

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЕКТ
на тему
**ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ
“ІНТЕРНЕТ-ФОРУМУ ДЛЯ ОБГОВОРЕННЯ КОМП'ЮТЕРНИХ
ЗБІРОК”**

студента Мельника Андрія Валерійовича

Науковий керівник: к.е.н., доцент.
Лозовик Ю. М.

**Випускний бакалаврський
проект допущений до захисту в
Екзаменаційній комісії з атестації
здобувачів вищої освіти**
Завідувач кафедри: к.е.н., доцент
Тішков Б. О.

Київ 2024

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:

Керівник проектної групи(гарант)

освітньо-професійної програми

_____ Іванченко Г.Ф.

“_____” _____ 2024 р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри

_____ Тішков Б.О.

“_____” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти ***Мельника Андрія Валерійовича***
очної (денної) форми навчання

на підготовку кваліфікаційного бакалаврського проекту
на тему: «Проектування інформаційної системи інтернет-форуму для обговорення комп'ютерних збірок»

Тему затверджено наказом ректора Університету від «11» 03 2024 р. № 529-ст.

Кваліфікаційний бакалаврський проект виконується на матеріалах дослідження предметної області джерела

План кваліфікаційного бакалаврського проекту

Розділ I. Характеристика та аналіз предметної галузі

Розділ II. Розробка вимог і моделювання інформаційної системи

Розділ III. Проектування та реалізація компонентів системи

Об'єкт дослідження: процес розробки інформаційної системи інтернет-форуму для обговорення комп'ютерних збірок

Предмет дослідження: інформаційна система інтернет-форуму для обговорення комп'ютерних збірок

Мета кваліфікаційного бакалаврського проекту: полягає в розробці інформаційної системи інтернет форуму, що надасть користувачам допомогу в пошуку порівнянні та підборі комплектуючих для збірки комп'ютера, а також дасть користувачам можливість шукати відповіді на питання що виникають при збірці та експлуатації персональних комп'ютерів

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі I Надати характеристику предметної області та об'єкта дослідження. Проаналізувати інформаційні систем, що використовуються для вирішення подібних задач в обраній предметній області.

У розділі II Розробити специфікацію вимог на основі результатів характеристики предметної області та аналізу прикладів. Розробити постановку задачі. Здійснити моделювання системи.

У розділі III Навести проектні рішення з інформаційного, технічного і програмного забезпечення, а також результати реалізації інформаційної системи.

Завдання підготував
науковий керівник _____

Лозовик Юрій Миколайович

“ _____ ” _____ 2024 р.

Завдання одержав
студент _____

Мельник Андрій Валерійович

“ _____ ” _____ 2024 р.

Відгук
про кваліфікаційний бакалаврський проект
здобувача навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

Мельника Андрія Валерійовича

на тему

**«Проектування інформаційної системи інтернет-форуму для обговорення
комп'ютерних збірок»**

1. Актуальність теми:

питання вибору нового ПК є важливим для розробників і користувачів різноманітних програм. Комп'ютери еволюціонують, - змінюються комплектуючі та програмне забезпечення, що вимагає від покупців знання актуальної інформації щодо характеристик і особливостей збірок ПК. Тема роботи є актуальною для кожного, хто має бажання придбати новий або оновити існуючий ПК. Для вирішення цього питання автор роботи, Мельник Андрій Валерійович, проектує і розробляє інтернет-форум для швидкої комунікації з виробниками ПК та обговорення можливих варіантів збірок.

2. Позитивні риси кваліфікаційного бакалаврського проекту:

Автор роботи, Мельник Андрій Валерійович, здійснив проектування та розробку інтернет-форуму з використанням сучасного стеку технологій, проаналізував аналоги та визначив напрями подальшого удосконалення продукту.

3. Наявність самостійних розробок автора:

У роботі важко виокремити текст автора від запозичень, бо відсутні посилання на використані джерела. Автор реалізує велику кількість структурних і поведінкових діаграм, розробляє прототип інтерфейсу системи та реалізує проектні рішення у власному авторському продукті. Продукт – цілісний і реалізує визначний функціонал.

4. Цінність теоретичних висновків та практичних рекомендацій:

розроблений програмний продукт буде корисним для всіх, хто бажає придбати новий або оновити наявний ПК.

5. Наявність недоліків:

- задовільний виклад матеріалу;
- у списку використаних джерел відсутні наукові джерела;
- робота більш практичного спрямування, ніж наукового;

- необхідність покращення інтерфейсу та серверної частини програми.

6. Загальна оцінка кваліфікаційного бакалаврського проекту та його допущення до захисту перед ЕК:

Висвітлене вище дозволяє констатувати, що дипломний проєкт відповідає встановленим вимогам, що висуваються до таких видів робіт, а його автор, Мельник Андрій Валерійович, володіє навиками проєктування та розробки інтернет-форуму для обговорення збірок ПК. Дипломна робота рекомендується до захисту в ДЕК з позитивною оцінкою.

Науковий керівник

Лозовик Юрій Миколайович

Рецензія
на кваліфікаційний бакалаврський проект
Мельника Андрія Валерійовича

тема
**«Проектування інформаційної системи інтернет-форуму для
обговорення комп'ютерних збірок»**

Актуальність теми кваліфікаційного бакалаврського проєкту і доцільність його розроблення. Тему кваліфікаційного бакалаврського проєкту можна вважати актуальною, оскільки на сьогоднішній день у більшості користувачів є персональні комп'ютери і значну їх частину складають саме стаціонарні комп'ютери.

Якість проведеного дослідження. Дослідження було проведено якісно, демонструє глибоке розуміння предметної області та методології проектування інформаційних систем.

Позитивні риси кваліфікаційного бакалаврського проєкту. Проєкт відповідає сучасним вимогам та тенденціям розвитку інформаційних технологій. В проєкті використовуються сучасні технології та інструменти проектування.

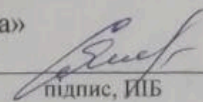
Зауваження. Хоча в цілому проєкт виконаний добре, для подальшого розвитку варто перейти від використання шаблонів до повноцінного фронтенду. Також необхідно реалізувати останній розділ форуму. Також можна виділити відсутність детального опису тестування.

Практична значимість висновків і рекомендацій. Висновки та рекомендації що представлені в проєкті є чіткими та в повному об'ємі описують недоліки і переваги розробленої системи, а також можливості для її подальшого розвитку і вдосконалення. Запропоновані рішення щодо структури форуму, функціональних можливостей, інтерфейсу користувача та технічної реалізації можуть бути легко адаптовані до потреб конкретного користувача або організації.

Вважаю, що робота виконана на високому професійному рівні, має теоретичну та практичну значущість й заслуговує на оцінку «Відмінно»

ВСП «Фаховий коледж інформаційних
систем і технологій КНЕУ ім. В. Гетьмана»
завідуюча відділення «Програмування»




підпис, ІПБ

Світлана ЯНЧУК

Місце печатки організації, де працює рецензент

Анотація

дипломного проекту студента 4 курсу
навчально-наукового інституту «Інститут інформаційних технологій в
економіці»

Мельника Андрія Валерійовича, виконаної на тему:
«Розробка інформаційної системи інтернет-форуму для обговорення
комп'ютерних збірок»

Київ: кафедра інформаційних систем в економіці, 2024 р.

Даний дипломний проект присвячено темі проектування та розробки інформаційної системи інтернет форуму для обговорення комп'ютерних збірок. В даному дипломному проекті описуються основні проектні рішення, наводяться приклади скріншотів клікабельного прототипу інтерфейсу користувача, приклади скріншотів реалізованої системи.

Кваліфікаційний бакалаврський проект складається з трьох розділів, логічно пов'язаних між собою.

В першому розділі наводиться опис предметної галузі.

Другий розділ присвячений проектуванню інформаційної системи, в даному розділі наводяться функціональні, нефункціональні та бізнес вимоги, а також в даному розділі розглядається постановка задачі та алгоритм функціонування системи.

В третьому розділі розглядається інформаційне, технічне та програмне забезпечення, також в даному розділі наводяться результати реалізації інформаційної системи.

У висновках дається оцінка проекту, наводяться рекомендації щодо подальшого розвитку проекту.

РЕФЕРАТ

Даний проект містить 87 сторінок, 89 рисунків, список літератури з 11 найменувань, 16 додатків.

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-ФОРУМУ ДЛЯ ОБГОВОРЕННЯ КОМП'ЮТЕРНИХ ЗБІРОК

Перелік ключових слів: Django, Python, MVC, PostgreSQL, інтернет-форум, комп'ютерні комплектуючі, комп'ютерна збірка.

Предметом дослідження виступає інформаційна система інтернет-форуму для обговорення комп'ютерних збірок.

Об'єктом дослідження виступає процес розробки інформаційної системи інтернет-форуму для обговорення комп'ютерних збірок.

Мета даного дипломного проекту полягає в розробці інформаційної системи інтернет форуму, що надасть користувачам допомогу в пошуку порівнянні та підборі комплектуючих для збірки комп'ютера, а також дасть користувачам можливість шукати відповіді на питання що виникають при збірці та експлуатації персональних комп'ютерів.

Завданням дипломного проекту є проектування та розробка інформаційної системи на базі клієнт-серверної архітектури.

Апаратура використана при проектуванні та розробці: персональний комп'ютер на базі процесора AMD Ryzen 5 5600g.

Результати, досягнуті під час проектування: було описано основні проектні рішення, розроблено прототип системи інтернет форуму.

Одержані результати можуть бути використані: для подальшого розвитку та покращення інтернет форуму для обговорення комп'ютерних збірок.

Рік виконання дипломного проекту: 2024

Рік захисту дипломного проекту: 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ	3
ВСТУП	4
Розділ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	5
1.1. Характеристика предметної області та об'єкта дослідження	5
1.2. Дослідження існуючих інформаційних систем для обраної предметної області.	8
Розділ 2. РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ	13
2.1. Аналіз і специфікація вимог до інформаційної системи	13
2.1.1. Бізнес-вимоги до системи	13
2.1.2. Функціональні вимоги до системи	14
2.1.3. Нефункціональні вимоги до системи	16
2.2. Постановка та алгоритм розв'язання задачі	18
2.2.1. Характеристика задачі	18
2.2.2. Алгоритм розв'язання задачі	23
2.3. Моделювання інформаційної системи	25
2.3.1. Моделювання поведінки системи	25
2.3.2. Моделювання структури системи	28
2.3.3. Розробка тест-кейсів	31
2.3.4. Трасування вимог до системи	39
2.4. Моделювання користувацького інтерфейсу	41
Розділ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ	58
3.1. Інформаційне забезпечення	58
3.2. Технічне забезпечення	68
3.3. Програмне забезпечення	70
3.4. Результати реалізації інформаційної системи	75
ВИСНОВКИ	84
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	86
ДОДАТКИ	87

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

AJAX	Asynchronous JavaScript and XML (Асинхронний JavaScript і XML)
CSS	Cascading Style Sheets (Каскадні таблиці стилів)
HTML	HyperText Markup Language (Мова гіпертекстової розмітки)
MVC	Model-View-Controller (Модель-представлення-контролер)
ORM	Object-relational mapping, Об'єктно-реляційна проекція
SQL	Structured Query Language (Мова структурованих запитів)
UML	Unified Modeling Language (Уніфікована мова моделювання)
БД	База даних
СКБД	Система Керування Базами Даних

ВСТУП

Актуальність: в епоху цифрових технологій персональні комп'ютери стали невід'ємною частиною нашого життя. Серед них значну частку складають стаціонарні ПК, що вирізняються гнучкістю конфігурації та потужністю. Проте, з розвитком програмного забезпечення та зростанням його системних вимог, виникає потреба в оновленні комп'ютерних компонентів. Це ставить користувачів перед складним завданням: самостійно підібрати сумісні та оптимальні за характеристиками комплектуючі. Складність вибору полягає в різноманітті комплектуючих, ринок пропонує широкий спектр компонентів від різних виробників, що відрізняються не лише ціною, але й характеристиками, сумісністю та продуктивністю. При виборі нових комплектуючих не слід забувати про сумісність, далеко не всі комплектуючі поєднуються між собою. Важливо правильно підібрати материнську плату, процесор, оперативну пам'ять, відеокарту, блок живлення та інші компоненти, щоб вони не лише фізично суміщались, але й ефективно взаємодіяли один з одним. Не всі користувачі володіють достатнім рівнем технічних знань, щоб самостійно розібратися в тонкощах вибору комплектуючих. Це може призвести до помилок при купівлі, несумісності компонентів або невиправданих витрат.

Мета даного дипломного проекту полягає в розробці інформаційної системи інтернет форуму, що надасть користувачам допомогу в пошуку порівнянні та підборі комплектуючих для збірки комп'ютера, а також дасть користувачам можливість шукати відповіді на питання що виникають при збірці та експлуатації персональних комп'ютерів.

Розділ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної області та об'єкта дослідження

Сьогодні стаціонарні ПК залишаються популярним вибором для багатьох користувачів, пропонуючи більшу потужність за меншу ціну порівняно з ноутбуками. Проте, з розвитком програмного забезпечення та зростанням вимог до ресурсів, виникає потреба в оновленні комп'ютера або його окремих компонентів. Однак, оновлення стаціонарного ПК може здатися складним завданням, адже воно потребує ретельного вибору та порівняння різних комплектуючих. Необхідно не лише знайти компоненти, які відповідають потребам та бюджету, але й переконатися, що вони всі сумісні один з одним. Цей процес може бути складним для не досвідчених користувачів що не збирали комп'ютер раніше, адже на ринку представлено безліч різних моделей комплектуючих від різних виробників.

Важливо також знати, що не всі комп'ютерні компоненти сумісні один з одним. Так до прикладу, материнська плата повинна мати відповідний сумісний сокет для процесора, а оперативна пам'ять повинна бути сумісна з материнською платою та процесором. Окрім вище згаданих нюансів існують і безліч інших значно менш очевидних тонкощів при підборі комп'ютерних комплектуючих.

Інформаційна система інтернет-форуму для обговорення комп'ютерних збірок це інтерактивний веб-сайт що допоможе користувачам в пошуку порівнянні та підборі комплектуючих для збірки комп'ютера, пошуку інформації про комп'ютерні комплектуючі, а також для пошуку інформації про можливі рішення проблем що виникають при збірці та налаштуванні нового комп'ютера.

Серед переваг що отримують користувачі використовуючи дану інформаційну систему інтернет-форуму для обговорення комп'ютерних збірок можна виділити можливість простого пошуку комп'ютерних комплектуючих з подальшою можливістю порівняння їх характеристик, а

також з можливістю залишити власний відгук та оцінку про певну комп'ютеру деталь, переглянути відгуки та оцінки інших користувачів.

Основною перевагою даного інтернет-форуму для користувачів буде можливість користуватися конфігуратором для того щоб створити власний варіант комп'ютерної збірки, при цьому конфігуратор може надавати користувачу певну допомогу, а саме пропонувати лише сумісні комплектуючі що найбільше підходять до попередньо обраних комплектуючих, підходять за вибраними параметрами, мають найкращі оцінки від інших користувачів. Таким чином для даний інтернет-форум з цим конфігуратором комп'ютерних збірок буде корисним для користувачів що не мають досвіду в підборі комп'ютерних комплектуючих. Проте для досвідчених користувачів дана функція форуму також буде корисна адже створеними варіантами комп'ютерних збірок можна буде ділитися з іншими користувачами та обговорювати їх.

Також при збірці і налаштуванні нового комп'ютера у багатьох користувачів можуть виникати труднощі і запитання відповіді на які інколи не просто знайти. Тому на даному інтернет-форумі буде передбачений розділ з запитаннями користувачів де користувачі будуть мати можливість створювати нові питання, та відповідати на питання інших користувачів.

Метою створення даної інформаційної системи інтернет-форуму для обговорення комп'ютерних збірок буде допомога користувачам в створенні конфігурацій комп'ютерних збірок, допомога з пошуком та порівнянням комп'ютерних комплектуючих, а також допомога з пошуком відповідей на запитання та вирішення можливих проблем що виникають при збірці нового комп'ютера або встановленні нових комплектуючих.

Для власника даної інформаційної системи користь буде полягати в залученні зацікавленої аудиторії, зборі інформації про те якими саме комп'ютерними комплектуючими найчастіше цікавляться користувачі, а також можливість в подальшому розміщувати рекламні оголошення на форумі для отримання прибутку.

Майбутніми користувачами інформаційної системи інтернет-форуму для обговорення комп'ютерних збірок будуть люди що хочуть зібрати новий комп'ютер, або оновити старий, а також люди що просто цікавляться новинками комп'ютерної техніки та комп'ютерними збірками.

1.2. Дослідження існуючих інформаційних систем для обраної предметної області.

Для порівняльної характеристики було знайдено 3 інтернет-форуми на споріднену тематику а також в якості четвертого учасника порівняння буде виступати інтернет магазин: tomshardware [1], linus tech tips [2], techpowerup [3], telemart [4].

Tomshardware це досить простий інтернет-форум на якому всі обговорення розбиті на категорії серед яких можна здійснювати пошук, також є можливість встановити базові фільтри наприклад по даті, а також сортування. Приклад сторінки форуму представлено на рисунку 1.2.1.

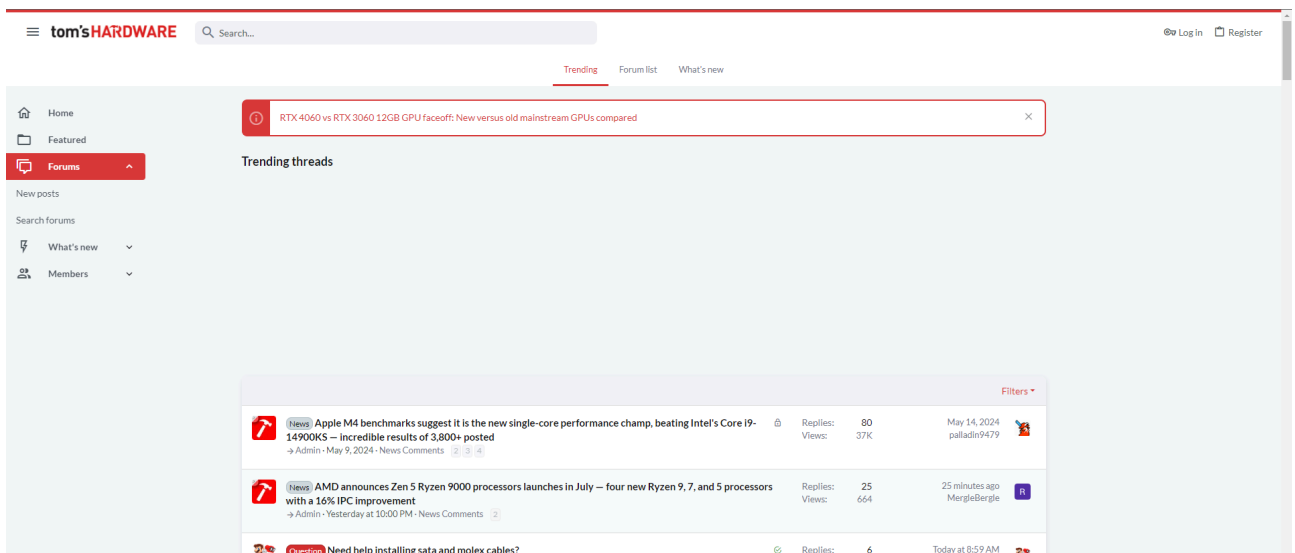


Рисунок 1.2.1 — Приклад сторінки форуму

Проте крім форуму є також сайт де можна знайти багато корисної інформації, оглядів комп'ютерних комплектуючих, типів процесорів і відеокарт та багато чого іншого. Серед мінусів можна виділити не найзручніший інтерфейс користувача.

Linustechtips подібно до попереднього має розбиття всіх обговорень на категорії, також можна використовувати просте сортування наприклад за датою публікації чи останнього оновлення, проте крім просто обговорень на цьому форумі також присутні блоги на різну тематику. Приклад головної сторінки форуму представлено на рисунку 1.2.2.

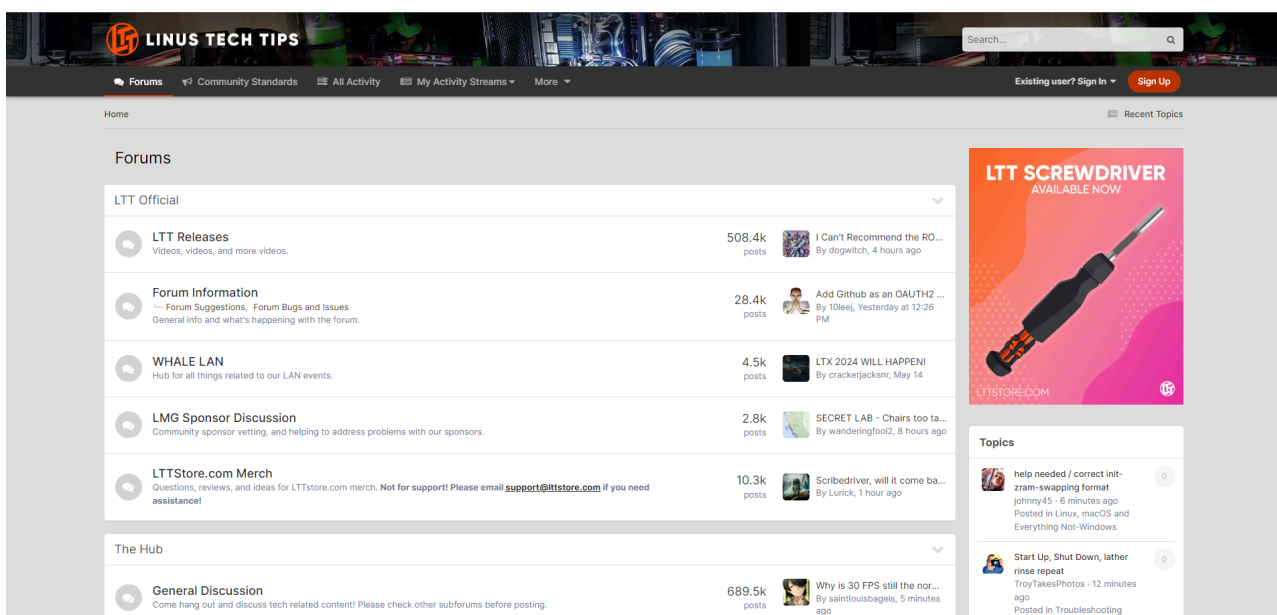


Рисунок 1.2.2 — Головна сторінка форуму linus tech tips

Серед плюсів порівняно з першим учасником порівняння даний форум має простий та зрозумілий інтерфейс.

Techpowerup як і попередні учасники порівняння має обговорення що розділені за категоріями, але на відміну від попередніх учасників даний форум також має багато додаткових матеріалів, таких як база даних комп'ютерних комплектуючих де можна знайти детальні характеристики комп'ютерних комплектуючих та їх опис. Також на даному форумі є розділ з оглядами комп'ютерних комплектуючих, а також розділ з програмним забезпеченням що може бути корисним при тестуванні нового ПК. Приклад сторінки форуму tech power up представлений на рисунку 1.2.3.

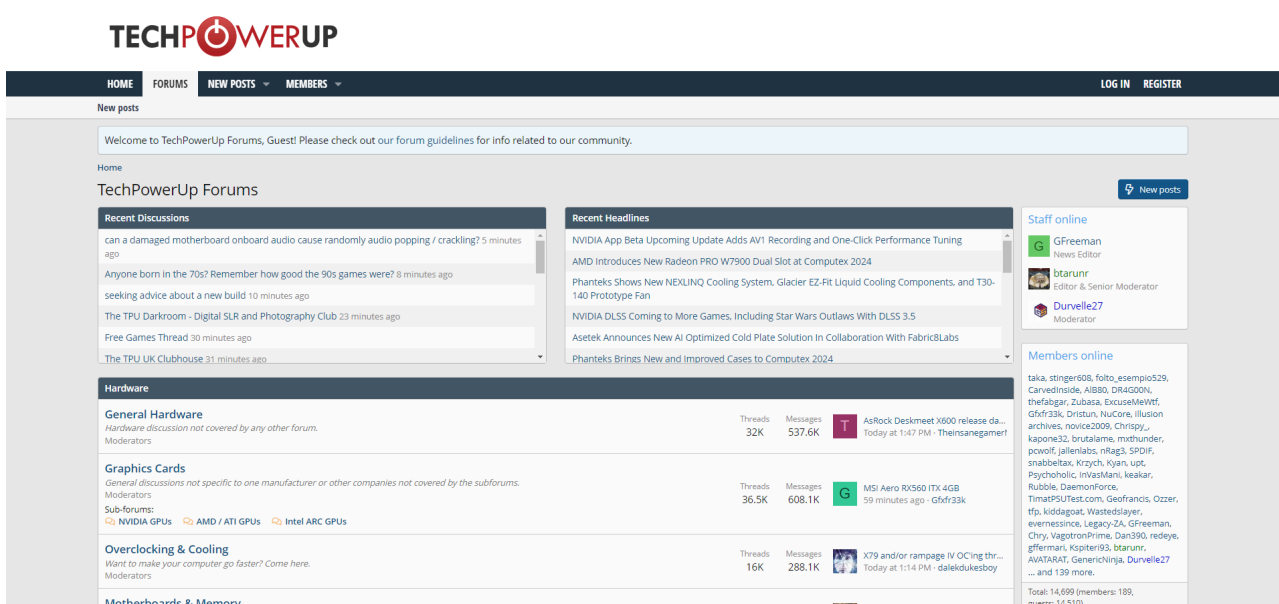


Рисунок 1.2.3 — Сторінка форуму tech power up

В підсумку даний форум має зручний інтерфейс та велику кількість різноманітної корисної інформації.

Telemart це інтернет магазин де можна знайти різноманітні комплектуючі для комп'ютера готові комп'ютери та ноутбуки, комп'ютерну периферію, а також деяку іншу техніку. Приклад головної сторінки інтернет магазину представлено на рисунку 1.2.4.

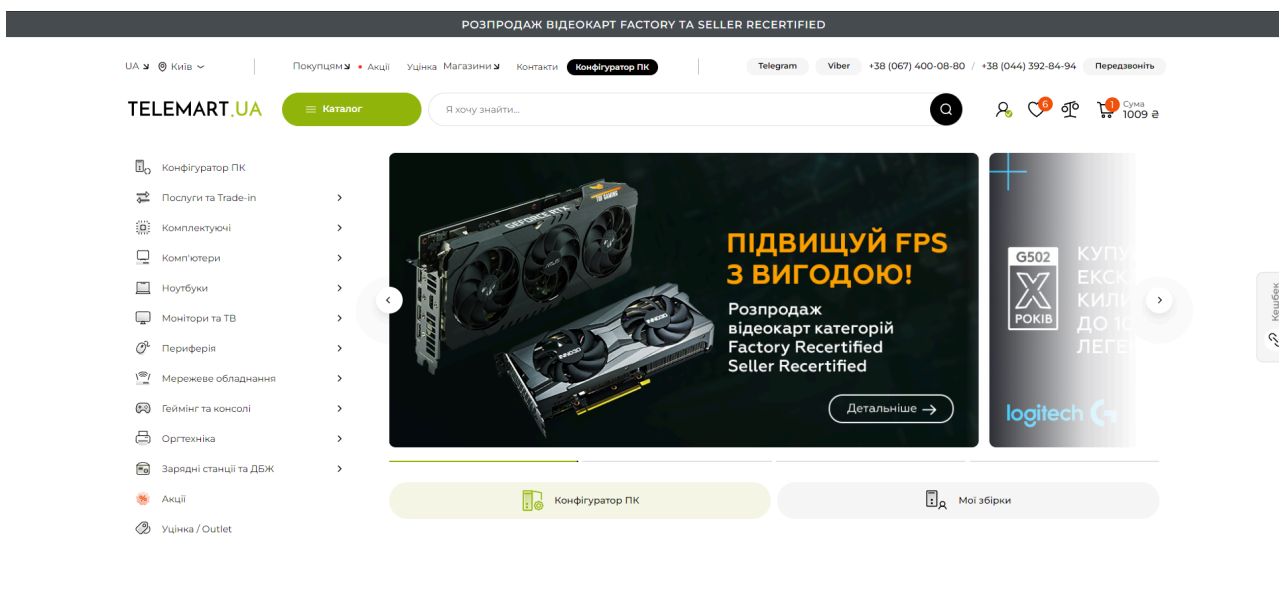


Рисунок 1.2.4 — Головна сторінка сайту telemart

До кожного товару тут можна задавати запитання на які можуть відповідати як адміністрація так і звичайні користувачі, також на даному сайті є розділ з блогами. Головною особливістю даного сайту є конфігуратор

ПК, за допомогою якого можна створити власний варіант комп'ютерної збірки, при цьому що важливо в конфігураторі є перевірка на сумісність комплектуючих, тобто якщо обрати несумісні комплектуючі то буде попередження про це, що є дуже корисною функцією. Приклад сторінки з конфігуратором представлено на рисунку 1.2.5.

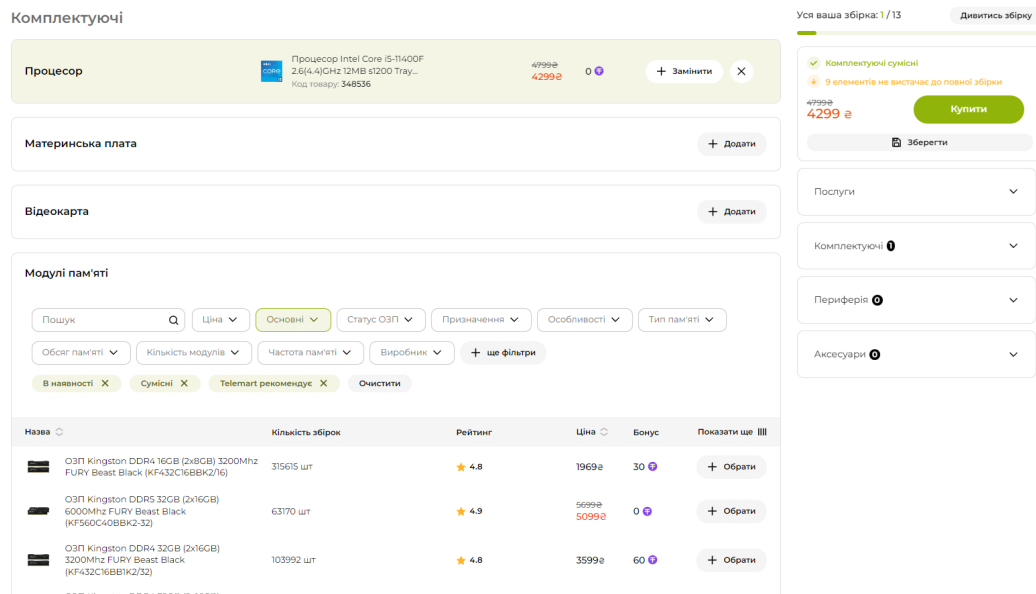


Рисунок 1.2.5 — Приклад сторінки конфігуратора на сайті telemart

Проте можливості вільно обговорювати комплектуючі чи комп'ютерні збірки на сайті нема. До плюсів також можна віднести зручний та зрозумілий користувацький інтерфейс, а також підтримку сайтом української мови.

Таблиця 1.1 — Порівняння інформаційних систем

№	Можливості	tomshardware	linustechtips	techpowerup	telemart
1	Стрічка новин	так	ні	так	ні
2	Блог	ні	так	так	так
3	Огляди на комплектуючі	так	ні	так	ні
4	Можливість задати питання	так	так	так	так
5	Можливість обговорення комплектуючих	так	так	так	так
6	Можливість пошуку інформації про специфікації комп'ютерних комплектуючих	ні	ні	так	так
7	Можливість створити власний варіант комп'ютерної збірки	ні	ні	ні	так
8	Можливість обговорення комп'ютерних збірок	так	так	так	ні
9	Інтуїтивна зрозумілість інтерфейсу	ні	так	так	так

Отже підсумовуючи можна виділити лідерів порівняння а саме techpowerup та telemart. Серед головних особливостей інформаційних систем які слід виділити для розгляду та відтворення у власній інформаційній системі: конфігуратор пк за допомогою якого можна створити власний варіант комп'ютерної збірки а також додаткові матеріали з інформацією про комп'ютерні комплектуючі та їх опис, можливість вільно обговорювати комп'ютерні збірки та комплектуючі.

Розділ 2. РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної системи

2.1.1. Бізнес-вимоги до системи

В цьому розділі представлені бізнес-вимоги до інтернет-форуму. Першою з цих вимог є орієнтованість на створення активної спільноти. Ця бізнес вимога є головною, адже для будьякого форуму спільнота є найважливішим аспектом успіху. Наступними йдуть бізнес-вимоги стосовно монетизації інтернет-форуму серед яких: рекламні можливості для компаній, які продають комп'ютерне обладнання або послуги, можливість партнерства з виробниками комплектуючих для просування їхніх продуктів серед спільноти. Діаграма бізнес-вимог представлена на рисунку 2.1.1.1, а на рисунку 2.1.1.2 представлені специфікації бізнес-вимог.

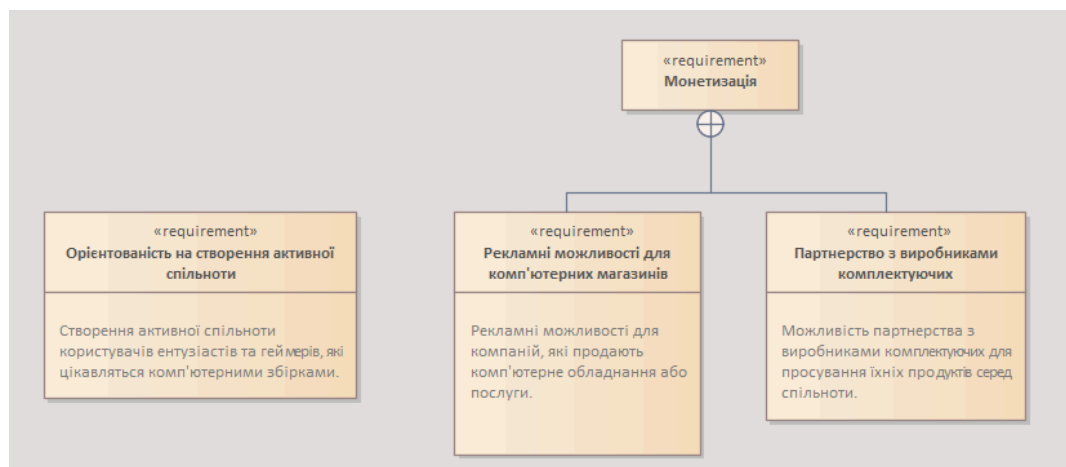


Рисунок 2.1.1.1 — Бізнес-вимоги

Item	Stereotype	Status	Difficulty	Priority
☒ Монетизація	requirement	Proposed	High	Medium
☒ Партнерство з виробниками комплектуючих Можливість партнерства з виробниками комплектуючих для просування їхніх продуктів серед спільноти.	requirement	Proposed	High	Medium
☒ Рекламні можливості для комп'ютерних магазинів Рекламні можливості для компаній, які продають комп'ютерне обладнання або послуги.	requirement	Proposed	Medium	High
☒ Орієнтованість на створення активної спільноти Створення активної спільноти користувачів ентузіастів та геймерів, які цікавляться комп'ютерними збірками.	requirement	Proposed	Medium	High

Рисунок 2.1.1.2 — Специфікація бізнес-вимог

2.1.2. Функціональні вимоги до системи

В цьому розділі представлені функціональні вимоги до системи інтернет-форуму для обговорення комп'ютерних збірок що розробляється. Серед функціональних вимог можна виділити такі групи як: коментарі користувачів де наведені вимоги до системи коментарів, система аутентифікації де наведено вимоги до системи реєстрації та входу в аккаунт, вимоги що до сторінки з запитаннями користувачів, вимоги до адмінпанелі, вимоги до навігації по форуму, а також вимоги до сторінки з варіантами комп'ютерних збірок користувачів. Розроблена діаграма функціональних вимог представлена на рисунку 2.1.2.1.

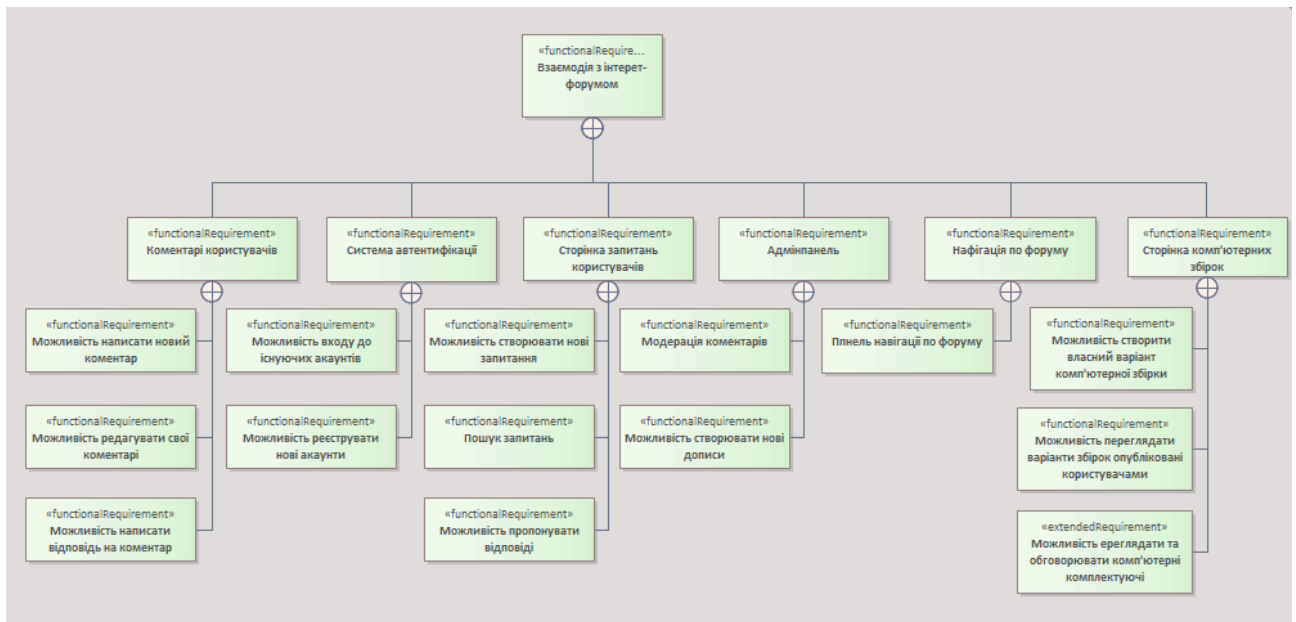


Рисунок 2.1.2.1 — Діаграма функціональних вимог

Також на рисунку 2.1.2.2 представлено специфікацію функціональних вимог створену за допомогою менеджера специфікацій. В специфікаціях вимог серед іншого зазначено: тип вимоги, статус вимоги, її складність та пріоритетність.

☒ Адмінпанель	functionalRequire...	Proposed	Medium	Medium
☒ Модерація коментарів	functionalRequire...	Proposed	High	Medium
☒ Можливість створювати нові дописи	functionalRequire...	Proposed	Low	High
☒ Коментарі користувачів	functionalRequire...	Proposed	High	High
☒ Можливість написати відповідь на коментар	functionalRequire...	Proposed	High	High
☒ Можливість написати новий коментар	functionalRequire...	Proposed	High	High
☒ Можливість редагувати свої коментарі	functionalRequire...	Proposed	Medium	Medium
☒ Навігація по форуму	functionalRequire...	Proposed	Low	High
☒ Панель навігації по форуму	functionalRequire...	Proposed	Low	High
☒ Система автентифікації	functionalRequire...	Proposed	Low	High
☒ Можливість входу до існуючих акаунтів	functionalRequire...	Proposed	Low	High
☒ Можливість реєструвати нові акаунти	functionalRequire...	Proposed	Low	High
☒ Сторінка запитань користувачів	functionalRequire...	Proposed	Low	Medium
☒ Можливість пропонувати відповіді	functionalRequire...	Proposed	Low	Medium
☒ Можливість створювати нові запитання	functionalRequire...	Proposed	Low	Medium

Рисунок 2.1.2.2 — Специфікація функціональних вимог

2.1.3. Нефункціональні вимоги до системи

В розділі 2.1.3 представлені нефункціональні вимоги до системи інтернет-форуму. Нефункціональні вимоги розділені на такі категорії як безпека, документація та доступність. Сама розроблена діаграма нефункціональних вимог зображена на рисунку 2.1.3.1.

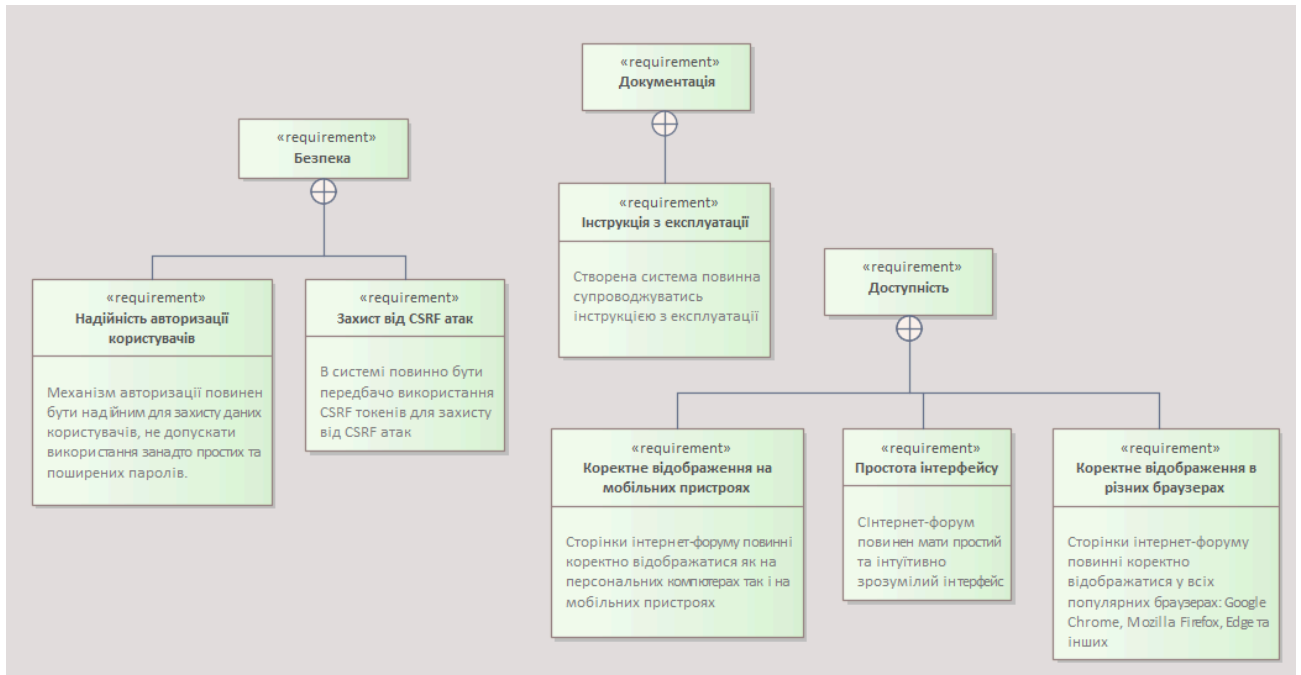


Рисунок 2.1.3.1 — Діаграма нефункціональних вимог

До нефункціональних вимог щодо безпеки відносяться надійність авторизації користувачів, та захист від CSRF атак. Під вимогою про надійність авторизації користувачів мається на увазі що механізм авторизації повинен бути надійним для захисту даних користувачів, не допускати використання занадто простих та поширених паролів. Під вимогою про захист від CSRF атак мається на увазі необхідність використання CSRF токенів для форм що будуть надсилатися користувачами.

До нефункціональних вимог про документацію є вимога про необхідність супроводження системи інструкцією з експлуатації.

До нефункціональних вимог про доступність є вимоги про коректне відображення на мобільних пристроях, простоту інтерфейсу, коректне відображення в різних браузерах. Під вимогою про коректне відображення на мобільних пристроях мається на увазі що сторінки інтернет-форуму повинні

коректно відображатися як на персональних комп'ютерах так і на мобільних пристроях. Під вимогою про простоту інтерфейсу мається на увазі що інтернет-форум повинен мати простий та інтуїтивно зрозумілий інтерфейс. Під вимогою про коректне відображення в різних браузерях мається на увазі що сторінки інтернет-форуму повинні коректно відображатися у всіх популярних браузерях: Google Chrome, Mozilla Firefox, Edge та інших.

Також на рисунку 2.1.3.2 представлено специфікацію до нефункціональних вимог.

Item	Stereotype	Status	Difficulty	Priority
☒ Документація	requirement	Proposed	High	Low
☒ Інструкція з експлуатації Створена система повинна супроводжуватись інструкцією з експлуатації	requirement	Proposed	High	Low
☒ Доступність	requirement	Proposed	Medium	High
☒ Коректне відображення в різних браузерях Сторінки інтернет-форуму повинні коректно відображатися у всіх популярних браузерях: Google Chrome, Mozilla Firefox, Edge та інших	requirement	Proposed	Low	High
☒ Коректне відображення на мобільних пристроях Сторінки інтернет-форуму повинні коректно відображатися як на персональних комп'ютерах так і на мобільних пристроях	requirement	Proposed	High	Medium
☒ Простота інтерфейсу Інтернет-форум повинен мати простий та інтуїтивно зрозумілий інтерфейс	requirement	Proposed	Medium	High
☒ Безпека	requirement	Proposed	Low	High
☒ Надійність авторизації користувачів Механізм авторизації повинен бути надійним для захисту даних користувачів, не допускати використання занадто простих та поширених паролів.	requirement	Proposed	Low	High
☒ Захист від CSRF атак В системі повинно бути передбачено використання CSRF	requirement	Proposed	Low	Medium

Рисунок 2.1.3.2 — Специфікація нефункціональних вимог

2.2 Постановка та алгоритм розв'язання задачі

2.2.1 Характеристика задачі

Задача з полегшення пошуку та обговорення комп'ютерних збірок, призначена для полегшення пошуку користувачами інформації про комп'ютерні збірки та для надання можливості їх зручного обговорення.

При розв'язанні задачі здійснюється керування наступними об'єктами: інформація про користувачів, інформація про комп'ютерні комплектуючі, інформація про комп'ютерні збірки, коментарі користувачів.

Вихідна інформація використовується користувачами інтернет-форуму, вхідну інформацію про комп'ютерні комплектуючі вносить адміністратор, а інформацію про коментарі та обговорення створюють користувачі в процесі функціонування системи.

Взаємозв'язок з іншими задачами можна прослідкувати на інформаційній моделі задачі (рисунк 2.2.1.1).

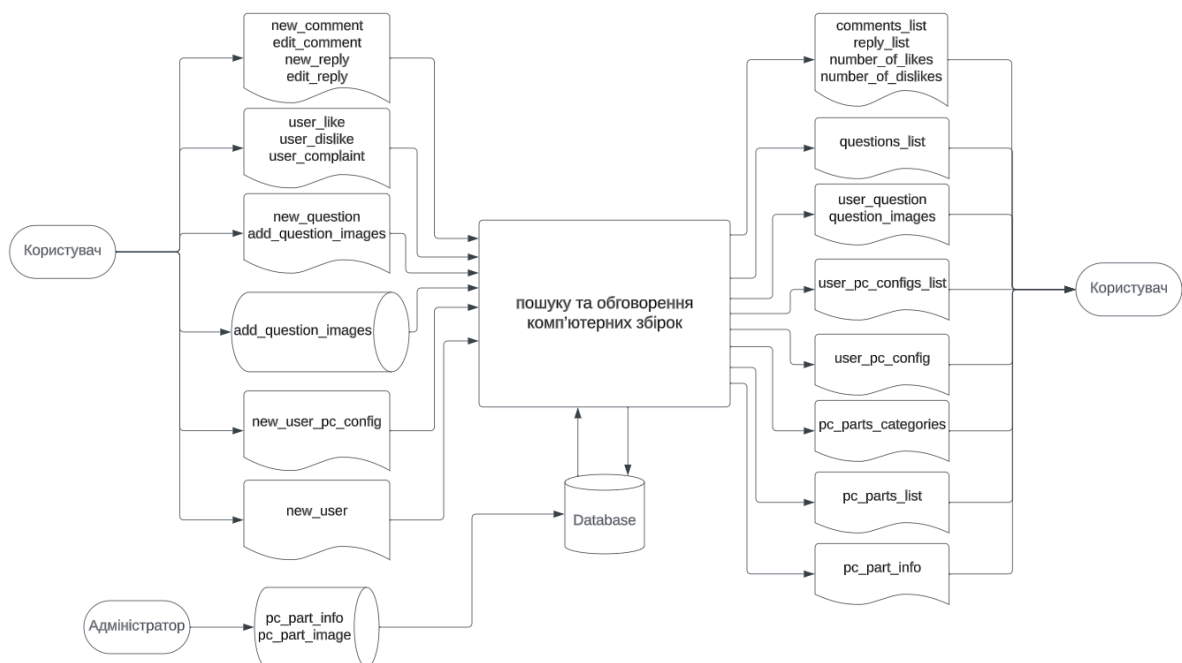


Рисунок 2.2.1.1 — Інформаційна модель задачі

Таблиця 2.1 — Перелік та опис вихідних повідомлень

№	Назва вихідного повідомлення	Ідентифікатор	Форма подання	Періодичність видання	Користувачі інформації
1	Список коментарів	comments_list	Екранна форма	По запити	Відвідувачі форуму
2	Список відповідей до коментаря	reply_list	Екранна форма	По запити	Відвідувачі форуму
3	Кількість вподобайок	number_of_likes	Екранна форма	По запити	Відвідувачі форуму
4	Кількість дизлайків	number_of_dislikes	Екранна форма	По запити	Відвідувачі форуму
5	Список запитань користувачів	questions_list	Екранна форма	По запити	Відвідувачі форуму
6	Запитання користувача	user_question	Екранна форма	По запити	Відвідувачі форуму
7	Зображення прикріплені до запитання	question_images	Екранна форма	По запити	Відвідувачі форуму
8	Список комп'ютерних збірок користувачів	user_pc_configs_list	Екранна форма	По запити	Відвідувачі форуму
9	Комп'ютерна збірка користувача	user_pc_config	Екранна форма	По запити	Відвідувачі форуму
10	Список категорій комп'ютерних комплектуючих	pc_parts_categories	Екранна форма	По запити	Відвідувачі форуму
11	Список комп'ютерних комплектуючих відповідно до категорії	pc_parts_list	Екранна форма	По запити	Відвідувачі форуму
12	Інформація про комп'ютерну деталь	pc_part_info	Екранна форма	По запити	Відвідувачі форуму

Список коментарів, це вихідне повідомлення з списком коментарів користувачів що будуть представлені на сторінці.

Список відповідей до коментаря, це вихідне повідомлення в якому буде міститися список з відповідей на коментар від інших користувачів.

Кількість вподобайок, це вихідне повідомлення буде використовуватися для відображення кількості вподобайок під коментарями та іншими об'єктами.

Кількість дизлайків, це вихідне повідомлення буде використовуватися для відображення кількості дизлайків під коментарями та іншими об'єктами.

Список запитань користувачів, в цьому вихідному повідомленні буде міститися список запитань користувачів, які необхідно буде відобразити на відповідній сторінці форуму.

Запитання користувача, в цьому вихідному повідомленні буде міститися інформація про конкретне запитання від користувача, його назва, опис, інформація про те хто створив це запитання.

Зображення прикріплені до запитання, це вихідне повідомлення буде містити зображення що було прикріплене до запитання.

Список комп'ютерних збірок користувачів, в цьому повідомленні буде міститися список комп'ютерних збірок від інших користувачів.

Комп'ютерна збірка користувача, в цьому повідомленні буде міститися інформація про конкретну комп'ютерну збірку, список комплектуючих що входить до цієї комп'ютерної збірки.

Список категорій комп'ютерних комплектуючих, це вихідне повідомлення буде містити список категорій комп'ютерних комплектуючих, наприклад: процесори, відеокарти, кулери і тд.

Список комп'ютерних комплектуючих відповідно до категорії, в цьому повідомленні буде міститися список з комплектуючих відповідно до певної категорії, які необхідно буде відобразити на відповідній сторінці.

Інформація про комп'ютерну деталь, це вихідне повідомлення буде містити інформації про комп'ютерну деталь та її детальні характеристики.

Таблиця 2.2 — Перелік та опис вхідних повідомлень

№	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Інформація про комп'ютерні комплектуючі	pc_part_info	Файл або екранна форма	При першому запуску, і по мірі виходу нових комплектуючих	Адміністратор
2	Зображення комп'ютерної деталі	pc_part_image	Файл	При першому запуску, і по мірі виходу нових комплектуючих	Адміністратор
3	Дані користувача	new_user	Екранна форма	При реєстрації чи зміні особистих даних	Користувач
4	Коментар	new_comment	Екранна форма	Коли користувач залишив коментар	Користувач
5	Зміна коментаря	edit_comment	Екранна форма	Коли користувач змінює свій коментар	Користувач
6	Відповідь на коментар	new_reply	Екранна форма	Коли користувач залишає відповідь на коментар	Користувач
7	Зміна відповіді на коментар	edit_reply	Екранна форма	Коли користувач змінює свою відповідь на коментар	Користувач
8	Вподобайка	user_like	Екранна форма	Коли користувач натискає вподобайку	Користувач
9	Дизлайк	user_dislike	Екранна форма	Коли користувач натискає дизлайк	Користувач
10	Скарга	user_complaint	Екранна форма	Коли користувач натискає "поскаржитися"	Користувач
11	Запитання від користувача	new_question	Екранна форма	Коли користувач створює нове питання	Користувач
12	Зображення до запитання	add_question_images	Файл	Коли користувач створює нове питання	Користувач
13	Конфігурація комп'ютерної збірки	new_user_pc_config	Екранна форма	Коли користувач створює нову конфігурації комп'ютерної збірки	Користувач

Інформація про комп'ютерні комплектуючі, це вхідна інформація що буде вводитися в базу даних адміністратором, в ній буде міститися інформація про комп'ютерні комплектуючі.

Зображення комп'ютерної деталі, це вхідна інформація що буде вводитися адміністратором, представляє собою файл зображення.

Дані користувача, це інформація що буде задаватися користувачем при реєстрації.

Коментар, це вхідна інформація що буде створюватися користувачем коли той буде залишати коментар.

Зміна коментаря, це вхідне повідомлення буде надходити від користувача коли той буде змінювати свій коментар.

Відповідь на коментар, це вхідне повідомлення буде надходити від користувача коли він буде писати відповідь на коментар.

Зміна відповіді на коментар, це вхідне повідомлення буде надходити від користувача коли той буде змінювати свою відповідь до коментаря.

Вподобайка, вхідне повідомлення від користувача, буде надходити коли користувач буде ставити вподобайку.

Дизлайк, вхідне повідомлення від користувача, буде надходити коли користувач буде ставити дизлайк.

Скарга, вхідне повідомлення, буде надходити коли користувач буде натискати кнопку “поскаржитися” під дописом іншого користувача.

Запитання від користувача, вхідне повідомлення, що надходить від користувача коли той створює нове запитання.

Зображення до питання, вхідне повідомлення, що надходить від користувача коли той прикріплює зображення до свого питання.

Конфігурація комп'ютерної збірки, це вхідне повідомлення буде надходити коли користувач буде створювати новий варіант комп'ютерної збірки за допомогою конфігуратора комп'ютерних збірок.

2.2.2 Алгоритм розв'язання задачі

Одна з важливих формул що буде використовуватися для підбору блоку живлення, це формула для розрахунку потужності блоку живлення:

$$W = \left(\sum_{i=0}^n consumption_i \right) * 1.3 \quad (1)$$

де:

$consumption_i$ - максимальне енергоспоживання конкретної комп'ютерної деталі що входить до збірки,

n - загальна кількість деталей що входять до збірки.

Як можна побачити в цій формулі відбувається множення суми максимального енергоспоживання всіх комп'ютерних комплектуючих що входять до збірки на коефіцієнт 1.3. Так відбувається через те що у блоку живлення має бути певний запас, а також через те що зазвичай блоки живлення мають найбільший коефіцієнт корисної дії при неповному навантаженні.

Одним з показників що розраховується можна виділити середню вартість комп'ютерної збірки. Для обрахування цього показника буде використовуватися наступна формула:

$$C_{avg} = \frac{\sum_{i=0}^n \sum_{j=0}^m cost_{ij}}{n} \quad (2)$$

де:

C_{avg} - середня вартість комп'ютерної збірки,

n - кількість комп'ютерних збірок,

m - кількість комплектуючих в конкретній комп'ютерній збірці,

$cost_{ij}$ - вартість конкретної комп'ютерної деталі.

Ще одним показником можна взяти індекс популярності, для його обрахунку використовується наступна формула:

$$P = \frac{2x - n}{n} \quad (3)$$

де:

P - індекс популярності,

x - кількість вподобайок,

n - загальна кількість відгуків.

Також було розроблено алгоритм розв'язання задачі (рисунок 2.2.2.1).

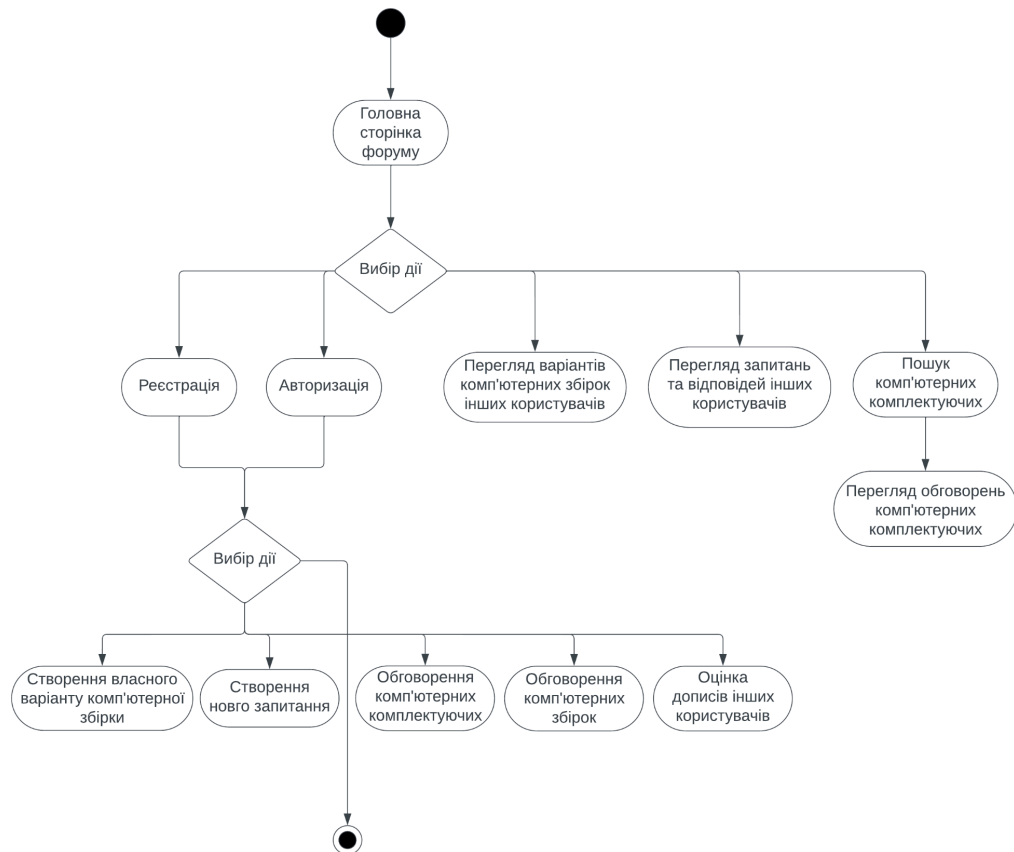


Рисунок 2.2.2.1 — алгоритм розв'язання задачі

2.3. Моделювання інформаційної системи

2.3.1. Моделювання поведінки системи

На діаграмі прецедентів що подана на рисунку 2.3.1.1 в узагальненому вигляді представлені функціональні вимоги у вигляді варіантів використання як типової взаємодії системи з акторами.

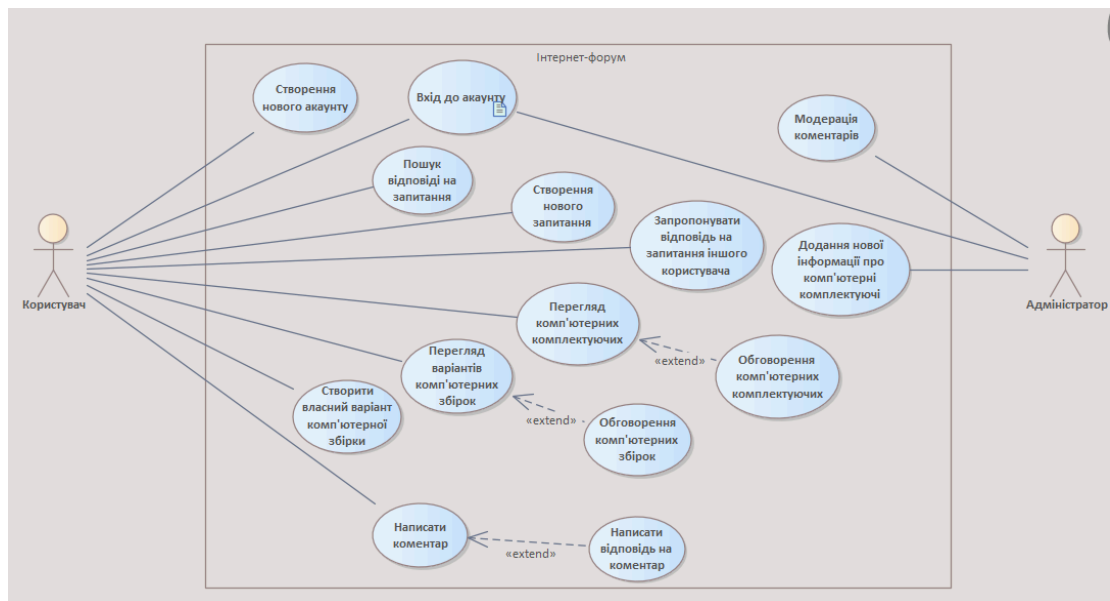


Рисунок 2.3.1.1 — діаграма прецедентів

Діаграма послідовності для авторизації користувача в системі представлена на рисунку 2.3.1.2, користувач вводить свої дані і вони відправляються на сервер де порівнюються з даними з бази даних і якщо все вірно то користувач входить в акаунт.

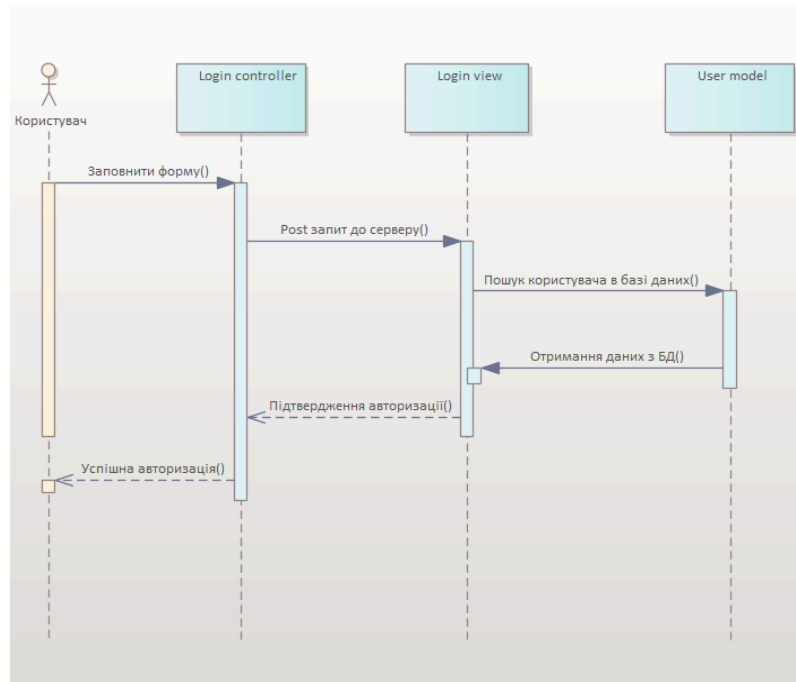


Рисунок 2.3.1.2 — діаграма послідовності входу

Ще однією діаграмою послідовності є діаграма реєстрації нового акаунту користувачем що представлена на рисунку 2.3.1.3, користувач заповнює форму і дані відправляються на сервер, далі відбувається перевірка що користувача з таким логіном ще не існує в системі і якщо вірно дані про користувача зберігаються в системі, а см користувач успішно реєструється.

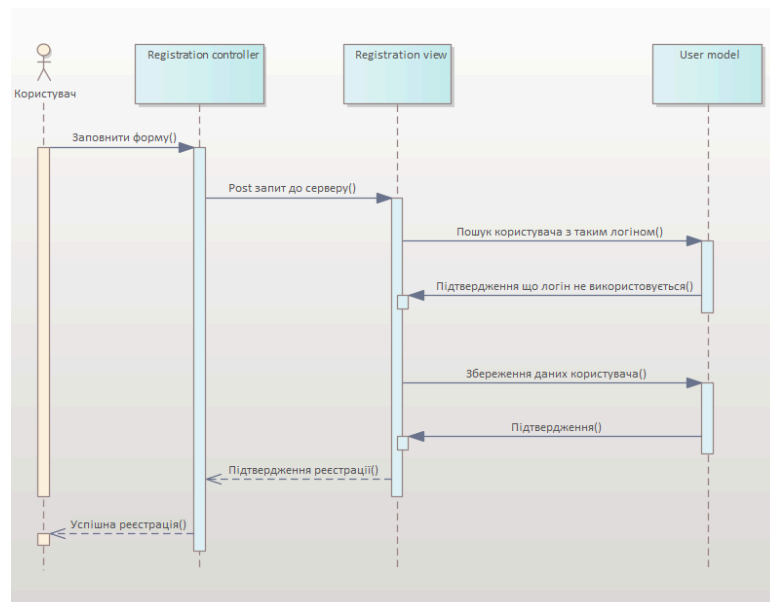


Рисунок 2.3.1.3 — діаграма послідовності реєстрації нового акаунту

Іншою діаграмою послідовності є діаграма послідовності створення коментаря користувачем, рисунок 2.3.1.4, користувач пише коментар

заповнюючи відповідну форму після чого натискає кнопку збереження та публікації і форма надсилається на сервер, де обробляється і новий коментар зберігається в базі даних.

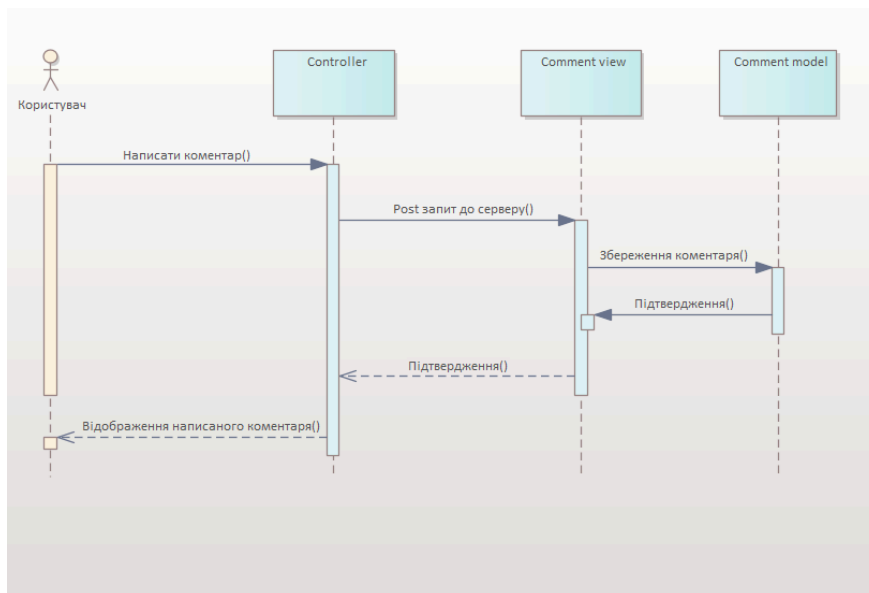


Рисунок 2.3.1.4 — діаграма послідовності написання коментаря

Також на рисунку 2.3.1.5 представлено діаграму послідовності створення нового запитання користувачем.

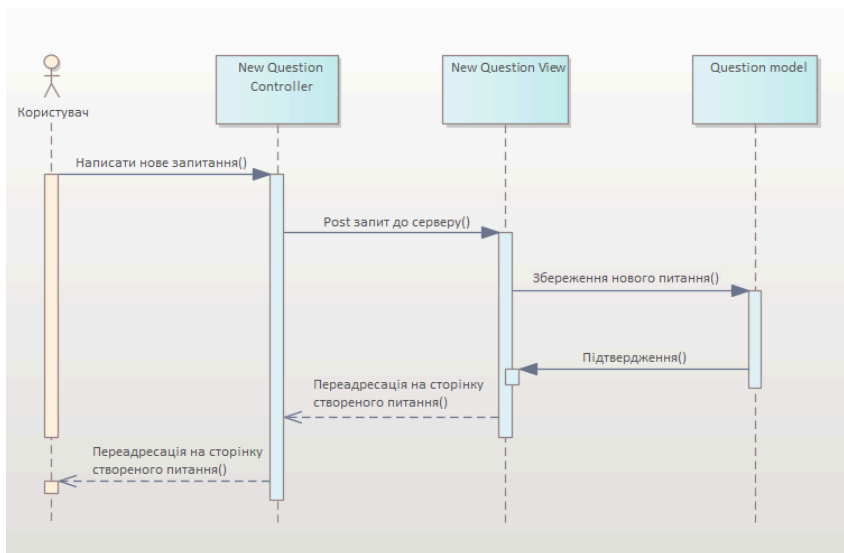


Рисунок 2.3.1.5 — діаграма послідовності створення нового запитання

2.3.2. Моделювання структури системи

В розділі 2.3.2 наводиться внутрішня структура системи, на рисунку 2.3.2.1 представлено подання внутрішньої структури системи у вигляді ієрархії блоків.

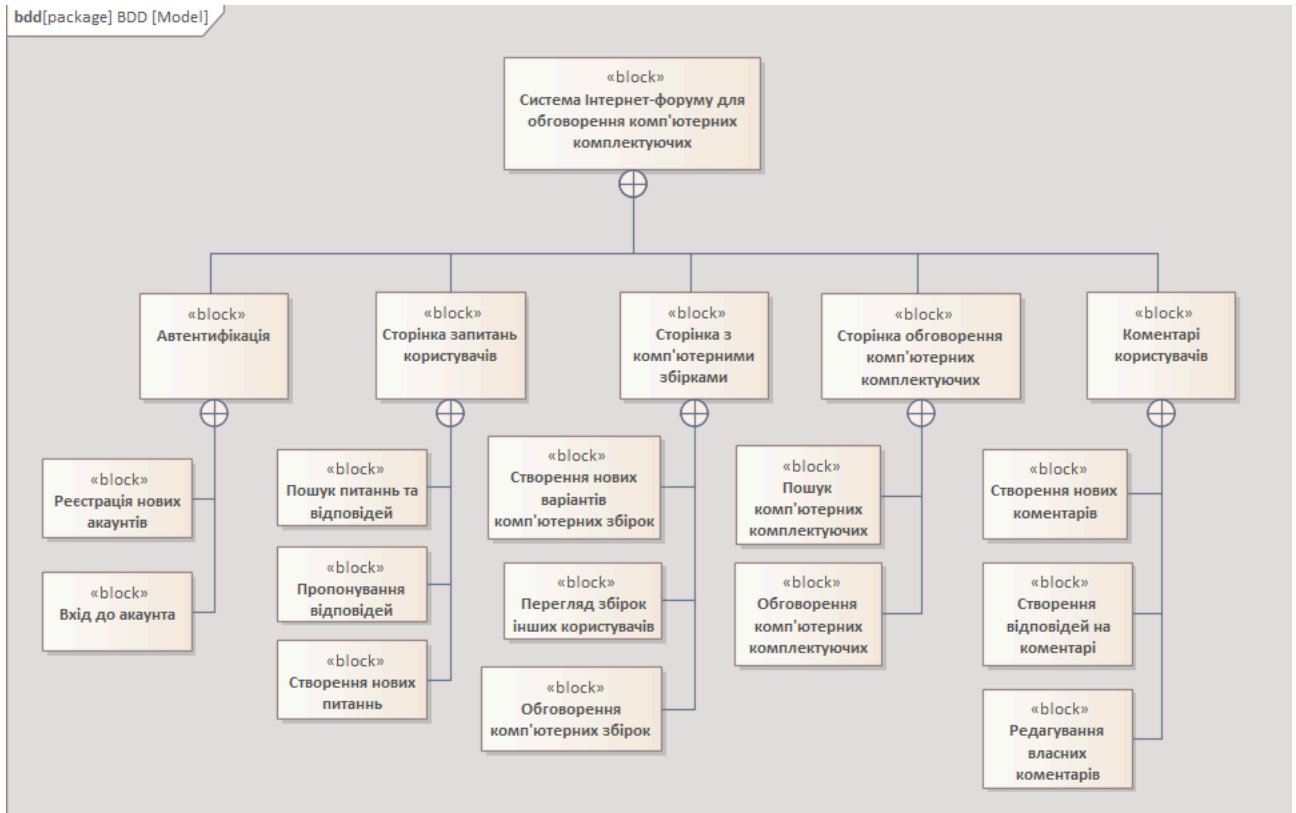


Рисунок 2.3.2.1 — Діаграма визначення блоків

На рисунку 2.3.2.2 представлена діаграма внутрішніх блоків системи, а також їх з'єднання між собою через порти, а також потоків, що існують між частинами і портами.

Загалом в системі передбачені наступні блоки: блок автентифікації який відповідає за реєстрацію та дані користувачів, блок коментарів користувачів в якому зберігаються коментарі користувачів їх обговорення, блок комп'ютерних комплектуючих в якому зберігатиметься інформація про характеристики комп'ютерних комплектуючих, блок з комп'ютерними збірками користувачів де зберігаються варіанти комп'ютерних збірок запропонованих користувачами, а також блок з запитаннями користувачів.

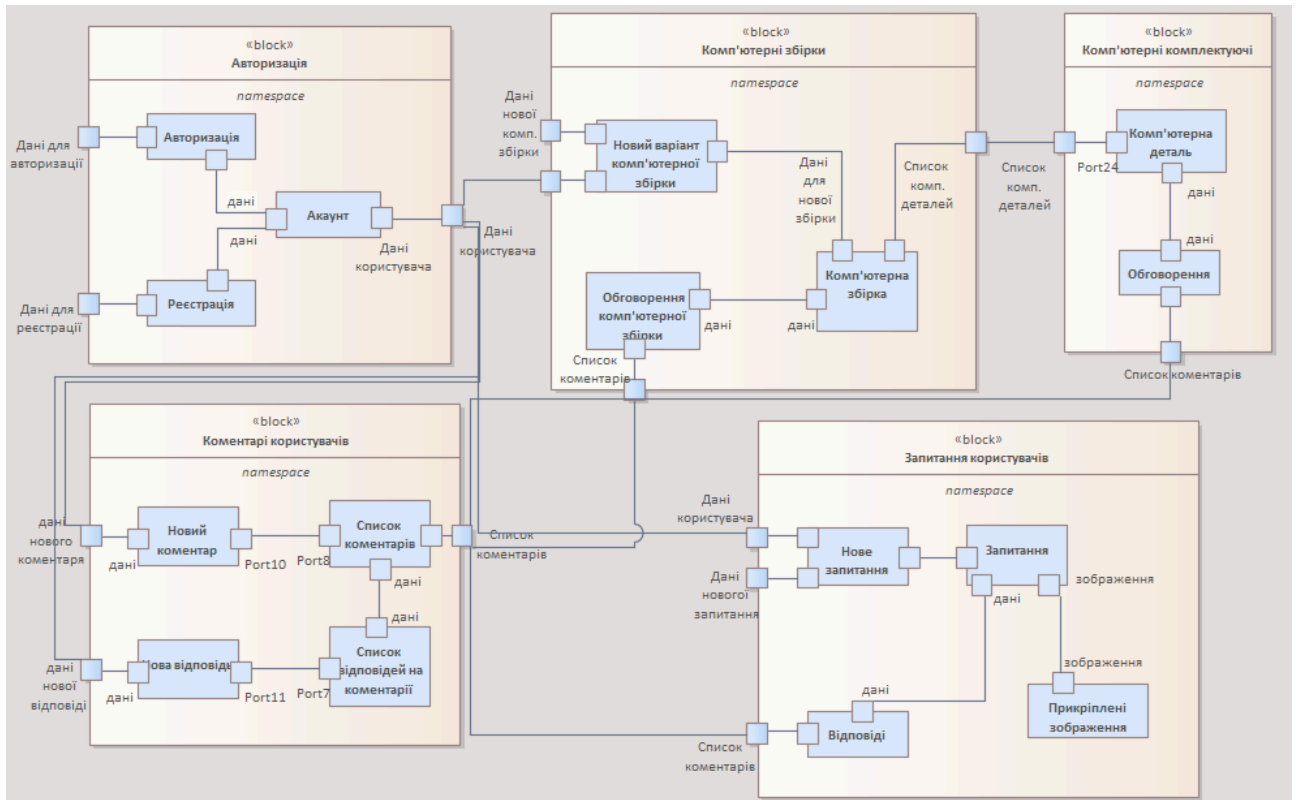
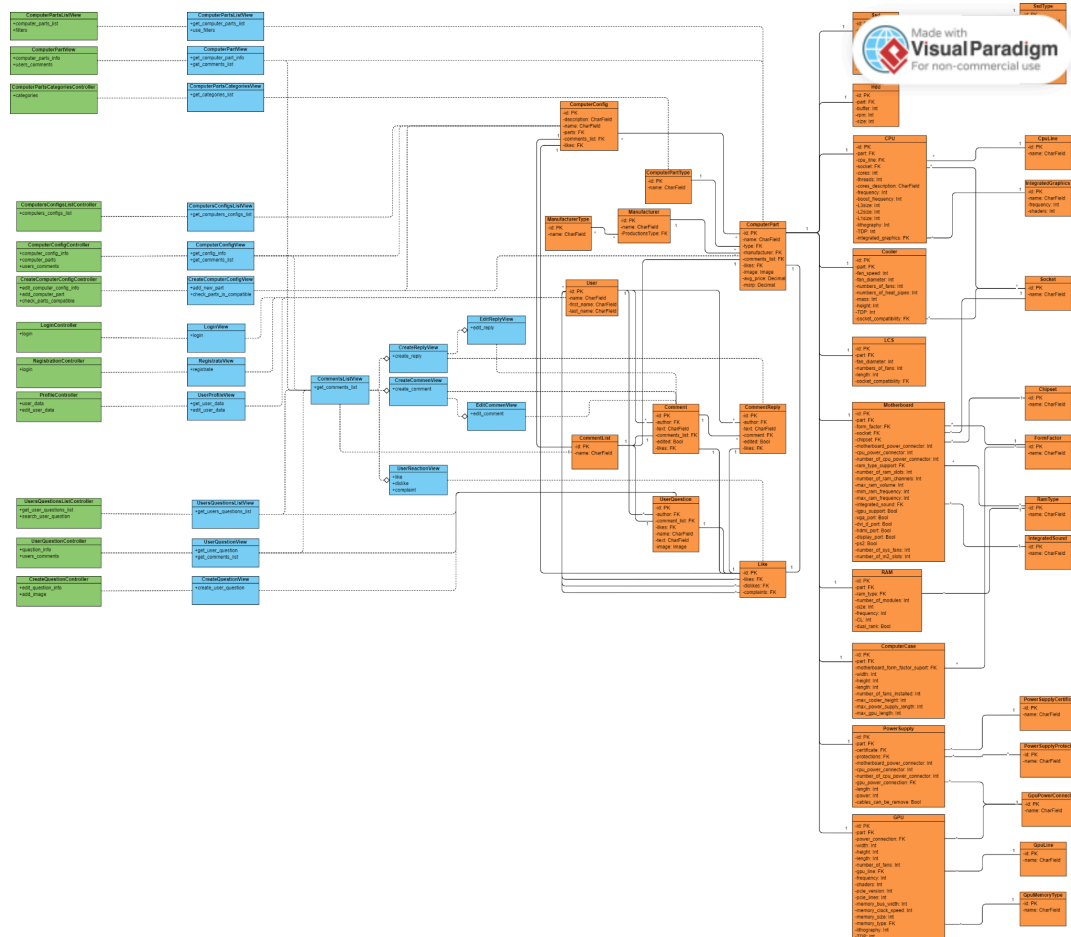


Рисунок 2.3.2.2 — Діаграма внутрішніх блоків

На рисунку 2.3.2.3 представлено діаграму класів в якій описані основні класи що присутні в системі а також зв'язки між ними, проте слід зазначити що на діаграмі зображені не всі класи що присутні в системі. Так наприклад клас “Computer Part” є класом до якого будуть мати зв'язок типу один до одного багато інших класів кожен з яких буде описувати конкретний вид комп'ютерної деталі з її характеристиками, проте на діаграмі всі ці підкласи не представлені.



Як можемо бачити з діаграми класів в базі даних будуть присутні такі класи як: “Comment List”, “Comment”, “Comment Reply” а також “Likes” ці класи будуть відноситися до модулю коментарів який буде використовуватися в інших модулях системи. Ще одним модулем системи є модуль з запитаннями користувачів до якого буде відноситися клас “Question”. В системі також передбачений модуль з даними про комп’ютерні комплектуючі до якого будуть відноситися такі класи як: “Computer Part Type”, “Computer Part”, “Manufacturer” і “Manufacturer Type”. Ще одним модулем системи буде модуль з варіантами комп’ютерних конфігурацій запропонованих користувачами до якого буде відноситися клас “Computer Config”. Також в системі передбачений модуль з даними про користувачів до якого відноситься клас “User”.

2.3.3. Розробка тест-кейсів

В даному розділі наведено тест-кейси та документацію до них підготовлену за допомогою засобів Enterprise Architect.

Першим тест-кейсом є тест створення нового акаунту, деталі тест-кейсу представлені на рисунку 2.3.3.1.

Рисунок 2.3.3.1 — Деталі тест-кейсу

Як бачимо виконання тест-кейсу передбачає шість пунктів: перехід на сторінку реєстрації, введення логіну а також паролю підтвердження паролю шляхом його повторного введення, введення додаткової інформації такої як прізвище та ім'я, після чого необхідно натиснути кнопку реалізації. При позитивному сценарій буде зареєстрований новий акаунт, а користувача буде перенаправлення на головну сторінку сайту. При негативному сценарію форма реєстрації буде оновлена з відповідним повідомленням в чому причина (наприклад форма заповнена не повністю, занадто короткий пароль, або те що користувач з таким логіном вже існує). Також за допомогою засобів Enterprise Architect було згенеровано відповідну документацію по даному тест-кейсу представлену на рисунку 2.3.3.2.

Model Report	10 April, 2024
Створення нового акаунту	
UseCase in package 'Use Cases'	
Створення нового акаунту Version 1.0 Phase 1.0 Proposed andru created on 01.04.2024. Last modified 01.04.2024	
EXTERNAL REQUIREMENTS	
☒ Requirement. Реєстрація нових акаунтів	
☒ Requirement. Система автентифікації	
OUTGOING STRUCTURAL RELATIONSHIPS	
↔ Realization from Створення нового акаунту to Система автентифікації [Direction is 'Source -> Destination'.]	
↔ Realization from Створення нового акаунту to Реєстрація нових акаунтів [Direction is 'Source -> Destination'.]	
ASSOCIATIONS	
↗ Association (direction: Unspecified) Source: Public (Actor) Користувач Target: Public (UseCase) Створення нового акаунту	
TESTING	
☒ Standard Unit test. Тест реєстрації нового акаунту Description: 1) Перейти на сторінку реєстрації 2) Придумати та ввести логін 3) Придумати та ввести пароль 4) Підтвердити пароль 5) Ввести допоміжну інформацію (прізвище, ім'я) 6) Натиснути на кнопку реєстрації Input: логін пароль прізвище, ім'я Results: зареєстрований новий акаунт 10.04.2024 - Pass Run by: andru Checked by: andru [Last run Pass at 10.04.2024 by andru and checked by andru.]	

Рисунок 2.3.3.2 — Документація тест-кейсу створення нового акаунту

Наступним тест-кейсом є тест входу до акаунту, на рисунку 2.3.3.3 представлено деталі даного тест-кейсу.

The screenshot shows a 'Test details' window with a blue title bar. Inside, there's a 'Test' section with a text input field containing 'Вхід до акаунту' and an 'Auto' button. Below this is a tabbed interface with 'Description', 'Input', 'Acceptance Criteria', and 'Results' tabs. The 'Description' tab is active, showing a list of four steps: 1) Перейти на сторінку входу, 2) Ввести логін, 3) Ввести пароль, and 4) Натиснути кнопку "увійти". The text in the description is underlined. Below the description, there are dropdown menus for 'Class' (set to 'Unit') and 'Type' (set to 'Standard'). At the bottom, there's a 'Run' section with fields for 'Run Date' (11.04.2024), 'Run By' (andru), 'Result' (Pass), and 'Checked By' (andru). At the very bottom, there are buttons for '<<', '>>', 'New', 'Apply', 'OK', 'Close', and 'Help'.

Рисунок 2.3.3.3 — Деталі тест-кейсу

Як можемо бачити з рисунка 2.3.3.3 даний тест-кейс складається з чотирьох кроків: перехід на сторінку авторизації, введення логіна, а також пароля, після чого необхідно натиснути кнопку входу. При позитивному сценарію буде здійснено успішний вхід до акаунту, а користувача буде переадресовано на головну сторінку. При негативному сценарію форма входу буде оновлена з відповідним повідомленням про помилку входу де має бути інформація про те що саме сталося не так, наприклад невірний логін або пароль. Також до цього тест-кейсу було згенеровано відповідну документацію що представлена на рисунку 2.3.3.4.

Вхід до акаунту

UseCase in package 'Use Cases'

Вхід до акаунту
Version 1.0 Phase 1.0 Proposed
andru created on 31.03.2024. Last modified 10.04.2024

EXTERNAL REQUIREMENTS
Requirement. Можливість входу в існуючі аканти
Requirement. Система автентифікації
SCENARIOS
Basic Path. Basic Path
OUTGOING STRUCTURAL RELATIONSHIPS
Realization from Вхід до акаунту to Система автентифікації [Direction is 'Source -> Destination'.]
Realization from Вхід до акаунту to Можливість входу в існуючі аканти [Direction is 'Source -> Destination'.]
ASSOCIATIONS
Association (direction: Unspecified) Source: Public (Actor) КористувачTarget: Public (UseCase) Вхід до акаунту
Association (direction: Unspecified) Source: Public (Actor) АдміністраторTarget: Public (UseCase) Вхід до акаунту
TESTING
Standard Unit test. Вхід до акаунту Description: 1) Перейти на сторінку входу 2) Ввести логін 3) Ввести пароль 4) Натиснути кнопку "увійти" Input: Login Пароль Results: Успішний вхід до акаунту 11.04.2024 - Pass Run by: andru Checked by: andru [Last run Pass at 11.04.2024 by andru and checked by andru.]

Рисунок 2.3.3.4 — Документація тест-кейсу входу до акаунту

Ще одним підготовленим тест-кейсом є тест створення нового питання, деталі якого представлені на рисунку 2.3.3.5.

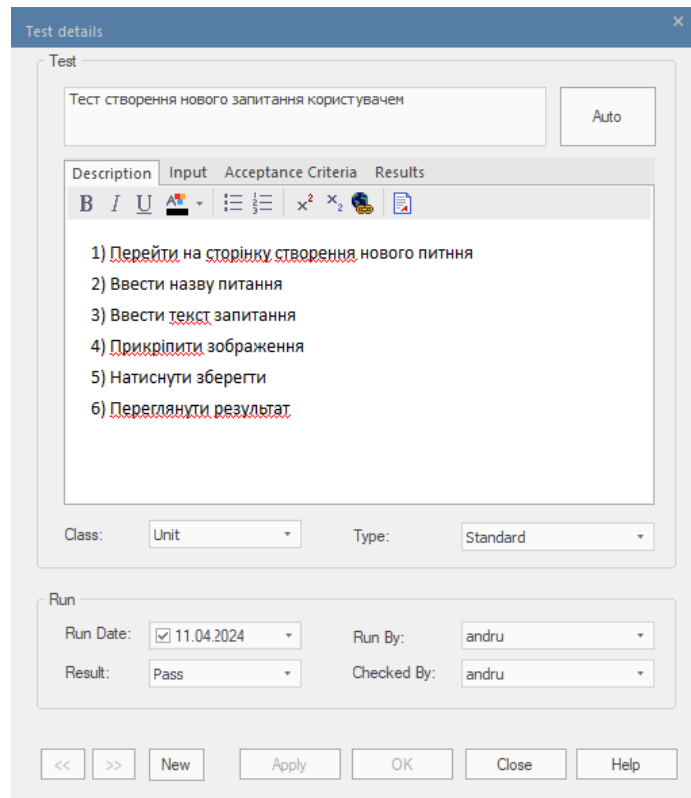


Рисунок 2.3.3.5 — Деталі тест-кейсу для створення нового запитання

Як можемо бачити з рисунка 2.3.3.5 для проведення даного тест-кейсу створення нового питання необхідно виконати наступні шість кроків, а саме: перейти на сторінку створення нового питання, ввести назву питання, ввести текст запитання, прикріпити зображення, натиснути зберегти та переглянути результат. В результаті на форумі має бути створено нове питання на яке інші користувачі можуть пропонувати відповіді. Також для даного тест-кейсу в Enterprise Architect було згенеровано відповідну документацію представлену на рисунку 2.3.3.6.

Test details

Test

Запропонувати відповідь на запитання

Auto

Description Input Acceptance Criteria Results

1) Перейти на сторінку з запитаннями користувачів
 2) Обрати запитання зі списку
 3) Ввести текст відповіді
 4) Натиснути на кнопку зберегти
 5) Переглянути результат

Class: Unit Type: Standard

Run

Run Date: 11.04.2024 Run By: andru
 Result: Pass Checked By: andru

<< >> New Apply OK Close Help

Рисунок 2.3.3.7 — Деталі тест-кейсу відповіді на запитання

Як можемо бачити для проходження даного тест-кейсу необхідно виконати п'ять кроків: перейти на сторінку з запитаннями користувачів, обрати запитання, ввести тест відповіді, натиснути кнопку зберегти щоб запропонувати відповідь після чого переглянути результат. В результаті на сторінці з запитанням користувача має з'явитися запропонована відповідь. Як і до всіх інших тест-кейсів до даного тест-кейсу було згенеровано відповідну документацію що представлена на рисунку 2.3.3.8.

Model Report

11 April, 2024

Запропонувати відповідь на запитання іншого користувача

UseCase in package 'Use Cases'

Запропонувати відповідь на запитання іншого користувача

Version 1.0 Phase 1.0 Proposed

andru created on 01.04.2024. Last modified 02.04.2024

EXTERNAL REQUIREMENTS
☑ Requirement. Можливість пропонувати відповідь
☑ Requirement. Сторінка запитань користувачів

OUTGOING STRUCTURAL RELATIONSHIPS
⬅ Realization from Запропонувати відповідь на запитання іншого користувача to Можливість пропонувати відповідь [Direction is 'Source -> Destination'.]
⬅ Realization from Запропонувати відповідь на запитання іншого користувача to Сторінка запитань користувачів [Direction is 'Source -> Destination'.]

ASSOCIATIONS
✓ Association (direction: Unspecified) Source: Public (Actor) Користувач Target: Public (UseCase) Запропонувати відповідь на запитання іншого користувача

TESTING
☒ Standard Unit test. Запропонувати відповідь на запитання Description: 1) Перейти на сторінку з запитаннями користувачів 2) Обрати запитання зі списку 3) Ввести текст відповіді 4) Натиснути на кнопку зберегти 5) Переглянути результат Input: Текст відповіді Results: Збережена відповідь до запитання користувача 11.04.2024 - Pass Run by: andru Checked by: andru [Last run Pass at 11.04.2024 by andru and checked by andru.]

Рисунок 2.3.3.8 — Тест-кейс відповіді на запитання користувача

2.3.4. Трасування вимог до системи

На рисунку 2.3.4.1 представлено діаграму трасування на якій присутні вимоги та прецеденти, а також показані зв'язки між ними.

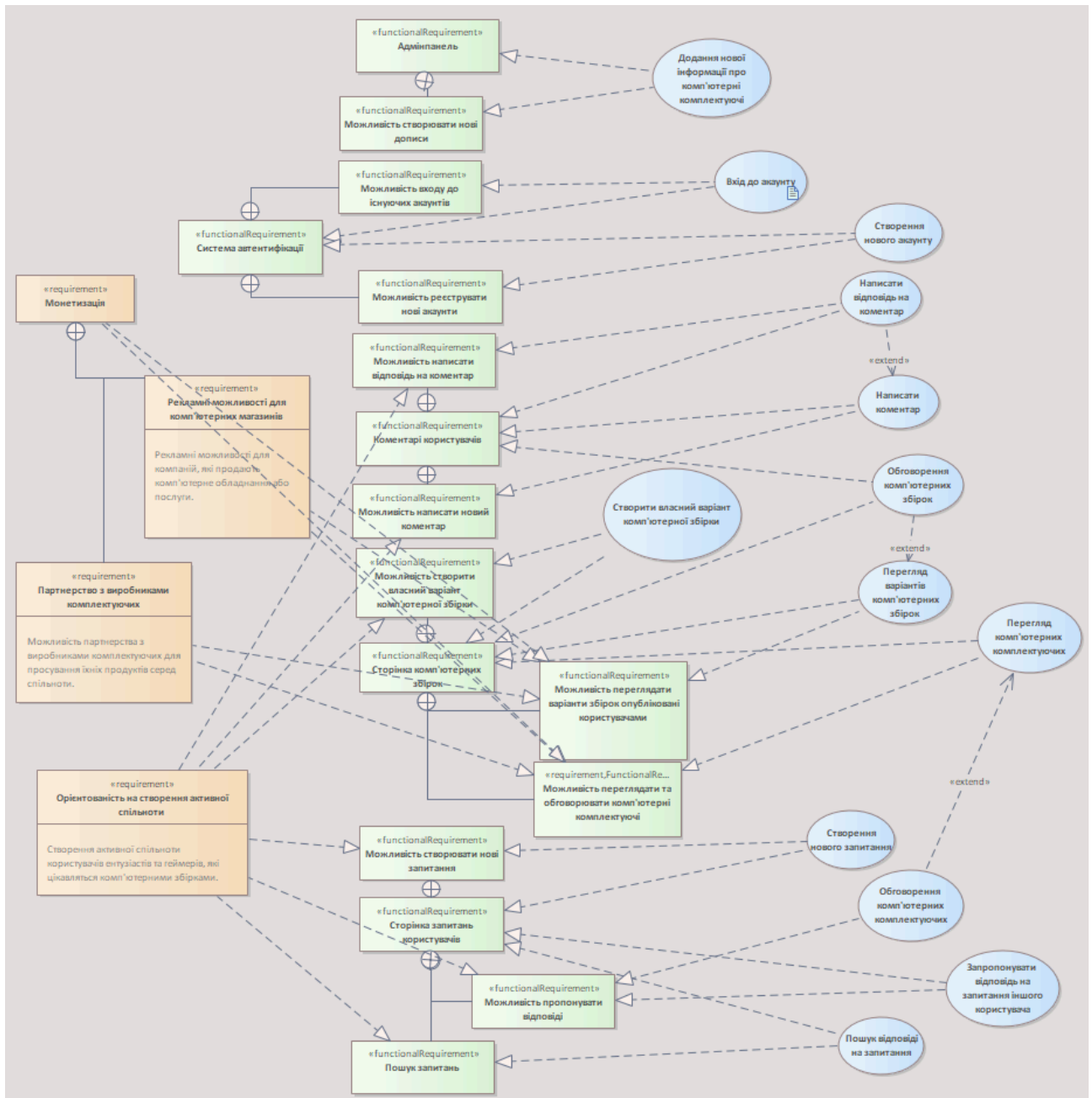


Рисунок 2.3.4.1 — Діаграма трасування

Також на рисунках 2.3.4.2 та 2.3.4.3 зображені матриці трасування на яких описані ті ж зв'язки що і на діаграмі трасування.

Target +	Requirements:Адмінпанель	Requirements:Взаємодія з інт	Requirements:Коментарі кор	Requirements:Модерація ком	Requirements:Можливість вх	Requirements:Можливість на	Requirements:Можливість на	Requirements:Можливість пе	Requirements:Можливість пе	Requirements:Можливість пр	Requirements:Можливість ре	Requirements:Можливість ре	Requirements:Можливість ст	Requirements:Можливість ст	Requirements:Можливість ст	Requirements:Навігація по ф	Requirements:Пошук запитан	Requirements:Панель навігац	Requirements:Система автент	Requirements:Сторінка запит	Requirements:Сторінка комп'
+ Source																					
Use Cases::Вхід до акаунту					↑														↑		
Use Cases::Додання нової ...	↑												↑								
Use Cases::Запропонуват...									↑											↑	
Use Cases::Модерація ком...	↑																				
Use Cases::Написати відп...			↑			↑															
Use Cases::Написати коме...			↑				↑														
Use Cases::Обговорення к...			↑																		↑
Use Cases::Обговорення к...									↑												↑
Use Cases::Перегляд варіа...								↑													↑
Use Cases::Перегляд комп...									↑												↑
Use Cases::Пошук відпові...																↑				↑	
Use Cases::Створення нов...											↑							↑			
Use Cases::Створення нов...													↑						↑		
Use Cases::Створити влас...												↑									↑

Рисунок 2.3.4.2 — Матриця трасування

Target +	Requirements:Адмінпанель	Requirements:Взаємодія з інтернет-форумом	Requirements:Коментарі користувачів	Requirements:Модерація коментарів	Requirements:Можливість входу до існуючих акаунтів	Requirements:Можливість написати відповідь на коментар	Requirements:Можливість написати новий коментар	Requirements:Можливість переглядати варіанти збірок опук	Requirements:Можливість переглядати та обговорювати ко	Requirements:Можливість пропонувати відповіді	Requirements:Можливість редагувати свої коментарі	Requirements:Можливість реєструвати нові акаунти	Requirements:Можливість створити власний варіант комент'с	Requirements:Можливість створювати нові дописи	Requirements:Можливість створювати нові запитання	Requirements:Навігація по форуму	Requirements:Пошук запитань	Requirements:Панель навігації по форуму	Requirements:Система автентифікації	Requirements:Сторінка запитань користувачів	Requirements:Сторінка комент'с
+ Source																					
Бізнес вимоги::Монетизація								↑	↑												
Бізнес вимоги::Орієнтованість на ств...				↑		↑	↑			↑			↑		↑		↑				
Бізнес вимоги::Партнерство з вироб...								↑	↑												
Бізнес вимоги::Рекламні можливост...								↑	↑												

Рисунок 2.3.4.3 — Матриця трасування

2.4 Моделювання користувацького інтерфейсу

Було розроблено клікабельний прототип дизайну інтерфейсу користувача за допомогою інструменту прототипування “Figma” який можна переглянути за посиланням [12].

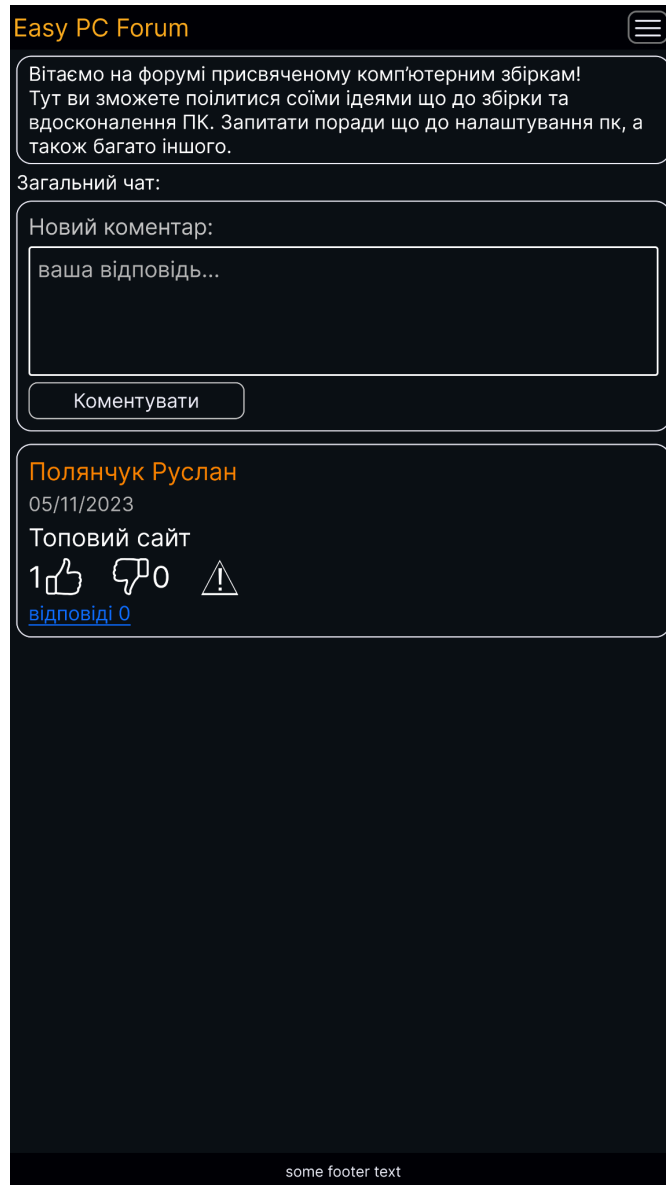


Рисунок 2.4.1 — Головна сторінка

На головній сторінці знаходиться привітальний текст, а також є загальний чат в якому усі користувачі можуть залишати свої коментарі.

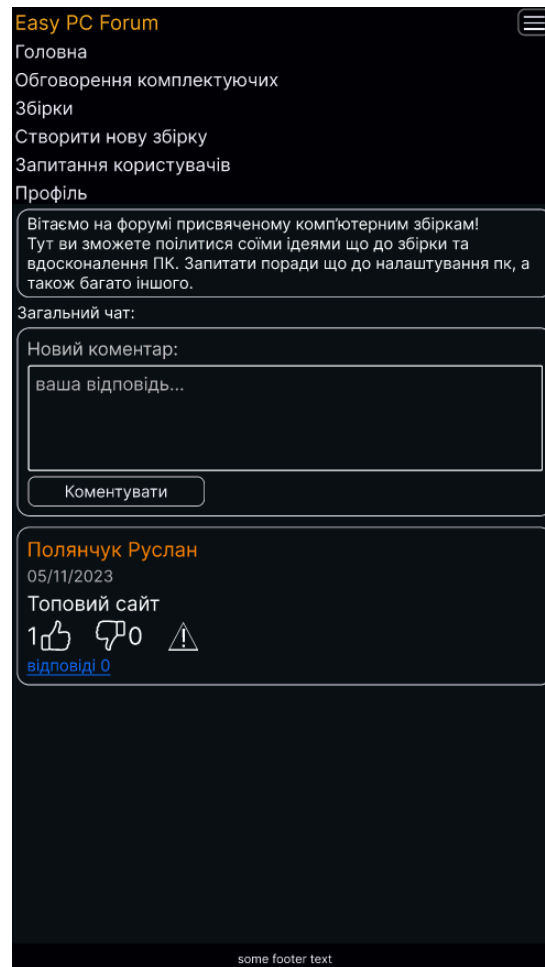


Рисунок 2.4.2 — Панель навігації

У верхній частині знаходиться навігаційна панель на якій знаходяться основні посилання на інші сторінки форуму такі як: головна сторінка, збірки користувачів, створення нової збірки, запитання користувачів та профіль. Слід зазначити що якщо користувач не буде авторизований, то замість посилання на профіль буде відображатися посилання на сторінку авторизації. Якщо ж користувач буде авторизований як адміністратор, то крім вищезгаданих посилань в навігаційній панелі також буде посилання на адмін панель.

В нижній частині сторінки знаходиться колонтитул.

Easy PC Forum

Звання: новий користувач

Ім'я

Андрій

Фамілія

Мельник

Зберегти зміни

Змінити акаунт

Мої збірки

some footer text

Рисунок 2.4.3 — Сторінка профілю користувача

На сторінці профілю користувача будуть відображатися його дані такі як прізвище та ім'я, які користувача зможе змінити в разі необхідності ввівши нові та натиснувши кнопку зберегти. Також нижче будуть відображатися збірки користувача.

Окрім кнопки зберегти ліворуч також буде кнопка “змінити акаунт” натиснувши яку користувача буде виходити з акаунту та попадати на сторінку авторизації.

Easy PC Forum

Логін:
Andriy

Пароль:

Увійти

[Немаєте акаунту? Зареєструйте новий!](#)

some footer text

Рисунок 2.4.4 — Сторінка авторизації

На сторінці авторизації користувач може увійти до свого акаунту заповнивши форму, ввівши свої логін та пароль і натиснувши кнопку “увійти”.

В разі якщо користувач ще не має власного акаунту, в нижній частині форми є посилання перейшовши за яким користувач попаде на сторінку реєстрації.

Easy PC Forum

Придумайте логін:
Andriy

Ім'я:
Андрій

Фамілія:
Мельник

Придумайте пароль:

Введіть пароль повторно для підтвердження:

Реєстрація!

[Вже маєте акаунт? Увійдіть тут!](#)

some footer text

Рисунок 2.4.5 — Сторінка реєстрації

На сторінці реєстрації користувач може зареєструвати новий акаунт заповнивши форму. Для заповнення форму користувачу необхідно буде придумати логін ввести такі дані як ім'я та фамілію, а також придумати пароль після чого ввести його повторно для підтвердження. Після заповнення форми і натискання кнопки реєстрації, якщо всі поля були заповнені коректно система створить новий акаунт і користувача буде авторизовано. Якщо ж при заповненні форми були введені некоректні дані, наприклад занадто простий пароль, то система видасть відповідне повідомлення і користувач може повторно заповнити форму.

В нижній частині форми також є посилання на сторінку авторизації.

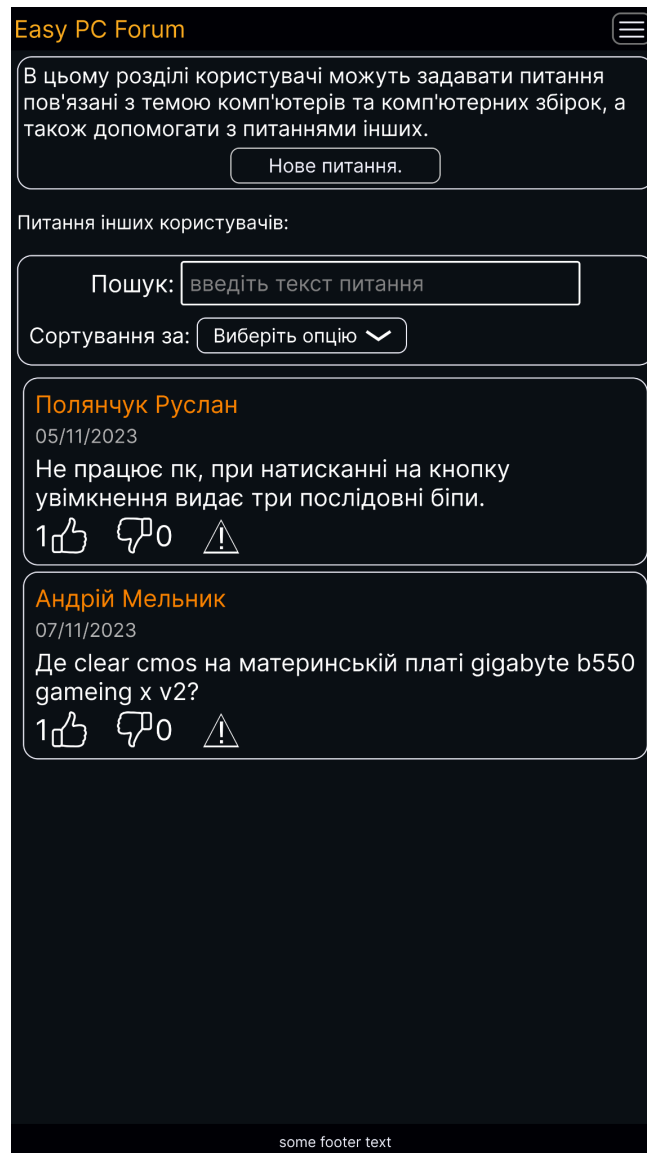


Рисунок 2.4.6 — Сторінка з запитаннями користувачів

Одним з розділів форуму є питання користувачів, тут користувачі можуть задавати питання а інші користувачі зможуть на них відповідати. На сторінці з запитаннями користувачів є коротке пояснення що до того як це працює.

Головною частиною даної сторінки є список запитань користувачів, в якому відображаються самі питання клікнувши на які можна перейти до обговорення конкретного питання. Також є можливість виставити сортування за датою створення питання та здійснювати пошук запитань.

Також на сторінці є можливість перейти на сторінку для створення нового питання.

The screenshot shows a web interface for 'Easy PC Forum'. At the top left is the forum name 'Easy PC Forum' in orange, and at the top right is a hamburger menu icon. The main form is titled 'Назва (коротко опишіть запитання):' and contains a text input field with 'text data'. Below this is a section titled 'введіть ваше запитання:' with a larger text area also containing 'text data'. Underneath is a file upload section with the text 'За необхідності прикріпіть зображення:' followed by a button labeled 'Вибрати файл'. At the bottom of the form is a button labeled 'Зберегти і опублікувати'. The entire form is set against a dark background. At the very bottom of the page, there is a small line of text: 'some footer text'.

Рисунок 2.4.7 — Сторінка створення нового питання

Для створення нового питання користувачу необхідно буде заповнити відповідну форму, де вказати назву запитання, ввести саме запитання, та за необхідності прикріпити фото.



Рисунок 2.4.8 — Приклад сторінки з запитанням користувача

На сторінці з запитанням користувача буде відображатися саме питання прикріплене зображення якщо таке є, а також оцінки користувачів.

Під самим запитанням буде можливість залишити свою відповідь, а також там відображатимуться відповіді інших користувачів.

Easy PC Forum

Відповіді:

Новий коментар:

ваша відповідь...

Коментувати

Полянчук Руслан

05/11/2023

Спробуй витягнути батарейку і трохи почекати, налаштування біос мають скинутися після цього.

1

0

[відповіді 1](#)

Додати відповідь:

ваша відповідь...

Коментувати

Мельник Андрій

05/11/2023

Нажаль не допомогло(

1

0

Гончарук Андрій

05/11/2023

На цій платі CLR_CMOS має бути в нижній правій частині (трохи під радіатором чіпсету).

1

0

[відповіді 1](#)

some footer text

Рисунок 2.4.9 — Приклад сторінки з запитанням користувача, відповіді

У кожного коментаря є свої оцінки, а також можуть бути відповіді від інших користувачів, для перегляду яких необхідно розгорнути коментар. Також до коментаря користувач зможе написати власну відповідь.

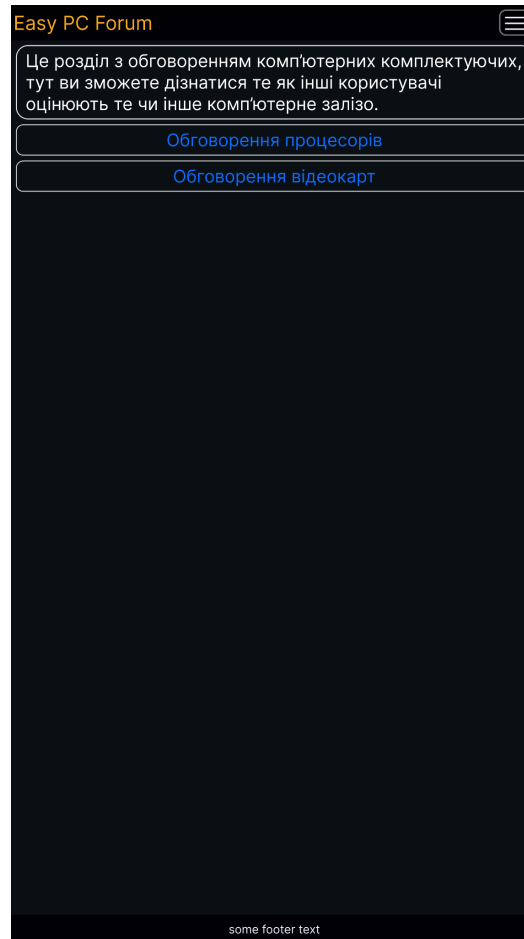


Рисунок 2.4.10 — Сторінка з посиланнями на обговорення комплектуючих

Ще одним з розділів є розділ з обговоренням комп'ютерних комплектуючих. Даний розділ сайту буде розділений на підрозділи по типу комплектуючих. В прототипі представлені два підрозділи, для процесорів та відеокарт.

Easy PC Forum

Відеокарти

Пошук:

Обсяг відеопам'яті: від до

Серія відеокарти:

☐ RX 7000
☐ RTX 4000

☐ RX 6000
☐ RTX 3000

☐ RX 5000
☐ RTX 2000

☐ RX 500
☐ GTX 1000

☐ RX 400

Виробник:
☐ Sapphire



Sapphire AMD Rx 6800xt Pulse
4608 поточкових процесори
частота ГП 2310 Mhz
шина 256 bit
відеопам'ять 16 Gb gddr6
3 вентилятори
TDP 300 ват



Gigabyte GeForce RTX 3060
кількість ядер CUDA 3584
частота ГП 1837 Mhz
обсяг пам'яті 12 Gb gddr6
шина пам'яті 192 бит
TDP 170 ват

some footer text

Рисунок 2.4.11 — Сторінка з списком посилань на обговорення відеокарт

На сторінці з списком відеокарт можна буде вибрати фільтри для пошуку відеокарт, а також можна буде здійснювати пошук за назвою. Після того як користувач знайде те що шукав він зможе натиснути на відповідне посилання щоб перейти до обговорення.

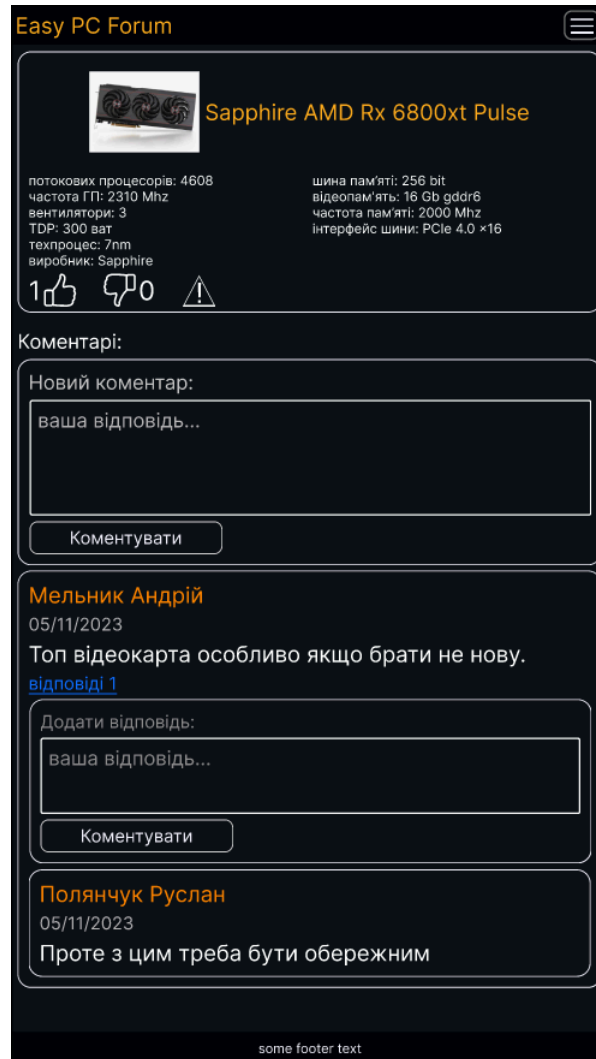


Рисунок 2.4.12 — Приклад сторінки з обговоренням відеокарти

На сторінці обговорення комп'ютерної деталі користувач зможе залишити свій коментар, відповідати на коментарі інших користувачів, а також може залишити свою оцінку.

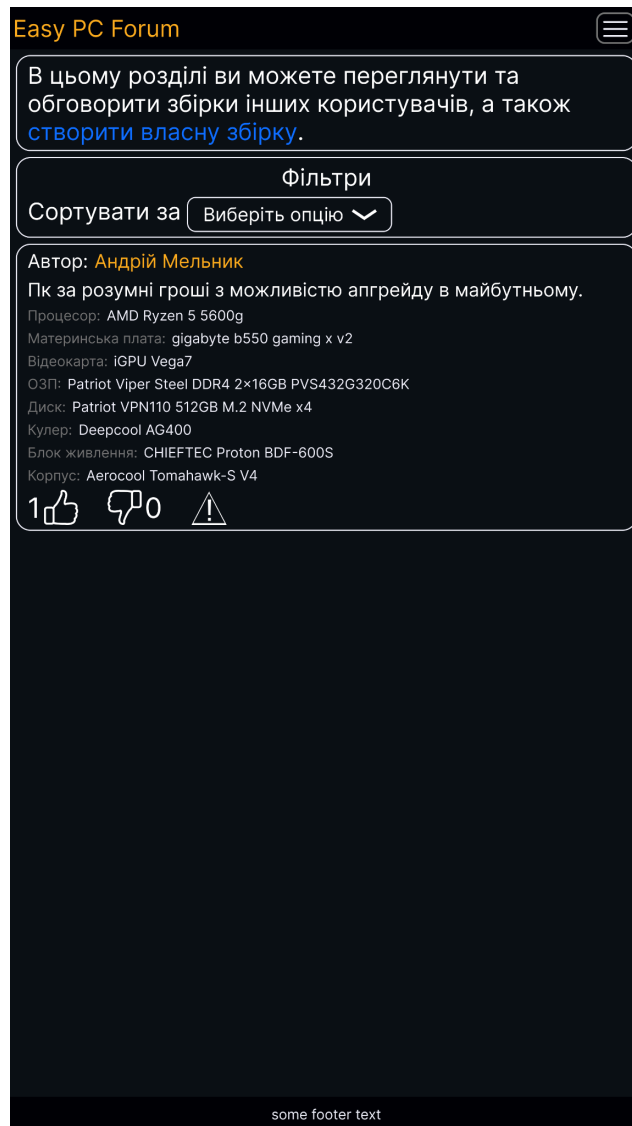


Рисунок 2.4.13 — Сторінка з комп'ютерними збірками користувачів

Розділ форуму з комп'ютерними збірками від користувачів є найважливішим. В цьому розділі користувачі можуть пропонувати варіанти своїх збірок, а інші користувачі можуть їх переглядати оцінювати та коментувати.

На сторінці з комп'ютерними збірками користувачів є короткий опис розділу з посиланням на сторінку створення власної комп'ютерної збірки, а також є список з збірок інших користувачів. Натиснувши на збірку в списку можна буде перейти до її сторінки, де можна переглянути деталі збірки і коментарі до неї.

Easy PC Forum

Дайте назву вашій збірці:

назва

Додайте опис вашої збірки:

опис

Оберіть комплектуючі що будуть у вашій збірці

Обрати процесор

Обрати материнську плату

Обрати відеокарту

Обрати ОЗП

Обрати блок живлення

Обрати SSD диск

Обрати Кулер

Обрати Корпус

Зберегти

Опублікувати

some footer text

Рисунок 2.4.14 — Сторінка створення нової комп'ютерної збірки

На сторінці для створення нової комп'ютерної збірки, користувач може обрати необхідні комплектуючі, а також ввести назву для своєї збірки і ввести опис, після чого зберегти та опублікувати її.

Easy PC Forum

Дайте назву вашій збірці:

назва

Додайте опис вашої збірки:

опис

Оберіть комплектуючі що будуть у вашій збірці

Обрати процесор

Пошук за назвою

Введіть назву

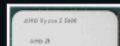
Пошук



AMD Ryzen 5 5600g
 6 ядер 12 потоків
 мікроархітектура Zen3
 частота в турбобусті 4450 Mhz
 техпроцес 7 нанометрів
 вбудована графіка Vega7
 TDP 65 Вт



Intel core i5 11400
 6 ядер 12 потоків
 ядра Rocket Lake
 частота в бусті 4400 Mhz
 техпроцес 14 нанометрів
 вбудована графіка Intel UHD Graphics 730
 TDP 65 Вт



AMD Ryzen 5 5600
 6 ядер 12 потоків

Обрати материнську плату

Пошук за назвою

Введіть назву

Пошук



Gigabyte B550 GAMING X V2
 сокет AM4
 чіпсет B550
 форм-фактор ATX
 кількість слотів пам'яті 4
 максимальний обсяг пам'яті 128 ГБ
 максимальна частота пам'яті 4733
 кількість слотів M.2 2 штуки



MSI B560M PRO-E

some footer text

Рисунок 2.4.15 — Приклад вибору деталей для комп'ютерної збірки

Для вибору деталей при створенні комп'ютерної збірки користувач зможе скористатися пошуком за назвою. Після вибору, деталь буде відображатися в самій збірці.

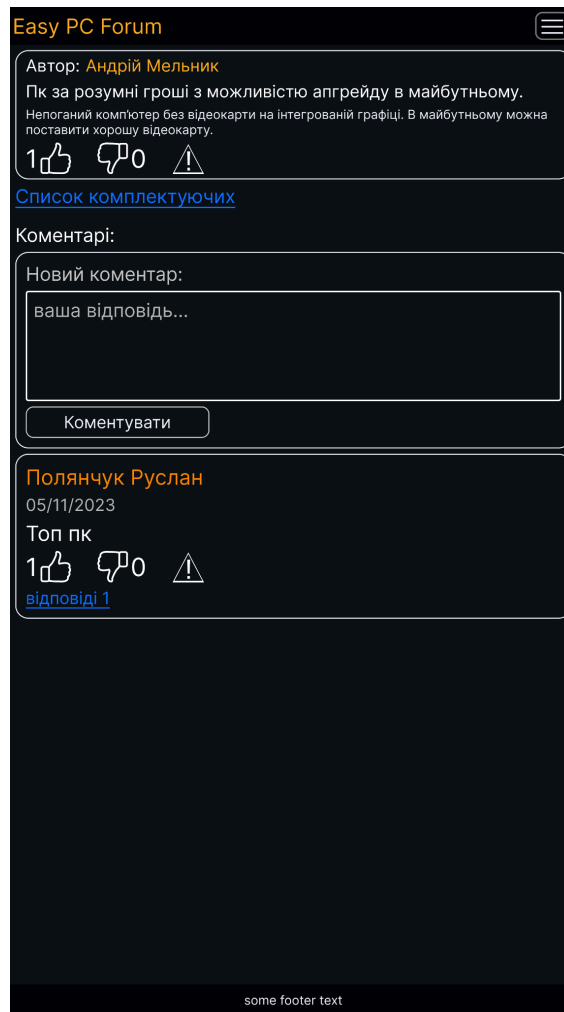


Рисунок 2.4.16 — Приклад сторінки обговорення збірки користувача

На сторінці з обговоренням збірки користувача форуму можна буде залишати свої коментарі та переглядати і відповідати на коментарі інших користувачів. Також буде можливість переглянути список деталей що входять до збірки і оцінити збірку.

Easy PC Forum

Автор: **Андрій Мельник**

Пк за розумні гроші з можливістю апгрейду в майбутньому.
Непоганий комп'ютер без відеокарти на інтегрованій графіці. В майбутньому можна поставити хорошу відеокарту.

1 👍 💬 ⚠️

[Список комплектующих](#)



AMD Ryzen 5 5600g
6 ядер 12 потоків
мікроархітектура Zen3
частота в турбобусті 4450 Mhz
техпроцес 7 нанометрів
вбудована графіка Vega7
TDP 65 Wat



Gigabyte B550 GAMING X V2
сокет AM4
чіпсет B550
форм-фактор ATX
кількість слотів пам'яті 4
максимальний обсяг пам'яті 128 ГБ
максимальна частота пам'яті 4733
кількість слотів M.2 2 штуки



Patriot Viper Steel
DDR4 2x16GB
PVS432G320C6K
Об'єм 32 GB
кількість модулів 2 (по 16GB)
частота 3200 Mhz
таймінги 16-20-20-40
дворангова пам'ять



Patriot VPN110 512GB
M.2 NVMe x4
об'єм 512 Gb
інтерфейс PCIe 3.0 x4
форм-фактор M.2 2280
швидкість читання 3100 МБ/с
швидкість запису 2300 МБ/с
TBW 400 TB



Deercool AG400
діаметр 120 мм
висота 150 мм
швидкість обертання 500–2000 обертів на хвилину
максимальний TDP 220 Wat
має 4 теплові трубки
вага 614 г

some footer text

Рисунок 2.4.17 — Приклад перегляду комплектуючих збірки користувача

Для перегляду деталей що входять до збірки необхідно буде натиснути на “список комплектуючих”.

Розділ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

Вибір СКБД, в якості системи керування базою даних для даної інформаційної системи було обрано PostgreSQL.

PostgreSQL це потужною, надійною і гнучкою системою керування базами даних з відкритим вихідним кодом, яка підходить для широкого спектра застосувань. Завдяки своїм розширеним функціональним можливостям, високій продуктивності та активній спільноті, PostgreSQL є чудовим вибором як для малих проектів, так і для великих корпоративних систем.

PostgreSQL, також відома як Postgres, є досить популярною системою керування базами даних з відкритим вихідним кодом. Дана СКБД була створена як нащадок проекту Ingres в університеті Каліфорнії в Берклі, з тих пір виросла у потужну та гнучку платформу для зберігання та управління даними. PostgreSQL розповсюджується за ліцензією PostgreSQL, що є дуже схожою до ліцензій MIT та BSD.

PostgreSQL підтримує стандартну мову для запитів до баз даних SQL, а також надає розширені можливості для роботи з даними.

Однією з переваг PostgreSQL є підтримка різних типів даних, крім стандартних типів, PostgreSQL підтримує користувацькі типи даних, масиви, JSON, XML, hstore (ключ-значення) та геометричні типи для роботи з геопросторовими даними.

Також до переваг PostgreSQL можна віднести розширюваність, дозволяє користувачам створювати власні функції, оператори та індекси, що в свою чергу дозволяє адаптувати систему під конкретні потреби.

Ще однією з переваг СКБД є підтримка ACID-транзакцій (Atomicity, Consistency, Isolation, Durability), що забезпечує високу надійність і цілісність даних.

[illegible]

На даній даталогічній моделі присутні досить багато таблиць, проте більшість з них насправді описують комп'ютерні комплектуючі різних типів та їх характеристики. Відповідно таблиці що описують конкретний тип комп'ютерної деталі містять в собі поля для опису характеристик деталі, і такі таблиці мають зв'язок типу один до одного з таблицею що узагальнено описує комп'ютерну деталь і має поля для назви, зображення, опису та деякі інші поля що спільні для всіх комп'ютерних деталей.

На рисунку 3.1.2 представлено таблицю бази даних “User” яка описує акаунт користувача.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	integer	☒	☒	
☐ password	character varying	☒	☐	
☐ last_login	timestamp with time zone	☐	☐	
☐ is_superuser	boolean	☒	☐	
☐ username	character varying	☒	☐	
☐ first_name	character varying	☒	☐	
☐ last_name	character varying	☒	☐	
☐ email	character varying	☒	☐	
☐ is_staff	boolean	☒	☐	
☐ is_active	boolean	☒	☐	
☐ date_joined	timestamp with time zone	☒	☐	

Рисунок 3.1.2 — Таблиця “User”

На рисунку 3.1.3 представлено таблицю “Like” що не має додаткових полів проте має зв'язки типу багато до багатьох до таблиці “User”, завдяки яким можна дізнатися які користувачі натиснули вподобайку чи дизлайк, а також підрахувати кількість вподобайок та дизлайків. Зв'язки типу багато до багатьох реалізовані за допомогою окремих таблиць, приклад такої таблиці наведено на рисунку 3.1.4.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	

Рисунок 3.1.3 — Таблиця “Like”

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ like_id	bigint	☒	☐	
☐ user_id	integer	☒	☐	

Рисунок 3.1.4 — Таблиця що реалізує зв'язок типу багато до багатьох між таблицями “Like” та “User”

На рисунку 3.1.5 представлено таблицю “Comments List” що описує список коментарів і має зв'язок з таблицею “Comment”.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ name	character varying	☐	☐	

Рисунок 3.1.5 — Таблиця “Comments List”

На рисунку 3.1.6 представлено таблицю “Comment” що описує коментар залишений користувачем. Дана таблиця має посилання до списку коментарів, об'єкту для підрахунку лайків та дизлайків, а також користувача що залишив цей коментар.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ text	text	☒	☐	
☐ created_time	time without time zone	☒	☐	
☐ created_date	date	☒	☐	
☐ edited	boolean	☒	☐	
☐ last_edited_datetime	timestamp with time zone	☐	☐	
☐ verified_by_moderator	boolean	☒	☐	
☐ user_id	integer	☐	☐	
☐ comment_list_id	bigint	☒	☐	
☐ likes_id	bigint	☒	☐	

Рисунок 3.1.6 — Таблиця “Comment”

На рисунку 3.1.7 представлено таблицю “Comment Reply” що описує відповідь на коментар. Дана таблиця має зв'язок з таблицею коментарів, зв'язок з таблицею “Like” для підрахунку лайків та дизлайків, а також зв'язок з таблицею користувачів.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ text	text	☒	☐	
☐ created_time	time without time zone	☒	☐	
☐ created_date	date	☒	☐	
☐ edited	boolean	☒	☐	
☐ last_edited_datetime	timestamp with time zone	☐	☐	
☐ verified_by_moderator	boolean	☒	☐	
☐ comment_id	bigint	☒	☐	
☐ user_id	integer	☐	☐	
☐ likes_id	bigint	☒	☐	

Рисунок 3.1.7 — Таблиця “Comment Reply”

На рисунку 3.1.8 представлено таблицю “Question” що описує запитання користувача. В даній таблиці також є поле типу “tsvector” що необхідне для швидкого текстового пошуку.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ question	character varying	☒	☐	
☐ description	text	☒	☐	
☐ create_date	date	☒	☐	
☐ create_time	time without time zone	☒	☐	
☐ answers_id	bigint	☒	☐	
☐ likes_id	bigint	☒	☐	
☐ user_id	integer	☐	☐	
☐ search_vector	tsvector	☐	☐	

Рисунок 3.1.8 — Таблиця “Question”

Таблиця “Question Image” зображена на рисунку 3.1.9 описує зображення що можуть бути прикріплені користувачем до запитання. Саме зображення буде завантажуватися та зберігатися на сервері у спеціально виділеній для зображень що прикріплюють користувачі папці, а в базі даних зберігається лише шлях до необхідного зображення.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ image	character varying	☒	☐	
☐ question_id	bigint	☒	☐	

Рисунок 3.1.9 — Таблиця “Question Image”

Таблиця “Manufacturer” представлена на рисунку 3.1.10 описує виробника комп’ютерних комплектуючих і має зв’язок типу багато до багатьох з таблицею “Manufacturer Type”, адже один виробник може виробляти одразу декілька видів комп’ютерних комплектуючих.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ name	character varying	☒	☐	

Рисунок 3.1.10 — Таблиця “Manufacturer”

В таблиці “Manufacturer Type” представлений на рисунку 3.1.11 описується тип виробника комп’ютерних комплектуючих, наприклад виробник процесорів чи до прикладу виробник відеокарт.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ name	character varying	☒	☐	

Рисунок 3.1.11 — Таблиця “Manufacturer Type”

На рисунку 3.1.12 зображено таблицю “Part Type” що описує тип комп’ютерної комплектуючої, наприклад: процесор, відеокарта, жорсткий диск.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ type_name	character varying	☒	☐	

Рисунок 3.1.12 — Таблиця “Part Type”

Таблиця “Computer Part” зображена на рисунку 3.1.13 містить в собі поля що спільні для всіх типів комп’ютерних комплектуючих, наприклад: назва, опис, зображення і так далі. Дана таблиця має зв’язок з таблицею списку коментарів коментарів “Comments List”, зв’язок з таблицею “Like” для підрахунку лайків та дизлайків, зв’язок з таблицею “Part Type”, а також зв’язок з таблицею “Manufacturer”. В даній таблиці також є поле типу “tsvector” що необхідне для швидкого текстового пошуку комп’ютерної комплектуючої за назвою.

☐ Name	Data type	Not NULL?	Primary key?	Comment
☐ id	bigint	☒	☒	
☐ name	character varying	☒	☐	
☐ description	text	☒	☐	
☐ msrp	numeric	☒	☐	
☐ avg_price	numeric	☒	☐	
☐ image	character varying	☐	☐	
☐ comments_id	bigint	☐	☐	
☐ likes_id	bigint	☐	☐	
☐ type_id	bigint	☒	☐	
☐ manufacturer_id	bigint	☐	☐	
☐ search_vector	tsvector	☐	☐	

Рисунок 3.1.13 — Таблиця “Computer Part”

На рисунку 3.1.14 зображено таблицю “Cpu” що має поля для опису характеристик центрального процесору. Дана таблиця пов’язана зв’язком типу один до одного з таблицею “Computer Part”.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	cores	integer	☒	☐	
☐	threads	integer	☒	☐	
☐	cores_description	character varying	☒	☐	
☐	frequency	integer	☒	☐	
☐	boost_frequency	integer	☒	☐	
☐	L3size	integer	☒	☐	
☐	L2size	integer	☒	☐	
☐	L1size	integer	☒	☐	
☐	lithography	integer	☒	☐	
☐	TDP	integer	☒	☐	
☐	computer_part_id	bigint	☒	☐	
☐	cpu_line_id	bigint	☐	☐	
☐	integrated_graphics_id	bigint	☐	☐	
☐	socket_id	bigint	☐	☐	

Рисунок 3.1.14 — Таблиця “Cpu”

На рисунку 3.1.15 зображено таблицю “Gpu” що має поля для опису характеристик відеокарти. Дана таблиця пов’язана зв’язком типу один до одного з таблицею “Computer Part”.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	frequency	integer	☒	☐	
☐	shaders	integer	☒	☐	
☐	pcie_version	integer	☒	☐	
☐	pcie_lines	integer	☒	☐	
☐	memory_bus_width	integer	☒	☐	
☐	memory_clock_speed	integer	☒	☐	
☐	memory_size	integer	☒	☐	
☐	lithography	integer	☒	☐	
☐	TDP	integer	☒	☐	
☐	slots	integer	☒	☐	
☐	width	integer	☐	☐	
☐	height	integer	☐	☐	
☐	length	integer	☐	☐	
☐	number_of_fans	integer	☒	☐	
☐	computer_part_id	bigint	☒	☐	
☐	gpu_line_id	bigint	☐	☐	
☐	memory_type_id	bigint	☐	☐	
☐	power_connection_id	bigint	☐	☐	

Рисунок 3.1.15 — Таблиця “Gpu”

На рисунку 3.1.16 зображено таблицю “Motherboard” що має поля для опису характеристик материнської плати. Дана таблиця пов’язана зв’язком типу один до одного з таблицею “Computer Part”.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	motherboard_power_connector	integer	☒	☐	
☐	cpu_power_connector	integer	☒	☐	
☐	number_of_cpu_power_connector	integer	☒	☐	
☐	number_of_ram_slots	integer	☒	☐	
☐	number_of_ram_channels	integer	☒	☐	
☐	max_ram_volume	integer	☒	☐	
☐	mim_ram_frequency	integer	☒	☐	
☐	max_ram_frequency	integer	☒	☐	
☐	igpu_support	boolean	☒	☐	
☐	vga_port	boolean	☒	☐	
☐	dvi_d_port	boolean	☒	☐	
☐	hdmi_port	boolean	☒	☐	
☐	display_port	boolean	☒	☐	
☐	ps2	boolean	☒	☐	
☐	number_of_internal_usb_v2	integer	☒	☐	
☐	number_of_internal_usb_v3	integer	☒	☐	
☐	number_of_external_usb_v2	integer	☒	☐	
☐	number_of_external_usb_v3	integer	☒	☐	
☐	number_of_sys_fans	integer	☒	☐	
☐	number_of_pcie_1x	integer	☒	☐	
☐	number_of_pcie_16x	integer	☒	☐	
☐	pcie_version	integer	☒	☐	
☐	number_of_sata3	integer	☒	☐	
☐	number_of_m2_slots	integer	☒	☐	

Рисунок 3.1.16 — Таблица “Motherboard”

На рисунку 3.1.17 зображено таблицю “Ram” що має поля для опису характеристик оперативної пам’яті. Дана таблиця як і інші таблиці що описують характеристики конкретного типу комп’ютерних деталей пов’язана зв’язком типу один до одного з таблицею “Computer Part”.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	number_of_modules	integer	☒	☐	
☐	size	integer	☒	☐	
☐	frequency	integer	☒	☐	
☐	CL	integer	☒	☐	
☐	tRCD	integer	☒	☐	
☐	tRP	integer	☒	☐	
☐	tRAS	integer	☒	☐	
☐	tRS	integer	☒	☐	
☐	dual_rank	boolean	☒	☐	
☐	computer_part_id	bigint	☒	☐	
☐	ram_type_id	bigint	☒	☐	

Рисунок 3.1.17 — Таблица “Ram”

На рисунку 3.1.18 зображено таблицю “Power Supply” що має поля для опису характеристик блоків живлення.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	motherboard_power_connector	integer	☒	☐	
☐	cpu_power_connector	integer	☒	☐	
☐	number_of_cpu_power_connector	integer	☒	☐	
☐	number_of_4pin_molex	integer	☒	☐	
☐	number_of_sata_connectors	integer	☒	☐	
☐	number_of_4pin_floppy	integer	☒	☐	
☐	width	integer	☒	☐	
☐	height	integer	☒	☐	
☐	length	integer	☒	☐	
☐	power	integer	☒	☐	
☐	cables_can_be_remove	boolean	☒	☐	
☐	computer_part_id	bigint	☒	☐	
☐	gpu_power_connection_id	bigint	☐	☐	
☐	certificate_id	bigint	☐	☐	

Рисунок 3.1.18 — Таблиця “Power Supply”

На рисунку 3.1.19 зображено таблицю “Ssd” що має поля для опису характеристик твердотілих накопичувачів.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	size	integer	☒	☐	
☐	read_speed	integer	☒	☐	
☐	write_speed	integer	☒	☐	
☐	TBW	integer	☒	☐	
☐	computer_part_id	bigint	☒	☐	
☐	tech_id	bigint	☐	☐	
☐	type_id	bigint	☐	☐	

Рисунок 3.1.19 — Таблиця “Ssd”

На рисунку 3.1.20 зображено таблицю “Hdd” що має поля для опису характеристик жорстких дисків.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	buffer	integer	☒	☐	
☐	rpm	integer	☒	☐	
☐	size	integer	☒	☐	
☐	computer_part_id	bigint	☒	☐	

Рисунок 3.1.20 — Таблиця “Hdd”

На рисунку 3.1.21 зображено таблицю “Cooler” що має поля для опису характеристик повітряного охолодження для центрального процесора.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	fan_speed	integer	☒	☐	
☐	fan_diameter	integer	☒	☐	
☐	numbers_of_fans	integer	☒	☐	
☐	numbers_of_heat_pipes	integer	☒	☐	
☐	mass	double precision	☒	☐	
☐	width	double precision	☒	☐	
☐	height	double precision	☒	☐	
☐	length	double precision	☒	☐	
☐	TDP	integer	☒	☐	
☐	computer_part_id	bigint	☒	☐	

Рисунок 3.1.21 — Таблица “Cooler”

На рисунку 3.1.22 зображено таблицю “Computer Case” що має поля для опису характеристик комп’ютерних корпусів.

☐	Name	Data type	Not NULL?	Primary key?	Comment
☐	id	bigint	☒	☒	
☐	number_of_usb_v2	integer	☒	☐	
☐	number_of_usb_v3	integer	☒	☐	
☐	number_of_usb_type_c	integer	☒	☐	
☐	microphone_and_headphone_port	boolean	☒	☐	
☐	width	integer	☒	☐	
☐	height	integer	☒	☐	
☐	length	integer	☒	☐	
☐	number_of_fans_installed	integer	☒	☐	
☐	number_of_front_fans	integer	☒	☐	
☐	number_of_back_fans	integer	☒	☐	
☐	number_of_top_fans	integer	☒	☐	
☐	max_cooler_height	integer	☒	☐	
☐	max_power_supply_length	integer	☒	☐	
☐	max_gpu_length	integer	☒	☐	
☐	computer_part_id	bigint	☒	☐	

Рисунок 3.1.22 — Таблица “Computer Case”

3.2 Технічне забезпечення

Для наочної ілюстрації технічного забезпечення було створено діаграму розгортання яку представлено на рисунку 3.2.1. На даній діаграмі зображено головні компоненти системи, а саме сервер з самою інформаційною системою інтернет-форуму та базою даних, а також комп'ютер користувача з інтернет браузером, через який користувач взаємодіє з інтернет-форумом, в якому відображаються підготовлені html сторінки та виконується JavaScript код клієнтської частини.

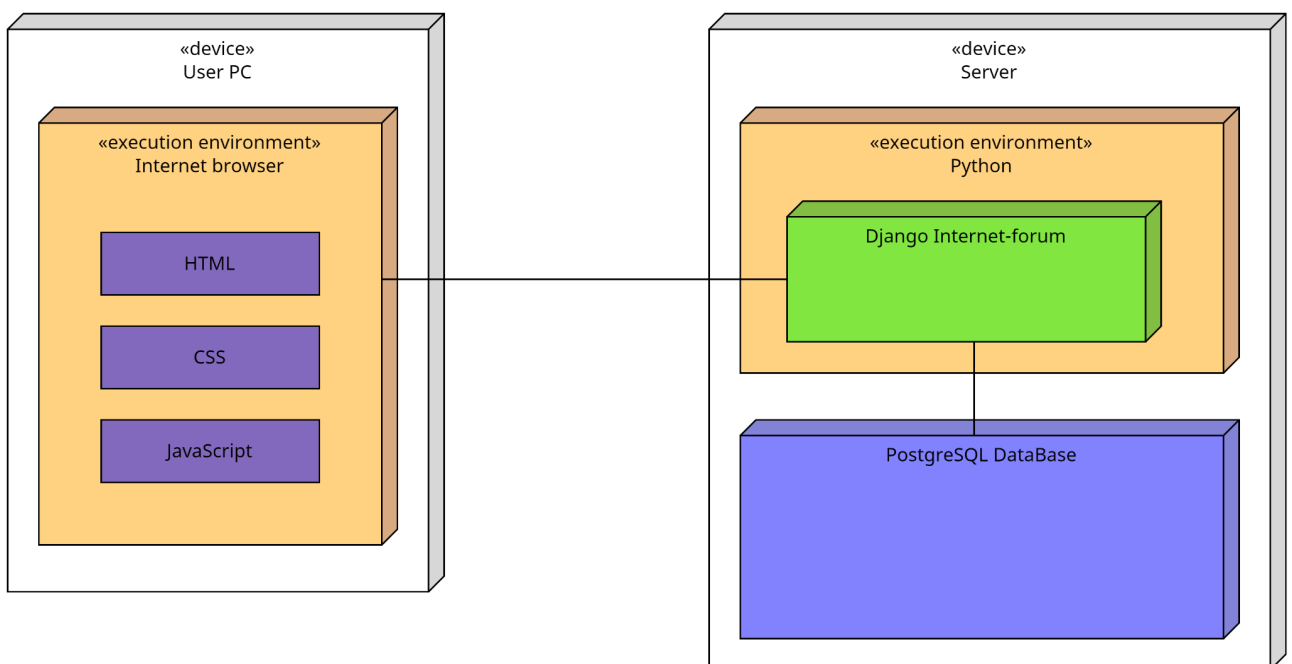


Рисунок 3.2.1 — діаграма розгортання

В якості операційної системи що встановлена на сервері може виступати як ОС Windows так і Linux, адже Python Django як і всі інші залежності включно з PostgreSQL доступні та можуть виконуватися на обох операційних системах.

При розгортанні на сервері необхідно попередньо встановити Python, всі необхідні залежності та бібліотеки та налаштувати базу даних PostgreSQL. Для встановлення всіх необхідних Python бібліотек необхідно буде перейти в кореневий каталог проекту та виконати команду “`pip install -r requirements.txt`”. Для налаштування бази даних необхідно встановити PostgreSQL, створити нову базу даних та користувача, після чого необхідно

встановити відповідну інформацію про користувача та назву бази даних в файлі “environment.env” що знаходиться в корені проекту. Також необхідно виконати команду “python manage.py migrate” для застосування міграцій до бази даних, це створить всі необхідні таблиці та зв’язки в базі даних. Після чого необхідно створити супер користувача за допомогою команди “python manage.py createsuperuser” і ввести необхідні дані логін та пароль для суперкористувача. Після чого необхідно заповнити базу даних даними про комп’ютерні комплектуючі, а також заповнити таблицю “User Status” що можна зробити через адмін панель.

3.3 Програмне забезпечення

В якості операційної системи при розробці використовувалась ОС Windows 11.

Для проектування системи та створення діаграм використовувався Enterprise Architect. Enterprise Architect (EA) – це багатофункціональне програмне забезпечення що використовується для аналізу, проектування, моделювання та розробки програмних систем, бізнес-процесів та архітектури підприємства. Enterprise Architect підтримує всі види діаграм UML, що дозволяє створювати детальні моделі систем. Також Enterprise Architect має підтримку моделювання бізнес-процесів за допомогою BPMN (Business Process Model and Notation).

В якості мови програмування було обрано Python. Python — це високорівнева мова програмування загального призначення, яка була вперше випущена в 1991 році. Python відомий своєю простотою та читабельністю, що робить її популярною серед початківців і професійних розробників. До переваг мови Python можна віднести: простий і зрозумілий синтаксис що робить код легким для читання і написання, Python має потужну стандартну бібліотеку яка включає модулі та пакети для самих різноманітних задач, Python є кросплатформною мовою тому код написаний на Python може працювати на різних операційних системах, Python є інтерпретованою мовою що значно спрощує написання коду, в Python використовується зручний менеджер пакетів `pip`, інтерпретатор має відкритий програмний код. До недоліків Python можна віднести відносну повільність виконання коду порівняно з іншими мовами програмування.

В якості середовища розробки було обрано IDE PyCharm Community, це інтегроване середовище розробки з досить широким функціоналом. PyCharm написаний на Python та Java і вперше випущений в липні 2010 року. Серед переваг PyCharm можна виділити: підказки щодо правопису на основі стандарту PEP8, контекстні підказки, є інструменти для швидкої взаємодії з кодом, присутня підтримка контролю версій, відкритий вихідний код,

статичний аналіз коду (дана функція підсвічує помилки). Починаючи з Python3 в мові з'явився такий елемент як підказки типу (type hints) за допомогою якого інтегроване середовище розробки може розпізнавати помилки з невідповідностями типів даних як в мовах з статичною типізацією, проте щоб ця функція працювала необхідно використовувати підказки типу в коді.

Для створення даної інформаційної системи було обрано фреймворк Django. Django - це відкритий високорівневий веб-фреймворк для розробки веб-додатків на мові програмування Python. Django надає ряд готових інструментів і підходів для вирішення рутинних завдань, дозволяючи розробникам фокусуватися на бізнес-логіці своєї програми.

До основних переваг Django можна віднести використання архітектурного шаблону MVT (Model View Template), що є варіацією популярної моделі MVC. Це дозволяє ефективно організувати код, розділяючи його на компоненти моделі (бізнес-логіка), виду (представлення) та шаблонів (представлення даних).

Django має вбудовану адміністративну панель, яка дозволяє легко управляти базою даних веб-додатку. Це значно полегшує розробку та адміністрування.

Django має вбудовану ORM, яка дозволяє взаємодіяти з базою даних, використовуючи об'єктно-орієнтовані моделі, замість прямих SQL-запитів. Це полегшує роботу з базою даних та забезпечує переносимість коду між різними СУБД.

Django автоматично генерує форми для створення, оновлення і видалення об'єктів бази даних, що заощаджує час розробки. Крім того в фреймворк вбудовано засоби безпеки, такі як CSRF.

Вбудована підтримка для роботи з URL адресами. Django має потужну систему маршрутизації, яка дозволяє легко визначати URL-шляхи для веб-сторінок.

Django активно використовується спільнотою розробників, і як наслідок має багато розширень (додаткових бібліотек), а також має гарну документацію, що полегшує вивчення та використання фреймворку.

Для реалізації інтерфейсу користувача при розробці системи було використано систему шаблонів яку за замовчуванням підтримує Django.

Шаблонізатор в Django — це потужний інструмент, що дозволяє розділити логіку додатку та представлення (інтерфейс користувача). Він допомагає легко створювати динамічні HTML-сторінки, використовуючи шаблони з вбудованими тегами та фільтрами.

Шаблони — це текстові файли, які можуть містити статичний HTML, а також теги та змінні Django. Файли шаблонів зазвичай зберігають в директоріях з назвою “templates” додатків Django.

Шаблони можуть містити змінні, які замінюються значеннями під час рендерингу шаблону. Змінні в шаблонах записуються в подвійні фігурні дужки.

Також шаблони підтримують теги які використовуються для логіки в шаблонах, такої як цикли та умовні вирази.

Ще однією корисною функцією шаблонів є фільтри що дозволяють модифікувати значення змінних в шаблоні.

Django дозволяє наслідувати шаблони для уникнення дублювання коду, для цього створюються базові шаблони з блоками, які можуть бути заповнені або перевизначені в дочірніх шаблонах. Приклад базового шаблону можна переглянути в додатку А, а приклад шаблону що наслідує базовий шаблон можна переглянути в додатку Б.

Ще однією надзвичайно корисною функцією шаблонів в Django є включення шаблонів. Шаблони можна розбивати на менші частини та включати їх в інші шаблони за допомогою тегу “{% include %}”. В додатку В можна переглянути приклад коду шаблону списку коментарів який включає в себе інший фрагмент, а саме коментар, код якого можна переглянути в додатку Г. При цьому сам фрагмент коментар включає в себе інші фрагменти

такі як фрагмент з вподобайками та дизлайками код якого представлено в додатку Г, а також фрагменти з відповіддю на коментар код якого можна переглянути в додатку Д. Сам шаблон списку коментарів також використовується для включення в багатьох інших шаблонах, наприклад на головній сторінці, або на сторінці запитання користувача код якої представлено в додатку Є.

Щоб передати дані шаблону для рендерингу в Django використовується змінна з назвою “context” що має тип словнику, тобто ключ значення, в цій змінній визначають всі ті дані які необхідні для рендерингу шаблону.

Також при розробці фронтальної частини застосунку було використано таку технологію як jQuery AJAX, це технологія яка використовується для здійснення асинхронних HTTP-запитів за допомогою бібліотеки jQuery. AJAX (Asynchronous JavaScript and XML) дозволяє завантажувати дані з сервера без перезавантаження сторінки, що значно покращує взаємодію користувача з веб-додатком. До прикладу коли користувач натискає вподобайку або дизлайк, це не призводить до перезавантаження сторінки що є дуже зручним. Приклад використання відповідної технології можна переглянути в коді фрагменту з вподобайками та дизлайками який представлено в додатку Г. Для підключення відповідної JavaScript бібліотеки jQuery в базовому шаблоні є відповідний скрипт, що можна побачити в додатку А.

Для візуального оформлення сторінок форуму використовувалися власноруч написані стилі CSS які можна переглянути в додатку З. Також при створенні сторінок форуму та для деяких його окремих частин було використано фреймворк Bootstrap. Bootstrap — це популярний відкритий фреймворк для розробки адаптивних веб-додатків та сайтів. Він містить набір інструментів для роботи з HTML, CSS та JavaScript, що спрощує процес створення сучасних веб-інтерфейсів.

Як вже зазначалося в розділі 3.1 в якості СКБД було обрано PostgreSQL який є потужною, надійною і гнучкою системою керування базами даних та має ряд переваг що були описані у відповідному розділі.

Слід також зазначити так як система розроблялась з використанням фреймворку Django який має власну ORM систему, при розробці не було необхідності власноруч писати SQL запити до бази даних що значно спростило процес розробки системи. Так для опису бази даних в Django необхідно було описати відповідні класи моделей що успадковуються від класу “Model”, також за допомогою спеціальних полів типу “OneToOneField”, “ForeignKey” та “ManyToManyField” описуються зв’язки між таблицями. Також в системі для деяких таблиць передбачається можливість повнотекстового пошуку, для реалізації якого були створені поля типу “tsvector”, які також передбачені в Django. За необхідності в класі моделі можна вказати мета клас в якому описати які індекси слід створити і для яких полів, що є зручним для оптимізації роботи бази даних. Переглянути приклади коду моделей можна в додатках II, I та ІІ.

3.4 Результати реалізації інформаційної системи

В даному розділі продемонстровані результати програмної реалізації інформаційної системи інтернет-форуму для обговорення комп'ютерних збірок. Переглянути програмний код “view” класів можна в додатках Й, К, Л.

На рисунку 3.4.1 представлено головну сторінку інтернет форуму на якій є привітальний текст, посилання на два інші розділи форуму, а також можливість залишити коментар.

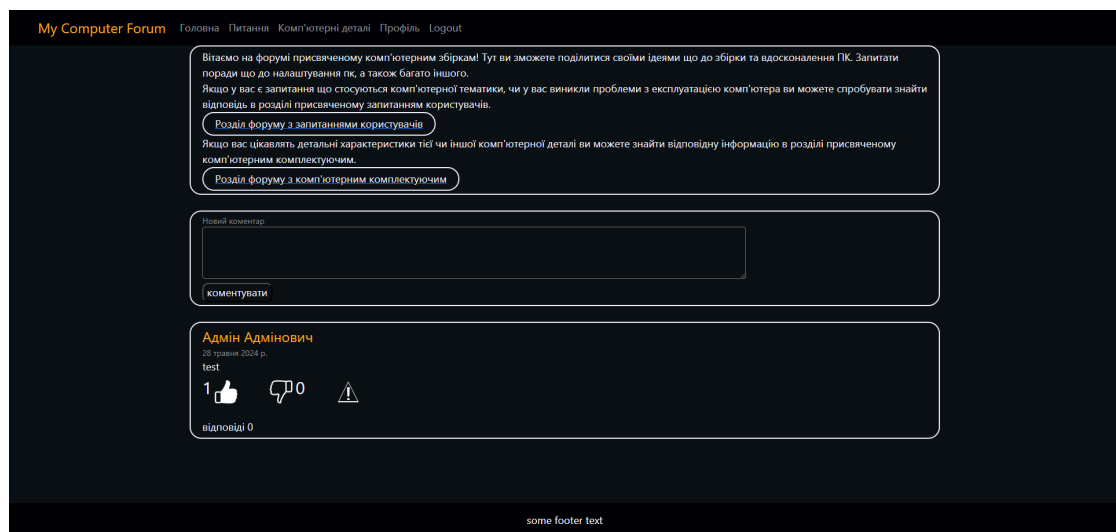


Рисунок 3.4.1 — Головна сторінка форуму

На рисунку 3.4.2 представлено сторінку авторизації.

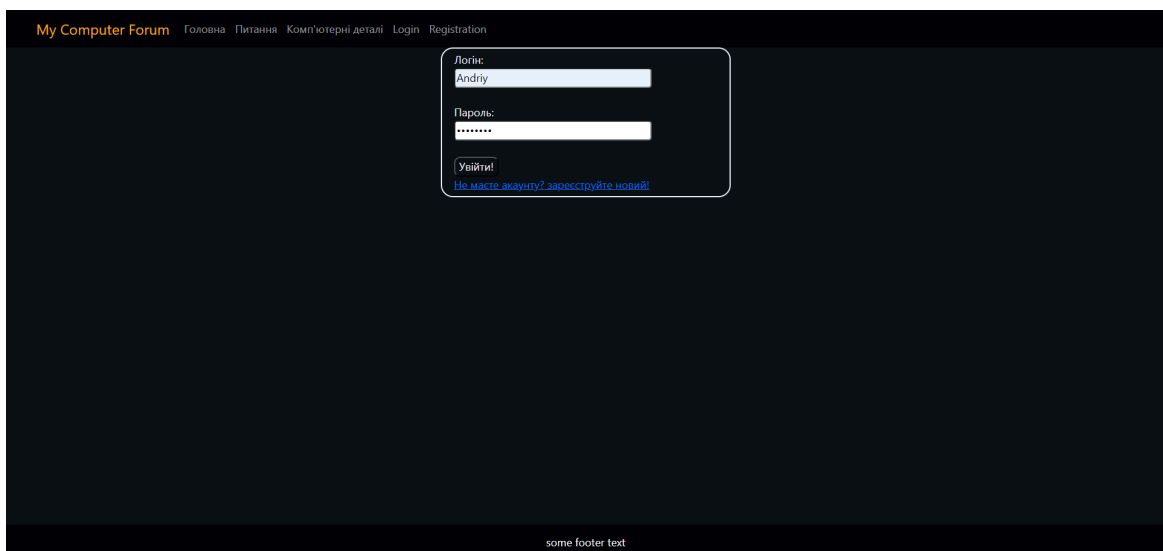


Рисунок 3.4.2 — Сторінка входу

В разі якщо у користувача ще немає акаунту передбачено посилання на сторінку реєстрації представлену на рисунку 3.4.3.

Рисунок 3.4.3 — Сторінка реєстрації

На рисунку 3.4.4 представлено сторінку з запитаннями користувачів.

Рисунок 3.4.4 — Сторінка з запитаннями користувачів

На даній сторінці передбачена можливість пошуку запитання за назвою, а також пагінація списку запитань що продемонстровано на рисунку 3.4.5.

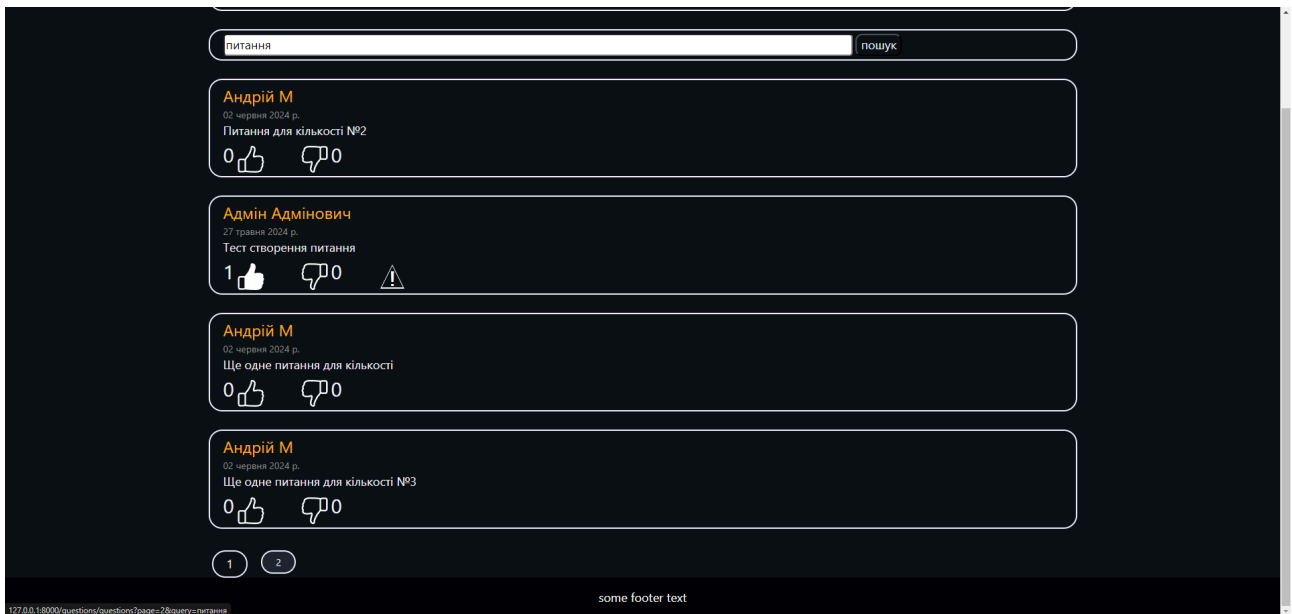


Рисунок 3.4.5 — Пошук за назвою

Приклад запитання користувача представлено на рисунку 3.4.6.

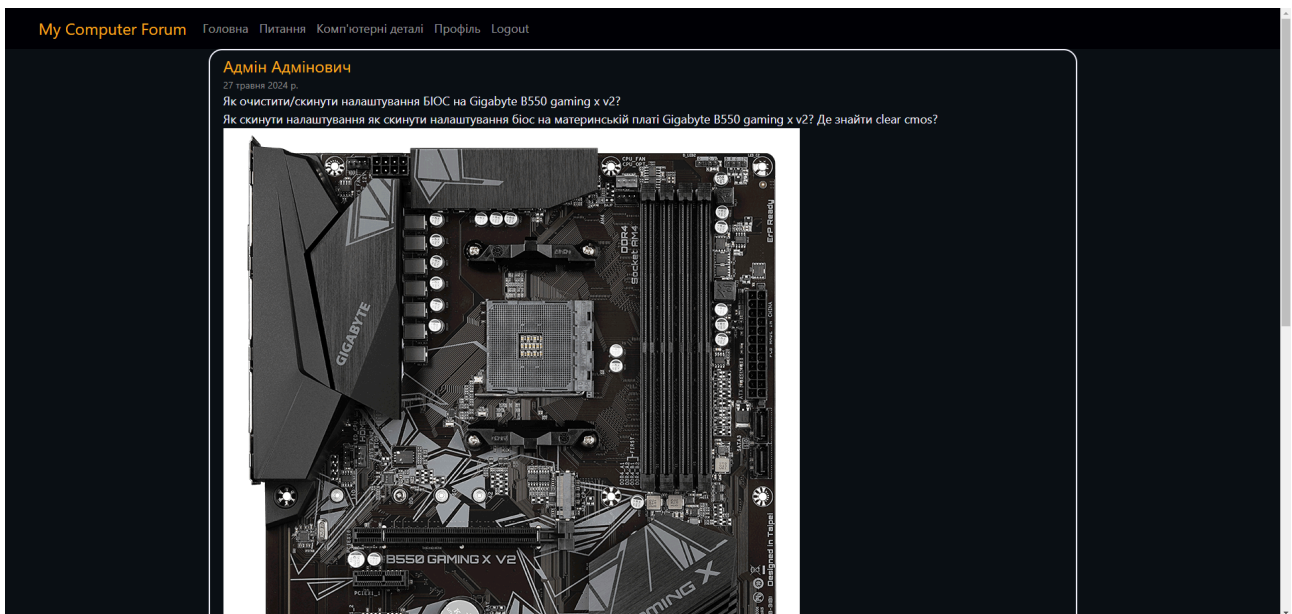


Рисунок 3.4.6 — Приклад сторінки з запитанням користувача

Також інші користувачі можуть пропонувати власні варіанти відповіді на запитання інших користувачів і писати відповіді на коментарі, що продемонстровано на рисунку 3.4.7.

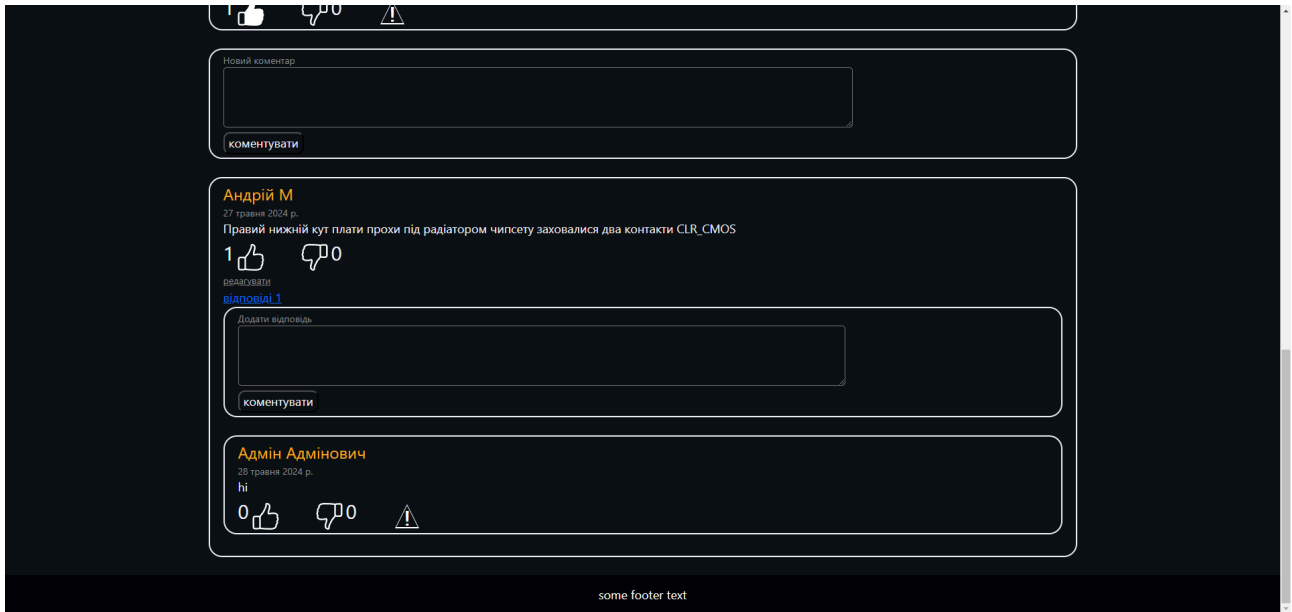


Рисунок 3.4.7 — Приклад відповіді на запитання

Також передбачено можливість редагування власних коментарів, що продемонстровано на рисунку 3.4.8.

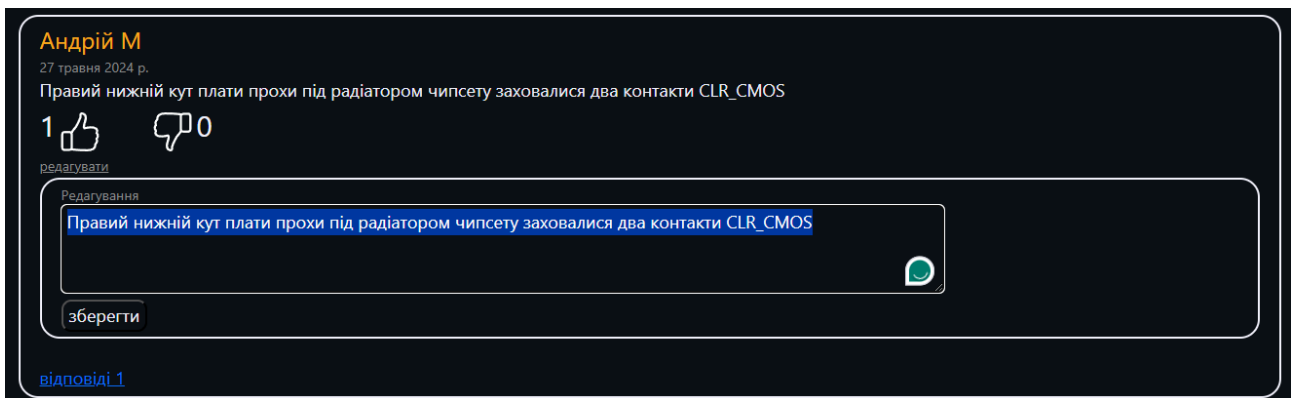


Рисунок 3.4.8 — Можливість редагувати власні коментарі

В разі якщо запитання ще не було створено на форумі раніше, зареєстровані користувачі можуть створювати нові питання. Форма створення нового запитання продемонстрована на рисунку 3.4.9, також за необхідності користувач при створенні запитання може прикріпити зображення.

Рисунок 3.4.9 — Сторінка створення нового запитання

На рисунку 3.4.10 представлено головну сторінку розділу з комп'ютерними комплектуючими, на якій є посилання на списки комп'ютерних комплектуючих різних типів.

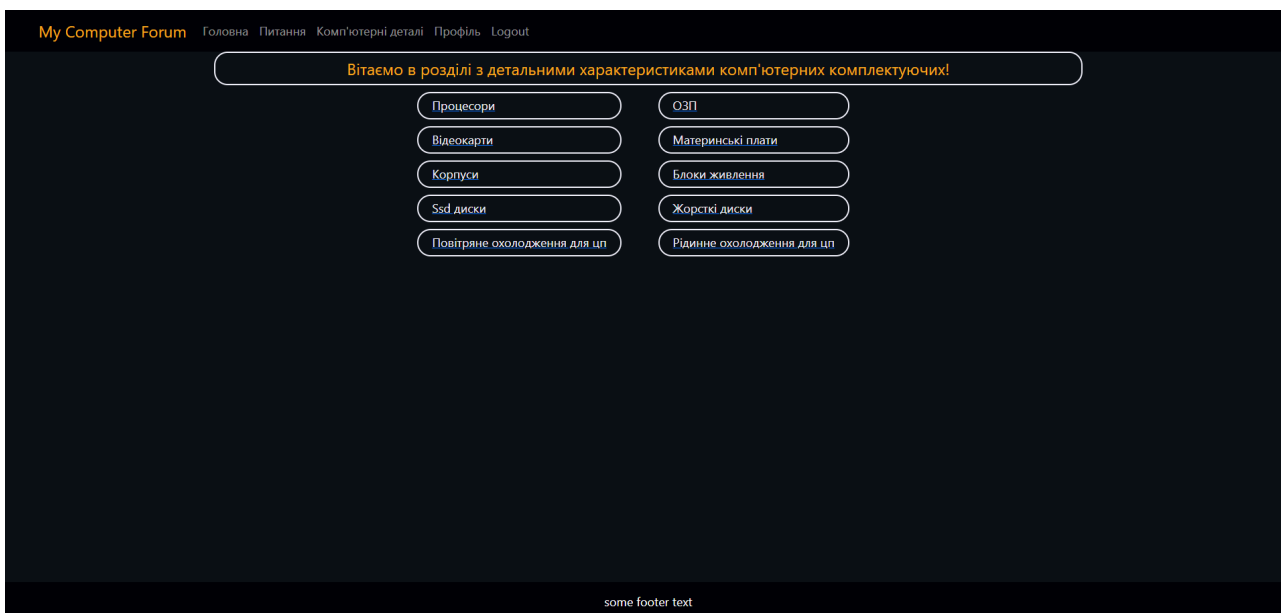


Рисунок 3.4.10 — Головна сторінка розділу з комп'ютерними комплектуючими

На рисунку 3.4.11 зображено приклад сторінки з списком процесорів, на даній сторінці також передбачений пошук за назвою процесора, а також пагінація списку.

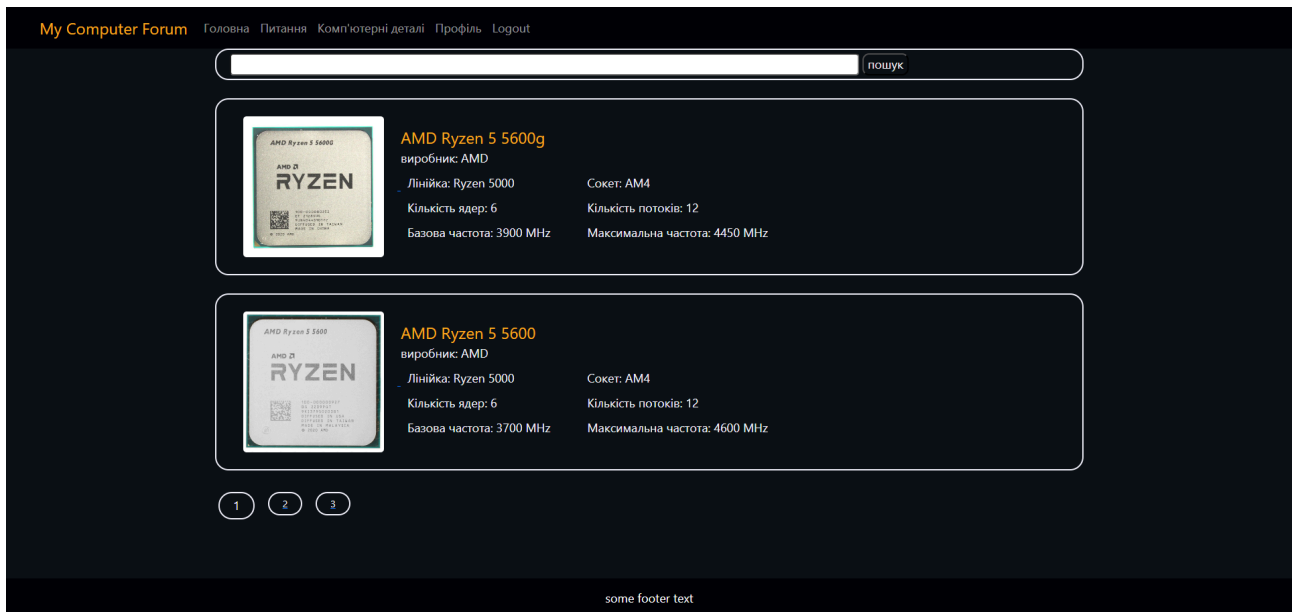


Рисунок 3.4.11 — Сторінка з списком процесорів

На рисунку 3.4.12 зображено приклад сторінки з детальними характеристиками процесора та його описом.

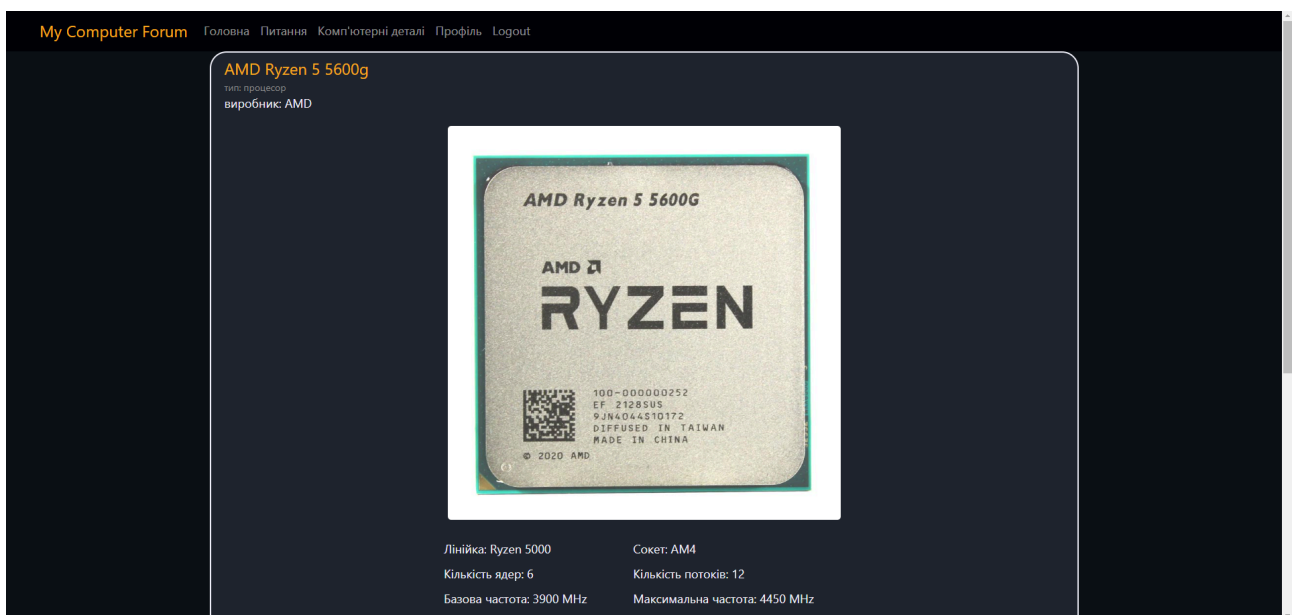


Рисунок 3.4.12 — Приклад сторінки з характеристиками процесора

Також на даній сторінці передбачена можливість залишати коментарі, що продемонстровано на рисунку 3.4.13.

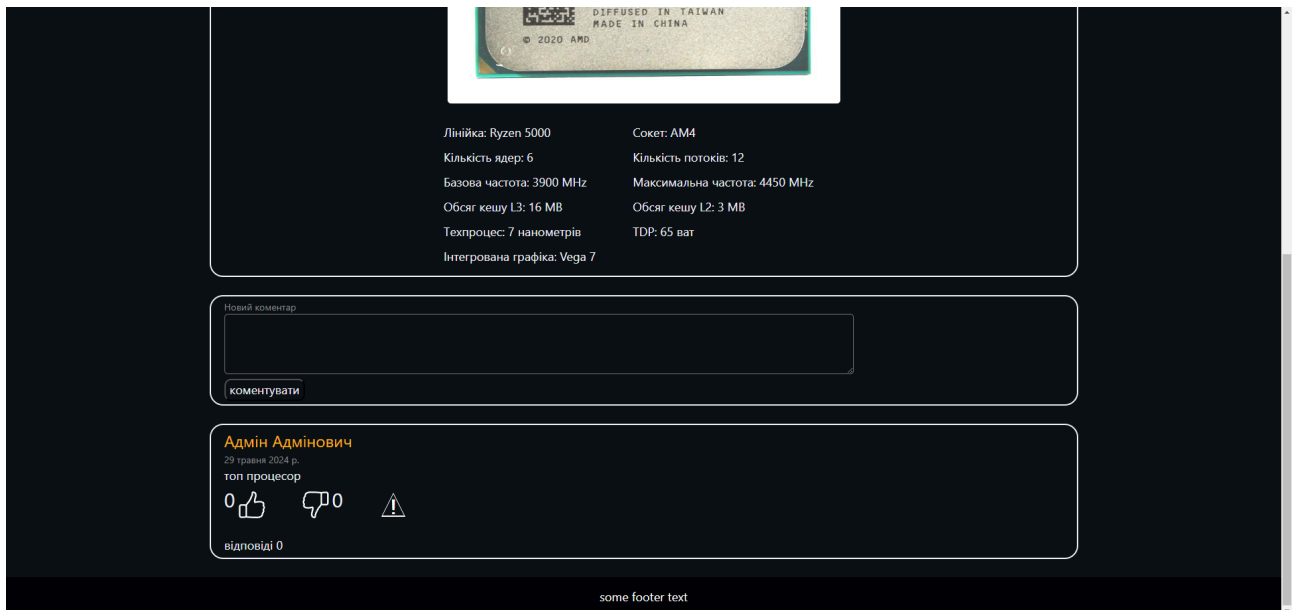


Рисунок 3.4.13 — Можливість обговорення комп'ютерних деталей

На рисунку 3.4.14 зображено приклад сторінки комп'ютерної комплектуючої іншого виду, а саме відеокарти з її детальними характеристиками процесора та його описом.

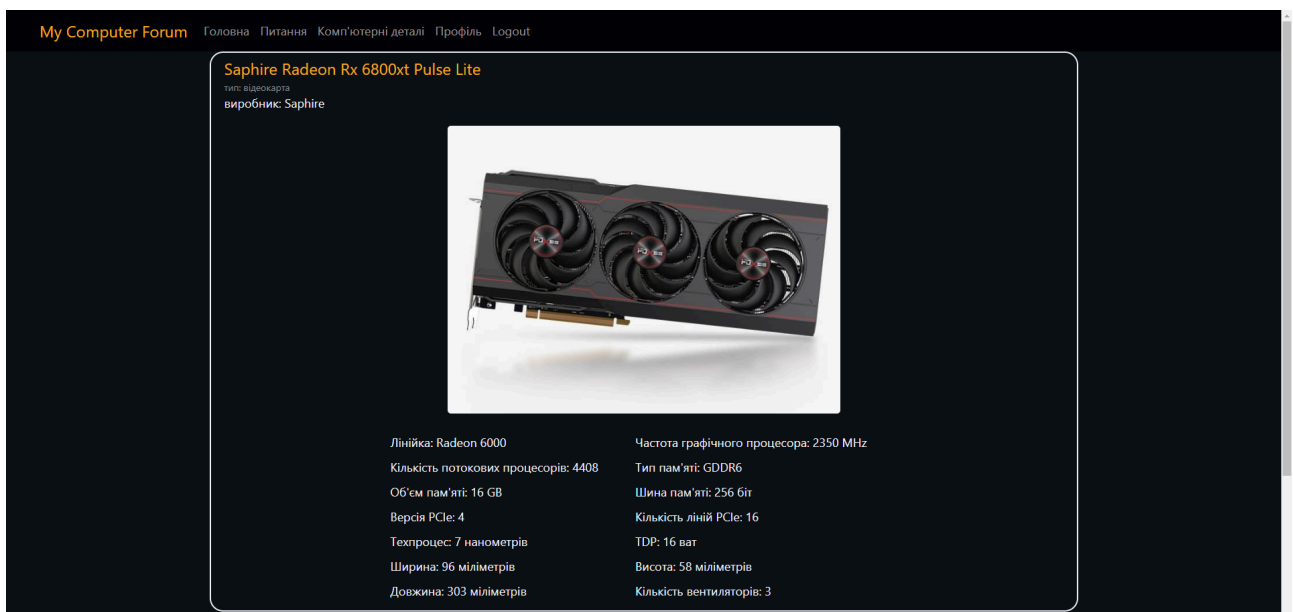


Рисунок 3.4.14 — Приклад сторінки з характеристиками відеокарти

Також якщо увійти в обліковий запис адміністратора сайту, то в навігаційні панелі з'явиться новий пункт для переходу на сторінку адмін панелі що продемонстрована на рисунку 3.4.15.

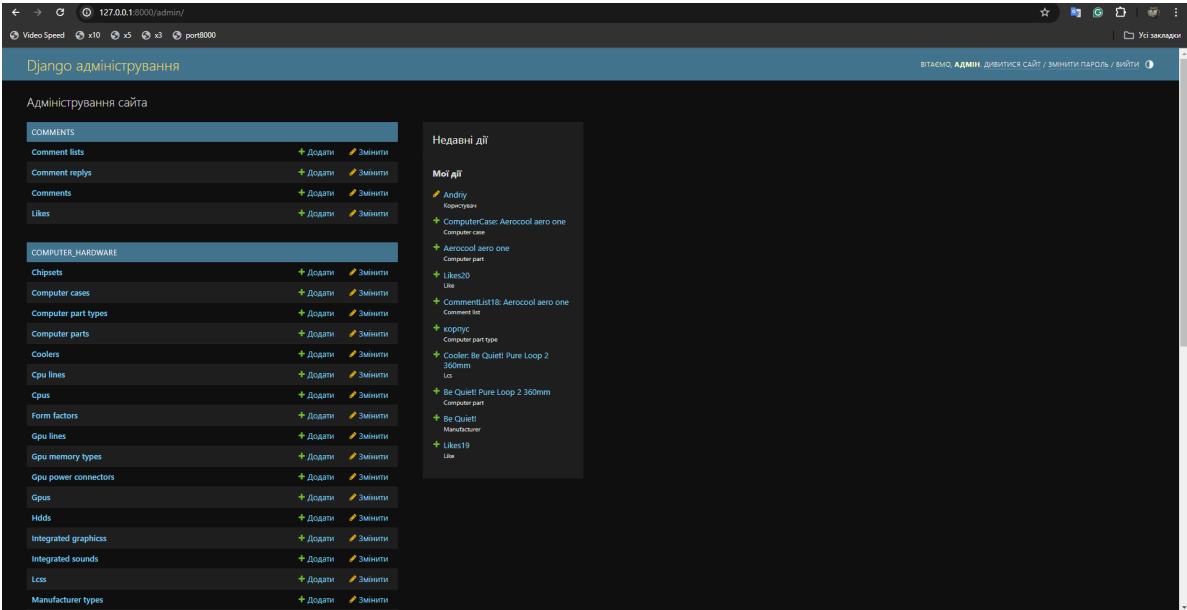


Рисунок 3.4.15 — Головна сторінка адмін панелі

На рисунку 3.4.16 продемонстровано приклад перегляду через адмін панель сторінки зі списком записів в таблиці “CPU”.

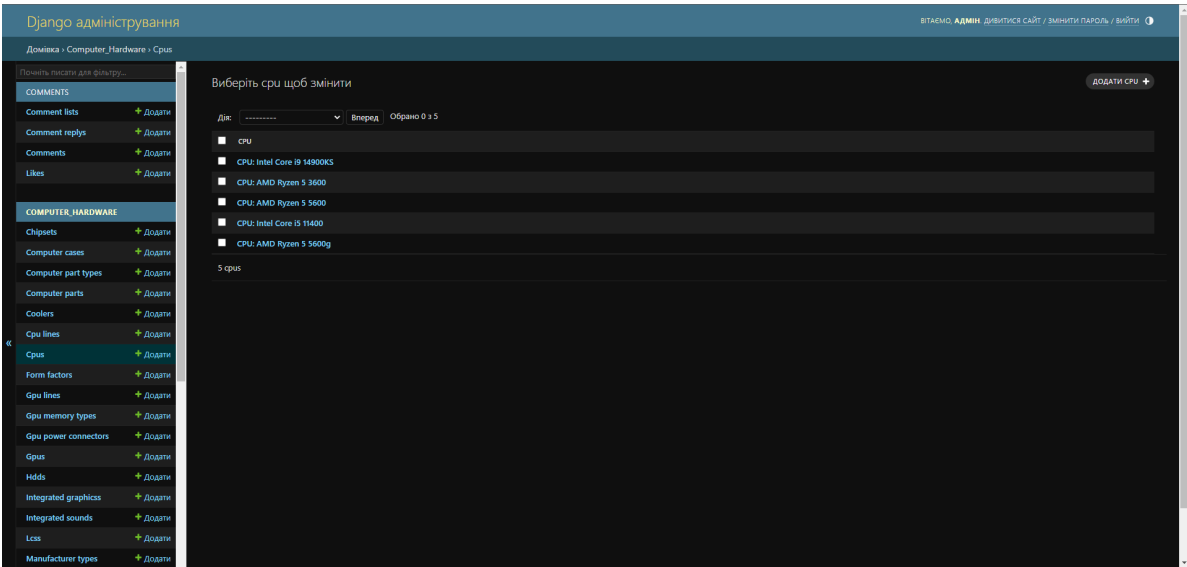


Рисунок 3.4.16 — Адмін панель список записів таблиці “CPU”

На рисунку 3.4.17 продемонстровано сторінку адмін панелі для перегляду та редагування або видалення запису з таблиці “CPU”.

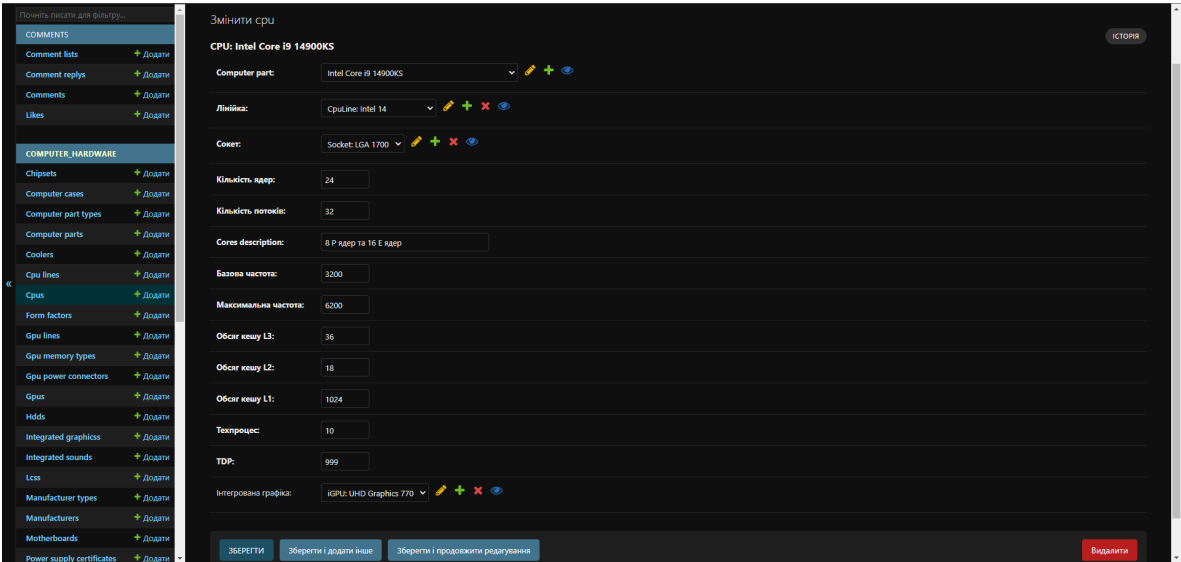


Рисунок 3.4.17 — сторінка адмін панелі для перегляду запису

На рисунку 3.4.18 продемонстровано перегляд іншого запису в таблиці “Computer part” через адмін панель.

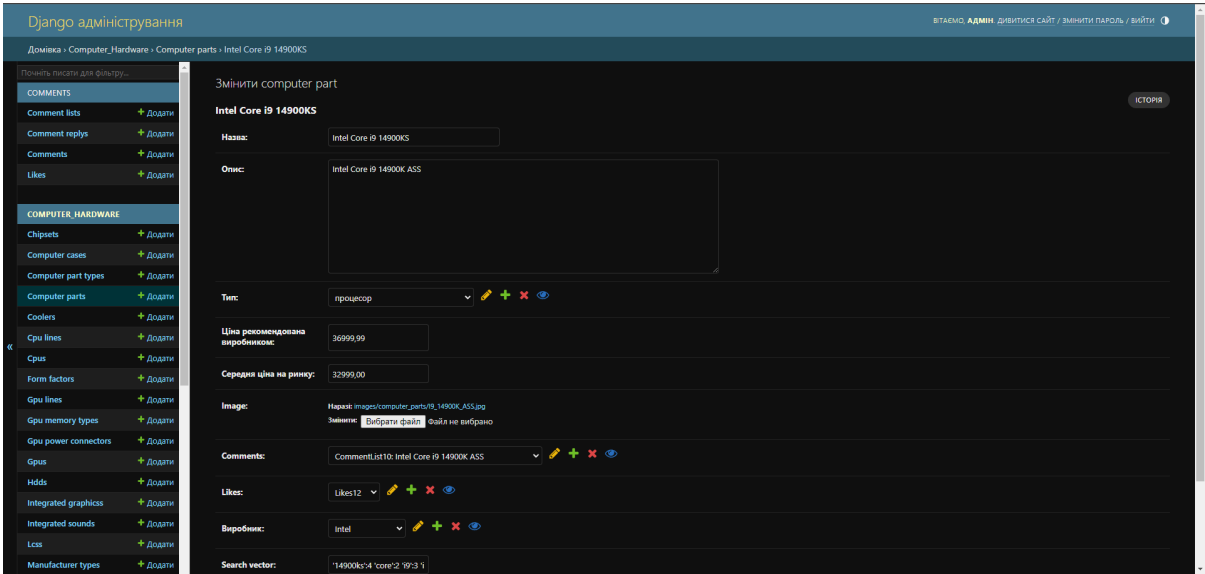


Рисунок 3.4.18 — Перегляд запису в таблиці “Computer part” через адмін панель.

ВИСНОВКИ

Було спроектовано та розроблено інформаційну систему інтернет-форуму для обговорення комп'ютерних збірок. Описано основні проектні рішення, розроблено клікабельний прототип інтерфейсу користувача, а також за допомогою фреймворку Django для мови Python реалізовано два розділи інтернет форуму і систему авторизації для користувачів.

В першому розділі було описано предметну галузь, обґрунтовано доцільність розробки системи. Наведено порівняння існуючих систем з їх перевагами та недоліками.

В другому розділі були описані вимоги до системи, зокрема бізнес-вимоги, функціональні вимоги та нефункціональні вимоги. Було сформовано характеристику задачі, наведено інформаційну модель задачі, описано вхідні та вихідні повідомлення, розроблено алгоритм розв'язання задачі. Також в другому розділі було проведено моделювання інформаційної системи, наведено діаграму прецедентів та діаграми послідовності, розроблено діаграми визначення блоків та діаграму класів, а також проведено трасування вимог.

В третьому розділі було описано результати проектування та реалізації інформаційної системи, зокрема обґрунтовано вибір СКБД та наведено даталогічну модель створеної бази даних. Було описано технічне забезпечення необхідне для запуску та функціонування інформаційної системи, коротко описано основні аспекти які необхідно враховувати при розгортанні системи. Було описано програмне забезпечення використане для проектування та розробки системи, описано використані технології з посиланнями на додатки з кодом де вони використовуються. Також в третьому розділі були наведені скріншоти створеної системи, описано розроблений функціонал.

Даний проект має перспективи подальшого розвитку, насамперед необхідно реалізувати третій розділ сайту з можливістю для користувачів

створювати власні варіанти комп'ютерних збірок. Також в розділі з запитаннями користувачів в майбутньому можливо розробити та впровадити систему тегів для покращення зручності пошуку відповіді на запитання. Також з можливих покращень можна виділити можливість переходу від використання шаблонів до розробки повноцінного фронтенду, що дозволить значно підвищити динамічність і інтерактивність системи, що забезпечить користувачам більш зручний та функціональний інтерфейс.

До переваг системи можна віднести: використання сучасних та гнучких технологій, відповідність поставленим вимогам, можливість подальшого вдосконалення та розширення.

Серед недоліків системи варто відзначити простий та примітивний фронтенд який потребує оновлення сторінки для відображення змін.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tomshardware [Електронний ресурс] – Режим доступу до ресурсу: <https://forums.tomshardware.com/>.
2. Linustechtips [Електронний ресурс] – Режим доступу до ресурсу: <https://linustechtips.com/>.
3. Techpowerup [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techpowerup.com/>.
4. Telemart [Електронний ресурс] – Режим доступу до ресурсу: <https://telemart.ua/ua/>.
5. Unified Modeling Language, v2.5.1 [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: <https://www.omg.org/spec/UML/2.5.1/PDF>.
6. Enterprise Architect User Guide [Електронний ресурс] – Режим доступу до ресурсу: https://sparxsystems.com/enterprise_architect_user_guide/16.1/welcome/index.html.
7. Django documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/5.0/>.
8. PostgreSQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/>.
9. PostgreSQL Text Search Types [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/datatype-textsearch.html>.
10. Get started with Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>.
11. jQuery Ajax [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.jquery.com/ajax/>.
12. Розроблений прототип інтерфейсу користувача [Електронний ресурс] – Режим доступу до ресурсу: <https://www.figma.com/file/e3rpd0RdrnUzXre3FZRNzP/MyComputerForumMobilePrototype?type=design&node-id=30%3A108&mode=design&t=ll3caUMKN71QYhyJ-1>.

ДОДАТКИ

ДОДАТОК А

Код базового шаблону

```
<!DOCTYPE html>
{% load static %}
<html lang="ua">
<head>
  <meta charset="UTF-8">
  <title>{% block title %}My computer forum{% endblock %}</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
    rel="stylesheet"

    integrity="sha384-T3c6CoI6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
    crossorigin="anonymous">
  <script defer src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"

    integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
    crossorigin="anonymous"></script>
  <script src="https://code.jquery.com/jquery-3.7.1.js"
    integrity="sha256-eKhai8LEQwp4NKxN+CfCh+3qOVUtJn3QNZ0TciWLP4="
    crossorigin="anonymous"></script>
  <link rel="stylesheet" href="{% static 'css/my_styles.css' %}">
</head>
<body class="text-white d-flex flex-column min-vh-100">

  <nav class="navbar navbar-expand-lg navbar-dark my-navbar">
    <a class="navbar-brand text-title" style="margin-left: 50px;" href="{% url 'index' %}">My Computer
    Forum</a>
    <button class="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarNav"
      aria-controls="navbarNav"
      aria-expanded="false"
      aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'index' %}">Головна</a>
        </li>
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'questions' %}">Питання</a>
        </li>
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'computer_hardware' %}">Комп'ютерні деталі</a>
        </li>
        {% if user.is_authenticated %}
        <li class="nav-item active">
          <a class="nav-link" href="#">Профіль</a>
        </li>
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'logout' %}">Logout</a>
        </li>
        {% if user.is_staff %}
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'admin:index' %}">Admin panel</a>
        </li>
        {% endif %}
        {% else %}
        <li class="nav-item active">

```

```
        <a class="nav-link" href="{% url 'login' %}">Login</a>
    </li>
    <li class="nav-item active">
        <a class="nav-link" href="{% url 'registration' %}">Registration</a>
    </li>
    {% endif %}
</ul>
</div>
</nav>

<div class="container">{% block content %}{% endblock %}</div>

<footer class="mt-auto text-center text-white my-footer">
    <div class="text-center p-3">
        some footer text
    </div>
</footer>
</body>
</html>
```

ДОДАТОК Б

Код шаблону “main_page.html”

```
{% extends "common/base.html" %}

{% block title %}My Computer Forum{% endblock title %}

{% block content %}
<div>
  <div class="my-container">
    <div>Вітаємо на форумі присвяченому комп'ютерним збіркам! Тут ви зможете поділитися своїми
    ідеями що до збірки та вдосконалення ПК. Запитати поради що до налаштування пк, а також багато
    іншого.</div>
    <div>Якщо у вас є запитання що стосуються комп'ютерної тематики, чи у вас виникли проблеми з
    експлуатацією комп'ютера ви можете спробувати знайти відповідь в розділі присвяченому запитанням
    користувачів.</div>
    <a href="{% url 'questions' %}"><div class="my-container" style="width: fit-content;">Розділ форуму з
    запитаннями користувачів</div></a>
    <div>Якщо вас цікавлять детальні характеристики тієї чи іншої комп'ютерної деталі ви можете знайти
    відповідну інформацію в розділі присвяченому комп'ютерним комплектуючим.</div>
    <a href="{% url 'computer_hardware' %}"><div class="my-container" style="width: fit-content;">Розділ
    форуму з комп'ютерним комплектуючим</div></a>
  </div>
  <br>
  <div>
    {% include "comments/comments_list_snippet.html" with comments_list=comments_list %}
  </div>
</div>
{% endblock content %}
```

ДОДАТОК В

Код шаблону “comments_list_snippet.html”

```
{% load comments_tags %}
<div>
  <div class="my-container">
    <div class="text-additional">Новий коментар</div>
    <form action="{% url 'add_comment' comments_list.pk %}" method="POST">
      {% csrf_token %}
      <textarea class="my-input-text" name="new_comment_text" maxlength="512"></textarea>
      <input name="reload_url" type="hidden" value="{{ request.resolver_match|my_url }}">
      <div><button class="my-btn" type="submit">коментувати</button></div>
    </form>
  </div><br>
  <div>
    {% for comment in comments_list.comments.all %}
    {% include "comments/comment_snippet.html" with comment=comment %}<br>
    {% endfor %}
  </div>
</div>
```

ДОДАТОК Г

Код шаблону “comment_snippet.html”

```
{% load comments_tags %}
<div class="my-container">
  <div class="text-title">
    {% if comment.user %}
      {{ comment.user.first_name }} {{ comment.user.last_name }}
    {% else %}
      ghost
    {% endif %}
  </div>
  <div class="text-additional">{{ comment.created_date }}</div>
  <div class="text-main">{{ comment.text }}</div>

  {% if comment.edited %}
    <div class="text-additional">ЗМІНЕНО</div>
  {% endif %}

  {% if comment.user == user %}
    <div>{% include "comments/like_view_snippet.html" with like_object=comment.likes %}</div>
    <a class="text-additional" data-bs-toggle="collapse"
data-bs-target="#collapse-comment-edit-{{ comment.pk }}">
      редагувати
    </a>
    <div class="collapse" id="collapse-comment-edit-{{ comment.pk }}">
      <div class="my-container">
        <div class="text-additional">Редагування</div>
        <form action="{% url 'edit_comment' comment.pk %}" method="POST">
          {% csrf_token %}
          <textarea class="my-input-text" name="comment_new_text"
maxlength="512">{{ comment.text }}</textarea>
          <input name="reload_url" type="hidden" value="{{ request.resolver_match|my_url }}">
          <div><button class="my-btn" type="submit">зберегти</button></div>
        </form>
      </div>
    </div>
  {% else %}
    <div>{% include "comments/like_snippet.html" with like_object=comment.likes %}</div>
  {% endif %}
  <br><a data-bs-toggle="collapse" data-bs-target="#collapse-comment-replies-{{ comment.pk }}">
    відповіді {{ comment.replies.count }}
  </a>
  <div class="collapse" id="collapse-comment-replies-{{ comment.pk }}">
    <div class="my-container">
      <div class="text-additional">Додати відповідь</div>
      <form action="{% url 'add_reply' comment.pk %}" method="POST">
        {% csrf_token %}
        <textarea class="my-input-text" name="new_reply_text" maxlength="512"></textarea>
        <input name="reload_url" type="hidden" value="{{ request.resolver_match|my_url }}">
        <div><button class="my-btn" type="submit">коментувати</button></div>
      </form>
    </div><br>
    <div>
      {% for reply in comment.replies.all %}
        {% include "comments/reply_snippet.html" with reply=reply %}<br>
      {% endfor %}
    </div>
  </div>
```

</div>

</div>

</div>

ДОДАТОК Г

Код шаблону “like_snippet.html”

```
{% load static %}
<div>
  <div id="like-num-{{ like_object.pk }}" style="display: inline-block; font-size: 30px;">{{
like_object.likes.count }}</div>
  <div style="display: inline-block; margin-right: 50px;">
    <button id="like-btn-{{ like_object.pk }}" data-url="{% url 'like' like_object.pk %}"
      style="padding: 0; border: none; background: none;" title="Вподобайка">
      
    </button>
  </div>

  <div style="display: inline-block;">
    <button id="dislike-btn-{{ like_object.pk }}" data-url="{% url 'dislike' like_object.pk %}"
      style="padding: 0; border: none; background: none;" title="Дизлайк">
      
    </button>
  </div>

  <div id="dislike-num-{{ like_object.pk }}" style="display: inline-block; font-size: 30px;
margin-right: 50px;">{{ like_object.dislikes.count }}</div>

  <div style="display: inline-block;">
    <button id="report-btn-{{ like_object.pk }}" data-url="{% url 'report' like_object.pk %}"
      style="padding: 0; border: none; background: none;" title="Поскаржитися">
      
    </button>
  </div>

<script type="text/javascript">
function UpdateLikesData({{ like_object.pk }})(response){
  $("#like-num-{{ like_object.pk }}").text(response["likes"]);
  if(response["liked"]){
    $("#like-img-{{ like_object.pk }}").attr("src","{% static 'images/like_on.svg' %}");
  }
  else{ $("#like-img-{{ like_object.pk }}").attr("src","{% static 'images/like.svg' %}"); }

  $("#dislike-num-{{ like_object.pk }}").text(response["dislikes"]);
  if(response["disliked"]){
    $("#dislike-img-{{ like_object.pk }}").attr("src","{% static 'images/dislike_on.svg' %}");
  }
  else{ $("#dislike-img-{{ like_object.pk }}").attr("src","{% static 'images/dislike.svg' %}"); }
}

$("#like-btn-{{ like_object.pk }}").click(() => {
  $.ajax({
    url: $("#like-btn-{{ like_object.pk }}").data("url"),
```

```

        success: UpdateLikesData{{like_object.pk}}
    })
})

$("#dislike-btn-{{like_object.pk}}").click(() => {
    $.ajax({
        url: $("#dislike-btn-{{like_object.pk}}").data("url"),
        success: UpdateLikesData{{like_object.pk}}
    })
})

$("#report-btn-{{like_object.pk}}").click(() => {
    $.ajax({
        url: $("#report-btn-{{like_object.pk}}").data("url"),
        success: (response) => {
            if(response["reported"]){
                $("#report-img-{{like_object.pk}}").attr("src","{% static 'images/alert_on.svg' %}");
            }else{
                $("#report-img-{{like_object.pk}}").attr("src","{% static 'images/alert.svg' %}");
            }
        }
    })
})
</script>
</div>

```

ДОДАТОК Д

Код шаблону “reply_snippet.html”

```
{% load comments_tags %}
<div class="my-container">
  <div class="text-title">
    {% if reply.user %}
      {{ reply.user.first_name }} {{ reply.user.last_name }}
    {% else %}
      ghost
    {% endif %}
  </div>
  <div class="text-additional">{{ reply.created_date }}</div>
  <div class="text-main">{{ reply.text }}</div>

  {% if reply.edited %}
    <div class="text-additional">змінено</div>
  {% endif %}

  {% if reply.user == user %}
    <div>{% include "comments/like_view_snippet.html" with like_object=reply.likes %}</div>
    <a class="text-additional" data-bs-toggle="collapse"
data-bs-target="#collapse-reply-edit-{{ reply.pk }}">
      редагувати
    </a>
    <div class="collapse" id="collapse-reply-edit-{{ reply.pk }}">
      <div class="my-container">
        <div class="text-additional">Редагування</div>
        <form action="{% url 'edit_reply' reply.pk %}" method="POST">
          {% csrf_token %}
          <textarea class="my-input-text" name="reply_new_text"
maxlength="512">{{ reply.text }}</textarea>
          <input name="reload_url" type="hidden" value="{{ request.resolver_match|my_url }}">
          <div><button class="my-btn" type="submit">зберегти</button></div>
        </form>
      </div>
    </div>
  {% else %}
    <div>{% include "comments/like_snippet.html" with like_object=reply.likes %}</div>
  {% endif %}
</div>
```

ДОДАТОК Е

Код шаблону “question.html”

```
{% extends "common/base.html" %}

{% block title %}Питання{% endblock title %}

{% block content %}
<div>
  <div class="my-container">
    {% if question.user %}
    <div class="text-title">{{ question.user.first_name }} {{ question.user.last_name }}</div>
    {% else %}
    <div class="text-title">ghost</div>
    {% endif %}
    <div class="text-additional">{{ question.create_date }}</div>
    <div class="text-main">{{ question.question }}</div>
    <div class="text-main">{{ question.description }}</div>
    {% if question.images.count %}
    <div>
      {% for img in question.images.all %}
      
      {% endfor %}
    </div>
    {% endif %}
    <div>
      {% if question.user == user %}
      {% include "comments/like_view_snippet.html" with like_object=question.likes %}
      {% else %}
      {% include "comments/like_snippet.html" with like_object=question.likes %}
      {% endif %}
    </div>
  </div><br>

  {% include "comments/comments_list_snippet.html" with comments_list=question.answers %}

</div>
{% endblock content %}
```

ДОДАТОК Є

Код шаблону “question_list.html”

```
{% extends "common/base.html" %}

{% block title %}Запитання користувачів{% endblock title %}

{% block content %}
<div>
  <div class="my-container">
    <h1 class="text-title">Запитання користувачів</h1>
    <div class="text-main">В цьому розділі користувачі можуть ставити питання пов'язані з темою
комп'ютерів та комп'ютерних збірок, а також допомагати з питаннями інших.</div>
    <div>
      <a href="{% url 'new_question' %}">
        <button class="my-btn">Нове питання</button>
      </a>
    </div>
  </div><br>

  <div class="my-container">
    <form method="GET">
      <input class="my-input" type="text" name="query" value="{{query}}">
      <button class="my-btn" type="submit">пошук</button>
    </form>
  </div><br>

  <div>
    {% for question in questions %}
    <div class="my-container">
      <a href="{% url 'question' question.pk %}" style="text-decoration: none;">
        {% if question.user %}
        <div class="text-title">{{question.user.first_name}} {{question.user.last_name}}</div>
        {% else %}
        <div class="text-title">ghost</div>
        {% endif %}
        <div class="text-additional">{{question.create_date}}</div>
        <div class="text-main">{{question.question}}</div>
      </a>
      <div>
        {% if question.user == user %}
        {% include "comments/like_view_snippet.html" with like_object=question.likes %}
        {% else %}
        {% include "comments/like_snippet.html" with like_object=question.likes %}
        {% endif %}
      </div>
    </div><br>
    {% endfor %}
  </div>

  <div>
    <ul class="my-list">
      {% for p in paginator.page_range %}

        {% if page_obj.number == p %}
        <li class="my-li" style="font-size: 18px;">
          <div class="my-container">{{p}}</div>
        </li>
        </div>
      </div>
    </ul>
  </div>
</div>
```

```
</li>
{% else %}
<li class="my-li">
  <a href="?page={{p}}&query={{request.GET.query}}">
    <div class="my-container" style="font-size: 14px;">{{p}}</div>
  </a>
</li>
{% endif %}

{% endfor %}
</ul>
</div>

</div>
{% endblock content %}
```

ДОДАТОК Ж

Код шаблону “new_question.html”

```
{% extends "common/base.html" %}

{% block title %}Нове питання{% endblock title %}

{% block content %}
<div>
  <h1 class="text-title">Нове запитання</h1>
  <div class="my-container">
    <form action="{% url 'new_question' %}" method="POST" enctype="multipart/form-data">
      {% csrf_token %}
      {% for field in question_form %}
        <div>{{ field.errors }}</div>
        <div>{{ field.label_tag }}</div>
        <div>{{ field }}</div>
      {% endfor %}
      {{ images.as_p }}
      <div><button class="my-btn" type="submit">Зберегти</button></div>
    </form>
  </div>
</div>
{% endblock content %}
```

ДОДАТОК 3

Код стилів CSS

```
.root {
  --bg1: rgb(13, 17, 23);
  --bg2: rgb(32, 38, 45);
  --bg3: rgb(1, 4, 9);
  --text1: rgb(240, 240, 250);
  --text2: rgb(15, 110, 255);
  --text3: rgb(250, 170, 30);
  --text-error: rgb(250, 70, 40);
  --text-additional: rgb(140, 140, 140);
  --active: rgb(250, 90, 30);
}

body {
  font-size: 18px;
  background: var(--bg1);
}

.my-navbar {
  background: var(--bg3);
}

.my-footer{
  background: var(--bg3);
}

.my-container{
  border: 2px solid var(--text1);
  border-radius: 20px;
  background-color: var(--bg1);
  color: var(--text1);
  padding: 5px 20px 5px 20px;
}

.my-container:hover{
  border: 2px solid var(--text1);
  border-radius: 25px;
  background-color: var(--bg2);
  padding: 5px 20px 5px 20px;
}

.my-title-container{
  background-color: var(--bg2);
  color: var(--text3);
  border-bottom: 2px solid var(--text1);
  border-radius: 20px;
  padding: 5px 20px 5px 20px;
}

.my-input{
  width: 75%;
  border-radius: 5px;
}

.my-input-text{
```

```
width: 75%;
height: 5em;
background-color: var(--bg1);
border-radius: 5px;
padding: 3px 5px 3px 5px;
color: var(--text1);
}

.my-btn{
background-color: var(--bg1);
border-radius: 10px;
color: var(--text1);
}

.my-btn:hover{
background-color: var(--bg2);
border: 2px solid var(--text1);
border-radius: 15px;
color: var(--text2);
}

.my-btn:active{
background-color: var(--active);
border: none;
border-radius: 15px;
color: var(--bg2);
}

.text-main{
color: var(--text1);
font-size: 18px;
}

.text-title{
font-size: 24px;
color: var(--text3);
}

.text-error{
font-size: 14px;
color: var(--text-error);
}

.text-additional{
font-size: 14px;
color: var(--text-additional);
}

.my-info-table{
width=75%;
border-collapse: separate;
border-spacing: 10px;
margin-left: auto;
margin-right: auto;
}

.my-info-table td {
padding: 0px 45px 0px 0px;
```

```
}
```

```
.my-image{  
  display: block;  
  margin-left: auto;  
  margin-right: auto;  
  border-radius: 25px;  
  width: 50%;  
  padding: 20px;  
}
```

```
.my-list {  
  list-style-type: none;  
  padding: 5px;  
}
```

```
.my-li {  
  float: left;  
  margin-right: 20px;  
}
```

ДОДАТОК И

Код моделей модуля “comments”

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_delete
from django.dispatch import receiver

class Like(models.Model):
    likes = models.ManyToManyField(User, related_name="likes", blank=True)
    dislikes = models.ManyToManyField(User, related_name="dislikes", blank=True)
    complaints = models.ManyToManyField(User, related_name="complaints", blank=True)

    def __str__(self):
        return f"Likes {self.pk}"

class CommentList(models.Model):
    name = models.CharField(max_length=32, null=True)

    def __str__(self):
        return f"CommentList {self.pk}: {self.name}"

class Comment(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    comment_list = models.ForeignKey(CommentList, on_delete=models.CASCADE,
related_name="comments")
    text = models.TextField(max_length=512)
    created_time = models.DateTimeField(auto_now=True)
    created_date = models.DateField(auto_now=True)
    edited = models.BooleanField(default=False)
    last_edited_datetime = models.DateTimeField(null=True)
    likes = models.OneToOneField(Like, on_delete=models.CASCADE)
    verified_by_moderator = models.BooleanField(default=False)

    def __str__(self):
        if self.user:
            return f"{self.user}: {self.text[:20]}"
        else:
            return f"no user: {self.text[:20]}"

@receiver(post_delete, sender=Comment, dispatch_uid="comment_delete_signal")
def on_delete_comment(sender, instance, using, **kwargs):
    instance.likes.delete()

class CommentReply(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    comment = models.ForeignKey(Comment, on_delete=models.CASCADE, related_name="replies")
    text = models.TextField(max_length=512)
    created_time = models.DateTimeField(auto_now=True)
    created_date = models.DateField(auto_now=True)
    edited = models.BooleanField(default=False)
    last_edited_datetime = models.DateTimeField(null=True)
```

```
likes = models.OneToOneField(Like, on_delete=models.CASCADE)
verified_by_moderator = models.BooleanField(default=False)

def __str__(self):
    if self.user:
        return f"{self.user}: {self.text[:20]}"
    else:
        return f"no user: {self.text[:20]}"

@receiver(post_delete, sender=CommentReply, dispatch_uid="comment_reply_delete_signal")
def on_delete_comment_reply(sender, instance, using, **kwargs):
    instance.likes.delete()
```

ДОДАТОК I

Код моделей модуля “questions”

```
from django.db import models
from django.contrib.auth.models import User
from django.contrib.postgres.search import SearchVectorField, SearchVector, SearchQuery
from django.contrib.postgres.indexes import GinIndex
from django.db.models.signals import post_delete, post_save
from django.dispatch import receiver
from comments.models import CommentList, Like


class SearchManager(models.Manager):
    def search_query(self, search_text: str):
        return self.filter(search_vector=SearchQuery(search_text))


class Question(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    question = models.CharField(max_length=128)
    description = models.TextField(max_length=1024)
    create_date = models.DateField(auto_now=True)
    create_time = models.TimeField(auto_now=True)
    answers = models.ForeignKey(CommentList, on_delete=models.CASCADE)
    likes = models.ForeignKey(Like, on_delete=models.CASCADE)
    search_vector = SearchVectorField(null=True)

    objects = SearchManager()

    class Meta:
        indexes = (GinIndex(fields=["search_vector"]),)

    @receiver(post_save, sender=Question, dispatch_uid="question_save_signal")
    def on_save_question(sender, instance, **kwargs):
        Question.objects.filter(id=instance.id).update(search_vector=SearchVector("question"))

    @receiver(post_delete, sender=Question, dispatch_uid="question_delete_signal")
    def on_delete_question(sender, instance, using, **kwargs):
        instance.answers.delete()
        instance.likes.delete()

    class QuestionImage(models.Model):
        question = models.ForeignKey(Question, on_delete=models.CASCADE, related_name="images")
        image = models.ImageField(upload_to="images/users_questions")
```

ДОДАТОК І

Код моделей модуля “computer hardware”

```
from django.db import models
from django.contrib.postgres.search import SearchVectorField, SearchVector, SearchQuery
from django.contrib.postgres.indexes import GinIndex
from django.db.models.signals import post_delete, post_save
from django.dispatch import receiver
from comments.models import CommentList, Like

class SearchManager(models.Manager):
    def search_query(self, search_text: str):
        return self.filter(search_vector=SearchQuery(search_text))

class ManufacturerType(models.Model):
    name = models.CharField(max_length=32)

    def __str__(self):
        return f"{self.name}"

class Manufacturer(models.Model):
    name = models.CharField(max_length=128)
    types = models.ManyToManyField(ManufacturerType, verbose_name="є виробником:")

    def __str__(self):
        return f"{self.name}"

class ComputerPartType(models.Model):
    type_name = models.CharField(max_length=32)

    def __str__(self):
        return f"{self.type_name}"

class ComputerPart(models.Model):
    name = models.CharField(max_length=128, verbose_name="назва")
    description = models.TextField(max_length=512, verbose_name="опис")
    type = models.ForeignKey(ComputerPartType, on_delete=models.SET_DEFAULT, default=1,
        verbose_name="тип")
    msrp = models.DecimalField(max_digits=9, decimal_places=2, default=0.0,
        verbose_name="ціна рекомендована виробником")
    avg_price = models.DecimalField(max_digits=9, decimal_places=2, default=0.0,
        verbose_name="середня ціна на ринку")
    image = models.ImageField(upload_to="images/computer_parts/", null=True)
    comments = models.ForeignKey(CommentList, on_delete=models.SET_NULL, null=True)
    likes = models.ForeignKey(Like, on_delete=models.SET_NULL, null=True)
    manufacturer = models.ForeignKey(Manufacturer, on_delete=models.SET_NULL, null=True,
        verbose_name="виробник")
    search_vector = SearchVectorField(null=True)

    objects = SearchManager()

    class Meta:
```

```

indexes = (GinIndex(fields=["search_vector"]),)

def __str__(self):
    return f"{self.name}"

@receiver(post_save, sender=ComputerPart, dispatch_uid="question_save_signal")
def on_save_question(sender, instance, **kwargs):
    ComputerPart.objects.filter(id=instance.id).update(search_vector=SearchVector("name"))

@receiver(post_delete, sender=ComputerPart, dispatch_uid="computer_part_delete_signal")
def on_delete_computer_part(sender, instance, using, **kwargs):
    instance.comments.delete()
    instance.likes.delete()

class RamType(models.Model):
    type_name = models.CharField(max_length=32)
    description = models.CharField(max_length=128)

    def __str__(self):
        return f"{self.type_name}"

class RAM(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    ram_type = models.OneToOneField(RamType, on_delete=models.SET_DEFAULT, default=1,
    verbose_name="тип пам'яті")
    number_of_modules = models.IntegerField(verbose_name="кількість модулів")
    size = models.IntegerField(null=False, verbose_name="обсяг") # in megabytes
    frequency = models.IntegerField(verbose_name="частота") # in megahertz
    CL = models.IntegerField(verbose_name="CL")
    tRCD = models.IntegerField(verbose_name="tRCD")
    tRP = models.IntegerField(verbose_name="tRP")
    tRAS = models.IntegerField(verbose_name="tRAS")
    tRS = models.IntegerField(verbose_name="tRS")
    dual_rank = models.BooleanField(default=False, verbose_name="дворангова пам'ять")

    def __str__(self):
        return f"RAM: {self.computer_part}"

class CpuLine(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"CpuLine: {self.name}"

class Socket(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"Socket: {self.name}"

class IntegratedGraphics(models.Model):

```

```

name = models.CharField(max_length=64)
frequency = models.IntegerField(verbose_name="частота") # in megahertz
shaders = models.IntegerField(verbose_name="кількість потокових процесорів")

def __str__(self):
    return f"iGPU: {self.name}"

class CPU(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    cpu_line = models.ForeignKey(CpuLine, on_delete=models.SET_NULL, null=True,
verbose_name="лінійка")
    socket = models.ForeignKey(Socket, on_delete=models.SET_NULL, null=True,
verbose_name="сокет")
    cores = models.IntegerField(verbose_name="кількість ядер")
    threads = models.IntegerField(verbose_name="кількість потоків")
    cores_description = models.CharField(max_length=128)
    frequency = models.IntegerField(verbose_name="базова частота") # in megahertz
    boost_frequency = models.IntegerField(verbose_name="максимальна частота") # in megahertz
    L3size = models.IntegerField(verbose_name="обсяг кешу L3") # in megabytes
    L2size = models.IntegerField(verbose_name="обсяг кешу L2") # in megabytes
    L1size = models.IntegerField(verbose_name="обсяг кешу L1") # in kilobytes
    lithography = models.IntegerField(verbose_name="техпроцес")
    TDP = models.IntegerField(verbose_name="TDP")
    integrated_graphics = models.ForeignKey(IntegratedGraphics, on_delete=models.SET_NULL,
null=True, blank=True,
verbose_name="інтегрована графіка")

    def __str__(self):
        return f"CPU: {self.computer_part}"

class GpuLine(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"GpuLine: {self.name}"

class GpuMemoryType(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"GpuMemoryType: {self.name}"

class GpuPowerConnector(models.Model):
    name = models.CharField(max_length=64)
    can_be_connected_to = models.ManyToManyField("self", blank=True)

    def __str__(self):
        return f"GpuPowerConnector: {self.name}"

class GPU(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    gpu_line = models.ForeignKey(GpuLine, on_delete=models.SET_NULL, null=True,
verbose_name="лінійка")

```

```

frequency = models.IntegerField(verbose_name="максимальна частота графічного чіпу") # in
megahertz
shaders = models.IntegerField(verbose_name="кількість потокових процесорів")
pcie_version = models.IntegerField(verbose_name="версія PCIe")
pcie_lines = models.IntegerField(verbose_name="кількість ліній PCIe")
memory_bus_width = models.IntegerField(verbose_name="шина пам'яті")
memory_clock_speed = models.IntegerField(verbose_name="частота пам'яті") # in megahertz
memory_size = models.IntegerField(verbose_name="обсяг пам'яті") # in megabytes
memory_type = models.ForeignKey(GpuMemoryType, on_delete=models.SET_NULL, null=True,
                                verbose_name="тип пам'яті") # in megabytes
lithography = models.IntegerField(verbose_name="техпроцес")
TDP = models.IntegerField(verbose_name="TDP")
power_connection = models.ForeignKey(GpuPowerConnector, on_delete=models.SET_NULL,
null=True)
slots = models.IntegerField(default=1, verbose_name="займає слотів розширення")
width = models.IntegerField(verbose_name="ширина", null=True) # in millimeters
height = models.IntegerField(verbose_name="висота", null=True) # in millimeters
length = models.IntegerField(verbose_name="довжина", null=True) # in millimeters
number_of_fans = models.IntegerField(verbose_name="кількість вентиляторів")

def __str__(self):
    return f"GPU: {self.computer_part}"

class FormFactor(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"FormFactor: {self.name}"

class Chipset(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"Chipset: {self.name}"

class IntegratedSound(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"IntegratedSound: {self.name}"

class Motherboard(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    form_factor = models.ForeignKey(FormFactor, on_delete=models.SET_NULL, null=True,
verbose_name="форм фактор")
    socket = models.ForeignKey(Socket, on_delete=models.SET_NULL, null=True,
verbose_name="сокет",
                                related_name="motherboards")
    chipset = models.ForeignKey(Chipset, on_delete=models.SET_NULL, null=True,
verbose_name="чіпсет")
    motherboard_power_connector = models.IntegerField(verbose_name="живлення материнської
плати")
    cpu_power_connector = models.IntegerField(verbose_name="живлення процесора")

```

```

number_of_cpu_power_connector = models.IntegerField(verbose_name="кількість конекторів для живлення процесора")
ram_type_support = models.ManyToManyField(RamType, verbose_name="підтримка типів пам'яті")
number_of_ram_slots = models.IntegerField(verbose_name="кількість слотів озп")
number_of_ram_channels = models.IntegerField(verbose_name="кількість каналів пам'яті")
max_ram_volume = models.IntegerField(verbose_name="максимальний обсяг пам'яті")
min_ram_frequency = models.IntegerField(verbose_name="мінімальна частота пам'яті")
max_ram_frequency = models.IntegerField(verbose_name="максимальна частота пам'яті")
integrated_sound = models.ForeignKey(IntegratedSound, on_delete=models.SET_NULL, null=True, verbose_name="вбудована звукова карта")
igpu_support = models.BooleanField(verbose_name="підтримка вбудованої графіки")
vga_port = models.BooleanField(verbose_name="роз'єм VGA")
dvi_d_port = models.BooleanField(verbose_name="роз'єм DVI-D")
hdmi_port = models.BooleanField(verbose_name="роз'єм HDMI")
display_port = models.BooleanField(verbose_name="роз'єм Display port")
ps2 = models.BooleanField(verbose_name="роз'єм PS/2")
number_of_internal_usb_v2 = models.IntegerField(verbose_name="кількість внутрішніх usb v2.0")
number_of_internal_usb_v3 = models.IntegerField(verbose_name="кількість внутрішніх usb v3")
number_of_external_usb_v2 = models.IntegerField(verbose_name="кількість зовнішніх usb v2.0")
number_of_external_usb_v3 = models.IntegerField(verbose_name="кількість зовнішніх usb v3")
number_of_sys_fans = models.IntegerField(verbose_name="кількість роз'ємів для корпусних вентиляторів")
number_of_pcie_1x = models.IntegerField(verbose_name="кількість PCI-E 1x")
number_of_pcie_16x = models.IntegerField(verbose_name="кількість PCI-E 16x")
pcie_version = models.IntegerField(verbose_name="версія PCI-E")
number_of_sata3 = models.IntegerField(verbose_name="кількість SATA3")
number_of_m2_slots = models.IntegerField(verbose_name="кількість слотів M.2")
m2_pcie_version = models.IntegerField(verbose_name="M.2 версія PCIe")
wifi = models.BooleanField(verbose_name="вбудований wifi")
bluetooth = models.BooleanField(verbose_name="вбудований bluetooth")

def __str__(self):
    return f"Motherboard: {self.computer_part}"

```

```

class ComputerCase(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    form_factor = models.ManyToManyField(FormFactor, verbose_name="формфактор", related_name="computer_cases")
    motherboard_form_factors = models.ManyToManyField(FormFactor, verbose_name="підтримувані формфактори материнської плати")
    number_of_usb_v2 = models.IntegerField(verbose_name="кількість usb v2.0")
    number_of_usb_v3 = models.IntegerField(verbose_name="кількість usb v3")
    number_of_usb_type_c = models.IntegerField(verbose_name="кількість usb type-c")
    microphone_and_headphone_port = models.BooleanField(verbose_name="роз'єм для навушників та мікрофону")
    width = models.IntegerField(verbose_name="ширина")
    height = models.IntegerField(verbose_name="висота")
    length = models.IntegerField(verbose_name="довжина")
    number_of_fans_installed = models.IntegerField(verbose_name="встановлено вентиляторів")
    number_of_front_fans = models.IntegerField(verbose_name="максимальна кількість вентиляторів на фронтальній панелі")
    number_of_back_fans = models.IntegerField(verbose_name="максимальна кількість вентиляторів на задній панелі")
    number_of_top_fans = models.IntegerField(verbose_name="максимальна кількість вентиляторів на верхній панелі")
    max_cooler_height = models.IntegerField(verbose_name="максимальна висота процесорного кулера")

```

```

max_power_supply_length = models.IntegerField(verbose_name="максимальна довжина блоку живлення")
max_gpu_length = models.IntegerField(verbose_name="максимальна довжина відеокарти")

def __str__(self):
    return f"ComputerCase: {self.computer_part}"

class PowerSupplyCertificate(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"SsdType: {self.name}"

class PowerSupplyProtection(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"Power supply protection: {self.name}"

class PowerSupply(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    certificate = models.ForeignKey(PowerSupplyCertificate, on_delete=models.SET_NULL, null=True,
                                    verbose_name="сертифікат")
    form_factor = models.ManyToManyField(FormFactor, verbose_name="формфактор")
    protections = models.ManyToManyField(PowerSupplyProtection, verbose_name="захисти")
    motherboard_power_connector = models.IntegerField(verbose_name="живлення материнської плати")
    cpu_power_connector = models.IntegerField(verbose_name="живлення процесора")
    number_of_cpu_power_connector = models.IntegerField(verbose_name="кількість конекторів для живлення процесора")
    gpu_power_connection = models.ForeignKey(GpuPowerConnector, on_delete=models.SET_NULL, null=True,
                                             verbose_name="підключення до відеокарт")
    number_of_4pin_molex = models.IntegerField(verbose_name="кількість роз'ємів 4pin Molex")
    number_of_sata_connectors = models.IntegerField(verbose_name="кількість роз'ємів SATA")
    number_of_4pin_floppy = models.IntegerField(verbose_name="кількість роз'ємів 4pin floppy")
    width = models.IntegerField(verbose_name="ширина")
    height = models.IntegerField(verbose_name="висота")
    length = models.IntegerField(verbose_name="довжина")
    power = models.IntegerField(verbose_name="потужність")
    cables_can_be_remove = models.BooleanField(verbose_name="кабелі відстібаються")

    def __str__(self):
        return f"PowerSupply: {self.computer_part}"

class SsdType(models.Model):
    name = models.CharField(max_length=64)

    def __str__(self):
        return f"SsdType: {self.name}"

class SsdTech(models.Model):
    name = models.CharField(max_length=64)

```

```

def __str__(self):
    return f"SsdTech: {self.name}"

class Ssd(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    type = models.ForeignKey(SsdType, on_delete=models.SET_NULL, null=True, verbose_name="тип")
    tech = models.ForeignKey(SsdTech, on_delete=models.SET_NULL, null=True,
verbose_name="технологія")
    size = models.IntegerField(verbose_name="обсяг пам'яті") # in gigabytes
    read_speed = models.IntegerField(verbose_name="швидкість читання") # in megabytes
    write_speed = models.IntegerField(verbose_name="швидкість запису") # in megabytes
    TBW = models.IntegerField(verbose_name="ресурс запису") # in terabytes

    def __str__(self):
        return f"Ssd: {self.computer_part}"

class Hdd(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    buffer = models.IntegerField(verbose_name="розмір буферу") # in megabytes
    rpm = models.IntegerField(verbose_name="швидкість обертання")
    size = models.IntegerField(verbose_name="обсяг пам'яті") # in gigabytes

    def __str__(self):
        return f"Hdd: {self.computer_part}"

class Cooler(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    fan_speed = models.IntegerField(verbose_name="максимальна швидкість обертання вентиляторів")
    fan_diameter = models.IntegerField(verbose_name="діаметр вентиляторів")
    numbers_of_fans = models.IntegerField(verbose_name="кількість вентиляторів")
    numbers_of_heat_pipes = models.IntegerField(verbose_name="кількість теплових трубок")
    mass = models.FloatField(verbose_name="вага")
    width = models.FloatField(verbose_name="ширина")
    height = models.FloatField(verbose_name="висота")
    length = models.FloatField(verbose_name="довжина")
    TDP = models.IntegerField(verbose_name="максимальний TDP")
    socket_compatibility = models.ManyToManyField(Socket, verbose_name="підтримка сокетів")

    def __str__(self):
        return f"Cooler: {self.computer_part}"

class LCS(models.Model):
    computer_part = models.OneToOneField(ComputerPart, on_delete=models.CASCADE)
    fan_diameter = models.IntegerField(verbose_name="діаметр вентиляторів")
    numbers_of_fans = models.IntegerField(verbose_name="кількість вентиляторів")
    pump_speed = models.IntegerField(verbose_name="швидкість обертання вентиляторів")
    length = models.FloatField(verbose_name="довжина радіатора")
    socket_compatibility = models.ManyToManyField(Socket, verbose_name="підтримка сокетів")

    def __str__(self):
        return f"Cooler: {self.computer_part}"

```

ДОДАТОК Й

Код “view” класів модуля “comments”

```
from datetime import datetime
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import get_object_or_404, redirect
from django.http.response import JsonResponse
from django.views import View
from .models import Like, CommentList, Comment, CommentReply

class AddCommentView(LoginRequiredMixin, View):
    login_url = "login"

    def post(self, request, comments_list_pk):
        txt: str = request.POST.get("new_comment_text")
        if txt:
            comments_list: CommentList = get_object_or_404(CommentList, id=comments_list_pk)
            likes: Like = Like.objects.create()
            comment: Comment = Comment.objects.create(user=self.request.user,
                                                       comment_list=comments_list,
                                                       text=txt,
                                                       likes=likes)

            comment.save()
        return redirect(request.POST.get("reload_url"))

class EditCommentView(LoginRequiredMixin, View):
    login_url = "login"

    def post(self, request, comment_pk):
        txt: str = request.POST.get("comment_new_text")
        if txt:
            comment: Comment = get_object_or_404(Comment, id=comment_pk)
            if comment.user == self.request.user:
                comment.text = txt
                comment.edited = True
                comment.last_edited_datetime = datetime.now()
                comment.save()
        return redirect(request.POST.get("reload_url"))

class AddReplyView(LoginRequiredMixin, View):
    login_url = "login"

    def post(self, request, comment_pk):
        txt: str = request.POST.get("new_reply_text")
        if txt:
            likes: Like = Like.objects.create()
            comment: Comment = get_object_or_404(Comment, id=comment_pk)
            reply: CommentReply = CommentReply.objects.create(user=self.request.user,
                                                              comment=comment,
                                                              text=txt,
                                                              likes=likes)

            reply.save()
        return redirect(request.POST.get("reload_url"))
```

```

class EditReplyView(LoginRequiredMixin, View):
    login_url = "login"

    def post(self, request, reply_pk):
        txt: str = request.POST.get("reply_new_text")
        if txt:
            reply: CommentReply = get_object_or_404(CommentReply, id=reply_pk)
            if reply.user == self.request.user:
                reply.text = txt
                reply.edited = True
                reply.last_edited_datetime = datetime.now()
                reply.save()
            return redirect(request.POST.get("reload_url"))

```

```

class LikeView(LoginRequiredMixin, View):
    login_url = "login"

    def get(self, request, pk):
        likes_object: Like = get_object_or_404(Like, id=pk)
        liked = self.request.user in likes_object.likes.all()
        disliked = self.request.user in likes_object.dislikes.all()
        if liked:
            likes_object.likes.remove(self.request.user)
            liked = False
        elif disliked:
            likes_object.dislikes.remove(self.request.user)
            likes_object.likes.add(self.request.user)
            disliked = False
            liked = True
        else:
            likes_object.likes.add(self.request.user)
            liked = True
        return JsonResponse({"likes": likes_object.likes.count(),
                            "dislikes": likes_object.dislikes.count(),
                            "liked": liked,
                            "disliked": disliked})

```

```

class DislikeView(LoginRequiredMixin, View):
    login_url = "login"

    def get(self, request, pk):
        likes_object: Like = get_object_or_404(Like, id=pk)
        liked = self.request.user in likes_object.likes.all()
        disliked = self.request.user in likes_object.dislikes.all()
        if liked:
            likes_object.likes.remove(self.request.user)
            likes_object.dislikes.add(self.request.user)
            liked = False
            disliked = True
        elif disliked:
            likes_object.dislikes.remove(self.request.user)
            disliked = False
        else:
            likes_object.dislikes.add(self.request.user)
            disliked = True

```

```
return JsonResponse({"likes": likes_object.likes.count(),
                    "dislikes": likes_object.dislikes.count(),
                    "liked": liked,
                    "disliked": disliked})
```

```
class ReportView(LoginRequiredMixin, View):
    login_url = "login"

    def get(self, request, pk):
        likes_object: Like = get_object_or_404(Like, id=pk)
        reported = self.request.user in likes_object.complaints.all()
        if reported:
            likes_object.complaints.remove(self.request.user)
        else:
            likes_object.complaints.add(self.request.user)
        return JsonResponse({"reported": not reported})
```

ДОДАТОК К

Код “view” класів модуля “questions”

```
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import redirect, render, get_object_or_404
from django.views import View
from django.views.generic import ListView
from .models import Question, QuestionImage
from .forms import QuestionForm, QuestionImageForm
from comments.models import CommentList, Like

class QuestionsListView(ListView):
    model = Question
    template_name = "questions/questions_list.html"
    context_object_name = "questions"
    paginate_by = 4

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context["query"] = self.request.GET.get("query", "")
        return context

    def get_queryset(self):
        search_text: str = self.request.GET.get("query", "").strip()
        if search_text:
            return self.model.objects.search_query(search_text)
        else:
            return super().get_queryset()

class QuestionView(View):
    template_name = "questions/question.html"

    def get(self, request, question_pk):
        question = get_object_or_404(Question, pk=question_pk)
        return render(request, self.template_name, {"question": question})

class NewQuestionView(LoginRequiredMixin, View):
    login_url = "login"
    template_name = "questions/new_question.html"

    def get(self, request):
        form = QuestionForm()
        form2 = QuestionImageForm()
        return render(request, self.template_name, {"question_form": form, "images": form2})

    def post(self, request):
        form = QuestionForm(request.POST)
        form2 = QuestionImageForm(request.POST, request.FILES)
        if form.is_valid() and form2.is_valid():
            question = form.cleaned_data["question"]
            description = form.cleaned_data["description"]
            like = Like.objects.create()
            comments_list = CommentList.objects.create(name=f"answers to question")
```

```
        question = Question.objects.create(user=self.request.user, question=question,
description=description,
                                         likes=like, answers=comments_list)
    question.save()
    img = form2.cleaned_data["image"]
    if img:
        QuestionImage.objects.create(question=question, image=img)
    return redirect("questions")
return render(request, self.template_name, {"question_form": form, "images": form2})
```

ДОДАТОК Л

Код “view” класів модуля “computer hardware”

```
from django.shortcuts import render
from django.http import Http404
from django.views import View
from django.views.generic import ListView
from .models import CPU, RAM, GPU, Motherboard, ComputerCase, PowerSupply, Ssd, Hdd, LCS,
Cooler
from django.contrib.postgres.search import SearchQuery
```

```
class ComputerHardwareView(View):
    template_name = "computer_hardware/computer_hardware.html"

    def get(self, request):
        return render(request, self.template_name, {})
```

```
class ComputerPartListView(ListView):
    slug = ""
    paginate_by = 2

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context["part_slug"] = self.slug
        context["query"] = self.request.GET.get("query", "")
        return context

    def get_queryset(self):
        search_text: str = self.request.GET.get("query", "").strip()
        if search_text:
            return self.model.objects.filter(computer_part__search_vector=SearchQuery(search_text))
        else:
            return super().get_queryset()
```

```
class CpuListView(ComputerPartListView):
    model = CPU
    slug = "cpu"
    template_name = "computer_hardware/cpu_list.html"
```

```
class RamListView(ComputerPartListView):
    model = RAM
    slug = "ram"
    template_name = "computer_hardware/ram_list.html"
```

```
class GpuListView(ComputerPartListView):
    model = GPU
    slug = "gpu"
    template_name = "computer_hardware/gpu_list.html"
```

```
class MotherboardListView(ComputerPartListView):
    model = Motherboard
```

```
slug = "motherboard"
template_name = "computer_hardware/motherboard_list.html"
```

```
class ComputerCaseListView(ComputerPartListView):
    model = ComputerCase
    slug = "computer_case"
    template_name = "computer_hardware/computer_case_list.html"
```

```
class PowerSupplyListView(ComputerPartListView):
    model = PowerSupply
    slug = "power_supply"
    template_name = "computer_hardware/power_supply_list.html"
```

```
class SsdListView(ComputerPartListView):
    model = Ssd
    slug = "ssd"
    template_name = "computer_hardware/ssd_list.html"
```

```
class HddListView(ComputerPartListView):
    model = Hdd
    slug = "hdd"
    template_name = "computer_hardware/hdd_list.html"
```

```
class CoolerListView(ComputerPartListView):
    model = Cooler
    slug = "cooler"
    template_name = "computer_hardware/cooler_list.html"
```

```
class LcsListView(ComputerPartListView):
    model = LCS
    slug = "lcs"
    template_name = "computer_hardware/lcs_list.html"
```

```
class ComputerPartView(View):
    collections = {"cpu": {"model": CPU, "template": "computer_hardware/cpu.html"},
                  "ram": {"model": RAM, "template": "computer_hardware/ram.html"},
                  "gpu": {"model": GPU, "template": "computer_hardware/gpu.html"},
                  "motherboard": {"model": Motherboard, "template":
"computer_hardware/motherboard.html"},
                  "computer_case": {"model": ComputerCase, "template":
"computer_hardware/computer_case.html"},
                  "power_supply": {"model": PowerSupply, "template":
"computer_hardware/power_supply.html"},
                  "ssd": {"model": Ssd, "template": "computer_hardware/ssd.html"},
                  "hdd": {"model": Hdd, "template": "computer_hardware/hdd.html"},
                  "cooler": {"model": Cooler, "template": "computer_hardware/cooler.html"},
                  "lcs": {"model": LCS, "template": "computer_hardware/lcs.html"}}

    def get(self, request, part_type, pk):
```

```
        if part_type not in self.collections:
            raise Http404
```

```
target_part = self.collections[part_type]["model"].objects.filter(pk=pk)

if target_part.count() == 0:
    raise Http404

target_part = target_part.select_related().first()

return render(request, self.collections[part_type]["template"],
              context={"target_part": target_part,
                      "computer_part": target_part.computer_part,
                      "comments_list": target_part.computer_part.comments})
```

Ім'я користувача:
Інформаційних систем в економіці Шкуратовська Те...

ID перевірки:
1016332857

Дата перевірки:
07.06.2024 16:32:21 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.06.2024 23:00:37 EEST

ID користувача:
100005745

Назва документа: Мельник_A_IN-403docx

Кількість сторінок: 88 Кількість слів: 8389 Кількість символів: 63060 Розмір файлу: 11.80 MB ID файлу: 1016132848

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7%
Схожість

Найбільша схожість: 1.88% з джерелом з Бібліотеки (ID файлу: 1015232085)

2.59% Джерела з Інтернету 311 Сторінка 90

6.91% Джерела з Бібліотеки 539 Сторінка 91

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

Підозріле форматування 25
сторінок