return an existing "Question" instance, given the validated data.

"""

instance.question\_text = validated\_data.get('question\_text', instance.question\_text)

instance.save()

return instance

class StudentQuestionSerializer(serializers.ModelSerializer):

"""

Serializer for student questions

"""

class Meta:

model = StudentQuestion

fields = ('id', 'lecture', 'question\_text')

class LectureListSerializer(serializers.ModelSerializer):

"""

Serializer for listing Lecture objects

"""

questions = QuestionSerializer(many=True, required=False)

student\_questions = StudentQuestionSerializer(many=True, required=False)

class Meta:

model = Lecture

fields = ('id', 'lecture\_name', 'questions', 'student\_questions', 'secret\_key', 'create\_date', 'update\_date', 'subject', 'place',

'start\_time', 'date', 'lesson\_number')

def create(self, validated\_data):

lecture = Lecture(\*\*validated\_data)

lecture.user = self.context['request'].user

lecture.save()

return lecture

class LectureDetailSerializer(serializers.ModelSerializer):

"""

Serializer for Lecture object details

"""

questions = QuestionSerializer(many=True, required=False)

student\_questions = StudentQuestionSerializer(many=True, required=False)

class Meta:

model = Lecture

fields = ('id', 'lecture\_name', 'questions', 'student\_questions', 'secret\_key', 'create\_date', 'update\_date', 'subject', 'place',

'start\_time', 'date', 'lesson\_number')

def update(self, instance, validated\_data):

"""

Update and return an existing "Lecture" instance, given the validated data.

"""

instance.lecture\_name = validated\_data.get('lecture\_name', instance.lecture\_name)

instance.secret\_key = validated\_data.get('secret\_key', instance.secret\_key)

instance.subject = validated\_data.get('subject', instance.subject)

instance.place = validated\_data.get('place', instance.place)

instance.start\_time = validated\_data.get('start\_time', instance.start\_time)

instance.date = validated\_data.get('date', instance.date)

instance.lesson\_number = validated\_data.get('lesson\_number', instance.lesson\_number)

instance.save()

return instance

class SubjectListSerializer(serializers.ModelSerializer):

"""

Serializer for listing Subject objects

"""

user = serializers.PrimaryKeyRelatedField(

read\_only=True,

default=serializers.CurrentUserDefault()

)

class Meta:

model = Subject

fields = ('id', 'name', 'user', 'create\_date', 'update\_date', 'groups')

def create(self, validated\_data):

subject = Subject(\*\*validated\_data)

subject.user = self.context['request'].user

subject.save()

return subject

class SubjectDetailSerializer(serializers.ModelSerializer):

"""

Serializer for Subject object details

"""

lectures = LectureListSerializer(many=True, required=False)

class Meta:

model = Subject

fields = ('id', 'name', 'user', 'create\_date', 'update\_date', 'groups', 'lectures')

def update(self, instance, validated\_data):

"""

Update and return an existing "Subject" instance, given the validated data.

"""

instance.name = validated\_data.get('name', instance.name)

instance.groups = validated\_data.get('groups', instance.groups)

instance.save()

return instance

Ж.3. Лістинг файлу urls.py.

# -\*- coding: utf-8 -\*-

from polls import views

from django.conf.urls import url

urlpatterns = [

url(r'^polls/$', views.QuestionList.as\_view()),

url(r'^polls/(?P<pk>\d+)/$', views.QuestionDetails.as\_view()),

url(r'^choices/$', views.ChoiceList.as\_view()),

url(r'^choices/(?P<pk>\d+)/$', views.ChoiceDetails.as\_view()),

url(r'^choices/(?P<pk>\d+)/vote/$', views.Voting.as\_view()),

url(r'^lectures/$', views.LectureList.as\_view()),

url(r'^lectures/(?P<pk>\d+)/$', views.LectureDetails.as\_view()),

url(r'^subjects/$', views.SubjectList.as\_view()),

url(r'^subjects/(?P<pk>\d+)/$', views.SubjectDetails.as\_view()),

url(r'^student-questions/$', views.StudentQuestionList.as\_view()),

url(r'^student-questions/(?P<pk>\d+)/$', views.StudentQuestionDetails.as\_view()),

]

Ж.4. Лістинг файлу views.py.

from django.shortcuts import render\_to\_response

from django.views.generic.base import View

from django.http import HttpResponse, HttpResponseBadRequest

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework import status, generics

from rest\_framework.permissions import IsAuthenticatedOrReadOnly, AllowAny

from polls.models import Question, Choice, Lecture, Subject, StudentQuestion

from polls.serializers import QuestionSerializer, ChoiceSerializer, LectureListSerializer, LectureDetailSerializer, \

SubjectDetailSerializer, SubjectListSerializer, StudentQuestionSerializer

class QuestionList(generics.ListCreateAPIView):

"""

List all Questions

"""

serializer\_class = QuestionSerializer

def get\_queryset(self):

return Question.objects.all()

class QuestionDetails(generics.RetrieveUpdateDestroyAPIView):

"""

Retrieve, update or delete a Question instance.

"""

serializer\_class = QuestionSerializer

permission\_classes = (IsAuthenticatedOrReadOnly, )

def put(self, request, \*args, \*\*kwargs):

return self.partial\_update(request, \*args, \*\*kwargs)

def get\_queryset(self):

return Question.objects.all()

class ChoiceList(generics.ListCreateAPIView):

"""

List all Choices.

"""

serializer\_class = ChoiceSerializer

permission\_classes = (IsAuthenticatedOrReadOnly,)

def get\_queryset(self):

return Choice.objects.all()

class ChoiceDetails(generics.RetrieveUpdateDestroyAPIView):

"""

Update choice instance.

"""

serializer\_class = ChoiceSerializer

permission\_classes = (IsAuthenticatedOrReadOnly,)

def put(self, request, \*args, \*\*kwargs):

return self.partial\_update(request, \*args, \*\*kwargs)

def get\_queryset(self):

return Choice.objects.all()

class Voting(APIView):

"""

View used for handling voting

"""

permission\_classes = (AllowAny,)

def post(self, request, \*args, \*\*kwargs):

id = kwargs.get("pk", None)

if id is None:

return HttpResponseBadRequest("Invalid input data. Please edit and try again.")

Choice.objects.get(id = id).vote()

return HttpResponse(status=200)

class IndexView(View):

def get(self, request):

return render\_to\_response('index.html')

class EmptyView(View):

def get(self, request):

return render\_to\_response('empty.html')

class LectureList(generics.ListCreateAPIView):

"""

List all Lectures

To get lectures list need to provide secret\_key param

In other case if user is not anonymous query will return all lectures created by user

"""

serializer\_class = LectureListSerializer

permission\_classes = (IsAuthenticatedOrReadOnly,)

def get\_queryset(self):

queryset = Lecture.objects.none()

secret\_key = self.request.query\_params.get('secret\_key', None)

user = self.request.user

if secret\_key is not None:

queryset = Lecture.objects.filter(secret\_key=secret\_key)

if not user.is\_anonymous:

queryset = Lecture.objects.filter(user=user)

return queryset

class LectureDetails(generics.RetrieveUpdateDestroyAPIView):

"""

Retrieve, update or delete a Lecture instance.

To get lecture need to provide secret\_key param

In other case if user is not anonymous query will return lecture if it's created by user

"""

serializer\_class = LectureDetailSerializer

permission\_classes = (IsAuthenticatedOrReadOnly,)

def put(self, request, \*args, \*\*kwargs):

return self.partial\_update(request, \*args, \*\*kwargs)

def get\_queryset(self):

queryset = Lecture.objects.none()

secret\_key = self.request.query\_params.get('secret\_key', None)

user = self.request.user

if secret\_key is not None:

queryset = Lecture.objects.filter(secret\_key=secret\_key)

if not user.is\_anonymous:

queryset = Lecture.objects.filter(user=user)

return queryset

class SubjectList(generics.ListCreateAPIView):

"""

List all Subjects

"""

serializer\_class = SubjectListSerializer

permission\_classes = (IsAuthenticatedOrReadOnly,)

def get\_queryset(self):

queryset = Subject.objects.none()

user = self.request.user

if not user.is\_anonymous:

queryset = Subject.objects.filter(user=user)

return queryset

class SubjectDetails(generics.RetrieveUpdateDestroyAPIView):

"""

Retrieve, update or delete a Subject instance.

"""

serializer\_class = SubjectDetailSerializer

permission\_classes = (IsAuthenticatedOrReadOnly,)

def put(self, request, \*args, \*\*kwargs):

return self.partial\_update(request, \*args, \*\*kwargs)

def get\_queryset(self):

queryset = Subject.objects.none()

user = self.request.user

if not user.is\_anonymous:

queryset = Subject.objects.filter(user=user)

return queryset

class StudentQuestionList(generics.ListCreateAPIView):

"""

Getting list of student questions by lecture and creating student questions

"""

serializer\_class = StudentQuestionSerializer

permission\_classes = (AllowAny, )

def get\_queryset(self):

queryset = StudentQuestion.objects.none()

lecture = self.request.query\_params.get('lecture', None)

if lecture is not None:

queryset = StudentQuestion.objects.filter(lecture=lecture)

return queryset

class StudentQuestionDetails(generics.DestroyAPIView):

"""

Deleting student questions

"""

serializer\_class = StudentQuestionSerializer

permission\_classes = (IsAuthenticatedOrReadOnly, )

queryset = StudentQuestion.objects.all()