МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

1 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформатика»

на тему: «Веборієнтована інформаційна система моніторингу та аналізу даних про навколишнє середовище»

здобувачки групи ІН-02 Подоляк Ангеліни Сергіївни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Ангеліна ПОДОЛЯК

(підпис)

Керівник

старший викладач,

кандидат фізико-математичних наук,

доцент Оксана ШОВКОПЛЯС _____

(підпис)

Суми – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 «Комп'ютерні науки», освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Подоляк Ангеліни Сергіївни

1. Тема роботи: «Веборієнтована інформаційна система моніторингу та аналізу даних про навколишнє середовище»
затверджую наказом по СумДУ від *«22» квітня 2024 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 1 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми та актуальності розробки веборієнтованих систем моніторингу та аналізу даних про навколишнє середовище з урахуванням потреб органів місцевої влади, конкурентів та цільової аудиторії. 2) Огляд та вибір програмних засобів для інформаційних систем. 3) Розроблення веборієнтованої інформаційної системи моніторингу та аналізу даних про навколишнє середовище. 4) Аналіз отриманих результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис)

Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми та актуальності розробки веборієнтованої інформаційної системи моніторингу та аналізу даних про навколишнє середовище, постановка й формування завдань дослідження*	06.05-08.05.2024	Виконано
2	*Огляд та вибір програмних засобів для інформаційних систем.*	08.05-11.05.2024	Виконано
3	*Розроблення веборієнтованої інформаційної системи моніторингу та аналізу даних про навколишнє середовище*	11.05-17.05.2024	Виконано
4	*Аналіз отриманих результатів*	17.05-20.05.2024	Виконано
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	20.05-28.05.2024	Виконано

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 97 стор., 36 рис., 6 таблиць, 2 додатки, 24 використаних джерела

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки спрямована на вирішення важливої соціальної проблеми управління навколишнім середовищем за допомогою розробки веборієнтованої інформаційної системи для моніторингу та аналізу даних про стан міського середовища.

Об'єкт дослідження – процес моніторингу та аналізу даних про навколишнє середовище.

Предмет дослідження – методи і моделі веборієнтованої інформаційної системи для моніторингу та аналізу даних про навколишнє середовище .

Мета роботи – розроблення веборієнтованої інформаційної системи, що дозволить користувачам спостерігати за станом навколишнього середовища, повідомляти про виявлені проблеми та залучати громадські ресурси для їх вирішення.

Методи дослідження – аналіз даних, алгоритми прийняття рішень та інструменти побудови веб-орієнтованих інформаційних систем.

Результати – розроблена веборієнтована інформаційна система, яка надає можливість моніторингу стану навколишнього середовища будь-якого населеного пункту, виявлення проблем та їх подальшого вирішення залученням громадських ресурсів. Проведено тестування системи на реальних даних міського середовища.

МОНІТОРИНГ, АНАЛІЗ ДАНИХ, НАВКОЛИШНЄ СЕРЕДОВИЩЕ,
ГРОМАДСЬКІ РЕСУРСИ, DJANGO, MYSQL

ЗМІСТ

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки спрямована на вирішення важливої соціальної проблеми управління навколишнім середовищем міста за допомогою розробки веборієнтованої інформаційної системи для моніторингу та аналізу даних про стан міського середовища.

Об’єкт дослідження – процес моніторингу та аналізу даних про навколишнє середовище міста з метою виявлення та вирішення проблем .

Предмет дослідження – методи і моделі, які застосовуються при створенні веборієнтованої інформаційної системи.

Гіпотеза. Застосування веборієнтованої інформаційної системи для моніторингу та аналізу даних про навколишнє середовище міста дозволить ефективно виявляти проблеми та шляхи їх вирішення, сприяючи покращенню якості життя мешканців та забезпечуючи сталий розвиток міста.

Новизна. Реалізовано унікальну схему винагород за активне залучення користувачів до вирішення екологічних проблем, що включає нагороди у вигляді можливостей для безкоштовного дозвілля та отримання знижок у кафе. Цей механізм стимулює не лише активну громадську участь у захисті навколишнього середовища, а й підтримує місцевий малий бізнес через розміщення реклами для невеликих закладів, що сприяє розвитку національної економіки.

Апробація матеріалів роботи. Основні результати роботи оприлюднені та обговорені на міжнародній науково-технічній конференції студентів та молодих вчених «Інформатика, математика, автоматика» (ІМА – 2024).

Зв’язок роботи з науковою темою. Кваліфікаційна робота виконана на кафедрі комп’ютерних наук та пов’язана з виконанням науково-дослідної роботи

№ 0118U006971 «Методи, математичні моделі та інформаційні технології аналізу і синтезу інфокомунікаційних систем» (2018-2023).

Структура. Дана робота організована у такий спосіб: у вступі розглядається актуальність та новизна теми дослідження; далі проводиться аналітичний огляд сучасного стану навколишнього середовища у певних населених пунктах; проводиться аналіз конкурентів; формулюються конкретні завдання дослідження; обґрунтовується вибір необхідних інструментів та технологій для розробки веборієнтованої інформаційної системи; подається детальний опис програмного забезпечення розробленого в ході дослідження; висновки систематизують отримані результати; у розділі «Список використаних джерел» перераховуються джерела, використані для підготовки роботи, та додаються необхідні додатки.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз предметної області

У сучасному світі інформаційні системи відіграють важливу роль у всіх сферах життя. Вони дозволяють збирати, зберігати, обробляти та передавати інформацію в ефективний та організований спосіб. Про це говорив у своїй роботі «The role of information technology in business success» Вівек Кулкарні [1]. Він описав деякі з аспектів, де інформаційні системи є надзвичайно корисними:

- підвищення продуктивності: інформаційні системи дозволяють автоматизувати багато рутинних завдань, що дозволяє працювати ефективніше та економити час і ресурси;
- покращення управління: вони надають керівництву доступ до важливої інформації для прийняття стратегічних рішень, що сприяє успішному управлінню організаціями;
- підвищення якості послуг: інформаційні системи дозволяють покращити обслуговування клієнтів, забезпечуючи їм доступ до інформації та сервісів у зручний для них спосіб;
- оптимізація процесів: вони допомагають оптимізувати бізнес-процеси, виявляючи можливості для покращення та ефективно впроваджуючи їх.

Тема моніторингу та аналізу даних про навколишнє середовище є надзвичайно важливою з кількох причин. Аналізуючи ці дані, ми отримуємо унікальну можливість зрозуміти стан природних екосистем, їх взаємозв'язки з господарською діяльністю та вплив на здоров'я людей.

Моніторинг дозволяє вчасно виявляти зміни в екосистемах, забруднення довкілля та інші проблеми, що виникають, що дає можливість приймати заходи для їх запобігання або зменшення. Це допомагає зберігати природні ресурси та зберегти різноманіття екосистем.

Аналіз даних про навколишнє середовище є важливим для оцінки впливу на здоров'я людей. Наприклад, він дозволяє виявити шкідливі впливи, такі як забруднення повітря, води та ґрунтів, і приймати заходи для їх зменшення.

Такий моніторинг і аналіз також сприяють підтримці сталого розвитку, оскільки вони допомагають забезпечити баланс між використанням ресурсів та їх збереженням для майбутніх поколінь.

Поліпшення управління середовищем є ще одним важливим аспектом. Об'єктивна інформація про стан довкілля дозволяє урядам, організаціям та громадянам приймати обґрунтовані рішення щодо збереження природи, зменшення забруднення та підтримки екологічно чистих технологій [2].

Саме тому розроблення веборієнтованої системи для збору та моніторингу даних про навколишнє середовище має велике значення для покращення якості життя в міських районах. Ця система дозволить мешканцям міста спостерігати за проблемами довкілля, такими як сміттєзвалища, пошкоджені дерева чи інші негативні явища, і швидко повідомляти про них відповідним службам.

Аналіз попередніх досліджень показує, що подібні системи вже успішно функціонують у різних країнах світу, таких як США, Велика Британія та Канада. Вони дозволяють ефективно виявляти, моніторити та вирішувати проблеми довкілля, залучаючи до цього процесу активну участь громадян [3].

Таким чином, розроблення веборієнтованої системи моніторингу та аналізу даних про навколишнє середовище є актуальним та перспективним напрямом для покращення якості життя в містах.

1.2 Аналіз аналогічних проєктів

На сучасному етапі розвитку технологій та діджиталізації, існують численні проекти та платформи, спрямовані на збір та моніторинг даних про навколишнє середовище з метою покращення якості життя мешканців міста. Деякі з них заслуговують на увагу завдяки своїм функціональностям та особливостям.

Один із таких проєктів – «Відкрите місто». Ця платформа дає можливість мешканцям створювати повідомлення про проблеми у місті, які потребують вирішення відповідними муніципальними органами. Однак, не дуже зручності її використання та відсутність стимулюючих механізмів для активної участі користувачів можуть стати перешкодою у її ефективності [4].

Ще один схожий проєкт – «Дорога», спрямований на створення звернень до Укравтодору стосовно стану доріг. Однак, обмежений функціонал та відсутність можливості залучення широкого кола користувачів обмежують його ефективність.

Також варто відзначити проєкт «SumDU bug tracker», який реалізовується студентами СумДУ. Ця платформа дозволяє створювати повідомлення про проблеми на території університету та направляти їх на розгляд відповідним службам. Проте, обмежений географією користувачів та відсутність широкомасштабного підходу обмежують його потенціал.

Необхідно зазначити, що наразі цей сервіс вже не працює, але саме він дав ідею для створення моєї веборієнтованої платформи ще у 2022 році, коли я брала участь у «Unido gcip regional accelerator center for innovation, technology, and start-ups in Sumy region of ukraine».

Таблиця 1.1 Порівняння проєктів зі схожою ідеєю

Назва	Функціонал	Вартість	Географія клієнтів	Канали взаємодії з клієнтом
Відкрите місто	Дає можливість створювати повідомлення про проблему, яку буде направлено у відповідну організацію для вирішення	Безкоштовно	Україна	Просування у Facebook та на сайтах міських рад

Продовження таблиці 1.1

Дорога	Дає можливість створювати звернення до Укравтодору	Безкоштовно	Україна	Статті на інформаційних сервісах про ідеї розвитку міста
SumDU bug tracker	Дає можливість створювати повідомлення про проблему, яку буде направлено на розгляд до адміністрації	Безкоштовно	м. Суми(СумДУ)	Реклама у Telegram та Istagram університету

Таблиця 1.2 Порівняння проєктів за їх перевагами та недоліками

Назва	Переваги	Недоліки
Відкрите місто	– Простий та зручний інтерфейс для створення повідомлень про проблеми. – Широке охоплення українського сегменту Facebook та сайтів міських рад, що сприяє залученню користувачів.	– Обмежена функціональність порівняно з іншими проєктами. – Відсутність інтеграції з іншими платформами для моніторингу та аналізу даних про навколишнє середовище
Дорога	– Можливість створювати звернення до Укравтодору, що є важливим для покращення дорожнього середовища. – Інформаційна підтримка на інформаційних сервісах стосовно ідей розвитку міста.	– Специфічна спрямованість лише на дорожнє середовище, що може обмежити аудиторію. – Відсутність можливості звертатися до інших міських служб та підприємств.

Продовження таблиці 1.2

SumDU bug tracker	– Зручна система створення повідомлень про проблеми, що дозволяє користувачам швидко і ефективно висловлювати свої заявки. – Широка публічність через рекламу у Telegram та Instagram університету.	– Обмежена географія клієнтів, оскільки платформа орієнтована на місто Суми та студентів СумДУ. – Можливість використання системи лише в рамках університетської спільноти, що зменшує потенційну аудиторію користувачів.

1.3 Аналіз цільової аудиторії

Аналіз цільової аудиторії є важливим етапом для розуміння потреб та очікувань вашої цільової аудиторії. Проводити такий аналіз є вкрай важливим етапом створення якісного програмного продукту, адже потрібно розуміти потреби аудиторії, аби повністю їх задовільнити.

Google Forms - зручний інструмент для створення опитувань та аналізу результатів. Саме тому я використовувала його для свого дослідження.

Загалом опитування пройшло 284 людини. Стосовно вікової групи можна виділити три основних:

1) 35 – 44 (22,2%)
2) 45 – 54 (29,9%)
3) 55 – 64 (21,1%)

Як бачимо це доволі зрілі люди зі свідомою позицією, більшість з яких (65,8 %) є жителями міст.

Абсолютна більшість(95,8%) дали відповідь: «Дуже важливо» на запитання: «**Як ви оцінюєте важливість збереження природного середовища для майбутніх поколінь?**», що означає готовність жителів нашої країни до змін та усвідомлення важливості даного питання.

Опитувані відокремлюють відразу декілька способів для поліпшення екологічної ситуації (рис. 1.1). Отже, вони розуміють, які є засоби для боротьби з такими проблемами.

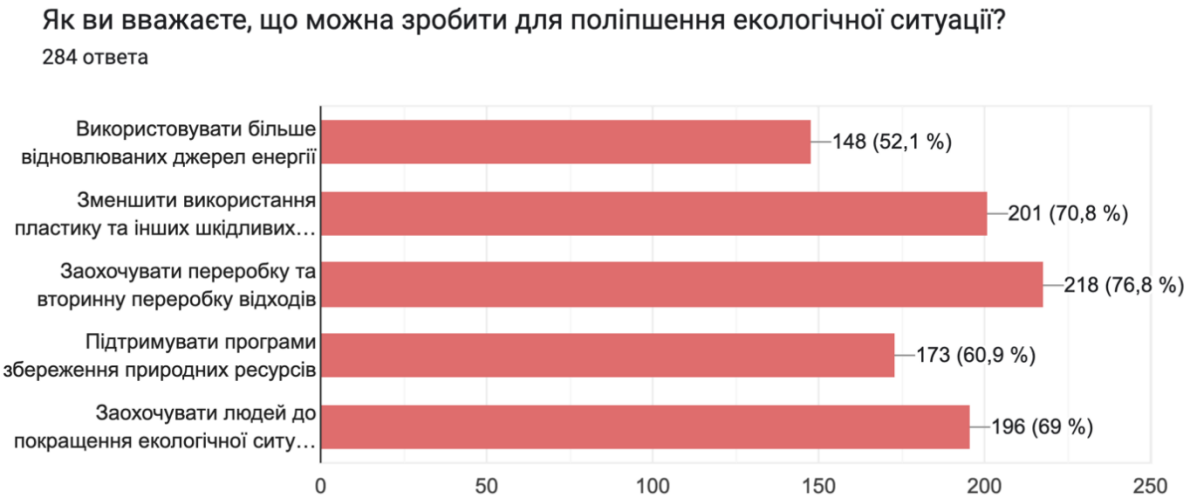

Рисунок 1.1 – Вибір методів для поліпшення екологічної ситуації

Більшість людей (82,7%) вважають, що загрози, які поставляються перед навколишнім середовищем є дуже серйозними, крім того більшість(81,%) вважають дуже важливим процес залучення громадськості до екологічних ініціатив та дій**.**

Розробка веб-сайтів, спрямованих на підвищення екологічної свідомості та залучення громадськості до заходів щодо охорони навколишнього середовища, може бути надзвичайно важливою. На це вказує відповідь на наступне питання(рис 1.2) , адже більшість опитуваних час від часу такими платформа користуються.

Рисунок 1.2 – Частота використання онлайн-платформ для отримання інформації про екологічні питання

Крім того, на питання: «**Чи підтримуєте ви ідею сприяння збереженню довкілля через використання онлайн-платформ?**» 78,2 відсотки людей відповіли «Так, я підтримую». Отже, для мого продукту уже є потенційна аудиторія, яка залюбки могла ним користуватись.

Попри все, на запитання: «**Чи згодні ви приймати участь у спільних екологічних ініціативах через онлайн-платформу?**» майже половина опитуванних відповіла: «Залежить від ситуації». Тому задача моєї платформи перш за все зацікавити аудиторію у вирішенні певних екологічних проблем колективним шляхом, аби відсіяти всі сумніви.

Більшість людей (56,3 %) оцінюють інформативність екологічної інформації на онлайн-платформах як частково інформативну, тому вкрай важливо наповнити мій продукт якісною інофрмацією, яка б зацікавлювала мою потенційну аудиторію.

Більшість людей хотіла б, аби проблеми населеного пункту у якому вони проживають знаходились в одному місці (рис. 1.3) Це запитання було сформоване для того, аби зрозуміти чи чекає аудиторія такий продукт на ринку.

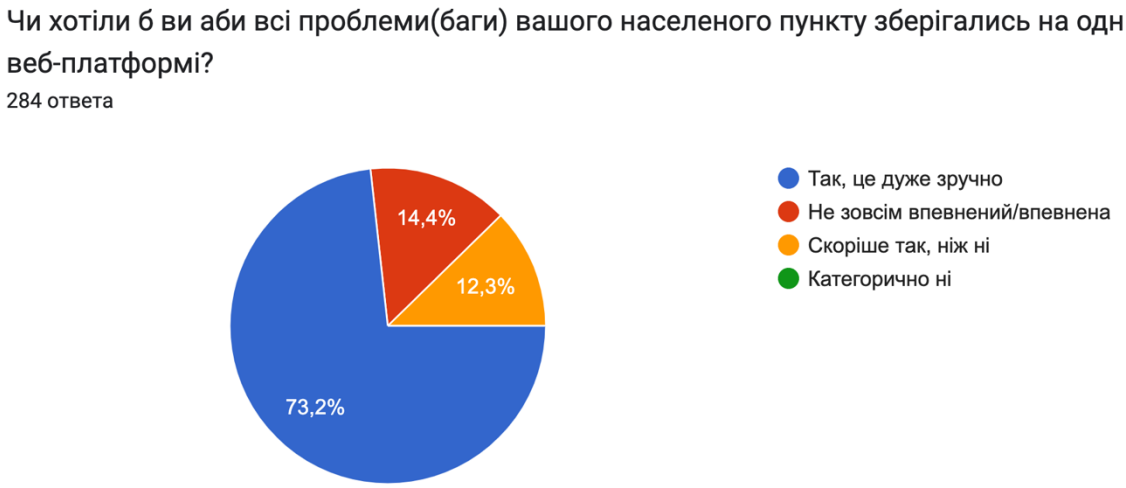

Рисунок 1.3 – Бажання опитуваних щодо зберігання інформації про проблеми населеного пункту на одній платформі

Крім того більшість опитуваних хотіла б сама повідомляти про проблеми у своєму населеному пункті і вважає це вкрай зручним механізмом роботи (рис. 1.4).

Рисунок 1.4 – Оцінка зручності вебплатформи, де можна повідомляти про проблеми у населеному пункті

Як бачимо, (рис. 1.5) серед опитуваних є ті, які б без роздумів хотіли б отримувати винагороду за свої добрі вчинки, а є ті, які не зовсім впевнені. Я

думаю, що це пов’язано, що аудиторія не зовсім розуміє які саме винагороди вони зможуть отримувати, адже чашка кави безкоштовно чи знижка на тістечко в улюбленому кафе завжди буде приємним бонусом.

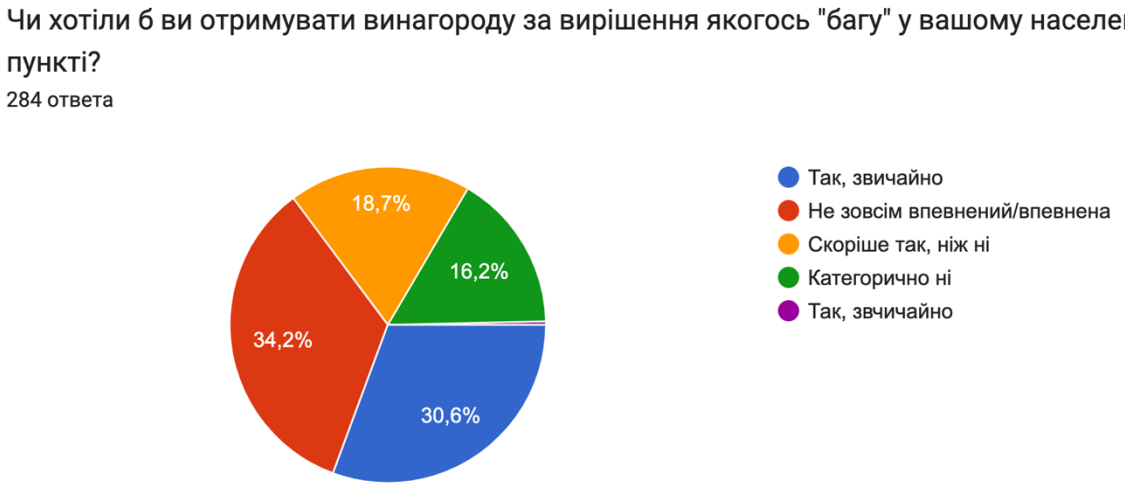

Рисунок 1.5 – Рівень бажання отримувати винагороду за вирішення багів

Після проведення аналізу цільової аудиторії можна прийти до висновку, що сайт, спрямований на підвищення екологічної свідомості та залучення громадськості до заходів щодо охорони навколишнього середовища, буде надзвичайно актуальним і важливим, адже про це свідчати відповіді опитуваних. Таким чином, розробка та запуск такого вебсайту буде відповідати очікуванням аудиторії та сприятиме підвищенню екологічної свідомості серед населення.

1.4 Постановка задачі

Метою роботи є створення веборієнтованої інформаційної системи для моніторингу та аналізу даних про навколишнє середовище з метою покращення якості життя мешканців міста.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) Розробити ім’я та концепцію вебплатформи для збору та аналізу даних про навколишнє середовище.

2) Реалізувати можливість функціонування кнопок «Create bug» та «Fix bug» для користувачів.
3) Розробити інтерфейс для завантаження фотографій багів та відмітки їх геолокації.
4) Забезпечити можливість авторизації та реєстрації користувачів для участі в системі та отримання бонусів.
5) Реалізувати систему сповіщень та повідомлень для користувачів щодо статусу їхніх заявок та бонусів.
6) Здійснити інтеграцію з органами місцевого самоврядування для ефективного вирішення зібраних проблем.
7) Створити гаманець для користувача, аби там він зберігав свої ecomoney
8) Провести тестування та оптимізацію роботи системи з метою підвищення її ефективності та надійності.
9) Підготувати документацію та інструкції користувача для успішного впровадження та використання вебплатформи.

Ці завдання спрямовані на реалізацію функціональних можливостей системи, забезпечення її коректної роботи та зручного використання користувачами.

Створення вебплатформи EcoINheart, яка буде дозволяти збирати дані про проблеми в місті та координувати їх вирішення, має на меті залучити активну участь мешканців у покращенні свого життєвого середовища. За допомогою цієї системи буде можливо надати зручний і швидкий спосіб повідомлення про проблеми, а також нагородити користувачів, які активно долучаються до вирішення цих проблем.

Ключові функції платформи EcoINheart включають:

- збір інформації про проблеми: користувачі можуть завантажувати дані та фотографії про проблеми(баги) в місті;

- моніторинг та аналіз даних: платформа аналізує надходячі дані та створює карту проблем, що допомагає місцевим органам управління приймати ефективні рішення;
- стимулювання активності користувачів: користувачі можуть отримувати заохочення за участь у вирішенні проблем, такі як знижки в місцевих закладах або екомонети, які можна витратити на придбання товарів та послуг.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір засобів для реалізації мети

1. Фреймворк для розроблення вебсайту

Для реалізації цілей та завдань, поставлених перед проектом EcoINheart, важливо обрати оптимальний програмний засіб, який забезпечить ефективний та надійний розвиток та функціонування веборієнтованої інформаційної системи.

1) Один з потенційних варіантів – використання фреймворку **Django** для розробки веб-платформи EcoINheart. Django є одним з найпопулярніших та потужних фреймворків для швидкої та безпечної розробки веб-додатків на мові програмування Python. Він надає готові компоненти та структуру, що спрощує створення веб-додатків, зокрема, сайтів, систем управління контентом, CRM та інших. Django широко використовується для розробки веб-додатків будь-якого рівня складності, від невеликих сайтів до великих корпоративних систем.

Функціонал:

- Django має вбудовану систему аутентифікації, управління сеансами та правами доступу, що спрощує розробку систем аутентифікації користувачів.
- Вбудована ORM (Object-Relational Mapping) для роботи з базою даних, що дозволяє використовувати різні типи баз даних та спрощує доступ до них.
- Django має вбудований адміністративний інтерфейс, що дозволяє адміністраторам легко керувати вмістом сайту без програмування.
- Підтримка шаблонів, яка дозволяє використовувати HTML та CSS для створення дизайну веб-сторінок [5].

2) Ще одним варіантом може слугувати **Flask**. Це легкий мікрофреймворк для розробки вебдодатків на Python. Він надає базовий функціонал та гнучкість, що дозволяє розробникам будувати додатки за їхніми власними потребами. Він використовується для створення невеликих та середніх веб-додатків, а також для розробки мікросервісів та RESTful API.

Функціонал:

- Flask має просту структуру та мінімалістичний дизайн, що дозволяє швидко розпочати роботу без додаткового коду.
- Це мінімальний фреймворк, який не нав'язує конкретних підходів до розробки та дозволяє розробникам вибирати та використовувати необхідні бібліотеки та інструменти.
- Flask забезпечує розширену підтримку розробки RESTful API, що робить його популярним у середовищах розробки мікросервісів [6].

3) Останній з розглянутих варіантів є фреймворк Pyramid. Це відкритий високорівневий Python-фреймворк для розробки вебдодатків. Він має просту архітектуру та потужні функції для швидкої розробки. Він часто використовується для розробки середніх та великих вебдодатків, а також для створення API та вебсервісів.

Функціонал:

- Pyramid забезпечує гнучку архітектуру та безліч розширень, що дозволяє розробникам створювати додатки будь-якої складності та функціональності.
- Вбудована система аутентифікації та авторизації дозволяє керувати доступом користувачів до різних частин додатку.
- Підтримка шаблонів і структура, що дає можливість розробникам швидко створювати та розгортати вебсторінки [7] .

Таблиця 2.1 Порівняння фреймворків для розроблення вебсайту

Фреймворк	Переваги	Недоліки
Flask	– Простий та мінімалістичний дизайн, що дозволяє швидко розпочати роботу. – Гнучкість у виборі бібліотек та інструментів.	– Відсутність вбудованих функцій, що може призвести до необхідності використання сторонніх бібліотек. – Менша продуктивність порівняно з Django.

Продовження таблиці 2.1

Flask	– Ідеально підходить для невеликих проектів та прототипів.	– Обмежені можливості для великих та складних проектів.
Pyramid	– Гнучка архітектура та велика кількість розширень, що дозволяє будувати додатки будь-якої складності. – Вбудована система аутентифікації та авторизації. – Підтримка шаблонів та структура, що спрощує розробку.	– Вимагає додаткових зусиль у налаштуванні та конфігурації порівняно з Django. – Менша спільнота та підтримка порівняно з Django. – Менша швидкодія порівняно з Flask. – Може виявитися складним для вивчення для початківців, особливо для тих, хто не має досвіду у розробці веб-додатків.

Висновок: розглянуті фреймворки мають свої унікальні особливості та застосування, але враховуючи особливості проекту EcoINheart, обрання Django для реалізації веб-платформи є належним рішенням. Його швидкість розробки, високий рівень безпеки, широкий спектр можливостей та розширюваність відповідають вимогам проекту та дозволять ефективно втілити поставлені завдання.

2. База даних

1) **SQLite** є потужною реляційною системою керування базами даних (СКБД), яка використовує мову структурованих запитів SQL для зберігання та управління даними. Вона відома своєю легкістю інтеграції, компактністю та високою продуктивністю.

Основні можливості:

- Легка інтеграція: SQLite є вбудованою базою даних, що означає, що вона не потребує окремого серверного програмного забезпечення. Це дозволяє легко інтегрувати її в мобільні додатки, десктопні програми та інші системи.
- Компактність: SQLite займає дуже мало місця на диску, і всі дані зберігаються в одному файлі. Це робить її ідеальною для застосувань з обмеженими ресурсами.
- Підтримка транзакцій: SQLite підтримує транзакції з атомарністю, послідовністю, ізольованістю та довговічністю (ACID), що забезпечує надійність зберігання даних.
- Автономна робота: Оскільки SQLite не потребує налаштування або адміністрування сервера, вона може працювати в автономному режимі без необхідності в мережевих з'єднаннях.
- Масштабованість: SQLite може обробляти великі обсяги даних та підтримує бази даних обсягом до терабайтів.
- Вбудована система безпеки: SQLite підтримує різні механізми безпеки, включаючи шифрування бази даних та контроль доступу користувачів.
- Підтримка стандартів SQL: SQLite підтримує більшість стандартів SQL, що дозволяє виконувати складні запити та маніпуляції з даними.
- Індексація даних: SQLite підтримує різні типи індексів, що полегшує пошук і доступ до даних, забезпечуючи високу швидкість виконання запитів [9].

2) База даних **PostgreSQL**. – це потужна об’єктно-реляційна система керування базами даних (ОРСКБД), яка використовує мову SQL для зберігання та управління даними. Вона відома своєю надійністю, розширюваністю та повнотою функцій.

Основні можливості:

- PostgreSQL пропонує розширену підтримку стандарту SQL, включаючи різноманітні типи даних, функції та операції.

- Вона підтримує географічні об'єкти та операції з геоданими, що робить її ідеальним вибором для застосунків, які працюють з географічними даними.
- PostgreSQL може обробляти великі обсяги даних і масштабуватися як вертикально, так і горизонтально.
- Ця база даних підтримує транзакції ACID, що забезпечує узгодженість даних і надійність операцій [10].

3) **MongoDB -** це потужна система керування базами даних (СКБД) з орієнтацією на документи, яка використовує JSON-подібні документи для зберігання та управління даними. Вона відома своєю гнучкістю, високою швидкодією та простотою у використанні.

Основні можливості:

- MongoDB має гнучку схему даних, яка дозволяє зберігати дані без фіксованих структур.
- Має високу продуктивність при роботі з великими обсягами неструктурованих даних, таких як документи та колекції.
- MongoDB підтримує горизонтальне масштабування, що полегшує розширення ресурсів для обробки збільшених обсягів даних.
- Ця база даних підтримує реплікацію та обмін даними, що забезпечує високу доступність і надійність системи [11].

Таблиця 2.2 Порівняння баз даних

База даних	Переваги	Недоліки
SQLite	– Висока продуктивність – Велика спільнота користувачів – Добре документована	– Менша гнучкість порівняно з деякими NoSQL базами даних – Обмежена масштабованість
PostgreSQL	– Повністю відкрита та безкоштовна	– Вимагає більше ресурсів системи порівняно з MySQL

Продовження таблиці 2.2

PostgreSQL	– Висока надійність та масштабованість – Підтримка розширень	– Складніше встановлення та налаштування
MongoDB	– Гнучка схема даних – Висока продуктивність при роботі з великими об'ємами неструктурованих даних	– Відсутність транзакцій та зовнішніх ключів – Обмежена підтримка складних операцій

Висновок: Для вебплатформи EcoINheart, що спеціалізується на зборі та аналізі даних про навколишнє середовище, використання бази даних SQLite може бути більш доцільним рішенням. SQLite відомий своєю легкістю в реалізації та управлінні, що дозволить зберігати та опрацьовувати дані ефективно при відносно невеликому обсязі. Ця база даних ідеально підходить для невеликих проектів, де важлива мінімальна конфігурація та простота використання. Такий вибір дозволить оптимізувати ресурси та спростить розгортання та підтримку проекту.Інструмент для прототипування сайту

3. Для реалізації цілей та завдань проекту EcoINheart, важливо обрати оптимальний метод прототипування, який забезпечить швидкий та ефективний процес розробки веб-інформаційної системи.

Існують такі методи прототипування:

1) Паперове прототипування: цей метод є найпростішим і найшвидшим способом створення ескізу веб-сайту чи програми. Він забезпечує масштабованість завдяки можливості додавати та видаляти елементи, а також доступність для всіх учасників проекту. Але його недоліком є відсутність інтерактивності та непрофесійний вигляд, що може ускладнити сприйняття клієнта [12].

2) Прототипування на дошці: головною перевагою цього методу є його масштабованість і можливість змін. Це дозволяє швидко й ефективно

створювати концепції та взаємодіяти з елементами. Однак обмежена доступність для учасників проекту може ускладнити спільну роботу та дискусії [13].

3) Прототипування за допомогою спеціальних програм: використання програмного забезпечення, такого як Axure Pro, Microsoft Visio, Adobe InDesign та Adobe Photoshop, надає розширені можливості та професійний вигляд прототипам. Ці програми часто використовуються для складних проектів з високими вимогами до дизайну.

4) Прототипування з використанням онлайн-сервісів: онлайн-сервіси, такі як Figma, надають швидкий та зручний спосіб створення прототипів. Вони можуть мати більш привабливий інтерфейс порівняно зі стандартними програмами, що робить їх популярними серед користувачів. Крім того, такі сервіси забезпечують можливість спільної роботи над проектом у реальному часі, що полегшує комунікацію та взаємодію всіх учасників проекту.

Таблиця 2.3 Порівняння методів прототипування

Метод прототипування	Переваги	Недоліки
Паперове прототипування	– Швидкий – Доступний для всіх – Масштабований	– Відсутність інтерактивності – Непрофесійний вигляд
Прототипування на дошці	– Масштаб прототипу – Можливість внесення змін	– Обмежена доступність для учасників проекту
Спеціальні програми	– Розширені можливості – Професійний вигляд	– Вимагає вивчення – Необхідність придбання ліцензій
Онлайн-сервіси	– Швидкий та зручний – Можливість спільної роботи	– Залежність від Інтернет-з'єднання – Обмежені можливості для роботи офлайн

Тепер слід більш детально розглянути одну із спеціальних програм для прототипування Adobe XD та онлайн-сервіс Figma, адже саме такі сучасні методи прототипування є більш ефективними та зручними згідно інформації поданої в таблиці 1.5.

Adobe XD - це інструмент для розробки та створення прототипів інтерфейсів користувача для веб-сайтів, мобільних програм та інших цифрових продуктів. Він був розроблений Adobe і призначений для дизайнерів і розробників, які хочуть створювати інтерактивні та естетично привабливі інтерфейси.

Основні функції та можливості:

– Створення макетів: Adobe XD дозволяє швидко створювати макети веб-сторінок та додатків за допомогою вбудованих інструментів для малювання та розміщення елементів.

- Прототипування: інструменти для створення інтерактивних прототипів дозволяють дизайнерам створювати динамічні переходи між сторінками та взаємодію користувача з інтерфейсом.

- Анімація: Adobe XD надає можливості для створення різноманітних анімаційних ефектів та переходів, що робить прототипи більш живими та привабливими.

- Спільна робота: інтеграція з хмарними сервісами Adobe дозволяє спільно працювати над проектом з іншими учасниками команди, обмінюватися відгуками та оновленнями в реальному часі.

- Векторний дизайн: можливість створення векторних графічних елементів дозволяє розширювати можливості для створення унікальних та адаптивних інтерфейсів.

- Автоматизація: Adobe XD підтримує різні інтеграції та плагіни, які дозволяють автоматизувати рутинні завдання та розширювати функціональність програми [14].

Figma - це онлайн-сервіс дизайну та створення прототипів, який дозволяє дизайнерам, розробникам та іншим членам команди співпрацювати над проектами, створювати веб-дизайн, мобільні програми та інші інтерактивні веб-продукти. Figma відома своїм спрощеним інтерфейсом і здатністю працювати в режимі реального часу, що дозволяє командам співпрацювати з будь-якого місця[15].

Основні функції та можливості:

– Інтерфейс з можливістю спільної роботи: Figma дозволяє кільком користувачам одночасно працювати над проектом та бачити зміни у реальному часі.

– Створення макетів та дизайну: в сервісі можна створювати векторні макети, малюнки, інтерфейси, іконки та інші елементи дизайну.

– Прототипування: Figma дозволяє створювати інтерактивні прототипи для тестування функціональності та навігації в додатках та веб-сайтах.

– Спільні бібліотеки та компоненти: можливість створення та використання спільних бібліотек компонентів для швидкого розроблення і забезпечення консистентності дизайну.

– Збереження проєктів у хмарному сховищі з можливістю легко ділитися доступом для спільної роботи з іншими користувачами [16].

Таблиця 2.4 Порівняльний аналіз Adobe XD і Figma

Критерій оцінювання	Adobe XD	Figma
Професійний вигляд	+	-
Зручний інтерфейс	+	+
Швидкість та зручність	-	+
Спільна робота	-	+
Інтеграція з Adobe	+	-
Вартість ліцензії	-	+
Залежність від установки	-	+

Примітка: "+" означає перевагу, "-" означає недолік.

Висновок: обираючи між Adobe XD і Figma для прототипування веб-інформаційної системи проекту EcoINheart, слід врахувати потреби команди та особливості проекту. У разі, якщо важливість професіонального вигляду та інтеграції з іншими продуктами Adobe важлива, Adobe XD може бути кращим вибором. Однак, у контексті розробки проекту EcoINheart, де спільна робота та зручність грають важливу роль, обрання Figma для прототипування веб-інформаційної системи є більш вигідним варіантом. Figma надає можливість спільної роботи над проектом у реальному часі, а також має інтуїтивний інтерфейс, що дозволяє швидко розробляти та тестувати прототипи . Крім того, він працює у веб-середовищі, що полегшує доступ до проекту з будь-якого пристрою та забезпечує зручність у використанні для всіх учасників команди.

3. Оформлення використаних джерел

1) Сервіс **Mendeley** - це важливий інструмент для дослідників і студентів, призначений для організації, управління та пошуку академічної літератури. Цей сервіс має численні переваги, які роблять його привабливим в порівнянні з іншими методами роботи з джерелами літератури.

Основні переваги Mendeley:

Сервіс Mendeley - це важливий інструмент для дослідників і студентів, призначений для організації, управління та пошуку академічної літератури. Цей сервіс має численні переваги, які роблять його привабливим в порівнянні з іншими методами роботи з джерелами літератури.

Основні переваги Mendeley:

– Бібліографічний менеджмент: Mendeley дозволяє створювати бібліотеку вашої наукової літератури, додавати, організовувати та класифікувати джерела. Ви можете легко завантажувати наукові статті, книги та інші джерела у свою бібліотеку та автоматично оновлювати метадані про них. Це робить процес збереження та організації літературних джерел більш ефективним.

– Пошук і імпорт літератури: Mendeley має інтеграцію з численними джерелами, такими як Google Scholar, PubMed, Scopus і іншими. Це дозволяє легко знаходити та імпортувати нові джерела безпосередньо до вашої бібліотеки.

– Автоматична генерація цитат і бібліографічних посилань: Mendeley вміє автоматично створювати цитати та бібліографічні посилання у різних форматах (APA, MLA, Chicago і багатьох інших). Це робить написання наукових робіт і звітів значно менш складним завданням.

– Спільна робота та обмін літературою: Mendeley дозволяє ділитися вашими бібліотеками з колегами та співавторами. Це особливо корисно при колективному дослідженні та написанні спільних наукових робіт.

– Доступ з будь-якого пристрою: Ваша бібліотека Mendeley синхронізується в хмарі, що дозволяє вам отримати доступ до неї з будь-якого пристрою з підключенням до Інтернету. Ви можете працювати над своєю літературою зі свого комп'ютера, смартфона або планшета.

– Безкоштовний базовий план: Багато основних функцій Mendeley доступні безкоштовно, що робить його доступним для більшості користувачів.

– Підтримка різних операційних систем: Mendeley підтримує Windows, macOS і Linux, що робить його універсальним інструментом для різних типів користувачів [17].

2) **Ручний пошук літератури** - це процес пошуку та аналізу джерел інформації з використанням традиційних джерел, таких як книги, журнали, архіви, інтерв'ю та інше, як протиставлення пошуку в онлайн-базах даних або в Інтернеті.

Особливості ручного пошуку літератури:

– Використання традиційних джерел: це означає використання книг, журналів, газет, архівів та інших традиційних джерел для збирання інформації.

– Фізичний доступ до матеріалів: ручний пошук літератури вимагає фізичного доступу до джерел, що може включати відвідування бібліотек, архівів або проведення інтерв'ю з експертами.

– Більша глибина аналізу: провідний ручний пошук може дозволити більш глибокий аналіз джерел, оскільки дослідник може працювати безпосередньо з оригінальними документами та матеріалами.

– Відсутність обмежень баз даних: ручний пошук не обмежується доступом до баз даних або веб-сайтів, тому він може допомогти знайти джерела, які можуть бути виключені з онлайн-пошуку.

– Розширення пошуку: ручний пошук може допомогти знайти джерела, які не зберігаються в електронному форматі або не доступні в Інтернеті.

Висновок: у порівнянні з ручним пошуком та введенням даних про джерела, Mendeley великою мірою автоматизує цей процес та спрощує організацію вашої літературної бази даних. Також він забезпечує докладний аналіз та зручний доступ до вашої літератури, що робить його потужним інструментом для будь-якого, хто займається академічним дослідженням чи написанням наукових робіт.

2.2 Проєктування бази даних

Під час розробки веб-орієнтованої інформаційної системи «EcoINheart» була виконана задача проєктування бази даних, яка має вирішити різноманітні завдання збереження та організації інформації, необхідної для функціонування системи. Процес проєктування бази даних передбачав створення таблиць, визначення відношень між ними та встановлення необхідних полів для збереження відповідної інформації.

Для моделювання бази даних використовувався програмний засіб **Vertabelo**, який забезпечує зручний інтерфейс для створення, редагування та візуалізації баз даних у вигляді ER-діаграм. Використання Vertabelo дозволило ефективно визначати сутності, атрибути та зв’язки між таблицями, а також забезпечити зручний процес роботи з базою даних у команді.

Для проектування бази даних було обрано методологію «логічного моделювання», що дозволило систематизувати зв’язки між різними сутностями

та атрибутами інформаційної системи «EcoINheart». Такий підхід дає можливість створити абстрактну модель бази даних, яка найкраще відображає бізнес-логіку системи.

Створення бази даних у Vertabelo включало наступні кроки:

- Створення таблиць: використовуючи інтерфейс Vertabelo, були створені таблиці для збереження різних типів даних, таких як дані про користувачів, виявлені проблеми, бонуси та повідомлення.
- Визначення полів: для кожної таблиці були визначені необхідні поля з відповідними типами даних (наприклад, рядок, ціле число, дата тощо), що відображають характеристики кожного об'єкта.
- Встановлення зв'язків: через Vertabelo були встановлені зв'язки між таблицями за допомогою зовнішніх ключів, що дозволяє забезпечити цілісність даних та ефективно виконувати запити для отримання зв'язаних даних.
- Генерація скриптів створення таблиць: Vertabelo автоматично генерував SQL-скрипти для створення таблиць, що спрощує процес реалізації створеної моделі бази даних у реальному середовищі [18].

Цей підхід до проєктування бази даних дозволяє створити структуровану та добре організовану базу даних, що відповідає потребам системи «EcoINheart» та забезпечує зручний доступ до інформації.

Для забезпечення ефективної роботи інформаційної системи «EcoINheart» були створені наступні таблиці:

1. users: зберігає дані про користувачів системи, такі як ім’я, електронна пошта та пароль та кількість ecomoney користувача.
2. bugs: містить інформацію про виявлені проблеми в навколишньому середовищі, такі як опис проблеми, фотографії та геолокація.
3. bonuses: зберігає дані про бонуси, які користувачі можуть отримати за участь у вирішенні проблем.

4. user_messages: містить повідомлення, що надсилаються користувачам системи, такі як сповіщення про статус їх заявок.
5. administrators: зберігає дані про адміністраторів системи, такі як ім'я, електронна пошта та роль.
6. bug_statuses: містить можливі статуси проблем, такі як «В очікуванні», «Вирішено» тощо.
7. discounted_establishments: зберігає дані про заклади, які надають знижки користувачам за участь у програмі.
8. bug_bonuses: встановлює зв'язок багів і бонусів, які користувач отримує за вирішення проблем.

У таблицях бази даних «EcoINheart»використовуються різні типи даних відповідно до призначення кожного поля. Основні типи даних, що використовуються, включають цілі числа (INT), рядки (VARCHAR), дата та час (DATETIME), логічний тип (BOOLEAN) та інші.

Для наочності наведено скріншот з проєктування бази даних у онлайн-інструменті Vertabelo (рис. 2.1).

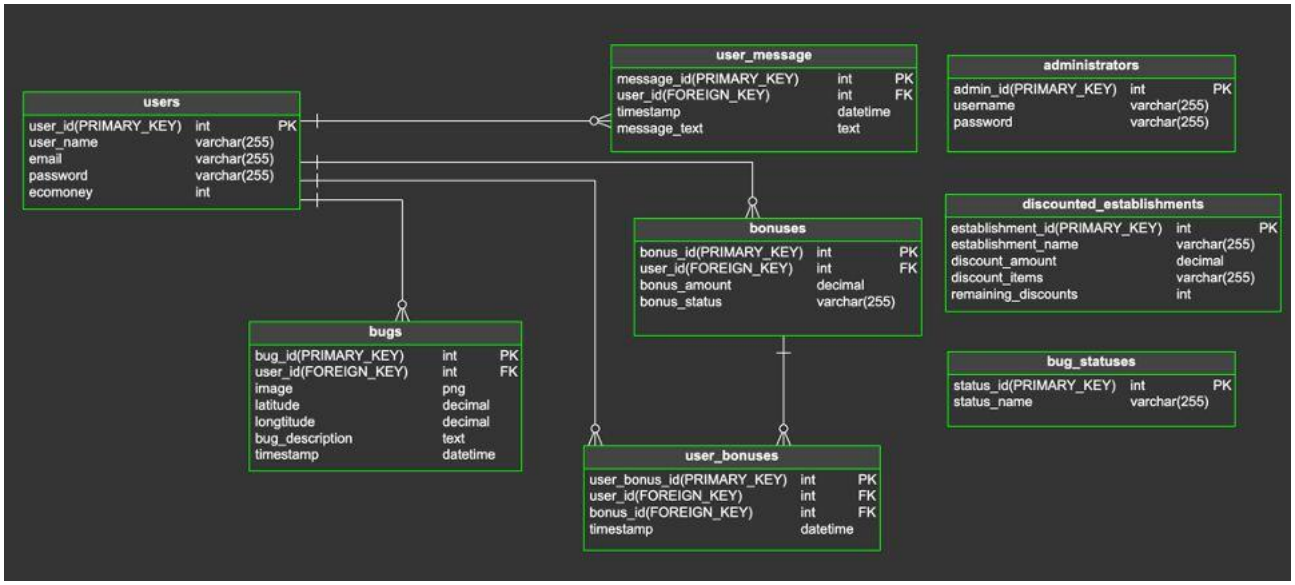

Рисунок 2.1 – Спроєктована схема бази даних

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Прототипування та дизайн

Прототипування сайту – важливий етап у розробці вебпроєктів, який дозволяє створити складний інтерфейс і функціональну модель ще до самої розробки. Основною метою прототипування є визначення структури, взаємодії елементів і навігації сторінки з метою покращення взаємодії з користувачем.

Створення прототипу сторінки дозволяє відтворити основні функції та особливості сторінки, щоб забезпечити належну взаємодію користувача. Він також визначає зручний і логічний спосіб переміщення користувачів по сайту і виявляє можливі проблеми в роботі сайту ще до його запуску [19].

При прототипуванні сайту були використані різні методи та стратегії, спрямовані на створення зручного та ефективного інтерфейсу для користувачів.

Вибір колірної палітри для сайту є надзвичайно важливим, адже кольори мають значний вплив на емоції та сприйняття користувачів. Правильно підібрані кольори можуть покращити користувацький досвід, зробити сайт більш привабливим і зрозумілим, а також підкреслити його фірмовий стиль. Наприклад, теплі кольори створюють атмосферу затишку та доброзичливості, а холодні — відчуття спокою та професіоналізму. Крім того, важливо враховувати психологічні аспекти кольору в різних культурах і групах цільової аудиторії [20].

Для створення мого сайту використовувались різні відтінки рожевого та зеленого кольорів. Ці кольори можуть узгоджуватися разом, оскільки вони створюють приємний контраст і доповнюють один одного. Зелений відтінок символізує природу, свіжість та життєву силу, тоді як рожевий виражає теплоту, ніжність і спокій. Разом вони створюють гармонійне поєднання, що сприймається як приємне для ока та емоційно стимулююче для користувачів.

Отже, вибір кольорів був узгоджений з загальною концепцією сайту, де природні відтінки та пастельні кольори викликають асоціації з екологічністю та природою (рис. 3.1).

Наприклад, використання кольору фону Жонкіль, що нагадує свіжість та чистоту, підкреслює екологічні цінності проекту.

Використання кнопок кольору баклажана для сторінки може бути хорошим рішенням з кількох причин. По-перше, баклажановий колір може додати дизайну концептуальної чіткості та індивідуальності, яка відрізняється від стандартних кольорів кнопок. Він привертає увагу і створює контраст з оточуючими елементами, роблячи кнопки більш помітними і привабливими для користувача. По-друге, колір баклажана може викликати такі емоції, як впевненість і рішучість, що може стати в нагоді для певних дій на сторінці, наприклад, створення звіту про наявний «баг» у місті [21].

Щодо рожевого кольору для забарвлення вікон з повідомленнями на сайті, він може бути ефективним через його асоціації з лагідністю, теплотою і спокоєм. Рожевий колір може створити враження доброзичливості та підтримки, особливо якщо вікна містять інформацію або повідомлення, що потребують уваги або розуміння.

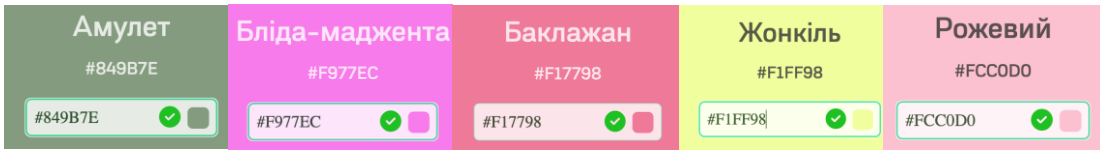

Рисунок 3.1 – Використані кольори на сайті

Щодо шрифтів, було обрано сучасні та легкочитабельні стилі, які забезпечують чітку передачу інформації користувачам. Основні шрифти, які використовуються у моєму сайті – це Kalam, Inter та Jacques Francois (рис. 3.2).

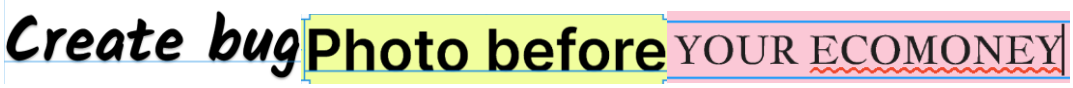

Рисунок 3.2 – Основні шрифти на сайті

Шрифт Kalam має рукописний та екзотичний вигляд, який допомагає відобразити тему природи та екології сайту. Його вигнуті лінії та невеликі

варіації ширини літер надають текстовому вмісту органічності та природності, що підкреслить співпрацю з екологічними цілями.

Шрифт Inter мінімалістичний і універсальний, що робить його ідеальним для веб-сайтів будь-якої тематики, включно з екологічними. Він легко читається на різних пристроях і з різною роздільністю екрана, що підвищує комфорт користувача.

Шрифт Jacques Francois має класичний і елегантний вигляд, який ідеально підходить для для стилізації вебсайту EcoINheart. Це додає дизайну автентичності та вишуканості, що відображатиме серйозність та важливість екологічної тематики [22] .

Однією з важливих задач у прототипуванні сайту є оптимізація розташування різних елементів на сторінці. Правильно сплановане розташування дозволяє досягти кращої зручності для користувача, покращити його взаємодію з сайтом та підвищити конверсію.

– Обговоримо головну сторінку мого сайту (рис. 3.3)

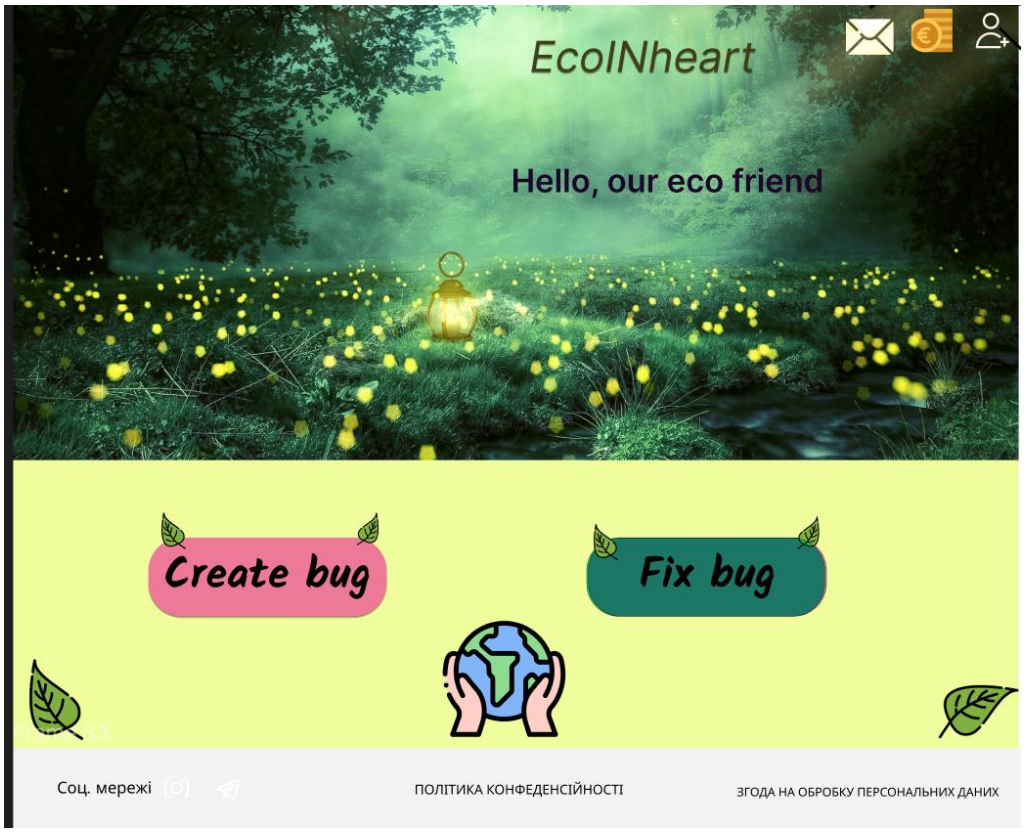

Рисунок 3.3 – Прототип головної сторінки сайту

Головне фото природи розташоване зверху сторінки, щоб здійснити перше враження та привернути увагу відвідувачів до екологічної тематики сайту.

Кнопки «Create Bang» та «Fix Bang» розміщені в центральній частині сторінки, є ключовими елементами, що стимулюють користувачів долучитись до дії та прийняти активну участь у програмі.

Крім того підібрані іконки, які розміщені у верхньому правому кутку сторінки, що символізують вікно з повідомленнями, поточний баланс ecomoney у користувача та форму для авторизації.

– Форма реєстрації (рис. 3.4) розташована на видному місці для зручності використання користувачами та має стандартний вигляд.

Рисунок 3.4 – Прототип форми для реєстрації

– Інформаційні вікна з повідомленнями про кількість ecomoney (рис. 3.5) та про наявні повідомлення для користувача (рис. 3.6) підходять під основний стиль сайту, не відволікають увагу користувача та легкі в розумінні.

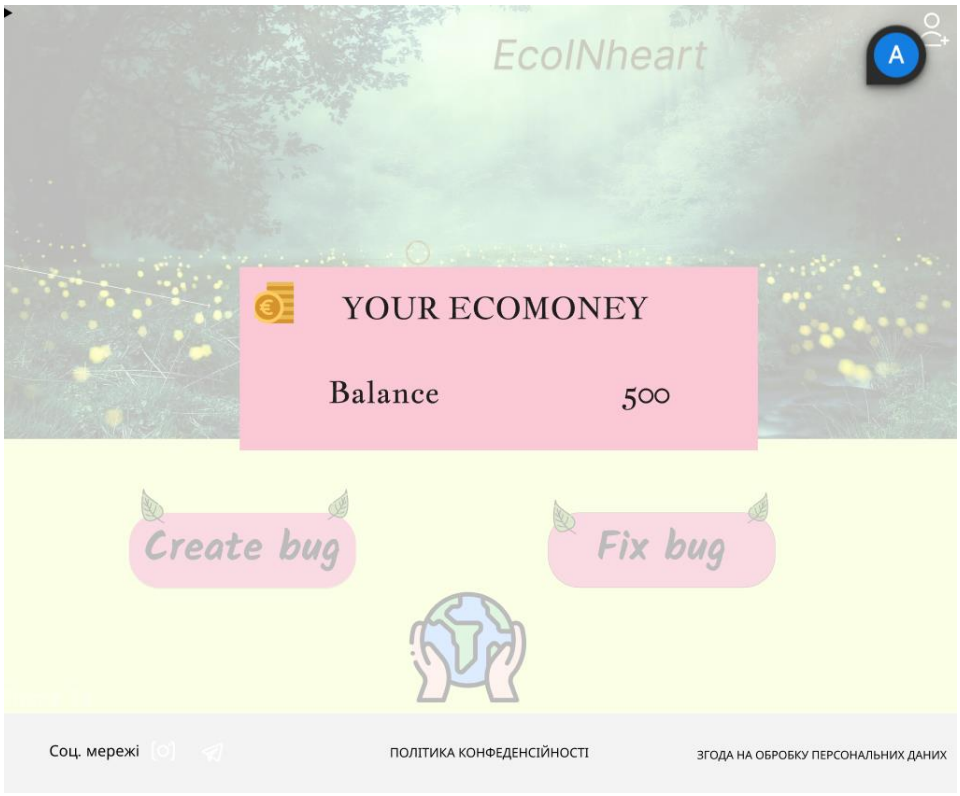

Рисунок 3.5 – Прототип вікна на головній сторінці, яке вміщує інформацію про поточний баланс бонусів

Рисунок 3.6 – Прототип вікна на головній сторінці з повідомленнями

– Сторінка для розміщення багів (рис. 3.6) відкривається при натискання кнопки «Create bug» має простий та лаконічний дизайн. Тут є можливість для розміщення пошти відвідувача сайту, також кнопка для завантаження фото багу і можливість описати його суть. Відразу після пророблених дій та натискання кнопки «Done», що має зручне розташування, користувач побачить вікно з повідомленням (рис. 3.7), де він побачить, що його запит успішно надіслано.

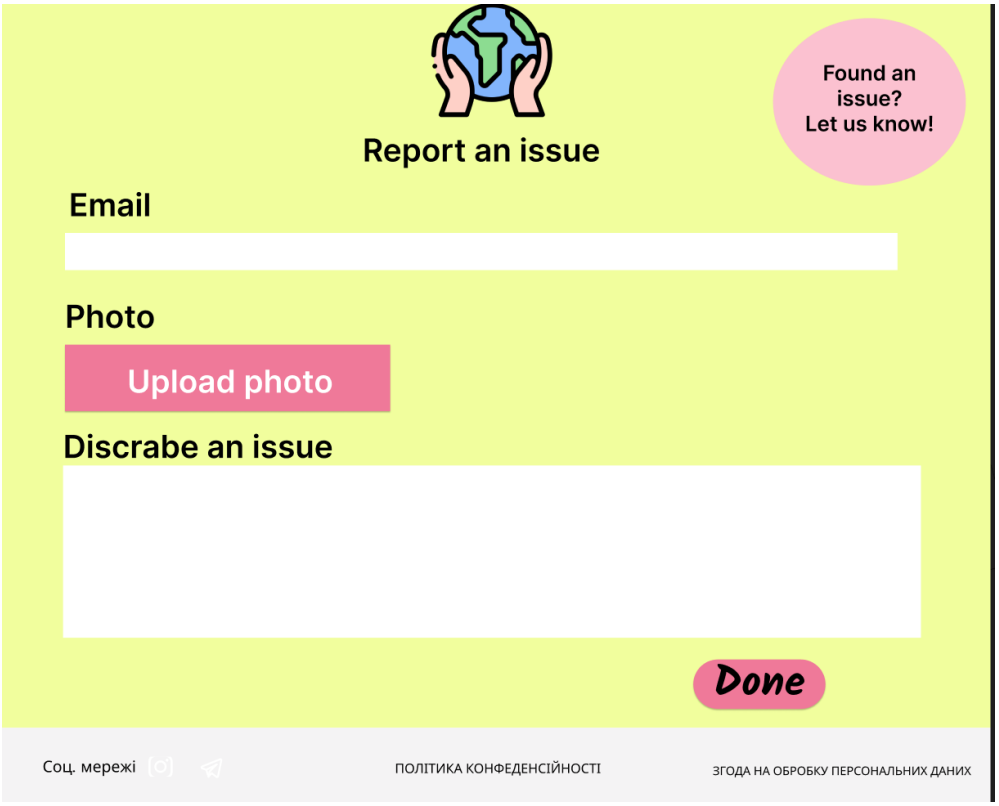

Рисунок 3.6 – Прототип сторінки для повідомлення про баг

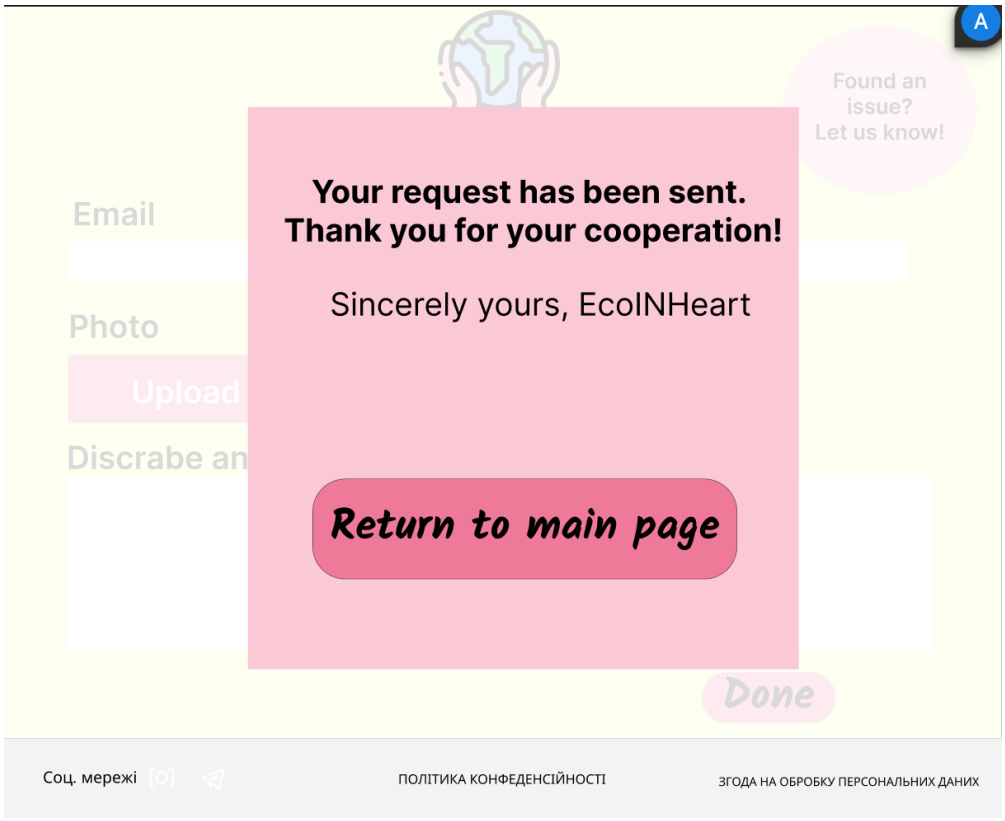

Рисунок 3.7 – Прототип вікна, яке спливає після успішного повідомлення про баг

– Дуже схожий дизайн має і сторінка сайту для виправлення багів (рис. 3.8), яка відкривається при натисканні кнопки «Fix bug», але тут ще є кнопка для завантаження фото після виправлення певного недоліку. Так само наявне вікно з повідомленням (рис. 3.9) про те, що цей запит надіслано до адміністрації сайту і що згодом користувачу будуть нараховані ecomoney.

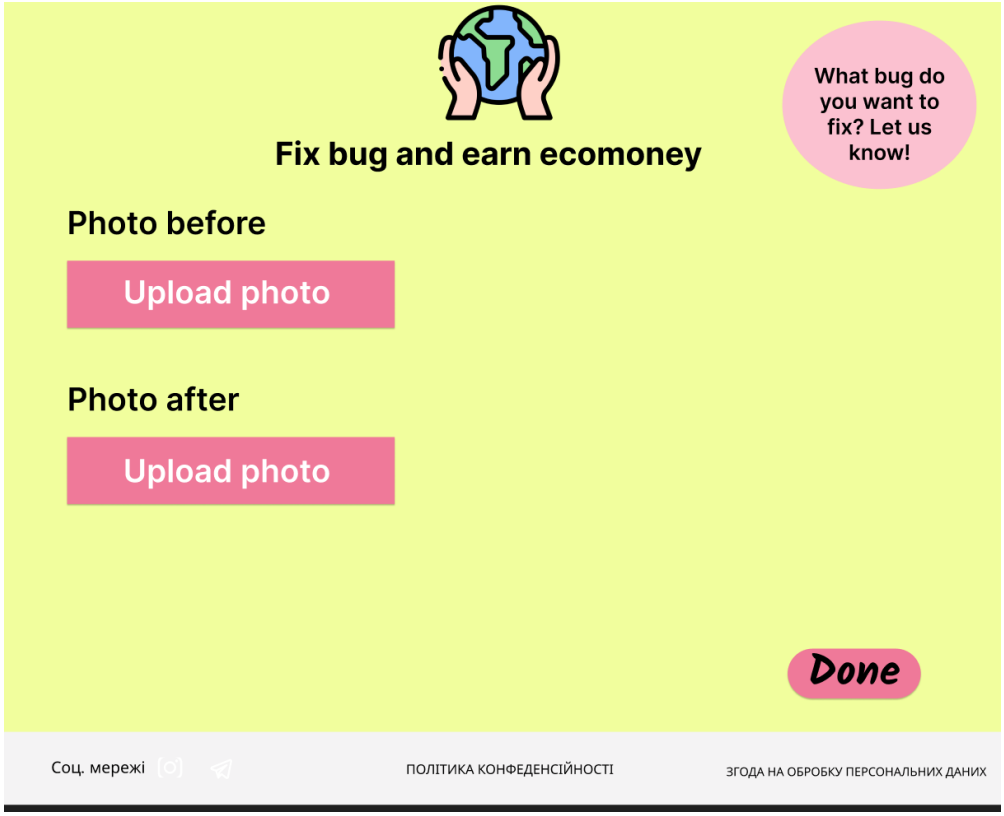

Рисунок 3.8 – Прототип сторінки для виправлення багу

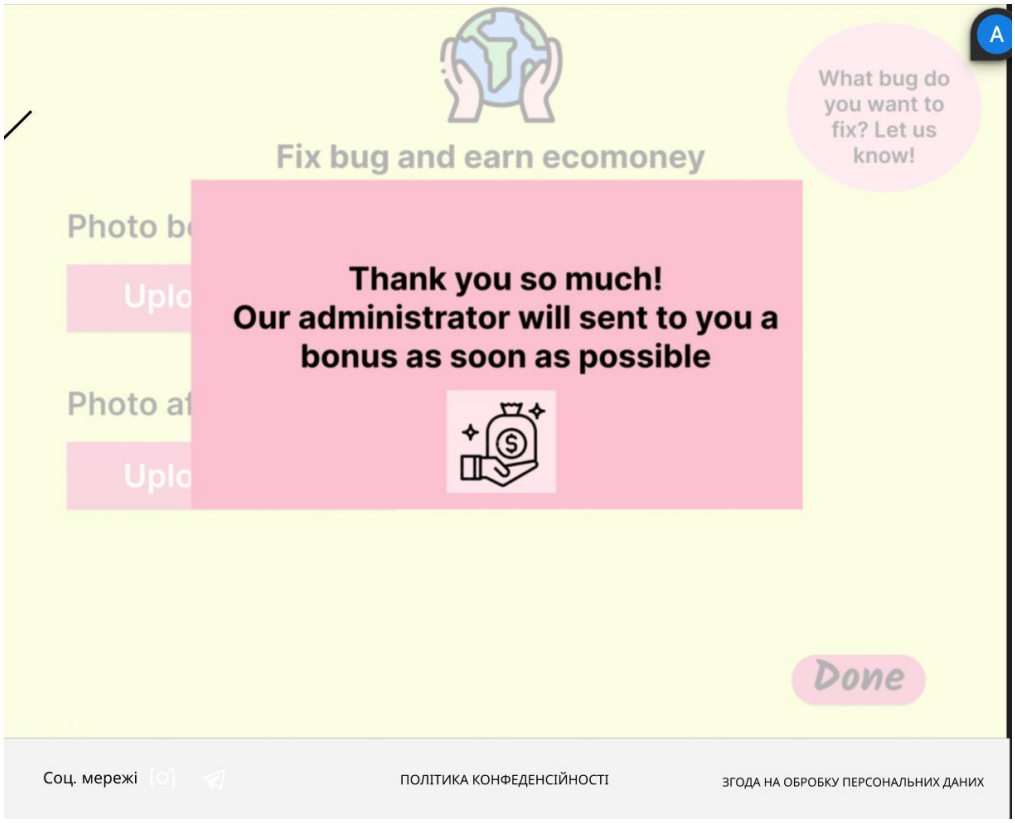

Рисунок 3.9 – Прототип вікна, яке спливає після виправлення багу користувачем

– Також на сайті є сторінка для витрачання зароблених бонусів (рис. 3.10). Вона допомагає користувачу обрати на що підуть його зароблені ecomoney. Представлено список із можливих варіантів і після натискання кнопки «Choose» користувач сайту побачить таке вікно з повідомленням (рис. 3.11) про те, що йому потрібно буде показати повідомлення з Message box, адміністратору кафе чи іншого закладу послуги якого обрав.

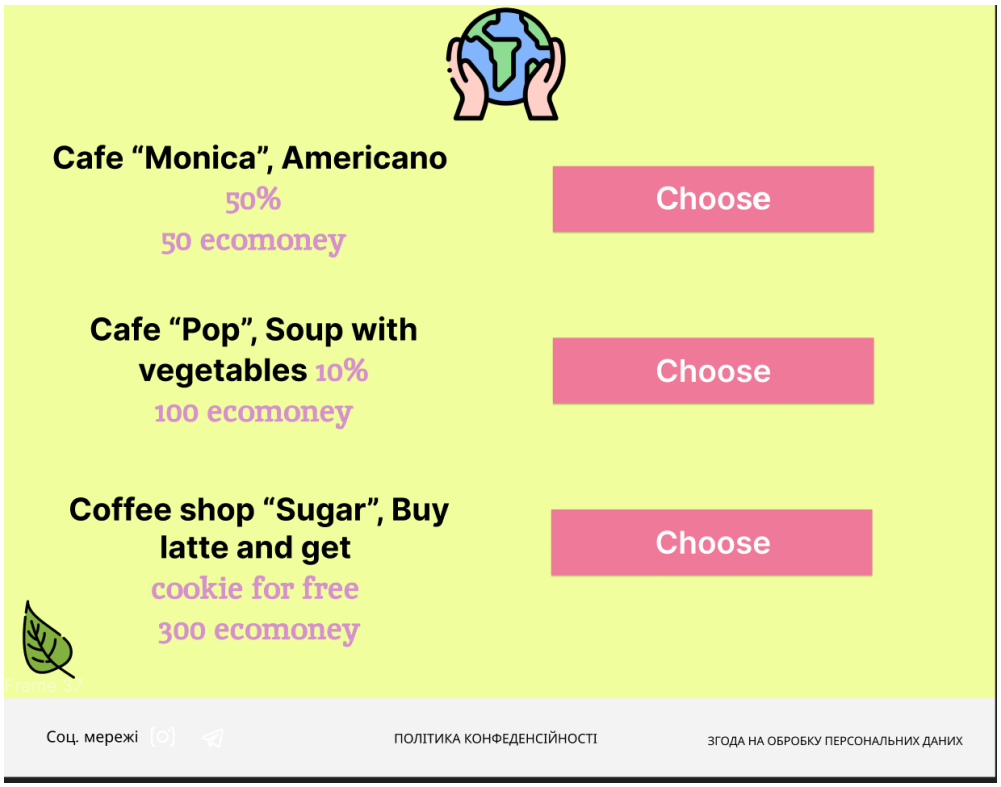

Рисунок 3.10 – Прототип сторінки сайту для витрачення бонусів

Рисунок 3.11 – Прототип повідомлення, яке спливає після витрачення бонусів

У результаті аналізу та прототипування була розроблена модель сайту, яка відображає його функціонал та дизайн. Цей прототип стане важливим етапом у розробці сайту, допомагаючи забезпечити максимальний комфорт та задоволення користувачів.

3.2 Інформаційна модель та структура вебсайту

Веб-сайт - це комплекс веб-сторінок, зображень, анімацій, електронних файлів та різних документів, баз даних, які розміщені у Всесвітній мережі Інтернет. Це структурована кореляційна інформаційна платформа, що має своє оформлення та власний інтерфейс функціонування з даними та інформацією.

Основні елементи веб-сайту:

Веб-сторінки: Це окремі сторінки веб-сайту, доступні через URL-адреси.

Зображення та анімації: Візуальний контент, що покращує користувацький досвід.

Електронні файли та документи: Додаткові ресурси, такі як PDF, Word-документи тощо.

Бази даних: Зберігання динамічного контенту та даних користувачів.

Створення інформаційної моделі для веб-сайту є першим кроком у розробці будь-якого типу веб-сайту. Інформаційна модель складається з основних елементів веб-сторінки. Розміщення та порядок розділів веб-сторінки відображає структуру веб-сайту.

Типи структури веб-сайту:

Внутрішня структура: Включає розробку бізнес-логіки веб-сайту, категорій, послуг тощо. Це стосується організації веб-сайту з боку бекенду, включаючи дизайн бази даних, систему управління контентом (CMS) та серверне програмування.

Зовнішня структура: Це те, що бачить користувач, включаючи дизайн користувацького інтерфейсу (UI), навігаційну структуру та те, як контент подається користувачам [23].

Побудуємо UML-діаграму внутрішньої структури сайту (рис. 3.12). Для цього ми використовуватимемо концепцію UML-діаграми класів для представлення взаємодії між різними компонентами вебсайту EcoINheart. Відобразимо основні функціональні модулі, користувачів та їхні взаємодії з системою.

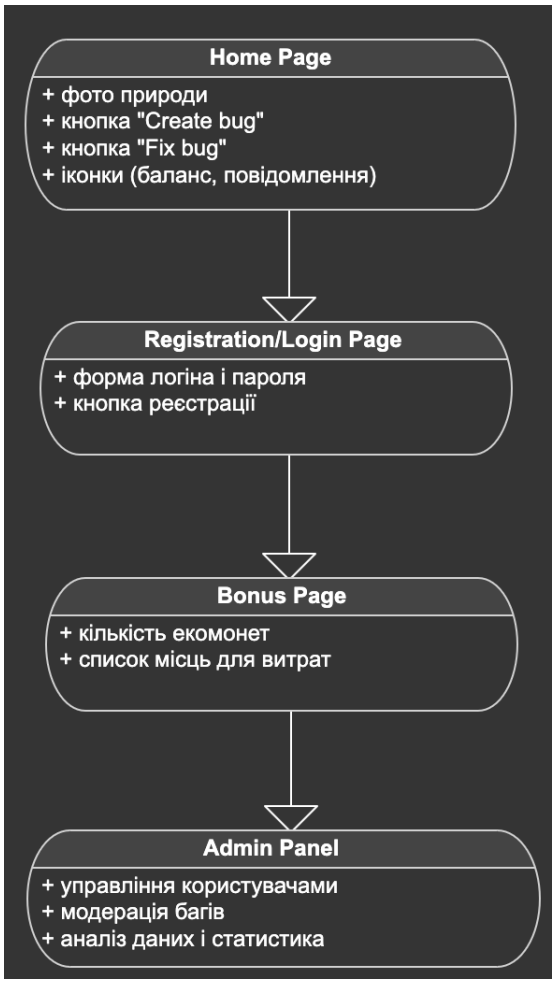

Рисунок 3.12 – UML-діаграма внутрішньої структури сайту

Зовнішня структура сайту відіграє ключову роль у створенні позитивного досвіду користувача, оскільки вона визначає, наскільки зручно та інтуїтивно користувачі зможуть взаємодіяти з веб-сайтом. Він впливає на такі аспекти, як:

• Навігація: добре спланована зовнішня структура допомагає користувачам легко знаходити потрібну інформацію та швидко орієнтуватися на сайті.

• Простота використання: інтуїтивно зрозумілі елементи керування та чітка організація інформації скорочують час, необхідний користувачам для виконання певних дій.

• Естетичне враження: візуальний дизайн і презентація впливають на сприйняття користувачами бренду чи організації, до якої належить сайт [23].

Сайт EcoINheart розроблений з акцентом на інтуїтивно зрозумілий інтерфейс і привабливий візуальний дизайн, що сприяє залученню користувачів до активної участі в покращенні міського середовища.

Головна сторінка

Головна сторінка є центральним елементом сайту, яка привертає увагу користувачів і забезпечує доступ до основних функцій платформи.

1) Фонове зображення природи:

Розташоване зверху сторінки, створюючи сприятливу атмосферу та підкреслюючи екологічну спрямованість платформи.

2) Кнопки "Create Bug" та "Fix Bug":

Розташовані в центрі сторінки нижче фонової фотографії.

Кнопка "Create Bug" має колір #F977EC, що робить її яскравою і помітною, спонукаючи користувачів повідомляти про проблеми.

Кнопка "Fix Bug" має колір #15C867, що символізує вирішення проблем і позитивні дії.

3) Інформаційні іконки:

Іконка з монеткою показує баланс екомонет.

Іконка з конвертом вказує на нові повідомлення.

4) Навігаційне меню

Навігаційне меню забезпечує зручний доступ до різних розділів сайту:

5) Авторизація та реєстрація:

Користувачі можуть швидко авторизуватися або зареєструватися, використовуючи відповідні кнопки.

Основний контент

Основний контент сайту містить різні елементи, що забезпечують взаємодію користувачів із платформою:

1) Інформаційні вікна:

Містять повідомлення про кількість екомонет та де їх можна витратити.

Користувачі можуть переглядати детальну інформацію про нараховані бонуси та їх використання.

2) Сторінка зі списком екомонет:

Окрема сторінка, де відображається детальний список нарахованих екомонет та їх використання.

3) Сторінка для повідомлень про баги:

Користувачі можуть завантажувати фотографії багів, давати доступ до їх геолокації та надавати опис проблеми.

Підвал (Footer)

Підвал містить додаткову інформацію та навігаційні посилання:

1) Соціальні іконки:

Посилання на сторінки в соціальних мережах, таких як Facebook та Twitter.

2) Політика конфіденційності

3.3 Налаштування взаємодії з базою даних

Для забезпечення коректної роботи з базою даних, необхідно створити таблиці баз даних в файлі **models.py** (рис. 3.15) і в файлі **settings.py** (рис. 3.16) налаштувати підключення до бази даних **SQLite.**

```python
class Role(models.Model):
    name = models.CharField(max_length=50)
    description = models.TextField()

    def __str__(self):
        return self.name

class ProUser(AbstractUser):
    role = models.ForeignKey(Role, on_delete=models.PROTECT, default=1, blank=True)
    invitation_code = models.CharField(max_length=10, unique=True, null=True, blank=True)
```

Рисунок 3.15 – Конфігураційний файл models.py

```python
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Рисунок 3.16 – Конфігураційний файл з settings.py

Після підключення до БД також потрібно створити моделі за концептуальною моделлю бази даних (Додаток А).

3.4 Механізми авторизації та реєстрації

Важливим компонентом в розробці інформаційних систем є авторизація та реєстрація. У Django, механізми авторизації та реєстрації вбудовані і легко налаштовуються за допомогою вбудованих модулів, таких як **django.contrib.auth** і **django.contrib.auth.views**.

1. **Авторизація (логін)**:
    - Користувач вводить свої дані (ім'я користувача або email та пароль) на сторінці, призначеній для входу.
    - Django перевіряє введені дані. Якщо вони вірні, користувач авторизується і отримує сеансовий токен.
    - Сеансовий токен зберігається в куках браузера або може передаватися в HTTP-заголовку. Він ідентифікує авторизованого користувача.
2. **Реєстрація (створення облікового запису)**:
    - Користувач вводить необхідну інформацію (ім'я, email, пароль тощо) на сторінці реєстрації.
    - Django перевіряє ці дані (наприклад, чи унікальний введений email) і, якщо все в порядку, створює новий обліковий запис користувача.
    - Після успішної реєстрації користувач може авторизуватися в системі.

Для реалізації авторизації та реєстрації в Django використовуються вбудовані класи та функції, такі як **User** модель (для представлення користувачів), **LoginView** та **LogoutView** (для сторінок входу та виходу) та **UserCreationForm** (для форми реєстрації).

У веборієнтованій інформаційній системі EcoINheart механізми авторизації та реєстрації мають такий вигляд (рис. 3.17-3.18).

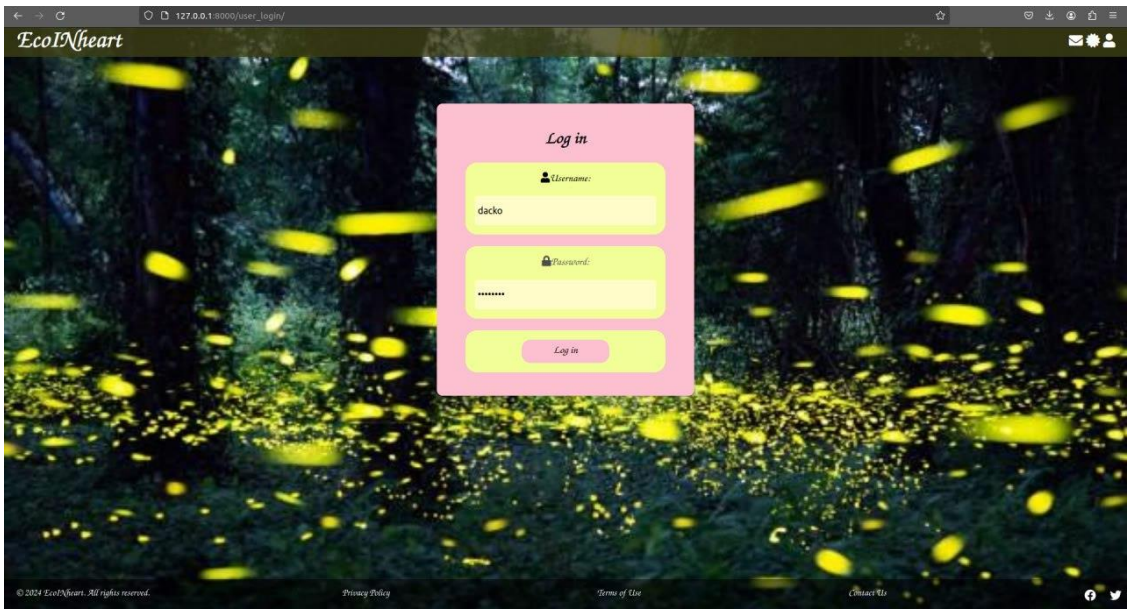

Рисунок 3.17 – Процес авторизації

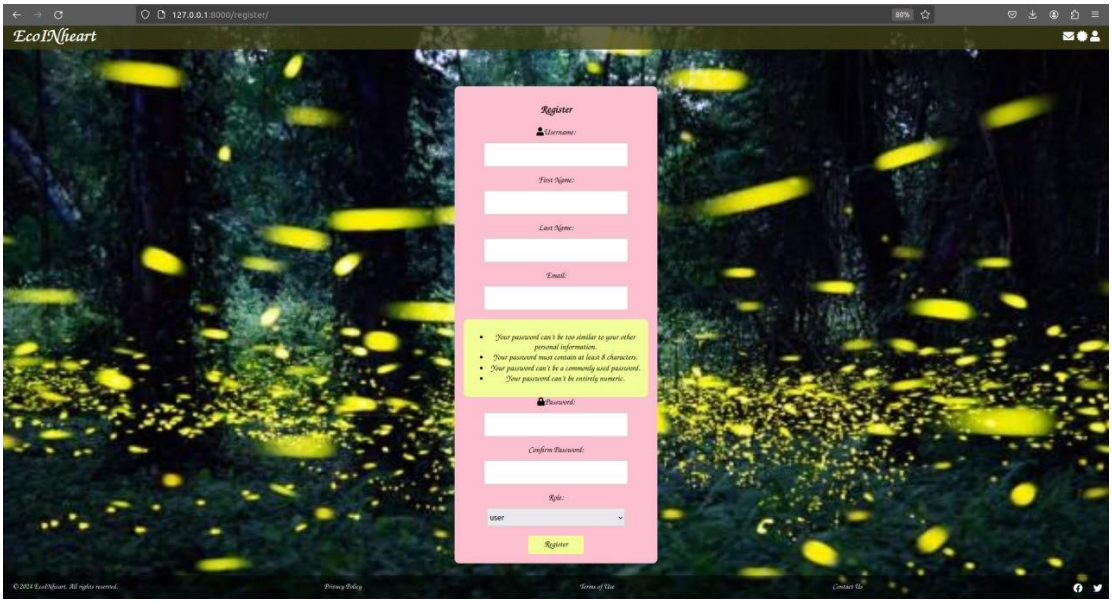

Рисунок 3.18 – Процес реєстрації

3.5 Тестування інформаційної системи

Перед тим, як випустити будь-яку інформаційну систему або додаток, необхідно провести відповідне тестування, щоб забезпечити передачу замовнику готового продукту та зберегти репутацію компанії в майбутньому.

У веборієнтованій інформаційній системі EcoINheart ми будемо тестувати наступні компоненти: авторизація, реєстрація, процес повідомлення про баг, процес виправлення багу.

Спочатку користувач, який зайшов на сайт потрапляє на стартову сторінку (рис. 3.19).

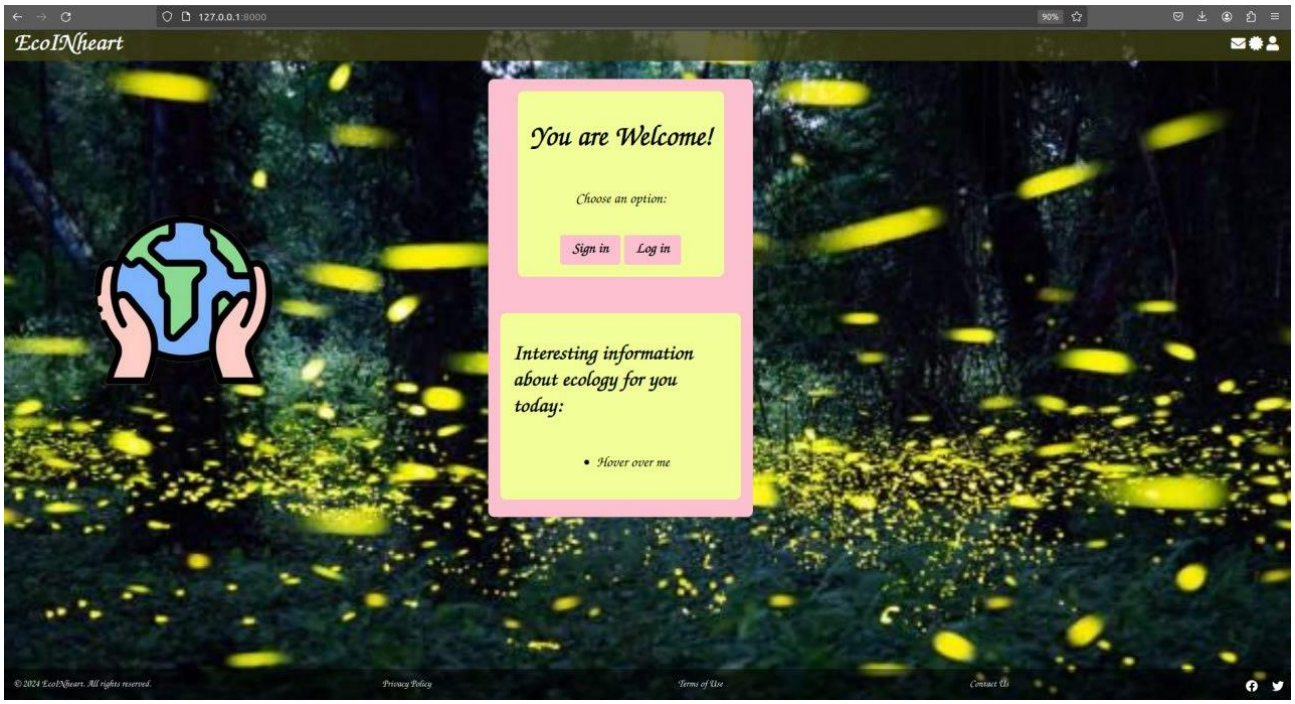

Рисунок 3.19 – Реакція бота на недопустимі дані

Далі він може або зареєструватись в системі або ввійти в уже створений ним акаунт. При натисканні кнопки «Sign in» користувач повинен потрапити на сторінку для реєстрації (рис. 3.20).

Перевіряються такі стани:

1) Електронна пошта повинна складатися з англійських літер, містити знак @ і крапку після нього. Пошта має бути новою і не збігатися з уже існуючою.

2) Пароль повинен містити не менше 8 символів, як мінімум 1 цифру та 1 велику літеру.

У випадку порожніх полів або некоректного формату, то при спробі залогінитись або зареєструватись ми перенаправляємось на ту саму сторінку з пустими полями для введення.

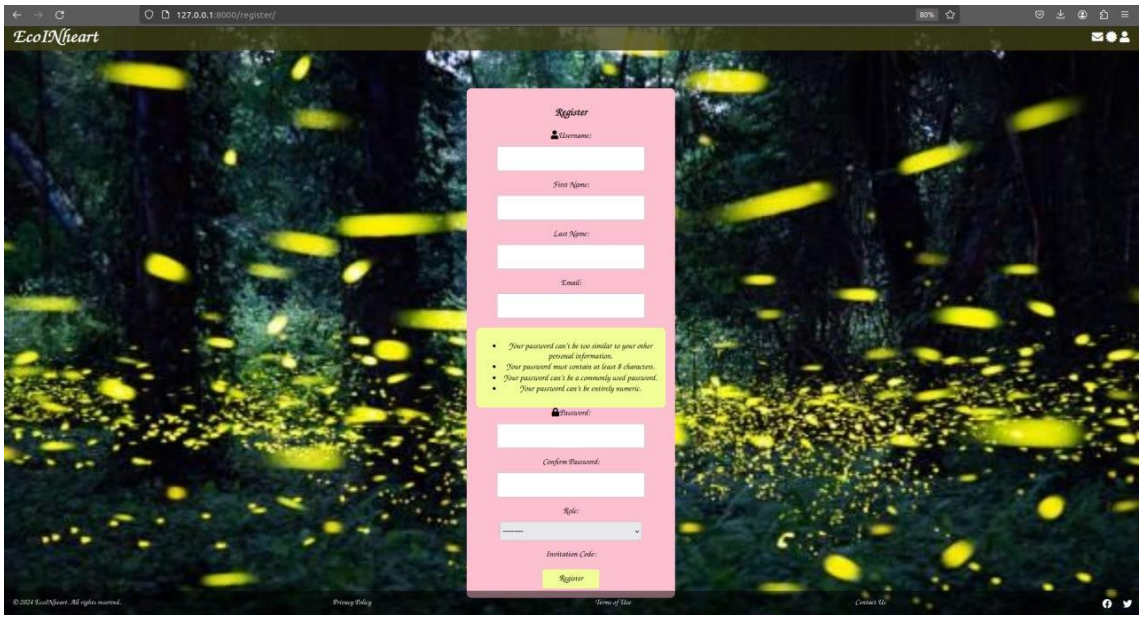

Рисунок 3.20 – Перевірка сторінки для реєстрації

Як бачимо тут є дві ролі Користувач та Адмін, для другої потрібно отримати спеціальний код-запрошення, який буде надсилатись на вказану пошту, якщо ж юзер обирає першу роль, то поле з кодом-запрошенням зникає (рис. 3.21).

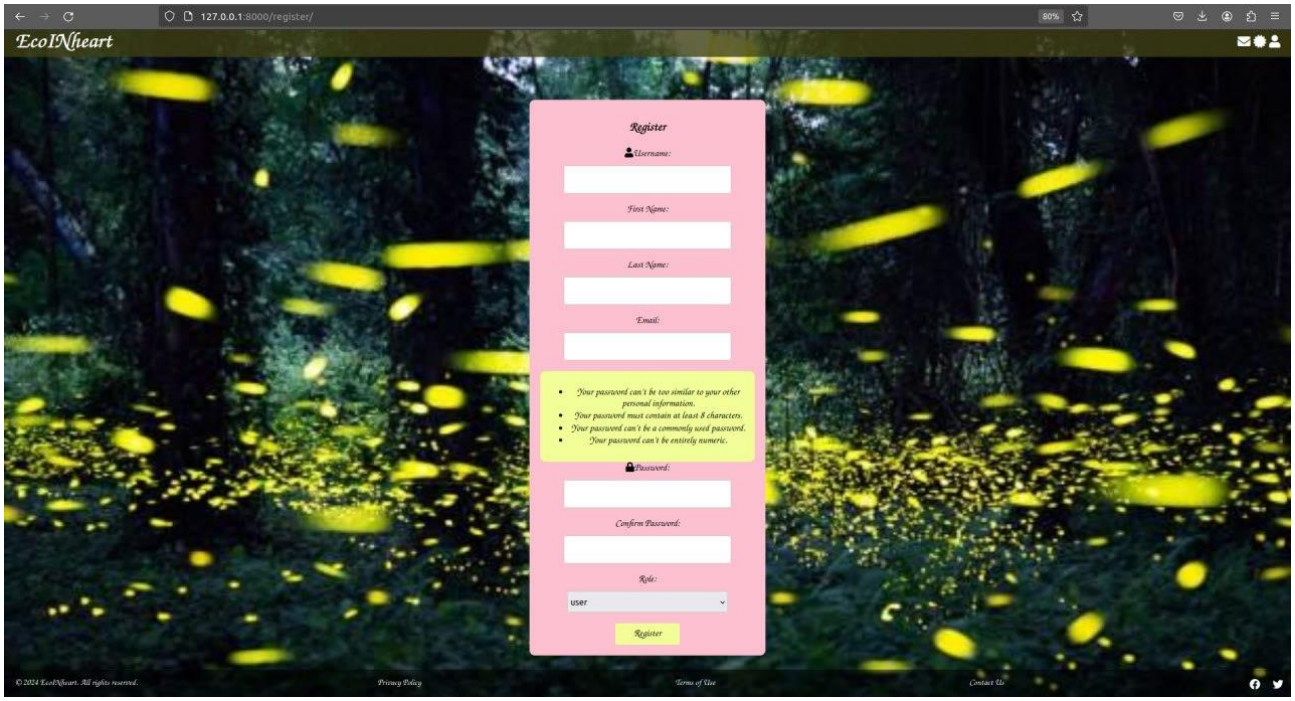

Рисунок 3.21 – Перевірка сторінки для реєстрації (зміна ролі)

Наступним компонентом є авторизація (рис. 3.22). Тут користувач вводить своє ім’я та пароль, при введенні некоректних даних аналогічно до системи при реєстрації він буде перенаправлятись на ту саму сторінку з пустими полями для введення. Якщо все введено коректно, то користувач натискає кнопку «Log in» та переходить на головну сторінку (рис. 3.23).

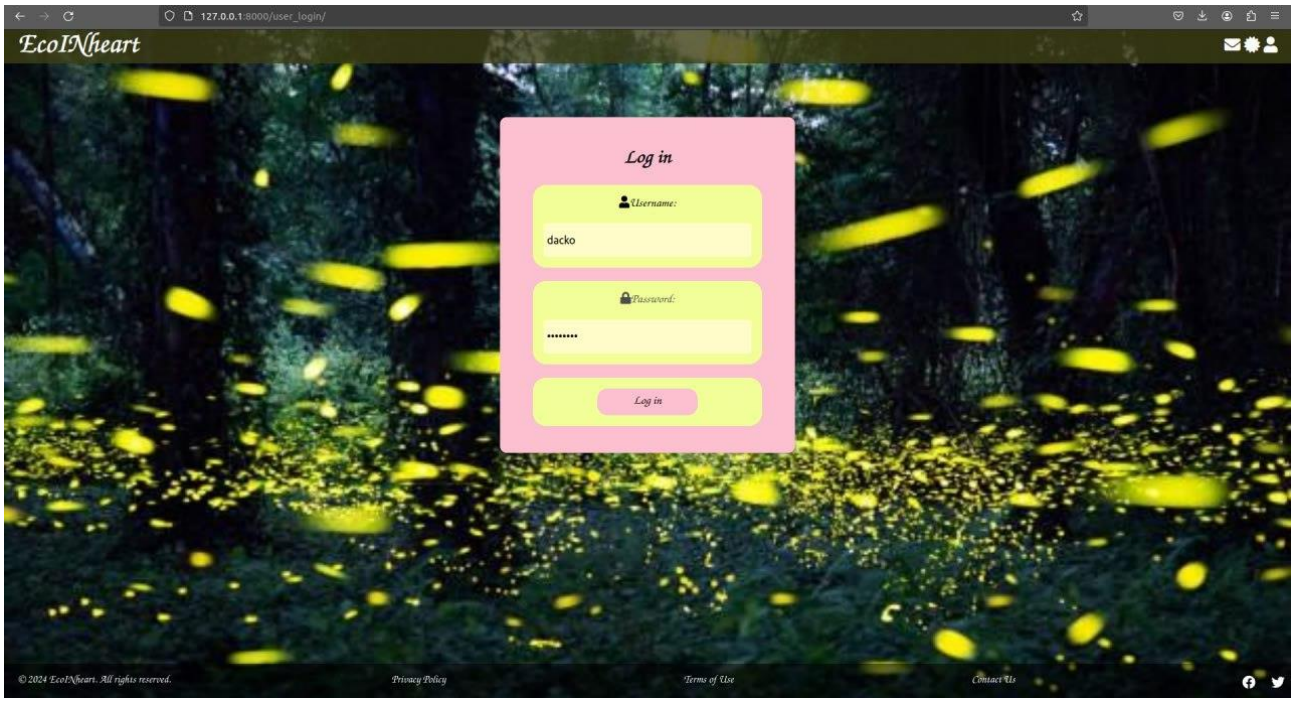

Рисунок 3.22 – Авторизація з валідними даними

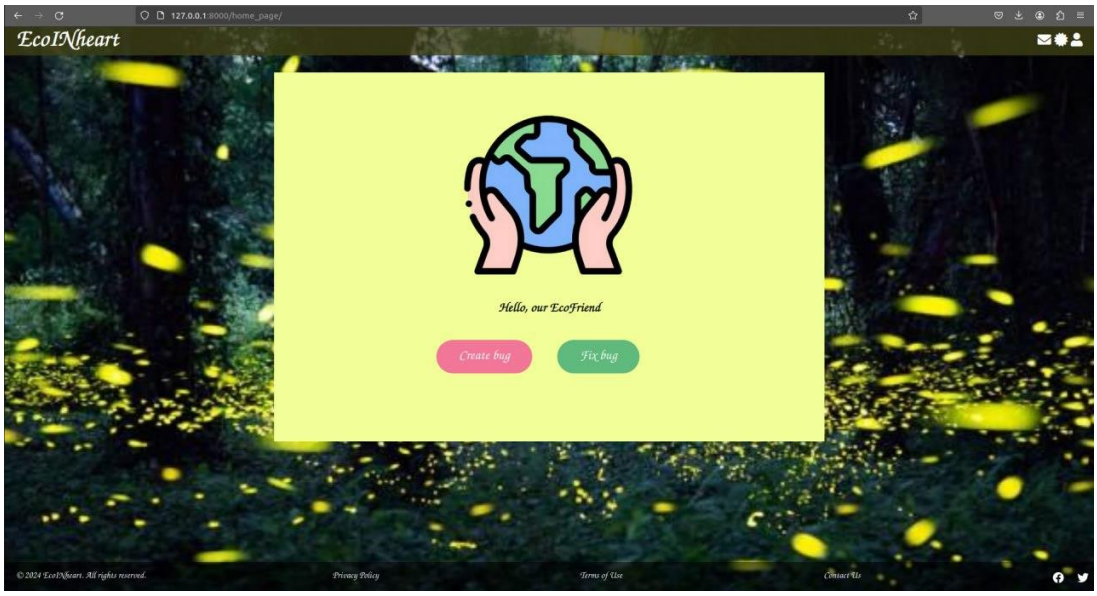

Рисунок 3.23 – Головна сторінка сайту

Тепер користувач потрапляє на головну сторінку і може звідси перейти на сторінку створення багу (рис. 3.24) та виправлення багу (рис. 3.25) за допомогою відповідних кнопок.

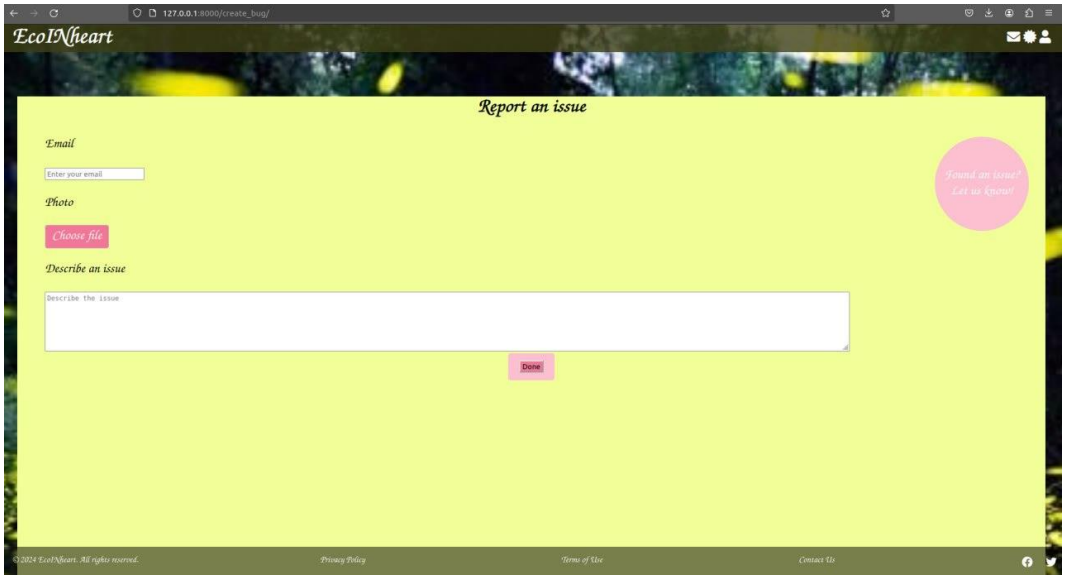

Рисунок 3.24 – Сторінка для створення багу

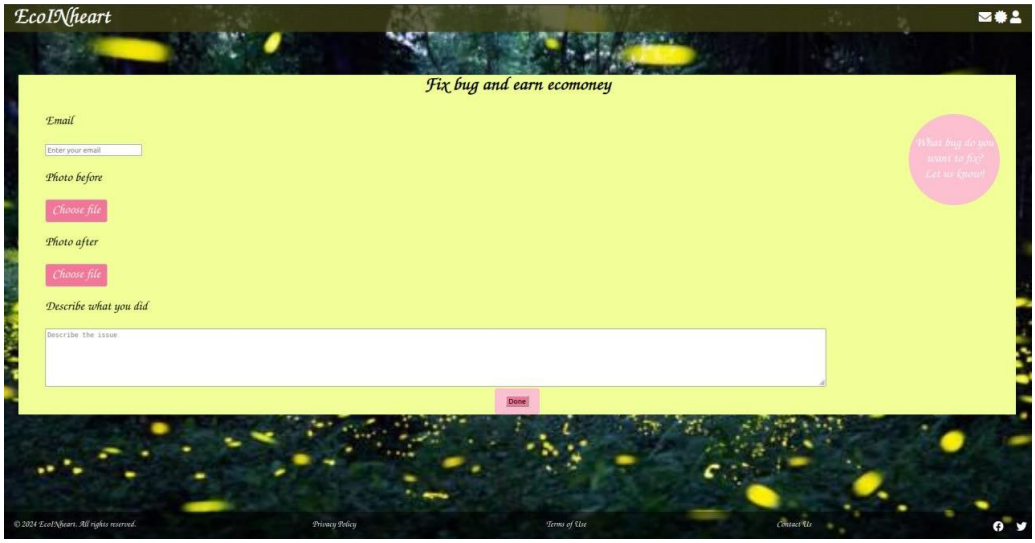

Рисунок 3.25 – Сторінка для виправлення багу

При натисканні на кнопку «Done» в цих двух формах запитується дозвіл на поширення геолокації (рис. 3.26), якщо користувач дозволяє це зробити, то бачить перед собою таку віконну форму (рис. 3.27).

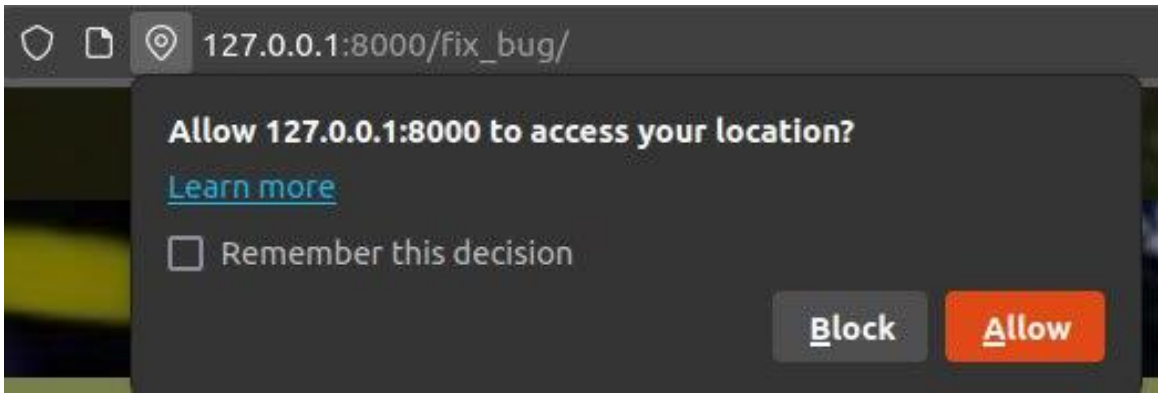

Рисунок 3.26 – Запит на дозвіл для поширення геолокації

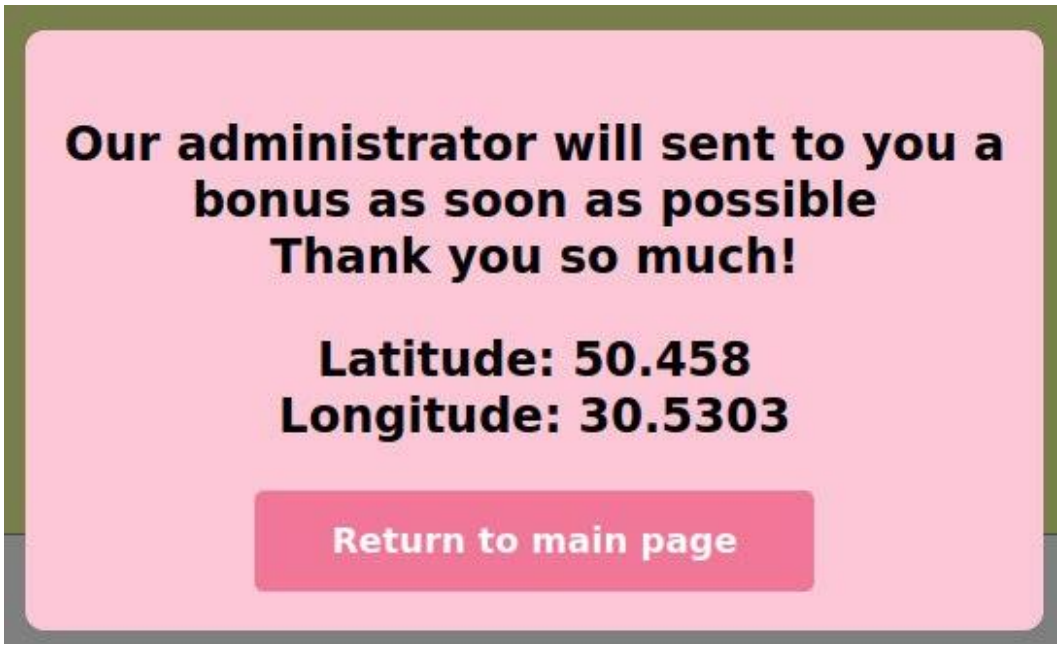

Рисунок 3.27 – Вікно з повідомленням про успішне завантаження фото

При натисканні на кнопку «Return to page» юзера повертає на головну сторінку і на сервер відправляється репорт.

На головній сторінці при натисканні на іконку з людиною юзер може переглядати свій власний профіль (рис. 3.28). Тут він може виходити з профіль, видаляти його та редагувати (рис. 3.29).

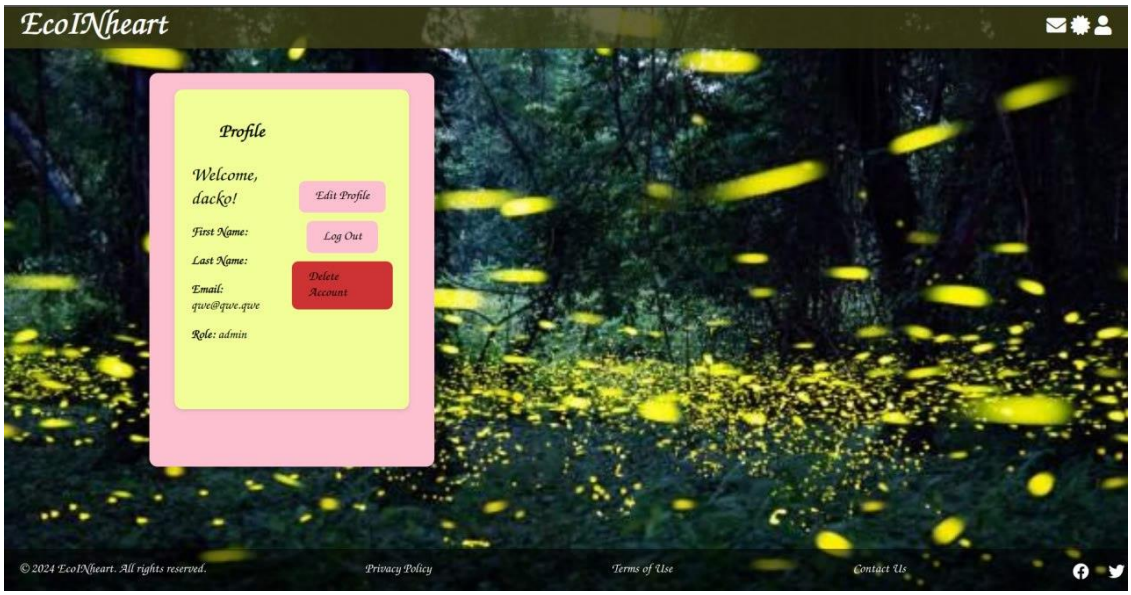

Рисунок 3.28 – Профіль користувача

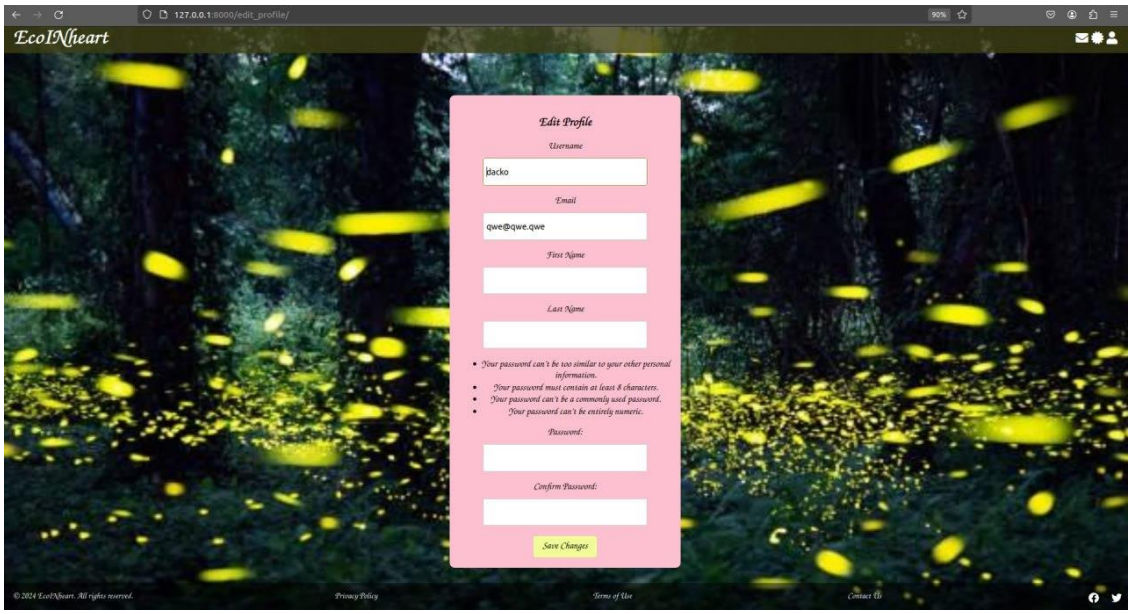

Рисунок 3.29 – Вікно з редагуванням профілю для користувача

Натиснувши на іконку монетки, користувач зможе потрапити на сторінку, де він витратить свої зароблені ecomoney (рис. 3.30).

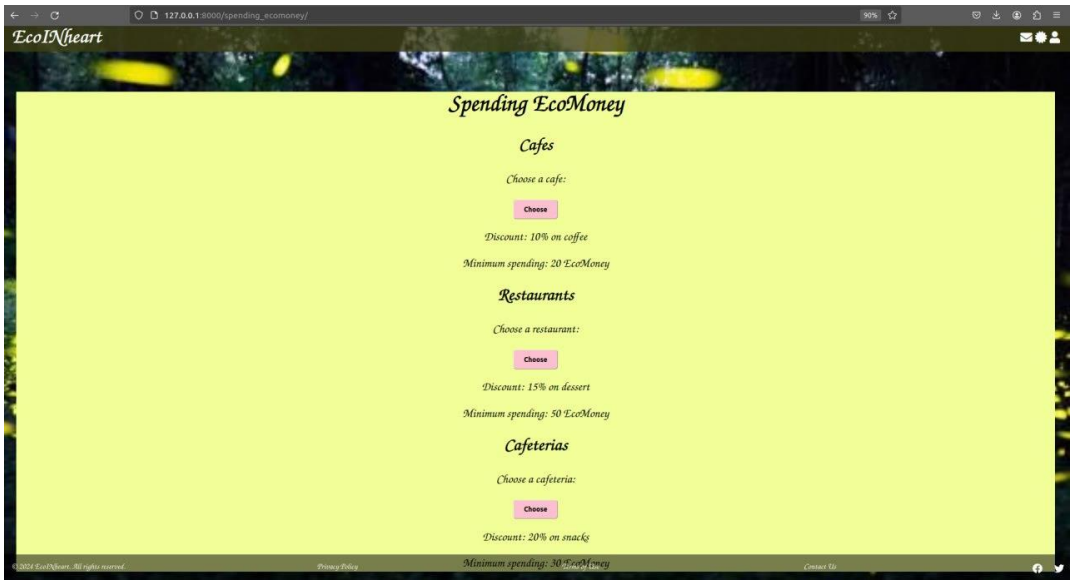

Рисунок 3.30 – Сторінка для витрачення бонусів

3.6 Алгоритм роботи системи

Авторизований користувач

- Він може створювати заявки про проблеми у місті за допомогою кнопки «Create bug».
- Може фіксити баги скориставшись кнопкою «Fix bug».
- Користувач може додавати фотографії багів та вказувати їх геолокацію.
- Він може переглядати статус своїх заявок та отримувати сповіщення про їх стан.
- Може отримувати ecomoney за виправлені баги та використовувати їх для знижок у різних закладах.

Адміністратор

- Адміністратор має всі можливості авторизованого користувача.
- Він має додаткові функції керування та моніторингу системи.
- Адміністратор може переглядати та змінювати дані користувачів та їх заявки.
- Він відповідає за модерацію та вирішення проблем, які надходять від користувачів.

ВИСНОВКИ

За результатами аналізу та розробки платформи EcoINheart можна зробити висновок про її значний потенціал у вирішенні екологічних проблем та покращенні якості життя в місті. Платформа EcoINheart спрямована на створення спільної мережі, яка допоможе зробити місто чистішим і забезпечить здоровіше та комфортніше середовище для всіх його мешканців.

В рамках кваліфікаційної роботи були виконані наступні завдання:

1. Аналіз актуальності даної роботи: проведено аналіз актуальності та виявлено, що дана тема є актуальною, оскільки спрямована на вирішення важливої соціальної проблеми управління та моніторингу за навколишнім середовищем міста.

2. Розробка концепції платформи EcoINheart: розроблено концепцію та функціональні вимоги платформи EcoINheart, щоб залучити мешканців до захисту довкілля та зробити місто більш зеленим та екологічно безпечним.

3. Вибір програмних засобів для вирішення мета: Вибрано програмні засоби, база даних, фреймфорк, середовище для прототипування та дизайну, які допомогли в створення сайту.

4. Прототипування та дизайн: у Figma створено прототип платформи, включаючи дизайн інтерфейсу, розташування елементів та функціональність платформи.

5. Проєктування бази даних: за допомогою Vertabelo спроєктовано базу даних мого сайту.

6. Налаштування авторизації та реєстрації користувачів: налаштований механізм авторизації та реєстрації для користувачів.

7. Впровадження та тестування: проведено функціональне впровадження платформи та її тестування з метою виявлення та усунення можливих проблем і недоліків.

У перспективі планується:

1. Впровадження та підтримка: планується впровадження платформи EcoINheart для широкого кола користувачів, а також підтримка та подальший розвиток системи з метою постійного підвищення її функціональності та зносостійкості.

2. Створення гаманця для користувача, аби там він зберігав свої ecomoney.

Порівняно з іншими подібними проектами EcoINheart має низку переваг, які роблять його найбільш привабливим та ефективним для користувачів. Система бонусів та заохочень стимулює активну участь мешканців у процесі збору та вирішення проблем міста. Зручний та інтуїтивно зрозумілий інтерфейс, можливість авторизації та реєстрації, а також широкі можливості взаємодії з іншими користувачами та органами місцевого самоврядування роблять EcoINheart найбільш прогресивним та популярним активним проектом у сфері екологічного моніторингу та управління містом.

Цей висновок підтверджується аналізом даних, отриманих під час використання форми Google, і відгуками користувачів, а також з різних джерел, включаючи офіційні сайти, документацію та відгуки.

Тому під час виконання кваліфікаційної роботи бакалавра була створена платформа EcoINheart, яка допоможе зробити місто чистішим та забезпечить здоровіше та комфортніше середовище для всіх його мешканців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vivek Kulkarni. The Role of Information Technology in Business Success | PDF | Internet | Cloud Computing [Electronic resource]. URL: https://www.scribd.com/document/436996853/The-Role-of-Information-Technology-in-Business-Success?language_settings_changed=English (дата доступу: 28.04.2024).
2. Monitoring and assessments | UNEP - UN Environment Programme [Electronic resource]. 2020. URL: https://www.unep.org/topics/air/monitoring-and-assessments (дата доступу: 28.04.2024).
3. George Wyeth L.C.P. The Impact of Citizen Environmental Science in the United States - CORE Reader [Electronic resource]. 2019. URL: https://core.ac.uk/reader/232646900 (дата доступу: 28.04.2024).
4. Відкрите Місто [Electronic resource]. URL: https://opencity.e-dem.ua/ (дата доступу: 28.04.2024).
5. Django documentation | Django documentation | Django [Electronic resource]. 2022. URL: https://docs.djangoproject.com/en/5.0/ (дата доступу: 28.04.2024).
6. Welcome to Flask — Flask Documentation (2.1.x) [Electronic resource]. 2022. URL: https://flask.palletsprojects.com/en/2.1.x/ (дата доступу: 28.04.2024).
7. The Pyramid Web Framework — The Pyramid Web Framework v2.0.2 [Electronic resource]. 2022. URL: https://docs.pylonsproject.org/projects/pyramid/en/2.0-branch/ (дата доступу: 28.04.2024).
8. Vincent W.S. Django for beginners : build websites with Python & Django. 335 p.
9. MySQL Documentation [Electronic resource]. URL: https://docs.oracle.com/cd/E17952_01/index.html (дата доступу: 28.04.2024).
10. PostgreSQL: Documentation: 9.1: psql [Electronic resource]. URL: https://www.postgresql.org/docs/9.1/app-psql.html (дата доступу: 28.04.2024).

11. MongoDB Atlas: Cloud Document Database | MongoDB [Electronic resource]. URL: https://www.mongodb.com/lp/cloud/atlas/try4?utm_content=controldbaasterms&utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_core_prosp-brand_gic-null_emea-pl_ps-all_desktop_eng_lead&utm_term=mongocloud&utm_medium=cpc_paid_search&utm_ad=p&utm_ad_campaign_id=12212624548&adgroup=115749720663&cq_cmp=12212624548&gad_source=1&gclid=CjwKCAjw57exBhAsEiwAaIxaZgdmaTwLSzLoMEpTd-CPWGHWiX4tqFOUXZelNrZlo89d9_8s_c3MVBoCAxMQAvD_BwE (дата доступу: 28.04.2024).
12. Osman A. et al. Paper Prototyping as a Rapid Participatory Design Technique // Computer and Information Science. Canadian Center of Science and Education, 2009. Vol. 2, № 3.
13. Lloyd W.J. et al. Tools and Techniques for Effective Distributed Requirements Engineering: An Empirical Study. 2001.
14. Welcome to the XD User Guide [Electronic resource]. URL: https://helpx.adobe.com/xd/user-guide.html (дата доступу: 28.04.2024).
15. Figma Learn - Help Center [Electronic resource]. URL: https://help.figma.com/hc/en-us (дата доступу: 28.04.2024).
16. Figma vs. Sketch vs. Adobe XD: Which Design Tool Is Better? | Coursera [Electronic resource]. URL: https://www.coursera.org/articles/figma-vs-sketch-vs-adobe-xd (дата доступу: 28.04.2024).
17. new release – Mendeley Blog [Electronic resource]. URL: https://blog.mendeley.com/category/new-release-2/ (дата доступу: 28.04.2024).
18. Vertabelo Database Modeler [Electronic resource]. URL: https://vertabelo.com/documentation/ (дата доступу: 28.04.2024).

19. Прототипування сайту: де і як зробити правильний прототип - WebTune [Electronic resource]. URL: https://webtune.com.ua/statti/dyzajn/prototypuvannya-sajtu/ (дата доступу: 28.04.2024).
20. Веб-дизайн складається з багатьох ключів, колірна палітра — один з них [Electronic resource]. URL: https://coi.ua/blog/DesignCo/color-palette-in-web-design-selection-psychology-and-trends/ (дата доступу: 28.04.2024).
21. Правильні поєднання та підбір кольору для сайту: рекомендації для новачків - Блог Impulse-design [Electronic resource]. URL: https://impulse-design.com.ua/ua/vybor-tsveta-dlya-sajta.html (дата доступу: 28.04.2024).
22. Browse Fonts - Google Fonts [Electronic resource]. URL: https://fonts.google.com/ (дата доступу: 28.04.2024).
23. Веб-дизайн. Інформаційна структура сайту [Electronic resource]. URL: https://webstudio2u.net/ua/design-web/443-information-structure.html (дата доступу: 20.05.2023).
24. Вікторія Драч. (2021). *Створення правильної структури сайту* https://project-seo.net/blog-uk/ (дата доступу: 20.05.2023).

ДОДАТОК А
МОДЕЛІ БАЗИ ДАНИХ

Файл models.py

```
from django.db import models

from django.contrib.auth.models import AbstractUser, Group, Permission

from django.db import migrations

from django.contrib.auth import get_user_model

from django.utils import timezone

from datetime import datetime

import secrets

from django.db.models import Sum


class Role(models.Model):

    name = models.CharField(max_length=50)

    description = models.TextField()


    def __str__(self):

        return self.name

class ProUser(AbstractUser):

    role = models.ForeignKey(Role, on_delete=models.PROTECT, default=2,
blank=True, related_name='pro_users')

    invitation_code = models.CharField(max_length=10, unique=True,
null=True, blank=True)

    # Переопределение обратной связи для групп
    groups = models.ManyToManyField(Group, verbose_name='groups',
blank=True, related_name='pro_user_groups')
    user_permissions = models.ManyToManyField(Permission,
verbose_name='user permissions', blank=True,

related_name='pro_user_permissions')

class User(AbstractUser):
    role = models.ForeignKey(Role, on_delete=models.PROTECT,
default=1, blank=True, related_name='normal_users')

    # Переопределение обратной связи для групп
    groups = models.ManyToManyField(Group, verbose_name='groups',
blank=True, related_name='normal_user_groups')
    user_permissions = models.ManyToManyField(Permission,
verbose_name='user permissions', blank=True,
```

```
related_name='normal_user_permissions')

class BugReport(models.Model):
    user = models.ForeignKey(ProUser, on_delete=models.CASCADE)
    email = models.EmailField()
    description = models.TextField()
    photo_before = models.ImageField(upload_to='bug_reports/')
    photo_after = models.ImageField(upload_to='bug_reports/',
blank=True, null=True)
    latitude = models.FloatField()
    longitude = models.FloatField()
    created_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return f'Bug Report by {self.user.username}'

    class Meta:
        verbose_name = 'Bug Report'
        verbose_name_plural = 'Bug Reports'
```

ДОДАТОК Б
КОД ПРОГРАМИ

Серверна частина

Файл admin.py

```
from .models import ProUser, BugReport, User
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import Group


class ProUserAdmin(UserAdmin):
    add_form = UserCreationForm
    model = ProUser
    list_display = ('username', 'email', 'is_staff')


    fieldsets = (
        (None, {'fields': ('username', 'password')}),
        ('Personal Info', {'fields': ('email', 'role', 'invitation_code')}),
        ('Permissions', {'fields': ('is_staff', 'is_superuser', 'groups')}),
    )


    add_fieldsets = (
        (None, {
            'classes': ('wide',),
            'fields': ('username', 'email', 'password1', 'password2', 'role',
'invitation_code', 'is_staff', 'is_superuser', 'groups')}
        ),
    )


admin.site.register(ProUser, ProUserAdmin)
admin.site.register(User)
admin.site.unregister(Group)
admin.site.register(BugReport)
```

Файл apps.py

```
from django.apps import AppConfig
```

```
class MainAppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'main_app'
```

Файл forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm,AuthenticationForm
from .models import Role, ProUser, Message, BloodRequest,Invitation, Donation, User
from django_select2.forms import Select2Widget


class InvitationForm(forms.ModelForm):
    class Meta:
        model = Invitation
        fields = ['email']
    def __init__(self, *args, **kwargs):
        self.user = kwargs.pop('user', None)  # Отримайте user з параметрів конструктора
        super().__init__(*args, **kwargs)


    def clean(self):
        cleaned_data = super().clean()
        if self.user.role.name != 'admin':
            raise forms.ValidationError("You don't have permission to add a donor.")
        return cleaned_data


class ProUserForm(UserCreationForm):
    email = forms.EmailField(required=True)
    role = forms.ModelChoiceField(queryset=Role.objects.all())
    invitation_code = forms.CharField(required=True)


    class Meta:
        model = ProUser
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2', 'role', 'invitation_code')


    def save(self, commit=True):
        user = super(ProUserForm, self).save(commit=False)
```

```
        user.email = self.cleaned_data['email']
        user.invitation_code = self.cleaned_data['invitation_code']
        if commit:
            user.save()
        return user

    class Meta:
        model = ProUser
        fields = ('username', 'first_name', 'last_name', 'email', 'password1',
'password2', 'role')


class UserForm(UserCreationForm):
    email = forms.EmailField(required=True)
    role = forms.ModelChoiceField(queryset=Role.objects.all())

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1',
'password2', 'role')

    def save(self, commit=True):
        user = super(UserForm, self).save(commit=False)
        user.email = self.cleaned_data['email']
        if commit:
            user.save()
        return user


class LoginForm(AuthenticationForm):
    # Додаткові поля, якщо потрібно

    # Приклад зміни мітки поля "username"
    username = forms.CharField(label='Ім\'я користувача')

    # Приклад додаткового поля
   # my_field = forms.CharField(label='Моє поле')
```

Файл signals.py

```
from django.apps import AppConfig


class MyappConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'myapp'


    def ready(self):
        import myapp.signals
```

Файл urls.py

```
from django.urls import path
from . import views
from django.contrib.auth import views as auth_views


urlpatterns = [
    path('', views.home, name='home'),
    path('home_page/', views.home_page, name='home_page'),
    path('fix_bug/', views.fix_bug, name='fix_bug'),
    path('create_bug/', views.create_bug, name='create_bug'),
    path('spending_ecomoney/', views.spending_ecomoney,
name='spending_ecomoney'),
    path('accounts/login/',
auth_views.LoginView.as_view(redirect_authenticated_user='profile'),
name='profile'),
    path('logout/', views.logout_user, name='logout'),
    path('register/', views.register, name='register'),
    path('profile/', views.profile, name='profile'),
    path('create_invitation/', views.create_invitation,
name='create_invitation'),
    path('invitation_success/', views.invitation_success,
name='invitation_success'),
    path('edit_profile/', views.edit_profile, name='edit_profile'),
    path('delete_profile/', views.delete_profile, name='delete_profile'),
    path('user_login/', views.user_login, name='user_login'),
    path('messages/', views.message_list, name='message_list'),]
```

Шаблони сторінок

all.css

```
body {
    margin: 0;
    padding: 0;
    color: #000; /* Колір тексту */
    font-family: "Kalam", cursive;
    background-color: #dde5f4; /* Set the background color */
    background-image: url('nature.jpg'); /* Set the background image */
    background-repeat: no-repeat; /* Prevent the background image from repeating */
    background-size: cover; /* Adjust the background image size to cover the entire
body */
}

main {
    padding: 30px;
}
```

fix_bug.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<div class="bug-main">
  <div class="container">
    <h2>Fix bug and earn ecomoney</h2>
    <div class="circle">
      <p>What bug do you want to fix?<br />Let us know!</p>
    </div>
    <form
      method="POST"
      action="{% url 'fix_bug' %}"
      enctype="multipart/form-data"
    >
      {% csrf_token %}
      <div class="email">
        <p>Email</p>
        <input type="email" name="email" placeholder="Enter your email" />
      </div>
      <div class="photo">
        <p>Photo before</p>
```

```
    <label for="photo_before" class="custom-file-upload">
      <span id="file-label-text-before">Choose file</span>
      <span id="file-icon-before" style="display: none">✔ </span>
    </label>
    <input
      id="photo_before"
      type="file"
      name="photo_before"
      accept="image/*"
      style="display: none"
    />
  </div>
  <div class="photo">
    <p>Photo after</p>
    <label for="photo_after" class="custom-file-upload">
      <span id="file-label-text-after">Choose file</span>
      <span id="file-icon-after" style="display: none">✔ </span>
    </label>
    <input
      id="photo_after"
      type="file"
      name="photo_after"
      accept="image/*"
      style="display: none"
    />
  </div>

  <div class="description">
    <p>Describe what you did</p>
    <textarea name="description" placeholder="Describe the issue"></textarea>
  </div>
  <div class="button">
    <button
      type="button"
      id="doneButton"
      style="background-color: #f17798; color: #000000"
    >
      Done
    </button>
  </div>
  <!-- Модальне вікно -->
  <div class="overlay" id="overlay"></div>
  <div class="modal" id="modal">
```

```
      <p>
        Our administrator will sent to you a<br />bonus as soon as possible<br />Thank
        you so much!
      </p>
      <p id="geolocation"></p>
      <input type="hidden" id="latitudeInput" name="latitude" />
      <input type="hidden" id="longitudeInput" name="longitude" />
      <button type="submit" id="returnButton">Return to main page</button>
    </div>
  </form>
 </div>
</div>

<script>
 document
  .getElementById("photo_before")
  .addEventListener("change", function () {
   var fileLabelText = document.getElementById("file-label-text-before");
   var fileIcon = document.getElementById("file-icon-before");
   if (this.files.length > 0) {
    fileLabelText.textContent = this.files[0].name;
    fileIcon.style.display = "inline";
   } else {
    fileLabelText.textContent = "Choose file";
    fileIcon.style.display = "none";
   }
  });
  document
  .getElementById("photo_after")
  .addEventListener("change", function () {
   var fileLabelText = document.getElementById("file-label-text-after");
   var fileIcon = document.getElementById("file-icon-after");
   if (this.files.length > 0) {
    fileLabelText.textContent = this.files[0].name;
    fileIcon.style.display = "inline";
   } else {
    fileLabelText.textContent = "Choose file";
    fileIcon.style.display = "none";
   }
  });
 function closeModalAndRedirect(url) {
  closeModal();
```

```
  window.location.href = url;
}
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    document.getElementById("geolocation").innerHTML =
      "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  document.getElementById("geolocation").innerHTML =
    "Latitude: " + latitude + "<br>Longitude: " + longitude;


  document.getElementById("latitudeInput").value = latitude;
  document.getElementById("longitudeInput").value = longitude;
}

function openModal() {
  getLocation();
  document.getElementById("overlay").style.display = "block";
  document.getElementById("modal").style.display = "block";
}

function closeModal() {
  document.getElementById("overlay").style.display = "none";
  document.getElementById("modal").style.display = "none";
}

function returnToMainPage() {
  closeModal();
  window.location.href = "{% url 'home_page' %}";
}
document
  .getElementById("returnButton")
  .addEventListener("click", function () {
    returnToMainPage();
  });
document.getElementById("doneButton").addEventListener("click", function () {
```

```
    openModal();
  });

  var map = L.map("map").setView([49.842957, 24.031111], 13);

  L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
    maxZoom: 19,
  }).addTo(map);

  var marker = L.marker([49.842957, 24.031111], { draggable: true }).addTo(map);

  marker.on("dragend", function (event) {
    var position = marker.getLatLng();
    console.log(position);
  });
</script>
{% endblock %}
```

index.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}

<div class="ekran" style="flex-direction: column;">
    <div class="hero">
      <img src="{% static 'logo.png' %}" class="profile-image" alt="Profile Image"
style="margin: 0px">
      <h2>Hello, our EcoFriend</h2>
    </div>
    <div class="buttons">
      <button
        id="createBugButton"
        class="create-bug"
        onclick="window.location.href = {% url 'create_bug' %};"
      >
        Create bug
      </button>
      <button
        class="fix-bug"
        onclick="window.location.href = {% url 'fix_bug' %};"
      >
```

```
      Fix bug
    </button>

  <div class="hero">
  </div>
  </div>
</div>
{% endblock %}
```

register.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
 <link rel="stylesheet" href="{% static 'register.css' %}">
 <div class="registration-container">
  <h2 class="registration-title">Register</h2>
  <form method="post">
   {% csrf_token %}
   <div class="form-group">
    <label for="id_username"><i class="fas fa-user"></i>Username:</label>
    {{ form.username }}
   </div>
   <div class="form-group">
    <label for="id_first_name">First Name:</label>
    {{ form.first_name }}
   </div>
   <div class="form-group">
    <label for="id_last_name">Last Name:</label>
    {{ form.last_name }}
   </div>
   <div class="form-group">
    <label for="id_email">Email:</label>
    {{ form.email }}
   </div>
   <div class="form-group">
    <div class="rules-container">
     <span class="password-help-text">{{ form.password1.help_text }}</span>
    </div>
    <label for="id_password1"><i class="fas fa-lock"></i>Password:</label>
    {{ form.password1 }}
   </div>
   <div class="form-group">
```

```
      <label for="id_password2">Confirm Password:</label>
      {{ form.password2 }}
    </div>
    <div class="form-group">
      <label for="id_role">Role:</label>
      {{ form.role }}
    </div>
    <div class="form-group" id="invitation-code-group">
      <label for="id_invitation_code">Invitation Code:</label>
      {{ form.invitation_code }}
    </div>
    <div class="button-container">
      <button type="submit">Register</button>
    </div>
  </form>
 </div>
 <script>
  document.addEventListener('DOMContentLoaded', function() {
    var roleSelect = document.getElementById('id_role');
    var invitationCodeGroup = document.getElementById('invitation-code-group');

    roleSelect.addEventListener('change', function() {
      if (roleSelect.value === "1") {
        invitationCodeGroup.style.display = 'none';
      } else {
        invitationCodeGroup.style.display = 'block';
      }
    });
  });
 </script>
{% endblock %}
```

edit_profile.html

```
<!-- edit_profile.html -->
{% extends 'base.html' %}
{% load static %}

{% block content %}
 <link rel="stylesheet" href="{% static 'register.css' %}">

 <div class="registration-container">
```

```
  <h2 class="registration-title">Edit Profile</h2>

  <form method="post">
   {% csrf_token %}
   <div class="form-group">
    <label for="{{ form.username.id_for_label }}">Username</label>
    {{ form.username }}
   </div>

   <div class="form-group">
    <label for="{{ form.email.id_for_label }}">Email</label>
    {{ form.email }}
   </div>

   <div class="form-group">
    <label for="{{ form.first_name.id_for_label }}">First Name</label>
    {{ form.first_name }}
   </div>

   <div class="form-group">
    <label for="{{ form.last_name.id_for_label }}">Last Name</label>
    {{ form.last_name }}
   </div>
   <div class="form-group">

    <span class="password-help-text">{{ form.password1.help_text }}</span>
    <label for="id_password1">Password:</label>
    {{ form.password1 }}
   </div>
   <div class="form-group">
    <label for="id_password2">Confirm Password:</label>
    {{ form.password2 }}
   </div>
   <div class="button-container">
    <button type="submit">Save Changes</button>
   </div>
  </form>
 </div>
{% endblock %}
```

profile.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}

<link rel="stylesheet" href="{% static 'profile.css' %}">
<div class="ekran">
   <div class="profile-container">
      <div class="profile-details">
         <div class="profile-info">
            <h2>Profile</h2>

            <p class="welcome-msg">Welcome, {{ user.username }}!</p>
            <p><strong>First Name:</strong> {{ user.first_name }}</p>
            <p><strong>Last Name:</strong> {{ user.last_name }}</p>
            <p><strong>Email:</strong> {{ user.email }}</p>
            <p><strong>Role:</strong> {{ user.role }}</p>
         </div>
         <div class="profile-actions">
            <div class="action-box">
               <a href="{% url 'edit_profile' %}" class="action-link">Edit Profile</a>
            </div>
            <div class="action-box">
               <a href="{% url 'logout' %}" class="action-link">Log Out</a>
            </div>
            <div class="action-box">
               <a href="{% url 'delete_profile' %}" class="action-link delete-
link">Delete Account</a>
            </div>
         </div>
      </div>
   </div>
</div>
{% endblock %}
```

spending_ecomoney.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<div class="bug-main">
   <div class="container">
      <h1>Spending EcoMoney</h1>
      <div class="list-item">
         <h2>Cafes</h2>
         <p>Choose a cafe:</p>
         <button class="button">Choose</button>
         <div class="discount-info">
            <p>Discount: 10% on coffee</p>
            <p>Minimum spending: 20 EcoMoney</p>
         </div>
      </div>
      <div class="list-item">
         <h2>Restaurants</h2>
         <p>Choose a restaurant:</p>
         <button class="button">Choose</button>
         <div class="discount-info">
            <p>Discount: 15% on dessert</p>
            <p>Minimum spending: 50 EcoMoney</p>
         </div>
      </div>
      <div class="list-item">
         <h2>Cafeterias</h2>
         <p>Choose a cafeteria:</p>
         <button class="button">Choose</button>
         <div class="discount-info">
            <p>Discount: 20% on snacks</p>
            <p>Minimum spending: 30 EcoMoney</p>
         </div>
      </div>
   </div>
</div>
{% endblock %}
```

home.html

```
{% extends 'base.html' %}
{% load static %}
<!--    <link rel="stylesheet" href="{% static 'styles.css' %}">-->
```

```
<!--    <link rel="stylesheet" href="{% static 'all.css' %}">-->
{% block content %}
   <div class="ekran">
   <img src="{% static 'logo.png' %}" class="profile-image" alt="Profile Image">
   <div class="profile-container">
      <div class="container interesting-facts" style="margin-top: 0px;">
      <h1>You are Welcome!</h1>
      <p>Choose an option:</p>
      <div class="buttons-home">
         <a class="button" href="{% url 'register' %}">Sign in</a>
         <a class="button" href="{% url 'user_login' %}">Log in</a>
      </div>
      </div>
   <div class="interesting-facts">
      <h2>Interesting information about ecology for you today:</h2>
      <ul>
         <li class="fact" id="initial-fact">Hover over me</li>
         <li class="fact hidden">Добавить 1</li>
         <li class="fact hidden">Добавить 2</li>
         <li class="fact hidden">Добавить 3</li>
         <!-- Додайте інші цікаві факти про екологію -->
      </ul>
   </div>

   <script>
      window.addEventListener('DOMContentLoaded', function() {
         var initialFact = document.querySelector('#initial-fact');

         initialFact.addEventListener('mouseover', function() {
            initialFact.style.display = 'none';
            var hiddenFacts = document.querySelectorAll('.fact.hidden');
            if (hiddenFacts.length > 0) {
               var randomIndex = Math.floor(Math.random() * hiddenFacts.length);
               hiddenFacts[randomIndex].classList.remove('hidden');
            }
         });
      });
   </script>

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

style.css

```
header {
  background-color: rgba(63, 64, 20, 0.8);
  background-size: cover;
  background-position: center;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 20px;
}

.logo h1 {
  color: #ffffff; /* Колір тексту логотипу */
  font-size: 36px; /* Збільшити розмір шрифту */
  font-style: italic; /* Наклонний шрифт */
  margin: 0;
}

.toolbar-link {
  display: inline-block;
  text-decoration: none;
  color: #ffffff;
  border-radius: 8px;
}

.icons {
  color: #fff; /* Колір іконок */
  font-size: 24px;
}

.hero {
  background-color: #f1ff98; /* Колір тла секції hero */
  padding: 50px 20px;
  width: 50%;
  text-align: center;
}

.hero h2 {
  font-weight: 600; /* Semi bold */
  font-size: 24px; /* Збільшити розмір шрифту */
}

.buttons {
```

```
  background-color: #f1ff98; /* Колір тла секції buttons */
  padding: 20px;
  width: 50%;
  text-align: center;
  position: relative; /* Для позиціонування кнопок */
  top: -50px; /* Зсунути кнопки вище */
}

.buttons-home {
    margin-top: 20px;
}

.button:hover {
    background-color: #8ba9c7;
}

.fact.hidden {
    display: none;
}

.create-bug {
  background-color: #f17798; /* Рожевий колір кнопки Create bug */
  color: #fff; /* Колір тексту кнопки Create bug */
}

.fix-bug {
  background-color: #60b97d; /* Зелений колір кнопки Fix bug */
  color: #fff; /* Колір тексту кнопки Fix bug */
}

.buttons button {
  font-family: "Kalam", cursive;
  font-size: 24px; /* Збільшити розмір шрифту кнопок */
  padding: 15px 40px; /* Поля кнопок */
  border-radius: 40px; /* Закругленість кутів кнопок */
  margin-right: 40px; /* Відступ між кнопками */
  cursor: pointer;
  border: none;
}

.button {
    display: inline-block;
    padding: 10px 20px;
```

```
    background-color: #fcc0d0;
    color: rgb(0, 0, 0);
    text-decoration: none;
    font-weight: bold;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

footer {
  background-color: rgba(0, 0, 0, 0.5); /* Колір фону футера */
  padding: 0px 0px 0px 20px;
  text-align: center;
  position: fixed;
  bottom: 0;
  width: 99%;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.footer-links {
  list-style: none;
  padding: 0;
  display: flex;
  justify-content: space-between; /* Рівномірний розподіл елементів */
  width: 50%; /* Змінено ширину, можна налаштувати відповідно до вашого
дизайну */
  margin: 0 auto; /* Центрування елементів */
}

.footer-links li {
  display: inline;
  margin: 0 10px;
}

.footer-links li a {
  color: #ffffff;
  text-decoration: none;
}

.social-icons {
  margin-top: 10px;
}
```

```
.social-icons a {
  color: #ffffff;
  font-size: 20px;
  margin: 0 10px;
  text-decoration: none;
}

.social-icons a:hover {
  color: #f977ec; /* Змініть колір при наведенні на іконку */
}

.interesting-facts{
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: #f1ff98;
    border-radius: 10px;
    padding: 20px;
    margin-top: 50px;
    margin-left: auto;
    margin-right: auto;
    max-width: 400px;
    font-size: 24px;
    margin-bottom: 10px;
    flex-basis: 100%;
}

.profile-image {
    width: 350px; /* Задайте бажаний розмір */
    height: auto; /* Залиште автоматичну висоту для збереження пропорцій */
    margin-right: 180px; /* Задайте відступ між зображенням та текстом */
    margin-left: 100px;
  }

.profile-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: #fcc0d0;
    border-radius: 10px;
    padding: 20px;
    margin-left: 150px;
```

```
    margin-right: auto;
    max-width: 400px;
}

.ekran{
    display: flex; /* Встановлюємо контейнер як flex-контейнер */
    align-items: center;
}

.container {
    text-align: center;
    margin-top: 50px;
    font-size: 24px;
}

  .circle {
    position: absolute;
    right: 60px;
    width: 170px;
    height: 170px;
    border-radius: 50%;
    background-color: #fcc0d0; /* Колір кружочка */
    display: flex;
    align-items: center;
    justify-content: center;
    color: #ffffff;
  }

  .email,
  .photo {
    margin-top: 20px;
    margin-left: 50px;
    text-align: left;
  }

  .description {
    margin-top: 20px;
    margin-left: 50px;
    text-align: left;
  }

  textarea {
    width: 80%;
```

```
  height: 100px;
}

input[type="file"] {
  display: none;
}

.custom-file-upload {
  border: 1px solid #ccc;
  display: inline-block;
  padding: 6px 12px;
  cursor: pointer;
  border-radius: 5px;
  background-color: #f17798;
  color: #fff;
}

.overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5); /* Затемнення */
  display: none;
}

.modal {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: #fcc7d6; /* Колір фону модального вікна */
  padding: 20px;
  border-radius: 10px;
  display: none;
}

.modal p {
  color: #000;
  font-weight: bold;
  font-family: inter, sans-serif; /* Змінив шрифт */
}
```

```
 .modal button {
   background-color: #f17798; /* Колір кнопки */
   color: #fff; /* Колір тексту на кнопці */
   padding: 15px 40px; /* Збільшуємо поля кнопки */
   border: none;
   border-radius: 5px;
   cursor: pointer;
   font-family: "Kalam", sans-serif; /* Змінив шрифт */
   font-weight: bold; /* Змінив на bold */
   font-size: 18px; /* Збільшуємо розмір шрифту */
 }

#map {
   height: 100vh;  /* Высота карты */
}

.bug-main {
background-color: #f1ff98;
}

#file-icon {
   margin-left: 10px;
   color: green;
}
```

report_issue.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<div class="bug-main">
   <div class="container">
      <h2>Report an issue</h2>
      <div class="circle">
         <p>Found an issue?<br />Let us know!</p>
      </div>
      <form method="POST" action="{% url 'create_bug' %}"
enctype="multipart/form-data">
         {% csrf_token %}
         <div class="email">
            <p>Email</p>
```

```
            <input type="email" name="email" placeholder="Enter your email" />
        </div>
        <div class="photo">
            <p>Photo</p>
            <label for="file-upload" class="custom-file-upload">
                <span id="file-label-text">Choose file</span>
                <span id="file-icon" style="display: none;">✔ </span>
            </label>
            <input id="file-upload" type="file" name="photo" accept="image/*"
style="display: none;" />
        </div>
        <div class="description">
            <p>Describe an issue</p>
            <textarea name="description" placeholder="Describe the
issue"></textarea>
        </div>
        <div class="button">
            <button type="button" id="doneButton" style="background-color:
#f17798; color: #000000">
                Done
            </button>
        </div>
        <div class="overlay" id="overlay"></div>
        <div class="modal" id="modal">
            <p>Your request has been sent.<br />Thank you for your cooperation!</p>
            <p id="geolocation"></p>
            <input type="hidden" id="latitudeInput" name="latitude" />
            <input type="hidden" id="longitudeInput" name="longitude" />
            <button type="submit" id="returnButton">
                Return to main page
            </button>
        </div>
    </form>
</div>

<div id="map"></div>


</div>
<script>
    document.getElementById("file-upload").addEventListener("change", function () {
        var fileLabelText = document.getElementById("file-label-text");
        var fileIcon = document.getElementById("file-icon");
        if (this.files.length > 0) {
```

```
        fileLabelText.textContent = this.files[0].name;
        fileIcon.style.display = "inline";
    } else {
        fileLabelText.textContent = "Choose file";
        fileIcon.style.display = "none";
    }
});
function closeModalAndRedirect(url) {
    closeModal();
    window.location.href = url;
}
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else {
        document.getElementById("geolocation").innerHTML =
            "Geolocation is not supported by this browser.";
    }
}

function showPosition(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    document.getElementById("geolocation").innerHTML =
        "Latitude: " + latitude + "<br>Longitude: " + longitude;


    document.getElementById("latitudeInput").value = latitude;
    document.getElementById("longitudeInput").value = longitude;
}

function openModal() {
    getLocation();
    document.getElementById("overlay").style.display = "block";
    document.getElementById("modal").style.display = "block";
}

function closeModal() {
    document.getElementById("overlay").style.display = "none";
    document.getElementById("modal").style.display = "none";
}

function returnToMainPage() {
```

```
        closeModal();
        window.location.href = "{% url 'home_page' %}";
    }
    document.getElementById("returnButton").addEventListener("click", function () {
        returnToMainPage();
    });
    document.getElementById("doneButton").addEventListener("click", function () {
        openModal();
    });

    var map = L.map("map").setView([49.842957, 24.031111], 13);

    L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
        maxZoom: 19,
    }).addTo(map);

    var marker = L.marker([49.842957, 24.031111], { draggable: true }).addTo(map);

    marker.on("dragend", function (event) {
        var position = marker.getLatLng();
        console.log(position);
    });
</script>
{% endblock %}
```

login.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
  <link rel="stylesheet" href="{% static 'login.css' %}">
  <div class="login-container">
    <h2 class="login-title">Log in</h2>
    <form method="post">
      {% csrf_token %}
      <div class="form-group">
        <div class="us-container">
          <label for="id_username"><i class="fas fa-user"></i>Username:</label>
          {{ form.username }}
        </div>
      </div>
      <div class="form-group">
```

```
      <div class="password">
        <label for="id_password"><i class="fas fa-lock"></i>Password:</label>
        {{ form.password }}
      </div>
    </div>
    <div class="form-group">
      <div class="us-container">
        <div class="button-container">
          <button type="submit">Log in</button>
        </div>
      </div>
    </div>
  </form>
 </div>
{% endblock %}
```

profile.css

```
.ekran{
  display: flex; /* Встановлюємо контейнер як flex-контейнер */
  align-items: center;
}

.profile-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: #fcc0d0;
    border-radius: 10px;
    padding: 20px;
    margin-left: 150px;
    margin-right: auto;
    max-width: 400px;
  }

  .profile-details {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap;
    background-color: #f1ff98;
    border-radius: 10px;
```

```
  padding: 20px;
  box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.1);
  width: 80%;
  margin-bottom: 50px;
}

.welcome-msg {
  font-size: 24px;
  margin-bottom: 10px;
  flex-basis: 100%;
}
p{
  text-align: left;
}
.profile-info {
  flex-basis: 50%;
  margin-bottom: 50px;
  text-align: center;
  align-items: center;
}
.iconka{
  align-items: center;
}
.profile-info p {
  margin-bottom: 10px;
}

.profile-actions {
  display: flex;
  flex-direction: column;
  align-items: center;
  flex-basis: 50%;
}

.action-box {
  margin-bottom: 10px;
}

.action-link {
  display: block;
  padding: 10px 20px;
  font-size: 16px;
  text-decoration: none;
```

```
    color: #000;
    background-color: #fcc0d0;
    border-radius: 8px;
  }

  .delete-link {
    background-color: #cc3333;
  }

.profile-image {
  width: 350px; /* Задайте бажаний розмір */
  height: auto; /* Залиште автоматичну висоту для збереження пропорцій */
  margin-right: 10px; /* Задайте відступ між зображенням та текстом */
  margin-left: 100px;
}
.profile-image1 {
  width: 350px; /* Задайте бажаний розмір */
  height: auto; /* Залиште автоматичну висоту для збереження пропорцій */
  margin-right: 100px; /* Задайте відступ між зображенням та текстом */
  margin-left: 10px;
}
```

register.css

```
.registration-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  margin-top: 50px;
  background-color: #fcc0d0;
  border-radius: 10px;
  padding: 20px;
  margin-left: auto;
  margin-right: auto;
  max-width: 400px;}

.registration-title {
  font-size: 24px;
  font-weight: bold;
  text-align: center;
  color: #000;
  margin-bottom: 20px;
```

```
}

.form-group {
  margin-bottom: 20px;
  text-align: center;
}
.rules-container{
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color: #f1ff98;
  border-radius: 10px;
  padding: 10px;
  margin-left: auto;
  margin-right: auto;
  max-width: 400px;

}
.form-group label {
  display: block;
  font-size: 18px;
  margin-bottom: 10px;
}

.form-group input,
.form-group select {
  width: 300px;
  height: 40px;
  font-size: 16px;
  padding: 5px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.password-help-text {
  font-size: 18px;
  color: rgb(0, 0, 0);
  list-style: none;
}

.button-container {
  display: flex;
```

```
  justify-content: center;
  margin-top: 20px;
}

button[type="submit"] {
  font-family: "Kalam", cursive;
  width: 120px;
  height: 40px;
  font-size: 18px;
  background-color: #f1ff98;
  color: #000;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button[type="submit"]:hover {
  background-color: #245588;
}

button[type="submit"]:focus {
  outline: none;
}
.link {
  display: flex;
  flex-direction: column;
  align-items: center;
  margin-bottom: 10px;
  padding: 10px 20px;
  font-size: 16px;
  background-color: #337ab7;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  margin-top: 10px;
  margin-right: 20px;
}

.link:hover {
  text-decoration:  none ;
}
.link1{
```

```
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

home_page.css

```
 body {
   margin: 0;
   padding: 0;
   font-family: "Kalam", cursive;
   color: #000; /* Колір тексту */
 }

 header {
   background-image: url("nature1.jpg"); /* Замініть 'your-image.jpg' на URL свого фото */
   background-size: cover;
   background-position: center;
   height: 300px; /* Висота заголовка */
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 0 20px;
 }

 .logo h1 {
   color: #ffffff; /* Колір тексту логотипу */
   font-size: 36px; /* Збільшити розмір шрифту */
   font-style: italic; /* Наклонний шрифт */
 }

 .icons {
   color: #000; /* Колір іконок */
   font-size: 24px;
 }

 .hero {
   background-color: #f1ff98; /* Колір тла секції hero */
   padding: 50px 20px;
   text-align: center;
 }
```

```
.hero h2 {
  font-weight: 600; /* Semi bold */
  font-size: 24px; /* Збільшити розмір шрифту */
}

.buttons {
  background-color: #f1ff98; /* Колір тла секції buttons */
  padding: 20px;
  text-align: center;
  position: relative; /* Для позиціонування кнопок */
  top: -50px; /* Зсунути кнопки вище */
}

.create-bug {
  background-color: #f17798; /* Рожевий колір кнопки Create bug */
  color: #fff; /* Колір тексту кнопки Create bug */
}

.fix-bug {
  background-color: #60b97d; /* Зелений колір кнопки Fix bug */
  color: #fff; /* Колір тексту кнопки Fix bug */
}

.buttons button {
  font-family: "Kalam", cursive;
  font-size: 24px; /* Збільшити розмір шрифту кнопок */
  padding: 15px 40px; /* Поля кнопок */
  border-radius: 40px; /* Закругленість кутів кнопок */
  margin-right: 40px; /* Відступ між кнопками */
  cursor: pointer;
  border: none;
}

footer {
  background-color: #f1ff98; /* Колір тла футера */
  padding: 20px;
  text-align: center;
  position: fixed;
  bottom: 0;
  width: 100%;
}
```

```
footer p {
  margin: 0;
}

.footer-links {
  list-style: none;
  padding: 0;
  margin-top: 10px;
}

.footer-links li {
  display: inline;
  margin: 0 10px;
}

.footer-links li a {
  color: #000; /* Колір посилань */
  text-decoration: none;
}

.social-icons {
  margin-top: 10px;
}

.social-icons a {
  color: #000; /* Колір іконок соціальних мереж */
  font-size: 20px;
  margin: 0 10px;
}
```

login.css

```
.login-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  margin-top: 50px;
  background-color: #fcc0d0;
  border-radius: 10px;
  padding: 20px;
  margin-left: auto;
```

```
  margin-right: auto;
  max-width: 400px;
}

.login-title {
  font-size: 30px;
  font-weight: bold;
  text-align: center;
  color: #000;
  margin-bottom: 20px;
}

.form-group {
  margin-bottom: 20px;
  text-align: center;
}
.us-container{
  background: #f1ff98;
  padding: 1em;
  display: flex;
  flex-direction: column;
  gap: 0.5em;
  border-radius: 20px;
  color: #000;
}
.us-container input {
  outline: none;
  border: none;
}
.password {
  background: #f1ff98;
  padding: 1em;
  display: flex;
  flex-direction: column;
  gap: 0.5em;
  border-radius: 20px;
  color: #4d4d4d;
}
.password input {
  outline: none;
  border: none;
}
```

```
.form-group label {
  display: block;
  font-size: 18px;
  margin-bottom: 10px;
}
.form-group input {
  width: 300px;
  height: 40px;
  font-size: 16px;
  padding: 5px;
  border-radius: 4px;
}
.password input:hover {
  border-bottom: 2px solid #336699; /* Bottom underline color */
}
.us-container input:hover {
  border-bottom: 2px solid #336699; /* Bottom underline color */
}
.button-container {
  display: flex;
  justify-content: center;
}
button[type="submit"] {
  font-family: "Kalam", cursive;
  width: 150px;
  height: 40px;
  font-size: 18px;
  background-color: #fcc0d0; /* Button color remains the same */
  color: #000;
  border: none;
  border-radius: 16px;
  cursor: pointer;
}
button[type="submit"]:hover {
  background-color: #245588;
}

button[type="submit"]:focus {
  outline: none;
}
```