

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ
СТВОРЕННЯ ВІДДАЛЕНИХ ЗАВДАНЬ
SOFTWARE FOR REMOTE TASKS CREATION

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

САВЕЛЬСВ Б.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
« » 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

САВЕЛЬЄВА Богдана Юрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для створення віддалених завдань. Software for Remote Tasks Creation

керівник проєкту (роботи) к.т.н., доцент, Шитікова Олена Вікторівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «28» квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ШИТІКОВА О.В., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання «14» квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Богдан САВЕЛЬСВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олена ШИТІКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
117 с., 14 табл., 50 рис., 4 дод., 20 джерел.

ВІДДАЛЕНІ ЗАВДАННЯ, РОЗРОБКА, МОВА ПРОГРАМУВАННЯ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, NODEJS, БАЗА ДАНИХ, ВІДДАЛЕНИЙ
СЕРВЕР, DOCKER, МОВА JAVASCRIPT, SQL.

Об'єкт дослідження – процес програмної реалізації застосунку для створення віддалених завдань.

Предмет дослідження – програмні засоби для реалізації застосунку для створення віддалених завдань.

Метою роботи є програмна реалізація застосунку для створення віддалених завдань.

Матеріали, методи та технічні засоби: мови програмування HTML, CSS, JavaScript та фреймворків React, Node.js.

Результати. Розроблено програмне забезпечення для створення віддалених завдань за допомогою мов програмування HTML, CSS, JavaScript та фреймворків React, Node.js.

Практична цінність роботи полягає у програмній реалізації застосунку для створення віддалених завдань у вигляді WEB-застосунку.

Висновки. Реалізовано застосунок для створення віддалених завдань у вигляді WEB-застосунку.

Галузь використання – користувачі інтернету.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 117 pages, 14 tables, 50 figures, 4 appendixes, 20 sources.

REMOTE TASKS, DEVELOPMENT, PROGRAMMING LANGUAGE, SOFTWARE, NODEJS, DATABASE, REMOTE SERVER, DOCKER, JAVASCRIPT LANGUAGE, SQL.

The object of the study is the process of software implementation of an application for creating remote tasks.

The subject of the study is software tools for implementing an application for creating remote tasks.

The purpose of the work is software implementation of an application for creating remote tasks.

Materials, methods and technical means: programming languages HTML, CSS, JavaScript and frameworks React, Node.js.

Results. Software for creating remote tasks has been developed using programming languages HTML, CSS, JavaScript and frameworks React, Node.js.

The practical value of the work lies in the software implementation of an application for creating remote tasks in the form of a WEB application.

Conclusions. An application for creating remote tasks in the form of a WEB application has been implemented.

Field of use - Internet users.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	12
1.1 Поняття та принципи виконання віддалених завдань	12
1.2 Основні відмінності виконання віддалених завдань від повної зайнятості	14
1.3 Категорій виконавців віддалених завдань.	16
1.4 Аналіз існуючих платформ для виконання віддалених завдань	23
1.5 Висновки за розділом 1	32
2 Матеріали і методи	33
2.1 Вибір мови програмування	33
2.2 Вибір фреймворку для розробки Frontend частини вебсайту	40
2.3 Вибір фреймворку для розробки Backend частини вебсайту	54
2.4 Висновки за розділом 2	61
3 Опис програми	62
3.1 Структура програмного забезпечення	62
3.2 Розробка бази даних	64
3.3 Основний алгоритм роботи програмного забезпечення	70
3.4 Проєктування інтерфейсу	72
3.5 Висновки за розділом 3	73
4 Експлуатація, тестування та експериментальне дослідження програми	74
4.1 Призначення й умови застосування програми	74
4.2 Умови виконання програми	75
4.3 Виконання програмного забезпечення	75
4.4 Тестування програмного забезпечення	83
4.5 Висновки за розділом 4	84
Висновки	85
Перелік джерел посилання	87

Додаток А Технічне завдання.....	89
Додаток Б Опис програми.....	92
Додаток В Текст програми	96
Додаток Д Слайди презентації	111

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JS – JavaScript;

MVC – model-view-controller;

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних.

ВСТУП

У XXI столітті цифрові технології стрімко трансформують усі сфери людської діяльності. Розвиток інформаційного простору, інтернету, мобільних технологій та хмарних сервісів сприяв зміні традиційних підходів до організації праці, освіти, керування проєктами та повсякденного життя. Однією з ключових змін стало активне впровадження віддаленого формату роботи. З кожним роком дедалі більше організацій, установ та індивідуальних користувачів віддають перевагу дистанційній взаємодії як ефективному, гнучкому та економічно вигідному способу реалізації робочих процесів

У таких умовах особливу роль відіграє програмне забезпечення, що забезпечує постановку, контроль та виконання завдань у дистанційному режимі. Відповідні інструменти дозволяють не лише організовувати командну роботу без фізичної присутності учасників, але й підтримувати високий рівень прозорості, продуктивності та відповідальності. Програмне забезпечення для створення віддалених завдань стає критично важливим компонентом цифрової інфраструктури, без якого неможливо уявити ефективне функціонування сучасного підприємства, стартапу, навчального закладу чи навіть індивідуального проєкту [1]-[3].

Особливої актуальності такі програмні рішення набули в період глобальних викликів, зокрема пандемії COVID-19, яка спричинила масовий перехід на дистанційну роботу та навчання. Також події, пов'язані з військовими конфліктами та вимушеною міграцією, стали каталізатором для розвитку інструментів віддаленого керування завданнями, що не залежать від географічного розташування користувачів. Відтак, попит на надійне, адаптивне й зручне програмне забезпечення в цій галузі продовжує зростати.

Програмне забезпечення для створення віддалених завдань охоплює широкий спектр функцій: від створення простих списків справ до комплексного управління проєктами з розподілом ролей, контролем дедлайнів, системами коментарів та звітності. Успішна реалізація таких

рішень потребує поєднання сучасних технічних засобів і методів розробки, врахування потреб цільової аудиторії, забезпечення безпеки даних та зручності користувацького інтерфейсу [1]-[3].

Сьогодні на ринку вже існує велика кількість готових сервісів і платформ, що пропонують функціонал для роботи із завданнями. Водночас багато з них мають обмеження у налаштуванні, закритий код, високу вартість або потребують значного часу на адаптацію. Саме тому створення власного вебзастосунку для управління віддаленими завданнями залишається актуальним завданням для розробників, студентів, дослідників та стартап-команд, які прагнуть створити індивідуальне рішення, що повністю відповідатиме конкретним вимогам [1]-[3].

Важливо зазначити, що подібне програмне забезпечення має бути орієнтоване на максимальну доступність. Використання вебтехнологій дозволяє забезпечити кросплатформеність – тобто можливість роботи на будь-якому пристрої, незалежно від операційної системи. Завдяки цьому користувачі можуть працювати зі своїми завданнями з комп'ютера, планшета чи смартфона – у будь-який час і з будь-якої точки світу [1]-[3].

Ще однією важливою особливістю є масштабованість та гнучкість системи. Важливо передбачити можливість розвитку застосунку: розширення функціоналу, інтеграцію з іншими сервісами (наприклад, поштовими клієнтами, календарями, месенджерами), реалізацію систем сповіщень та адаптацію під різні сценарії використання – від особистого планування до командної роботи в корпоративному середовищі [1]-[3].

Таким чином, розробка програмного забезпечення для створення віддалених завдань є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Дослідження цієї теми спрямоване на розуміння основних вимог до подібних систем, вибір оптимальних технологій для реалізації, проєктування зручного та функціонального інтерфейсу, а також створення надійного вебзастосунку, який відповідатиме сучасним вимогам користувачів. Результатом такої роботи є програмний продукт, що має практичне значення і потенціал до подальшого розвитку в умовах зростаючої цифровізації суспільства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та принципи виконання відалених завдань

Економіка концертів швидко зростає, оскільки все більше професіоналів використовують свої навички як виконавці відалених завдань (фрілансери). Дослідження Upwork виявило, що у 2023 році 64 мільйони професіоналів займалися фрілансом [4]-[5].

Виконання відалених завдань (фріланс) – це виконання конкретної роботи для клієнтів без повної зайнятості. Фрілансери часто беруться за кілька проєктів з різними клієнтами одночасно.

Для деяких фріланс – це побічне заняття. Для інших це все, що вони роблять. Хоча це не повна зайнятість в одного роботодавця, фріланс може стати малим бізнесом і забезпечити повний робочий день [4]-[5].

Фрілансери працюють за контрактом на неповний або повний робочий день і часто підписують угоди перед початком проєктів.

При фрілансі клієнт платить за проєкт, за завдання або за годину, залежно від угоди. Фрілансні проєкти зазвичай включають короткострокові завдання, хоча задоволені клієнти часто вимагають додаткової роботи. Більшість позаштатних вакансій доступні в сферах навичок, послуг і творчості, таких як копірайтинг, програмування, інженерія та маркетинг.

47% фрілансерів надають послуги, засновані на знаннях, включаючи маркетинг і програмування [4]-[5].

Найбільш високооплачувані ролі. Менеджери зі зв'язків з громадськістю (PR) і бізнес-консультанти є одними з найбільш високооплачуваних позаштатних ролей: PR-менеджери отримують від 50 до 100 доларів на годину, а бізнес-консультанти – від 28 до 98 доларів на годину [4]-[5].

Середня погодинна ставка для фрілансерів у Північній Америці становить 44 долари США .

Третина фрилансерів витрачають 10-20 годин на тиждень на концертну роботу. 72% фрилансерів стверджують, що головною причиною фрилансу є контроль над своїм фінансовим майбутнім. Прогнози показують, що до 2027 року понад 50% робочої сили будуть фрилансерами [4]-[5].

Переваги фрилансу:

- гнучкість. Як фрилансер можна самостійно обирати години та вирішувати, коли та де працювати. Наприклад, можна працювати вдома, або на пляжі. Гнучкість фрилансу допомагає сприяти кращому балансу між роботою та особистим життям;

- вибір клієнтів і проєктів. Робота фрилансером дозволяє обирати проєкти, які відповідають навичкам та інтересам фрилансера. Також можна вибрати клієнтів, які відповідають певним критеріям;

- встановлення власних ставок. Можна насолоджуватися більшим контролем над своїми потенційними прибутками, оскільки зазвичай фрилансер сам встановлює ставки. Обсяг роботи, яку фрилансер може виконати щодня, або щотижня, є ще одним фактором, який, ймовірно, визначатиме заробіток;

- покращений набір навичок. Фрилансер може мати можливість працювати над більш спеціалізованими проєктами, що дасть більше досвіду в тематичних галузях;

- вплив світових брендів. Оскільки фрилансер може обирати своїх клієнтів і проєкти, у нього є можливість розвивати значущі відносини з компаніями з усього світу.

Недоліки фрилансу:

- ізоляція. На відміну від роботи на неповний або повний робочий день, фрилансерський бізнес може бути самотнім, особливо якщо фрилансер працює віддалено;

- невизначена безпека роботи. Фрилансерська робота, ймовірно, буде нестабільною. Можливо, фрилансеру доведеться продовжувати працювати на звичайній роботі, щоб підтримувати стабільний дохід на ранніх етапах кар'єри фрилансера;

– адміністративні обов'язки. Фрілансеру доведеться займатися юридичною документацією, виставляти рахунки, маркетинг та виконувати інші відповідні адміністративні обов'язки. Ці обов'язки можуть вимагати значних знань у сфері бухгалтерського обліку, податкового законодавства щодо самозайнятості та офісних технологій, а також відіймати час від основної сфери уваги;

– немає пільг, які фінансуються роботодавцем. Як само зайнята особа, у фрілансера немає роботодавця, який би сплачував внески на медичне страхування, пенсії 401 (k) та інші пільги, які часто пов'язані з роботою на повний робочий день [4]-[5].

1.2 Основні відмінності виконання віддалених завдань від повної зайнятості

Вибір між фрілансом і повною зайнятістю є дуже особистим. В таблиці 1.1 наведено порівняння роботи на фрілансі та повної зайнятості [5].

Таблиця 1.1 – Порівняння роботи на фрілансі та повної зайнятості

	Фріланс	Повна зайнятість
Гнучкість	Свобода працювати будь-де в будь-який час, залежно від уподобань чи обставин. І можна вибрати своїх клієнтів і проекти.	Робота на повний робочий день означає, що робітник виконує певні обов'язки протягом встановленого робочого часу та має певні обов'язки.
Стабільність	На відміну від повної зайнятості, фрілансерська робота може бути дещо непередбачуваною, що ускладнює планування майбутнього.	Працівники, які працюють повний робочий день, зазвичай мають стабільний, передбачуваний дохід і графік роботи, що полегшує планування майбутніх фінансових питань.

Продовження таблиці 1.1

	Фріланс	Повна зайнятість
Переваги	Незалежні працівники зазвичай не мають доступу до фінансованих роботодавцем пільг, таких як оплачувана відпустка, пенсійні плани, медичне страхування та лікарняні.	Такі переваги, як оплачувана відпустка, медичне страхування та пенсійні плани, є значним плюсом повної зайнятості.
Мотивація	Фрілансер працює як сам собі бос, тому самотивація є необхідністю.	Як штатний працівник повинен відповідати певним стандартам продуктивності, щоб отримати підвищення по службі.
Заробіток	Фрілансери мають більший контроль над своїм потенціалом заробітку, оскільки вони самі встановлюють тарифи та години роботи.	Заробітки штатного працівника зазвичай визначаються структурою заробітної плати та політикою роботодавця.
Набір навичок	Незалежні підрядники мають можливість, або спеціалізуватися в конкретній галузі, або отримати досвід і навички в різних предметних областях.	На відміну від фрілансерів, штатні працівники можуть не мати такого ж рівня гнучкості для розвитку конкретних навичок, оскільки їхні посадові ролі та обов'язки в основному визначаються роботодавцем.
Досвід	Фрілансери мають можливість працювати з кількома брендами по всьому світу, щоб розширити свою професійну мережу та створити різноманітне портфоліо.	На відміну від фрілансерів, працівники, як правило, обмежені роботою в одній компанії, що може обмежити їх вплив на різні галузі чи клієнтів.

1.3 Категорій виконавців віддалених завдань

Однією з найкращих рис економіки є те, що ви можна працювати в різних секторах, вибраних відповідно до особистого набору навичок [6]-[7]. Нижче наведено деякі популярні вакансії для незалежних підрядників:

а) розробка в ІТ. Нижченаведені посади в галузі розвитку та ІТ передбачають розробку та підтримку технологічних систем і цифрового програмного забезпечення, які забезпечують роботу організацій [6]-[7]:

1) веброзробник. Веброзробники пишуть код для макетів вебсайтів, які створюють повністю функціональні сайти. Веброзробники відповідають за створення, підтримку та масштабування вебсайтів клієнтів відповідно до конкретних вимог [6]-[7];

2) UX дизайнер. Розробники взаємодії з користувачем допомагають забезпечити безперебійну роботу в Інтернеті, розробляючи зручні елементи вебсайту, програмного забезпечення або мобільного додатка. Вони пропонують рішення щодо зручності використання на основі обширних досліджень і аналізу конкурентів [6]-[7];

3) програміст. Програмування – це популярна сфера, яка включає роботу з комп'ютером і розробку програмного забезпечення. Ця високотехнічна сфера є хорошим вибором, якщо ви хочете заробляти на життя написанням коду;

4) інженер машинного навчання (ML). Інженери ML відповідають за створення та розгортання моделей AI (штучного інтелекту), які можуть аналізувати дані та робити прогнози. Наприклад, вони можуть розробити алгоритм, який може розпізнавати зображення об'єктів, таких як автомобілі чи тварини, і класифікувати їх за різними категоріями. Для цієї роботи потрібні знання інформатики та знання мов програмування, таких як Python і R [6]-[7];

5) аналітик даних. Аналітики даних збирають і аналізують дані, щоб виявити інформацію, яка може стати основою для прийняття бізнес-рішень. Вони використовують такі інструменти, як SQL і Excel , щоб очищати та впорядковувати набори даних. Вони працюють у багатьох галузях, таких як охорона здоров'я та маркетинг [6]-[7];

6) дослідник ШІ. Дослідник штучного інтелекту аналізуватиме дані та планує експерименти для розробки нових алгоритмів і моделей для вдосконалення систем штучного інтелекту. Ця роль вимагає ґрунтовних знань з інформатики, математики та статистики [6]-[7];

б) дизайнерсько-креативні працівники. Незалежні творчі професіонали створюють візуальні елементи, продукти, щоб привернути увагу користувачів [6]-[7]:

1) графічний дизайнер. Фрілансер графічний дизайнер допомагає клієнтам створити привабливі логотипи, корпоративну літературу або елементи брендування вебсайту;

2) редактор відео. Залежно від обсягу проєкту відео редактор може виконувати базове або більш повне редагування. Загальні обов'язки включають об'єднання графіки, аудіо та ефектів для забезпечення переконливого кінцевого продукту [6]-[7];

3) дизайнер сайтів. Роль вебдизайнера, полягає у створенні та підтримці високоякісних вебсайтів. Він допомагає клієнтам приймати правильні рішення під час вибору тем, інтеграції плагінів або вибору систем керування вмістом (CMS). Крім того, його робота – оновити код і об'єднати елементи дизайну, наприклад банери та графіку [6]-[7];

4) фотограф. Фотографи-фрілансери співпрацюють з організаціями, щоб зробити найкращі фотографії для маркетингу, вмісту вебсайту та інших бізнес-цілей. Фотографів можуть найняти для фотозйомки під час весіль, днів народжень та інших особливих подій;

5) відеоператор. Відеоператор спеціалізується на створенні відеоконтенту, наприклад для маркетингу чи освіти. Вони знімають

високоякісне відео за допомогою камер та іншого обладнання, а потім редагують його для створення кінцевого продукту [6]-[7];

в) збут і маркетинг. Компанії повинні залишатися конкурентоспроможними та охоплювати цільову аудиторію, а позаштатні спеціалісти з продажу та маркетингу користуються великим попитом:

1) маркетолог соціальних мереж. Ця роль передбачає створення цікавих публікацій у соціальних мережах, керування обліковими записами соціальних мереж і моніторинг коментарів і повідомлень. Вони також аналізують ефективність своїх маркетингових кампаній, щоб оптимізувати їх для максимального залучення та охоплення [6]-[7];

2) маркетинговий стратег. Клієнти наймають маркетингових стратегів, щоб допомогти розробити програми охоплення клієнтів і рекламні кампанії, які приносять результати. Співпраця з досвідченим позаштатним експертом з маркетингу може полегшити досягнення цілей продажів;

3) менеджер по роботі з клієнтами. Менеджер по роботі з клієнтами гарантує, що якість обслуговування відповідає очікуванням клієнтів. Час очікування клієнтів є частиною мінімальних стандартів обслуговування, які контролює менеджер. Цей професіонал-фрілансер також встановлює цілі обслуговування клієнтів і створює план розвитку персоналу [6]-[7];

4) творець контенту. Творці контенту створюють цікавий контент для соціальних медіа, вебсайтів, блогів і відеоканалів, використовуючи текст, зображення та відео, щоб ділитися цінною інформацією або рекламувати продукт;

5) SEO спеціаліст. Ця посада передбачає оптимізацію вебсайтів для покращення їх видимості на сторінках результатів пошуку. Цей процес передбачає проведення дослідження ключових слів, створення зворотних посилань і аналіз трафіку вебсайту, щоб збільшити кількість конверсій і продажів для бізнесу [6]-[7];

г) письмо та переклад. Ця категорія включає ряд робіт, пов'язаних із створенням письмового вмісту та перекладом матеріалів на різні мови. Ці ролі вимагають виняткових навичок спілкування та уваги до деталей [6]-[7]:

1) копірайтер. Професіонал із копірайтингу може нести відповідальність за створення інформаційних бюлетенів, вебсайтів, публікацій у соціальних мережах та інших типів вмісту. Його робота – писати привабливий вміст, який навчає та надихає читачів. Деякі письменницькі роботи зосереджені на створенні вмісту, опублікованого в друкованих ЗМІ [6]-[7];

2) автор контенту. Ці автори створюють навчальний, освітній або розважальний вміст, який інформує читача та залучає його. Вони надають письмові матеріали для вебсайтів, соціальних мереж та інших цифрових платформ. Їхній підхід відрізняється від підходу копірайтерів, оскільки вони зазвичай зосереджуються на створенні більш переконливого контенту [6]-[7];

3) технічний письменник. Ця роль передбачає створення посібників користувача, посібників з експлуатації та технічних звітів, які допомагають користувачам зрозуміти складну технічну інформацію. Вони співпрацюють із експертами, щоб переконатися, що вміст точний і легкий для розуміння;

4) редактор. Позаштатне редагування передбачає перевірку вмісту на наявність друкарських помилок, граматичних помилок і загальне дотримання стислого змісту та керівництва по стилю. Він також відповідає за перевірку фактів перед подачею готових документів клієнтам [6]-[7];

5) перекладач. Для точного перетворення тексту з однієї мови на іншу наймають позаштатних перекладачів. Перекладений вміст може з'являтися на вебсайтах, у маркетингових матеріалах і посібниках із продуктів;

д) адміністративна та клієнтська підтримка. Ця категорія пропонує широкий вибір можливостей роботи в багатьох функціональних сферах. Фрілансери в цих ролях підтримують безпосередньо клієнта, або працюють із системами, пов'язаними з потребами клієнта та транзакціями:

1) координатор по роботі з клієнтами. Може процвітати як позаштатний координатор обслуговування клієнтів, якщо у нього є чудові навички спілкування. Його обов'язки можуть включати забезпечення того, щоб клієнти отримували ефективне обслуговування, проведення опитувань щодо якості та розгляд скарг клієнтів [6]-[7];

2) менеджер бази даних. Роль менеджера БД полягає в управлінні базами даних клієнтів. Він несе відповідальність за безпеку та впорядкованість конфіденційної інформації;

3) менеджер електронної комерції. Популярність онлайн-шопінгу сприяє збільшенню потреби в досвідчених менеджерах електронної комерції. Він відповідає за керування запасами, нагляд за обслуговуванням сайту та оптимізацію маркетингових стратегій [6]-[7];

4) віртуальний помічник. Ця роль передбачає надання віртуальної адміністративної підтримки окремим особам або підприємствам. Це включає такі завдання, як планування зустрічей, організація поїздок і проведення досліджень. Ці завдання допомагають клієнтам заощадити час і підвищити продуктивність;

5) керівник проєкту. Керівник проєкту несе відповідальність за управління графіками, фінансами та конфліктами, щоб забезпечити виконання проєктів вчасно та в межах бюджету. Керівники проєктів використовують свої лідерські та комунікаційні навички, щоб керувати командами та зацікавленими сторонами протягом усього проєкту [6]-[7];

е) фінанси та бухгалтерський облік. Багато клієнтів шукають фінансових експертів, які б допомогли їм організувати записи, прийняти

обґрунтовані фінансові рішення та керувати податками (зокрема, керувати сплатою податків із заробітної плати та подавати податкові декларації):

1) бухгалтер. Клієнти наймають позаштатних бухгалтерів, щоб вони допомагали готувати звіти, керувати фінансовою інформацією та складати бюджети. Багато проєктів також передбачають підготовку податкових декларацій, переважно для стартапів [6]-[7];

2) фінансовий радник. Фінансовий консультант тісно співпрацює з клієнтами, щоб допомогти їм керувати особистими чи діловими фінансами. Клієнти очікують надійних порад і відповідності на основі досвіду та поточних досліджень;

3) інвестор. Багато клієнтів шукають незалежних інвестиційних професіоналів, щоб отримати експертну пораду перед прийняттям критичних рішень. У багатьох випадках експерт аналізує власний капітал цільової компанії, щоб визначити її фінансовий стан, перш ніж надавати рекомендації клієнту [6]-[7];

ж) людські ресурси. Фахівці з кадрів:

1) менеджер по зарплаті. Позаштатні менеджери з розрахунку заробітної плати допомагають клієнтам із функціями розрахунку заробітної плати та забезпечують ефективну та точну обробку [6]-[7];

2) рекрутер. Позаштатний рекрутер відповідає за перевірку кандидатів на роботу та розробку стратегій найму. Він може працювати з кількома клієнтами одночасно, щоб допомогти їм найняти своїх ідеальних кандидатів;

3) загальний експерт з кадрів. Клієнти наймають позаштатних експертів з кадрів (HR) для допомоги в управлінні розвитком працівників. У деяких випадках проєкти передбачають встановлення кадрової політики та процедур;

з) право:

1) юрист. Фрілансні проєкти для юристів дозволяють працювати з компаніями, фізичними особами та юридичними фірмами.

Залежно від їх спеціальності вони можуть допомагати клієнтам у корпоративних правових питаннях або сімейних справах [6]-[7];

2) віддалений секретар. Віддалений секретар, ви може працювати над проєктами, пов'язаними з різними адміністративними обов'язками, такими як планування зустрічей, надсилання рахунків-фактур і ведення кореспонденції;

3) помічник юриста. Клієнти наймають позаштатних помічників юристів для допомоги в складанні судових документів і роботі з електронними документами. Додаткові обов'язки можуть включати патентну роботу, управління документами та електронне відкриття (форма цифрового розслідування) [6]-[7];

и) інженерія та архітектура. Категорія «Інженерія та архітектура» включає роботи, пов'язані з проєктуванням і будівництвом конструкцій, виробів і систем. Вони можуть працювати в різних місцях, включаючи будівельні фірми та державні установи [6]-[7]:

1) архітектор. Багато клієнтів публікують проєкти, пов'язані з проєктуванням будівель і поданнями дозволів. Архітектор може вибрати проєкти, що включають структурні, або цивільні проєкти, залежно від досвіду та набору навичок [6]-[7];

2) дизайнер інтер'єру. Дизайнер інтер'єру, дає поради клієнтам, щодо розташування меблів і варіантів матеріалів. Також він може використовувати цифрові інструменти для віртуальної візуалізації [6]-[7];

3) віддалений інженер-механік. Незалежні інженери-механіки допомагають клієнтам проєктувати та розробляти машини, датчики та інструменти. Деякі проєкти передбачають усунення механічних несправностей і ремонт [6]-[7];

4) 3D модельєр. 3D-моделіст створює тривимірні моделі об'єктів за допомогою спеціалізованого програмного забезпечення. Вони тісно співпрацюють з іншими професіоналами, включаючи

дизайнерів ігор, арт-директорів і архітекторів, щоб переконатися, що моделі відповідають бажаним характеристикам [6]-[7].

1.4 Аналіз існуючих платформ для виконання віддалених завдань

Проаналізуємо програмне забезпечення для виконання віддалених завдань. На сьогоднішній день існує велика кількість подібних платформ, кожна з яких містить свої плюси та мінуси [8]-[9].

Fiverr – це платформа, яка об’єднує фрілансерів із власниками бізнесу, які шукають послуги в різних цифрових проєктах, включаючи дизайн вебсайтів, написання контенту та озвучення. Крім того, це один із найкращих вебсайтів фрілансерів для студентів (рисунок 1.1) [8].

Тип фрілансу: цифровий маркетинг, веброзробка, соціальні мережі.

Плюси: різноманітна категорія, безкоштовна реєстрація, онлайн-курси.

Мінуси: висока комісія, тривалий процес оплати.

Цей вебсайт для фрілансерів забезпечує спрощений спосіб взаємодії як для фрілансерів, так і для власників бізнесу. Зазначимо, що на сайті фрілансери називаються продавцями, їхні послуги – гігами, а власники бізнесу – баєрами [8]-[9].

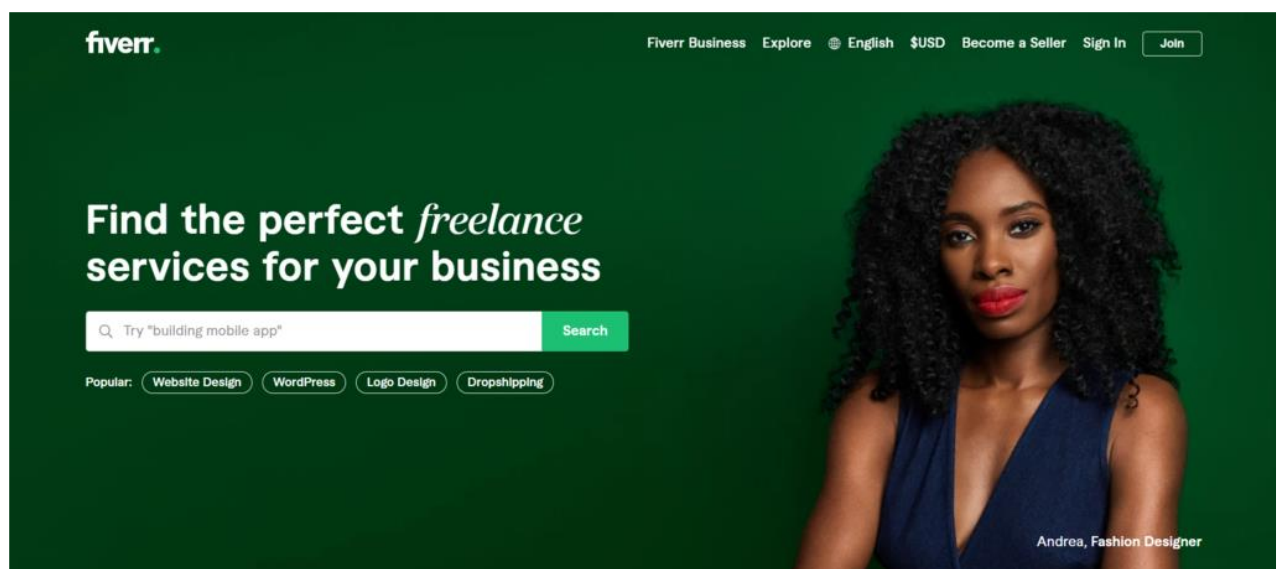


Рисунок 1.1 – Платформа Fiverr [8]

Замість того, щоб розсилати ставки, продавці можуть безкоштовно зареєструватися, вказати список своїх концертів і встановити ціни. Тим часом покупці можуть просто переглядати різні категорії, щоб зробити покупку.

Щоразу, коли покупець купує пропозицію, система автоматично стягує плату з його рахунку, а гроші призупиняє. Потім система візьме комісію та надішле фрілансерам 80% від запропонованої ціни [8]-[9].

Залежно від рівня продавця кожного фрілансера, виведення прибутку може тривати до 14 днів. Є кілька варіантів зняття платежів, наприклад дебетова картка, PayPal або банківський переказ [8]-[9].

Fiverr відкриває світ потенційних продажів із 3,42 мільйонами активних покупців. Немає погодинної ставки; можна визначити ціну на основі проєкту.

Продавцям гарантовано захищені платежі за допомогою сертифіката SSL, який захищає платіжну та особисту інформацію. А також цілодобова служба підтримки клієнтів, яка завжди готова допомогти [8]-[9].

Toptal – це вебсайт позаштатних спеціалістів, який об'єднує висококваліфікованих експертів у галузі позаштатних спеціалістів із компаніями. Toptal має широкий спектр фрілансерів, від веб розробників і вебдизайнерів до фінансових консультантів і менеджерів із продуктів (рисунок 1.2) [8].

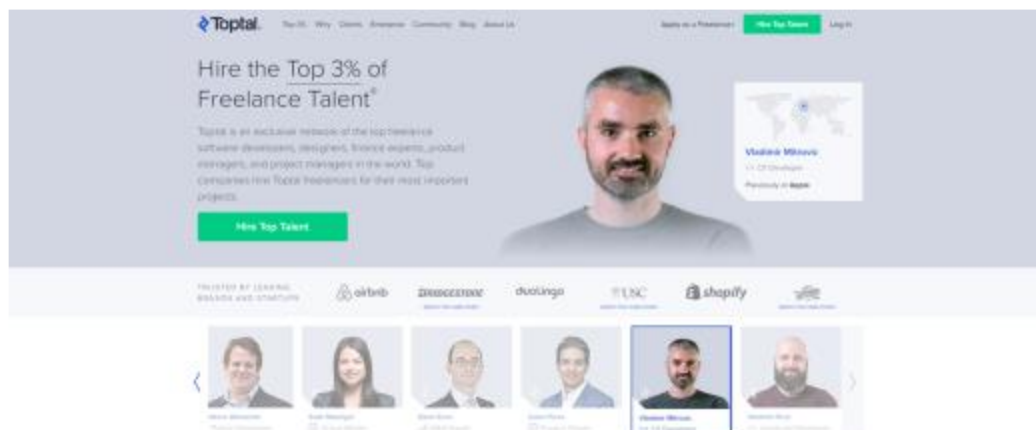


Рисунок 1.2 – Вебсайт Toptal [8]

Вид фрілансу: розробка програмного забезпечення, фінансовий консалтинг, тимчасове управління [8]-[9].

Плюси: компанії вищого рівня та таланти, безкоштовні рахунки та платежі.

Мінуси: інтенсивний процес перевірки, лише великі проєкти.

Для того, щоб подати заявку на роботу в TopTal як фрілансера, потрібно пройти п'ять етапів перевірки, починаючи від комплексної оцінки англійською мовою і закінчуючи оцінкою проєкту. Цей процес гарантує, що на платформу потраплять лише 3% найкращих [8]-[9].

Пройшовши всі кроки, фрілансер отримає доступ до різноманітних оголошень про роботу від провідних клієнтів і компаній, таких як Motorola та Airbnb. Однак якщо не пройти іспит, процес подання заявки призупиниться на кілька місяців.

У TopTal є спеціальний сервіс для відстеження часу та виставлення рахунків під назвою TopTracker. За допомогою цієї послуги фрілансери отримають загальну ціну, яку вони пропонують, і отримають платежі через PayPal, PayPal або прямий місцевий банківський переказ [8]-[9].

Jooble – це пошукова система для пошуку вакансій, яка збирає дані з понад 140 000 джерел по всьому світу, включаючи корпоративні вебсайти, онлайн-дошки вакансій, платформи фрілансерів, соціальні мережі та оголошення (рисунок 1.3) [8].

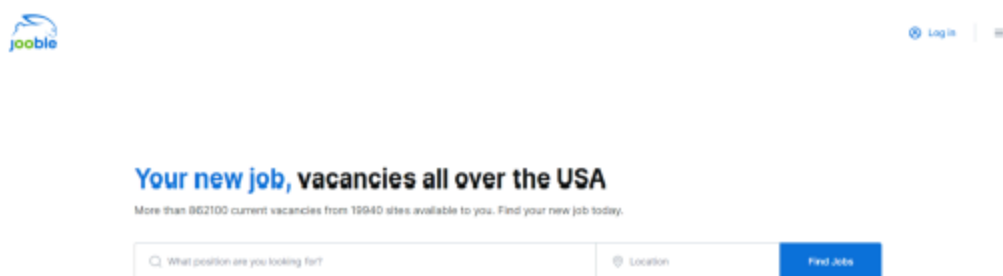


Рисунок 1.3 – Платформа Jooble [8]

Тип фрілансу: письмо, графічний дизайн, введення даних.

Плюси: централізований пошук вакансій, сповіщення електронною поштою, реєстрація не потрібна [8]-[9].

Мінуси: відсутність швидкої заявки – необхідно подати заявку з джерела кожної вакансії.

Значною перевагою використання Jooble є те, що не потрібно перевіряти численні онлайн-дошки вакансій і платформи для фрілансерів, оскільки це централізує процес пошуку можливостей фрілансу.

Jooble дозволяє знаходити роботу лише в певному регіоні. На головній сторінці можна вибрати конкретне місто у своїй країні, щоб знайти вакансії, обмежені цією місцевістю [8]-[9].

Не потрібно створювати обліковий запис, щоб знайти роботу фрілансера в Jooble. Все, що потрібно зробити, це ввести ключове слово для потрібної вакансії в пошукову систему Jooble. Наявні різні фільтри, такі як діапазон зарплати, досвід роботи та дата посади, щоб допомогти знайти бажану роботу.

Оскільки Jooble є агрегатором, вибір результату пошуку перенаправить вас до джерела вакансії. Там можна знайти докладнішу інформацію про вакансію та подати заявку на неї.

Підписавшись на сповіщення електронною поштою від Jooble, можна отримувати сповіщення щоразу, коли він знаходить нову роботу за ключовим словом [8]-[9].

Freelancer.com – це ще один вебсайт для фрілансерів, де професіонали та компанії з усього світу співпрацюють над різними проєктами. Завдяки широкому спектру знань клієнти можуть легко знайти фахівця будь-якого профілю. (рисунок 1.4) [8].

Тип фрілансу: переклад контенту, веброзробка, маркетинг у соціальних мережах [8]-[9].

Плюси: відстеження прогресу, живий чат, цілодобова підтримка клієнтів.

Мінуси: складний інтерфейс, спам-заявники та підроблені клієнти.



Рисунок 1.4 – Вебсайт Freelancer.com [8]

На платформі можна зареєструватися як фрілансер, або власник бізнесу. Щоб зареєструватися як фрілансер, потрібно буде заповнити коротку реєстраційну форму. Надати детальну інформацію, включаючи свої навички, рівень освіти та досвід роботи.

Окрім проєктів, є багато конкурсів, наприклад, візуальних чи дизайнерських робіт. Фрілансери можуть отримати вигоду від цих конкурсів, заробляючи гроші разом із відгуками у своїх профілях [8]-[9].

Кожен фрілансер отримає повну оплату після виконання роботи. Однак платформа забирає 10% від загального заробітку на конкурси та фіксовані проєкти. Як і більшість сайтів фрілансерів, можна використовувати PayPal, або банківський переказ для виведення коштів [8]-[9].

Upwork – це вебсайт фрілансерів, який об'єднує клієнтів і фрілансерів з усього світу (рисунок 1.5) [9].

Вид фрілансу: бренд-маркетинг, програмування, дизайн сайтів [8]-[9].

Плюси: захист платежів, надійні клієнти, бюджетні проєкти.

Мінуси: тривалий процес відбору, висока плата за обслуговування.



Рисунок 1.5 – Платформа Upwork [9]

Платформа надає широкий спектр категорій, від вебдизайну та розробки програмного забезпечення до обслуговування клієнтів і бухгалтерського обліку [8]-[9].

Клієнти та фрілансери мають кілька способів зв'язку – клієнти можуть опублікувати вакансію та найняти таланти, або придбати попередньо визначену послугу з каталогу проєкту. Подібним чином фрілансери можуть отримати доступ до дошки вакансій і пропонувати послуги для продажу.

Щоб розпочати роботу як фрілансер із Upwork, треба почати із створення профілю. Потрібно заповнити інформацію про фрілансера, його знання та досвід роботи. Клієнти вирішуватимуть, чи відповідає фрілансер кваліфікації, переглядаючи профіль щоразу, коли він подає заявку на роботу чи презентує проєкт [8]-[9].

Upwork має ковзну шкалу, тобто чим більше працює фрілансер, тим менше грошей він платите. Наприклад, комісія починається з 20% за перший рахунок у 500 доларів США та поступово зменшується, коли фрілансер отримує більше грошей [8]-[9].

Є кілька варіантів виведення коштів, включаючи прямий переказ, PayPal і банківський переказ [8]-[9].

Flexjobs – це вебсайт, який спеціалізується на гнучких і дистанційних можливостях працевлаштування, зокрема на фрілансі, неповний і повний

робочий день. Є широкі категорії, від освіти та навчання до журналістики та письменницької роботи (рисунок 1.6) [9].

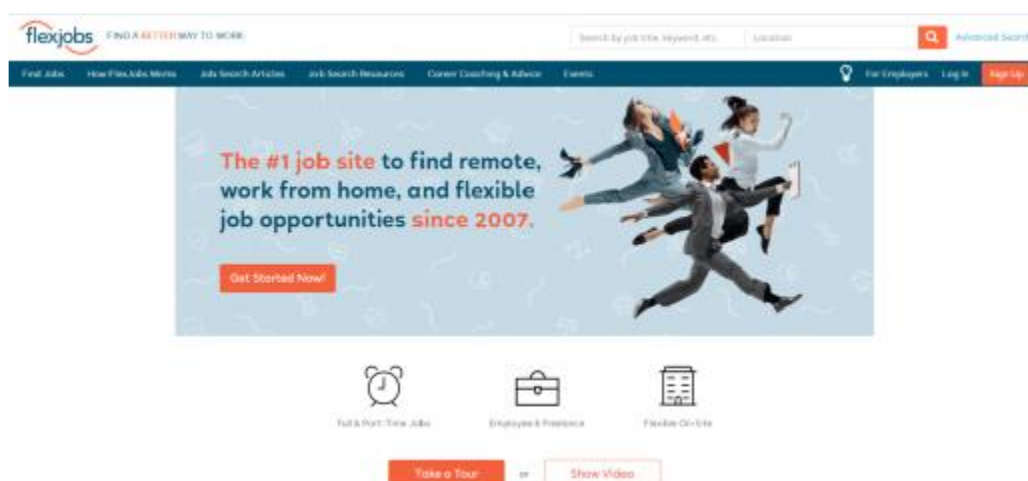


Рисунок 1.6 – Платформа Flexjobs [9]

Вид фрілансу: написання, контент-маркетинг, транскрипція.

Плюси: відсутність шахрайства, 30-денна гарантія повернення грошей, надійна підтримка.

Мінуси: деякі вакансії преміум-класу можна знайти за межами платформи.

Flexjobs гарантує, що кожна вакансія, представлена на вебсайті, є законною. Він перевіряє та перевіряє всі відкриття, фільтруючи шахрайство та фальшиві компанії. На відміну від інших онлайн-дошок вакансій, цей вебсайт фрілансерів не показує жодної реклами на своїй сторінці [8]-[9].

Щоб знайти роботу фрілансера, користувачам необхідно оформити підписку. Доступно чотири плани від 6,95 доларів США на тиждень. Кожна підписка надає необмежений доступ до списків вакансій, а також персональне робоче портфоліо, поради щодо кар'єри та безкоштовне тестування навичок.

Користувачі можуть оплатити підписку за допомогою передплаченої картки або PayPal. Існує 30-денна гарантія задоволення, тобто можна отримати повне відшкодування, якщо клієнт не задоволений послугою [8]-[9].

SimplyHired – це дошка вакансій, де компанії можуть безкоштовно публікувати пропозиції про вакансії. Він також збирає пропозиції роботи з усього Інтернету на одній сторінці. Сайт має простий користувацький інтерфейс, що дозволяє легко переглядати вакансії за категоріями (рисунок 1.7) [9].

Тип фрілансу: кадри, фінанси, введення даних.

Плюси: сповіщення електронною поштою, корисні посібники, безкоштовні інструменти для роботи.

Мінуси: спам із ненадійних сторонніх вебсайтів.

Існують різні можливості фрілансу в різних сферах, включаючи фінанси та маркетинг. Щоб знайти певну вакансію, просто заповніть поле пошуку відповідними ключовими словами. Щоб отримати релевантніші результати, звузіть список, вказавши своє місто, штат або поштовий індекс

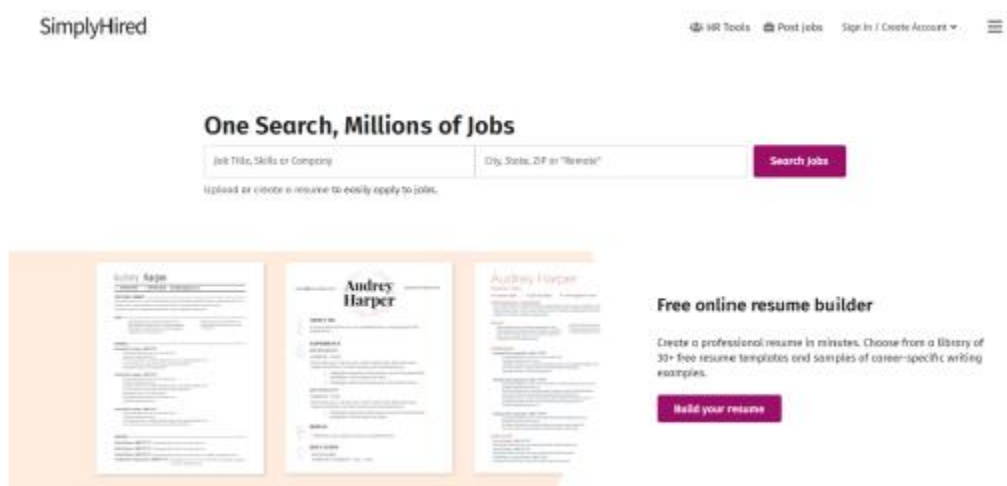


Рисунок 1.7 – Платформа SimplyHired [9]

Шукачі роботи також можуть отримати доступ до сторінок компанії, щоб отримати додаткову інформацію, таку як огляд заробітної плати компанії, перелік пропонованих переваг і відгуки, надіслані їй справжніми працівниками [8]-[9].

Люди можуть переглядати SimplyHired без профілю. Однак реєстрація облікового запису дасть вам доступ до різноманітних інструментів роботи, таких як оцінка зарплати та створення резюме.

Guru – це місце для спільної роботи фрілансерів і компаній з усього світу. Компанії можуть наймати професіоналів з різними навичками, наприклад програмістів, графічних дизайнерів і менеджерів із продуктів (рисунк 1.8) [9].

Вид фрілансу: маркетинг, програмування, адміністрування [8]-[9].

Плюси: безпечна оплата, індивідуальні списки вакансій, безкоштовне членство.

Мінуси: потенційні фальшиві клієнти.

Цей вебсайт для фрілансерів дозволяє легко переглядати списки вакансій і надсилати пропозиції клієнтам. Робота починається із реєстрації та створення профілю фрілансера. Система надасть рекомендації щодо роботи на основі навичок і досвіду роботи.

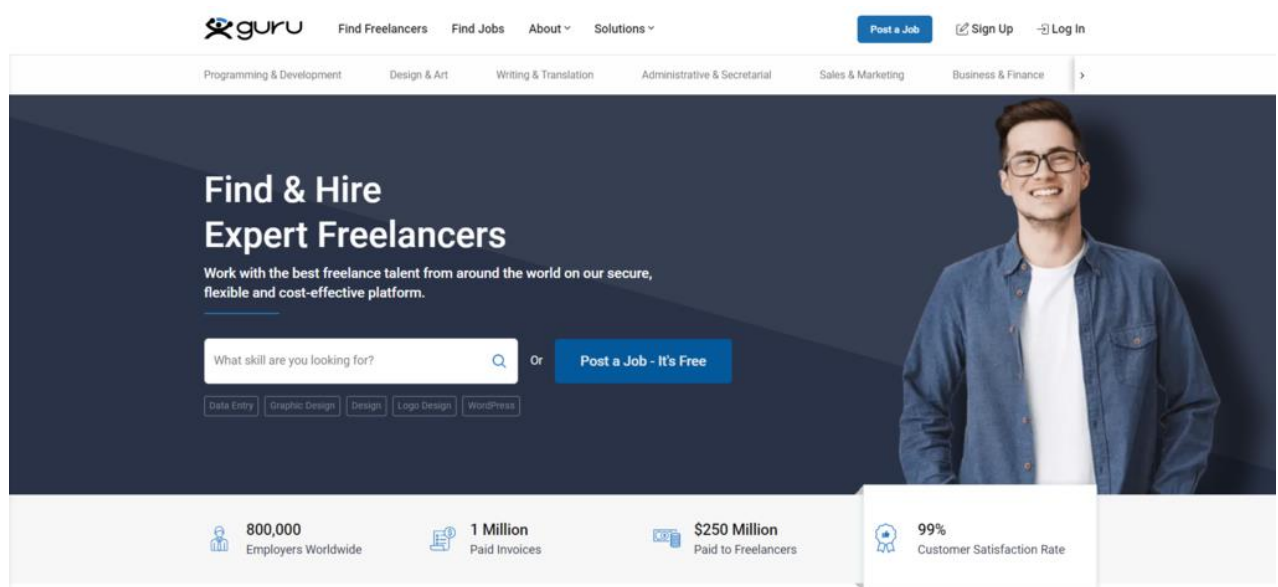


Рисунок 1.8 – Платформа SimplyHired [9]

Фрілансери можуть брати участь у торгах на довгострокові, або короткострокові проєкти та давати фіксовану ціну за кожен проєкт. Є кілька

методів виведення коштів, включаючи PayPal і банківський переказ. Сайт гарантує, що фрілансери отримують гроші вчасно завдяки захисту SafePay.

Кожен фрілансер, який зареєструвався на цьому вебсайті отримує безкоштовне базове членство. Однак платні плани членства включають більше інструментів, щоб зробити профіль помітним [8]-[9].

1.5 Висновки за розділом 1

Перший розділ пояснювальної записки до роботи було присвячено ознайомленню із переметною областю, яка стосується виконання віддалених завдань. Визначено поняття та принципи виконання віддалених завдань, Сформульовані основні відмінності виконання віддалених завдань від повної зайнятості. Проаналізовані категорій виконавців віддалених завдань. Виконано аналіз існуючих платформ для виконання віддалених завдань

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

На сьогоднішній день понад 98,8% вебсайтів використовують JavaScript на стороні клієнта для відтворення вебсторінок, що підкреслює його домінування та всюдисущість у веброзробці.

Ця приголомшлива статистика не тільки підкреслює широке впровадження JavaScript, але й готує основу для тонкого обговорення його сильних і слабких сторін [10]-[11].

JavaScript – широко поширена мова програмування, відома своєю універсальністю у створенні інтерактивних елементів. Брендан Айх розробив його всього за 10 днів у 1995 році, працюючи в Netscape Communications Corporation [10]-[11].

Спочатку спрямований на додавання інтерактивності до статичних HTML-сторінок, JavaScript перетворився на важливий інструмент для веброзробки. На відміну від мов, призначених переважно для серверних сценаріїв або настільних додатків, JavaScript зосереджується на покращенні взаємодії з інтерфейсом користувача у веббраузерах. Він працює з 32-розрядними операндами та зберігає числа як 64-розрядні числа з плаваючою комою.

Протягом багатьох років JavaScript розвивався разом з Інтернетом, з появою потужних фреймворків і технологій, таких як Node.js, які розширили його застосування за межі браузера [10]-[11].

Сьогодні JavaScript залишається основним елементом веброзробки, стимулюючи інновації та інтерактивність цифрових платформ.

JavaScript служить основою для безлічі програм, надаючи їм потужність завдяки своїй універсальності та широкому застосуванню. Ось кілька яскравих прикладів:

- платформи соціальних медіа: провідні соціальні медіа-платформи, такі як Facebook, Twitter і Instagram, значною мірою покладаються на

JavaScript для своїх динамічних інтерфейсів, що забезпечує оновлення в реальному часі та безперебійну роботу користувачів;

- односторінкові програми (SPA): такі програми, як Gmail, Google Maps і GitHub, використовують такі фреймворки JavaScript, як Angular, React або Vue.js, для створення SPA. Ці SPA надають користувачам плавний, адаптивний інтерфейс без незручностей постійного перезавантаження сторінок [10]-[11];

- сервіси потокового передавання: такі гіганти потокової індустрії, як Netflix, Spotify і YouTube, використовують JavaScript, щоб забезпечити захоплюючий досвід потокового передавання медіа. JavaScript забезпечує плавне відтворення та інтерактивні функції для мільйонів користувачів у всьому світі;

- інструменти для співпраці: такі програми для співпраці, як Slack і Trello, використовують потужність JavaScript для створення середовищ для співпраці. Завдяки оновленням у реальному часі та безперебійному спілкуванню JavaScript дозволяє командам ефективно працювати разом незалежно від їх фізичного розташування [10]-[11];

- платформи електронної комерції: великі торговельні вебсайти, такі як Amazon і eBay, використовують JavaScript для покращення досвіду покупок. Завдяки таким функціям, як динамічні рекомендації продуктів і оновлення в реальному часі, JavaScript дозволяє цим платформам надавати зручні для користувача інтерфейси, які залучають клієнтів;

- фінансові платформи: безпека та сучасність є найважливішими для фінансових платформ, і JavaScript відіграє вирішальну роль у досягненні обох. Такі платформи, як PayPal і різноманітні системи онлайн-банкінгу, включають JavaScript, щоб полегшити безпечні фінансові транзакції, пропонуючи сучасні інтерфейси, які забезпечують зручну роботу користувача [10]-[11].

В таблиці 2.1 стило описані переваги та недоліки JavaScript.

Таблиця 2.1 – Переваги та недоліки JavaScript

Переваги JavaScript	Недоліки JavaScript
Забезпечує динамічний та інтерактивний вебвміст.	Вразливий до загроз безпеці, таких як міжсайтовий сценарій (XSS).
Покращує взаємодію з користувачем завдяки обробці на стороні клієнта.	Обмеження продуктивності через однопотокове виконання.
Широка підтримка бібліотек через фреймворки, такі як React і Angular.	Невідповідності в різних браузерях можуть вимагати додаткового тестування.
Повна інтеграція з HTML і CSS для веброзробки.	Залежність від обробки на стороні клієнта може перешкоджати доступності.
Сприяє створенню адаптивних, зручних для мобільних пристроїв інтерфейсів.	Налагодження та усунення несправностей коду JavaScript може бути складним завданням.

Переваги JavaScript:

– підвищена швидкість і ефективність. Швидкість JavaScript пояснюється його здатністю виконуватися на стороні клієнта, зводячи до мінімуму залежність від ресурсів сервера. Це означає, що коли користувач отримує доступ до вебсайту, код JavaScript запускається безпосередньо в його браузері, що забезпечує швидке завантаження та динамічну взаємодію без постійного зв'язку з сервером. Крім того, він ефективно керує передачею даних, забезпечуючи передачу великих обсягів інформації з мінімальною затримкою, підвищуючи загальну ефективність і швидкість реагування вебдодатків [10]-[11];

– спрощений процес розробки. Простота JavaScript оптимізує розробку вебсайтів, дозволяючи виконувати як зовнішні, так і внутрішні завдання однією мовою. Його нескладний синтаксис і велика кількість навчальних матеріалів роблять його доступним для розробників усіх рівнів. Цей уніфікований підхід зменшує складність кодування, полегшуючи створення та підтримку вебсайтів. За допомогою JavaScript розробники

можуть зосередитися на створенні функціональності без додаткових витрат на керування кількома мовами, підвищуючи ефективність і сприяючи більш простому процесу розробки [10]-[11];

- безперервні оновлення. JavaScript регулярно оновлюється та вдосконалюється під наглядом таких організацій, як ECMA International. Ці оновлення, керовані специфікацією ECMAScript, вводять нові функції та підвищують продуктивність. Для розробників це означає доступ до сучасних інструментів і можливостей, завдяки яким їхні програми залишаються актуальними та конкурентоспроможними. Залишаючись динамічним і реагуючи на постійні потреби, JavaScript гарантує, що розробники можуть використовувати останні досягнення в веброботці, сприяючи інноваціям і ефективності своїх проєктів [10]-[11];

- зниження навантаження на сервер. JavaScript зменшує навантаження на сервер, дозволяючи певним завданням, таким як перевірка форм і динамічний рендеринг вмісту, виконуватися браузером користувача, а не сервером. Це означає, що замість того, щоб постійно запитувати інформацію з сервера, браузер може обробляти деякі завдання локально, що призводить до швидшого часу відповіді та покращеної масштабованості вебдодатків. Це розвантаження завдань допомагає розподілити робоче навантаження, роблячи загальну систему більш ефективною та чутливою до взаємодії користувача [10]-[11];

- повна сумісність. Інтероперабельність JavaScript означає, що незалежно від того, хто пише код, він може бездоганно працювати разом. Це спрощує такі завдання, як керування елементами вебсторінки та доступ до функцій браузера. Він сприяє співпраці між розробниками та полегшує створення потужних програм. Завдяки сумісності код JavaScript з різних джерел може бути інтегрований без конфліктів, підвищуючи ефективність і функціональність проєктів веброботки [10]-[11];

- універсальність застосування. JavaScript є універсальним, тобто він може адаптуватися до різних способів програмування. Він служить мовою для

інтерпретації коду, роботи з об'єктами та створення сценаріїв для вебсторінок. Він широко використовується у веброзробці, дозволяючи розробникам створювати візуально привабливі та інтерактивні інтерфейси. Незалежно від того, чи йдеться про зміну зображень чи додавання розширених функцій, JavaScript є домінуючим і гнучким, що робить його найкращим вибором для створення різноманітних вебдодатків із багатим користувацьким досвідом [10]-[11];

- модель асинхронного програмування. Модель асинхронного програмування JavaScript дозволяє завданням виконуватися незалежно, запобігаючи блокуванню одним завданням інших. Наприклад, під час отримання даних із сервера JavaScript не зупиняє всю програму, дозволяючи користувачеві продовжувати взаємодіяти з інтерфейсом. Ця можливість покращує швидкість реагування та продуктивність програми, забезпечуючи більш плавну роботу користувача. Такі завдання, як вибірка даних, обробка вхідних даних і паралельні процеси, можуть виконуватися одночасно, зменшуючи затримки та підвищуючи загальну ефективність [10]-[11];

- рясна екосистема. JavaScript отримує переваги від великої екосистеми бібліотек і фреймворків, таких як React, Angular і Vue.js. Ці інструменти спрощують веброзробку, пропонуючи готові компоненти та структури. Розробники можуть використовувати ці ресурси для прискорення створення складних програм, не починаючи з нуля. Наприклад, React надає багаторазові компоненти інтерфейсу користувача, Angular пропонує надійне зв'язування даних, а Vue.js вирізняється своєю простотою та гнучкістю. Загалом ця багата екосистема підвищує продуктивність і дозволяє розробникам легко створювати потужні вебдодатки.

Недоліки JavaScript:

- ризик безпеки на стороні клієнта. Реалізація JavaScript у різних браузерах відрізняється, створюючи вразливі місця для користувачів. Хакери використовують ці відмінності для впровадження шкідливого коду, що ставить під загрозу безпеку клієнтських систем. Це становить значний ризик,

оскільки модифіковані сценарії можуть виконуватися непоміченими, дозволяючи несанкціонований доступ і потенційні порушення даних. Як наслідок, користувачі стикаються з такими загрозами, як несанкціонований доступ до даних і маніпулювання системою. Такі ризики безпеки підкреслюють важливість пильної практики кодування та надійних заходів безпеки для захисту від уразливостей, пов'язаних із JavaScript [10]-[11];

- видимість коду. Код JavaScript, видимий для всіх на вебсторінці, означає, що будь-хто може отримати до нього доступ і потенційно змінити його, що збільшує ризик несанкціонованого доступу та витоку даних. Ця видимість також уповільнює виконання коду, оскільки JavaScript повинен відстежувати весь видимий код. По суті, така відкритість полегшує зловмисникам використання вразливостей, створюючи значний ризик для безпеки вебсайтів та їхніх користувачів. Тому розробники повинні застосувати додаткові заходи безпеки для захисту конфіденційних даних;

- відмінності міжбраузерної інтерпретації. Інтерпретація JavaScript відрізняється в різних вебпереглядачах, що призводить до неузгодженості в тому, як сценарії поведуться на різних платформах. Це означає, що код, написаний для одного браузера, може не працювати належним чином в іншому. Розробникам часто доводиться писати окремий код для кожного браузера або використовувати додаткові методи, щоб забезпечити узгоджену роботу в різних середовищах. Ці відмінності можуть ускладнити процес розробки та збільшити ймовірність помилок, ускладнюючи створення вебпрограм, які бездоганно працюють у всіх браузерах [10]-[11];

- модель єдиного успадкування. Єдина модель успадкування JavaScript дозволяє об'єктам успадковувати властивості та поведінку від одного об'єкта-прототипу. На відміну від мов, які підтримують множинне успадкування, обмеження JavaScript означає, що об'єкт може успадкувати лише від одного батька. Це обмеження може ускладнити організацію коду, оскільки розробники повинні знайти альтернативні підходи, такі як міксини або композиція, щоб імітувати множинне успадкування. Хоча ці методи

пропонують рішення, вони також можуть внести додаткову складність у кодову базу [10]-[11];

- проблеми налагодження. Динамічний і вільно типізований характер JavaScript ускладнює налагодження, оскільки помилки можуть виникати лише під час виконання. На відміну від мов зі статичною типізацією, проблеми, пов'язані з типами, важче виявити під час розробки. Крім того, старі браузері часто не мають надійних інструментів налагодження, що ще більше ускладнює процес. Це означає, що розробники повинні вкладати більше часу та зусиль у виявлення та виправлення помилок, які можуть уповільнити цикл розробки та потенційно вплинути на роботу вебдодатків;

- виконання займає багато часу. Модель обробки об'єктів документа (DOM) JavaScript може працювати мляво, особливо в повільних мережах, спричиняючи затримки під час завантаження вебвмісту. Крім того, постійне перетворення даних між різними типами може ще більше сповільнити обробку. Це означає, що хоча вміст HTML завантажується швидко, операції JavaScript можуть тривати довше, що впливає на загальну взаємодію з користувачем. У результаті користувачі з повільнішим мережевим з'єднанням можуть подовжити час очікування, поки вебсторінки повністю завантажаться та стануть інтерактивними [10]-[11];

- залежність від JavaScript. Вимкнення JavaScript на вебсайті, яке часто робиться з міркувань безпеки, може спричинити значні проблеми з функціональністю. Деякі важливі функції, як-от інтерактивні карти на основі JavaScript, можуть не завантажуватися, що порушує роботу користувача. Хоча сучасні вебпереглядачі можуть не помічати незначні помилки JavaScript, один недолік може зупинити роботу всього вебсайту. Ця залежність від JavaScript підкреслює важливість ретельного кодування для забезпечення безперебійної роботи в різних сценаріях перегляду [10]-[11];

- проблеми пошукової оптимізації (SEO). Пошукові системи мали труднощі з розумінням та індексуванням вмісту, створеного за допомогою JavaScript, що ускладнювало пошук вебсайтів, які широко використовують

JavaScript. Незважаючи на те, що технологію пошукової системи було вдосконалено, залишаються проблеми щодо забезпечення того, щоб вміст на основі JavaScript належним чином індексувався та ранжувався в результатах пошуку. Це може вплинути на видимість і доступність вебсайтів, потенційно зменшивши їх охоплення та ефективність охоплення цільової аудиторії через звичайний пошуковий трафік [10]-[11].

Зрештою, рішення про використання JavaScript залежить від вимог проєкту, навичок команди та цілей взаємодії з користувачем. Сумлінно оцінюючи його плюси та мінуси, розробники можуть ефективно використовувати можливості JavaScript, зменшуючи пов'язані з цим ризики.

Враховуючи розповсюдженість JavaScript та перераховані переваги і недоліки цієї мови, в якості мови програмування для розробки додатку було обрано JavaScript [10]-[11].

2.2 Вибір фреймворку для розробки Frontend частини вебсайту

JavaScript широко використовується для інтерфейсної розробки протягом майже двох десятиліть. Такі популярні фреймворки, як React, Vue.js і Angular, продовжують домінувати в рейтингу, тоді як нові претенденти, такі як Svelte, Solid і Qwik, стабільно набирають обертів. Ландшафт розвивається, оскільки розробники досліджують нові технології разом із усталеними фреймворками. Базуючись на останніх тенденціях опитування State of JS 2024, на рисунку 2.1 представлені найкращі фреймворки JavaScript для інтерфейсної розробки у 2025 році [12]-[13].

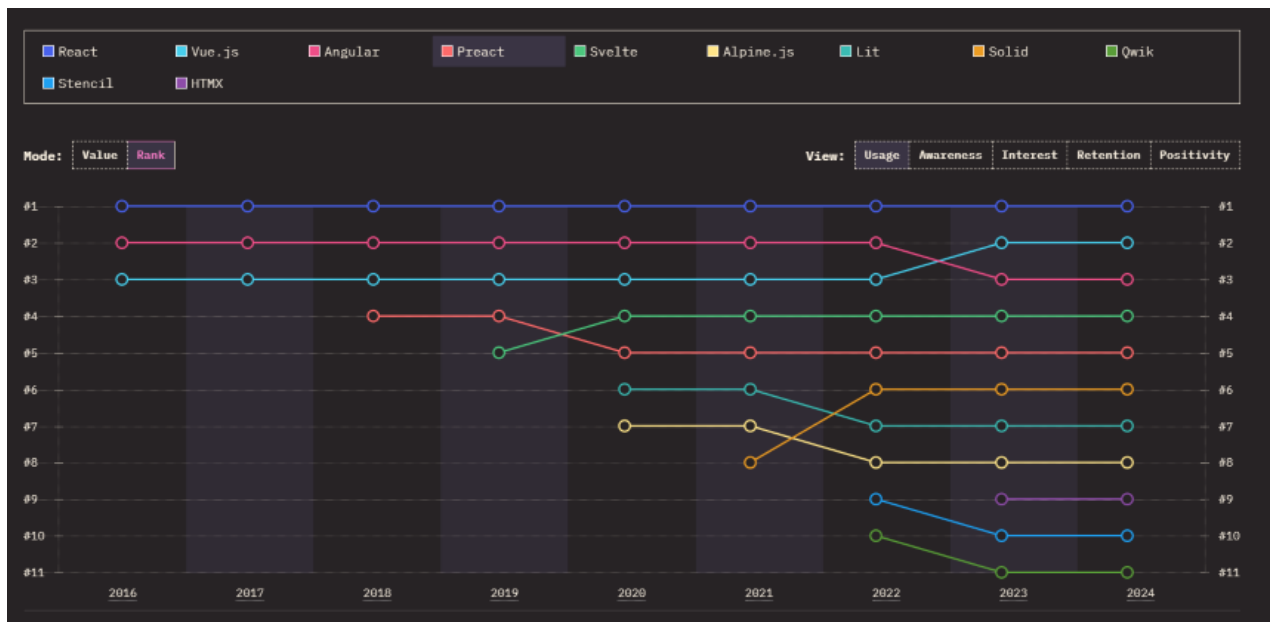


Рисунок 2.1 – Найкращі фреймворки JavaScript для інтерфейсної розробки у 2025 році [12]

Розглянемо ці фреймворки більш детально.

1. React, також відомий як React.js, є одним із найкращих фреймворків JavaScript у категорії інтерфейсу. Це інтерфейсна бібліотека JavaScript з відкритим вихідним кодом, створена командою розробників Facebook на чолі з Джорданом Волке ще в 2011 році та стала відкритою в червні 2013 року. Початковий прототип отримав назву «FaxJS» і вперше експериментував із ним у стрічці новин Facebook. React можна вважати одним із найбільших руйнівників у сфері веброзробки, який зробив справжній прорив у формуванні вебдодатків, які ми бачимо сьогодні [12]-[13].

React, також відомий як React.js, є одним із найкращих фреймворків JavaScript у категорії інтерфейсу. Це інтерфейсна бібліотека JavaScript з відкритим вихідним кодом, створена командою розробників Facebook на чолі з Джорданом Волке ще в 2011 році та стала відкритою в червні 2013 року. Початковий прототип отримав назву «FaxJS» і вперше експериментував із ним у стрічці новин Facebook. React можна вважати одним із найбільших руйнівників у сфері веброзробки, який зробив справжній прорив у формуванні вебдодатків, які ми бачимо сьогодні [12]-[13].

React представив компонентний, функціональний і декларативний стиль програмування для створення інтерактивних інтерфейсів користувача для переважно односторінкових вебдодатків. React забезпечує неймовірно швидке рендеринг, використовуючи «Віртуальний DOM». DOM розшифровується як Document Object Model, яка відтворює лише ті компоненти, які змінилися, замість відтворення всієї сторінки. Іншою ключовою особливістю React є простіший синтаксис JSX замість JavaScript.

Незважаючи на те, що React демонструє дещо крутішу криву навчання, ніж інші найкращі фреймворки JavaScript у цьому списку, він підтримується величезною спільнотою розробників, щедрими навчальними ресурсами та масовим впровадженням у галузі в усіх куточках світу [12]-[13].

Статистика використання React:

За даними State of JS, сьомий рік поспіль він посідає перше місце в списку найкращих інтерфейсних фреймворків JavaScript у всьому світі! React займає перше місце з 2016 року, випереджаючи основних конкурентів Vue.js і AngularJS як найкращі фреймворки JavaScript [12]-[13].

Згідно з опитуванням State of JS 2022, 81,8% розробників JavaScript наразі використовують React.js, тоді як ще 47,2% виявили великий інтерес до вивчення цього фреймворку JavaScript у майбутньому [12]-[13].

З 40,58% голосів React випередив jQuery і посів друге місце в опитуванні розробників Stack Overflow щодо найкращих фреймворків JavaScript у 2025 році, залишивши його на третьому місці.

Поточна версія – 19.0.0

GitHub – 233 тисячі зірок і 1651+ учасників [12]-[13].

NPM – у середньому близько до історичного максимуму в 32 млн+ завантажень на тиждень. Статистика використання React зображена на рисунку 2.2 [12].

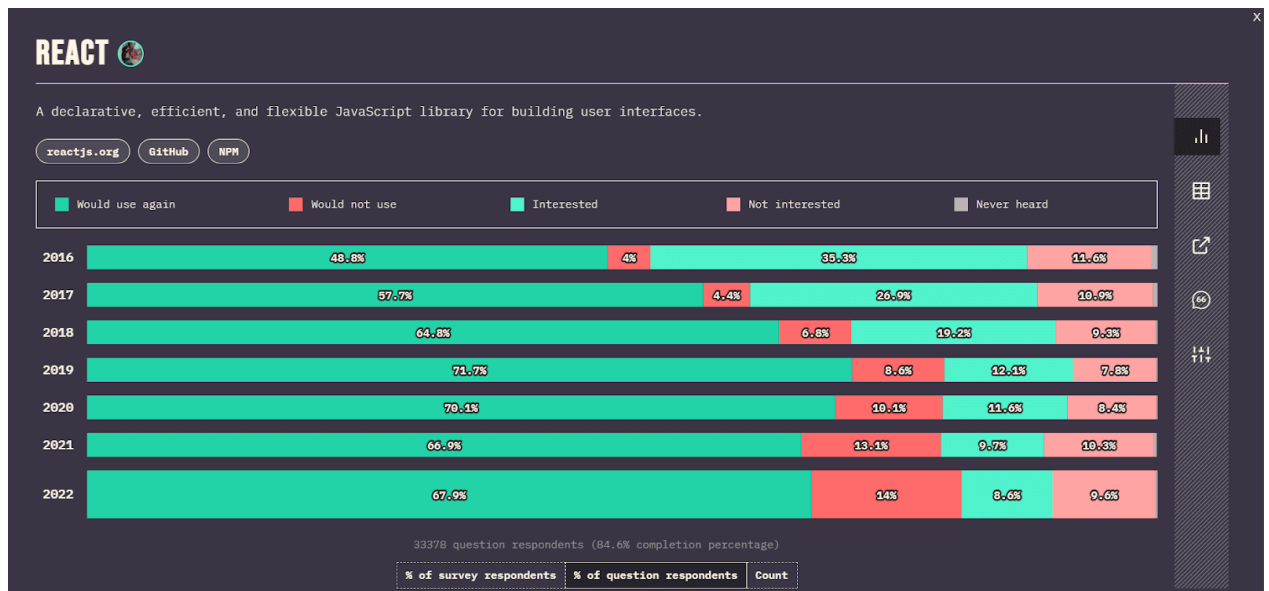


Рисунок 2.2 – Статистика використання React [12]

Плюси використання React:

- багаторазові компоненти – його компонентний підхід допомагає розробникам імпортувати або повторно використовувати компоненти інтерфейсу користувача, не докладаючи жодних зусиль, а не кодувати їх з нуля [12]-[13];
- легка інтеграція – забезпечує легку інтеграцію з іншими зовнішніми та внутрішніми фреймворками, такими як популярна PHP-фреймворк Laravel, щоб працювати без збоїв;
- напрямок потоку даних – головна відмінність полягає в тому, що React.js дотримується архітектури потоку даних, спрямованого вниз. Це гарантує, що дочірній елемент не може впливати на батьківський, що робить код більш стабільним і менш схильним до помилок [12]-[13];
- простіший синтаксис – використовує розширення JSX JavaScript замість звичайного JavaScript, HTML-подібний синтаксис, який полегшує читання, розуміння та підтримку коду [12]-[13];
- віртуальний DOM – створює віртуальне представлення або копію DOM під назвою Virtual DOM або vDOM. React порівнює віртуальний DOM із справжнім DOM, щоб відобразити лише компоненти, які змінилися, замість відтворення всієї сторінки. Це ключ до неймовірної продуктивності React;

- SEO – після багатьох удосконалень React.js є одним із найбільш сприятливих для SEO інтерфейсних фреймворків, які полегшують сканування сторінок. Традиційні фреймворки JavaScript, які динамічно відтворювали вміст, були сумно відомі своєю низькою ефективністю SEO та несумісністю з сканерами пошукових систем [12]-[13];

- постійне вдосконалення – React знаходиться в стані постійної еволюції та вдосконалення, додаючи ключові нові функції, такі як Hooks, Fiber, Concurrent Mode, Suspense тощо, для зменшення шаблонного коду, покращення паралелізму та швидкого рендерингу, а також підвищення продуктивності, що робить його сильним претендентом у категорії найкращих фреймворків JavaScript [12]-[13];

- багата підтримка спільноти – React підтримується великою спільнотою розробників із майже 1,6 тисячами співавторів на GitHub і широким набором навчальних ресурсів, посібників і довідкових посібників, доступних кожному, хто починає. Також не забувайте, що React.js підтримується однією з найбільших і найпотужніших компаній світу – Facebook.

Мінуси використання React:

- JSX є простішим синтаксисом, який легше читати та розуміти, він може стати додатковим бар'єром для навчання новачків [12]-[13];

- юагато розробників можуть вважати, що React має набагато крутішу криву та більшу складність, ніж такі ключові конкуруючі фреймворки, як Vue.js або будь-які інші найкращі фреймворки JavaScript;

- React часто страждає від скарг на погану документацію, яка не може наздогнати нові випуски та функції, що змушує розробників;

- проблема з рівнем перегляду моделі MVC, не залишаючи іншого вибору, окрім як покладатися на інші технології для роботи з моделлю та рівнями контролера [12]-[13];

Відомі вебсайти, створені за допомогою React: Airbnb, Asana, BBC, Cloudflare, Codecademy, Dropbox, Facebook, GitHub, Imgur, Instagram, Medium,

Netflix, OkCupid, Paypal, Periscope, Pinterest, Product Hunt, Reddit, Salesforce, Scribd, Shopify, Slack, Snapchat, Squarespace, Tesla, The New York Times, Typeform, Twitter, Uber, Udemy, WhatsApp, Zendesk [12]-[13].

2. Angular – це фреймворк Google із відкритим кодом для створення динамічних односторінкових вебдодатків. Він використовує TypeScript для безпеки типів, використовує повторно використовувані компоненти для структури та пропонує двостороннє зв'язування даних для спрощеної розробки, що робить його популярним вибором для масштабованих вебпроектів [12]-[13].

Angular другий рік поспіль посідає друге місце серед найкращих інтерфейсних фреймворків JavaScript в опитуванні про стан JS 2022 [12]-[13].

В опитуванні розробників Stack Overflow 2023 Angular зайняв 5 місце серед найкращих інтерфейсних фреймворків JavaScript із 17,46% позитивних відповідей [12]-[13].

Поточна версія – v19.2.0.

GitHub – 97,1 тис.+ зірок і 1873 тис.+ учасників.

NPM – Angular зареєстрував величезні 3,9 млн+ завантажень на тиждень на NPM. Статистика використання Angular зображена на рисунку 2.3 [13].

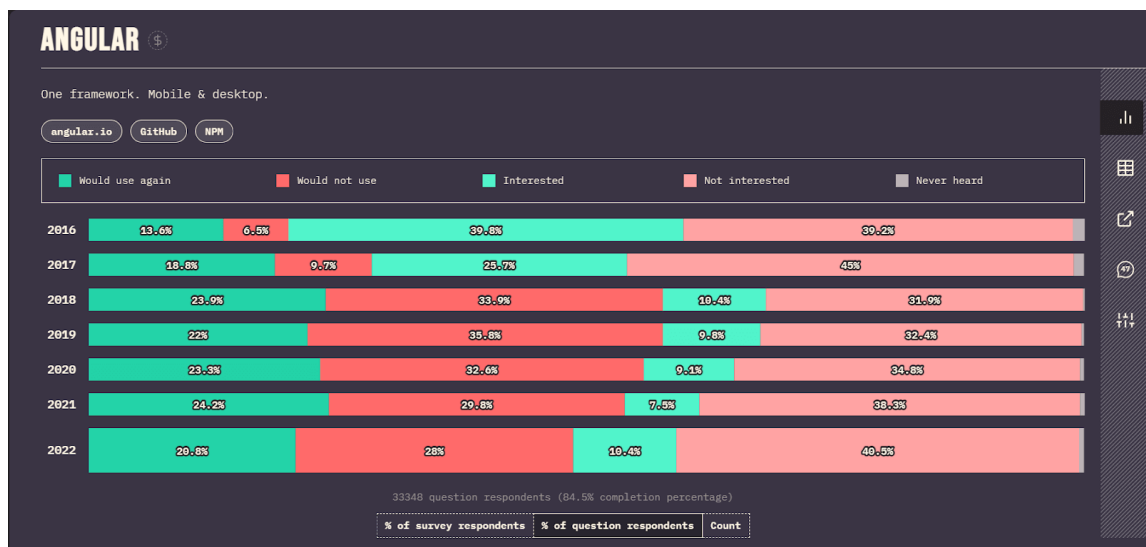


Рисунок 2.3 – Статистика використання Angular [12]

Плюси використання Angular:

- компонентна архітектура . Відмовившись від архітектури MVW (model-view-whatever), Angular 2 і новіші версії прийняли суто компонентну архітектуру, яка полегшує повторне використання. Компоненти можна використовувати знову і знову в додатку. Це також покращує читабельність коду та полегшує обслуговування [12]-[13];
- продуктивність сервера – Angular підтримує готове кешування та багато функцій для забезпечення швидкої роботи сервера [12]-[13];
- MVC – контролер представлення моделі в angular встановлює ключові функції, такі як області видимості та зв'язування даних. Це також забезпечує ізоляцію інтерфейсу користувача та логіки додатка одна від одної;
- двостороннє зв'язування даних . Однією з ключових особливостей Angular є двостороннє зв'язування даних, яке формує зв'язок між шаром Model і View таким чином, що кожен з них відображає зміни в іншому. Однак інші конкуруючі фреймворки прийняли односторонню прив'язку даних для спрощення [12]-[13];
- інтеграції сторонніх розробників – Angular пропонує багату колекцію інтеграцій сторонніх розробників для подальшого покращення їх вебдодатків [12]-[13];
- масштабна підтримка спільноти – Angular благословляється підтримкою від Google і величезної спільноти розробників, яка збирає понад 1,6 тисяч учасників на GitHub. AngularJS також має чудову документацію, величезну бібліотеку ресурсів і підтримку спільноти [12]-[13].

Мінуси використання Angular:

- TypeScript – щоб використовувати Angular, ви повинні використовувати TypeScript, який пропонує багато переваг, але все ще створює серйозний виклик для нових розробників. TypeScript є великою перешкодою для входу і пояснює, чому angular боровся з React і Vue.js [12]-[13];
- важкий і роздутий – порівняно з такими легкими альтернативами, як

React і Vue, AngularJS надзвичайно громіздкий і великий, приблизно близько 550-600 Кб, що може не підходити для більшості невеликих програм;

– погана SEO – Angular бореться з SEO через гострі проблеми з доступністю, загнані в кут сканерами/павуками пошукових систем. Якщо ваша основна увага спрямована на SEO, React є набагато кращою альтернативою для SEO [12]-[13].

Відомі вебсайти, створені за допомогою Angular: Google, Allegro, Blender, Clickup, Clockify, Delta, Deutsche Bank, DoubleClick, Freelancer, Forbes, Guardian, IBM, Instapage, iStock, JetBlue, Lego, Mailerlite, Microsoft Office, Mixer, Udacity, Upwork, Vevo, Walmart, Weather, WikiWand, Xbox, YouTube.

3. Vue.js або Vue – це легкий зовнішній фреймворк JavaScript із відкритим кодом, який використовується для створення креативних інтерфейсів користувача та високопродуктивних односторінкових вебдодатків із мінімальними зусиллями [12]-[13].

Vue.js вперше був запущений у 2014 році Еваном Ю, розробником, який працює на Google, який черпав натхнення з AngularJS, щоб створити просту, легку та ефективну альтернативу у вигляді Vue.js. І з того часу Vue.js є одним із найкращих фреймворків JavaScript для розробки інтерфейсу. Незважаючи на те, що Vue.js запозичив більшість своїх функцій від ReactJS і AngularJS, він значно вдосконалив ці функції, щоб створити кращу, просту у використанні та безпечну структуру. Найбільшим прикладом цього підходу є те, що Vue.js пропонує двостороннє зв'язування даних, як це видно в AngularJS, і «віртуальний DOM», як це видно в React [12]-[13].

Vue.js посів третє місце в списку найкращих фреймворків JavaScript для інтерфейсу в опитуванні розробників про стан JS 2022 [12]-[13].

46,2% розробників JavaScript наразі використовують Vue.js і обіцяють продовжувати його використовувати, тоді як 50,7% виявили великий інтерес до його використання.

Поточна версія – v2.7.16.

GitHub – 208 тисяч зірок і 364+ співавтори.

NPM – Vue.js зареєстрував вражаючі 6,4 млн+ завантажень на тиждень на NPM.

Плюси використання Vue.js:

- простота – найсильніша перевага Vue.js – це його простота. Будь-який розробник React або AngularJS може миттєво підібрати Vue без жодних потовиділень. Структура Vue проста і дозволяє кодувати більше за допомогою безтипового синтаксису. Vue використовує однофайлову систему компонентів, тобто HTML, CSS і JavaScript знаходяться в одному файлі для будь-якого компонента [12]-[13];
- вбудований MVC – є вбудований MVC, який забезпечує швидку та просту конфігурацію, на відміну від ReactJS [12]-[13];
- легко освоїти – не вимагає вивчення JSX або TypeScript, таких як React або AngularJS. Vue використовує прості для сприйняття шаблони HTML, які не ускладнюють код [12]-[13];
- невеликий розмір – ще однією важливою перевагою є його напрочуд легкий розмір, враховуючи його вражаючу майстерність. Архівована версія Vue.js має розмір ледве 18-20 Кб, набагато легша, ніж її роздуті, громіздкі конкуренти, такі як React або AngularJS;
- інтеграція – його можна використовувати не тільки для надзвичайно швидкого створення додатків з нуля, але також можна легко інтегрувати в існуючі вебдодатки без будь-яких проблем зі створенням нових компонентів [12]-[13].

Мінуси використання Vue.js:

- невелика спільнота – Vue.js не користується підтримкою таких гігантів, як Facebook або Google, як у випадку з React або Angular. Але Vue справді вдалося створити спільноту прихильників. На GitHub Vue має 360+ співавторів, що є пристойним, але далеко не таким, яким підтримує React з 1,6 тис. співавторів [12]-[13].
- обмежені можливості працевлаштування – Vue, безсумнівно, надзвичайно популярний у Китаї та США, але це може бути не так в решті

світу. Традиційно США є лідером із впровадження нових технологій, решта світу відстає.

Відомі вебсайти, створені за допомогою Vue.js: 9gag, Adobe, Apple Swift UI, Behance, Bilibili, BMW, Chess, Font Awesome, GitLab, Hack The Box, Laravel, Laracasts, Louis Vuitton, Namecheap, Netlify, Netguru, Nintendo, Pluralsight, Shien, Stack Overflow, Trivago, Trustpilot, Upwork, Wizzair, Zoom.

4. Ember.js – це фреймворк JavaScript з відкритим кодом для створення масштабованих односторінкових вебдодатків для бізнесу. Архітектура Model-View-ViewModel (MVVM) є основою для Ember, на відміну від інших фреймворків, які ми обговорювали [13].

Спочатку Ember.js був фреймворком SproutCore 2.0, який був перейменований на Ember.js його творцем Єгудіо Кацом, досвідченим розробником, якого вважають одним із головних творців jQuery. Однією з найулюбленіших і ключових функцій Ember є інструмент інтерфейсу командного рядка Ember, який є джерелом продуктивності. Такі функції роблять його одним із найкращих фреймворків JavaScript [12]-[13].

NPM – Ember наближається до 1200+ щотижневих завантажень на NPM. Статистика використання Ember.js зображена на рисунку 2.4 [13].



Рисунок 2.4 – Статистика використання Ember.js [13]

Незважаючи на те, що Ember.js є застарілою інтерфейсною платформою JavaScript порівняно з React, Vue та Svelte, вона все ще має потужні можливості та має велику базу користувачів у таких великих компаніях, як Microsoft, LinkedIn, Netflix, Twitch тощо [12]-[13].

Ember.js посідає 4 місце в списку найкращих фреймворків JavaScript для категорії інтерфейсу в опитуванні State of JS 2022.

GitHub – 22,5 тисячі зірок і 824+ співавторів на GitHub [12]-[13].

Плюси використання Ember.js:

- API – Ember надає простий механізм роботи з API;
- Convention over Configuration – Ember.js дотримується підходу «CoC – Convention over Configuration», який гарантує, що в більшості випадків немає потреби у будь-якій конфігурації, тож ви можете одразу перейти до кодування та створення вебдодатку [12]-[13];

- багата документація та підтримка – Ember.js постачається з детальною документацією, докладними посібниками та довідковими посібниками, які допоможуть вам легко розпочати роботу [12]-[13];

- Ember CLI – інструмент командного рядка Ember – це засіб підвищення продуктивності, який не лише допомагає створювати нові проекти, але й створює компоненти та контролери за допомогою автоматичної генерації. Ember CLI також допомагає встановлювати сторонні залежності;

- інструмент Inspector – Ember.js має розширення інструменту розробки Inspector для Chrome і Firefox, яке допомагає відстежувати та редагувати програми в режимі реального часу [12]-[13];

- підтримує двостороннє зв'язування даних, подібне до AngularJS;
- зворотна сумісність з попередніми версіями.

Мінуси використання Ember.js:

- крута крива навчання – розробники-новачки можуть вважати Ember.js досить складним порівняно з Vue.js або React [12]-[13];

- складний для великомасштабних застосувань і може бути надто громіздким і важким для простих невеликих застосувань [12]-[13];

- будь-яке відхилення від стандартного виконання завдань може призвести до неприємних проблем [12]-[13];
- застій у популярності – зазнав застою протягом останніх кількох років, не змігши залучити нових розробників у своє коло.

Відомі вебсайти, створені за допомогою Ember: CrowdStrike, Ghost, GroupOn, Heroku, Intercom, LinkedIn, Microsoft, Netflix, Skylight, Square, TED, Twitch, Yahoo та Zendesk [12]-[13].

5. Preact – це легка, швидка, високопродуктивна бібліотека та альтернатива React. Preact має розмір лише 3 Кб (мінімізований і стиснутий у gzip), але він надає вам усі необхідні функції React, що робить його одним із найкращих фреймворків JavaScript. Preact був створений Джейсоном Міллером, старшим інженером програм для розробників у Google, і його можна розглядати як підмножину React, яка позбавляє деяких функцій [12]-[13].

Preact створено за тим же основним принципом, що й React – компонентний підхід із віртуальною DOM, але повністю сумісний із React. Він навіть може використовувати пакети React без шкоди для швидкості, продуктивності та меншого розміру. У випадках, коли повний потенціал React не потрібен, більшість розробників навіть використовують React під час розробки та переходять на Preact для виробництва. Багато великих компаній, як-от Tencent, Uber і Lyft, використовують Preact [12]-[13].

В опитуванні State of JS 2022 Preact посів п'яте місце серед найкращих інтерфейсних фреймворків JavaScript.

9,5% опитаних розробників JavaScript вже використовували Preact і планують продовжувати його використовувати, тоді як 22,6% виявили великий інтерес до вивчення та використання його в майбутньому [12]-[13].

Поточна версія – 10.26.4.

GitHub – 37,2 тис.+ зірок, 2 тис.+ форків, 316+ учасників.

NPM – Preact записав понад 5 мільйонів завантажень на тиждень на NPM. Статистика використання Preact зображена на рисунку 2.5 [13].



Рисунок 2.5 – Статистика використання Preact [13]

Плюси використання Preact:

- менший за розміром – завдяки своєму легкому розміру Preact все ще забезпечує швидшу продуктивність візуалізації порівняно з іншими фреймворками JavaScript, такими як React і Vue, який має розмір 40-45 Кб (у стиснутому файлі gzip [12]-[13];
- висока ефективність – ефективніше, ніж React та інші найкращі фреймворки JavaScript у використанні пам'яті. Це означає, що збирач сміття повинен виконувати менше роботи [12]-[13];
- Preact CLI – пропонує надзвичайно потужний інструмент інтерфейсу командного рядка, який допомагає легко створювати нові проєкти, заощаджуючи зусилля, пов'язані з Babel або вебпакетом [12]-[13];
- модуль стану посилення – хоча Preact є підмножиною React, він має кілька додаткових функцій. Найважливішим є модуль LinkState, який оптимізує зміни стану. React, який використовує функції зі стрілками всередині методу render() для оновлення стану. Цей спосіб неефективний, з великою кількістю непотрібного коду. Preact вирішує цю проблему за допомогою модуля LinkState, який повертає функцію обробки, коли її передано в подію, оновлення стану компонента змінюється автоматично;
- сумісність . Використовує той самий API ES6, що й React, тому розробники можуть легко перейти на Preact із React у існуючому проєкті під

час виробництва [12]-[13].

Мінуси використання Preact:

- Preact не підтримує propTypes, як інші фреймворки JavaScript, такі як React [12]-[13];
- Preact не підтримує createClass. Він підтримує лише функціональні компоненти без збереження стану та компоненти на основі класу ES6 [12]-[13].

Відомі вебсайти, створені за допомогою Preact: Bluehive, Bustle, Dailymotion, Dev.to, Dominos, Financial Times, GroupOn, Hashi Corp, Housing.com, Lyft, Pepsi, Rocket Chat, Smashing Magazine, Synacor, Sogou Wenen, Tencent, Treebo, Uber [12]-[13].

В результаті аналізу популярних фреймворків для розробки FrontEnd частини вебсайту було обрано фреймворк React [12]-[13].

2.3 Вибір фреймворку для розробки Backend частини вебсайту

Однією з ключових причин шаленого успіху JavaScript і його сходження на трон найпопулярнішої мови програмування в світі є його універсальність. Завдяки Node.js JavaScript можна використовувати не тільки для клієнтської інтерфейсної розробки, але й для серверної серверної розробки.

На вибір є кілька серверних фреймворків JavaScript, це і Node.js, Next.js і Express.js є найпопулярнішим вибором серед розробників. Проаналізімо найкращі фреймворки JavaScript на 2025 рік для серверної частини [14]-[15].

1. Node.js – це серверна платформа з відкритим кодом, яку можна використовувати з Windows, Linux, Unix, macOS та іншими операційними системами. Node.js використовує V8 JavaScript Engine, серверне середовище виконання для JavaScript, щоб виконувати код JavaScript поза веббраузером [14]-[15].

Приблизно через тринадцять років після випуску першого серверного середовища JavaScript, LiveWire Pro Web від Netscape, Райан Дал у 2009 році створив оригінальну версію Node.js. У початковій версії підтримувалися лише Linux і Mac OS X. І щоб створити OpenJS Foundation, JS Foundation і Node.js Foundation, об'єднані в 2019 році, і з того часу Node.js займає перше місце серед найкращих фреймворків JavaScript [14]-[15].

Node.js лише нативно підтримує JavaScript, хоча кілька мов можна перекласти на JS. Через це програми Node.js можна створювати різними мовами, включаючи Dart, TypeScript, ClojureScript і CoffeeScript.

Node.js підтримується в macOS, Linux і Microsoft Windows, а також підтримується рівень 2 для SmartOS, IBM AIX і FreeBSD. Крім того, OpenBSD працює на IBM I (AS/400), доступні версії LTS [14]-[15].

Node.js раніше домінував як найпопулярніша технологія в щорічному огляді Stack Overflow протягом останніх двох років; однак він займає перше

місце серед найпопулярніших вебфреймворків і технологій в опитуванні 2023 року [14]-[15].

Типовий час відповіді для ідентичної сторінки в PayPal зменшився на 35%. Сторінки обслуговуються на 200 мс швидше, коли Java замінено на NodeJS.

Поточна версія -23.9.0.

GitHub – 110 тис.+ зірок, 30,8 тис.+ форків, 3562+ учасників.

NPM – Node.js зафіксував понад 222 тисячі завантажень на NPM.

Плюси використання Node js:

- із зростанням ефективності зростатиме вартість серверів і персоналу, необхідного для їх експлуатації. Крім того, можна уникнути втрати продуктивності, пов'язаної з масштабуванням. Хоча більшість розробників віддають перевагу запуску Node на масштабованих хмарних кластерах, завдання можна виконати за допомогою сервера VPS або недорогого виділеного сервера [14]-[15];

- може зменшити витрати на розробку до 58%, і він розвивається швидше, ніж будь-який інший фреймворк JavaScript [14]-[15];

- організації, які використовують Node.js більше двох років, спостерігали зниження витрат на розробку на 12%.

Мінуси використання Node js:

- Node.js використовує цикл подій для паралелізму та працює в одному потоці. Це дизайнерське рішення дає змогу ефективно обробляти багато з'єднань, але це також означає, що тривалі процеси блокування призупинять весь процес і перешкоджатимуть обробці додаткових запитів. Через це можуть виникати вузькі місця продуктивності, особливо в програмах, які використовують багато процесора або мають багато операцій вводу/виводу;

- у Node.js немає рідного методу для багатопоточності або паралельної обробки. Він однопотоковий і працює на одному процесі, що ускладнює використання багатоядерних систем з максимальним потенціалом;

– JavaScript, який використовується Node.js і відомий своєю універсальністю та зручністю використання, може бути складним для розробників, які не знайомі з ним. Оскільки JavaScript відрізняється від інших широко використовуваних мов програмування, таких як Java, C++ і Python.

Відомі вебсайти, створені за допомогою Node.js: LinkedIn, eBay, Netflix, GoDaddy, Groupon, Uber, NASA, Walmart тощо [14]-[15].

2. Next.js – це наскрізний серверний фреймворк візуалізації на основі React. Next.js створила голландська компанія Vercel (раніше відома як ZEIT) у 2017 році. Хоча і Next.js, і Gatsby базуються на React, між ними є одна ключова відмінність. У той час як Gatsby є генератором статичного сайту, Next є рендерером на стороні SSR-сервера [14].

Next.js також пропонує статичні генератори. Next.js створено за принципом «Створіть один раз, працює всюди», що означає, що ви можете створювати вебпрограми за допомогою Next.js, мобільні програми, програми для настільних комп'ютерів і прогресивні вебпрограми [14]-[15].

Next.js займає перше місце серед найбільш використовуваних внутрішніх фреймворків JavaScript у звіті про стан JS за 2022 рік, випереджаючи Express і Nuxt [14]-[15].

48,6% опитаних розробників JavaScript зараз використовують Next.js. Але ще більше вражає те, що колосальні 65,2% розробників зацікавлені в спробі наступного.

Поточна версія – v15.2.1.

GitHub – 130 тис.+ зірок, 27,8 тис.+ форків, 3548+ учасників.

NPM – Next має близько 8,78 млн+ завантажень на тиждень на NPM.

Плюси використання Next.js:

– візуалізація на стороні сервера – Next.js має SSR (візуалізація на стороні сервера) із коробки, щоб забезпечити набагато швидшу роботу, оскільки йому не потрібно чекати, поки браузер клієнта завантажить JavaScript, щоб показати вміст. SSR робить Next.js одним із найкращих фреймворків JavaScript для внутрішньої розробки [14]-[15];

- зручність для SEO – традиційні клієнтські програми мають низьку ефективність для SEO. Роботи пошукових систем стикаються з труднощами під час індексації таких сторінок. Однак Next.js із можливістю візуалізації на стороні сервера забезпечує чудову продуктивність SEO. Next.js дозволяє нам створювати динамічні мета-теги в розділі head для оптимізації SEO;

- автоматичне поділ коду – ця функція в Next.js дозволяє нам розбивати код нашої програми на низку невеликих пакетів, завантажуваних окремо за потреби, без завершення всього коду JavaScript. Це допомагає забезпечити надзвичайно швидку оптимальну продуктивність [14]-[15];

- попередня вибірка – ця функція в Next.js дозволяє згаданим вище легким пакетам коду JavaScript відкладено завантажуватись поза сценою;

- перезавантаження гарячого коду – Next.js автоматично перезавантажує сторінку, коли виявляється зміна [14]-[15];

- розгортайте скрізь – за допомогою Next.js ви можете створювати невеликі статичні вебсайти та корпоративні програми. Крім того, ви можете створювати не лише вебпрограми, але й мобільні програми, настільні програми, а також прогресивні вебпрограми [14], [15].

Мінуси використання Next.js:

- Gatsby може бути кращим варіантом, ніж Next.js, для невеликих блогів, де статична збірка забезпечує набагато сприятливішу та кращу продуктивність, ніж рендеринг на сервері Next.js;

- тести продуктивності Next.js у більшості випадків поступаються Gatsby і Nuxt [14]-[15];

- для надання даних потрібна сторона внутрішнього сервера;

- Next.js створено спеціально для роботи лише з React [14]-[15].

Відомі вебсайти, створені за допомогою Next.js: Auth0, Binance, Coinbase, Docker, GitHub, Hulu, Invision, Magic Leap, Netflix, Sesame, Starbucks, Trulia, Tencent, Twitch і Uber [14]-[15].

3. Express, також відомий як Express.js, – це мінімальний фреймворк JavaScript із відкритим кодом для Node.js, який використовується для

створення API та вебдодатків. Він настільки популярний, що вважається одним із найкращих фреймворків JavaScript для серверної роботи з Node.js, особливо в стеках MEAN або MERN. З майже 48 тисячами зірок, це найпопулярніший фреймворк Node.js на GitHub [14]-[15].

TJ Holowaychuk створив Express у 2010 році, перш ніж його придбала IBM (StrongLoop) у 2015 році. Зараз його підтримує Node.js Foundation. Основною причиною того, що Express є одним із найкращих фреймворків JavaScript, є його швидке кодування на стороні сервера. Складні завдання, кодування яких займе години за допомогою чистого Node.js, можна вирішити за кілька хвилин завдяки Express. Крім того, Express пропонує надійний механізм маршрутизації для обробки дуже динамічних URL-адрес порівняно з базовим і рудиментарним механізмом, який пропонує Node.js.

Згідно з опитуванням розробників Stack Overflow у 2023 році, Express займає 4-е місце серед найпопулярніших вебфреймворків у цілому та серед професіоналів [14]-[15].

Поточна версія – 4.21.2.

GitHub – 66,4 тис.+ зірок, 17,7 тис.+ форків, 314+ учасників.

NPM – Express набирає вражаючі 36,66 млн+ завантажень на тиждень на NPM.

Плюси використання Express:

- підтримка спільноти – Express – це зріла 13-річна структура з величезною підтримкою спільноти [14]-[15];
- хороша документація – Express був благословенний добре написаною та ретельною документацією та багатьма навчальними посібниками та ресурсами [14]-[15];
- швидка розробка – спрощує та прискорює розробку вебдодатків за допомогою Node.js [14]-[15];
- проміжне програмне забезпечення – Express технічно є серією викликів функцій проміжного програмного забезпечення. Функції проміжного програмного забезпечення виконують різні завдання, як-от виконання коду,

внесення змін до об'єктів запиту та відповіді та завершення циклу запит-відповідь. Express має п'ять типів проміжного програмного забезпечення – на рівні програми, на рівні маршрутизатора, для обробки помилок, вбудоване та стороннє проміжне програмне забезпечення [14]-[15];

- маршрутизація – пропонує потужний механізм маршрутизації для визначення маршрутів на основі URL-адрес і методів HTTP;

- швидка інтеграція з базою даних – Express пропонує зручне підключення до основних популярних баз даних, таких як MongoDB, MySQL і Redis [14]-[15];

- інтеграція з механізмами шаблонів – усі популярні механізми шаблонів, як-от Pug, HAML і EJS, одразу працюють із Express.

Мінуси використання Express:

- забезпечення безпеки вебпрограми повністю залежить від розробника та якості коду. Express сам по собі не пропонує жодного рішення безпеки [14]-[15];

- незважаючи на те, що в більшості випадків це не є недоліком, особливо досвідченим розробникам може бути складно освоїти Express через його бездумність. Немає чітко визначеного структурованого способу виконання завдань для початківців. Хоча досвідчені розробники вважатимуть необачність великою перевагою, яка пропонує більше гнучкості та свободи.

Відомі вебсайти, створені за допомогою Express: Accenture, Coursera, Exove, Fox, Godaddy, IBM, Myntra, Nike, PayPal, Pluralsight, QuizUp, Ripjar, Sony PlayStation, Twitter, Uber, Unsplash, Yandex [14]-[15].

4. Nuxt – це прогресивний фреймворк із відкритим вихідним кодом, заснований на екосистемі Vue.js, який використовується для створення ефективних вебдодатків, особливо програм, які відображаються на стороні сервера. Його можна вважати покращувачем для Vue.js. Але пам'ятайте, що Nuxt не є заміною для Vue.js, оскільки він не може працювати окремо. І також не можна вважати його повноцінним серверним фреймворком, таким як Express [15].

Nuxt. поєднує офіційні бібліотеки та компоненти Vue.js – Vue, Vue Router, Vuex, Vue server renderer і Vue meta, а також укомплектовує майстерність webpack babel і PostCSS. Nuxt є одним із найкращих фреймворків JavaScript для створення цих трьох типів вебдодатків – попередньо відрендерених статичних сторінок, односторінкових вебдодатків (SPA) і вебдодатків, відтворених на стороні сервера (SSR) або універсальних додатків.

Посідає 4 місце в рейтингу найкращих фреймворків JavaScript для бекенда в опитуванні State of JS 2022 [14]-[15].

14,4% опитаних розробників JavaScript зараз використовують Nuxt, тоді як 30,5% чули про нього та бажають спробувати в майбутньому [14]-[15].

GitHub – 56,3 тис.+ зірок, 5,2 тис.+ форків, 812+ учасників.

NPM – Nuxt зафіксував у середньому 746 тисяч+ завантажень на NPM.

Плюси використання Nuxt:

- швидке налаштування та розробка. Налаштування нового проєкту в Nuxt займає лише кілька хвилин завдяки стартовому шаблону, який можна легко створити за допомогою командного рядка [14]-[15];

- створити універсальну програму – Nuxt може створювати універсальні програми – код JavaScript, який працює як на стороні клієнта, так і на сервері, що зміцнює позицію однієї з найкращих фреймворків JavaScript. У той час як традиційні односторінкові програми страждають від повільного часу завантаження та низької продуктивності SEO, універсальні програми забезпечують величезне підвищення швидкості завантаження та покращують SEO, роблячи вміст доступним швидше для пошукових роботів [14]-[15];

- структура проєкту – Nuxt за замовчуванням організовує ваш код у логічну, масштабовану та просту для розуміння структуру прямо з коробки. Основні каталоги, встановлені за замовчуванням: ресурси, компоненти, макети, сторінки, плагіни та магазини [14]-[15];

- розділення коду – Nuxt генерує статичну версію вашої програми з найефективнішою конфігурацією вебпакета. У результаті кожна статично згенерована сторінка має свій файл JavaScript. Це допомагає розділити

загальний код JavaScript програми на менші файли, підвищуючи швидкість і загальну продуктивність [14]-[15].

Мінуси використання Nuxt:

- можна зіткнутися з проблемами інтеграції спеціальних бібліотек із Nuxt [12]-[13];
- проблеми з налагодженням під час використання Nuxt часто можуть бути складними та неприємними [14]-[15];
- розмір спільноти, яка підтримує Nuxt, відносно невеликий порівняно з Next.js або Gatsby [14]-[15];
- документація не дуже обширна, і немає великої кількості хороших ресурсів [14]-[15].

Відомі вебсайти, створені за допомогою Nuxt: Aircall, Amplitude, Backmarket, Bitpay, Bootstrap Vue, Fox News, Gitlab, Icons8, Instrument, MyScript, Nespresso, Note.com, Ozon.ru, Roland Garros, System76, Todoist, Upwork, Wappalyzer [14]-[15].

В результаті аналізу популярних фреймворків для розробки Backend частини вебсайту було обрано фреймворк Node.js.

2.4 Висновки за розділом 2

Другий розділ пояснювальної записки до роботи було присвячено висвітленню матеріалів та методів, що буде використано для програмної реалізації застосунку для створення віддалених завдань. Було досліджено, обрано та аргументовано вибір мови програмування, фреймворків для розробки FrontEnd та Backend частин вебсайту.

В результаті аналізу популярних фреймворків для розробки FrontEnd частини вебсайту було обрано фреймворк React, для розробки Backend частини вебсайту було обрано фреймворк Node.js. В якості мови програмування для розробки додатку було обрано JavaScript.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення

Загальна структурна схема системи за обраною темою представлена на рисунку 3.1.

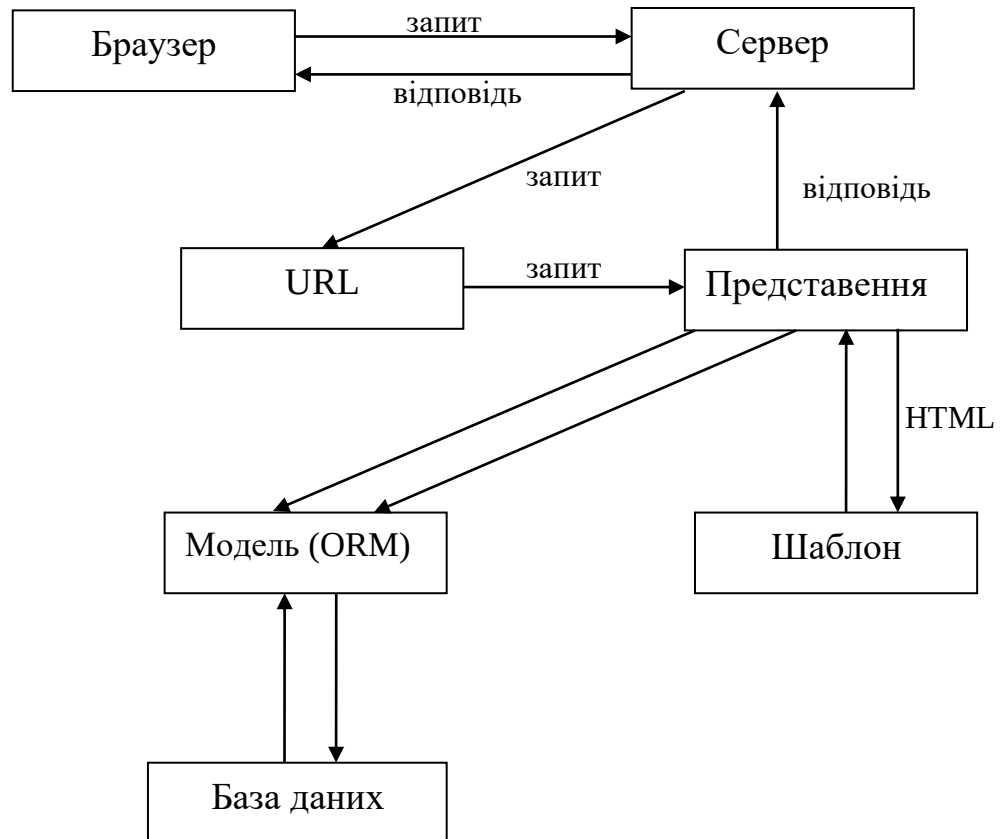


Рисунок 3.1 – Загальна структурна схема системи за обраною темою

Ця структурна схема демонструє архітектуру вебдодатку на основі шаблону MVC (Model-View-Controller).

Загальний потік запиту:

- браузер надсилає запит на сервер;
- сервер отримує запит і передає його до URL диспетчера (роутера);
- URL визначає, яке представлення (View/Controller) має обробити запит, і передає його до Представлення.

Обробка логіки:

- представлення звретається до Моделі (ORM), щоб отримати або змінити дані в Базі даних;
- представлення звретається може використати Шаблон, щоб згенерувати HTML на основі отриманих даних.

Відповідь:

- HTML, згенерований у Шаблоні, повертається через Представлення назад до Сервера, а потім – у Браузер як відповідь.

Компоненти:

- браузер – клієнтський інтерфейс користувача;
- сервер – обробляє запити та надсилає відповіді;
- URL – маршрутизація запитів;
- представлення – контролер, логіка обробки запитів;
- модель (ORM) – взаємодія з базою даних через об'єкти;
- база даних – зберігання даних;
- шаблон – генерація HTML-вмісту для користувача.

Розроблювана системи складається з наступних модулів, призначення кожного з модулів представлено у таблиці 3.1.

Таблиця 3.1 – Опис модулів

Назва директорії	Опис модулю
controllers	Модуль який відповідає за основний функціонал серверу.
models	Модуль призначений для опису моделей бази даних.
components	Тут знаходиться основна візуальна частина сайту
sagas	Модуль який відповідає за запити до сервера
public	Модуль який зберігає усі файли, які будуть додані до сайту

3.2 Розробка бази даних

Для розробки даного web-додатку використано об'єктно-орієнтовану модель бази даних. База даних розроблена в СКБД PostgreSQL [16]-[17].

Об'єктно-орієнтована модель – модель, що реалізує об'єктно-орієнтований підхід. Вона обробляє дані як абстрактні об'єкти, наділені властивостями, у вигляді неструктурованих даних, і використовуючи методи взаємодії з іншими об'єктами навколишнього світу [16]-[17].

Реляційна база даних – це набір даних із зумовленими зв'язками з-поміж них. Ці дані організовані як набір таблиць, з стовпців і рядків. У таблицях зберігається інформація про об'єкти, які у є базі даних. У кожному стовпці таблиці зберігається певний тип даних, у кожному осередку – значення атрибута. Кожен рядок таблиці є набір пов'язаних значень, які зв'язані один з одним. Кожен рядок у таблиці може бути унікальним ідентифікатором, який називається первинним ключем, а рядки з кількох таблиць можуть бути пов'язані за допомогою зовнішніх ключів. До цих даних можна отримати доступ багатьма способами і при цьому реорганізовувати таблиці БД не потрібно [16]-[17].

Об'єкт – сукупність властивостей, включаючи унікальний ідентифікатор id [16]-[17].

Властивість – сукупність назви і відповідного йому типу і значення.

Типи властивостей – рядок, ціле число, число з плаваючою точкою, масив, об'єкт, бінарна рядок, байт, символ, дата, boolean, null та інші [16]-[17].

В процесі проєктування даного програмного продукту для побудови об'єктно-орієнтованої моделі було виділено наступні об'єктні множини: «catalogs», «conversations», «messages», «category», «Ratings», «Selects», «Contests», «Users», «Offers», «RefreshTokens», «CreditCards».

Аналіз визначених об'єктів і атрибутів дозволяє виділити сутності бази даних і побудувати її логічну схему.

Спроектowana об'єктно-орієнтована модель зображена на рисунку 3.2.

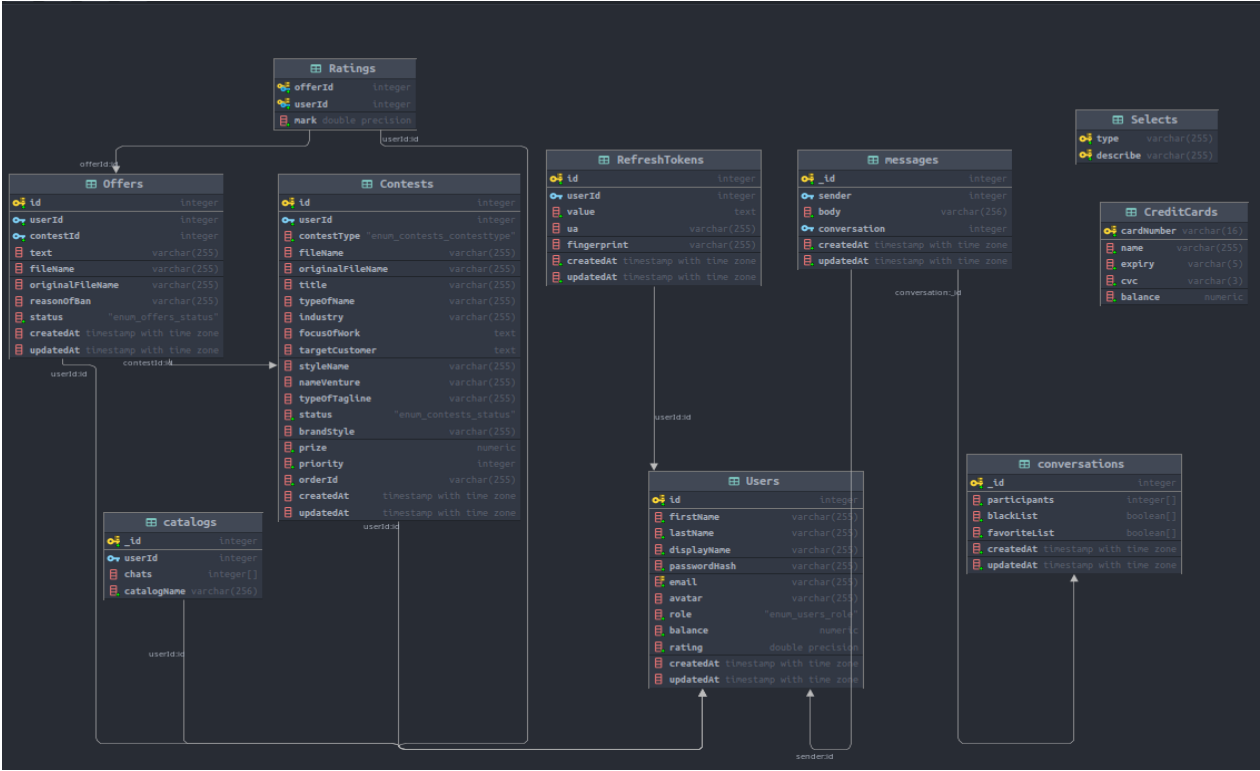


Рисунок 3.2 – Об'єктно-орієнтована модель бази даних

Таблиця «catalogs» зберігає інформацію про усі текстові каталоги. Опис таблиці «catalogs» представлений у таблиці 3.2.

Таблиця 3.2 – Опис таблиці «catalogs»

Назва ключа	Тип значення	Опис
<code>_id</code>	Integer	Забезпечує унікальність документа
<code>userId</code>	ForeignKey	Посилання на користувача.
<code>chats</code>	Integer	Чати в каталозі.
<code>catalogName</code>	Varchar	Назва каталогу.

Таблиця «Ratings» зберігає інформацію про успішність користувачів. Опис таблиці «Ratings» представлений у таблиці 3.3.

Таблиця 3.3 – Опис таблиці «Ratings»

Назва ключа	Тип значення	Опис
offerId	ForeignKey	Посилання на пропозицію користувача.
userId	ForeignKey	Посилання на користувача.
mark	Double	Оцінка пропозиції.

Таблиця «conversations» зберігає інформацію про чати. Опис таблиці «conversations» представлений в таблиці 3.4.

Таблиця 3.4 – Опис таблиці «conversations»

Назва ключа	Тип значення	Опис
_id	Integer	Забезпечує унікальність документа
participants	Integer[]	Користувачі
blackList	Boolean[]	Чи є користувачі в чорному списку
favoriteList	Boolean[]	Чи є користувачі в списку улюблених
createdAt	Timestamp	Дата створення запису
updatedAt	Timestamp	Дата останнього оновлення запису

Таблиця «Selects» зберігає інформацію про всі можливі поля вибору. Опис таблиці «Selects» представлений в таблиці 3.5.

Таблиця 3.5 – Опис таблиці «Selects»

Назва ключа	Тип значення	Опис
type	Varchar	Тип поля
describe	Varchar	Текст поля

Таблиця «messages» зберігає інформацію про чати. Опис таблиці «messages» представлений в таблиці 3.6.

Таблиця 3.6 – Опис таблиці «conversations»

Назва ключа	Тип значення	Опис
_id	Integer	Забезпечує унікальність документа
sender	ForeignKey	Посилання на користувача
body	Varchar	Чи є користувачі в чорному списку
conversation	ForeignKey	Посилання на чат
createdAt	Timestamp	Дата створення запису
updatedAt	Timestamp	Дата останнього оновлення запису

Таблиця «Contests» зберігає інформацію про конкурси. Опис таблиці «Contests» представлений в таблиці 3.7.

Таблиця 3.7 – Опис таблиці «Contests»

Назва ключа	Тип значення	Опис
id	Integer	Забезпечує унікальність документа
userId	ForeignKey	Посилання на користувача
contestType	Enum	Тип конкурсу
fileName	Varchar	Назва файлу
originalFileName	Varchar	Назва файлу що зберігається на сервері
title	Varchar	Заголовок
typeName	Varchar	Тип назви
focusOfWork	Text	Основна робота
targetCustomer	Text	Основні користувачі
styleName	Varchar	Назва стилю
nameVenture	Varchar	Назва підприємства
typeOfTagline	Varchar	Тип слогану
status	Enum	Статус конкурсу
brandStyle	Varchar	Стиль бренду
price	Numeric	Ціна за конкурс

Продовження таблиці 3.7

Назва ключа	Тип значення	Опис
priority	Integer	Пріоритет конкурс
orderId	Varchar	Унікальність запису
createdAt	Timestamp	Дата створення запису
updatedAt	Timestamp	Дата останнього оновлення запису

Таблиця «Users» зберігає інформацію про користувачів. Опис таблиці «Users» представлений в таблиці 3.8.

Таблиця 3.8 – Опис таблиці «Users»

Назва ключа	Тип значення	Опис
id	Integer	Забезпечує унікальність документа
firstName	Varchar	Ім'я користувача
lastName	Varchar	Прізвище користувача
displayName	Varchar	Нікнейм
passwordHash	Varchar	Хеш пароля
avatar	Varchar	Фото користувача
email	Varchar	Електронна пошта
role	Enum	Роль користувача
balance	Numeric	Кількість «поінтів» користувача
rating	Double	Рейтинг користувача
createdAt	Timestamp	Дата створення запису
updatedAt	Timestamp	Дата останнього оновлення запису

Таблиця «Offers» зберігає інформацію про пропозиції користувачів. Опис таблиці «Offers» представлений в таблиці 3.9.

Таблиця 3.9 – Опис таблиці «Offers»

Назва ключа	Тип значення	Опис
id	Integer	Забезпечує унікальність документа
userId	ForeignKey	Посилання на користувача
contestId	ForeignKey	Посилання на конкурс
text	Varchar	Текст пропозиції
fileName	Varchar	Назва файлу
originalFileName	Varchar	Назва файлу що зберігається на сервері
reasonOfBan	Varchar	Причина блокування
status	Enum	Статус пропозиції
createdAt	Timestamp	Дата створення запису
updatedAt	Timestamp	Дата останнього оновлення запису

Таблиця «CreditCards» зберігає інформацію про банківські дані користувачів. Опис таблиці «CreditCards» представлений в таблиці 3.10.

Таблиця 3.10 – Опис таблиці «CreditCards»

Назва ключа	Тип значення	Опис
cardNumber	Varchar	Номер карти
name	Varchar	Ім'я власника карти
expiry	Varchar	Дата закінчення строку дії карти
cvc	Varchar	CVC код карти
balance	Numeric	Баланс карти

Таблиця «RefreshTokens» зберігає інформацію про пропозиції користувачів. Опис таблиці «RefreshTokens» представлений в таблиці 3.11.

Таблиця 3.11 – Опис таблиці «RefreshTokens»

Назва ключа	Тип значення	Опис
id	Integer	Забезпечує унікальність документа
userId	ForeignKey	Посилання на користувача
value	Text	Дані токена
ua	Varchar	Дані для роз шифровки токена
fingerprint	Varchar	Відбиток пальця
createdAt	Timestamp	Дата створення запису
updatedAt	Timestamp	Дата останнього оновлення запису

3.3 Основний алгоритм роботи програмного забезпечення

Основний алгоритм роботи програмного забезпечення зображений на рисунку 3.3. Програма створена на основі MVC моделі і алгоритм її роботи базується на принципах цієї моделі.

MVC – це аббревіатура від Model-View-Controller, і це архітектурний шаблон, який використовується для побудови структурованих і зручних у підтримці програм. Його головна мета – розділити логіку програми на три незалежні частини, щоб зробити розробку, тестування та масштабування більш зручними [18].

Model (Модель) – це частина, яка відповідає за роботу з даними: зберігає бізнес-логіку, взаємодіє з базою даних, перевіряє й обробляє дані [18].

View (Подання) – це інтерфейс користувача, отримує дані від контролера/моделі і показує їх, не містить логіки, або обробки запитів [18].

Controller (Контролер) – це посередник між користувачем, поданням і моделлю, приймає вхідні запити, визначає, що з ними робити, викликає модель для обробки даних, вибирає, яке подання показати у відповідь [18].

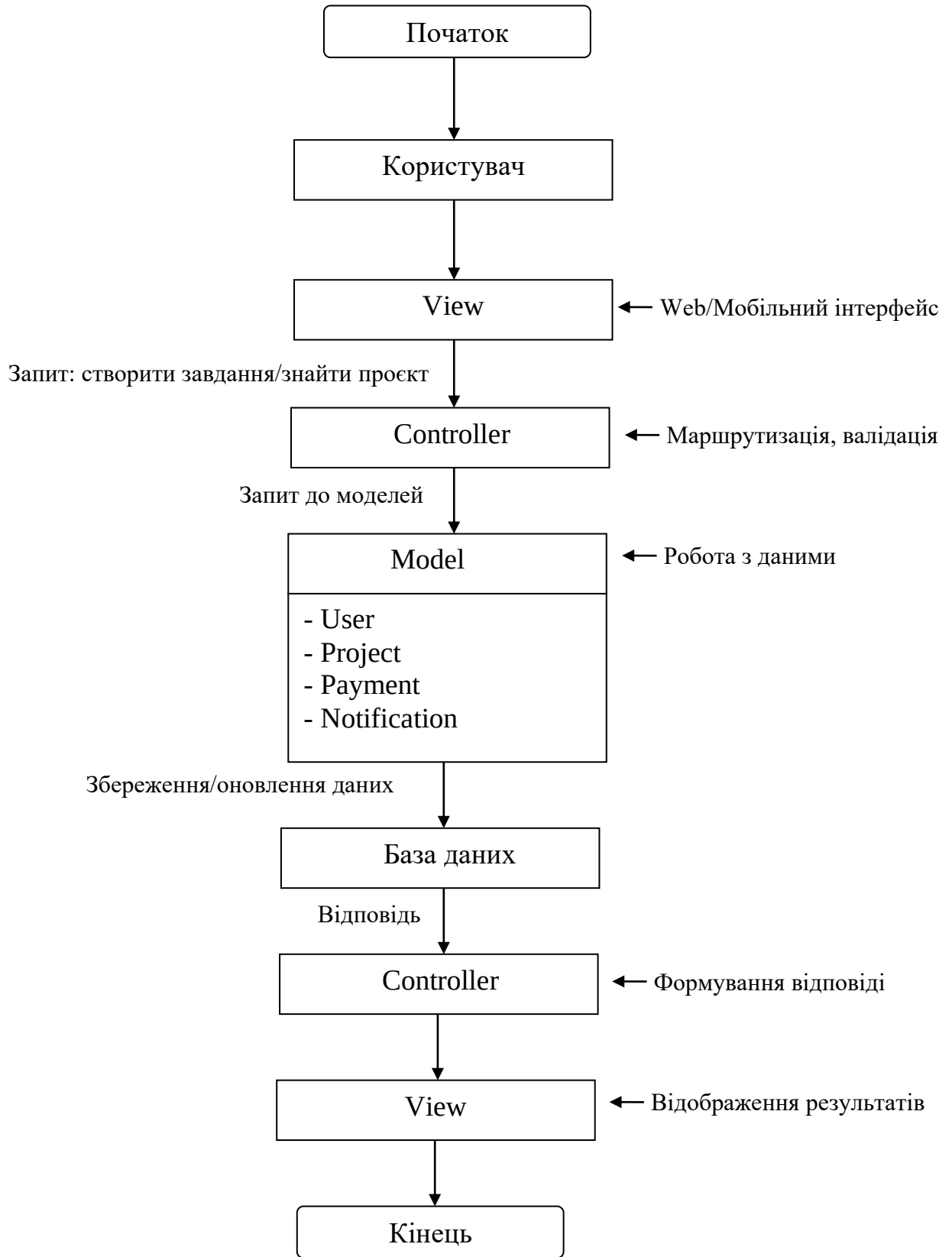


Рисунок 3.3 – Основний алгоритм роботи програмного забезпечення

Опис алгоритму. Користувач фрілансер, або замовник взаємодіє з платформою через Вебінтерфейс. На етапі представлення (View) відображається форма проєкту, список доступних завдань та інше.

Коли алгоритм доходить до етапу «Controller» відбувається валідація даних, автентифікація та маршрутизація запитів.

На етапі «Model» (Моделі даних) відбувається для моделі «User» ініціалізуються дані користувачів (полі: фрілансер/замовник), методи: register(), login(). Для моделі «Project» ініціалізуються поля (title, description, budget, deadline, status) та методи: create(), assign_freelancer().

Для моделі «Payment» відбувається інтеграція з платіжними системами та ініціалізується Метод: process_payment(). Для моделі «Notification» відбувається відправка email/повідомлень про нові заявки.

Далі «Model» оновлює інформацію в базі даних. Формується відповідь, яка передається в «Controller», який в свою чергу передає результат у «View». «View» відображає результат користувачу.

3.4 Проєктування інтерфейсу

На сучасному етапі розвитку програмного забезпечення велика увага приділяється інтерфейсу користувача. Саме інтерфейс у значній мірі впливає на успішність програмного продукту на ринку. Він має бути інтуїтивно зрозумілим, зручним у користуванні, містити помірну кількість кольорів, а також забезпечувати користувача достатнім обсягом інформації для розуміння стану автоматизованих процесів. При цьому важливо уникати перевантаження екрану надмірною кількістю елементів. Також варто враховувати естетичні аспекти оформлення та дотримуватися загальноприйнятих стандартів побудови інтерфейсів [19].

Інтерфейс даного програмного продукту реалізований з використанням бібліотеки React, мультипарадигмальної мови програмування JavaScript, мови розмітки HTML5 та каскадних таблиць стилів CSS3.

Основні компоненти інтерфейсу представлені наступними сторінками:

- Home – головна сторінка;
- LoginPage – сторінка входу до системи;
- RegistrationPage – сторінка реєстрації нового користувача;
- Dashboard – сторінка для перегляду конкурсів;
- StartContestPage – сторінка створення нового конкурсу.

На рисунку 3.4 наведено концепт сторінки головної сторінки вебсайту.

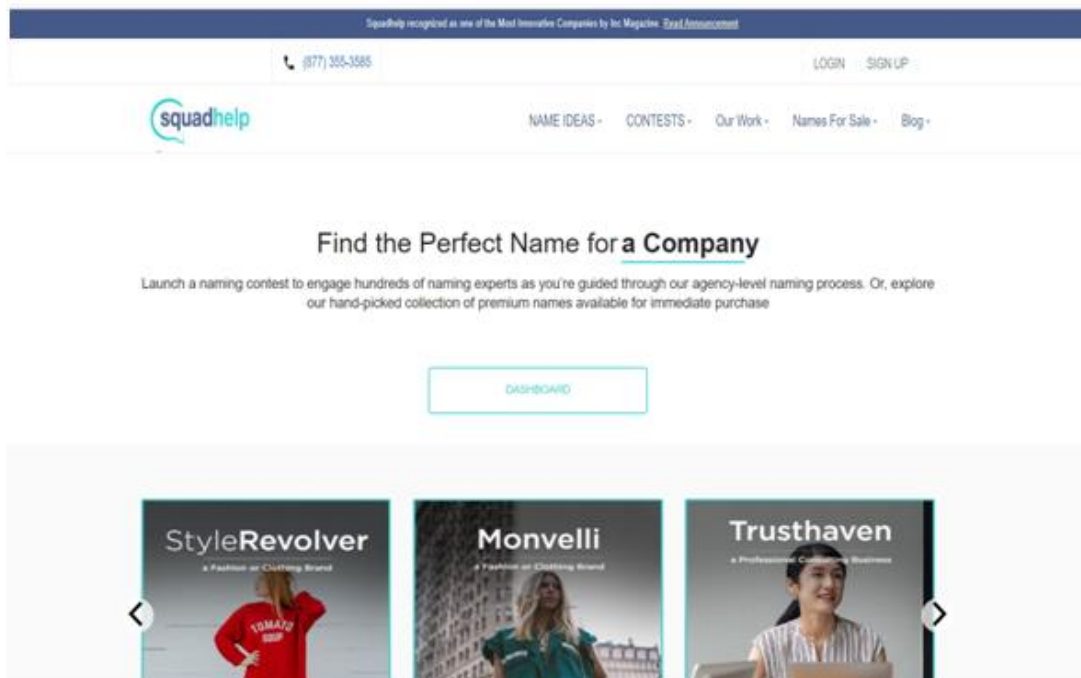


Рисунок 3.4 – Концепт головної сторінки вебсайту

3.5 Висновки за розділом 3

Розроблено структуру програмного забезпечення для створення віддалених завдань. Спроектвана структура бази даних додатку. Розроблено основний алгоритм роботи програмного забезпечення для створення віддалених завдань. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для створення віддалених завдань. Розроблено програмне забезпечення для створення віддалених завдань.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Метою роботи є розробка сайту для створення віддалених завдань. Користувач сайту має можливість знаходити та переглядати або створювати пропозиції, які його зацікавляють, а також оцінювати їх та виконавців, також має можливість спілкуватися с ними. Виконавці мають можливість знаходити необхідні їм замовлення та приймати участь у їх виконанні, та отримувати винагороду в випадку якщо їх пропозиція переможе.

Сайт має наступні функції:

- відображення інформації про пропозиції та замовлення;
- можливість авторизації на сайті;
- пошук за різними категоріями;
- модерація пропозицій;
- чат між виконавцем та замовником;
- створення конкурсів.

Сайт розроблений в середовищі IntelliJIdea на мові JavaScript (React, Node), HTML, CSS.

Однаково швидко працює на всіх платформах та пристроях, адже сайт є адаптивним, має зрозумілий інтерфейс, що відповідає сучасним стандартам дизайну.

Для запуску платформи у програміста має бути встановлені такі компоненти:

- Docker;
- Docker-Compose;
- Node.js v. 14.16.1;
- PostgreSQL v. 13.0.

4.2 Умови виконання програми

Для функціонування сайту, а також в разі його доопрацювання, необхідно мати таку мінімальну апаратну платформу: ПК з тактовою частотою мікропроцесора не менше 2 ГГц, з оперативною пам'яттю з мінімальним об'ємом 4 Гб, з дисковим простором 5 Гб, необхідним для інсталяції браузера. На персональному комп'ютері повинно бути встановлено наступне програмне забезпечення: операційна система, браузер.

4.3 Виконання програмного забезпечення

Платформа призначена для створення віддалених завдань.

Користувач сайту має можливість знаходити та переглядати, або створювати пропозиції, які його зацікавляють, а також оцінювати їх та виконавців, також має можливість спілкуватися с ними. Виконавці мають можливість знаходити необхідні їм замовлення та приймати участь у їх виконанні, та отримувати винагороду в випадку якщо їх пропозиція переможе.

Спочатку розберемо функції сайту від обличчя звичайного користувача. Для початку роботи з сайтом потрібно перейти за посиланням <http://localhost:5000>, перед нами відкриється головна сторінка сайту його шапка (рисунок 4.1), головний блок на (рисунок 4.2), та розділ контактів зображений на рисунку 4.3.



Рисунок 4.1 – Шапка головної сторінки сайту

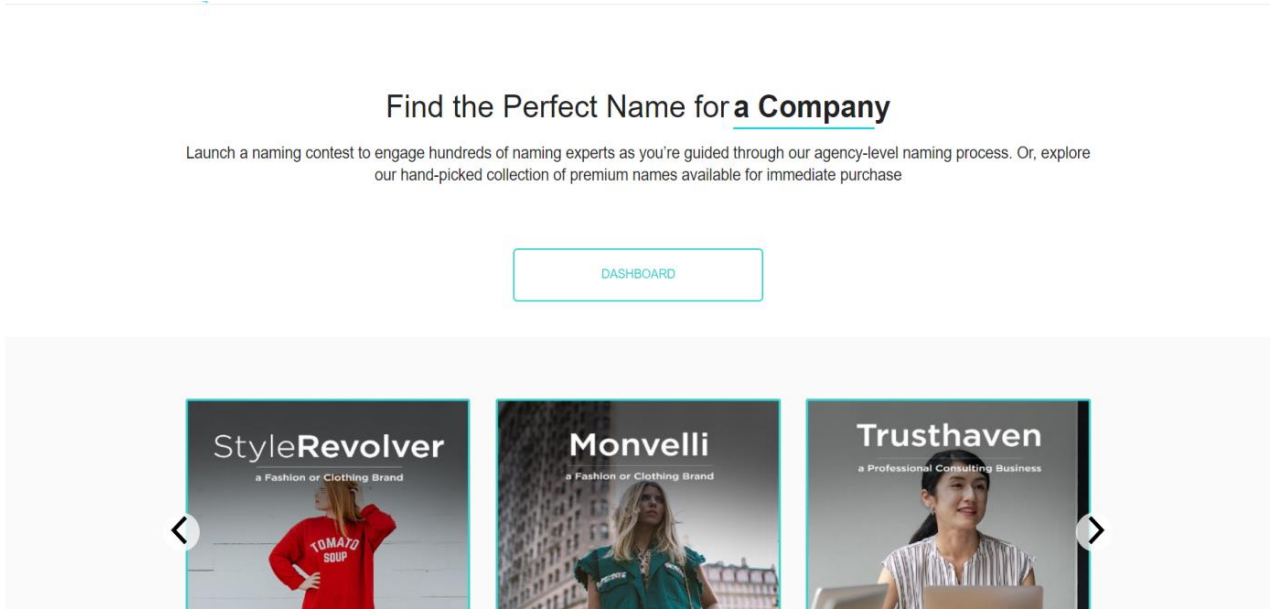


Рисунок 4.2 – Головна сторінка сайту

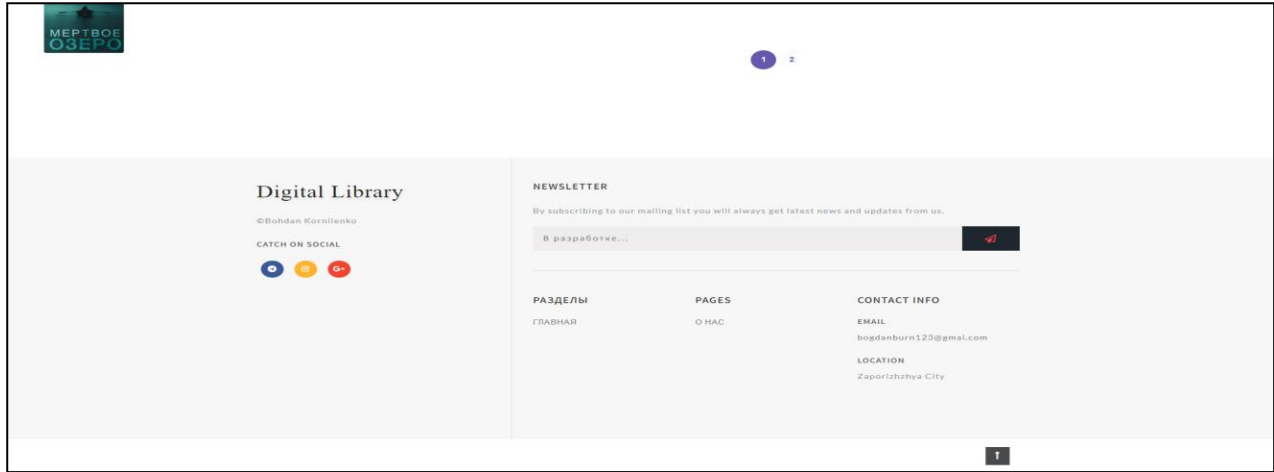


Рисунок 4.3 – Розділ контактів

На рисунку 4.4 представлена сторінка авторизації. Якщо форма буде заповнена вірно, її рамка стане зеленою (рисунок 4.5), якщо ні то червоною, і виведеться помилка (рисунок 4.6).

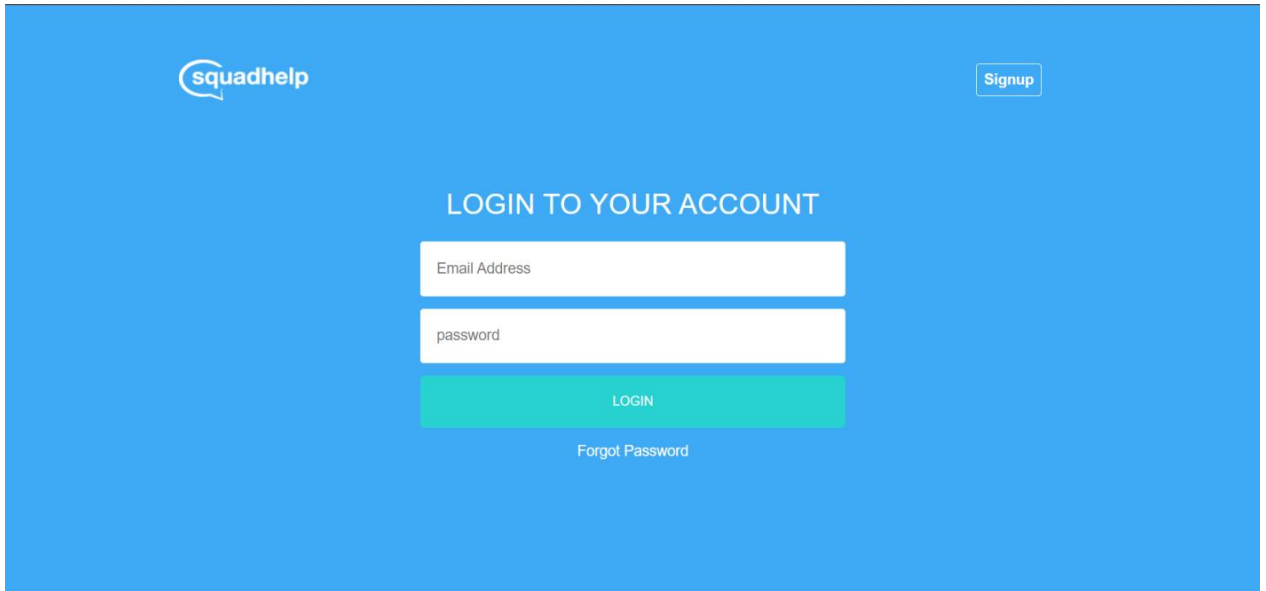


Рисунок 4.4 – Сторінка авторизації

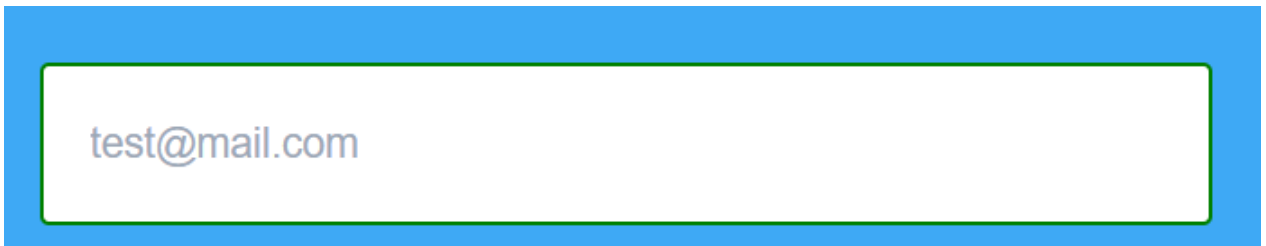


Рисунок 4.5 – Вірно заповнена форма

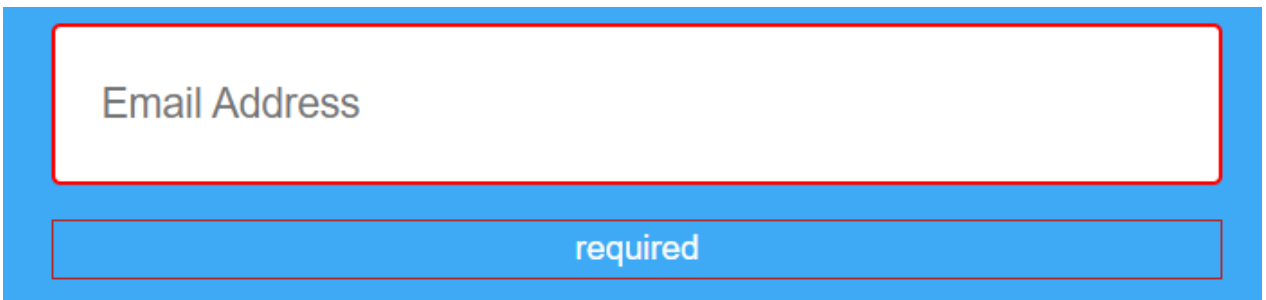


Рисунок 4.6 – Хибно заповнена форма

Вид сторінки авторизації представлено на рисунку 4.7.

В разі якщо у користувача не має акаунту на цьому ресурсі, можна його створити, перейшовши за відповідними посиланнями (рисунок 4.8).

Рисунок 4.7 – Сторінка створення облікового запису

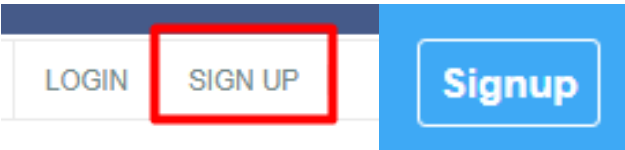


Рисунок 4.8 – Кнопки для переходу на сторінку реєстрації

Також реалізована особиста сторінка користувача її вигляд представлено на рисунку 4.9, перейти на неї можна натиснувши відповідну кнопку (рисунок 4.10).

Рисунок 4.9 – Особиста сторінка користувача

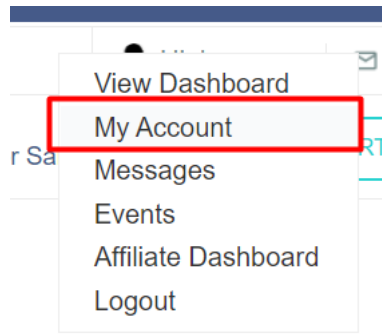


Рисунок 4.10 – Кнопка для пеходу на особисту сторінку

Якщо обліковий запис створений під роллю «customer» (рисунок 4.11) або «creator» (рисунок 4.12), сторінка «Dashboard» буде мати різний вигляд (Рисунки 4.13, 4.14).

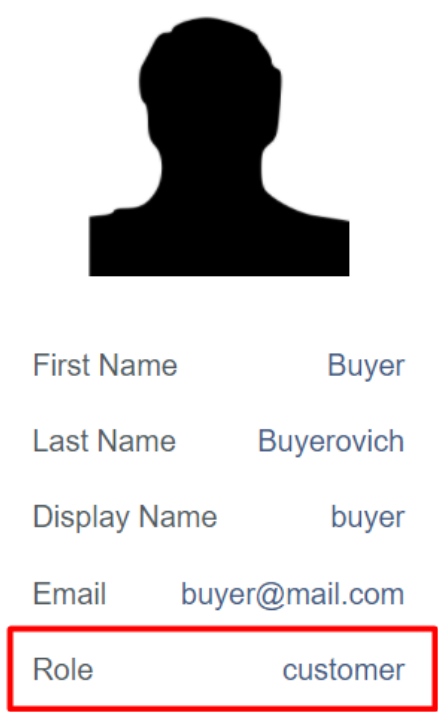


Рисунок 4.11 – Роль «customer»



First Name	Creator
Last Name	Creatorvich
Display Name	creator
Email	creator@mail.com
Role	creator
Balance	0\$

Рисунок 4.12 – Роль «creator»

Active Contests

Completed contests

Inactive contests

test (#1)

Name / Company

This is an Invitation Only Contest and is only open to those Creatives who have achieved a Tier A status.

Guaranteed prize \$100

0 Entries

14d 22h Going

Рисунок 4.13 – Вигляд сторінки «Dashboard» під роллю «customer»

Filter Results

My Entries

By contest type

name,tagline,logo

By contest ID

By industry

Choose industry

By amount award

Ascending

test (#1)

Name / Company

This is an Invitation Only Contest and is only open to those Creatives who have achieved a Tier A status.

Guaranteed prize \$100

0 Entries

14d 22h Going

Рисунок 4.14 – Вигляд сторінки «Dashboard» під роллю «creator»

Якщо натиснути на конкурс, то відкриється сторінка з детальною інформацією (рисунок 4.15), на якій можна переглянути повну інформацію про конкурс та пропозиції до нього, також якщо ви є автором конкурсу ви можете його редагувати.

Field	Value	Action
Contest Type	name	Edit
Title of the Project	test	
Type of Name	Company	
Style of Name	Classic	
What is your Business/ Brand about?	dwada	
Description target customers of company	adwadaw	
Industry of company	Creative Agency	

Рисунок 4.15 – Повна інформація про конкурс

Також буде відрізнятись сторінка пропозицій «Offers» (Рисунки 4.16, 4.17).

Offer ID	Creator	Rating	Actions
test	Creator Creatorvich creator@mail.com	☆☆☆☆☆	Resolve, Reject
test2	Creator Creatorvich creator@mail.com	☆☆☆☆☆	Resolve, Reject
test 3	Creator Creatorvich creator@mail.com	☆☆☆☆☆	Resolve, Reject

Contest Stats

Total Entries: 3

Рисунок 4.16 – Сторінка «Offers» у полі «customer»

У ролі «customer» буде показано які пропозиції зараз доступні, можна обрати, або відхилити пропозиції. Також, можливо зв'язатись з автором через особисті повідомлення (рисунок 4.18).

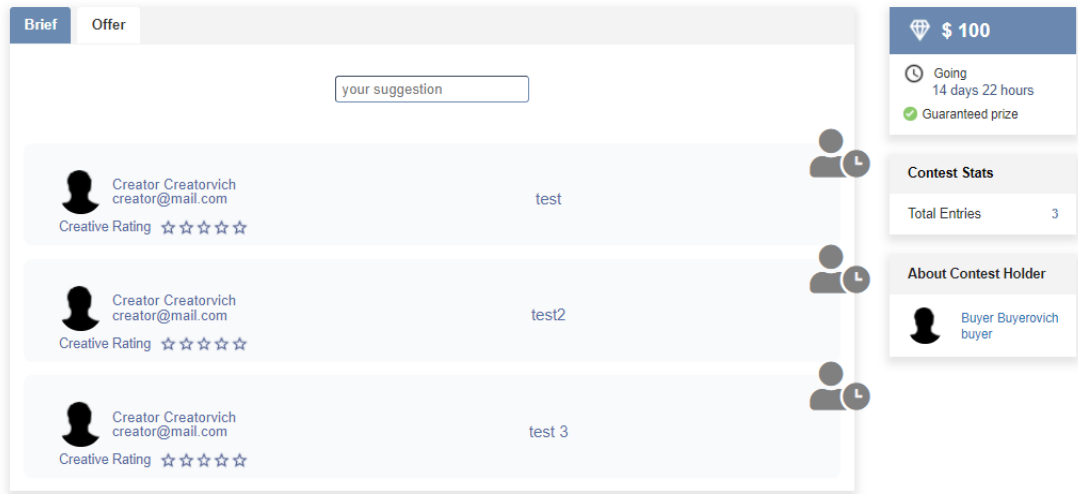


Рисунок 4.17 – Сторінка «Offers» у ролі «creator»

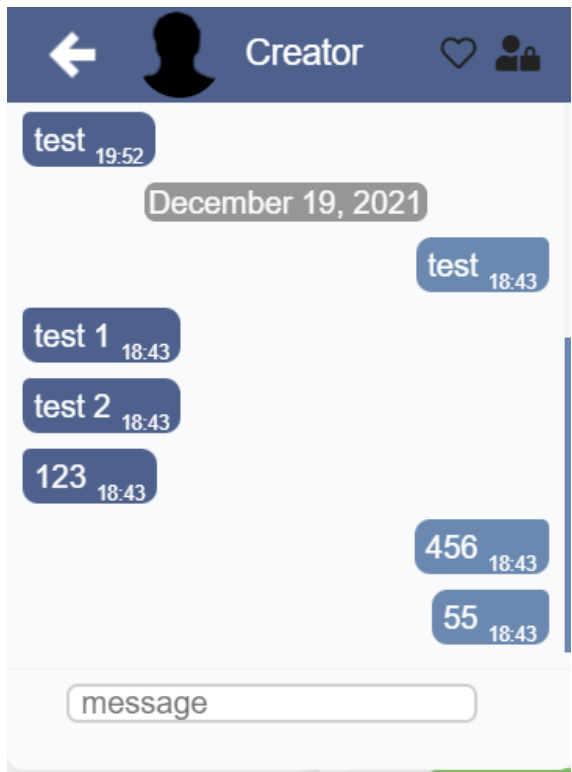


Рисунок 4.18 – Приклад особистих повідомлень

У ролі «creator» показано поточний статус пропозицій (рисунок 4.19).

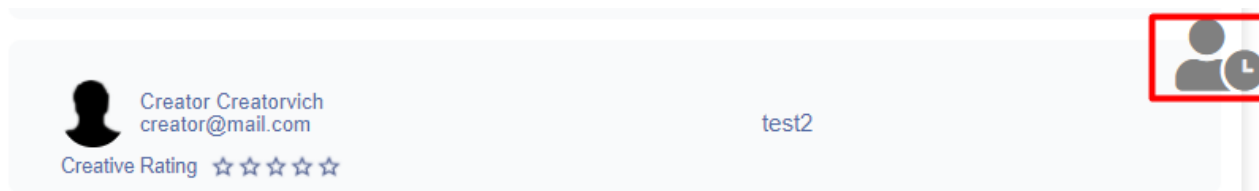


Рисунок 4.19 – Статус пропозиції

Для створення конкурсу необхідно мати роль «creator», та натиснути кнопку «Start contest» (рисунок 4.20), після чого слідуючи інструкціям заповнити форми на спеціальній сторінці (рисунок 4.21).

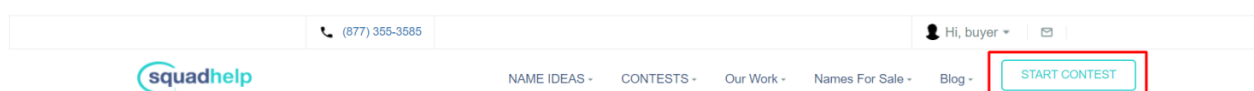


Рисунок 4.20 – Кнопка для переходу на сторінку створення конкурсу

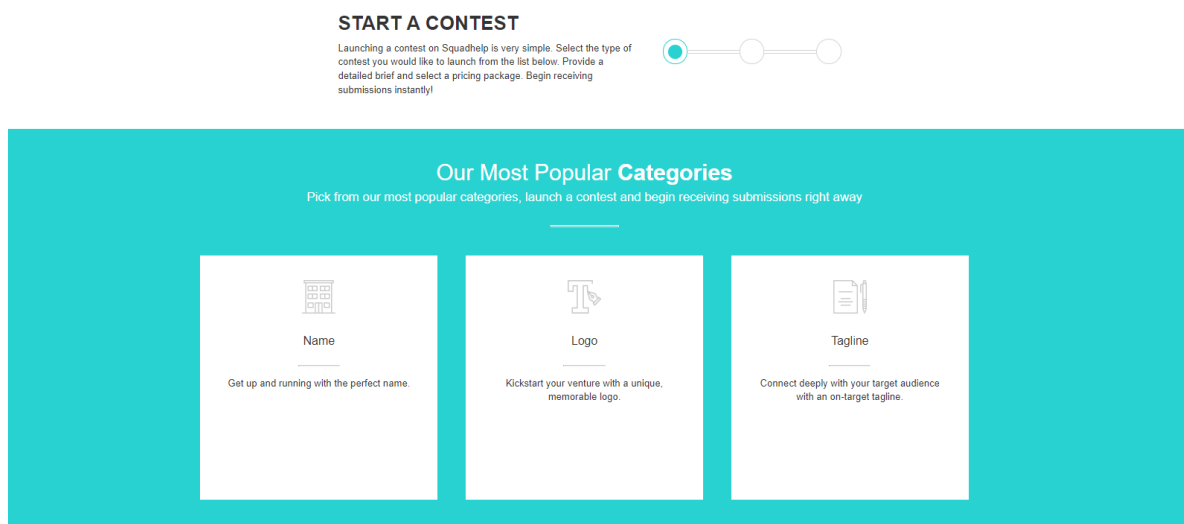


Рисунок 4.21 – Вигляд сторінки для створення конкурсу

4.4 Тестування програмного забезпечення

Було здійснено тестування програмного забезпечення, розробленого для створення віддалених завдань. Система пройшла модульне та інтеграційне тестування із застосуванням стандартних засобів фреймворку React і

бібліотеки React Testing Library. Перевірці підлягали програмні модулі, що реалізують ключову функціональність додатку [20].

Результати тестування підтвердили, що програмне забезпечення працює стабільно та без збоїв. Воно повністю відповідає всім функціональним вимогам і успішно виконує передбачені завдання. Всі виявлені незначні недоліки були оперативно усунені в процесі доопрацювання коду.

Окрема увага приділялася тестуванню сценаріїв взаємодії користувача з інтерфейсом – таких як створення, редагування, перегляд і видалення завдань. Було перевірено правильність обробки введених даних, валідацію форм, а також реакцію системи на типові помилки користувача [20].

Таким чином, розроблене ПЗ демонструє високу надійність, функціональність і відповідність сучасним вимогам до якості вебзастосунків.

4.5 Висновки за розділом 4

В четвертому розділі було розглянуто призначення, умови використання та функціональність програмного забезпечення для створення віддалених завдань. Описано процес та умови виконання програмного забезпечення.

Також було здійснено запуск і тестування розробленого програмного забезпечення. Проведено модульне та інтеграційне тестування із використанням бібліотеки React Testing Library. Усі ключові компоненти додатку протестовані на коректність виконання функцій. В результаті перевірки встановлено, що програма стабільно виконує всі передбачені задачі, відповідає функціональним вимогам і готова до практичного використання.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку для створення віддалених завдань. Для цього було виконано ряд завдань:

- проаналізована переметна область, яка стосується виконання віддалених завдань;
- визначено поняття та принципи виконання віддалених завдань;
- сформульовані основні відмінності виконання віддалених завдань від повної зайнятості;
- проаналізовані категорії виконавців віддалених завдань;
- виконано аналіз існуючих платформ для виконання віддалених завдань;
- досліджено, обрано та аргументовано вибір мови програмування, фреймворків для розробки FrontEnd та Backend частин вебсайту;
- розроблено структуру програмного забезпечення для створення віддалених завдань;
- спроектована структура бази даних додатку;
- розроблено основний алгоритм роботи програмного забезпечення для створення віддалених завдань.;
- виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням;
- розроблено програмне забезпечення для створення віддалених завдань.
- розглянуто призначення, умови використання та функціональність програмного забезпечення для створення віддалених завдань;
- описано процес та умови виконання програмного забезпечення;
- здійснено запуск і тестування розробленого програмного забезпечення.

Сайт розроблений в середовищі IntelliJIdea на мові JavaScript (React, Node), HTML, CSS.

Застосунок було протестований за допомогою модульного та інтеграційного тестування із використанням бібліотеки React Testing Library. Усі ключові компоненти додатку протестовані на коректність виконання функцій. В результаті перевірки встановлено, що програма стабільно виконує всі передбачені задачі, відповідає функціональним вимогам і готова до практичного використання.

Застосунок реалізує такі функціональні вимоги:

- відображення інформації про пропозиції та замовлення;
- можливість авторизації на сайті;
- пошук за різними категоріями;
- модерація пропозицій;
- чат між виконавцем та замовником;
- створення конкурсів.

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Navigating Remote Development Challenges in Ukraine - Top Tools and Recommendations [Electronic resource] – Access mode: <https://moldstud.com/articles/p-navigating-remote-development-challenges-in-ukraine-top-tools-and-recommendations>.
2. Remote Work for Developers: Essential Tools and Platforms [Electronic resource] – Access mode: <https://daily.dev/blog/remote-work-for-developers-essential-tools-and-platforms>.
3. 20 Best Task Management Software Tools to Elevate Your Workflows [2025 Edition] [Electronic resource] – Access mode: <https://clickup.com/blog/task-management-software>.
4. The Art of Successful Freelancing: From Cubicle to Freedom [Electronic resource] – Access mode: <https://www.ie.edu/insights/articles/from-cubicle-to-freedom-the-art-of-successful-freelancing>.
5. What Is Freelancing? Basics and Popular Jobs in 2025 [Electronic resource] – Access mode: <https://www.upwork.com/resources/what-is-freelancing>.
6. What You Need to Know About Freelancing [Electronic resource] – Access mode: <https://www.illinoisworknet.com/Pages/Article.aspx?ArticleId=291>.
7. What Is a Freelancer? Definition, Benefits, and Types [Electronic resource] – Access mode: <https://ca.indeed.com/career-advice/finding-a-job/what-is-freelancer>.
8. 25 best freelance websites to find work in 2025 [Electronic resource] – Access mode: <https://www.hostinger.com/tutorials/best-freelance-websites>.
9. The ultimate guide to the 29 best freelance websites for 2025 [Electronic resource] – Access mode: <https://arc.dev/talent-blog/best-freelance-websites>.
10. Advantages and Disadvantages of Javascript [Electronic resource] – Access mode: <https://ellow.io/advantages-and-disadvantages-of-javascript/#respond>.

11. Usage statistics of JavaScript as client-side programming language on websites [Electronic resource] – Access mode: <https://w3techs.com/technologies/details/cp-javascript>.

12. 27 Best JavaScript Frameworks For 2025 [Electronic resource] – Access mode: <https://www.lambdatest.com/blog/best-javascript-frameworks>.

13. 6 Best JavaScript Frameworks for 2025 [Electronic resource] – Access mode: <https://strapi.io/blog/best-javascript-frameworks>.

14. JavaScript frameworks and libraries [Electronic resource] – Access mode: https://developer.mozilla.org/enUS/docs/Learn_web_development/Core/Frameworks_libraries.

15. Top 15 JavaScript frameworks for 2025 [Electronic resource] – Access mode: <https://hygraph.com/blog/javascript-frameworks>.

16. What is PostgreSQL? [Electronic resource] – Access mode: <https://www.ibm.com/think/topics/postgresql>.

17. What is a relational database? [Electronic resource] – Access mode: <https://cloud.google.com/learn/what-is-a-relational-database>.

18. MVC Architecture – What is a Model View Controller Framework? [Electronic resource] – Access mode: <https://www.freecodecamp.org/news/mvc-architecture-what-is-a-model-view-controller-framework>.

19. Принципи вебдизайну інтерфейсу користувача [Електрон. ресурс]. – Режим доступу: <https://stfalcon.com/uk/blog/post/user-interface-web-design-principles>.

20. Тестування вебпроектів: основні етапи та поради [Електрон. ресурс]. – Режим доступу: <https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi>.

ДОДАТОК А
Технічне завдання

Вступ

Даний застосунок присвячений для створення віддалених завдань.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для створення віддалених завдань», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Основні вимоги до програми, що розробляється

Розробка ПЗ для створення віддалених завдань є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для створення віддалених завдань.

A.2.1 Вимоги до функціональних характеристик

- відображення інформації про пропозиції та замовлення;
- можливість авторизації на сайті;
- пошук за різними категоріями;
- модерація пропозицій;
- чат між виконавцем та замовником;
- створення конкурсів.

A.2.2 Вимоги до інтерфейсу програми

Інтерфейс системи повинен відрізнятися лаконічністю, інтуїтивною зрозумілістю та зручністю для користувачів. Дизайн має бути адаптивним для зручності роботи на будь-якому дисплеї.

A.2.3 Вимоги до надійності

Забезпечення конфіденційності та безпеки даних забезпечується дотримання відповідних правил та рекомендацій щодо захисту даних користувачів та захисту інформації користувача.

A.2.4 Вимоги до складу та параметрів технічний засобів

Для функціонування застосунку, а також в разі його доопрацювання, необхідно мати таку мінімальну апаратну платформу: ПК з тактовою частотою мікропроцесора не менше 2 ГГц, з оперативною пам'яттю з мінімальним об'ємом 4 Гб, з дисковим простором 5 Гб, необхідним для інсталяції браузера. На персональному комп'ютері повинно бути встановлено наступне програмне забезпечення: операційна система, браузер.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Розробка ПЗ для створення віддалених завдань є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для створення віддалених завдань.

Б.2 Функціональне призначення

Система має реалізовувати наступні функціональні вимоги:

- відображення інформації про пропозиції та замовлення;
- можливість авторизації на сайті;
- пошук за різними категоріями;
- модерація пропозицій;
- чат між виконавцем та замовником;
- створення конкурсів.

Б.3 Опис логічної структури

Загальна структурна схема системи за обраною темою представлена на рисунку Б.1.

Ця структурна схема демонструє архітектуру вебдодатку на основі шаблону MVC (Model-View-Controller).

Загальний потік запиту:

- браузер надсилає запит на сервер;
- сервер отримує запит і передає його до URL диспетчера (роутера);
- URL визначає, яке представлення (View/Controller) має обробити

запит, і передає його до Представлення.

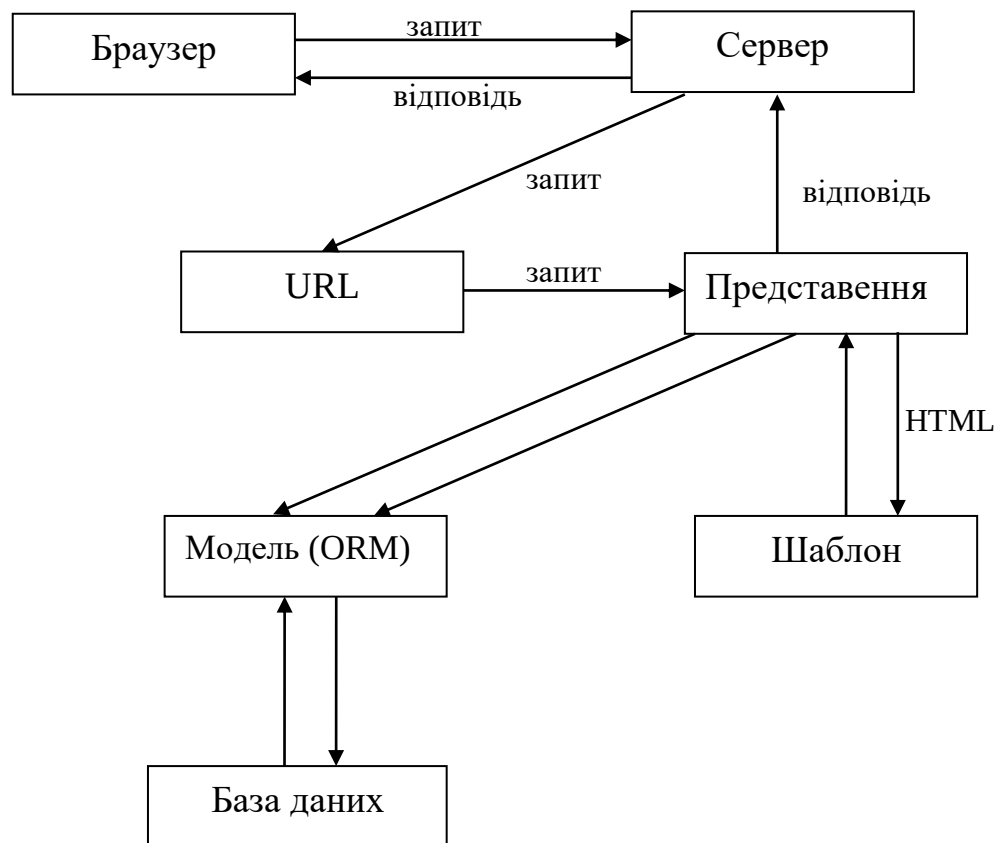


Рисунок Б.1 – Загальна структурна схема системи за обраною темою

Обробка логіки:

- представлення звертається до Моделі (ORM), щоб отримати або змінити дані в Базі даних;
- представлення звертається може використати Шаблон, щоб згенерувати HTML на основі отриманих даних.

Відповідь:

- HTML, згенерований у Шаблоні, повертається через Представлення назад до Сервера, а потім – у Браузер як відповідь.

Компоненти:

- браузер – клієнтський інтерфейс користувача;
- сервер – обробляє запити та надсилає відповіді;
- URL – маршрутизація запитів;
- представлення – контролер, логіка обробки запитів;

- модель (ORM) – взаємодія з базою даних через об'єкти;
- база даних – зберігання даних;
- шаблон – генерація HTML-вмісту для користувача.

Система складається з наступних модулів, призначення кожного з модулів представлено у таблиці Б.1.

Таблиця Б.1 – Опис модулів

Назва директорії	Опис модулю
controllers	Модуль який відповідає за основний функціонал серверу.
models	Модуль призначений для опису моделей бази даних.
components	Тут знаходиться основна візуальна частина сайту
sagas	Модуль який відповідає за запити до сервера
public	Модуль який зберігає усі файли, які будуть додані до сайту

Б.4 Використані технічні засоби

Для функціонування застосунку, а також в разі його доопрацювання, необхідно мати таку мінімальну апаратну платформу: ПК з тактовою частотою мікропроцесора не менше 2 ГГц, з оперативною пам'яттю з мінімальним об'ємом 4 Гб, з дисковим простором 5 Гб, необхідним для інсталяції браузера. На персональному комп'ютері повинно бути встановлено наступне програмне забезпечення: операційна система, браузер.

ДОДАТОК В
Текст програми

```

contestController.js
const nodemailer = require('nodemailer');
const db = require('../models/index');
const ServerError = require('../errors/ServerError');
const contestQueries = require('../queries/contestQueries');
const userQueries = require('../queries/userQueries');
const controller = require('../socketInit');
const UtilFunctions = require('../utils/functions');
const CONSTANTS = require('../constants');

module.exports.dataForContest = async (req, res, next) => {
  try {
    const payload = {};
    const {
      body: { characteristic1, characteristic2 },
    } = req;
    const predicate = [characteristic1, characteristic2, 'industry'].filter(
      Boolean,
    );

    const characteristics = await db.Select.findAll({
      where: {
        type: {
          [db.Sequelize.Op.or]: predicate,
        },
      },
    });
    if (!characteristics) {
      return next(new ServerError());
    }
    characteristics.forEach((characteristic) => {
      if (!payload[characteristic.type]) {
        payload[characteristic.type] = [];
      }
      payload[characteristic.type].push(characteristic.describe);
    });
    res.send(payload);
  } catch (err) {
    next(new ServerError('cannot get contest preferences'));
  }
};

module.exports.getContestById = async (req, res, next) => {
  try {
    let contestInfo = await db.Contest.findOne({
      where: { id: req.headers.contestid },
      order: [[db.Offer, 'id', 'asc']],
      include: [
        {
          model: db.User,
          required: true,
          attributes: {

```

```

        exclude: ['password', 'role', 'balance'],
      },
    },
    {
      model: db.Offer,
      required: false,
      where:
        req.tokenData.role === CONSTANTS.ROLES.CREATOR
        ? { userId: req.tokenData.userId }
        : {},
      attributes: { exclude: ['userId', 'contestId'] },
      include: [
        {
          model: db.User,
          required: true,
          attributes: {
            exclude: ['password', 'role', 'balance'],
          },
        },
        {
          model: db.Rating,
          required: false,
          where: { userId: req.tokenData.userId },
          attributes: { exclude: ['userId', 'offerId'] },
        },
      ],
    },
  ],
});
contestInfo = contestInfo.get({ plain: true });
contestInfo.Offers.forEach((offer) => {
  if (offer.Rating) {
    offer.mark = offer.Rating.mark;
  }
  delete offer.Rating;
});
res.send(contestInfo);
} catch (err) {
  next(new ServerError());
}
};

module.exports.downloadFile = async (req, res, next) => {
  const file = CONSTANTS.CONTESTS_DEFAULT_DIR + req.params.fileName;
  res.download(file);
};

module.exports.updateContest = async (req, res, next) => {
  if (req.file) {
    req.body.fileName = req.file.filename;
    req.body.originalFileName = req.file.originalname;
  }
};

```

```

const { contestId } = req.body;
delete req.body.contestId;
try {
  const updatedContest = await contestQueries.updateContest(req.body, {
    id: contestId,
    userId: req.tokenData.userId,
  });
  res.send(updatedContest);
} catch (e) {
  next(e);
}
};

module.exports.setNewOffer = async (req, res, next) => {
  const obj = {};
  if (req.body.contestType === CONSTANTS.CONTEST_TYPES.LOGO) {
    obj.fileName = req.file.filename;
    obj.originalFileName = req.file.originalname;
  } else {
    obj.text = req.body.offerData;
  }
  obj.userId = req.tokenData.userId;
  obj.contestId = req.body.contestId;
  try {
    const result = await contestQueries.createOffer(obj);
    delete result.contestId;
    delete result.userId;
    const User = Object.assign({}, req.tokenData, { id: req.tokenData.userId });
    res.send(Object.assign({}, result, { User }));
  } catch (e) {
    return next(new ServerError());
  }
};

const rejectOffer = async (offerId, creatorId, contestId) => {
  const rejectedOffer = await contestQueries.updateOffer(
    { status: CONSTANTS.OFFER_STATUSES.REJECTED },
    { id: offerId },
  );
  controller
    .getNotificationController()
    .emitChangeOfferStatus(
      creatorId,
      'Someone of yours offers was rejected',
      contestId,
    );
  return rejectedOffer;
};

const resolveOffer = async (
  contestId,
  creatorId,

```

```

    orderId,
    offerId,
    priority,
    transaction,
  ) => {
    const finishedContest = await contestQueries.updateContestStatus(
      {
        status: db.sequelize.literal(` CASE
          WHEN "id"=${contestId} AND "orderId"=${orderId}' THEN '${
            CONSTANTS.CONTESTS_STATUSES.FINISHED
          }':"enum_Contests_status"
          WHEN "orderId"=${orderId}' AND "priority"=${priority + 1} THEN '${
            CONSTANTS.CONTESTS_STATUSES.ACTIVE
          }':"enum_Contests_status"
          ELSE '${
            CONSTANTS.CONTESTS_STATUSES.PENDING
          }':"enum_Contests_status"
        END
        `),
      },
      { orderId },
      transaction,
    );
    await userQueries.updateUser(
      { balance: db.sequelize.literal(`balance + ${finishedContest.prize}`) },
      creatorId,
      transaction,
    );
    const updatedOffers = await contestQueries.updateOfferStatus(
      {
        status: db.sequelize.literal(` CASE
          WHEN "id"=${offerId} THEN
            '${CONSTANTS.OFFER_STATUSES.WON}':"enum_Offers_status"
          WHEN "status"=${CONSTANTS.OFFER_STATUSES.PENDING}' THEN
            '${CONSTANTS.OFFER_STATUSES.REJECTED}':"enum_Offers_status"
          ELSE '${CONSTANTS.OFFER_STATUSES.BANNED}':"enum_Offers_status"
        END
        `),
      },
      {
        contestId,
      },
      transaction,
    );
    transaction.commit();
    const arrayRoomsId = [];
    updatedOffers.forEach((offer) => {
      if (
        offer.status === CONSTANTS.OFFER_STATUS_REJECTED
        && creatorId !== offer.userId
      ) {
        arrayRoomsId.push(offer.userId);
      }
    });
  }
}

```

```

    }
  });
  controller
    .getNotificationController()
    .emitChangeOfferStatus(
      arrayRoomsId,
      'Someone of yours offers was rejected',
      contestId,
    );
  controller
    .getNotificationController()
    .emitChangeOfferStatus(creatorId, 'Someone of your offers WIN', contestId);
  return updatedOffers[0].dataValues;
};

module.exports.banOrPandingOffer = async (req, res, next) => {
  const {
    body: {
      newStatus, offerId, reasonOfBan, customerId, email, text,
    },
  } = req;
  console.log('email', email);
  try {
    const offer = await db.Offer.update(
      { status: newStatus, reasonOfBan },
      {
        where: { id: offerId },
      },
    );
  };
  if (newStatus === CONSTANTS.OFFER_STATUSES.PENDING) {
    controller
      .getNotificationController()
      .emitEntryCreated(customerId);
  }

  const transporter = nodemailer.createTransport({
    service: 'Gmail',
    auth: {
      user: 'testsender615',
      pass: 'testsender87o(p)',
    },
  });

  await transporter.sendMail(
    {
      from: '"EXAMsqyadhhelp.com"',
      to: email,
      subject: 'One of your offer have new status!',
      text: `Your offer "${text}" was ${newStatus} ===
CONSTANTS.OFFER_STATUSES.BANNED ? `banned` by moderator. Reason:
${reasonOfBan}` : 'resolve by moderator'`.`,
    },
  );

```

```

      (error) => {
        if (error) {
          console.log(error);
        }
      },
    );

    console.log('email was send!');

    res.send(offer);
  } catch (err) {
    next(err);
  }
};

module.exports.setOfferStatus = async (req, res, next) => {
  let transaction;
  if (req.body.command === 'reject') {
    try {
      const offer = await rejectOffer(
        req.body.offerId,
        req.body.creatorId,
        req.body.contestId,
      );
      res.send(offer);
    } catch (err) {
      next(err);
    }
  } else if (req.body.command === 'resolve') {
    try {
      transaction = await db.sequelize.transaction();
      const winningOffer = await resolveOffer(
        req.body.contestId,
        req.body.creatorId,
        req.body.orderId,
        req.body.offerId,
        req.body.priority,
        transaction,
      );
      res.send(winningOffer);
    } catch (err) {
      transaction.rollback();
      next(err);
    }
  }
};

module.exports.getCustomersContests = (req, res, next) => {
  db.Contest.findAll({
    where: { status: req.headers.status, userId: req.tokenData.userId },
    limit: req.body.limit,
    offset: req.body.offset ? req.body.offset : 0,
  });
};

```



```

    );
    res.send({
      contests,
      haveMore: !contests.length === 0,
    });
  })
  .catch((err) => {
    next(new ServerError());
  });
};

module.exports.getModeratorOffers = async (req, res, next) => {
  try {
    console.log('getting offers>>>>>>>>>');
    const { body: pagination } = req;

    const offers = await db.Offer.findAll({
      where: { status: CONSTANTS.OFFER_STATUSES.MODERATED },
      attributes: { exclude: ['userId', 'contestId'] },
      order: [['id', 'asc']],
      ...pagination,
      include: [
        {
          model: db.User,
          required: true,
          attributes: {
            exclude: ['password', 'role', 'balance'],
          },
        },
        {
          model: db.Contest,
          required: true,
          attributes: ['userId'],
        },
      ],
    });
    res.send(offers);
  } catch (err) {
    next(new ServerError());
  }
};

basicMiddlewares.js
const bd = require('../models/index');
const RightsError = require('../errors/RightsError');
const ServerError = require('../errors/ServerError');
const CONSTANTS = require('../constants');

module.exports.parseBody = (req, res, next) => {
  req.body.contests = JSON.parse(req.body.contests);
  for (let i = 0; i < req.body.contests.length; i++) {
    if (req.body.contests[i].haveFile) {
      const file = req.files.splice(0, 1);

```

```

    req.body.contests[i].fileName = file[0].filename;
    req.body.contests[i].originalFileName = file[0].originalname;
  }
}
next();
};

```

```

module.exports.canGetContest = async (req, res, next) => {
  let result = null;
  try {
    if (req.tokenData.role === CONSTANTS.ROLES.CUSTOMER) {
      result = await bd.Contest.findOne({
        where: { id: req.headers.contestid, userId: req.tokenData.userId },
      });
    } else if (req.tokenData.role === CONSTANTS.ROLES.CREATOR) {
      result = await bd.Contest.findOne({
        where: {
          id: req.headers.contestid,
          status: {
            [bd.Sequelize.Op.or]: [
              CONSTANTS.CONTESTS_STATUSES.ACTIVE,
              CONSTANTS.CONTESTS_STATUSES.FINISHED,
            ],
          },
        },
      });
    }
    result ? next() : next(new RightsError());
  } catch (err) {
    next(new ServerError(err));
  }
};

```

```

module.exports.onlyForCreative = (req, res, next) => {
  if (req.tokenData.role === CONSTANTS.ROLES.CUSTOMER) {
    next(new RightsError());
  } else {
    next();
  }
};

```

```

module.exports.onlyForCustomer = (req, res, next) => {
  if (req.tokenData.role === CONSTANTS.ROLES.CREATOR) {
    return next(new RightsError('this page only for customers'));
  }
  next();
};

```

```

module.exports.canSendOffer = async (req, res, next) => {
  if (req.tokenData.role === CONSTANTS.ROLES.CUSTOMER) {
    return next(new RightsError());
  }
}

```

```

    try {
      const contest = await bd.Contest.findOne({
        where: {
          id: req.body.contestId,
        },
        attributes: ['status'],
      });
      if (contest.get('status') ===
CONSTANTS.CONTESTS_STATUSES.ACTIVE) {
        next();
      } else {
        return next(new RightsError());
      }
    } catch (e) {
      next(new ServerError());
    }
  };

module.exports.onlyForCustomerWhoCreateContest = async (req, res, next) => {
  try {
    const result = await bd.Contest.findOne({
      where: {
        userId: req.tokenData.userId,
        id: req.body.contestId,
        status: CONSTANTS.CONTESTS_STATUSES.ACTIVE,
      },
    });
    if (!result) {
      return next(new RightsError());
    }
    next();
  } catch (e) {
    next(new ServerError());
  }
};

module.exports.canUpdateContest = async (req, res, next) => {
  try {
    const result = bd.Contest.findOne({
      where: {
        userId: req.tokenData.userId,
        id: req.body.contestId,
        status: {
CONSTANTS.CONTESTS_STATUSES.FINISHED },
      },
    });
    if (!result) {
      return next(new RightsError());
    }
    next();
  } catch (e) {
    next(new ServerError());
  }
};

```

[bd.Sequelize.Op.not]:

```
}
```

```
UserController.js
```

```
const moment = require('moment');
const uuid = require('uuid/v1');
const CONSTANTS = require('../constants');
const db = require('../models/index');
const controller = require('../socketInit');
const userQueries = require('../queries/userQueries');
const bankQueries = require('../queries/bankQueries');
const ratingQueries = require('../queries/ratingQueries');

function getQuery(offerId, userId, mark, isFirst, transaction) {
  const getCreateQuery = () => ratingQueries.createRating({
    offerId,
    mark,
    userId,
  }, transaction);
  const getUpdateQuery = () => ratingQueries.updateRating({ mark },
    { offerId, userId }, transaction);
  return isFirst ? getCreateQuery : getUpdateQuery;
}

module.exports.changeMark = async (req, res, next) => {
  let sum = 0;
  let avg = 0;
  let transaction;
  const {
    isFirst, offerId, mark, creatorId,
  } = req.body;
  const { userId } = req.tokenData;
  try {
    transaction = await db.sequelize.transaction(
      {
        isolationLevel:
db.Sequelize.Transaction.ISOLATION_LEVELS.READ_UNCOMMITTED },
    );
    const query = getQuery(offerId, userId, mark, isFirst, transaction);
    await query();
    const offersArray = await db.Rating.findAll({
      include: [
        {
          model: db.Offer,
          required: true,
          where: { userId: creatorId },
        },
      ],
      transaction,
    });
    for (let i = 0; i < offersArray.length; i++) {
      sum += offersArray[i].dataValues.mark;
    }
    avg = sum / offersArray.length;
```

```

    await userQueries.updateUser({ rating: avg }, creatorId, transaction);
    transaction.commit();
    controller.getNotificationController().emitChangeMark(creatorId);
    res.send({ userId: creatorId, rating: avg });
  } catch (err) {
    transaction.rollback();
    next(err);
  }
};

module.exports.payment = async (req, res, next) => {
  const transaction = await db.sequelize.transaction();
  try {
    await bankQueries.updateBankBalance({
      balance: db.sequelize.literal(`
        CASE
          WHEN "cardNumber"='${req.body.number.replace(/ /g,
            "')}' AND "cvc"='${req.body.cvc}' AND "expiry"='${req.body.expiry}'
          THEN "balance"-${req.body.price}
          WHEN
            "cardNumber"='${CONSTANTS.SQUADHELP_BANK_NUMBER}'          AND
            "cvc"='${CONSTANTS.SQUADHELP_BANK_CVC}'                  AND
            "expiry"='${CONSTANTS.SQUADHELP_BANK_EXPIRY}'
          THEN "balance" + ${req.body.price} END
        `),
    },
    {
      cardNumber: {
        [db.Sequelize.Op.in]: [
          CONSTANTS.SQUADHELP_BANK_NUMBER,
          req.body.number.replace(/ /g, ""),
        ],
      },
    },
    transaction);
    const orderId = uuid();
    req.body.contests.forEach((contest, index) => {
      const prize = index === req.body.contests.length - 1 ? Math.ceil(
        req.body.price / req.body.contests.length,
      )
        : Math.floor(req.body.price / req.body.contests.length);
      contest = Object.assign(contest, {
        status: index === 0 ? 'active' : 'pending',
        userId: req.tokenData.userId,
        priority: index + 1,
        orderId,
        createdAt: moment().format('YYYY-MM-DD HH:mm'),
        prize,
      });
    });
    await db.Contest.bulkCreate(req.body.contests, transaction);
  }
};

```

```

    transaction.commit();
    res.send();
  } catch (err) {
    transaction.rollback();
    next(err);
  }
};

```

```

module.exports.updateUser = async (req, res, next) => {
  try {
    if (req.file) {
      req.body.avatar = req.file.filename;
    }
    const updatedUser = await userQueries.updateUser(req.body,
      req.tokenData.userId);
    res.send({
      firstName: updatedUser.firstName,
      lastName: updatedUser.lastName,
      displayName: updatedUser.displayName,
      avatar: updatedUser.avatar,
      email: updatedUser.email,
      balance: updatedUser.balance,
      role: updatedUser.role,
      id: updatedUser.id,
    });
  } catch (err) {
    next(err);
  }
};

```

```

module.exports.cashout = async (req, res, next) => {
  let transaction;
  try {
    transaction = await db.sequelize.transaction();
    const updatedUser = await userQueries.updateUser(
      { balance: db.sequelize.literal(`balance - ${req.body.sum}`) },
      req.tokenData.userId, transaction,
    );
    await bankQueries.updateBankBalance({
      balance: db.sequelize.literal(`CASE
        WHEN "cardNumber"='${req.body.number.replace(/ /g,
          " ")}' AND "expiry"='${req.body.expiry}' AND "cvc"='${req.body.cvc}'
        THEN "balance" + ${req.body.sum}
        WHEN
"cardNumber"='${CONSTANTS.SQUADHELP_BANK_NUMBER}'          AND
"expiry"='${CONSTANTS.SQUADHELP_BANK_EXPIRY}'              AND
"cvc"='${CONSTANTS.SQUADHELP_BANK_CVC}'
        THEN "balance" - ${req.body.sum}
        END
      `),
    },
    {

```

```
cardNumber: {
  [db.Sequelize.Op.in]: [
    CONSTANTS.SQUADHELP_BANK_NUMBER,
    req.body.number.replace(/ /g, ""),
  ],
},
},
transaction);
transaction.commit();
res.send({ balance: updatedUser.balance });
} catch (err) {
  transaction.rollback();
  next(err);
}
};
};
```

ДОДАТОК Д
Слайди презентації

**Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

Програмне забезпечення для створення віддалених завдань

Виконав: Богдан САВЕЛЬЄВ

ст. групи КНТ-111

Керівник: Олена ШИТІКОВА

к.т.н., доцент

Рисунок Д.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- ✳ Об'єкт дослідження – процес програмної реалізації застосунку для створення віддалених завдань.
- ✳ Предмет дослідження – програмні засоби для реалізації застосунку для створення віддалених завдань.
- ✳ Метою роботи є програмна реалізація застосунку для створення віддалених завдань.

Рисунок Д.2 – Слайд 2

ПОСТАНОВКА ЗАВДАННЯ

Розробка ПЗ для створення віддалених завдань є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для створення віддалених завдань.

4

Рисунок Д.3 – Слайд 3

ПОРІВНЯННЯ РОБОТИ НА ФРІЛАНСІ ТА ПОВНОЇ ЗАЙНЯТОСТІ

	Фріланс	Повна зайнятість
Гнучкість	Свобода працювати будь-де в будь-який час, залежно від уподобань чи обставин. І можна вибрати своїх клієнтів і виконувати певні обов'язки протягом встановленого робочого часу та мати певні обов'язки.	Робота на повний робочий день означає, що робітник виконує певні обов'язки протягом встановленого робочого часу та має певні обов'язки.
Стабільність	На відміну від повної зайнятості, фрілансерська робота може бути дещо непередбачуваною, що ускладнює планування майбутніх фінансових питань.	Працівники, які працюють повний робочий день, зазвичай мають стабільний, передбачуваний дохід і графік роботи, що полегшує планування майбутніх фінансових питань.
Переваги	Незалежні працівники зазвичай не мають доступу до фінансованих роботодавцем пільг, таких як оплачувана відпустка, пенсійні плани, медичне страхування та лікарняні.	Такі переваги, як оплачувана відпустка, медичне страхування та пенсійні плани, є значним плюсом повної зайнятості.
Мотивація	Фрілансер працює як сам собі бос, тому самотивація є необхідною.	Як штатний працівник повинен відповідати певним стандартам продуктивності, щоб отримати підвищення по службі.
Заробіток	Фрілансери мають більший контроль над своїм потенціалом заробітку, оскільки вони самі встановлюють тарифи та години роботи.	Заробітки штатного працівника зазвичай визначаються структурою заробітної плати та політикою роботодавця.
Набір навичок	Незалежні підрядники мають можливість спеціалізуватися в конкретній галузі, або отримати досвід і навички в різних предметних областях.	На відміну від фрілансерів, штатні працівники можуть не мати такої ж рівня гнучкості для розвитку конкретних навичок, оскільки їхні посадові ролі та обов'язки в основному визначаються роботодавцем.
Досвід	Фрілансери мають можливість працювати з кількома брендами по всьому світу, щоб розширити свою професійну мережу та створити різноманітне портфоліо.	На відміну від фрілансерів, працівники, як правило, обмежені в одній компанії, що може обмежити їх вплив на різні галузі чи клієнтів.

Рисунок Д.4 – Слайд 4

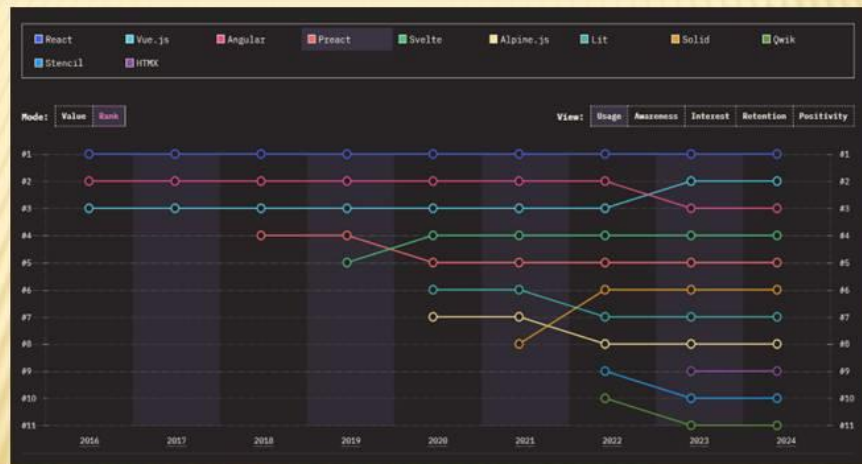
ВИБІР МОВИ ПРОГРАМУВАННЯ

Переваги JavaScript	Недоліки JavaScript
Забезпечує динамічний та інтерактивний веб-вміст.	Вразливий до загроз безпеці, таких як міжсайтовий сценарій (XSS).
Покращує взаємодію з користувачем завдяки обробці на стороні клієнта.	Обмеження продуктивності через однопотокове виконання.
Широка підтримка бібліотек через фреймворки, такі як React і Angular.	Невідповідності в різних браузерах можуть вимагати додаткового тестування.
Повна інтеграція з HTML і CSS для веб-розробки.	Залежність від обробки на стороні клієнта може перешкоджати доступності.
Сприяє створенню адаптивних, зручних для мобільних пристроїв інтерфейсів.	Налагодження та усунення несправностей коду JavaScript може бути складним завданням.

5

Рисунок Д.5 – Слайд 5

ВИБІР ФРЕЙМВОРКУ ДЛЯ РОЗРОБКИ FRONTEND ЧАСТИНИ ВЕБ-САЙТУ



6

Рисунок Д.6 – Слайд 6

ЗАГАЛЬНА СТРУКТУРНА СХЕМА СИСТЕМИ

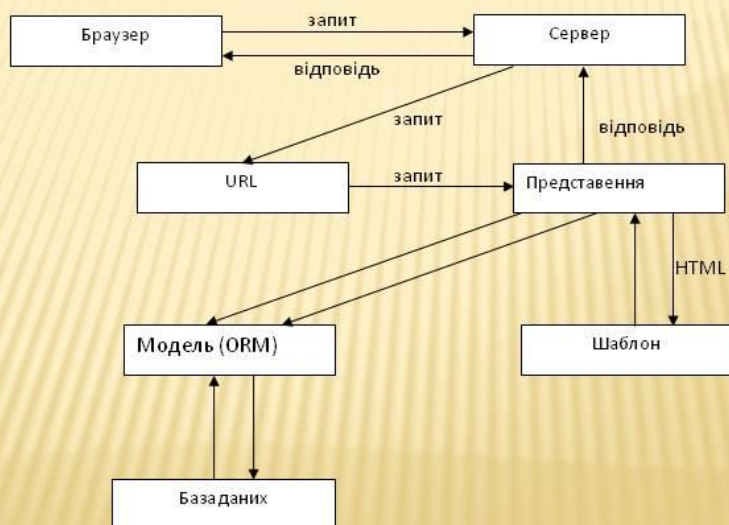


Рисунок Д.7 – Слайд 7

ОБ'ЄКТНО-ОРІЄНТОВАНА МОДЕЛЬ БАЗИ ДАНИХ

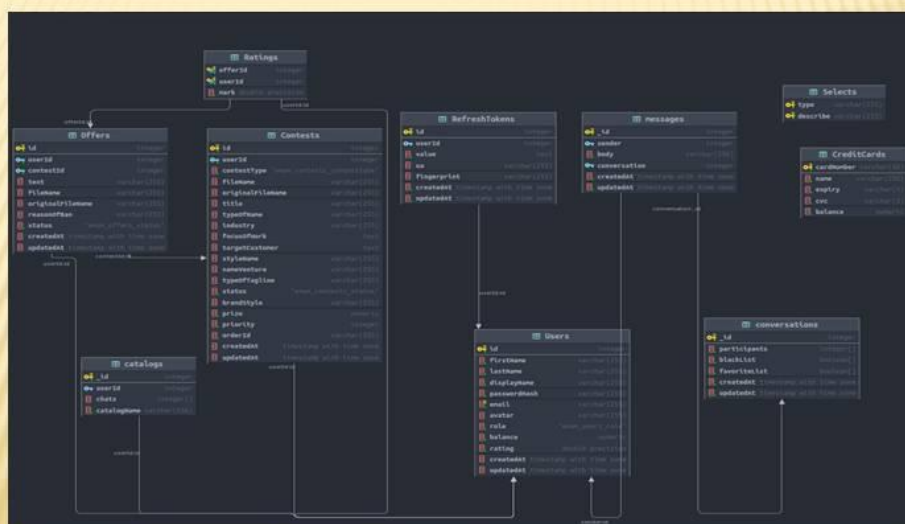
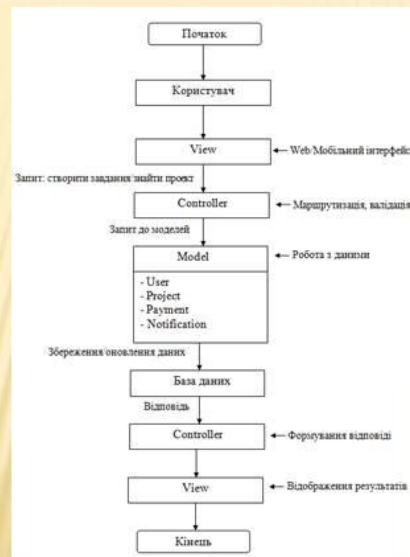


Рисунок Д.8 – Слайд 8

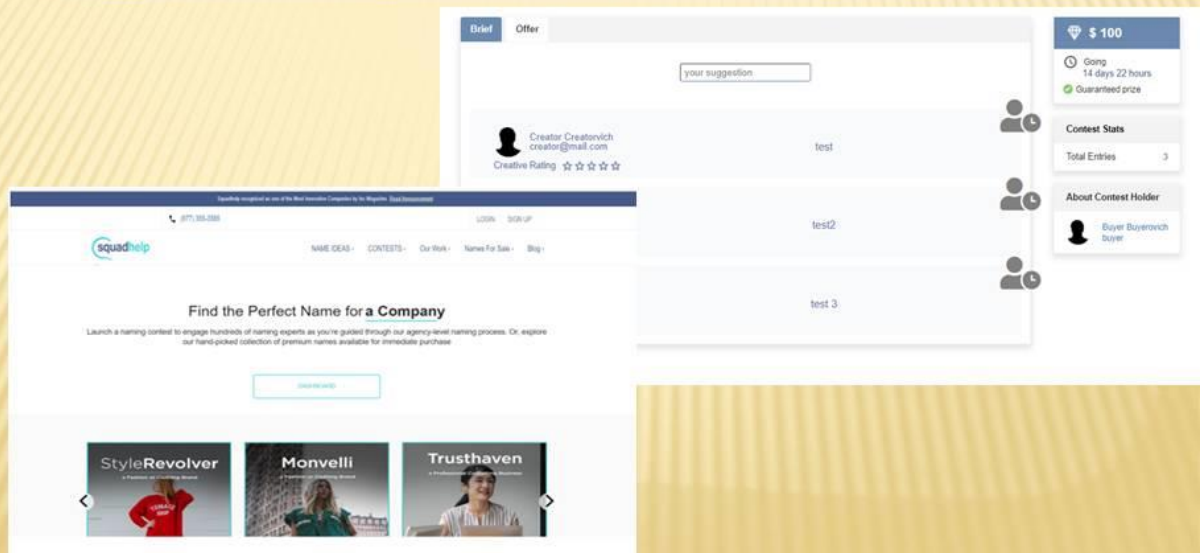
ОСНОВНИЙ АЛГОРИТМ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



10

Рисунок Д.9 – Слайд 9

ІНТЕРФЕЙС ЗАСТОСУНКУ



11

Рисунок Д.10 – Слайд 10

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку для створення віддалених завдань. Для цього було виконано ряд завдань:

- + проаналізована периметрна область, яка стосується виконання віддалених завдань;
- + визначено поняття та принципи виконання віддалених завдань;
- + сформульовані основні відмінності виконання віддалених завдань від повної зайнятості;
- + проаналізовані категорії виконавців віддалених завдань;
- + виконано аналіз існуючих платформ для виконання віддалених завдань;
- + досліджено, обрано та аргументовано вибір мови програмування, фреймворків для розробки FrontEnd та Backend частин веб-сайту;
- + розроблено структуру, основний алгоритм роботи та базових програмного забезпечення для створення віддалених завдань;
- + виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням;
- + розроблено програмне забезпечення для створення віддалених завдань.
- + описано процес та умови виконання програмного забезпечення;
- + здійснено запуск і тестування розробленого програмного забезпечення.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена система є зручною для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

12

Рисунок Д.11 – Слайд 11