НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмне забезпечення для дизайну ітер'єрів приміщень»

Виконав:
студент IV курсу, групи КП-11
Бондарчук Данііл Сергійович _____

Керівник:
доцент кафедри ПЗКС, к.т.н., доцент,
Новак Дмитро Сергійович _____

Консультант з нормоконтролю:
доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
доцент кафедри ІТШІК, к.т.н., доцент,
Мошенський Андрій Олександрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2024 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Бондарчуку Даніілу Сергійовичу

1. Тема проєкту «Програмне забезпечення для дизайну інтер'єру приміщень», керівник проєкту Новак Дмитро Сергійович, затверджені наказом по університету № 1808-С від «29» травня 2025 р.

2. Термін подання студентом проєкту «13» червня 2025 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - огляд предметної області;
   - обґрунтування вибору засобів розроблення;
   - архіткетура та дизайн додатку;
   - аналіз розробленого додатка.

5. Перелік обов'язкового графічного матеріалу:
   - діаграма діяльності основного сценарію користувача (креслення);
   - діаграма прецендентів (креслення);
   - архітектура програмного забезпечення (плакат);
   - основні дії користувача додатку "Tumba"(плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	08.03.2025	
2.	Розроблення та узгодження технічного завдання	18.03.2025	
3.	Розроблення структури програмного забезпечення	02.04.2025	
4.	Підготовка матеріалів першого розділу дипломного проєкту	15.04.2025	
5.	Розроблення дизайну системи та графічних елементів	22.04.2025	
6.	Підготовка матеріалів другого розділу дипломного проєкту	01.05.2025	
7.	Програмна реалізація дипломного проєкту	16.05.2025	
8.	Тестування програмного забезпечення	26.05.2025	
9.	Підготовка матеріалів третього розділу дипломного проєкту	27.03.2025	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	01.06.2025	
11.	Підготовка графічної частини дипломного проєкту	02.06.2025	
12.	Оформлення документації дипломного проєкту	04.06.2025	

Студент Даніїл БОНДАРЧУК

Керівник проєкту Дмитро НОВАК

АНОТАЦІЯ

У сучасних умовах зростання популярності доповненої реальності виникає потреба у створенні інструментів, які полегшують дизайн інтер'єрів за допомогою інтеракції з 3D-об’єктами. Це дозволяє користувачам не лише оцінити просторові рішення, але й експериментувати з різними варіантами меблів та декору без необхідності фізичних змін у приміщенні. У рамках дипломного проєкту було розроблено додаток "Tumba", що дозволяє користувачам візуалізувати інтер’єрні рішення в реальному просторі за допомогою AR-технологій.

Основними функціональними можливостями "Tumba" є інтерактивне розміщення 3D-об’єктів, налаштування їх розмірів, кольору та текстур, а також взаємодія з віртуальними предметами через зручний користувацький інтерфейс.

Серед переваг застосунку – використання Unity та ARCore для реалізації високоякісної візуалізації та забезпечення стабільної роботи навіть на мобільних пристроях. Оптимізація графічних ресурсів дозволяє забезпечити плавну роботу навіть при великій кількості 3D-об’єктів у сцені. Крім того, інтуїтивний інтерфейс сприяє швидкому освоєнню програми як новачками, так і досвідченими дизайнерами.

Таким чином, "Tumba" виступає ефективним інструментом для створення персоналізованих інтер’єрних рішень, використовуючи можливості доповненої реальності для інтерактивного проектування та візуалізації простору. Це рішення дозволяє значно спростити процес прийняття рішень під час проєктування інтер’єру та мінімізувати ризики помилок у кінцевому результаті.

ABSTRACT

In the current context of growing popularity of augmented reality, there is a growing demand for tools that facilitate interior design through interaction with 3D objects. This approach allows users not only to assess spatial solutions but also to experiment with various furniture and decor options without making physical changes to the space. As part of the diploma project, the "Tumba" application was developed, enabling users to visualize interior design solutions in real space using AR technology.

The main functional features of "Tumba" include interactive placement of 3D objects, adjustment of their size, color, and texture, as well as interaction with virtual objects through a user-friendly interface. Additionally, users can create collections of objects, allowing them to save and reuse previously created interior compositions.

Among the advantages of the application is the use of Unity and ARCore to implement high-quality visualization and ensure stable performance even on mobile devices. Graphics optimization provides smooth performance even with a large number of 3D objects in the scene. Moreover, the intuitive interface facilitates quick learning of the program for both beginners and experienced designers.

Thus, "Tumba" serves as an effective tool for creating personalized interior design solutions, utilizing the potential of augmented reality for interactive design and space visualization. This solution significantly simplifies the decision-making process during interior design and minimizes the risk of errors in the final result.

ДП.045480-01-90 Програмне забезпечення для дизайну інтер’єрів приміщень.
Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045480-02-91	Програмне забезпечення	5	
	для дизайну інтер’єрів		
	приміщень. Технічне		
	завдання		
ДП.045480-03-81	Програмне забезпечення	51	
	для дизайну інтер’єрів		
	приміщень.		
	Пояснювальна записка		
ДП.045480-04-51	Програмне забезпечення	4	
	для дизайну інтер’єрів		
	приміщень. Програма та		
	методика тестування		
ДП.045480-05-34	Програмне забезпечення	4	
	для дизайну інтер’єрів		
	приміщень. Керівництво		
	користувача		
ДП.045480-06-99	Програмне забезпечення	1	
	для дизайну інтер’єрів		
	приміщень. Блок-схема		
	алгоритму		
ДП.045480-07-99	Програмне забезпечення	1	
	для дизайну інтер’єрів		
	приміщень. Діаграма		
	прецендентів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045480 -08-98	Програмне забезпечення	1	
	для дизайну інтер’єрів		
	приміщень. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

«____» _____ 2024 р.

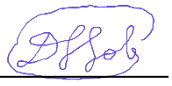
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДИЗАЙНУ ІНТЕР'ЄРІВ ПРИМІЩЕНЬ

Технічне завдання

ДП.045480-02-91

"ПОГОДЖЕНО"

Керівник проєкту:



_____ Дмитро НОВАК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Данііл БОНДАРЧУК

2024

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для дизайну інтер’єру приміщень.

Галузь застосування: Програмне забезпечення для малих архітектурних рішень.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. МЕТА РОЗРОБКИ

Створити дружній для користувача та інтуїтивний додаток для дизайну приміщень, який допомагає візуалізувати свої ідеї та створювати ефективні інтер'єрні рішення.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Застосунок "Tumba" має забезпечувати такі ключові функції:

1. Перегляд 3D моделей та ознайомлення з детальною інформацією про них.
2. Функція сортування об’єктів за категоріями.
3. Функція пошуку об’єкта за категорією/назвою.
4. Можливість розміщати 3d об’єкти в просторі за допомогою AR технології.
5. Можливість переміщувати та масштабувати об’єкти в просторі за допомогою AR технології.
6. Можливість змінювати текстури об’єктів.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту має бути розроблена наступна документація:

1. Пояснювальна записка, яка міститиме опис предметної області, огляд існуючих рішень, обґрунтування вибраних технологій, опис архітектури додатку, особливостей реалізації, інструкції з розгортання та налаштування системи.
2. Програма та методика тестування додатку, що включатиме тестові сценарії, стратегії тестування, вимоги до тестового середовища та звіти про проведене тестування.
3. Керівництво користувача, в якому буде докладно описано функціональні можливості, інструкції з встановлення та використання "Tumba", а також посібник з вирішення типових проблем.

Графічна частина, що міститиме:

- Діаграму послідовності процесу планування завдання;
- Діаграму компонентів архітектури додатку;
- Макети користувацького інтерфейсу для ключових сторінок.

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури та аналіз предметної області.....08.03.2025
Розроблення та узгодження технічного завдання..... 18.03.2025
Проєктування архітектури вебдодатку.....02.04.2025
Розробка дизайну користувацького інтерфейсу 22.04.2025
Тестування функціональності "Tumba”.....24.05.2025
Підготовка матеріалів пояснювальної записки..... 25.05.2025
Підготовка графічної частини та діаграм проєкту..... 28.05.2025
Оформлення проєктної документації згідно з вимогами..... 29.05.2025

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДИЗАЙНУ ІНТЕР'ЄРІВ ПРИМІЩЕНЬ

Пояснювальна записка

ДП.045480-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

 Дмитро НОВАК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Даніїл БОНДАРЧУК

2025

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API (Application Programming Interface) – набір визначень, протоколів та засобів для взаємодії з певним програмним забезпеченням.

UI (User Interface) – користувацький інтерфейс, засіб взаємодії між людиною та програмним забезпеченням.

UX (User Experience) – досвід взаємодії користувача з продуктом, що включає зручність, ефективність та задоволеність.

AR (Augmented Reality) – накладання 3D-об'єктів на реальний світ через камеру.

Unity – багатоплатформне середовище розробки інтерактивних 2D- та 3D-додатків, ігор і візуалізацій. Платформа підтримує створення застосунків для мобільних пристроїв, комп’ютерів, консолей та веббраузерів і забезпечує зручні інструменти для роботи з графікою, фізикою, анімацією та доповненою реальністю.

Android – відкрита операційна система для мобільних пристроїв, розроблена компанією Google.

iOS – операційна система для мобільних пристроїв компанії Apple.

Вебсайт – сукупність пов’язаних між собою вебсторінок, що розміщені на одному сервері та доступні в мережі Інтернет за спільною адресою.

Вебсторінка – документ, створений за допомогою мов розмітки, який відображається у веббраузері та є частиною вебсайту.

AI (Artificial Intelligence) – галузь інформатики, що займається створенням систем, здатних виконувати завдання, які зазвичай потребують людського інтелекту.

Ігровий рушій – програмне забезпечення, що забезпечує базову функціональність для створення відеоігор та інтерактивних застосунків, включаючи рендеринг графіки, фізику, анімацію, звук та керування об’єктами.

Рендеринг – процес створення зображення з цифрової 3D-сцени шляхом обчислення освітлення, текстур, тіней та інших візуальних ефектів для отримання фінального кадру.

Фреймворк – набір готових програмних компонентів і бібліотек, що забезпечують основу для розробки програмного забезпечення та визначають загальну структуру застосунку.

Бібліотека – набір готових функцій, класів або модулів, які використовуються розробниками для спрощення процесу програмування та повторного використання коду в різних проєктах.

Текстура – зображення або візерунок, який накладається на поверхню 3D моделі для надання їй кольору, деталей та реалістичного вигляду.

Матеріал – набір властивостей, що визначає зовнішній вигляд поверхні 3D-об’єкта, включаючи колір, блиск, прозорість, текстуру та реакцію на освітлення.

3D-модель – цифрове тривимірне представлення об’єкта, створене з використанням геометричних елементів (вершин, граней, полігонів), яке використовується у візуалізації, іграх, AR/VR та дизайні.

Основний колір – головний колір, що використовується в дизайні інтерфейсу або стилістичному оформленні та задає загальний візуальний стиль усього проєкту.

Вторинний колір – допоміжний колір, що доповнює основний у візуальному оформленні, використовується для виділення другорядних елементів інтерфейсу та створення гармонійної кольорової палітри.

Комплементарний колір – колір, що розташований напроти іншого на колірному колі та створює з ним максимальний контраст, підсилюючи його візуальне сприйняття.

Бренд – сукупність візуальних, емоційних та асоціативних характеристик, що формують унікальний образ компанії, продукту або послуги в уявленні споживачів.

Брендовий елемент – візуальний або текстовий компонент, що асоціюється з конкретним брендом і допомагає його впізнавати, наприклад логотип, кольорова палітра, шрифт або слоган.

Колізія – ситуація у 3D-середовищі, коли об’єкти стикаються або взаємодіють один з одним у фізичному просторі, зазвичай з використанням спеціальних алгоритмів виявлення зіткнень.

ВСТУП

Ефективна візуалізація продуктів та передбачуваність їх вигляду стали невіддільним складником сучасного ринку дизайну. Покупці бажають бачити як дизайнерське рішення буде виглядати у реальному середовищі, наскільки воно вписуватиметься по вигляду перед остаточним рішенням про реалізацію проєкту.

Протягом останніх років AR технології активно розвиваються та знаходять своє застосування в багатьох сферах, особливо в дизайні інтер'єрів. Однак більшість теперішніх рішень або мають дуже обмежений функціонал або потребують близько професійних навичок дизайнера створюючи занадто високий поріг входу для звичайних користувачів.

Проєкт має на меті створити дружній для користувача та інтуїтивний додаток для дизайну приміщень який допоможе користувачеві візуалізувати свої ідеї та створити власні інтер'єрні рішення. Додаток дозволяє об'єднувати сучасні AR технології та Unity для забезпечення розміщення 3d моделей фурнітури та інших елементів декору та дає широкі можливості персоналізації – від зміни розміру та положення до зміни кольору чи текстури моделі.

Розроблений додаток дозволяє користувачам зручно розміщувати меблі та елементи дизайну на виявлених поверхнях, отримуючи миттєвий та персоналізований результат.

Важливим акцентом при розробці є широкі можливості редагування моделей, зміну їх габаритів, положення та текстур, що має на меті задовольнити якомога більше різноманітних естетичних рішень користувачів.

Інтуїтивний інтерфейс забезпечує зручність роботи на мобільних пристроях, а проста та зрозуміла система маніпуляції моделями дозволяє легко взаємодіяти з 3D-моделями в AR.

Підсумовуючи, цей дипломний проєкт спрямований на створення інструменту для ефективної візуалізації та персоналізації дизайну приміщень, що надасть користувачеві можливість самостійно створити свої унікальні рішення.

1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність задачі персоналізованого дизайну приміщень

Сучасний ринок дизайну приміщень все більше орієнтується на індивідуальні потреби покупців, котрі прагнуть створити унікальний та комфортний життєвий простір. Проте вибір елементів декору та меблів часто супроводжується труднощами з неможливістю оцінити до моменту покупки чи пасуватиме той, чи інший вибір в обраному інтер'єрі, що ускладнює роботу користувачам без дизайнерського досвіду.

З розвитком смартфонів та широким їх розповсюдженням з'являються все нові й нові можливості та способи вибору облаштування помешкань, зокрема фурнітури та інших елементів дизайну розміщених у них. Останнім часом, особливу роль у цих процесах відіграють AR технології, що дозволяють віртуально та у реальному часі помістити бажані об'єкти у простір навколо користувача [1] ще до здійснення покупки чи пошуку подібного виробу у меблевих магазинах.

Android, ioS чи інша мобільна платформа та інтуїтивний інтерфейс робить такі рішення масовими, а процес взаємодії користувача простим та доступним [1]. Робота додатків у реальному часі та широкі можливості редагування об'єктів дозволяють користувачам швидко налаштовувати зовнішній вигляд власних інтер'єрів та спрощують процес пошуку та покупки необхідних меблів та об'єктів інтер'єру.

У цьому дипломному проєкті розглядається розробка мобільного застосунку для дизайну інтер'єрів приміщень за допомогою AR технології. Додаток "Tumba" покликаний створити процес такого дизайну більш інтуїтивним, наочним, персоналізованим та зручним, що підтверджує актуальність цієї розробки.

1.2. Огляд теперішніх додатків для дизайну приміщень

Wayfair – View in Room 3D це функція всередині мобільного застосунку магазину меблів Wayfair, що дозволяє покупцям віртуально розміщувати меблі у власному просторі за допомогою AR. Меблі розміщуються у масштабі 1:1, та деякі моделі мають опцію зміни текстури, що робить можливість вибору наочнішим.

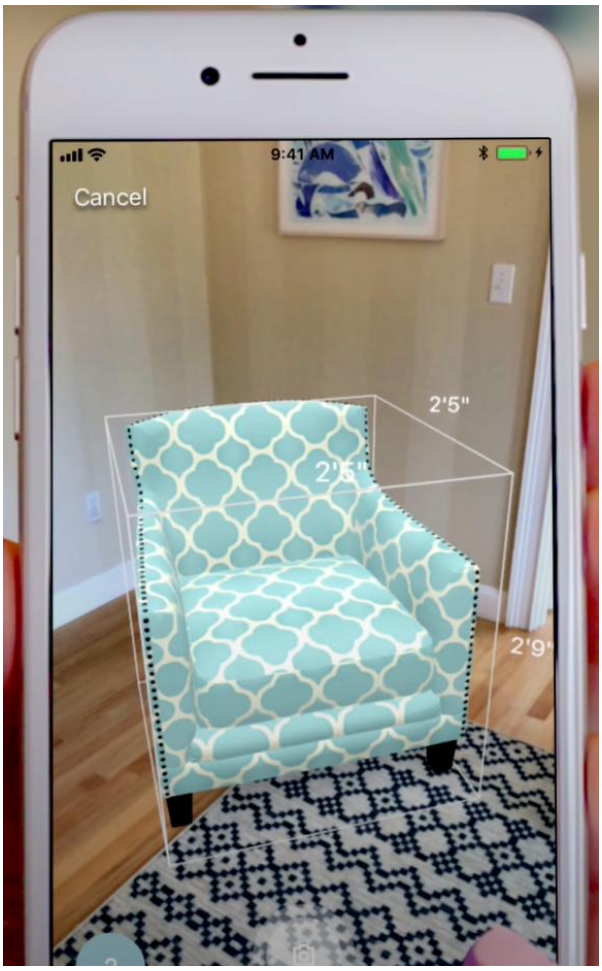


Рис. 1.1. Wayfair

Переваги:

1. Інтеграція з онлайн-магазином – користувач може примірити товари з каталогу та одразу примірити її.
2. Підтримка кросплатформності – існує вебсайт магазину, мобільний додаток доступний на Android, iOS та інших популярних платформах.

3. Висока якість 3D моделей – моделі деталізовані, виглядають реалістично в AR. Також для деяких моделей доступна базова можливість редагування окремих елементів.

Недоліки:

1. AR це доповнення – Wayfair не окремий додаток для дизайну, а лише допоміжна функція для інтернет-магазину.
2. Неможливість зміни розміру моделей – відсутній функціонал масштабування моделей за бажанням користувача, що може завадити зручно використовувати додаток в обжитих приміщеннях.
3. Не орієнтованість на український ринок – додаток орієнтований на покупців у США. Товари всередині додатку дорогі у порівнянні з українськими магазинами, також за доставлення стягуватиметься чималий чек.

Houzz – мобільний додаток, що містить всередині себе інструмент доповненої реальності, котрий дозволяє віртуально розміщувати меблі та предмети декору у власному помешканні.

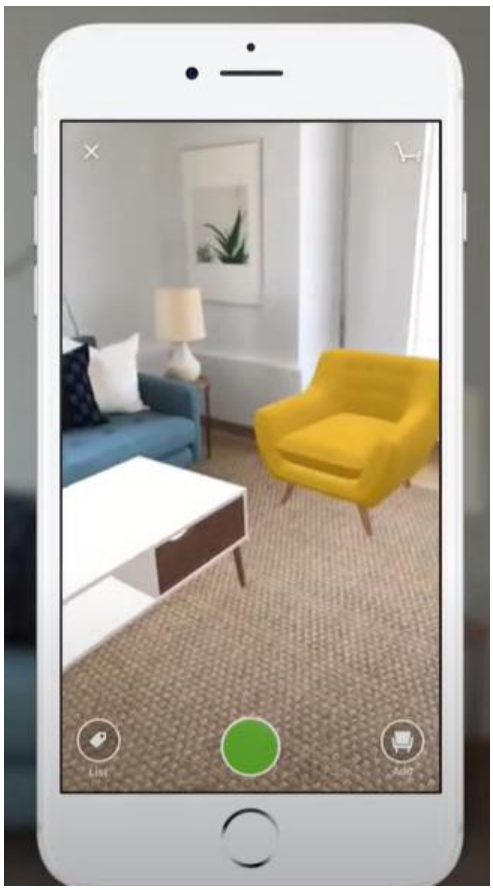


Рис. 1.2. Houzz

Переваги:

1. Широкий вибір реальних брендів та готових інтер’єрних рішень – користувач може обрати не лише окремий предмет, а й цілі дизайнерські композиції.
2. Комплексна платформа – у додатку є інструменти планування кімнат, перегляду ідей, фото, документів, статей.
3. Вбудована соціальна платформа – користувачі можуть зберігати свої роботи та ділитись ними з іншими людьми.
4. Локалізація – Houzz працює глобально та підтримує багато мов.

Недоліки:

1. AR це доповнення – Houzz не окремий додаток для дизайну, а лише допоміжна функція для інтернет-магазину.

2. Складний інтерфейс для AR користувачів – через велику кількість функцій інтерфейс не інтуїтивний для тих, хто шукає тільки AR-візуалізації.
3. Відсутність редагування – немає змін кольору, розміру, матеріалу в режимі реального часу. В додатку присутня лише відображення тільки готових товарів.

Amikasa – це застосунок, котрий поєднує функції планування приміщення з опцією розміщення елементів інтер'єру у доповненій реальності. У цьому додатку користувач може створювати 3D плани кімнати, додавати меблі з зазначеного каталогу та редагувати їх.

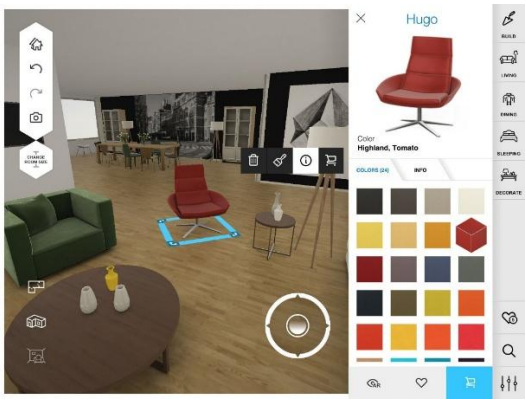


Рис. 1.3. Amikasa

Переваги:

1. Планування приміщення – користувач може створити модель власної кімнати перед розміщенням меблів.
2. Офлайн режим – частина функцій, включно з AR, можна використовувати без підключення до Інтернету.
3. Реалістичне освітлення – модель кімнати враховує джерела світла, чим ми додаємо глибини відображенню.

Недоліки:

1. Взаємодія з моделями – після розміщення в AR користувач не може змінити параметри елементів інтер'єру.

2. Високий поріг входу – додаток орієнтований на більш професійну аудиторію, UI та можливості глибші та важчі за аналоги.

DecorMatters – це мобільний застосунок, орієнтований на поєднанні AR дизайнів приміщень та соціальної мережі.

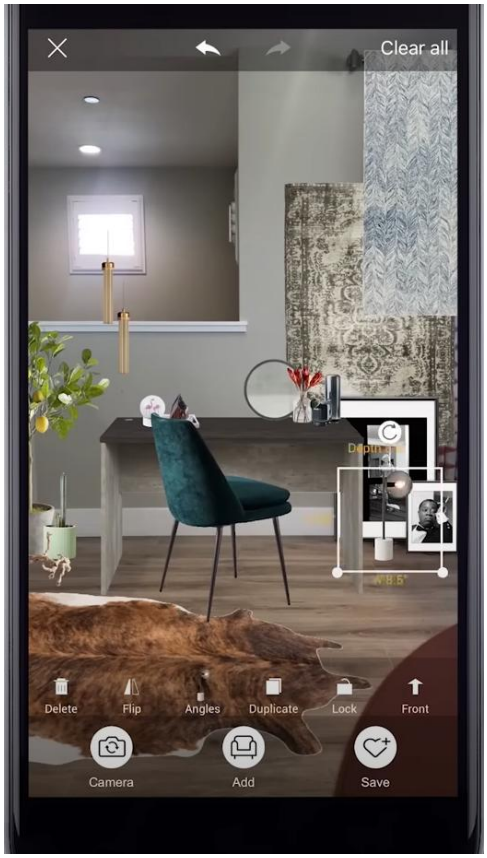


Рис. 1.4. DecorMatters

Переваги:

1. Орієнтація на соціальний складник – широкі можливості ділитись дизайнами, брати участь в тематичних роботах та взаємодіяти з іншими користувачами.
2. AI рекомендації – додаток пропонує варіанти декору на основі аналізу вподобань користувача.

Недоліки:

1. AR це доповнення – Houzz не окремий додаток для дизайну, все більше зосереджено на естетиці, ніж на точності.
2. Відсутність редагування – користувач не може змінювати розмір, стиль чи положення меблів у просторі.

1.3. Аналіз огляду додатків для дизайну приміщень

Сучасний підхід до оформлення інтер’єру все частіше базується на використанні різноманітних інструментів, в тому числі AR функції. Зазвичай рішення у цій сфері орієнтовані та створені переважно великими компаніями, через що дизайнерський складник є далеко не основним пріоритетом. Вони виконують базову функцію візуалізації, однак нечасто надають можливість гнучкого редагування об’єктів, будучи обмеженими лише власним брендом чи брендами-партнерами.

Такі рішення є лише однією з багатьох функцій мобільних застосунків магазинів меблів, котрі першочергово мають на меті комерцію та зручність покупки, а не результат дизайну як такий. Моделі в таких додатках часто дуже якісні та добре деталізовані, але мають обмежені можливості для змін текстури чи приміщення в AR просторі.

З іншого боку існує велика кількість додатків які ставлять акцент саме на соціальному складнику [2]. Вони більш дружні для користувачів, інтуїтивніший UI, спільноту чи внутрішню AI систему порадника, мають більше можливостей для змін та редагування, але часто не дозволяють використовувати AR функцію на повну. Також деякі додатки можуть бути занадто глибокими та професійними для пересічного користувача, чим можуть відштовхнути від себе новачка, якому важливий простіший та швидший результат.

Спираючись на тези наведені вище, розробка застосунку "Tumba" має на меті створення додатка, котрий буде уникати технологічних обмежень як і сторонніх платформ, так і переобтяжених функцій котрі не близькі новачкам. При створенні проєкту буде враховано як і потреба в якості моделей та можливості їх змін у вигляді та просторі так і бажання у простому, зрозумілому для новачка UI.

1.4. Висновки до першого розділу

У цьому розділі було проаналізовано передумови та обґрунтовано актуальність створення мобільного застосунку для візуалізації дизайну приміщень в AR. Встановлено, що наявні рішення здебільшого обмежуються базовим розміщенням моделей у просторі без можливості гнучкого налаштування їх зовнішнього вигляду, або є просто придатком до більшого проєкту, котрий має в пріоритеті комерцію, а не дизайн інтер'єру. Це створює потребу в інструменті, який дозволяє поєднати точну візуалізацію об'єктів з індивідуальним підходом до дизайну.

Значну увагу було приділено аналізу потреб цільової аудиторії – користувачів, які не мають професійних навичок роботи з архітектурним ПЗ, але бажають самостійно створити або змінити дизайн власного житлового або комерційного простору. Для них важливою є простота інтерфейсу, можливість швидко орієнтуватись у функціоналі застосунку та отримати візуальний результат без тривалого навчання чи попереднього досвіду.

Також було вивчено сучасні підходи до UX/UI мобільних застосунків з AR-функціональністю. Встановлено, що оптимальним є мінімалістичний, але інформативний інтерфейс з контекстним управлінням об'єктами в сцені, а також підтримкою жестів для масштабування, обертання та переміщення елементів. Такі функції підвищують зручність використання та сприяють глибшому залученню користувача до процесу моделювання.

Таким чином, створення "Tumba" є не лише актуальним, але і стратегічно обґрунтованим кроком, який забезпечить кращу персоналізацію, гнучкість та новачкову інтуїтивність у сфері дизайну приміщень за допомогою AR технології.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБЛЕННЯ

2.1. Вибір засобів розроблення мобільного AR-застосунку

Клієнтською частиною для цього проєкту є мобільний застосунок для 3D дизайну приміщень за допомогою AR. Створення такого застосунку потребує врахувати не тільки всю специфіку розробки мобільного додатку так і звернути особливу увагу на способи візуалізації тривимірних об’єктів у реальному часі, відстеження простору та інші важливі складники для коректної та повної роботи AR технології.

Загалом, для мобільної розробки існують два основних підходи – нативний та кросплатформний [8]. Нативна розробка у своїй суті передбачає створення додатків для конкретної мобільної операційної системи. Такий підхід до розробки передбачає використання таких мов, як Kotlin чи Java, якщо цільовою операційною системою є Android, в той час як Swift чи Objective-C є основними мовами для операційної системи iOS. Головною з багатьох переваг такого підходу є повний доступ до всіх функцій та можливостей обраної платформи та дуже висока оптимізація продуктивності, що особливо важливо у складних графічних застосунках як "Tumba".

На відміну від нативної, кросплатформна розробка забезпечує роботу одразу на кількох ОС одночасно, використовуючи єдину кодову базу. У цій сфері існує багато різних фреймворків, зокрема Flutter, React Native, Xamarin та рушії ігрового типу – Unity, Unreal Engine та Godot. З одного боку ці рішення забезпечені великою кількістю бібліотек, котрі дозволяють розробникам ефективно працювати з більшою частиною функцій операційних систем, що робить їх по-справжньому незамінним та універсальним рішенням багатьох кейсів. З іншого боку, нижча продуктивність додатків, обмеження доступу до можливостей платформи

та в роботі з 3D графікою й AR технологіями можуть стати вагомим аргументом проти такого підходу розробки.

Зважаючи на характер та специфіку розробки додатка для дизайну приміщень за допомогою AR технології ключовим критерієм при виборі підходу стали підтримка багатьох мобільних платформ, простота розгортання, а також легкий і надійний доступ до AR бібліотек та роботи з 3D моделями в режимі реального часу. У проєкті не передбачено використання складних обчислень та глибокої інтеграції з операційною системою, тож було прийнято рішення проводити розробку за допомогою кросплатформного підходу. Такий вибір дозволяє одночасно підтримувати дві найпопулярніші мобільні операційні системи – Android та iOS, зберігаючи гнучкість, зручність, швидкість підтримки й обслуговування додатка та високий темп впровадження нових функцій, тим самим дозволяючи сфокусувати зусилля на якості візуального досвіду та взаємодії користувача з AR, не витрачаючи обмежені ресурси на дублювання розробки для інших систем.

2.1.1. Flutter

Flutter – кросплатформний фреймворк, створений компанією Google, що дозволяє створювати мобільні додатки з єдиним кодом для Android та iOS. Він написаний мовою програмування Dart та використовує власний модифікований рендеринг рушій для відображення інтерфейсу, котрий забезпечує високу швидкодію та стабільну і якісну роботу на різних пристроях. Flutter популярний для створення застосунків зі складним інтерфейсом, має велику бібліотеку вже готових елементів котрі автоматично підлаштовуються до обраного дизайну та функцію Hot-reload, яка значно пришвидшує процес розробки, миттєво візуалізуючи зміни в інтерфейсі користувача [9].

Попри всі переваги у роботі з UI, спроможності Flutter в проєктах що мають акцент на роботі з тривимірною графікою є дуже обмеженим.

Фреймворк не має повноцінної підтримки чи власно створеної AR компоненти, що змушує залучати сторонні нестабільні бібліотеки чи створювати інші складні інтеграції. Реалізація таких рішень спричинює нестабільність роботи додатка та складність у розмежуванні дизайнерського елементу програми та логічного.

2.1.2. React Native

React Native – фреймворк для кросплатформної розробки мобільних застосунків створений компанією Meta. Він дозволяє створювати застосунки за допомогою JavaScript та бібліотеки React на Android та iOS, спираючись на нативні компоненти платформи а не через вбудований браузер. Такий підхід забезпечує кращу продуктивність, особливо у порівнянні з іншими кросплатформеними рішеннями [9].

React Native є популярним вибором розробників завдяки розвиненій спільноті та багатій системі бібліотек і плагінів. Гнучкость та ефективність створення UI, впроваджені в цьому фреймворку, дозволяють досягнути високої швидкості розробки та повного використання ресурсів мобільних систем.

Однак цей фреймворк не є орієнтованим на роботу з 3D графікою, тим паче з AR. Існують такі сторонні рішення як ViroReact, що дають реалізовувати базовий AR функціонал у React Native додатках, але вони мають непостійну підтримку, рідко оновлюються та не зібезпечують достатнього контролю над 3D моделями. Ці недоліки є критичними для виконання функціональних вимог додатку для дизайну інтер’єрів приміщень за допомгою AR технології "Tumba".

2.1.3. Xamarin

Xamarin – кросплатформний фреймворк, що дозволяє створювати мобільні додатки для операційних систем Android та iOS, використовуючи мову програмування C# та платформу .Net. Цей фреймворк був оригінально створений як альтернатива нативній розробці з можливістю

використання 90% коду для обох найпопулярніших платформ повторно. Xamarin легко інтегрується з офіційними інструментами розробки створених компанією Microsoft, такими як Visual Studio та компілює код у нативні збірки, що забезпечує високу продуктивність додатків.

Основною перевагою Xamarin є прямий доступ до обміну даними та взаємодією з операційною системою напряму, що дозволяє реалізувати нативний підхід та функціональність попри оригінальну кросплатформність. Також важливою перевагою є використання бібліотеки елементів Xamarin.Forms, які автоматично адаптуються до платформи, на якій встановлено додаток.

Однак іноді для точного відтворення унікальних платформенних елементів потрібно вручну створювати певні окремі інтерфейси та елементи вручну, що може сповільнювати розробку та суперечити кросплатформному принципу. Також, попри свою ефективність в інших напрямках Xamarin не має способів зручної реалізації 3D рушію. Підтримка AR реалізується лише через окремі бібліотеки та не мають певного централізованого фреймворку, що змушує шукати обхідні рішення Це значно ускладнює інтеграцію AR та вимагає глибоке розуміння внутрішньої структури та специфіку роботи Xamarin на різних операційних систем.

2.1.4. Unity

Unity – універсальний рушій, котрий в тому числі може бути використаний для розробки мобільних застосунків. Він містить в собі потужний набір інструментів для роботи з графікою, фізикою, анімацією та візуальним інтерфейсом, та глибоку підтримку мультиплатформної розробки, оскільки оригінально націлений на розробку комп’ютерних ігор.

Цей рушій підтримує імпорт 3D моделей та дає гнучкі інструменти для обробки текстур, матеріалів, світла та фізики, що робить його надзвичано зручним для розробки додатків, у яких візуальна 3D складова

відіграє ключову роль. Другою, не менш важливою перевагою Unity є інтеграція з AR Foundation – фреймворком для створення доповненої реальності, котрий об'єднує в собі рішення як і для Android так і для iOS платформи. Це дозволяє реалізувати важкі в реалізації фізичні компоненти AR як розпізнавання площини, відслідковування камери та контроль освітлення без необхідності реалізації цих механізмів для кожної операційної системи окремо [7].

Водночас попри значенні переваги Unity має певні недоліки, наприклад великий розмір фінальних збірок. Навіть найпростіші додатки можуть важити десятки мегабайт, що може бути критичним для мобільного сегменту. Також через глибоко реалізовані AR механізми додатки можуть мати високі вимоги до оптимізації, що потребує ретельної роботи над продуктивністю та швидкодією всього додатку.

2.1.5. Unreal Engine

Unreal Engine – один з найпотужніших ігрових рушіїв, розроблений компанією Epic Games. Він став широко відомий завдяки своїй фотореалістичності та широким графічним можливостям, глибокій системі анімації та рендерингу. Підтримка та розробка передових технологій у сфері 3D та віртуальної реальності робить його фаворитом для розробки великобюджетних ігрових проєктів та кіновиробництві [10].

Unreal Engine підтримує створення мобільних застосунків з інтегрованим AR завдяки підтримці ARKit та ARCore для iOS та Android відповідно. Він забезпечує повний контроль над рендерингом, що є великою перевагою при створенні глибоких та деталізованих 3D сцен з високоякісним та пропрацьованим освітленням, тінями та анімацією. Іншим значним плюсом є візуальне програмування через систему Blueprints, що за допомогою інтуїтивних автоматизованих скриптів дозволяє створювати логіку для об'єктів та 3D моделей без глибоких знань у цій платформі чи у мові програмування C++.

З іншого боку кінематографічність та реалістичність Unreal Engine може стати як і великою перевагою, так і великим недоліком. Через величезний об’єм навіть простих збірок розробка мобільних додатків на цьому рушії часто вимагає орієнтуватись на вужчий сегмент гаджетів. Також реалістична картинка та пропрацьована фізика можуть сильно навантажувати навіть сучасні мобільні пристрої, що ще сильніше зменшує кількість потенційних користувачів. AR функції для цього рушія для різних систем реалізуються частково по-різному, що подовжує розробку та робить її важчою через дублювання важкої AR логіки для іншої операційної системи.

2.1.6. Godot

Godot – відкритий та безкоштовний рушій для створення 2D та 3D засосунків, котрий останніми роками набрав популярності серед незалежних розробників завдяки активній спільноті, гнучкості та модульності. Він реалізує мультиплатформну розробку для більшості видів цільових платформ, від персональни ком’ютерів до ігрових консолей.

Головною перевагою Godot є повна відкритість коду, що дозволяє розробникам модифікувати рушій відповідно до потреб та функціональних вимог проєкту. Проста мова сценаріїв GDScript, котра синтаксисом дуже нагадує популярну та широко використовувану мову програмування Python є неодмінним плюсом у розробці, адже вимагає значно меншого часу на засвоєння, аніж всі аналоги, котрі були описані у підпунктах вище.

Однак, Godot не є практичним вибором для розробки проєктів з AR, оскільки підтримка цієї технології на цьому рушії є мінімальною та реалізованою лише через сторонні модулі або користувацькі, експерементальні збірки рушія. Також цей рушій не підтримує високоякісний рендеринг 3D сцен, особливо для складних об’єктів чи текстур, що робить його неприваблювою платформою для AR додатків.

2.2. Висновки проведеного аналізу

Після детального вивчення сучасних популярних платформ для створення мобільних застосунків було прийнято рішення обрати Unity як рушій клієнтської частини, адже серед усіх розглянутих вище варіантів саме Unity найкраще відповідає технічним та функціональним вимогам додатка для інтерактивного дизайну приміщень за допомогою AR технології "Tumba".

Аналіз альтернативних фреймворків чітко показав, що жоден з них не підходить для якісної та ефективної роботи з 3D моделями, особливо у режимі реального часу. Хоч більшість з них відмінно підходять для створення чудових UI та можуть бути дуже стабільними в роботі, відсутність глибоких можливостей у роботі з 3D унеможливлює їх вибір для цього проєкту. Також жоден з них не містить в собі хоч якогось легко реалізованого способу роботи з AR, чим значно ускладнюють та розтягують потенційну розробку.

Ігрові рушії, такі як Unreal Engine та Godot є вже більш відповідними кандидатами, але в них ряд суттєвих мінусів, котрі переважають всі переваги. Хоч Unreal Engine і продукує кінематографічне зображення та є високотехнологічним, така потужність є абсолютно надмірною для мобільних пристроїв. Великий об’єм навіть базових збірок та потреба у новіших гаджетах робить цей рушій непріоритетним для мобільної розробки застосунків для дизайну. З іншого боку, Godot, попри свою простоту, невибагливість та можливість підлаштуватись під будь-який проєкт, не містить в собі інструментів 3D візуалізації та впровадженої AR функції, так само як і фреймворки, що робить його паганим вибором.

Натомість Unity забезпечує не лише стабільну підтримку мобільних платформ (Android та iOS), але й має потужні засоби для роботи з тривимірними сценами, імпорту 3D-моделей, створення інтерактивного інтерфейсу та реалізації взаємодії у доповненому просторі. Його глибока інтеграція з AR Foundation, яка уніфікує роботу з ARKit і ARCore, дозволяє

реалізувати ключову функціональність застосунку – розміщення, масштабування та редагування меблів та елементів декору в реальному середовищі користувача.

Окрім технічної відповідності, важливою перевагою Unity є його широка спільнота розробників, постійна підтримка та велика кількість готових рішень, плагінів і навчальних матеріалів. Це значно скорочує час розробки, знижує ризики при виникненні технічних труднощів і забезпечує швидкий старт навіть для менш досвідчених учасників команди.

Завдяки гнучкості Unity, у рамках цього проєкту можливо реалізувати не лише базовий функціонал додатку, а й передбачити подальше масштабування – наприклад, синхронізацію зі сторонніми сервісами, можливість розширення під інші платформи, такі як Windows чи WebGL. Такий підхід робить Unity найкращим вибором для створення стабільного, зручного та функціонального інструменту для AR дизайну інтер’єру приміщень.

3. АРХІТЕКТУРА ТА ДИЗАЙН ДОДАТКУ

3.1. Загальний опис структури застосунку

Перед описом загальної структури застосунку для дизайну приміщень за допомогою AR технології "Tumba" слід враховувати ключову мету, котра лежить в основі розробки даного проєкту – створення дружнього для користувача та інтуїтивного додатка для дизайну приміщень, який допоможе візуалізувати користувацькі ідеї та створити ефективні інтер'єрні рішення.

ПЗ в основному орієнтоване на пересічних користувачів, котрі прагнуть за допомогою технології доповненої реальності та власного смартфона створити унікальний інтер’єр свого приміщення [12]. Також проєкт може стати в нагоді професійним дизайнерам як інструмент для скетчів та швидкої візуалізації думок, що може значно полегшити процес розробки дизайну чи комунікацію з замовником.

Спираючись на ці фактори, визначимо основні вимоги до розроблюваного додатка. Вимога – певна задача чи властивість, що має містити система. Вимога може бути функціональною, тобто описувати певну функцію, котру система повинна виконувати або може бути нефункціональною, описуючи властивості системи що можна виміряти, наприклад параметр швидкодії, надійніості, безпеки, відновлюваності, масштабування тощо.

Отже, наведемо список функціональних вимог:

1. Перегляд 3D моделей та ознайомлення з детальною інформацією про них.
2. Функція сортування об’єктів за категоріями.
3. Функція пошуку об’єкта за категорією/назвою.

4. Можливість розміщати 3d об’єкти в просторі за допомогою AR технології.
5. Можливість переміщувати та масштабувати об’єкти в просторі за допомогою AR технології.
6. Можливість змінювати текстури об’єктів.
7. Можливість власноруч створювати текстури для 3D моделей.

Також наведемо список нефункціональних вимог, котрі допоможуть визначити якість розроблюваного ПЗ:

1. Коректна версія роботи на різних версіях роботи Android (починаючи з Android 7.0 “Lollipop” та вище).
2. Час завантаження додатка та навігації між вікнами не має перевищувати поріг 5 секунд.
3. Встановлення обмеження на 50 3D об’єктів в сцені, щоб уникнути лагів та перевантаження, особливо на менш сучасних пристроях.
4. Оптимізована робота з пам’яттю завдяки стисненню об’єктів та текстур.

Спираючись на описані вимоги, у застосунку буде створено декілька ключових меню, які забезпечать користувачам зручний доступ до основних функцій.

Користувач базово знаходиться у головному меню, з котрого і здійснюється основна навігація додатком залежно від поставленої задачі.

Якщо користувач хоче детально ознайомитись з усіма 3D моделями що наявні у його бібліотеці, то він має скористуватись меню “Collection”. У ньому він обирає потрібну категорію та переходить до перегляду відповідної групи об’єктів. Він може детально розглянути всі моделі категорії, подивитись їх назву, рейтинг за параметрами “Comfort”, “Ergonomics”, “Universality” та прочитати загальний опис об’єкту. Після завершення сесії користувач може або переглянути іншу категорію, або повернутись у головне меню.

Далі користувач за бажанням може відредагувати будь-яку модель в меню “Redactor”. У ньому користувач спершу знаходить потрібну модель чи моделі за допомогою текстового пошуку або фільтрації усіх доступних об’єктів за категоріями чи рейтингом. Після цього в окремому меню можна змінювати текстури моделі та її колір на запропоновані додатком або створити власний колір для текстури та додати цю текстуру на обраний об’єкт. Всі надані зміни збережуться в пам’яті пристрою та будуть відображатись при роботі з цим та іншими меню додатка в майбутньому. Після завершення сесії користувач може або редагувати інший об’єкт, або повернутись у головне меню.

Найголовнішим меню є “Create design” котре спроєктовано для розміщення 3D моделей меблів та інших елементів дизайну на площині за допомогою технології AR. Після надання користувачем доступу до камери смартфона, додаток сам знайде та позначить пласкі робочі площини, на котрі можливо розмістити 3D об’єкти. Після цього користувач обирає бажані моделі за допомогою підменю пошуку та фільтрації та розміщує їх на площині. Після розміщення об’єкта користувач може за допомогою відповідних рухів пальцями переміщувати, обертати чи масштабувати об’єкта для створення кращого дизайнерського рішення. Після завершення сесії користувач може повернутись у головне меню.

Основний сценарій роботи додатка описує створення нового дизайну після дослідження колекції та редагування потрібних моделей (рис. 3.1).

Після запуску потрапляє до головного меню, яке є точкою входу до всіх інших меню додатка. У ньому у користувачу надається вибір наступного меню для роботи.

Для детального ознайомлення з моделями має бути натиснута кнопка “Collection”, після чого користувач має обрати категорію об’єктів для перегляду в наступному підменю. Після того як користувач ознайомився з детальною інформацією про кожен з об’єктів обраної категорії він вертається в головне меню.

Для редагування моделей користувач має натиснути кнопку “Redactor” в головному меню. Відфільтрувавши або знайшовши модель він переходить до підменю редагування, у якому він змінює чи створює бажану текстуру. Після того як зміни автоматично збережуться він повертається в головне меню для початку роботи над дизайнерським проєктом.

Для того, щоб створити дизайнерський проєкт користувач натискає кнопку “Create Design”, після чого програма сама автоматично знаходить всі необхідні робочі поверхні. Користувач може розміщати обрані за допомогою системи фільтрації чи пошуку об’єкти на площину та взаємодіяти з ними.

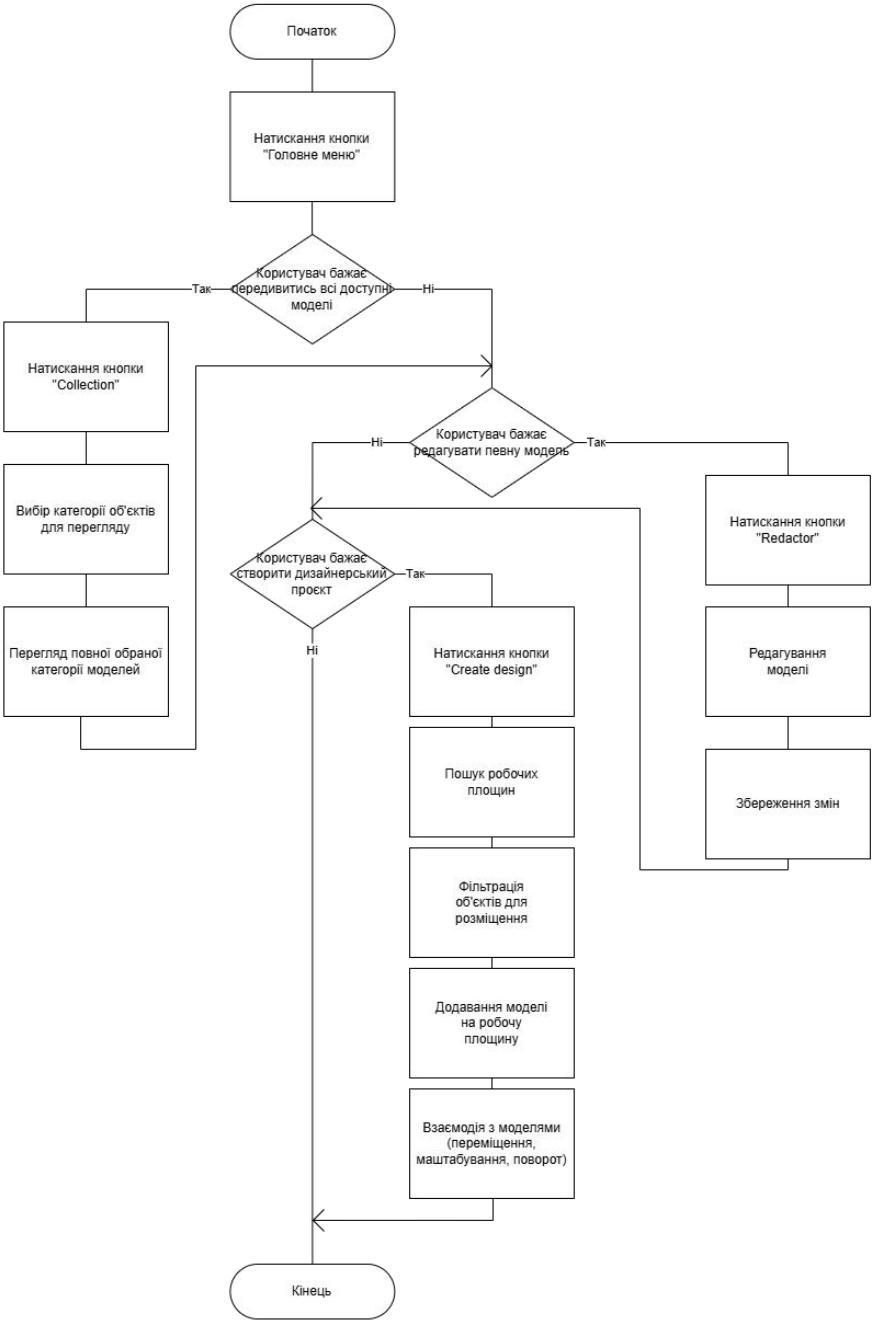


Рис. 3.1 Діаграма діяльності основного сценарцію використання додатку

3.2. Архітектура додатка “Tumba”

Структура застосунку для інтерактивного дизайну приміщень за допомогою AR технології “Tumba” є компонентною, забезпечуючи розподіл логіки програми між окремими блоками та функціями. Такий підхід дозволяє реалізувати гнучку логічну структуру, у котрій кожен модуль відповідає за певний набір дій та функцій системи, що значно пришвидшує та полегшує розробку додатка.

Розглянемо детальніше структуру модулів та їх призначення (рис. 3.2). Через особливості роботи рушія Unity кожен модуль є суто логічним та складається з багатьох менших підпрограм та підпрограм бібліотек.

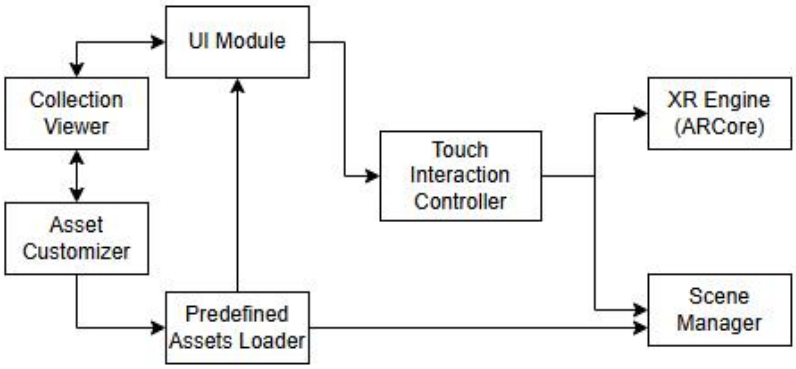


Рис. 3.2 Архітектура застосунку “Tumba”

UI Module є відповідальним за все, що пов’язано з користувацьким інтерфейсом, а його елементи реалізовані в кожному меню додатка. Основною структурною одиницею цього модуля є Canvas, котрий виступає як основний контейнер для всіх елементів інтерфейсу користувача. У кожному меню використовувався свій Canvas-об’єкт, котрий наповнювався всіма необхідними елементами та логікою залежності від потреб меню. Також однією з головних функцій цього модуля є збір зворотного зв’язку від користувача, котрий реалізований через стандартні UI компоненти рушія Unity: Button, Slider, Dropdown тощо. Наприклад, у меню Redactor елементи Slider та їх обробник, що зберігається в елементі Canvas використовується для зміни параметрів користувацького кольору для текстури редагованої моделі. Всі ці та інші компоненти модуля UI Module інтегровані через подійну систему Unity Events, що дозволяє оброблювати їх у єдиному місці без зміни та переписування логіки, що значно спрощує та пришвидшує розробку користувацького інтерфейсу.

Collection Viewer забезпечує логіку роботи меню Collection. Основна функція цього модуля у співпраці з UI Module надати зручний інтерфейс та

логіку для перегляду, ознайомленням з детальною інформацією про елемент декору та перемкнення між цими елементами. Центральний елемент Canvas меню Collection містить в собі містить контейнер, у котрому зберігаються посилання на всі моделі, котрі представлені для даної категорії. Для реалізації циклічної навігації кожен елемент має два посилання – на попередній та наступний об’єкт. Це дозволяє користувачу переходити до будь-якої моделі у колекції без потреби прокрутки або повернення до початку списку. Якщо користувач досягає останнього елемента, наступний перехід приводить його до першого, забезпечуючи безперервність циклу. Навігація відбувається за допомогою кнопок-стрілок, розташованих праворуч та ліворуч від моделі. При натисканні на відповідну стрілку підпрограма CollectionManager робить всі моделі, окрім наступної в списку невидимими та переписує текст та її рейтинги залежно від потрібної моделі.

Модуль Asset Customizer дає можливість користувачу змінювати візуальні параметри 3D моделей, зокрема текстуру їх матеріалу. Основна ідея модуля полягає в застосуванні глобальних змін, отже будь-які зміни, внесені до певної моделі, автоматично поширюються на всі її екземпляри в інших меню. Такий підхід дозволяє підтримувати узгоджений стиль оформлення моделей меблів та елементів дизайну, навіть після переходу в інші меню чи перезапуску додатка. Основна логіка реалізована за допомогою класів MaterialChanger та GlobalMaterialChanger. Перша, отримуючи зворотній зв’язок від вибору користувача знаходить обрану текстуру та модель і вносить шлях до них у пам’ять програми. Другий, спираючись на дані з такого словника що поєднує модель та актуальну текстуру вносить відповідні зміни та їх зовнішнього вигляду. Екземпляр класу GlobalMaterialChanger присутній в кожному меню додатка, в якому є взаємодія з 3D моделями, тому будь-яка зміна виконується глобально.

Predefined Assets Loader відповідальний за управління завантаженням та відображення заздалегідь підготовлених 3D моделей та

файлів програми загалом. Основна структура цього модуля базується на використанні Resources Folder – спеціального каталогу Unity, де зберігаються всі моделі, матеріали, текстури, зображення тощо. Всі основні функції цього модуля реалізовані автоматично рушієм Unity, що значно пришвидшує процес розробки додатка.

Touch Interaction Controller працює у синергії з UI Module, забезпечуючи взаємодію користувача з об'єктами у сцені за допомогою сенсорних жестів. Основною функцією цього модуля є обробка дотиків, розпізнавання жестів та перетворення цих подій на конкретні дії логіки програми. Більшість функцій цього модуля прописані автоматично у бібліотеках рушія Unity, а ті, котрі були створені під час розробки програми будуть детально розглянуті при розборі SceneManager.

Модуль XR Engine є основним логічним модулем додатка "Tumba", що відповідає за відображення 3D моделей у реальний простір, їх трекінг за допомогою камери смартфона та логіку взаємодії користувача зі спроєктованими об'єктами. Цей модуль є модифікацією бібліотеки AR Core, який дозволяє ідентифікувати горизонтальні поверхні, розміщувати об'єкти на цих поверхнях, відстежувати їх положення та коригувати їх розташування відповідно до руху камери. Це забезпечує не лише статичне відображення об'єктів, але й повноцінну інтерактивну взаємодію з ними у доповненій реальності.

Scene Manager є головним контролером для управління 3D об'єктами в AR під час роботи в меню Create design. Основною його функцією є контроль за додаванням, видаленням, переміщенням, обертанням та масштабуванням об'єктів, забезпечуючи цілісну інтерактивну взаємодію між користувачем та AR-середовищем. При завантаженні сцени Scene Manager ініціалізує сканування простору за допомогою ARRaycastManager, який обробляє дотики до екрана та визначає точку зіткнення з площиною. Ця точка стає базовою для розміщення об'єкта, а Scene Manager одразу створює ARAnchor для його стабілізації у просторі. Такий підхід дозволяє

системі зберігати коректне положення об’єкта навіть при русі камери або зміні кута огляду. Дані про площини та їх координати зберігаються у внутрішньому словнику Scene Manager, що забезпечує швидкий доступ до них при оновленні позиції об’єктів. Після розміщення об’єкта, через взаємодію модулів Touch Iteration Controller, XR Engine та Scene Manager реагуючи на характерні рухи користувача по екрану модель змінює своє положення в AR сцені, свій масштаб, положення, розмір тощо.

3.3. Дизайн користувацького інтерфейсу

Дизайн користувацького інтерфейсу у застосунку для дизайну інтер’єрів за допомогою AR технології “Tumba”, оскільки він має на меті поєднати зовнішньо привабливий вигляд програми, і спосіб, у котрому користувачі можуть легко та й зручно взаємодіяти з 3D об’єктами в AR у режимі реального часу при роботі меню “Create design”. Основна мета UI-дизайну полягає у створенні максимально інтуїтивного середовища для редагування об’єктів, їх редагування та розміщення у просторі, забезпечуючи при цьому цілісність користувацького досвіду. У цьому розділі буде детально прописано основні принципи дизайну користувацького інтерфейсу, елементи, кольори та шрифти, котрі були використані для створення максимально позитивного користувацького досвіду при роботі з додатком “Tumba” [12].

3.3.1. Основні принципи UI

Інтерфейс у застосунку “Tumba” розроблений з урахуванням як і специфіки AR програм, так і класичних мобільних додатків. Інтерфейс має бути простим та зрозумілим для користувача без необхідності вивчення всіх відповідних посібників та інструкцій [13]. Також, важливо побудувати UI навколо логічних дій користувача, що дозволяє перейти за основним сценарієм роботи програми (рис. 3.1). без зайвих кроків та маніпуляцій. Неодмінним складником успішного дизайну мобільного застосунку є узгодженість інтерфейсу: усі елементи мають бути в одній колірній палітрі,

мати єдиний спільний стиль та використовувати контрастні кольори для акцентуації. Такий підхід дозволяє зберегти єдність візуального сприйняття навіть при переміщенні користувача між різними розділами застосунку. Важливим елементом при розробці дизайну користувацького інтерфейсу було врахування всі особливостей AR при роботі меню “Create Design”. Основна мета дизайну цього меню – створити інтерфейс, який не перекриває віртуальні об’єкти та забезпечує природну взаємодію з ними у реальному просторі. Це досягається через впровадження мінімалістичного стилю, чіткої структури та прозорих елементів інтерфейсу, які дозволяють користувачу зосередитись на об’єктах у сцені без відволікання на зайві деталі.

3.3.2. Основні використані елементи інтерфейсу

Розглянемо основні елементи UI, котрі були використані при розробці додатка. Всі названі елементи є частиною стандартної бібліотеки Unity UI та оптимізовані під використання на різних платформах та операційних системах.

Image є найбільш широковживаним елементом в користувацькому інтерфейсі. Може використовуватись для зберігання зображень, фону та інших елементів дизайну, починаючи від заголовків до кнопок. Часто поєднує свої функції з компонентом Button.

Button це інтерактивний елемент інтерфейсу, що реагує на натискання. Використовується для виконання основних дій, таких як навігація додатком, впровадження або збереження змін, та інша активація значної частини логіки програми.

TextMeshPro використовується для зберігання високоякісних текстових полів. Основною його властивістю є відображення користувацьких шрифтів з можливістю тонкого їх налаштування.

Input Field це текстове поле для введення даних, що використовує TextMeshPro для чіткого відображення тексту. Використовується у полі пошуку для введення назви моделі.

Dropdown виконує функції випадаючого списку, який дозволяє обрати категорію об’єктів для фільтрації. Використовує TextMeshPro для відображення тексту.

Canvas є основним контейнером для всіх UI елементів. Використовується для розміщення статичних елементів та World Space для контекстних меню у просторі AR.

Slider це інтерактивний елемент, що дозволяє регулювати інтенсивність кольору створюваного користувацького матеріалу.

3.3.3. Візуальний дизайн

Візуальний дизайн додатка “Tumba”, на яскравій та контрастній кольоровій схемі, що забезпечує легку орієнтацію інтерфейсом. Основним кольором є яскрава маджента, а комплементарним – насичений фіолетовий, створюючи яскраві візуальні доповнення та акценти, що збільшує впізнаваність додатка та якість користувацького досвіду. Вторинний колір було обрано білим, він застосовується для текстових елементів та полів вводу, підвищуючи їх контрастність на фіолетовому фоні. Така палітра створює відчуття яскравості та ігрового стилю, водночас зберігаючи візуальну гармонію між інтерфейсними компонентами.

3.3.4. Шрифти

У застосунку використовується шрифт Irish Grover, який має характерний декоративний стиль з плавними, округлими лініями. Він надає інтерфейсу впізнаваний вигляд, створюючи легку ігрову атмосферу, що відповідає стилю додатка. Irish Grover застосовується для заголовків та основних акцентів, підкреслюючи важливі елементи інтерфейсу. Таким чином, використання Irish Grover не лише додає візуальної привабливості інтерфейсу, але й створює чіткий акцент на ключових елементах,

підкреслюючи характерний стиль додатка та забезпечуючи цілісність його дизайну.

3.4. Висновки до розділу

У даному розділі було проведено детальний аналіз структури застосунку Tumba, що побудований на основі AR технології для інтерактивного дизайну приміщень.Визначено, що інтерфейс програми має просту та зрозумілу структуру, побудовану навколо основних меню Collection, Redactor та Create Design. Кожне меню виконує конкретну задачу, але водночас інтегровано у загальну логіку програми для забезпечення цілісності користувацького досвіду та підтримки принципу модульної розробки.

Аналіз основних модулів додатка показав, що використання компонентної архітектури дозволило розподілити функціональність між окремими елементами, такими як UI Module, Asset Customizer, Predefined Assets Loader, Scene Manager, XR Engine та Touch Interaction Controller. Це забезпечило незалежність роботи кожного модуля та спростило процес підтримки та оновлення додатка. Взаємодія між модулями реалізована через подійну систему Unity Events, що дозволяє ефективно обробляти дії користувача без дублювання логіки.

Було також проаналізовано використання компонентів Unity UI, таких як Button, TextMeshPro, Slider, Canvas та інших. Вони забезпечують чітку та стабільну реалізацію інтерфейсу, зберігаючи цілісність стилю програми та підтримуючи адаптивність під різні екрани пристроїв та операційних систем. Водночас використання Irish Grover та гармонійної кольорової схеми в ігровому стилі сприяє створенню виразних та запам’ятовуваних текстових елементів, що підкреслюють ключові дії у процесі взаємодії з 3D-об’єктами та покращують взаємодію між користувачем та інтерфейсом.

Загалом, структура додатка Tumba побудована з урахуванням вимог до мобільних AR-застосунків та забезпечує інтуїтивну взаємодію як і з 3D об’єктами в AR, так і в інших меню додатка. Використання модульної архітектури спрощує масштабування проєкту, а дотримання принципів мінімалізму та послідовності у візуальному дизайні дозволяє уникнути перевантаженості інтерфейсу, що особливо важливо в AR середовищі.

4. АНАЛІЗ РОЗРОБЛЕНОГО ДОДАТКА

4.1. Опис UI

Головне меню застосунку “Tumba” виконує роль основного навігаційного центру, забезпечуючи доступ до ключових функцій інтерфейсу. У верхній частині екрана розташована назва проєкту, стилізована у впізнаваному шрифті, що підкреслює тематику інтер'єрного дизайну. Центральним візуальним елементом є зображення тумби, яке символізує тематику застосунку та слугує своєрідним брендовим елементом. У центральній частині екрана розміщено три основні кнопки, що контрастно виділяються на маджентовому фоні – “Create Design”, “Collection” та “Model Redactor”, котрі є переходами до основних розділів застосунку. У нижній частині екрана розташовано інформаційний блок із зазначенням імені розробника, навчальної групи та року створення проєкту, що сприяє візуальній завершеності екрана головного меню.

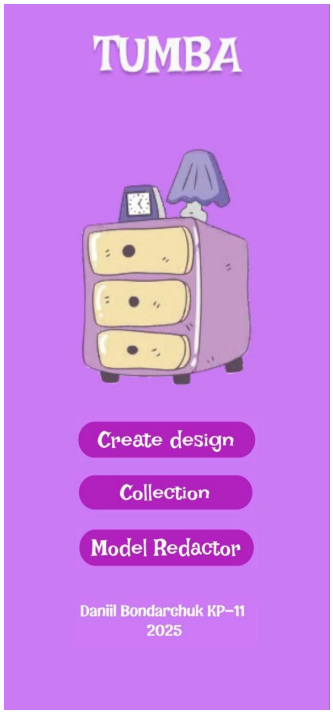


Рис. 4.1. Головне меню застосунку “Tumba”

Проміжна сторінка колекції об’єктів реалізована у вигляді вертикального списку категорій меблів, на даний представлених у застосунку. У верхній частині розміщено заголовок “Collection”, а під ним категорії “Chairs”, “Tables”, “Sofas” та “Other”. Вибір будь-якої категорії переводить користувача на відповідну сторінку з переліком об’єктів цієї групи. В нижній частині екрана розміщено кнопку «Back», яка повертає користувача до головного меню, забезпечуючи простий і зрозумілий спосіб навігації між розділами застосунку.

Рис. 4.2. Проміжна сторінка меню “Collection”

Сторінка перегляду об’єкта колекції призначена для детального ознайомлення з характеристиками вибраного елемента. У верхній частині екрана розташовано заголовок “Collection”, під заголовком подано короткий опис об’єкта, його рейтинг комфорту, ергономічності та універсальності. Центральне місце на сторінці займає зображення об’єкта у вигляді 3D-моделі, що дозволяє візуально оцінити його дизайн та конструкцію. У нижній частині сторінки розміщено назву об’єкта та кнопки для переходу до наступного або попереднього елемента колекції,

котра реалізована як кільцевий двозв’язний список. Для повернення на попередню сторінку передбачено кнопку “Back”.

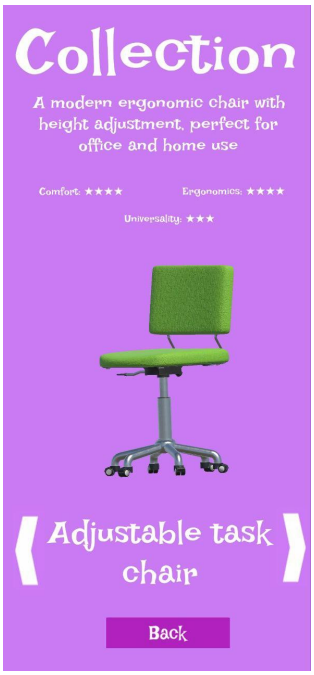


Рис. 4.3. Сторінка меню “Collection”

У верхній частині сторінки “Redactor” розташовано відповідний заголовок, під яким реалізовано панель керування, що включає елементи фільтрації та пошуку, забезпечуючи зручну навігацію серед об’єктів та прискорення процесу вибору необхідного елемента. Секція фільтрації представлена випадаючим списком “Filter by item/rating”, який дозволяє користувачу впорядковувати об’єкти за категоріями меблів або рейтингом. При виборі певної категорії у вікні відображаються лише ті об’єкти, які належать до цієї групи, що значно полегшує процес вибору об’єкта. Додатково, у фільтрі передбачена опція сортування за рейтингом, що дозволяє користувачу переглянути об’єкти з високими оцінками комфорту, ергономіки чи універсальності. Нижче розміщено текстове поле “Search by name”, яке виконує функцію миттєвого пошуку за назвою об’єкта. Після введення тексту система автоматично відображає відповідні моделі, які містять задану ключову фразу, чим додає гнучкості у пошуку. Основна

частина сторінки представлена сіткою з іконками об’єктів, кожна з яких оформлена у вигляді невеликого зображення у рамці. Така структура забезпечує візуальну компактність та дозволяє відобразити значну кількість елементів на одному екрані без перенасичення інтерфейсу. У нижній частині сторінки розташовано кнопку “Back”, яка дозволяє повернутися до головного меню.

Рис. 4.4. Проміжна сторінка меню “Redactor”

Сторінка редагування моделі забезпечує користувача можливістю налаштовувати зовнішній вигляд обраного об’єкта, змінюючи його кольори та текстури. У верхній частині екрана розташовано заголовок “Redactor”, нижче відображено назву об’єкта “Model Name”, що вказує на поточну модель, обрану для редагування. Поруч зазначено поточну назву текстури, під якою реалізовано блок вибору стилю, який складається з набору текстурних варіантів, представлених у вигляді кольорових квадратів. Користувач може обрати один із варіантів текстури, після чого зображення об’єкта миттєво оновлюється з урахуванням вибраного матеріалу. Такий інтерактивний підхід сприяє візуальній наочності

редагування та полегшує процес прийняття дизайнерських рішень. У центральній частині сторінки розміщено 3D-зображення об’єкта, що дозволяє користувачу оцінити застосовані зміни у реальному часі. Також це означає, що зміна матеріалу автоматично оновлюється у всіх місцях, де використовується дана модель, забезпечуючи узгодженість візуального стилю та уникнення розбіжностей у дизайні. Такий підхід оптимізує робочий процес, оскільки користувачеві не потрібно вручну змінювати матеріали для кожного екземпляра об'єкта. Під зображенням знаходяться три слайдери, що відповідають за коригування кольорових компонентів об’єкта – червоного, зеленого та синього. Змінюючи положення повзунків, користувач може динамічно налаштовувати колірну палітру моделі, отримуючи миттєвий результат. У нижній частині сторінки реалізовано дві основні кнопки. Кнопка “Create design” зберігає внесені зміни та підтверджує новий стиль об’єкта. Кнопка “Back” повертає користувача до попереднього меню без збереження змін, забезпечуючи зручну та зрозумілу навігацію між розділами.

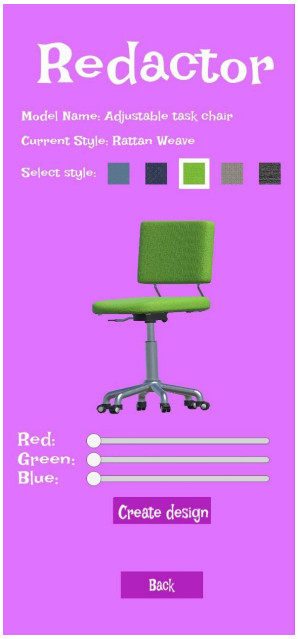


Рис. 4.5. Сторінка меню “Redactor”

Сторінка AR-режиму призначена для розміщення об’єктів із колекції у реальному просторі за допомогою камери пристрою. У верхній частині екрана розташовано кнопку «Back», яка забезпечує швидке повернення до попереднього меню, а також випадаючий список, що дозволяє користувачу обирати категорію об’єктів для відображення. Основна частина екрана відведена під область перегляду камери, через яку користувач може спостерігати простір у режимі реального часу. У нижній частині екрана реалізовано панель вибору об’єктів, яка містить горизонтальний скролінг зі зображеннями меблів з колекції. Користувач може перегортати доступні об’єкти за допомогою кнопок навігації обираючи потрібний елемент для розміщення у просторі. Обраний об’єкт автоматично накладається на зону відображення, дозволяючи користувачу оцінити його розмір, пропорції та розташування в інтер’єрі. Користувач також має змогу змінювати положення об’єкта, масштабувати його за допомогою жестів на екрані та повертати навколо вертикальної осі. Така функціональність забезпечує точну і наочну взаємодію з 3D-моделями, що дозволяє краще адаптувати їх до особливостей конкретного простору.

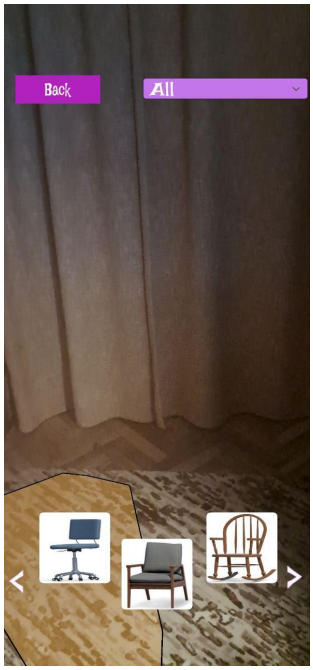


Рис. 4.6. Сторінка меню “Create Design”

4.2. Тестування застосунку

Під час розробки застосунку “Tumba” проведення тестування відіграло головну роль у забезпеченні якості програмного забезпечення. Основний фокус був на проведенні ручному наскрізному тестуванню [14], що дозволило оцінити роботу застосунку в умовах максимально наближених до реальних сценаріїв використання.

Наскрізне тестування охоплювало перевірку функціональності всіх основних модулів, включаючи розміщення об’єктів у просторі, зміну їхнього розміру та текстури, взаємодію з бібліотекою об’єктів режимом редагування тощо. Тестування проводилося з урахуванням типових сценаріїв користувача, що дозволило не лише перевірити коректність виконання окремих операцій, але й виявити потенційні проблеми у взаємодії між компонентами.

Особлива увага була приділена перевірці стабільності роботи застосунку при великій кількості об’єктів у сцені та тестуванню колізій між об’єктами. Такий підхід сприяв ідентифікації сценаріїв, які могли бути пропущені під час розробки, забезпечуючи більшу надійність кінцевого продукту [15].

Таблиця 4.1

Таблиця тестування

№	Мета тест-кейсу	Дія	Очікуваний результат
1	Виявлення колізій під час масштабування	Збільшення об’єкта у розмірі на більше ніж на 100%	Додаток не дозволяє збільшувати об’єкт більше ніж на 100% від початкового розміру. При спробі надмірного збільшення ця дія над об’єктом не відбувається

Продовження табл. 4.1

2	Виявлення колізій під час масштабування	Зменшення об’єкта у розмірі на більше ніж на 50%	Додаток не дозволяє зменшувати об’єкт менше ніж на 50% від початкового розміру. При спробі надмірного збільшення ця дія над об’єктом не може відбутися.
3	Виявлення колізій під час масштабування	Збільшення та зменшення об'єкта з інтервалом у 2 секунди	Система коректно реагує на всі зміни, відсутні затримки.
4	Тестування точності	Швидкі оберти об’єктів навколо своєї осі в AR	Оберти об’єкта відтворюється синхронно, кадри не втрачаються
5	Тестування точності	Масштабування об’єкта	Об’єкт зберігає свої пропорції за будь-яких умов
6	Тестування точності	Зміна кута огляду в AR	Всі об'єкти залишаються на своїх позиціях, без зсуву
7	Тестування точності	Швидкий рух об’єктів по AR площині	Рух об’єкта відтворюється синхронно, кадри чи координати не втрачаються
8	Тестування продуктивності	Додавання понад 20 об’єктів у сцену	Всі об’єкти залишаються на своїх місцях, відсутні затримки
9	Взаємодія з моделями у AR	Швидкі оберти об’єктів навколо своєї осі в AR	Оберти об’єкта відтворюється синхронно, кадри не втрачаються

10	Вплив зміни освітлення на відображення об'єктів	При оберті чи переміщенні об’єктів на об’єкті з’являються відповідні тіні.	Тіні змінюються динамічно, затримки відсутні.

Продовження табл. 4.1

11	Перевірка збереження стану текстур	Зміна текстури трьох об'єктів, вихід у головне меню та повернення до перегляду цих об’єктів	Всі зміни текстур залишаються після повернення до об’єктів
12	Перевірка збереження стану текстур	Зміна текстур понад десяти елементів	Всі зміни збережено, відсутні затримки
13	Перевірка узгодженості стану текстур	Заміна текстури об’єкта в режимі редагування	Текстура на відповідному об’єктові змінилась глобально
14	Перевірка узгодженості стану текстур	Спроба одночасного додавання власної текстури та вибору зі списку готових	Пріоритет віддається останньому вибору користувача

Під час тестування було виявлено низку проблем, що впливали на коректну роботу застосунку. Зокрема, виникали проблеми з відображенням текстур після перезапуску сцени, коли обрані матеріали не завантажувались коректно або поверталися до стандартних значень. Також було виявлено проблеми з взаємодією з об’єктами у режимі AR. Зокрема, при спробі масштабування та обертання об’єктів спостерігалося

неправильне розташування та масштабування моделей, що призводило до їхнього спотворення та невідповідності реальним пропорціям об’єкта. Також було зафіксовано зсув координат при переміщенні об’єктів, внаслідок чого вони відображалися не у заданій точці, а зі значним відхиленням.

Усі зазначені помилки було проаналізовано та усунуто, що дозволило забезпечити стабільну роботу застосунку при роботі всіх його меню.

4.3. Пропозиції для майбутнього покращення роботи додатка

Для подальшого розвитку застосунку пропонується низка напрямків, що сприятимуть підвищенню його функціональності та розширенню цільової аудиторії. Насамперед, доцільним є розширення бібліотеки об’єктів шляхом співпрацювання з меблевими компаніями для створення нових моделей меблів та декору. Це не лише збільшить різноманіття доступних моделей, але й дозволить створити простір для комерційних інтеграцій та спільних проєктів.

Крім того, корисним буде впровадження системи рекомендацій, яка допомагатиме користувачам у виборі меблів та їх розміщенні відповідно до загального стилю інтер’єру. Це сприятиме підвищенню інтерактивності застосунку та може слугувати інструментом для навчання нових користувачів. У цьому контексті варто розглянути можливість співпраці з освітніми платформами, що дозволить “виростити” нову аудиторію користувачів та поширити популярність застосунку.

Також перспективним напрямком розвитку є реалізація функції поширення створених дизайнів, що дозволить користувачам демонструвати свої роботи у соціальних мережах чи ділитися ними з іншими користувачами. Це сприятиме популяризації застосунку, створенню активної спільноти користувачів і забезпечить зворотний зв’язок з користувачами.

ВИСНОВКИ

Основною метою даного дипломного проєкту було створення програмного забезпечення для дизайну інтер'єру приміщень, яке дозволяє користувачам візуалізувати свої ідеї за допомогою AR технології. Застосунок надає можливість інтегрувати 3D моделі меблів та елементів декору у реальний простір, змінювати їх розмір, текстури та кольори, забезпечуючи високий рівень персоналізації.

Вибір технології AR був обґрунтований потребою у створенні інтуїтивного інструменту, що дозволить користувачам попередньо оцінити дизайн приміщення до його реалізації. Для розробки застосунку було обрано платформу Unity, яка забезпечила гнучку інтеграцію AR, імпорт 3D моделей та їх динамічне редагування у просторі. Крім того, застосунок використовує ARCore для точного розпізнавання площин та взаємодією з об'єктами в AR меню.

Сформовано та реалізовано основні функціональні вимоги до цього проєкту: створення дизайнів у AR, редагування текстур та кольорів моделей, бібліотека об'єктів з можливістю сортування та фільтрації, Кожне з основних меню застосунку було структуровано з урахуванням зручності користувача та інтерактивності інтерфейсу.

Застосунок «Tumba» складається з чотирьох основних меню: головного, колекції, редактора моделей та AR-режиму. Головне меню забезпечує доступ до ключових функцій програми, тоді як у меню колекції користувач може переглядати всі доступні об'єкти та ознайомитись з детальною інформацією про них. Редактор моделей дозволяє змінювати текстури та матеріали об'єктів, а AR-режим відповідає за їх розміщення у реальному просторі. Це дозволяє не лише створювати окремі інтер'єрні елементи, але й оцінювати їхню взаємодію у контексті загального дизайну

приміщення. Така структура забезпечує логічну послідовність роботи із застосунком та сприяє зручності користування.

Під час тестування було виявлено ряд проблем, а саме некоректне поводження об’єктів у AR та недосконала реалізація функцій зміни текстур. Усі зазначені проблеми було проаналізовано та усунуто, що дозволило забезпечити стабільність та коректність роботи застосунку.

Результатом роботи є мобільний застосунок “Tumba”, що дозволяє користувачам створювати дизайн інтер’єрів приміщень у AR. У подальшому пропонується розширити бібліотеку об’єктів, додати можливість інтеграції з соціальними мережами для поширення дизайнів та впровадити систему рекомендацій для оптимізації розташування об’єктів у просторі, що сприятиме підвищенню зручності застосунку та розширенню його функціональних можливостей.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. S. Siltanen, V. Oksman, and M. Ainasoja, "User-Centered Design of Augmented Reality Interior Design Service," International Journal of Arts & Sciences, vol. 6, no. 1, pp. 547-563, 2013. Available: https://universitypublications.net/ijas/0601/pdf/SPQ643.pdf
2. V. T. Phan and S. Y. Choo, "Interior Design in Augmented Reality Environment," Int. J. Comput. Appl., vol. 5, no. 5, pp. 16-22, Aug. 2010. Available: https://doi.org/10.5120/1021-1380
3. A. Neba et al., "Usability study of a user-friendly AR assembly assistance," Procedia CIRP, vol. 104, pp. 74-79, 2021. Available: https://doi.org/10.1016/j.procir.2021.11.013
4. X. Zhang, X. Wang, and W. Xu, "Research on User Demands and Functional Design of an AR-Based Interior Design and Display Platform for Recreational Vehicles," Appl. Sci., vol. 14, no. 10568, 2024. Available: https://doi.org/10.3390/app142210568
5. N. Statham, "Modular Architecture for 3D Game Environment Art with Photogrammetry," Ph.D. thesis, Univ. of Porto, Portugal, 2022. Available: https://repositorio-aberto.up.pt/handle/10216/567562
6. M. Hlayel et al., "Enhancing unity-based AR with optimal lossless compression for digital twin assets," PLoS ONE, vol. 19, no. 12, pp. e0314691, 2024. Available: https://doi.org/10.1371/journal.pone.0314691
7. M. Aurelius, "Mobile Application Development Framework with key criteria for choosing native or cross-platform application development," B.Sc. thesis, Karlstad University, Sweden, 2020. Available: https://www.diva-portal.org/smash/get/diva2:1443427/FULLTEXT01.pdf
8. N. Nguyen, "Developing a multiplayer AR game using AR Foundation and Unity," B.Eng. thesis, Metropolia Univ. Appl. Sci., Finland, 2020. Available: https://www.theseus.fi/bitstream/handle/10024/338650/Nguyen_Ngoc.pdf

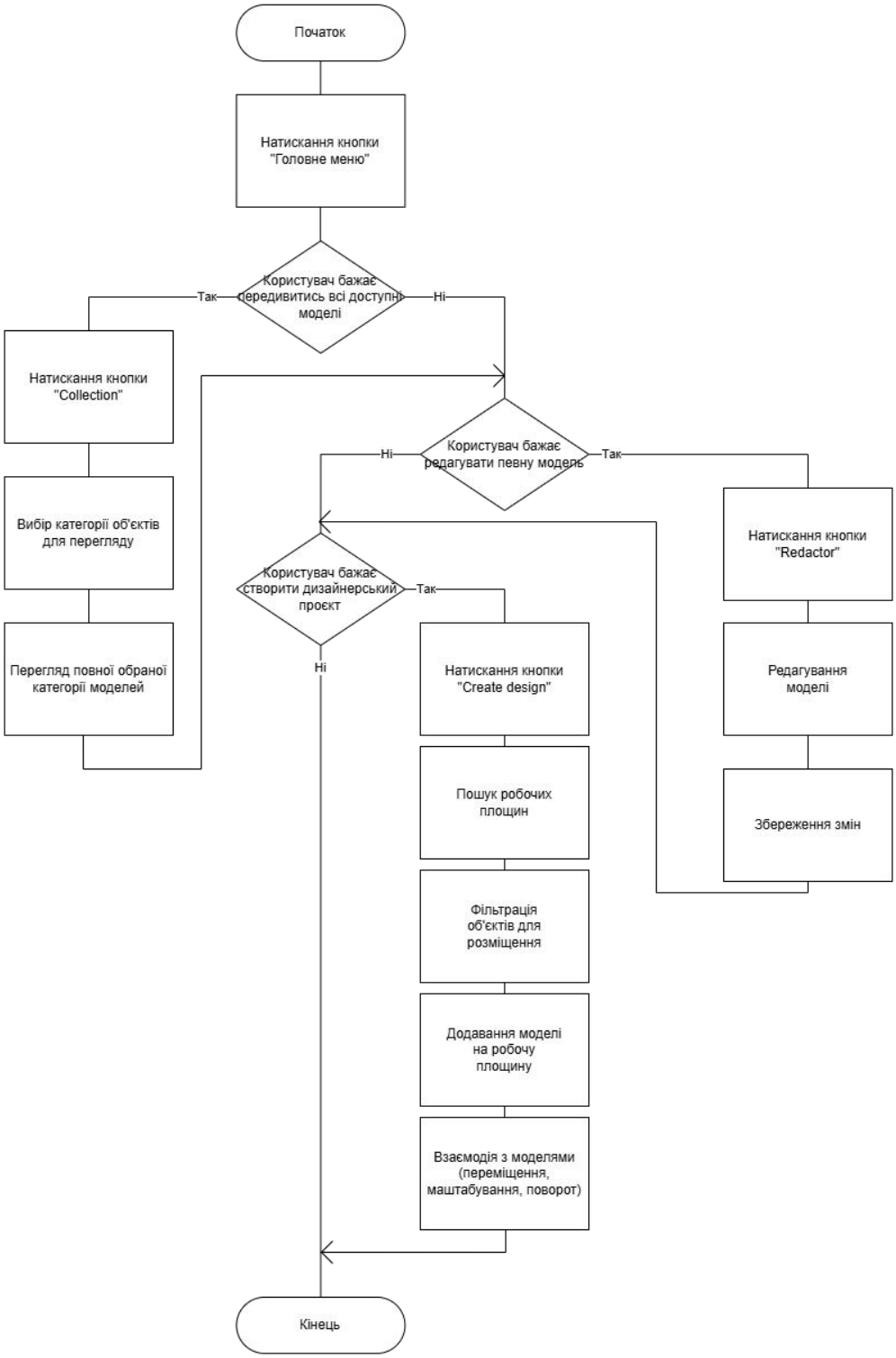
9. N. A. Shevtsiv and A. M. Striuk, "Cross platform development vs native development," in CEUR Workshop Proc., 2020. Available: http://ceur-ws.org/Vol-2731/paper10.pdf
10. A. E. Fentaw, "Cross platform mobile application development: a comparison study of React Native Vs Flutter," M.Sc. thesis, Univ. Jyväskylä, 2020. Available: https://jyx.jyu.fi/handle/123456789/70045
11. S. Xanthopoulos and S. Xinogalos, "A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications," in *Proceedings of the 5th Balkan Conference in Informatics (BCI '13)*, Thessaloniki, Greece, Sep. 2013, pp. 213–220. Available: https://doi.org/10.1145/2490257.2490302
12. A. Šmíd, "Comparison of Unity and Unreal Engine," B.Sc. thesis, Czech Tech. Univ., Prague, Czech Republic, 2017. Available: https://dspace.cvut.cz/handle/10467/84832291
13. J. M. F. Rodrigues et al., "Adaptive Card Design UI Implementation for an Augmented Reality Museum Application," in Proc. UAHCI, Las Vegas, NV, USA, 2017, pp. 433-443. Available: https://doi.org/10.1007/978-3-319-58706-6_35
14. L. Punchoojit and N. Hongwarittorrn, "Usability Studies on Mobile User Interface Design Patterns: A Systematic Literature Review," Advances in Human-Computer Interaction, vol. 2017, Article ID 6787504, 22 pages, Nov. 2017. Available: https://doi.org/10.1155/2017/6787504
15. A. Nikolarakis and P. Koutsabasis, "Mobile AR Interaction Design Patterns for Storytelling in Cultural Heritage: A Systematic Review," Multimodal Technol. Interact., vol. 8, no. 52, pp. 1-32, 2024. Available: https://doi.org/10.3390/mti8060052.
16. T. Höllerer et al., "User Interface Management Techniques for Collaborative Mobile Augmented Reality," Comput. Graph., vol. 25, no. 5, pp. 799-810, 2001. Available: https://doi.org/10.1016/S0097-8493(01)00124-7

17. J. Scheibmeir and Y. Malaiya, "Quality Model for Testing Augmented Reality Applications," in Proceedings of the IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON 2019), New York, NY, USA, Oct. 2019, pp. 1-6. Available: https://doi.org/10.1109/UEMCON47517.2019.8992974
18. H. Muñoz, C. Scully-Allison, V. Le, S. Strachan, F. C. Harris Jr., and S. Dascalu, "A Mobile Quality Assurance Application for the NRDC," University of Nevada, Reno, Reno, NV, USA. [Online]. Available: https://sensor.nevada.edu/NRDC/

ДОДАТКИ

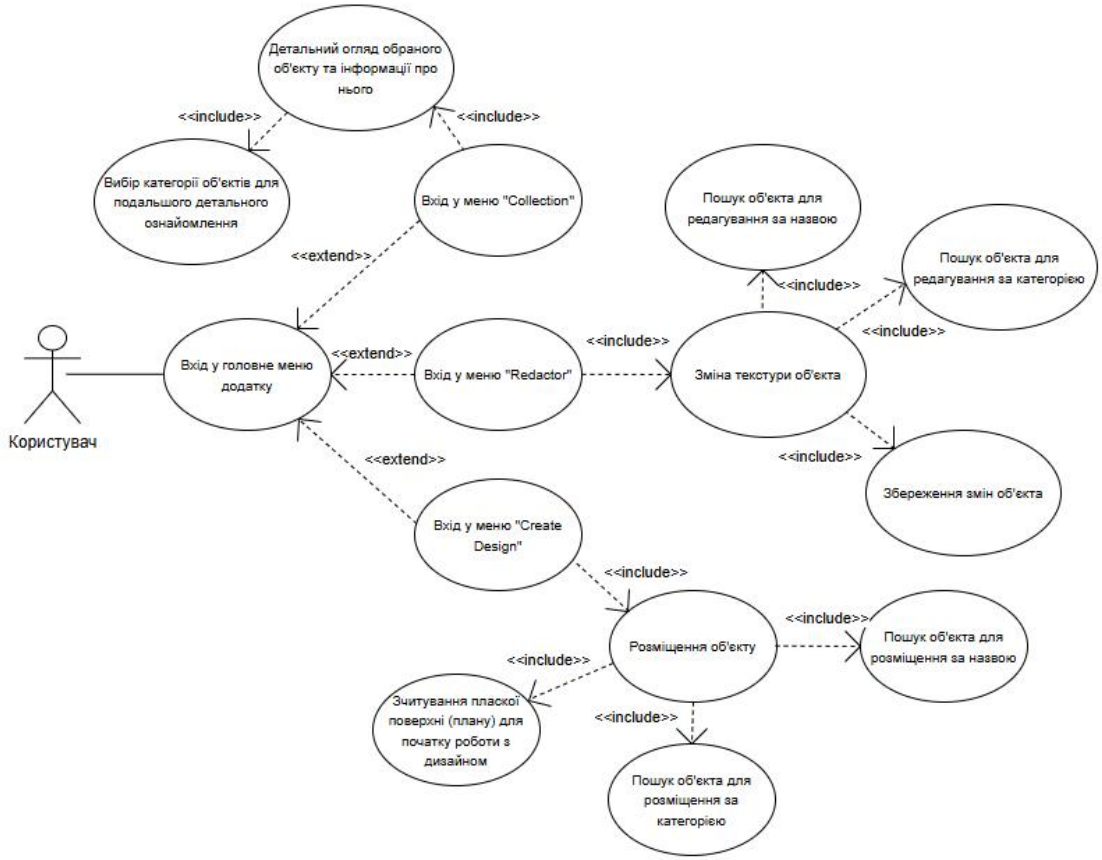
Додаток 1

Копії графічних матеріалів



ДП.045480-06-99

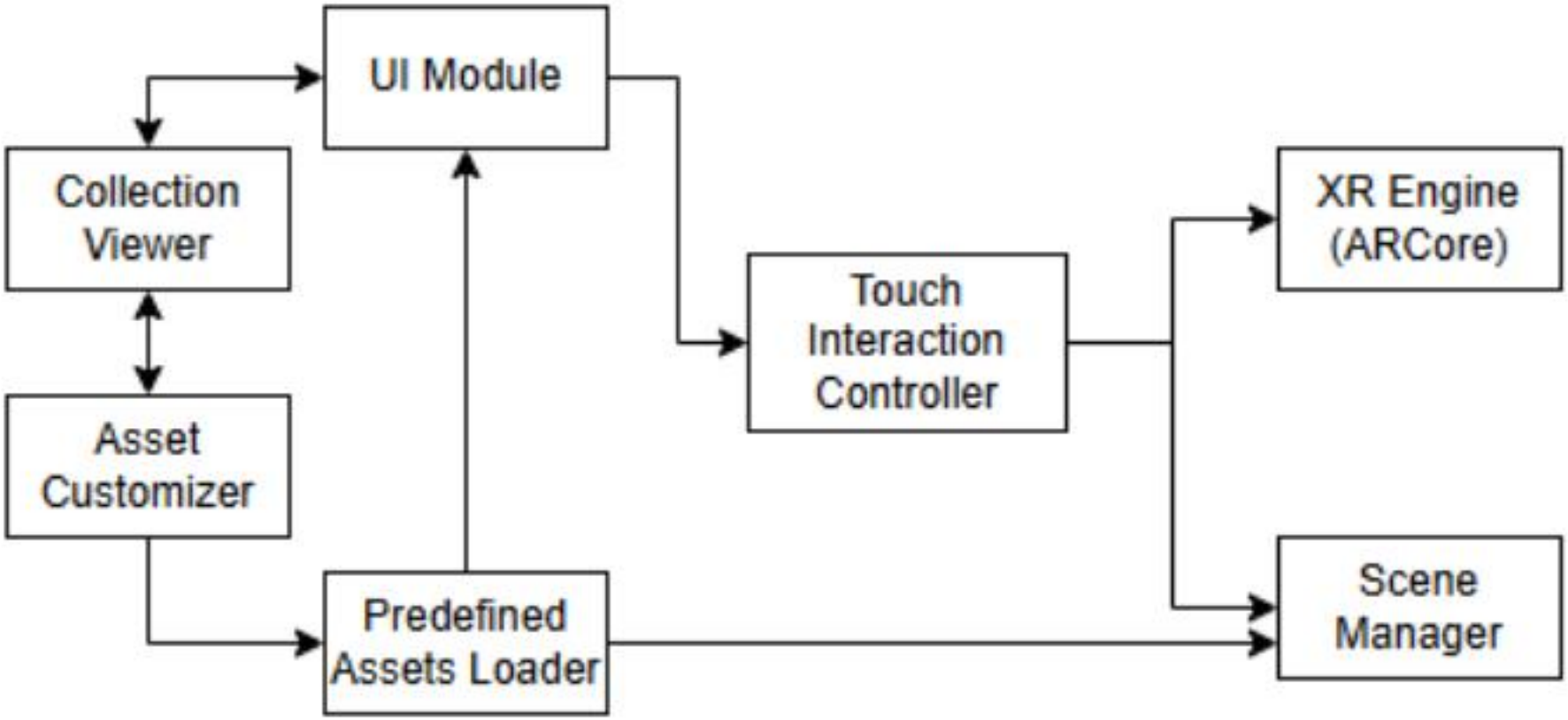
Програмне забезпечення для дизайну інтер’єрів приміщень. Діаграма діяльності основного сценарію використання. Схема алгоритму



ДП.045480-06-99

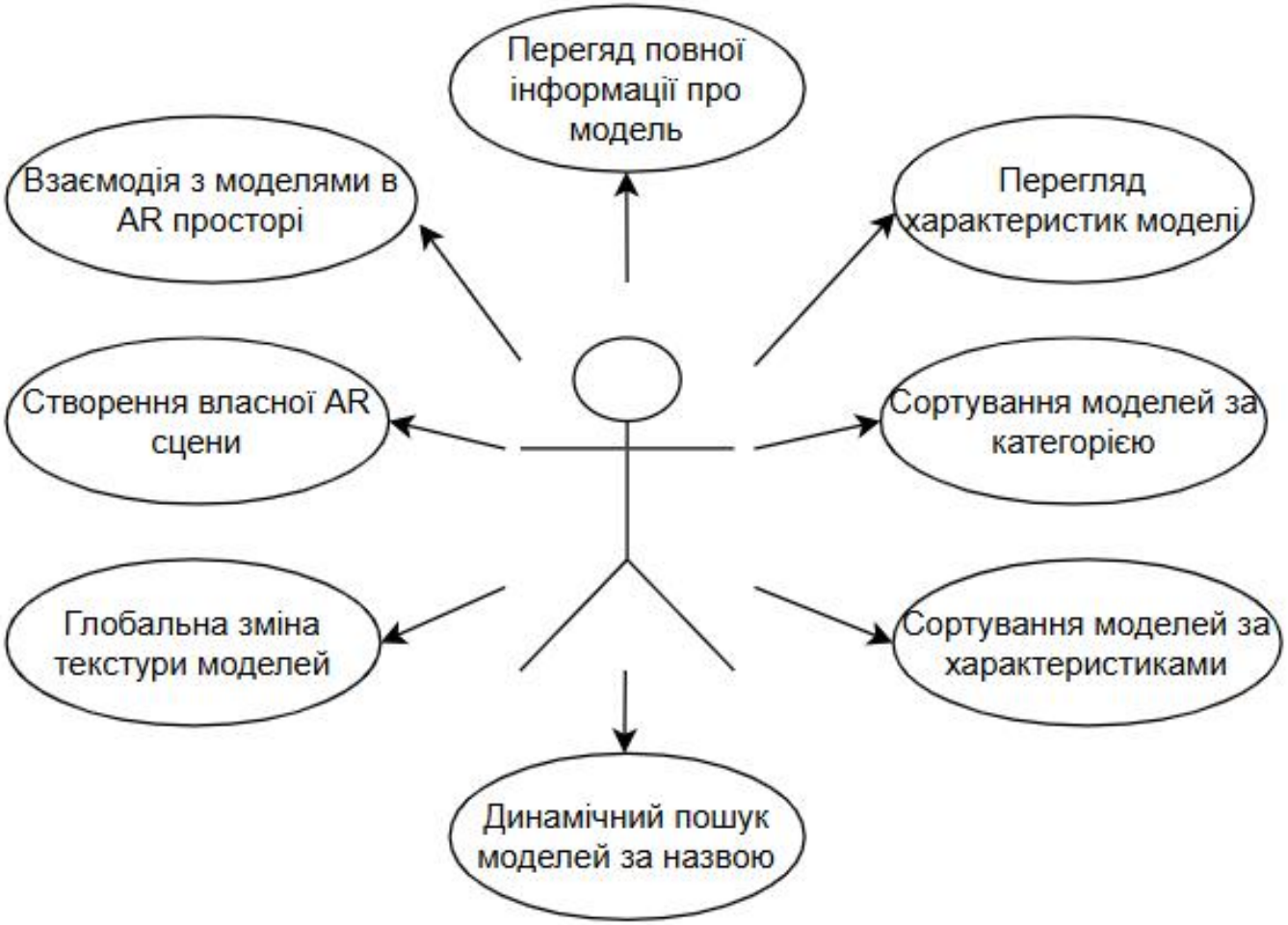
Програмне забезпечення
для дизайну інтер’єрів приміщень.
UML діаграма. Діаграма
прецендентів

Архітектура програмного забезпечення для дизайну інтер'єру приміщень



Бондарчук Данііл, група КП-11

Основні дії користувача додатку “Tumba”



Бондарчук Данііл, група КП-11

Додаток 2

Лістинг програми

```
using UnityEngine;
using UnityEngine.UI;

public class ApplyCustomMaterial : MonoBehaviour
{
    [Header("UI")]
    public Button applyMaterialButton;

    private Renderer targetRenderer;
    private Material customMaterial;
    private string MaterialKey =>
$"material_{PlayerPrefs.GetString("SelectedModelName", "default")}";

    void Start()
    {
        LoadCustomMaterialFromModelData();
        applyMaterialButton.onClick.AddListener(ApplyMaterial);
        LoadMaterial();
    }

    private void LoadCustomMaterialFromModelData()
    {
        Transform modelHolder =
GameObject.Find("ModelHolder")?.transform;
        if (modelHolder == null) return;

        foreach (Transform child in modelHolder)
        {
            if (child.gameObject.activeSelf)
            {
                RedactorModelData data =
child.GetComponent<RedactorModelData>();
                if (data != null && data.modelPrefab != null)
                {
                    targetRenderer =
data.modelPrefab.GetComponent<Renderer>();
                    customMaterial = data.custom;
                    break;
                }
            }
        }

        if (targetRenderer == null || customMaterial == null)
        {
            Debug.LogWarning("ApplyCustomMaterial: Renderer or custom
material not found.");
        }
    }

    private void ApplyMaterial()
    {
```

```
        if (targetRenderer == null || customMaterial == null) return;

        targetRenderer.sharedMaterial = customMaterial;

        PlayerPrefs.SetString(MaterialKey + "_name",
customMaterial.name);
        PlayerPrefs.DeleteKey(MaterialKey);
        PlayerPrefs.Save();
    }

    private void LoadMaterial()
    {
        if (targetRenderer == null || customMaterial == null) return;

        string savedName = PlayerPrefs.GetString(MaterialKey +
"_name", "");
        if (savedName == customMaterial.name)
        {
            targetRenderer.sharedMaterial = customMaterial;
        }
    }
}

using UnityEngine;

public class AutoScaler : MonoBehaviour
{
    [Header("Налаштування масштабування")]
    public float targetHeight = 1.0f;

    void Start()
    {
        NormalizeHeight();
    }

    void NormalizeHeight()
    {
        Bounds bounds = GetRenderableBounds(gameObject);
        float currentHeight = bounds.size.y;

        if (currentHeight > 0.001f)
        {
            float scaleMultiplier = targetHeight / currentHeight;
            transform.localScale *= scaleMultiplier;

            Debug.Log($"[AutoScaler] Масштабовано '{gameObject.name}'
до висоти {targetHeight} м. Множник масштабу: {scaleMultiplier}");
        }
        else
        {
```

```
            Debug.LogWarning($"[AutoScaler] '{gameObject.name}' має
нульову висоту! Перевір MeshRenderer.");
        }
    }

    Bounds GetRenderableBounds(GameObject go)
    {
        Renderer[] renderers = go.GetComponentsInChildren<Renderer>();

        if (renderers.Length == 0)
            return new Bounds(go.transform.position, Vector3.zero);

        Bounds combinedBounds = renderers[0].bounds;
        foreach (Renderer r in renderers)
        {
            combinedBounds.Encapsulate(r.bounds);
        }

        return combinedBounds;
    }
}

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class ButtonFilter : MonoBehaviour
{
    public TMP_Dropdown categoryDropdown;
    public Transform buttonContainer;

    private Dictionary<string, List<GameObject>> categoryButtons =
new Dictionary<string, List<GameObject>>();

    void Start()
    {
        if (categoryDropdown == null || buttonContainer == null)
            return;

        categoryButtons["all"] = new List<GameObject>();
        categoryButtons["chairs"] = new List<GameObject>();
        categoryButtons["sofas"] = new List<GameObject>();
        categoryButtons["tables"] = new List<GameObject>();
        categoryButtons["other"] = new List<GameObject>();

        foreach (Transform button in buttonContainer)
        {
            RaitingParameters category =
button.GetComponent<RaitingParameters>();
            if (category != null)
```

```
            {
                string categoryName = category.category.ToLower();

                if (!categoryButtons.ContainsKey(categoryName))
                    categoryButtons[categoryName] = new
List<GameObject>();

                categoryButtons[categoryName].Add(button.gameObject);
                categoryButtons["all"].Add(button.gameObject);
            }
        }

        categoryDropdown.onValueChanged.AddListener(delegate
{ FilterButtons(); });

        FilterButtons();
    }

    void FilterButtons()
    {
        string selected =
categoryDropdown.options[categoryDropdown.value].text.Trim().ToLower()
;

        if (selected == "all")
        {
            foreach (var button in categoryButtons["all"])
                button.SetActive(true);
            return;
        }

        if (selected == "comfort" || selected == "ergonomics" ||
selected == "universality")
        {
            List<(GameObject button, int rating)> sorted = new
List<(GameObject, int)>();

            foreach (GameObject button in categoryButtons["all"])
            {
                RaitingParameters rp =
button.GetComponent<RaitingParameters>();
                if (rp != null)
                {
                    int value = 0;
                    switch (selected)
                    {
                        case "comfort": value = rp.comfort; break;
                        case "ergonomics": value = rp.ergonomics;
break;
                        case "universality": value = rp.universality;
break;
```

```
                    }

                    sorted.Add((button, value));
                    button.SetActive(true);
                }
            }

            sorted.Sort((a, b) => b.rating.CompareTo(a.rating));

            for (int i = 0; i < sorted.Count; i++)
                sorted[i].button.transform.SetSiblingIndex(i);

            return;
        }

        foreach (var category in categoryButtons)
        {
            foreach (var button in category.Value)
                button.SetActive(false);
        }

        if (categoryButtons.ContainsKey(selected))
        {
            foreach (var button in categoryButtons[selected])
                button.SetActive(true);
        }
    }
}
using UnityEngine;
using UnityEngine.UI;

public class ColorSliderToMaterial : MonoBehaviour
{
    public Slider redSlider;
    public Slider greenSlider;
    public Slider blueSlider;

    public Button applyButton;

    private Material customMaterial;

    void Start()
    {
        LoadCustomMaterialFromActiveModel();
        applyButton.onClick.AddListener(ApplyColorFromSliders);
    }

    private void LoadCustomMaterialFromActiveModel()
    {
        Transform modelHolder =
GameObject.Find("ModelHolder")?.transform;
```

```
        if (modelHolder == null) return;

        foreach (Transform child in modelHolder)
        {
            if (child.gameObject.activeSelf)
            {
                RedactorModelData data =
child.GetComponent<RedactorModelData>();
                if (data != null)
                {
                    customMaterial = data.custom;
                    break;
                }
            }
        }
    }

    private void ApplyColorFromSliders()
    {
        if (customMaterial == null)
            return;

        Color newColor = new Color(
            redSlider.value,
            greenSlider.value,
            blueSlider.value
        );

        customMaterial.color = newColor;

        PlayerPrefs.SetFloat("color_r", redSlider.value);
        PlayerPrefs.SetFloat("color_g", greenSlider.value);
        PlayerPrefs.SetFloat("color_b", blueSlider.value);
        PlayerPrefs.Save();
    }
}

using UnityEngine;

public class ObjectRemover : MonoBehaviour
{
    public float doubleTapTime = 0.3f;

    private float lastTapTime = 0f;
    private GameObject lastTappedObject;

    void Update()
    {
        if (Input.touchCount == 1)
        {
            Touch touch = Input.GetTouch(0);
```

```
            if (touch.phase == TouchPhase.Began)
            {
                Ray ray =
Camera.main.ScreenPointToRay(touch.position);
                RaycastHit hit;

                if (Physics.Raycast(ray, out hit))
                {
                    GameObject tappedObject =
hit.transform.gameObject;

                    if (lastTappedObject == tappedObject && Time.time
- lastTapTime < doubleTapTime)
                    {

Destroy(tappedObject.transform.root.gameObject);
                    }

                    lastTapTime = Time.time;
                    lastTappedObject = tappedObject;
                }
            }
        }
    }
}
using UnityEngine;
using UnityEngine.UI;

public class DropdownMenuController : MonoBehaviour
{
    private Button menuButton;
    private GameObject menuOptionsHolder;
    private GameObject background;

    private bool menuOpen = false;

    void Start()
    {
        menuButton =
GameObject.Find("MenuButton").GetComponent<Button>();
        menuOptionsHolder = GameObject.Find("MenuOptionsHolder");
        background = GameObject.Find("Background");

        if (menuButton == null || menuOptionsHolder == null ||
background == null)
        {
            Debug.LogError("Не знайдено один з об'єктів: MenuButton,
MenuOptionsHolder або Background");
            return;
        }
```

```
        SetMenuVisible(false);

        menuButton.onClick.AddListener(ToggleMenu);

        Button[] childButtons =
menuOptionsHolder.GetComponentsInChildren<Button>(true);
        foreach (Button btn in childButtons)
        {
            btn.onClick.AddListener(CloseMenu);
        }
    }

    void ToggleMenu()
    {
        menuOpen = !menuOpen;
        SetMenuVisible(menuOpen);
    }

    void CloseMenu()
    {
        menuOpen = false;
        SetMenuVisible(false);
    }

    void SetMenuVisible(bool visible)
    {
        menuOptionsHolder.SetActive(visible);
        background.SetActive(visible);

        foreach (Transform child in menuOptionsHolder.transform)
        {
            child.gameObject.SetActive(visible);
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class NewBehaviourScript : MonoBehaviour
{
    public TMP_InputField searchInput;
    public Transform buttonContainer;

    private List<GameObject> allButtons = new List<GameObject>();

    void Start()
    {
        if (searchInput == null || buttonContainer == null)
        {
```

```
            Debug.LogError("SearchInput або ButtonContainer не
призначені!");
            return;
        }

        foreach (Transform button in buttonContainer)
        {
            allButtons.Add(button.gameObject);
        }

        searchInput.onValueChanged.AddListener(delegate
{ FilterByName(); });

        FilterByName();
    }

    void FilterByName()
    {
        string searchText = searchInput.text.ToLower();
        Debug.Log("Пошук: " + searchText);

        foreach (GameObject button in allButtons)
        {
            RaitingParameters param =
button.GetComponent<RaitingParameters>();
            if (param != null)
            {
                string objName = param.name.ToLower();
                bool match = objName.Contains(searchText);
                button.SetActive(match ||
string.IsNullOrEmpty(searchText));
            }
            else
            {
                button.SetActive(false);
            }
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using Unity.XR.CoreUtils;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class FurniturePlacementManager : MonoBehaviour
{
    [Header("UI")]
```

```
    public Button[] imageButtons;
    public GameObject[] furniturePrefabs;
    public Sprite[] furnitureIcons;

    [Header("AR")]
    public XROrigin xrOrigin;
    public ARRaycastManager raycastManager;
    public ARPlaneManager planeManager;

    private GameObject SpawnableFurniture;
    private List<ARRaycastHit> raycastHits = new List<ARRaycastHit>();

    private float lastTapTime = 0f;
    private const float doubleTapThreshold = 0.3f;

    void Start()
    {

        foreach (var button in imageButtons)
        {
            button.onClick.AddListener(() =>
            {
                Sprite clickedSprite = button.image.sprite;
                GameObject foundPrefab =
FindPrefabBySprite(clickedSprite);

                if (foundPrefab != null)
                {
                    SwitchFurniture(foundPrefab);
                }
                else
                {
                    Debug.LogWarning("Prefab not found for sprite: "
+ clickedSprite.name);
                }
            });
        }
    }

    void Update()
    {
        if (Input.touchCount > 0 && Input.GetTouch(0).phase ==
TouchPhase.Began)
        {
            if (Time.time - lastTapTime < doubleTapThreshold &&
isButtonPressed() == false)
            {
                bool collision =
raycastManager.Raycast(Input.GetTouch(0).position, raycastHits,
TrackableType.PlaneWithinPolygon);
                if (collision && SpawnableFurniture != null)
```

```
                {
                    GameObject _object =
Instantiate(SpawnableFurniture);
                    _object.transform.position =
raycastHits[0].pose.position;
                    _object.transform.rotation =
raycastHits[0].pose.rotation;

                    foreach (var plane in planeManager.trackables)
                    {
                        plane.gameObject.SetActive(false);
                    }

                    planeManager.enabled = false;
                }
            }

            lastTapTime = Time.time;
        }
    }

    GameObject FindPrefabBySprite(Sprite sprite)
    {
        for (int i = 0; i < furnitureIcons.Length; i++)
        {
            if (furnitureIcons[i] == sprite)
            {
                return furniturePrefabs[i];
            }
        }

        return null;
    }

    public void SwitchFurniture(GameObject furniture)
    {
        SpawnableFurniture = furniture;
    }

    public bool isButtonPressed()
    {
        return
EventSystem.current.currentSelectedGameObject?.GetComponent<Button>()
 != null;
    }
}
using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;
using TMPro;
```

```
public class GalleryScroll : MonoBehaviour
{
    [Header("UI Elements")]
    public Image[] displaySlots;
    public Button nextButton;
    public Button prevButton;
    public TMP_Dropdown categoryDropdown;

    [Header("Data")]
    public List<Sprite> allImages;
    public List<Sprite> chairImages;
    public List<Sprite> tableImages;
    public List<Sprite> sofaImages;
    public List<Sprite> otherImages;
    public List<Sprite> comfortImages;
    public List<Sprite> ergonomicsImages;
    public List<Sprite> universalityImages;

    private Dictionary<string, List<Sprite>> imageCategories;
    private List<Sprite> currentImageList;
    private int currentIndex = 0;

    void Start()
    {

        nextButton.onClick.AddListener(Next);
        prevButton.onClick.AddListener(Previous);

categoryDropdown.onValueChanged.AddListener(OnCategoryChanged);


        imageCategories = new Dictionary<string, List<Sprite>>()
        {
            { "All", allImages },
            { "Chairs", chairImages },
            { "Tables", tableImages },
            { "Sofas", sofaImages },
            { "Other", otherImages },
            { "Comfort", comfortImages },
            { "Ergonomics", ergonomicsImages },
            { "Universality", universalityImages }
        };


        currentImageList = allImages;
        UpdateGallery();
    }

    void OnCategoryChanged(int index)
    {
        string selected = categoryDropdown.options[index].text;
```

```
        if (imageCategories.ContainsKey(selected))
        {
            currentImageList = imageCategories[selected];
            currentIndex = 0;
            UpdateGallery();
        }
        else
        {
            Debug.LogWarning("No category found for: " + selected);
        }
    }

    void Next()
    {
        if (currentImageList.Count == 0) return;
        currentIndex = (currentIndex + 1) % currentImageList.Count;
        UpdateGallery();
    }

    void Previous()
    {
        if (currentImageList.Count == 0) return;
        currentIndex = (currentIndex - 1 + currentImageList.Count) %
currentImageList.Count;
        UpdateGallery();
    }

    void UpdateGallery()
    {
        for (int i = 0; i < displaySlots.Length; i++)
        {
            if (currentImageList.Count == 0)
            {
                displaySlots[i].sprite = null;
                continue;
            }

            int index = (currentIndex + i) % currentImageList.Count;
            displaySlots[i].sprite = currentImageList[index];
        }
    }
}
using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;
using TMPro;

public class GlobalMaterialChanger : MonoBehaviour
{
    [Header("UI та Стилі")]
    [SerializeField] private GameObject[] containers;
```

```
    [SerializeField] private TextMeshProUGUI styleNameText;

    [Header("Цільовий об'єкт (заповнюється автоматично)")]
    private Material[] materials;
    private string[] styleNames;
    private Sprite[] styleIcons;
    private Renderer targetRenderer;

    private Dictionary<Button, int> buttonMaterialPairs = new
Dictionary<Button, int>();
    private Dictionary<int, GameObject> indexToOutline = new
Dictionary<int, GameObject>();

    private string MaterialKey =>
$"material_{PlayerPrefs.GetString("SelectedModelName", "default")}";

    void Start()
    {
        LoadDataFromActiveModel();
        InitializeButtons();
        LoadSavedMaterial();
    }

    private void LoadDataFromActiveModel()
    {
        Transform modelHolder =
GameObject.Find("ModelHolder")?.transform;
        if (modelHolder == null) return;

        foreach (Transform child in modelHolder)
        {
            if (child.gameObject.activeSelf)
            {
                RedactorModelData data =
child.GetComponent<RedactorModelData>();
                if (data != null)
                {
                    materials = data.materials;
                    styleNames = data.styleNames;
                    styleIcons = data.styleIcons;

                    if (data.modelPrefab != null)
                    {
                        targetRenderer =
data.modelPrefab.GetComponent<Renderer>();
                    }
                }
            }
        }
    }
```

```
    private void InitializeButtons()
    {
        for (int i = 0; i < containers.Length; i++)
        {
            int materialIndex = i;
            Button btn =
containers[i].GetComponentInChildren<Button>();
            Transform outline =
containers[i].transform.Find("Outline");


            Image img = btn?.GetComponent<Image>();
            if (img != null && styleIcons != null && materialIndex <
styleIcons.Length)
            {
                img.sprite = styleIcons[materialIndex];
            }

            if (btn != null)
            {
                buttonMaterialPairs.Add(btn, materialIndex);
                btn.onClick.AddListener(() =>
OnMaterialButtonClicked(materialIndex));

                if (outline != null)
                {
                    indexToOutline[materialIndex] =
outline.gameObject;
                    outline.gameObject.SetActive(false);
                }
            }
        }
    }

    private void OnMaterialButtonClicked(int materialIndex)
    {
        if (materialIndex < 0 || materialIndex >= materials.Length)
return;

        ApplyMaterialByIndex(materialIndex);
        UpdateOutlines(materialIndex);
        UpdateStyleName(materialIndex);
    }

    public void ApplyMaterialByIndex(int index)
    {
        if (materials == null || targetRenderer == null) return;

        if (index >= 0 && index < materials.Length)
        {
            targetRenderer.sharedMaterial = materials[index];
```

```
            PlayerPrefs.SetInt(MaterialKey, index);
            PlayerPrefs.DeleteKey(MaterialKey + "_name");
            PlayerPrefs.Save();
        }
    }

    private void LoadSavedMaterial()
    {
        string customName = PlayerPrefs.GetString(MaterialKey +
"_name", "");

        if (!string.IsNullOrEmpty(customName))
        {
            for (int i = 0; i < materials.Length; i++)
            {
                if (materials[i] != null && materials[i].name ==
customName)
                {
                    targetRenderer.sharedMaterial = materials[i];
                    UpdateOutlines(i);
                    UpdateStyleName(i);
                    return;
                }
            }
        }

        int savedIndex = PlayerPrefs.GetInt(MaterialKey, -1);
        if (savedIndex >= 0 && savedIndex < materials.Length)
        {
            targetRenderer.sharedMaterial = materials[savedIndex];
            UpdateOutlines(savedIndex);
            UpdateStyleName(savedIndex);
        }
    }

    private void UpdateOutlines(int activeIndex)
    {
        foreach (var kvp in indexToOutline)
        {
            kvp.Value.SetActive(kvp.Key == activeIndex);
        }
    }

    private void UpdateStyleName(int index)
    {
        if (styleNameText != null && styleNames != null && index <
styleNames.Length)
        {
            styleNameText.text = "Current Style: " +
styleNames[index];
        }
```

```
    }
}

using UnityEngine;

public class MaterialChanger : MonoBehaviour
{
    public Material[] materials;
    public Renderer targetRenderer;

    private string MaterialKey =>
$"material_{PlayerPrefs.GetString("SelectedModelName", "default")}";

    void Start()
    {
        if (targetRenderer != null)
        {
            LoadSavedMaterial();
        }
        else
        {
            Debug.LogWarning("MaterialChanger: targetRenderer не
призначений!");
        }
    }

    public void ApplyMaterialByIndex(int index)
    {
        if (materials == null || targetRenderer == null) return;

        if (index >= 0 && index < materials.Length)
        {
            targetRenderer.sharedMaterial = materials[index];
            PlayerPrefs.SetInt(MaterialKey, index);
            PlayerPrefs.DeleteKey(MaterialKey + "_name");
            PlayerPrefs.Save();
        }
    }

    private void LoadSavedMaterial()
    {
        if (targetRenderer == null) return;

        string customName = PlayerPrefs.GetString(MaterialKey +
"_name", "");

        if (!string.IsNullOrEmpty(customName))
        {
            foreach (var mat in materials)
            {
                if (mat != null && mat.name == customName)
```

```
                {
                    targetRenderer.sharedMaterial = mat;
                    return;
                }
            }
        }

        int savedIndex = PlayerPrefs.GetInt(MaterialKey, -1);
        if (savedIndex >= 0 && savedIndex < materials.Length)
        {
            targetRenderer.sharedMaterial = materials[savedIndex];
        }
    }
}

using UnityEngine;
using TMPro;

public class ModelActivator : MonoBehaviour
{
    [SerializeField] public TextMeshProUGUI modelNameText;

    void Start()
    {
        string selectedName =
PlayerPrefs.GetString("SelectedModelName", "");

        foreach (Transform child in transform)
        {
            var data = child.GetComponent<RedactorModelData>();
            if (data != null)
            {
                bool isActive = data.modelName == selectedName;
                child.gameObject.SetActive(isActive);

                if (isActive && modelNameText != null)
                {
                    modelNameText.text = "Model Name: " +
data.modelName;
                }
            }
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
```

```
public class ObjectDragger : MonoBehaviour
{
    public float holdTimeThreshold = 0.3f;
    private float touchStartTime;
    private bool isDragging = false;

    private GameObject selectedObject;

    private ARRaycastManager raycastManager;
    private Camera arCamera;
    private static List<ARRaycastHit> hits = new List<ARRaycastHit>();

    void Start()
    {
        raycastManager = FindObjectOfType<ARRaycastManager>();
        arCamera = Camera.main;
    }

    void Update()
    {
        if (Input.touchCount == 1)
        {
            Touch touch = Input.GetTouch(0);


            if (touch.phase == TouchPhase.Began)
            {
                touchStartTime = Time.time;
                Ray ray = arCamera.ScreenPointToRay(touch.position);
                if (Physics.Raycast(ray, out RaycastHit hit))
                {

                    selectedObject = hit.transform.root.gameObject;
                }
            }


            else if (touch.phase == TouchPhase.Stationary &&
selectedObject != null)
            {
                if (!isDragging && Time.time - touchStartTime >=
holdTimeThreshold)
                {
                    isDragging = true;
                }
            }
            else if (touch.phase == TouchPhase.Moved && isDragging &&
selectedObject != null)
            {
                if (raycastManager.Raycast(touch.position, hits,
TrackableType.PlaneWithinPolygon))
```

```
                {
                    Pose hitPose = hits[0].pose;
                    selectedObject.transform.position =
hitPose.position;
                }
            }


            else if (touch.phase == TouchPhase.Ended || touch.phase
== TouchPhase.Canceled)
            {
                isDragging = false;
                selectedObject = null;
            }
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RaitingParameters : MonoBehaviour
{
    public string name;
    public string category;
    public byte comfort;
    public byte ergonomics;
    public byte universality;
}

using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;

public class OutlineButtonManager : MonoBehaviour
{
    [SerializeField] private GameObject[] containers;

    private Dictionary<Button, GameObject> buttonOutlinePairs = new
Dictionary<Button, GameObject>();

    void Start()
    {

        foreach (GameObject container in containers)
        {
            Button btn = container.GetComponentInChildren<Button>();
            Transform outline = container.transform.Find("Outline");

            if (btn != null && outline != null)
```

```
            {
                buttonOutlinePairs.Add(btn, outline.gameObject);

                outline.gameObject.SetActive(false);

                btn.onClick.AddListener(() => OnButtonClicked(btn));
            }
            else
            {
                Debug.LogWarning($"Button або Outline не знайдені у
контейнері {container.name}");
            }
        }
    }

    private void OnButtonClicked(Button clickedButton)
    {
        foreach (var pair in buttonOutlinePairs)
        {

            pair.Value.SetActive(pair.Key == clickedButton);
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RedactorModelData : MonoBehaviour
{
    public string modelName;
    public GameObject modelPrefab;
    public Material[] materials;
    public string[] styleNames;
    public Sprite[] styleIcons;
    public Material custom;
}

using UnityEngine;

public class SimpleObjectRotator : MonoBehaviour
{

    public float rotationSpeed = 0.2f;

    private GameObject selectedObject;

    void Update()
    {
```

```
        if (Input.touchCount == 1)
        {
            Touch touch = Input.GetTouch(0);

            if (touch.phase == TouchPhase.Began)
            {

                Ray ray =
Camera.main.ScreenPointToRay(touch.position);
                RaycastHit hit;

                if (Physics.Raycast(ray, out hit))
                {

                    selectedObject = hit.transform.root.gameObject;
                }
            }
            else if (touch.phase == TouchPhase.Moved &&
selectedObject != null)
            {

                float deltaX = touch.deltaPosition.x;
                selectedObject.transform.Rotate(Vector3.up, -deltaX *
rotationSpeed, Space.World);
            }
            else if (touch.phase == TouchPhase.Ended || touch.phase
== TouchPhase.Canceled)
            {

                selectedObject = null;
            }
        }
    }
}

using UnityEngine;

public class ObjectScaler : MonoBehaviour
{
    public float minScale = 0.1f;
    public float maxScale = 3.0f;
    public float scaleSpeed = 0.01f;

    private float previousDistance;
    private bool isScaling = false;

    void Update()
    {
        if (Input.touchCount == 2)
        {
            Touch touchZero = Input.GetTouch(0);
```

```
            Touch touchOne = Input.GetTouch(1);

            Ray ray0 =
Camera.main.ScreenPointToRay(touchZero.position);
            Ray ray1 =
Camera.main.ScreenPointToRay(touchOne.position);

            RaycastHit hit0, hit1;

            bool isTouchingObject0 = Physics.Raycast(ray0, out hit0);
            bool isTouchingObject1 = Physics.Raycast(ray1, out hit1);


            if (isTouchingObject0 && isTouchingObject1 &&
                hit0.transform.root == hit1.transform.root)
            {
                GameObject targetRoot =
hit0.transform.root.gameObject;

                float currentDistance =
Vector2.Distance(touchZero.position, touchOne.position);

                if (!isScaling)
                {
                    previousDistance = currentDistance;
                    isScaling = true;
                }
                else
                {
                    float distanceDelta = currentDistance -
previousDistance;
                    float scaleFactor = 1 + (distanceDelta *
scaleSpeed);

                    Vector3 newScale =
targetRoot.transform.localScale * scaleFactor;
                    newScale.x = Mathf.Clamp(newScale.x, minScale,
maxScale);
                    newScale.y = Mathf.Clamp(newScale.y, minScale,
maxScale);
                    newScale.z = Mathf.Clamp(newScale.z, minScale,
maxScale);

                    targetRoot.transform.localScale = newScale;
                    previousDistance = currentDistance;
                }
            }
            else
            {
                isScaling = false;
            }
```

```
        }
        else
        {
            isScaling = false;
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneController : MonoBehaviour
{
    public void SwitchScenes(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }

}

using UnityEngine;
using TMPro;

public class ScreenLogger : MonoBehaviour
{
    public TextMeshProUGUI logText;
    [TextArea] public string logContent = "";
    public int maxLines = 15;

    void OnEnable()
    {
        Application.logMessageReceived += HandleLog;
    }

    void OnDisable()
    {
        Application.logMessageReceived -= HandleLog;
    }

    void HandleLog(string logString, string stackTrace, LogType type)
    {
        string prefix = "";
        switch (type)
        {
            case LogType.Warning: prefix = "<color=yellow>[W]</color>
"; break;
            case LogType.Error:
            case LogType.Exception: prefix = "<color=red>[E]</color>
"; break;
```

```
            default: prefix = "<color=white>"; break;
        }

        logContent += prefix + logString + "</color>\n";


        var lines = logContent.Split('\n');
        if (lines.Length > maxLines)
        {
            logContent = string.Join("\n", lines, lines.Length -
maxLines - 1, maxLines);
        }

        if (logText != null)
        {
            logText.text = logContent;
        }
    }
}

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Toggle3DObjects : MonoBehaviour
{
    public List<GameObject> objectList;
    public Button nextButton;
    public Button prevButton;

    private int currentIndex = 0; // Індекс поточного об'єкта

    void Start()
    {

        if (objectList == null || objectList.Count == 0)
        {
            Debug.LogError("objectList не заданий або порожній!");
            return;
        }


        HideAllObjects();


        objectList[currentIndex].SetActive(true);


        if (nextButton != null)
            nextButton.onClick.AddListener(ShowNextObject);
```

```
        if (prevButton != null)
            prevButton.onClick.AddListener(ShowPreviousObject);
    }

    private void HideAllObjects()
    {

        foreach (var obj in objectList)
        {
            if (obj != null)
                obj.SetActive(false);
        }
    }

    private void ShowNextObject()
    {

        objectList[currentIndex].SetActive(false);

        currentIndex = (currentIndex + 1) % objectList.Count;

        objectList[currentIndex].SetActive(true);
    }

    private void ShowPreviousObject()
    {

        objectList[currentIndex].SetActive(false);


        currentIndex = (currentIndex - 1 + objectList.Count) %
objectList.Count;

        objectList[currentIndex].SetActive(true);
    }
}

using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;

public class ImageCarousel : MonoBehaviour
{
    public Button nextButton;
    public Button previousButton;
    public Image[] imageButtons;
    public List<Sprite> imageList;

    private int currentStartIndex = 0;

    void Start()
```

```
    {

        nextButton.onClick.AddListener(ShowNextImages);
        previousButton.onClick.AddListener(ShowPreviousImages);


        UpdateImages();
    }

    void UpdateImages()
    {

        for (int i = 0; i < imageButtons.Length; i++)
        {
            int index = currentStartIndex + i;

            if (index < imageList.Count)
            {
                imageButtons[i].sprite = imageList[index];
                imageButtons[i].gameObject.SetActive(true);
            }
            else
            {
                imageButtons[i].gameObject.SetActive(false);
            }
        }

        previousButton.interactable = currentStartIndex > 0;
        nextButton.interactable = currentStartIndex +
imageButtons.Length < imageList.Count;
    }

    void ShowNextImages()
    {

        if (currentStartIndex + imageButtons.Length < imageList.Count)
        {
            currentStartIndex += imageButtons.Length;
            UpdateImages();
        }
    }

    void ShowPreviousImages()
    {

        if (currentStartIndex - imageButtons.Length >= 0)
        {
            currentStartIndex -= imageButtons.Length;
            UpdateImages();
        }
    }
```

```
}

using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class TextNavigator : MonoBehaviour
{

    [SerializeField] private string[] texts;

    private int[] nextIndex;
    private int[] prevIndex;
    private int currentIndex = 0;

    public TextMeshProUGUI inputField;
    public Button nextButton;
    public Button prevButton;

    void Start()
    {
        if (texts == null || texts.Length == 0)
        {
            Debug.LogError("TextNavigator: Масив текстів порожній!");
            return;
        }

        nextIndex = new int[texts.Length];
        prevIndex = new int[texts.Length];

        for (int i = 0; i < texts.Length; i++)
        {
            nextIndex[i] = (i + 1) % texts.Length;
            prevIndex[i] = (i - 1 + texts.Length) % texts.Length;
        }

        UpdateText();

        nextButton.onClick.AddListener(ShowNextText);
        prevButton.onClick.AddListener(ShowPreviousText);
    }

    private void UpdateText()
    {
        if (inputField != null && texts.Length > 0)
        {
            inputField.text = texts[currentIndex];
        }
    }

    private void ShowNextText()
```

```
{
    currentIndex = nextIndex[currentIndex];
    UpdateText();
}

private void ShowPreviousText()
{
    currentIndex = prevIndex[currentIndex];
    UpdateText();
}
```

Додаток 3

Копії графічних матеріалів

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

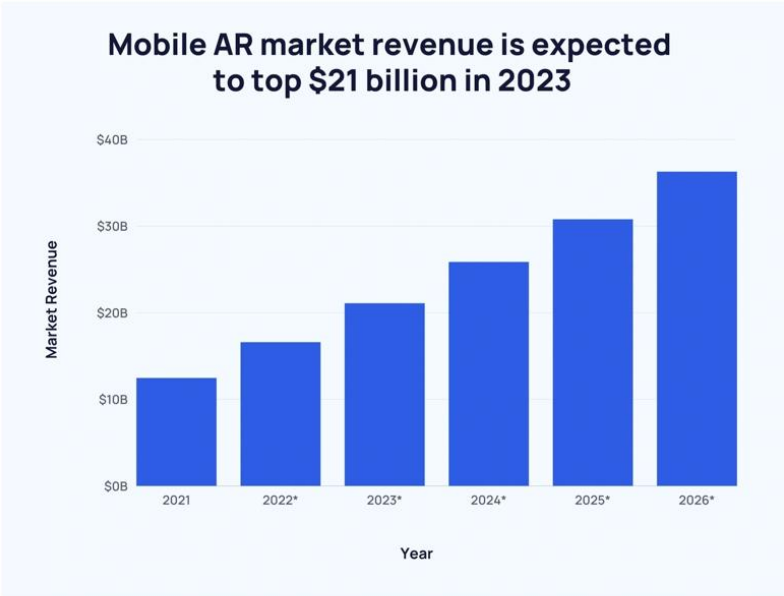
Програмне забезпечення для дизайну інтер’єрів приміщень

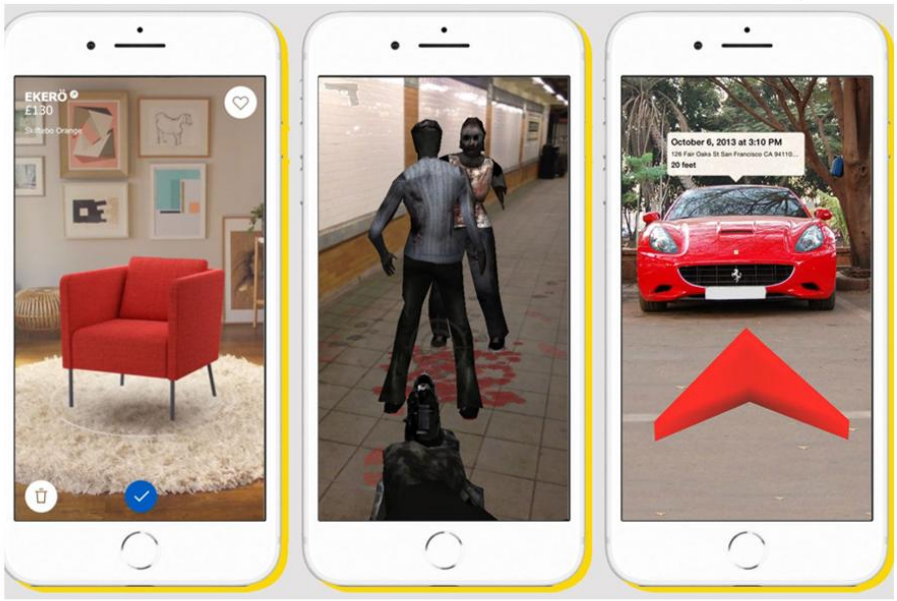
Виконав: Бондарчук Данііл Сергійович

Керівник: доцент кафедри ПЗКС, к.т.н, доцент Новак Д.С.

Київ – 2025

АКТУАЛЬНІСТЬ





МЕТА ДИПЛОМНОГО ПРОЄКТУ

Створити дружній для користувача та інтуїтивний додаток для дизайну приміщень, який допомагає візуалізовувати дизайнерські ідеї та створювати ефективні інтер’єрні рішення.

ЗАВДАННЯ ДИПЛОМНОГО ПРОЄКТУ

1. Дослідити проблеми, що виникають при дизайні інтер'єрів приміщень.
2. Проаналізувати ринок ПЗ для дизайну приміщень.
3. Визначити вимоги до розроблюваного ПЗ.
4. Проаналізувати та обрати засоби реалізації ПЗ
5. Розробити програмне забезпечення для дизайну інтер'єрів приміщень відповідно до поставлених вимог
6. Прровести тестування створеного додатку та проаналізувати результати
7. Сформулювати шляхи подальшого розвитку проєкту

ДЕРЕВО ПРОБЛЕМ


КАФЕДРА • ПРОГРАМНОГО • ЗАБЕЗПЕЧЕННЯ
СИСТЕМ • ФПМ • КОМП'ЮТЕРНИХ
КПІ ім. Ігоря Сікорського

АНАЛОГИ


Національний технічний університет України
"Київський політехнічний інститут"


wayfair


amikasa


houzz


DecorMatters

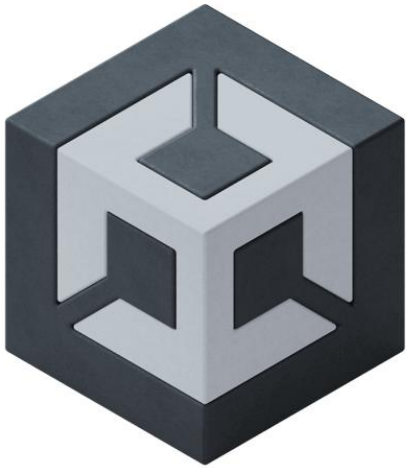
ПОРІВНЯННЯ З АНАЛОГАМИ

	Wayfair	Amikasa	Tumba
Є онлайн-магазином?			
Чи наявний високий “поріг входження” для пересічного користувача?			
Чи містить можлдивість динамічної взаємодії з моделями у AR просторі?			
Чи наявна можливість редагування текстур та матеріалів для усіх моделей?			
Чи можлива робота додатку без підключення до мережі інтернет?			

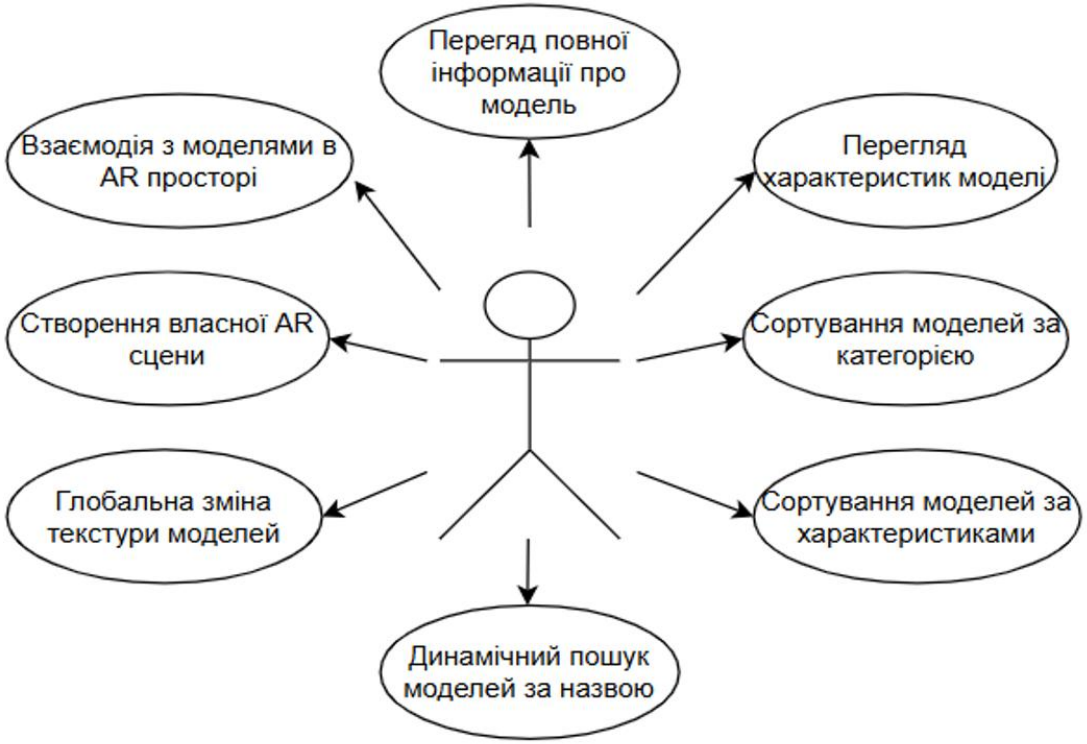
ФУНКЦІОНАЛЬНІ ВИМОГИ

1. Перегляд 3D моделей та ознайомлення з детальною інформацією про них.
2. Функція сортування об’єктів за категоріями.
3. Функція пошуку об’єкта за категорією/назвою.
4. Можливість розміщати 3d об’єкти в просторі за допомогою AR технології.
5. Можливість переміщувати та масштабувати об’єкти в просторі за допомогою AR технології.
6. Можливість змінювати текстури об’єктів.
7. Можливість власноруч створювати матеріали для 3D моделей.

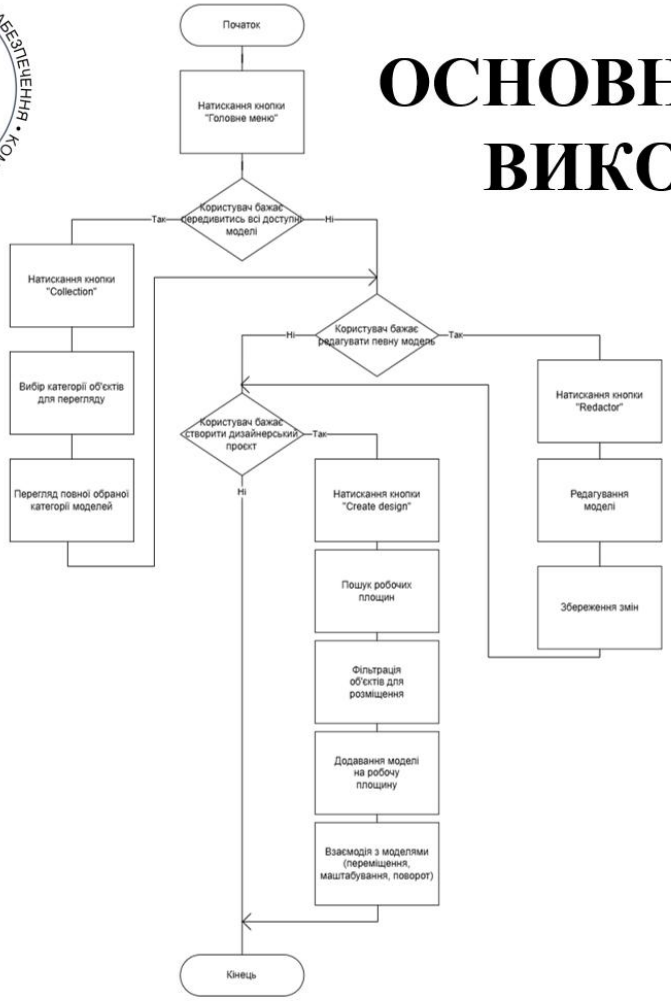
ВИБІР ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ



ДІАГРАМА ОСНОВНИХ ДІЙ КОРИСТУВАЧА



ОСНОВНИЙ СЦЕНАРІЙ ВИКОРИСТАННЯ



ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПЗ

- Основний фокус був на проведенні ручному наскрізному тестуванню;
- Наскрізне тестування охоплювало перевірку функціональності всіх основних модулів;
- Тестування проводилося з урахуванням типових сценаріїв користувача;

ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ

1. Співпрацювання з меблевими компаніями.
2. Впровадження системи рекомендацій.
3. Реалізація функції поширення створених дизайнів.
4. Колаборація з освітніми платформами.
5. Розширення бібліотеки моделей.

ВИСНОВКИ

1. Проаналізовано існуючі програмні рішення.
2. Спроєктовано архітектуру додатку.
3. Розроблено UI та ПЗ.
4. Додаток протестовано ручним наскрізним тестуванням.
5. Визначено шлях подальшого розвитку.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

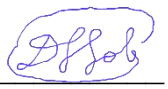
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДИЗАЙНУ ІНТЕР'ЄРУ ПРИМІЩЕНЬ

Програма та методика тестування

ДП.045480-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

 Дмитро НОВАК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Даніїл БОНДАРЧУК

2024

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Об'єктом випробувань є програмне забезпечення для дизайну інтер’єрів приміщень "Tumba" – інтуїтивного мобільного додатку, який допоможе візуалізувати користувацькі ідеї та створити ефективні інтер'єрні рішення.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів UI;
2) забезпечення належного рівня взаємодії з моделями в AR;
3) зручність роботи з мобільним додатком;
4) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні “системного тестування”.

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу;
4) тестування сценаріїв використання з метою оцінки зручності взаємодії користувача з інтерфейсом застосунку.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування застосунку "Tumba" виконується шляхом перевірки основних функцій і компонентів програми. Працездатність вебзастосунку перевіряється шляхом:

1) динамічного ручного тестування введенням граничної кількості моделей у простір AR;
2) динамічного моделювання функціональних можливостей через перевірку коректності відображення 3D об’єктів, зміни їх параметрів;
3) статичного тестування скриптів та коду;
4) тестування продуктивності;
5) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р.

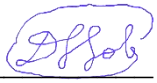
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДИЗАЙНУ ІНТЕР'ЄРУ ПРИМІЩЕНЬ

Керівництво користувача

ДП.045480-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

 Дмитро НОВАК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Даніїл БОНДАРЧУК

2025

ЗМІСТ

1. ОПИС СТРУКТУРИ МОБІЛЬНОГО ЗАСТОСУНКУ

Мобільний додаток "Tumba" складається з набору меню, які дозволяють користувачам візуалізувати інтер'єрні рішення у форматі доповненої реальності. Інтерфейс реалізовано українською мовою та адаптовано для мобільних пристроїв на платформі Android.

Для користувачів доступні головне меню, меню редагування, меню колекції та меню AR.

2. ПРОЦЕДУРА ПЕРЕГЛЯДУ КОЛЕКЦІЇ

Сторінка перегляду об’єкта колекції призначена для детального ознайомлення з характеристиками вибраного елемента та його 3D модель. У нижній частині сторінки розміщено назву об’єкта та кнопки для переходу до наступного або попереднього елемента колекції. Для повернення на попередню сторінку передбачено кнопку “Back”.

Рис. 1. Сторінка меню “Collection”

3. ПРОЦЕДУРА СОРТУВАННЯ МОДЕЛЕЙ ДЛЯ РЕДАГУВАННЯ

У верхній частині сторінки “Redactor” є секція фільтрації, що представлена випадаючим списком “Filter by item/rating”, яка дозволяє користувачу впорядковувати об’єкти за категоріями меблів або рейтингом. При виборі певної категорії у вікні відображаються лише ті об’єкти, які належать до цієї групи, що значно полегшує процес вибору об’єкта. Додатково, у фільтрі передбачена опція сортування за рейтингом, що дозволяє користувачу переглянути об’єкти з високими оцінками комфорту, ергономіки чи універсальності. Нижче розміщено текстове поле “Search by name”, яке виконує функцію миттєвого пошуку за назвою об’єкта. Після введення тексту система автоматично відображає відповідні моделі, які містять задану ключову фразу, чим додає гнучкості у пошуку.

Рис. 2. Проміжна сторінка меню “Redactor”

4. ПРОЦЕДУРА РЕДАГУВАННЯ МОДЕЛЕЙ

Сторінка редагування моделі забезпечує користувача можливістю налаштовувати зовнішній вигляд обраного об’єкта, змінюючи його кольори та текстури. Зазначено поточну назву текстури, під якою реалізовано блок вибору стилю, який складається з набору текстурних варіантів, представлених у вигляді кольорових квадратів. Користувач може обрати один із варіантів текстури, після чого зображення об’єкта миттєво оновлюється з урахуванням вибраного матеріалу. У центральній частині сторінки розміщено 3D-зображення об’єкта, що дозволяє користувачу оцінити застосовані зміни у реальному часі. Під зображенням знаходяться три слайдери, що відповідають за коригування кольорових компонентів об’єкта — червоного, зеленого та синього. Змінюючи положення повзунків, користувач може динамічно налаштовувати колірну палітру моделі, отримуючи миттєвий результат. Кнопка “Create design” зберігає внесені зміни та підтверджує новий стиль об’єкта. Кнопка “Back” повертає користувача до попереднього меню без збереження змін, забезпечуючи зручну та зрозумілу навігацію між розділами.

Рис. 3. Сторінка меню “Redactor”

5. ПРОЦЕДУРА РОБОТИ В AR

Сторінка AR-режиму призначена для розміщення об’єктів із колекції у реальному просторі за допомогою камери пристрою. У верхній частині екрана розташовано кнопку «Back», яка забезпечує швидке повернення до попереднього меню, а також випадаючий список, що дозволяє користувачу обирати категорію об’єктів для відображення. Основна частина екрана відведена під область перегляду камери, через яку користувач може спостерігати простір у режимі реального часу. У нижній частині екрана реалізовано панель вибору об’єктів, яка містить горизонтальний скролінг зі зображеннями меблів з колекції. Користувач може перегортати доступні об’єкти за допомогою кнопок навігації обираючи потрібний елемент для розміщення у просторі. Обраний об’єкт автоматично накладається на зону відображення, дозволяючи користувачу оцінити його розмір, пропорції та розташування в інтер’єрі. Модель можна пальцем пересувати по простору, якщо розтягнути модель то вона змінить свій розмір, при русі пальцем по екрану праворуч чи ліворуч модель відповідно обернеться, а при потрійному натисканні на неї вона видалиться з AR сцени.

Рис. 5. Сторінка меню “Create Design”