НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.4

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Метод та засіб створення програмного забезпечення формування робочої групи»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІП-02мп
Ковинєв Кирило Олексійович _____

Керівник:
Старший викладач,
Вітковська Ірина Іванівна _____

Рецензент:
к.т.н., проф.,
Писарчук Олексій Олександрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Ковинєву Кирилу Олексійовичу

1. Тема дисертації «Метод та засіб створення програмного забезпечення формування робочої групи», науковий керівник дисертації старший викладач Вітковська Ірина Іванівна, затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с.

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об’єкт дослідження – процеси побудови та функціонування робочої групи.

4. Предмет дослідження – моделі формування робочих груп і засіб створення програмного забезпечення для формування робочої групи проекту.

5. Перелік завдань, які потрібно розробити – аналіз проблеми формування робочої групи; аналіз існуючих рішень; вдосконалення математичного методу для формування робочої групи; розробка програмного забезпечення формування робочої групи; дослідження ефективності розробленого програмного забезпечення.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – Схема структурна опису баз даних, схема структурна архітектури програмного забезпечення, діаграма варіантів використання.

7. Орієнтовний перелік публікацій – одна доповідь на науковій конференції.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9.Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Аналіз проблеми формування робочої групи	15.09.2021	
2	Аналіз існуючих рішень	1.10.2021	
3	Аналіз та вдосконалення математичного методу для формування робочої групи	15.10.2021	
4	Розробка програмного забезпечення формування робочої групи	29.10.2021	
5	Виконання експериментальних досліджень	7.11.2021	
6	Оформлення пояснювальної записки	19.11.2021	
7	Подання дисертації на попередній захист	22.11.2021	
8	Подання дисертації на захист	6.12.2021	

Студент Кирило КОВИНЄВ

Науковий керівник Ірина ВІТКОВСЬКА

РЕФЕРАТ

Актуальність. Вперше розглядається структура робочої групи проекту як економічна складова, адже якість результату на пряму залежить від виконавців. Створення програмного забезпечення для вирішення проблеми формування робочої групи в залежності від розміру проекту та скорочення цього процесу є актуальною проблемою на сьогоднішній день.

Мета дослідження: основна мета даної роботи полягає в покращенні процесу формування робочих груп шляхом дослідження математичних методів і моделей та розробки програмного засобу для вирішення проблеми формування робочої групи з урахуванням впливу структури групи на ефективність проекту.

Для досягнення поставленої мети були сформульовані **наступні завдання** дослідження:

- аналіз проблеми формування робочої групи;
- аналіз існуючих рішень формування робочих груп та команд;
- аналіз та вдосконалення математичного забезпечення для побудови набору працівників робочої групи;
- вибір та обґрунтування необхідних програмних засобів для розробки програмного забезпечення розрахункової частини;
- розробка бібліотеки на основі методу оцінки вартості програмного забезпечення та веб-додатку.

Об’єкт дослідження: робоча група, процеси її життєдіяльності.

Предмет дослідження: програмне забезпечення для формування робочої групи проекту.

Наукова новизна. Найбільш суттєвими науковими результатами магістерської дисертації є:

- проведений аналіз актуальності задачі вибору та формування підходу до створення групи проекту, існуючих моделей і методів для її вирішення;
- вперше використано у програмному забезпеченні моделі оцінки вартості ПЗ для вирішення проблеми формування робочої групи.

Практичне значення отриманих результатів визначається тим, що запропоновані математичні методи доведені до практичної реалізації у рамках програмного забезпечення, для формування робочої групи.

Зв'язок роботи з науковими програмами, планами і темами. Робота виконувалась на кафедрі інженерії програмного забезпечення Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» в рамках теми «Метод та засіб створення програмного забезпечення формування робочої групи».

Апробація. Основні положення роботи доповідались і обговорювались на конференції «Інженерія програмного забезпечення і передові технології».

Публікації. Наукові положення дисертації опубліковані в матеріалах першої всеукраїнської науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021).

Ключові слова: формування робочої групи, формування команди ІТ-проекту, моделі оцінки вартості ПЗ, COCOMO, COCOMO II.

ABSTRACT

Actuality. The structure of the project working group has been considered as an economic component for the first time because the quality of the result directly depends of the performers. Development of software for solving the problem of forming a working group depending of the project size and reducing this process is an actual problem today.

The purpose of the study: the main purpose of this work is to improve the process of forming working groups by studying mathematical methods and models and developing software to solve the problem of forming a working group taking into account the impact of group structure on project effectiveness.

To achieve this goal, the following **research objectives** were formulated:

- analysis of the problem of forming a working group;
- analysis of existing decisions on forming working groups and teams;
- analysis and improvement of mathematical software for building a set of employees of a working group;
- selection and justification of the necessary software tools for developing software for the calculation part;
- development of a library based on the method of estimating the cost of software and web application.

The object of research: a working group, its activity processes.

The subject of research: software for forming a project working group.

Scientific novelty. The most significant scientific results of the master's dissertation are:

- analysis of actuality of the problem of a choice and formation of the approach to creation of a project group, existing models and methods for its decision;
- the model of software evaluation to solve the problem of forming a working group was used in the software for the first time.

Practical significance of the obtained results is determined by the fact that the proposed mathematical methods have been brought to practical implementation within the software for formation of a working group.

Connection of the work with scientific programs, plans, and topics. The work was performed at the Department of Software Engineering of the National Technical University of Ukraine "Kyiv Polytechnic Institute named after Igor Sikorsky" within the theme "Method and means of creating software for the formation of the working group".

Approbation. The main provisions of the work were reported and discussed at the conference "Software Engineering and Advanced Technologies".

Publications. The scientific provisions of the dissertation were published in the materials of the first all-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies" (SoftTech-2021).

Keywords: formation of a working group, formation of an IT project team, software evaluation models, COCOMO, COCOMO II.

ЗМІСТ

ВСТУП 10

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 11

1.1 Опис предметної області 11

1.2 Аналіз існуючих рішень 18

1.2.1 Калькулятор COCOCMO 18

1.2.2 COCOMO II – Constructive Cost Model 19

1.2.3 Monday.com 20

1.3 Постановка задачі 22

1.4 Висновки по розділу 24

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ФОРМУВАННЯ РОБОЧОЇ ГРУПИ 25

2.1 Моделі оцінки вартості програмного забезпечення 25

2.1.1 Модель Путнема 25

2.1.2 Модель COCOMO 26

2.1.3 Модель COCOMO II 32

2.1.4 Вибір моделі оцінки вартості ПЗ 38

2.2 Висновки до розділу 41

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧОГО ЗАБЕЗПЕЧЕННЯ 42

3.1 Засоби розробки 42

3.1.1 .NET 42

3.1.2 C# 43

3.1.3 Microsoft SQL Server 44

3.1.4 Angular 45

3.1.5 PrimeNG 45

3.2 Вимоги до програмного продукту 46

3.3 Опис структури бази даних 47
3.4 Інструкція користувача 52
3.5 Випробування програмного продукту 60
3.6 Експериментальне дослідження запропонованих моделей 64
3.7 Висновки до розділу 74
4 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ 75
4.1 Опис ідеї проєкту 75
4.2 Технологічний аудит ідеї проекту 76
4.3 Аналіз ринкових можливостей запуску стартап-проєкту 77
4.4 Розроблення ринкової стратегії проєкту 86
4.5 Висновки до розділу 93
ВИСНОВКИ 94
ПЕРЕЛІК ПОСИЛАНЬ 96
ДОДАТОК А Структурна схема бази даних 98
ДОДАТОК Б Діаграма варіантів використання 99
ДОДАТОК В Схема архітектури програмного забезпечення 100
ДОДАТОК Г Звіт перевірки на плагіат 101
ДОДАТОК Д Лістинг коду бібліотеки на основі методу оцінки вартості 102

ВСТУП

Створення робочої групи для виконання нового проекту – одне з основних завдань менеджера проекту на першому етапі виконання роботи над проектом. На даний момент велика кількість проект-менеджерів вирішують питання формування групи за допомогою відбору працівників, в залежності від компетенції, здібностей роботи у команді та аналітичних навичок кожного окремого працівника, на власний розсуд.

Такий підхід може призвести до:

- збільшення термінів формування команди;
- формування набору працівників, що не вкладуться у визначені терміни виконання проекту, або не виконають проект взагалі;
- формування набору працівників, що не вкладуться у визначений бюджет виконання проекту.

Наявність системи для формування робочої групи, яка розрахує показники кожного набору працівників, вирішить проблему зриву термінів, визначених бюджетів, та прискорить процес формування команди. У ході виконання порівняльного аналізу існуючих систем формування робочих груп для виконання проектів наявного рішення не було виявлено.

Отже існує потреба у створенні програмного забезпечення, що на основі існуючих вхідних даних про проект та працівників компанії, розрахує інформацію про кожен набір співробітників. Таким чином ПЗ прискорить час та збільшить ефективність виконання етапу роботи формування команди проект-менеджером, а також знизить ризики невдалого виконання проекту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

В умовах постійних змін, зокрема, загострення конкуренції, все частіше багато товарів, робіт та послуг створюються на основі проектного підходу, що дозволяє досягти запланованих результатів. Проектне управління пронизує усі функції підприємства та ефективне для всіх рівнів організації, що доведено практичним застосуванням розвиненими країнами. У той же час, не дивлячись на популярність та ефективність методів управління проектами, згідно зі статистичними даними, більше 40% проектів виявляються невдалими або не завершуються взагалі. Причини невдач, частіш за все, пов’язані з культурою підприємництва при формуванні команди проекту. Тому, актуальним питанням на сьогоднішній день є ефективне управління командою проекту, зокрема, формування та розвиток команди [1].

Створення програмного продукту є дуже великою задачею, тому її вирішення самотужки є доволі тяжким або інколи взагалі неможливим. Коли окремі люди бачать завдання, які вони не можуть вирішити поодинці, формуються групи за допомогою менеджера. Кожна група володіє наступними основними характеристиками: взаємодія, цілі, взаємозалежність, структура та єдність. Відмінністю команди від групи є інтенсивність кожного з цих атрибутів. Рівень взаємодії у командах концентрований та неперервний, та включає як цілеспрямовану дію, так і взаємодію, яка підтримує взаємовідносини (соціальна підтримка, взаємодопомога). Обов’язкова умова для роботи у команді – це прагнення до цілей, причому колективним. У команді успіх та невдача трапляються там, де усі учасники розділяють результат незалежно від їх особистих показників. Таким чином команди підкреслюють результати, що їх існування знаходиться під загрозою, якщо вони не зможуть досягти узгоджених цілей. Усі члени групи у певній мірі взаємозалежні, але члени команди настільки тісно пов’язані, що кожен

результат кожного окремого учасника нерозривно пов’язаний з результатом одне одного. Передбачається, що кожен член команди має спеціалізовані знання, навички та здібності, які він або вона вносять у команду. Успіх команди залежить від ефективного об’єднання цих окремих внесків. Також команди є добре структурованою групою. Члени спортивних команд, таких як футбол або бейсбол, знають у чому їх роль через особисте положення, що вони займають у команді. Нарешті, тісний зв’язок учасників команд означає, що вони мають велику міру єдності: команди зазвичай є згуртованими, особливо у тому сенсі, що їх члени єдині у своїх зусиллях для досягнення загальної цілі. Зовнішній тиск може збільшити єдність, наприклад: велике навантаження, обмежений час або змагання з іншими групами [2].

З цього випливає, що під поняттям команда мається на увазі колектив (об’єднання людей, що здійснюють спільну діяльність і володіють спільними інтересами), здатний досягати мети автономно й узгоджено, при мінімальних керуючих впливах [3]. Суттєвими в наведеному визначенні команди є два аспекти. Перший – досягнення мети, тобто кінцевий результат спільної діяльності є для команди системоутворюючим фактором. Другий аспект – автономність і узгодженість діяльності – означає, що кожен з членів команди демонструє поведінку, яка вимагається у даних умовах [4].

Команди бувають найрізноманітніших форм і можуть виконувати безліч різних функцій у військових, освітніх, промислових, корпоративних та науково-дослідних сферах. Однак можна зробити загальну відмінність між командами, які обробляють інформацію, та командами які планують, практикують та виконують діяльність:

– *виконавчі команди* – визначають і вирішують проблеми, приймають рішення про повсякденні операції та виробництва, а також ставлять цілі для майбутнього організації;

- *проектні команди*, до складу яких входять особи з різним походженням та галуззю знань, які об’єднуються для розробки інноваційних продуктів та визначення нових рішень для існуючих проблем;
- *консультативні команди,* такі як групи по огляду та керівні комітети, звуться інколи паралельними командами, тому що вони працюють поза звичних наглядових структур компаній;
- *робочі команди*, такі як виробничі бригади та бригади з технічного обслуговування, несуть відповідальність за матеріальний результат організації. Вони створюють продукцію або надають послуги;
- *дійові команди*, до складу яких входять спортивні команди, хірургічні команди, наряди поліції, військові частини та оркестри. Вони є спеціалізованими командами, що надають послугу за рахунок високо координованих дій [2].

Таким чином, можна сказати, що команда ІТ-проекту – це колектив або робоча група, що здійснює спільну діяльність, та до якої входять особи з різним походженням та галуззю знань, які об’єднуються для розробки інноваційних продуктів. Також кожен з членів групи повинен демонструвати поведінку (роль), яка вимагається у даних умовах.

ІТ-проектом може бути будь-який тип проекту, який стосується ІТ-інфраструктури, інформаційних систем або комп’ютерних технологій. Це може включати діяльність з розробки програмного забезпечення, наприклад програмування простого мобільного додатка або великомасштабної програмної системи. Управління ІТ-проектами для початківців іноді включає веб-розробку, включаючи оновлення веб-сторінки, створення сайту для онлайн-покупок або навіть розробку всієї веб-інфраструктури.

Інші поширені приклади ІТ-проектів включають проектування ІТ-інфраструктури організації, розгортання систем і програмного забезпечення та застосування заходів безпеки ІТ.

Нижче наведено кілька категорій IT-проектів:

– дослідження;
– обслуговування;
– розробка програмного забезпечення;
– розгортання системи;
– управління змінами;
– інфраструктура;
– оцінка потреб [5].

Виходячи з цього, для вирішення проблеми, коли проект виявляється невдалим або не завершується взагалі, треба сформувати такий набір робітників компанії, відштовхуючись від галузі їх знань та поведінки (досвід розробки, аналітичні здібності та інше), що зможуть вдало завершити проект.

Існують традиційні методи та підходи формування команди:

– системний підхід;
– метод аналогій;
– експериментально-аналітичний підхід;
– параметричний підхід;
– блочний підхід;
– метод моделювання;
– метод структуризації цілей;
– підхід на основі досвіду.

Суть цих методів та підходів, їх переваги та недоліки вказані далі.

Системний підхід – підхід, у якому будь-яка система сприймається як сукупність взаємозалежних елементів, має вихід, вхід, зв'язок із довкіллям та зворотний зв'язок.

Переваги: Концентрує увагу на цілісність структури. Показує взаємозалежність.

Недоліки: Складний підхід для застосування на практиці. Потребує багаторівневого вивчення предмета. Не дає можливості розглядати проблему ізольовано. Не враховує розвитку команди проекту.

Метод аналогій – метод, суть його полягає в тому, що структура проекту створюється на основі розглянутих групою експертів прямих чи непрямих аналогій компанії.

Переваги: Використовується минулий досвід для виключення проблем у майбутньому. Формування різних команд проектів містить спільні риси.

Недоліки: Потрібно витрачати багато часу. Легко отримати непотрібні та невідповідні результати. Аналогія може бути некоректною. Не впливає на розвиток команди проекту.

Відповідно до *експериментально-аналітичного підходу*, команда може бути сформована експериментальним шляхом. Це передбачає внесення змін та аналіз сформованої команди доти, доки її ефективність не підвищиться.

Переваги: Комбінований метод, дає точніший результат. Сприяє формуванню ефективної команди.

Недоліки: При побудові доводиться підлаштовуватися під дані експерименту, а в експериментальних моделях закладаються апріорні відомості про об'єкт. Не враховує компетентності та розвитку команди проекту. Потрібна думка експертного співробітника. Труднощі у залученні незалежних експертів. Суб'єктивність оцінок.

Параметричний підхід визначає параметри та характеристики, які мають відповідати членам групи. Тільки після цього здійснюється підбір команди.

Переваги: Дає можливість підібрати команду за певними параметрами, компетенціями. Об'єктивний метод.

Недоліки: Не всі психологічні характеристики можна висловити у якості параметрів. Психологічний стан команди проекту складно висловити у кількісному та якісному виразі. Необхідне залучення професіоналів для отримання більш точного результату.

При *блочному підході* спочатку створюються невеликі групи, які можна назвати блоками, і лише потім формується команда шляхом їх об'єднання.

Переваги: Дозволяє уникнути помилок. Помилки будуть виявлені наперед, на рівні блоків.

Недоліки: Потрібен тимчасовий ресурс для трансформації блоків у команду. Не враховує розвитку команди проекту.

Метод моделювання передбачає формування команди, спираючись на наукову економічну модель.

Переваги: Дає можливість сформувати складні системи та об'єкти.

Недоліки: Не враховує розвитку персоналу у команді проекту.

Метод структуризації цілей визначає цілі та завдання, виходячи з яких відбувається формування команд.

Переваги: Чітко визначено цілі та завдання. Дозволяє підібрати команду проекту виходячи з цілей, функціональних обов'язків.

Недоліки: Завдання під час реалізації проекту можуть змінюватися, збільшуватись, або розширюватись. Чи не передбачає розвиток.

Підхід на основі досвіду має безліч схожостей з експериментально-аналітичним методом – різниця лише в тому, що оцінку дає не група експертів, а керівник, спираючись на результати діяльності.

Переваги: Дає точний результат. Сприяє формуванню ефективної команди. Спирається на попередній досвід.

Недоліки: Можливе використання помилкових відомостей. Керівник може неправильно оцінити критерії відбору. Не згадується розвиток персоналу [1].

Таким чином, кожен підхід має свої переваги та недоліки. Однак, не для кожного проекту потрібне використання всіх перерахованих підходів.

Так як проект – це діяльність, спрямована на створення певного продукту чи послуги протягом визначеного терміну та за певних фінансових обмежень [6], оцінкою якості сформованої команди можна вважати вартість та час виконання проекту. Для визначення вартості та часу виконання ІТ-проекту використовуються методи та моделі оцінки вартості програмного забезпечення.

До популярних моделей належать COCOCMO та COCOMO II. Для підрахунку вартості ПЗ вони використовують такі фактори витрат, як:

- характеристики продукту;
- характеристики персоналу;
- характеристики апаратного забезпечення;
- характеристики проекту [7].

Таким чином, для підрахунку факторів витрат виникає потреба у збереженні інформації про характеристики кожного окремого працівника компанії та заявлені замовником характеристики проекту та продукту.

1.2 Аналіз існуючих рішень

На сьогоднішній день, існуючі рішення можна розділити на дві категорії:

– калькулятори оцінки вартості програмного забезпечення, за допомогою яких, на основі вхідних даних характеристик проекту та персоналу, вираховується вартість та час виконання проекту;
– інформаційні системи для управління IT-проектами, де команди формуються вручну.

1.2.1 Калькулятор COCOCMO

Калькулятор COCOCMO – веб-застосунок, що використовує конструктивну модель вартості (COCOCMO) для підрахунку трудомісткості, часу виконання та кількості персоналу для проекту.

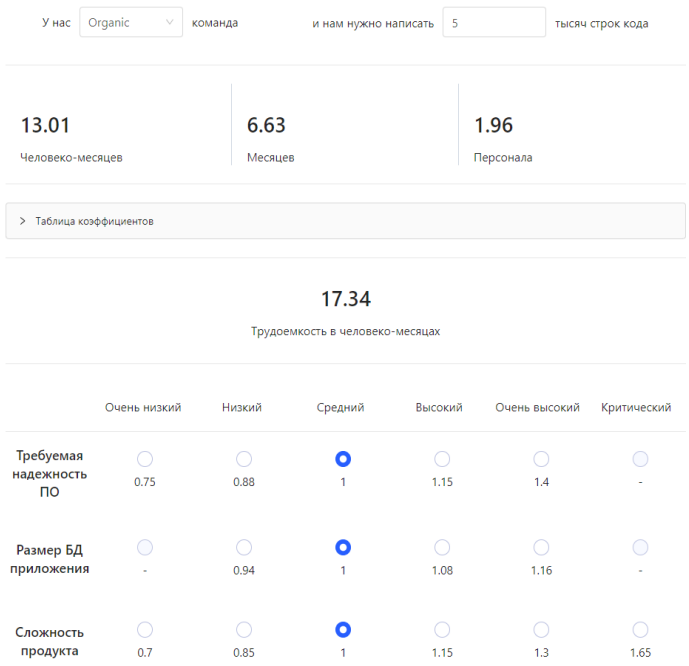


Рисунок 1.1 – Калькулятор COCOMO.

На рис.1.1 показано скріншот використання застосунку.

Переваги рішення:

– калькулятор має зручний та динамічний інтерфейс;
– застосунок має інформаційну довідку про те, що таке COCOCMO та як користуватися калькулятором.

Недоліки рішення:

– калькулятор не вираховує вартість проекту;
– не має можливості зберегти дані про персонал та проект, щоб система сама сформувала вхідні дані та сформувала робочу групу проекту;
– не має можливості зберегти результат підрахунків.

1.2.2 COCOMO II – Constructive Cost Model

COCOMO II – Constructive Cost Model – веб-застосунок, що використовує COCOMO Model 2 для підрахунку трудомісткості, часу виконання та кількості персоналу для проекту.

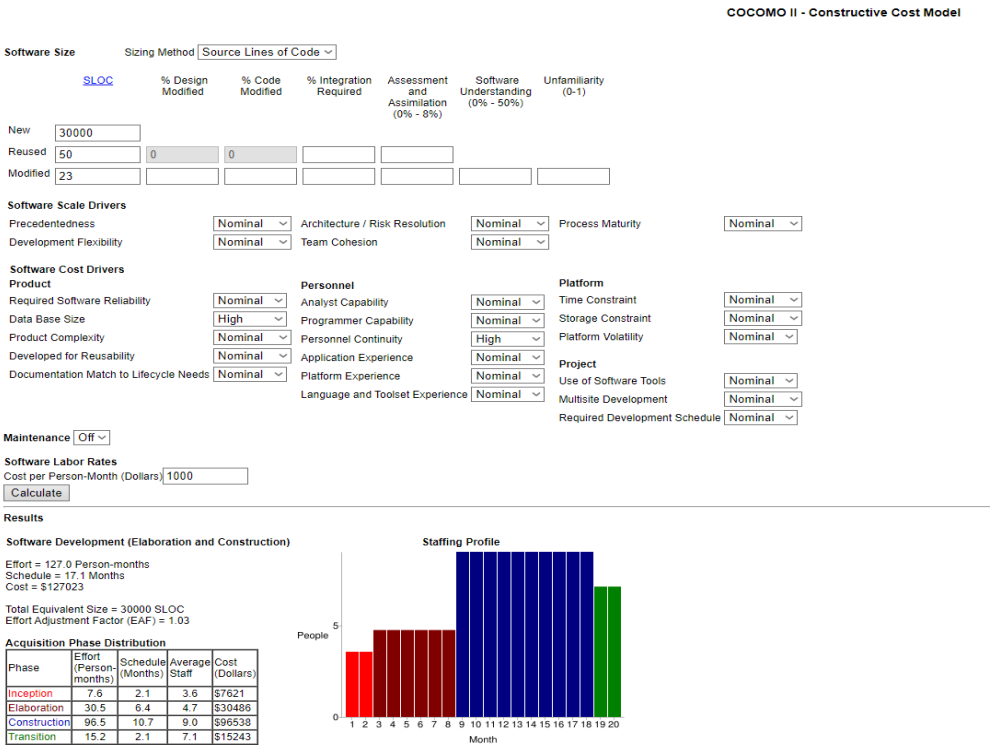


Рисунок 1.2 – COCOMO II – Constructive Cost Model

Переваги рішення:

– порівняно з попереднім рішенням, калькулятор підраховує більше оцінок, має таблицю розподілів фаз проекту та графік виконання фаз.

Недоліки рішення:

– калькулятор має застарілий інтерфейс;
– не містить довідки про користування калькулятором;
– не має можливості зберегти дані про персонал та проект, щоб система сама сформувала вхідні дані та сформувала робочу групу проекту;
– не має можливості зберегти результат підрахунків.

1.2.3 Monday.com

Monday.com – інструмент управління проектами, розрахований як на команди, так і на поодиноких користувачів. Інструмент управління, моніторингу та обміну даними з співробітниками, партнерами і замовниками, він дозволяє управляти як проектами, так і окремими оперативними або особистими завданнями, координувати діяльність в команді і з партнерами.

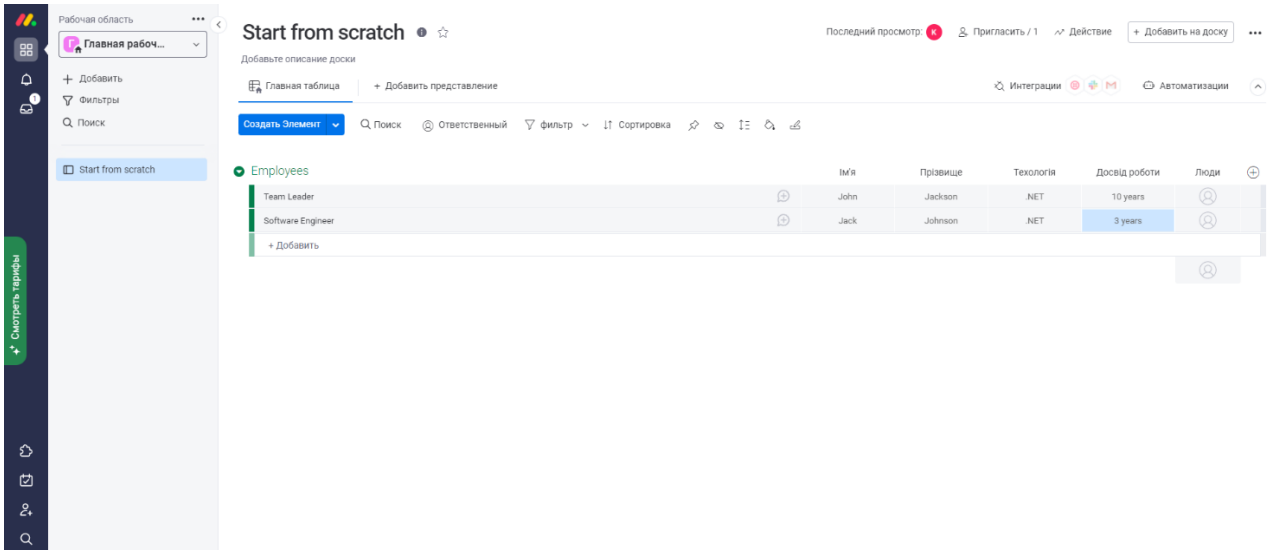


Рисунок 1.3 – Monday.com

Переваги рішення:

– застосунок має зручний сучасний інтерфейс, легкий у використанні;

– застосунок надає можливість зберігати необхідну інформацію про характеристики персоналу та проекту для оцінки вартості програмного забезпечення.

Недоліки рішення:

– немає можливості підрахування вхідних даних для методів оцінки вартості програмного забезпечення;
– немає калькулятору оцінки вартості програмного забезпечення для підрахунку якості формування робочої групи;
– розподілення персоналу для проектів відбувається вручну.

Також до існуючих рішень другої групи, куди входять інформаційні системи для управління IT-проектами, де команди формуються вручну, і які дуже схожі за функціоналом на Monday.com, можна віднести:

– Asana;
– Scoro;
– Basecamp;
– Trello;
– Quire.

Проаналізувавши існуючі рішення, можна зробити висновок, що на сьогоднішній день не має аналогів, які б задовольнили потребу у вирішенні проблеми формування робочої групи IT-проекту.

1.3 Постановка задачі

Метою роботи є покращення процесу формування робочих груп проекту шляхом вдосконалення математичного методу оцінки вартості програмного забезпечення для створення програмного забезпечення формування робочої групи IT-проекту.

Створений продукт повинен відповідати наступним функціональним вимогам:

- можливість збереження/редагування/видалення інформації про характеристики кожного окремого працівника, проекту та продукту;
- можливість підрахунку вхідних даних для методів оцінки вартості програмного забезпечення;
- можливість підрахунку вартості та часу виконання проекту за допомогою методів оцінки вартості програмного забезпечення;
- формування наборів працівників, чия робота у команді буде найшвидшою при мінімальних затратах для кожного окремого проекту;
- можливість перегляду докладної інформації розрахунків методів оцінки вартості програмного забезпечення.

Також створений продукт повинен відповідати наступним вимогам:

- інтуїтивно зрозумілий інтерфейс;
- простота у використанні;
- можливість додання нових модулів для оцінки вартості проекту;
- велика швидкість підрахунків для формування робочої групи.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

- проаналізувати існуючі методи оцінки вартості програмного забезпечення;
- спроектувати та створити базу даних;

- спроектувати архітектуру програмного забезпечення;
- вдосконалити формування вхідних даних для методів оцінки вартості ПЗ;
- розробити бібліотеку на основі методу оцінки вартості ПЗ та веб-додаток;
- розробити стартап-проект для ПЗ.

1.4 Висновки по розділу

У першому розділі було виконано аналіз предметної області, висвітлення та аналіз вирішення проблеми вдалості та завершеності проекту, та її взаємозв’язок з формуванням робочої групи.

Також було проаналізовано сучасні розробки для вирішення проблеми формування команди, поділено їх на дві категорії та визначено їх основні переваги та недоліки. На основі цього аналізу був зроблений висновок про актуальність розробки програмного забезпечення для формування робочої групи проекту та сформульовані функціональні вимоги.

На основі аналізу предметної області та існуючих рішень сформульована мета розробки, вимоги до продукту та задачі розробки.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ФОРМУВАННЯ РОБОЧОЇ ГРУПИ

2.1 Моделі оцінки вартості програмного забезпечення

2.1.1 Модель Путнема

Модель Лоуренса Путнема описує час і зусилля, необхідні для завершення програмного проекту певного розміру. Путнем помітив, що кадровий склад програмного забезпечення слідує за добре відомим розподілом Релея. Путнем використав своє спостереження про рівні продуктивності, щоб отримати формулу (1):

$$L = C_k K^{\frac{1}{3}} t_d^{\frac{4}{3}}, \quad (1)$$

де L – оцінка продукту у кількості строк коду;

K – це загальні зусилля витрачені на розробку продукту, вимріюється у людино-рік;

t_d – співвідноситься з часом тестування системи та інтеграції. Тому t_d можна відносно розглядати як час необхідний для розробки продукту.

C_k – це є константою стану технології і відображає вимоги, що перешкоджають розробці програми.

Основними значеннями C_k = 2 для поганого середовища розробки, C_k = 8 для доброго середовища розробки програмного забезпечення, так C_k = 11 для бездоганного середовища розробки.

Точне значення C_k для конкретного завдання можна обчислити з історичних даних організації, яка його розробляє [8].

Так як програмне формула не містить змінних залежних від характеристики робочої групи, модель Путнема не підійде для розрахунків формування робочої групи проекту.

2.1.2 Модель COCOMO

Конструктивна модель вартості (COCOMO) – алгоритмічна модель оцінки вартості програмного забезпечення, розроблена на початку 80-х років Баррі Боемом [9].

Модель складається з трьох рівнів:

– базовий рівень;
– проміжний рівень;
– детальний рівень.

На кожному рівні усі проекти розбиваються на три групи по рівню складності:

– розповсюджений тип;
– вбудований тип;
– напівнезалежний тип;

До розповсюдженого типу відносяться наступні характеристики: невелика команда розробників, що має досвід у створені подібних проектів; продукту необхідні невеликі нововведення; умови роботи та середовище розробки є стабільними; проект має невисоку складність виконання.

До вбудованого типу відносяться: велика команда розробників проекту; високий степінь новизни; жорсткі обмеження і терміни здачі; вимагає великих витрат на зміни і виправлення; середовище розробки продукту має велику кількість складних інтерфейсів (деякі надаються замовниками продукту разом з апаратним забезпеченням).

Напівзалежний тип можна віднести до проміжного положення між вбудованим і розповсюдженим типами. Йому притаманні наступні характеристики: невелика команда розробників; продукт має невеликі елементи новизни; помірні обмеження і наявність кінцевого терміну; частина вимогу до продукту фіксується, в іншому спеціалісти мають вибір.

2.1.2.1 Базовий рівень

Базовий рівень розраховує трудоємність та вартість розробки як функцію від розміру програми. Розмір виражається в оціночних тисячах рядків коду.

Базовий рівень моделі має наступні формули:

$$PM = a_i * (SIZE)^{b_i}, \quad (2)$$

$$TM = c_i * (PM)^{d_i}, \quad (3)$$

$$SS=PM/TM, \quad (4)$$

$$P=SIZE/PM, \quad (5)$$

де PM – трудоємність (людино-місяць);

SIZE – об’єм програмного продукту (вимірюється в тисячах рядків вихідного коду);

TM – термін розробки або тривалість (вимірюється в місяцях);

SS – середня чисельність персоналу;

P – продуктивність.

Коефіцієнти a_i, b_i, c_i, d_i для формул (2) та (3) вибираються за таблиці 2.1.

Таблиця 2.1 – Коефіцієнти формули базового рівня COCOMO

Тип проекту	a	b	c	d
Розповсюджений	2.4	1.05	2.5	0.38
Напівнезалежний	3.0	1.12	2.5	0.35
Вбудований	3.6	1.20	2.5	0.32

Формула базового рівня моделі підходить для приблизної оцінки витрат розробки програмного продукту. Проте, із-за відсутності врахування чинників кваліфікації та характеристики команди, що працює над продуктом, базовий рівень не підходить для розрахунків формування робочої групи проекту.

2.1.2.2 Середній рівень

Середній рівень розраховує трудоємність розробки як функцію від розміру програми та множини «факторів вартості», що включають суб’єктивні оцінки характеристик продукту, проекту, команди розробників та апаратного забезпечення. Це розширення включає у себе множину з чотирьох факторів, кожен з яких має декілька дочірніх характеристик:

– характеристики продукту;
– характеристики проекту;
– характеристики апаратного забезпечення;
– характеристики персоналу.

Характеристики продукту мають такі дочірні характеристики:

– Required Software Reliability – RELY (Потрібна надійність програмного забезпечення);
– Size of Application Database – DATA (Розмір бази даних продукту);
– Complexity of Product – CPLX (Складність продукту).

Характеристики проекту мають такі дочірні характеристики:

- Use of Software Tools – TOOL (Використання інструментів розробки програмного забезпечення);
- Application of Software Engineering Methods – MODP (Застосування методів розробки програмного забезпечення);
- Required Development Schedule – SCED (Вимоги дотримання графіку роботи).

Характеристики апаратного забезпечення мають такі дочірні характеристики:

- Run-Time Performance Constraints – TIME (Обмеження швидкодії при виконанні програми);
- Memory Constraints – STOR (Обмеження пам’яті);
- Required Turnabout Time – TURN (Необхідний час відновлення);
- Volatility of the Virtual Machine Environment – VIRT (Нестійкість оточення віртуальної машини).

Характеристики персоналу мають такі дочірні характеристики:

- Analyst Capability – ACAP (Аналітичні здібності);
- Programming Language Experience – LEXP (Досвід розробки на мовах програмування);
- Applications Experience – AEXP (Досвід розробки);
- Software Engineer Capability – PCAP (Здібності до розробки програмного забезпечення);
- Virtual Machine Experience – VEXP (Досвід використання віртуальних машин).

Формула середнього рівня моделі має наступний вигляд:

$$PM = EAF * a_i * (SIZE)^{b_i}, \quad (6)$$

де PM – трудоємність (людей*місяць);

EAF – добуток обраних атрибутів вартості;

SIZE – об’єм програмного продукту (вимірюється в тисячах рядків вихідного коду).

Атрибути вартості вибираються з таблиці 2.2.

Таблиця 2.2 – Значення атрибутів вартості

Атрибути вартості	Рейтинг					
	дуже низький	низький	середній	високий	дуже високий	критичний
Характеристики продукту						
1. Необхідна надійність ПЗ	0.75	0.88	1.00	1.15	1.40	n/a
2. Розмір БД продукту	n/a	0.94	1.00	1.08	1.16	n/a
3. Складність продукту	0.70	0.85	1.00	1.15	1.30	1.65
Характеристики апаратного забезпечення						
4. Обмеження швидкодії при виконанні програми	n/a	n/a	1.00	1.11	1.30	1.66
5. Обмеження пам'яті	n/a	n/a	1.00	1.06	1.21	1.56
6. Нестійкість оточення віртуальної машини	n/a	0.87	1.00	1.15	1.30	n/a
7. Необхідний час відновлення	n/a	0.87	1.00	1.07	1.15	n/a
Характеристики персоналу						
8. Аналітичні здібності	1.46	1.19	1.00	0.86	0.71	n/a
9. Досвід розробки	1.29	1.13	1.00	0.91	0.82	n/a
10. Здібності до розробки ПЗ	1.42	1.17	1.00	0.86	0.70	n/a
11. Досвід використання віртуальних машин	1.21	1.10	1.00	0.90	n/a	n/a
12. Досвід розробки на мовах програмування	1.14	1.07	1.00	0.95	n/a	n/a

Кінець таблиці 2.2

Характеристики проекту						
13. Застосування методів розробки ПЗ	1.24	1.10	1.00	0.91	0.82	n/a
14. Використання інструментарію розробки	1.24	1.10	1.00	0.91	0.83	n/a
15. Вимоги дотримання графіка розробки	1.23	1.08	1.00	1.04	1.10	n/a

Коефіцієнти a_i, b_i для формули (6) середнього рівня вибираються з таблиці 2.3.

Таблиця 2.3 – Коефіцієнти формули середнього рівня COCOMO

Тип проекту	a	b
Розповсюджений	3.2	1.05
Напівнезалежний	3.0	1.12
Вбудований	2.8	1.20

Наявність у формулі середнього рівня моделі таких чинників, як аналітичні здібності команди, досвід розробки, здібності до розробки програмного забезпечення, досвід використання віртуальних машин та досвід розробки на мовах програмування дозволяє використовувати цей рівень для підрахунків формування робочої групи проекту.

2.1.2.3 Детальний рівень

Детальний рівень включає у себе усі характеристики середнього рівня з оцінкою впливу даних характеристик на кожен етап процесу розробки програмного забезпечення.

2.1.3 Модель COCOMO II

Модель COCOMO II, котра була придумана у 1997 році, є вдосконаленою версією моделі COCOMO. Оцінку проекту розбивають на дві стадії: попередня оцінка на початковій фазі та детальна оцінка після проробки архітектури.

Формула оцінки трудоємності проекту має наступний вигляд:

$$PM = EAF * \text{A} * (SIZE)^E, \quad (7)$$

$$E = B + 0.01 * \sum_{j=1}^{5} SF_j; \quad (8)$$

де EAF – добуток обраних атрибутів вартості;

B=0.91, A=2.94 для попередньої оцінки, A=2.45 для детальної оцінки;
SIZE – об’єм програмного продукту (вимірюється в тисячах рядків вихідного коду);

SF_j – фактори масштабу.

В моделі COCOMO II використовують п’ять факторів масштабу SF_j (на обох стадіях оцінки проекту), які визначаються наступними характеристиками проекту:

- Precedentedness – PREC (Прецедентність) – визначається за наступними категоріями: абсолютно безпрецедентний, багато в чому безпрецедентний, дещо безпрецедентний, загалом знайомий, багато в чому знайомий, абсолютно знайомий;
- Development Flexibility – FLEX (Гнучкість проекту розробки) – визначається за наступними категоріями: процес строго детермінований, процес нестрого детермінований, загальні положення, деякі положення, визначені тільки загальні цілі;
- Architecture / Risk Resolution – RESL (Архітектура і дозвіл ризиків) – вимріюється у відсотках від 20% (ризики невідомі) до 100% (ризики вирішені на 100%);
- Team Cohesion – TEAM (Спрацьованість команди) – визначається за наступними категоріями: дуже важкі взаємодії, іноді важкі взаємодії, стандартні кооперативні взаємодії, загалом кооперативні взаємодії, у більшості кооперативні взаємодії, повна взаємодія та взаємодопомога;
- Process Maturity – PMAT (Зрілість процесів).

Всі фактори масштабу мають певну оцінку свого рівня: дуже низький, низький, середній, високий, дуже високий та критичний.

Числові значення фактору масштабу в залежності від оцінки його рівня наведені в таблиці 2.4.

Таблиця 2.4 – Значення чинника масштабу

Чинник масштабу	Оцінка рівня чинника					
	Very Low	Low	Nominal	High	Very High	Extra High
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	4.38	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Кількість і значення множників трудоємності ЕМ відрізняються в залежності від стадії оцінки проекту.

Для першої (попередньої) стадії оцінки трудоємності програмного продукту необхідно оцінити рівень семи множників трудоємності ЕМ:

- Personnel Capability – PERS (Кваліфікація персоналу) – вимірюється у відсотках текучості від більше ніж 45% до менш ніж 4%;
- Personnel Experience – PREX (Досвід персоналу) – визначається за наступними категоріями: нові інструменти та платформа, є невеликий досвід використання інструментів та платформи, інструменти та платформа загалом відомі, інструменти та платформа у більшості відомі, інструменти та платформа добре відомі;
- Product Reliability and Complexity – RCPX (Складність і надійність продукту) – визначається за наступними категоріями: спеціальних вимог немає, невелика кількість вимог, середня кількість вимог, велика кількість вимог, жорсткі вимоги;
- Developed for Reusability – RUSE (Розробка для повторного використання) – визначається за наступними категоріями: не потребує, повторне використання у іншому проекті, повторне використання у іншій програмі, повторне використання у іншому продукті, повторне використання у інших продуктах;

– Platform Difficulty – PDIF (Складність платформи розробки);
– Facilities – FCIL (Обладнання);
– Required Development Schedule – SCED (Необхідність виконання графіка робіт) – вимірюється у відсотках номінальної тривалості від 75% до 160%.

Числові значення множників трудоємності для попередньої стадії в залежності від оцінки рівня наведені в таблиці 2.5.

Таблиця 2.5 – Значення множників трудоємності

Множник трудоємності	Оцінка рівня множника						
	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.22	1.00	0.87	0.74	0.62
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	n/a	n/a	0.95	1.00	1.07	1.15	1.24
PDIF	n/a	n/a	0.87	1.00	1.29	1.81	2.61
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	n/a	1.43	1.14	1.00	1.00	n/a	n/a

Для другої (детальної) стадії оцінки трудоємності програмного продукту необхідно оцінити рівень сімнадцять множників трудоємності ЕМ:

– Analyst Capability – ACAP (Можливості аналітика) – вимірюється у відсотках (від 15% до 90%);
– Applications Experience – AEXP (Досвід розробки доданків) – вимріюється у роках (від 0 до 6 років);
– Programmer Capability – PCAP (Можливості програміста) – вимірюється у відсотках (від 15% до 90%);
– Personnel Continuity – PCON (Тривалість роботи персоналу);
– Platform Experience – PEXP (Досвід роботи з платформою) – вимріюється у роках (від 0 до 6 років);

- Language and Tool Experience – LTEX (Досвід роботи з мовами програмування і інструментальних засобів) – вимріюється у роках (від 0 до 6 років);
- Required Software Reliability – RELY (Необхідна надійність програми) – визначається за наступними категоріями: незначна надійність, невисокий рівень надійності, середній рівень надійності, високі фінансові втрати, ризик людському життю;
- Database Size – DATA (Розмір бази даних);
- Software Product Complexity – CPLX (Складність програми);
- Required Reusability – RUSE (Необхідна можливість багаторазового використання) – визначається за наступними категоріями: не потребує, повторне використання у іншому проекті, повторне використання у іншій програмі, повторне використання у іншому продукті, повторне використання у інших продуктах;
- Documentation Match to Life-Cycle Needs – DOCU (Відповідність документації потребам життєвого циклу);
- Execution Time Constraint – TIME (Обмеження часу виконання) – вимріюється у відсотках використання доступного часу виконання від 50% до 95%;
- Main Storage Constraint – STOR (Обмеження пам’яті) – вимріюється у відсотках використання доступного сховища від 50% до 95%;
- Platform Volatility – PVOL (Змінність платформи) – визначається за наступними категоріями: кожні 12 місяців, кожні 6 місяців, кожні 2 місяці, кожні 2 тижні;
- Use of Software Tools – TOOL (Використання інструментальних програмних засобів);
- Multisite Development – SITE (багатоабонентська розробка) – визначається за наступними категоріями: інтернаціональна, декілька міст або компаній, одне місто, одна будівля, одне приміщення;

– Required Development Schedule – SCED (необхідність виконання графіка робіт) – вимірюється у відсотках (від 75% до 160% номінальної тривалості).

Числові значення множників трудоємності для детальної стадії в залежності від оцінки рівня наведені в таблиці 2.6.

Таблиця 2.6 – Значення множників трудоємності

Множник трудоємності	Оцінка рівня множника					
	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	1.42	1.29	1.00	0.85	0.71	n/a
AEXP	1.22	1.10	1.00	0.88	0.81	n/a
PCAP	1.34	1.15	1.00	0.88	0.76	n/a
PCON	1.29	1.12	1.00	0.90	0.81	n/a
PEXP	1.19	1.09	1.00	0.91	0.85	n/a
LTEX	1.20	1.09	1.00	0.91	0.84	n/a
RELY	0.84	0.92	1.00	1.10	1.26	n/a
DATA	n/a	0.23	1.00	1.14	1.28	n/a
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	n/a	0.95	1.00	1.07	1.15	1.24
DOCU	0.81	0.91	1.00	1.11	1.23	n/a
TIME	n/a	n/a	1.00	1.11	1.29	1.63
STOR	n/a	n/a	1.00	1.05	1.17	1.46
PVOL	n/a	0.87	1.00	1.15	1.30	n/a
TOOL	1.17	1.09	1.00	0.90	0.78	n/a
SITE	1.22	1.09	1.00	0.93	0.86	0.80
SCED	1.43	1.14	1.00	1.00	1.00	n/a

Час розробки проекту ТМ в моделі COCOMO II для обох рівнів можна розрахувати за формулою:

$$TM = SCED * C * (PM_{NS})^{B+0.2*(E-B)}, \quad (9)$$

де C = 3.67, D =0.28; PM_{NS} – розрахована трудоємність проекту без урахування множника SCED, що визначає стиснення розкладу.

2.1.4 Вибір моделі оцінки вартості ПЗ

Проаналізував моделі оцінки вартості програмного забезпечення, можна зробити висновок, що для розрахунків формування робочої групи підійдуть:

– COCOMO – середній рівень;
– COCOMO II для попередньої оцінки;
– COCOMO II для детальної оцінки.

Їх переваги та недоліки вказані далі.

До переваг середнього рівня COCOMO відносяться:

– п’ять чинників характеристики персоналу, що впливають на оцінку вартості програмного забезпечення.

За їх наявності є можливість формування набору працівників, що можуть розробити проект за найменшу вартість.

До недоліків середнього рівня COCOMO відносяться:

– десять чинників характеристики проекту та програмного продукту;
– середній рівень COCOMO не має формули розрахунку часу виконання проекту.

Для оцінки вартості ПЗ ці значення чинників мають бути відомі. Також для розрахунку досвіду роботи на мовах програмування персоналу мають бути відомі мови програмування, за допомогою яких створюється проект.

Єдиною оцінкою якості формування робочої групи може виступати тільки трудоємність.

До переваг COCOMO II для попередньої оцінки відносяться:

– наявність формули розрахунку часу виконання проекту, що дозволить формувати робочі групи за двома оцінками якості.

До недоліків COCOMO II для попередньої оцінки відносяться:

– тільки три чинника характеристики персоналу – спрацьованість команди з факторів масштабу та кваліфікація і досвід персоналу з множників трудоємності;
– дев’ять чинників характеристики проекту та програмного продукту.

За такою кількістю чинників важче дати оцінки якості формування робочої групи.

Для оцінки вартості ПЗ ці чинники мають бути відомі. Також для оцінки спрацьованості команди необхідно мати додаткову інформацію про досвід роботи працівників один з одним, а для кваліфікації персоналу потрібно мати інформацію про мови програмування, за допомогою яких створюється проект, та платформу і технології.

До переваг COCOMO II для детальної оцінки відносяться:

– наявність формули розрахунку часу виконання проекту, що дозволить формувати робочі групи за двома оцінками якості;
– сім чинників характеристики персоналу, що впливають на оцінку вартості програмного забезпечення.

За їх наявності є можливість формування набору працівників, що можуть розробити проект за найменшу вартість та час виконання.

До недоліків COCOMO II для детальної оцінки відносяться:

– п’ятнадцять чинників характеристики проекту та програмного продукту.

Для оцінки вартості ПЗ ці значення чинників мають бути відомі. Також для оцінки спрацьованості команди необхідно мати додаткову інформацію про досвід роботи працівників один з одним, а для досвіду роботи з платформою та мовами програмування потрібно мати інформацію про мови програмування, за допомогою яких створюється проект, та платформу і технології.

Для розрахунків формування робочої групи ІТ-проекту будуть використовуватися моделі COCOMO – середній рівень та COCOMO II для детальної оцінки.

2.2 Висновки до розділу

У другому розділі було розглянуто математичне забезпечення для створення програмного забезпечення для формування робочої групи.

Були проаналізовані моделі оцінки вартості програмного забезпечення, такі як модель Путмана, три рівні моделі COCOMO та дві стадії оцінки моделі COCOMO II. Було розглянуто їх недоліки та переваги, залежність характеристик персоналу від оцінки вартості та часу виконання проекту.

Також було обрано моделі для розрахунків формування робочої групи IT-проекту.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧОГО ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

3.1.1 .NET

Для розробки програмного забезпечення було обрано платформу .Net. .Net – це безкоштовна платформа з відкритим кодом для створення багатьох різних типів додатків. .Net дозволяє використовувати кілька мов, редакторів і бібліотек для створення веб-, мобільних, десктопних програм та ігор [10].

Основні характеристики .Net:

– .NET є відкритим вихідним кодом і доступний на Github для безкоштовного завантаження та відкритий для внесків спільноти; хоча напрямок розробки .NET очолює спеціальна команда експертів Microsoft, більшість розробок вирішує велика спільнота з відкритим кодом;
– .NET є кросплатформним і підтримує всі Windows, Linux і Mac OS; додаток .NET можна розробити з використанням єдиної бази коду та запустити на кількох платформах;
– .NET – це сучасна платформа для розробки програмного забезпечення, яка підтримує більшість сучасних потреб розробки програмного забезпечення, включаючи нативні, мобільні, хмарні, веб-послуги та сервіси;
– .NET підтримує кілька мов, включаючи C#, F# і Visual Basic;
– .NET є хмарним і дозволяє розробникам створювати хмарні веб-додатки, служби та API;
– .NET пропонує одні з найкращих програмних IDE та інструментів, включаючи Visual Studio та Visual Studio Code;
– .NET швидкий і дружній і добре працює з іншими мовами та платформами.

.NET зріла, але все ще розвивається. Перша версія .NET Framework була випущена 14 лютого 2002 року. Поточна версія .NET — .NET 5.0, яка була випущена в листопаді 2020 року.

Команда .NET оголосила дати випуску майбутніх версій і очікує випускати одну велику версію щороку в листопаді. Як видно з наведеного вище розкладу випуску, .NET 6.0, 7.0. і 8.0 будуть випущені в листопаді 2021, 2022 і 2023 років відповідно [11].

У нашому випадку, для створення веб-застосунку ми використовуємо ASP.NET. ASP.NET — це набір бібліотек та інструментів для створення веб-додатків, включаючи інтерфейсні веб-сайти, API та мікросервіси. Як і ASP, ASP.NET є технологією на стороні сервера. Схему монолітної архітектури для програмного забезпечення формування робочої групи з використанням платформи .Net наведено в Додатку В. Лістинг коду бібліотеки на платформі .Net на основі методу оцінки вартості ПЗ наведено у Додатку Д.

3.1.2 C#

Мовою програмування, для створення серверної частини застосунку, було обрано C#. C# є однією з найпопулярніших мов програмування у світі, відноситься до об’єктно-орієнтованих мов програмування, а такожмає свої коріння в сімействі мов C і буде відразу знайомий програмістам C, C++, Java та JavaScript. C# також є найбільш універсальною мовою програмування серед усіх.

Основні характеристики C#:

- простий, чистий та сучасний;
- безпечний, продуктивний і швидкий;
- є відкритим вихідним кодом;
- є мовою для всіх видів програмних додатків;
- є мовою для нових технологій;

– розвивається [11].

3.1.3 Microsoft SQL Server

Для збереження даних було обрано Microsoft SQL Server. Microsoft SQL Server — це система управління реляційною базою даних (RDBMS), яка підтримує широкий спектр додатків для обробки транзакцій, бізнес-аналітики та аналітики в корпоративних IT-середах. Microsoft SQL Server є однією з трьох провідних технологій баз даних, поряд з Oracle Database і IBM DB2 [12].

Microsoft SQL Server (MSSQL) широко використовується в корпоративних розгортаннях. MSSQL — це масштабована платформа даних, яка включає в себе кілька інструментів ETL (Extract, Transform and Load) і служби звітності, де дані можна додавати, змінювати та запитувати за допомогою стандартизованої мови структурованих запитів (SQL). MSSQL – це платформа даних, що розвивається та використовується для критично важливих бізнес-рішень і рішень для обробки даних локально, в хмарі та на гібридних платформах.

Сервер MSSQL є неймовірно популярним рішенням для баз даних, який використовується сьогодні, і однією з його найсильніших переваг є простота використання. MSSQL постачається з багатьма чудовими інструментами, які роблять розробку бази даних швидким і гнучким процесом. Студія керування SQL Server дозволяє будь-якому затвердженому користувачеві керувати та підтримувати бази даних, виконувати SQL-запити, створювати резервні копії та аналізувати графіки продуктивності. MSSQL інтегрується з Visual Studio, щоб надати вашій команді DevOps потужну, звичну платформу для створення та керування користувацькими програмами, які легко інтегруються з MSSQL Server [13].

3.1.4 Angular

Для клієнтської частини був використаний сучасний фреймворк Angular. Angular — це платформа JavaScript з відкритим вихідним кодом, написана на TypeScript. Google підтримує його, і його основна мета — розробляти односторінкові програми. Як фреймворк, Angular має очевидні переваги, а також надає стандартну структуру для роботи розробників. Це дозволяє користувачам створювати великі додатки в обслуговуваний спосіб.

TypeScript визначає набір типів для JavaScript, що допомагає користувачам писати код JavaScript, який легше зрозуміти. Весь код TypeScript компілюється за допомогою JavaScript і може безперебійно працювати на будь-якій платформі. TypeScript не є обов'язковим для розробки програми Angular. Однак його використання настійно рекомендується, оскільки він пропонує кращу синтаксичну структуру, полегшуючи розуміння та обслуговування кодової бази [14].

3.1.5 PrimeNG

Для покращення інтерфейсу клієнтської частини, була використана бібліотека PrimeNG. PrimeNG — це набір компонентів інтерфейсу користувача для Angular. Усі віджети мають відкритий вихідний код і безкоштовні для використання за ліцензією MIT. PrimeNG розроблено PrimeTek Informatics, постачальником з багаторічним досвідом розробки рішень інтерфейсу користувача з відкритим кодом [15].

3.2 Вимоги до програмного продукту

Розглянемо функціональні вимоги до програмного продукту на основі діаграми використання, що наведена у Додатку Б.

Функціональні вимоги до серверної частини:

- забезпечувати реєстрацію та авторизацію нових користувачів;
- забезпечувати додавання/редагування/видалення/збереження нового проекту клієнтом;
- забезпечувати додавання/редагування/видалення/збереження нового працівника клієнтом;
- забезпечувати додавання/редагування/видалення/збереження нової мови програмування клієнтом;
- забезпечувати додавання/редагування/видалення/збереження нової платформи клієнтом;
- роботи підрахунки наборів працівників за допомогою моделі оцінки вартості ПЗ в залежності від вхідних параметрів;
- надавати клієнту результати підрахунків формування робочої групи.

Функціональні вимоги до клієнтської частини:

- надавати змогу реєстрації та авторизації клієнту;
- надавати змогу переглянути/додати/змінити/видалити інформацію про новий проект;
- надавати змогу переглянути/додати/змінити/видалити інформацію про нового працівника;
- надавати змогу переглянути/додати/змінити/видалити інформацію про нову мову програмування;
- надавати змогу переглянути/додати/змінити/видалити інформацію про нову платформу;
- надавати змогу перегляду результатів підрахунків формування робочої групи.

3.3 Опис структури бази даних

Структурна схема бази даних застосунку наведена у Додатку А. Програма використовує сутність User для збереження даних користувача. Користувач має змогу додавати/видаляти/змінювати інформацію про мови програмування, платформи програмування, працівників, проекти та команди, тому сутність має відношення один до багатьох до наступних сутностей: Employee, Team, Project, ProgrammingLanguage, ProgrammingPlatform.

Працівник може знати декілька мов програмування та декілька платформ програмування, тому сутність Employee має відношення багато до багатьох до сутностей ProgrammingLanguage та ProgrammingPlatform за допомогою сутностей LanguageExperiences та PlatformExperiences, у яких зберігається досвід роботи працівника.

Сутність Cohesion дозволяє зберігати досвід роботи працівників одне з одним.

Сутність Team формується як результат роботи програмного забезпечення з набору працівників для конкретного проекту.

Детальний опис сутностей бази даних наведений у таблицях 3.1-3.11.

Таблиця 3.1 – Перелік полів сутності User

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
UserName	nvarchar(max)	Логін користувача
Email	nvarchar(max)	Електронна пошта користувача
Password	nvarchar(max)	Пароль користувача
Image	nvarchar(max)	Зображення користувача

Таблиця 3.2 – Перелік полів сутності Employee

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
Name	nvarchar(max)	Ім’я працівника
Lastname	nvarchar(max)	Прізвище працівника
Image	nvarchar(max)	Зображення працівника
Salary	int	Заробітна плата працівника
AnalystCapability	nvarchar(max)	Аналітичні здібності працівника
ProgrammerCapability	nvarchar(max)	Здібності до програмування працівника
VirtualMachineExperience	int	Досвід використання віртуальних машин
UserId	int	Унікальний ідентифікатор користувача

Таблиця 3.3 – Перелік полів сутності Project

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
Image	nvarchar(max)	Зображення проекту
SoftwareReliability	nvarchar(max)	Потрібна надійність програмного забезпечення
DataBaseSize	nvarchar(max)	Розмір бази даних продукту
Complexity	nvarchar(max)	Складність продукту
PerformanceLimitation	nvarchar(max)	Обмеження швидкодії при виконанні програми
MemoryLimitation	nvarchar(max)	Обмеження пам’яті
VirtualMachineInstability	nvarchar(max)	Нестійкість оточення віртуальної машини
RecoveryTime	nvarchar(max)	Необхідний час відновлення
DevelopmentTools	nvarchar(max)	Використання інструментів розробки програмного забезпечення
DevelopmentSchedule	nvarchar(max)	Необхідність виконання графіка робіт
RequiredReusability	nvarchar(max)	Необхідна надійність програми
DocumentationMatch	nvarchar(max)	Відповідність документації потребам життєвого циклу
ExecutionTimeLimitation	nvarchar(max)	Обмеження часу виконання
PlatformVolatility	nvarchar(max)	Змінність платформи
MultisiteDevelopment	nvarchar(max)	Багатоабонентська розробка
DevelopmentFlexibility	nvarchar(max)	Гнучкість проекту розробки
RiskResolution	nvarchar(max)	Архітектура і дозвіл ризиків
ProcessMaturity	nvarchar(max)	Зрілість процесів
UserId	int	Унікальний ідентифікатор користувача

Таблиця 3.4 – Перелік полів сутності Team

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
Name	nvarchar(max)	Назва команди
UserId	int	Унікальний ідентифікатор користувача

Таблиця 3.5 – Перелік полів сутності ProgrammingLanguage

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
Name	nvarchar(max)	Назва мови програмування
UserId	int	Унікальний ідентифікатор користувача

Таблиця 3.6 – Перелік полів сутності ProgrammingPlatform

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
Name	nvarchar(max)	Назва платформи програмування
UserId	int	Унікальний ідентифікатор користувача

Таблиця 3.7 – Перелік полів сутності Cohesion

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
FirstEmployeeId	int	Унікальний ідентифікатор користувача №1
SecondEmployeeId	int	Унікальний ідентифікатор користувача №2
WorkExperience	int	Досвід роботи

Таблиця 3.8 – Перелік полів сутності EmployeeTeam

Поле	Тип даних	Призначення
EmployeesId	int	Унікальний ідентифікатор працівника
ProjectsId	int	Унікальний ідентифікатор проекту

Таблиця 3.9 – Перелік полів сутності LanguageExperience

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
EmployeesId	int	Унікальний ідентифікатор працівника
ProgrammingLanguageId	int	Унікальний ідентифікатор мови програмування
Experience	int	Досвід роботи

Таблиця 3.10 – Перелік полів сутності PlatformExperience

Поле	Тип даних	Призначення
Id	int	Унікальний ідентифікатор
EmployeesId	int	Унікальний ідентифікатор працівника
ProgrammingPlatformId	int	Унікальний ідентифікатор платформи програмування
Experience	int	Досвід роботи

Таблиця 3.11 – Перелік полів сутності ProjectTeam

Поле	Тип даних	Призначення
ProjectsId	int	Унікальний ідентифікатор проекту
TeamsId	Int	Унікальний ідентифікатор команди

3.4 Інструкція користувача

Надалі буде наведено інструкцію використання веб-застосунку:

Реєстрація. Використання застосунку починається з реєстрації користувача у системі.

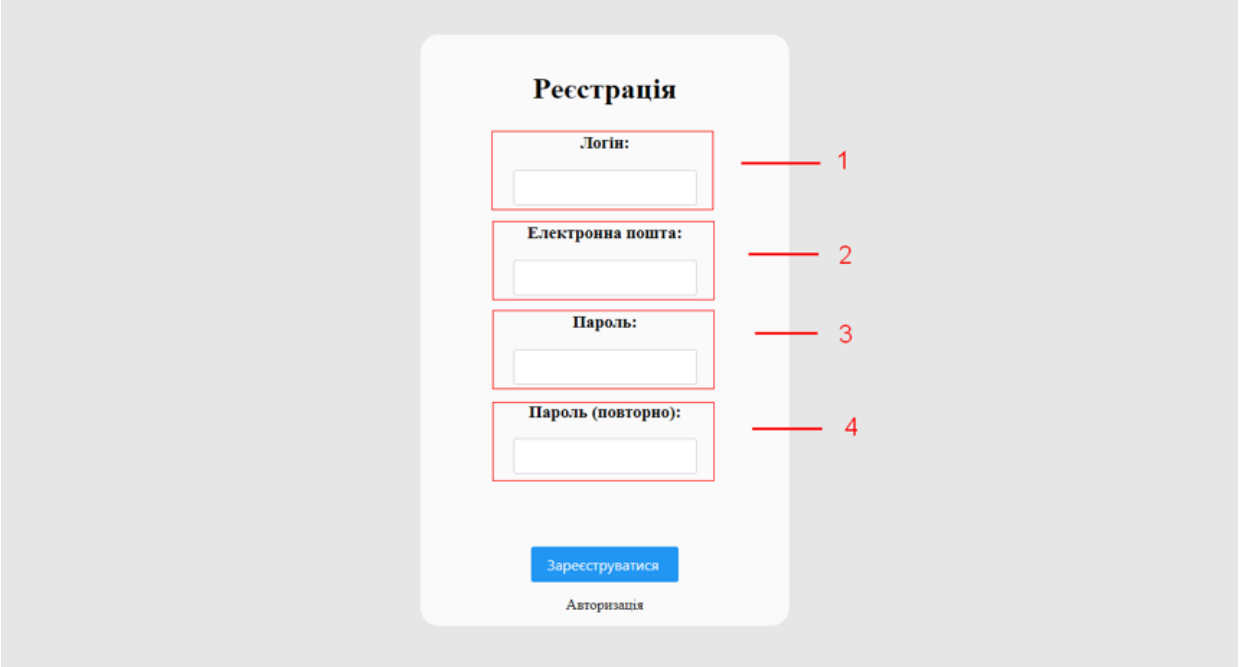


Рисунок 3.1 – Форма реєстрації користувача у системі

На сторінці можна побачити форму реєстрації користувачів:

– поле для вводу ім’я користувача – це ім’я буде відображатися у веб-застосунку;
– поле для вводу електронної пошти користувача – потрібно для контакту з користувачем (наприклад, для відновлення пароля);
– поле для вводу паролю;
– поле для повторного вводу паролю.

Головна сторінка. Після реєстрації користувач опиняється на головні сторінці.

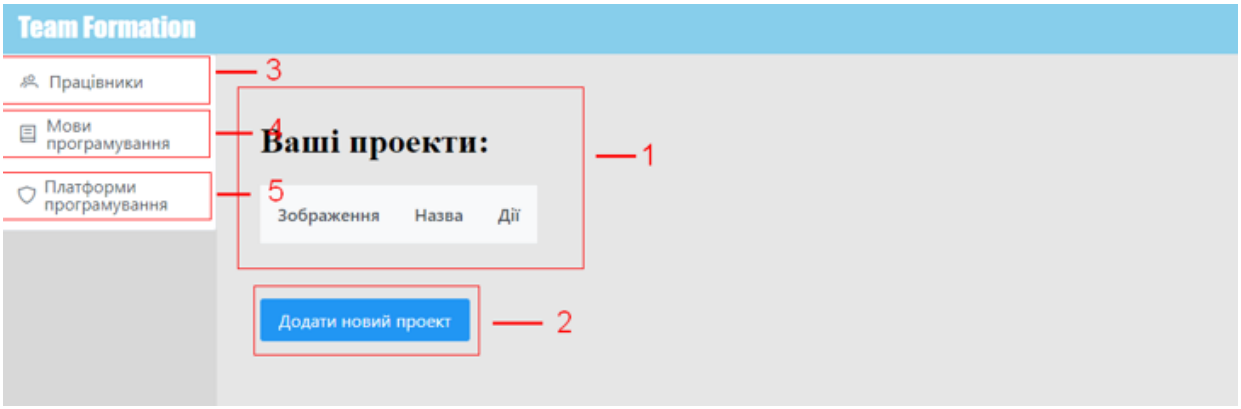


Рисунок 3.2 – Головна сторінка веб-застосунку

На сторінці можна побачити:

– таблицю проектів користувача, де міститься інформація про його проекти;
– кнопку «Додати проект», що перенаправляє користувача на сторінку створення нового проекту;
– посилання «Працівники» на боковій панелі, що перенаправляє користувача на сторінку з інформацією про працівників;
– посилання «Мови програмування» на боковій панелі, що перенаправляє користувача на сторінку з інформацією про мови програмування;
– посилання «Платформи програмування» на боковій панелі, що перенаправляє користувача на сторінку з інформацією про платформи програмування.

Додання нового проекту. При натисканні «Додати проект», користувача перенаправить на сторінку створення нового проекту.

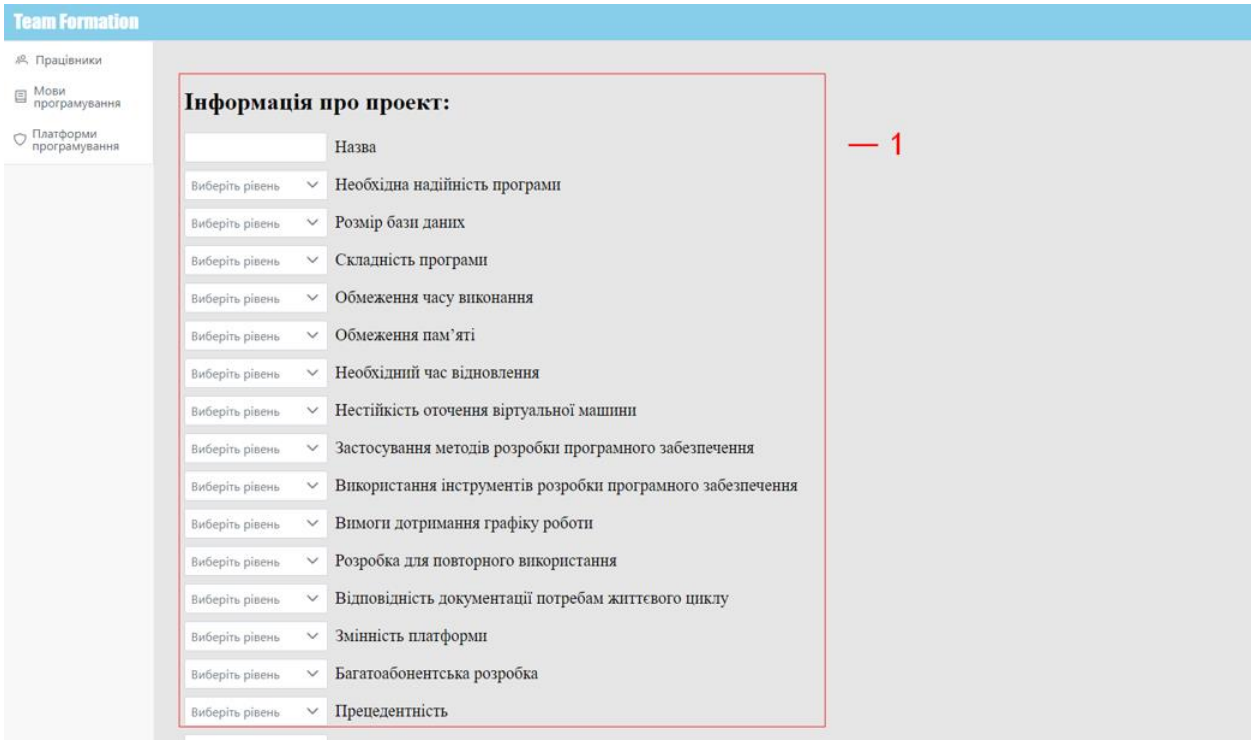


Рисунок 3.3 – Сторінка створення нового проекту

На сторінці можна побачити:

– форму для заповнення інформації про проект;
– кнопку «Зберегти зміни» для збереження інформації про проект до бази даних (під формою).

Сторінка «Працівники». При натисканні на посилання «Працівники», користувача перенаправить на сторінку з інформацією про його працівників.

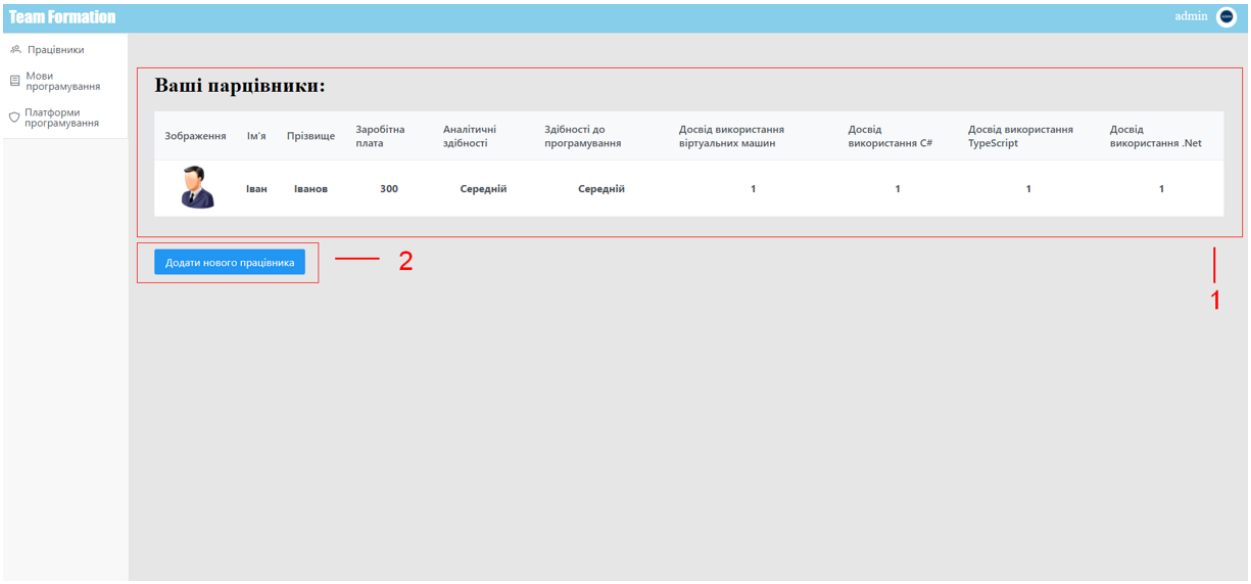


Рисунок 3.4 – Сторінка «Працівники»

На сторінці можна побачити:

– таблицю працівників користувача, де міститься інформація про його працівників;
– кнопку «Додати працівника», що перенаправляє користувача на сторінку створення нового працівника.

Додання нового працівника. При натисканні «Додати працівника», користувача перенаправить на сторінку створення нового працівника.

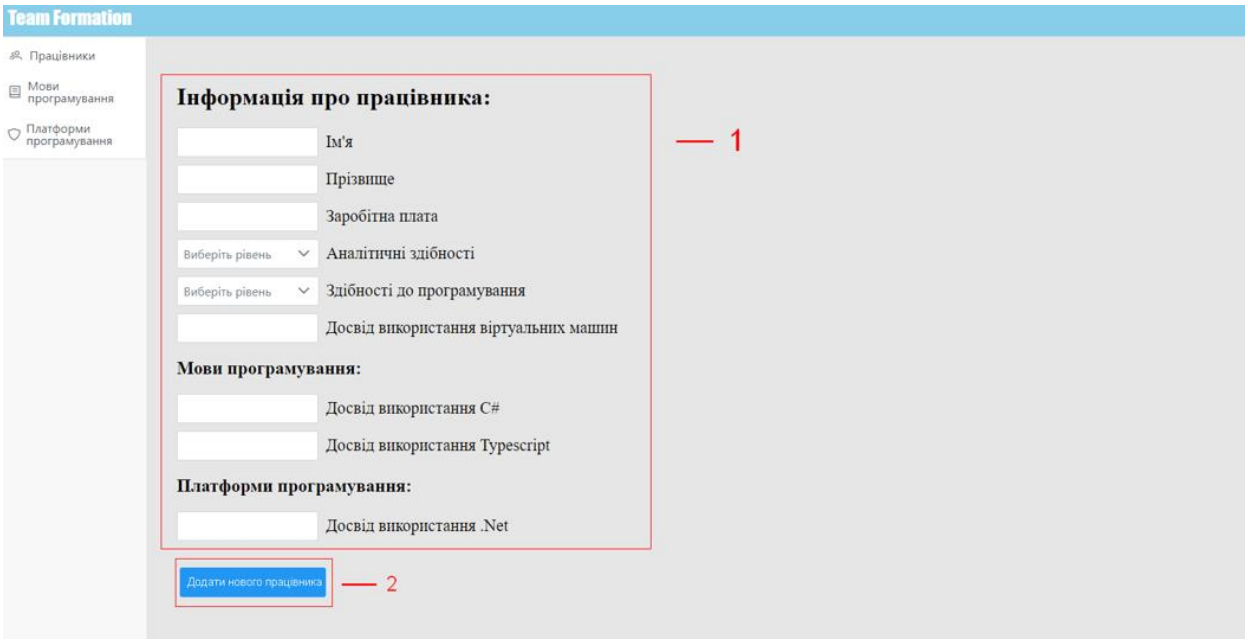


Рисунок 3.5 – Сторінка створення нового працівника

На сторінці можна побачити:

– форму для заповнення інформації про працівника;
– кнопку «Зберегти зміни» для збереження інформації про працівника до бази даних.

Сторінка «Мови програмування». При натисканні на посилання «Мови програмування», користувача перенаправить на сторінку з інформацією про мови програмування.

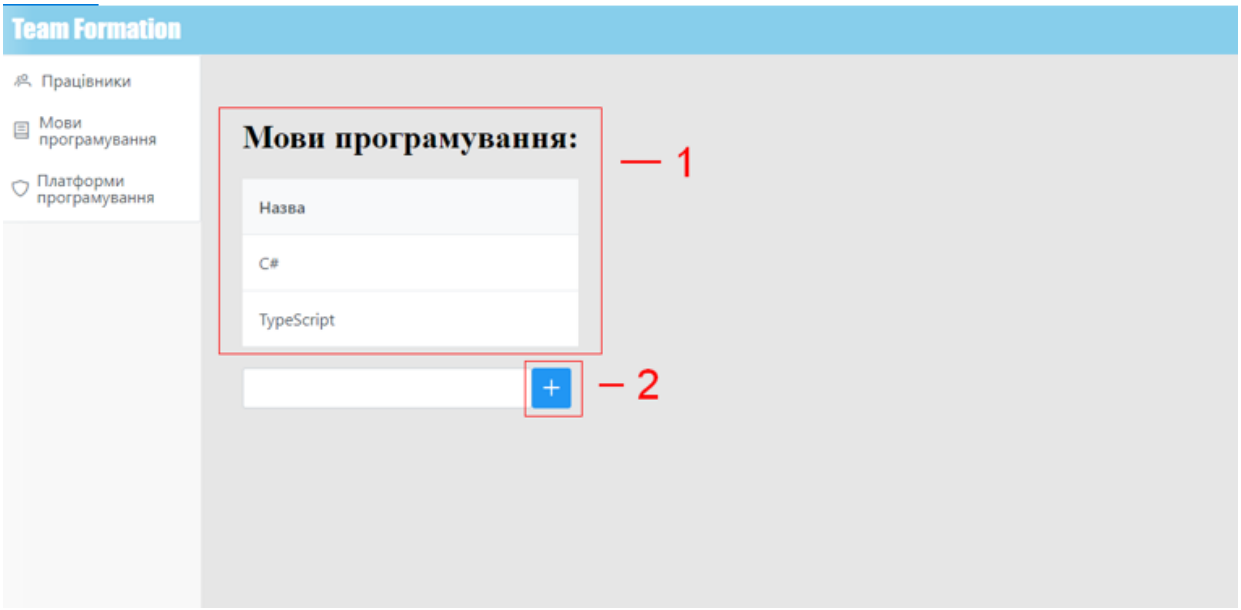


Рисунок 3.6 – Сторінка «Мови програмування»

На сторінці можна побачити:

– таблицю мов програмування користувача, що редагується;
– кнопку «Додати мову програмування» для додання нової мови програмування.

Сторінка «Платформи програмування». При натисканні на посилання «Платформи програмування», користувача перенаправить на сторінку з інформацією про платформи програмування.

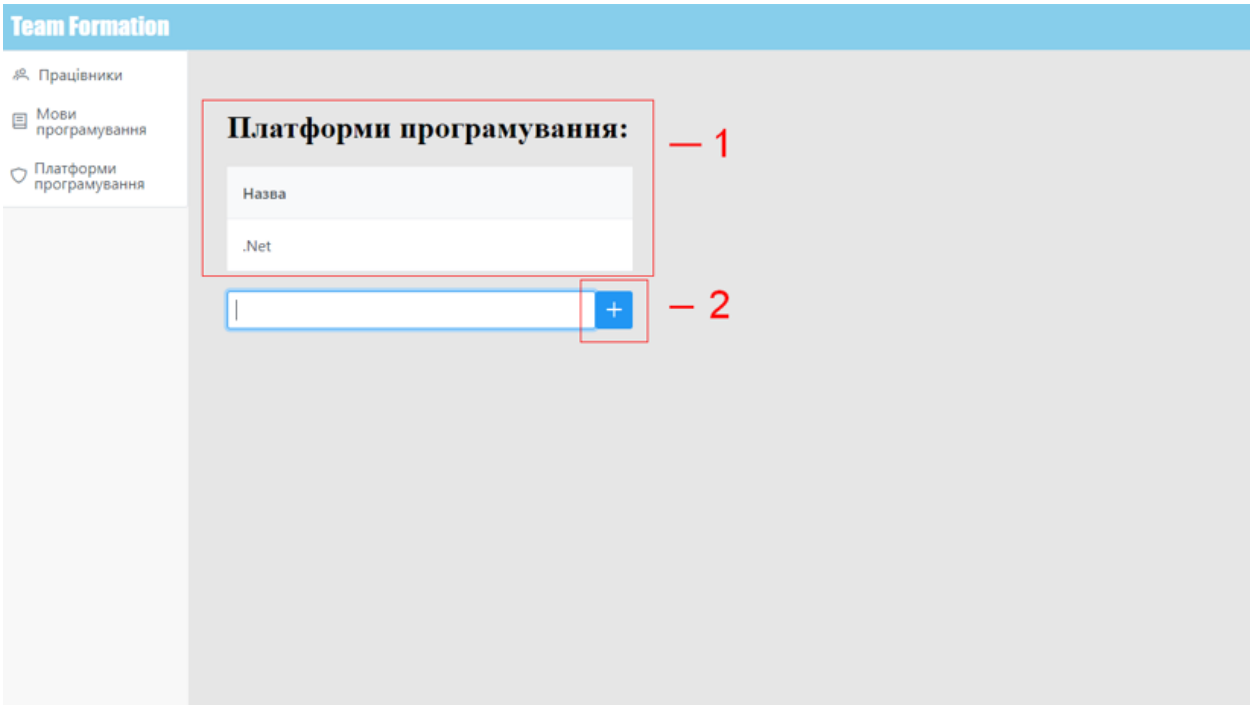


Рисунок 3.7 – Сторінка «Платформи програмування»

На сторінці можна побачити:

– таблицю платформ програмування користувача, що редагується;
– кнопку «Додати платформу програмування» для додання нової мови програмування.

Головна сторінка після додання проекту. Після додання хоча б одного проекту у таблиці проектів користувача з’явиться про нього інформація.

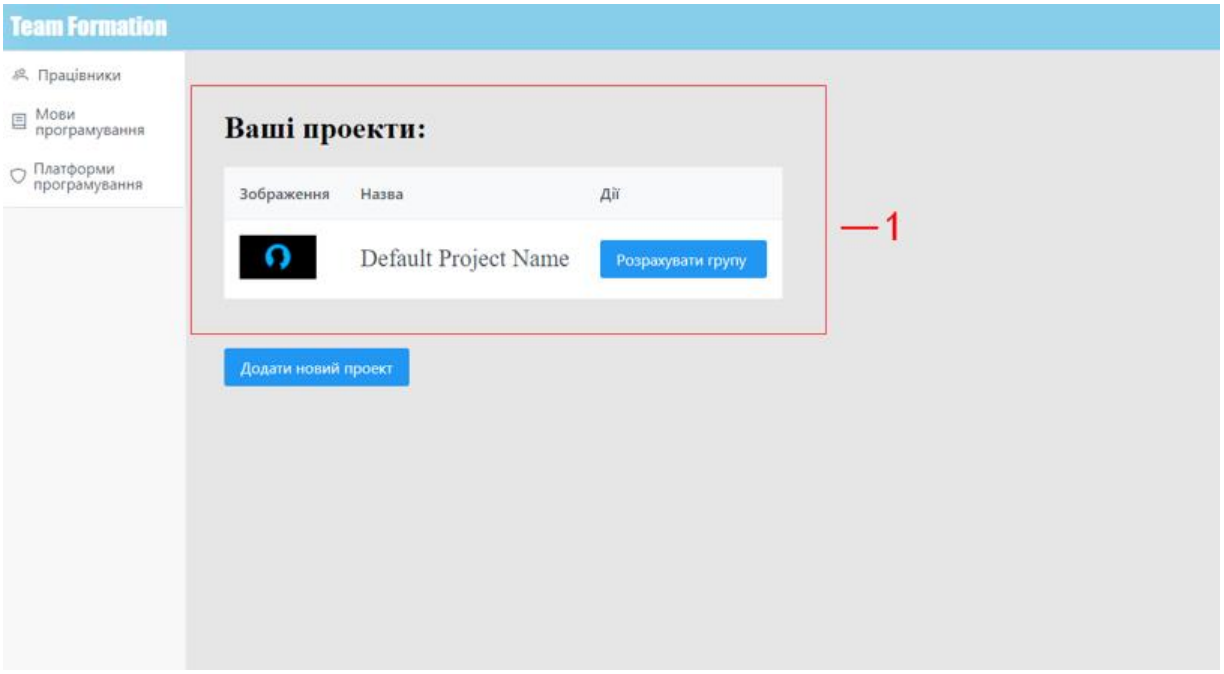


Рисунок 3.8 – Головна сторінка після додання проекту

На сторінці можна побачити:

– таблицю проектів користувача, де з’явилася інформація про новий проект та кнопка «Сформувати групу», яка перенаправляє користувача на результати розрахунків формування груп для обраного проекту.

Результати розрахунків формування груп. При натисканні на кнопку «Сформувати групу», користувача перенаправить на сторінку з результатами розрахунків формування груп для обраного проекту.

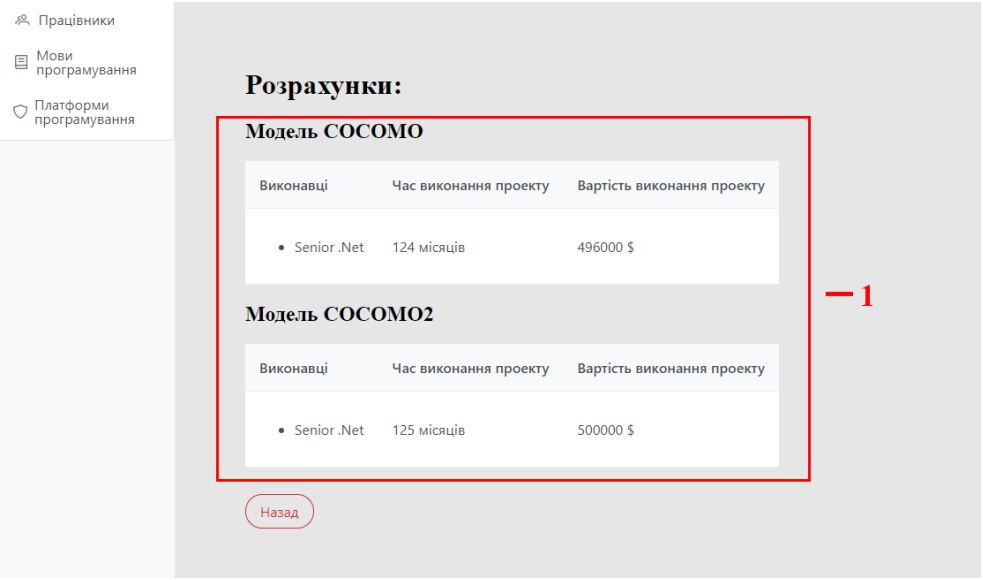


Рисунок 3.9 – Сторінка розрахунків формування груп

На сторінці можна побачити:

– таблицю з наборами працівників та розрахунками вартості роботи та часу виконання проекту цими працівниками;
– фільтри часу виконання проекту та вартості роботи.

3.5 Випробування програмного продукту

Метою випробування є оцінка якості та надійності програмного продукту та його відповідності функціональним вимогам.

Для тестування було створено сценарії роботи програмного забезпечення на основі діаграми використання, що наведена у Додатку Б. Результати виконання тестових сценаріїв наведені у таблицях 3.12-3.17.

Перелік тестових сценаріїв:

– «Спроба реєстрації з неправильним паролем у полі повторного вводу паролю»;
– «Спроба реєстрації»;
– «Спроба додання проекту з незаповненими полями форми»;
– «Додання проекту з заповненими полями форми»;
– «Формування групи обраного проекту без працівників»;
– «Формування групи обраного проекту»;
– «Спроба видалення працівника».

Таблиця 3.12 – Тестовий сценарій «Спроба реєстрації з неправильним паролем у полі повторного вводу паролю»

Мета тесту	Перевірка роботи
Початковий стан ПЗ	Відкрита сторінка реєстрації.
Вхідні дані	Ім’я користувача, електронна пошта, пароль, неправильно введений повторно пароль.
Схема проведення тесту	Натискання на кнопку «Зареєструватись».
Очікуваний результат	Поява повідомлення про неправильно введений повторно пароль.
Стан ПЗ після проведення випробувань	Відкрита сторінка реєстрації з повідомленням про неправильно введений повторно пароль.

Таблиця 3.13 – Тестовий сценарій «Спроба реєстрації»

Мета тесту	Перевірка роботи
Початковий стан ПЗ	Відкрита сторінка реєстрації.
Вхідні дані	Ім’я користувача, електронна пошта, пароль, правильно введений повторно пароль.
Схема проведення тесту	Натискання на кнопку «Зареєструватись».
Очікуваний результат	Переадресація на головну сторінку веб-застосунку.
Стан ПЗ після проведення випробувань	Відкрита головна сторінка.

Таблиця 3.14 – Тестовий сценарій «Спроба додання проекту з незаповненими полями форми»

Мета тесту	Перевірка роботи
Початковий стан ПЗ	Відкрита сторінка додання нового проекту.
Вхідні дані	Незаповнені форми.
Схема проведення тесту	Натискання на кнопку «Додати проект».
Очікуваний результат	Поява повідомлення «Одне чи декілька полів не заповнені».
Стан ПЗ після проведення випробувань	Відкрита сторінка додання нового проекту з повідомленням «Одне чи декілька полів не заповнені».

Таблиця 3.15 – Тестовий сценарій «Спроба додання проекту з заповненими полями форми»

Мета тесту	Перевірка роботи
Початковий стан ПЗ	Відкрита сторінка додання нового проекту.
Вхідні дані	Заповнені форми з інформацією про проект.
Схема проведення тесту	Натискання на кнопку «Додати проект».
Очікуваний результат	Переадресація на головну сторінку.
Стан ПЗ після проведення випробувань	Відкрита головна сторінка з таблицею проектів, де є інформація про новий проект.

Таблиця 3.16 – Тестовий сценарій «Формування групи обраного проекту без працівників»

Мета тесту	Перевірка роботи
Початковий стан ПЗ	Відкрита головна сторінка з таблицею проектів.
Вхідні дані	Відсутність працівників на аккаунті користувача.
Схема проведення тесту	Натискання на кнопку «Сформувати групу» будь-якого проекту.
Очікуваний результат	Поява повідомлення «Немає працівників для формування групи».
Стан ПЗ після проведення випробувань	Відкрита головна сторінка з таблицею проектів.

Таблиця 3.17 – Тестовий сценарій «Формування групи обраного»

Мета тесту	Перевірка роботи
Початковий стан ПЗ	Відкрита головна сторінка з таблицею проектів.
Вхідні дані	На аккаунті користувача є декілька працівників.
Схема проведення тесту	Натискання на кнопку «Сформувати групу» будь-якого проекту.
Очікуваний результат	Переадресація на сторінку розрахунків формування груп.
Стан ПЗ після проведення випробувань	Відкрита головна сторінка розрахунків формування груп.

3.6 Експериментальне дослідження запропонованих моделей

Метою дослідження є визначення залежності результату використання запропонованих моделей від вхідних даних характеристик команди проекту на основі побудованого програмного забезпечення. Для цього визначимо вхідні дані трьох проектів різної складності та вхідні дані трьох команд проекту з різним досвідом розробки. Тип проекту: розповсюджений. Розмір проекту: 10 тисяч строк коду. Для експериментального дослідження кожна команда побудована з однієї людини. Вхідні дані команд наведені на Рисунках 3.10-3.12.

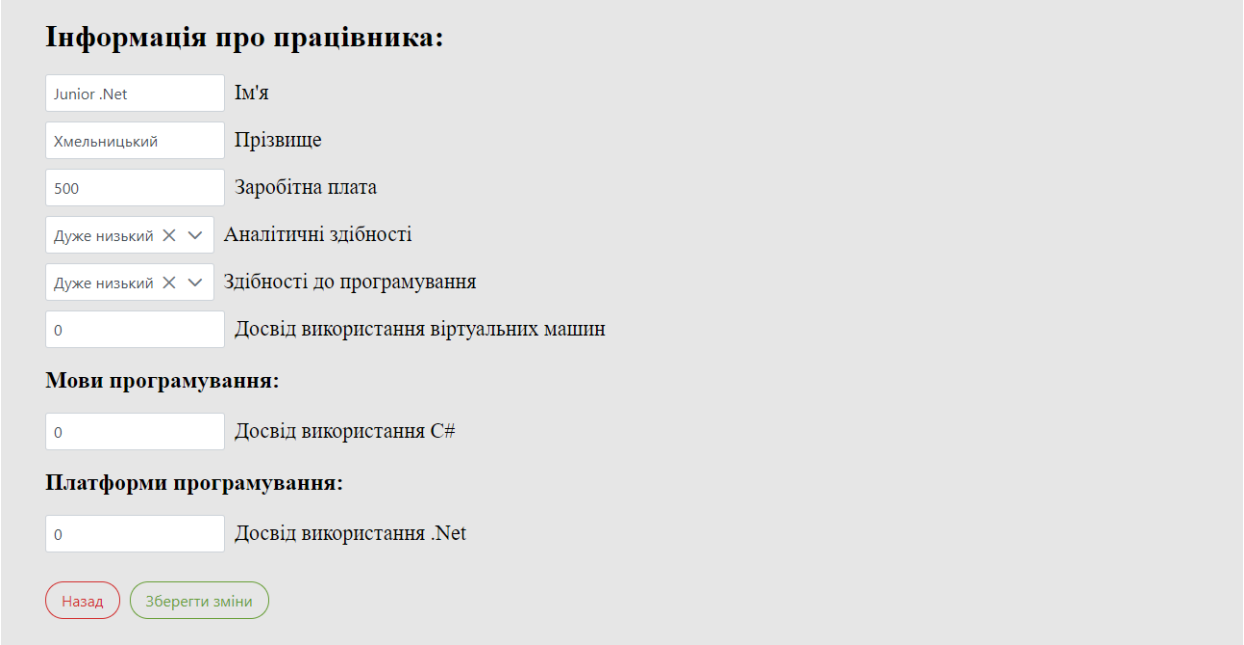


Рисунок 3.10 – Вхідні дані працівника для команди з низьким досвідом роботи

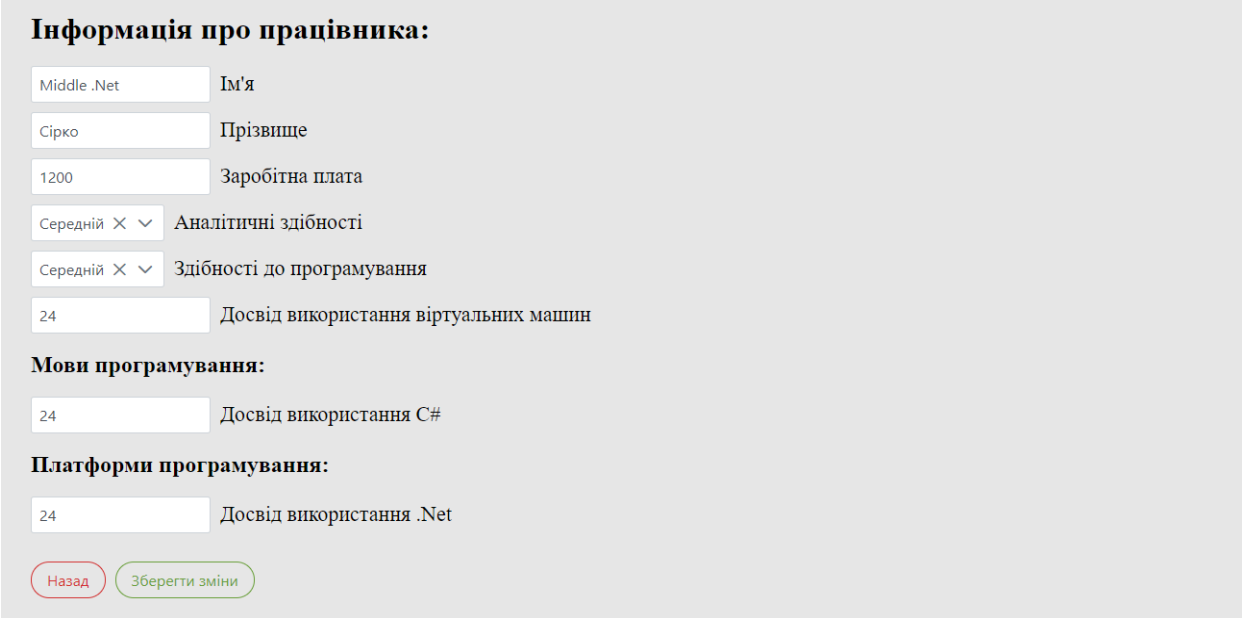


Рисунок 3.11 – Вхідні дані працівника для команди з середнім досвідом роботи

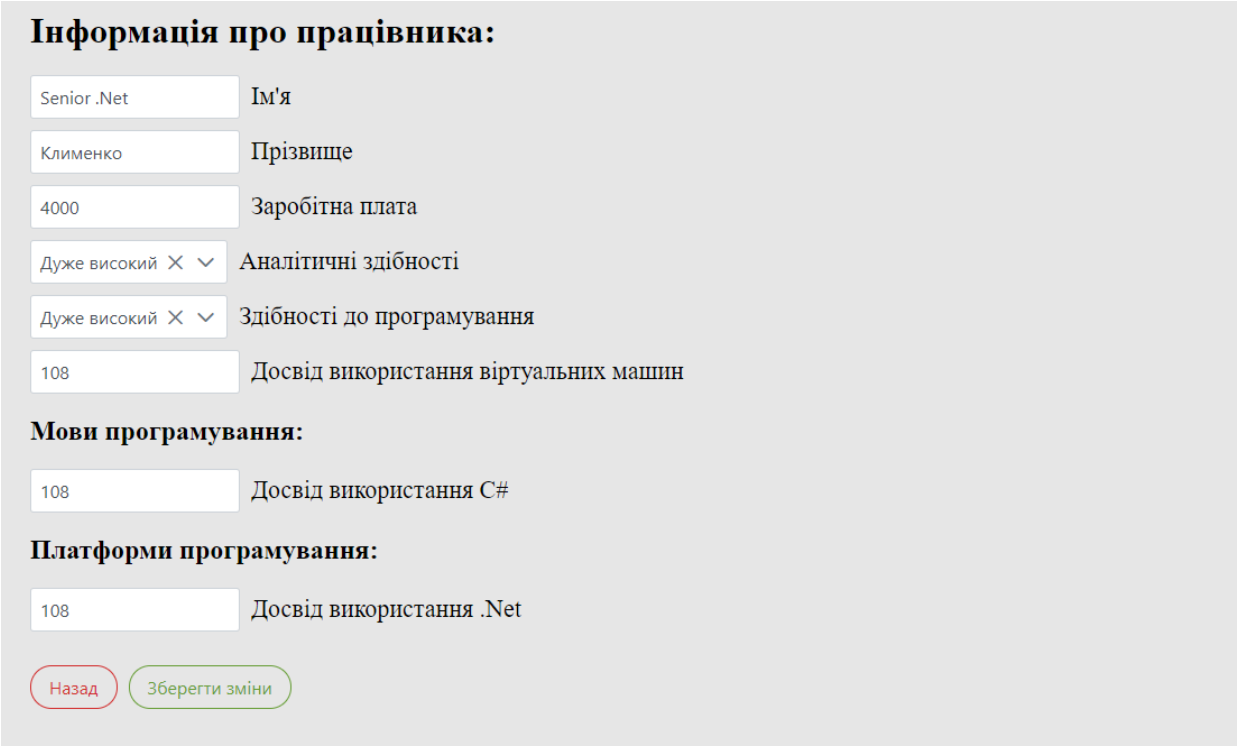


Рисунок 3.12 – Вхідні дані працівника для команди з середнім досвідом роботи

Вхідні дані для проектів наведені у Таблицях 3.18-3.19.

Таблиця 3.18 – Вхідні дані проектів для моделі COCOMO

Атрибути вартості	Рейтинг		
	Проект низької складності	Проект середньої складності	Проект високої складності
RELY	Дуже низький	Середній	Дуже високий
DATA	Низький	Середній	Дуже високий
CPLX	Дуже низький	Середній	Дуже високий
TIME	Середній	Середній	Дуже високий
STOR	Середній	Середній	Дуже високий
MODP	Дуже високий	Середній	Дуже низький
TOOL	Дуже високий	Середній	Дуже низький
SCED	Дуже низький	Середній	Високий
TURN	Низький	Середній	Дуже високий
VIRT	Низький	Середній	Дуже високий

Таблиця 3.19 – Вхідні дані проектів для моделі COCOMO II

Атрибути вартості	Рейтинг		
	Проект низької складності	Проект середньої складності	Проект високої складності
PREC	Дуже високий	Середній	Дуже низький
FLEX	Дуже високий	Середній	Дуже низький
RESL	Дуже високий	Середній	Дуже низький
PMAT	Дуже високий	Середній	Дуже низький
RUSE	Низький	Середній	Дуже високий
SCED	Дуже низький	Середній	Високий
RELY	Дуже низький	Середній	Дуже високий
DATA	Низький	Середній	Дуже високий
CPLX	Дуже низький	Середній	Дуже високий
DOCU	Дуже низький	Середній	Дуже високий
TIME	Середній	Середній	Дуже високий
STOR	Середній	Середній	Дуже високий
PVOL	Низький	Середній	Дуже високий
TOOL	Дуже високий	Середній	Дуже низький
SITE	Дуже високий	Середній	Дуже низький

Результати виконання програми наведені на Рисунках 3.13-3.21.

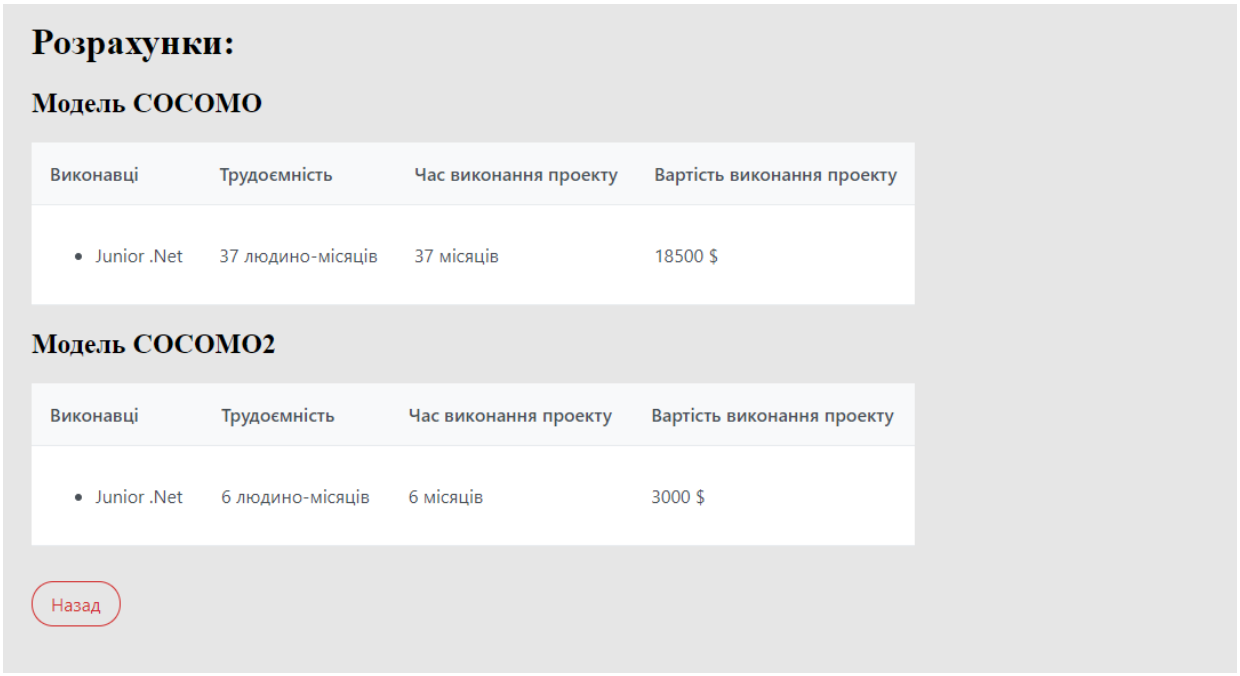


Рисунок 3.13 – Розрахунки трудоємності для команди з низьким досвідом роботи та проекту низької складності

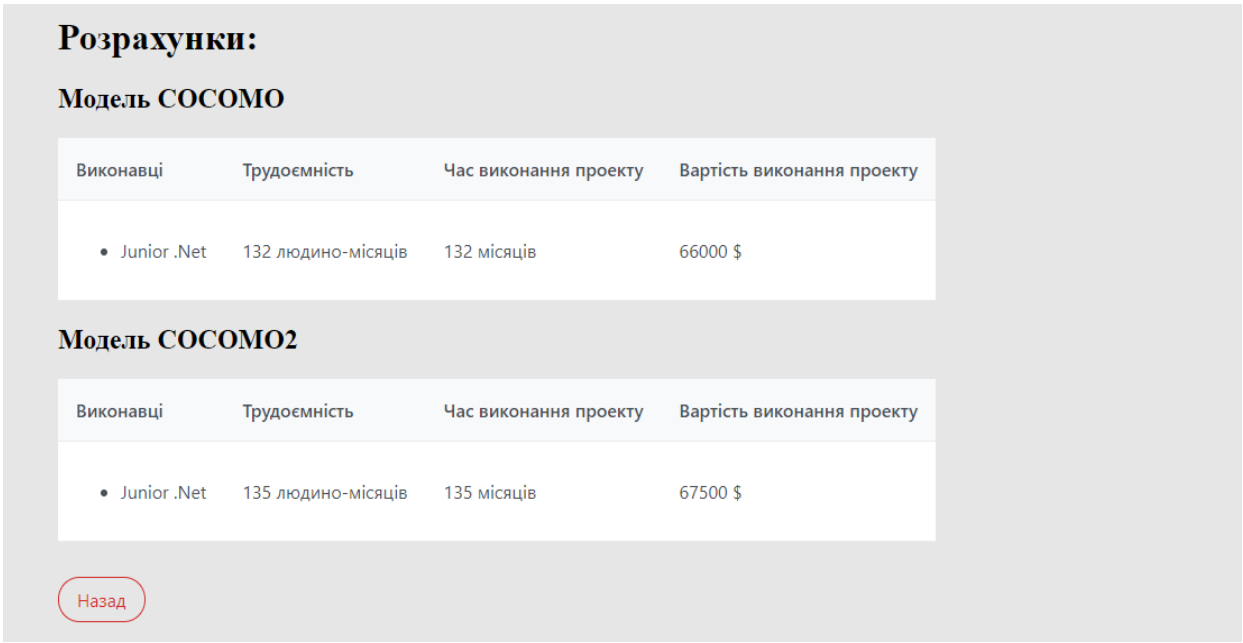


Рисунок 3.14 – Розрахунки трудоємності для команди з низьким досвідом роботи та проекту середньої складності

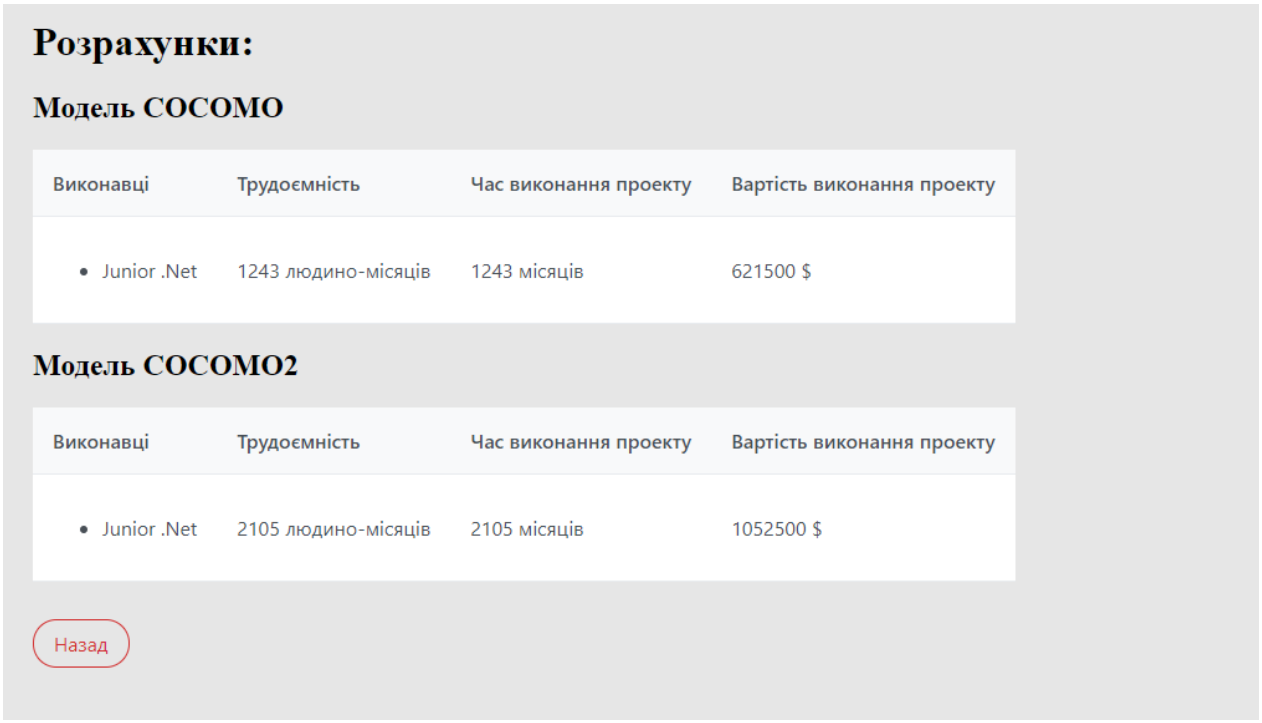


Рисунок 3.15 – Розрахунки трудоємності для команди з низьким досвідом роботи та проекту високої складності

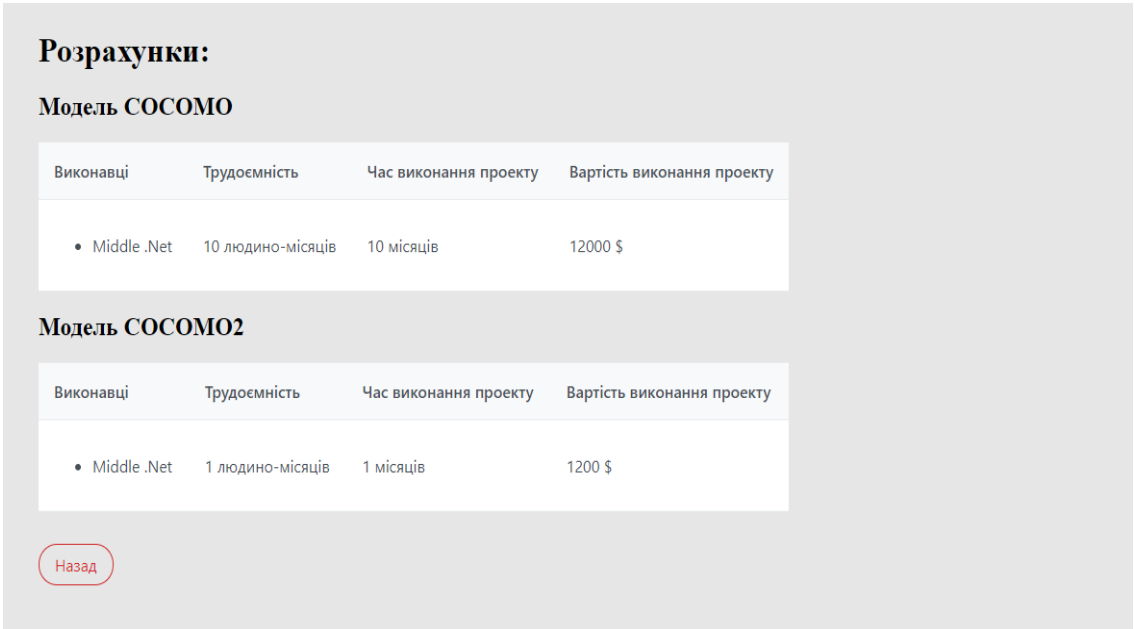


Рисунок 3.16 – Розрахунки трудоємності для команди з середнім досвідом роботи та проекту низької складності

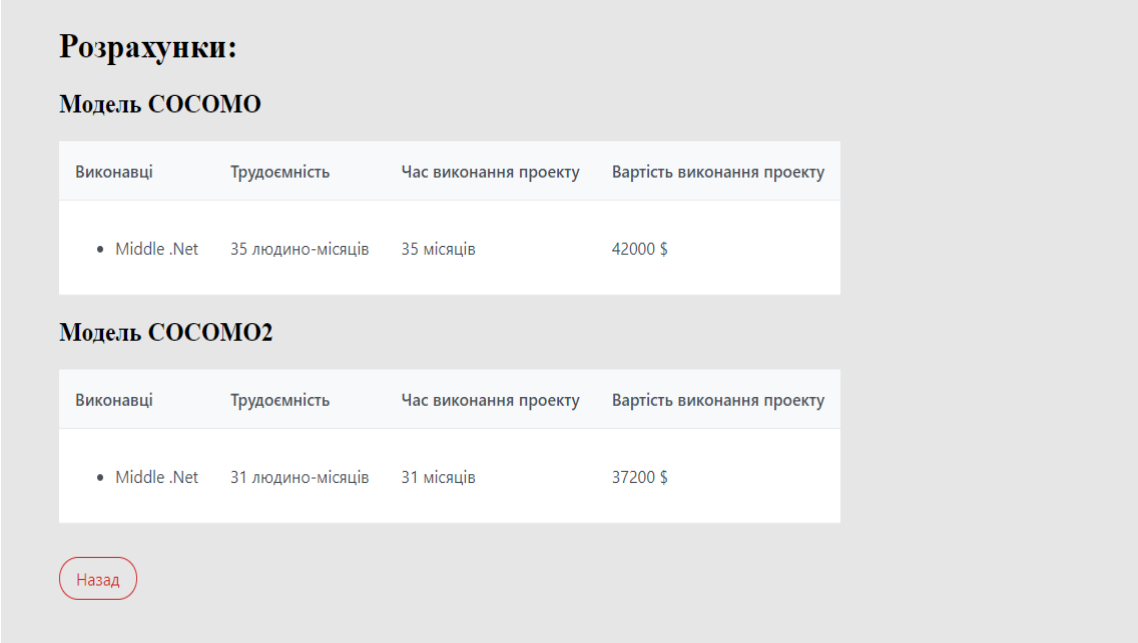


Рисунок 3.17 – Розрахунки трудоємності для команди з середнім досвідом роботи та проекту середньої складності

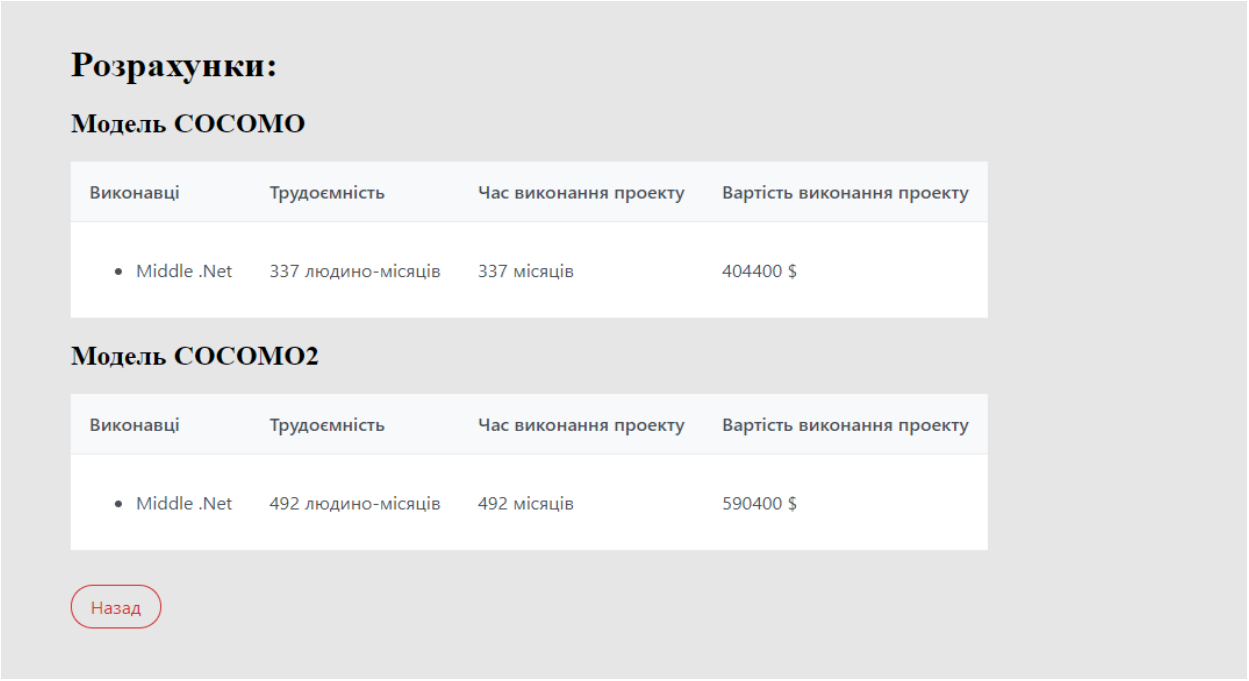


Рисунок 3.18 – Розрахунки трудоємності для команди з середнім досвідом роботи та проекту високої складності

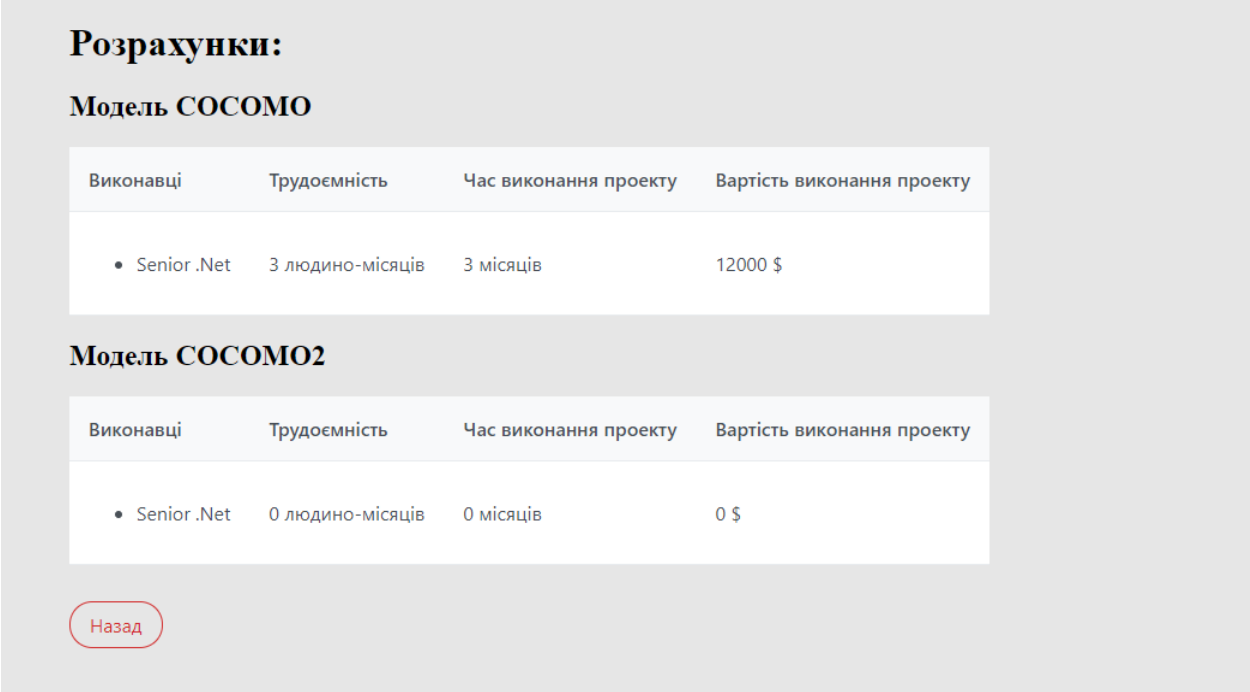


Рисунок 3.19 – Розрахунки трудоємності для команди з високим досвідом роботи та проекту низької складності

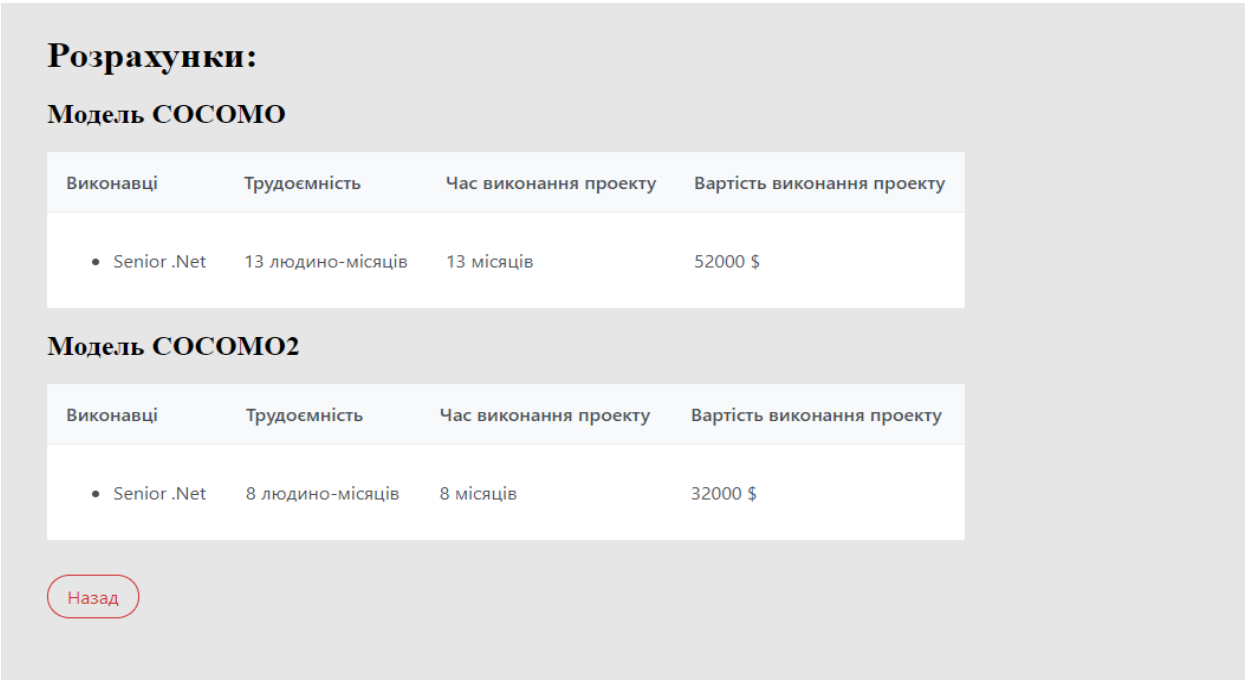


Рисунок 3.20 – Розрахунки трудоємності для команди з високим досвідом роботи та проекту низької складності

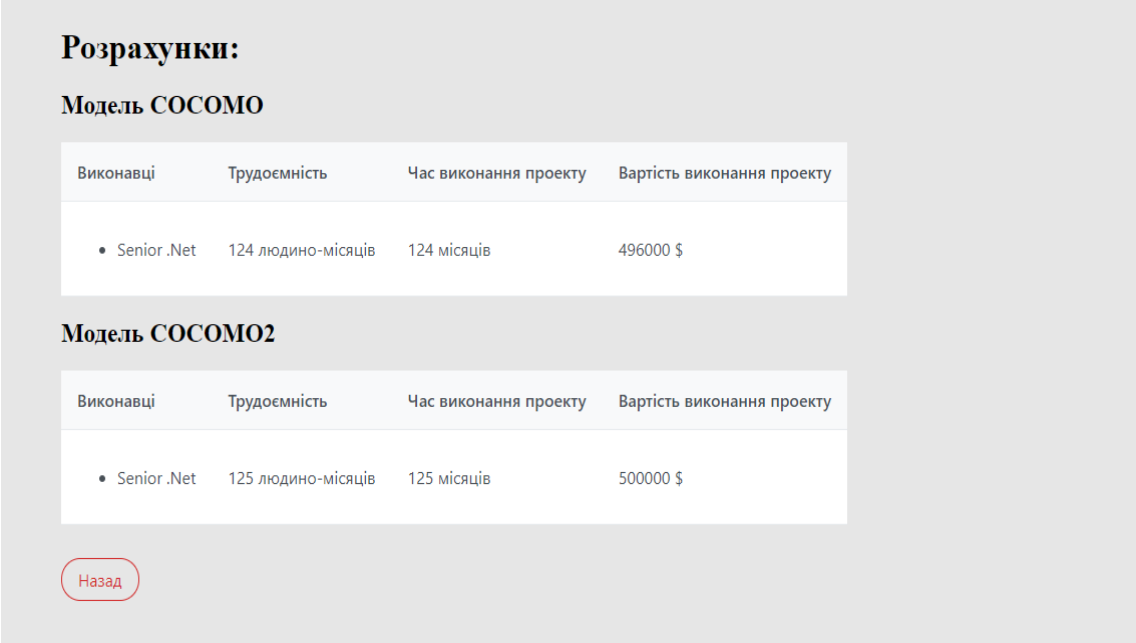


Рисунок 3.21 – Розрахунки трудоємності для команди з високим досвідом роботи та проекту високої складності

Порівняння результатів наведено у Таблицях 3.20-3.21.

Таблиця 3.20 – Значення підрахунків трудоємності моделі COCOMO

	Команда з низьким досвідом роботи	Команда з середнім досвідом роботи	Команда з високим досвідом роботи
Проект низької складності	37	10	3
Проект середньої складності	132	35	13
Проект високої складності	1243	332	124

Таблиця 3.21 – Значення підрахунків трудоємності детальної оцінки моделі COCOMO II

	Команда з низьким досвідом роботи	Команда з середнім досвідом роботи	Команда з високим досвідом роботи
Проект низької складності	6	1	0
Проект середньої складності	135	31	8
Проект високої складності	2105	492	125

За результатами роботи експериментів виявлено, що трудоємність залежить від команди, яка планує виконувати проект. Результати підрахунків трудоємності відрізняються у декілька разів, в залежності від досвіду команди проекту. Це спостерігається на різних рівнях складності проекту. Таким чином, можна зробити висновок, що формування команди проекту напряму впливає на результати підрахунку трудоємності, часу виконання та вартості проекту за допомогою моделей COCOMO та COCOMO II.

3.7 Висновки до розділу

У цьому розділі було розглянуто засоби розробки програмного забезпечення, до яких належить:

- .Net – безкоштовна платформа з відкритим кодом для створення багатьох різних типів додатків;
- C# – найбільш універсальна мова програмування серед усіх;
- MSSQL, що є неймовірно популярним рішенням для баз даних, яке використовується сьогодні, і однією з його найсильніших переваг є простота використання;
- Angular – платформа JavaScript з відкритим вихідним кодом, написана на TypeScript.

Також, у цьому розділі було висунуто функціональні вимоги до серверної та клієнтської частини застосунку.

Був наданий опис бази даних застосунку та детально розглянуто кожну з сутностей, які використовуються у системі.

Для полегшення роботи із застосунком було розглянуто інструкцію користувача з докладним описом використання основних функцій програмного забезпечення.

Для оцінки якості ПЗ було створено декілька сценаріїв тестування:

- «Спроба реєстрації з неправильним паролем у полі повторного вводу паролю»;
- «Спроба реєстрації»;
- «Спроба додання проекту з незаповненими полями форми»;
- «Додання проекту з заповненими полями форми»;
- «Формування групи обраного проекту без працівників»;
- «Формування групи обраного проекту»;
- «Спроба видалення працівника».

4 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження [16].

Таблиця 4.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Веб-застосунок для формування робочої групи ІТ-проекту	Використання сервісу у ІТ-компаніях для підрахунку вартості та часу виконання ІТ-проекту	Користувач отримує змогу сформувати робочу групу ІТ-проекту в залежності від вартості виконання ІТ-проекту та часу виконання
	Використання сервісу при створенні стартап-проекту для підрахунку вартості та часу виконання проекту	

Таблиця 4.2 — Опис ідеї стартап-проєкту

№	Техніко-економічні характеристики ідеї	Продукція конкурентів				W	N	S
		Проєкт	COCOMO	COCOMO II	Monday			
1	Підрахунок вартості та часу виконання проекту	+	+	+	-	-	-	+
2	Збереження даних про працівників проекту	+	-	-	+	-	+	-
3	Формування робочої групи проекту	+	-	-	-	-	-	+
4	Збереження інформації сформованої робочої групи для проекту	+	-	-	+	-	+	-

W / N / S – слабка / нейтральна / сильна сторона

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових:

– за якою технологією буде виготовлено товар згідно ідеї проекту?
– чи існують такі технології, чи їх потрібно розробити/доробити?
– чи доступні такі технології авторам проекту? [16]

Таблиця 4.3 — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Збереження інформації про проект та працівників	MS SQL SERVER, ORACLE	Наявна	Доступна
2	Веб-застосунок	.Net, Java, Python, Angular	Наявна	Доступна
3	Підрахунок вартості та часу виконання проекту	Моделі оцінки вартості ПЗ	Наявна	Доступна

Обрана технологія реалізації ідеї проєкту 1: MS SQL Server, .Net, Angular, Моделі оцінки вартості ПЗ.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів [16].

Таблиця 4.4 — Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, грн./ум.од	Немає точної публічної статистики
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	ДСТУ 2050-94. Системи оброблення інформації. Організація даних. ДСТУ 3918-99. Інформаційні технології. Процеси життєвого циклу ПО.
6	Середня норма рентабельності в галузі або по ринку, %	10-15%

Таблиця 4.5 — Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Покращення ефективності формування робочої групи ІТ-проекту	Власники ІТ-компаній, проект-менеджери	Немає	– підрахунок вартості та часу виконання проекту; – збереження даних про працівників проекту; – формування робочої групи проекту; – збереження інформації сформованої робочої групи для проекту.
2	Прискорення процесу формування робочої групи ІТ-проекту	Власники ІТ-компаній, проект-менеджери	Немає	
3	Прогнозування результатів формування робочої групи ІТ-проекту	Власники ІТ-компаній, проект-менеджери	Немає	

Таблиця 4.6 — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява конкурентів на ринку	При появі конкурентів на ринку, компанія може почати втрачати клієнтів та нести фінансові збитки	Поширення функціоналу продукту, програми лояльності для споживачів
2	Низька кількість потенційних споживачів	Вузька тематика проекту може не надати потрібної кількості клієнтів для отримання прибутку	Використання альтернатив розвитку проекту
3	Маленькі прибутки	Відсутність безкоштовних версій продукту може відштовхнути потенційних клієнтів від продукту	Зміна цінової політики, розробка безкоштовних версій продукту
4	Відмова співпраці іноземних компаній	Слабкість бренду може бути причиною відмови співпраці іноземних компаній	Поширення впливу на українському ринку
5	Відсутність потенційних інвесторів	Вузька тематика проекту може не надати потрібної кількості інвесторів для продовження існування проекту	Закриття компанії
6	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	– залучення додаткових інвесторів, мотивація роботи на перспективу; – ітеративна розробка продукту задля покрокового виведення продукту на ринок та отримання відповіді користувачів.
7	Вихід аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	– вихід товару на ринок в коротші строки з не повною, але достатньою, функціональністю для зацікавлення усіх цільових аудиторій; – проведення рекламної компанії.

Таблиця 4.7 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Відсутність конкуренції на ринку	Відсутність конкурентів дозволяє отримати вищі прибуток та клієнтську базу	Розвиток проекту, встановлення монополії
2	Отримання підтримки з боку IT-компаній	Відсутність аналогів дає можливість попросити підтримки від IT-компаній	Ведення перемов з представниками IT-компаній для отримання додаткового бюджету
3	Розширення компанії	Можливість виходу на міжнародний ринок дає змогу розширити компанію для роботи з іноземними партнерами	Збільшення кількості робітників
4	Покращення продукції	Можливість збільшення функціоналу для потреб споживачів дає змогу покращити продукт	Отримання додаткового прибутку на потреби компанії

Таблиця 4.8 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар від кожної компанії на ринку, являється недосконалим замінником товару, реалізованого іншими фірмами; На ринку є умови для входу та виходу; Ціна корелює між суперниками.	– розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники; – кореляція цін у відповідності до товарів замінників; – різни типи ліцензій.
2	Рівень конкурентної боротьби: світовий	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	– вихід на ринок збуту продукту з клієнто-необхідною функціональністю; – налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів; – надання бета-версій продукту.
3	Галузева ознака: внутрішньогалузева	Даний тип продукту може використовуватися тільки у сфері розробки IT додатків \ продуктів	– надання зручного, інтуїтивно зрозумілого інтерфейсу; – підтримка всім відомих методів взаємодії з середовищем розробки; – наявність документації та он-лайн підтримки.

Кінець таблиці 4.8

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція – конкуренція між товарами одного виду	– впровадження функціональності яка відсутня у товарів-замінників; – спрощення інтерфейсів; – надання підтримки.
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація; Не цінова – надання функціональності, що відсутня у товарах-замінниках.	– надання платних ліцензій лише на критично важливу функціональність для клієнта з певним строком підтримки, що зазначена у відповідній ліцензії; – впровадження унікальної функціональності.
6	За інтенсивністю: марочна	Наявність унікального знаку що відрізняє даний продукт від продуктів-замінників	Впровадження власної назви та власного знаку

Таблиця 4.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Відсутні	ІТ-компанії	-	ІТ-компанії, стартап-проекти	Неавтоматизовані аналоги
Висновки	Конкурентна боротьба не інтенсивна	Існують можливості виходу на ринок	-	Клієнти визначають потрібний функціонал	Обмежень для роботи на ринку немає

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею [16].

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

Таблиця 4.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Технічне обслуговування	Встановлення та підтримка свого продукту
2	Потреби споживачів	Потреби споживачів обумовлюють необхідність розробки проекту
3	Результативність	Завжди досягається кінцевий результат
4	Відсутність конкурентів	Більш раціональне використання ресурсів
5	Ціна та собівартість продукції	Не завищена, відповідна ціна

Таблиця 4.11 — Порівняльний аналіз сильних та слабких сторін програмного забезпечення формування робочої групи

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Технічне обслуговування	17						+	
2	Потреби споживачів	19			+				
3	Результативність	15			+				
4	Відсутність конкурентів	13		+					
5	Ціна та собівартість продукції	17							+

Таблиця 4.12 — SWOT аналіз стартап-проєкту

Сильні сторони (S): – висока лояльність до клієнтів; – оригінальність продукту; – якість продукту.	Слабкі сторони (W): – слабкість бренду, уперше на ринку; – вузька тематика продукту; – висока залежність від потреб клієнта.
Можливості (O): – співпраця з закордонними компаніями; – встановлення монополії; – отримання підтримки з боку IT-компаній; – покращення продукту; – розширення компанії.	Загрози (T): – поява конкурентів; – низька кількість потенційних клієнтів; – маленькі прибутки; – відмова співпраці іноземних компаній; – відсутність потенційних інвесторів.

Таблиця 4.13 — Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс – наявний	2-3 місяці
2	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс – наявний	2-3 тижні
3	Презентація товару на хакатонах й інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці

4.4 Розроблення ринкової стратегії проєкту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів [16]:

Таблиця 4.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Проект-менеджери IT-компаній	Висока	80%	Низька, тому що не існує аналогів на тему автоматизованого формування команд IT-проектів, проте є аналоги на тему формування команд IT-проектів вручну.	Легко, тому що веб-застосунок не потребує установки додаткових елементів та має зручний інтерфейс.
2	Власники IT-компаній	Висока	80%		

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є працівники IT сфери, IT компанії в цілому. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу, оскільки для підприємств, IT працівників та IT компаній у цілому надається стандартизований продукт з можливістю розширення функціональності за домовленістю.

Таблиця 4.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів	– зниження ступеню замінності товару; – прихильність клієнтів; – відмітні властивості товару; – відмітні характеристики товару.	Стратегія диференціації

Таблиця 4.16 — Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу	Стратегія заняття конкурентної ніші

Таблиця 4.17 — Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Простота використання	Стратегія диференціації	– технічна підтримка системи; – вартість.	– зручний інтерфейс; – адаптивність; – кросплатформеність.
2	Швидкодія	Стратегія диференціації	– технічна підтримка системи; – вартість.	– швидкий вхід в систему; – оптимізація; – зменшення розміру інформації.
3	Функціонал	Стратегія диференціації	– технічна підтримка системи; – вартість.	– зберігання інформації; – обробка інформації; – експонування та перегляд інформації.
4	Надійність	Стратегія диференціації	– технічна підтримка системи; – вартість.	– збереження інформації; – наявність складних паролей; – наявність двофакторної авторизації.

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації.

Таблиця 4.18 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Адаптивність	Веб-застосунок адаптований під різні розміри смартфонів, планшетів та комп’ютерів.	Систему можна використовувати з різних ґаджетів.
2	Кросплатформеність	Веб-застосунок можна використовувати на різних операційних системах.	Немає залежності від операційної системи ґаджета для використання системи.
3	Зручний інтерфейс	Веб-застосунок має інтуїтивно зрозумілий користувачу інтерфейс.	Зростає швидкість використання системи користувачем завдяки інтерфейсу.
4	Оптимізація	Веб-застосунок використовує швидкі алгоритми збереження та обробки інформації.	Зростає швидкість використання системи завдяки оптимізації.
5	Надійність	Веб-застосунок використовує технології для захисту інформації про користувача та його колекцію.	Користувачу не потрібно переживати про захист власних даних.

Таблиця 4.19 — Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Веб-застосунок для формування робочої групи ІТ-проекту.</td></tr>
<tr><td rowspan="11">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Адаптивність</td><td>М</td><td>Тх</td></tr>
<tr><td>2. Кросплатформеність</td><td>М</td><td>Тх</td></tr>
<tr><td>3. Зручний інтерфейс</td><td>М</td><td>Тх</td></tr>
<tr><td>4. Оптимізація</td><td>М</td><td>Тх</td></tr>
<tr><td>5. Надійність</td><td>М</td><td>Тх</td></tr>
<tr><td>6. Браузер</td><td>М</td><td>Тх</td></tr>
<tr><td>7. Процесор</td><td>М</td><td>Тх</td></tr>
<tr><td>8. Обсяг ОЗУ</td><td>М</td><td>Тх</td></tr>
<tr><td colspan="3">Якість: Відповідає рекомендаціям по вдосконаленню сайтів комітетів Верховної Ради України.</td></tr>
<tr><td colspan="3">Пакування: Електронна документація, що містить таку інформацію:
– загальна назва продукту, власна назва;
– найменування та адреса виробника, місце виготовлення;
– інструкція щодо користувача необхідні технічні вимоги.</td></tr>
<tr><td rowspan="2">3. Товар із підкріпленням</td><td colspan="3">До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товару</td></tr>
<tr><td colspan="3">Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, патент</td></tr>
</table>

Таблиця 4.20 — Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
0 грн	0 грн	Проект-менеджери: 30000 грн; Приватні компанії: 150000 грн.	Проект-менеджери: 100-300 грн; Приватні компанії (ліцензія на 10 людей): 1500-4500 грн.

Таблиця 4.21 — Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти купують продукт безпосередньо у компанії-розробника	– аналіз ринку; – технічна підтримка; – обслуговування.	0-Канал нульового рівня (виробник безпосередньо продає товар клієнту)	Через сайт виробника

Таблиця 4.22 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти дізнаються про веб-застосунок з соціальних мереж (instagram, facebook, twitter)	Інтернет; Соціальні мережі; Інтернет-сервіси з проект-менеджменту	– унікальний продукт; – потреби клієнта; – низька ціна.	Проінформувати про існування продукту	Рекламне звернення: «Створення команди IT-проекту вийшло на новий рівень!»
2	Клієнти дізнаються про веб-застосунок з інтернет-сервісів з проект-менеджменту		– унікальний продукт; – потреби клієнта; – низька ціна.	Проінформувати про існування продукту	

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

4.5 Висновки до розділу

В четвертому розділі описано стратегії та підходи з розроблення стартап-проєкту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проєкту. Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проєкту було встановлено що проєкт є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проєкту та доведено доцільність подальшої імплементації проєкту.

ВИСНОВКИ

У ході виконання магістерської дисертації розглянуто питання пов’язані з процесом формування робочих груп проекту.

На основі даних, отриманих в процесі аналізу, сформульовано мету, яка полягає в покращенні процесу формування робочих груп шляхом дослідження математичних методів і моделей та розробки програмного засобу для вирішення проблеми формування робочої групи з урахуванням впливу структури групи на ефективність проекту.

Наданий опис предметного середовища, проведений аналіз та зроблено постановку задачі. У якості методу для створення програмного забезпечення формування робочої групи обрано метод оцінки вартості ПЗ, а саме моделі COCOMO середнього рівня та COCOMO II для детальної оцінки.

Для розробки бібліотеки на основі методу оцінки вартості ПЗ використано платформу .Net та мову програмування C#. Для розробки програмного забезпечення формування робочої групи було використано платформу .Net, мови програмування C# та TypeScript, фреймворк Angular та СУБД MS SQL Server.

Проведений маркетинговий аналіз стартап-проєкту. Проведено опис ідеї проєкту, технологічний аудит ідеї проєкту, аналіз ринкових можливостей запуску стартап-проєкту. Розроблено ринкову стратегію проєкту та маркетингову програму стартап-проєкту.

Основні положення роботи доповідались і обговорювались на конференції «Інженерія програмного забезпечення і передові технології».

Найбільш суттєвими науковими результатами магістерської дисертації є:

- проведений аналіз актуальності задачі вибору та формування підходу до створення групи проекту, існуючих моделей і методів для її вирішення;
- вперше використання у програмному забезпеченні моделі оцінки вартості ПЗ для вирішення проблеми формування робочої групи.

Практичне значення отриманих результатів визначається тим, що запропоновані математичні методи доведені до практичної реалізації у рамках програмного забезпечення, котре використовується для формування робочої групи.

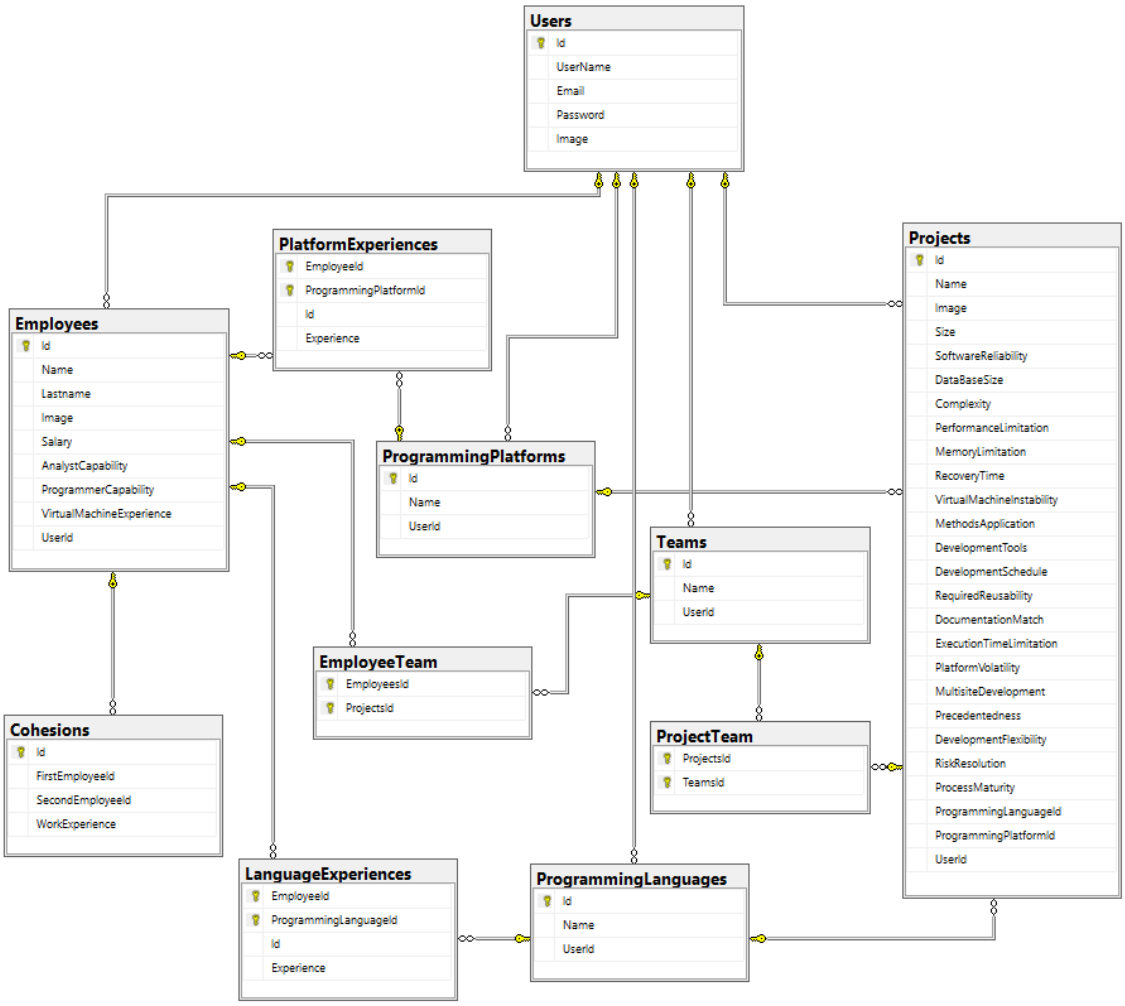
ПЕРЕЛІК ПОСИЛАНЬ

1) **Торгашова, А.В. Костіна, Г.Д.** Управління та розвиток команди проекту: методи та алгоритми (Стаття) [Електронний ресурс]. – 2017. Режим доступу: cyberleninka.ru/article/n/upravlenie-i-razvitie-komandy-proekta-metody-i-algoritmy/pdf
2) **Донелсон Р. Форсіт.** Group Dynamics. – 5-е вид. – Wadsworth Cengage Learning, 2010.
3) **Новиков Д.А.** Теорія управління організаційними системами. – 2-е вид. – М.: Физматлит, 2007.
4) **Новиков Д.А.** Математичні моделі формування та функціонування команд. – М.: Физмалит, 2008.
5) Workfront. It project-management for beginners [Електронний ресурс]. Режим доступу: https://www.workfront.com/blog/it-project-management-for-beginners
6) Підручники: Інформаційні системи і технології на підприємствах [Електронний ресурс].
   Режим доступу: https://pidru4niki.com/1057011647752/informatika/sutnist_ponyattya_proektu
7) Docplayer. Конструктивна модель вартості COCOMO [Електронний ресурс].
   Режим доступу: https://docplayer.com/71575863-Laboratornaya-rabota-1-konstruktivnaya-model-stoimosti-cocomo.html
8) Javatpoint. Putnam Resource Allocation Model [Електронний ресурс]. Режим доступу: https://www.javatpoint.com/putnam-resource-allocation-model
9) **Баррі Боем.** Software cost estimation with COCOMO II. Englewood Cliffs, NJ: Prentice Hall, 2000

10) Microsoft. .Net tutorial [Електронний ресурс].
Режим доступу: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/intro
11) C# Corner. What is .Net [Електронний ресурс].
Режим доступу: https://www.c-sharpcorner.com/article/what-is-net/
12) Search Data management. Microsoft SQL Server [Електронний ресурс]
Режим доступу:
https://searchdatamanagement.techtarget.com/definition/SQL-Server
13) Atlantic. What is MS SQL [Електронний ресурс].
Режим доступу: https://www.atlantic.net/vps-hosting/what-is-mssql/
14) Simplilearn. What is Angular [Електронний ресурс].
Режим доступу: https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular
15) Primeng [Електронний ресурс].
Режим доступу: https://www.primefaces.org/primeng/showcase/#/
16) Розроблення стартап-проекту [Електронний ресурс].
Режим доступу:
https://ep.kpi.ua/files/navchannia/mag/roz_startap_proektiv_met_vk.pdf

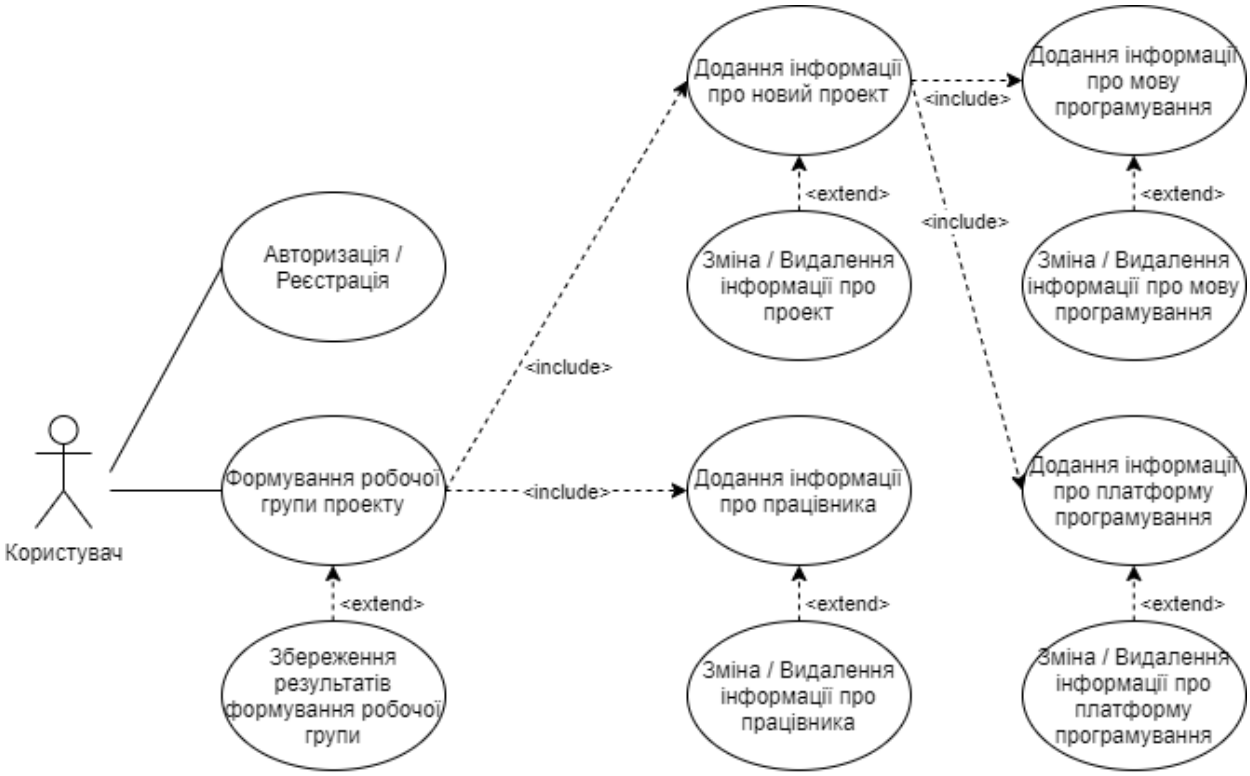
ДОДАТОК А

Структурна схема бази даних



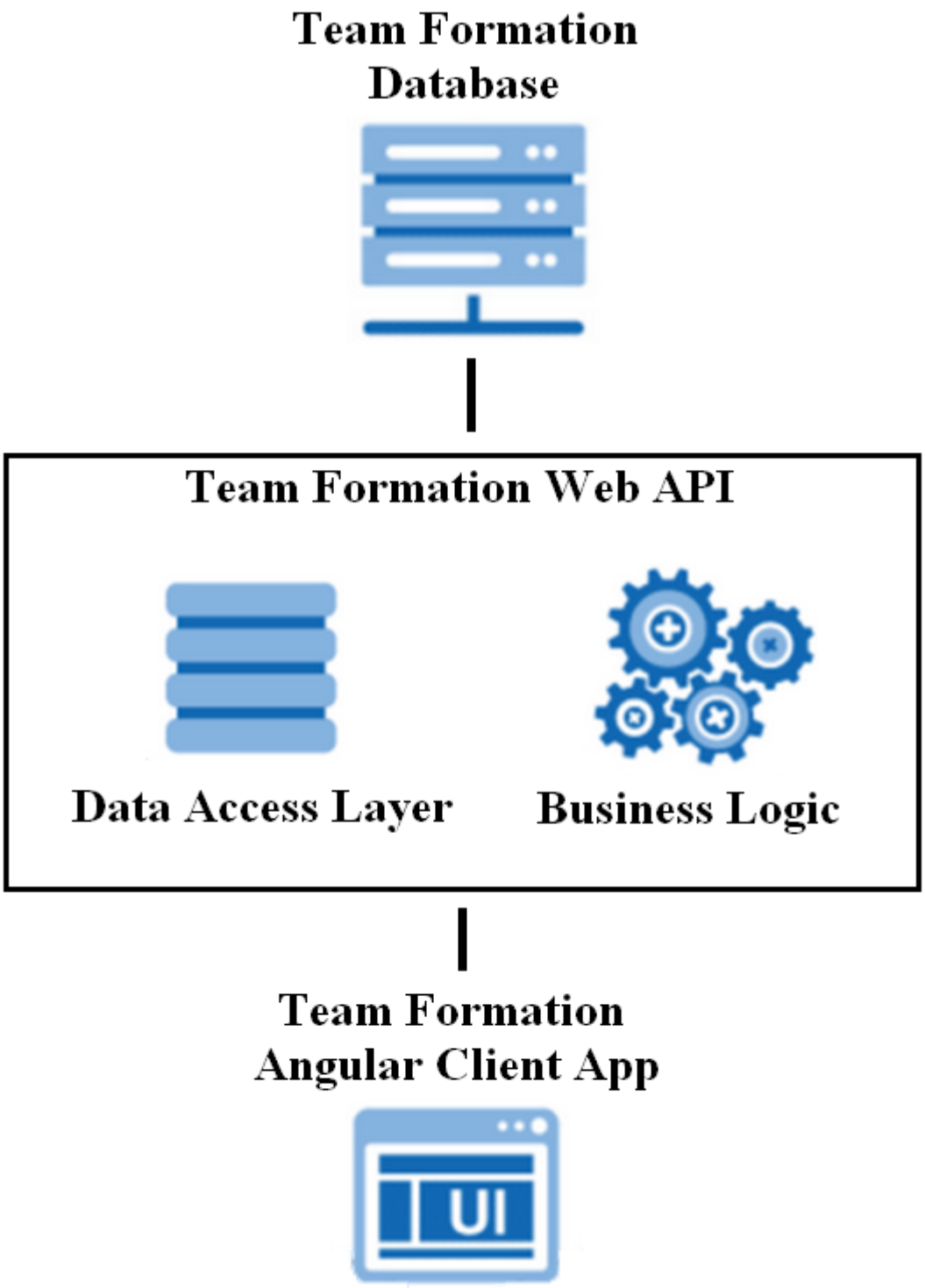
ДОДАТОК Б

Діаграма варіантів використання



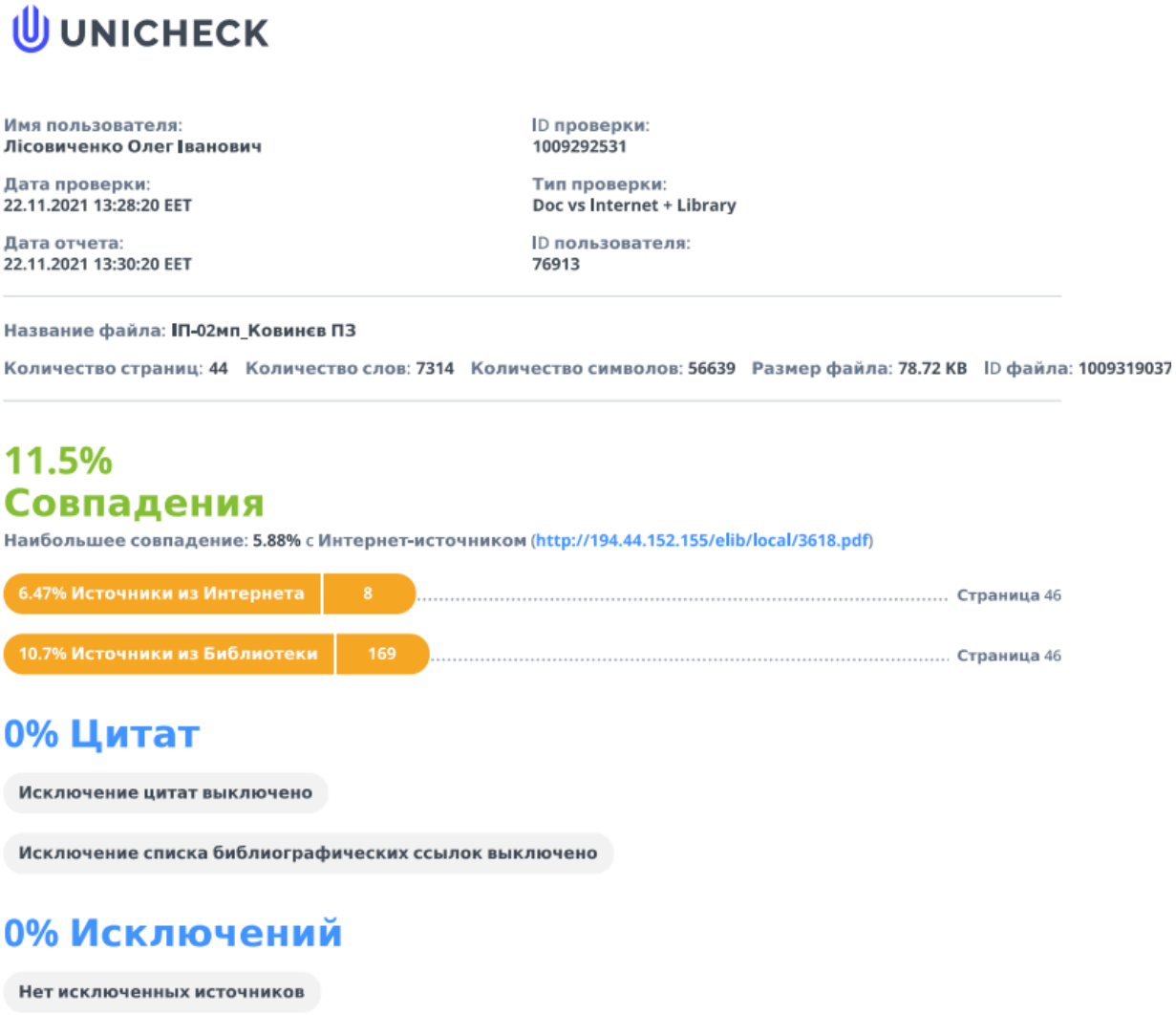
ДОДАТОК В

Схема архітектури програмного забезпечення



ДОДАТОК Г

Звіт перевірки на плагіат



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009292531

Дата проверки:
22.11.2021 13:28:20 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
22.11.2021 13:30:20 EET

ID пользователя:
76913

Название файла: ІП-02мп_Ковинєв ПЗ

Количество страниц: 44 Количество слов: 7314 Количество символов: 56639 Размер файла: 78.72 KB ID файла: 1009319037

11.5%
Совпадения

Наибольшее совпадение: 5.88% с Интернет-источником (http://194.44.152.155/elib/local/3618.pdf)

6.47% Источники из Интернета 8 ..... Страница 46

10.7% Источники из Библиотеки 169 ..... Страница 46

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

ДОДАТОК Д

Лістинг коду бібліотеки на основі методу оцінки вартості

```
public class COCOMO
    {
        public Product Product { get; set; }
        public Project Project { get; set; }
        public Hardware Hardware { get; set; }
        public List<User> Users { get; set; }
        private List<Group> Groups { get; set; } = new List<Group>();
        private int[] UsersId;
        public COCOMO(Product Product, Project Project, Hardware Hardware, List<User>
Users)
        {
            this.Product = Product;
            this.Project = Project;
            this.Hardware = Hardware;
            this.Users = Users;
        }
        public List<Group> GroupsFormating()
        {
            float a = 0;
            float b = 0;
            switch (Project.Type)
            {
                case 1:
                    a = 3.2f;
                    b = 1.05f;
                    break;
                case 2:
                    a = 3.0f;
                    b = 1.12f;
                    break;
                case 3:
                    a = 2.8f;
                    b = 1.20f;
                    break;
            }
            GroupsInitialization();
            foreach(Group group in this.Groups)
            {
                List<User> groupUsers = new List<User>();
                foreach(int Id in group.UsersId)
                {
                    groupUsers.Add(Users.Where(x => x.UserId == Id).FirstOrDefault());
                }
                var ab = CountEAF(this.Product, this.Project, this.Hardware, groupUsers);
                group.PM = CountEAF(this.Product, this.Project, this.Hardware,
groupUsers) * a * Math.Pow(Project.Size, b);
            }
            return this.Groups;
        }

        private float CountEAF(Product Product, Project Project, Hardware Hardware,
List<User> Users)
        {
            var a = ProductCharacteristicCount(Product);
            var b = ProjectCharacteristicCount(Project);
            var c = HardwareCharacteristicCount(Hardware);
            var d = PersonalCharacteristicCount(Users);
            return ProductCharacteristicCount(Product) *
                ProjectCharacteristicCount(Project) *
```

```
                HardwareCharacteristicCount(Hardware) *
                PersonalCharacteristicCount(Users);

        }
        private void GroupsInitialization()
        {
            List<int> Ids = new List<int>();
            foreach (User user in this.Users)
            {
                Ids.Add(user.UserId);
            }
            this.UsersId = new int[Ids.Count];
            Ids.CopyTo(this.UsersId);
            AddListGroup(0);
        }
        private void AddListGroup(int i, List<int> list = null)
        {
            List<int> copyList;

            for (int j = i; j < UsersId.Length; j++)
            {
                if (list != null)
                {
                    copyList = new List<int>(list);
                }
                else
                {
                    copyList = new List<int>();
                }
                copyList.Add(UsersId[j]);
                Groups.Add(new Group { UsersId = copyList, PM = 0 });
                if (j + 1 != UsersId.Length)
                    AddListGroup(j + 1, copyList);
            }

        }

    private float ProductCharacteristicCount(Product Product)
        {
            return Coefficients.SoftwareReliability[Product.SoftwareReliability] *
                Coefficients.DataBaseSize[Product.DatabaseSize] *
                Coefficients.ProductComplexity[Product.ProductComplexity];
        }
        private float ProjectCharacteristicCount(Project Project)
        {
            return Coefficients.SoftwareTools[Project.SoftwareTools] *
                Coefficients.EngineeringMethods[Project.EngineeringMethods] *
                Coefficients.DevelopmentShedule[Project.DevelopmentShedule];
        }
        private float HardwareCharacteristicCount(Hardware Hardware)
        {
            return Coefficients.PerformanceConstraints[Hardware.PerformanceConstraints] *
                Coefficients.MemoryConstraints[Hardware.MemoryConstraints] *
                Coefficients.TurnaboutTime[Hardware.TurnaboutTime] *
                Coefficients.VirtualMachineVolatility[Hardware.VirtualMachineVolatility];
        }
        private float PersonalCharacteristicCount(List<User> Users)
        {
            int analystCapability = 0;
            int engineerCapability = 0;
            int languageExperienceQuality = 0;
            int applicationsExperienceQuality = 0;
            int virtualMachineExperienceQuality = 0;
            foreach (User user in Users)
```

```
{
    analystCapability += user.AnalystCapability;
    engineerCapability += user.EngineerCapability;
    foreach(int key in Coefficients.WorkQuality.Keys)
    {
        if (user.LanguageExperience <= key)
        {
            languageExperienceQuality += Coefficients.WorkQuality[key];
            break;
        }
    }
    if (user.LanguageExperience > 120)
        languageExperienceQuality += Coefficients.WorkQuality[120];
    foreach (int key in Coefficients.WorkQuality.Keys)
    {
        if (user.ApplicationsExperience <= key)
        {
            applicationsExperienceQuality += Coefficients.WorkQuality[key];
            break;
        }
    }
    if (user.ApplicationsExperience > 120)
        applicationsExperienceQuality += Coefficients.WorkQuality[120];
    foreach (int key in Coefficients.WorkQuality.Keys)
    {
        if (user.VirtualMachineExperience <= key)
        {
            virtualMachineExperienceQuality += Coefficients.WorkQuality[key];
            break;
        }
    }
    if (user.VirtualMachineExperience > 120)
        virtualMachineExperienceQuality += Coefficients.WorkQuality[120];
}
analystCapability /= Users.Count;
engineerCapability /= Users.Count;
languageExperienceQuality /= Users.Count;
applicationsExperienceQuality /= Users.Count;
virtualMachineExperienceQuality /= Users.Count;

int meanLanguageExperience = 0;
int meanApplicationsExperience = 0;
int meanVirtualMachineExperience = 0;
foreach (int key in Coefficients.WorkQuality.Keys)
{
    if (languageExperienceQuality <= Coefficients.WorkQuality[key])
    {
        meanLanguageExperience = key;
        break;
    }
}
if (languageExperienceQuality > Coefficients.WorkQuality[120])
    meanLanguageExperience = 120;
foreach (int key in Coefficients.WorkQuality.Keys)
{
    if (applicationsExperienceQuality <= Coefficients.WorkQuality[key])
    {
        meanApplicationsExperience = key;
        break;
    }

}
if (applicationsExperienceQuality > Coefficients.WorkQuality[120])
    meanApplicationsExperience = 120;
```

```
            foreach (int key in Coefficients.WorkQuality.Keys)
            {
                if (virtualMachineExperienceQuality <= Coefficients.WorkQuality[key])
                {
                    meanVirtualMachineExperience = key;
                    break;
                }

            }
            if (virtualMachineExperienceQuality > Coefficients.WorkQuality[120])
                meanVirtualMachineExperience = 120;

            int languageExperience = 0;
            int appliccationsExperince = 0;
            int virtualMachineExperience = 0;
            foreach(int key in Coefficients.Experience.Keys)
            {
                if(meanLanguageExperience <= key)
                {
                    languageExperience = Coefficients.Experience[key];
                    break;
                }

            }
            if (meanLanguageExperience > 72)
                languageExperience = 5;
            foreach (int key in Coefficients.Experience.Keys)
            {
                if (meanApplicationsExperience <= key)
                {
                    appliccationsExperince = Coefficients.Experience[key];
                    break;
                }

            }
            if (meanApplicationsExperience > 72)
                appliccationsExperince = 5;
            foreach (int key in Coefficients.Experience.Keys)
            {
                if (meanVirtualMachineExperience <= key)
                {
                    virtualMachineExperience = Coefficients.Experience[key];
                    break;
                }

            }
            if (meanVirtualMachineExperience > 72)
                virtualMachineExperience = 5;

            return Coefficients.AnalystCapability[analystCapability] *
                Coefficients.EngineerCapability[engineerCapability] *
                Coefficients.LanguageExperience[languageExperience] *
                Coefficients.ApplicationsExperience[appliccationsExperince] *
                Coefficients.VirtualMachineExperience[virtualMachineExperience];
        }
    }

public class Product
    {
        public int SoftwareReliability { get; set; }
        public int DatabaseSize { get; set; }
        public int ProductComplexity { get; set; }

        public Product()
```

```
        {
            this.SoftwareReliability = 3;
            this.DatabaseSize = 3;
            this.ProductComplexity = 3;
        }
    }

    public class Project
    {
        public int Size { get; set; }
        public int Type { get; set; }
        public int SoftwareTools { get; set; }
        public int EngineeringMethods { get; set; }
        public int DevelopmentShedule { get; set; }

        public Project()
        {
            this.Size = 10;
            this.Type = 1;
            this.SoftwareTools = 3;
            this.EngineeringMethods = 3;
            this.DevelopmentShedule = 3;
        }
    }

    public class Hardware
    {
        public int PerformanceConstraints { get; set; }

        public int MemoryConstraints { get; set; }

        public int TurnaboutTime { get; set; }

        public int VirtualMachineVolatility { get; set; }

        public Hardware()
        {
            this.PerformanceConstraints = 3;
            this.MemoryConstraints = 3;
            this.TurnaboutTime = 3;
            this.VirtualMachineVolatility = 3;
        }
    }

    public class User
    {
        public int UserId { get; set; }
        public int AnalystCapability { get; set; }
        public int EngineerCapability { get; set; }
        public int LanguageExperience { get; set; }
        public int ApplicationsExperience { get; set; }
        public int VirtualMachineExperience { get; set; }
    }


public static class Coefficients
    {
        public static Dictionary<int, float> SoftwareReliability = new Dictionary<int,
float>
        {
            [1] = 0.75f,
            [2] = 0.88f,
            [3] = 1.0f,
```

```
            [4] = 1.15f,
            [5] = 1.4f
        };
        public static Dictionary<int, float> DataBaseSize = new Dictionary<int, float>
        {
            [2] = 0.94f,
            [3] = 1.0f,
            [4] = 1.08f,
            [5] = 1.16f
        };
        public static Dictionary<int, float> ProductComplexity = new Dictionary<int,
float>
        {
            [1] = 0.7f,
            [2] = 0.85f,
            [3] = 1.0f,
            [4] = 1.15f,
            [5] = 1.30f,
            [6] = 1.65f
        };
        public static Dictionary<int, float> SoftwareTools = new Dictionary<int, float>
        {
            [1] = 1.24f,
            [2] = 1.10f,
            [3] = 1.0f,
            [4] = 0.91f,
            [5] = 0.83f
        };
        public static Dictionary<int, float> EngineeringMethods = new Dictionary<int,
float>
        {
            [1] = 1.24f,
            [2] = 1.10f,
            [3] = 1.0f,
            [4] = 0.91f,
            [5] = 0.82f
        };
        public static Dictionary<int, float> DevelopmentShedule = new Dictionary<int,
float>
        {
            [1] = 1.23f,
            [2] = 1.08f,
            [3] = 1.0f,
            [4] = 1.04f,
            [5] = 1.10f
        };
        public static Dictionary<int, float> PerformanceConstraints = new Dictionary<int,
float>
        {
            [3] = 1.0f,
            [4] = 1.11f,
            [5] = 1.30f,
            [6] = 1.66f
        };
        public static Dictionary<int, float> MemoryConstraints = new Dictionary<int,
float>
        {
            [3] = 1.0f,
            [4] = 1.06f,
            [5] = 1.21f,
            [6] = 1.56f
        };
        public static Dictionary<int, float> TurnaboutTime = new Dictionary<int, float>
        {
```

```
            [2] = 0.87f,
            [3] = 1.0f,
            [4] = 1.07f,
            [5] = 1.15f
        };
        public static Dictionary<int, float> VirtualMachineVolatility = new
Dictionary<int, float>
        {
            [2] = 0.87f,
            [3] = 1.0f,
            [4] = 1.15f,
            [5] = 1.30f
        };
        public static Dictionary<int, float> AnalystCapability = new Dictionary<int,
float>
        {
            [1] = 1.46f,
            [2] = 1.19f,
            [3] = 1.0f,
            [4] = 0.86f,
            [5] = 0.71f
        };
        public static Dictionary<int, float> EngineerCapability = new Dictionary<int,
float>
        {
            [1] = 1.42f,
            [2] = 1.17f,
            [3] = 1.0f,
            [4] = 0.86f,
            [5] = 0.70f
        };
        public static Dictionary<int, float> LanguageExperience = new Dictionary<int,
float>
        {
            [1] = 1.14f,
            [2] = 1.07f,
            [3] = 1.0f,
            [4] = 0.95f
        };
        public static Dictionary<int, float> ApplicationsExperience = new Dictionary<int,
float>
        {
            [1] = 1.29f,
            [2] = 1.13f,
            [3] = 1.0f,
            [4] = 0.91f,
            [5] = 0.82f
        };
        public static Dictionary<int, float> VirtualMachineExperience = new
Dictionary<int, float>
        {
            [1] = 1.21f,
            [2] = 1.10f,
            [3] = 1.0f,
            [4] = 0.90f
        };
        public static Dictionary<int, int> WorkQuality = new Dictionary<int, int>
        {
            [0] = 500,
            [3] = 500,
            [6] = 630,
            [12] = 878,
            [18] = 1200,
            [24] = 1600,
```

```
        [36] = 2300,
        [48] = 2890,
        [60] = 3380,
        [72] = 3980,
        [84] = 4000,
        [96] = 4100,
        [108] = 4200,
        [120] = 4300,
    };
    public static Dictionary<int, int> Experience = new Dictionary<int, int>
    {
        [6] = 1,
        [12] = 2,
        [36] = 3,
        [72] = 4,
        [120] = 5
    };
}
public class Group
{
    public List<int> UsersId = new List<int>();
    public double PM;
}
```