НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.9:624

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету	Завідувач кафедри
інформаційних технологій	комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____ _____	_____
«___»_____ 2021 р.	«___»_____ 2021 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Інтелектуальна система генерування дизайну інтер'єру на основі уподобань користувача

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма Програмне забезпечення інформаційних систем

Орієнтація освітньої програми освітньо-професійна

Гарант освітньої програми

к.т.н. доцент _____ Голуб Б. Л.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

к.т.н. доцент _____ Ткаченко О. М.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Іманов А. М.
(підпис) (ПІБ студента)

КИЇВ-2021

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних наук
к.т.н. доцент Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"" 20 року

ЗАВДАННЯ

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ
ІМАНОВУ АНТОНУ МАМЕДОВИЧУ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Програмне забезпечення інформаційних систем»
Орієнтація освітньої програми освітньо-професійна

1. Тема магістерської кваліфікаційної роботи Інтелектуальна система генерування дизайну інтер'єру затверджена наказом ректора НУБіП України від 29.10.2020 № 1636 "С"
2. Термін подання завершеної роботи на кафедру
рік, місяць, число
3. Вихідні дані до магістерської кваліфікаційної роботи матеріали переддипломної практики
4. Перелік питань, що підлягають дослідженню:
   1. Аналіз предметної області
   2. Проектування системи аналізу та видачі результатів
   3. Реалізація рішення згідно 2 пункту

Дата видачі завдання 2020 . 10 . 29
рік, місяць, число

Керівник магістерської роботи _____ Ткаченко О. М. доц. к.т.н.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Іманов О. М.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Від проблеми «вибору дизайну інтер'єру» найбільше потерпають мешканці квартир [1], в своїй більшості ті, які не мають великого бюджету, або часу для вивчення детального дизайну нових квартир, чи при ремонті старої. Зазвичай такі люди не мають можливості найняти собі окрему людину, яка б займалась цим ділом, адже навіть на сам ремонт скоріше всього вони витрачають більшу частину своїх збережень, якщо не всі.

Серед кількох варіантів вирішення проблеми, найбільш прийнятним в поточних умовах є використання сучасних технологій й мережі інтернету для швидкого та дешевого вибору адаптивного дизайну [2], які допомагають автоматизувати процес вибору нового дизайну інтер'єру, при цьому вимагаючи від користувача мінімальних витрат зусиль і часу. Наразі такі технології не надають допомоги у виборі дизайну, тим не менш, вони пропонують самому спроектувати інтер'єр квартири, чи будинку [3]. Цей спосіб не сприяє покращенню ситуації й тим більш, таким чином збільшується навантаження на користувача цих систем. Адже за нинішніх умов потрібно витратити ще додатковий час, не тільки на вивчення інформації по дизайну, але ще й по деталям інтер'єру.

Якщо ж змінити підхід системи до вирішення проблем користувача, не методом ручного проектування інтер'єру, а збором суттєво важливих даних (побажань) щодо нового дизайну інтер'єру, то можливо скоротити час й навантаження на користувачів систем [4].

Новий підхід, який потрібно буде використати, представляє собою попереднє опитування користувача, при якому система збирає усі суттєво важливі дані для аналізу та подальшого вирішення результату, такі як кількість кімнат у квартирі, їх метрики, тобто висота, довжина та ширина, тощо. Але маючи лише кількісні показники, не достатньо для розуміння уподобань користувача, а тому йому буде надано ряд питань, де буде запропоновано вибрати фотографії інтер'єру, який сподобався користувачу. Самі фотографії

представляють собою відображення частини квартири спроектованої за певним дизайном інтер’єру. В першому абзаці вище було згадано про бюджет, який витрачається на ремонт, або побудову нового дизайну для квартири, а тому попередньо зібрані дані також буде використано для розрахунку вартості даної послуги. Але на цьому робота системи не зупиняється й під результатами будуть вказані магазини, де можливо оплатити дану послугу.

Метою даної магістерської роботи є визначення та розробка ефективних засобів для обробки інформації для прийняття рішень щодо вибору дизайну інтер’єру.

Об’єктом дослідження являється процес формування дизайну інтер’єру.

Предмет дослідження – це системи підтримки прийняття рішень у сфері дизайну інтер’єру.

Під час виконання магістерської роботи були використані такі засоби: мова програмування Python та Javascript, система управління базами даних PostgreSQL, середовище Visual Studio Code.

Для реалізації поставленої мети визначено такі основні завдання:

1) проаналізувати предметну область;
2) змоделювати систему аналізу та видачі результатів;
3) розробити інтелектуальну систему генерування дизайну інтер’єру на основі вподобань користувачів.

Дана магістерська робота складається з **N** сторінок. Вона включає 4 розділи.

Апробація результатів дослідження:

- XI міжнародна науково-практична конференція молодих вчених “Інформаційні технології: економіка, техніка, освіта” - публікація “Інтелектуальна система генерування дизайну інтер’єру на основі вподобань користувачів”
- IV Всеукраїнська науково-практична інтернет-конференція студентів та аспірантів “Теоретичні та прикладні аспекти розробки

комп’ютерних систем” – публікація “ Інтелектуальна система генерування дизайну інтер’єру на основі вподобань користувачів”

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Перш за все, варто зазначити, що «Системний аналіз – це методологія теорії систем, що полягає в дослідженні будь-яких об'єктів, що представляються в якості систем, проведенні їх структуризації і подальшого аналізу. Головна особливість системного аналізу полягає в тому, що він включає в себе не тільки методи аналізу (від грец. analysis - розчленування об'єкта на елементи), а й методи синтезу (від грец. synthesis - з'єднання елементів в єдине ціле). Головна мета системного аналізу – виявити й усунути невизначеність при вирішенні складної проблеми на основі пошуку найкращого рішення з існуючих альтернатив.»[5]

Для даної задачі щодо створення сервісу для вибору дизайну інтер'єра та прорахунку вартості ремонту, виокремлюються наступні задачі:

1. Обширний аналіз вподобань та очікувань потенціальних користувачів щодо зовнішнього вигляду житла, окрема увага приділена останнім популярним тенденціям.

2. Аналіз існуючих рішень та будматеріалів, що можуть бути використані для створення інтер'єру приміщень.

3. Розробка проекту інтелектуальної системи підбору дизайну інтер'єру та прорахунку вартості житла на основі вподобань користувача.

4.Створення основного коду даного програмного продукту.

1.2 Актуальність сервісу

Безумовно, ми живемо у XXI сторіччі з блискавичним розвитком все нових технологій. Інтернет, що отримав своє визнання у 2000-х

роках, значно полегшив отримання інформації, лише на пошук якої люди могли витрачати роки життя не в такому й далекому минулому.

Дуже вчасно розвинулась і швидкість доступу: з 56 кілобайт, через які нам хвилинами доводилось чекати завантаження однією сторінки, прийшов й широкосмуговий доступ, що зробив реальним перегляд відео високої якості з мінімальною затримкою.

Нині, користувач може легко знайти найрізноманітніші варіанти інтер'єру, а купа відеоблогерів на платформі Youtube допоможуть визначитися з всіма тонкими моментами.

Дійсно, але перегляд лише самих фото віднімає купу часу. А як відомо, саме час – наша найдорогоцінніша валюта.

Кожен має унікальне почуття стилю і свою думку про те, що являє собою добре спроектований простір. Дехто відчуває себе найбільш задоволеним в затишних межах французького сільського декору, а інші вважають мінімалістичні атрибути спокійними та центруючими. Деякі люди жадають інтимності невеликих кімнат; а інші вважають за краще жити в просторих приміщеннях відкритого планування.

Але, хоча естетичні уподобання дуже індивідуальні, певні просторові атрибути є універсально (підсвідомо) привабливими, створюючи комфорт і благополуччя на неврологічному рівні [6]. Краєвид на природу, рясне (але не яскраве) денне світло та хороша якість повітря в приміщенні — це бажані риси. Можливість контролювати навколишнє середовище — від температури до рівня освітленості — є ще одним важливим критерієм. Простір також має бути простим та інтуїтивно зрозумілим у використанні та маневруванні. Ці особливості стосуються будь-якого місця проживання, від модерністського міського таунхаусу до пляжної відли в середземноморському стилі.

Вони також підкріплені наукою. Психологи-екологи, географи, ергономісти та експерти з ефективності є одними з спеціалістів, які з'ясували складні відносини між людиною та місцем. Студенти, які займаються дизайном інтер'єрів, проводять багато часу в школі, вивчаючи нюанси цих міждисциплінарних досліджень, які спільно називають дослідженнями поведінки середовища.

Для наших просторових уподобань є еволюційна основа [7]. Оскільки люди еволюціонували шляхом адаптації до боротьби за виживання в дикій природі, типи просторів, які ми підсвідомо віддаємо перевагу, мають риси, які допомогли нашим предкам боротися з хижаками та знаходити засоби для існування; сьогодні вони вселяють спокій і розслаблення. Люди мають вроджену прихильність до природного середовища, наповненого водою, зеленню, сонячним світлом і квітами, що сигналізує про наявність їжі та води. (Термін для цієї любові до природи — «біофілія».) Відповідно, їх важливо враховувати під час визначення розміру та проектування простору.

Ось огляд лише пари параметрів, розрахунок яких звичайний користувач зазвичай ігнорує, або невірно використовує:

Бюджет:

Зазвичай, користувач спочатку підбирає дизайн інтер'єру, що підходить йому по смаку, а вже після цього намагається влізти в дозволений бюджет, економлячи на важливих нюансах та псуючи картину дешевими пластиковими аналогами, яскравим прикладом є плитка «під цеглу», або полістирольні плінтуси.

Розробка бюджету проекту може бути неприємною, особливо для тих, хто тільки починає працювати в процесі дизайну інтер'єру. Новачки, як правило, мають неточне уявлення про ціни на різні елементи та послуги. (І

справді існує досить широкий діапазон; розглянемо металевий консольний стіл довжиною 70 см, який може коштувати 400 або 100 000 доларів США.) Хоча деякі люди можуть легко визначити максимальну суму, яку вони відчувають комфортною, щоб виділити її на проект, адже для них це можливо, у багатьох домовласників бюджет стоїть під знаком питання.

Тим не менш, важливо визначити прийнятну цифру на початку проекту, перш ніж приймати будь-які рішення — або навіть приступати до проектування, якщо на те пішло. «Немає сенсу йти шляхом, який може бути абсолютно нереальним», – каже дизайнер Джеймі Дрейк (насправді, багато професіоналів не почнуть працювати з клієнтом, доки бюджет не буде затверджено; це часто записано в контракті.) [8] Створення бюджету, особливо для великої чи складної роботи, є завданням, від якого може скористатися неспеціаліст від професійних порад. Досвідчені дизайнери можуть приблизно оцінити, скільки може коштувати проект, з огляду на порівнянні роботи, які вони виконали, або запропонувати відповідний діапазон.

Після оформлення замовлення на товари розраховується вартість доставки та доставки (приблизно 15 відсотків вартості товару); також враховують відповідні податки з продажів на послуги з дизайну та покупки. Ці збори можуть становити додаткові 10 відсотків від базової вартості меблів та матеріалів. Незалежно від очікуваного бюджету, власник розумного будинку або дизайнер виділить подушку в розмірі щонайменше 25 відсотків загального бюджету. Несподіване обов'язково виникне, а несподіване завжди коштує грошей.

Розмір приміщення:

Важливо оцінити просторові характеристики приміщення з об’єктивної точки зору, незалежно від майбутнього дизайну чи навіть цільового

використання. Оцініть наявні елементи та їх стан — від підлогових дощок і стель до розташування дверей та розеток. Уважно перевірте розташування вікон, верхнього освітлення та будь-яких наявних вбудованих елементів. Чи суттєво відрізняються площі кімнати при огляді на одному футі над підлогою, на п’яти футах вище чи ближче до стелі? Недосконалості з’являються на кожному рівні, і їх потрібно враховувати під час розробки дизайну. Якщо проект ще не побудований, судити про простір можна, вивчивши архітектурні плани та будівельну документацію.

Наведений вище контрольний список допоможе розпізнати потенціал та обмеження приміщення, але не є виключним.

Зверніть увагу на загальний масштаб приміщення (тобто його розмір у порівнянні з тілом людини). Кімната оцінюється як затишна чи вражаюча за тим, наскільки велика чи маленька людина почувається в ній. Наскільки висока стеля? Наскільки широка, велика чи затишна кімната? Чи відчуваєте себе там карликом, переповненим чи комфортним? На сприйняття масштабу часто впливає шлях або перехід з однієї кімнати в іншу.

Френк Ллойд Райт чудово використовував цю якість, часто проектуючи дуже низький, обмежувальний вхід, який відкривав кімнату з високою стелею. Це створило поштовх розширення, який посилив враження, схоже на лофт [9].

Масштаб кімнати найбільше впливає на те, що потрібно зробити, щоб створити бажану атмосферу. Якщо в кімнаті висока стеля і грандіозні ліпнини, створити відчуття затишку буде складно. Аналогічно, якщо кімната розташована з маленької сторони і метою є формальність, то особливу увагу доведеться приділити вікнам і входам, розміру меблів і освітленню, щоб реалізувати бачення.

У той час як масштаб стосується відношення однієї речі до іншої (тобто людини до кімнати), пропорція — це відношення частин до цілого. Вважається, що прямокутний простір, висота якого приблизно дорівнює ширині його найкоротшої сторони, має рівновагу та стабільність.

Так само, як люди шукають рівноваги в житті, вони також шукають його в оточенні. Відчуття візуальної рівноваги часто встановлюється за допомогою симетрії, як у випадку з парними вікнами, які однаково розташовані на стіні. Але рівновага також може випливати з асиметрії; розглянемо групу для сидіння з диваном, закріпленим з одного боку, і, навпроти, двома кріслами. Зверніть увагу на будь-які можливості створити симетричний або асиметричний баланс за допомогою архітектурних елементів або декору. Шукайте наявність сильної центральної лінії або фокусної точки. Чи є камін у центрі, наприклад? Також зверніть увагу на те, що називається «локальною» симетрією: лише одна частина загального плану дозволяє симетричне розташування, наприклад, альков для сидіння у більшій кімнаті [10].

Вікна, двері та мансардні вікна оцінюються і як портали для огляду, і як канали денного світла. Вони можуть обрамлювати вигляд або створювати моменти паузи — переходи між одним простором і іншим. Зверніть увагу, який тип огляду на кожну дверну чи віконну раму, і чи слід приховувати ці краєвиди або відзначати їх. Скільки прямого чи непрямого освітлення допускає кожне вікно? Чи потрібно обробляти вікна або відкриті дверні прорізи для візуальної конфіденційності або контролю світла?

Кольорова схема

Колір є основоположним елементом будь-якої палітри, але його складно використовувати майстерно та відповідно до принципів балансу та пропорцій.

Колір має стратегічну цінність, допомагаючи дизайнеру маніпулювати архітектурою та властивим характером простору.

Колір також має емоційний, символічний зміст. Колір говорить: він може заспокоїти і заспокоїти, створити напругу або енергію, залежно від бажаного ефекту.

Як правило, схема повинна мати пропорційний розподіл кольорів — у різних тонах — на стінах, підлозі, стелі, меблях та аксесуарах. На основі концепції та образів, що надихають, визначають:

• Домінуючий відтінок, який буде використовуватися для приблизно 50 відсотків усіх тканин або матеріалів. Колір може бути однотонним, текстурованим або з'являтися на принтах.

• Вторинні відтінки або два, що становлять близько 30 відсотків схеми.

• Інша частина схеми повинна включати акцентний колір, який може бути яскравішим або більш драматичним. Метал, дерево, листя або інший природний матеріал також можуть виступати в якості акценту або допоміжного кольору.

Зібрати кольорову гаму спочатку може бути непросто. Дизайнери інтер'єру вчаться використовувати колір, змішуючи пігменти фарби та розуміючи науку, яка лежить в основі того, як око «читає» відтінки в контексті та між різними елементами кімнати. Відповідно, відмінною відправною точкою є улюблена картина, яка, незалежно від зображуваного зображення, має правильне хроматичне відчуття для кімнати, яку ви хочете створити. Зрештою, художники роками вчаться, щоб стати експертом у розповіді історії з кольором: використаними пропорціями, текстурою і навіть візерунком [11].

Настільний пейзаж Вінсента Ван Гога, якщо його ретельно проаналізувати на предмет відтінків, цінності, насиченості та пропорції кольору, можна відтворити як їдальню у французькому стилі з жовтими стінами та акцентами насиченого синього, теплого білого, помаранчевого відтінку, і чорний [12].

Розробляючи колірну схему, задайте собі наступні запитання:

• Чи збалансовано?

• Чи правильно розподілені кольори по кімнаті? Деякі дизайнери застосовують до створюваних ними кімнат принцип розподілу тонів: імітуючи природний ландшафт, на стелю наноситься найсвітліший колір, як небо; середній тон або середній колір використовується для найбільшої площі (стіни та меблі); і найтемніший колір використовується для підлоги, як у землі.

• Чи достатньо нейтральних? Ці більш бліді кольори можна найбільш успішно нанести на найбільший відсоток поверхні, наприклад, на стіни, штори і навіть на підлогу, де яскраво-хроматичний колір може бути переважаючим. Схема, розрахована на багато десятиліть, повинна використовувати нейтральні кольори для більш постійних речей і яскраві або барвисті акценти на тих аксесуарах, які можна легко змінити (подушки, абажури і навіть твори мистецтва).

• Чи утримується загальна колірна гамма та схема матеріалів?

Правила щодо кольору — це тверді рекомендації для початківців, але немає загальновизнаних формул, яких можна було б дотримуватися, як рецепт, щоб досягти певної мети. Дійсно, фахівці з кольору настільки добре знають свою тему, що можуть грати з так звані правила і порушувати їх [13]. З огляду на це, більшість інтер'єрних схем потрапляє в одну з наступних категорій, і їх сила або делікатність буде змінюватися

відповідно до інтенсивності, яскравості або блідості основного відтінку. Зауважте, що наведені нижче правила грунтуються на барвниках або пігментах, які використовуються в або на твердих речах, таких як тканина або дерево; різні теорії та правила застосовуються до кольору, який використовується в освітленні [14].

Нижченаведена таблиця підсумовую дану інформацію, показуючи, що онлайн сервіс буде задіянним максимальною кількістю користувачів.

Таблиця 1.1

Сам	Онлайн сервіс	Дизайнер
Купа вільного часу	Мало вільного часу	Мало, або немає вільного часу
Майже немає грошей	Є гроші, але є бажання до їх правильного використання	Широкий фінансовий бюджет
Є певні дизайнерські здібності	Може не мати дизайнерських здібностей	Відсутні дизайнерські здібності
Пріоритет доступності над дизайном	Пріоритет дизайн – вартість врівноважений	Пріоритет дизайну над вартістю
Є потрібні інструменти	Як є, так і можуть бути відсутніми інструменти	Відсутні інструменти

З огляду на все вищезазначене приходимо до висновку, що даний сервіс дійсно актуальний. Так, особа може обійтися і без нього, якщо у неї є купа вільного часу, але при цьому вона з великою вірогідністю зазнає фінансових збитків як під час закупівлі матеріалів, так і при розрахунку з підрядниками.

1.3 Аналіз сервісів-конкурентів

Створення простору самостійно вимагає багато часу, інструментів і навченого ока для якісного дизайну. З іншого боку, найняти особистого дизайнера інтер'єру коштує дорого (у середньому коштує 2000-5000 доларів США за кімнату за дизайнерські послуги і вимагає кількох особистих зустрічей).

Здавалося б, в мережі має існувати безліч подібних продуктів, але авіть поглиблений пошук мережі інтернет, зокрема ресурсу reddit.com приводить до висновку, що нині на ринку існує лише три конкурентоздатні системи, що є дійсно працездатними:

- www.designfor-me.com
- www.decorilla.com
- www.stucco.com

Однак, всі вони націлені та можуть ефективно використовуватися виключно громадянами США, Канади та деякими країнами Західної Європи.

Після довгого вивчення та тестування вищезгаданих систем я прийшов до висновку, що саме розробка Stucco у підсумку є найбільш завершеним продуктом.

Таблиця 1.2

Назва сервісу	**Переваги**	**Недоліки**
design-for-me.com	Великий вибір дизайну інтер'єрів	Застарілий інтерфейс, відсутність підтримки HTML5, немає підтримки української мови та українських користувачів.
decorilla.com	Підтримка великої кількості онлайн магазинів з продажу товарів для ремонту інтер'єру	Застарілий інтерфейс, немає підтримки української мови та українських користувачів
stuccco.com	Додатковий сервіс замовлення послуг дизайнера	Зависока ціна за сервіс, перевага сервісу дизайнер-клієнт, немає підтримки української мови та українських користувачів

Процес онлайн-дизайну інтер'єру дещо відрізняється для кожного з вищезазначених сервісів, що займаються дизайном інтер'єру в Інтернеті, тому далі я наведу огляд лише системи Stuccoo.

Це покроковий огляд процесу онлайн-дизайну інтер'єру на Stuccco [15]:

1. Кожен майбутній онлайн-проект дизайну інтер'єру починається з безкоштовного онлайн-дзвінка про дизайн інтер'єру з членом команди дизайнерів Stuccco. 30-хвилинний дзвінок використовується як можливість без тиску поділитися інформацією про свої стильові уподобання та кімнату(и), яку(-і) ви хочете створити.

2. Після дзвінка ви отримуєте електронною поштою список із трьох рекомендованих дизайнерів інтер'єру Stuccco Online (Stuccco враховує стиль дизайну, часовий пояс, особистість, часову шкалу, тип кімнати, наявність дизайнера тощо, коли рекомендує дизайнерів).

3. Ви переглядаєте рекомендовані профілі Stuccco дизайнерів, які включають минулі проекти, відгуки клієнтів тощо, а потім вибираєте один для роботи.

4. У вас є дзвінок 1:1 з вашим дизайнером, щоб обговорити вимоги до вашого проекту дизайну інтер'єру онлайн.

5. Протягом кількох днів ви отримаєте електронний лист із запропонованим дизайном від вашого дизайнера.

6. Ви надсилаєте відгук про запропонований дизайн електронною поштою, і ваш дизайнер оновлює ваш дизайн, поки ви не будете повністю задоволені (це може вимагати одного або кількох раундів перегляду залежно від ваших унікальних потреб).

7. Мета полягає в тому, щоб кожен проект Stuccco Online Design Interior Design був завершений протягом 14 днів після проведення дзвінка 1:1 з обраним вами дизайнером.

Після завершення процесу дизайну інтер'єру Stuccco Online ви отримуєте:

• 3D-рендерінг вашого нещодавно спроектованого приміщення, щоб ви могли побачити, як він виглядатиме, коли всі рекомендовані елементи будуть придбані та встановлені.

• План приміщення з усіма рекомендованими елементами, показаними в масштабі.

• Список покупок із можливістю натискання з усіма рекомендованими товарами, щоб ви могли купувати їх самостійно безпосередньо на веб-сайтах роздрібних продавців.

• Детальні інструкції щодо дизайну, яким ви або хтось інший можете дотримуватись, щоб легко перетворити свій дизайн у реальність.

За останнє десятиліття дизайн житлових будинків став культурною пристрастю. Телевізійні шоу про ремонт будинків, блоги «Зроби сам» та журнали про притулок розглядають домашнє святилище як головного героя, а не лише як декорації. Таке відкриття викликало широке ентузіазм щодо всього домашнього та дому та привернуло увагу до професії. Дизайн інтер'єру більше не сприймається як сувора сфера компетенції тих, у кого великий будинок і глибокі кишені; дизайн інтер'єру є універсальним, релевантним і застосовним до будь-якого місця проживання — неважливо, наскільки скромним — і доступним для кожного покупця.

Для ентузіастів дизайну така доступність є джерелом не тільки розширення можливостей, але й жаху, оскільки демократизація — це не те саме, що демістифікація. Можна прочитати статтю про те, як розрахувати довжину, необхідну для переоббивки дивана, але для того, щоб зрозуміти, що і скільки тканини купувати, все одно необхідно зрозуміти типи пряжі та переплетення, а також як наносяться візерунки, щоб вони вибудовувалися в ряд. шви. І так, завдяки Інтернету споживачі тепер можуть насолоджуватися миттєвими розпродажами дизайнерських продуктів і доступом до раніше ексклюзивних ресурсів, таких як виставкові зали «тільки для торгівлі». Але структуру цін на продукцію на замовлення, декоративну обробку стін і вбудовану шафу може бути важко зрозуміти, що призведе до нереалістичних

очікувань щодо бюджету. Перенасиченість заплутаною (і часто суперечливою) інформацією є ведмежою послугою для будь-кого, хто приступає до дизайнерського проекту, будь то новий власник будинку, який прагне облаштувати все своє житло, орендар на ринку нестандартних штор, щоб прикрасити свою квартиру, або будь-який житель, який планує найняти професіонала для внутрішніх робіт.

Багато людей просто обожнюють проектувати власний простір і воліють займатися цим самостійно, озброївшись енциклопедичною та авторитетною інформацією. Інші можуть просто не мати фінансового становища, щоб зберегти професійні послуги. Мій сервіс покаже мотивованим любителям, як зробити усвідомлений вибір у всьому, від вибору настінного покриття до визначення правильної висоти вбудованої робочої поверхні в домашньому офісі.

Ті, хто вперше в цьому процесі, отримають погляд на все, що входить в схему інтер'єру, щоб вони могли самостійно приймати розумні рішення або визначити, чи потрібно найняти професіонала — декоративного маляра, виробника обробки вікон на замовлення, ремісника для створити килим на замовлення, щоб відповідати дивним пропорціям їхньої вітальні, столяр, щоб побудувати стелажі в бібліотеці, генеральний підрядник для відновлення ванної кімнати для гостей, або дизайнер інтер'єру, щоб реалізувати будинок своєї мрії зверху донизу.

Люди, які вперше наймають дизайнера інтер'єру, дізнаються, як знайти професіонала, чия філософія збігається з їхньою. Пропонуються рекомендації щодо різних способів, включаючи контракти (що вони охоплюють і як їх розшифрувати); як структурована плата за проектування; в якій послідовності протікає процес; як дизайнер буде взаємодіяти не лише з клієнтами, а й з архітекторами, генеральними

підрядниками та іншими суміжними спеціалістами; і чому під час встановлення рекомендується звільнити будинок на кілька днів.

Дизайн інтер'єру - це рівна частина мистецтва і науки. Це вирішення проблем із додатковою цінністю: унікальний творчий поворот дизайнера та здатність удосконалювати рішення з поглядом на красу та емпатію. Хороший дизайнер інтер'єру – вирішує проблеми та оповідає. Його роль полягає в тому, щоб створити розповідь для простору, який з'єднає все це разом. Таким чином, дизайн інтер'єру вимагає як мрій про блакитне небо, так і логічних міркувань. Практикам і любителям дизайну потрібні далекоглядність і панорамне бачення, щоб мати на увазі загальну картину, а також лазерний фокус, щоб побачити, як кожна хвилинна деталь стає на місце. Але по суті дизайн інтер'єру – це набагато більше, ніж сума його частин. Це виходить за рамки вибору красивої обробки та створення індивідуальних меблів. Зрештою, мова йде про створення місця: наповнення простору сенсом і актуальністю, серцем і душею.

Саме ці завдання і буде допомагати виконувати мій сервіс.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні положення

Переважна більшість існуючих на сьогоднішній день методів об'єктно-орієнтованого аналізу і проектування (ООАП) включають включають в себе мову моделювання та, відповідно, опис процесу моделювання. Мова моделювання – це будь-яка штучна мова, котра може використовуватися для вираження інформації (даних) або знань чи систем у структурі, і яка визначається послідовним набором правил [16].

Мова може бути текстовою, або графічною, одним з варіантів якої є UML, що і використовувалася мною.

Уніфікована мова моделювання (UML) — це сімейство графічних позначень, підкріплених єдиною метамоделью, які допомагають описувати та проектувати програмні системи, зокрема програмні системи, створені з використанням об’єктно-орієнтованого (ОО) стилю. Це дещо спрощене визначення. Насправді, UML — це кілька різних речей для різних людей. Це випливає як з його власної історії, так і з різних поглядів людей на те, що робить ефективний процес розробки програмного забезпечення. У результаті моє завдання в більшій частині цього розділу полягає в тому, щоб створити основу для цієї книги, пояснивши різні способи, якими люди бачать і використовують UML.

Мови графічного моделювання існують в індустрії програмного забезпечення протягом тривалого часу. Основна рушійна сила, яка стоїть за всіма ними, полягає в тому, що мови програмування не мають достатньо високого рівня абстракції, щоб полегшити дискусії про дизайн [17].

Незважаючи на те, що мови графічного моделювання існують протягом тривалого часу, в індустрії програмного забезпечення існує величезна кількість суперечок щодо їх ролі. Ці суперечки безпосередньо впливають на те, як люди сприймають роль самого UML.

UML є відносно відкритим стандартом, який контролюється Object Management Group (OMG), відкритим консорціумом компаній. OMG було створено для розробки стандартів, які підтримували сумісність, зокрема взаємодію об'єктно-орієнтованих систем. OMG, мабуть, найбільш відомий за стандартами CORBA (Архітектура посередника загального об’єктного запиту).

UML народився в результаті об’єднання багатьох об’єктно-орієнтованих мов графічного моделювання, які процвітали наприкінці 1980-х і на початку 1990-х років. З моменту своєї появи в 1997 році вона віднесла цю Вавилонську вежу в історію. Я та багато інших розробників дуже вдячні за це сервіс [18].

В основі ролі UML у розробці програмного забезпечення лежать різні способи, якими люди хочуть його використовувати, відмінності, які переходять від інших мов графічного моделювання. Ці відмінності призводять до довгих і важких аргументів щодо того, як слід використовувати UML.

UML, у його поточному стані, визначає нотацію та мета-модель. Позначення — це графічні елементи, які ви бачите в моделях; це графічний синтаксис мови моделювання. Наприклад, нотація діаграми класів визначає, як відображаються елементи та поняття, такі як клас, асоціація та множинність.

Звичайно, це призводить до питання, що саме мається на увазі під асоціацією чи множинністю чи навіть класом. Звичайне використання передбачає деякі неформальні визначення, але багато людей хочуть більшої суворості [19].

Ідея суворої специфікації та мов проектування найбільш поширена в області формальних методів. У таких техніках проекти та специфікації представлені за допомогою деякої похідної від обчислення предикатів. Такі визначення є математично точними і не допускають двозначності. Однак значення цих визначень аж ніяк не є універсальним. Навіть якщо ви можете

довести, що програма задовольняє математичну специфікацію, неможливо довести, що математична специфікація відповідає реальним вимогам системи.

Більшість мов графічного моделювання мають дуже мало суворості; їх позначення звертається до інтуїції, а не до формального визначення. Загалом, здається, це не завдало великої шкоди. Ці методи можуть бути неформальними, але багато людей все ще вважають їх корисними — і це корисність має значення [20].

Проте методисти шукають шляхи підвищення точності методів, не жертвуючи їх корисністю. Один із способів зробити це — визначити мета-модель: діаграму, зазвичай діаграму класів, яка визначає поняття мови.

Таблиця 2.1, невеликий фрагмент мета-моделі UML, показує взаємозв’язок між функціями.

Таблиця 2.1

Багато людей, які беруть участь у поточній розробці UML, цікавляться в першу чергу мета-моделю, особливо тому, що це важливо для використання UML та мови програмування. Проблеми нотації часто займають друге місце, що важливо мати на увазі, якщо ви коли-небудь спробуєте ознайомитися з самими документами стандартів.

Коли ми дивимося на діаграму UML, ми повинні мати на увазі, що загальний принцип UML полягає в тому, що будь-яка інформація може бути прихована для певної діаграми. Це придушення може відбуватися або взагалі — приховати всі атрибути — або конкретно — не показувати ні три класи. Таким чином, на діаграмі ми ніколи не зможете нічого зробити за її відсутністю. Якщо кратність відсутня, ми не можемо визначити, яке це

значення може бути. Навіть якщо мета-модель UML має значення за замовчуванням, наприклад для атрибутів, якщо ви не бачите інформації на діаграмі, це може бути пов'язано з тим, що вона йде за замовчуванням або тому, що вона «пригнічена» [21].

2.2 Планування робочого процесу, вибір методів

Вибір ітераційного методу

Однією з найбільших дискусій щодо процесу є те, що між стилями водоспаду та ітеративними стилями. Терміни часто вживаються неправильно, особливо тому, що ітерація вважається модною, тоді як процес водоспаду, здається, носить картаті штани. Як наслідок, багато проектів стверджують, що виконують ітераційну розробку, але насправді роблять каскад [22].

Істотна відмінність між ними полягає в тому, як ви розбиваєте проект на менші частини. Якщо у вас є проект, на який, на вашу думку, знадобиться рік, мало кому буде зручно казати команді піти на рік і повернутися, коли закінчиться. Потрібна деяка розбивка, щоб люди могли підійти до проблеми та відстежити прогрес.

Стиль водоспаду розбиває проект на основі діяльності. Щоб створити програмне забезпечення, ви повинні виконати певні дії: аналіз вимог, проектування, кодування та тестування. Таким чином, наш 1-річний проект може мати 2-місячний етап аналізу, за яким слідує 4-місячний етап проектування, за яким слідує 3-місячний етап кодування, а потім 3-місячний етап тестування.

Ітеративний стиль розбиває проект на підмножини функціональних можливостей. Ви можете витратити рік і розбити його на 3-місячні повторення. У першій ітерації ви взяли б чверть вимог і виконали повний життєвий цикл програмного забезпечення для цього кварталу: аналіз,

проектування, код і тестування. Наприкінці першої ітерації у вас буде система, яка виконує чверть необхідної функціональності. Потім ви зробили б другу ітерацію, щоб наприкінці 6 місяців у вас була система, яка виконує половину функціональних можливостей [23].

З розвитком водоспаду, як правило, існує певна форма формального передачі обелуговування між кожною фазою, але часто виникають зворотні потоки. Під час кодування може виникнути щось, що змусить вас переглянути аналіз і дизайн. Ви, звичайно, не повинні вважати, що весь дизайн закінчено, коли починається кодування. Неминуче, що аналіз і проектні рішення доведеться переглянути на наступних етапах. Однак ці зворотні потоки є винятком, і їх слід звести до мінімуму, наскільки це можливо.

За допомогою ітерації ви зазвичай бачите певну форму дослідження до початку справжніх ітерацій. Принаймні, це дозволить отримати уявлення про вимоги високого рівня: принаймні достатньо, щоб розбити вимоги на наступні ітерації. Деякі проектні рішення високого рівня також можуть прийматися під час розвідки. З іншого боку, хоча кожна ітерація має створювати готове до виробництва інтегроване програмне забезпечення, воно часто не досягає цього моменту і потребує періоду стабілізації, щоб виправити останні помилки. Крім того, деякі дії, такі як навчання користувачів, залишаються до кінця [24].

Ви можете не запускати систему у виробництво в кінці кожної ітерації, але система повинна бути якісної. Однак часто ви можете запускати систему у виробництво через регулярні проміжки часу; це добре, тому що ви отримуєте цінність від системи раніше і отримуєте зворотний зв'язок кращої якості. У цій ситуації ви часто чуєте про проект, який має кілька випусків, кожен з яких розбивається на кілька ітерацій.

Ітераційний розвиток зустрічається під багатьма назвами: інкрементальний, спіральний, еволюційний і спадає на думку джакузі. Різні

люди розрізняють їх, але ці відмінності не є ані загальноприйнятими, ані такими важливими в порівнянні ітерацією /водоспадом.

Ви можете мати гібридні підходи, існує поетапний життєвий цикл доставки, коли аналіз і високорівневе проектування виконуються спочатку в стилі водоспаду, а потім кодування та тестування поділяються на ітерації. Такий проект може мати 4 місяці аналізу та проектування, а потім чотири 2-місячні ітераційні збірки системи.

Більшість авторів програмного забезпечення за останні кілька років, особливо в об'єктно-орієнтованому співтоваристві, не люблять водоспадний підхід. З багатьох причин для цього найосновнішою є те, що дуже важко сказати, чи справді проект йде по шляху водоспаду. Занадто легко оголосити перемогу на ранніх етапах і приховати збій у розкладі. Зазвичай єдиний спосіб, за допомогою якого можна справді визначити, чи йдете ви на правильному шляху, — це створити перевірене інтегроване програмне забезпечення. Роблячи це багаторазово, ітеративний стиль краще попереджає, якщо щось піде не так [25].

Саме через це мною і було вибрано підхід з ітераціями.

Поширеною технікою з ітераціями є використання тайм-боксу. Це змушує ітерацію мати фіксований проміжок часу. Якщо виявляється, що ви не можете побудувати все, що збиралися побудувати під час ітерації, ви повинні вирішити вилучити деякі функції з ітерації; ви не повинні вказувати дату ітерації.

Більшість проектів, які використовують ітераційну розробку, використовують однакову довжину ітерацій протягом усього проекту; таким чином ви отримуєте регулярний ритм побудов.

Однією з найпоширеніших проблем ітераційної розробки є питання переробки. Ітеративна розробка явно передбачає, що ви будете переробляти та видаляти існуючий код під час наступних ітерацій проекту. У багатьох сферах, таких як виробництво, переробка вважається марною тратою. Але програмне забезпечення не схоже на виробництво; як результат, часто

ефективніше переробляти існуючий код, ніж виправляти код, який був погано розроблений. Ряд технічних прийомів може значно допомогти зробити переробку більш ефективною.

Прогнозне та адаптивне планування

Однією з причин того, що водоспад витримає, є прагнення до передбачуваності в розробці програмного забезпечення. Немає нічого більш неприємного, ніж не мати чіткого уявлення, скільки буде коштувати створення певного програмного забезпечення та скільки часу займе його створення.

Прогнозний підхід передбачає виконання роботи на початку проекту, щоб краще зрозуміти, що потрібно зробити пізніше. Таким чином, ви можете досягти точки, коли останню частину проекту можна буде оцінити з розумним ступенем точності. При прогнозному плануванні проект має два етапи. Перший етап складається з планів і його важко передбачити, але другий етап набагато більш передбачуваний, оскільки плани є.

Це не обов'язково чорно-біла справа. У міру реалізації проекту ви поступово отримуєте більшу передбачуваність. І навіть коли у вас є прогнозний план, все піде не так. Ви просто очікуєте, що відхилення стануть менш значними, коли буде розроблено надійний план.

Тим не менш, існує велика дискусія щодо того, чи можна передбачити багато програмних проектів. В основі цього питання лежить аналіз вимог. Одним з унікальних джерел складності програмних проектів є складність розуміння вимог до програмної системи. Більшість програмних проектів відчувають значний відтік вимог: зміни вимог на пізніх стадіях проекту. Ці зміни руйнують основи прогнозного плану. Ви можете боротися з цими змінами, заморозивши вимоги на ранньому етапі та не дозволивши внесення змін, але це створює ризик створення системи, яка більше не відповідає потребам своїх користувачів [26].

Ця проблема призводить до двох дуже різних реакцій. Один із шляхів — докласти більше зусиль до самого процесу вимог. Таким чином, ви можете отримати більш точний набір вимог, що зменшить відтік.

Інша школа стверджує, що відтік вимог неминучий, що багатьом проектам надто важко стабілізувати вимоги достатньою мірою, щоб використовувати прогнозний план. Це може бути пов'язано або з великими труднощами уявити, що може робити програмне забезпечення, або через те, що ринкові умови змушують непередбачувані зміни. Ця школа думок виступає за адаптивне планування, завдяки якому прогнозування розглядається як ілюзія. Замість того, щоб обманювати себе ілюзорною передбачуваністю, ми повинні зіткнутися з реальністю постійних змін і використовувати підхід до планування, який розглядає зміни як константу в програмному проекті. Ця зміна контролюється таким чином, щоб проект забезпечував найкраще програмне забезпечення, яке тільки може; але хоча проект піддається контролю, він не передбачуваний.

Різниця між прогнозним проектом і адаптивним проектом виявляється в багатьох аспектах, коли люди говорять про те, як проходить проект. Коли люди говорять про проект, який працює добре, тому що він йде за планом, це прогнозна форма мислення. Не можна сказати «за планом» в адаптивному середовищі, тому що план постійно змінюється. Це не означає, що адаптивні проекти не планують; вони зазвичай багато планують, але план розглядається як базовий план для оцінки наслідків змін, а не як передбачення майбутнього.

За допомогою прогнозного плану ви можете розробити контракт з фіксованою ціною/фіксованим обсягом. У такому договорі точно зазначено, що потрібно побудувати, скільки це буде коштувати і коли буде поставлено. Таке виправлення неможливо за допомогою адаптивного плану. Ви можете встановити бюджет і час доставки, але не можете визначити, які функції будуть надані. Адаптивний контракт передбачає, що користувачі будуть співпрацювати з командою розробників, щоб регулярно переоцінювати, яку функціональність необхідно створити, і скасують проект, якщо прогрес буде

занадто повільним. Таким чином, адаптивний процес планування може бути фіксованою ціною/змінною сферою застосування [27].

Природно, адаптивний підхід менш бажаний, оскільки будь-хто віддає перевагу більшій передбачуваності в програмному проекті. Проте передбачуваність залежить від точного, точного та стабільного набору вимог. Якщо ви не можете стабілізувати свої вимоги, прогнозний план базується на піску, і висока ймовірність того, що проект зрушиться з курсу. Маю визнати, що головним чином через брак досвіду тут мною був вибраний адаптивний підхід.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 ОРГАНІЗАЦІЙНА СТРУКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Дана Інтелектуальна система генерування дизайну інтер'єру на основі вподобань користувача була розроблена як веб-додаток.

Веб-програми - це динамічні веб-сайти в поєднанні з програмуванням на стороні сервера [28], які забезпечують такі функції, як взаємодія з користувачами, підключення до внутрішніх баз даних та отримання результатів для браузерів.

Прикладами веб-додатків є Інтернет-банкінг, соціальні мережі, онлайн-бронювання, додатки для електронної комерції / кошика, інтерактивні ігри, онлайн-навчання, Інтернет-опитування, блоги, Інтернет-форуми, системи управління контентом тощо.

Існує дві основні категорії кодування [29], сценарію та програмування для створення веб-додатків:

- Сценарії / кодування на стороні клієнта - Сценарії на стороні клієнта - це тип коду, який виконується або інтерпретується браузерами
- Сценарії / кодування на стороні сервера - Сценарії на стороні сервера - це тип коду, який виконується або інтерпретується веб-сервером

Клієнтські сценарії зазвичай переглядає будь-який відвідувач веб-сайту (у меню перегляду натисніть "Переглянути джерело", щоб переглянути вихідний код).

Нижче наведено деякі загальні технології сценаріїв на стороні клієнта [30]:

- HTML (мова розмітки HyperText)
- CSS (каскадні таблиці стилів)
- JavaScript (мова програмування високого рівня призначена для розробки частини клієнту, яку пізніше перетворили на мову сценаріїв загального призначення з можливостю використання як на стороні сервера, так і клієнта)
- Ajax (асинхронний JavaScript та XML)
- jQuery (JavaScript Framework Library - часто використовується в розробці Ajax)

Сценарії / кодування на стороні сервера - Сценарії на стороні сервера - це тип коду, який виконується або інтерпретується веб-сервером.

Сценарії на стороні сервера не можуть бути переглянуті або доступні для відвідувачів чи широкої громадськості.

Нижче наведено загальні технології сценаріїв на стороні сервера [31]:

- PHP + Laravel (дуже поширена мова сценаріїв на стороні сервера - з відкритим кодом на основі Linux / Unix - безкоштовний перерозподіл, зазвичай поєднується з базою даних MySQL)
- ASP (мова сценаріїв веб-сервера Microsoft (IIS))
- ASP.NET та ASP.MVC (фреймворк веб-застосунків Microsoft - наступник ASP)

- Ruby + Ruby on Rails (Веб-програма для програмування Ruby - безкоштовний перерозподіл)
- Perl (мова програмування високого рівня загального користування та мова сценаріїв на стороні сервера - безкоштовний перерозподіл - втратив свою популярність через PHP)
- Javascript + NodeJS/NextJS/ExpressJS (мова програмування високого рівня призначена для розробки частини клієнту, яку пізніше перетворили на мову сценаріїв загального призначення з можливістю використання як на стороні сервера, так і клієнта)
- Python + Django/Flask/FastAPI (мова програмування загального призначення та мова сценаріїв на стороні сервера - безкоштовне перерозподіл)

Бібліотеки програм [32] - це сукупність загальновживаних функцій, класів або підпрограм, що забезпечують легкість розробки та обслуговування, дозволяючи розробникам легко додавати або редагувати функціональні можливості в програму з рамковим або модульним типом.

Web Application Frameworks - це набори бібліотек програм, компонентів та інструментів, організованих в архітектурній системі, що дозволяє розробникам створювати та підтримувати складні проекти веб-додатків, використовуючи швидкий та ефективний підхід.

Фреймворки веб-програм розроблені для спрощення програмування та сприяння повторному використанню коду, викладаючи організацію та структуру папок, документацію, керівні принципи та бібліотеки (коди багаторазового використання для загальних функцій та класів).

Дії програми та логіка відокремлені від файлів HTML, CSS та дизайну. Це допомагає дизайнерам (без будь-якого досвіду програмування) мати

можливість редагувати інтерфейс та вносити зміни в дизайн без допомоги програміста.

Збірки базуються на модулі, бібліотеках та інструментах, що дозволяє програмістам легко обмінюватися бібліотеками та швидко та ефективно реалізовувати складні функціональні можливості та функції.

Структура допомагає створити найкраще практичне кодування з узгодженою логікою та стандартами кодування, а також надає іншим розробникам можливість ознайомитися з кодом за короткий час.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Для написання серверної частини веб-додатку було обрано скриптову мову Python.

Клієнтська частина написана на мові JavaScript.

Графічний інтерфейс побудований за допомогою HTML5 і CSS3.

3.3 Алгоритмізація та програмування програмних модулів

3.3.1 Реалізація інтерфейсу користувача

Графічний інтерфейс даної системи використовує HTML5 і CSS3 та JavaScript.

Мова розмітки гіпертексту (HTML) - основний мовний стандарт, що використовується для впорядкування та форматування веб-сторінок та інших документів у Всесвітній павутині. Він часто використовується разом із каскадними таблицями стилів (CSS) та JavaScript для створення повністю адаптивного дизайну веб-сторінки.

HTML визначає, які частини тексту - це абзаци основних текстів, заголовки, гіперпосилання, марковані / нумеровані списки, блок-лапки, курсив, напівжирний шрифт тощо, а CSS визначає, які ці частини візуально

виглядають на зовнішній панелі. JavaScript, навпаки, додає на сторінку динамічні елементи, такі як спливаючі вікна, анімована графіка, прокрутка банерів та багато іншого.

HTML використовується з самого початку Інтернету, яким ми його знаємо сьогодні. У 1991 році, коли Тім Бернерс-Лі представив Інтернет[33], він також винайшов систему, за допомогою якої веб-браузери могли перекладати текст на візуальні веб-сторінки. Оригінальний дизайн HTML був відносно простим (він містив лише 18 тегів) і прийняв структуру позначення Стандартною узагальненою мовою розмітки (SGML).

З часу свого існування HTML бачив багато оновлень від Консорціуму всесвітньої павутини (W3C). До наступних версій додано 140 тегів HTML, найвідоміший з яких - HTML5. У 2014 році HTML5 представив семантичні теги для частин веб-сайту, які раніше не були визнані, включаючи заголовок, нижній колонтитул, меню навігації, а також аудіо- та відео-елементи серед інших.

CSS скорочується від Cascading Style Sheets (Каскадні таблиці стилів) і є найкращим способом налаштування зовнішнього вигляду веб-сайту. Таблиці стилів визначають колір, розмір і положення тексту та інших тегів HTML, тоді як файли HTML визначають вміст та спосіб його організації. Їх розділення дозволяє змінити колірну схему без необхідності переписувати весь веб-сайт.

Каскадування означає, що стиль, застосований до батьківського елемента, також застосовуватиметься до всіх дочірніх елементів у батьківському елементі. Наприклад, встановлення кольору основного текету означатиме, що всі заголовки та абзаци в тілі також будуть однакового кольору.

Визначення та використання стилів [34]

Існує три основних способи включення таблиці стилів для веб-сторінки або сайту:

- Встановлення стилю вказання значення напряму в атрибуті тегу, <img src=’/image.png’ style=”text-align:center;”> наприклад, де «style=”text-align:center;”» - вказання стилю
- Використання <style> тегу в заголовці HTML
- Створення та посилання на зовнішній файл CSS через тег <link>

Основні таблиці стилів зазвичай змінюють зовнішній вигляд HTML-тегів, таких як <body> та <p>. Використовуючи файли CSS або таблиці стилів у заголовку, ми також можемо визначити класи стилів і застосувати їх до будь-якого елемента, використовуючи class = "?" атрибут, але це виходить за рамки цього простого посібника.

JavaScript - це мова програмування, що використовується переважно веб-браузерами для створення динамічного та інтерактивного досвіду для користувача. Більшість функцій та додатків, що роблять Інтернет необхідним для сучасного життя, кодуються у певній формі JavaScript.

Найбільш ранні втілення JavaScript були розроблені наприкінці 1990-х років для веб-браузера Netscape Navigator. На той час веб-сторінки були статичними, пропонуючи мало взаємодії з користувачами, окрім клацання

посилань та завантаження нових сторінок. Вперше JavaScript активував анімацію, адаптивний вміст та перевірку форми на сторінці.

Протягом багатьох років JavaScript функціонував лише на обмеженій кількості браузерів. Microsoft Internet Explorer, найбільша база браузерів, підтримувала JavaScript лише набагато пізніше. Натомість Microsoft створила власний власний сценарій на стороні клієнта під назвою JScript. У перші дні веб-розробки програмісти, які бажали створювати динамічні веб-сайти, часто були змушені вибирати одну сім'ю браузерів над іншими. Це було менш ніж ідеально, оскільки робило Інтернет менш загальнодоступним.

JavaScript не став стандартизованим та широко прийнятим до 1999 року. Навіть після стандартизації сумісність браузерів залишалася проблемою протягом десяти років.

JavaScript - це те, що називається сценарієм на стороні клієнта. Більшість веб-додатків, таких як пошукова система, працюють завдяки взаємодії між пристроєм користувача (наприклад, комп'ютером, телефоном або планшетом) та віддаленим сервером. Програмне забезпечення на віддаленому сервері надсилає інформацію клієнту (тобто машині користувача), а програмне забезпечення на стороні клієнта зчитує інформацію та відображає веб-сторінку на екрані.

Клієнтський сценарій - це мова програмування, яка повністю виконує свої завдання на машині клієнта, і для її функціонування не потрібно взаємодіяти з сервером. Наприклад, якщо на вашому комп'ютері завантажена веб-сторінка, а постачальник послуг Інтернету не працює, ви все ще можете взаємодіяти з веб-сторінками, вже завантаженими у ваш браузер. Однак ви не зможете переходити до нових веб-сторінок або отримувати доступ до будь-яких даних, розташованих віддалено.

3.3.2 Забезпечення інтерфейсу з базою даних

Для написання серверної частини веб-додатку було обрано скриптову мову Python, яка співпрацює в парі з PostgreSQL.

Абстракція сховища бази даних, що найчастіше використовується у веб-розробці на Python, — це набори реляційних таблиць.

Реляційні бази даних зберігають дані в серії таблиць. Взаємозв'язки між таблицями задаються як зовнішні ключі. Зовнішній ключ — це унікальне посилання з одного рядка реляційної таблиці на інший рядок таблиці, яка може бути тією ж таблицею, але найчастіше є іншою таблицею.

Реалізації зберігання баз даних відрізняються за складністю. SQLite, база даних, що входить до складу Python, створює один файл для всіх даних у базі даних. Інші бази даних, такі як PostgreSQL, MySQL, Oracle і Microsoft SQL Server, мають складніші схеми збереження, пропонуючи додаткові розширені функції, корисні для зберігання даних веб-додатків. Ці розширені функції включають, але не обмежуються ними:

1. Реплікація даних між головною базою даних і одним або декількома підлеглими екземплярами лише для читання
2. Розширені типи стовпців, які можуть ефективно зберігати напівструктуровані дані, такі як JavaScript Object Notation (JSON)
3. sharding, що дозволяє горизонтальне масштабування кількох баз даних, кожна з яких використовується як екземпляри читання-запису за рахунок затримки узгодженості даних
4. моніторинг, статистика та інша корисна інформація під час виконання для схем і таблиць бази даних

Зазвичай веб-додатки починаються з одного екземпляра бази даних, такого як PostgreSQL, із простою схемою. З часом схема бази даних перетворюється на більш складну структуру з використанням міграції схем і розширених функцій, таких як реплікація, розподілення та моніторинг, стають все більш корисними, оскільки використання бази даних збільшується відповідно до потреб користувачів програми.

PostgreSQL є рекомендованою реляційною базою даних для роботи з веб-додатками Python відповідно до джерела [35]. Набір функцій, активна розробка та стабільність PostgreSQL сприяють його використанню в якості серверної частини для мільйонів додатків, які існують сьогодні в Інтернеті.

Щоб працювати з реляційною базою даних за допомогою Python, потрібно використовувати бібліотеку коду. Найпоширенішими бібліотеками для реляційних баз даних є:

- psycopg2 (вихідний код) для PostgreSQL.
- MySQLdb (вихідний код) для MySQL. Зауважте, що розробка цього драйвера в основному заморожена, тому оцінка альтернативних драйверів є розумною, якщо ви використовуєте MySQL як бекенд.
- cx_Oracle для бази даних Oracle (вихідний код).

Підтримка SQLite вбудована в Python 2.7+ відповідно до [36], тому окрема бібліотека не потрібна. Просто «імпортуйте sqlite3», щоб розпочати взаємодію з базою даних одного файлу.

PostgreSQL — вибір бази даних за замовчуванням для багатьох розробників Python, включаючи команду Django під час тестування Django ORM відповідно до джерела [37].

PostgreSQL часто розглядають як більш надійну і стабільну функцію в порівнянні з MySQL, SQLServer і Oracle. Усі ці бази даних є розумним вибором. Однак, оскільки PostgreSQL зазвичай використовується розробниками Python, драйвери та приклад коду для використання бази даних, як правило, краще задокументовані та містять менше помилок для типових сценаріїв використання.

Якщо ви спробуєте використовувати базу даних Oracle з Django, ви побачите, що прикладів коду для цього налаштування набагато менше, ніж у серверних налаштуваннях PostgreSQL.

Щоб абстрагувати зв'язок між таблицями та об'єктами, багато розробників Python використовують об'єктно-реляційний картограф (ORM), щоб перетворити реляційні дані з PostgreSQL на об'єкти, які можна використовувати в їхній програмі Python. Наприклад, хоча PostgreSQL надає реляційну базу даних, а psycopg є загальним конектором бази даних, існує багато ORM, які можна використовувати з різними веб-фреймворками, як показано в таблиці 3.1.

Таблиця 3.1

Web framework	Bottle	Flask	Flask	Django
ORM	Peewee	Pony ORM	SQLAlchemy	Django ORM
Database connector	psycopg	psycopg	psycopg	psycopg
Relational database	PostgreSQL	PostgreSQL	PostgreSQL	PostgreSQL

3.3.3 Діаграма діяльності та реалізація алгоритму

Діаграми діяльності - одна з найбільш доступних діаграм UML [37], оскільки вони використовують символи, подібні до широко відомих позначень блок- схем; тому вони корисні для опису процесів для широкої аудиторії. Насправді діаграми діяльності кореняться в блок-схемах, а також діаграмах стану UML, діаграмах потоків даних та мережах Петрі.

Діаграми активності показують дії високого рівня, прикуті до ланцюга, щоб представляти процес, що відбувається у вашій системі. Діаграми діяльності особливо вдалі для моделювання бізнес-процесів.

Бізнес-процес - це сукупність узгоджених завдань [38], що дозволяють досягти бізнес-мети, наприклад, доставка замовлення клієнтів. Деякі інструменти управління бізнес-процесами (BPM) дозволяють визначати бізнес-процеси за допомогою діаграм діяльності або подібних графічних позначень (наприклад, BPMN), а потім виконувати їх. Це дозволяє визначити

та виконати, наприклад, процес затвердження платежу, коли один із етапів викликає веб-службу затвердження кредитної картки, використовуючи прості графічні позначення, такі як діаграми діяльності.

Між початковим вузлом та кінцевим вузлом діяльності є дії, які намальовані у вигляді округлених прямокутників. Дії - це важливі кроки, які відбуваються у загальній діяльності, наприклад, Виберіть тип облікового запису, Введіть дані про автора тощо. Дією може бути виконана поведінка, обчислення або будь-який ключовий крок у процесі.

Потік діяльності відображається за допомогою стрілочкових ліній, які називаються краями або контурами. Стрілка на краю діяльності показує напрямок потоку від однієї дії до іншої. Лінія, що заходить у вузол, називається вхідним ребром, а лінія, що виходить із вузла, - вихідним ребром. Грані нав'язують дії разом, щоб визначити загальний потік активності: спочатку початковий вузол стає активним, потім перший ромбоподібний вузол називається рішенням, аналогічним оператору if-else у коді.

Зверніть увагу, що з рішення, наведеного на малюнку вище, є два вихідних ребра, кожен позначений булевими умовами. З вузла прийняття рішення прослідковується лише один край залежно від того, чи уповноважений автор. Другий ромбоподібний вузол називається злиттям. Вузол злиття поєднує краї, починаючи від вузла прийняття рішення, позначаючи кінець умовної поведінки.

Блок-схема зображена на Рис 3.1 відображає алгоритм перегляду даних.

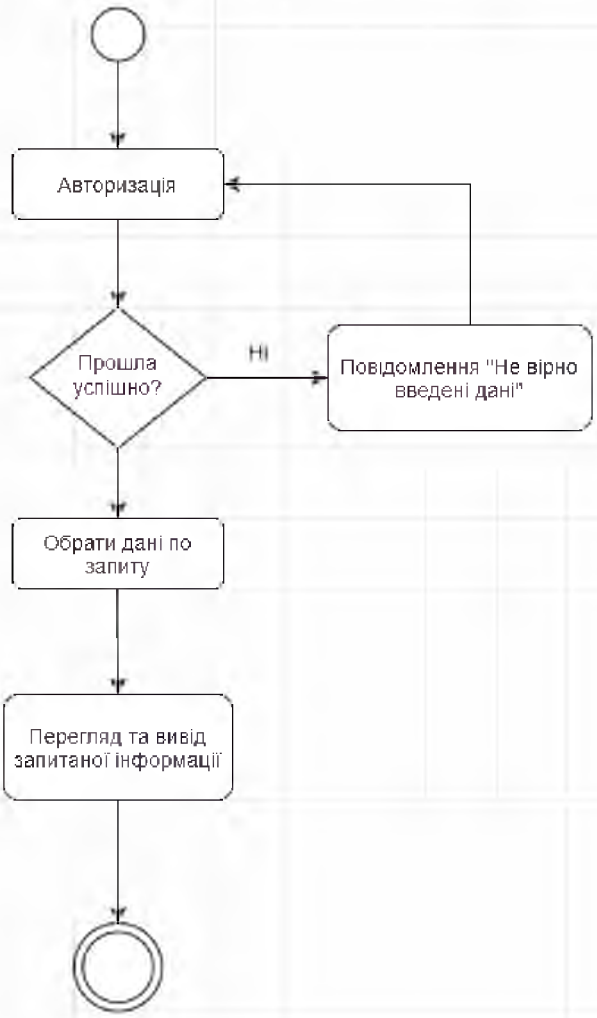

Рис 3.1 Блок-схема «Перегляд даних»

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

В даній роботі було розроблено веб-додаток, який має на меті отримання користувачем спеціально підібраного під його уподобання дизайну, а також отримання даних, щодо проведення ремонту простору за вказаним дизайном.

Серед основних кроків, користувачу для початку потрібно вказати основні параметри простору, як зображено на рис 4.1.

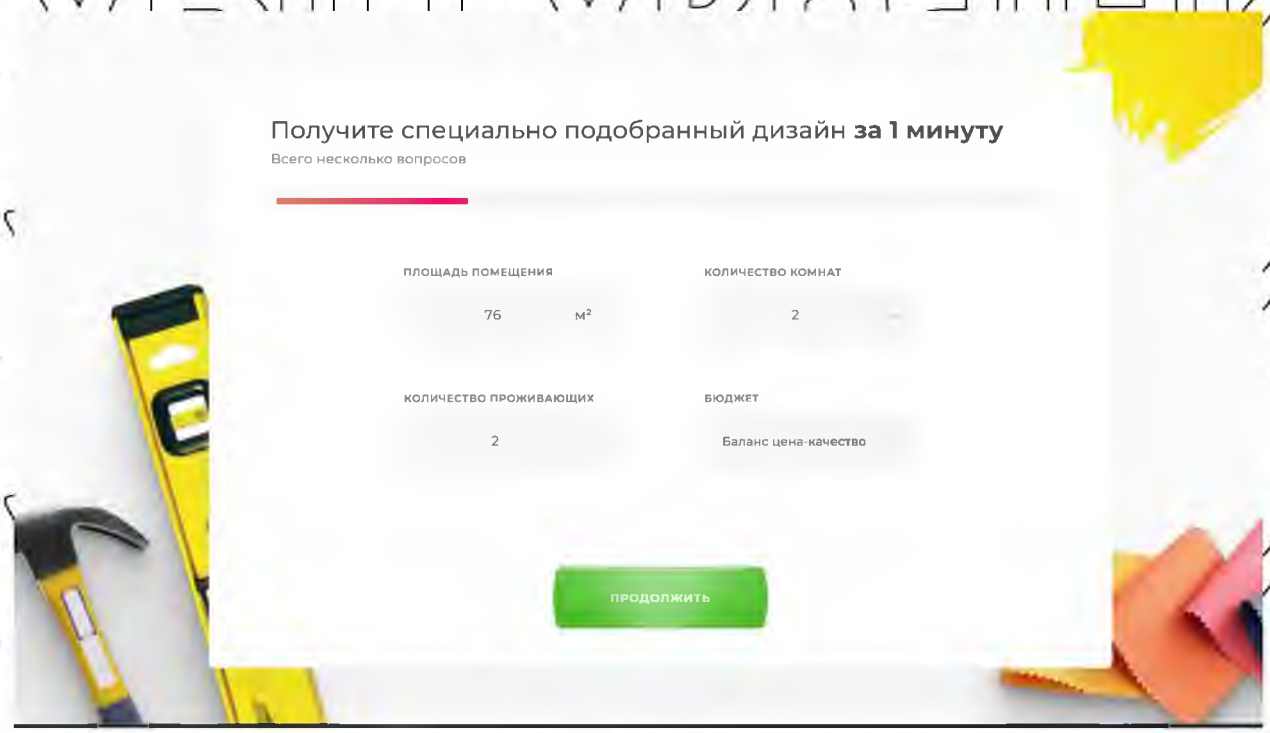

Рис 4.1 Перший крок для підбору дизайну

Під час підбору дизайну все вище вказані критерії мають вагу, оскільки одні види дизайну розраховані на велику кількість вільного простору в той час, як інші є менш вимогливими в цьому плані.

Кількість людей, які живуть у просторі напряму впливаю на кількість меблів, які потрібні (наприклад шкафи), оскільки усім людям потрібні речі для життя. Бюджет же вказує напряму чи має користувач можливість оплатити дизайн на свій розмір простору з усіма вказаними вище параметрами.

Наступним кроком є вибір кольорової гами та матеріалу стінок: перший варіант також дозволяє відокремити дизайн, які не підходять для користувача. Матеріал стінок може здатися не важливим на перший погляд, але якщо є особливий матеріал стінок, то є можливість підбору дизайну, який саме орієнтований на них. Сам крок показаний на Рис 4.2

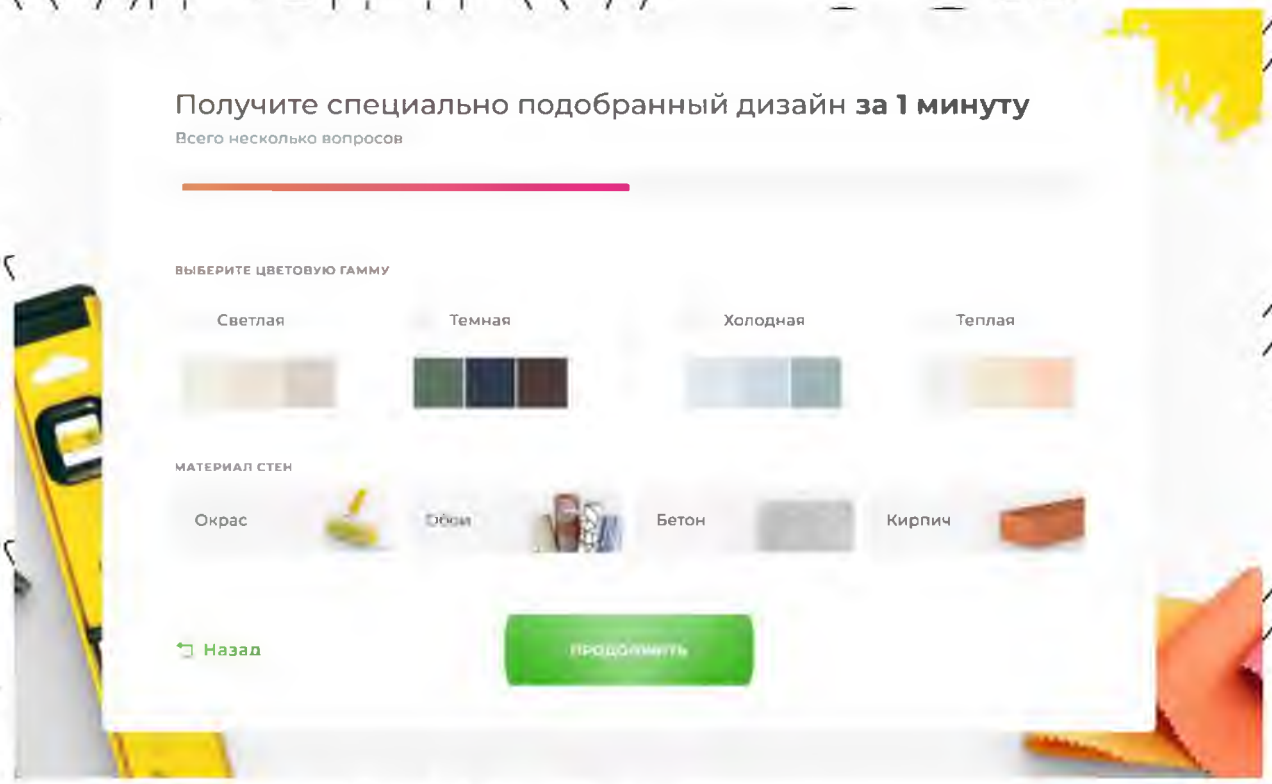

Рис 4.2 Другий крок для підбору дизайну

На сьогоднішній день є доволі велика кількість різноманітних видів дизайну, а тому навіть вказавши всі дані як на першому кроці, так і на другому, може бути не достатньо для відокремлення самого найкращого варіанту для користувача.

Тому на випадок коли система підібрала за вказаними даними декілька варіантів дизайну, користувачу надасться можливість вибрати між ними, що йому більше до вподоби. Цей крок показаний на Рис 4.3

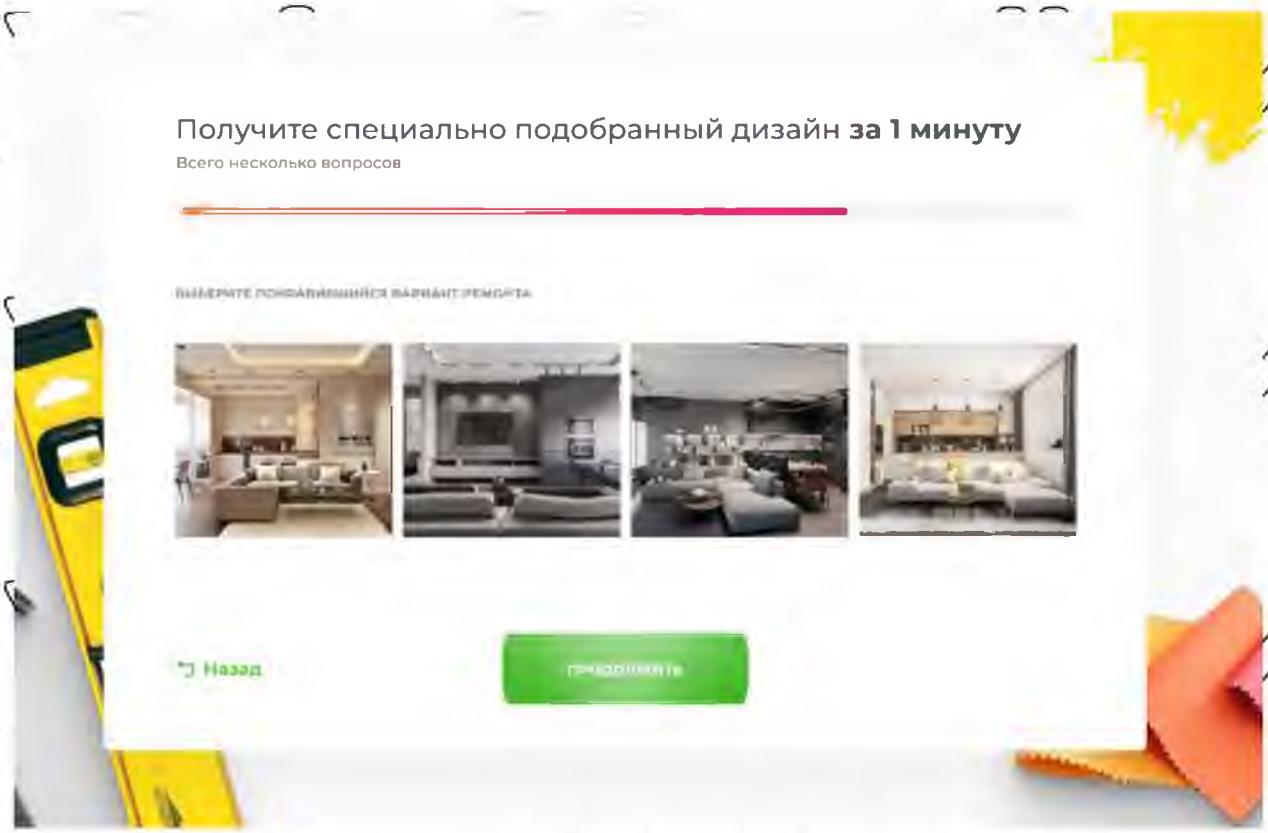

Рис 4.3 Третій крок для підбору дизайну

На четвертому кроці користувачу потрібно вести ще своє місто та бюджет (це потрібно щоб підібрати найкращі варіанти серед магазинів). Рис 4.4

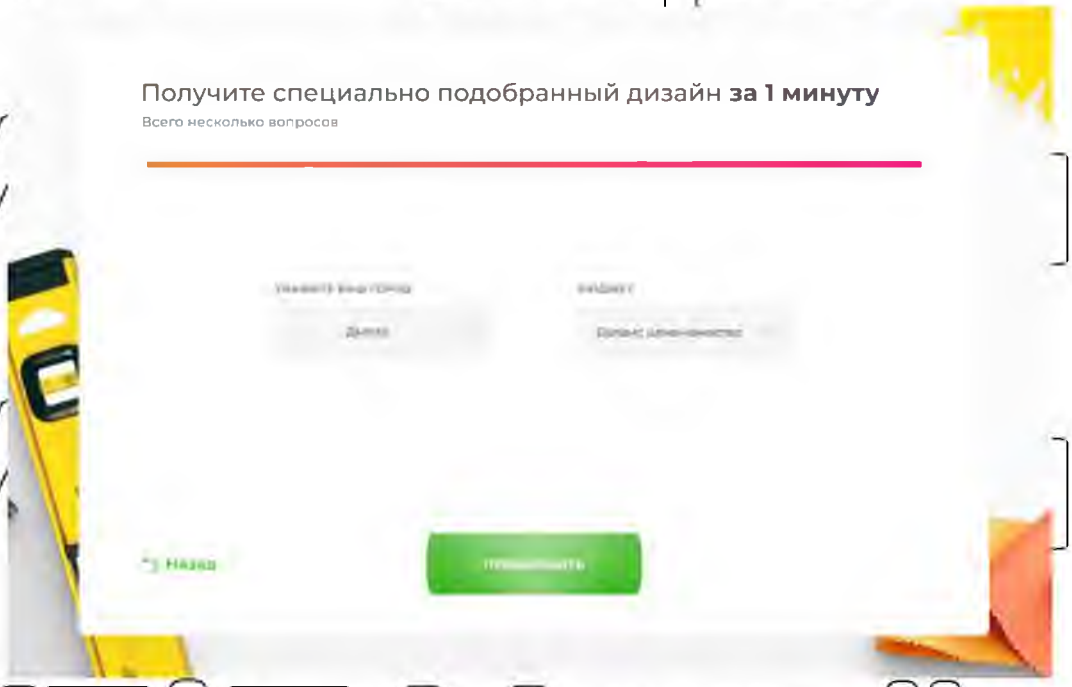

Рис 4.4 Четвертий крок для підбору дизайну

Після пройдених усіх вище кроків, користувачу буде виведена загальна сума для проведення даного дизайну за веденими вище параметрами, а також

підібрані магазини, в яких можна купити потрібні матеріали як показано на Рис 4.5:

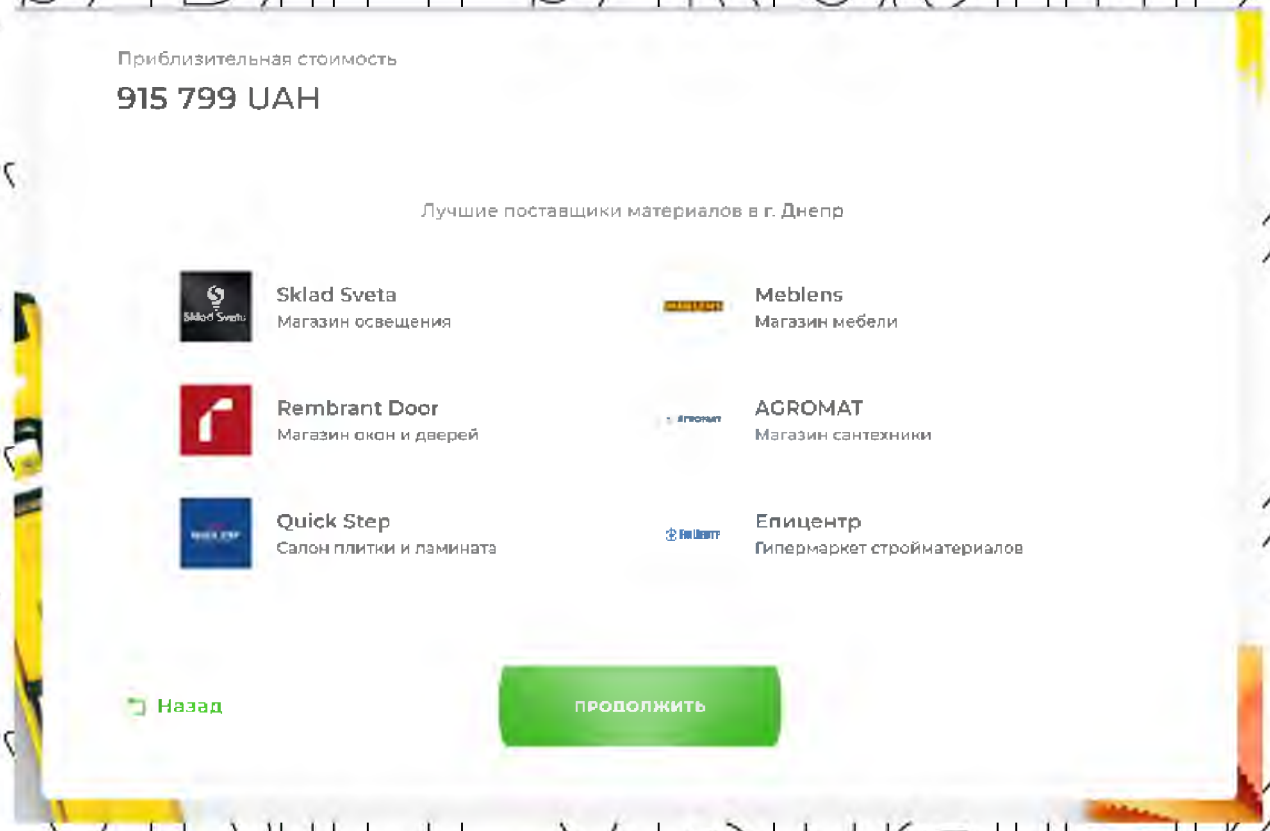

Рис 4.5 Вивід списку магазинів та приблизної вартості послуги

0.2 Вимоги до апаратного забезпечення

В ході тестування системи було визначено, що потрібен мінімальний набір технічних вимог для надійного та швидкого користування системою, що дасть змогу отримувати актуальні дані. Вимоги наведені в таблиці 4.1

Таблиця 4.1

Процесор	Intel Core
SDD	8 GB
Пам’ять оперативна	6 GB
Ethernet-порт	2000 mb/s

Мінімальні характеристики клієнтського пристрою

Мінімальні технічні вимоги до апаратного забезпечення користувача для роботи з інформаційною системою:

- Процесор Intel Core;
- Оперативна пам’ять 4 ГБ;
- Відеокарта
- Звукова карта
- Активне під’єднання до інтернету
- Клавіатура
- Миша
- Монітор

0.3 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Потрібне надійне під’єднання до мережі Інтернет, а також мати встановлений веб-браузер на пристрої, що дозволить максимально швидко та безпечно користуватися даним веб-додатком.

0.4 СКЛАД ІНСТАЛЯЦІЙНОГО ПАКЕТУ

Використовуючи дану систему, користувач не може зтикнутися з проблемами її установки, тому що це веб-застосунок і лише потрібно перейти за посиланням у браузері.

1. ВИСНОВКИ

Метою даної магістерської роботи є визначення та розробка ефективних засобів для обробки інформації для прийняття рішень щодо вибору дизайну інтер'єру.

Ця система генерування дизайну була розроблена на основі даних, які були зібрані в ході аналізу предметної області, існуючих рішень та статистичних даних попиту користувачів в сфері дизайну інтер'єру.

У результаті виконання магістерської роботи було проведено:

- Аналіз предметної області дизайну інтер'єру;
- Розроблено діаграму діяльності для обробки даних;
- Створено сховище даних в тому ж MS SQL Server;
- Було розроблено та впроваджено у вигляді системи ефективний засіб для обробки інформації для прийняття рішень щодо вибору дизайну інтер'єру

Результати проведеної роботи дають змогу будь-якому користувачеві системи використовувати дану систему для аналізу та найбільш кращого варіанту дизайну під свої уподобання в Україні. Система має змову продумати та підібрати дизайн на основі уподобань користувача, серед яких об'єм простору, кількість кімнат, людей які живуть у просторі, кольорової гами та інші.

Розроблена підсистема підтримки може зменшити витрати в часі та коштах для підбору та оплати послуг при підборі дизайну інтер'єру.

На мою думку, мета та задача, що була поставлена в даній магістерській роботі була виконана успішно.

2. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Warner C. How interior design can affect the health and well-being of seniors [Електронний ресурс] / Cynthia Warner // McKnight's Senior Living. – Режим доступу: https://www.mcknightsseniorliving.com/home/columns/guest-columns/how-interior-design-can-affect-the-health-and-well-being-of-seniors/
2. Кузнецова І. О. Інноваційні напрями в дизайні інтер'єру / І. О. Кузнецова, І. О. Русаков, О. В. Руденко, К. О. Гербич // Теорія та практика дизайну. - 2017. - Вип. 12. - С. 157-165. - Режим доступу: http://nbuv.gov.ua/UJRN/tprd_2017_12_17.
3. 27 Best Online Interior Design Software Programs [Електронний ресурс] // Home Stratosphere– 2020 – Режим доступу: https://www.homestratosphere.com/online-interior-design-software/
4. Кардаш О. В. Дизайн інтер'єру як об'єкт теоретичних досліджень / О. В. Кардаш // Теорія та практика дизайну. – 2012. – Вип. 1. – С. 33-40. – Режим доступу: http://nbuv.gov.ua/UJRN/tprd_2012_1_8
5. Принципи і методи системного аналізу [Електронний ресурс]. URL: https://stud.com.ua/45001/investuvannya/printsipi_metodi_sistemnogo_analizu
6. Desfray P., Raymond G. - Modeling Enterprise Architecture with TOGAF. A Practical Guide Using UML and BPMN - 2014+
7. Furniture for Interior Design - Booth, Sam & Plunkett, Drew Published in 2014 by Laurence King Publishing Ltd.
8. Interior Design Reference & Specificatron Book Updated & Revisegners Need to Know Every Day, The - Grimley, Chris & Love, Mimi Published in 2018 by Rockport Publishers, Inc.
9. Mola F.Z. - 150 Best of the Best House Ideas – 2016 Published by Harper Design (September 27, 2016)

10. New York School of Interior Design _ Home - Fisher, Ellen S. & Renzi, Jen by Clarkson Potter; Illustrated edition (March 27, 2018)
11. Made for Living: Collected Interiors for All Sorts of Styles by Clarkson Potter; Illustrated edition (October 27, 2020)
12. Down to Earth: Laid-back Interiors for Modern Living by Harry N. Abrams; Illustrated edition (October 8, 2019)
13. Feels Like Home: Relaxed Interiors for a Meaningful Life by Harry N. Abrams (October 19, 2021)
14. The Curated House: Creating Style, Beauty, and Balance by Rizzoli; Illustrated edition (October 27, 2015)
15. Веб сайт Stuccco [Електронний ресурс]. URL: www.stuccco.com
16. UML Distilled: A Brief Guide to the Standard Object Modeling Language 3rd Edition by Martin Fowler Addison-Wesley Professional; 3rd edition (September 15, 2003)
17. Systems Engineering Demystified: A practitioner's handbook for developing complex systems using a model-based approach published by Jon Holt
18. Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development, 3rd edition by Craig Larman, published by Prentice Hall; 3 edition (October 30, 2004)
19. OCUP Certification Guide: UML 2.5 Foundational Exam; Preparing for the OCUP 2 by Michael Jesse Chonoles, published by Morgan Kaufmann; 1st edition (September 8, 2017)
20. Large-scale software architecture: a practical guide using UML by Richard Anthony and Jeff Garland by Wiley; 1st edition (December 30, 2002)
21. UML 2 Certification Guide: Fundamental and Intermediate Exams (The MK/OMG Press) 1st Edition Publisher, published by Morgan Kaufmann; 1st edition (December 18, 2006)
22. Learning UML 2.0: A Pragmatic Introduction to UML 1st Edition by Russ Miles, published by O'Reilly Media; 1st edition (May 16, 2006)

23. UML 2.0 in a Nutshell: A Desktop Quick Reference by Dan Pilone, published by O'Reilly Media; 2nd edition (July 12, 2005)

24. UML 2 For Dummies 1st Edition by Michael Jesse Chonoles, published by Amazon.com (July 2, 2003)

25. Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition by Erich Gamma, published by Addison-Wesley Professional; 1st edition (November 10, 1994)

26. Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software 2nd Edition by Eric Freeman, published by O'Reilly Media; 2nd edition (December 29, 2020)

27. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems 1st Edition by Martin Kleppmann, published by O'Reilly Media; 1st edition (April 18, 2017)

28. Курс лекцій з дисципліни «Web-проектування» для студентів спеціальності 7.080402 – «інформаційні технології проектування»/ Уклад.: О.П. Цурін. – К.: НТУУ «КПІ», 2011 р. – С. 14-18.

29. What Is Full Stack Development? [Електронний ресурс] // Trio. – 2020. – Режим доступу до ресурсу: https://trio.dev/blog/full-stack-development.

30. Всесвітня павутина Тіма Бернерса-Лі [Електронний ресурс] // ЦЕЙ ДЕНЬ В ІСТОРІЇ. – 2006. – Режим доступу до ресурсу: https://www.jnsm.com.ua/h/0517M/.

31. Top 10 Frameworks for Web Applications [Електронний ресурс] // GeeksforGeeks. – 2020. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/top-10-frameworks-for-web-applications/.

32. Бібліотека підпрограм [Електронний ресурс] // Wikipedia. – 2017. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%BF%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC

33. Питання для інтерв'ю веб-розробки [Електронний ресурс] // uk.education-wiki. – 2021. – Режим доступу до ресурсу: https://uk.education-wiki.com/7971701-web-development-interview-questions.

34. HTML Styles - CSS [Електронний ресурс] // w3schools. – 2021. – Режим доступу до ресурсу: https://www.w3schools.com/html/html_css.asp.

35. Databases [Електронний ресурс] // fullstackpython. – 2015 – Режим доступу до ресурсу: https://www.fullstackpython.com/databases.html.

36. sqlite3 — DB-API 2.0 interface for SQLite databases [Електронний ресурс] // docs.python.org. – 2020. – Режим доступу до ресурсу: https://docs.python.org/2.7/library/sqlite3.html.

37. PostgreSQL [Електронний ресурс] // fullstackpython. – 2016. – Режим доступу до ресурсу: https://www.fullstackpython.com/postgresql.html.

38. Марголін О. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] / Олександр Марголін // Evergreen. – 2021. – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html.

39. Бізнес-процеси підприємства: сутність та класифікація [Електронний ресурс] // SOPHUS – 2016 – Режим доступу до ресурсу: http://sophus.at.ua/publ/2016_12_14_kampodijsk/sekcija_section_3_2016_12_14/biznes_procesi_pidpriemstva_sutnist_ta_klasifikacija/136-1-0-2192.

40. Microsoft SQL Server – реализация языка программирования SQL [Електронний ресурс]. URL: http://progopedia.ru/implementation/microsoft-sql-server/

41. Моделі і методи проектування інформаційних систем [Електронний ресурс]. URL: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8cc4d8e5e/20160217112601/170352/index.html

42. Побудова логічної моделі даних [Електронний ресурс]. URL: https://studfile.net/preview/7144845/page:29/

43. OLAP-технологии [Електронний ресурс]. URL: https://kaidev.ru/Pages/Article.aspx?p=OlapAbout

44. MySQL 4 - Строкові функції [Електронний ресурс] Режим доступу: http: //www.codenet.ru/db/mysql/mystring4

45. Web-Розробка [Електронний ресурс] Режим доступу: http://fcit.tneu.org/webrozrobka/

46. Стратегія и основні кроки при розробці web-сайту. - Режим доступу: http://ruszura.in.ua/neobhidno-znaty/stratehiya-i-osnovni-kroky-pry-rozrobtsiwebsajta.html.

47. Технології розробки та тестування програм [Електронний ресурс]. Режим доступу: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php

48. Створення сайту, web-дизайн. - Режим доступу: http://www.artus.ru/.

49. Двигуни для сайтів, платні і безкоштовні CMS системи, каталог систем управління сайтами. – Режим доступу: http://www.cmsmagazine.ru/catalogue/