Міністерство освіти і науки України
Харківський національний університет
радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Методи оцінювання ефективності програмних проектів та оптимізації процесів розробки
(тема)

Виконала: студент 2 курсу, групи ІПЗм-17-1
спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-професійної програми

Інженерія програмного забезпечення
(повна назва освітньої програми)

Лук'янова К.Ю.
(прізвище, ініціали)

Керівник проф. Дудар З.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

20___р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук_____

Кафедра Програмної інженерії_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність 121-Інженерія програмного забезпечення_____
(код і повна назва)

освітньо-наукова програма _Інженерія програмного забезпечення_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«____»_____20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Лук'яновій Катерині Юріївні_____
(прізвище, ім'я, по батькові)

Тема роботи _Методи оцінювання ефективності програмних проектів та оптимізації процесів розробки _____
затверджена наказом по університету від "___" _____20 ___ р № _____
заповнюється вручну після отримання наказу

2. Термін подання студентом роботи до екзаменаційної комісії
11 червня 2019 р.

3. Вихідні дані до роботи *алгоритми оцінки ефективності програмних проекті, алгоритми оптимізації процесів розробки пояснювальна записка. Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів оцінювання ефективності керування програмними проектами, ефективність керування проектами _успішність IT проектів , пошук оптимізації керування проектами за допомогою розробленного програмного додатку* _____

Зворотний бік бланку завдання

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Мета завдання, обґрунтування доцільності розроблення, постановка задачі, об'єктна модель системи, базові моделі, методи й алгоритми, структура бази даних, структурно-логічна схема взаємодії даних, план захисту інформації (за необхідністю), інтерфейс програмної системи, результати тестування програмної системи, демонстраційні матеріали*

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Снецчастина	проф. Дудар. З. В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі	19 квітня 2019р.	
2.	Огляд існуючих методів	27 квітня 2019р.	
3.	Методи швидкого детектування відрізків ліній		
4.	Підготовка пояснювальної записки	25 травня 2019р.	
5.	Спецчастина	26 травня 2019р.	
6.	Підготовка презентації та доповіді	31 травня 2019р.	
7.	Попередній захист	05 червня 2019р.	
8.	Нормоконтроль, рецензування	06 червня 2019р.	
9.	Занесення диплома в електронний архів	07 червня 2019р.	
10.	Допуск до захисту у зав. кафедри	10 червня 2019р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____2019 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Дудар З.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 97 с., 33 рис., 4 табл., 20 джерел.

КЕРУВАННЯ ПРОЕКТАМИ, ЕФЕКТИВНІСТЬ, ПЛАНУВАННЯ ПРОЕКТУ, ДІАГРАМА ГАНТА, PMBOK, SERVERLESS, AWS, WEB-ДОДАТОК.

Об’єктом дослідження є IT індустрія та аналіз ефективності управління проектами.

Метою роботи є дослідження методів оцінки ефективності програмних проектів та розробка сучасної зручної системи для оптимізації процесів розробки.

Методи розробки базуються на технології Node JS, React, сервері бази даних NoSQL, AWS Lambda та Serverless.

У результаті атестаційної роботи здійснена програмна реалізація веб-додатка який допомагає менеджерам в організаціях з розробки програмного забезпечення оцінити, моніторити та підвищити ефективність управління проектами.

PROJECT MANAGEMENT, EFFECTIVENESS, PLANNING, GANTT DIAGRAM, PMBOK, SERVERLESS, AWS, WEB-APPLICATION.

The object of research is the IT industry and the analysis of the effectiveness of project management. The purpose of the work is to study the methods of evaluating the effectiveness of software projects and the development of a modern convenient system for optimizing development processes. The development methods are based on the technology NodeJS, React, the database NoSQL and Serverless architecture with AWS Lambda. As the result of attestation diploma work, existing metrics for assessing the effectiveness of software management were analyzed, and a WEB application was developed that helps project managers in software development organizations evaluate, monitor and increase the effectiveness of project management.

ЗМІСТ

ВСТУП

Щоб бути конкурентоспроможними в сучасному світі, в будь-якій галузі, організації мають справу з безпрецедентними змінами, обумовленими такими факторами, як темпи змін, зміна глобальної економічної сили, зміни поколінь, вплив цифрової економіки і поява глобальних гравців, які порушують традиційні сектори. Швидкість, якість, пріоритетність, дисципліна, адаптивність - це всі ключові якості, які залучають клієнтів та інших зацікавлених сторін до використання продуктів або послуг та побудови довготривалих відносин. У зв'язку з цими змінами традиційні методології управління проектами часто більше не вважаються ефективними, особливо в IT-галузі. Щоб досягти успіху в середовищі зі змінюваними вимогами, не тільки бізнес, але й керування проектами має бути адаптивним і гнучким.

Оцінка ефективності IT-проекту є непростим завданням для сучасних проектних менеджерів, бо за останні 30 років розробка програмного забезпечення динамічно змінювалася. Серед дев'яти індустрій, лише в галузі програмного забезпечення, перевитрати бюджету та низька продуктивність явно вказані як проблеми серед інших. Зазвичай середній IT-проект відстає від графіку від 6 до 12 місяців і від 50 до 100 відсотків мають перевитрати бюджету [1]. Можна було б очікувати, що з усіма досягненнями в технічних аспектах інженерії програмного забезпечення результат розробки програмних проектів повинен бути набагато кращим. Однак, недостатньо покладатися лише на технічний прогрес. Потрібні також значні досягнення в галузі управління проектами програмного забезпечення для досягнення кращих результатів у програмних проектах.

Неефективне управління IT-проектами є одним з основних причин провалів програмних проектів [2]. Насамперед, ефективне керування проектами програмного забезпечення є ключовим елементом в досягненні успіху проекту. Щоб поліпшити якість управління і зосередити зусилля на вирішувані проблем, важливо спочатку визначити та виміряти ефективність управління програмними

проектами [2]. Демарк і Лістер стверджують, що «Для переважної більшості збанкрутілих проектів, які ми розглядали, там не було жодної технологічної проблеми, яка б могла пояснити провал проекту». Робертсон та інші підкреслювали, що “за кілька десятиліть досвіду проекту ми ніколи не бачили, щоб проект не закінчився успіхом з технічних причин. Це завжди були людські невдачі, які призвели до того, що хороші проекти припинялися». Погане управління може збільшити витрати на програмне забезпечення скоріше за будь-який інший фактор. Інші дослідники та практики також акцентують увагу щодо важливості управління проектами програмного забезпечення для успішності програмних проектів [3].

Багато алгоритмічних і не алгоритмічних методів використовуються для оцінки ефективності проектів, такі як SLIM (управління життєвим циклом програмного забезпечення), Halstead Model, модель Bailey-Basil, модель COCOMO [3] і аналіз функцій тощо, але дані методи не оцінюють точно всі види програмного забезпечення. Сьогодні ці традиційні методики не популярні. Управління проектами є складним завданням, і розробка метрики для ефективності управління проектами також не є легким завданням. Проте вимірювання та оцінювання ефективності управління в програмних проектах відкриває багато можливостей для вдосконалення.

У цій атестаційній роботі магістра буде розглянуто та проаналізовано кілька найпоширеніших методів оцінки ефективності управління проектами, які було згадано вище. Також будуть представлені результати дослідження успіху проекту, присвяченого конкретно українському IT ринку та розроблено сучасного зручного Web-додатку для керівників проектів, який буде допомагати в оптимізації процесів розробки.

1 ПОСТАНОВКА ЗАДАЧІ

Упродовж останнього десятиліття управління проектами стає все більш складним. Сьогодні ще й досі існують проблеми при розробці програмного забезпечення. Проекти мають тенденцію тривати довше, ніж очікувалося, і витрачається більше коштів, ніж передбачалося. Проаналізувавши роботу сотень американських корпорацій і підсумки виконання декількох десятків тисяч проектів, пов'язаних з розробкою ПЗ, Standish Group [4] прийшла до наступних невтішних висновків. Тільки 29% проектів завершилися в строк, не перевищили запланований бюджет і реалізували всі необхідні функції і можливості. 52% проектів завершилися із запізненням, витрати перевищили запланований бюджет, необхідні функції не були реалізовані в повному об'ємі. Середнє перевищення термінів склало 120%, середнє перевищення витрат 100%, зазвичай виключалося значне число функцій. 19 % проектів повністю провалилися і були анулюванні до завершення.

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

Рисунок 1.1. – Статистика с успіху всіх проектів з 2011 по 2015 роки

Цей результат, через 4 роки після звіту, дозволяє проаналізувати той факт, що відсоток успіху проекту все ще є фактором дослідження [5]. Провал IT-проекту пояснюється тим, що програмісти та менеджери приділяють більше уваги до етапу розробки продукту, ніж етапу планування. Мета, сфера застосування, час

виконання і оснащена команда важливі як уявлення про те, що буде отримано внаслідок реалізації IT проекту. Широкі теоретичні дослідження з чинних веб-ресурсів, форумів і знань експертів в області інформаційних технологій проектування, безсумнівно, мають значення. Але найважливішими характеристиками в будь-якому IT проекті буде чітка постановка завдання, зобов'язання довести рішення до працюючого результату або переконаного рішення про неможливість його досягнення. Тому планування має великий вплив на управління проектом, розподіл ресурсів і коштів.

1.1 Особливості управління IT проектами

У сучасному світі інформаційні технології проникли практично в усі сфери нашої життєдіяльності, промисловості та бізнесу. З кожним днем зростає кількість користувачів інтернету, різних гаджетів, з'являються «розумні» будинки, «розумні» машини тощо. У зв'язку з цим збільшується потреба в високоякісних технологіях і програмному забезпеченні, яке використовує ці технології.

Управління IT-проектом вдає із себе процес планування, організації і розмежування відповідальності за завершення конкретних цілей, пов'язаних з інформаційними технологіями (IT) організації.

Управління IT-проектами включає в себе, здійснення нагляду над проектами з розробки програмного забезпечення, налаштування обладнання, модернізації мережі, хмарних обчислень, бізнес-аналітиці, управління даними і впровадження IT-послуг. На додаток до звичайних проблем, які можуть призвести до провалу проекту, є фактори, які також можуть негативно вплинути на успіх IT-проекту:

- розвиток в технології на стадії впровадження проекту;
- інфраструктурні зміни, які впливають на безпеку і управління даними;
- невідомі залежні відносини між обладнанням, програмним забезпеченням мережевої інфраструктури та даних.

Оскільки IT-проекти як правило, являють собою нові технології, які не були реалізовані або використані раніше в організації, практично завжди існує ймовірність ускладнення, яка вплине на успішність проекту.

Існують п'ять груп процесів, що включають в себе життєвий цикл управління проектом і є універсальними для всіх проектів. Однак конкретні етапи в межах проекту є унікальними для кожного проекту і являють собою життєвий цикл проекту [6]. Ці п'ять груп представлені нижче:

– ініціалізація - визначені цілі проекту, потреби або проблеми. Призначено менеджер проекту і створений статут проекту;
– планування - керівник проекту і команда проекту працюють разом, щоб спланувати всі необхідні кроки, для досягнення успішного завершення проекту. Процеси планування проекту мають ітеративний характер і очікується, що планування буде проходити часто протягом усього проекту;
– виконання - як тільки план проекту був створений, команда проекту виконує завдання за планом виконання проекту для досягнення результатів проекту. Протягом реалізації проект може перейти до стадії планування за необхідністю;
– моніторинг і контроль - під час виконання проекту командою, менеджер моніторить і контролює роботу за часом, витратами, якістю, ризиками та іншими факторами проекту. Моніторинг і контроль також є безперервним процесом, для гарантування того що проект вирішує свої завдання за кожною ціллю;
– закриття - в кінці кожного етапу і в кінці всього проекту, відбувається закриття проекту для того, щоб перевірити і переконатися в завершеності всіх робіт, після чого відбувається здача проекту.

Для досягнення стабільних позитивних результатів проекті повинні бути легкокерованими. Розробка легкокерованих проектів вимагає діяльності у двох паралельних напрямах. По-перше, необхідно залучити всіх, хто зацікавлений у виконанні проекту (тобто всіх, хто може виграти при здійсненні проекту), до

визначення конкретних цілей проекту і засобів їх досягнення. По-друге, необхідний пошук такого варіанту (серед наявних варіантів), який би забезпечував економне витрачання ресурсів при реалізації.

Управління проектом — це процес керівництва та координації людських, матеріальних та фінансових ресурсів протягом життєвого циклу проекту шляхом застосування сучасних методів та техніки управління для досягнення визначених у проекті результатів за складом та обсягом робіт; вартістю, часом, якістю та задоволенню інтересів учасників проекту.

1.2 Успішність проекту

Успіх проекту - це одна з найменш опрацьованих концепцій управління проектами. "Для тих, хто бере участь в проекті, успіх проекту, як правило, розглядається як досягнення певної наперед визначеної мети проекту" [1], а широка громадськість має різні думки, які базуються на задоволеності користувачів. У традиційній методології управління проектами, в тому числі PMbok [6], вважається, що проект успішний, якщо він виконаний в зазначений термін, в рамках затвердженого бюджету і задовольняє замовника якістю. Всі ці аспекти не тільки однаково важливі для проекту, але і взаємопов'язані між собою. Цей взаємозв'язок отримала назву «трикутник проекту».

Рисунок 1.2. «Трикутник проекту»

Ідея трикутника полягає в наданні можливості учасникам і зацікавленим сторонам проекту визначити рамки проекту і врівноважувати його конкуруючі потреби: гроші - бюджет проекту, час - тривалість виконання проекту, зміст - обсяг необхідних робіт для досягнення цілей проекту і якість - якість кінцевого продукту. Даний підхід став способом моніторингу та контролю проектів. Пізніше, цей підхід, де факто, став методом визначення та вимірювання успішності проекту [1]. В даному підході не можливо оптимізувати всі чотири обмеження, одна зі сторін завжди буде в програші.

1.3 Метрики оцінювання ефективності керування програмними проектами

Успіх управління проектами не теж саме що і успіх проекту. Незважаючи на те, що більшість практикуючих проектних менеджерів підкреслюють, що успіх проектів програмного забезпечення тісно пов'язаний з якістю та успішністю управління проектами, не існує встановлених емпіричних доказів такого зв'язку в літературі з управління проектами. Пов'язані емпіричні дослідження в галузі програмної інженерії або навіть у літературі з управління проектами досить обмежені. Це не випадково. Для цього є кілька причин. На сьогодні запропоновано багато моделей оцінювання.

В даній роботі їх було розподілено на 3 категорії вимірювання ефективності керування проектами:

– на основі анкети;
– на основі метрик;
– на основі моделі.

1.3.1 Оцінювання на основі анкети

У першому підході вимірювання ефективності управління базується на оцінці відповідей на анкету. Оцінки, що базуються на анкетах, є загальними в управлінських та організаційних областях соціології. Це пояснюється тим, що абстрактні поняття, такі як робота в команді, організаційна прихильність, комунікація, лідерство тощо, важко аналізувати кількісно. Цей підхід був використаний у розробці метрики управління якістю для розробки програмного забезпечення [7].

У дослідженні Osmundson et al, 2003, було розроблено анкету для вивчення того, які найкращі практики управління дотримуються для керування IT - проектом. На основі відповідей на питання вимірюється якість управління проектом. Вони також порівнювали отриману метрику (QMM) з метрикою, зібраною через суб'єктивну оцінку, розглянуту в попередньому розділі. Анкета досліджує чотири важливі сфери управління проектами програмного забезпечення.

Це управління вимогами до проекту, планування та оцінка проектів, управління ризиками та управління людьми [8]. Управління людьми далі поділяється на чотири області: людські ресурси, лідерство, комунікації та технічні компетенції менеджера програми. Повний комплект анкет включав 457 запитань. Показник QMM базується на шкалі від 0 до 10, причому 0 є найнижчим показником якості, а 10 - найвищим показником якості. Важливість дослідження QMM полягає у фокусі на розробці метрики якості або ефективності управління проектами в програмних проектах.

COCOMO II включає фактор зрілості процесу (PMAT) як масштабний фактор для оцінки зусиль [9]. Важливо відзначити, що масштабні коефіцієнти експоненціально впливають на оцінку зусиль. У COCOMO II цей фактор PMAT визначається з використанням одного з двох методів. Перший метод ґрунтується на рейтингу організації SW-CMM, якщо такий існує. Другий метод

використовується, коли організація не має рейтингу SW-CMM. Другий метод використовує інший рейтинг, еквівалентний рівень зрілості процесу (EPML), який базується на відсотку відповідності для кожної ключової цільової області процесу в моделі SW-CMM. Ця відповідність (рейтинг EPML) оцінюється за допомогою відповідей на анкету, отриману з вісімнадцяти ключових областей процесу.

Боем [9] надав ретельні огляди 38 підходів і різних моделей, розглянув їхні сильні та слабкі сторони і прийшов до наступних висновків: жодна методика не може бути пріоритетною для усіх ситуацій; поєднання різних методів у більшості випадків дає хороші результати; метод використання нейронних мереж та динамічні методи не є достатньо зрілими на момент дослідження. В даному дослідженні методи машинного навчання і регресійні методи використовуються для певних задач або їхні елементи вже інтегровані у моделі.

1.3.2 Оцінювання на основі метрик

Інший підхід до вимірювання ефективності керуванням ІТ-проектом здійснюється за допомогою інших програмних метрик. Наприклад, такі показники, як кількість дефектів у часі, складність програмного забезпечення, якість вимог до ПЗ, коефіцієнт текучості персоналу тощо, можуть використовуватися як вхідні дані для метричної моделі для оцінки ефективності управління програмним проектом. Цей тип вимірювання насправді є непрямим вимірюванням. Коли комплексні атрибути вимірюються в термінах простіших суб-атрибутів, це вимірювання є непрямим [10].

Наскільки відомо, не було спроби розробки метрики для оцінки ефективності управління програмними проектами з використанням такого підходу. Тому в даній роботі пропонується метрична модель для такого вимірювання, щоб орієнтувати майбутні дослідження.

Модель показана нижче:

$$EMM = \sum_{i=1}^{n} w_i * m_i$$

У наведеній вище моделі m є показником, який виявляє відношення до метрики якості управління. Може бути n число метрик. Також може бути лише одна метрика, і в цьому випадку n дорівнює 1. Приклади таких метрик можуть включати продуктивність програміста, швидкість виправлення дефектів, певні зароблені метрики значення (EVM) і т.д. w_i - вага, пов'язана з певною метрикою, m_i. Такі ваги можуть знадобитися, оскільки різні метрики можуть бути пов'язані з результуючою метрикою управління якістю інакше. Тому у кожної метриці повинна бути вага, щоб можна було виміряти ефективність управління проектом за допомогою метричної моделі. Розробка ефективності управління або метрики якості для програмних проектів з використанням такого підходу вимагає значних подальших досліджень на основі емпіричних досліджень.

1.3.3 Оцінювання на основі моделі

У цьому підході показники ефективності або якості управління виводяться з моделей управління проектами програмного забезпечення. В даний час цей підхід також є концептуальним і не існує прикладів. Не було жодної спроби виміряти ефективність управління програмними проектами на основі моделі управління проектами.

Досить довгий час дослідники були зосереджені на розробці методологій життєвого циклу розробки програмного забезпечення. Існує безліч прикладів таких методологій, як Waterfall, Spiral, Win-Win, швидке створення прототипів, гнучкі розробки, SCRUM тощо.

У червні 2005 року Ініціатива управління бізнес-процесами (BPMI) та група управління об'єктами об'єднали свою діяльність і сформували Бізнес-модель та

інтеграцію Task Force. Вони розробили різні стандартні пропозиції для різних поглядів на управління процесами, наприклад, Модель бізнес-мотивації (BMM) і визначення бізнес-процесів Metamodel (BPDM). Навіть діаграми Ганта і PERT та схеми CPM (Critical Path Analysis) є моделями процесу, а розвиток діаграм Ганта датується 1910-ими роками [11]. Для того щоб краще зрозуміти, як можна оцінювати успіх проекту завдяки діаграмам Ганта та критичному шляху, нижче приведено більше інформації щодо кожного з варіантів.

Critical Path - це послідовність етапів, на яких менеджер визначає найменшу кількість часу, необхідну для виконання завдання з найменшою кількістю затримок. Отже, критичний шлях дійсно є найдовшим періодом часу, необхідним для виконання завдань проекту. Те, що зараз ми розуміємо як критичний шлях, було вперше розроблено в кінці 1950-х років Морганом Р. Уолкером і Джеймсом Е. Келі. Вони підійшли до цій ідеї приблизно в той же час, коли Booz Allen Hamilton і американський флот працювали в подібному дусі. Коріння критичного шляху можна знайти в деяких практиках DuPont на початку 1940-х років, і навіть у проекту Манхеттен [11].

З цього часу, метод критичного шляху був використаний в різних проектах, від будівництва, аерокосмічної галузі та захисту до розробки програмного забезпечення та продуктів, техніки, технічного обслуговування та інших. Проекти з взаємозалежною діяльністю можуть виграти від цього. Хоча оригінальна програма критичного шляху більше не використовується, підхід залишається незмінним.

Метод критичного шляху в управлінні термінами проекту в IT галузі — покрокова система управління завданнями окремого проекту. Визначивши всі дії на збудованому критичному шляху, можна адекватно оцінити майбутні витрати і скласти прогноз термінів проекту.

Для того, щоб побудувати діаграму критичного шляху, потрібно слідувати зазначеним крокам:

– визначити задачі;
– опрацювати їх послідовність за допомогою таблиці;

– скористатися діаграмою на манер, який зображено на рис. 1.3., і заповнити отриманими даними;
– оцінити час, який необхідно буде витратити на кожну із задач;
– якщо потрібно додати або навпаки видалити задачі, то треба оновлювати діаграму критичного шляху.

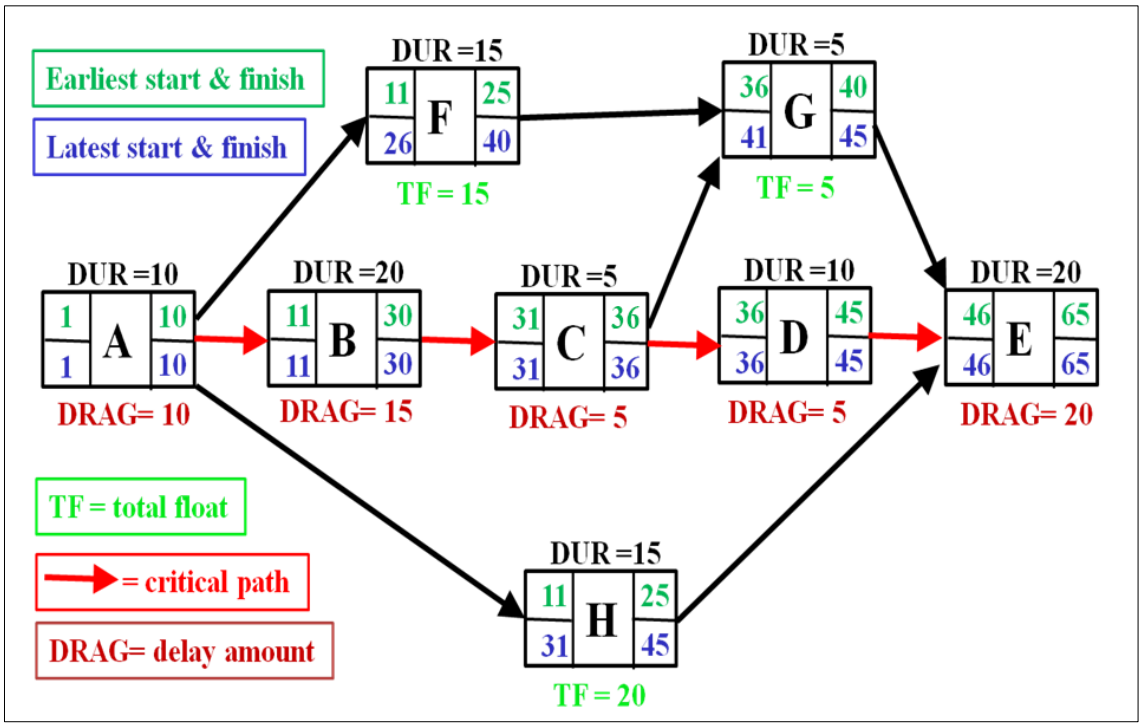

Рисунок 1.3. – Метод критичного шляху

Діаграма Ганта є одним із засобів планування та керування проектами. Вона використовується для ілюстрації плану, графіка робіт для будь-якого проекту.

Діаграма Ганта являє собою відрізки, розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, закінченню та тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями [6].

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується

вертикальна лінія, що відповідає моменту «сьогодні». Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу, наприклад хто працює над задачею, або що саме потрібно розробити.

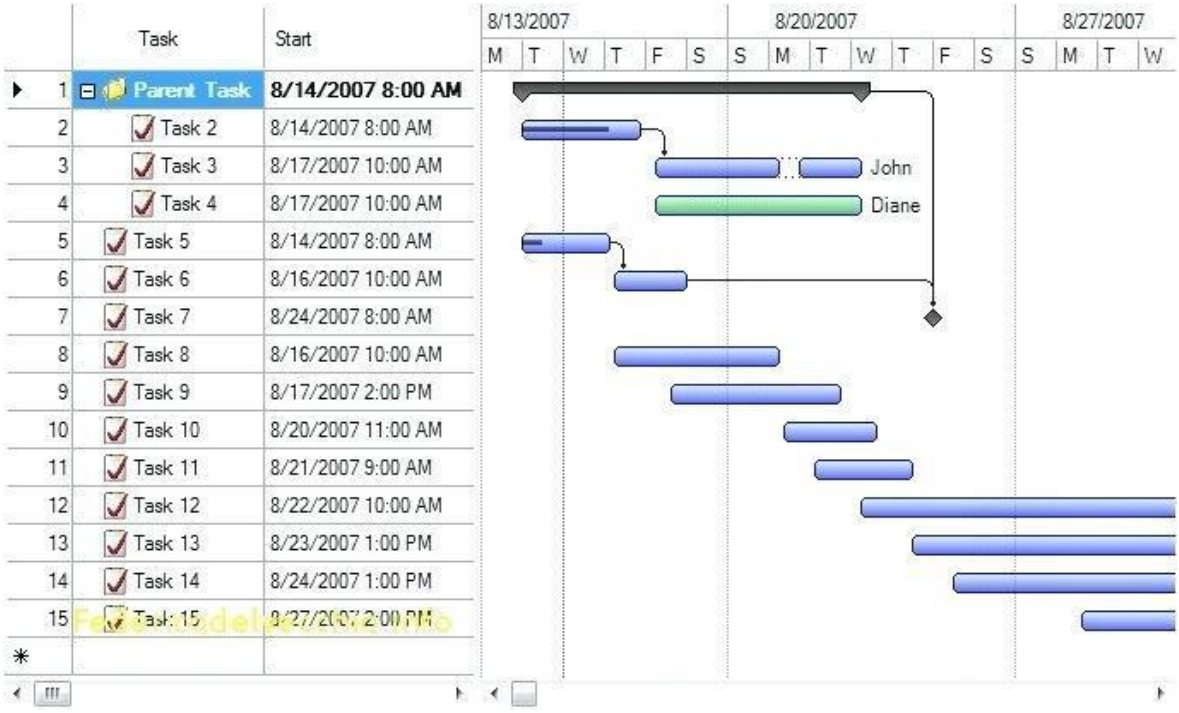

Рисунок 1.4. – Діаграма Ганта

Однак існує значна різниця між вищезгаданими PML (Process Modeling Languages) і моделями процесу. Хоча моделі процесу (такі як Gantt, PERT і CPM) отримали широке визнання в промисловості, вони є дуже технічними та формальними. Широке впровадження діаграм Ганта, PERT і CPM показує, що практикуючі керівники проектів хотіли б щоб ці моделі керування проектами були простіші. Дані моделі потрібно б ще вдосконалити, щоб була можливість їх використання в проектах, де часто змінюються вимоги до програмного проекту [12].

Життєздатність вимірювань ефективності для програмного забезпечення управління проектами вимагає отримання фактичних даних від проектів, яких у нас зараз немає. Моделі процесу розроблені для однієї конкретної мети і вони

зосереджуються лише на одному аспекті управління проектами. Наприклад, для прогнозування графіка проекту використовуються діаграми PERT. Однак, управління проектами програмного забезпечення має багато аспектів.

Успіх вимірювання на основі моделі буде сильно залежати від здатності проектування моделі управління проектом. Коли ці моделі управління проектами далекі від задовільного, то отриманий результат, ймовірно, буде теж незадовільним.

1.4. Мета і завдання роботи

Кількість досліджень та опитувань на глобальному рівні, що проводяться майже щороку про невдачі в управлінні проектами, багато [12]. Але після аналізу статичних даних, ще неможливо досконально зрозуміти:

- залежність успішності проекту від типу контракту (fixed-price, time and material, dedicated team);
- як розмір проекту та методологія управління впливають один на одного;
- який з етапів управління проектами найбільш ризиковий;
- які фактори часто забувають оцінити на етапі планування проекту;
- як змінюються проект в рамках успішності, якщо замовник проекту має досвід в IT-розробці або керуванні.

Тому в рамках цієї атестаційної роботи, будуть представлені результати дослідження успіху проекту, присвяченого конкретно українському IT ринку. Метою дослідження цієї роботи є формування емпіричних знань про характеристики успішних програмних проектів та визначення дій та факторів пов'язаних з успіхом програмних. Ця атестаційна робота буде поєднанням двох категорій вимірювання успіху проекту.

Практична значущість роботи полягає в розробці WEB-додатку, який допомагає менеджерам проекту оптимізувати процес керування проекту, керуючи ресурсами та витратами.

Розроблений додаток повинен мати можливість:

- додавати нові проекти, людські ресурси, прейскурант цін до системи;
- включати розподіл завдань, та додавати взаємозв'язки між ними;
- візуалізації діаграми Ганта для відстеження часу та статусу проекту;
- складання бюджету;
- автоматичне оптимальне планування ресурсів, завдяки розробленому алгоритму, який підбирає команду на проект, дивлячись на рівень позиції, стек технологій та дати проекту;
- відстеження ризиків проекту;
- планування дати релізів, беручи до уваги не тільки проектні рамки, але і святкові та вихідні дні в усіх проектних локаціях;
- планування дати закінчення проекту шляхом визначення найдовшого відрізка залежних завдань та вимірювання часу, необхідного для завершення їх від початку до кінця;
- скорочення тривалості проекту завдяки алгоритму скорочення часу однієї або декількох важливих завдань до меншого часу, ніж було заплановано;
- відстеження загальної статистики.

2 ОПИС ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

2.1 Побудова анонімного анкетного опитування

Запрошеними респондентами були українські розробники ПЗ, тестувальники, бізнес-аналітики та менеджери проектів і менеджери портфелей проектів. Опитування проходило онлайн за допомогою Google Forms щоб забезпечити анонімність респондентів та їхніх проектів. Анонімність відповідей було запроваджено для підвищення мотивації для надання точних та достовірних відповідей. Опитування використовувалося тільки для цілей цього дослідження. Всім учасникам опитування було запропоновано надати інформацію про характеристики та результати останнього завершеного IT-проекту, у якому вони брали участь. Вибір останнього проекту зменшує ризик того, що вибірка проектів буде зміщена до найбільш успішних або найбільших програмних проектів легко змінивши результати в той чи інший бік.

2.2 Результати опитування

В даному опитуванні 81 респонденти прийняли участь з 22 різних IT компаній України. Найбільшу частку з опитуваних зайняли розробники проектів (33.3%), другу і третю сходинку посіли менеджери проектів (27.3%) та менеджери портфелей проектів (18.2%).

У широкому сенсі, керування проектом – це керування за розробкою програмного забезпечення, процесом аналізу, створенням архітектури, написанням коду, створенням дизайну, тестуванням та реалізацією. Але насправді, це дещо складніше, оскільки керування проектом починається з вибору моделі ціноутворення на розробку програмного забезпечення, яку ринок сьогодні надає бізнесу. Однією метою цього опитування було дослідити залежність успіху

проекту від типу контракту. На сьогодні найпоширеніші 3 типи контрактів – fixed price, time & material та dedicated team.

В рамках цього дослідження, майже 50% це контракт з фіксованою ціною. Даний тип контракту передає чітке розуміння вимог, коли обсяг проекту задокументований у специфікації, UI / UX і критеріях прийнятності. На другому місці dedicated team – тип контракту, який підходить для клієнтів, які знають про складність і підводних каменів власної розробки проекту і тих, хто готовий інвестувати в поступовому і безперервному розвитку якості продукту. Менш ніж 15% отримав тип контракту time & material.

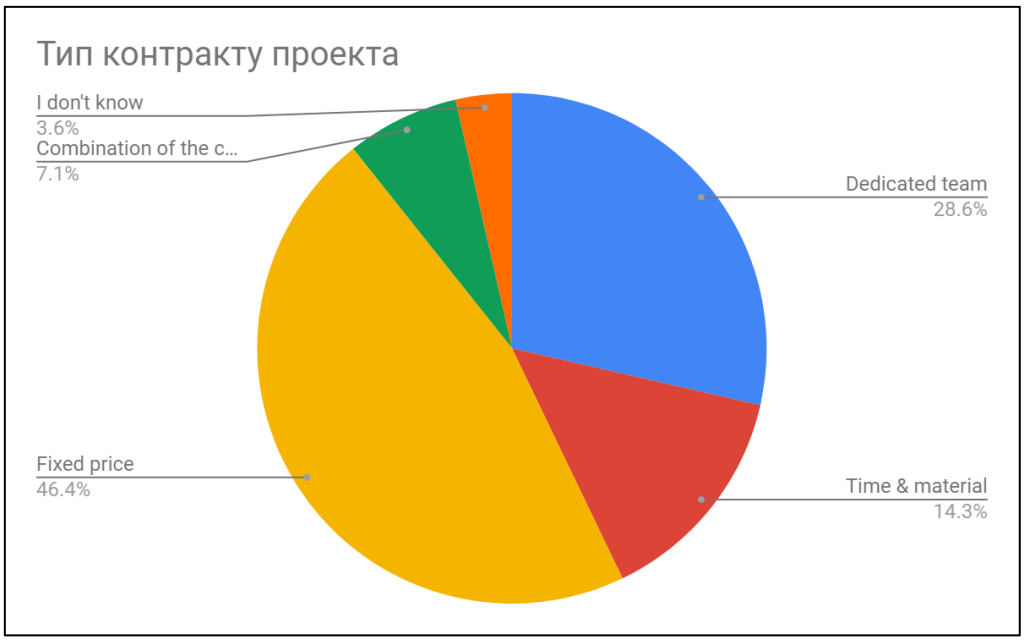

Рисунок 2.1 – Графік розподілення видів контракту у проектах

Багато з цих статистичних даних свідчать про велику кількість проектів, які в кінцевому підсумку завершуються з запізненням і надмірним бюджетом. Коли виконано опрацювання цих різноманітних досліджень, можна швидко співвідносити багато факторів провалу з однією змінною в проекті і це розмір проекту. Говорячи про великі та малі проекти, обговорюються фактори складності, тривалості та витрат. Тому в межах проведеного опитування респондентам було запропоновано оцінити по 5-ти бальній шкалі наскільки успішний проект, над яким вони зараз працюють з точки зору контролю якості,

бюджету, наскільки проект розробляється за планом (вчасно) та наскільки розроблена функціональність співпадає з вимогами клієнта. До результатів було обрано проекти, які тривали більше ніж 1 рік (54 проекти). Гарна новина в тому, що жоден з респондентів не обрав найгірший результат – 1, коли оцінював проект з точки зору успішності. Але найбільший відсоток успішності займає якість продукту. З боку часових та бюджетних обмежень, менеджерам проектів ще потрібна допомога в точності оцінювання. Це дослідження лише підтвердило відносно легке припущення про те, що більший проект буде важче завершити успішно, ніж менший проект.

Таблиця 2.1 – Результати опитування щодо успішності проекту

	5 (відмінно)	**4**	**3**	**2**	**1**
Проект розроблено в межах бюджету	28%	24%	40%	8%	0%
Проект розроблено вчасно	30%	28%	28%	14%	0%
Якість проекту	53%	35%	12%	0%	0%
Розроблена функціональність співпадає з вимогами клієнта	45%	39%	11%	5%	0%

Згідно з даними опитування, поширені причини невдач у проектах включають різноманітні фактори – від зміни пріоритетів організації (40%) до обмеженості ресурсів (20%). На деякі з цих факторів керівник проекту не може вплинути, але з іншими йому під силу впоратися. Надалі було наведено типи робіт в межах розробки проекту найчастіше забувають врахувати на етапі планування:

– час, на адаптацію в проекті нових співробітників;

– налаштування необхідного програмного забезпечення та отримання облікових даних;
– розгортання проекту на виробництві;
– уточнення вимог проекту;
– покращення продуктивності розробленого продукту;
– проектні комунікації в команді та зустрічі з замовником;
– стабілізація проекту та виправлення дефектів;
– передача проектних знань;
– створення технічної документації.

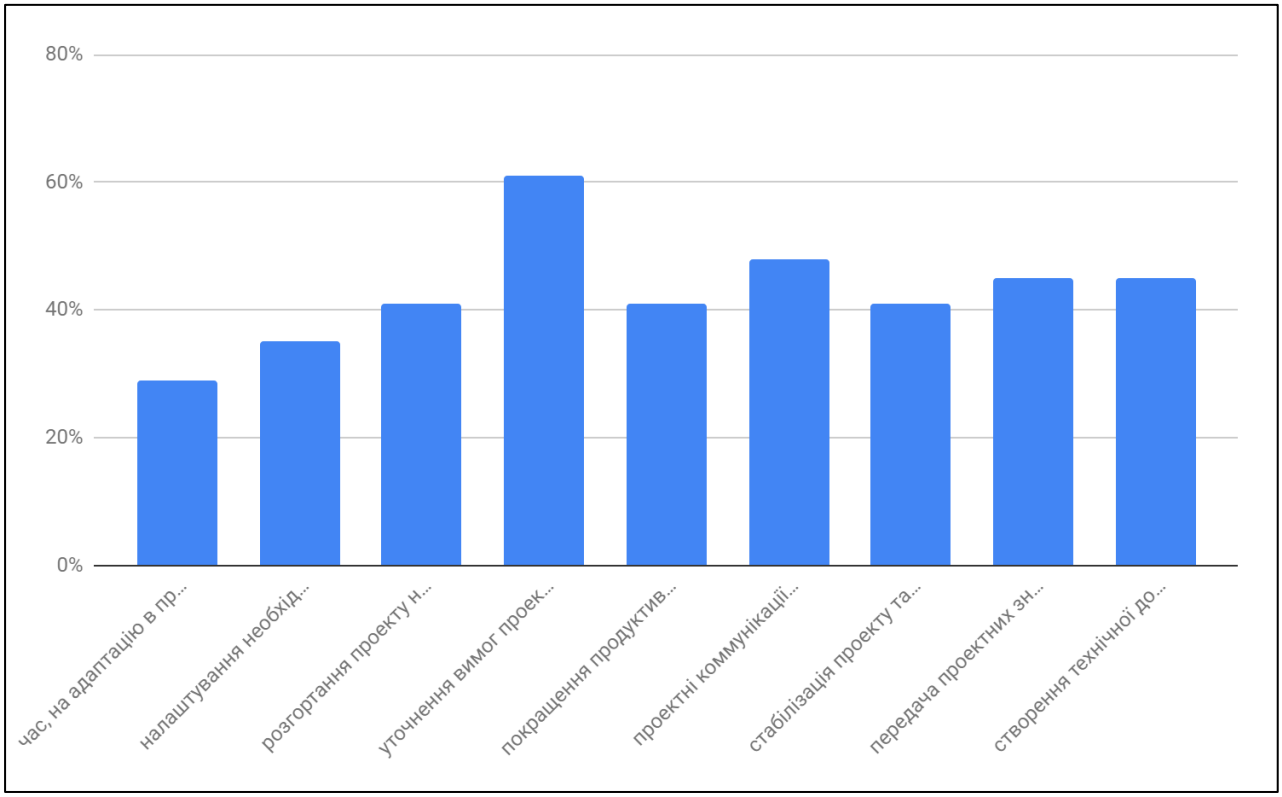

Рисунок 2.2 – Роботи, які найчастіше не враховують на етапі планування

В більш ніж 60% проектів не враховується час на уточнення вимог проекту. Незважаючи на те, наскільки чітко описані вимоги до завдання, завжди виникають питання на етапі розробки продукту, тому щоб більш ефективно спланувати роботу команди та спрогнозувати дати закінчення розробки чи тестування, потрібно на етапі планування додавати час на уточнення вимог до проекту.

На сьогодні гнучкі методи керування проектами стали трендом останніх кількох років, підхід Scrum є найпопулярнішою (43%) гнучкою методологією для виконання складних завдань. Наступні за популярністю після нього – Lean & Test Driven Development (11%) та eXtreme Programming (10%). Хоча гнучкі організації збільшують прибутки на 37% швидше й отримують на 30% більше доходів, ніж інші компанії, проекти все одно потрібно оцінювати детально. У багатьох організаціях існує переконлива думка про те, що "тільки робота за ноутбуком - це робота". Для розробників це схоже на те, що «тільки кодування - це робота». Ці переконання часто підкріплюються, класифікуючи “роботу ноутбука” як оплачувану роботу, тоді як зустрічі не підлягають оплаті. Основне припущення полягає в тому, що ми розробляємо проект тільки тоді, коли сидимо за нашими ноутбуками, і що ми повинні максимізувати цей час.

Ідея, що «тільки робота стоїть за моїм ноутбуком», є старомодною. Розробка проекту не є рутинною роботою. Команди часто мають комунікації про те, в якому стані вони знаходяться, щоб зрозуміти, що відбувається, і подумати про те, який наступний крок має сенс. Ці розмови необхідні і цінні. Хоча це може виявитися неефективним проводити час з усією командою в кімнаті (це 3 години з 6 людьми; 18 годин не оплачуваної роботи), робота буде більш ефективною в результатах, які вона генерує. Зрештою, творчість, мудрість і досвід кожного використовується для досягнення спільного розуміння, яке здійснюють усі. Тому коли вже усі знають скільки часу мінімально часу витрачати на комунікацію, дуже соромно отримувати результати, що у 45% проектні комунікації не враховуються на етапі планування проекту.

Ще одними із факторів успіху керування проектів, це наскільки часто команді потрібно виходити на роботу у вихідні або й навіть у святкові дні, щоб встигнути за проектним планом.

Результати опитування лише підтвердили мою думку. 43% серед респондентів виходять на роботу у вихідні раз на місяць. Майже 37 відсотків перероблять у вихідні тільки 2-3 рази на рік. Але є і гарна новина, майже 17 %

серед респондентів, ніколи не були змушені затримуватись на роботі або виходити на вихідних.

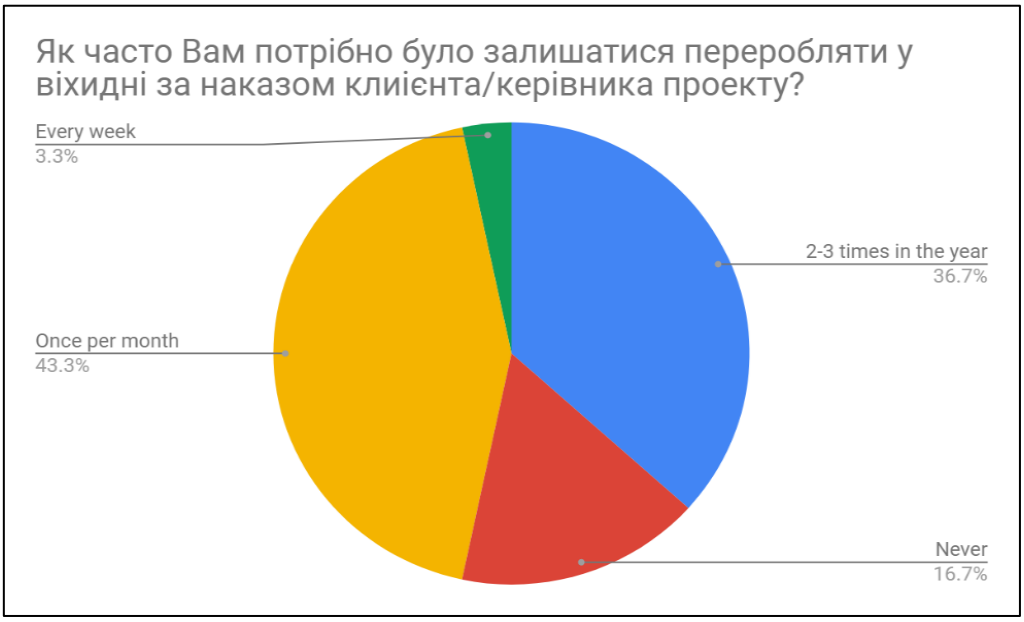

Рисунок 2.3 – Інфографіка роботи у вихідні дні

Щодо роботи у святкові дні, результати були більш кращими. Майже 44% ніколи не працювали за наказом керівника проекту у національні вихідні дні. Але приблизно така ж сама частина респондентів (41%) 2-3 рази на рік повинна була працювати у святкові дні. Всі інші (14%) лише один раз залучаються до роботи у неробочі дні.

Щоб виявнити чи впливає на успіх проекту компетентність клієнта в IT сфері було додано наступні два питання до анкети: «Скільки було IT-компетентності у клієнта?». Можливі відповіді: «Дуже багато», «Багато», «Деяка», «Мало» і «Дуже мало». Гіпотеза полягала в тому, що чим більше знань у клієнта в IT галузі, тим більш успішним буде проект з точки зору співпадання розробленого продукту до клієнтських очікувань від проекту. Наш аналіз відповідей показав, що було лише невелике зростання, на 7%, у показниках успішності для клієнтів у категорії «Компетентний клієнт» та зниження рівня успішності, на 3% для тих проектів, де була вибрана категорія «Дуже мало IT компетенції». В цілому, результати показують, що компетентний клієнт може

відігравати важливу роль для успіху проектів програмного забезпечення, і навіть більше, щоб уникнути невдачі проекту.

Маючи на увазі визначення успіху проекту, ми запитали респондентів, які саме елементи вони сприймають як найбільший внесок у успіх проекту. Вони вказали, що визначення та узгодження чітких вимог проекту (адекватної підтримки цілей бізнес-стратегії) є найважливішим елементом, що сприяє успіху проекту. Ці головні характеристики, що сприяють успіху проекту, підтверджені респондентами, які займають ролі у проектах. Слід зазначити, що респонденти, які входять до складу виконавчого керівництва (менеджери проектів або менеджери портфелей проектів), вважали «використання інструментів управління проектами» більш важливим, ніж «високоефективні команди» порівняно з іншими респондентами.

Базуючись на дослідженні та передовій практиці, в рамках цієї атестаційної роботи було розроблено WEB-додаток, який би дозволяв менеджерам проектів автоматично слідкувати за станом проекту та оптимізувати процес розробки.

3 АРХІТЕКТУРНА МОДЕЛЬ WEB ДОДАТКА

3.1 Сучасні способи організації архітектури

Веб-застосування складають основну частину сучасних застосувань. У епоху, коли Інтернет став доступним майже кожному у будь-якій точці планети, розроблені програмні веб-застосування складають основу сучасних продуктів. Зараз неможливо уявити компанію, яка б не мала власного сайту або веб-застосунку, що дозволяє отримати доступ до основних функцій продукту у будь-який час.

На тлі цього технології розробки веб-додатків розвиваються неабиякими темпами, створюються нові способи організації, технології, мови програмування. Однак основою всіх сучасних додатків є архітектура. Від вибору правильної архітектури залежить не тільки швидкість і зручність доступу та розробки, а й весь успіх продукту.

Під час роботи біло проведено ознайомлення з сучасними способами організації архітектури веб-додатків, їх порівняння і аргументований вибір архітектури для розробки програмного продукту.

Спочатку, найбільш використовуваним підходом була монолітна архітектура, що дозволяє будувати величезні додатки, що складаються з безлічі функціональних частин в рамках однієї великої структури. Одним з основних переваг цього підходу є незалежність від будь-яких сторонніх ресурсів на користь організації всього продукту в рамках однієї програми.

На зміну монолітною архітектури прийшов мікросервісний підхід. У ньому використовується абсолютно інша стратегія організації. В даному підході широко застосовуються різні сторонні сервіси, програма містить кілька невеликих компонент, кожен з яких є окремим веб-додатком. Ці компоненти взаємопов'язані між собою. Кожному компоненту даної архітектури відповідає певна роль у функціонуванні програми. Основною перевагою даного підходу є зручність і

гнучкість додатки, що дозволяє вдаватися до масштабування окремих компонент. На даний момент даних підхід є найбільш популярним.

Serverless (безсерверна архітектура) є активно набираючим популярність підходом до організації архітектури веб-додатків. Взявши всі переваги мікросервісного підходу, без серверний підхід пішов далі і приніс відсутність будь-якої апаратної частини в організацію веб-додатків [13].

Serverless має як ряд переваг, так і деякі недоліки. Перевагами даного підходу є:

– відсутність апаратної частини - серверів;
– відсутність прямого контакту і адміністрування серверної частини;
– практично безмежне горизонтальне масштабування додатку;
– оплата тільки за використаний час.

До недоліків даного підходу можна віднести:

– відсутність чіткого контролю за виконанням програми;
– відсутність цілісності додатка;
– холодний старт може займати кілька секунд.

В цілому даний підхід є досить підходящим для вирішення завдань певного роду в рамках великого додатка або для початкових етапів розробки. Особливий наголос в даному випадку ставиться на незалежність від серверної інфраструктури і безмежний приріст продуктивності в автоматичному режимі.

Одночасно з безсерверною архітектурою вводиться поняття FaaS (Function-as-a-Service, функція як сервіс), що характеризується підходом до розбиття сервісів на найменші функціональні частини спільно з відокремленням цих частин в окремі функціональні одиниці. Наприклад, веб-сервер розбивається на окремі шляхи і для кожного такого шляху створюється окрема функція, яка викликається при кожному зверненні до шляху. Також, безсерверні системи набагато простіше підтримувати, більш того, будь-яка оптимізація продуктивності FaaS-додатки автоматично знижує операційні витрати [14].

У роботі був обраний даний підхід не тільки для організації веб-сервера прив'язаного до призначеного для користувача інтерфейсу, але і для створення

непостійній архітектури збору даних з різних джерел. Основними причинами вибору цього напрямку є можливість створення серверної архітектури на обмежений час без постійного використання і можливість стабільної роботи незалежно від навантаження.

3.2 Огляд сучасних загроз безпеці веб-застосувань

Сучасний світ несе в собі безліч загроз і потенційних небезпек на кожному кроці. В епоху сучасного інтернету, який став невід'ємною частиною нашого життя, складно знайти причину для відчуття безпеки. Інформація, яка надається у всесвітній мережі, є одним з найбільш важливих багатств сучасного світу [15]. У той же час кіберзлочинність є дуже високорозвиненою, адже майже кожна компанія має свій веб-сайт, а значить і потенційне джерело для витоку інформації. На тлі розвинутої кіберзлочинності з'являється потреба у фахівцях з безпеки та створенні систем з високим рівнем захисту від різних атак. Сучасний веб-розробник зобов'язаний бути знайомий з векторами атак, що найбільш зустрічаються, як на сервер так і на призначений для користувача інтерфейс.

У рамках роботи було проведено ознайомлення з сучасними векторами атак на веб-застосунки, механізмами їх дії, можливими наслідками та способами захисту від них.

За даними відкритого проекту з безпеки веб-застосунків (OWASP) основними ризиками в веб-додатку [15] зазначені нижче.

Впровадження коду–ін'єкції таких типів як SQL, NoSQL, OS, і LDAP. Ін'єкції виникають, коли ненадійні дані відправляються інтерпретатором як частина команди або запиту. Ворожі дані зловмисника можуть обманути інтерпретатор для ненавмисного виконання ним команд або доступу до даних без належної авторизації.

Некоректна автентифікація і управління сесією. Функції програми, пов'язані з автентифікацією і управлінням сеансами, часто реалізуються неправильно, що дозволяє зловмисникам скомпрометувати паролі, ключі або токени сесії, або використовувати інші недоліки реалізації, щоб тимчасово або постійно використовувати ідентифікатори інших користувачів.

Витік чутливих даних. Багато веб-застосунків не захищають конфіденційні дані, такі як фінансові або медичні дані. Зловмисники можуть красти або модифікувати такі слабо захищені дані для здійснення шахрайства: з кредитними картами, крадіжки особистих даних або інших злочинів.

Впровадження зовнішніх XML-сутностей. Багато застарілих або погано конфігурованих XML-процесорів мають посилання на зовнішні XML-ресурси, вони можуть бути використані для розкриття загальних або внутрішніх файлів, сканування портів, віддаленого виконання коду.

Порушення контролю доступу. Обмеження на те, що дозволено користувачам, які пройшли перевірку автентичності, часто вже не виконуються належним чином.

Небезпечна конфігурація. Дана проблема пов'язана з використанням конфігурації за замовчуванням, неповної або специфічної конфігурації, відкритого сховища даних.

Міжсайтовий скрипт. Дані помилки виникають, коли програма включає недостовірні дані в новій веб-сторінці без правильної перевірки або оновлює існуючу веб-сторінку з наданими зловмисником даними, що дозволяє йому виконувати код в браузері, який може захоплювати призначені для користувача сеанси.

Використання компонентів з відомими вразливостями. Компоненти, такі як бібліотеки, фреймворки та інші програмні модулі, працюють за тими ж правами доступу, що й застосунок. Недостатній моніторинг. Дана проблема в поєднанні з відсутньою або неефективною інтеграцією з реагуванням на інциденти дозволяє зловмисникам продовжувати атакувати систему різними способами: підробляти, витягувати або знищувати дані.

В роботі веб-безпеці приділено окрему увагу в силу наявності роботи з конфіденційними даними користувача, такими як особисті дані користувача, його проектами, а так само роботі з фінансами.

3.3 Огляд моделі веб-додатка

Беручи до уваги усі вищезазначені дослідження та аналіз сучасних підходів, було вирішено створити програмну модель, яка б вирішувала проблеми моніторингу та мала б за основу безсерверну архітектуру із захистом від потенційних загроз. На рис. 2.1 можна побачити повну модель серверної частини веб-застосунку.

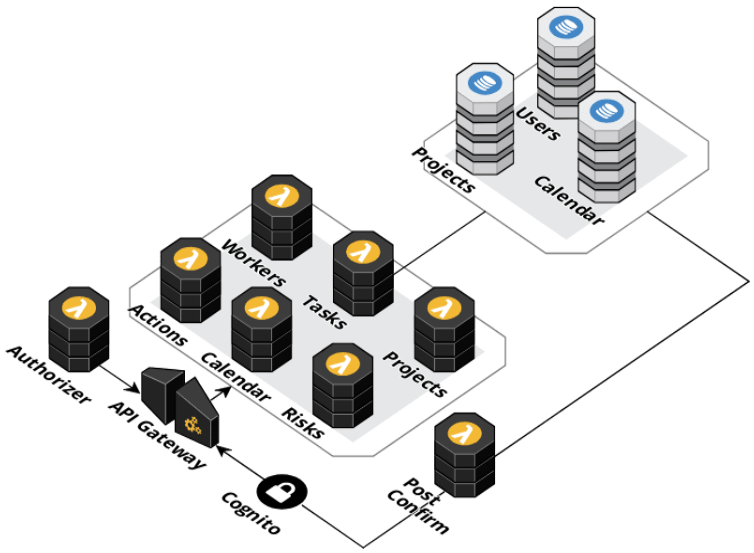

Рисунок 2.1 – Діаграма серверної частини веб-застосунку

Amazon API Gateway – це повністю керований сервіс для розробників, який спрощує створення, публікацію, обслуговування, моніторинг та забезпечення безпеки API в будь-яких масштабах [16]. Цей сервіс бере на себе всі завдання, пов’язані з прийомом і обробкою сотень тисяч одночасних викликів API, включаючи управління трафіком, авторизацію і контроль доступу, моніторинг та

управління версіями API. Він використовується як зовнішня точка доступу для будь-яких сервісів або веб-застосувань, що сховані за ним та має багато переваг.

По-перше, це низька вартість та ефективність. З Amazon API Gateway ви платите лише за виклики ваших API і вихідну передачу даних. Мінімальні та авансові платежі відсутні.

По-друге, продуктивність при будь-якому масштабі. Інтеграція з сервісом Amazon CloudFront, який було використано в роботі у візуальній частині, дозволяє використовувати при роботі з API Gateway переваги глобальної мережі периферійних розташувань, щоб забезпечити кінцевим користувачам мінімально можливу затримку при викликах API та отриманні відповідей. Також дозволяє регулювати трафік з використанням обмежень, допомагаючи серверним процесам витримувати різке зростання його обсягу. Крім того Amazon API Gateway допомагає підвищити продуктивність API шляхом кешування відповідей на виклики, що дозволяє уникнути повторних звернень до серверних систем.

Також це зручний моніторинг активності. Після розгортання API сервіс Amazon API Gateway дозволяє візуально відслідковувати виклики на адресу відповідних сервісів на панелі управління за допомогою Amazon CloudWatch. Надана інформація включає метрики продуктивності, а також статистику викликів API, затримку при передачі даних та коефіцієнт помилок.

Гнучкі налаштування безпеки – ще одна з переваг. API Gateway надає інструменти авторизації доступу до API і функцій контролю роботи сервісу. Для авторизації доступу до ваших API можна використовувати такі інструменти управління та забезпечення безпеки як AWS Identity and Access Management (IAM) і Amazon Cognito, що було використано в роботі. Також є можливість верифікувати підписані виклики API та використовувати AWS Lambda з метою перевірки вхідних запитів.

Хотілось би зазначити, що також добре це - API без серверів. API Gateway тісно інтегрований з AWS Lambda, що дозволяє створювати без серверні API, які потів використовуються мобільними та веб-додатками для виклику публічних сервісів. AWS Lambda запускає код у високопродуктивному

обчислювальному середовищі і виконує складні завдання і адміністрування обчислювальних ресурсів [16].

В роботі API Gateway виконує усі з вищезазначених дій за допомогою інтеграції з іншими сервісами Amazon.

AWS Lambda, що є представленою такими складовими як Authorizer Lambda, Workers, Projects, Post Confirm Lambda є основою для створення безсерверної архітектури. AWS Lambda – це сервіс безсерверних обчислень, що запускає програмний код у відповідь на деякі події та автоматично керує обчислювальними ресурсами, необхідними для його виконання. Його можна використовувати для розширення можливостей інших веб-сервісів Amazon за допомогою спеціальної логіки або для створення власних веб-сервісів, що використовують масштабованість, продуктивність та безпеку Amazon. Lambda запускає код у високопродуктивному обчислювальному середовищі і займається адміністративною підтримкою всіх ресурсів, включаючи обслуговування серверів і оперативних систем, розподіл продуктивності та автоматичне масштабування, установку програмного забезпечення і виправлення вразливостей, а також моніторинг коду та ведення логування. Розробник має тільки завантажити код.

Amazon Lambda має низку переваг. По-перше, не потрібно управління сервером Amazon Lambda дозволяє автоматично запускати програмний код без необхідності у виділенні серверів або управлінні ними. Достатньо написати програмний код і завантажити його в Lambda.

По-друге, це - безперервне масштабування. AWS Lambda автоматично масштабує веб-додаток, запускаючи виконання коду в відповідь на кожен тригер. Всі запущені коди виконуються паралельно, при цьому кожен тригер обробляється індивідуально, що забезпечує масштабування відповідно до робочого навантаження.

Також однією з переваг –це оплата з точністю до часток секунди. При роботі з AWS Lambda оплачуються кожні 100 мілісекунд виконання програмного коду і кількість його тригерів. Коли програмний код не виконується, оплата не потрібна.

В роботі Amazon Lambda автоматично виконує код у відповідь на HTTP запити через API Gateway (API Lambda), виконує роль додатку до існуючого сервісу аутентифікації (Post Registration Lambda) та верифікує підписані виклики (Authorizer Lambda). API Lambda виконує роль звичайного REST-сервісу, що дозволяє виконувати автентифікацію з допомогою сервісу Cognito та має доступ до сервісу збереження даних DynamoDB Storage. Post Registration Lambda є додатком до сервісу Cognito, що виконує автентифікацію, проте не може зберігати у собі деяку додаткову інформацію, потрібну для роботи веб-застосунку. Саме тому ця лямбда після реєстрації користувача додає деякі дані до нього у DynamoDB Storage, які потім будуть використані у подальшій роботі сервісу. Authorizer Lambda використовується для перевірки автентифікованих запитів та дозволяє контролювати до яких веб-шляхів є доступ у того чи іншого користувача.

Amazon Cognito є широко використовуваним сервісом для аутентифікації користувачів. Він дозволяє швидко і просто додавати можливості реєстрації, авторизації і контролю доступу користувачів в мобільні та інтернет-додатки. Amazon Cognito масштабується до мільйонів користувачів і підтримує авторизацію за допомогою соціальних постачальників посвідчень (Facebook, Amazon), а також постачальників корпоративних посвідчень на основі SAML 2.0.

Останнім компонентом, що є зв’язком між веб-частиною та частиною моніторингу є база даних DynamoDB. Amazon DynamoDB - це швидкий і гнучкий сервіс баз даних NoSQL. Він підходить для будь-яких додатків, які вимагають стабільної роботи з затримкою не більше декількох мілісекунд при будь-якому масштабі. Ця повністю керована хмарна база даних підтримує роботу на основі як документів, так і пар «ключ-значення».

Гнучка модель даних, стабільна продуктивність і автоматичне масштабування пропускної здатності роблять цей сервіс відмінною платформою для мобільних і інтернет-додатків.Одними з найбільш важливих переваг DynamoDB є:

Швидка і стабільна робота. Рішення DynamoDB стабільно і швидко працює в системах будь-якого масштабу в будь-якій області застосування. Середній час

обробки запиту на сервері становить не більше десяти мілісекунд. У міру зростання обсягів даних і підвищення необхідної продуктивності система DynamoDB забезпечує відповідність вимогам до пропускної спроможності і часу обробки запиту за допомогою технологій автоматичного розбиття на розділи і SSD в системах будь-якого масштабу.

Висока масштабованість. DynamoDB автоматично масштабує ресурси в бік збільшення або зменшення згідно зі збільшенням або зменшенням обсягу запитів додатків. Автомасштабування включено за замовчуванням, потрібно тільки вказати цільовий рівень використання. Фактичне споживання пропускної здатності безперервно відстежується в фоновому режимі за допомогою попереджень Amazon CloudWatch, а виділена пропускна здатність коригується при кожному відхиленні рівня використання від цільового значення. Якщо необхідно масштабувати додаток для обслуговування користувачів, що знаходяться по всьому світу, можна використовувати глобальні таблиці, які забезпечують автоматичне реплицирування даних в задані регіони AWS.

Повністю керований сервіс. DynamoDB - це повністю керований сервіс нереляційних бази даних в хмарі. Можна просто створювати таблицю бази даних і задавати вимоги до цільового рівня використання для автомасштабування, після чого сервіс працює в автоматичному режимі. Не потрібно турбуватися про управління базою даних: про виділення апаратного і програмного забезпечення, налаштування і конфігурації, оновлення програмного забезпечення, використання нерозподіленого кластера бази даних або поділі інформації на кілька інстансів в міру зміни масштабів системи.

DynamoDB також забезпечує відновлення на заданий момент часу, резервне копіювання і відновлення для всіх використовуваних таблиць, що дозволяє задовольнити корпоративним і нормативним вимогам.

Подієво-орієнтоване програмування. DynamoDB інтегрований з сервісом AWS Lambda для створення тригерів, які дозволяють розробляти програми, автоматично реагують на зміну даних.

Точний контроль доступу. DynamoDB інтегрований з сервісом Identity and Access Management (IAM), що забезпечує повний і точний контроль доступу всіх користувачів організації. Ви можете призначити кожному користувачеві унікальні безпечні дані для доступу і дозволу, а потім відстежувати доступ кінцевих користувачів до всіх сервісів і ресурсів [16].

Гнучкість. DynamoDB підтримує роботу зі структурами даних на основі як документів, так і пар «ключ-значення», завдяки чому можна вибрати оптимальну архітектуру з урахуванням особливостей свого застосування.

У роботі DynamoDB зберігає 3 таблиці даних: Users, Calendars та Projects. Таблиця Users використовується для збереження даних про співробітників, для використання їх як ресурси проекту. Схему бази Users можна побачити в табл. 3.1.

Таблиця 3.1 – Схема бази даних Resources

Назва поля	Тип	Опис
UserId	String	Ідентифікатор співробітника.
First Name	String	Ім’я співробітника у системі.
Last Name	String	Прізвище співробітника
Location	String	Офіс в якому знаходиться співробітник
Main Skill	String	Головний навичок
Role	String	Роль у проекті
Seniority level	String	Рівень професіоналізму
Skills	String	Якими навичками володіє співробітник
Vacations	{}	Відпустки
Vacation start date	String	Дата початку наступної відпустки
Vacation end date	String	Дата закінчення наступної відпустки
Project	{}	Проект, де працює співробітник
Project Id	String	Ідентифікатор проекту
Project end date	String	Дата закінчення проекту

Таблиця Users використовується для збереження даних про співробітників компанії.

Таблиця Projects використовується для збереження даних про проекти та сумісні структури даних. Схему бази можна побачити в табл. 2.3.

Таблиця 3.2 – Схема бази даних Projects

Назва поля	Тип	Опис
ProjectId	String	Ідентифікатор проекту.
Project Name	String	Назва проекту
Start date	String	Дата початку
End Date	String	Дата закінчення
Project duration	Number	Довжина проекту
Project Cost	Number	Бюджет проекту
Overhead cost per day	Number	Вартість за накладні витрати за день
Risks	[{}]	Ризики
Risks description	String	Опис ризика
Risks probability	Number	Ймовірність ризику
Risks impact	String	Вплив ризика на проект
Risks mitigation action	String	Заходи щодо зменшення ризику
Risks resolution cost	Number	Вартість за вирішення ризиків
Tasks	[{}]	Технічне завдання
Task summary	String	Резюме завдання
Task work type	String	Який ресурс потрібен для виконання роботи
Task type	String	Тип завдання
Task components	String	Компоненти завдання
Task priority	String	Пріоритет
Task assignee	String	Виконавець завдання
Task labels	[String]	Помітки
Task description	String	Опис роботи
Task estimates min	Number	Мінімальний час на виконання
Task estimates max	Number	Максимальний час для виконання
Task logged time	Number	Час виконання завдання
Task cost	Number	Вартість виконання завдання
Task crash cost	Number	Вартість зменшення часу на виконання завдання
Task reporter	String	Автор завдання
Task comments	String	Коментарі
Task sprint	String	Спринт
Task status	String	Статус завдання
Task predecessor	[String]	Залежне завдання
Rate card	String	Прейскурант

В даній таблиці зберігається інформація про завдання проекту, ризики, та ціни на ресурси, та інші додаткові деталі проекту, які можуть впливати на його успішність.

Іншою частиною веб-додатку є календар. Схема цієї бази в табл. 3.3

Таблиця 3.3 – Схема бази даних Calendar

Назва поля	Тип	Опис
Location	String	Локація офісу
Holidays	[{}]	Святкові вихідні дні
Holidays date	String	Дата святкових вихідних
Holidays description	String	Додаткова інформація

Іншою складовою є CloudWatch. Amazon CloudWatch - це сервіс моніторингу хмарних ресурсів AWS і додатків, які запускаються з їх допомогою. Amazon CloudWatch можна використовувати для збору і відстеження метрик, накопичення та аналізу файлів журналів, створення попереджень, а також автоматичного реагування на зміни ресурсів AWS. Можна використовувати Amazon CloudWatch для отримання зведеної інформації про систему, що включає в себе інформацію про використовувані ресурси, продуктивності додатків і загальний стан системи. Ці дані застосовуються для оперативного реагування та забезпечення стабільної роботи додатків.

Основними перевагами є:

– моніторинг метрик користувача. Можна додавати призначені для користувача метрики, які генеруються власними додатками, в Amazon CloudWatch для моніторингу, скориставшись простим API-запитом. Також додавати і зберігати метрики, які мають важливе значення для продуктивності застосування, дозволяють усувати неполадки і стежити за динамікою;

– моніторинг та зберігання лог-файлів. Можна використовувати логи CloudWatch для моніторингу та усунення неполадок в роботі систем і додатків, використовуючи існуючі лог-файли наявних систем, додатків і користувальницьких метрик. Можна відправляти лог-файли використовуваних систем, додатків і призначених для користувача даних в логи CloudWatch для їх моніторингу в режимі, близькому до реального

часу. Це допомагає краще зрозуміти потреби систем і додатків, а також дозволяє краще управляти ними, при цьому можна зберігати логи для подальшого доступу в надійному і недорогому сховищі;

– відображення графіків і статистики. Панелі управління Amazon Cloudwatch дозволяють створювати графіки ресурсів AWS, придатні для багаторазового використання, а також настроюються метрики. Це дозволяє оперативно вести моніторинг робочого стану і миттєво виявляти проблеми. Дані метрик зберігаються протягом п'ятнадцяти місяців. Це дозволяє переглядати як найсвіжіші дані, так і історію змін. Amazon CloudWatch може завантажити всі метрики облікового запису для перегляду і виведення графіків за допомогою консолі управління AWS, в тому числі логи, метрики ресурсів AWS і зазначені розробником метрики додатків;
– подієво-орієнтоване програмування. CloudWatch дозволяє створювати події, які потрібні розробнику на основі технології Cron jobs. Це дозволяє виконувати деякі дії на задньому плані, які не потребують участі користувача системи.

Вищеописаний сервіс Amazon Lambda представлений у частини API шістьма компонентами, кожен з яких виконує різні задачі. Workers відповідає за роботу з працівниками та командами, Project надає можливість працювати з проектами, Tasks та Risks працюють з сумісними проекту задачами та ризиками, Calendar дає доступ до роботи з різними календарями локацій та святами, а Actions виконує такі дії як розрахунок критичного шляху, крешів, підбір команди, тощо.

Використана система сервісів дозволяє надавати користувачу гнучкий і швидкий доступ до даних у будь-який час.

4 ОСНОВНІ РЕЗУЛЬТАТИ РОБОТИ

4.1 Загальний опис програмних частин

Основним результатом роботи є готова до використання система, що дозволяє проводити моніторинг вартості продукції без безпосередньої участі користувача. Як було описано вище систему можна поділити на такі основні частини: серверна та веб частина.

Для реалізації серверної частини було обрано мову JavaScript, а саме Node.JS. Це дає кілька переваг перед іншими варіантами. По-перше, Node.JS добре пристосовано до великої кількості одночасних запитів. Базуючись на таких речах як Event Loop та асинхронне виконання коду, серверна частина на Node, вдало реагує на будь-який запит, не ставить користувачів у так звану чергу. Також маючи веб-частину на JavaScript можна отримати гарний зв’язок між нею та серверною частиною заснований на рівності технологій. Також Amazon обрав Node.JS як одну з основних мов для реалізації своїх сервісів.

Для полегшення роботи був взят Serverless Framework, що полегшує розгортання проектів Node.js на AWS Lambda і API Gateway. Для виконання веб-частини було обрано React та MobX.

React – це найпопулярніша бібліотека для створення інтерфейсів користувача, заснована на мові JavaScript. React дозволяє створювати інтерактивні інтерфейси користувача безболісно для кожного стану у програмі та ефективно оновлює та відтворює лише потрібні компоненти, коли дані змінюються. Основною концепцією є компонент, який керує своїм станом. Розробник створює інтерфейс з компонентів, що розроблені кимось або може створювати свої.

MobX – це передбачуваний контейнер для додатків JavaScript. Він допомагає писати програми, які працюють послідовно зі зміною стану, працюють у різних середовищах (клієнтському, серверному або мобільному) та легкі для тестування.

Для кращого розуміння вимог до програми, та варіанти використання продукту була розроблена Use Case діаграма.

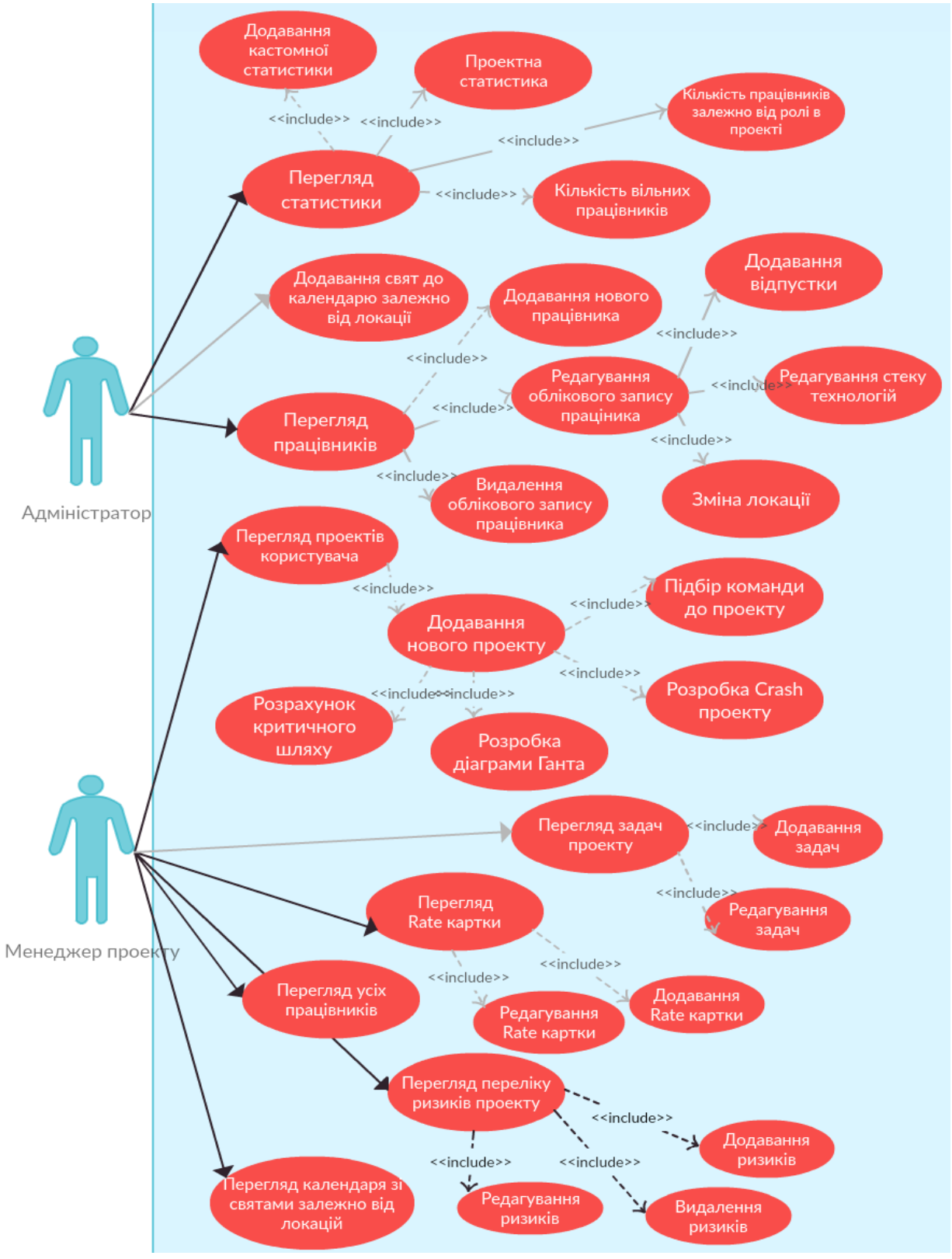

Рисунок 4.1 – Use-case діаграма

4.2 Розроблення алгоритму оптимального підбору команди до проекту

Одним із головних факторів успішності керування проектом, це високопродуктивна та кваліфікована команда. Побудова високопродуктивних команд не є легкою роботою. Для встановлення високопродуктивної культури праці та розробки високопродуктивних команд потрібно керівникам проектів дотримуватися наступних кроків.

Крок перший. Створювати мотивуючи і складні цілі, пов'язані з виконанням завдань для членів команди, особливо, коли члени мають високу мотивацію досягнення.

Крок другий. Розвивати особисті лідерські компетенції та заохочувати індивідуальну ініціативу.

Крок третій. Наймати співробітників до команди, які мають найкраще поєднання навичок і здібностей та особистих якостей, які дозволяють ефективно і гармонійно взаємодіяти з іншими людьми.

Крок четвертий. Приділяти увагу розвитку членів команди, щоб вони могли застосовувати інноваційні методи та нові підходи до роботи для досягнення цілей команди.

В межах цієї атестаційної роботи, пропонуємо рейтинговий алгоритм підбору команди. Таким чином ми спробуємо оптимізувати 3 крок по створенню високопродуктивних команд.

Головна ідея алгоритму полягає в тому, що кожен член команди підбирається на основі помноження двох рейтингів (рейтингу за технологіями проекту та рейтингу, який проставляється за підсумками попередніх проектів). Рейтинг за технологіями проекту знаходимо завдяки перетину серед множини технологій які потребуються на проекті, та вмінь та навичок, які є у співробітника.

Спершу при пошуку члена команди, робиться фільтрація серед усіх співробітників по наступним критеріям:

– рівень сеньорності (Senior, Middle, Junior, Trainee);

– кваліфікація (Project Manager, Developer, QA, Designer, та інші);
– первинний навик чи мова програмування (Java, .Net, C++, Manual QA, Python, та інші);
– на проекті людина, чи без проекту;
– чи має відпустку на момент старту проекту;
– в якій локації знаходиться кандидат.

Серед кандидатів, які пройшли усі етапи фільтрації починається рейтинговий відбір. Чим більше співпадає технологій та навичок, які знає кандидат та які потрібні на проекті – тим більше в нього рейтинг. Взагалі, якщо кандидат ідеально підходить до вакансії, то він отримує максимальний рейтинговий бал – 1.

Після того як проект закінчується, менеджер проекту оцінює кожного співробітника за наступними критеріями:

– робота в команді: наскільки людина вміє працювати та знаходити спільну мову з іншими членами команди;
– комунікація зі стороною замовника та всередині команди;
– володіння англійською мовою;
– цілеспрямованість і гнучкість;
– стресостійкість і відповідальність;
– вміння швидко адаптуватись до нових умов проекту;

Після того, як обидва рейтинги отримано, система помножує їх та віддає результат. Це зроблено за для того, щоб підібрати найефективнішу команду на проект. Якщо керівник погоджується, то нова команда буде обрана на новий програмний проект.

Нажаль недостатньо тільки професіональних знань та навичок, щоб отримати роботу, особисті якості, які дозволяють ефективно працювати в команді відіграють теж досить важливу роль. Тому даний алгоритм враховує ці якості теж до підбору найкращої команди на проект.

4.3 Реалізація інтерфейсу WEB-застосунку

Інтерфейс користувача було виконано з використанням підходу Material Design. Material Design – це принцип дизайну сайтів, програмного забезпечення та застосунків, а також правила дизайну інтерфейсів від компанії Google. Ідея дизайну полягає в інтерфейсі, поведінка і вигляд якого наслідують правила поведінки і вигляду паперових карток в реальному житті. Material design за допомогою системи управління тінями створює візуальну ілюзію простору між за стосунком та екраном пристрою.

Вигляд основної сторінки можна побачити на рис. 4.2

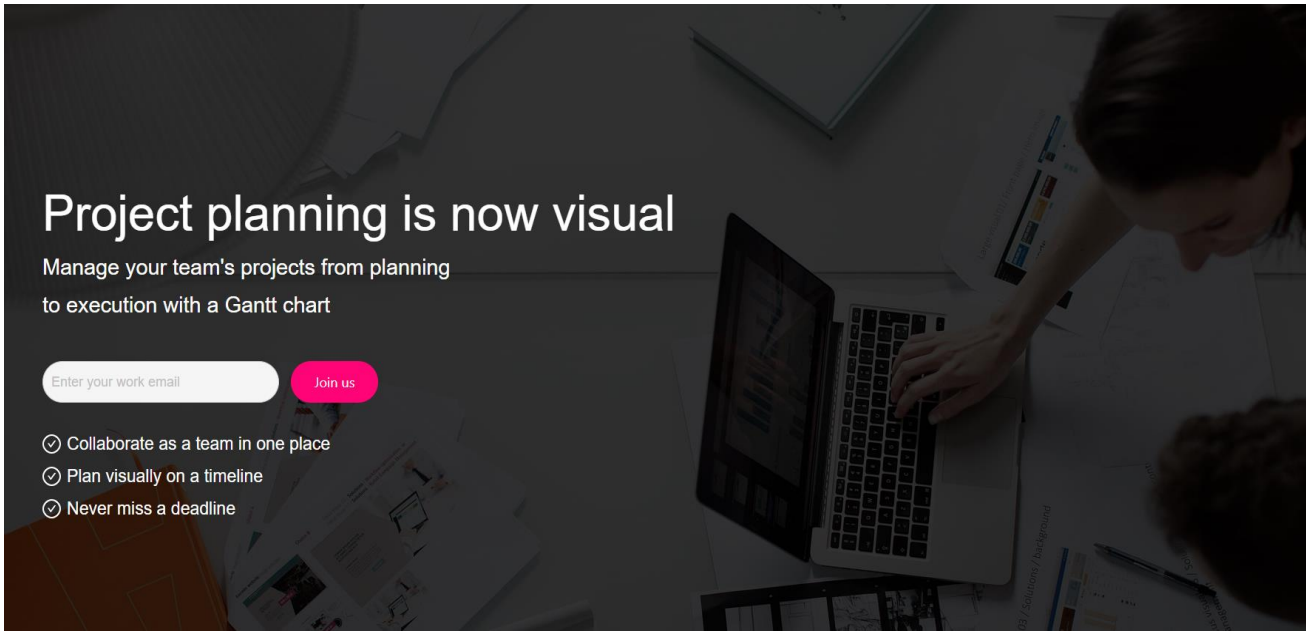

Рисунок 4.2 – Основна сторінка застосунку

Основна сторінка веб-застосунку виконана у мінімалістичному стилі та носить базовий, не функціональний характер. Головна мета, це надати відомості користувачеві, які головні функції та фішки є в даному веб-додатку, та як можна використовувати додаток. На рис 4.2 можна побачити кнопку “Join us”, яка дозволяє перейти до реєстрації та автентифікації користувача. Реєстрація нового користувача відбувається двома засобами: через заповнення персональних даних,

або можна виконати авто-реєстрацію через соціальні мережі: Facebook, Twitter, Google. На рис 4.3 зображено сторінку реєстрації.

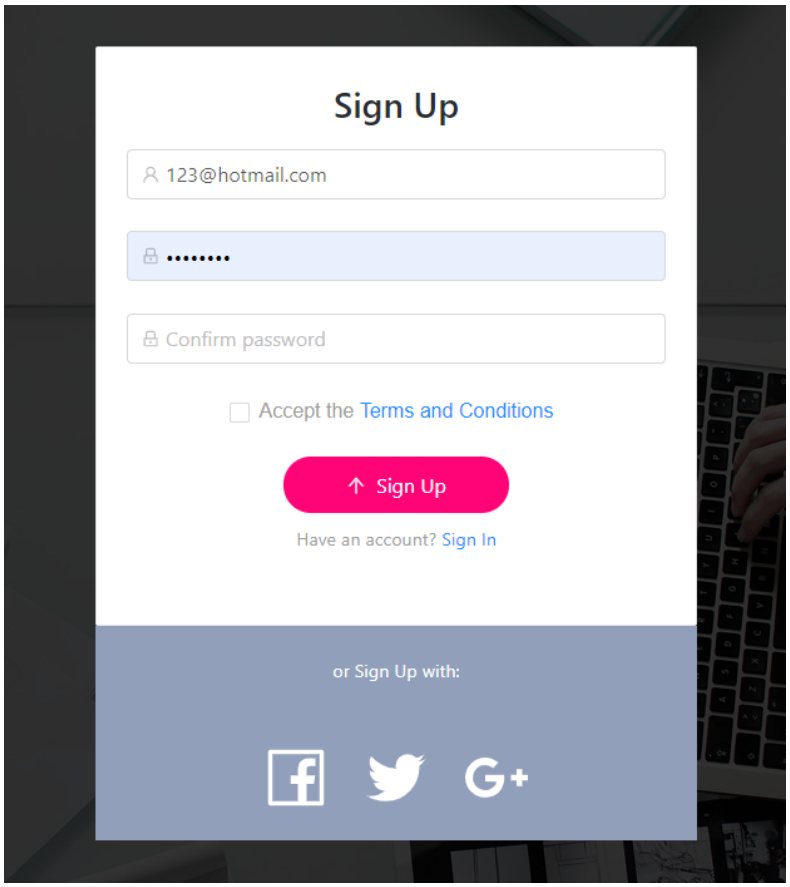

Рисунок 4.3 – Сторінка реєстрації користувача

На рис. 4.3 можна побачити саме перший крок реєстрації та три поля у формі: E-mail, Password та Confirm Password. Поле Password відображає закодований пароль. Також щоб завершити реєстрацію потрібно прийняти умови правил та угоди користувача. Якщо випадково потрапив на сторінку з реєстрацію, можна повернутись на сторінку в авторизацією натиснувши на Sign in.

Після натискання на кнопку Sign Up запит до Amazon Cognito збереже введені дані до бази, якщо вони відповідають вимогам безпеки, наприклад, пароль не може бути використаний, бо не має жодної цифри.

Після збереження даних Cognito створює системний лист до користувача з верифікацією поштової адреси та наміру реєстрації у той час коли веб-застосунок

перенесе користувача на сторінку верифікації. На рис. 4.4. можна побачити системний лист з кодом.

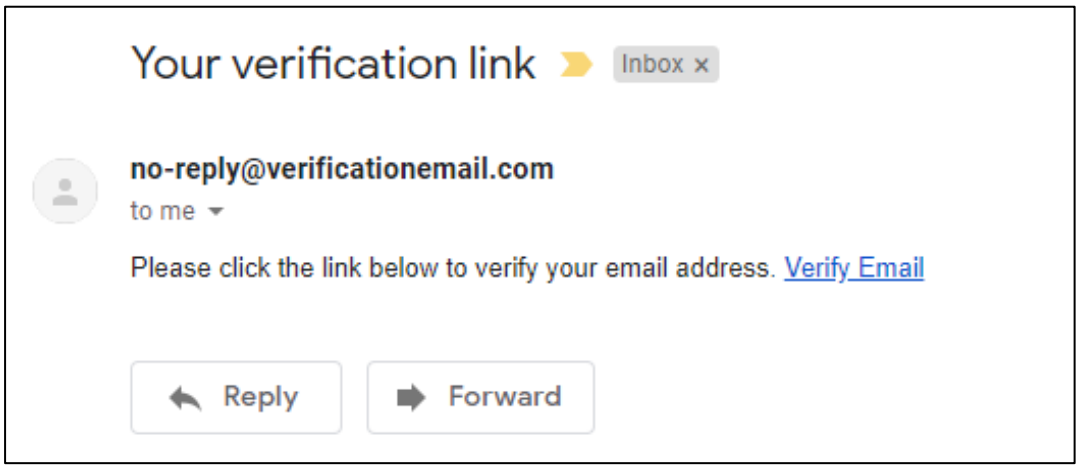

Рисунок 4.4 – Системний лист з верифікацією поштової адреси.

Після успішної верифікації веб-застосунок перенесе користувача на сторінку автентифікації та пропонує ввести ім’я користувача та пароль

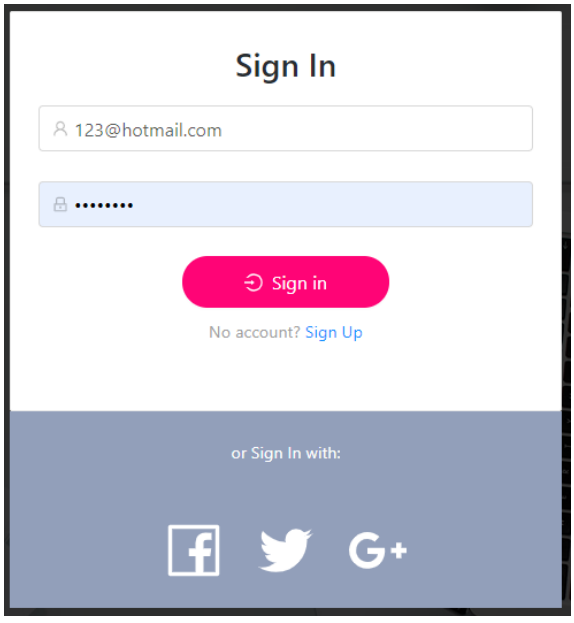

Рисунок 4.5 – Сторінка автентифікації

У випадку правильно введених імені користувача та паролю система перенесе менеджера проектів на основну сторінку зі списком проектів, що вже є зареєстрованими для моніторингу та керування. Якщо менеджер ще не має

зареєстрованих проектів та команд, то його основна сторінка буде виглядати як показано на рис. 4.6.

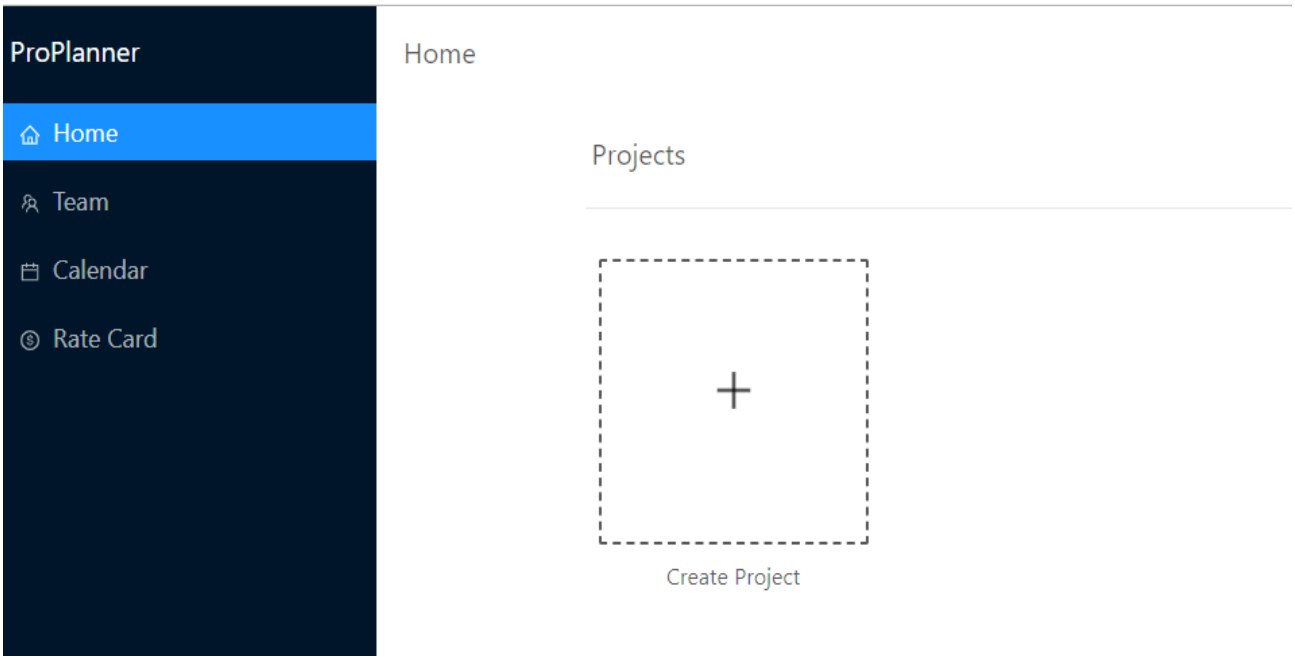

Рисунок 4.6 – Пуста основна сторінка нового користувача

Для того щоб зареєструвати в системі новий проект, потрібно натиснути на кнопку Create Project. Після того з’явиться модальне вікно для заповнення проектних даних. Обов’язково потрібно заповнити поля Title, Project start date та Project color. Поле Description заповнюється за бажанням.

Також вже на етапі створення проекту, треба зазначити у полі “Overhead cost per day” скільки коштів витрачається на підтримку інфраструктури проекту. Сюди включають ціни на оренду приміщення, оплату Інтернету та електроенергії, оренду технічного обладнання (комп’ютери, роутери, та інше). Ці витрати дуже часто забувають включити до остаточної ціни проекту, і тому наприкінці робіт, з’ясовується, що прибуток від виконаного проекту був замалий, або і зовсім його не було. Також дана інформація допоможе точніше з’ясувати, які будуть збитки при витрачанні часу на проект більше ніж було заплановано.

Після заповнення всіх даних, потрібно натиснути на кнопку Create для створення проекту в системі. Модальне вікно для створення проекту зображено на рис. 4.7.

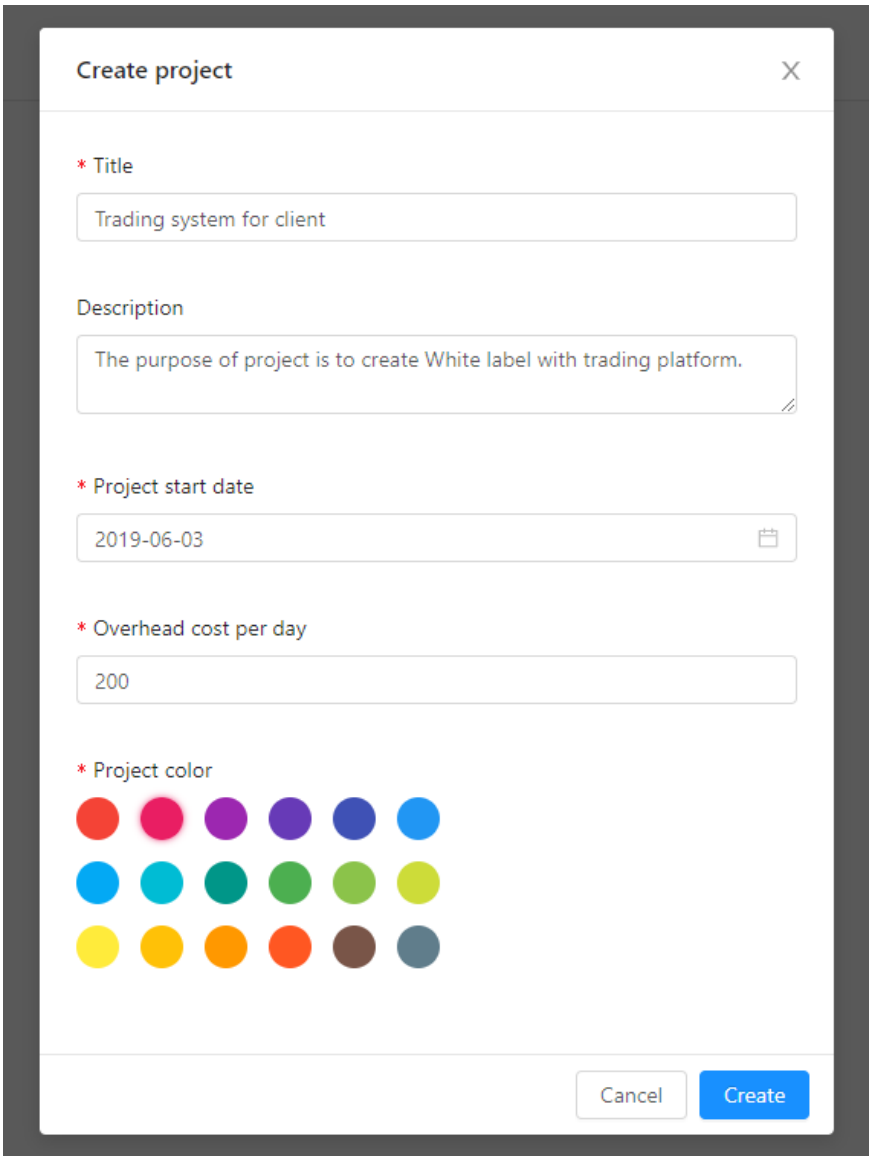

Рисунок 4.7 – Сторінка створення проекту

Після того як додана загальна інформація проекту, користувач потрапляє на сторінку проекту. Сторінка має декілька вкладок: Main Info, Tasks, Risks, Team, Gantt Diagram та Crash. Завдяки такому засобу побудови сторінки, нові дані демонструються без перезавантаження та перехід від однієї порції контенту до іншої здійснюється плавно й тому практично непомітний для користувача. Унаслідок користувач бачить кілька порцій інформації як одну єдину велику

сторінку. На вкладці “Main Info” відображається головна інформація проекту. На рис. 4.8 інформація у полях End Date, Duration, Project Cost зараз відсутня. Це розраховується, після того, як завдання до проекту додано, та також призначені ресурси, які будуть розробляти проект. Також кінцева вартість проекту може змінюватись залежно від ризиків, які описані на вкладці Risks.

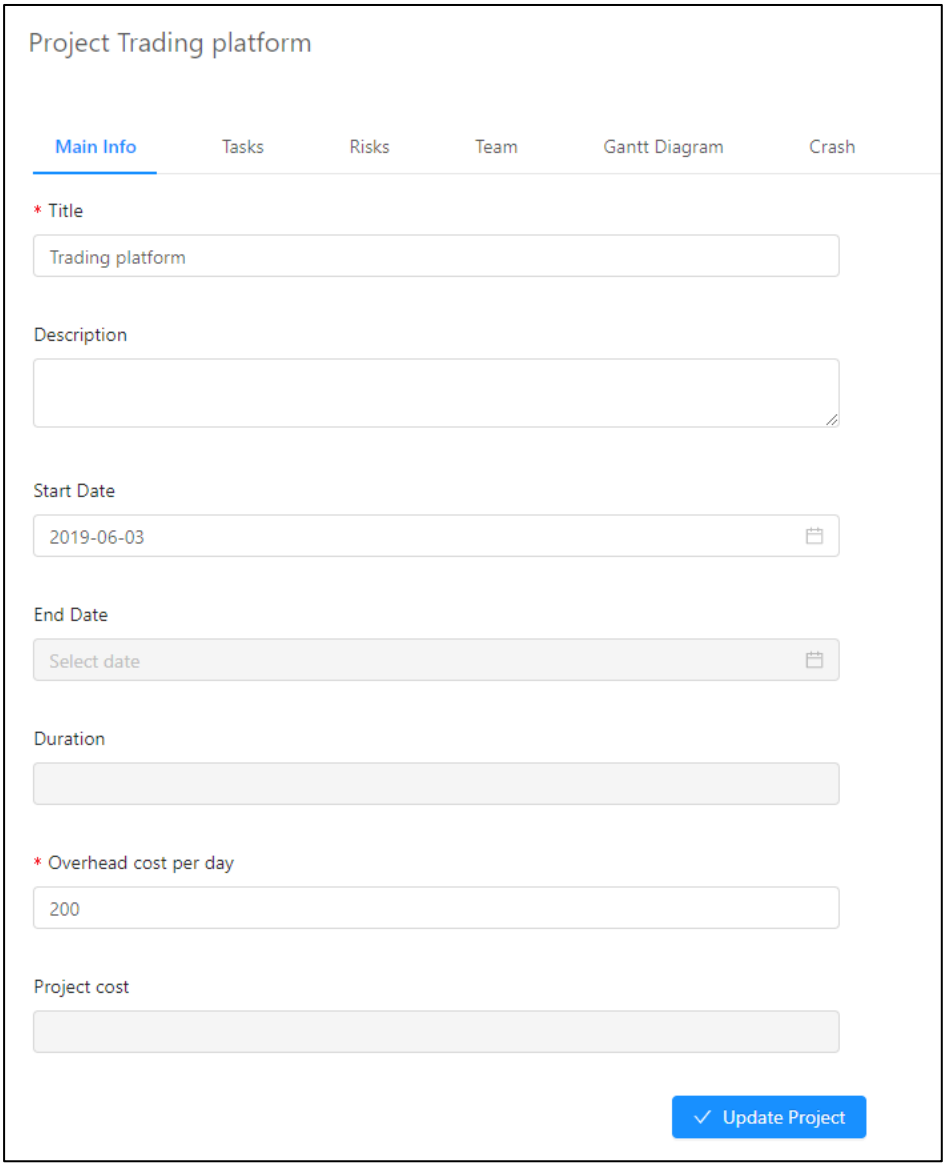

Рисунок 4.8 – Сторінка загальної інформації проекту

Розглянемо функціонал додавання проектних завдань. Після переходу до вкладки Tasks, якщо до того у проекту не було ніяких завдань, то сторінка буде незаповнена. Після натискання на кнопку +, з’явиться діалогове вікно, де потрібно буде заповнити інформацію щодо завдання. На рис. 4.9 частково зображено яка

інформація повинна бути обов’язково заповнена. Поля Description, Task Cost, Crash Cost, Predecessors та Assignee заповнюються пізніше.

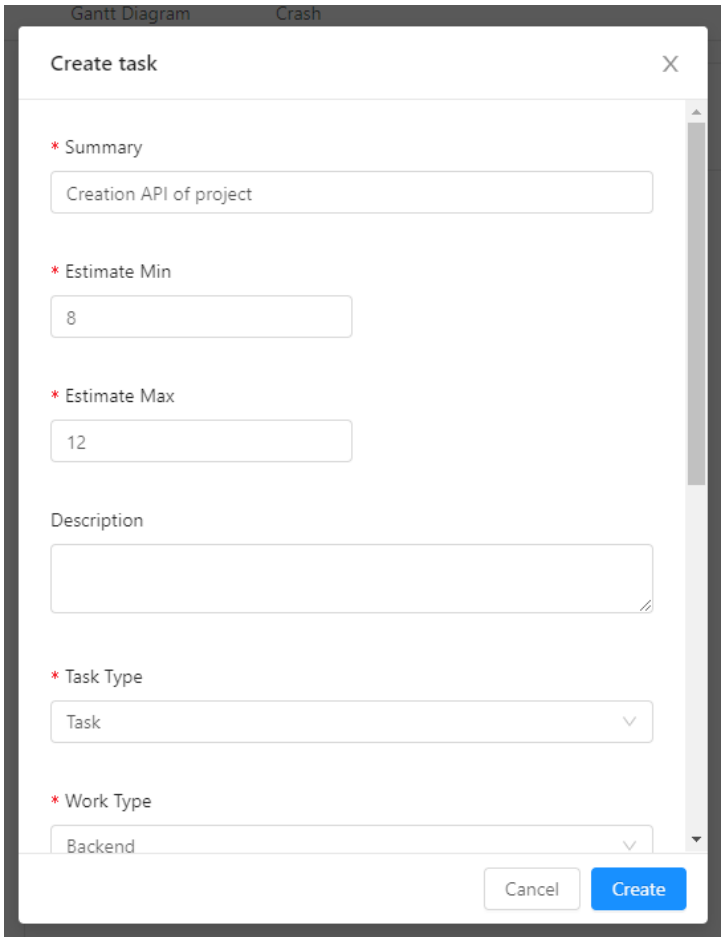

Рисунок 4.9 – Сторінка створення проектного завдання

Насамперед поле Task Cost автоматично розраховується, після того, як обрано тип завдання: бекенд, фронтенд або тестування, та додана оцінка часу на це завдання. Це можливо завдяки тому, що у кожного проекту також є своя Rate card. Про це більш детальніше буде описано пізніше.

Після того, як усі завдання внесені до проекту, вони відображаються списком з лівої сторони. Якщо потрібно більше деталей, ніж назва завдання та час виконання, то після натискання на завдання, з правої сторони з’являється інформація про тип роботи, опис завдання, мінімальна та максимальна оцінка на час виконання, ціна, та чи є інші завдання, які потрібно зробити спочатку. Також

зверху є 2 кнопки Edit Task, Delete Task, щоб або винести зміни до завдання, або видалити його з проекту.

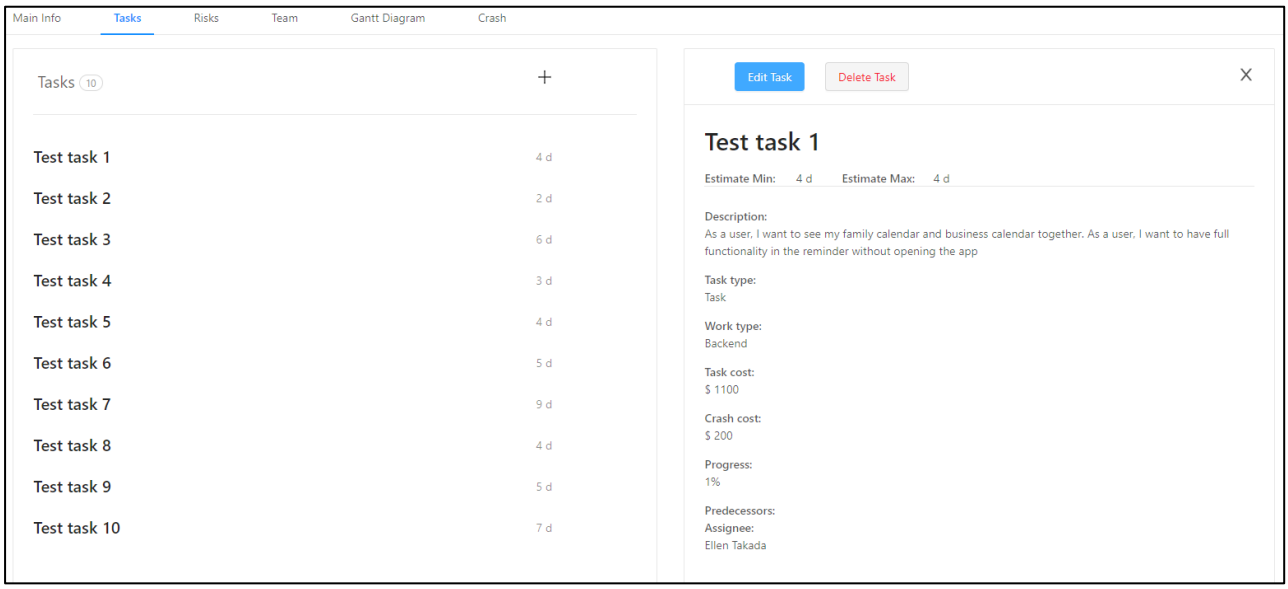

Рисунок 4.10 – Сторінка зі списком проектних задач

Після того, як задачі поставлено, то можна подивитись на діаграмі Ганта, як проект буде виглядати та коли може бути закінчений. Треба зазначити, що дана оцінка стане точніше, коли буде додана команда на проект, ризики будуть вказані. Але ми розглянемо послідовний варіант. Тому перейдемо до підбору команди на проект.

В системі передбачено 2 ролі користувачів: менеджер проекту, котрий може бачити тільки свої проекти та адміністратор, який відповідає до внесення усіх співробітників компанії до системи. Також він може бачити усі проекти, додавати відпустки співробітникам та редагувати їхні особисті профілі (додавати нові навички, кваліфікації та інше). В рамках цієї дипломної роботи, ми вже добавили 54 співробітника до додатку.

При пошуку команди на новий проект, менеджер проекту може спочатку подивитися які ресурси зараз без проектів, у кого коли відпустка та інше. Також на даній сторінці якщо натиснути на знак +, то можна подивитися більш детальну інформацію щодо навичок співробітника, які технології він використовує та який

має досвід. Також можна подивитися в якому офісі та в якій локації знаходиться співробітник. Сторінка усієї команди компанії зображена на рис. 4.11.

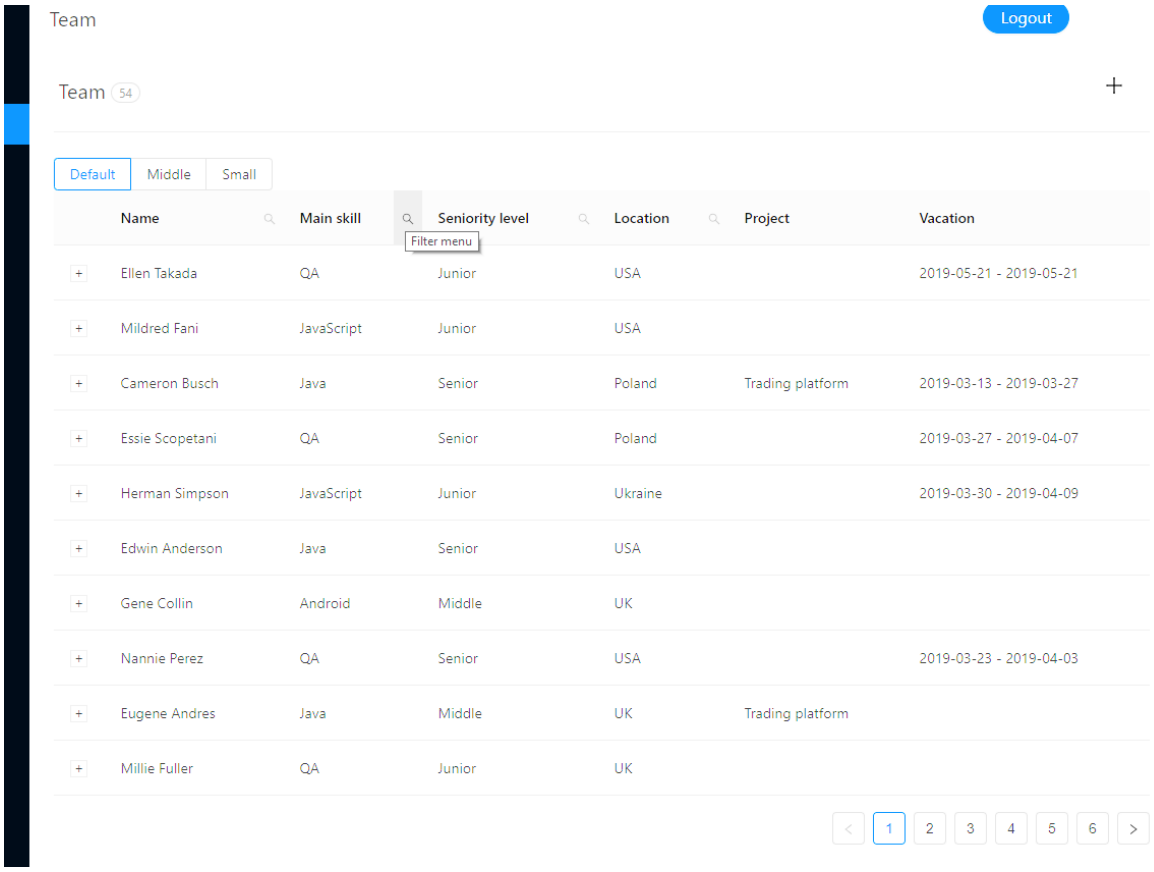

Рисунок 4.11 – Сторінка зі списком всіх співробітників

Для того щоб призначити людей на проект, потрібно перейти на вкладку Team на сторінці проекту. Даний функціонал можна побачити на рис. 4.12.

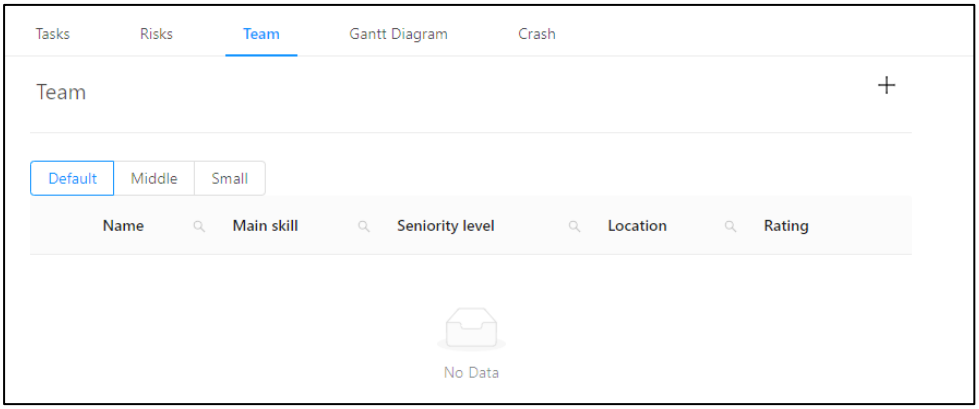

Рисунок 4.12 – Сторінка проектної команди

Після натискання на кнопку +, з’явиться модальне вікно для пошуку команди. Як було описано в алгоритмі пошуку команди, це абсолютно оптимізований етап планування.

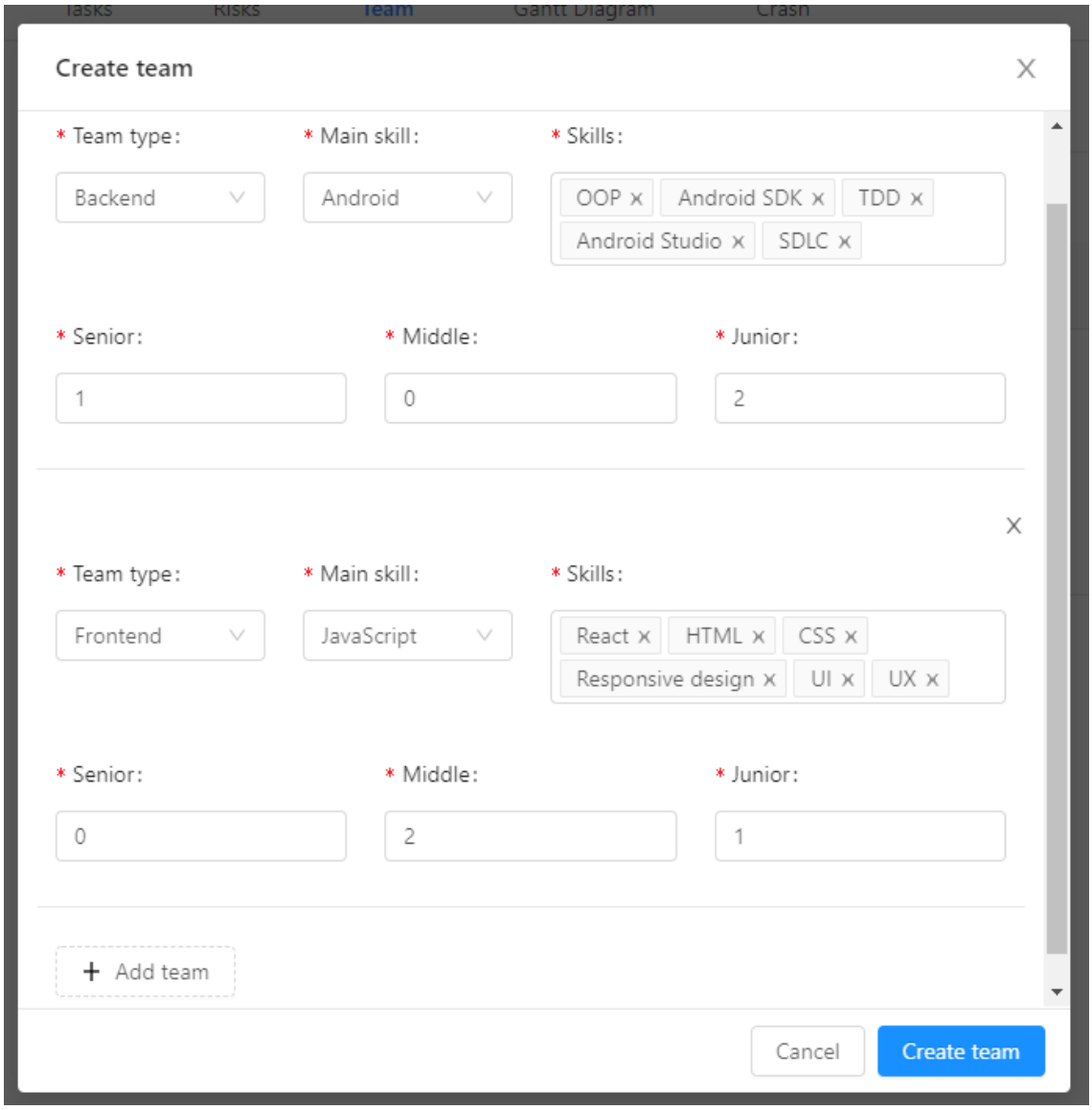

Рисунок 4.13 – Модальне вікно пошуку команди на проект

Для пошуку команди, потрібно заповнити поле Team type, яке включає в себе такі напрямки: backend, frontend, QA, mobile, design, devOps. Обов’язково заповнити і головну технологію, яка потрібна на проекті: Main skill - .Net, Java, Python, JS, QA Automation, Android, iOS, та інше. Після цього потрібно визначитись які додаткові навички потрібні. Для цього поле Skills зроблено по типу мульті вибору. Останнім кроком треба вибрати який висококваліфікованих

спеціалістів потрібно до обраного типу команди. Якщо на проект потрібен більше ніж 1 тип команди, то можна додати ще після натискання на кнопку Add team. Коли все обрано, менеджер натискає на кнопку Create team та система починає пошук співробітників, які краще підходять під вимоги проекту.

Після виконання алгоритму, менеджеру буде запропонована команда, яка найбільше підходить зараз на проект, серед вільних ресурсів компанії. Запропоновану команду можна побачити на рис. 4.14.

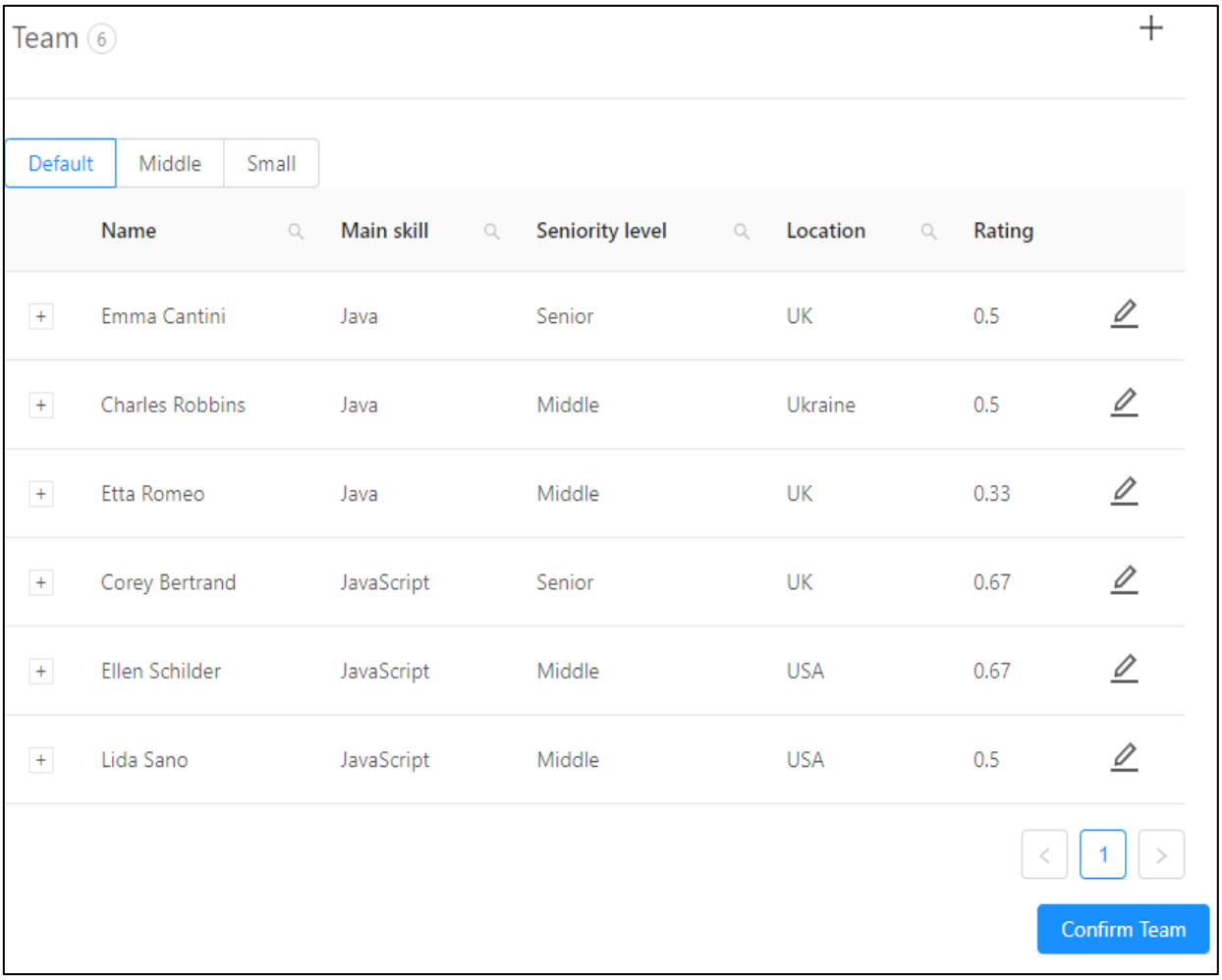

Рисунок 4.14 – Запропонована команда на проект

На сторінці буде зображена така ж сама інформація як і на загальній сторінці всіх співробітників, але також буде додаткова інформація, як рейтинг кожного з претендентів на посаду в проекті. Якщо менеджер згоден з вибором то тоді після натискання на кнопку Confirm team команда буде вже записана на обраний проект.

Якщо менеджера не задовольнили пошуки кандидатів, він може спробувати змінити критерії пошуку, наприклад шукати на проект не тільки Junior QA, але й додати Middle QA.

Коли команда сформована, то потрібно повернутися до сторінки з завданнями проекту, та призначити хто над яким завданням буде працювати. Таким чином, це допоможе врахувати в рамках проекту не тільки голі цифри по довжині проекту, але й зрозуміти скільки насправді займе час виконання проекту. Зробити це можливо, натиснувши на кнопку Edit, та обравши члена команди на виконання даного завдання. Детальніше можна оглянути на рис. 4.15.

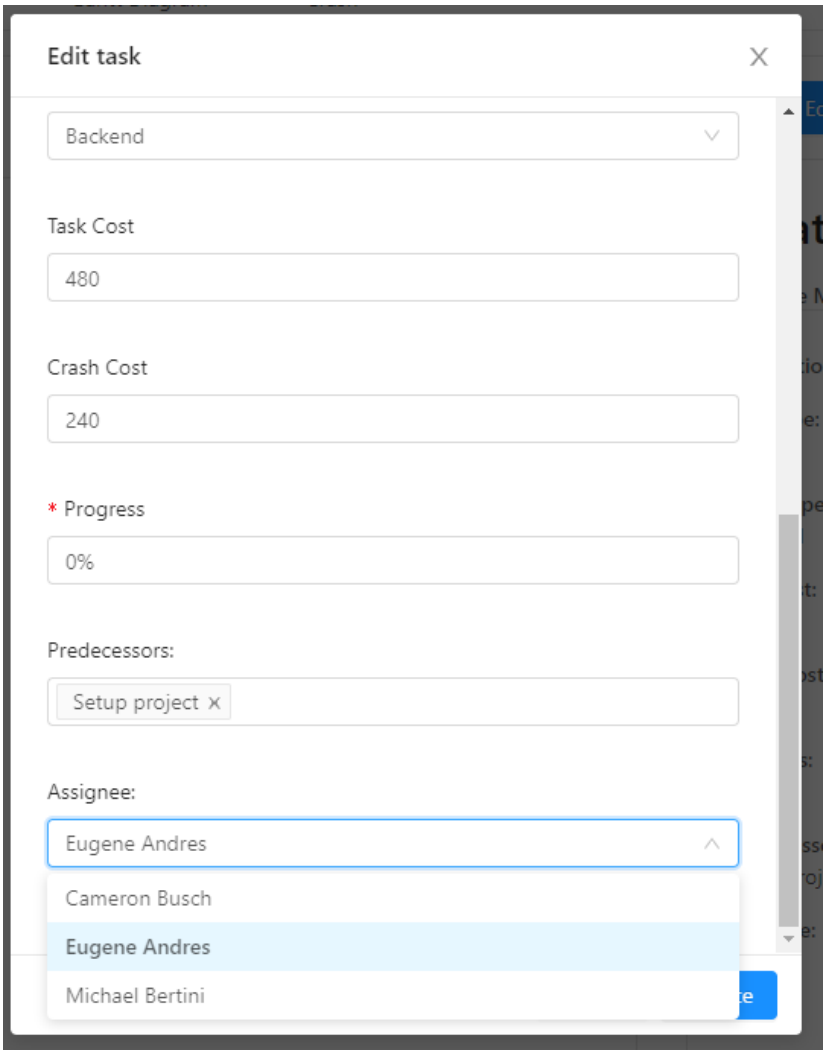

Рисунок 4.15 – Призначення членів команди на завдання

В полі Assignee будуть знаходитись тільки ті співробітники, що були обрані до проекту, і чия кваліфікація співпадає з типом завдання. Після призначення

команди на проект та завдання, можна повернутися до головної сторінки проекту та натиснути на перерахунок критичного шляху та визначити дати закінчення проекту. На рис 4.16 зображено проект, коли завдання вже додані, є команда, але якраз дати завершення проекту не визначені.

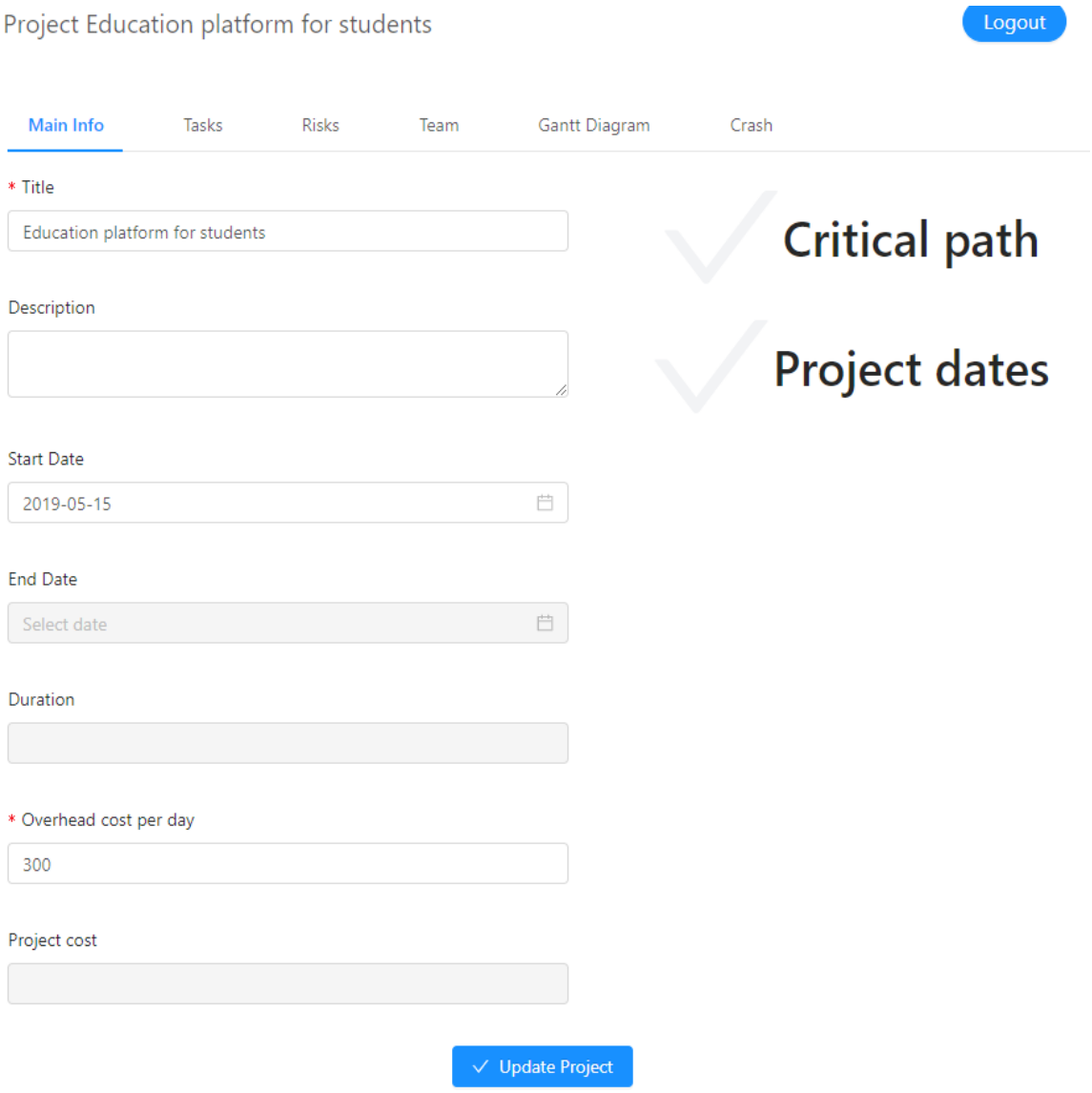

Рисунок 4.16– Приклад проекту до визначення дати закінчення

Для того щоб розрахувати кінцеві дати проекту, спершу потрібно натиснути Critical Path для розрахунку критичного шляху. Після того, як це буде виконано, можна натиснути на Project Dates, та в полях End Date, Duration з’явиться ця інформація. Дана інформація обчислюються автоматично, та змінити її можна, якщо ще раз натистнути на кнопку. Project Dates. Але спершу потрібно змінити

або додати нове завдання проекту. Як виглядає сторінка з оновленими даними можна побачити на рис.4.17.

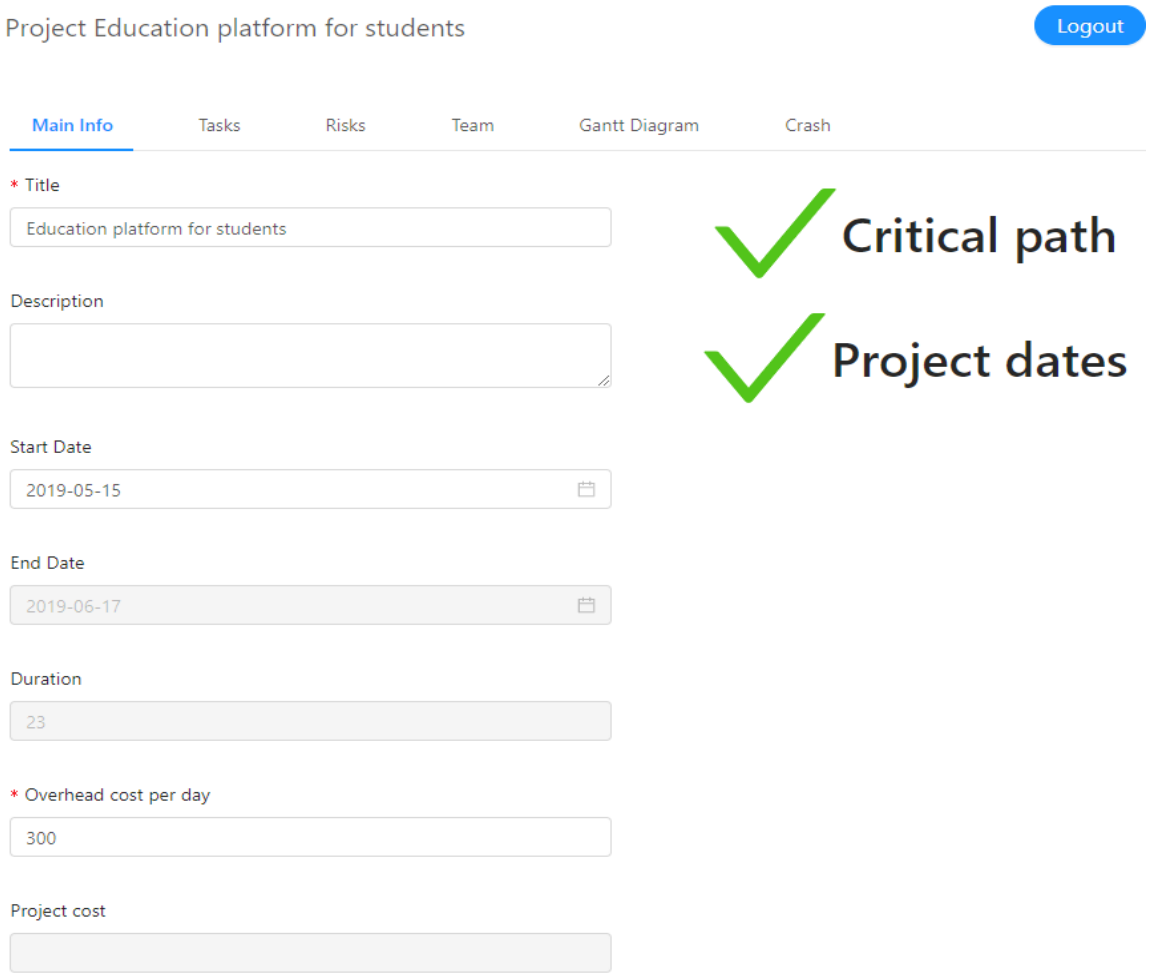

Рисунок 4.17– Приклад проекту після визначення дати закінчення

Як можна побачити з наведеного скріншоту, довжина проекту становить 23 робочих дні.

Після того як розраховані дати проекту та визначений критичний шлях проекту, можна переходити на вкладку Gantt Diagram, щоб побачити вже детально як виглядає проект в межах календарного плану. З лівого боку буде зазначено визначений список завдань, хто над яким завданням працює, яка тривалість виконання завдання, та коли почали виконувати вибране завдання. З правої сторони буде намальована діаграма Ганта. Також завдяки позначці, який сьогодні день, можна зрозуміти де знаходиться зараз проект в межах успіху для завершення

проекту в строк. Найбільш важливою інформацією на діаграмі Ганта є відображення критичного шляху проекту який можна побачити на рис. 4.18.

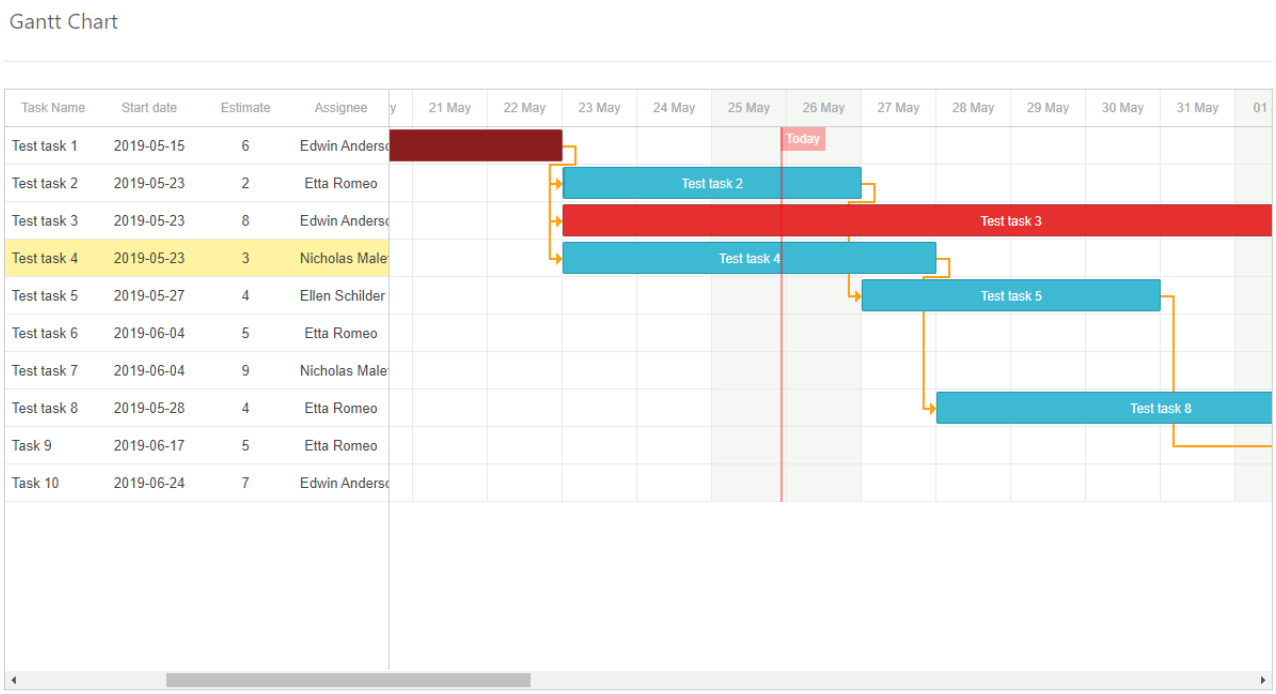

Рисунок 4.18 – Діаграма Ганта проекту

Критичний шлях – це ланцюжок зв’язаних завдань, що безпосередньо впливають на дату завершення проекту [6]. Якщо будь-яке завдання на критичному шляху виконується із запізненням, то увесь проект також виконується із запізненням. Всі завдання, які входять до критичного шляху пов’язані між собою, та на етапі створення завдання залежність позначається через поле Predecessors.

Коли завершено останнє завдання на критичному шляху, то проект автоматично теж буде завершено. Хотілось би зазначити, що в даній роботі, на діаграмі Ганта можна побачити вихідні дні, але вони не входять до розрахунків тривалості завдання. Наприклад, якщо візьмемо Task 4, то на діаграмі він відображається тривалістю у 5 днів, але якщо уважно подивитись на поле Estimate, то можна побачити, що саме завдання займає 3 дні для завершення. Цей формат відображення було обрано для того, щоб не виникало уяви, що вихідні дні це додатковий час, куди можна ще добавити нових завдань. Ще однією з переваг

даного додатку, що не можна обрати 1 і ту саму людину на виконання завдань, які можуть виконуватися паралельно. Якщо інших кандидатів немає, то обране завдання також буде у черзі і це впливатиме на остаточні дати завершення проекту.

Для більш детального планування проекту на етапі дослідження факторів успішності IT проектів, було визначено, що потрібно мати чітке розуміння, коли є додаткові вихідні. Особливо це важливо, якщо проектна команда знаходиться в різних локаціях. Тому в даній атестаційній роботі було добавлено також сторінку календаря, де можна вибрати локацію та подивитися коли є святкові вихідні. Вигляд календаря можна побачити на рис. 4.19.

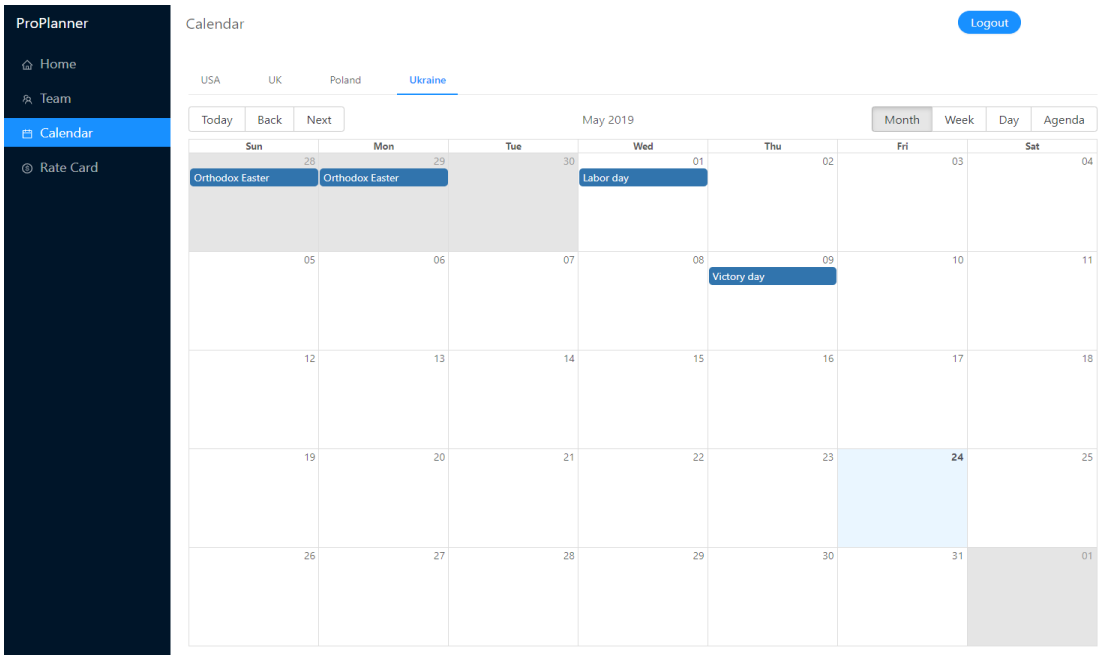

Рисунок 4.19 – Календар зі святковими днями в різних локаціях

Менеджер має можливість редагувати та додавати свята, якщо нові вихідні було призначено. Для цього потрібно натиснути на обраний день, та після того як з’явиться діалогове вікно, додати назву свята, та змінити дату, якщо помилково була обрана інша дата. Також можна міняти вигляд календаря, натиснувши на кнопки з правої сторони. Можна відображати як на цілий місяць, тиждень або день.

Керівник проекту може виконувати різні заходи для досягнення успіху проекту, а саме виконання проекту у строк. Деякі з поширених методів це

наприклад додавання додаткових ресурсів до вирішення задач критичного шляху. Цей варіант має свої обмеження, такі як збільшення бюджету для додавання ресурсів та доступність ресурсів. Інший варіант, це зменшити кількість вимог проекту. Це можна зробити, лише якщо спонсор і головні зацікавлені сторони погодились скоротити кількість завдань проекту. Тому в даній атестаційній роботі ми пропонуємо варіант - розрахунку крешу проекту. Project crash - є технікою стиснення графіка, яка використовується для зменшення або скорочення розкладу проекту. Ця методика стиснення розкладу, передбачає додавання додаткових ресурсів до проекту для стиснення графіка. Під час збою менеджер проекту переглядаєте критичний шлях і обирає, які дії можна завершити, додавши додаткові ресурси. Тому при створенні завдань, є поля Crash cost та Minimal Estimate. Керівник додає інформацію, щодо ціни на зменшення тривалості виконання завдання на 1 день. Як виглядає сторінка зі скороченням графіку проекту можна подивитись на рис. 4.20.

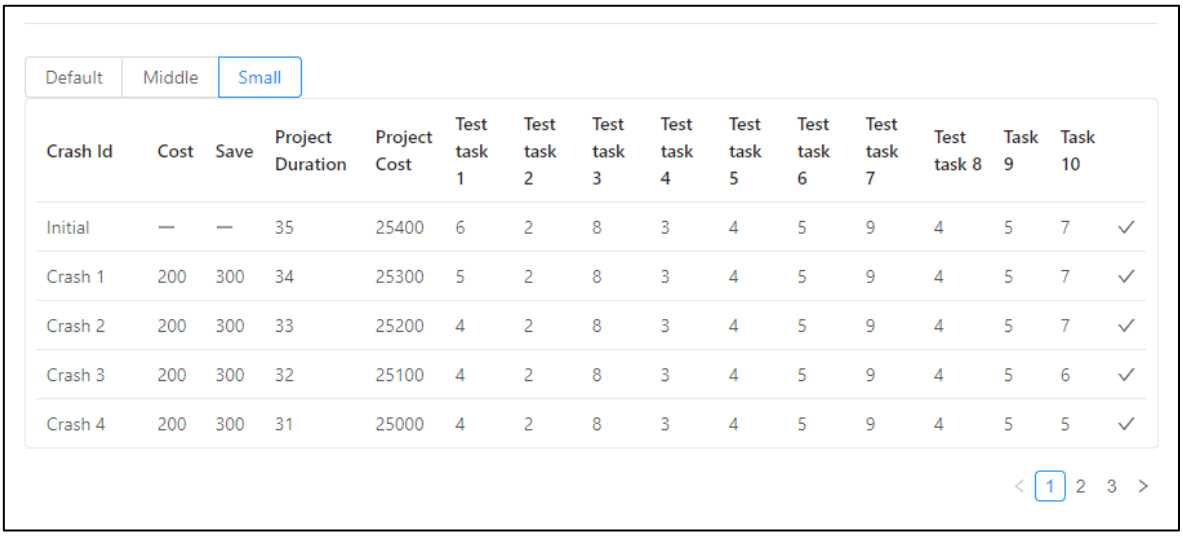

Рисунок 4.20 – Графік скорочення проекту

З лівої сторони можна побачити скільки коштуватиме зменшення тривалості проекту на 1 день, як це відобразиться на загальній тривалості проекту. Та чи вигідно буде зменшувати тривалість проекту. В колонці Save можна побачити скільки коштів можна буде заощадити при зменшенні проекту на 1 день. Ці кошти

як раз ті що заповнювалися на початку проекту. Усі витрати на електроенергію, Інтернет, оренду та технічне обладнання. Якщо керівник бажає отримати більш детальну інформацію, то треба натиснути на вибрану строку, і з'явиться модальне вікно з перерахованою діаграмою Ганта. Також буде зазначено скільки початково планувалась тривалість проекту, та яка нова кінцева дата.

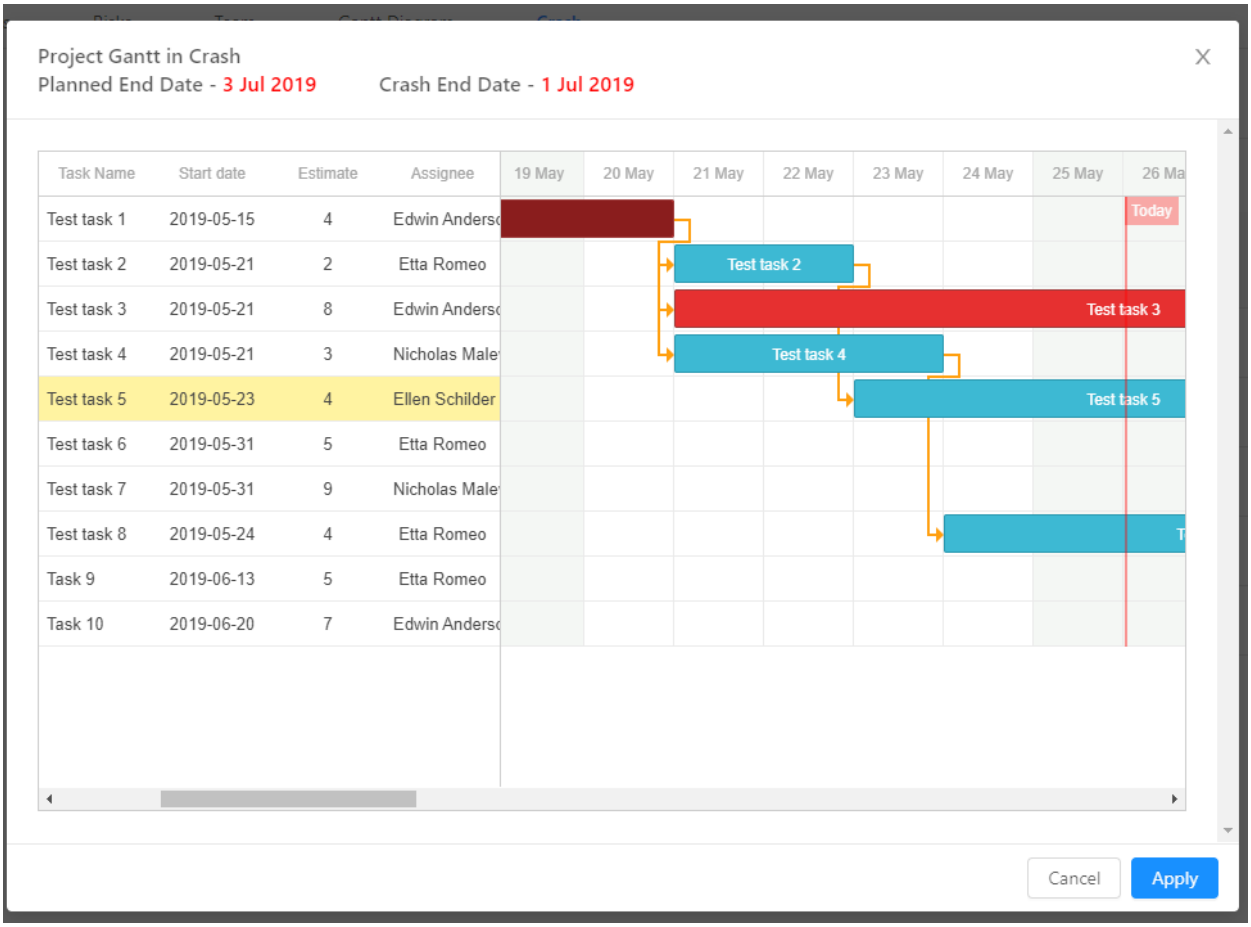

Рисунок 4.21 – Графік скорочення проекту та оновлена діаграма Ганта

Для порівняння візьмемо Task 1, на рисунку 4.18 дата закінчення завдання була 23 травня, а зараз після зменшення тривалості 1 завдання на 2 дні, дата закінчення становить 21 травня. Таким чином залежні від нього завдання на критичному шляху також починаються раніше ніж очікувалось. Щодо економії з боку бюджету, в даному випадку можна буде заощадити 200 доларів. Але бувають випадки коли ціна на зменшення тривалості завелика. Найчастіше це ті завдання які неможливо ніяк зробити паралельно, або додавання нових ресурсів призведе

навпаки до перевитрачання часу, так як більше часу потрібно буде новій людині на отримання проектних знань. Якщо менеджер проекту задоволений результатом скорочення тривалості проекту, то він може застосувати нову версію проекту натиснувши на кнопку Apply. Після цього проект буде перераховано за новими даними.

Ще одна цікава функціональність цього додатку – це можливість додавання ціни для кожної технології. Детальніше можна розглянути на рис. 4.22.

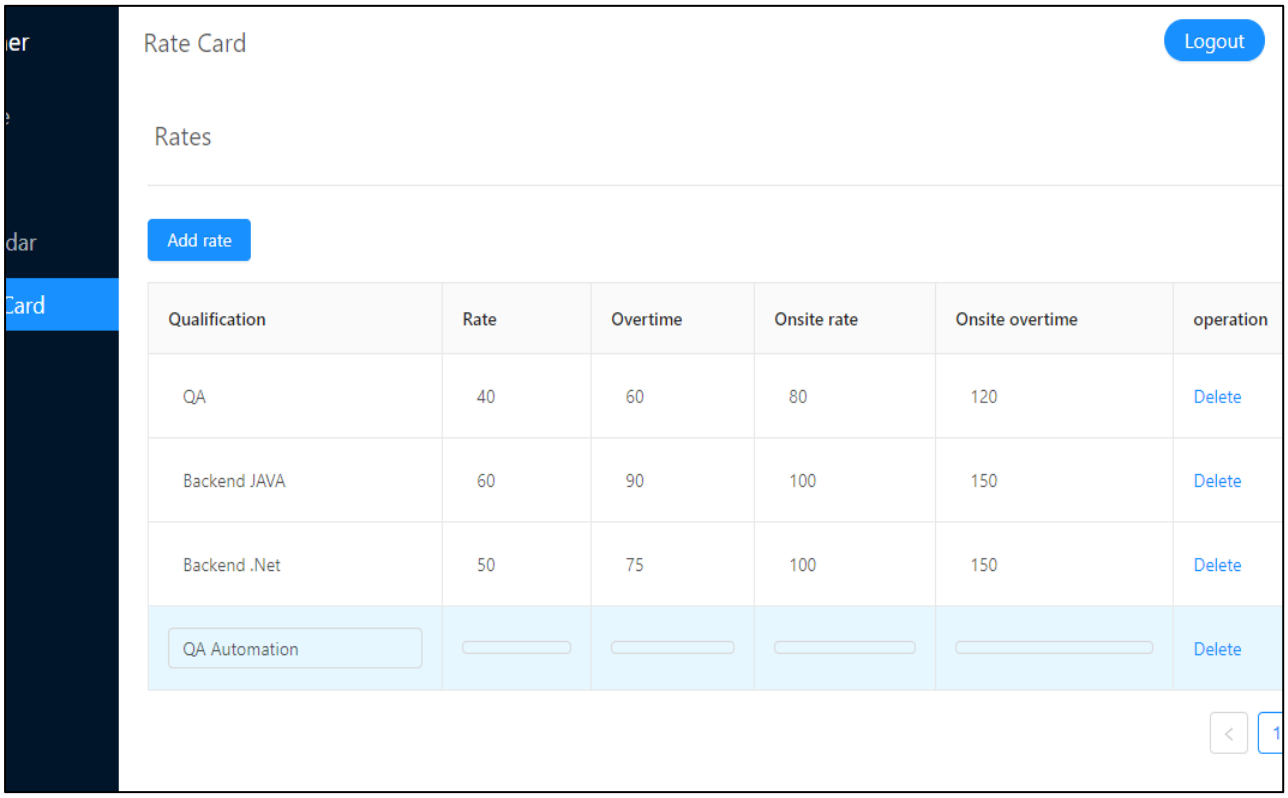

Рисунок 4.22 – Rate card проекту

Так як ціни на ресурси залежать не тільки від компанії, але і від домовленостей між клієнтом на компанією, було запропоновано зробити свою рейт карту під кожен проект. Менеджер має можливість додавати ціни на різний стек технологій просто натискаючи кнопку Add rate.

Після цього як показано на рис. 4.22 буде додана нова строчка у таблицю де менеджер вводить дані. Наприклад скільки коштує 1 час програміста Java, скільки буде коштуватиме робота в неробочі години. Скільки коштуватиме програміст,

якщо він знаходиться в команді клієнта, та який рейт якщо він працює в неробочі години.

На даному етапі це загальна функціональність яка доступна менеджеру проектів. Тепер розглянемо цей додаток з точки зору адміністратора.

Після того як адмінстратор залогінився на веб додаток за допомогою спеціального облікового запису, він потрапляє на Web-сторінку, яка зображена на рисунку 4.23.

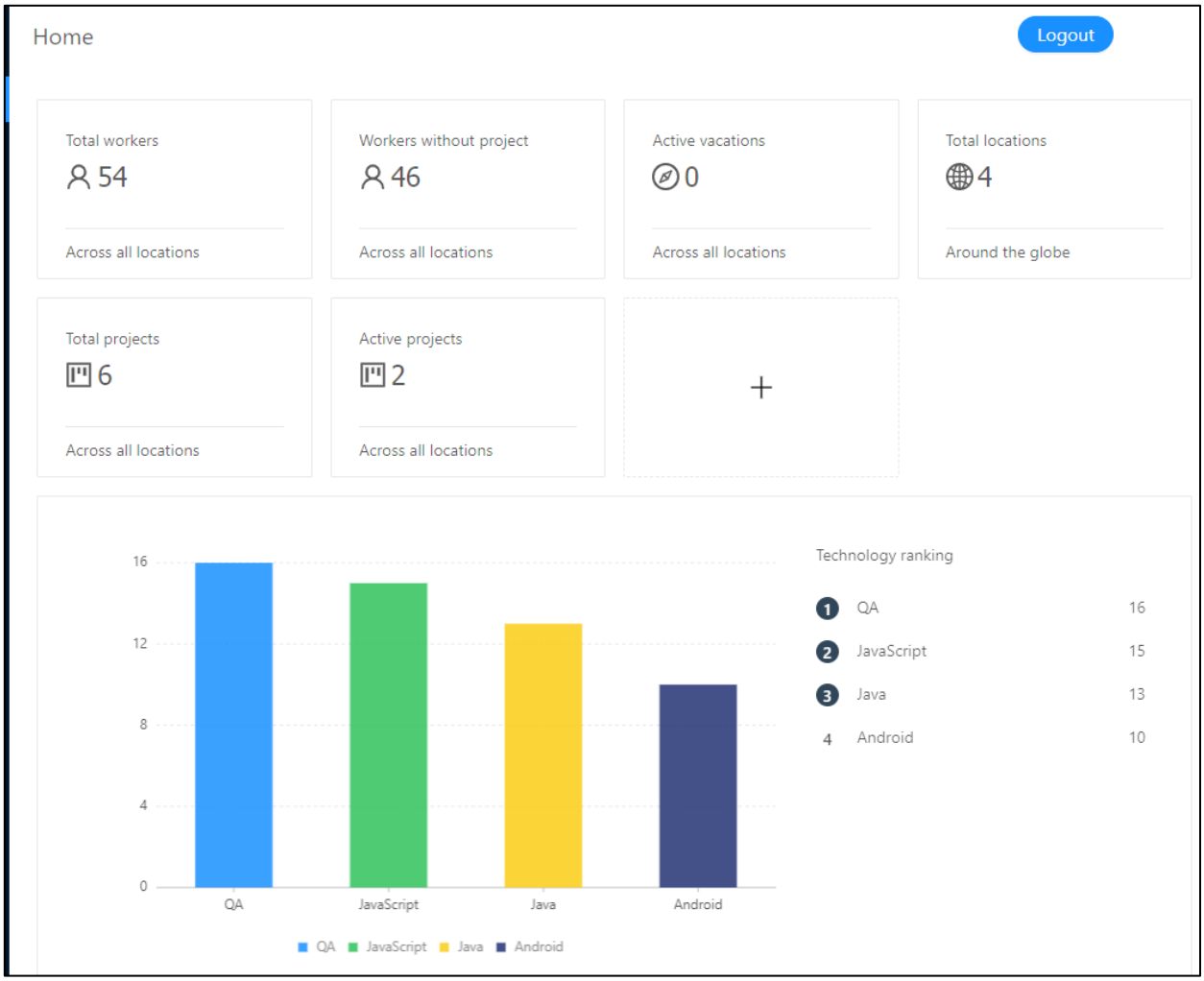

Рисунок 4.23 – Статистика компанії

На сторінці можна побачити різну статистику, наприклад, скільки проектів зараз у розробці, скільки всього співробітників в компанії, та скільки серед них знаходяться без активного проекту. Також можна побачити хто зараз знаходиться у відпустці, та скільки всього офісів розробки у компанії. Також є графік який

показує скільки всього людей в компанії по різним технологіям. На рис. 4.23. можна побачити, що на сьогодні найбільш всього це тестувальників програмного забезпечення – 16 чоловік. Також можна додати свою кастомну статистику, для цього потрібно натиснути на квадрат з іконкою хрестика.

В даній дипломній роботі, тільки адміністратор компанії може додавати нових співробітників. Для цього потрібно зайти на вкладку Team рис. 4.24.

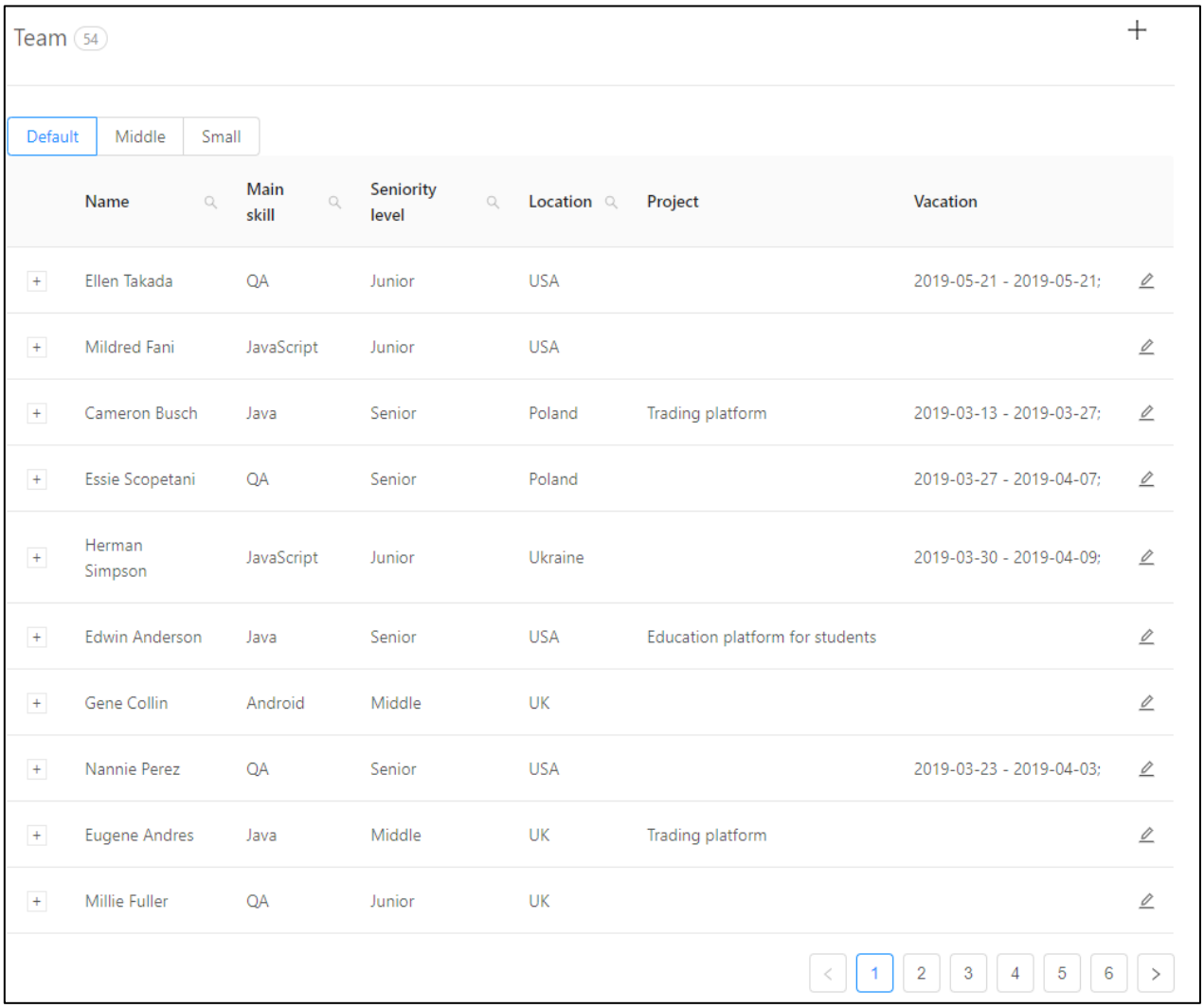

Рисунок 4.24 – Таблиця зі співробітниками компанії

На цій сторінці розташована така ж сама інформація про команду яка є доступної і для менеджера проекту, але за одним винятком – функцією додавання нових ресурсів у компанію. Для того щоб додати потрібно натиснути на кнопку з

іконкою хрестика, та після цього з’явиться діалогове вікно як зображено на рис. 4.25 для створення нового облікового запису співробітника компанії.

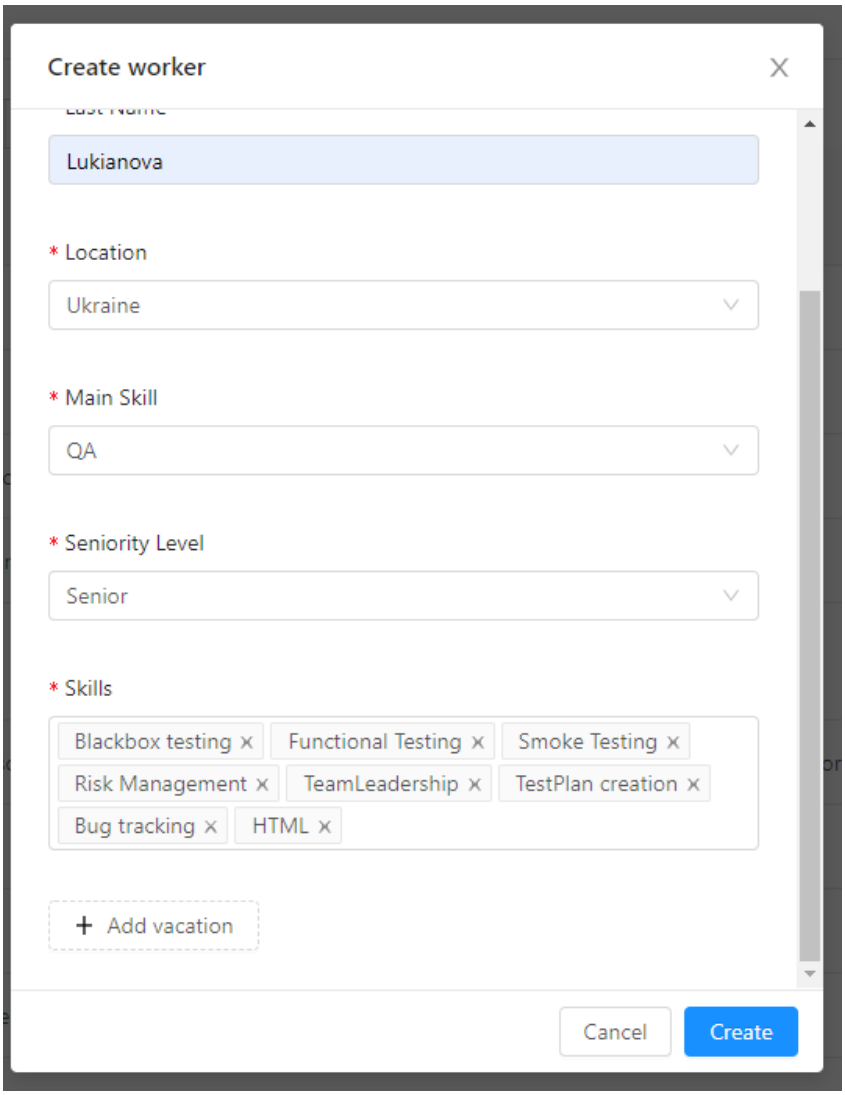

Рисунок 4.25 – Форма внесення даних нового співробітника

Після того як всі дані були внесені, можна також додати інформацію про відпустки співробітника. Для цього потрібно натиснути на кнопку Add vacation. Після натискання з’явиться можливість вибрати дати відпустки та додати їх до облікового запису співробітника.

Також можна додати декілька відпусток за 1 раз. Якщо відпустка була додана помилково, то її можна видалити. Більш детальна інформація щодо додавання відпусток зображена на рис. 4.26.

Після того як все додано, адміністратор натискає на кнопку Create, та новий співробітник з’являється в системі.

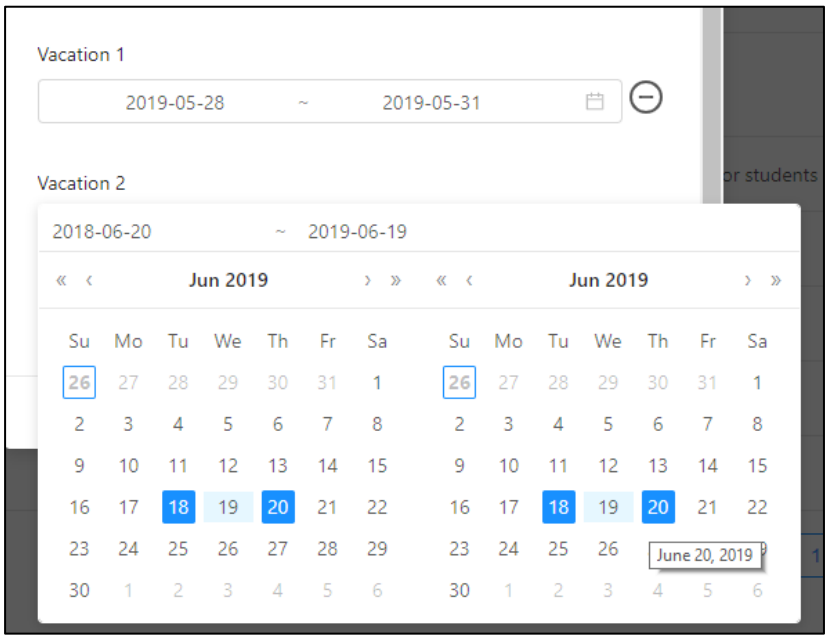

Рисунок 4.26 – Додавання відпустки співробітнику

Ще одна можливість яку має адміністратор – це додавання нових локацій до календаря, або видалення їх із системи. Цей функціонал можна переглянути на вкладці Calendar.

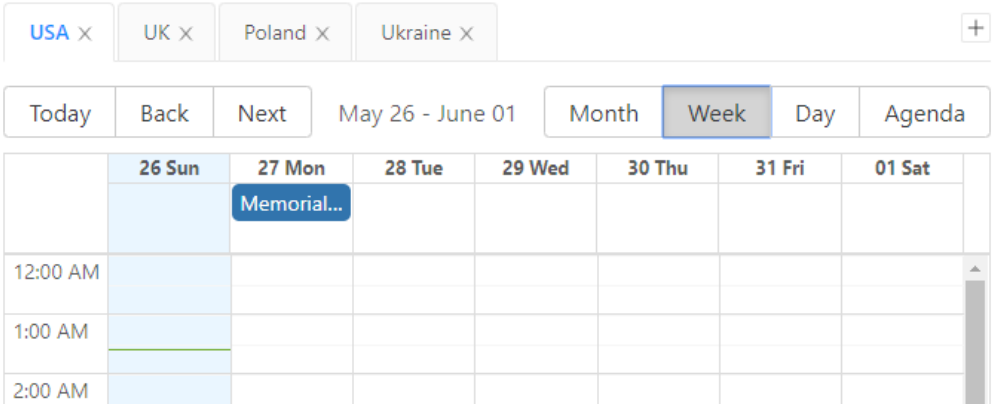

Рисунок 4.27 – Редагування локацій компанії

Для того щоб видалити локацію, потрібно натиснути на хрестик на обраній вкладці.

Для додавання нової локації потрібно лише натиснути на кнопку + та ввести інформацію, щодо місцезнаходження нового офісу. Після того як додано нову локацію, потрібно внести усі святкові дні, для більш чіткого планування ресурсів на проекті.

В результаті атестаційної роботи було зроблено зручний веб-додаток, який допомагає керівникам проекту моніторити та спостерігати за планом виконання робіт проекту. Також є можливість дослідити як змінюється бюджет проект, якщо клієнт бажає стиснути час виконання проекту завдяки алгоритму зменшення тривалості розробки завдань. Реалізація веб-додатку була створена за допомогою Serverless архітектури. Так як цей проект розгорнуто на Amazon сервері, то його можна з легкістю використовувати іншим компаніям.

5 МОЖЛИВІСТЬ ВИКОРИСТАННЯ РОЗРОБЛЕНОГО WEB-ДОДАТКУ

Проаналізувавши дані в розділі 2 Опис проведених досліджень, можна зробити висновок, що здатність менеджера проекту точно оцінити вартість та спланувати процес розробки програмного забезпечення безпосередньо впливає на успіх будь-якої компанії-розробника програмного забезпечення. Статистичний аналіз опитування респондентів, які мають досвід розробки чи керування програмними проектами, показує що є сильна позитивна кореляція з рейтингами успішності програмного проекту від того наскільки детально запланована розробка проекту. Застосування сучасних методологій управління проектами і програмами є чинником, що визначає ефективність використання капіталу в процесі функціонування і розвитку підприємств і організацій. Але розширення ринку і збільшення складності проектів супроводжується значним збільшенням кількості інформації, яку менеджери повинні обробляти в обмежені терміни і з найвищою ефективністю. При високих темпах розвитку IT індустрії, стає зрозумілим, що керівникам проектів досить складно відстежувати всі процеси в проектах, збирати і оперативно обробляти всю необхідну інформацію [17]. Зараз в IT сфері неможливо обійтися без засобів автоматизації діяльності будь яких співробітників та менеджерів. Тому виникла потреба в сучасних інформаційних системах управління проектами, які могли б стати помічником менеджерів не тільки з позицій збору необхідної інформації, але й для надання посильної допомоги у вирішенні управлінських, облікових, фінансово-економічних та інших задач. Подібним чином, компанії сьогодні можуть легше пом'якшити ризик, виявивши неспроможні аспекти проекту, використовуючи програмне забезпечення для управління проектами.

Останнім часом все більшого розповсюдження набуває одночасне управління декількома проектами, у таких умовах більше уваги має приділятися контролю виконання етапів проектів. IT дають можливість реалізувати мультипроектне управління, при якому управління декількома проектами виконуються

паралельно, незалежно один від одного, але використовують спільні ресурси. У мульти проектному управлінні ІТ дозволяють описувати склад та характеристику робіт, ресурсів, прибутків та видатків проектів, створювати розклад виконання робіт із урахуванням проектних обмежень, виявляти критичні операції та резерви часу для виконання інших операцій, розраховувати бюджет проектів, потреби проектів у матеріалах і ресурсах, планове завантаження ресурсів проектів, аналізувати ризики та резерви, розрахувати успішність виконання проектів, вести облік та аналіз виконавців проектів, отримувати необхідну звітність за проектами [18].

Таким чином веб-додаток, який було описано в 4 розділі можна використовувати в подальшій діяльності. На даний момент цей проект має кілька переваг серед аналогів, такі як:

- алгоритм автоматичного аналізу та пошуку ресурсів на проект;
- розрахунок критичного шляху проекту;
- врахування до тривалості проекту святкових вихідних днів, у тих локаціях, де знаходиться команда;
- аналіз вигідності зменшення тривалості проекту, та як це впливає на бюджет проекту.

Даний веб-додаток планується подальше розвивати, наприклад зробити інтеграцію з JIRA, та додати можливість співробітникам вносити в систему час, котрий було витрачено на виконання завдання. Також планується розробити додаткові функції: ведення архіву та документообороту, аналітичні функції мережевого мультипроектного планування

ВИСНОВКИ

Реалії сьогоднішньої ситуації на ринку України й світу в цілому диктують компаніям певні вектори діяльності, які пов’язані як з виживанням, так і розвитком. Світовий досвід показує, що єдиним універсальним підходом до рішення величезної кількості завдань є проекти і управління ними. Досягнення високої ефективності в управлінні проектами допомагає забезпечити успішне завершення проекту та отримання очікуваних результатів від продукту [19]. В даний час в довіднику з управління проектами (англ. A Guide to the Project Management Body of Knowledge або PMBOK Guide) не вистачає відповідних інструментів, які допоможуть кількісно оцінити ефективність управління проектами в інженерії програмного забезпечення. Крім того, здатність оцінювати або вимірювати прогрес є необхідною для проведення будь-якого формального процесу з поліпшення ефективності управління [20].

В рамках атестаційної роботи було досліджено та проаналізовано існуючі методи оцінювання ефективності управління програмним продуктом. Також було проведено опитування українського IT ринку, щоб визначити, які помилки найчастіше виникають при плануванні процесу розробки проекту та яка можливість покращити процес керування програмними проектами. та розроблено веб-додатка який допомагає менеджерам в організаціях з розробки програмного забезпечення оцінити, моніторити та підвищити ефективність управління проектами. Простіше кажучи, цей продукт завдяки власне розробленому алгоритму пошуку ресурсів на проект і врахування вихідних та святкових днів у локаціях проекту допоможе зацікавленим сторонам проекту досягти кращих результатів проекту, таких як завершення проекту вчасно, в межах бюджету, з необхідною функціональністю та створенням бажаного проекту для замовника.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Agarwal, N., & Rathod, U. (2006). “Defining ‘Success’ for Software Projects: An Explanatory Revelation.” International Journal of Project Management 24, 358-370.

2. Glass, R. L. (2005). “IT Failure Rates-70 Percent or 10-15 Percent?” IEEE Software, 22(3), 110-112

3. Barry Boehm, et al. «Software cost estimation with COCOMO II». Englewood Cliffs, NJ:Prentice-Hall, 2000

4. Standish Group 2015 Chaos Report - Q&A with Jennifer URL: http://www.infoq.com/articles/standish-chaos-2015 (Дата звернення: Квітень 1, 2019)

5. Lukianova K. Yu. Methods of forecasting of changes in product’s value under its modifications in agile projects. [Text]/ K. Yu. Lukianova/ Zbiór artykułów naukowych. Konferencji Miedzynarodowej Naukowo-Praktycznej - Inżynieria i technologia. Nowoczesne badania podstawowe i stosowane (29.04.2017 - 30.04.2017) - Warszawa: 2017. – 6-8 pp. - ISBN: 978-83-65608-56-7

6. Project Management Institute. A Guide to The Project Management Body of Knowledge - PMBOK (r) Guide – Fourth edition, Project Management Institute, pmi.org, 2008, GM Product=00101095501

7. Fenton, N., & Pfleeger, S. L. (1997). Software Metrics: A Rigorous and Practical Approach, Revised Printing. Massachusetts: PWS Publishing Company.

8. CMMI Product Team. (2006). Capability Maturity Model Integration Version 1.2. Software Engineering Institute, Carnegie Mellon University.

9. Barry Boehm, et al. «Software cost estimation with COCOMO II». Englewood Cliffs, NJ:Prentice-Hall, 2000

10. Holgeid K. A Reflection on Why Large Public Projects Fail. [Electronic resource] / K. Holgeid M. Thompson – Mode of access: www/URL: https://www.jbs.cam.ac.uk/fileadmin/user_upload/programmes/emba/downloads/A_Re

flection_on_Why_Large_Public_IT_Projects_Fail_Kjetil__Mark_Thompson_s_chapter.pdf (Дата звернення:Лютий 19 2019)

11. Usmani F. Tools to Estimate Costs in the Project Management [Electronic resource] / F. Ushmani - Mode of access: www/URL: https://pmstudycircle.com/2012/06/4-tools-to-estimate-costs-in-the-project-management/ (Дата звернення: Лютий 9 2019)

12. Yeo, K. T. (2002). Critical failure factors in information system projects. International Journal of Project Management 20(3): 241-246.

13. Serverless Architectures / Fowler, M. URL: https://martinfowler.com/articles/serverless.html (Дата звернення: Лютий 9 2019)

14. Düüna, K. Secure Your Node.js Web Application/ K. Düüna// The Pragmatic Programmers, LLC. – 2016.

15. OWASP Top 10 Application Security Risks URL: https://www.owasp.org/index.php/Top_10-2017_Top_10. (Дата звернення: Квітень 8, 2019)

16. Sbarski, P. Serverless Architectures on AWS/ P. Sbarski// Manning Publications Co. – 2017.

17. DeMarco, T., & Lister, T. (1987). Peopleware: Productive Projects and Teams. New York, NY: Dorset House Publishing Company

18. Devaux, Stephen A. (2015). *Total Project Control (2nd Edition): A Practitioner's Guide to Managing Projects as Investments*. CRC Press. ISBN 978-1498706773.

19. Hyvari, I. (2006). “Project Management Effectiveness in Project-oriented Business Organizations.” International Journal of Project Management, 24, 216-225.

20. Verner, J. M., Evanco, W. M. (2005). “In-house Software Development: What Project Management Practices Lead to Success?” IEEE Software, 22(1), Jan.-Feb., 86– 93.