Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2023 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Комп'ютерна система планування ІТ-проєктів»

Захищено на засіданні
Атестаційної комісії № 40
протокол № __ від __.06.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ Скоб Ю. О.
(підпис) (прізвище та ініціали)

Виконала:
студентка 4 курсу, групи КІ– 41
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – «Комп'ютерна інженерія»
Кринична Аліна Олександрівна
(підпис)

Керівник:
канд. техн.наук, доцент
Булавін Дмитро Олексійович
(підпис)

Рецензент:
д.т.н., доц., професор кафедри
теоретичної та прикладної інформатики
Руккас Кирило Маркович _ _

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 87 сторінок, із яких 53 сторінки основної частини з 25 рисунком, 3 таблицями, 16 найменуваннями списку використаних джерел та чотирма додатками.

Метою кваліфікаційної роботи є розробка сервісу планування та планування складу команди IT-проєктів, що допоможе вдосконалити та спростити роботу, а також підібрати оптимальний склад команди.

Об'єкт дослідження – IT-проєкти різної складності, що потребують ефективного планування та управління для досягнення успіху за рахунок підбору команди спеціалістів. У роботі зроблено акцент на розгляд різноманітних проєктів, але тема також досліджує, які спеціалісти можуть бути залучені до виконання проєкту.

Предмет дослідження – комп'ютерні системи планування, які застосовуються для керування IT-проєктами. Основний акцент зроблено на вивченні принципів та методологій, які використовуються в таких системах, а також на описі можливостей, функцій та інструментів, які вони надають для ефективного планування та управління проектами.

Завдання, яке вирішується в кваліфікаційній роботі полягає в тому, щоб за допомогою існуючих мов програмування розробити сервіс для зменшення об'єму роботи при плануванні складу команди виконання IT-проєктів.

Область застосування – розробка веб-додатків, що призначена для створення програмного продукту, який має широкі можливості застосування в галузі IT-індустрії.

Ключові слова: IT-проєкт, планування, epic, story, tasks, JavaSript, HTML.

ABSTRACT

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and four appendices. The total volume of the work is 87 pages, including 53 pages of the main part with 25 figures, 3 tables, 16 references and four appendices.

The purpose of the qualification work is to develop a service for planning and scheduling the composition of the IT project team, which will help to improve and simplify the work, as well as to select the optimal team composition.

The object of research is IT projects of varying complexity that require effective planning and management to achieve success through the selection of a team of specialists. The work focuses on a variety of projects, but the topic also explores which specialists can be involved in the project.

The subject of the research is computer-based planning systems used to manage IT projects. The main focus is on studying the principles and methodologies used in such systems, as well as describing the capabilities, functions, and tools they provide for effective project planning and management.

The task that is solved in the qualification work is to develop a service using existing programming languages to reduce the amount of work when planning the composition of the IT project team.

The field of application is the development of web applications, which is intended to create a software product that has wide application possibilities in the IT industry.

Keywords: IT project, planning, epic, story, tasks, JavaSript, HTML.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

CC	–	Cloud Computing;
AI	–	Artificial intelligence;
IoT	–	Internet of Things;
SM	–	Sales Manager;
PM	–	Project Manager;
BA	–	Бізнес-аналітик;
QA	–	Quality Assurance;
JS	–	JavaScript;
HTML	–	HyperText Markup Language;
API	–	Application Programming Interface.

ВСТУП

У сучасному світі комп'ютеризація зайняла важливе місце в більшості сфер життєдіяльності людини. Інформаційні технології стали необхідними для успішної роботи в будь-якій галузі, включаючи IT-індустрію. Однак, зростання складності проектів та збільшення обсягу робіт вимагає більш ефективного планування та управління проектами. У зв'язку з цим, розробка комп'ютерних систем планування стає надзвичайно важливою задачею.

Моя дипломна робота присвячена розробці комп'ютерної системи планування IT-проєктів, яка допоможе підвищити ефективність управління проектами в IT-індустрії. У цій роботі буде розглянуто питання планування проектів, їх виконання та контролю, а також вивчено сучасні методики та інструменти управління проектами. Розроблена система буде враховувати особливості IT-проектів та включати в себе необхідні функції для ефективного планування та управління проектами.

У роботі буде проведено дослідження сучасних методологій та інструментів управління проектами, а також будуть запропоновані власні рішення для покращення ефективності управління проектами в IT-індустрії. Для розробки системи буде використана сучасна технологія програмування, що дозволить створити потужну та надійну систему.

Результатом дипломної роботи буде функціональна комп'ютерна система планування складу команди IT-проєктів, яка буде дозволяти забезпечити ефективну організацію та управління проектами в IT-індустрії. Використання розробленої системи дозволить підвищити ефективність виконання проектів, зменшити час та зусилля на їх планування та контроль, а також забезпечити вчасне та якісне виконання завдань. Ще однією з головних переваг моєї розробки буде те, що система буде підбирати команду IT-спеціалістів, які будуть брати участь в розробці того чи іншого проєкту.

Актуальність теми. Тема "Комп'ютерна система планування ІТ-проєктів" залишається актуальною в сучасному світі, оскільки ІТ-індустрія продовжує швидко розвиватися, технології стають більш складними, а обсяги проєктів зростають. Компанії, що розробляють програмне забезпечення та інші ІТ-продукти, повинні забезпечувати ефективне планування, управління та контроль проєктами, щоб забезпечити успішне завершення проєктів вчасно, в межах бюджету та з високою якістю.

Комп'ютерні системи планування ІТ-проєктів допомагають організаціям забезпечити ефективне планування ресурсів, встановлення реалістичних термінів та оцінки вартості проєктів. Вони також дозволяють керувати ризиками, забезпечувати співпрацю між командами та збільшувати продуктивність проєктів. Такі системи є незамінним інструментом для успішного управління проєктами в сучасній ІТ-індустрії.

Окрім цього, зростання конкуренції в ІТ-індустрії зумовлює необхідність ефективного використання ресурсів та зниження витрат. Комп'ютерні системи планування ІТ-проєктів дозволяють знизити витрати та збільшити ефективність проєктів, що є важливим для бізнесу та його успішної діяльності.

РОЗДІЛ 1
АНАЛІЗ СТРУКТУРИ ІТ-ПРОЄКТУ, ТИПІВ КОМАНД РОЗРОБКИ ТА СКЛАДОВИХ ЧАСТИН ПРОЦЕСУ ПЛАНУВАННЯ

1.1 Аналіз ІТ-проєктів та їх структури.

Насамперед, потрібно розібратися що собою являє ІТ-проєкт. "ІТ-проєкт" - це проєкт, що передбачає розробку та впровадження інформаційних технологій для вирішення конкретної бізнес-проблеми або завдання в іншій галузі. Проєкт може включати розробку програмного забезпечення, баз даних, веб-сайтів, мобільних додатків та інших інформаційних продуктів. ІТ-проєкти можуть бути внутрішніми або позаштатними, а їхній успіх залежить від правильного планування, розробки та реалізації проєкту. Проєкт може бути як і самостійним, так і частиною програми.

Важливою складовою ІТ-проєкту є цінність, що являє собою корисність чогось, використання певних властивостей чи функцій. «Існують різні компоненти, такі як портфелі, програми, проєкти, продукти та операційна діяльність, які можна використовувати окремо чи разом для створення цінності. У поєднанні ці компоненти складають систему постачання цінності, що відповідає стратегії організації. На рис.1.1 зображено зразок системи постачання цінності з двома портфелями, які складаються з програм та проєктів. Тут також зображено окрему програму з проєктами та окремі проєкти, не пов’язані з портфелями та програмами . Будь-які проєкти або програми можуть містити продукти. Операційна діяльність може безпосередньо підтримувати та впливати на портфелі, програми та проєкти, а також на інші бізнес-функції, такі як нарахування заробітної плати, управління ланцюгами постачання тощо. Портфелі, програми та проєкти впливають як один на одного, так і на операційну діяльність.

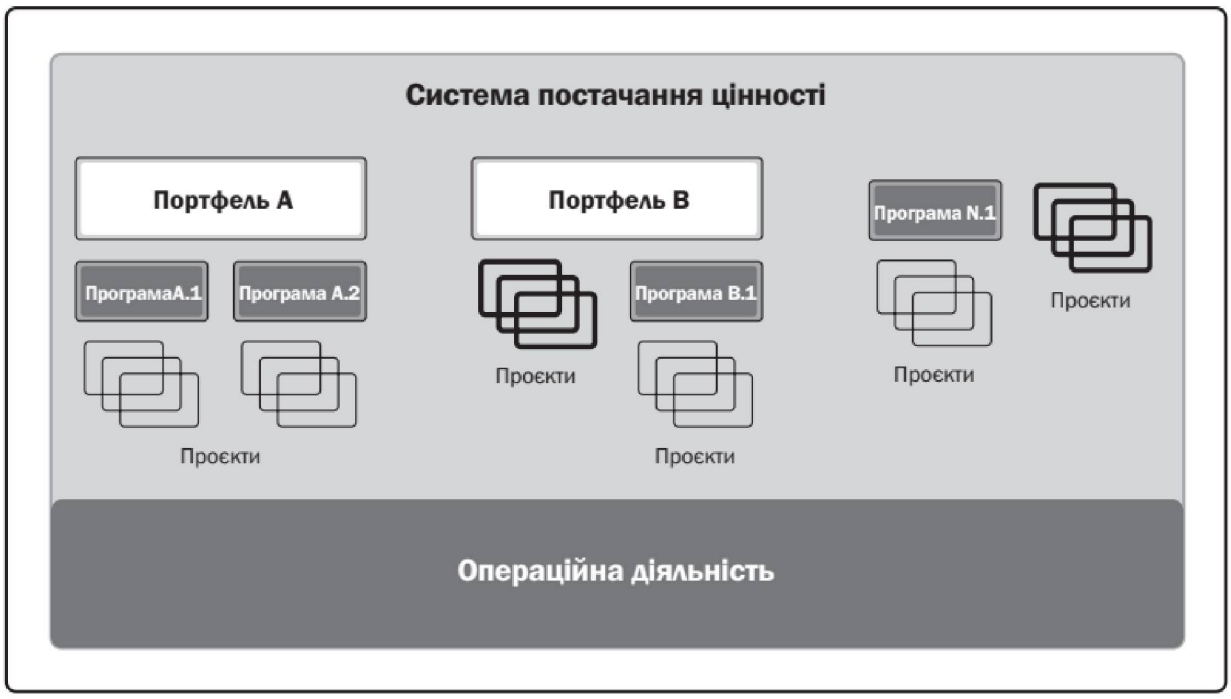

Рис 1.1 - Зразок системи постачання цінності

Як зображено на рис. 1.2, система постачання цінності є частиною внутрішнього середовища організації, яке залежить від політик, процедур, методологій, структур, конструкцій врядування тощо. Це внутрішнє середовище існує в більш широкому зовнішньому середовище, що охоплює економіку, конкурентне середовище, законодавчі обмеження тощо [1].

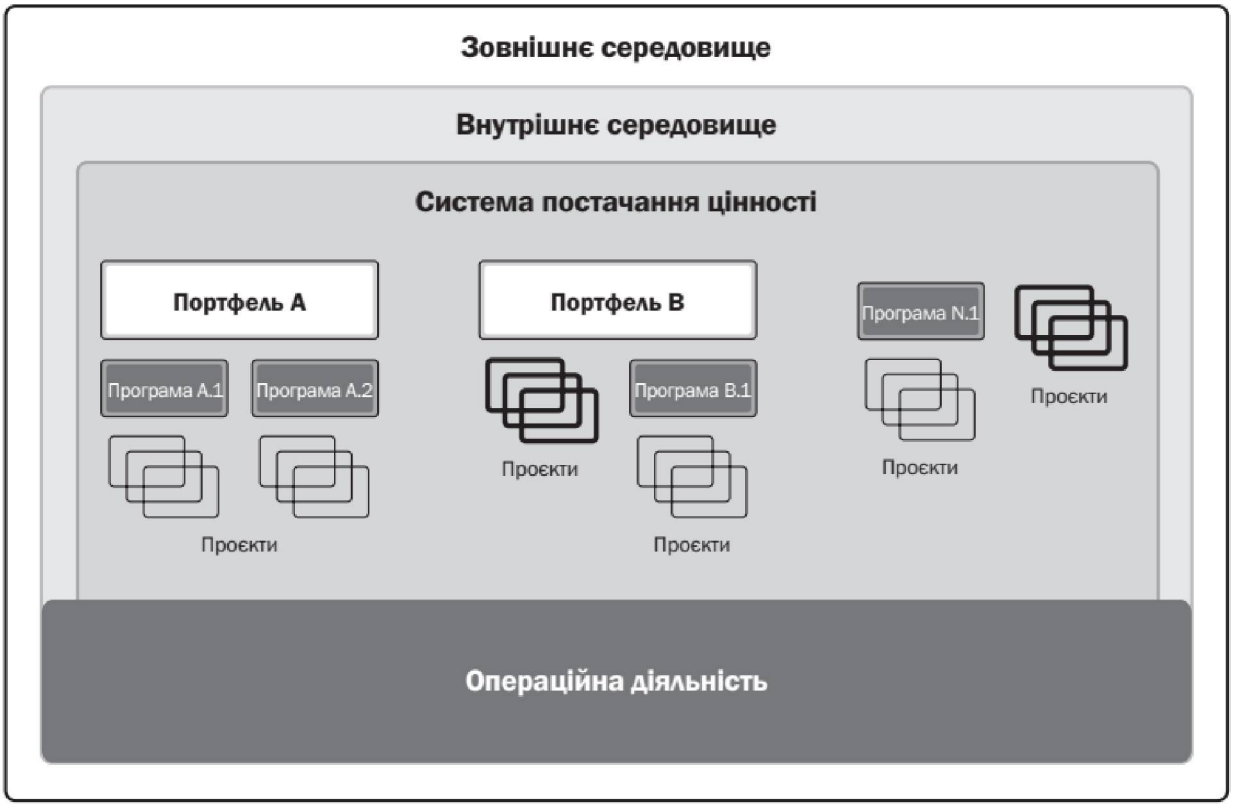

Рис.1.2 - Компоненти зразка системи постачання цінності

Для отримання кінцевих результатів використовують компоненти в системі постачання цінності, тобто доробки. «Розглянемо, що таке кінцевий результат – це підсумок чи наслідок проєкту чи процесу.» Для того, щоб підкреслити довгострокову ефективність проєкту, зосереджують увагу на кінцевих результатах. Здобутками є кінцеві результати, що створюють вигоди. Вигоди формують цінність, яка є вартісною, важливою або корисною.

Що розуміється під управлянням проєкту? Спочатку треба розібратися з тим, що таке продукт. «Продукт – це створений артефакт, що піддається кількісному оцінюванню та може бути як самостійним кінцевим виробом так і компонентом іншого виробу. Управління продуктом передбачає інтеграцію людей, даних, процесів та бізнес-систем для створення, підтримки та розвитку продукту або послуги впродовж їх життєвого циклу. Життєвий цикл продукту – це серія фаз, що відображає еволюцію продукту від концепту через зростання та зрілість до виходу з ринку.

Управління продуктом може ініціювати програми або проєкти в будь-який момент життєвого циклу продукту для створення або покращення конкретних компонентів, функцій або можливостей. Початковий продукт може бути результатом програми або проєкту. Упродовж свого життєвого циклу нова програма або проєкт можуть додавати або покращувати певні компоненти, атрибути або можливості, які створюють додаткову цінність для замовників та спонсорської організації. У деяких випадках програма може охоплювати повний життєвий цикл продукту або послуги для безпосереднього управління вигодами та створення цінності для організації [1][2].

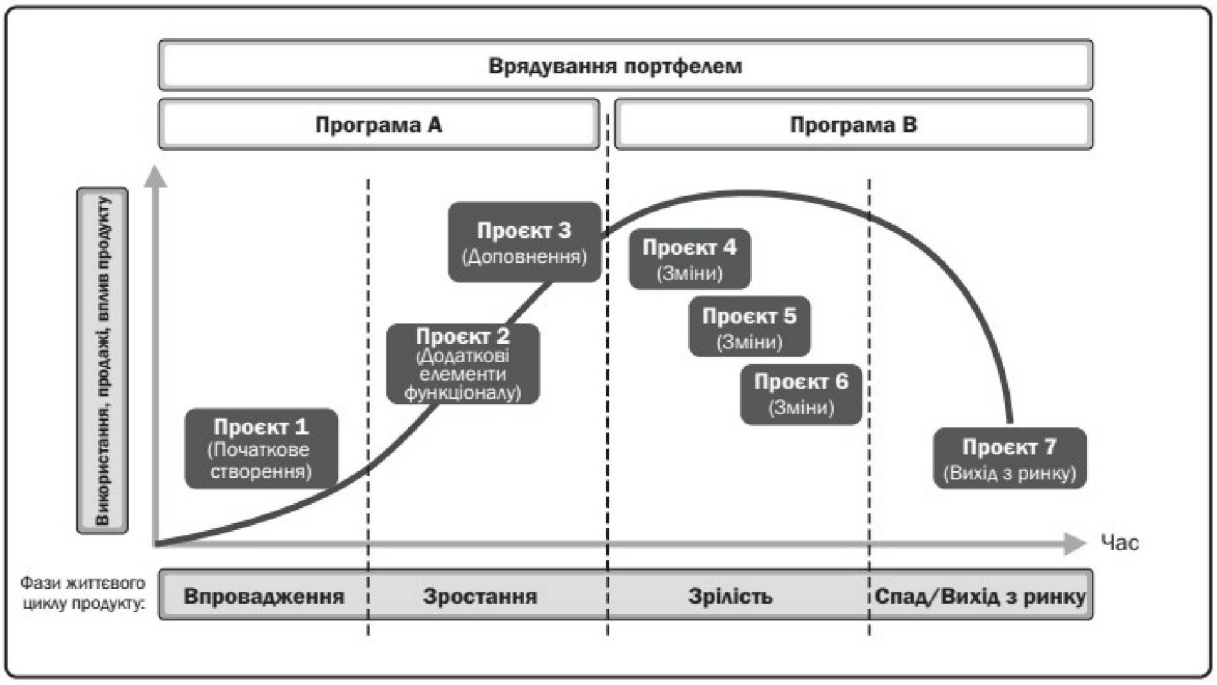

Рис 1.3 - Зразок життєвого циклу продукту

Головним у процесі виділення фаз, стадій та етапів проєкту полягає у визначенні контрольних точок, під час проходження яких використовується додаткова інформація і визначаються або оцінюються можливі напрямки розвитку проекту. Прийнятий поділ відображає взаємодію проекту з середовищем.

Ефективний виробничий процес ґрунтується на ітеративності, інкрементальності, самокерованості команди та адаптивності. Головним

принципом успіху в управлінні програмним проєктом є те, що не люди повинні пристосовуватися під обрану модель процесу, а модель процесу повинна підлаштовуватися під конкретну команду для забезпечення її найвищої продуктивності [3].

1.2. Аналіз принципів та моделей формування проєктної команди

Важливою складовою проєкту є проєктна команда, яка його виконує та веде до поставлених цілей. Команда – це група осіб, яка кваліфікована в певній галузі, об'єднана спільною метою, для досягнення якої взаємно погоджує роботу між собою.

Важливо розуміти потрібна команда взагалі чи можна обійтися без неї. Є деякі критерії, за допомогою яких можемо зробити висновок необхідна команда чи ні:

- для розв'язання важких завдань або питань;
- потрібна висока самовіддача;
- для ухвалення рішення необхідної згоди;
- необхідний багатобічний ключ;
- відповідний розширений діапазон кваліфікованості;
- багато різновидів невизначеності;
- використання командної праці для розробки стратегії.

Для блискучого результату проєкту треба ефективна команда. Наведу таблицю 1.1, де вказані чинники ефективності.

Таблиця 1.1

Чинники ефективності команди

Ефективна команда	Неефективна команда
1. Вдала співдія в команді.	1. Відсутність одного з чинників ефективності.
2. Індивідуальний підхід задля задоволення потреб членів команди.	2. Обрання не найсприятливішого варіанту розв'язання задачі, а загальноприйнятого.

Закінчення таблиці 1.1

3. Досягнення командою спільних поставлених цілей.	3. Різноманітний рівень сучасного мислення членів команди
	4. Прийняття командного рішення займає більше часу, ніж індивідуального.

Кількість спеціалістів та професійний ступінь залежить від особливостей проєкту. Для кожного з членів команди визначені їх особливі функції, повноваження та зона відповідальності. Передусім, для успішної реалізації проєкту треба гармонійна робота всього колективу задля досягнення спільної мети.

Для управління проєктом існує два основних принципу формування команд: створення власних груп, тобто одна група від замовника, а інша від виконавця, або створення єдиної команди [4].

В цілому виділяють чотири головні підходи щодо формування самої команди:

1. Цілевизначальний підхід – посилається на цілі проєкту, за загальну мету можна обрати стратегічну чи оперативну мету, бо саме даний підхід дає можливість вільно розбиратися при обранні групової цілі.

2. Рольовий підхід – формується на переговорах серед членів групи, де вирішується хто, яку роль займає.

3. Проблемно-орієнтований підхід – основується на досягненні очікуваної цілі за допомогою вирішення деяких проблем.

4. Міжособистісний підхід – сконцентрований на покращенні стосунків між членами команди.

Для команд в сфері ІТ характерне те, що чотири головні підходи об’єднуються. Завдяки цьому можна розподілити ролі, сформулювати

функціональні зони, досягти бажаної мети. Важливим є виокремлення проблем, які існують на даний момент та які потребують негайного рішення, та ступінь невизначеності на перших етапах проєкту.

При утворенні внутрішньоорганізаційного проєкту зазвичай у пріоритеті співпрацівники зі штату фірми. Проведення оцінки кадрового персоналу, навантаження сотрудників, взаємини між працівниками фірми або окремого відділу [5].

За потреби створення команди для реалізації проєкту за допомогою декількох компаній, потрібно належним чином знати вимоги в рівні кваліфікованості, психологічний портрет, фахові характеристики, а ще й визначити ролі членів команди, їхню кількість та рівень професіоналізму.

Виділяють п’ять етапів для створення команди ІТ-проєкту [3]:

1. Сформульовані завдання – окреслюється чорновий варіант задуму проєкту та орієнтовна думка щодо етапів та завдань.

2. Формування рольової моделі – відбувається розподіл завдань за кваліфікацією та досвідом роботи, враховується необхідна кількість спеціалістів. Обирається певна модель, за якою команда працюватиме.

3. Узгодження – обмін думками й схвалення рольової моделі, що визначили на попередньому етапі.

4. Формування умов і потенціалу – визначення місць та умов праці, розбиття завдань.

5. Створення команди – розгляд потенційних претендентів на необхідні ролі.

Команда виникає з наявного штату чи з нових кандидатів. У свою чергу даний етап включає в себе формування сприятливих взаємин у колективі та поєднання спеціалістів конкретної технології. Цей процес розпочинається після етапу підписання усіх договорів щодо проєкту. PM розпочинає процедуру підбору команди, згідно зі складеним планом розробки ІТ-проєкту. Необхідно відібрати спеціалістів різного рівня з обраних технологій та провести співбесіду на предмет підтвердження знань та вмінь [6].

Для результативного виконання всіх поставлених задач потрібно створити «ефективну команду» (табл.1.1).

Ефективна команда має певні характеристики та особливості:

1. Професійна ефективність – вся команда працює на здобуток остаточного результату проєкту, креативне бачення та ініціатива членів команди на плідну працю.

2. Організаційно-суб'єктивна ефективність – атмосфера, в якій присутні узгоджені рішення, взаємоповага, кожен з учасників прислухається до думки іншого, усі питання вирішуються на підставі загальних думок, ідей та йдуть на компроміс, а не з точки зору більшості голосів.

Робота по створенню команди ІТ-проєкту відбувається протягом усього робочого процесу, а не завершується затвердженням штату. Тому й надалі будуть з'ясовувати виконання певних задач, ключові моменти проєкту, відносини між членами команди, склад тощо.

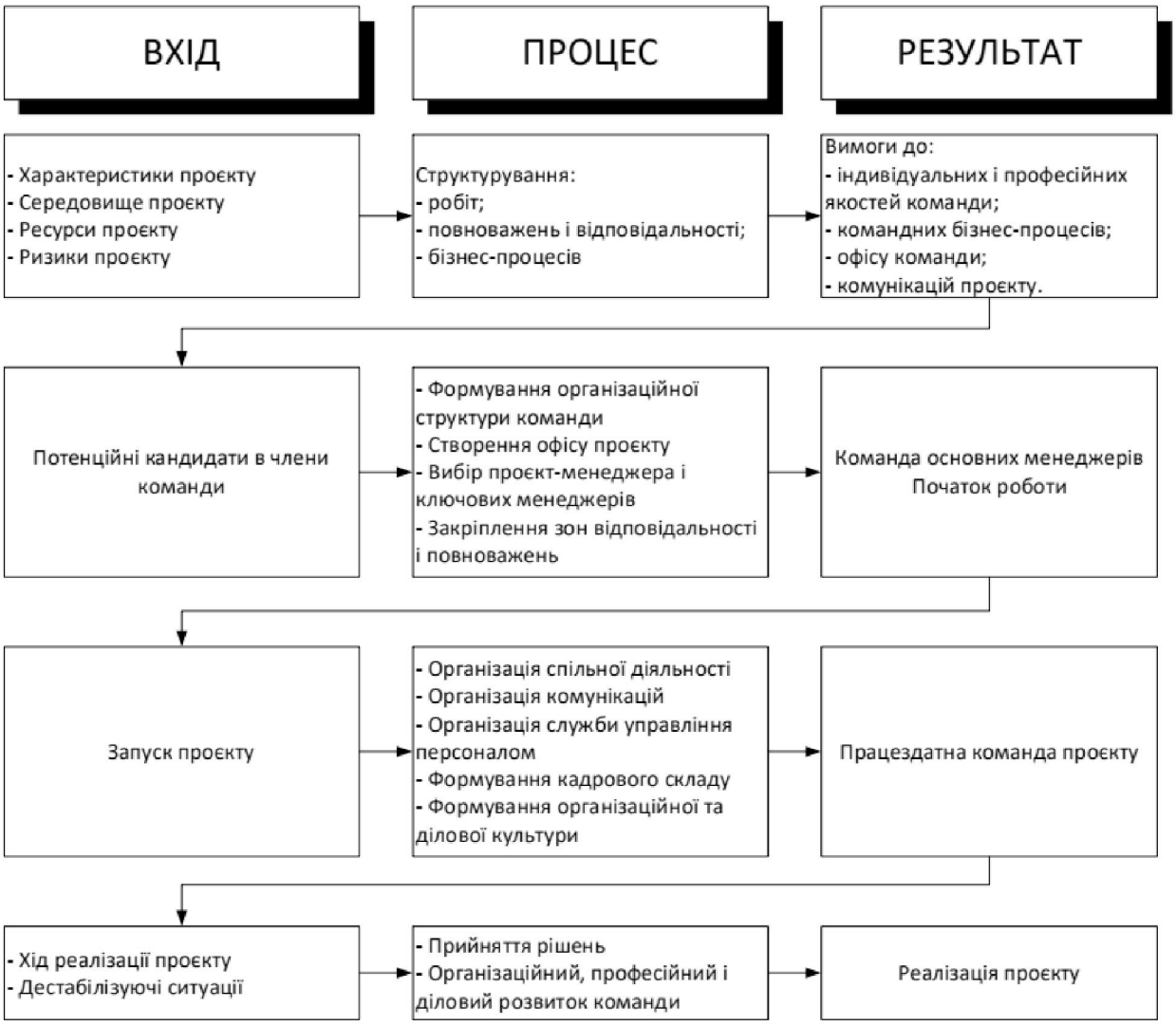

Рис. 1.4 - Зразок створення команди проєкту, що вважається ефективною

Команда проєкту по його завершенню розформовується. Якщо в подальшому штат команди залишиться незмінним, то в цьому випадку все одно команда стане іншою через різні задачі, ключові моменти та рівень взаємодії тощо.

Побудова команди ІТ-проєкту залежить масштабу самого проєкту, способу управління та особливості цілі. Виділяють декілька різновидів побудов проєктних команд, який має позитивні та негативні властивості:

1. Ізоморфна побудова – команда, що складається зі спеціалістів, які однакові за знаннями та працюють паралельно, завдяки чому їх простіше контролювати. У даному випадку члени команди можуть легко ділитися

порадами та досвідом. Негативним моментом є те, що у більш складних проєктах таку побудову не використовуватимуть.

2. Експертна побудова – визначається автономією команди при нагляді менеджеру проєкту. З недоліків: пливке розбиття відповідальності між учасниками проєкту, недоречний розподіл зобов'язань.

3. Колегіальна побудова – формується на основі інформаційного обігу знань та ідей, об'єднаності рольових груп та високій працездатності. Мінусом можна зазначити самоврядування окремих підгруп та велике навантаження потоком даних.

4. Хірургічна побудова – характеризується злиттям окремих підгруп в одну єдину, розбиттям проєкту на менші за розміром проєкти, але також взаємозалежні між собою. Можна сказати, що такою побудовою гарно управляти. Але без слабких сторін не обійтися і в такому випадку: усе підпорядковано лідером проєкту, висока можливість появи суперечностей та перевантаження через кількість інформації.

Виокремлюють ще такі типи команд IT-проєктів:

- «In-House» - цілісне територіальне місцезнаходження всієї команди. Здебільшого все зосереджено в одній організації, у якій штатний склад реалізує свої задачі, паралельно обговорюючи це.
- «Розподілена команда» - відмінне територіальне місцезнаходження членів команди чи дистанційна робота. Також важливо відмітити, що команда для IT-проєкту може бути створена не зі співпрацівників організації, а залученням з інших фірм для скорочення витрат.

Зараз в основному зазвичай використовується саме розподілений тип команди, бо він також є ефективним. Нижче наведена таблиця з плюсами та мінусами використання таких типів команд.

Таблиця 1.2

Властивості притаманні командам «In-House» та «Розподілена команда»

«In-House»	«Розподілена команда»
Плюси	
- Миттєвий контакт. - Розмови в команді в реальному житті. - Хвилинне розповсюдження даних, усі одночасно проінформовані. - Нагляд за роботою та усіма процесами.	- Працівники, що працюють з дому – недорогий розрахунок за роботу. - Своєрідне налагодження виконання завдань.
Мінуси	
- Відволікання на інші розмови, які не стосуються проєкту.	- Знаходження робітників у різних країнах і , як наслідок, розбіжність у часі. - Дефіцит «живого» спілкування та з'являються складності у тому, що можуть бути певні розбіжності у командних ідеалах.

Для принципів розробки IT-проєктів вдаються до різних методів. Їх можна класифікувати за структурою та властивостями. Обрання моделі залежить від деяких факторів: терміну виконання, бюджетування, деталізацій проєкту, прерогативи керівника тощо [8].

1.3 Аналіз методологій управління розробкою IT-проєкту.

В IT-галузі представлені два головних методології щодо управління проєктною командою: Waterfall та Agile.

Підхід Waterfall є стандартною моделлю, яка була популярною на витоках епохи розробки, але водночас використовується й на сьогодення. Ідея

полягає в наступному: кожний черговий етап реалізується тільки тоді, коли цілком закінчено колишній. Етапи чітко окреслені та технологія виконується начебто каскадом, крок за кроком йти згідно інструкції [9].

Дана модель розробки відносна строга й присутні вимогливі правила. Негативною стороною є те, що повернутися на попередній етап нелегко. Внесення змін до дійсного проєкту має велику ціну та труднощі. Описаний метод відповідає стандартам лаконічно розпланованих проєктів, у якому є чіткий зміст створюваного та слідування точних цілей та вимог.

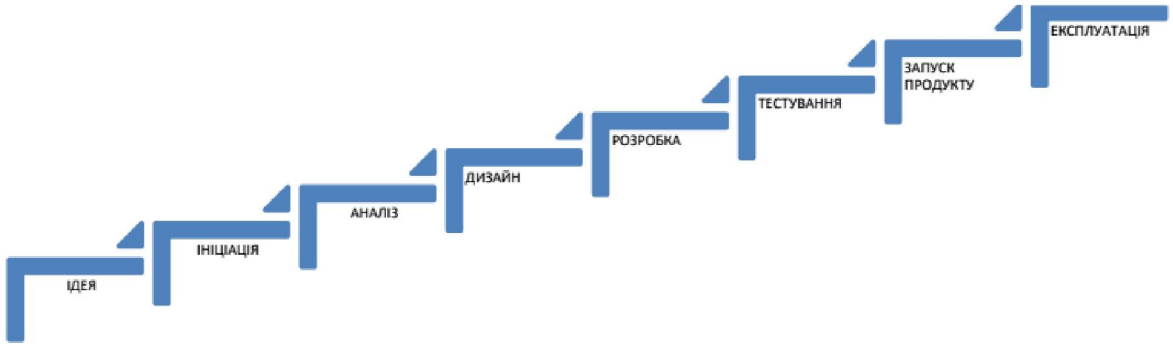

Рис. 1.5 - Схема дій у Waterfall

При користуванні цим способом в IT-проєктах були внесені коректування, тобто він став ітераційним. Також зробили етапи перевірки для запобігання виникненню помилок.

Другим підходом є більш гнучка методологія Agile. Вона є кращою для деяких розробників через те, що можна створювати продукт, у якого не повністю сформована ідея. Головний плюс цього способу припадає на те, що замовник дивитиметься за процесом та може вносити корективи. Це стало можливим за допомогою визначення спринтів – визначених часових проміжків, за які втілюються задачі. При обміні думками визначаються цілі, часові діапазони, де розробники реалізують задачі, які потрібно. Пізніше призначається обговорення для коригування та додавання нових відрізків.

Слабкою стороною є неможливість визначення точної ціни через нерозуміння кінцевого продукту та тривалість втілення ідеї.

Метод Agile безумовно підходить для проєктів з довготривалим життєвим циклом та адаптованим під перетворення на ринку. Це дозволяє оновлювати продукт та змінювати його відповідно до нових вимог [9].

Нижче наведені рис. 1.6 та табл. 1.3, де описані ключові особливості моделей Waterfall та Agile, і ще й поняття, які підкажуть коли доречніше використовувати одну з двох методологій.

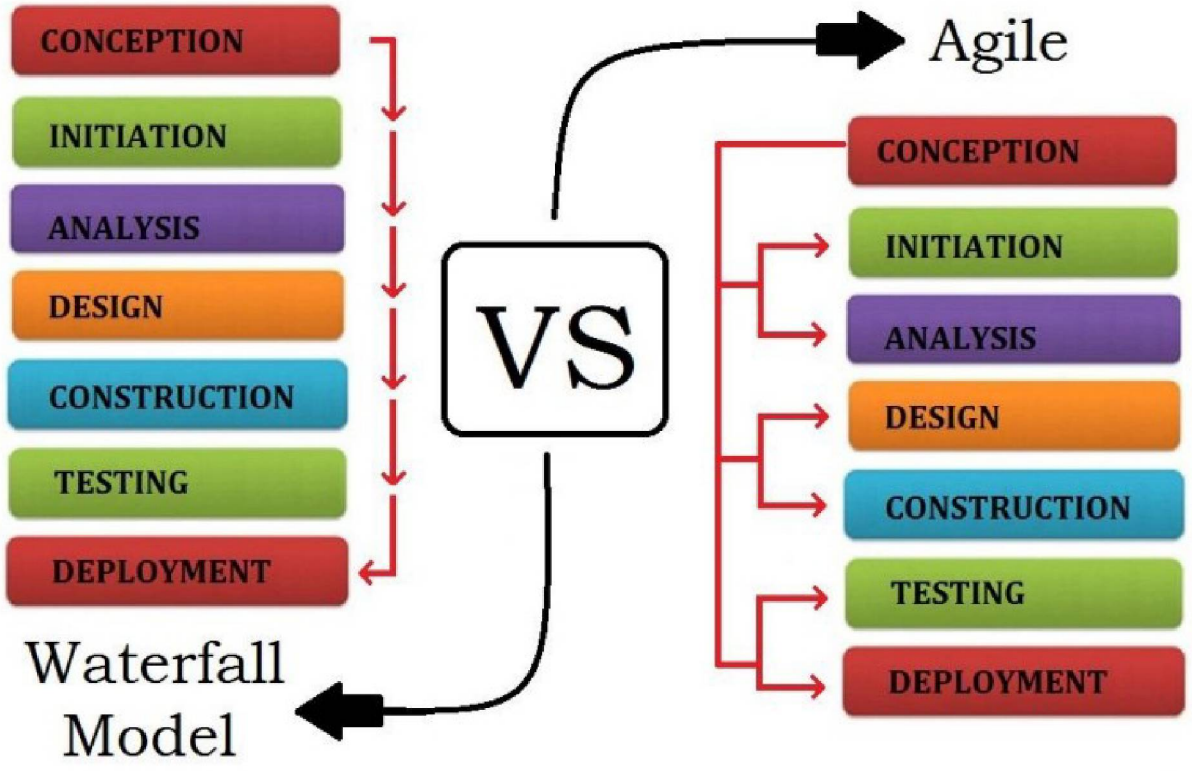

Рис. 1.6 - Різниця методів Waterfall та Agile.

Scrum, Kanban та Scrumban - це методології управління проектами, які спрямовані на створення високоякісних продуктів в ефективний та гнучкий спосіб. Ось коротка характеристика кожної з них [10]:

Scrum: Scrum - це гнучкий фреймворк для управління проектами, який наголошує на командній роботі, ітеративному розвитку та постійному вдосконаленні. Scrum -команди працюють у спринтах, які є обмеженими у часі періодами роботи, і організовані навколо конкретних ролей, включаючи Scrum -майстра, власника продукту та команду розробників.

Kanban: Kanban - це методологія ощадливого виробництва, яка наголошує на візуалізації роботи, обмеженні незавершеного виробництва та управлінні потоком. Команди Kanban використовують дошку для візуалізації своєї роботи і переміщення робочих елементів через ряд етапів, від "потрібно зробити" до "зроблено".

Scrumban: Scrumban - це гібридний підхід, який поєднує в собі елементи Scrum та Kanban. Як і Scrum, Scrumban передбачає роботу в спринті, але також наголошує на постійному вдосконаленні, обмеженні незавершеного виробництва та візуалізації роботи, як і в Kanban. Scrumban особливо добре підходить для проектів, які вимагають високого ступеня гнучкості та адаптивності.

У процесі створення команди для ІТ-проєкту переважна кількість моделей команд базується на функціональному підході, який включає розбиття учасників команди та їх співдружність. Для обрання керівника/лідера проводять оцінку індивідуального та психологічного підходу.

Проєктна робота – одна з недостатньо мотивованих праць, бо суцільного результату можна досягти при завершенні проєкту, але при зміні термінів мотивація працівників прямує до нуля, що схиляє до появи різноманітних ризиків проєкту [11].

Таблиця 1.3

Зіставлення методів Waterfall та Agile

Waterfall	Agile
- Черговість. - Чітко окреслена інструкція. - Суворе виконання процесів. - Гарний продукт.	- Гнучкість . - Просто додавати зміни в проєкт. - Продукт удосконалюється в процесі розробки. - Гарний продукт.
Коли користуватися Waterfall?	Коли користуватися Agile?

- Чітке уявлення про продукт та відповідність вимогам. - Замовник одразу каже, що хоче отримати по завершенню, та не має змоги щось змінювати. - Після виконання роботи кожен отримує попередньо оговорену оплату.	- Нечітко сформовано уявлення про кінцевий результат. - Можливість внесення коректівів від замовника. - Головним завдання є швидке отримання придатного до роботи продукту.

Також важливо зазначити, що тільки 30% IT-проєктів увінчалися успіхом, решта «провалилася» чи завершилися з недотриманням спринтів. У більшості випадків причиною провалів виявилися умови та ризики, що частіше за все залежні від людського фактору. Запобігти виникненню таких проблем можна було б з допомогою більшої концентрації уваги на етапі створення команди для IT-проєкту, а точніше на розподіленні задач та виконанні функцій.

Взагалі треба відповідально підходити до процесу планування усього проєкту. Розглянемо більш детально тему планування IT-проєкту. Процес структурного планування включає в себе кілька етапів.

По-перше, проєкт розбивається на низку окремих завдань, які мають бути виконані, щоб проєкт був успішним. Потім будується мережева діаграма, яка описує послідовність виконання цих завдань. Потім оцінюються терміни виконання кожного завдання й аналізується мережевий графік.

Мережевий графік є найважливішим компонентом цього етапу, оскільки він забезпечує візуальне представлення діяльності проєкту та їхніх часових взаємозв'язків. Мережева діаграма має відповідати певним правилам, наприклад, кожне завдання має бути представлене тільки однією вершиною, і жодне завдання не може розпочатися, доки не будуть завершені всі попередні завдання. Крім того, жодне завдання не може розпочатися, якщо воно слідує

безпосередньо за іншим завданням, поки не буде завершено попереднє завдання[12].

Нарешті, віхи використовуються для позначення початку або кінця найважливіших етапів проєкту, і їх представлено завданнями з нульовою тривалістю на початку і наприкінці проєкту.

Мережевий графік є корисним інструментом для визначення критичних робіт і критичного шляху проекту на основі тривалості кожної роботи.

Критичний вид діяльності - це вид діяльності, затримка якого призведе до затримки всього проєкту, оскільки він не має запасу часу. З іншого боку, некритичні роботи можуть бути відкладені в межах їхнього часового запасу без впливу на час завершення проєкту.

Критичний шлях - це шлях, який починається з початкової вершини і закінчується в кінцевій вершині мережевої діаграми, що проходить тільки через критичні роботи. Загальна тривалість робіт на критичному шляху - це мінімальний час, необхідний для завершення проєкту.

Пошук критичного шляху включає два етапи: розрахунок часу раннього початку для кожного виду діяльності, тобто найранішого часу, коли робота може розпочатися, і розрахунок часу пізнього початку для кожного виду діяльності, тобто найпізнішого часу, коли робота може розпочатися без збільшення тривалості проєкту.

Коли йдеться про критично важливі види діяльності, резерву часу не існує. Тому менеджери проєкту мають зосередитися насамперед на забезпеченні своєчасного завершення цих робіт. Навпаки, некритичні роботи мають резерв часу, більший за нуль, що дає менеджерам певну гнучкість у коригуванні термінів цих робіт і відповідному розподілі ресурсів.

Два можливих варіанти управління некритичними видами діяльності включають відтермінування початку робіт на суму, що не перевищує резерв часу, і перерозподіл ресурсів на критичні види діяльності.

Таким чином можна скоротити тривалість критичних робіт і проєкту загалом.

Інший варіант - допустити недовикористання ресурсів для некритичних робіт, що дасть змогу збільшити тривалість некритичних робіт у межах резерву часу. Ресурси, що звільнилися, можуть бути використані для завершення критичних робіт, що в кінцевому підсумку скоротить тривалість критичних робіт і всього проєкту.

На етапі календарного планування створюється діаграма Ганта для представлення графіка проєкту. Діаграма Ганта надає інформацію про такі аспекти проєкту:

- розташування завдань, яке визначається мережевою діаграмою;
- ресурси, необхідні для кожного завдання, і порядок їх розподілу;
- конкретні дати початку та закінчення виконання завдань.

Використання трудових ресурсів протягом усього проєкту можна зобразити за допомогою графіка, побудованого на основі діаграми Ганта. Цей графік ілюструє відсоток роботи, виконаної певним ресурсом протягом усього терміну проекту. Вісь х являє собою часову шкалу проєкту, а вісь у показує загальний відсоток завантаження працівника за всіма завданнями проєкту, які виконуються на цей момент [13].

Зазвичай працівник повністю виконує одне завдання, перш ніж перейти до наступного, внаслідок чого його завантаженість становить 100%. Однак трапляються випадки, коли працівник одночасно виконує кілька завдань, розподіляючи свій час відповідним чином. Наприклад, два завдання по 50% кожне, що означає, що половина робочого дня йде на кожне з них. Графік використання ресурсів дає змогу відстежувати загальне робоче навантаження працівника, виявляючи випадки перевантаження, коли запланована ним робота перевищує його можливості щодо виконання протягом робочого дня, про що свідчить загальне робоче навантаження понад 100%.

Етап оперативного управління охоплює виконання робіт за проєктом і постійний контроль за їхнім перебігом. Хоча початковий план може бути добре продуманий, під час роботи неминуче доведеться вносити корективи.

Тому до завдань менеджера входить:

- відстеження фактичного графіка виконання робіт;
- порівняння фактичного графіка із запланованим;
- ухвалення рішень щодо усунення відхилень від плану, що виникають;
- зміна графіка проєкту в разі виникнення значних відхилень.

Перші два завдання розв'язують за допомогою діаграми Ганта, яка дає змогу легко відстежувати відсоток фактичного виконання кожного завдання і виявляти будь-які відхилення. Метод усунення відхилень залежить від наявних у менеджера ресурсів. Для завершення роботи, що відстає, можна залучити додаткових працівників (ресурси) або змусити ту саму кількість працівників працювати понаднормово, але це збільшить вартість проєкту [14].

Якщо відхилення занадто велике, щоб виправити його за допомогою додаткових ресурсів або понаднормової роботи, або якщо вартість проєкту не може бути збільшена, проєкт має бути перепланований. Для цього необхідно присвоїти завершеним роботам нульові значення тривалості, установити значення тривалості частково завершених завдань відповідно до обсягу робіт, що залишився, внести структурні зміни до мережевої діаграми для усунення непотрібних робіт і додавання нових робіт, перерахувати критичний шлях і змінити графік проєкту. Після того як скоригований план проєкту затверджено вищим керівництвом, можна приступати до його реалізації та оперативного управління. Цей процес може знадобитися повторити кілька разів.

1.4. Мета, предмет, об’єкт та постановка задачі.

Мета дослідження "Комп'ютерна система планування ІТ-проєктів" - дослідження та розробка комп’ютерної системи планування складу команди для проєктів різної тривалості.

Предмет дослідження - комп'ютерні системи планування, які застосовуються для керування ІТ-проєктами. Основний акцент зроблено на вивченні принципів та методологій, які використовуються в таких системах, а

також на описі можливостей, функцій та інструментів, які вони надають для ефективного планування та управління проектами.

Об'єкт дослідження - ІТ-проєкти різної складності, що потребують ефективного планування та управління для досягнення успіху за рахунок підбору команди спеціалістів. У роботі зроблено акцент на розгляд різноманітних проєктів, але тема також досліджує, які спеціалісти можуть бути залучені до виконання проєкту.

Постановка задачі полягає в дослідженні комп'ютерних систем планування ІТ-проєктів та їх впровадженні в практику управління проєктами. В роботі досліджуються наступні питання:

1. Вивчення методологій та принципів, які використовуються в комп'ютерних системах планування ІТ-проєктів;
2. Дослідження функцій та інструментів, які надають комп'ютерні системи планування ІТ-проєктів для ефективного планування та управління проєктами;
3. Вивчення можливостей використання комп'ютерних систем планування ІТ-проектів для різних видів ІТ-проєктів.

Висновки за розділом 1

Було розглянуто основні поняття, які дозволили отримати загальне уявлення про основні складові частини процесу планування ІТ-проектів, типи команд розробки та важливість забезпечення якості в процесі планування проектів. Загальна структура ІТ-проекту та його етапи дають можливість чітко визначити всі необхідні етапи проекту та організувати їх в логічну послідовність, що дозволить досягти успішної реалізації проекту.

РОЗДІЛ 2
ПРОЄКТУВАННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ЗАСОБУ ПЛАНУВАННЯ ІТ-ПРОЄКТІВ

2.1. Різні типи команд в ІТ.

ІТ-сфера охоплює багато різних технологій, які допомагають створювати програмне забезпечення, мережі та інші інформаційні системи. Розглянемо декілька з найбільш поширених технологій в ІТ-сфері:

1. Cloud computing (CC) - це технологія, що дозволяє доступатися до ресурсів (включаючи обчислювальну потужність, зберігання даних та програмне забезпечення) через Інтернет. За допомогою цієї технології компанії можуть зменшувати витрати на обладнання та забезпечувати більшу мобільність для своїх працівників.
2. Artificial intelligence (AI) - це технологія, що дозволяє комп'ютерам виконувати завдання, які зазвичай вимагають людської інтелектуальної діяльності, такі як розпізнавання образів, мови та прийняття рішень. AI використовується для створення різноманітних продуктів, від мобільних додатків до роботів та автономних автомобілів.
3. Internet of Things (IoT) - це технологія, що дозволяє різним пристроям підключатися до Інтернету та взаємодіяти між собою. За допомогою цієї технології можна створювати різноманітні системи, такі як "розумний дім", "розумне місто" та "розумна фабрика".
4. Blockchain - це технологія, що дозволяє створювати безпечні та надійні системи збереження даних. В основі цієї технології лежить розподілена база даних, яка дозволяє кожному користувачеві внести свій внесок до системи та перевіряти цілісність даних.

5. DevOps - це технологія, що поєднує розробку програмного забезпечення та операції, що дозволяє автоматизувати процеси розробки, тестування, випуску та моніторингу програмного забезпечення. DevOps допомагає зменшити час випуску програмного забезпечення та забезпечує більшу стабільність та безпеку програмного забезпечення.
6. Agile - це методологія розробки програмного забезпечення, яка дозволяє команді розробників більш гнучко підходити до змін вимог та швидше відповідати на потреби клієнтів. Agile підтримує постійну взаємодію між розробниками та клієнтом, що допомагає зменшити ризик невдачі та підвищити якість програмного забезпечення.
7. Virtualization - це технологія, що дозволяє використовувати віртуальні пристрої замість фізичних. Віртуалізація дозволяє зменшити кількість фізичних пристроїв, що потрібні для роботи, знизити витрати на обслуговування та забезпечити більшу мобільність для працівників.
8. Big Data - це технологія, що дозволяє обробляти та аналізувати великі обсяги даних. Big Data допомагає компаніям отримувати цінну інформацію про своїх клієнтів та ринок, що дозволяє їм приймати більш обґрунтовані рішення.
9. Cybersecurity - це технологія, що дозволяє захищати комп'ютерні системи від злочинних атак та витоку конфіденційної інформації. Cybersecurity включає в себе різні методи та технології, такі як антивірусне програмне забезпечення, мережеві файрволи, системи виявлення вторгнень, системи автентифікації та шифрування даних. У світі, де все більше даних зберігається в електронному вигляді, кібербезпека є критично важливою для захисту конфіденційної інформації, відкриття нових бізнес-можливостей

та забезпечення надійності функціонування комп'ютерних систем [15].

Як можна помітити сфера технологій в IT багатоманітна та кожен може знайти застосування своїх знань. Можна робити спроби в різних напрямках та остаточно зупинитись на тому, що цікавить найбільше.

Проананалізуємо спеціалістів, які обов'язково повинні бути присутні в кожній IT-компанії та є затребуваними:

- Delivery Manager відповідає за координацію інтеграції та поставку продукту або послуги в зазначений термін та бюджет. Він забезпечує взаємодію між різними командами, що беруть участь у процесі розробки та впровадження продукту, забезпечує планування та управління ресурсами та бюджетом, а також забезпечує звітність про прогрес поставки.
- Sales Manager (SM) - основна роль полягає в нагляді за діяльністю компанії або команди з продажу та забезпеченні досягнення поставлених цілей і завдань.
- Project Manager (PM) - відповідає за планування, координацію та управління проєктом, включно з управлінням бюджетом, контролем термінів виконання робіт і управлінням ризиками.
- Архітектори відповідають за проектування архітектури програмного забезпечення та інфраструктури в IT. Вони визначають, як різні компоненти програмного забезпечення будуть працювати разом, які технології використовуватимуться та як буде забезпечено безпеку та масштабованість системи. Розрізняють 6 типів архітекторів: архітектор програмного забезпечення, інфраструктурний архітектор, архітектор даних, архітектор безпеки, архітектор веб-додатків, архітектор мережі. Кожний з них займається вузькою спеціалізацією свого профілю.
- Бізнес-аналітик (BA) відповідає за збір, аналіз та документування вимог до програмного забезпечення від бізнес-замовника. Основна мета бізнес-аналітика полягає в тому, щоб зрозуміти бізнес-проблеми та

потреби клієнта, і допомогти команді розробників створити програмне забезпечення, яке відповідатиме цим потребам.

- Tech Lead - це технічний експерт, який відповідає за технічне керівництво проектом або командою розробників. Основна мета технічного лідера полягає в тому, щоб забезпечити високу якість технічного рішення та забезпечити успішне виконання проекту з технічної точки зору.
- QALead відповідає за забезпечення якості програмного забезпечення. Основна мета QALead полягає в тому, щоб забезпечити високу якість продукту та надати сприятливі умови для ефективної роботи команди розробників.
- Backend-розробники - це фахівці, які займаються розробкою серверної частини програмного забезпечення. Основна мета спеціалістів Backend полягає в тому, щоб забезпечити ефективну та безперебійну роботу серверної частини продукту.
- Frontend-розробники - це фахівці, які займаються розробкою клієнтської частини програмного забезпечення, тобто того, що бачить користувач на сторінках веб-сайту або в мобільних додатках. Основна мета спеціалістів Frontend полягає в тому, щоб забезпечити користувачам зручний та привабливий інтерфейс, з яким вони можуть легко взаємодіяти.
- QA (Quality Assurance) - це фахівці, які займаються забезпеченням якості програмного забезпечення. Основна мета QA спеціалістів полягає в тому, щоб перевірити, чи відповідає продукт вимогам замовника, чи він працює правильно та ефективно, і чи не містить він помилок, які можуть призвести до негативного впливу на користувача.
- Дизайнери - це фахівці, які займаються розробкою дизайну для веб-сайтів, мобільних додатків та інших цифрових продуктів. Вони використовують свої знання в галузі дизайну та технологій, щоб

створювати естетичні та привабливі продукти, які були б зручні та функціональні для користувачів

- Тестувальники - це фахівці, які виконують тестування програмного забезпечення, щоб перевірити, чи відповідає воно вимогам замовника, чи воно працює правильно та ефективно, і чи не містить воно помилок, які можуть призвести до негативного впливу на користувача.

Також варто зазначити, що існує декілька ступенів програмістів, які відрізняються рівнем досвіду та навичок. Основні з них:

1. Junior Developer - це початківець, який має базові знання та досвід у програмуванні. Він може брати участь у розробці простих додатків та функціоналу під керівництвом більш досвідченого розробника.
2. Middle Developer - це розробник середнього рівня, який має досвід у програмуванні та розуміє основні концепції розробки програмного забезпечення. Він може самостійно розробляти додатки та функціонал, вирішувати складні технічні завдання та працювати зі складним кодом.
3. Senior Developer - це розробник високого рівня, який має глибокі знання у програмуванні та великий досвід у розробці програмного забезпечення. Він може брати участь у складних проектах, керувати командою розробників та вирішувати складні технічні завдання

Залежно від рівня складності проєкту, IT-команди можуть мати різний склад. Основними складовими IT-команд є розробники, тестувальники та менеджери проекту. Детальніші складові команд для виконання проєкту в залежності від рівня складності можуть бути наступними:

1. Простий проєкт.

Для простих проєктів, таких як створення простого веб-сайту або мобільного додатку, IT-команда може складатися з наступних складових:

- Один або декілька розробників (back-end та/або front-end);

- Один тестувальник;
- Один менеджер проєкту;
- Один бізнес-аналітик;
- Один дизайнер.

2. Середній проєкт.

Для середніх проєктів, таких як розробка складної мобільної або веб-платформи, склад IT-команди може бути наступним:

- Декілька back-end та/або front-end розробників;
- Декілька тестувальників, включаючи автоматизоване тестування;
- Один або декілька UI/UX дизайнерів;
- Один менеджер проєкту та один бізнес-аналітик.

3. Складний проєкт.

Для складних проєктів, таких як створення великої корпоративної системи або розробка складної мережі, склад IT-команди може включати наступні складові:

- Декілька back-end та/або front-end розробників;
- Кілька тестувальників, включаючи автоматизоване тестування та тестування безпеки;
- Декілька UI/UX дизайнерів та/або інформаційних архітекторів;
- Один або декілька технічних лідерів та один архітектор системи;
- Один менеджер проєкту та один бізнес-аналітик.

Важливо, щоб склад IT-команди відповідав потребам проєкту і був достатньо гнучким, щоб можна було змінювати його у процесі розробки, якщо це необхідно. Також важливо, щоб кожен член команди міг працювати в команді та співпрацювати з іншими членами команди. Завдяки правильному складу IT-команди можна забезпечити успішне виконання проєкту, якісний продукт та задоволеність клієнтів.

2.2. Складові IT-проєкту.

IT-проєкти стали важливим інструментом розробки програмного забезпечення, веб-додатків та інших технологічних рішень. Їх використовують для управління різноманітними видами розробок, від великих корпоративних проєктів до маленьких стартапів. Для ефективного керування IT-проєктом необхідно мати чітку структуру та організацію роботи, і це досягається за допомогою різних складових, таких як epic, story та task.

Epic - це великі категорії або групи робіт, які включають в себе великі функціональні або бізнесові вимоги, які можуть виконуватися протягом тривалого періоду часу. Epics є високорівневими ідеями або концепціями, які описують загальну мету або бажану функціональність проєкту. Вони можуть бути розподілені на менші фрагменти для розробки та впровадження. Epics допомагають командам зорієнтуватися в загальному напрямку розробки проєкту та встановлювати пріоритети [15].

Story - це короткі описи функціональності або вимог до продукту, виражені з точки зору кінцевого користувача. Вони є засобом збору вимог та співпраці між командами розробки та зацікавленими сторонами. Story мають бути короткими, зрозумілими, мінімальними та готовими до реалізації одиницями роботи. Вони допомагають командам розробки розуміти роботу на менші фрагменти, встановлювати пріоритети, оцінювати та відстежувати прогрес розробки.

Tasks - це конкретні завдання, які виконуються командами розробки для реалізації функціональності, описаної в сторі. Tasks є дрібними одиницями роботи, які можуть бути виконані в межах кількох годин або декількох днів. Вони можуть містити деталізовані кроки, необхідні ресурси, терміни виконання та відповідальних виконавців. Tasks є базовими одиницями в плануванні та організації роботи команди розробки, допомагаючи керувати виконанням проєкту на кожному етапі.

У процесі розробки IT-проєкту epics, story та tasks можуть змінюватися, оновлюватися та доповнюватися залежно від вимог та потреб проєкту. Важливо мати гнучкий підхід до управління цими складовими, щоб забезпечити успішну реалізацію проєкту.

Правильне управління epics, story та tasks включає кілька важливих етапів. По-перше, epics повинні бути чітко визначені, містити вимоги та бізнес-цілі проєкту. Вони повинні бути відповідні стратегії компанії та допомагати досягти більшої мети проєкту. Далі, story повинні бути деталізовані та зрозумілі, містити опис функціональності, вимоги до користувача, терміни виконання та оцінку зусиль. Вони також повинні бути пріоритезовані, щоб команда могла зосередитися на найважливіших завданнях.

Далі, на основі сторі команда розробки створює tasks, які є конкретними завданнями для виконання функціональності, описаної в сторі. Таски повинні бути деталізовані, містити всі необхідні кроки для виконання, відповідальних виконавців, оцінку зусиль та терміни виконання. Вони можуть бути організовані в канбан-дошках, проектних системах або інших інструментах управління проєктом, що допомагає відстежувати прогрес та впроваджувати зміни.

Для ефективного управління складовими IT-проєкту, такими як epics, story та tasks, рекомендується використовувати агільні методики розробки програмного забезпечення, такі як Scrum, Kanban або Lean. Вони надають структуру та рамки для організації роботи команди та сприяють високій продуктивності.

Epics використовуються для визначення стратегічних цілей проєкту та допомагають забезпечити зв'язок між бізнес-вимогами та розробкою. Вони можуть бути представлені у вигляді великих функціональних блоків або функціональних можливостей, які розробляються протягом певного періоду часу.

Story використовуються для деталізації епіків на більш дрібні фрагменти, які можуть бути виконані командою розробки протягом одного ітераційного циклу (наприклад, двотижневого спринту в методиці Scrum). Story містять опис функціональності, вимоги до користувача, терміни виконання та оцінку зусиль. Вони також можуть бути пріоритезовані в залежності від бізнес-вимог та технічних обмежень.

Tasks використовуються для конкретизації сторі на окремі завдання, які можуть бути виконані в рамках одного робочого дня або кількох годин. Tasks містять опис конкретного завдання, відповідального виконавця, оцінку зусиль та терміни виконання. Вони можуть бути організовані в канбан-дошках, що допомагає команді відстежувати прогрес виконання та визначати пріоритетність завдань.

Для успішного управління складовими ІТ-проєкту рекомендується використовувати наступні підходи та практики:

1. Визначення чітких бізнес-вимог: epics, story та tasks повинні бути засновані на чітко визначених бізнес-вимогах. Важливо забезпечити взаєморозуміння між командою розробки та бізнес-замовником, щоб уникнути непорозумінь та забезпечити відповідність розроблюваного продукту вимогам бізнесу.
2. Розбиття на менші частини: epics, story та tasks мають бути розбиті на менші, конкретні та досяжні одиниці роботи. Це допомагає команді легше розуміти обсяг роботи, виконувати її ефективно та відстежувати прогрес.
3. Пріоритезація: важливо визначити пріоритети між epics, story та tasks відповідно до стратегічних цілей проєкту та бізнес-вимог. Це допомагає забезпечити фокус на найважливіших завданнях та досягненні більшої вартості для бізнесу.
4. Комунікація та співпраця: ефективна комунікація та співпраця між усіма членами команди є ключовим фактором успіху проєкту.

Важливо забезпечити відкритий обмін інформацією, вирішення проблем та взаємодію між різними ролями в проєкті.

5. Використання візуалізації: використання візуальних засобів, таких як канбан-дошки, графіки Ганта або інші інструменти, допомагає візуалізувати прогрес роботи, виявляти можливі буттєві точки та робити корективи в планах, а також взаємодіяти з командою та зацікавленими сторонами.
6. Використання інструментів управління проєктами: використання спеціалізованих інструментів управління проєктами, таких як Jira, Trello, Asana або інші, допомагає відстежувати, оцінювати та керувати роботою над епіками, сторі та тасками. Ці інструменти надають зручний інтерфейс для керування задачами, визначення термінів виконання, розподілу ресурсів та відстеження прогресу.
7. Гнучкість та адаптабельність: IT-проєкти можуть зазнавати змін вимог, пріоритетів та ресурсів. Важливо мати гнучкість та бути готовим до змін, адаптуватися до нових умов та робити корективи в планах проєкту, epics, story та tasks відповідно до поточної ситуації.
8. Тестування та якість: важливим аспектом IT-проєктів є тестування та забезпечення якості розроблюваного продукту. Варто враховувати включення тестування в робочий процес, визначення вимог до якості, використання автоматизованих тестів та забезпечення високої якості продукту перед його впровадженням.
9. Моніторинг та звітність: важливо встановити механізми моніторингу та звітності про стан epics, story та tasks в проєкті. Це дозволяє відстежувати прогрес роботи, виявляти можливі ризики та забезпечувати своєчасне повідомлення зацікавлених сторін про стан проєкту.

Складові IT-проєкту, такі як epics, story та tasks, відіграють важливу роль у процесі розробки програмного забезпечення. Їх правильне визначення,

організація та керування є вирішальними для успішної реалізації проєкту. Використання агільних методів, використання спеціалізованих інструментів управління проєктами, комунікація та співпраця в команді, тестування та забезпечення якості, моніторинг та звітність, а також постійне вдосконалення, допомагають досягти успіху в розробці IT-проєктів [16].

2.3. Модель рішення задачі планування.

IT-проєкти являють собою важливу складову сучасного світу повсякденного життя. Багато хто залучений до роботи в IT-сфері та з кожним роком попит на місце в IT збільшується. Бажаючих спеціалістів на одне місце іноді перевищує 200 чоловік на 1 посаду. Усі намагаються проходити курси задля підвищення кваліфікації та навичок, щоб не втратити своє місце.

Для більшого розуміння формування команди можна зробити математичні розрахунки.

T{T1, T2, T3, T4} – час необхідний для повного виконання проєкту, розподілений по виробничим напрямам.

N_d – кількість робочих днів.

N_h – кількість годин.

N_w – кількість вихідних днів.

N_c – кількість календарних днів.

D_s - дата початку проєкту.

D_f – дата закінчення проєкту.

H – константа = 8 годинам на день.

Q{Q1, Q2, Q3, Q4} – необхідна кількість працівників за виробничим напрямом.

$Q1 = T1/(Df – Ds)*H$.

По-перше, зробимо розрахунки часу роботи для простого проєкту:
Для розрахунку часу необхідного для виконання проєкту потрібно спочатку перевести дати в години, враховуючи вихідні дні. Для цього можна скористатися формулою:

$N_d = (D_f – D_s) * (5/7)$.

$N_h = N_d * H$.

Наприклад, для проєкту з Ds=10.05.2023 та Df=19.11.2023, кількість робочих днів можна розрахувати так:

$N_c = (19.11.2023 - 10.05.2023) = 193$.

$N_w = (193 / 7) * 2 = 54$, бо маємо 2 вихідні дні на тиждень.

$N_d = (193 * 5 / 7) - 54 = 89$.

$N_h = 89 * 8 = 712$.

1. Розрахунок часу роботи Backend-розробника:

Нехай час виконання Backend-розробника на проєкт дорівнює $T_1 = 2160$ годин.

$Q_1 = T_1 / (D_f – D_s) * H = 2160 / 712 = 3.04$.

Отже, необхідна кількість Backend-розробників - це близько 3 осіб.

2. Розрахунок часу роботи Frontend-розробника:

Нехай час виконання Frontend-розробника на проєкт дорівнює $T_2 = 1440$ годин.

$Q_2 = T_2 / (D_f – D_s) * H = 1440 / 712 = 2.03$.

Отже, необхідна кількість Frontend-розробників - це близько 2 осіб.

3. Розрахунок часу роботи QA-тестувальника:

Нехай час виконання QA-тестувальника на проєкт дорівнює $T_3 = 1080$ годин.

$Q_3 = T_3 / (D_f – D_s) * H = 1080 / 712 = 1.52$.

Отже, необхідна кількість QA-тестувальників - це близько 2 осіб.

4. Розрахунок часу роботи Designer:

Нехай час виконання дизайнера на проєкт дорівнює $T_4 = 720$ годин.

$Q_4 = T_4 / (D_f – D_s) * H = 720 / 712 = 1.01$

Отже, необхідна кількість дизайнерів - це близько 1 особа.

$T\{T_1, T_2, T_3, T_4\} = 2160 + 1440 + 1080 + 720 = 5400$.

$Q\{Q_1, Q_2, Q_3, Q_4\} = 3 + 2 + 2 + 1 = 8$.

У результаті отримуємо, що для виконання проєкту строком у півроку нам знадобиться 3 backend-розробника, 2 frontend-розробника, 2 QA-тестувальника, 1 designer.

По-друге, зробимо розрахунки часу роботи для проєкту середньої складності:

Для розрахунку часу необхідного для виконання проєкту потрібно спочатку перевести дати в години, враховуючи вихідні дні. Для цього можна скористатися формулою:

$N_d = (D_f – D_s) * (5/7)$.

$N_h = N_d * H$.

Наприклад, для проєкту з Ds=01.06.2023 та Df=05.06.2024, кількість робочих днів можна розрахувати так:

$N_c = (05.06.2024 - 01.06.2023) = 370$.

$N_w = (370 / 7) * 2 = 106$, бо маємо 2 вихідні дні на тиждень.

$N_d = (370 * 5 / 7) - 106 = 159$.

$N_h = 159 * 8 = 1272$.

1. Розрахунок часу роботи Backend-розробника:

Нехай час виконання Backend-розробника на проєкт дорівнює T1 = 4320 годин.

$Q_1 = T_1 / (D_f – D_s) * H = 4320 / 1272 = 3.4$

Отже, необхідна кількість Backend-розробників - це близько 4 осіб.

2. Розрахунок часу роботи Frontend-розробника:

Нехай час виконання Frontend-розробника на проєкт дорівнює T_2 = 3600 годин.

$Q_2 = T_2 / (D_f – D_s) * H = 3600 / 1272 = 2.83$

Отже, необхідна кількість Frontend-розробників - це близько 3 осіб.

3. Розрахунок часу роботи QA-тестувальника:

Нехай час виконання QA-тестувальника на проєкт дорівнює $T_3 = 2160$ год

$Q_3 = T_3 / (D_f – D_s) * H = 2160 / 1272 = 1.7$

Отже, необхідна кількість QA-тестувальників - це близько 2 осіб.

4. Розрахунок часу роботи Designer:

Нехай час виконання дизайнера на проєкт дорівнює $T_4 = 1440$ годин.

$Q_4 = T_4 / (D_f – D_s) * H = 1440 / 1272 = 1.13$

Отже, необхідна кількість дизайнерів - це близько 2 осіб.

$T\{T_1, T_2, T_3, T_4\} = 4320 + 3600 + 2160 + 1440 = 11520.$

$Q\{Q_1, Q_2, Q_3, Q_4\} = 4 + 3 + 2 + 2 = 11.$

У результаті отримуємо, що для виконання проєкту строком у рік нам знадобиться 4 backend-розробника, 3 frontend-розробника, 2 QA-тестувальника, 2 designers.

По-третє, зробимо розрахунки часу роботи для складного проєкту:

Для розрахунку часу необхідного для виконання проєкту потрібно спочатку перевести дати в години, враховуючи вихідні дні. Для цього можна скористатися формулою:

$N_d = (D_f – D_s) * (5/7).$

$N_h = N_d * H.$

Наприклад, для проєкту з Ds=15.07.2023 та Df=22.07.2025, кількість робочих днів можна розрахувати так:

$N_c = (22.07.2025 - 15.07.2023) = 745.$

$N_w = (745 / 7) * 2 = 212$, бо маємо 2 вихідні дні на тиждень.

$N_d = (745 * 5 / 7) - 212 = 264.$

$N_h = 264 * 8 = 2112.$

1. Розрахунок часу роботи Backend-розробника:

Нехай час виконання Backend-розробника на проєкт дорівнює $T_1 = 9360$ годин.

$Q_1 = T_1 / (D_f – D_s) * H = 9360 / 2112 = 4,43.$

Отже, необхідна кількість Backend-розробників - це близько 5 осіб.

2. Розрахунок часу роботи Frontend-розробника:

Нехай час виконання Frontend-розробника на проєкт дорівнює $T_2 = 7920$ годин.

$Q_2 = T_2 / (D_f – D_s) * H = 7920 / 2112 = 3,75.$

Отже, необхідна кількість Frontend-розробників - це близько 4 осіб.

3. Розрахунок часу роботи QA-тестувальника:

Нехай час виконання QA-тестувальника на проєкт дорівнює $T_3 = 7320$ год.

$Q_3 = T_3 / (D_f – D_s) * H = 7320 / 2112 = 3,47.$

Отже, необхідна кількість QA-тестувальників - це близько 4 осіб.

4.Розрахунок часу роботи Designer:

Нехай час виконання дизайнера на проєкт дорівнює $T_4 = 5200$ годин.

$Q_4 = T_4 / (D_f – D_s) * H = 5200 / 2112 = 2,46.$

Отже, необхідна кількість дизайнерів - це близько 3 осіб.

$T\{T_1, T_2, T_3, T_4\} = 9360 + 7920 + 4320 + 3600 = 25200.$

$Q\{Q_1, Q_2, Q_3, Q_4\} = 5 + 4 + 2 + 2 = 13.$

У результаті отримуємо, що для виконання проєкту строком у рік нам знадобиться 5 backend-розробників, 4 frontend-розробника, 2 QA-тестувальника, 2 designers.

Також варто враховувати, що в кожному з проєктів буде присутній PM та BA, де у кожного з них своя кількість робочих годин.

Ми розглянули три проєкти з різними рівнями складності, розрахували кількість годин кожного зі спеціалістів та отримали загальну кількість членів команди та витрачений час на проєкт.

Висновки за розділом 2

У даному розділі було розглянуто різні сфери застосування IT-технологій та було зазначено, що їх застосовують у різноманітних галузях. Також ми продивилися та проаналізували типи команд та їх склад для різної складності проєктів. Ще одним з важливих пунктів було те, що ми зробили розрахунки та показали, як саме будуть робитися розрахунки.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ ПЛАНУВАННЯ ІТ-ПРОЄКТІВ

3.1. Використані мови програмування та їх переваги.

Для моєї розробки було використано дві мови програмування: JavaScript (JS) та HTML. JS і HTML - це дві різні мови програмування, які використовуються для розробки веб-додатків.

HTML (HyperText Markup Language) - це мова розмітки, яка використовується для створення веб-сторінок. Вона дозволяє встановлювати структуру документа, включаючи заголовки, абзаци, списки, таблиці, форми та інші елементи, які допомагають організувати контент на сторінці. HTML забезпечує зручний інтерфейс для збереження та передачі інформації з сервера на клієнт.

JS - це мова програмування, яка використовується для створення динамічних веб-додатків та взаємодії з користувачем. JS дозволяє змінювати зовнішній вигляд сторінки, додавати анімацію, валідувати форми, взаємодіяти зі сторонніми API та багато іншого. JS є однією з найбільш поширених мов програмування в світі.

Переваги використання JS і HTML у розробці веб-додатків:

1. Легкість використання: HTML і JS дуже легкі для вивчення, і майже кожен може почати розробляти веб-додатки.
2. Кросплатформенність: HTML і JS можуть працювати на будь-якому комп'ютері з будь-якою операційною системою та на будь-якому браузері.
3. Багатофункціональність: HTML і JS дозволяють створювати динамічні та інтерактивні веб-сторінки, що дозволяє розробникам створювати складні веб-додатки.

4. Підтримка: HTML і JS підтримуються більшістю браузерів та мають велику кількість фреймворків, бібліотек та інструментів розробки, що значно спрощує роботу розробників.
5. Взаємодія з користувачем: JS дозволяє створювати веб-додатки, які реагують на дії користувача без необхідності відправляти запити на сервер. Це дозволяє створювати більш інтерактивні та відзивчиві веб-додатки.
6. Масштабованість: HTML і JS дозволяють створювати веб-додатки будь-якої складності та масштабу, що робить їх ідеальними для розробки великих проектів.
7. Підтримка різних платформ: HTML та JS підтримують розробку веб-додатків для різних платформ, таких як настільні комп'ютери, мобільні пристрої та інші.

Загалом, використання HTML і JS є важливою складовою розробки веб-додатків. Вони дозволяють розробникам створювати динамічні та ефективні веб-додатки з високою ступеню функціональності та безпеки. Більшість веб-сайтів та веб-додатків використовують HTML та JS як основні мови програмування, тому їх вивчення є важливим для будь-якого розробника веб-додатків.

3.2 Опис роботи сайту.

Метою моєї розробки є онлайн-сервіс, що надає користувачам зручний і швидкий спосіб підібрати необхідний та оптимальний склад команди для реалізації IT-проект.

Розглянемо фазу проєктування, яка передує старту роботи команди над IT-проєктом:

1. Клієнт обговорює мету проєкту з SM та PM.
2. Створення технічної документації проєкту.
3. Декомпозиція проєкту на epics/stories/tasks/sub-tasks.

4.Оцінка скоупу робіт з розбиттям на спеціалізації.

5. Оцінка складу команди, необхідної для реалізації проєкту у вказаний термін.

Основна робота сайту полягає в тому, що користувачі можуть ввести параметри проєкту, такі як терміни виконання, потрібні навички, основні вимоги до розробки, і отримати рекомендації щодо складу команди, яка може ефективно виконати проєкт з урахуванням заданих параметрів, що в сукупності аналізуються та видають результат.

Сервіс може запропонувати різні варіанти складу команди, які можуть включати різні ролі, такі як менеджер проекту, розробник, дизайнер, тестувальник тощо.

Крім підбору команди, сайт також може надати користувачам інформацію щодо розрахунку бюджету проєкту. Користувачі можуть ввести потрібні витрати на різні ресурси, такі як зарплата команди, оренда офісу, програмне забезпечення, хостинг тощо, і отримати розрахунок загального бюджету проєкту.

Для забезпечення безпеки та конфіденційності даних, сервіс може використовувати захист інформації, такий як шифрування, щоб захистити дані користувачів від несанкціонованого доступу.

Сайт може бути корисним інструментом для бізнесу, стартапів та фрілансерів, які шукають спосіб ефективно організувати та виконати IT-проєкт. Розробка може допомогти користувачам знайти потрібну команду, що відповідає їх вимогам та бюджету, та розрахувати загальний бюджет проєкту, що дозволить користувачам раціонально розподілити ресурси та планувати проєкт на майбутнє.

Окрім цього, сайт може мати інші корисні функції, такі як збереження списку проєктів, що користувачі планують реалізувати, можливість відстеження прогресу проєкту та звітності про затрати.

Для ефективної роботи сервісу може використовуватися різноманітний програмний код, такий як Python, Java, JS та інші, але в нашому випадку ми

вище зазначили, що використовували мови програмування JS та HTML та вказали їх переваги.

Створення сайту вимагає ретельного планування та розробки, з урахуванням потреб користувачів та можливих функцій, що можуть бути вимогливі. Проте, при правильному виконанні, такий сайт може бути корисним інструментом для різноманітних направлень. Також сайт можна доповнювати додатковим інструментарієм та розвивати в майбутньому.

3.3 Тестування сервісу для планування IT-проєктів

Данний сайт було реалізовано на мові програмування JS та на мові розмітки HTML&CSS, та дана програма володіє широким набором можливостей підбору команди : головна сторінка IT-Service , на якій зображено логотип, та клавіши для переходу між сторінками «Авторізація», котра потрібна для входу на головну сторінку підбору команди, та «Головна», яка поверне вас на головну сторінку сайту (рис 3.1).

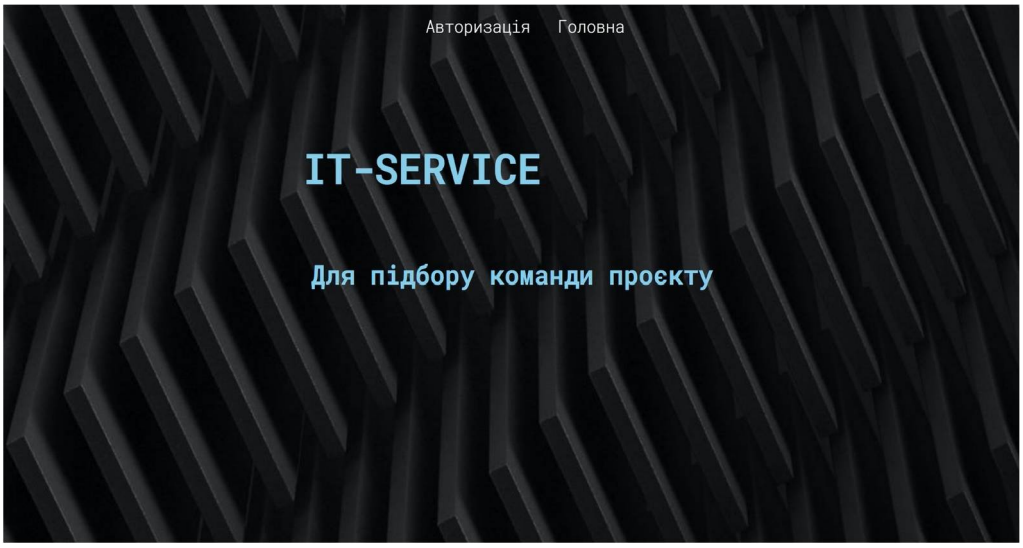

Рис 3.1 – Головна сторінка

На наступній сторінці «Авторизація», для виконання авторизації на веб-сайті, потрібно ввести особисті дані користувача. Ввести електрону пошту та пароль в поле ввода паролю. Потрібно звернути увагу на те, що пароль чутливий до регістру (залежить від використання великих і малих літер), тому

потрібно переконатися, що вводите його правильно. Після чого, натискаємо на кнопку "Вхід": Після введення своїх облікових даних і пароля натисніть на відповідну кнопку, щоб виконати авторизацію. Система перевірятиме Вашу інформацію і переконається, що вона збігається з даними вашого акаунта. Якщо надані облікові дані правильні, Ви будете авторизовані й отримаєте доступ до свого облікового запису (рис 3.2).

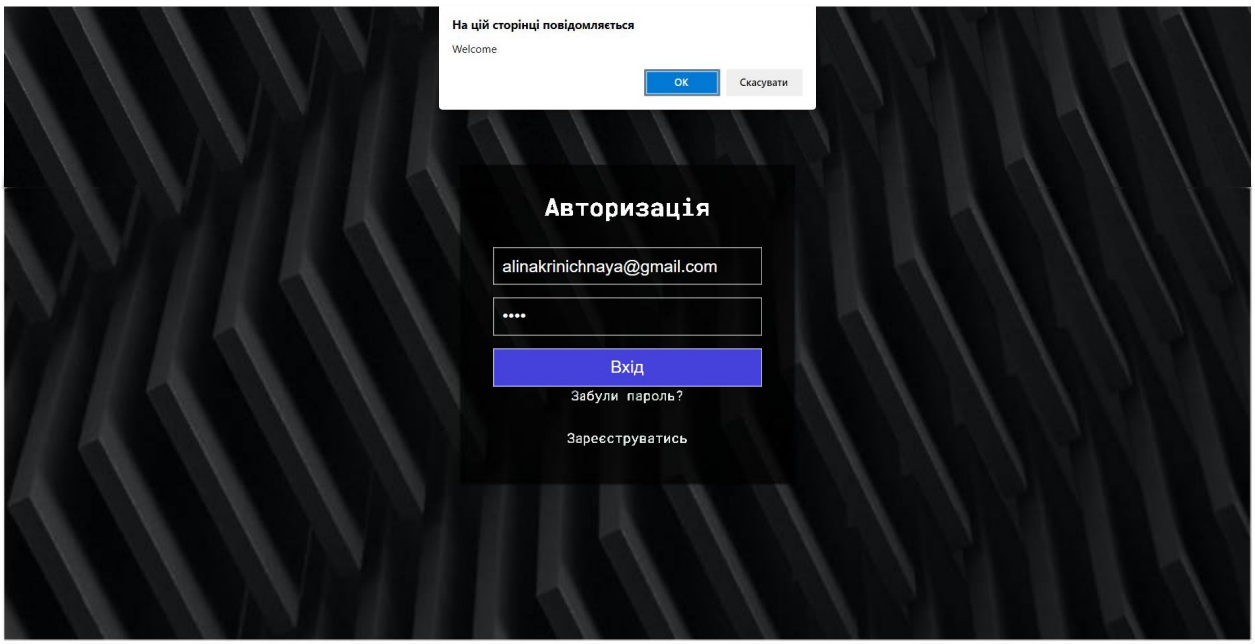

Рис 3.2 – Сторінка авторизації

Якщо під час авторизації виникає помилка через неправильний логін або пароль, зазвичай з'являється повідомлення, що інформує користувача про проблему. Опис такої помилки може мати такий вигляд (рис 3.3)

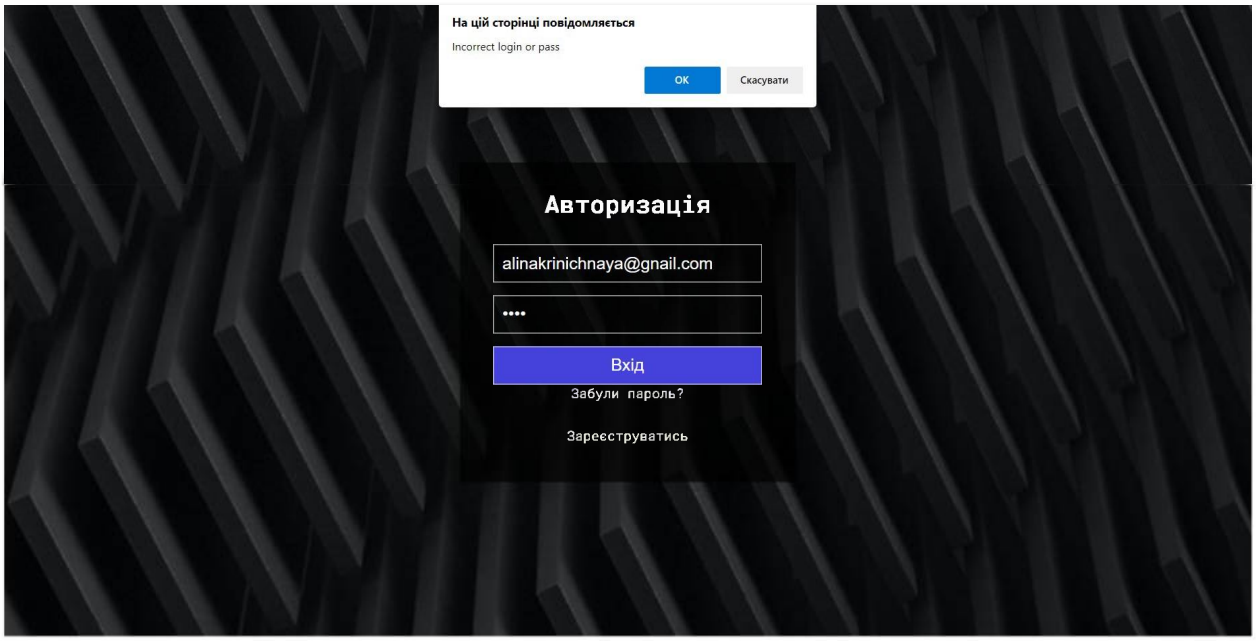

Рис 3.3 – Сторінка авторизації (з неправильними логіном або паролем)

Це повідомлення вказує на те, що введені вами дані для входу не збігаються з інформацією в системі. У разі неправильного логіна або пароля рекомендується виконати такі дії:

1. Перевірте правильність введення даних: Переконайтеся, що ви правильно ввели свій логін (електронну пошту) і пароль. Зверніть увагу на використання великих і малих літер, а також на можливу наявність пробілів або інших спеціальних символів.
2. Перевірте розкладку клавіатури: Іноді помилка авторизації може бути пов'язана з неправильною розкладкою клавіатури. Переконайтеся, що ви використовуєте правильну розкладку і вводите дані на відповідних клавішах.

Якщо ви забули пароль для входу на веб-сайт, зазвичай доступна функція скидання пароля, яка допомагає відновити доступ до вашого облікового запису. Опис процесу скидання пароля може виглядати наступним чином:

1. Знайдіть посилання на скидання пароля: На сторінці авторизації присутнє посилання або кнопка з написом "Забули пароль?".Вона знаходиться під полем для введення пароля (рис 3.4).

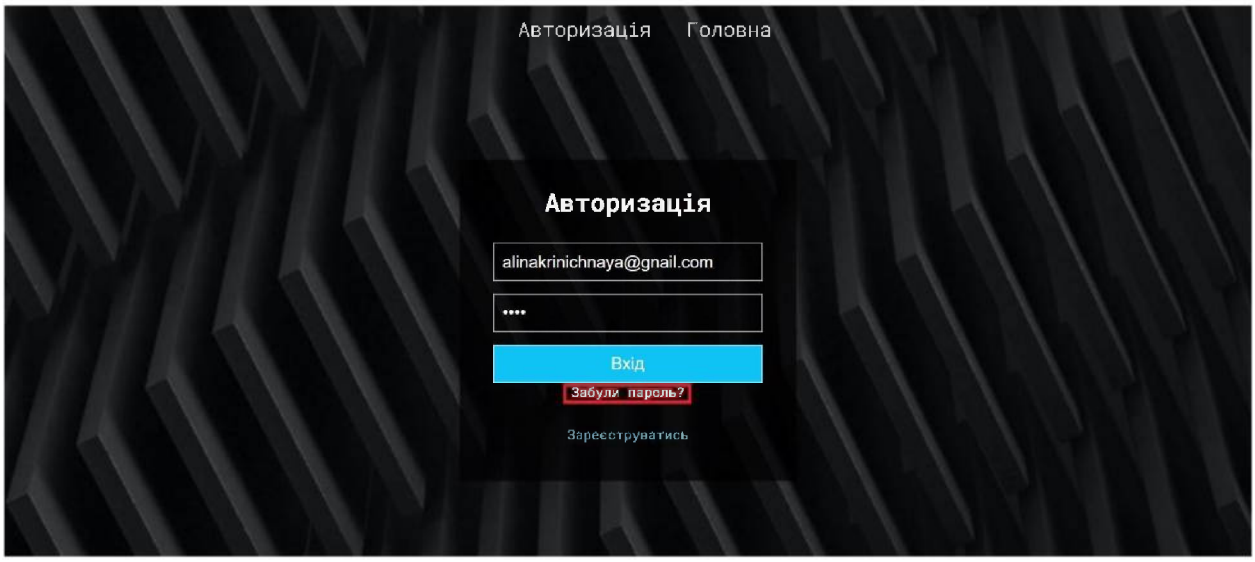

Рис 3.4 – Головна сторінка

2. Натисніть на посилання "Забули пароль?": Клацніть на посилання, щоб перейти на сторінку скидання пароля.
3. Введіть свою адресу електронної пошти: На сторінці скидання пароля буде запропоновано ввести адресу електронної пошти, пов'язану з вашим акаунтом. Переконайтеся, що ви вводите ту адресу, яку використовували під час реєстрації на сайті.
4. Натисніть на кнопку "Відновити".Після введення адреси електронної пошти натисніть на відповідну кнопку, щоб запросити скидання пароля (рис 3.5).

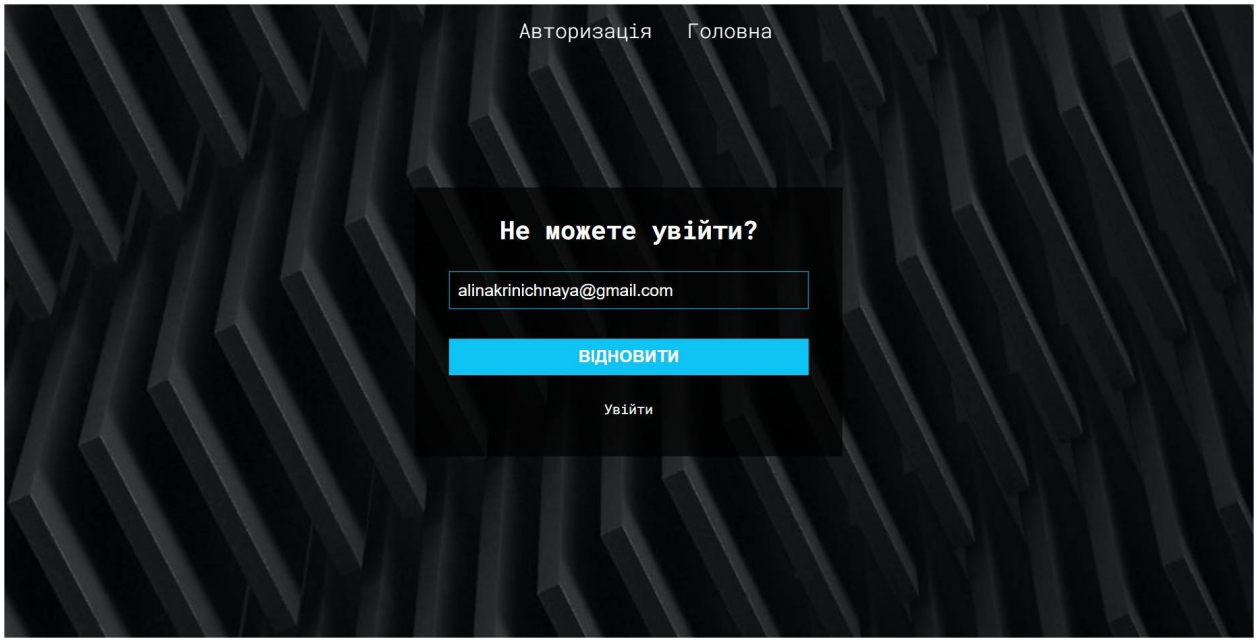

Рис 3.5 – Сторінка відновлення пароля

Після чого вас буде переведно на сторінку з подальшою інструкцією про відновлення пароля.

Якщо ви ще не зареєстровані на сайті, то на сторінці авторизації натисніть на кнопку зареєструватися(рис 3.6):

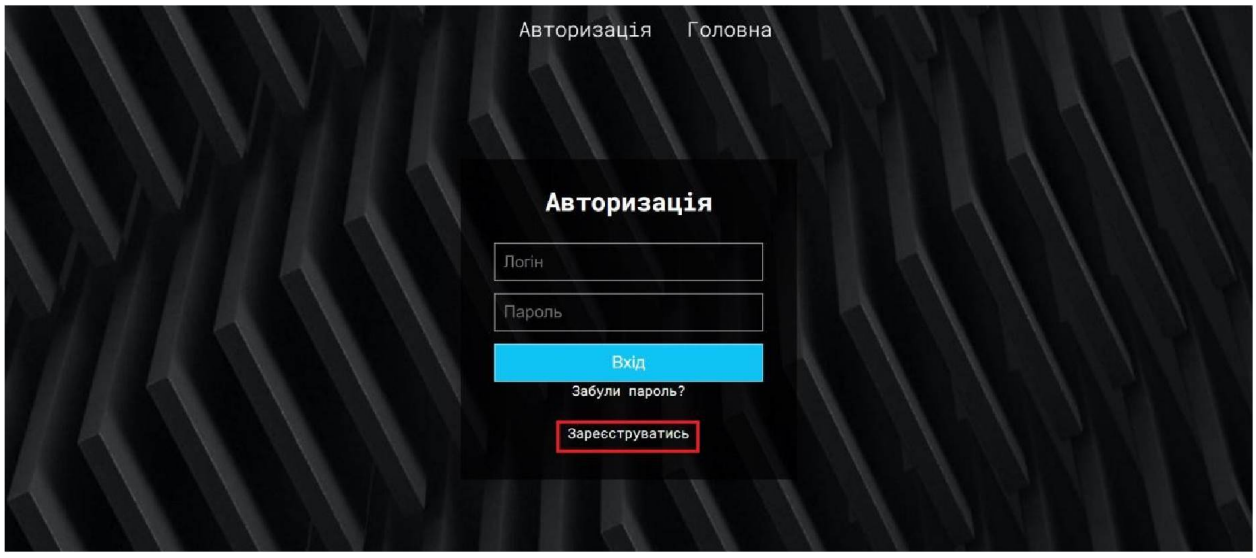

Рис 3.6 – Сторінка авторизації (кнопка зареєструватись)

1. Заповніть реєстраційну форму: На сторінці реєстрації пропонується реєстраційна форма, в яку потрібно ввести необхідні дані. Це включає

поля для введення вашого адреси електронної пошти, пароля(рис 3.7).

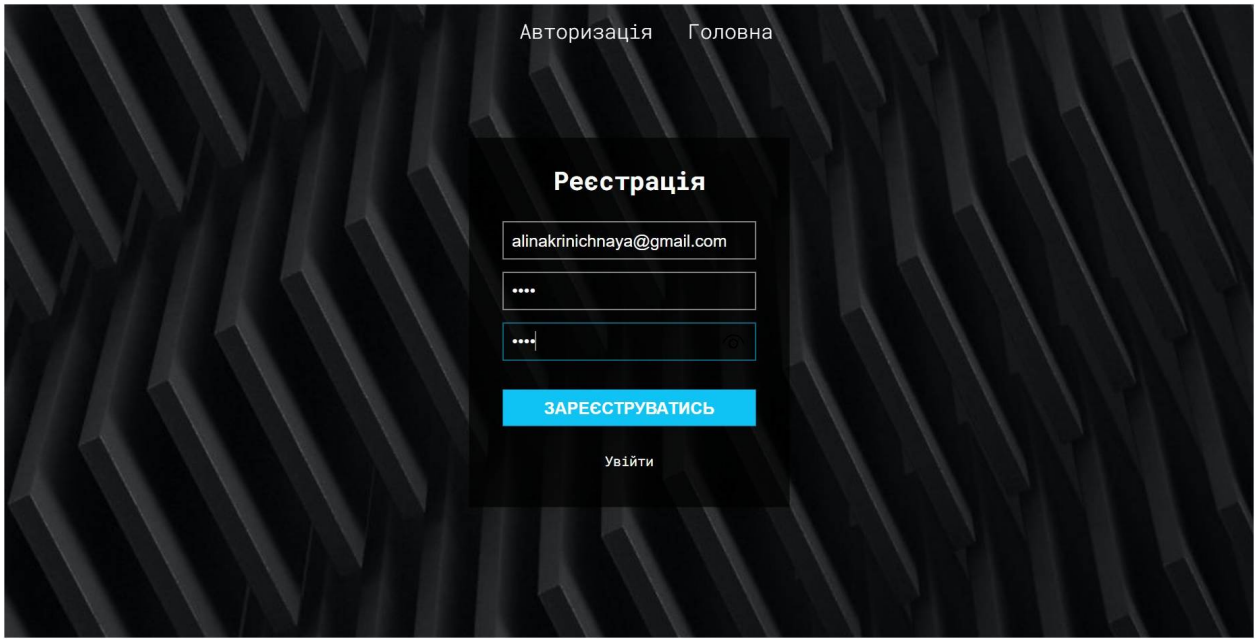

Рис 3.7 – Сторінка Реєстрації

2. Введіть дані: Внесіть запитувану інформацію у відповідні поля.
3. Створіть пароль: Придумайте надійний пароль для вашого облікового запису. Пароль має містити комбінацію букв, цифр і спеціальних символів, щоб забезпечити безпеку вашого облікового запису. Переконайтеся, що запам'ятайте свій пароль або збережіть його в надійному місці.

Після завершення реєстрації ви будете перенаправлені на головну сторінку веб-сайту. Тепер у вас є обліковий запис на сайті, і ви можете використовувати свій логін і пароль для входу в систему.

Перейдемо до головної сторінки сайту для підбору команди

Сайт надає можливість надавання команди за сроком роботи над проєктом. Користувачі можуть редагувати та вказати свої уподобання й отримати список відповідних фахівців(рис 3.8 та рис 3.9).

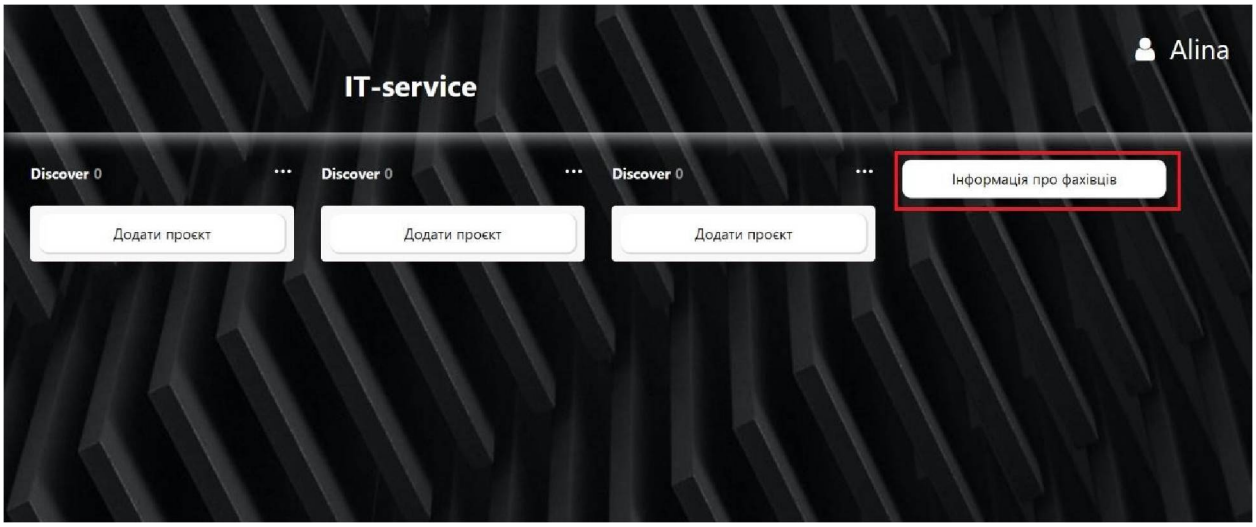

Рис 3.8 – Головна сторінка для підбору команди

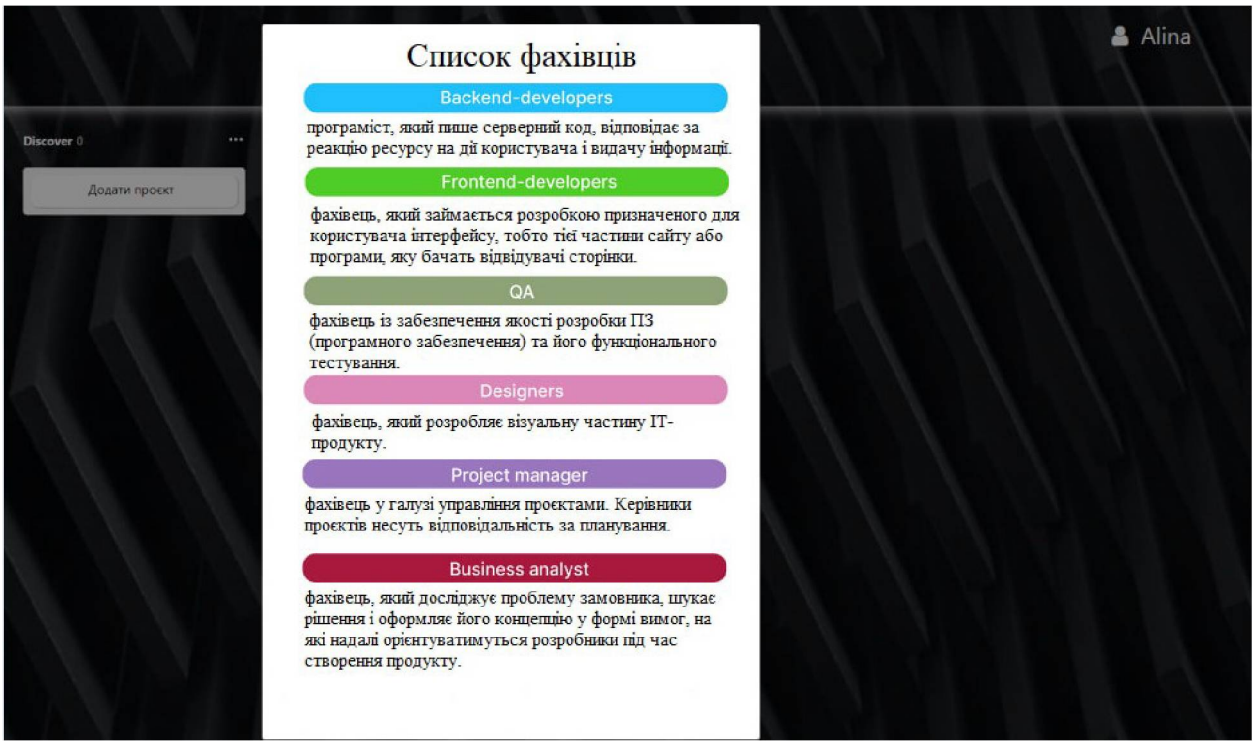

Рис 3.9 – Інформація про фахівців

Навігація до розділу створення проєкту: Після входу в систему натисніть на кнопку, що позначає "Додати проєкт"(рис 3.10).

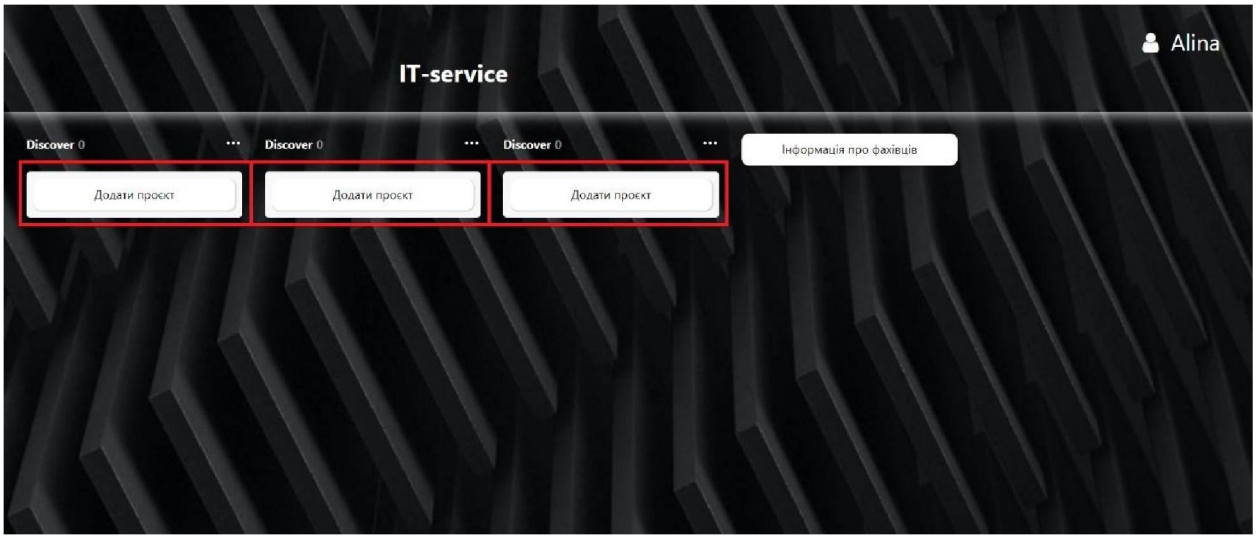

Рис 3.10 – Інформація про фахівців

Потрібно заповнити інформацію про проєкт: На сторінці створення проєкту вам запропоновано заповнити інформацію про проєкт. Це включає назву проєкту, опис, цілі, вимоги до проєкту та вибору дати початку та дати кінця проєкту(рис 3.11).

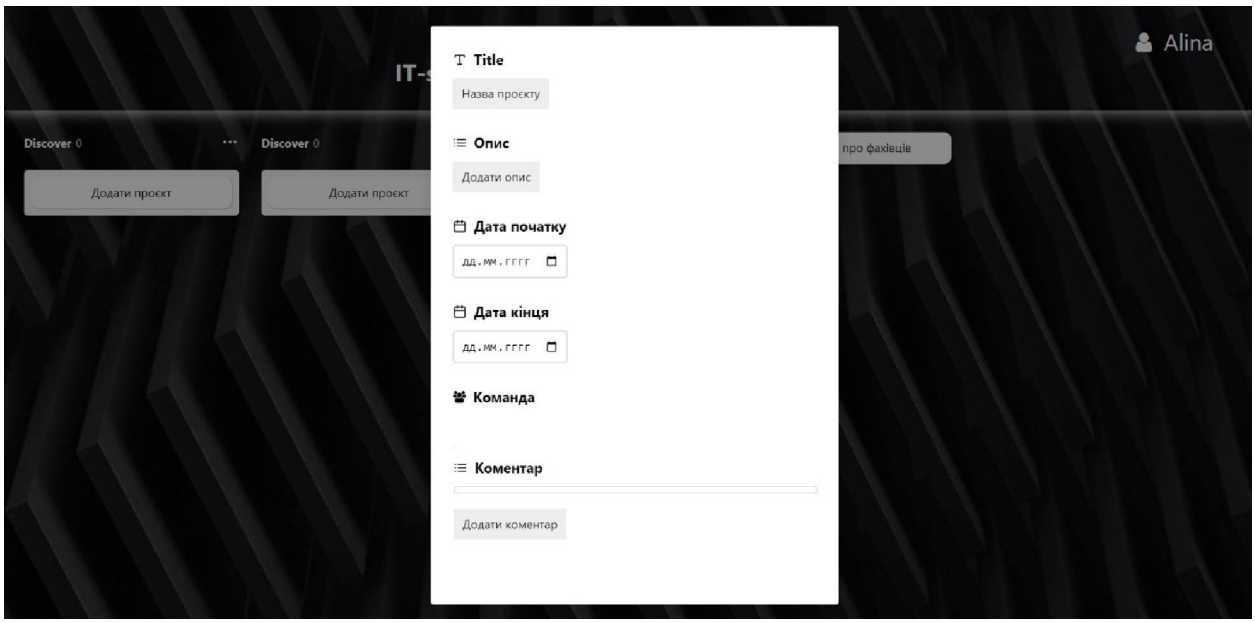

Рис 3.11 – Головна сторінка для заповнення проєкту

Створимо для початку простий проєкт на півроку.Визначемо мету проєкту: розкриємо, що саме ми хочемо досягти за півроку. Наприклад, це буде мобільний додаток «E-Health»(рис 3.12).

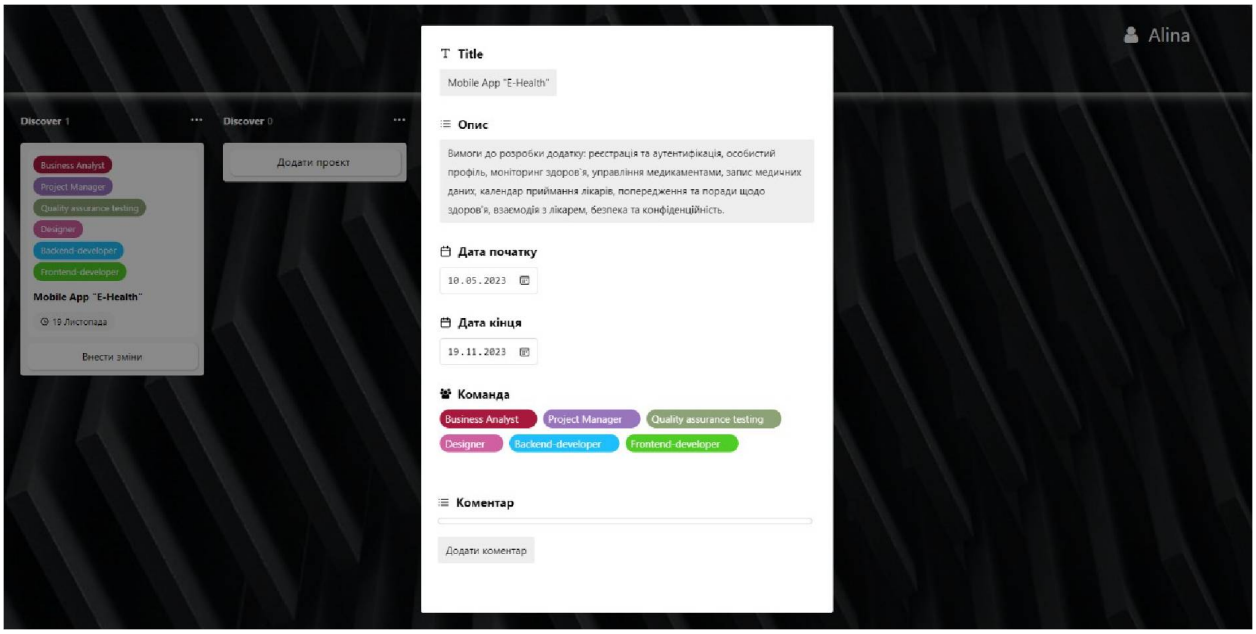

Рис 3.12 – Головна сторінка заповненого проєкту «E-Health»

Після того як ми вибрали дату початку та дату кінця , система підібрала нам команду та скільки фахівців буде працювати над цим проєктом, для відображення ми натискаємо на кнопку "...", то відображається інформація про «Склад команди» та «Додати проєкт» (Рис 3.13 та Рис 3.14).

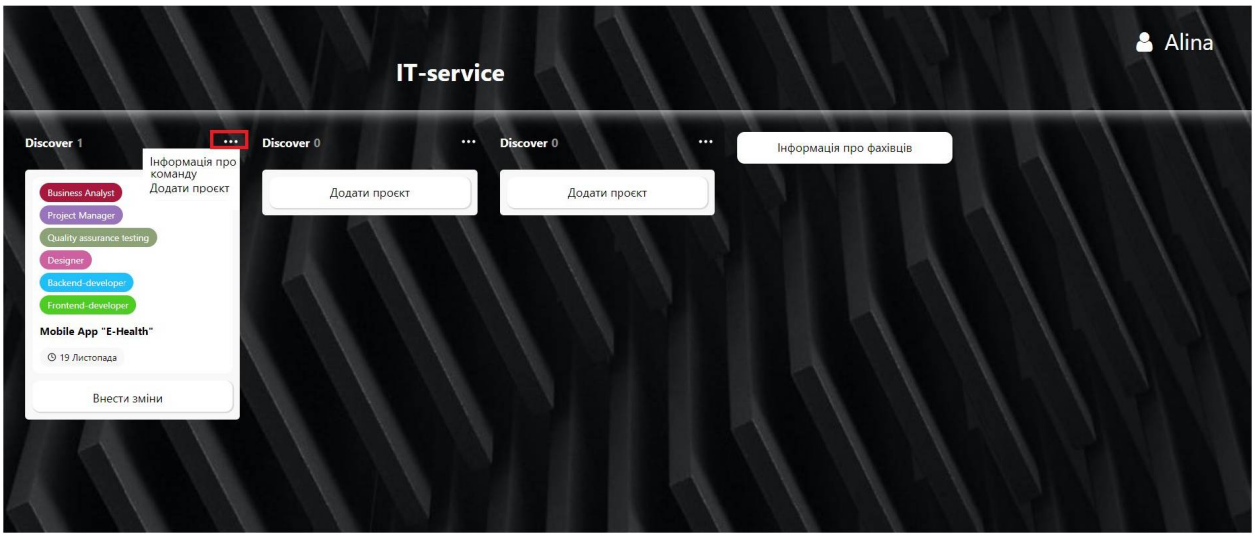

Рис 3.13 – Перегляд інформації після натискання клавіші «…»

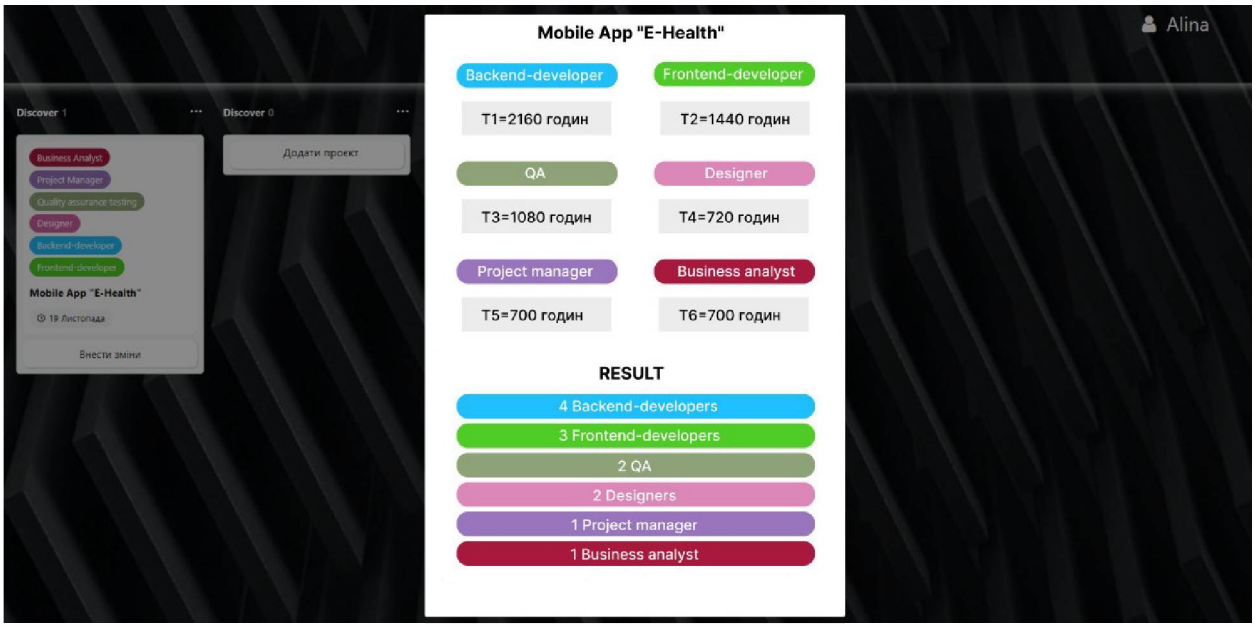

Рис 3.14 – Інформація команди над проєктом «E-Health»

Створимо тепер проєкт середньої складності. Відобразимо, що має бути в проєкті виконано за рік. Створимо веб-сайт для замовлення квітів «Flowers.ua»(рис 3.15).

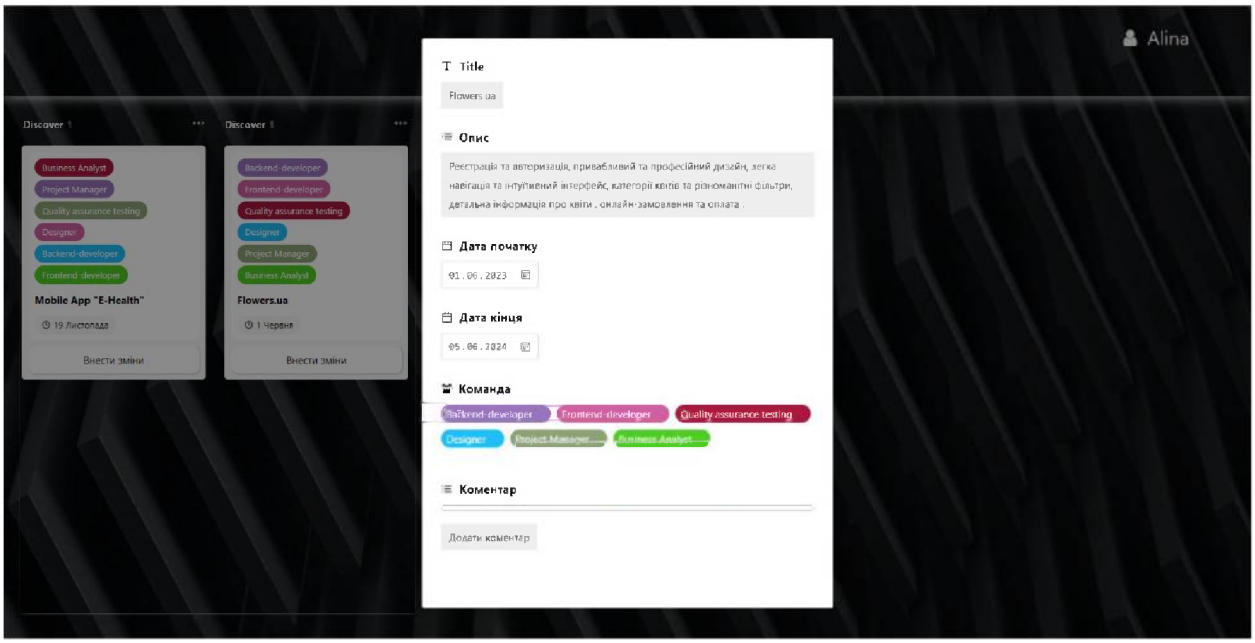

Рис 3.15 – Головна сторінка проєкту «Flowers.ua»

Після вибору дати початку та закінчення проекту, система автоматично підібрала нам команду та визначила кількість фахівців, які будуть працювати

над проєктом(рис 3.16).

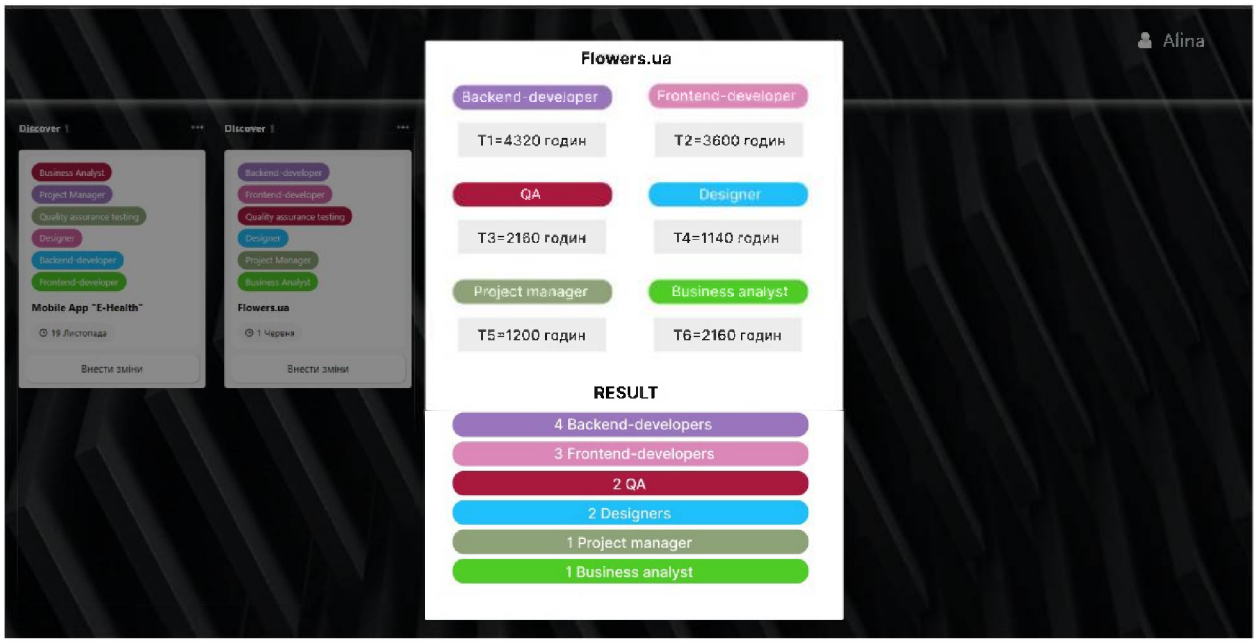

Рис 3.16 – Інформація команди над проєктом «Flowers.ua»

Створимо теперь складний проєкт. Відобразимо що має бути у проєкті виконано за 2 роки. Створимо програму «Artificial intelligence image recognition»(рис 3.17).

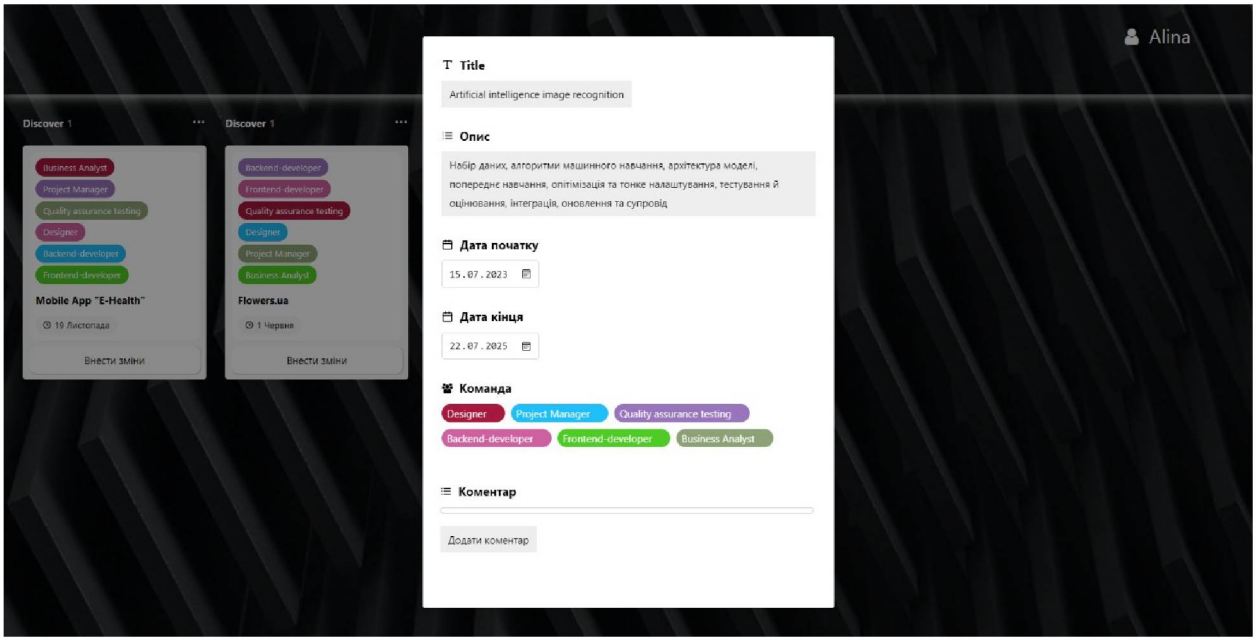

Рис 3.17 – Головна сторінка проєкту «Artificial intelligence image recognition»

Після визначення дати початку та закінчення проекту, наша система

автоматично скомпонувала команду та визначила оптимальну кількість фахівців, які будуть займатися реалізацією цього проекту(рис 3.18).

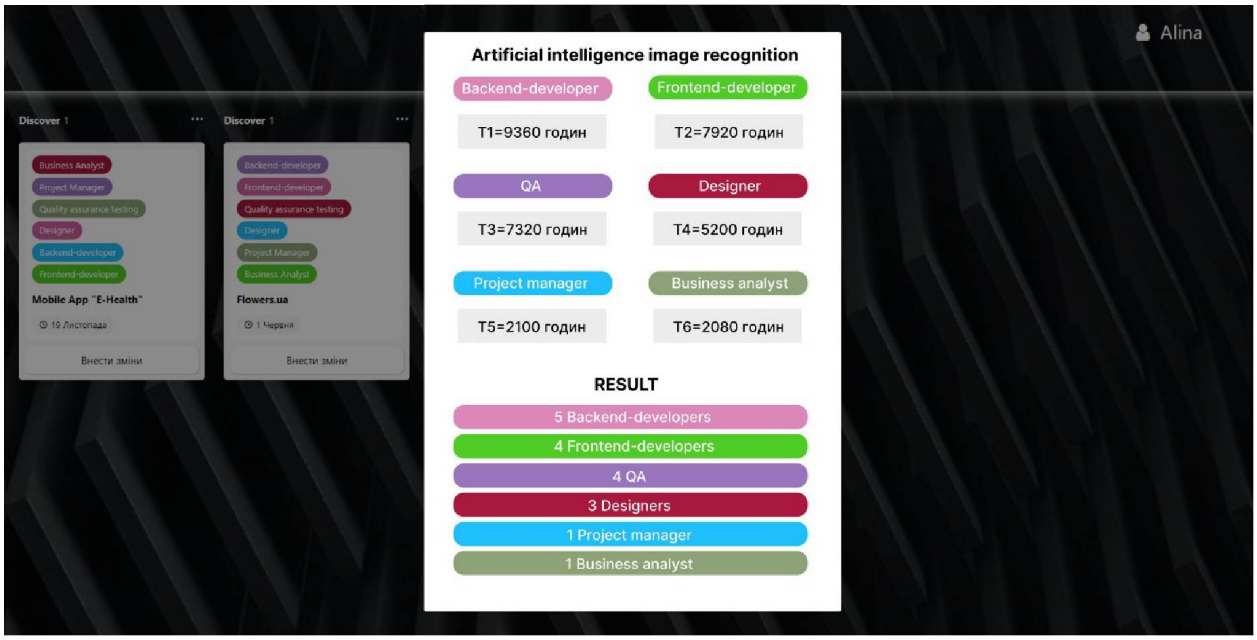

Рис 3.18 – Інформація команди над проєктом «Artificial intelligence image recognition»

На рисунку 3.19, уже можна побачити головну сторінку с проєктами , які ми розглядали раніше.

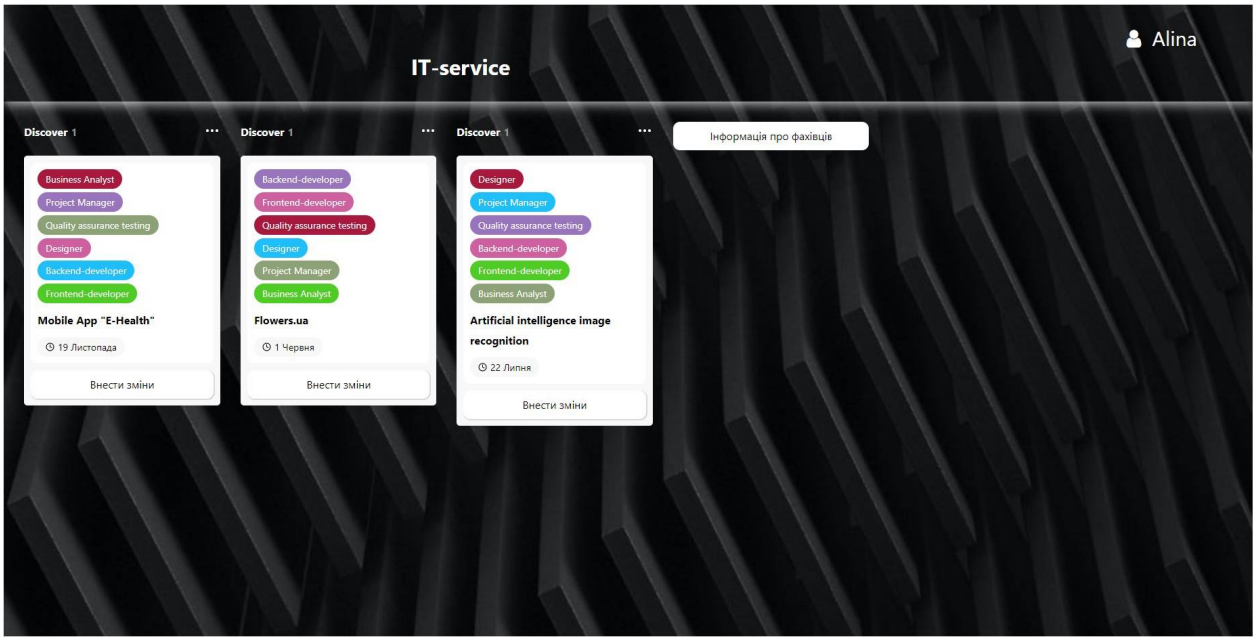

Рис 3.19 – Головна сторінка з проєктами

Висновки за розділом 3

У даному розділі було створено веб-сайт для підбору команди є потужним інструментом, що допомагає знаходити та формувати ефективні команди для виконання проектів. Завдяки широкому функціоналу цього сайту, ви можете зручно встановлювати критерії відбору команди, включаючи дату початку та закінчення проекту.

У цілому, веб-сайт для підбору команди є незамінним інструментом для успішного виконання проєктів. Він допомагає знайти правильну кількість фахівців, організувати робочий процес та забезпечити успішну реалізацію проєкту. З його допомогою Ви можете зосередитися на важливих завданнях і досягти бажаних результатів.

ВИСНОВКИ

У ході дослідження було розглянуто питання планування ІТ-проєктів та розроблено комп'ютерну систему, яка дозволяє автоматизувати процес планування та контролювання виконання ІТ-проєктів. Було проведено аналіз існуючих рішень та визначено основні вимоги до системи планування ІТ-проєктів.

Для розробки системи були використані сучасні технології програмування. Розроблена система дозволяє автоматизувати процеси контролювання виконання завдань, відстежувати прогрес проєкту та підбирати склад спеціалістів.

У результаті дослідження було визначено, що застосування комп'ютерної системи планування ІТ-проєктів дозволяє підвищити ефективність управління проєктами та зменшити ризики неуспішного завершення проекту. Розроблена система може бути використана в компаніях, які займаються розробкою програмного забезпечення та інших ІТ-проєктах.

Отже, можна стверджувати, що розроблена комп'ютерна система планування ІТ-проєктів є актуальним та перспективним рішенням для автоматизації управління ІТ-проєктами, що дозволяє збільшити ефективність процесів та зменшити ризики невиконання проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition. Newtown Square, PA: Project Management Institute, 2021. Print.
2. Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley, 2017.
3. Монастирський Г. Л. Теорія організації: підручник. 2-е вид. Тернопіль: Крок, 2019. 368 с.
4. Wysocki, R. K. Effective Project Management: Traditional, Agile, Extreme. Wiley, 2013.
5. Cleland, D. I., & Ireland, L. R. Project Management: Strategic Design and Implementation. McGraw-Hill, 2006.
6. Gido, J., & Clements, J. P. Successful Project Management. Cengage Learning, 2014.
7. Lewis, J. P. Fundamentals of Project Management. AMACOM, 2006.
8. Heagney, J. Fundamentals of Project Management. AMACOM, 2016.
9. Turner, J. R. The Handbook of Project-Based Management: Leading Strategic Change in Organizations. McGraw-Hill, 2014.
10. Verzuh, E. The Fast Forward MBA in Project Management. Wiley, 2015.
11. Marchewka, J. T. Information Technology Project Management. Wiley, 2014.
12. Phillips, J. IT Project Management: On Track from Start to Finish. McGraw-Hill Education, 2016.
13. Fleming, Q. W., & Koppelman, J. M. Earned Value Project Management. Project Management Institute, 2016.
14. Crawford, J. K. The Strategic Project Office: A Guide to Improving Organizational Performance. CRC Press, 2002.
15. Heldman, K. PMP Project Management Professional Exam Study Guide. Sybex, 2018.

16. Schwalbe, K. Information Technology Project Management. Cengage Learning, 2015.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр
Галузь знань: 12 – Інформаційні технології.
Спеціальність 123 – Комп'ютерна інженерія.

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
_____ д.т.н., проф. Шматков С. І.

«17»листопада 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Криничної Аліни Олександрівни
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«Комп'ютерна система планування ІТ-проєктів»**

керівник роботи Булавін Дмитро Олексійович, канд. техн.наук, доцент, доцент кафедри теоретичної та прикладної системотехніки.
затверджені наказом по університету від «23» травня 2023 року № 4101-5/895

2. Строк подання студентом роботи 26 травня 2023

3. Перелік питань, які потрібно розробити
1. Аналіз структури ІТ-проєкту, принципи формування команд та складові частини процесу планування.
2. Фази планування, підготовка початкових даних моделі планування ІТ-проєктів.
3. Математична модель автоматизованого планування ІТ-проєктів.
4. Алгоритм програмної моделі планування ІТ-проєктів.
5. Розробка програмного засобу планування ІТ-проєкту.
6. Тестування програмного засобу планування ІТ-проєкту.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1.	Затвердження теми роботи	Жовтень 2022
2.	Аналіз предметної області та пошук літератури	Листопад 2022
3.	Аналіз фази планування ІТ-проєктів	Листопад 2022-Грудень 2022
4.	Проєктування математичної моделі засобу планування ІТ-проєктів	Січень 2023-Лютий 2023
5.	Розробка та тестування програмного засобу планування ІТ-проєктів	Березень 2023-Квітень 2023
6.	Оформлення пояснювальної записки	Травень 2023
7.	Передзахист кваліфікаційної роботи	Травень 2023
8.	Представлення кваліфікаційної роботи керівнику та рецензенту	Травень 2023

5. Дата видачі завдання 19 листопада 2022

Студент Кринична А. О.
ініціали, прізвище
підпис

Керівник роботи Булавін Д.О.
ініціали, прізвище
підпис

Додаток Б

Затверджую

_____

«___» _____ 2023 р.

Технічне завдання
на розробку програмного виробу «Комп'ютерна система планування ІТ-проєктів»

1.	Введення	1.1. Назва: Комп'ютерна система планування ІТ-проєктів. 1.2. Галузь застосування: Інформаційні технології
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія 2.2. Завдання на кваліфікаційну роботу бакалавра № __4101-5/895__ від «_23_» _____05_____2023 (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: створення сервісу, який допоможе управляти проєктами та підбирати ІТ-команду в галузі інформаційних технологій. 3.2. Призначення розробки надає можливість системі створювати та керувати проєктом, слідкувати за виконанням завдання, визначати терміни їх виконання, контролювати витрати та вирішувати проблеми, що виникають у процесі виконання проєкту. 3.3. Вихідні дані розробки: сервіс планування ІТ-проєкту з підбором команди; вхідні дані розробки: кількість годин по кожному з напрямків.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: можливість контролю завдань, ресурсів та термінів виконання проєктів з можливістю моніторингу та звітності про їх реалізацію, підбирати команду спеціалістів. Також вона повинна мати інтерфейс для взаємодії з користувачами. 4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу . 4.3.Вимоги до умов експлуатації: немає 4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows,

		Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи потрібний інтерпретатор мови програмування. 4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 10, підтримка мови програмування, підтримка різних платформ. 4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються. 4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються. 4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.	
5.	Вимоги до програмної документації	Програмною документацією до виробу «Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.2 та 3.3 пояснювальної записки до кваліфікаційної роботи). 3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)	
6.	Вимоги до техніко-економічних показників	Програмною документацією до виробу «Комп'ютерна система планування IT-проєктів» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи). 3) Джерела базової інформації.	
7.	Стадії і етапи розробки	Дата	Назва етапу
		від 20 листопада 2022 до 10 грудня 2022	Аналіз предметної області та пошук літератури.
		від 25 листопада 2022 до 30 грудня 2023	Аналіз фази планування IT-проєктів
		від 1 січня 2023	Проєктування математичної моделі засобу планування IT-

Додаток В

Програма і методика випробувань програмного виробу

«Комп'ютерна система планування ІТ-проєктів»

1 Об'єкт випробувань

1. Назва програмного виробу : «Комп'ютерна система планування ІТ-проєктів»
2. Галузь застосування : Інформаційні технології
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

4.1. Вимоги до функціональних характеристик: можливість контролю завдань, ресурсів та термінів виконання проектів з можливістю моніторингу та звітності про їх реалізацію, розраховувати бюджет. Також вона повинна мати

інтерфейс для взаємодії з користувачами.
4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу .
4.3.Вимоги до умов експлуатації: немає
4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Спеціальні вимоги (не пред'являються). Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи потрібний інтерпретатор мови програмування.
4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 10, підтримка мови програмування, підтримка різних платформ.
4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються.
4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються.
4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.

5. Вимоги до програмної документації

Документацією до виробу «Комп'ютерна система планування IT-проєктів» вважати:

1) Документація по мові програмування та додаткові мануали.

2) Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).

4) Джерела базової інформації.

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Засоби випробувань представлено на ПК на яких встановлено наступні програмні засоби: інтерпретатор мови програмування.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

- ознайомчий (1-й етап);
- власне випробування програмного виробу (2-й етап).

lead.html

G: > work > Diplom > HTML&CSS > lead.html > html > body > nav > ul > li

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="./css/style.css">
    <title>IT-service</title>
    <link rel="shortcut icon" href="./images/icon.png" type="image/x-icon">
    <link rel="ss" href="./images/ss.png"  type="image/png"sizes="16x16">
</head>
<body >
   <nav>
      <ul>
          <li>
             <a href="log_in.html">Авторизація</a>
          </li>
          <li>
            <a href="lead.html">Головна</a>
         </li>
      </ul>
    </nav>

<div class="slog">
   <h1>IT-SERVICE</h1>
</div>
<div class="slog_1">
   <h2>Для розрахунку бюджету проєкту</h2>
</div>
</body>
</html>
```

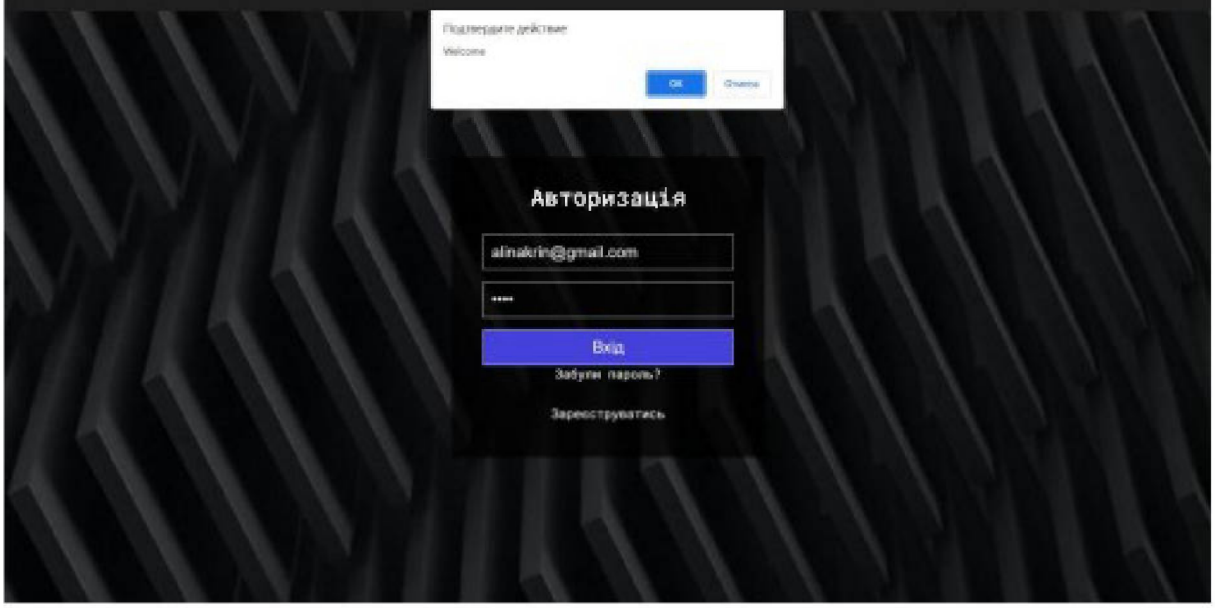

Рис. В.1 Тест 1

Тест 2

1. Перевірка виконання програми
2. Ознайомлення з додатковою інформацією;
3. Отримання результатів.

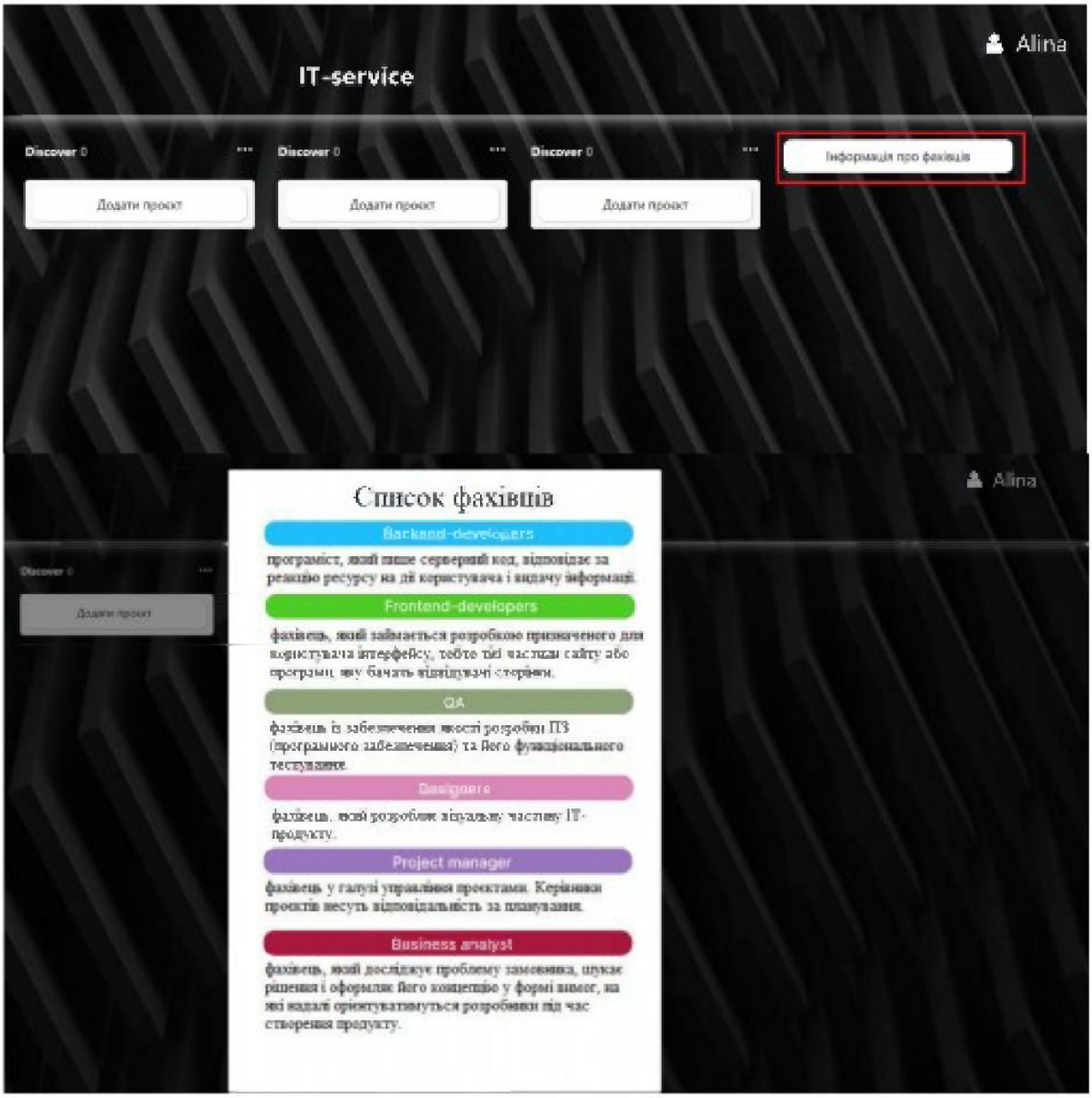

Рис. В.2 Тест 2

Тест 3

4. Перевірка виконання програми
5. Рекомендації щодо проєкту за нашим замовленням;
6. Отримання результату.

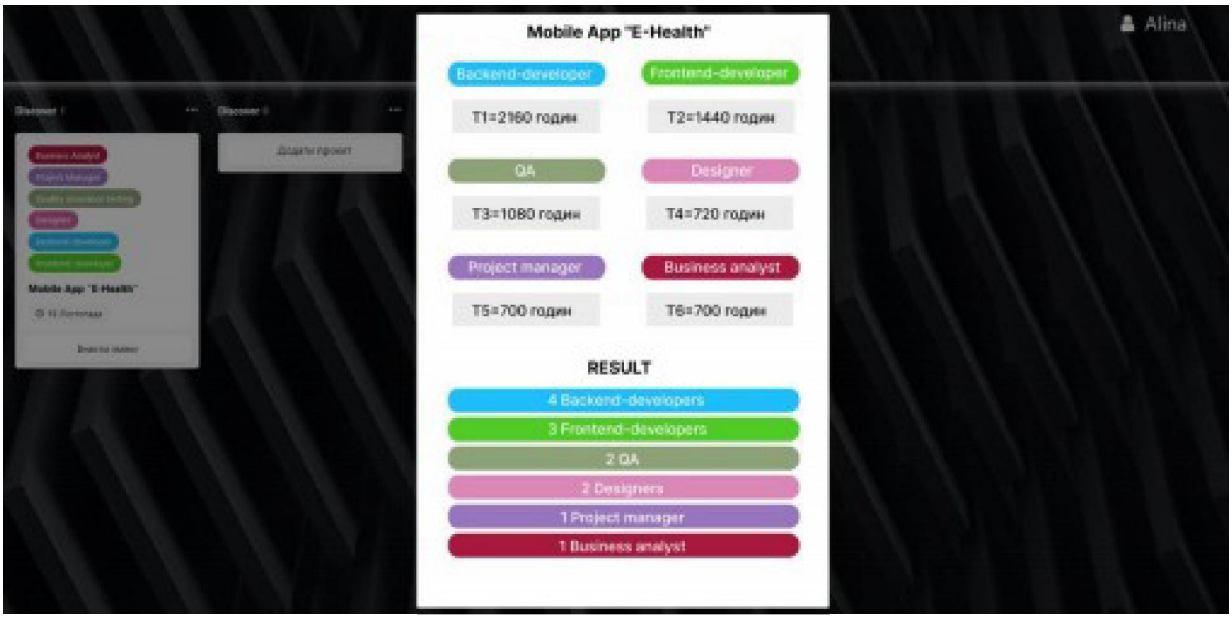

Рис. В.3 Тест 3

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: тест 1 успішно пройшов випробування, тест 2 успішно пройшов випробування і тест 3 успішно пройшов випробування. Випробування пройшло успішно.

Виконавець: студентка групи КІ-41, Кринична А. О.

Додаток Г

Лістинг коду

«Комп'ютерна система планування ІТ-проєктів»

Головна сторінка «IT-Service»

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" type="text/css" href="./css/style.css">
   <title>IT-service</title>
   <link rel="shortcut icon" href="./images/icon.png" type="image/x-icon">
   <link rel="ss" href="./images/ss.png"  type="image/png"sizes="16x16">
</head>
<body >
  <nav>
    <ul>
      <li>
        <a href="log_in.html">Авторизація</a>
      </li>
      <li>
        <a href="lead.html">Головна</a>
      </li>
    </ul>
  </nav>

<div class="slog">
  <h1>IT-SERVICE</h1>
</div>
<div class="slog_1">
  <h2>Для підбору  команди проєкту</h2>
</div>
</body>
```

Авторизація користувача

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" type="text/css" href="./css/style.css">
   <title>IT-service</title>
   <link rel="shortcut icon" href="./images/icon.png" type="image/x-icon">
   <link rel="ss" href="./images/ss.png"  type="image/png"sizes="16x16">



</head>
<body >
  <nav>
    <ul>
      <li class="login">
        <a href="log_in.html">Авторизація</a>
      </li>

      <li class="leadt">
        <a href="lead.html">Головна</a>
      </li>



    </ul>
  </nav>

  <div class="form"> <h1>Авторизація</h1>
    <div class="input-form">
      <input type="text" placeholder="Логін" id="login">

    </div>
    <div class="input-form">
      <input type="password" placeholder="Пароль" id="password">
```

```
    </div>

    <div class="input-form">

        <button class="butt"   id="check">Вхід</button>
      </a>
      <a href="forget_password.html" class="forget">Забули пароль?</a>

     <a href="regestration.html" class="registretion">Зареєструватись</a>
    </div>


  </div>
<script src="log_in.js"></script>

</body>
</html>
document.getElementById('check').onclick = function() {
    let login = document.getElementById('login');
    let password = document.getElementById('password');


    if (login.value === 'alinakrinichnaya@gmail.com' && password.value === '2032'){

  if (confirm ('Welcome')){
   window.location.href = 'http://localhost:3200/'
  }

  }
  if (login.value === 'alinakrinichnaya@gmail.com' && password.value === '1032'){

  else alert ('Incorrect login or pass');


}
}
```

Відновлення паролю

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/ess" href="./css/style.css">
    <title>IT-service</title>
    <link rel="shortcut icon" href="./images/icon.png" type="image/x-icon">




</head>
<body >

   <nav>
     <ul>

       <li>
         <a href="log_in.html">Авторизація</a>
       </li>

         <li class="leadl">
           <a href="lead.html">Головна</a>
         </li>


     </ul>
    </nav>



    <div class="recovery1">
      <p class="success-response"> На пошту надіслано інструкцію з подальшими діями.</p>
    </div>

<a href="log_in.html">
```

```
    <button class="btn">Вхід</button>
   </a>



</body>
</html>
```

Реєстрація аккаунта

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="./css/style.css">
    <title>IT-service</title>
    <link rel="shortcut icon" href="./images/free-icon-computer-9696506.png" type="image/x-
icon">
    <link rel="ss" href="./images/ss.png"  type="image/png"sizes="16x16">



</head>
<body >
  <nav>
    <ul>
      <li class="login">
        <a href="log_in.html">Авторизація</a>
      </li>

      <li class="leadt">
        <a href="lead.html">Головна</a>
      </li>


    </ul>
  </nav>

  <div class="form"> <h1>Реєстрація</h1>
```

```
    <div class="input-form">
      <input type="text" placeholder="Логін">

     </div>
    <div class="input-form">
      <input type="password" placeholder="Пароль">

    </div>
    <div class="input-form">
      <input type="password" placeholder="Підтвердити пароль">

    </div>
    <div class="input-form">

      <a href="log_in.html">

        <input type="submit" value="Зареєструватись">
      </a>

      <a href="log_in.html" class="forget">Увійти</a>


    </div>


  </div>


</body>
</html>
```

Відновлення паролю

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="./css/style.css">
```

```
        <title>IT-service</title>
        <link rel="shortcut icon" href="./images/free-icon-computer-9696506.png" type="image/x-
icon">
        <link rel="ss" href="./images/ss.png"  type="image/png"sizes="16x16">


    </head>
    <body >
      <nav>
        <ul>
          <li class="login">
            <a href="log_in.html">Авторизація</a>
          </li>

          <li class="leadt">
            <a href="lead.html">Головна</a>
          </li>

            Відновлення паролю
        </ul>
      </nav>

      <div class="form"> <h1>Не можете увійти?</h1>
        <div class="input-form">
          <input type="text" placeholder="E-mail">

        </div>

        <div class="input-form">

          <a href="recovery.html">
            <input type="submit" value="Відновити">
          </a>

          <a href="log_in.html" class="forget">Увійти</a>
        </div>

      </div>
```

```
</body>
</html>
```

Головна сторінка розрахунків бюджету

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
import React, { useState } from "react";
import { CheckSquare, Clock, MoreHorizontal } from "react-feather";

import Dropdown from "../Dropdown/Dropdown";

import "./Card.css";
import CardInfo from "./CardInfo/CardInfo";

function Card(props) {
  const [showDropdown, setShowDropdown] = useState(false);
  const [showModal, setShowModal] = useState(false);

  const { id, title, date, tasks, labels } = props.card;

  const formatDate = (value) => {
    if (!value) return "";
    const date = new Date(value);
    if (!date) return "";

    const months = [
```

```
    "Січня",
    "Лютого",
    "Березня",
    "Квітня",
    "Травня",
    "Червня",
    "Липня",
    "Серпня",
    "Вересня",
    "Жовтня",
    "Листопада",
    "Грудня",
  ];

  const day = date.getDate();
  const month = months[date.getMonth()];
  return day + " " + month;
};


return (
  <>
    {showModal && (
      <CardInfo
        onClose={() => setShowModal(false)}
        card={props.card}
        boardId={props.boardId}
        updateCard={props.updateCard}
      />
    )}
    <div
      className="card"
      draggable
      onDragEnd={() => props.dragEnded(props.boardId, id)}
      onDragEnter={() => props.dragEntered(props.boardId, id)}
      onClick={() => setShowModal(true)}
    >
      <div className="card_top">
```

```
<div className="card_top_labels">
  {labels?.map((item, index) => (
    <label key={index} style={{ backgroundColor: item.color }}>
      {item.text}
    </label>
  ))}
</div>
<div
  className="card_top_more"
  onClick={(event) => {
    event.stopPropagation();
    setShowDropdown(true);
  }}
>
  <MoreHorizontal />
  {showDropdown && (
    <Dropdown
      class="board_dropdown"
      onClose={() => setShowDropdown(false)}
    >
      <p onClick={() => props.removeCard(props.boardId, id)}>
       Видалити проєкт
      </p>
    </Dropdown>
  )}
</div>
</div>
<div className="card_title">{title}</div>
<div className="card_footer">
  {date && (
    <p className="card_footer_item">
      <Clock className="card_footer_icon" />
      {formatDate(date)}
    </p>
  )}
  {tasks && tasks?.length > 0 && (
    <p className="card_footer_item">
```

```
            <CheckSquare className="card_footer_icon" />
            {tasks?.filter((item) => item.completed)?.length}/{tasks?.length}
          </p>
        )}
      </div>
    </div>
  </>
 );
}

export default Card;
import React, { useEffect, useState } from "react";

import {
 Calendar,
 CheckSquare,
 Command,
 List,
 Tag,
 Trash,
 Type,
 X,
} from "react-feather";

import Modal from "../../Modal/Modal";
import Editable from "../../Editabled/Editable";

import "./CardInfo.css";

function CardInfo(props) {
 const colors = [
  "#a8193d",
  "#4fcc25",
  "#1ebffa",
  "#8da377",
  "#9975bd",
  "#cf61a1",
```

```
];

const [selectedColor, setSelectedColor] = useState();
const [values, setValues] = useState({
  ...props.card,
});

const updateTitle = (value) => {
  setValues({ ...values, title: value });
};

const updateDesc = (value) => {
  setValues({ ...values, desc: value });
};

const addLabel = (label) => {
  const index = values.labels.findIndex((item) => item.text === label.text);
  if (index > -1) return;

  setSelectedColor("");
  setValues({
    ...values,
    labels: [...values.labels, label],
  });
};

const removeLabel = (label) => {
  const tempLabels = values.labels.filter((item) => item.text !== label.text);

  setValues({
    ...values,
    labels: tempLabels,
  });
};

const addTask = (value) => {
  const task = {
```

```
    completed: false,
    text: value,
  };
  setValues({
    ...values,
    tasks: [...values.tasks, task],
  });
};

const removeTask = (id) => {
  const tasks = [...values.tasks];

  const tempTasks = tasks.filter((item) => item.id !== id);
  setValues({
    ...values,
    tasks: tempTasks,
  });
};

const updateTask = (id, value) => {
  const tasks = [...values.tasks];

  const index = tasks.findIndex((item) => item.id === id);
  if (index < 0) return;

  tasks[index].completed = value;

  setValues({
    ...values,
    tasks,
  });
};

const calculatePercent = () => {
  if (!values.tasks?.length) return 0;
  const completed = values.tasks?.filter((item) => item.completed)?.length;
  return (completed / values.tasks?.length) * 100;
```

```
const updateDate = (date) => {
  if (!date) return;

  setValues({
    ...values,
    date,
  });
};

useEffect(() => {
  if (props.updateCard) props.updateCard(props.boardId, values.id, values);
}, [values]);

return (
  <Modal onClose={props.onClose}>
    <div className="cardinfo">
      <div className="cardinfo_box">
        <div className="cardinfo_box_title">
          <Type />
          <p>Title</p>
        </div>
        <Editable
          defaultValue={values.title}
          text={values.title}
          placeholder="Введіть назву"
          onSubmit={updateTitle}
        />
      </div>

      <div className="cardinfo_box">
        <div className="cardinfo_box_title">
          <List />
          <p>Опис</p>
        </div>
        <Editable
          defaultValue={values.desc}
```

```
    placeholder="Введіть опис"
    onSubmit={updateDesc}
   />
  </div>

  <div className="cardinfo_box">
   <div className="cardinfo_box_title">
    <Calendar />
    <p>Дата початку</p>
   </div>
   <input
    type="Date"
    defaultValue={values.date}
    min={new Date().toISOString().substr(0, 10)}
    onChange={(event) => updateDate(event.target.value)}
   />
  </div>
  <div className="cardinfo_box">
   <div className="cardinfo_box_title">
    <Calendar />
    <p>Дата кінця</p>
   </div>
   <input
    type="Date"
    defaultValue={values.date}
    min={new Date().toISOString().substr(0, 10)}
    onChange={(event) => updateDate(event.target.value)}
   />
  </div>


  <div className="cardinfo_box">
   <div className="cardinfo_box_title">
   <i class="fa fa-users icon"></i>
    <p>Команда</p>
   </div>
   <div className="cardinfo_box_labels">
```

```
      <label
        key={index}
        style={{ backgroundColor: item.color, color: "#fff" }}
      >
        {item.text}
        <X onClick={() => removeLabel(item)} />
      </label>
    ))}
  </div>
  <ul>
    {colors.map((item, index) => (
      <li
        key={index + item}
        style={{ backgroundColor: item }}
        className={selectedColor === item ? "li_active" : ""}
        onClick={() => setSelectedColor(item)}
      />
    ))}
  </ul>
  <Editable
    text="Додати команду"
    placeholder="Додати команду розробників"
    onSubmit={(value) =>
      addLabel({ color: selectedColor, text: value })
    }
  />
</div>

<div className="cardinfo_box">
  <div className="cardinfo_box_title">
    <List/>
    <p>Коментар</p>
  </div>
  <div className="cardinfo_box_progress-bar">
    <div
      className="cardinfo_box_progress"
      style={{
```

```
            backgroundColor: calculatePercent() === 100 ? "limegreen" : "",
          }}
        />
      </div>
      <div className="cardinfo_box_task_list">
        {values.tasks?.map((item) => (
          <div key={item.id} className="cardinfo_box_task_checkbox">
            <input
              type="checkbox"
              defaultChecked={item.completed}
              onChange={(event) =>
                updateTask(item.id, event.target.checked)
              }
            />
            <p className={item.completed ? "completed" : ""}>{item.text}</p>
            <Trash onClick={() => removeTask(item.id)} />
          </div>
        ))}
      </div>
      <Editable
        text={"Додати коментар"}
        placeholder="Додати коментар"
        onSubmit={addTask}
      />
    </div>
  </div>
 </Modal>
 );
}

export default CardInfo;
```