**Реферат**

Випускна робота бакалавра складається з програмного комплексу для організації та виконання завдань за винагороду з вбудованим чатом та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 104 сторінках друкованого тексту, має 116 сторінок додатків. Список використаних джерел містить 30 найменувань і займає 3 сторінки. В роботі наведено 59 рисунків та 11 таблиць. Загальний обсяг роботи – 220 сторінки. Метою даної розробки є створення інтегрованої та зручної платформи, яка забезпечить ефективне виконання завдань за винагороду, сприятиме взаємодії користувачів та забезпечить безпеку та надійність фінансових операцій. Під час виконання завдання були визначені вимоги користувача та аргументовано створення концептуальної моделі. Також проведено аналіз аналогічних програмних продуктів, здійснено відбір інструментів для реалізації, розроблено структуру бази даних та алгоритм роботи. Здійснено вибір архітектури та розроблено об'єктно-орієнтовану структуру системи. Також здійснено опис програмної реалізації платформи.

КЛЮЧОВІ СЛОВА: АДМІНІСТРУВАННЯ, БАЗА ДАНИХ, ПЛАТФОРМА, КОРИСТУВАЧ, АДМІНІСТРАТОР, ЧАТ, СОКЕТИ, ЗАПИТИ, ЗАВДАННЯ, АВТОРИЗАЦІЯ, ЗАМОВНИК, ВИКОНАВЕЦЬ, НАГОРОДА.

**Скорочення**

БД – База Даних

MVC – Модель–вигляд–контролер (або Модель–представлення–контролер) — Концепція архітектури програм, що використовується при їхньому проектуванні та розробці.

ПЗ – Програмне забезпечення

РРЧ – режим реального час

ПДОТВЗЗВЗВЧ - платформа для організації та виконання завдань за винагороду з вбудованим чатом

ВЧ – вбудований чат

АПЗ – архітектура програмного забезпечення

MVC - Model-View-Controller

**Вступ**

Актуальність теми полягає в швидко зростаючій популярності використання веб-технологій для створення інноваційних платформ, які сприяють ефективній організації та виконанню завдань за винагороду. З великим розширенням цифрового середовища важливо надати користувачам зручний та інтерактивний інтерфейс для ефективного спілкування та виконання завдань.

**Актуальність теми** полягає у зростанні попиту на заробіток коштів за допомогою фріланс платформ. Крім того, у зв’язку з війною, пандемією та деякими іншими локальними причинами міграції населень різних країн, іноді виникає потреба зробити певну дію в регіоні, до якого фізично неможливо дібратись, або це занадто дорого, чи просто немає бажання, що змушує людей, шукати вихід за допомогою знайомих, що не завжди є зручним, як для того, хто робить запит, так і для того, кого просять допомоги. Так як на платформі завжди будуть відповідати лише користувачі, що реально зацікавлені у виконанні завдання, то це спростить як відчуття незручності у обох сторін, так і підніме коефіцієнт якості виконання завдань. А так як інтернет сьогодні є майже в кожній точці світу, то користувачі зможуть легко комунікувати між собою де завгодно та виконувати задачі коли завгодно Тому дана платформа буде користуватись попитом, як серед виконавців, так і серед тих, хто буде потребувати допомоги у вирішенні задач.

Пошук задач по локації, дозволять користувачам швидко брати нові завдання та переходити до їх виконання. Це дозволить зарорбляти легше, швидше та більше. Крім того, у користувачів буде рейтингова система, що дозволить передбачати якість виконання задачі і обирати замовникам кращого виконавця.

Спілкування у РРЧ з можливістю надсилання фото та відео дозволять користувачам краще взаємодіяти між собою для підвищення результату виконанная, що буде приємним моментом для обох сторін. Основні функції платформи включають в себе вбудований чат для зручного спілкування між користувачами, організацію завдань з можливістю їх виконання та визначення винагороди, а також систему оцінювання та зворотного зв'язку. Використання технологій node.js та react гарантує швидкість та ефективність роботи інтерфейсу, а сокети дозволяють забезпечити реальний час для обміну повідомленнями в чаті.

**Метою даної розробки** є створення інтегрованої та зручної платформи, яка забезпечить ефективне виконання завдань за винагороду, сприятиме взаємодії користувачів та забезпечить безпеку та надійність фінансових операцій.

Поставлена мета обумовлює необхідність виконання наступних завдань:

1) Аналіз теоретичних та функціональних потреб:

- Ретельний аналіз вимог та потреб користувачів у відношенні до організації та виконання завдань за винагороду.

- Визначення основних функціональних можливостей, які має надавати платформа.

2) Аналіз існуючих веб-сайтів та визначення унікальності:

- Оцінка існуючих платформ з аналогічною спрямованістю.

- Визначення особливостей, які роблять платформу унікальною та конкурентоспроможною.

3) Визначення технологій для реалізації онлайн-платформ:

- Вибір технологічного стеку, включаючи Node.js та React для розробки веб-інтерфейсу.

- Використання сокетів для забезпечення РРЧ.

- Інтеграція з MySQL для збереження даних користувачів та іншої необхідної інформації для правильної роботи платформи.

4) Визначення інформаційних потреб предметної області:

- Розуміння основних вимог користувачів та замовників.

- Визначення видів завдань, які можуть бути виконані на платформі.

5) Обрання оптимальної архітектури програмного комплексу:

- Вивчення сучасних тенденцій у розробці веб-додатків та вибір оптимальної архітектури для платформи.

6) Проектування бази даних:

- Розробка структури бази даних для збереження інформації про завдання та користувачів.

8) Визначення об’єктної структури додатку:

- Створення моделей та об’єктної структури для ефективної роботи платформи.

9) Розробка адаптивного інтерфейсу веб-сайту:

- Створення інтерфейсу, що адаптується до різних типів пристроїв, включаючи комп’ютери та мобільні пристрої.

10) Розробка математичної та алгоритмічної моделі:

- Розробка алгоритмів для ефективної роботи системи на основі бази даних.

10) Написання серверної частини:

- Реалізація серверного коду для обробки запитів, взаємодії з базою даних та зі сторонніми API.

11) Реалізація CMS:

- Розробка системи управління контентом для зручного адміністрування та модерації платформи.

**Об’єктом дослідження** є методи та засоби розробки платформи за

визначеними предметними областями.

**Предметом дослідження** є моделі, технології, структури, засоби, методи та технології розробки та взаємодії платформи та модулів управлінням нею

**РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ**

**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ**

**КІНОТЕАТРУ**

**1.1 Опис проекту та постановка задачі**

На сьогоднішній день, в умовах війни, пост-пандемійного періоду та інших ситуацій, що впливають на рішення людей щодо зміни місця перебування на довгий термін, дуже складно вирішити якісь прості дії самотужки в місцях, в які немає можливості фізично потрапити, так як для цього потрібно витратити багато сил, коштів, енергії та інших ресурсів, щоб виконати задачу. Наприклад, дуже складно, привітати свою подругу чи друга з якоюсь подією в червні, подарувавши букетик ромашок та кавун, без допомоги інших друзів, що досі живуть в тому ж місці, що і людина, яку хочеться привітати.

Даний проект, має на меті полегшити та зробити більш доступними такі види комунікації та взаємодії в умовах глобальної нестабільності. Дана іноваційна платформа дозволить людям як віддалено обмінюватися реальними матеріальними подарунками, так і просто: співати гарно пісні, розповідати виразно вірші, яких ще не бачив світ чи просто передавати пакет продуктів рідній бабусі, яку вже не бачив декілька років, не маючи змоги приїхати і допомогти.

Тож одним з найважливіших моментів продукту є забезпечення можливість віртуальної присутності та активного участі у подіях, навіть якщо фізично люди знаходяться далеко один від одного. Така платформа сприятиме відновленню та зміцненню зв'язків між людьми в умовах сучасної нестабільності та важливості віддаленої спільності.

Крім того, враховуючи сучасні технології та потреби співтовариства, інтерфейс продукту повинен бути інтуїтивним та ефективним для користувачів будь-якої вікової категорії та будь-якого досвіду у використанні гаджетів та інтернету, який дозволить навіть бабусям та дідусям, у похилому віці, передавати своїм онукам приємні враженнями та висловлювати теплі слова через віртуальну присутність.

Також в контексті розробки проекту важливим аспектом є впровадження чату у РРЧ з розширеними можливостями відправки різноманітного контенту. Це обґрунтовано необхідністю встановлення надійного та швидкого зв'язку між замовником та виконавцем задачі. Реальний час дозволяє інтерактивно обмінюватися думками, реакціями та динамічно вмішуватись у виконання задачі на різних рівнях виконання, та змінювати її хід, напрямок, тривалість і якість, тим самим надаючи повний контроль замовнику над задачею створюючи відчуття, наче він і є виконавцем.

Ще одним важливим аспектом чату є можливість відправки фото, відео та інших файлів в межах чату важливо для розширення спектру виразних можливостей та можливостей комунікації. Відправлення зображень та відео дозволяє виконавцям краще розуміти, що відбувається і що від них вимагається, а замовникам – переконуватись у якості виконання задачі, детальнішому поглибленні у якусь непередбачену проблему та бути “Тут і зараз” в будь-якій точці світу о будь-якій годині доби.

Крім цього у проекті акцентується увага на технічних інноваціях, спрямованих на забезпечення найвищого рівня зручності та функціональності для користувачів. Платформа побудована з використанням передових технологій, що дозволяють забезпечити надійний та швидкий обмін повідомленнями, надійне виконання операцій з коштами, та надійне збереження даних для подальшого відтворення у зв’язку з різними причинами.

Також однією з найважливіших складових платформи є система коментарів як виконавців, так і замовників, оскільки вона сприяє взаємодії, довірі та взаєморозумінні між користувачами. Також система коментарів надає можливість користувачам висловлювати свої думки та враження від співпраці. Замовники, переглядаючи відгуки, можуть отримати об'єктивне уявлення про рівень професіоналізму та надійності виконавців. Рейтинг та відгуки формують довіру, що є критичним аспектом в прийнятті рішення, так як на основі цього, у замовника та виконавця формується перше відчуття довіри один до одного.

**1.2 Аналіз аналогів програмного продукту**

На момент розробки платформи, проведено ретельний аналіз та порівняння з існуючими аналогами, які спрямовані на подібне призначення, функціонал та мають схожу цільову аудиторію замовників. Цими програмами виявились Upwork, Fiverr та TaskRabbit, вже зарекомендували себе на ринку як визнані лідери у сфері фрілансу та надання послуг.

**Upwork** – це додаток, визнаний своєю орієнтацією на фріланс та надає можливість замовникам шукати виконавців для виконання різноманітних завдань, починаючи від програмування та дизайну до письмових послуг та перекладу. Платформа дозволяє створювати проекти, розміщувати завдання та взаємодіяти з фрілансерами через вбудований чат. Оплата за виконану роботу здійснюється через систему електронного платежу, що забезпечує безпечні та ефективні транзакції між замовниками та фрілансерами.

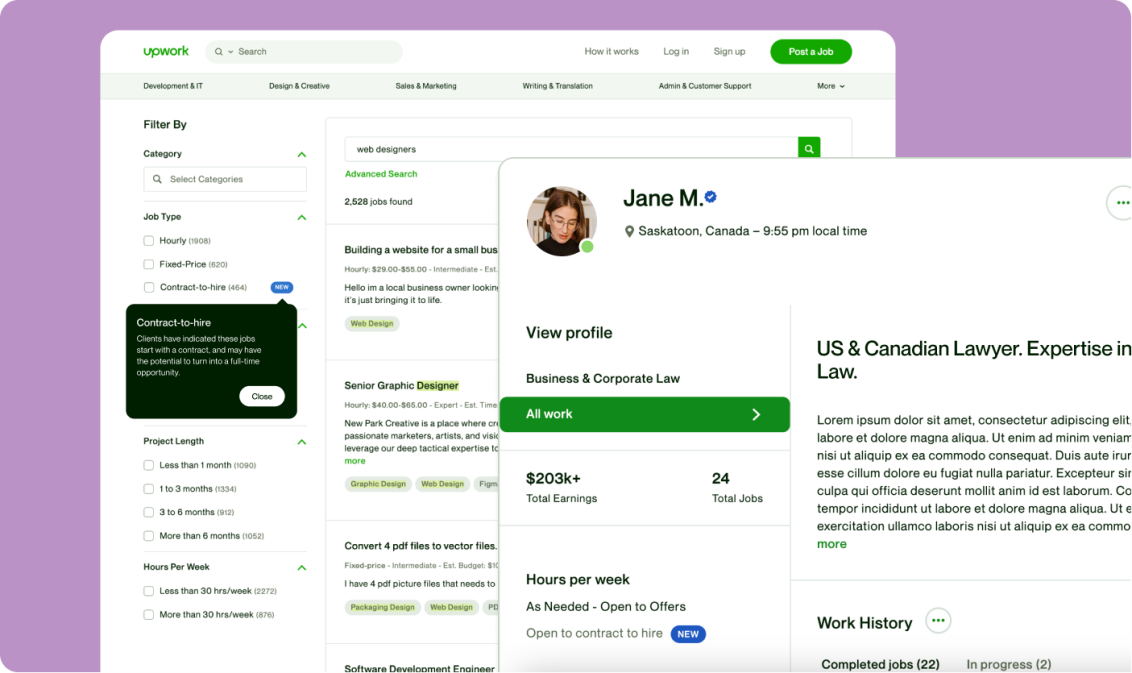


Рис. 1.1 – Робоча сторінка додатку Upwork

**Fiverr** – це онлайн-платформа, спрямована на надання послуг у різних областях. Замовники можуть знаходити фрілансерів, готових виконати конкретні завдання або надати конкретні послуги. Fiverr відомий своєю простою та інтуїтивно зрозумілою платформою, де користувачі можуть замовляти різноманітні послуги, від графічного дизайну до відеомонтажу. Оплата за послуги здійснюється через вбудовану систему оплати, забезпечуючи зручний та безпечний обмін коштами між замовниками та фрілансерами. Комунікація між клієнтом та виконавцем здійснюється через вбудовані інструменти спілкування, включаючи чатову систему та можливість обміну повідомленнями. Це створює ефективний та прозорий зв'язок, дозволяючи обговорювати деталі проекту, уточнювати вимоги та отримувати зворотний зв'язок для досягнення найкращого результату. Також на платформі є прозорий механізм відгуків, який допомагає іншим користувачам обирати виконавців на основі їхньої репутації та якості послуг. Крім того, важливою є можливість перегляду портфоліо виконавців, щоб забезпечити відповідність їхніх навичок та стилю вимогам конкретного проекту.

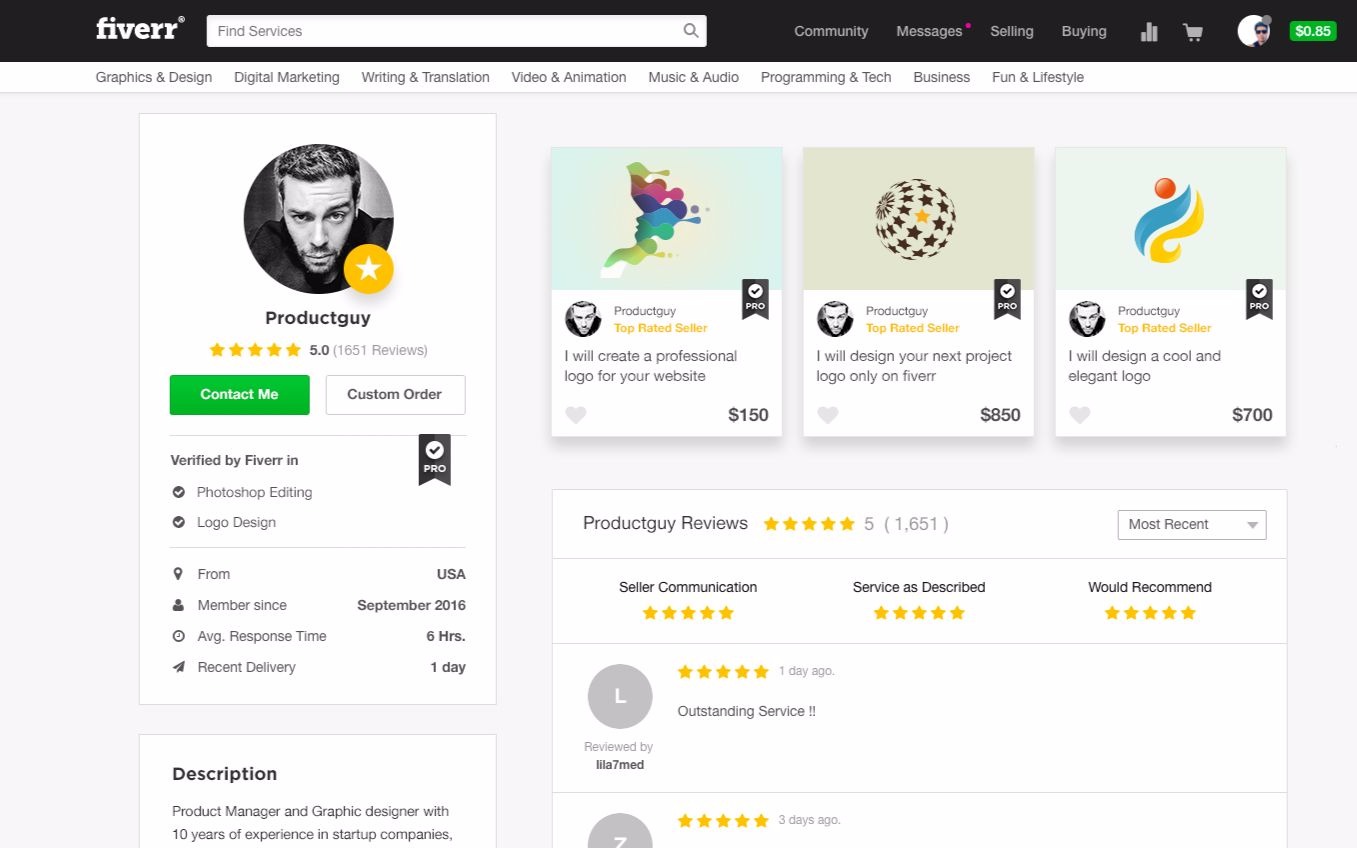


Рис. 1.2 – Робоча сторінка додатку Fiverr

**TaskRabbit** – це платформа, орієнтована на надання послуг у сфері реального життя та допомоги у вирішенні різноманітних повсякденних завдань. Замовники можуть шукати фахівців, готових виконати різні завдання, такі як прибирання, ремонт, переїзд та інші ділові послуги. Платформа дозволяє легко знаходити та спілкуватися з фахівцями, а оплата за виконані послуги також проводиться через систему електронного платежу, щоб забезпечити прозорий та безпечний процес транзакцій між сторонами. TaskRabbit визначається своєю акцентуацією на реальних потребах користувачів і підтримкою вирішення їхніх повсякденних завдань. Комунікація на платформі відбувається через будований чат, що дозволяє сторонам взаємодіяти, обговорювати деталі завдань, узгоджувати умови та надавати необхідні пояснення. Цей механізм сприяє ясному розумінню вимог та забезпечує ефективний обмін інформацією для успішного виконання завдань на платформі TaskRabbit. Крім цього на платформі існує система оглядів та оцінок, яка грає важливу роль у виборі виконавця для конкретного завдання.

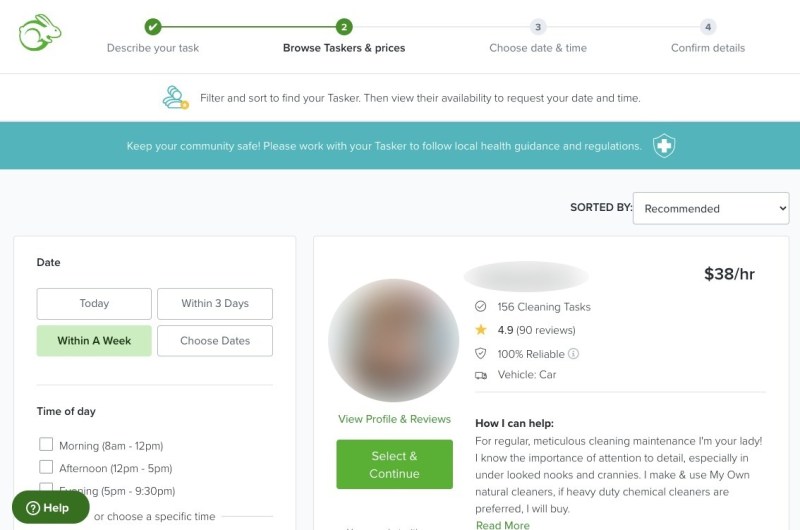


Рис. 1.3 – Робоча сторінка додатку TaskRabbit

В таблиці 1.1 наведена загальна характеристика і порівняння всіх

досліджуваних сайтів.

Таблиця 1.1

Порівняльна таблиця

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Upwork | Fiverr | TaskRabbit |
| Підсилюючі Функції | Рейтинг, глобальна аудиторія | Рейтинг, чітка категоризація послуг | Локальна орієнтація, рейтинг |
| Тип Виконавців | Глобальні фрілансери з різних країн | Індивіди та агентства, різноманітні категорії | Локальні виконавці для реальних завдань |
| Категорії Послуг | Програмування, дизайн, копірайтинг | Графічний дизайн, відеоролики, письменницькі послуги | Допомога в різних сферах життя |
| Система Оцінок та Відгуків | Є | Є | Є |
| Глобальний охоплення | Так | Так | Ні (Локально орієнтований) |
| Інтерфейс | Сучасний, зручний | Простий, з чіткою категоризацією | Легкий та зрозумілий |
| Адаптивність інтерфейсу | Так | Так | Так |
| Можливість Відправки Контенту | Файли | Фото, відео, файли | Фото, відео, файли |
| Вартість Послу | Різноманітна, визначається виконавцем | Різноманітна, визначається виконавцем | Залежить від виду та обсягу послуг |
| Зручність Використання | Так | Так | Так |

**1.3 Вибір архітектури веб-орієнтованої платформи**

Оскільки платформа повинна бути надійною, безпечною, конкурентно-спроможною, легкою та зрозумілою для користувачів різного віку та націлена існувати досить тривалий час, то вона повинна бути гарно побудована на архітектурному рівні. Це необхідно, так як відразу зробити додаток у найкращій своїй версіії – не можливо. Завжди потрібні додаткові тести, перевірки та новий функціонал. Щоб баги швидко виправлялись, а нові опції швидко додавались до платформи, потрібно правильно підібрати АПЗ.

Архітектура програмного забезпечення - це високорівневе структурне представлення програмної системи, яке визначає її ключові компоненти, їх взаємодії та організацію для досягнення визначених функціональних і якісних характеристик. Архітектура програмного забезпечення визначає фундаментальні аспекти системи, такі як структура, поведінка, взаємодія компонентів, а також розподіл ресурсів, забезпечуючи основу для подальшого розвитку, тестування і супроводження програмного продукту.

Під час аналізу можливих АПЗ, було зроблено висновок, що для платформи, спрямованої на обмін послуг та завдань між замовниками та виконавцями, використання клієнт-серверної АПЗ має ряд вагомих переваг.

По-перше, взаємодія із сервером. Клієнт-серверна архітектура дозволяє ефективно взаємодіяти із сервером, вимагаючи або передаючи дані лише тоді, коли це необхідно. Це дозволяє зменшити навантаження на мережу та забезпечити більш ефективний обмін інформацією.

По-друге, асинхронність. Клієнт-серверна архітектура може працювати в асинхронному режимі, що є важливим для сучасних веб-застосунків. Це дозволяє обробляти багато запитів одночасно, покращуючи продуктивність та відзначаючи систему як більш масштабовану.

По-третє, відокремлення інтерфейсу та логіки. Клієнт-серверна архітектура дозволяє чітко відокремити логіку додатку на стороні сервера від інтерфейсу користувача на стороні клієнта. Це розділення полегшує розробку, тестування та супровід, оскільки зміни в одній частині системи майже не впливають на іншу.

По-четверте, сприяння масштабованості: Клієнт-серверна архітектура забезпечує гнучкість у розгортанні, що сприяє масштабованості системи. Можна легко додавати нові клієнти чи розширювати функціонал серверу без значних змін в коді інших компонентів.

По-п’яте, більша безпека. Через те, що логіка додатку зберігається на сервері, це дозволяє ефективніше керувати безпекою та обробкою даних. Чутливі операції можна виконувати тільки на сервері, що зменшує ризик витоку чутливої інформації на клієнтську сторону.

По-шосте, підтримка різних клієнтів. Клієнт-серверна архітектура дозволяє реалізувати різноманітні клієнтські додатки для різних платформ (веб, мобільні пристрої, настільні додатки), обслуговуючи різні потреби користувачів.

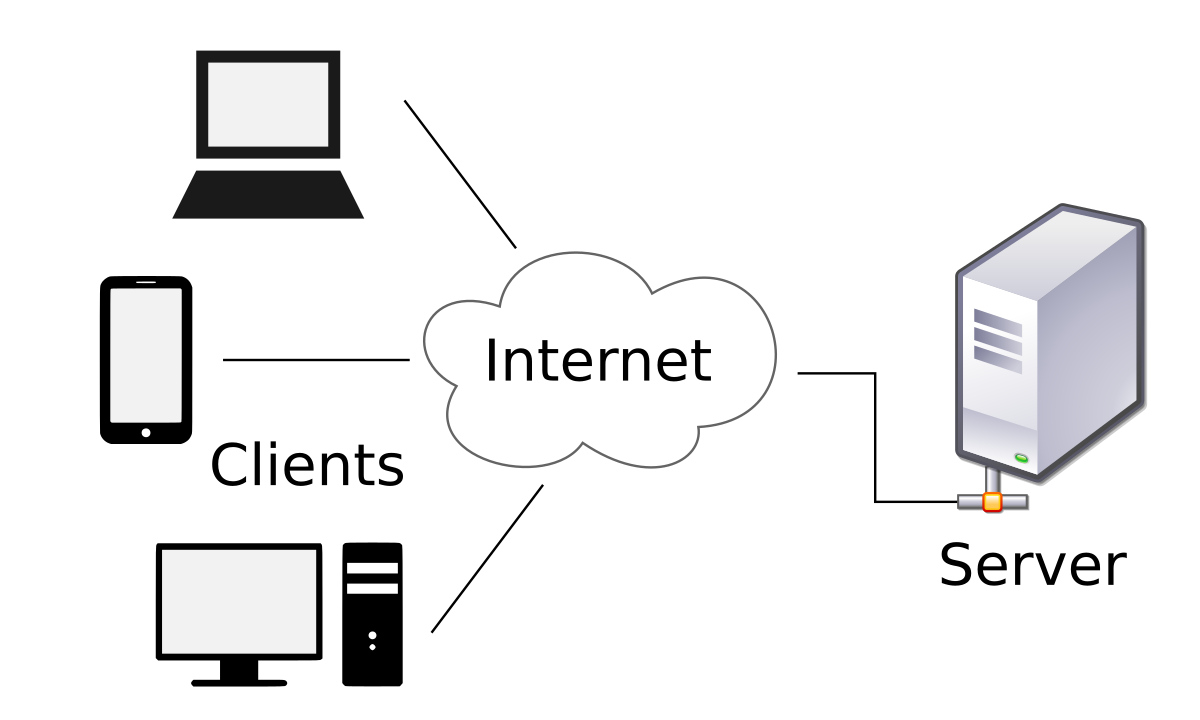


Рис. 1.4 – Візуалізації клієнт-серверної архітектури

Під час вибору основного патерну(патерну розробки), було закцентовано увагу на двох з них – на MVC (Model-View-Controller) та MVVM (Model-View-ViewModel). У результаті порівняння, було прийнято взяти саме MVC за основу, так як цей патерн простіший у реалізації та підтриці, не вимагає спеціальних бібліотек для реалізації та є не таким об’ємеим за рахунок меншої кількості компонентів.

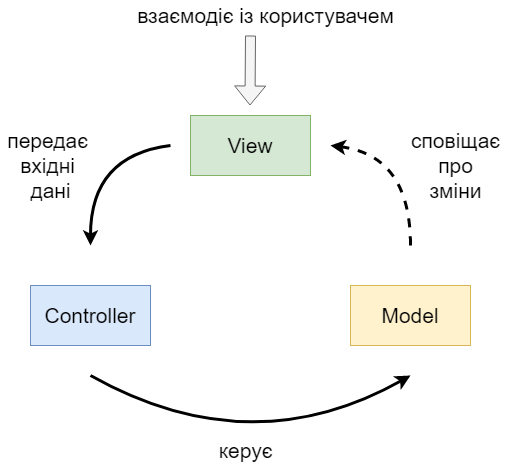


Рис. 1.5 – Візуалізації MVC патерну

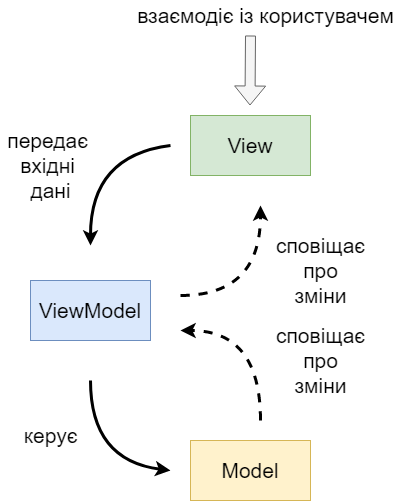


Рис. 1.6 – Візуалізації MVVM патерну

Крім цього, патерн MVC є дуже доцільним для проекту платформи обміну послуг та завдань. Ось кілька конкретних причин, чому цей патерн підходить для реалізації платформи:

- Розділення відповідальностей. MVC визначає три основні компоненти. Модель (Model), Вигляд (View) та Контролер (Controller). Це дозволяє чітко визначити та розділити відповідальності між різними частинами додатку. Модель відповідає за логіку додатку та роботу з даними, Вигляд - за представлення інформації користувачеві, а Контролер - за обробку введених користувачем дій.

- Легке розширення та зміни. Одна з переваг MVC полягає в тому, що зміни в одній частині системи майже не впливають на інші. Це забезпечує гнучкість та легкість у внесенні змін або розширенні функціоналу.

- Покращена організація коду. Використання MVC дозволяє розділити код на логічні шари, що полегшує організацію та управління кодовою базою.

- Підтримка різних інтерфейсів. Контролер дозволяє легко взаємодіяти з різними видами інтерфейсів, такими як веб-сторінки, мобільні додатки, API і т.д. Модель і Вигляд можуть залишатися практично незмінними, що спрощує розробку та супровід.

Для виконання етапу кодування обрано середовище розробки Visual Studio Code (VS Code), в якому буде проводитись подальша розробка додатку, використовуючи інструменти та мови програмування, такі як Node.js, React та MySQL. Visual Studio Code відомий своєю легкістю використання та розширюваністю.

VS Code — це інтегроване середовище розробки, яке надає інтелектуальний редактор з можливостями аналізу коду, автоматизованими засобами виявлення та виправлення помилок, а також широким спектром розширень для підтримки різних мов та технологій.

Проект розроблюватиметься за допомогою Node.js для серверної частини, React для клієнтського інтерфейсу та MySQL для управління базою даних. Використання VS Code сприяє зручній та ефективній розробці завдяки його інтуїтивному інтерфейсу та широкому спектру можливостей для роботи з кодом.

**1.4 Обгрунтування вибору засобів реалізації та вимоги до апаратного**

**забезпечення**

Обгрунтування вибору засобів реалізації та вимоги до апаратного забезпечення є критичним етапом у розробці будь-якого програмного продукту. Доцільність вибору конкретних технологій, мов програмування та апаратного забезпечення визначається широким спектром факторів, таких як функціональні вимоги продукту, потреби користувачів, вимоги до продуктивності, ефективність розробки та підтримки системи.

В даному проекті, буде використано Node.js для серверної частини проекту, так як він є відкритою та ефективною платформою, побудованою на движку V8 веб-переглядача Google Chrome. Його основна перевага - це здатність виконувати JavaScript на сервері, що робить його ідеальним вибором для розробки масштабованих веб-додатків.

Крім цього Node.js використовує асинхронність, що дозволяє виконувати багато операцій одночасно без очікування завершення кожної з них. Це особливо корисно для веб-серверів, які повинні ефективно обслуговувати багато запитів від користувачів одночасно. Для проекту це означає швидший обмін даними між клієнтом і сервером, зменшуючи час очікування для кінцевого користувача.

Також Node.js здатний легко масштабуватись горизонтально, додаванням нових серверів для обробки збільшеного навантаження. Це дозволяє підтримувати високу доступність та швидкість при зростанні кількості користувачів. Це дає можливість легко розширювати систему з ростом її популярності та користувацької бази.

Крім цього, одним з плюсів Node.js є велика та активна спільнота розробників, яка забезпечує підтримку та постійний розвиток платформи. Широкий вибір пакетів (за допомогою npm - менеджера пакетів Node.js) спрощує використання готових рішень та бібліотек для розширення функціональності проекту.

Також завдяки використанню JavaScript, Node.js робить можливим використання однієї мови програмування як на клієнтській, так і на серверній частині. Це спрощує взаємодію між розробниками, зменшує поріг входження в проект та зменшує потребу у використанні великої кількості розробників для підтрики коду та проекту у майбутньому.

Для клієнтської частини, в д аному проекті, буде використано React. React - це бібліотека JavaScript для розробки інтерфейсів користувачів, що дозволяє створювати високопродуктивні та ефективні односторінкові додатки (Single Page Applications, SPA).

Однією з переваг реакту є компонентна архітектура. Платформа, яка дозволяє людям допомагати один одному в різних задачах за грошові нагороди, може включати різноманітні функції та елементи інтерфейсу. React дозволяє розділити ці функції на компоненти, які можуть бути незалежно розроблені, тестовані та підтримувані. Це полегшує масштабування та утримання коду.

Також React може похвалитись своєю швидкодією, адже він використовує віртуальний DOM та ефективний алгоритм оновлення, що робить його особливо швидким для SPA. Користувачі будуть мати швидкий та плавний досвід взаємодії з платформою.

Крім цього React є однією з найпопулярніших та найбільш підтримуваних бібліотек для розробки інтерфейсів. Його активна спільнота забезпечує багатством ресурсів, документацією та готових рішень для різних аспектів розробки.

Також у реакта є свій синтаксиз. JSX полегшує опис інтерфейсу, надаючи можливість використовувати синтаксис, схожий на HTML, для створення компонентів. Це робить код більш зрозумілим та легким для розробки. Крім цього всього, у реакта дуже зручна взаємодія з backend. React можна легко інтегрувати з серверною частиною, написаною на Node.js. Це забезпечить гармонійну взаємодію між клієнтом та сервером на обох частинах вашого проекту.

Для збереження даних, даний проект, використовуватиме MySQL, так як вона є однією з найпоширеніших та добре вивчених реляційних систем управління базами даних. Вона вже довгий час використовується у великих проектах та корпоративних системах, що свідчить про її стабільність та надійність.

Також MySQL має широку спільноту користувачів, а також обширну документацію. Це забезпечує необхідними ресурсами для розробки, оптимізації та управління базою даних.

Крім цього, одним з найважливіших моментів є те, що MySQL є вільною та відкритою системою, що робить її доступною для використання без великих витрат. Це може бути важливим фактором, особливо для початкових етапів розробки та стартапів.

Також MySQL добре інтегрується з серверними застосунками, написаними на Node.js, завдяки наявності бібліотек та драйверів для Node.js, таких як "mysql" або "sequelize". Ці бібліотеки дозволяють виконувати SQL-запити, отримувати та взаємодіяти з даними швидко, зручно і безпечно, що дозволяє в майбутньому оновлювати та покращувати функціонал платформи.

Вимог до апаратного забезпечення небагато, проте вони є, адже навіть з гарно оптимізованим кодом, додаток буде просто набором символів, що не буде приносити жодної користі та задоволення.

Перш за все, для платформи потрібно чотирьохядерний або вище процесор. Так як такий процесор часто використовується для обробки паралельних запитів та оптимізації роботи серверної частини додатку. Це забезпечує ефективну роботу, особливо в умовах великого обсягу одночасних з'єднань.

Ефективна робота серверної частини додатку вимагає значної кількості оперативної пам'яті. Оперативна пам'ять використовується для тимчасового зберігання та швидкого доступу до активних процесів та даних. Використання обсягу оперативної пам'яті у розмірі 8 ГБ, або більше, дозволить серверу зручно та швидко обробляти великі обсяги інформації, таким чином покращуючи загальну продуктивність додатку. Також важливо мати на увазі, що сервер, який оптимально використовує ресурси оперативної пам'яті, може обслуговувати більше одночасних запитів та швидше реагувати на зміни в навантаженні.

Також, на сервері має бути SSD жорсткий диск. Використання SSD (Solid State Drive) є ключовим елементом для оптимізації швидкості операцій з даними на сервері. Оперативна система та серверні програми можуть швидше зчитувати та записувати дані на SSD, оскільки він не має рухомих частин, які спричиняють затримки у порівнянні з традиційним жорстким диском HDD (Hard Disk Drive). SSD також забезпечує високий рівень надійності та тривалості роботи, оскільки він менше схильний до механічних поломок, що робить його ідеальним вибором для систем, які працюють без перерв, таких як сервери. Пришвидшення швидкості доступу до даних сприяє зменшенню часу очікування для клієнтів та покращенню загального користувацького досвіду. Так як платформа буде зберігати велику кількість даних, фото та відео-матеріалів, то краще всього, щоб розмір диску був більшим за 500 ГБ.

**Висновки до першого розділу**

У першому розділі дипломної роботи детально проаналізовано особливості сучасних платформ, подібних за ціллю та функціоналом до розробляємої, визначено чітку мету та поставлені завдання проекту.

Також визначено середовище програмного забезпечення для розробки сайту та визначено набір інструментів, які будуть використовуватися. Зокрема, вибір був зроблений на користь Node.js для серверної частини проекту та React для клієнтської. Цей вибір обгрунтовано їх ефективністю, швидкодією та можливістю легкої масштабованості.

У третьому підрозділі нашої дипломної роботи була проведена глибока робота над архітектурними аспектами платформи. Увага була зосереджена на визначенні ефективної та добре структурованої архітектури, яка відповідає потребам нашого проекту. З цією метою було обрано паттерн проектування MVC (Model-View-Controller), який дозволяє чітко розділити логічні компоненти системи.

Таким чином, перший розділ проекту встановлює ключові принципи та фундаментальні рішення для подальшої розробки та реалізації платформи, визначаючи її стратегічний напрямок та основні технічні параметри.