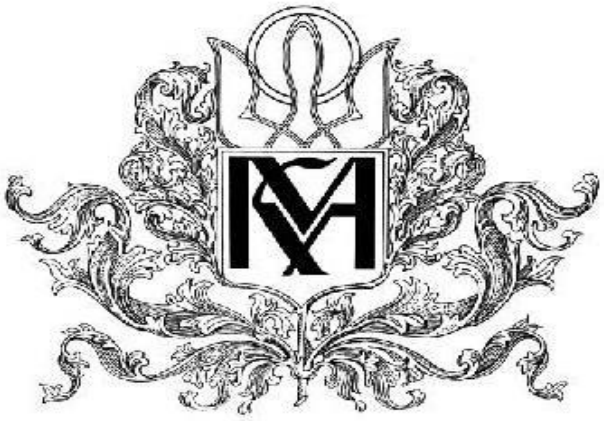
Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики



Дослідження методів та засобів реалізації спеціалізованої інформаційної системи
Текстова частина до курсової роботи за спеціальністю „ Комп'ютерні науки"

Керівник курсової роботи
к.ф.-м.н, ст. викладач Ющенко Ю. О
_____
(підпис)
"____" _____ 2021 р.

Виконала студентка
Настенко А. Е
"____" _____ 2021 р.

Київ 2021

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці Настенко Ангеліні Едуардівні

4 курсу факультету інформатики

ТЕМА: Дослідження методів та засобів реалізації спеціалізованої інформаційної системи

Індивідуальне завдання
Вступ
1. Аналіз методів розробки інформаційних систем
2. Аналіз основних засобів розробки інформаційних систем
3. Розробка спеціалізованої інформаційної системи
Висновок
Список літератури
Додатки (за необхідністю)

Дата видачі „___” _____ 2020 р. Керівник _____
(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проєкту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	05.10.2020	
2.	Огляд технічної літератури за темою роботи.	14.10.2020	
3.	Аналіз методів та засобів спеціалізованих інформаційних систем	07.11.2020	
4.	Розробка бази даних для інформаційної системи	26.12.2020	
5.	Порівняльний аналіз існуючих аналогічних додатків	14.01.2021	
6.	Програмування додатку	20.01.2021	
7.	Тестування розробленого додатку	25.02.2021	
8.	Написання пояснювальної роботи.	01.03.2021	
9.	Створення слайдів для доповіді та написання доповіді.	23.03.2021	
10.	Остаточне оформлення пояснювальної роботи та слайдів.	09.04.2021	
11.	Захист курсової роботи	21.04.2021	

Студент _____Настенко А.Е._____ _____

Керівник _____Ющенко Ю.О._____ _____

“_____” _____ _____ р.

Зміст

Анотація

У роботі розглядаються поняття, основні методи, етапи та засоби розробки інформаційної системи з прикладами. В рамках проєкту розроблена власна спеціалізована інформаційна система з використанням попередньо теоретично досліджених технічних засобів.

Вступ

Інформаційні системи сьогодення відіграють дуже важливу, або навіть незамінну роль у багатьох сферах життєдіяльності сучасної цивілізації починаючи від освітньої, фінансової, урядової та закінчуючи економічною. Остання, наприклад, передбачає потребу у використанні ІС в апараті такої державноважливої рушійної сили як галузь промисловості. Інформаційні системи стали основою більшості організацій. Банкова сфера не в змозі обробляти платежі, лікарні не можуть лікувати пацієнтів, уряд не може організувати податкову систему, магазини не могли б забезпечувати свої полиці без підтримки даної технологічної структури. Проте дане багатофункціональне рішення активно використовується не тільки в сферах такого рівня важливості для держав як вищезазначені, але й в сфері не менш важливої для людини – соціальної. Саме ця царина використання інформаційних технологій буде детальніше розглянута у даній роботі.

Актуальність розробки та досліджуваної теми безперечна. Щоденна праця, захоплення, спілкування, збір інформації та прийняття рішень – в усьому цьому людство все впевненіше покладається на комп’ютерну автоматизацію. Проблеми розробки ефективних систем обробки інформації актуальні вже деякий час. З самого початку створення подібних систем спеціалісти в області інформаційних технологій намагаються втілити з одного боку ефективні засоби інтеграції функціональних та структурних компонентів, а з іншого – прості способи керування кожним із них в окремості. Такі спеціалізовані інформаційні системи як соціальні мережі втілюють ту саму інтеграцію вищеназваних складових

приватного життя кожної людини, яких суспільство так потребує. Разом із розвитком технологій виникла можливість реалізувати свої потреби у соціалізації та самореалізації шляхом використання комп'ютерних технологій. Тоді почали з'являтися такі інформаційні технологічні конструкції як соціальні мережі – платформи, що використовуються для спілкування, знайомств, створення соціальних взаємовідносин між людьми за їх інтересами, а також для розваг та роботи. Без них нині неможливо уявити сучасне життя. На сьогодні існує безліч платформ, спеціалізованих для певного конкретного виду інтересів та діяльності – інформаційні системи для програмістів, веб-дизайнерів, геймерів, архітекторів, художників і тд.

Мета курсової роботи - дослідження методів та засобів реалізації спеціалізованої інформаційної системи, розгляд аналогічних систем, їх переваг та недоліків, та розробка власної аналогічної системи з врахуванням попередніх досліджень.

Текстова частина курсової роботи включає три розділи.

В першому розділі розкривається загальний механізм та спосіб побудови спеціалізованої інформаційної системи. В другому розділі постає аналіз методів та засобів побудови спеціалізованих інформаційних систем на конкретних прикладах. Третій розділ присвячений дослідженню засобів розробки ІС шляхом розроблення та демонстрації власної в рамках курсової роботи спеціалізованої інформаційної системи з описом обраних програмних засобів розробки.

РОЗДІЛ 1: Аналіз методів розробки інформаційних систем

Інформаційна система (ІС) – соціотехнічна організаційна система, призначена для збору, обробки, збереження і розповсюдження інформації. З соціотехнічної точки зору, інформаційні системи складаються з чотирьох компонентів:

- Задачі
- Люди
- Структури (або ролі)
- Технології

Інформаційні системи можна визначити як інтеграцію компонентів для збору, збереження та обробки даних, котрі використовуються для надання інформації, вкладу в знання, а також цифрових продуктів.

1.1 Стратегії розробки спеціалізованої інформаційної системи

Існують різноманітні методи розробки спеціалізованих комп'ютерних інформаційних систем. Структурний аналіз – найпопулярніший метод. Також використовується нова стратегія – об'єктно-орієнтований аналіз. Кожен з методів пропонує множину варіацій. Деякі організації займаються розробкою своїх власних підходів та переймають ті методи, що пропонуються постачальниками програмного забезпечення, інструментів CASE, або ж консультантами. Більшість експертів в області розробки згодні з твердженням, що єдиної найкращої стратегії розробки інформаційної системи не існує. Натомість, системний аналітик має розуміти альтернативні методології та їх сильні та слабкі сторони.

1.1.1 Сумісна й швидка розробка спеціалізованої інформаційної системи

В минулому IT-відділи іноді розробляли системи без достатньої участі користувачів, в результаті чого останні часто були незадоволені кінцевим результатом роботи спеціалістів. З часом, багато компаній визнали, що групи для розробки систем, що складаються з IT-персоналу, користувачів та менеджерів можуть швидше виконувати свою роботу та досягати кращих результатів. Популярними стали такі дві методології : сумісна розробка (JAD) та швидка розробка (RAD). Обидві використовують команди, що складаються з користувачів, менеджерів та IT-персоналу. Різниця полягає в тому, що JAD приділяє особливу увагу груповому пошуку факторів, що є тільки однією з фаз процесу розробки, тоді як RAD – більш зжата версія загального процесу. Схема роботи RAD проілюстрована на рис.1.1

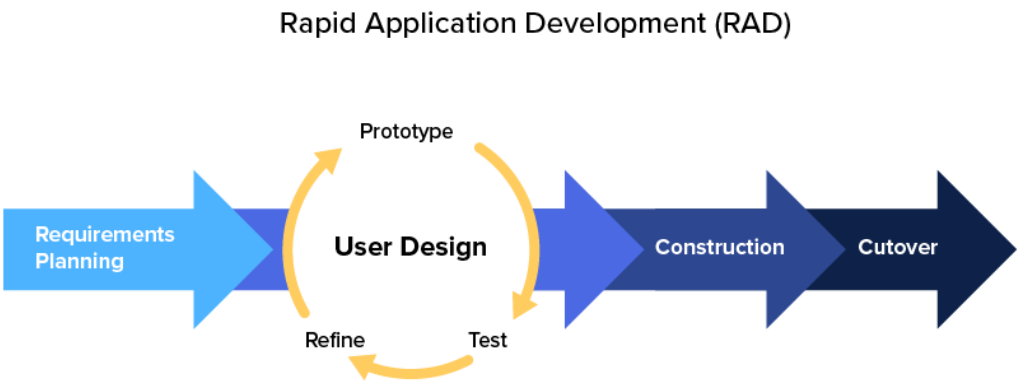


Рисунок. 1.1 Схема роботи RAD

1.1.2 Структурний аналіз спеціалізованої інформаційної системи

Структурний аналіз – традиційний метод розробки інформаційних систем. Структурний аналіз використовує серію етапів, що називаються циклом розробки системи (SDLC), з метою аналізу, планування, проєктування, впровадження та підтримки інформаційної системи.

Структурний аналіз використовує набір моделей процесів для графічного опису системи. Оскільки він фокусується на процесах, що трансформують дані в корисну інформацію, структурний аналіз називають методикою, орієнтованою на процес. В додаток до моделювання, процеси структурного аналізу також включають в себе структуру та організацію даних, проєктування реляційної бази даних та проблеми користувацького інтерфейсу.

Моделювання процесу визначає дані, які поступають в процес, бізнес-правила, за допомогою яких ці дані перетворюються та результуючий потік вихідних даних. Схема процесу SDLC проілюстрована на рис.1.2

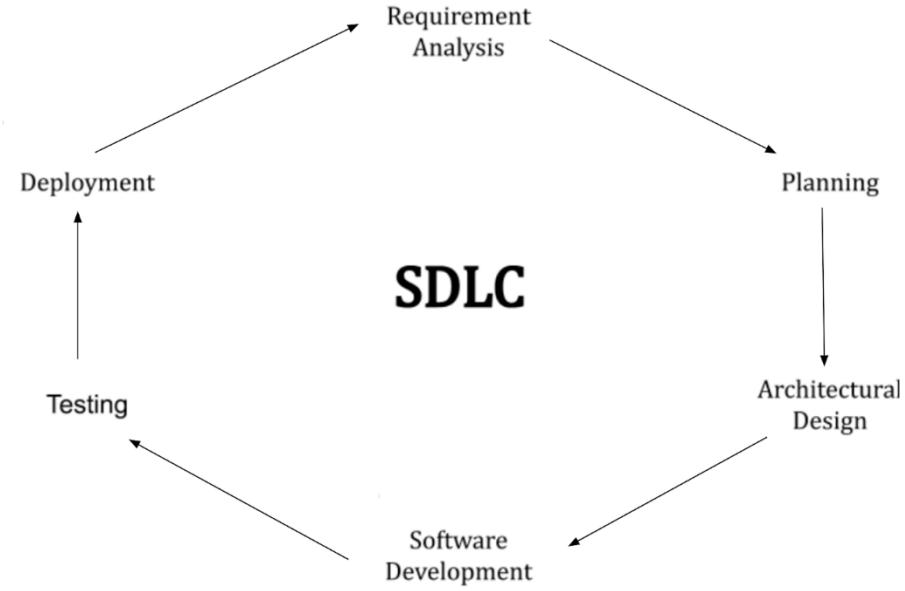


Рисунок 1.2 Схема циклу розробки SDLC

1.1.3 Об’єктно-орієнтований аналіз спеціалізованої інформаційної системи

В той час як структурний аналіз розглядає процеси й дані як окремі компоненти, об’єктно-орієнтований аналіз розглядає дані та процеси, що впливають на дані як структури, що називають об’єктами. Системний аналітик використовує дану методологію для моделювання реальних бізнес-процесів та операцій. Результатом є набір програмних об’єктів, котрі є представленнями реальних людей, речей, транзакцій, та подій. Далі, з використанням об’єктно-орієнтованої мови програмування, програміст пише код, що створює об’єкти.

Об’єкт є членом класу, котрий являє собою набір схожих об’єктів. Об’єкти володіють характеристикою, що називається властивостями, котрі об’єкти наслідують від свого класу або мають самі по собі.

1.1.4 Інші стратегії розробки

Окрім структурного та об’єктно-орієнтованого аналізу, існують також інші альтернативні методології. Наприклад, Microsoft пропонує підхід під назвою Microsoft Solution Framework (MSF), котрий документує досвід попередніх команд.

Використовуючи MSF, системні аналітики розроблюють серію моделей, включно з моделлю керування ризиками, моделлю команди та моделлю процесів. Кожна модель має конкретне призначення та результат, котрий вносить як свій вклад до розробки системи. Хоч процеси Microsoft різняться від орієнтованої на етапи SDLC, розробники MSF по суті виконують один й той же вид планування, задають одного й того ж типу запитання при встановленні фактів, вирішують одного типу проблеми при проєктуванні та реалізації. MSF використовує концепцію об'єктно-орієнтованого аналізу, втім також вивчає більш широкий бізнес- та організаційний контексти, що оточують розробку інформаційної системи.

1.2 Етапи розробки спеціалізованої інформаційної системи

Впровадження системи – це процес:

1) Визначення того, як має бути побудована інформаційна система (наприклад, фізична конструкція системи)
2) Забезпечення продуктивності та доцільності проєктованої інформаційної системи
3) Забезпечення відповідності інформаційної системи стандарту якості.

Розробка інформаційної системи (рис.1.3):

1) Концептуальний дизайн – визначення призначення інформаційної системи
2) Логічний дизайн – визначення того, якою має бути система безпосередньо для користувача
3) Фізичний дизайн – визначення того, як має бути побудована система

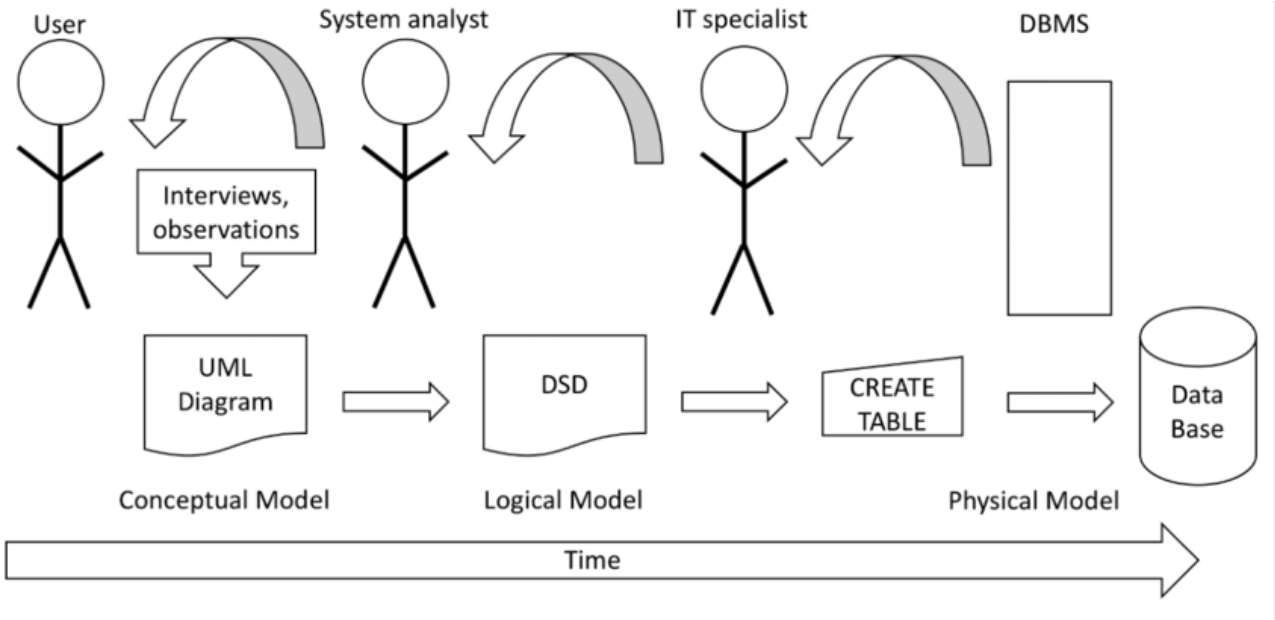


Рисунок 1.3 Проєктування інформаційної системи

1.2.1 Концептуальний дизайн спеціалізованої інформаційної системи

В концептуальному дизайні головна мета – оцінка можливості досягнення цілей управління інформаційною системою та аналізується загальна картина системи. Даний процес включає в себе наступні етапи:

1) Визначення проблеми

На першому етапі концептуального дизайну важливо сформувати чітке розуміння та визначення проблеми що підлягає вирішенню. Інформаційні потреби цільової групи користувачів, власне для яких створюється спеціалізована система, повинні бути ідентифіковані та зрозумілі на даному етапі, вони можуть бути визначені шляхом проведення опитувальної або консультаційної роботи з представниками цієї цільової аудиторії, що дозволить краще сформулювати розуміння мети, цілей та операційних планів спеціалізованої діяльності.

Формально, етап визначення проблеми можна розв’язати за допомогою ітеративного процесу. Оскільки інформаційні потреби – це власне проблеми, які мають бути вирішені за допомогою функціональності спеціалізованої ІС, то звідси випливає, що формулювання потреб достатньо для процесу проєктування. Для втілення даної мети потрібно слідувати наступним крокам:

- Постанова факту про потребу в інформації
- Задавання питань про поставлену потребу
- Пропонування інтерпретації даної потреби
- Деталізація початкового твердження
- Розгляд більш деталізованого опису потреби з керівництвом

Вищезазначені пункти повторюються до тих пір, поки потреба в інформації та проблема, висунута для вирішення, не стануть зрозумілими. Цей процес уточнення проблеми перетворюється в задачі системи.

2) Встановлення системних цілей

В більшості випадках доволі складно сформулювати цілі для систем, що охоплюють декілька функціональних областей. Цінність ІС полягає в корисності для її користувачів. Тож, не дивлячись на складність, конкретизація намірів системи необхідна, вони мають бути сформульовані в твердженнях, що саме зможуть робити користувачі після того, як будуть виконані інформаційні вимоги.

Таким чином перші кроки в проектуванні спеціалізованих систем намагаються дати відповіді на такі питання як: “Яка мета системи?”, “Навіщо вона потрібна?”, “Хто такі її користувачі, які їх цілі?”.

3) Встановлення обмежень для системи

При розгляді третього етапу в процесі встановлення обмежень легко побачити та зрозуміти ітераційний характер процесу проектування систем. Обмеження дають можливість проектувальникові з’ясовувати умови, за яких можуть бути досягнені цілі та враховувати кордони що обмежують проєкт. Обмеження можуть бути сприйняті як у негативній конотації по відношенню до проєктуванню системи, так і в позитивній, адже вони дозволяють забезпечити реалістичність дизайну.

Обмеження класифікуються як зовнішні та внутрішні.

- **Зовнішні обмеження**

Зовнішнє середовище організації орієнтоване та непокоється власне замовником. Тобто, різні системи виставлення рахунків, введення замовлень та інші системи, які мають враховувати потреби користувача. Якщо ж деякі вихідні результати

системи не задовольняють потребам клієнта з будь-яких причин, необхідно накласти обмеження.

Наприклад уряд може накладати деякі обмеження на обробку даних, що є необхідним для підтримання безпеки конкретних класів інформації у відповідності з законодавством та нормативними актами як варіант в області ведення бізнесу (наприклад, податки й звітність).

- **Внутрішні обмеження**

По перше, керівництвом, тобто групою, що виступає замовником спеціалізованої інформаційної системи має бути забезпечене сприятливе середовище для інформаційних систем та технологій, інакше дизайн та втілення системи матимуть серйозні обмеження.

По друге, організаційні міркування часто встановлюють обмеження на задачі й змінюють початковий підхід до проектування системи. Тобто корпораційна політика сама часто визначає або обмежує підхід до проектування систем.

По третє, доступність та потреба в персоналі нерідко є головною обмежувальною силою при проектуванні, розробці та використанні спеціалізованих інформаційних системи. Системні та комп'ютерні вміння та навички є одними з найважливіших нині. Тож, найзначнішим обмеженням з усіх є саме обмеження пов'язане з людьми.

Також, наступною причиною внутрішніх обмежень є вартість. При проєктуванні системи замовник повинен раціонально порівняти кількість замовлених для проєктування задач із отримуваної з них вигоди.

Обмеження, що накладаються самовільно формулюються від імені менеджера чи дизайнера. Менеджер також обмежує зашальну кількість часу та зусиль, які мають бути витрачені на дослідження проєкту. В процесі розрахунку витрат на досягнення кінцевої мети менеджеру може знадобитися зменшити кількість вимог, щоб система відповідала решті виходам, обмеженням чи обладнанню.

4) Визначення інформаційних потреб та інформаційних джерел

Для вірного та грамотного проєктування спеціалізованої ІС дуже важливо мати чіткий виклад інформаційних потреб користувачів. На жаль, часто багато ресурсів замовниками витрачаються на обладнання та підтримку вже існуючих

систем чи створення складних банків даних при цьому попередньо не визначивши реальні інформаційні потреби користувачів. Оптимальні результати не можуть бути досягнуті у випадку якщо менеджери не надають специфікації того, що саме вони очікують від функцій інформаційної системи. Джерела інформації важливі для визначення інформаційних потреб, системі може знадобитися зовнішня або внутрішня інформація.

5) Альтернативні концептуальні дизайни й обрання з них одного

Розробка концептуального дизайну спеціалізованої ІС представляє собою творчий процес, котрий є нічим іншим як комбінуванням знань в один певний зразок. Концепція є ескізом, скелетом інформаційної системи, який складається з моделей інформаційних потоків, джерел, каналів інформації, ролей користувачів, інформаційних точок входів та виходів. Якщо концептуальний дизайн – це скелет, то детальний дизайн – плоть.

Для ілюстрації даного етапу дизайну, можна навести такий приклад : дві групи студентів отримали завдання розробити проєкт туристичного путівника та системи контактної інформації. Перша зі створених концепцій – це нарис, що детально описує конкретні місцевості з описом їх культури, спадщини, та коледжів, готелей та торгівлі. В той час як друга концепція являє собою нарис опису коледжів разом із їх факультетами, а також структур сплати для різних потреб. Очевидно, що обидві версії системи мають свої переваги та недоліки. Проте іноді якась з концепції буде перевершувати інші за основними критеріями.

6) Документування найкращого дизайну

Після зібрання на попередніх етапах достатньої кількості потрібної інформації, починається останній етап концептуального проєктування – більш детального опису концепції спеціалізованої інформаційної системи. Даний опис може включати в себе блоксхему, або деяку іншу документацію потоку інформації через систему, входів та виходів.

На цьому етапі під час створення документації, менеджер має переконатися та слідкувати, щоб проєктована система забезпечувала необхідну інформацію. Задача дизайнера – цікавитися природою матеріалів та обладнання в якості міркувань технічної обробки.

Таким чином, усі деталі, що пізніше мають бути опрацьовані безпосередньо розробником матимуть чіткі інструкції відносно того які дані й коли мають бути охоплені, які файли підлягають використанню, деталі того, як саме має відбуватися обробка, які вихідні дані мають генеруватися системою і т.д.

1.2.2 Логічне проєктування спеціалізованої інформаційної системи

Другий етап проєктування спеціалізованих інформаційних систем отримав назву логічного проєктування. Безпосередня мета етапу – створення логічної моделі для досліджуваної предметної області. Суть процесу – створена на попередньому етапі концептуальна модель перетворюється з уточненням в деяку логічну модель даних. Логічна модель даних надає змогу врахувати особливості обраної моделі організації даних в цільовій системі управління базами даних, наприклад в реляційній. Тоді як концептуальна модель не має прямої залежності від фізичних аспектів, логічна – модель даних, в якій на основі обраної моделі створюється цільова система керування базами даних (СКБД).

На даному етапі має бути відомо яка саме система керування базами даних буде цільовою. В якості цільової бази даних може бути обрана як об’єктно-орієнтована система, так і ієрархічна, мережева або реляційна СКБД.

Під час розробки логічної моделі, розробники виконують постійне тестування та перевірку виконаної роботи на відповідність вимогам користувачів. З метою перевірки вірності збудованої логічної моделі даних, розробниками застосовується метод нормалізації, що дозволяє отримати підтвердження того, що відношення, які були введені в модель даних не мають надлишковості даних, та логічна модель підтримує усі необхідні для користувачів спеціалізованої системи транзакції.

Логічна модель даних є для розробника джерелом інформації, необхідним на етапі фізичного проєктування ІС. За її допомогою розробник досягає поставлених перед ним цілей, що є дуже важливим для ефективного проєктування.

В процесі експлуатації та супроводу інформаційної системи логічна модель даних відіграє дуже важливу роль. Саме тому необхідно уділяти багато уваги організаційному супроводу, котрий має підтримуватися в актуальному стані.

1.2.3 Фізичне проєктування спеціалізованої інформаційної системи

Головна мета етапу фізичного проєктування – визначити, як саме буде реалізована логічна структура бази даних. Для реляційної бази даних цей процес включає:

- Визначення структурного набору таблиць, типів даних, полів, та обмежень для цих таблиць, таких як первинний та зовнішній ключі, унікальні поля, значення not null, а також перевірка чи не виходить інформація за межі визначених границь.
- Ідентифікація конкретних структур сховища та методів доступу з метою ефективного витягу даних.
- Розробка функцій захисту для системи баз даних включно зі створенням облікового запису, захисту несанкціонованого доступу, та встановленням рівня безпеки.

Фізичний дизайн специфічний для СКБД, в той час як логічний, навпаки, не залежить від конкретної СКБД. Фізичний дизайн – процес впровадження БД у вторинному сховищі з конкретною СКБД.

РОЗДІЛ 2: Аналіз основних засобів розробки інформаційних систем

CASE (англ. computer-aided software engineering) – сукупність методів та засобів проєктування інформаційних систем з використанням CASE-засобів, що застосовуються системними аналітиками для полегшення процесу розробки та обслуговування інформаційних систем. Інструменти CASE забезпечують загальну основу для розробки систем та підтримують широкий спектр методологій проєктування, включно зі структурним та об’єктно-орієнтованим аналізом.

Засоби для автоматизації розробки систем (CASE-засоби) – інструменти для автоматизації процесів проєктування та розробки програмного забезпечення,

призначені для системного аналітика, розробника програмного забезпечення та програміста. З першу, CASE-засоби виконували функції лише для спрощення найбільш трудомістких процесів аналізу та проєктування, проте згодом, з призодом стандарту ISO/IEC 14102 CASE-засоби почали ідентифікувати як програмні засоби для підтримки процесів життєвого циклу ПЗ.

Інструменти CASE забезпечують підвищення продуктивності та якості інформаційних систем за рахунок спрощення процесу розробки. Окрім традиційних інструментів CASE, розробники також часто використовують інструменти управління проєктами, такі як Microsoft Project та спеціальні інструменти, призначені для побудови діаграм накшталт Microsoft Visio. Системний аналітик може використовувати Visio з метою побудови різних типів діаграм, включно із блок-схемами.

Компоненти CASE-засобів:

Інструменти CASE можуть бути умовно розподілені на наступні частини, в залежності від їх використання на конкретному етапі SDLC:

- **Центральний репозиторій**

CASE-засоби потребують певного центрального репозиторію, котрий буде слугувати в якості джерела спільної, інтегрованої та узгодженої інформації. Центральний репозиторій – сховище, де зберігаються специфікації продукта, документи з вимогами, відповідні звіти та діаграми, та інша корисна інформація.

- **CASE-засоби “верхнього рівня” (Upper CASE Tools)** – інструменти, що використовуються на етапах планування, аналізу та проєктування SDLC.

В основному застосовуються при аналізі та документації потреб користувачів. Підходять, перш за все, для візуалізації, для створення різних схем та генерування документації. Підтримують використання традиційних мов діаграм (діаграми сутність-зв'язок, моделей даних, UML-схеми і т.д)

- **CASE-засоби “нижнього рівня” (Lower CASE Tools)** – інструменти, що використовуються при впровадженні, тестуванні та обслуговуванні.

Підтримують генерування структури бази даних, генерування коду, проведення тестування, керування версіями коду, керування конфігурацією, реверсивне проєктування і т.д.

- **Інтегровані CASE-засоби** (рис.2.1)

Інтегровані CASE-засоби є корисними на всіх етапах SDLC, від збору вимог і до тестування та документації.

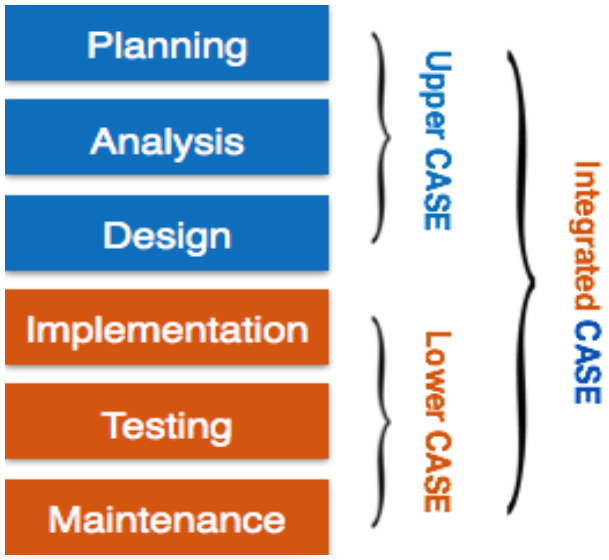


Рисунок 2.1 Компоненти інтегрованих CASE-засобів

CASE-засоби можуть бути згруповані разом, якщо вони мають схожі функції, процеси та можливість інтеграції з іншими інструментами.

Типи CASE-засобів:

- **Інструменти діаграм**

Дані інструменти використовуються для представлення компонентів системи, даних, взаємодії між різними програмними компонентами та структури системи в графічній формі. Наприклад, інструмент Flow Chart Maker для створення блок-схем, Cacoo, ConceptDraw Diagram, Creately, Edge Diagrammer, Edraw Max, Gliffy, Lucidchart, SmartDraw.

- **Приклад Cacoo :**

  Комплексний інструмент для побудови, використання та публікацій діаграм з функцією групової роботи.

  Переваги:

- Гарне хмарне підключення
- Є можливість підключення до Slack
- Є можливість групової роботи
- Доступність на 18 мовах
- Невелика ціна

Недоліки:

- Обмежена кількість шаблонів

Візуалізація інтерфейсу інструменту (рис 2.2):

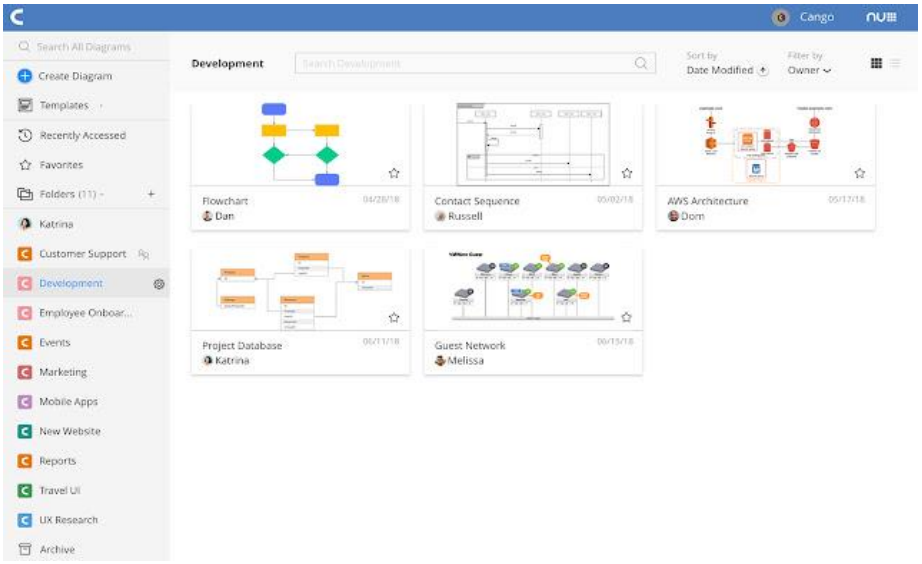


Рисунок 2.2

- **Інструменти для моделювання процесів**

Моделювання процесу – метод створення моделі програмного процесу, котрий створюється для розробки інформаційної системи. Дані інструменти дозволяють менеджерам обирати модель процесу або змінювати її у відповідності з вимогами програмного продукту. Наприклад, EPF Composer, IBM Method Composer.

- **Приклад EPF Composer :**

  Відкрите програмне забезпечення, на якому перевіряється новий функціонал. Містить функціонал створення процесу та публікації.

  Переваги:

  - Поставляється разом із OpenUP, який може бути розширений та налаштований в залежності від потреб організації
  - Підходить для малих проєктних груп

Недоліки:

- Відсутність інтеграції з іншими інструментами IBM
- Відсутність можливості міграції із Rational Process Workbench

Візуалізація інтерфейсу інструменту (рис 2.3):

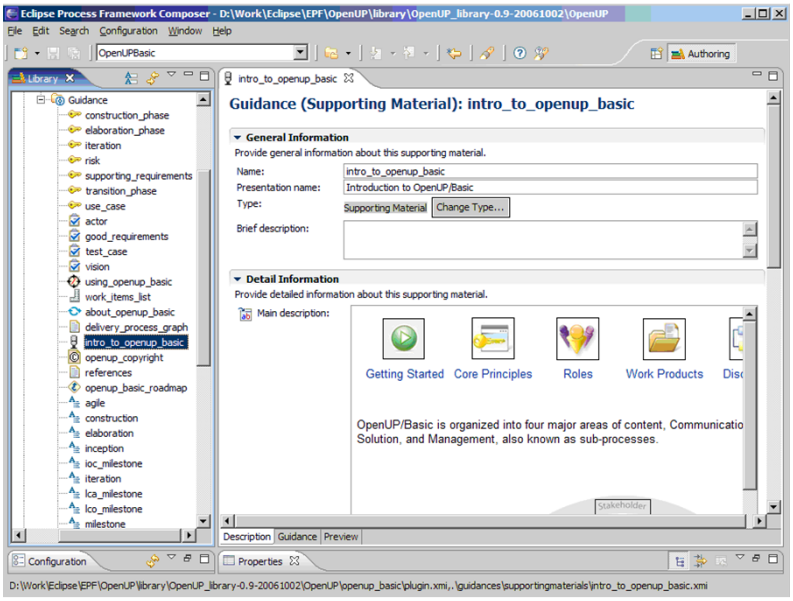


Рисунок 2.3

- **Інструменти для керування проєктами**

Дані інструменти використовуються для планування проєкту, оцінки витрат та зусиль, планування ресурсів та розкладу. Інструменти керування проєктами дають змогу зберігати та обмінюватися інформацією про проєкт в режимі реального часу у рамках всієї організації. Наприклад, Creative Pro Office, Trac Project, Basecamp, Springloops, No Kahuna, Pbwiki, 16bugs, JIRA.

- **Приклад Basecamp :**

  Інструмент для керування проєктами, сумісної роботи та постановки задач по проєктам.

  Переваги:

  - Зручний для використання у невеликих компаніях
  - Сумісний із багатьма додатками, віджетами та іншими програмами
  - На офіційному сайті є в доступі платні та безкоштовні доповнення
  - Є можливість самостійно створювати доповнення для Basecamp використовуючи API в стилі REST.

  Недоліки:

- Недостатньо пристосований для ведення довгострокових та складних проєктів
- Незручний для використання в великих компаніях
- Дорогий
- Неможливість доступу до системи під час виникнення несправностей із серверами компанії

Візуалізація інтерфейсу інструменту (рис 2.4):

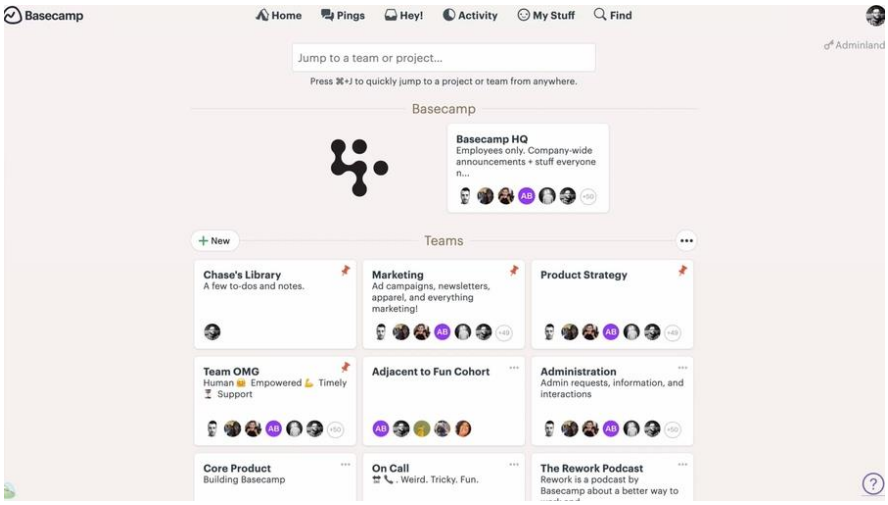


Рисунок 2.4

- **Інструменти для документації**

Документація в програмному проєкті починається до програмного процесу, відбувається впродовж усіх етапів SDLC та після завершення проєктів.

Дані інструменти призначені створювати документацію для технічних та кінцевих користувачів. Технічні користувачі – це, в основному, штатні спеціалісти групи розробників, котрі звертаються до системного, довідкового, навчального посібників та посібників з інсталяції і т.д. Документи ж кінцевого користувача дають опис функціонування системи та інструкцій використання, інструкція користувача. Наприклад, Doxygen, DrExplain, StepShot, Dozuki, Adobe RoboHelp для документації.

- **Приклад DrExplain :**

  Інструмент, призначений для швидкого створення файлів довідки, посібників користувача, довідкових систем, посібників та документацій до програмного забезпечення, виробам, технічним та бізнес-системам.

Переваги:

- Програма здатна аналізувати користувацький інтерфейс та створювати скріншоти вікон, автоматично роблячи пояснювальні виноски для елементів інтерфейсу
- Інтегрована утиліта аналізу та захвату програмних екранів та вікон
- Моментальний перегляд результату в друкованому та web-форматах
- Адаптивність під мобільні пристрої web-форматів
- Автоматизація рутинних операцій
- Є безкоштовна демоверсія
- Лаконічний та зручний інтерфейс
- Підтримка сумісної роботи над проєктом для багатьох користувачів

Недоліки:

- Програма платна

Візуалізація інтерфейсу інструменту (рис 2.5):

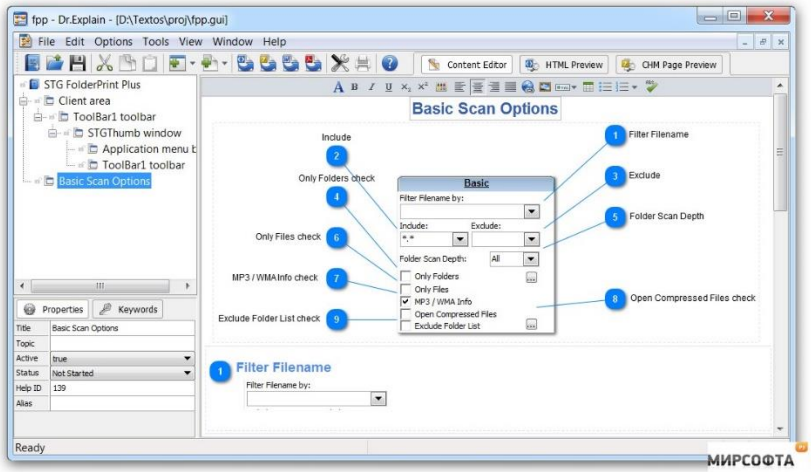


Рисунок 2.5

- **Інструменти для аналізу**

Дані інструменти допомагають збирати вимоги, автоматично перевіряють будь-які невідповідності, неточності в діаграмах, надлишковість даних або помилкові упущення. Наприклад, Accept 360, Accompa, Jama, Rational DOORS, CaseComplete для аналізу вимог, Visible Analyst для загального аналізу.

- **Приклад Accompa :**

Інструмент, що є повністю орієнтований на роботу у хмарі, дає можливість співробітникам співпрацювати в режимі реального часу з метою простого та продуктивного керування вимогами.

Переваги:

- Збереження та керування вимогами в центральному репозиторії
- Механізм налаштування під потреби користувача
- Автоматизація збору вимог
- Функція автоматичного відстеження змін та залежностей
- Можливість сумісної роботи та сумісного використання вимог в режимі реального часу
- Створення документів з вимогами за допомогою інтуїтивно-зрозумілих інструментаріїв
- Широкий набір вбудованих інтеграцій (Jira, FogBugz, HP Quality Center, IBM Rational, IBM Rational ClearQuest, Bugzilla, trac, VersionOne, Rally, Agilefant, Seapine Software, SmartBear та VisualStudio. Сервер Team Foundation)

Недоліки:

- Невелика кількість функціоналу у порівнянні з аналогами
- Програма платна

Візуалізація інтерфейсу інструменту (рис 2.6):

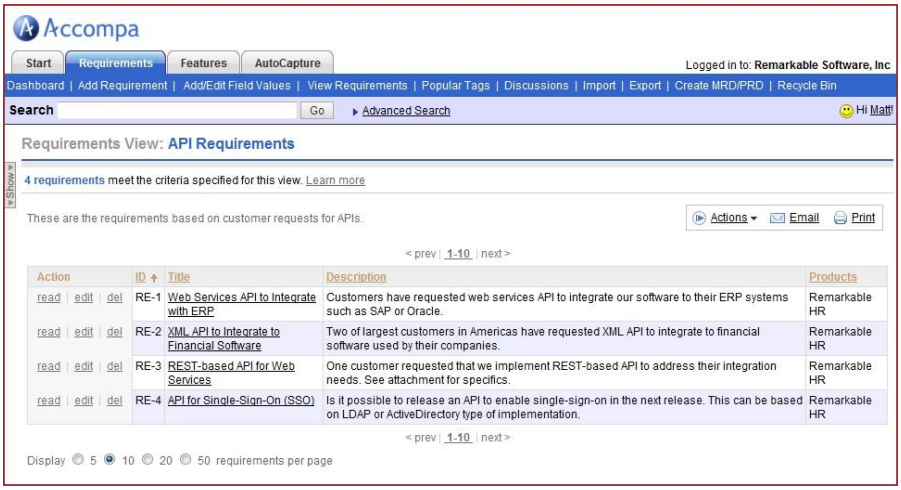


Рисунок 2.6

- **Інструменти для дизайну**

Дані інструменти допомагають розробити блокову структуру для ПЗ, котра може бути розбита на менші модулі. Ці інструменти забезпечують деталізацію кожного модуля та взаємозв'язок між ними.

- **Інструменти керування конфігурацією**

Екземпляр ПЗ випускається під однією версією. Інструменти керування конфігурацією мають справу з –

- Керуванням версіями
- Керування базовою конфігурацією
- Керування змінами

CASE-засоби допомагають в цьому за рахунок автоматичного відстеження, керування версіями та випусками. Наприклад, Fossil, Git, Mercurial, CVS, SVN, Accu REV.

- **Приклад Git :**

  Інструмент, що являє собою розподілену систему контроля версіями.

  Переваги:

  - Високий рівень продуктивності
  - Наявні розвинені засоби інтеграції з іншими VCS (в тому числі з CVS та SVN)
  - Продумана система команд, яка надає змогу зручно вбудовувати git скрипти
  - Для роздачі репозиторія мережею достатньо будь-якого веб-серверу

  Недоліки:

  - Система не вміє відстежувати пусті каталоги
  - Великі часові затрати в порівнянні із файл-орієнтованими системами на формування історії конкретного файлу, історії правок певного конкретного користувача, пошуку змін, які мають відношення до заданого місця конкретного файлу
  - Деякі команди працюють неочікувано

  Візуалізація інтерфейсу інструменту (рис 2.7):

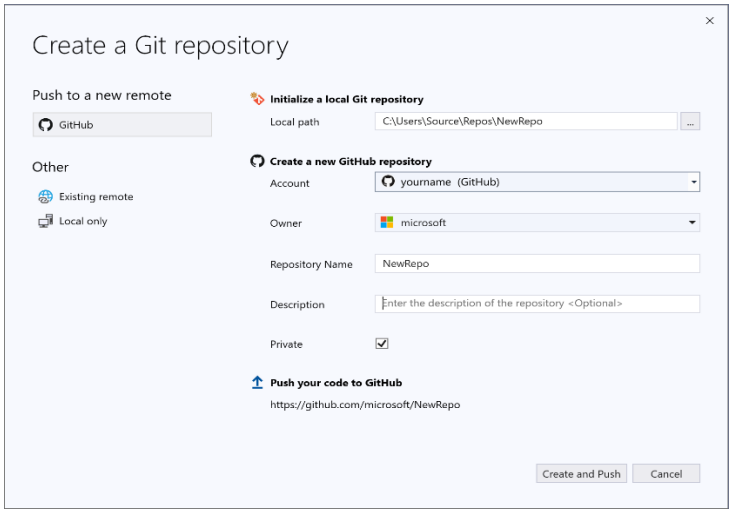


Рисунок 2.7 Інтерфейс Git у VisualStudio

- **Інструменти керування змінами**

Дані інструменти вважаються частиною інструментів конфігураційного керування. Мають справу зі змінами, що були внесені до розроблюваної інформаційної системи після виправлення його базової версії або ж при першому релізі. Засоби CASE автоматизують відстеження змін, керування файлами, кодом і т.д.

- **Інструменти для програмування**

Дані інструменти складаються із програмних середовищ, таких як IDE (інтегроване середовище розробки), вбудованої бібліотеки модулів та інструментів моделювання. Перераховані інструменти надають змогу створення програмного продукту, коду та включають в себе функції для моделювання та тестування. Наприклад, Eclipse, Visual Studio, WebStorm, Atom.

- **Приклад WebStorm :**

  Інструмент, що є комерційним середовищем розробки для JavaScript, CSS та HTML, розроблений на основі JetBrains.

  Переваги:

  - Гарна валідація коду на помилки, підказки по реорганізації коду, розстановка відступів
  - Зручний та розумний редактор JavaScript, HTML, CSS
  - Підтримка безлічі інших мов програмування (наприклад, TypeScript, CoffeeScript, Dart, Less, Sass та Stylus й фреймворки, наприклад, Angular, React та Meteor)

- Автодоповнення коду, швидка навігація та рефакторинг
- Має сильні інструменти інтеграції з іншими системами керування версій (Git, GitHub, Subversion, Perforce, Mercurial, CVS)
- Влаштовані інструменти тестування
- Ефективна розробка проєктів на Node.js

Недоліки:

- Займає багато пам’яті
- Програма платна

Візуалізація інтерфейсу інструменту (рис 2.8):

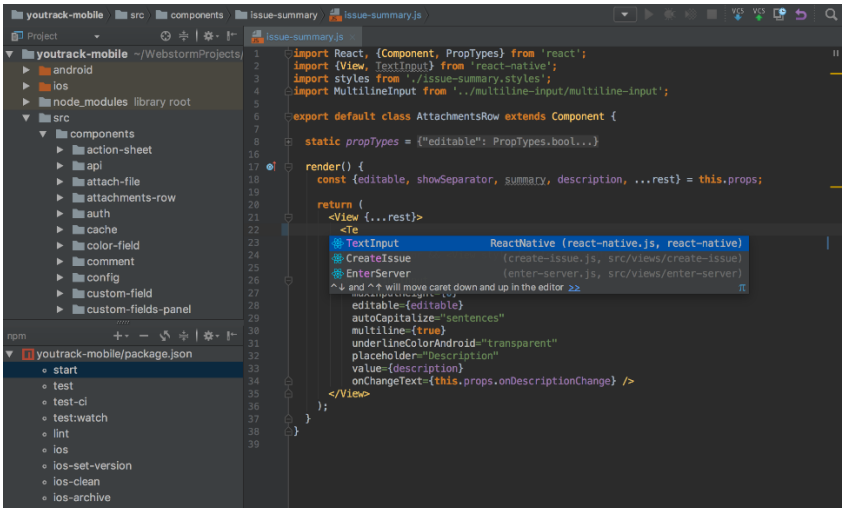


Рисунок 2.8

- **Інструменти Веб-розробки**

Мета інструментарію – створення безпосередньо веб-сторінок з усіма необхідними елементами накшталт форм, тексту, сценарію, графіки та ін. Веб-інструменти також забезпечують попередній перегляд роботи та її вигляду після закінчення. Приклади інструментів - Fontello, Adobe Edge Inspect, Foundation 3, Flaticon, Brackets.

- **Приклад Brackets :**

  Текстовий редактор з відкритим вихідним кодом для веб-розробників, орієнтований на роботу з HTML, CSS та javaScript.

  Переваги:

  - Інструмент написаний на JavaScript, HTML, CSS, що надає можливість дописувати й покращувати початковий код програми та писати для неї розширення

- Наявність функції life preview, яка надає змогу в реальному часі бачити всі зміни без попереднього збереження
- Підтримка програми на більшості платформах таких як Windows, Linux і т.д.
- Наявна вбудована функція, що дозволяє при наведенні курсора на HEX та RGB показувати їх візуально в спливаючій підказці в будь-який підтримуваних документах. Аналогічно функція працює з градієнтами та зображеннями
- При вказуванні шляхів до файлів редактор підказує не тільки шлях до них, але й імена файлів

Недоліки:

- Інтерактивний перегляд працює лише з браузером Google Chrome
- Багато некоректно працюючих розширень, тому встановлювати безпечно лише перевірені, яких не дуже багато

Візуалізація інтерфейсу інструменту (рис 2.9):

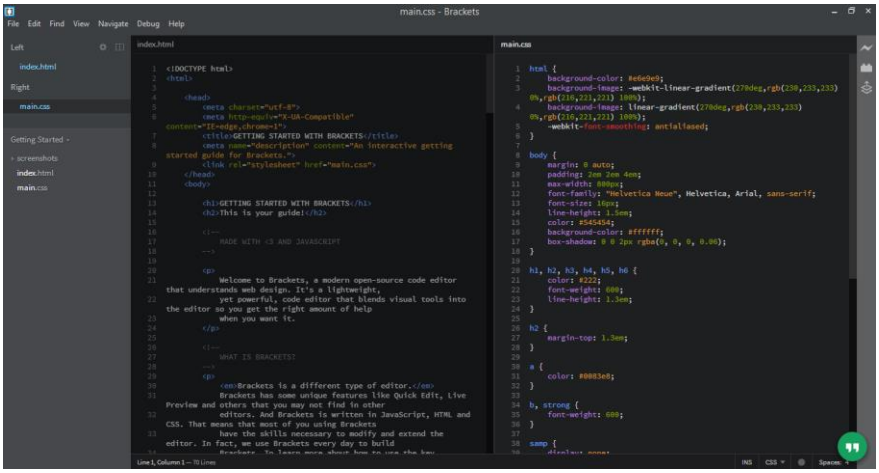


Рисунок 2.9

- **Інструменти для створення прототипів**

Програмний прототип – деяка змодельована версія ймовірного програмного продукту. Ціль прототипу – забезпечення попереднього зовнішнього вигляду продукту та імітування деяких аспектів реального продукту.

Інструменти CASE для створення прототипів поставляються з графічними бібліотеками. Вони націлені на створення апаратно-незалежних користувацьких інтерфейсів та дизайну та допомагають користувачам швидко створювати прототипи на основі вже отриманої інформації. Наприклад, Serena prototype

composer, Mockup Builder, Moqups, Mockplus, Balsamiq, Mockflow, Hotgloo, Minjamock, Invision, Uxpin, Proto.Io, Fluid UI, Axure RP, Pidoco, Justinmind.

- **Приклад Mockup Builder :**

  Інструмент для створення прототипів, макетів та каркасів, котрий надає можливість легко ділитися своєю роботою з клієнтами та колегами. Надає можливість створення прототипів для мобільних та веб-додатків.

  Переваги:

  - Різноманітна бібліотека елементів дизайну
  - Спільне використання проєктів, експорт в PNG та PDF
  - Безкоштовні каркасні шаблони
  - Зрозумілий інтерфейс

  Недоліки:

  - Інструмент є платним

  Візуалізація інтерфейсу інструменту (рис 2.10):

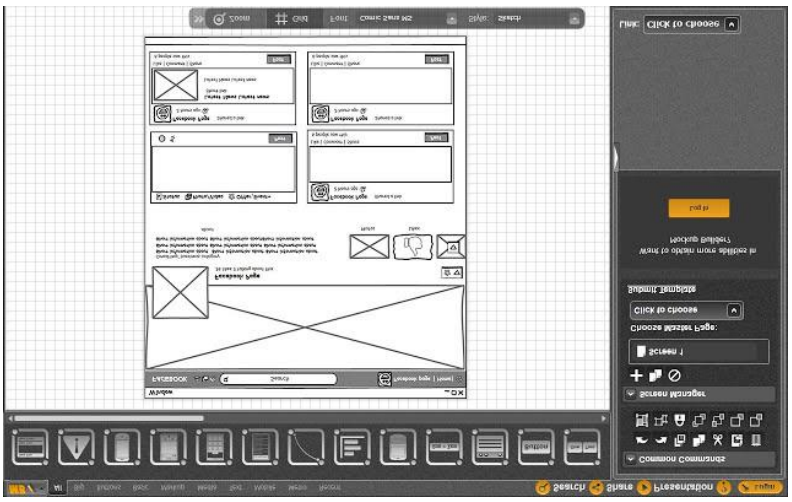


Рисунок 2.10

- **Інструменти для обслуговування**

Супровід інформаційної системи включає в себе модифікацію програмного продукту після його поставки. Основний функціонал - методи автоматичного ведення журналів та звітів про помилки, генерування інформації про помилки та аналіз першопричин. Наприклад, Bugzilla для відстеження дефектів, HP Quality Center.

- **Приклад HP Quality Center :**

Інтегрована система для забезпечення управління процесами контролю якості на всіх етапах розробки програмного забезпечення. HP Quality містить п’ять модулів, що тісно інтегровані між собою та забезпечують неперервність процесу тестування : Management, Requirements management, Test Plan, Test Lab, Defects management.

Переваги:

- Функціонал визначення та керування вимогами
- Прискорення ручного тестування
- Робота з планами тестування
- Моніторинг якості релізів та циклів
- Інструментарій графіків та робочих випробувань
- Функціонал відстеження дефектів
- Функція централізованого створення звітів
- Автоматизація гнучких методик розробки

Недоліки:

- Інструмент є платним

Візуалізація інтерфейсу інструменту (рис 2.11):

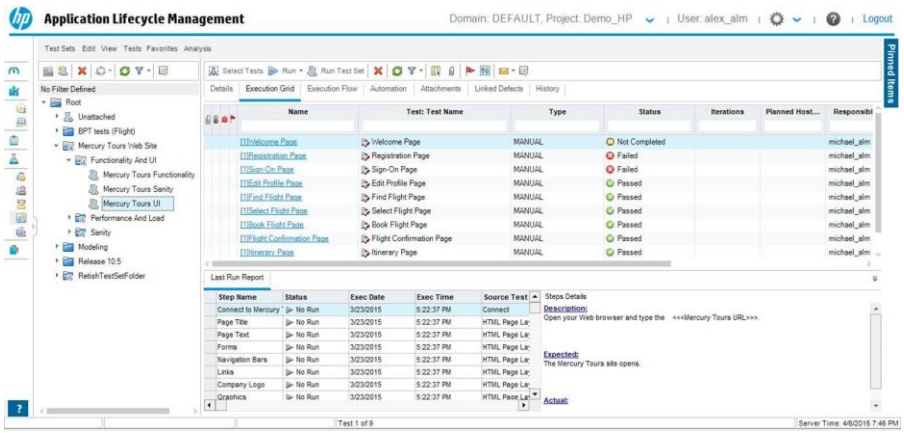


Рисунок 2.11

- **Інструменти забезпечення якості**

Інструменти відповідають за моніторинг процесу розробки та методів, що застосовуються для створення продукту з метою забезпечення відповідності якості організаційним стандартам. Інструменти контролю якості включають у себе інструменти налаштування та контролю змін, та інструментарій тестування. Наприклад, SoapTest, AppsWatch, JMeter.

- **Приклад HP Quality Center :**

  Інструмент надає можливість моніторингу усіх необхідних користувачеві додатків з будь-яких місцезнаходжень. Забезпечує отримання скріншотів про збої в роботі, визначення першопричин помилок та ліквідування проблем завчасно, перегляд тенденцій поведінки додатків з плином часу.

  Переваги:

  - Наявність інструментів діагностики
  - Повна діагностика транзакцій
  - Інструменти аналізу продуктивності
  - Наявність механізму керування ресурсами
  - Аналіз причин
  - Відстеження окремих транзакцій
  - Моніторинг серверу
  - Простий у використанні

  Недоліки:

  - Інструмент є платним
  - Інколи виникають збої автоматизації й доводиться використовувати ручне керування

  Візуалізація інтерфейсу інструменту (рис 2.12):

Рисунок 2.12

- **Інструменти СКБД**

Даний інструмент являє собою певну сукупність програмних та лінгвістичних засобів загального або спеціального призначення, які слугують з метою

забезпечення функціоналу створення, керування та використання баз даних. Оскільки бази даних – невід'ємна та основна частина складу спеціалізованих інформаційних систем, даний інструмент є головним серед використовуваних в процесі розробки системи. Тобто, СКБД, або система керування базами даних – комплекс програм, які надають можливість створювати БД (базу даних), й маніпулювати даними, себто додавати, оновлювати, видаляти та отримувати. Система надає можливість адміністрування БД та забезпечує безпечність, надійність збереження та цілісність даних.

Залежно від моделі даних, СКБД типізуються як ієрархічні, мережеві, реляційні, об'єктно-орієнтовані та об'єктно-реляційні. За ступенем розподіленості – локальні (усі частини СКБД розміщені на одному комп'ютері), та розподілені (частини СКБД можуть бути розміщені на двох та більше комп'ютерах). За способом доступу до БД, СКБД поділяються на файл-серверні, клієнт-серверні та вбудовувані.

Наприклад, Oracle Database, Firebird, Interbase, IBM DB2, Informix, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MySQL, Microsoft Access, Paradox, dBase, SQLite, BerkeleyDB, Firebird Embedded, Microsoft SQL Server Compact.

- **Приклад MySQL :**

  Реляційна система керування базами даних. Є гарним рішенням для невеликих та середніх додатків. Зазвичай використовується в якості сервера, до якого відбувається підключення локальних або віддалених клієнтів, проте до дистрибутиву також входить бібліотека внутрішнього серверу, що дозволяє включати MySQL до програм автономних.

  Переваги:

  - Простота використання. MySQL легко інсталюється, в той час як множина плагінів та допоміжних програм спрощують роботу з базами даних (наприклад наявність графічного користувацького інтерфейсу MySQL Workbench).
  - Широкий функціонал. Система володіє усім необхідним для реалізації БД функціоналом.

- Безпечність. Структура системи передбачає роботу більшості вбудованих функцій безпеки за замовчуванням.
- Масштабованість. Система є достатньо універсальною та працює однаково ефективно з даними як малих так и великих об'ємів.
- Швидкість. За рахунок спрощення деяких використовуваних стандартів забезпечується висока продуктивність.

Недоліки:

- Порівняно з іншими СКБД недостатньо надійна.

Візуалізація інтерфейсу інструменту MySQL Workbench (рис 2.13):

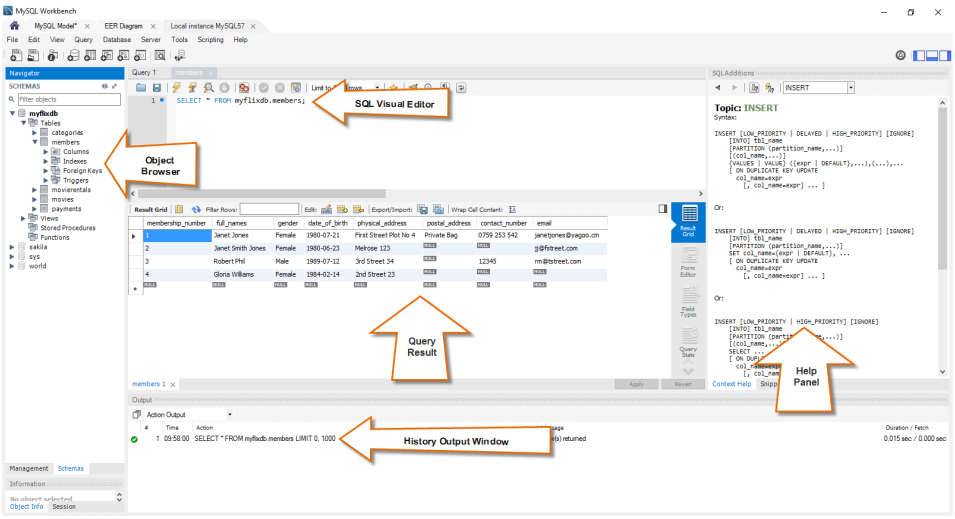


Рисунок 2.13 MySQL Workbench

РОЗДІЛ 3: Розробка спеціалізованої ІС

3.1 Технічне завдання

Головна мета розробки – створення спеціалізованої інформаційної системи (або прототипу соціальної мережі) для художників початківців. Основний функціонал системи забезпечуватиме користувачам можливості ділитися своїми художніми витворами або переглядати роботи інших користувачів, відмічати або коментувати картини що сподобалися, підписуватися на користувачів, виставляти свої роботи на продаж або аукціон, або ж придбати чиюсь роботу напряму чи через участь в аукціоні. Для отримання повного доступу до можливостей системи передбачена необхідність авторизації, або ж створення нового облікового запису якщо користувач його не має. Зберігатися вся інформація системи має у базі даних. Інтерфейс системи має бути максимально

простим та приємним користувачеві. В процесі створення спеціалізованої системи передбачається дослідження засобів розробки шляхом їх використання та інтеграції в проєкті.

3.2 Обґрунтування вибору засобів розробки

Для розробки back-end частини було обрано мову програмування C# та інтегроване середовище розробки Microsoft Visual Studio. Вибір зупинено саме на даній мові програмування через її беззаперечні переваги – C# простий та водночас дуже потужний інструмент розробки, реалізовує об’єктно-орієнтовано парадигму програмування, відноситься до типу мов що компілюються й володіє усіма здобутками таких мов. C# є інтеграцією усіх найкращих ідей, приналежних таким сучасним мовам програмування як Java та C++ та через значну кількісну різноманітність синтаксичних конструкцій та можливістю роботи з .NET фреймворком дозволяє підвищити швидкість розробки більше за будь-які інші мови. Вирішальним моментом у виборі стала ідея використання платформи ASP.NET для розробки інформаційної системи. Даний інструментарій дозволяє зручно та продуктивно розробляти веб-сервіси, веб-додатки та веб-сайти будь-якого масштабу. З її допомогою реалізується бізнес-логіка та доступ до даних в інформаційній системі.

Середовище Visual Studio було обране з наступних міркувань: оскільки дане середовище, як і обрана мова програмування C# створені Microsoft, то максимально функціональне рішення для розробки – інтеграція цих двох засобів, особливо що в деяких випадках (наприклад, при використанні технологій UWP та WPF) Visual Studio незамінна. Крім того, велика перевага – безкоштовність версії «Community edition», яка забезпечує увесь необхідний рядовому користувачеві функціонал. Також, дане середовище є дуже функціональним засобом, оскільки підтримує безліч якісних плагінів, за допомогою яких виникає можливість розширення функціональності розроблюваного додатку й підключення інших мов. Підтримання фреймворку .NET(включно з ASP.NET, що був використаний у розробці) – головна причина, оскільки сама мова програмування була обрана зі спонукання використання даної платформи, й дана

можливість надає середовищу широкі можливості розробки під операційну систему Windows.

Для підтримки бази даних інформаційної системи була обрана об’єктно-реляційна СКБД PostgreSQL, яка базується на мові SQL та підтримує багаточисленні можливості. Даний інструмент є досить популярним та був обраний через численні переваги, в числі яких – підтримка БД необмежених розмірів, потужні та надійні механізми транзакцій та наявність дуже зручного графічного клієнту pgAdmin 4. З його допомогою відбувається підтримка БД інформаційної системи, забезпечується збереження даних та виконання усіх необхідних для ІС транзакцій.

Для розробки front-end частини інформаційної системи були використані такі технології як React.JS, Node.js та Axios. В якості інтегрованого середовища розробки був обраний WebStorm.

Бібліотека React.JS була обрана, оскільки є надзвичайно зручною та ефективною для великих веб-додатків, де дані змінюються на регулярній основі, й через це є, за статистикою (рис. 3.2), найпопулярнішою серед аналогічних фреймворків JavaScript. Даний фреймворк представляє структуру в якості окремих компонентів та використовує розширення JSX. Досить легкий у вивченні.

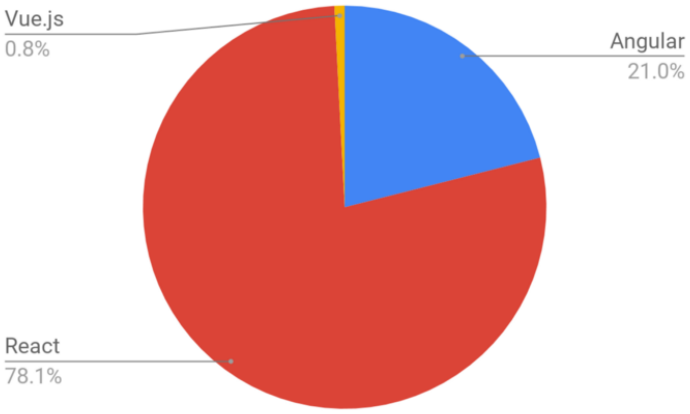


Рисунок 3.2

Node.js – середовище виконання JavaScript й потрібне для виконання програми.

Axios був обраний оскільки він є одним із самих популярних HTTP клієнтів для браузерів та node.js, володіє можливостями підтримки запитів, отримання відповідей від сервера, їх трансформація та конвертація у формат JSON.

Інтегроване середовище розробки WebStorm було обране через його зручність, функціональність та підтримку усіх мов та фреймворків, що використовуються для розробки front-end частини й є зручнішим для цього аніж Visual Studio.

Також, для створення графічного інтерфейсу були використані технології HTML, CSS та бібліотека Bootstrap. Bootstrap є простим та зручним рішенням з безліччю готових класів для створення будь-якого дизайну.

3.3 Розробка спеціалізованої інформаційної системи

Програмний додаток був спроєктований на базі клієнт-серверної архітектури, тобто система працює наступним чином (рис. 3.3.1): користувач через клієнтський інтерфейс відправляє запити до програми-сервера, яка в свою чергу їх обробляє, направляє запит до бази даних, отримує відповідь та у стандартизованому вигляді відправляє клієнтові.

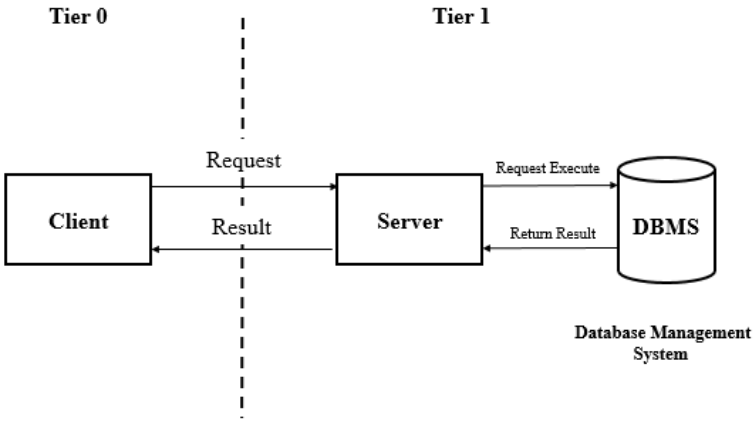


Рисунок 3.3.1 Схема клієнт-серверної архітектури

З метою створення максимально зручного та пристосованого до системних потреб відповідної специфіки користувацького інтерфейсу були проаналізовані такі аналогічні інформаційні веб-системи з точки зору як візуальної зручності так і функціональності: DeviantArt (рис. 3.3.2), Behance (рис. 3.3.3), Dribble (рис. 3.3.4) та інші менш популярні сайти.

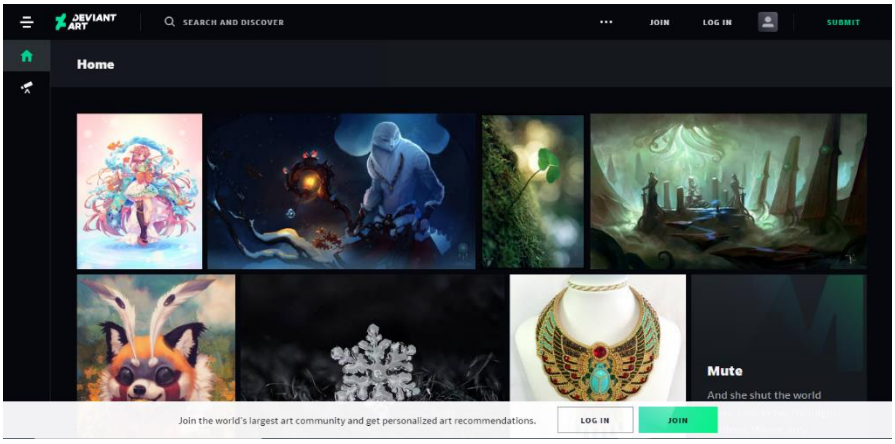


Рисунок 3.3.2 DeviantArt.com

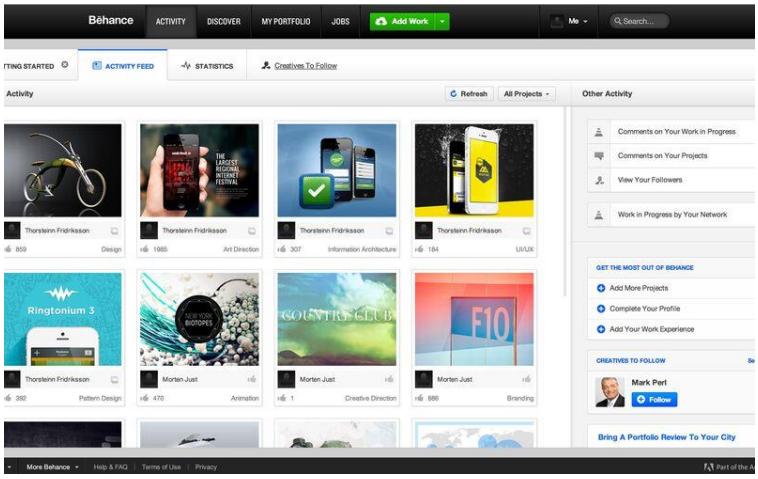


Рисунок 3.3.3 Behance

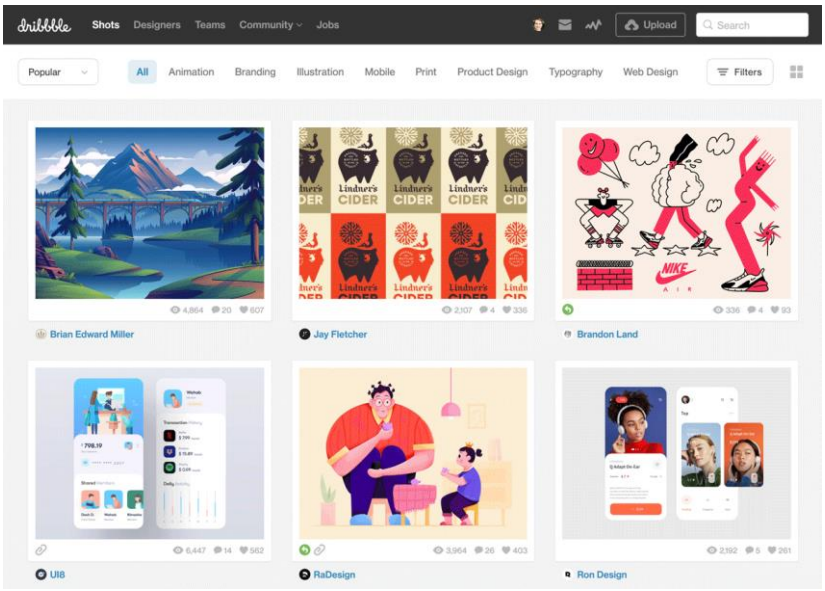


Рисунок 3.3.4 Dribble

Далі постало питання про програмне втілення концепції системи. Розробка серверної частини застосунку відбувалася наступним чином: спочатку були продумані та розроблені сутності, які будуть головними у програмі. Для таких

сутностей були створені класи Album, Bid, Comment, Genre, Like, Order, Painting, Role, User, які були поміщені у namespace Models. Усі ці класи відповідають кожній раніше створеній таблиці у базі даних. Далі потрібно було розробити механізм зв’язку програми з базою даних. Для цього було використано шаблон Data Access Object (DAO) – один із найбільш розповсюджений патернів проєктування. Ідея цього патерну – доступ до даних. Він створює прошарок, який відповідає лише за доступ до даних. Саме він займається внесенням, зміною, видаленням та вибіркою інформації із бази даних. З цією метою був створений namespace DataAccessLayer, в якому за допомогою бібліотеки Npgsql були написані запити за допомогою яких безпосередньо і відбувається взаємодія з базою даних, а саме додаваються, редагуються та отримуються дані. Відповідно було створені класи AlbumAdapter, AuctionAdapter, GenreAdapter, OrderAdapter, PaintingAdapter та UserAdapter щоб відокремити логіку зв’язку з базою даних різних сутностей. Для реалізації бізнес логіки було створено namespace Providers. У даному неймспейсі аналогічно з namespace DataAccessLayer було розділено логіку різних сутностей в окремі класи, так було створено AlbumProvider, AuctionProvider, GenreProvider, OrderProvider, PaintingProvider та UserProvider. У кожному такому класі була реалізована логіка обробки даних для подальшої передачі їх на фронт частину. На останньому етапі розробки серверної частини постало завдання передавати спершу отримані з бази даних, а потім оброблені за допомогою провайдерів дані на фронт частину програми. Для цього було вирішено скористатися контролерами, які досить легко реалізуються засобами фреймворку .NET. Для того щоб відділити логіки передачі даних що стосуються різних сутностей було створено декілька контролерів, а саме AlbumsController, AuctionsController, GenresController, OrdersController, PaintingsController та UsersController. Кожному контролеру було присвоєно свій унікальний route, для того щоб робити запити саме на потрібний контролер. Крім цього, для кожного методу у контролері також було виділено унікальний в рамках контролера роут. Таким чином лише прописавши правильний роут можна було отримувати потрібні дані з бек частини. На цьому етапі розробка бек-екнд частини була завершена.

Реалізовувати фронт частину було вирішено за допомогою бібліотеки React. Спершу постало питання як саме отримувати дані з back-end частини. На допомогу прийшла бібліотека axios, що знаходилась у node.js. Вона дозволяє робити запити на потрібні роути, а у відповідь повертати потрібні дані. Тому були розроблені класи AlbumService, AuctionService, GenreService, OrderService, PaintingService та UserService, в кожному з яких розроблені методи, які за допомогою бібліотеки axios реалізують зв'язок з бекенд-частиною, а саме використовуючи потрібний метод, наприклад get, post і т.д роблять запити на відповідні роути контролерів, що знаходяться на бек-енд частині, які в свою чергу повертають дані, які запрошує фронт.

Так як питання про отримання даних з бек-енд частини було вирішене, потрібно було переходити до розробки дизайну сайту. Використовуючи ліцензійну програму Adobe Photoshop, був розроблений макет дизайну застосунку. На наступному етапі поставло завдання реалізувати розроблений дизайн за допомогою фреймворку React.js. Весь фронт був поділений на окремі логічні частини – компоненти, кожен з яких відповідав за певну частину сайту. Так були створені компоненти для картин, жанрів, альбомів, вспливаючих вікон та інших елементів. Загалом було розроблено сорок компонентів. Далі потрібно було кожному компоненту надати відповідний дизайн. Для цього був використаний інструментарій CSS3 та його фреймворк Bootstrap.

На наступному етапі потрібно було зв'язати всі компоненти в одне ціле. Для цього спочатку було розроблено п'ятнадцять головних компонентів, кожен з яких відповідав за певну логічну частину сайту, так компонент PaintingPage відповідає за сторінку картини, Home – за головну сторінку, Login – за сторінку авторизації, Albums – за сторінку на якій знаходяться всі альбоми, UserPage – за сторінку користувача і т.д. Залишалось тільки зробити навігацію між даними компонентами, щоб користувач мав змогу переміщуватися між сторінками. Для цього був використаний такий пакет бібліотеки React як React Router. З його допомогою можна зробити переключання компонентів за певними шляхами, так компоненту PaintingPage був наданий роут /painting/:id, Home – /home, Login – /login, Albums – /albums, UserPage – /user і т.д. При переході на певний роут

відбувається рендер компонента, який знаходиться по даному роуті, у цей момент викликається потрібний метод із сервіс класу та підгружаються дані з бек-енд частини, далі засобами мови JavaScript дані обробляються та виводяться у компонентах.

На останньому етапі потрібно було протестувати систему, для цього були використані юніт та мок тести, за допомогою мок тестів, були відтворені сценарії роботи програми, додавання або редагування даних в БД, а за допомогою юніт тестів, була протестована правильність роботи бізнес логіки застосунку.

Важливим етапом у створенні інформаційної системи була розробка бази даних для неї, де має зберігатися уся інформація про користувачів, картини, та іншу системну інформацію. В процесі роботи з БД використовувалася СКБД PostgreSQL. Була створена БД, що складається з 15 таблиць наступного вмісту:

- **Albums**: Таблиця містить інформацію про створені користувачем альбоми й має наступні атрибути: PK – album_id, FK – user_id та обов’язкові атрибути title (назва альбому) та description (опис альбому).
- **Paintings**: Таблиця містить інформацію про кожну картину та має наступні атрибути: PK – painting_id, та обов’язкові атрибути title (назва картини), materials (матеріали виготовлення картини), painter (автор картини), price (ціна), description (опис роботи) upload_date (дата викладення), status (статус продано/непродано), image (фото картини).
- **Users**: Таблиця містить інформацію про користувачів системи, тобто інформація з облікового запису й має наступні атрибути: PK – user_id, FK – role_id, та обов’язкові атрибути firstname, lastname (ім’я та прізвище), address, email, phone_number, location, date_of_birth, gender, login, password.
- **Roles**: Таблиця містить інформацію про ролі користувачів в системі (в залежності від ролі користувач має певний набір можливостей) й має наступні атрибути: PK – role_id, та обов’язкові атрибути role_name, role_description.

- **Orders**: Таблиця містить інформацію про транзакції, тобто про покупки клієнтів й має наступні атрибути: PK – order_id, FK – user_id, та обов’язкові атрибути date (дата покупки), amount (сума), status.
- **Likes**: Таблиця містить інформацію про лайки й має наступні атрибути: PK – like_id, FK – user_id, painting_id.
- **Comments**: Містить інформацію про коментарі користувачів й має наступні атрибути: PK – comment_id, FK – user_id, painting_id, та обов’язкові атрибути content, date.
- **Genres**: Таблиця містить інформацію про жанри й має наступні атрибути: PK – genre_id, та обов’язкові атрибути title, description, ismovement, image.
- **Follows**: Таблиця містить інформацію про підписників та має наступні атрибути: PK – follower_id, following_id.
- **AlbumPaintings**: Таблиця містить інформацію про те, які картини містяться в яких альбомах и має наступні атрибути: PK – album_id, painting_id.
- **BidHistory**: Таблиця містить інформацію про історію аукціонних ставок й має наступні атрибути: PK – bet_id, FK – bid_id, user_id, та обов’язкові атрибути bet, placetime.
- **BidProducts**: Таблиця містить інформацію про кожний картинний аукціон та має наступні атрибути: PK – bid_id, FK – user_id, painting_id, та обов’язкові атрибути bid_price, status, start_time.
- **OrderPaintings**: Таблиця містить системну інформацію про замовлення і має наступні атрибути: PK – order_id, painting_id, count.
- **PaintingsGenres**: Таблиця містить опис зв’язку картин із жанрами й має наступні атрибути: PK – painting_id, genre_id.
- **PaintingsUsers**: Таблиця містить зв’язок користувачів з картинами, що були викладені ними й має наступні атрибути: PK – painting_id, user_id.

3.4 Демонстрація основного функціоналу системи

При переході на головну сторінку (рис. 3.4.1) користувач відразу побачить стрічку, яка відображає найпопулярніше в системних категоріях – топ найпопулярніших картин, аукціонів, альбомів, художніх жанрів та, в кінці, список усіх наявних картин. Система визначає популярність в залежності від кількості лайків та коментарів. Для візуальної зручності, у стрічці картини зображені у вигляді карток із зображенням самої картини, та нижче – короткої загальної інформаційної довідки про неї, а саме – назва роботи, її ціна, кількість лайків (кількість користувачів, котрим сподобалася картина) та коментарів. Альбоми та жанри зображені схожим чином, але без характерної додаткової інформації. Також на головній сторінці є два типи меню – вертикальне, де міститься навігація системою, тобто покликання на інші її розділи, а саме сторінки жанрів, альбомів й аукціонів, та горизонтальне меню, де містяться інструмент пошуку картин системою, а також посилання на авторизацію та реєстрацію у системі. Без авторизації, більшість головних функцій системи будуть недоступні. Дані два типи меню видні та доступні з будь-якого розділу системи. Також, у нижньому колонтитулі сторінки (або футері), міститься інформація про компанію даної системи із зазначенням авторських прав і покликаннями на соціальні мережі та додаткова навігація системою, аналогічна навігації у вертикальному меню.

Сторінка жанрів (рис. 3.4.2) виглядає аналогічним чином як і домашня сторінка – на стрічці показані спочатку топи найпопулярніших жанрів, художніх напрямів, та, вже в кінці, список усіх наявних у системі. Вертикальне меню може бути у згорнутому (як на рис. 3.4.1) або розгорнутому (рис. 3.4.3) стані, залежно від зручності для користувача. Для детального перегляду кожного жанру або напряму користувач має натиснути на той, що його цікавить та система відобразить подробиці у новій сторінці (рис. 3.4.4), де він зможе прочитати додаткову пізнавальну інформацію про обраний жанр/напрям та побачить список картин в системі, що належать до відповідної категорії.

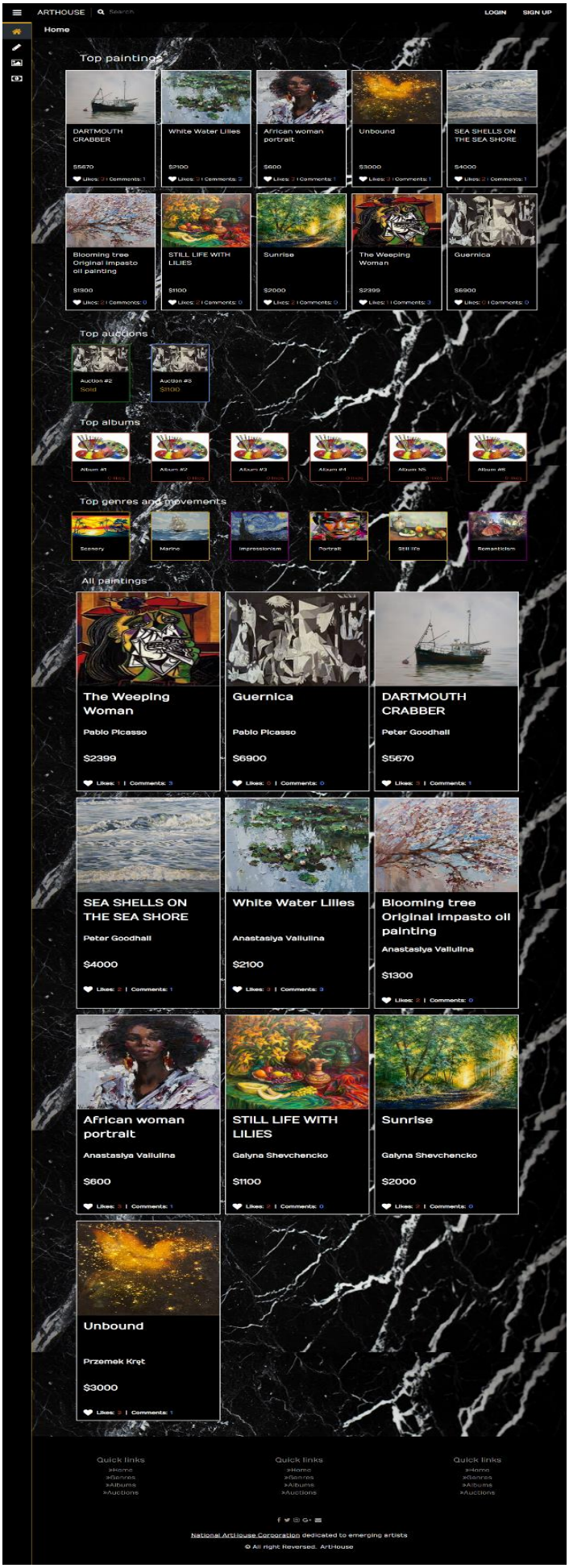

ARTHOUSE
Search
LOGIN
SIGN UP
Home
Top paintings
DARTMOUTH CRABBER
$5670
White Water Lilies
$2100
African woman portrait
$600
Unbound
$3000
SEA SHELLS ON THE SEA SHORE
$4000
Blooming tree Original impasto oil painting
$1300
STILL LIFE WITH LILIES
$1100
Sunrise
$2000
The Weeping Woman
$2399
Guernica
$6900
Top auctions
Auction #2
Sold
Auction #3
$1100
Top albums
Top genres and movements
Scenery
Marine
Impressionism
Portrait
Still life
Romanticism
All paintings
The Weeping Woman
Pablo Picasso
$2399
Guernica
Pablo Picasso
$6900
DARTMOUTH CRABBER
Peter Goodhall
$5670
SEA SHELLS ON THE SEA SHORE
Peter Goodhall
$4000
White Water Lilies
Anastasiya Vallulina
$2100
Blooming tree Original impasto oil painting
Anastasiya Vallulina
$1300
African woman portrait
Anastasiya Vallulina
$600
STILL LIFE WITH LILIES
Galyna Shevchencko
$1100
Sunrise
Galyna Shevchencko
$2000
Unbound
Przemek Krąt
$3000
Quick links
>Home
>Genres
>Albums
>Auctions
Quick links
>Home
>Genres
>Albums
>Auctions
Quick links
>Home
>Genres
>Albums
>Auctions
National ArtHouse Corporation dedicated to emerging artists
© All right Reversed. ArtHouse

Рисунок 3.4.1

Рисунок 3.4.2

Решта розділів – альбоми та аукціони – мають дизайн інтерфейсу аналогічний попереднім. Спочатку зображені найпопулярніші об’єкти розділу, а потім – усі наявні. Проте в цьому випадку, для доступу до більш детальної інформації про аукціони або альбоми, користувач повинен або авторизуватися у системі шляхом введення своїх логіну та паролю, або ж створити новий обліковий запис, якщо користувач його ще не має.

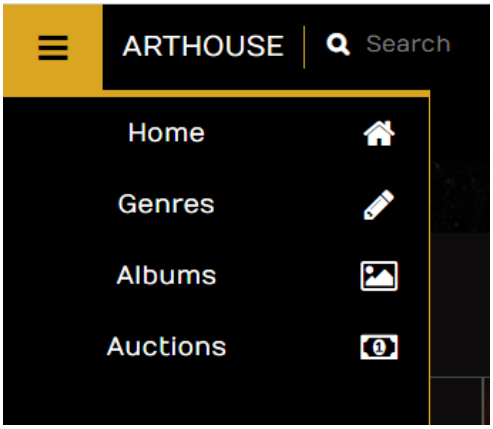


Рисунок 3.4.3

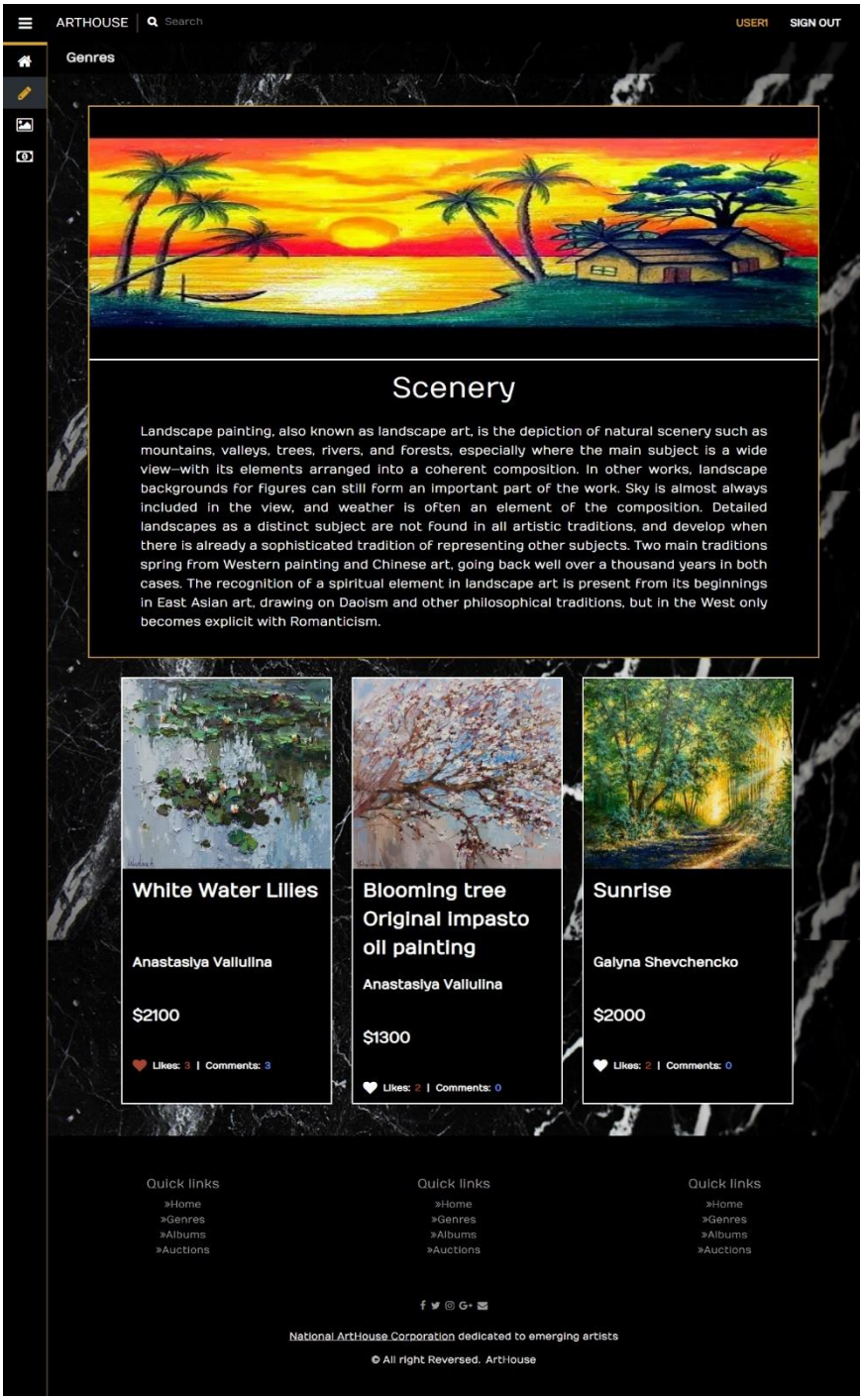


Рисунок 3.4.4

Форми для авторизації та реєстрації зображені на рис.3.4.5 та рис.3.4.6. Як вже було зазначено раніше, для авторизації з уже існуючим обліковим записом, користувач має ввести лише своє користувацьке ім’я та пароль, котрий відразу шифрується системою за допомогою алгоритму шифрування sha256 й зберігається у базі даних в зашифрованому вигляді, що забезпечує повну безпеку конфіденційності користувача.

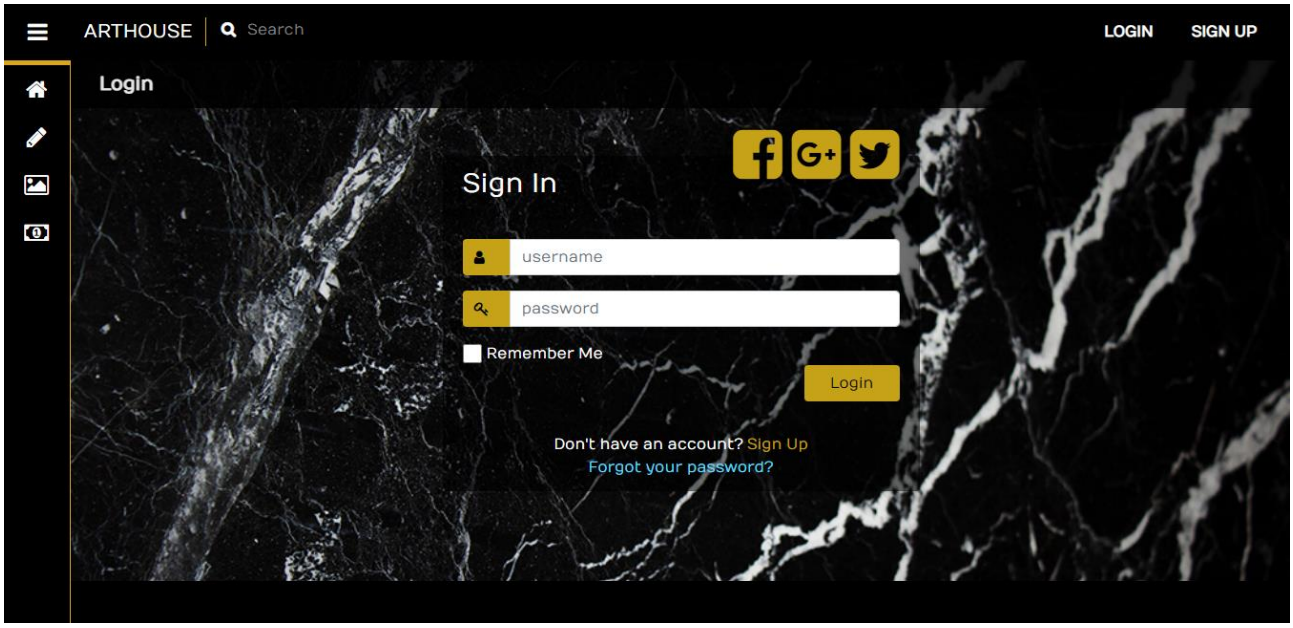


Рисунок 3.4.5

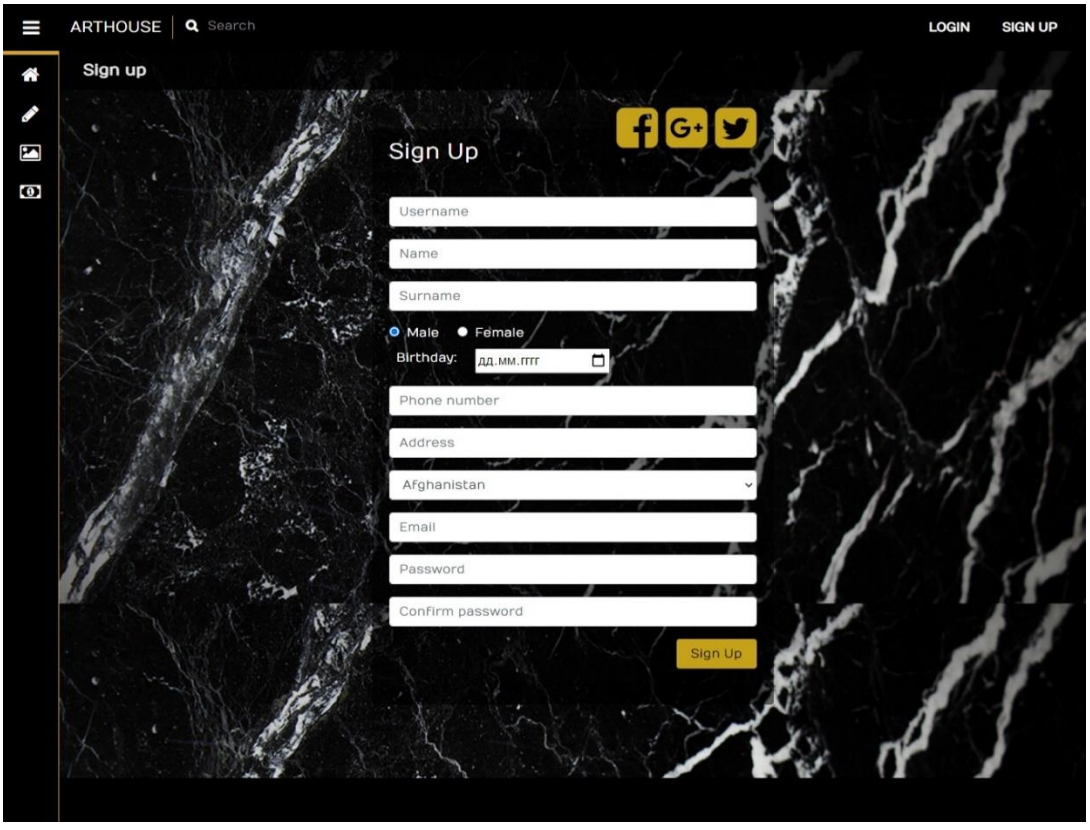


Рисунок 3.4.6

Для реєстрації нового облікового запису, користувач має заповнити усі запропоновані поля форми, а саме нікнейм, справжні ім’я та прізвище, стать, дату народження, номер телефону, адресу, країну перебування, електронну пошту та пароль. Поля номера телефону, електронної пошти та паролю перевіряються системою на валідність даних. Номер телефону та адреса електронної пошти мають відповідати визначеній структурі, а пароль, з метою надійності, не може бути менший аніж 7 символів.

Після авторизації, користувач може перейти у свій особистий кабінет, котрий зображено на рис.3.4.7.

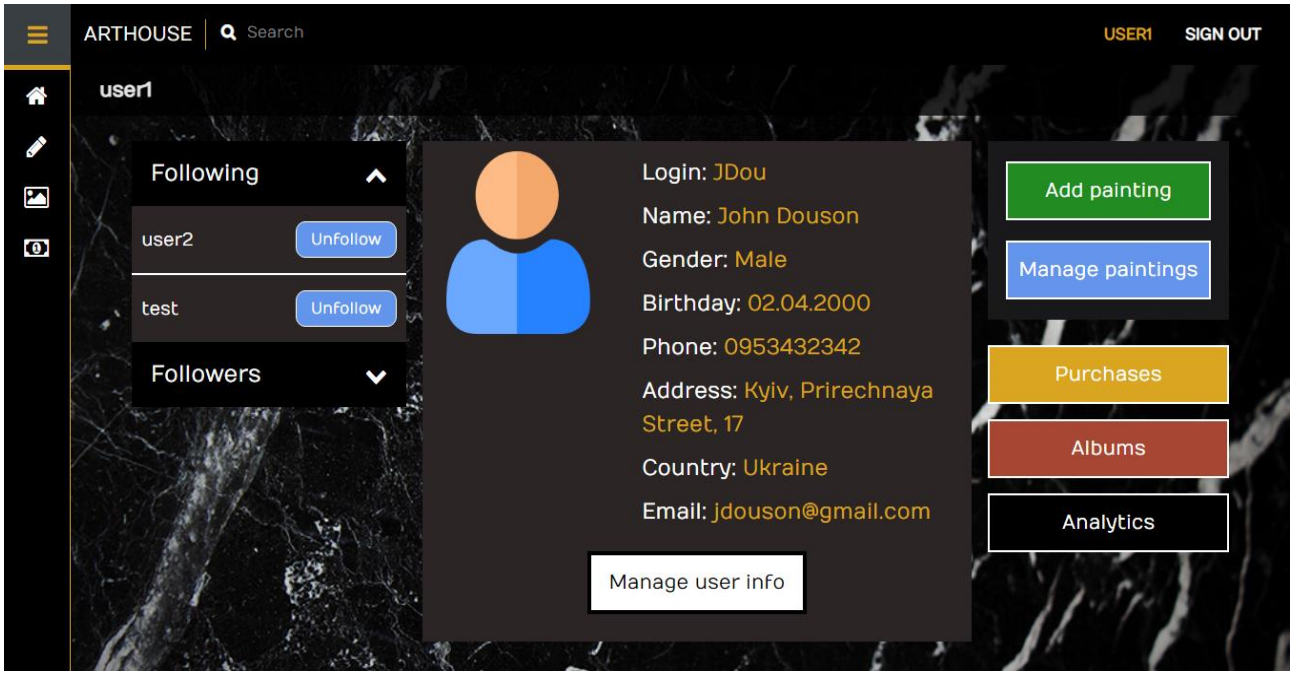


Рисунок 3.4.7

Тут користувач має можливість переглядати своїх підписників та тих, на кого він підписаний, редагувати інформацію про себе, додавати свої картини та керувати ними (рис.3.4.8), додавати нові альбоми (рис.3.4.9) й картини до них, а також робити покупки. Користувач має можливість додавати нові картини, редагувати інформацію про вже існуючі (назва картини, опис матеріалів та самої суті роботи, ціна), переглядати кількість лайків або ж видаляти картини. Аналогічний функціонал доступний і для роботи з альбомами.

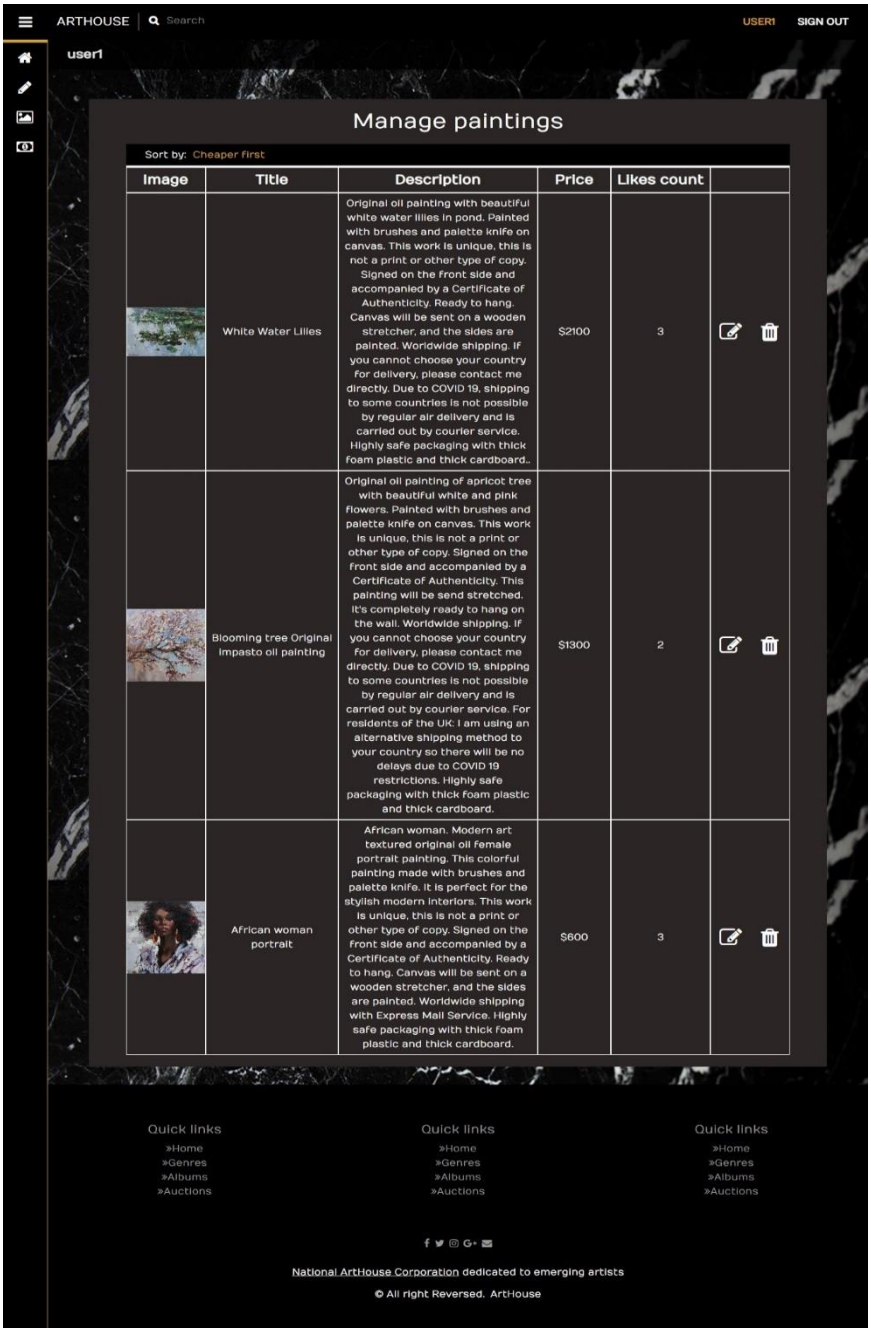


Рисунок 3.4.8

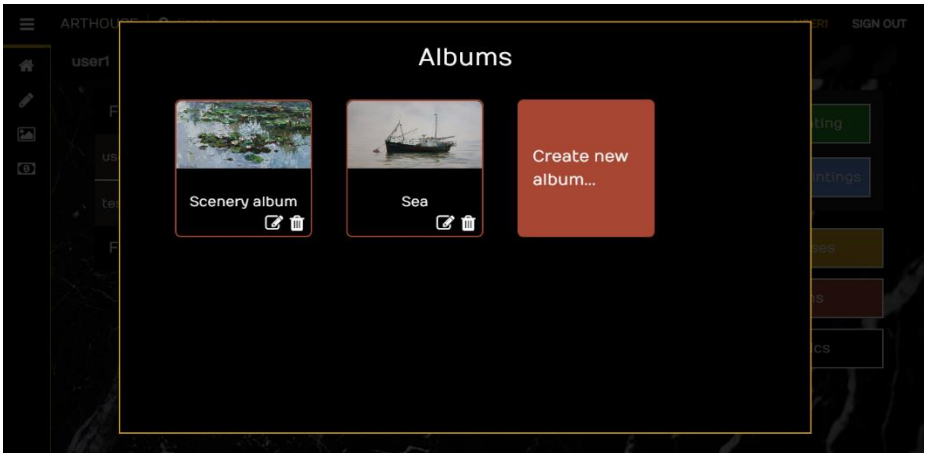


Рисунок 3.4.9

Для проведення платежу потрібно перейти у відповідне вікно (рис.3.4.10).

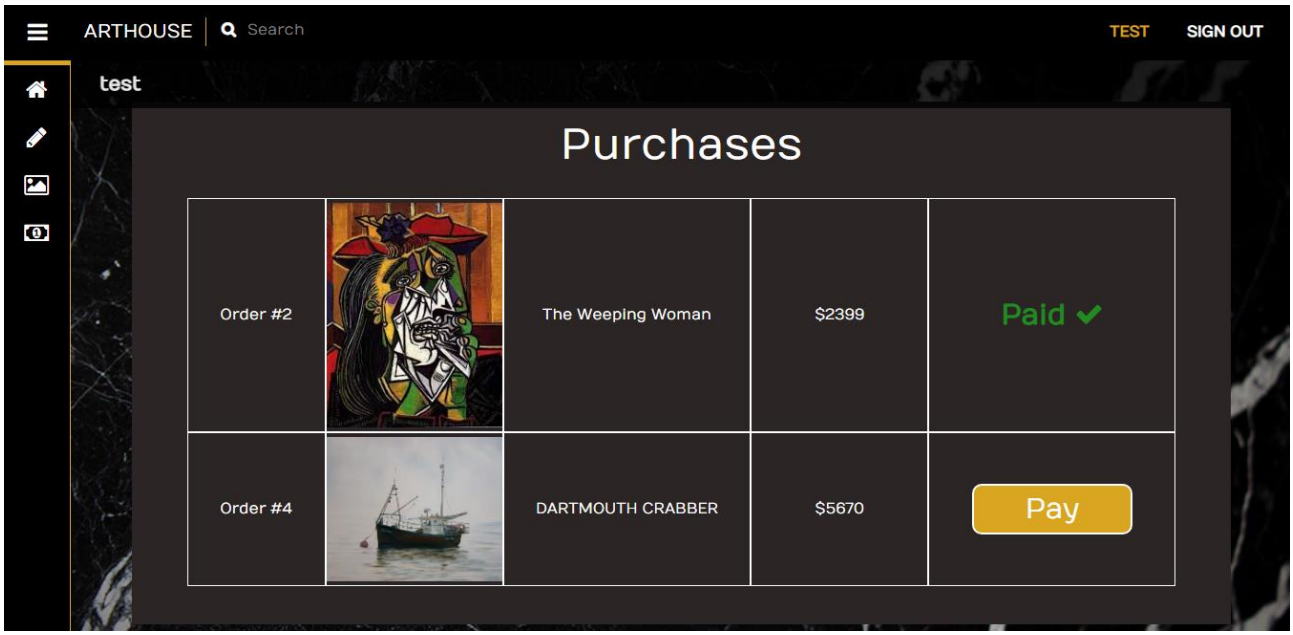


Рисунок 3.4.10

Якщо покупка вже проплачена, система сповістить про це. Якщо ні – буде активна кнопка для відповідної дії. Після оплати користувач отримає чек (рис.3.4.11), де будуть прописані товари, їх ціни, та загальна сума сплати. Приємний бонус для покупця – передбачення на день.

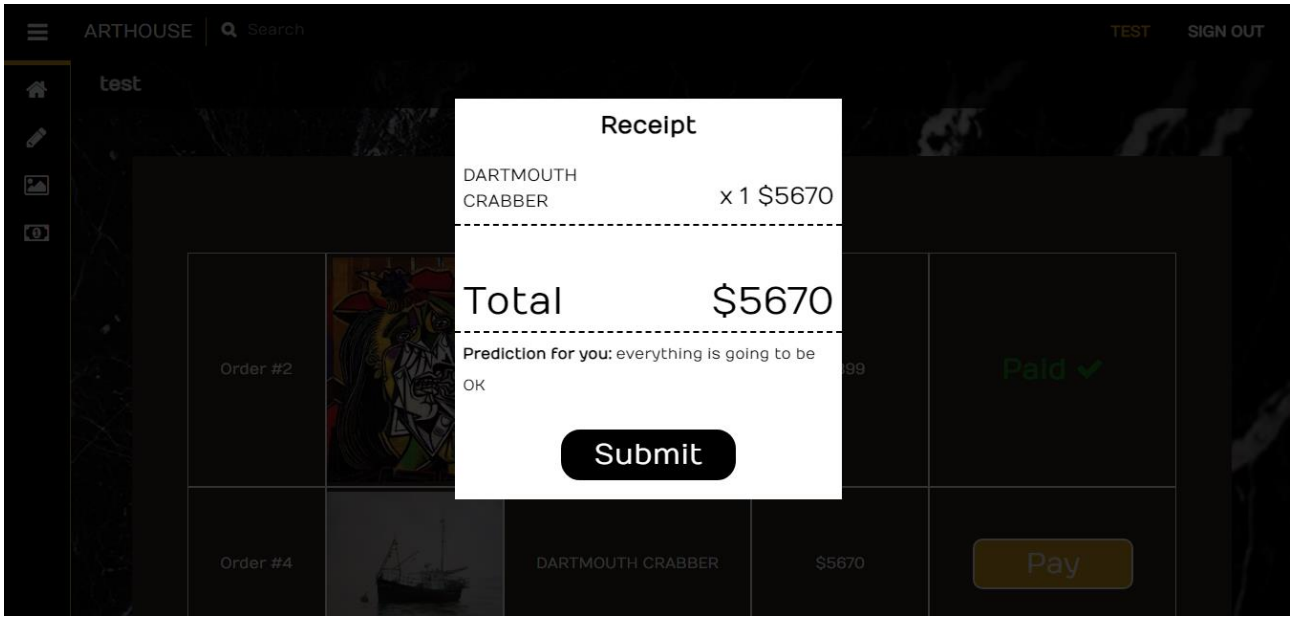


Рисунок 3.4.11

Також, система передбачає користувачам детальний перегляд опублікованих картин. Сторінка публікації зображена на рис.3.4.12. Тут можна

побачити детальну інформацію про картину (автор, жанр, опис матеріалів та роботи), її назву, користувача, що її виклав та ціну.

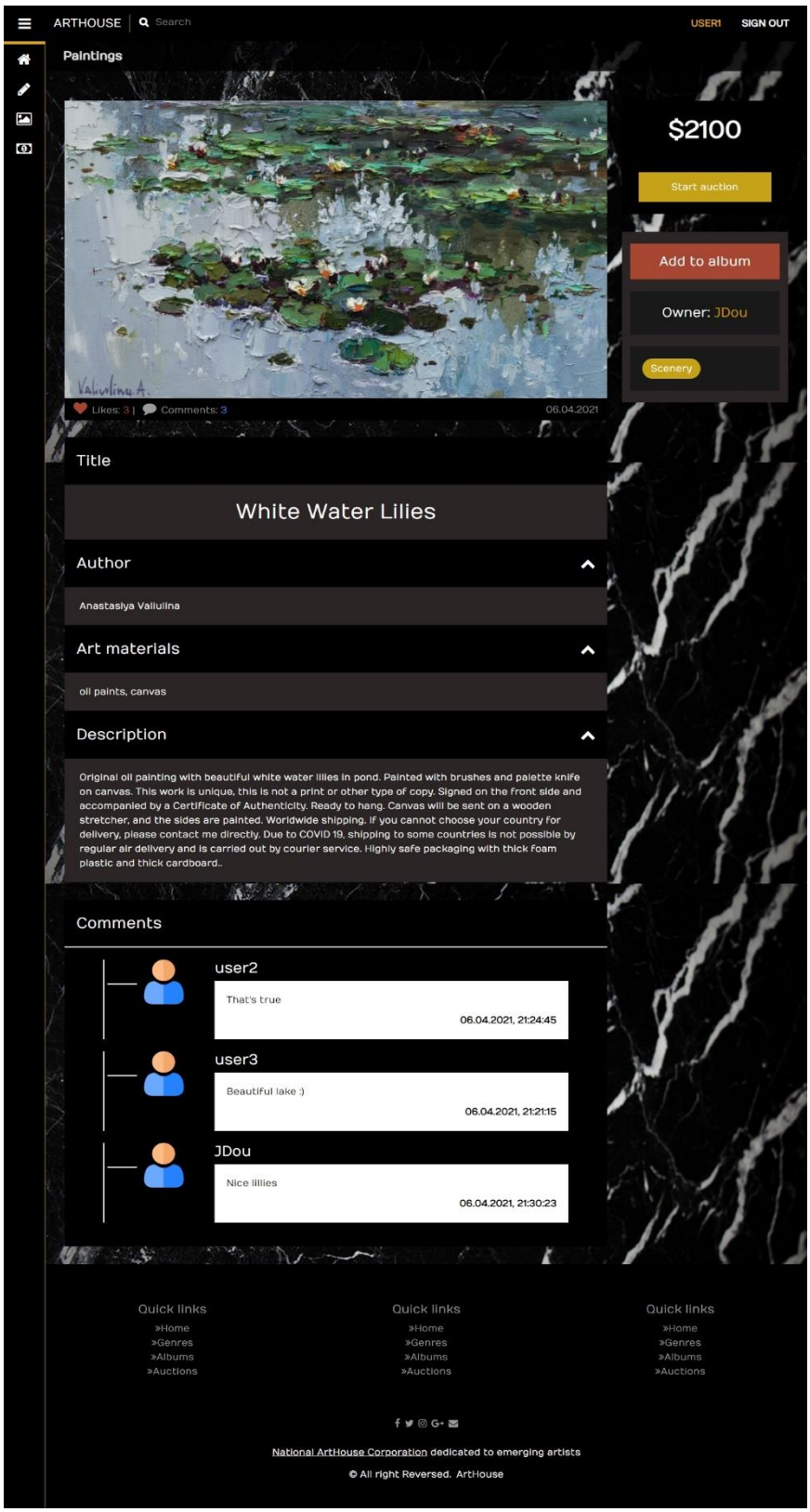


Рисунок 3.4.12

Також, користувач має можливість поставити лайк, залишити свій коментарій або додати картину до свого власного альбому. В системі доступні близько двадцяти художніх жанрів та напрямів.

Система надає користувачу можливість продажу картини через аукціон. Сторінка аукціону показана на рис.3.4.13. Для цього користувач має створити аукціон, вказавши картину, себто лот, та час, який він буде тривати. Тоді інші користувачі мають робити свої ставки, які будуть відображатися у вікні історії ставок й перемагає користувач, чия ставка була останньою по завершенні часу. Тоді цей самий учасник матиме можливість придбати виграний лот.

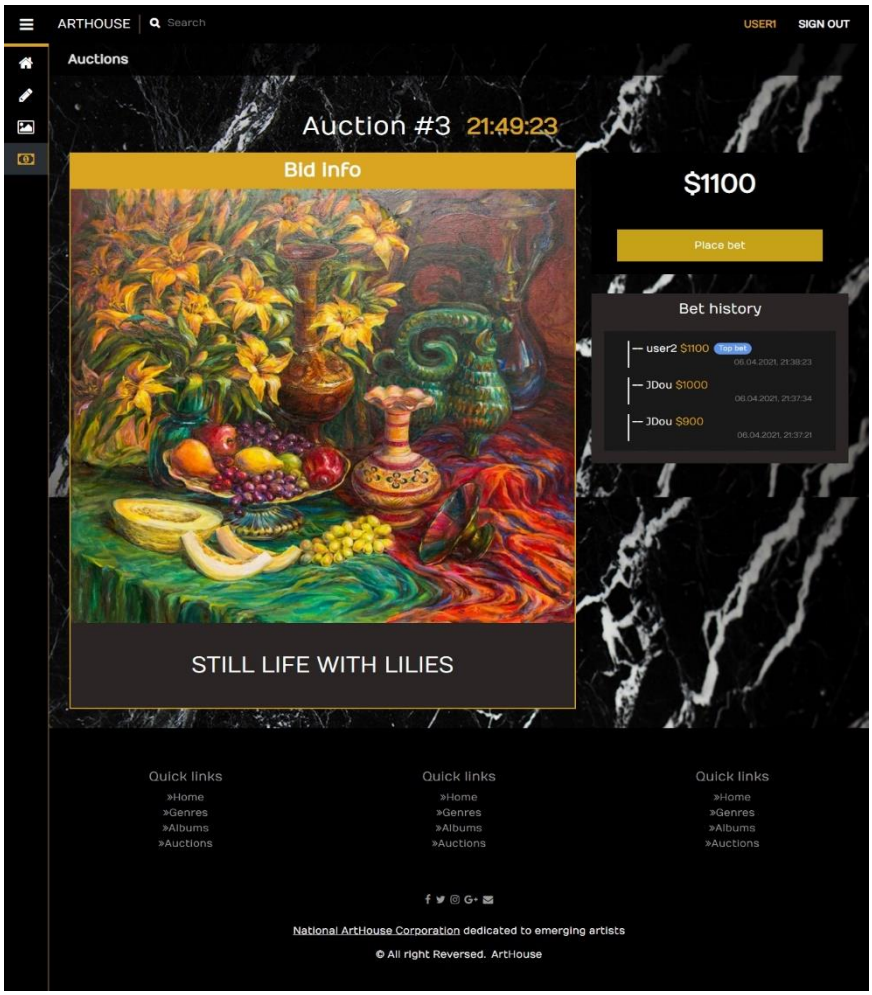


Рисунок 3.4.13

Система також надає інструмент пошуку картин за назвою, робота якого показана на рис.3.4.14.

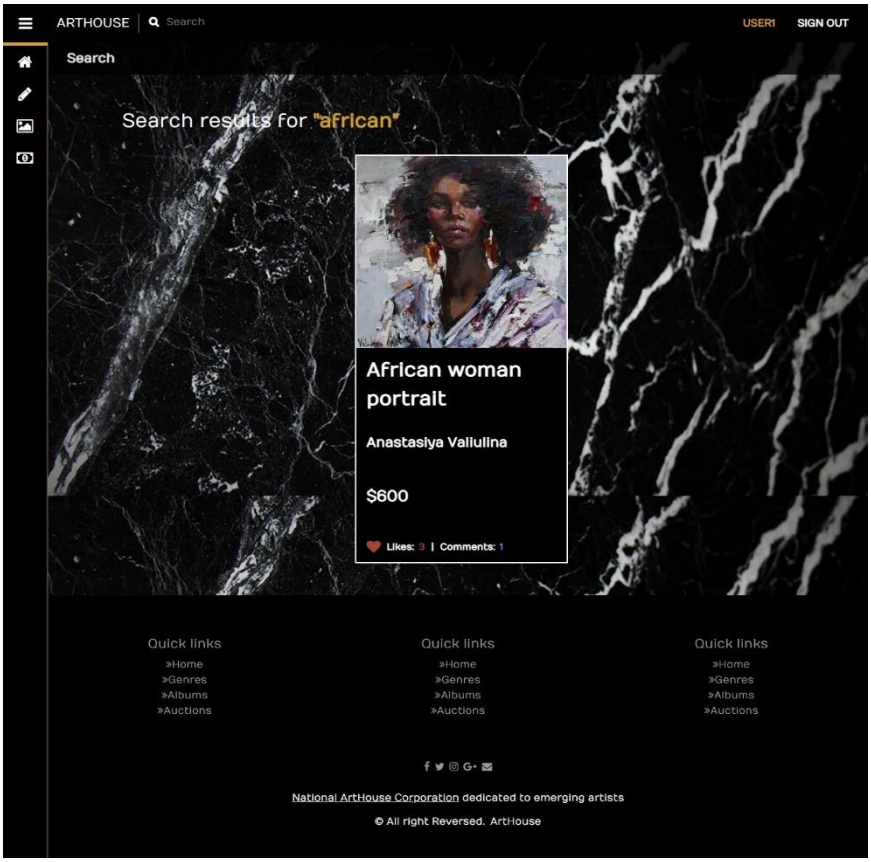


Рисунок 3.4.14

Отже, система є функціонально досить розвинутою та задовільняє усім основним користувацьким потребам, передбаченим особливою спеціалізацією системи, а саме - допомога художникам-початківцям у їх творчому та кар’єрному просуванні. Користувачі системи мають змогу переглядати творчі здобутки та ідеї інших користувачів, зберігати їх на пам’ять у власні альбоми, ставити лайки, коментувати роботи та ділитися своїми власними художніми витворами. Також, користувачі мають змогу придбати чиюсь роботу або продавати власні прямо або через механізм аукціону. Користувачі мають змогу перегляду робіт за жанрами, напрямами або ж художниками, які їх цікавлять. Окрім того, ІС несе також просвітню функцію, надаючи додаткову цікаву інформацію про мистецькі напрями. Інтерфейс є простим та зручним, а система захищеною та безпечною, адже конфіденційні дані зберігаються в базі даних у зашифрованому вигляді. Розроблена спеціалізована інформаційна система є гідним прототипом для повноцінного потенційного робочого програмного продукту.

Висновок

В процесі виконання курсової роботи були досліджені поняття інформаційної системи, її структури, проаналізовані методи та стратегії розробки спеціалізованої ІС, детально розглянуті етапи її створення, сукупність методів та засобів проєктування з використанням CASE-засобів, розгляд усіх їх компонентів з приведенням конкретних прикладів. Для закріплення та більш глибокого розуміння вивченого матеріалу у рамках виконання технічного завдання була розроблена власна спеціалізована інформаційна система, яка призначена для використання художниками-початківцями. На етапі концептуального дизайну були детально проаналізовані аналогічні системи, тому створений програмний продукт передбачає увесь основний функціонал, необхідний для специфіки обраної предметної області. Користувачі системи мають можливість створення власного облікового запису та авторизації, публікації та продажу за допомогою системи власних художніх творів або перегляд та придбання чужих напряму або через участь в аукціоні, коментування та вподобання постів, підписки на інших користувачів з метою слідкування за їх творчістю, а також передбачені механізми створення власних альбомів з роботами, пошуку картин системою за назвою, жанрами та художніми напрямами. Крім того кожний напрям та жанр мають власну окрему сторінку з описом в освітніх цілях. Збереження та підтримання даних в актуальному вигляді, необхідних для функціонування інформаційної системи реалізується засобами СКБД PostgreSQl, зв’язок зі створеною БД та бізнес-логіка програми втілюється за допомогою механізмів мови програмування C# та платформи .NET, клієнтська частина виконана на JavaScript-бібліотеці React. Використання поданих засобів розробки ІС на практиці поглибило та розширило розуміння викладеного у даній роботі теоретичного матеріалу, сприяло опануванню нових програмних технологій та покращенню навичок із вже знайомими.

Актуальність даної роботи є беззаперечною. Нині, інформаційні системи виступають єдиним основним функціональним рішенням проблеми необхідності в ефективних та зручних способах збереження, пошуку, обробки, та безлічі інших видів маніпулювань даними, яких гостро потребують усі сучасні сфери

життя, починаючи від бізнесу та закінчуючи державними структурами. Без інформаційних систем будь-яка продуктивна автоматизована діяльність стає неможливою, тому дуже важливо вміти та розуміти як правильно та ефективно налагодити процес розробки ІС будь-яких масштабів, враховуючи усі етапи, найкращу для кожної окремої ситуації стратегію та засоби розробки, вірно розподілити роботу на кожному кроці між учасниками процесу створення ІС. Матеріал, досліджений та викладений в рамках роботи над курсовим проєктом розглядає усі сторони життєвого циклу інформаційних систем від методів та стратегій до конкретних засобів розробки продукту. Тому, враховуючи сучасні тенденції розвитку інформаційних систем, набуті знання та навички впродовж роботи над теоретичною та технічною частинами курсового проєкту є актуальними та корисними.

Отже, наприкінці виконання курсової роботи можна зазначити, що поставлена мета проєкту досягнена, поставлені задачі виконані, практична частина – реалізована. Досліджений теоретичний матеріал був цікавим та пізнавальним, розглянуті теми – актуальними в сучасній технологічній реальності. В перспективі функціональні можливості розробленого програмного продукту можуть бути розширені шляхом додання, наприклад, статистичного механізму, який міг би збирати та демонструвати такі корисні для маркетингу дані як співвідношення популярності певних жанрів, художніх напрямів або конкретних художників. Також має потенціал ідея інтеграції ІС з технологією рекомендаційних систем, які б могли аналізувати вподобання користувача за певними критеріями та згідно з отриманими даними формувати його домашню стрічку. Розроблена інформаційна система може бути залучена до роботи як повноцінний програмний продукт в наявному або вдосконаленому вигляді.

Список використаної літератури

1. [Електронний ресурс] Systems implementation & evaluation. Дата використання 01.03.2021 https://www.uky.edu/~dsianita/695A&D/lecture5.html#:~:text=Systems%20implementation%20is%20the%20process,(i.e.%2C%20quality%20assurance)
2. [Електронний ресурс] Conceptual modeling of information system. Дата використання 10.03.2021 https://www.researchgate.net/publication/220691699_Conceptual_Modeling_of_Information_Systems
3. [Електронний ресурс] Conceptual Design of a system. Дата використання 10.03.2021 https://ecomputernotes.com/mis/system-design/what-is-meant-by-the-conceptual-design-of-mis-discuss-various-steps-involved-in-the-conceptual-design-of-a-system
4. [Електронний ресурс] Logical and Physical design. Дата використання 11.03.2021 https://slideplayer.com/slide/254376/
5. [Електронний ресурс] Концептуальний та фізичний рівні представлення даних. Дата використання 11.03.2021 https://studfile.net/preview/1752926/page:4/
6. [Електронний ресурс] CASE Tools. Дата використання 13.03.2021 http://www.umsl.edu/~sauterv/analysis/F08papers/View.html
7. [Електронний ресурс] Components of Database Management System. Дата використання 13.03.2021 https://www.techradar.com/best/best-flowchart-software
8. [Електронний ресурс] Using CASE-Tools for Database Design. Дата використання 14.03.2021 https://www.sciencedirect.com/science/article/pii/B9780123747303000127
9. [Електронний ресурс] Computer Aided Software Engineering. Дата використання 16.03.2021

https://www.geeksforgeeks.org/computer-aided-software-engineering-case/

10. [Електронний ресурс] DeviantArt. Дата використання 15.01.2021 https://www.deviantart.com/
11. [Електронний ресурс] Огляд PostgreSQL СКБД. Дата використання 20.03.2021 https://www.postgresql.org/docs/
12. [Електронний ресурс] Diagram Software. Дата використання 16.04.2021 https://www.capterra.com/diagram-software/
13. [Електронний ресурс] Project Management Software. Дата використання 16.04.2021 https://project-management.com/top-10-project-management-software/
14. [Електронний ресурс] Essential Data Analyst Tools. Дата використання 17.04.2021 https://www.datapine.com/articles/data-analyst-tools-software

Перелік прийнятих скорочень

VCS – Version Control System

IDE - Integrated Development Environment

DBMS - Database Management System

ІС – Інформаційна система

СКБД – Система керування базами даних

БД – база даних

CASE - computer-aided software engineering

ПЗ – Програмне забезпечення

SDLC - Software Development Lifecycle

RAD - Rapid application development

JAD - Joint application development