**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра програмного забезпечення**

**Звіт**  
про переддипломну практику за темою бакалаврської кваліфікаційної роботи

**Програмне забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина)**

тема

студента (ки) IV курсу групи ПІ-44

Кулявця В.Р.

(прізвище, ініціали) 

**база практики ФОП Гулий А.М.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва підприємства)

**термін практики** з «03.03.2017» до «29.04.2017»   
 

**Керівники практики**

від кафедри \_\_\_\_\_\_\_ доцент Левус Є.В.

(підпис) (посада, ПІП)

від бази практики \_\_\_\_\_\_\_ Гулий А.М.

(підпис) (посада, ПІП)   
 

**Керівник бакалаврської** \_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доцент Тушницький Р.Б.

(підпис) (наук.ст., вч.звання, ПІП)

**кваліфікаційної роботи**

**Оцінка**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Дата\_**\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 2017

ЗМІСТ

[1. Опис бази практики 3](#_Toc481699527)

[2. Завдання отримане на базі практики 3](#_Toc481699528)

[3. Результат виконання завдання на базі практики 3](#_Toc481699529)

[3.1 Ознайомлення з веб-фреймоворком на Python - Django 3](#_Toc481699530)

[3.2 Розробка застосування з використанням веб-фреймоворку Django 5](#_Toc481699531)

[4. Завдання отримане стосовно дипломного проекту 8](#_Toc481699532)

[4.1. Огляд IT – засобів для організації комунікації студентів з викладачем на лекції 8](#_Toc481699533)

[4.2. Постановка завдання для розробки програмного забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина) 13](#_Toc481699534)

[4.3 Специфікація вимог до програмного забезпечення для організації комунікації студентів з викладачем на лекції (серверна частина) 14](#_Toc481699535)

[5. Проектування архітектури застосування 21](#_Toc481699536)

[5.1 Проектування архітектури 21](#_Toc481699537)

[5.2 Вибір інструментальних засобів для реалізації веб-системи 24](#_Toc481699538)

[5.3 Проектування графічного інтерфейсу користувача 24](#_Toc481699539)

[6. Висновки про отримані під час практики результати 27](#_Toc481699540)

[СПИСОК ОПРАЦЬОВАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 28](#_Toc481699541)

## Опис бази практики

ФОП Гулий А.М. – це аутсоурсингова компанія з розробки програмного забезпечення, яка знаходиться у Львові. Основним напрямком є розробка веб-сайтів або додатків для мобільних платформ на операційних системах Android та iOS. Щодо розробки веб-сайтів, то клієнтські частини будуються на основі сучасних JavaScript веб-фреймоврків Angular та React. Щодо серверної частини для своїх проектів, компанія використовує Python, NodeJS та Ruby on Rails. Основна методологія розробки програмного забезпечення, яка є впроваджена у компанії це Scrum, хоча є команди програмістів, які працюють за методологіями Kanban і Waterfall. Найбільшими клієнтами компанії є інші великі компанії, які потребують програмного забезпечення типу B2B (Business-to-Business), хоча присутні індивідуальні клієнти та стартапи.

## Завдання отримане на базі практики

Згідно переддипломної практики на базі практики ФОП Гулий А.М., було отримано наступні завдання:

1. Ознайомитися з веб-фреймоворком на Python - Django.
2. Розробити прототип програмного забезпечення для організації комунікації студентів з викладачем на лекції.

## Результат виконання завдання на базі практики

### Ознайомлення з веб-фреймоворком на Python - Django

Django - це високорівневий фреймворк з відкритими вихідними кодами написаний на мові програмування Python призначення для побудови веб-систем [1].

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), основна відмінність 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 [2].

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.

Враховуючи вищезазначене хочеться виділити такі модулі Django:

* **ORM**

ORM (Object-relational mapping) – це технологія програмування, яка зв’язує базу даних з об’єктами мов програмування, які підтримують об’єктно-орієнтовану парадигму розробки програмного забезпечення. Об’єкти бази даних у Django іменуються моделями і він дозволяє працювати з ними як зі звичайними об’єктами мови програмування Python, а також у більшості випадків не має потреби писати SQL-запити.

* **вбудований інтерфейс адміністратора**

Django автоматично створює адміністративну частину сайту, яка дозволяє виконувати основні CRUD-операції (Create, Read, Update, Delete) з сутностями бази даних.

* **диспетчер URL на основі регулярних виразів**

Django дозволяє будувати таку схему посилань, яка потрібна програмісту і не накладає ніяких додаткових обмежень, адже парсинг URL побудований на регулярних виразах.

* **система фільтрів («middleware»)**

Middleware – це спеціальні модулі, які обробляють запити до того як вони будуть оброблені View, вони використовуються для побудови додаткових обробників запитів, наприклад включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій.

* **система кешування**

Django дозволяє кешувати SQL-вибірки, окремі змінні, шаблони та іншу інформацію. Також він можу бути легко налаштований для робити з Memcached або Redis.

* **Інтернаціоналізація**

Переклад проекту побудованого на Django є досить простим, адже цей веб-фреймворк одразу надає усі необхідні інструменти для цього. Інтернаціоналізація будується на основі «лінивого» перекладу, тобто якщо якийсь текст не має перекладу, то буде використано базовий текст, а не повідомлення про помилку.

### Розробка застосування з використанням веб-фреймоворку Django

Щоб в повній мірі показати всі переваги, та використати якомога більше можливостей цього фреймворка, було вирішено зробити прототип програмного забезпечення для організації комунікації студентів з викладачем на лекції.

Демонстрацію роботи прототипу можна побачити на рис. 3.1-3.4. Лістинг коду файлу models.py можна знайти у додатку А.

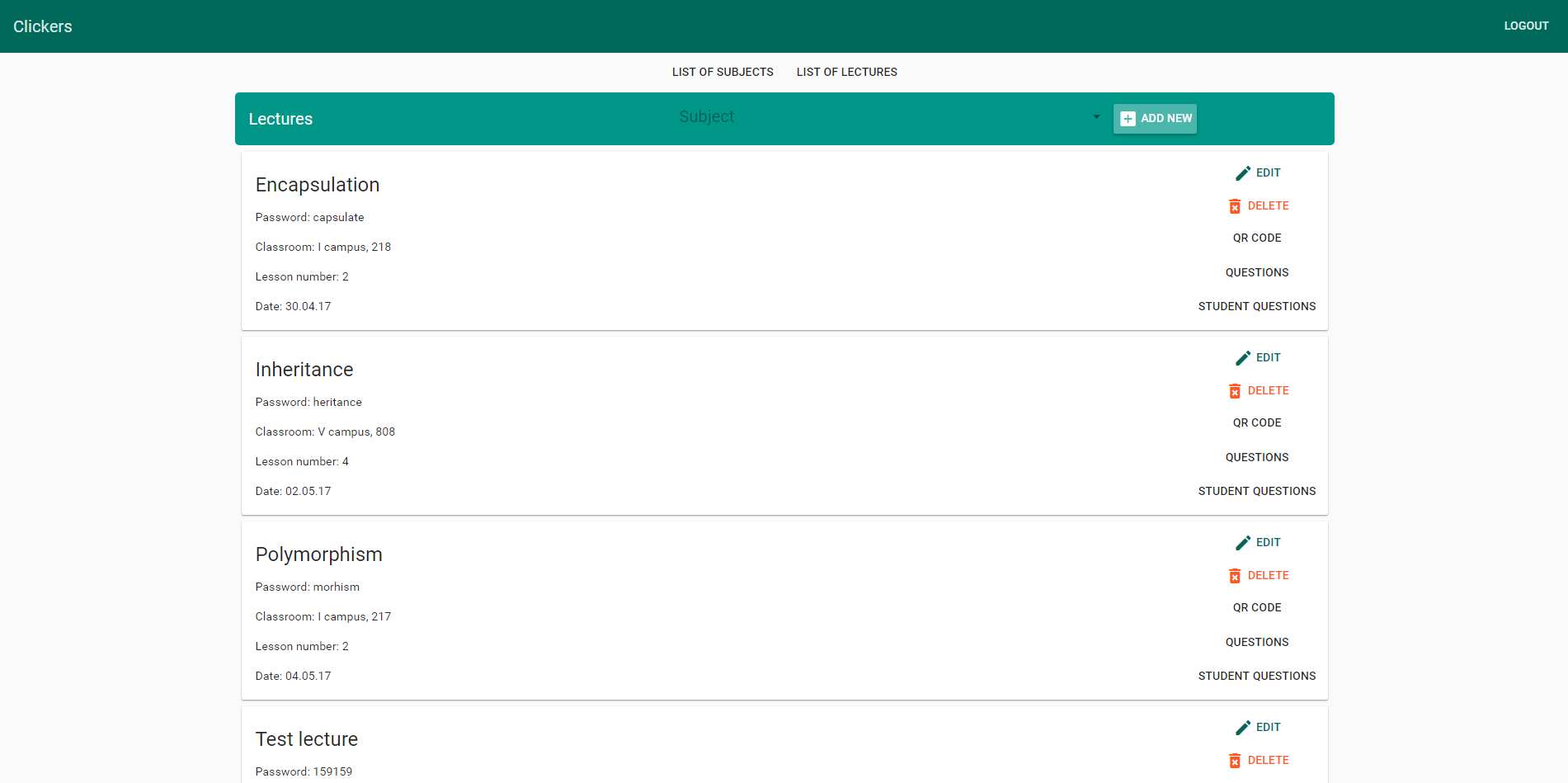


Рис. 3.1. Вигляд сторінки з списком лекцій, який отримується з сервера через REST API і вдображається на клієнті з використанням Angular

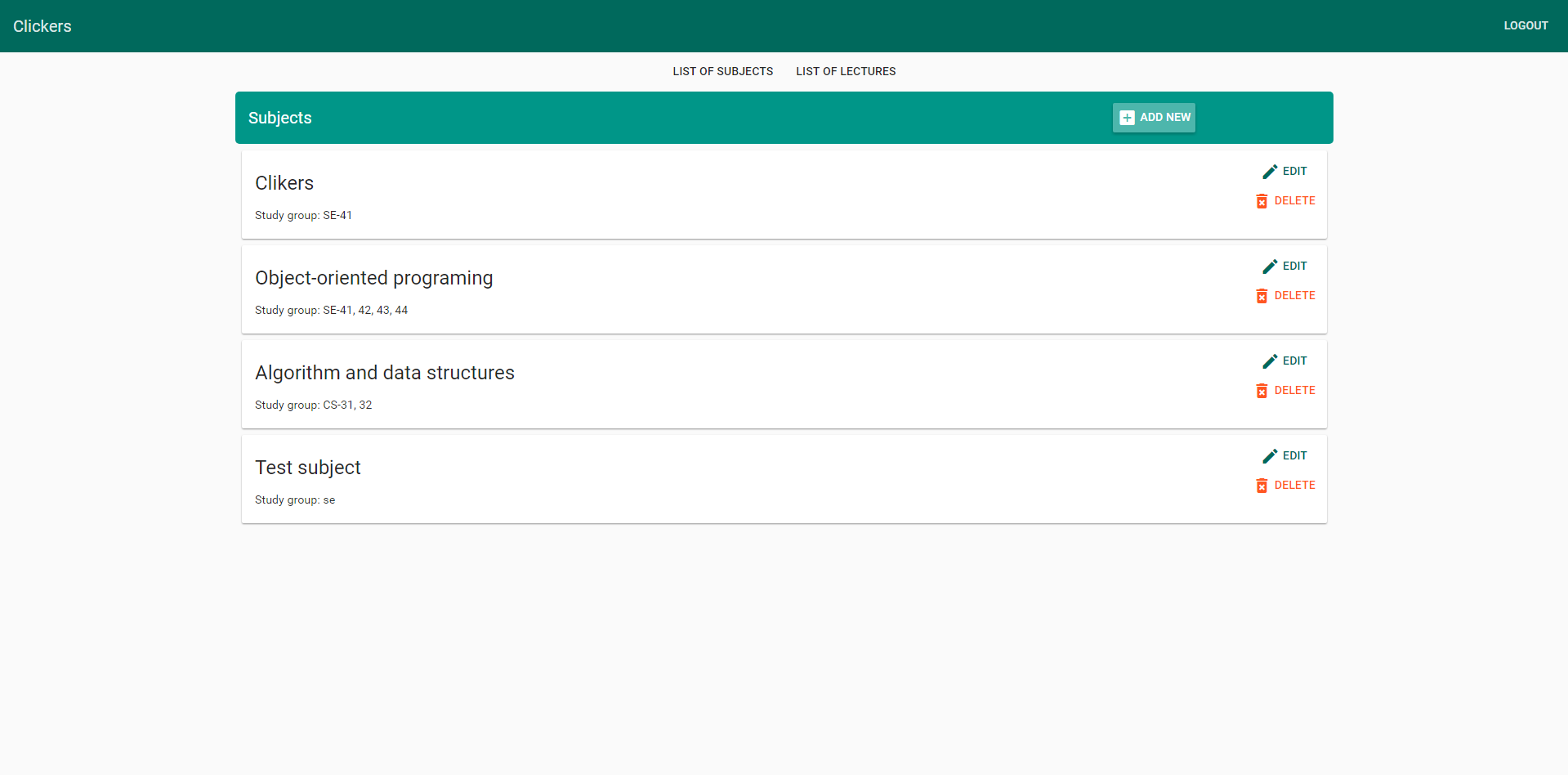


Рис. 3.2. Вигляд сторінки з списком навчальних предметів

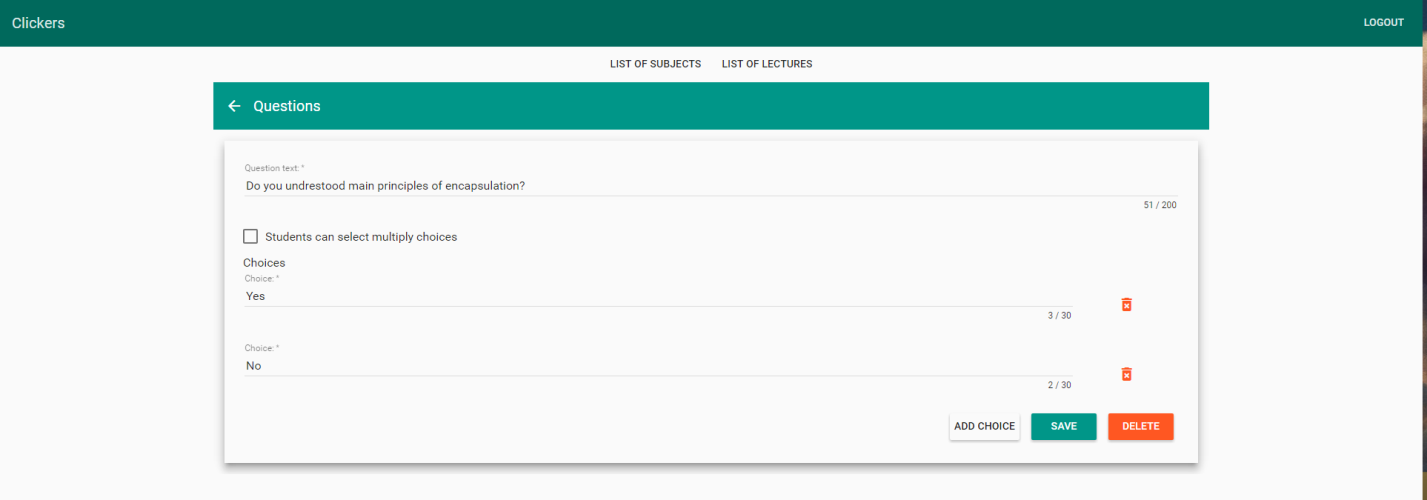


Рис. 3.3. Сторінка створення нового питання для студентів

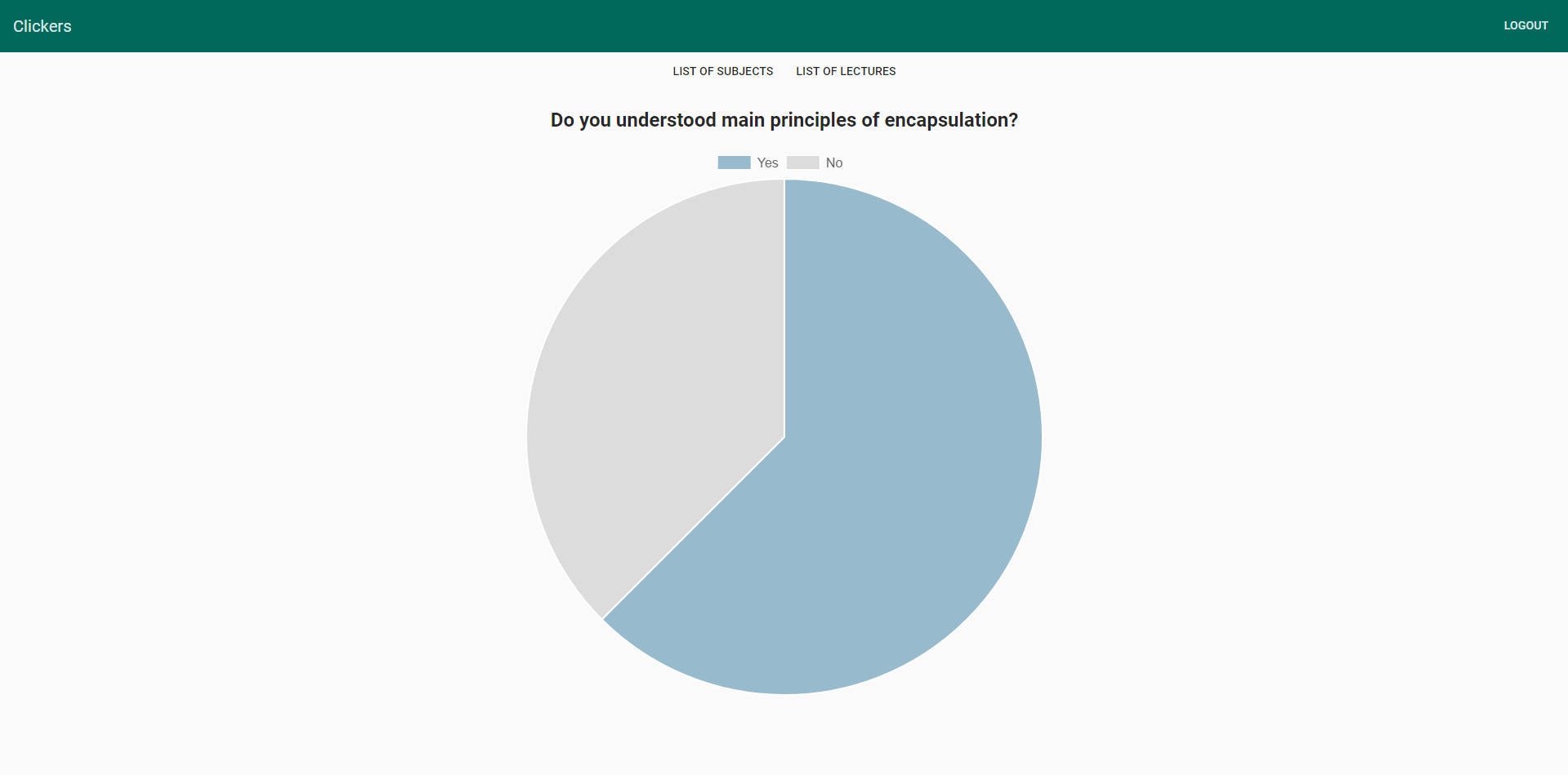


Рис. 3.4. Відображення графіку з відповідями студентів, який оновлюється у реальному часі

## Завдання отримане стосовно дипломного проекту

Стосовно дипломного проекту було винесено наступні завдання:

1. Провести аналіз програмного забезпечення для організації комунікації студентів з викладачем на лекції.
2. Скласти специфікацію вимог до ПЗ
3. Обґрунтувати вибір технологій розробки
4. Розробити прототип інтерфейсу користувача

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

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

У сфері освіти для комунікації студентів з викладачем все більш популярним стає використання різноманітних технологій реагування [3]. 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 [4].



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 [5].

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 [6].

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 [7].

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.

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

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 [1, 2], бібліотеці Django REST Framework [8], фреймоворку AngularJS, специфікація вимог.

Вихідні дані: результуюче застосування, яке реалізовує поставлену задачу.

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

1. **Вступ**
   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.

1. **Загальний опис**
   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.
  1. ***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.

* 1. ***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.

1. **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.
   1. ***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
   1. ***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.

1. **Вимоги до зовнішніх інтерфейсів**
   1. **Користувацькі інтерфейси**

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

* 1. **Апаратні інтерфейси**

Поточна веб-система не буде використовувати жодних апаратних інтерфейсів.

* 1. **Програмні інтерфейси**

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

* 1. **Комунікаційні інтерфейси**

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

1. **Інші нефункційні вимоги**
   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)

* 1. **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.

* 1. **Атрибути якості програмного продукту**
* Переносимість.
* Надійність.
* Зручність використання.
* Зручність супроводу.

1. **Інші вимоги**

* Для клієнтського веб-сайту необхідно забезпечити переклад українською та англійською мовами.

# 5. Проектування архітектури застосування

# 5.1 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].

Для кращого проектування, була створена UML - діаграма розгортування (рис. 5.1) та діаграма прецедентів (рис. 5.2).



Рис. 5.1. Діаграма розгортування.



Рис. 5.2. Діаграма прецедентів використання.

# 5.2 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.

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 (адже зараз ніяких альтернатив для неї у створенні клієнтських програм для веб-браузерів немає), з допомогою фреймворку AngularJS.

# 5.3 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. Оглянути основні зображення прототипу графічного інтерфейсу можна на рис. 5.3 – 5.6.

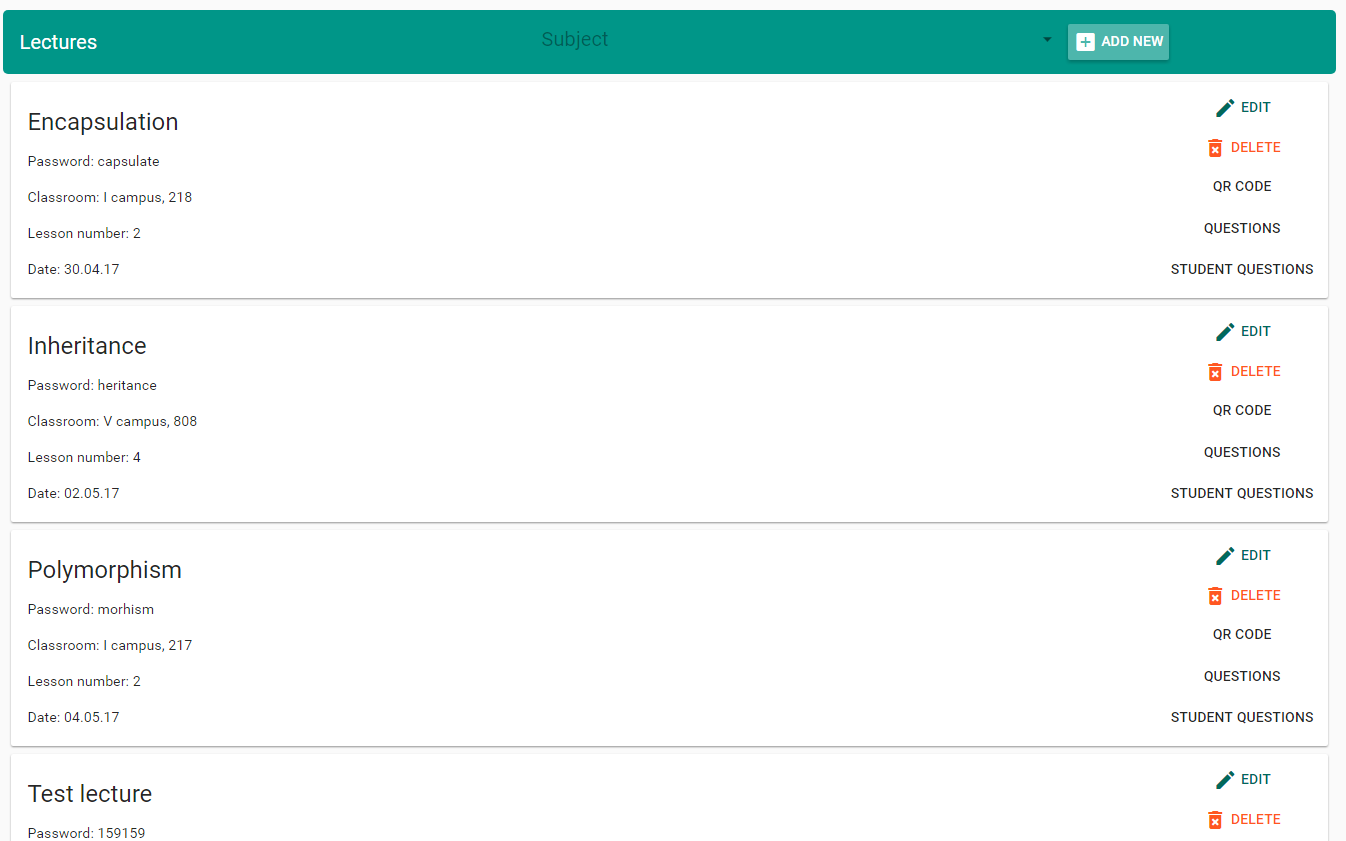


Рис. 5.3. Вигляд списку лекцій, які реалізовані у вигляді карток Material Design

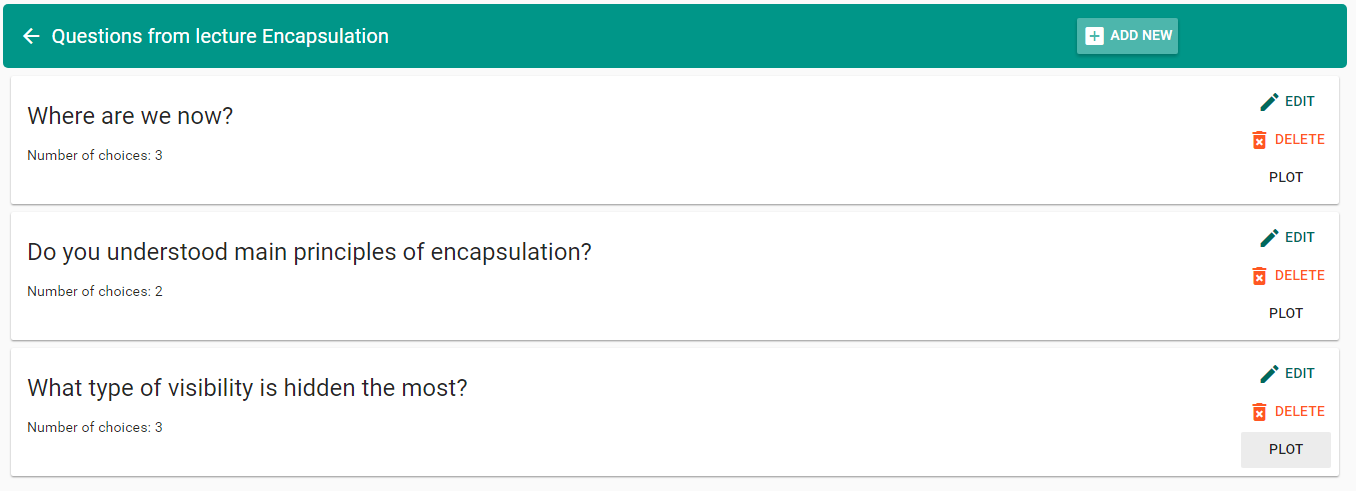


Рис. 5.4. Вигляд список питань створених викладачем до конкретної лекції

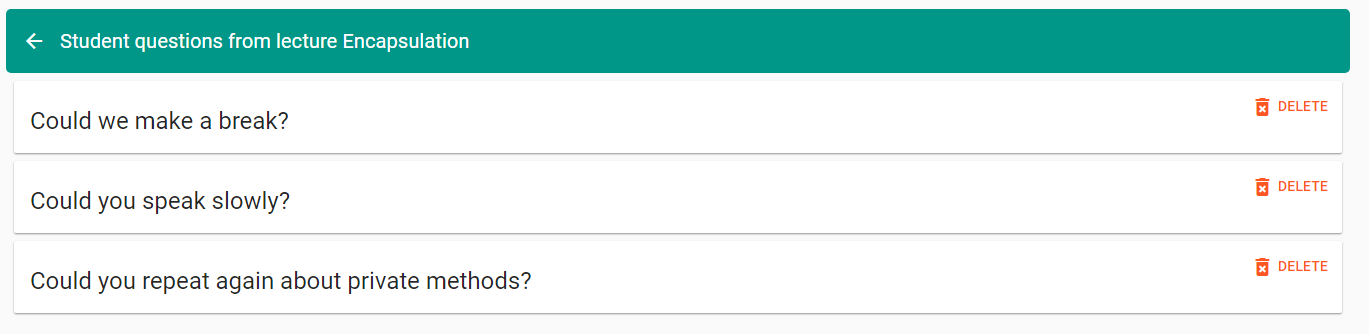


Рис. 5.5. Вигляд списку питань, які створюють студенти

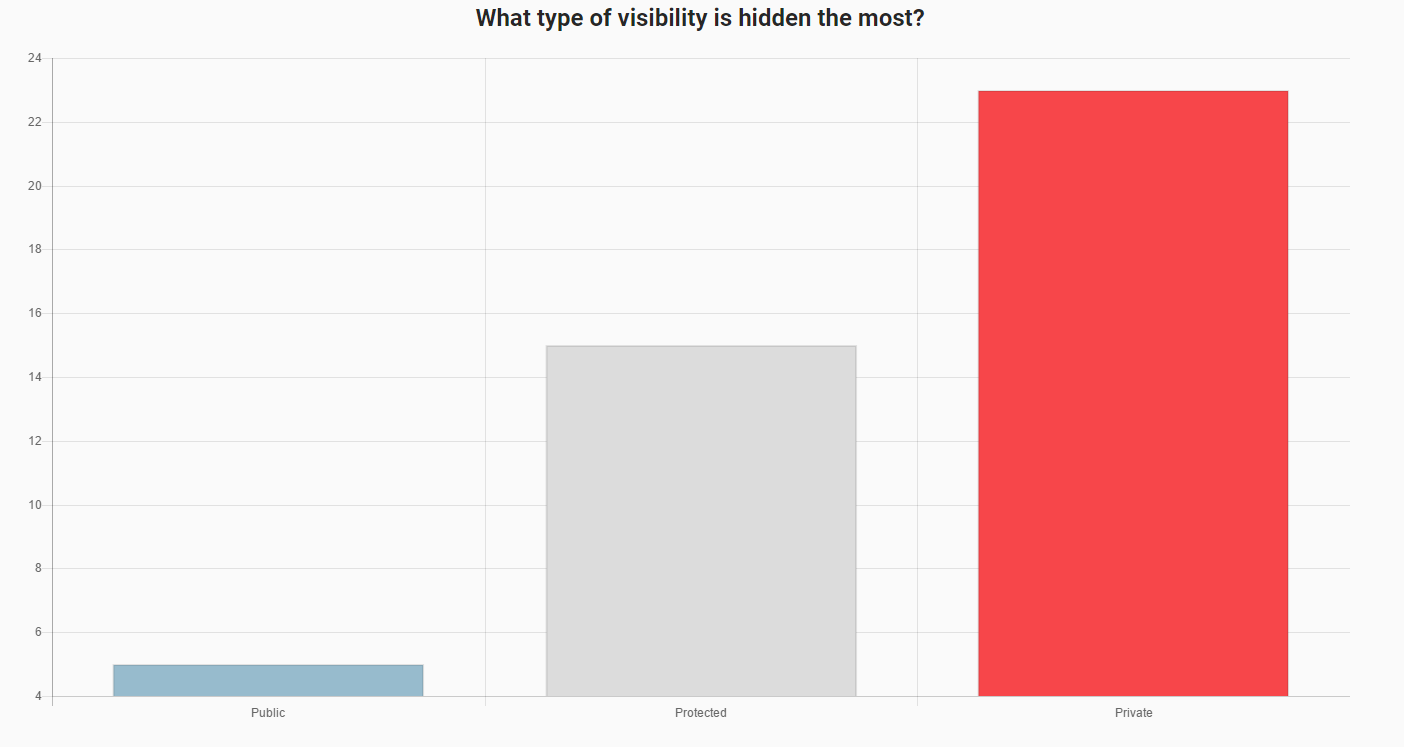


Рис. 5.6. Вигляд графіку з відповідями студентів

## 6. Висновки про отримані під час практики результати

На базі практики я отримав досвід роботи у команді, роботи у реальній компанії з розробки ПЗ. Я краще освоїв мову програмування Python, вивчив новий для себе веб-фреймворк для створення веб-систем Django. Також застосував на практиці отримані знання, розробивши прототип програмного забезпечення для організації комунікації студентів з викладачем на лекції.

Для дипломного проекту була розглянута його предметна область, а саме, IT-засоби для організації комунікації студентів з викладачем на лекції, були проаналізовані інформаційні джерела, щоб зібрати якомога більше інформації для розробки необхідного веб-сервісу. Також були обрані архітектура та технології, необхідні для розробки, складена специфікація вимог до розробки ПЗ, розроблений графічний інтерфейс.

# СПИСОК ОПРАЦЬОВАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Django documentation [Електронний ресурс]. – Веб-доступ до сторінки: https://docs.djangoproject.com/en/ (2017).
2. The Django book [Електронний ресурс]. – Веб-доступ до сторінки: http://djangobook.com/ (2017).
3. ClassroomResponse Systems ("Clickers"). [Електронний ресурс]. – Веб-доступ до сторінки: https://cft.vanderbilt.edu/guides-sub-pages/clickers/ (2017).
4. Classroom Response Systems [Електронний ресурс]. – Веб-доступ до сторінки: http://serc.carleton.edu/sp/library/classresponse/index.html (2017).
5. Clickerstechnologies [Електронний ресурс]. –Веб-доступ до сторінки: presswestern.uwo.ca/faculty.html#tech (2017).
6. Using Clickers in the Classroom Systems [Електронний ресурс]. – Веб-доступ до сторінки: https://www.youtube.com/watch?v=CnnP0uCqD4k (2017).
7. 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.
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. Лістинг файлу models.py.

Код моделей прототипу програмного забезпечення для роботи через Django ORM.

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