Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії і управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти магістр

Модель системи розподілу командного навантаження IT-проєкту

(тема)

Виконав:

студент 2 курсу, групи СПм-22-2

Пальвальов О.П.
(прізвище, ініціали)

Спеціальність 123 – Комп’ютерна інженерія

(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування

(повна назва освітньої програми)

Керівник: проф. Фесенко Т.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти магістр

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Пальвальову Олександру Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель системи розподілу командного навантаження IT-проєкту

затверджена наказом по університету від " 06 " листопада 2023 р. № 1299 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 січня 2024 р.

3. Вхідні дані до роботи 1)Тема роботи

4. Перелік питань, що потрібно опрацювати у роботі 1) Розробити модель системи розподілу командного навантаження 2) Оцінити релевантність по відношенню до сучасних проектних методологій

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд теми та суті завдання	06.11.23	
2	Аналіз існуючих рішень	10.11.23	
3	Розробка нового методу	18.11.23	
4	Реалізація методу в існуючих методологіях	27.11.23	
5	Підготовка презентації	05.01.24	
6	Відправка роботи на нормоконтроль	10.01.24	
7	Перевірка роботи на антиплагіат	12.01.24	

Дата видачі завдання 06.11.2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Фесенко Т.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 57 с., 30 рис., 12 джерел, 1 додаток

УПРАВЛІННЯ ІТ ПРОЕКТАМИ, РОЗПОДІЛ, НАВАНТАЖЕННЯ, БІЗНЕС, КОМАНДА, ЗАВДАННЯ

Об'єкт дослідження – ІТ-проекти, процес розподілу командного навантаження.

Предмет дослідження – вдосконалення методу розподілу командного навантаження в ІТ-проектах.

Метою кваліфікаційної роботи є розробка моделі комп’ютерної системи розподілу навантаження ІТ проекту з використанням методології Agile Scrum.

Метод дослідження – вивчення літератури, приєднання методу до існуючих методологій за допомогою програми на комп'ютері, теоретичні розрахунки.

Метод зручно використовувати в бізнес цілях для управління ІТ проектом.

Метод складений теоретично та може застосуватися у реальних проектах.

ABSTRACT

57 pages, 30 figures, 12 sources, 1 appendix.

IT PROJECT MANAGEMENT, DISTRIBUTION, LOADING, BUSINESS, TEAM, TASK

The object of research is IT projects, the process of team workload distribution.

The subject of the research is the improvement of the method of distribution of team workload in IT projects.

The purpose of the qualification work is to develop a model of the IT project load distribution computer system using the Agile Scrum methodology.

The purpose of the work is to develop a program that is connected to the Internet and organizes support for the work of the client system of those stations.

The research method is the study of the literature, joining the method to existing methodologies using a computer program, theoretical calculations.

The method is convenient to use for business purposes for IT project management.

The method is developed theoretically and can be applied in real projects.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ОДИНИЦЬ, СКОРОЧЕНЬ

РП – результативна продуктивність

СЕФ – середній емоційний фон

SP – Story Point (одиниця оцінки важкості завдань)

ВСТУП

У сучасному світі, де технології та інформаційні системи відіграють ключову роль у бізнесі та інших сферах діяльності, успіх IT-проектів стає надзвичайно важливим завданням. Однак здійснення цих проектів може бути вкрай складним завданням через різноманітні фактори, які впливають на їх успішне виконання. Один із таких факторів - ефективний розподіл командного навантаження. Розподіл командного навантаження в IT-проектах є процесом, який визначає, як завдання та відповідальності розподіляються між членами команди для досягнення поставлених цілей та завдань проекту. Цей аспект є критичним для досягнення успіху проекту, оскільки неефективний розподіл навантаження може призвести до затримок, недосягнення цілей та зростання витрат.

Об'єкт дослідження – IT-проекти, процес розподілу командного навантаження.

Предмет дослідження – вдосконалення методу розподілу командного навантаження в IT-проектах.

Метою кваліфікаційної роботи є розробка моделі комп’ютерної системи розподілу навантаження IT проекту з використанням методології Agile Scrum.

Для вирішення поставленої мети були сформульовані завдання:

1 проаналізувати актуальні методології управління проектами в контексті розподілу ресурсів в IT проектах;

2 дослідити можливості розподілу навантаження на основі Scrum;

3 визначити універсальність розробленої моделі системи;

4 розробити прототип системи розподілу командного навантаження на базі Agile Scrum;

5 розглянути та вирішити проблеми ефективного розподілу командного навантаження в IT-проектах, надати практичні рекомендацій для покращення

системи;

6 протестувати модель або прототип системи на ІТ проекті.

Робота має на меті допомогти фахівцям у галузі управління проектами та розробки ПЗ зрозуміти важливість та методи оптимізації системи розподілу командного навантаження в ІТ-проектах. Вона розглядається як практичний внесок у підвищення результативності та успішності проектів у цій динамічній галузі.

У цій роботі будуть розглянуті ключові аспекти розподілу командного навантаження, інструменти та методи, що допомагають оптимізувати цей процес, а також важливі практичні приклади та рекомендації для впровадження результатів дослідження в реальній практиці.

Також у цій роботі буде розроблено прототип системи, який буде реалізовувати розроблену модель. Розроблений прототип буде або додатком до вже існуючої системи управління проектом, або новою системою. Основними цілями таких додатку є полегшення та оптимізація процесів керування проектом, збільшення продуктивності, покращення комунікації в команді, а також забезпечення ефективного використання ресурсів і виконання завдань у встановлені строки.

Загалом результат роботи сприятиме покращенню управління проектами та досягненню успішних результатів у цій важливій галузі.

1 АНАЛІЗ ПОТРЕБ ТА ОСНОВНІ ВИМОГИ ДО РОЗПОДІЛУ ЗАВДАНЬ В ІТ-ПРОЕКТАХ

1.1 Особливості формування і розподілу завдань в команді ІТ проекту

У будь-якому проекті перш за все необхідно сформувати команду. Це є дуже важливо, адже саме команда є ключовим двигуном для успішного закінчення проекту. Команда повинна бути злагоджена, а завдання розподілені так, щоб проект розвивався. Від складу команди вже можна сформувати приблизну особливість розподілу завдань у проекті. Особливості формування і розподілу завдань в команді ІТ-проекту визначаються не лише професійними здібностями кожного учасника, але й комплексом методів та стратегій, що розглядаються у дослідженні Schwalbe [1]. Ці методи враховують різноманітні аспекти від психології спілкування у команді до аналізу та оцінки потенціалу кожного учасника, а також впроваджують корисні правила щодо організації навантаження людей, наприклад: “Інші тенденції, що впливають на майбутнє управління людськими ресурсами, стосуються годин що організації очікують роботи багатьох ІТ-фахівців і як вони винагороджують результативність. Сьогодні люди щасливіші, коли можуть працювати менше 40 годин на тиждень або працювати від додому. Якщо компанії добре планують свої проекти, вони можуть уникнути необхідності понаднормової роботи, або вони може дати зрозуміти, що понаднормова робота необов’язкова.”[1 с.346]. Дослідник акцентує увагу на тому, що навантаження має бути розподілене, а люди не мають працювати понаднормово.

Важливою особливістю формування завдань є управління часом. Є дві основні форми, які менеджери та інженери проектів можуть використовувати для кращого управління часом. Перша форма — це блокнот “що робити” (Рисунок 1.1)[2].

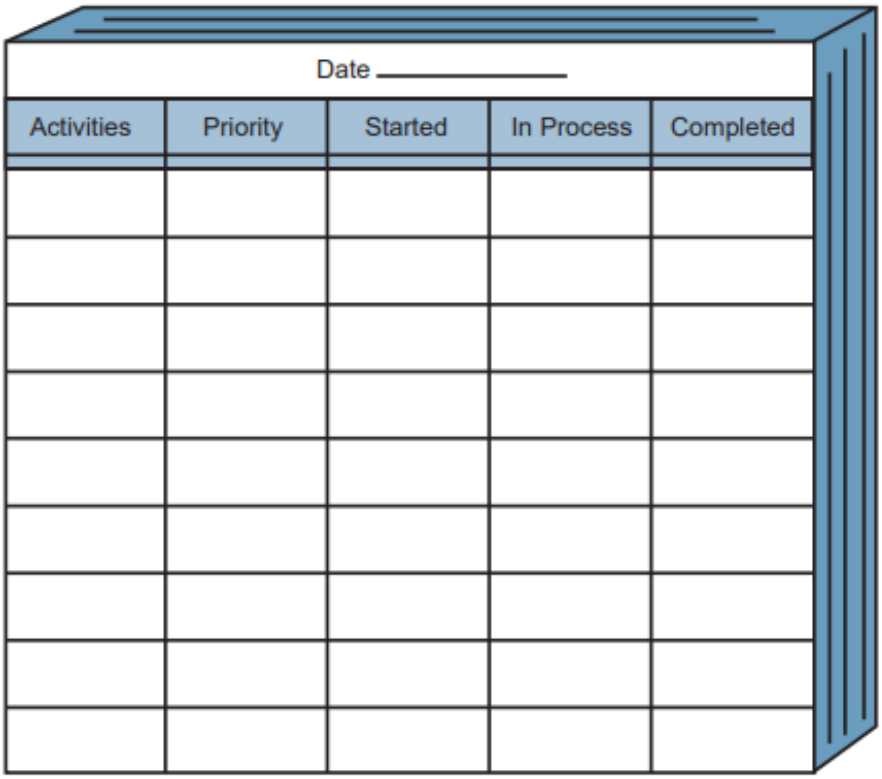
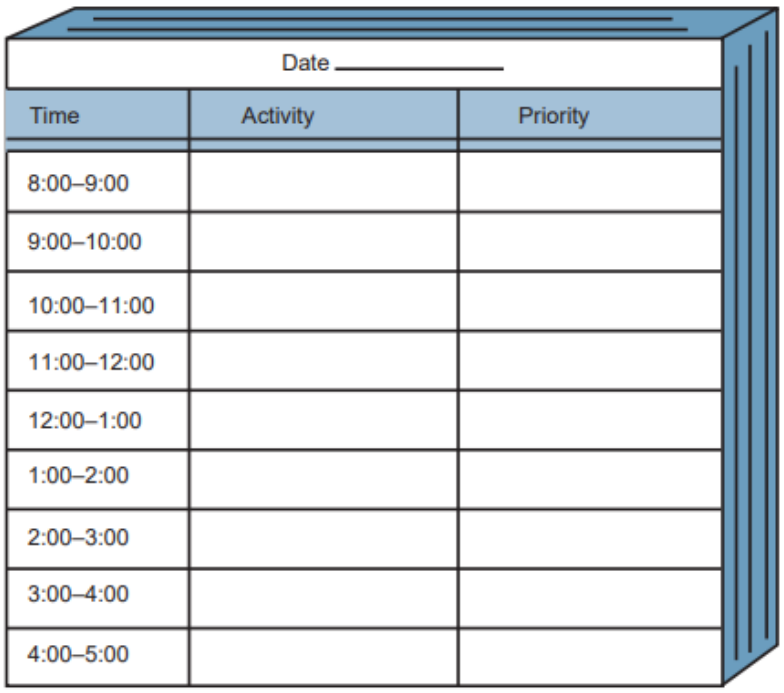


Рисунок 1.1 – блокнот “Що робити”[2]

Менеджер проекту або секретар готує список справ, які потрібно зробити. Потім керівник проекту вирішує, які дії він повинен виконувати сам, і призначає відповідні пріоритети. Дії з найвищим пріоритетом потім переносяться в “щоденний календар-журнал” (рисунок 1.2).



Date _____

Time	Activity	Priority
8:00–9:00		
9:00–10:00		
10:00–11:00		
11:00–12:00		
12:00–1:00		
1:00–2:00		
2:00–3:00		
3:00–4:00		
4:00–5:00		

Рисунок 1.2 – Щоденний календар-журнал[2]

Керівник проекту призначає ці дії до відповідних часових блоків на основі власного енергетичного циклу. Потім використовуються незаповнені блоки часу неочікуваних криз або для менш пріоритетних заходів. Якщо пріоритетних елементів більше, ніж часових інтервалів, керівник проекту може спробувати це зробити розклад завчасно. Зазвичай це не є хорошою практикою, оскільки це створює відставання пріоритетних видів діяльності. Крім того, діяльність, яка сьогодні є пріоритетом «B», може легко стати пріоритетом «A» за день-два. Мораль тут – не відкладай на завтра що ви або ваша команда можете зробити сьогодні.[2]. Управління часом може сприяти покращенню розподілення навантаження в IT проекті, бо робота виконується у більшості випадків за час, тому це дуже впливає на подальше розподілення завдань на проекті.

Якщо дивитись на проект з точки зору ресурсів, то Pinto[3] приводить приклад розподілення ресурсів та конфлікти якщо їх не правильно розподілити. Поняття завантаження ресурсів стосується кількості окремих

ресурсів, необхідних для розкладу протягом певних періодів часу. Ми можемо завантажити або розмістити в детальному розкладі ресурси з урахуванням конкретних завдань або проекту в цілому. Однак, як правило, загалом корисно робити обидва: створити загальну таблицю завантаження ресурсів проекту, а також визначити потреби в ресурсах для кожного окремого завдання. З практичної точки зору, завантаження ресурсів намагається призначити відповідний ресурс у відповідному ступені або кількості для кожної діяльності проекту.[3]

Saldaña та López [4] акцентують увагу на важливості правильного підбору команди, враховуючи їхні здібності та попередній досвід. Це допомагає створити команду, яка працює як добре налаштована машина, де кожен виконує відповідальність, що максимізує продуктивність проекту.

Існує ще два способи керування навантаженням ресурсів: завантаження ресурсів і вирівнювання[5]. Вирівнювання ресурсів — це техніка вирішення конфліктів ресурсів відкладаючи завдання. Це форма аналізу мережі, у якій ресурс управління рішень відповідає плануванню (дати початку і кінця). Основною метою вирівнювання ресурсів є створення більш плавного розподілу використання ресурсів. Керівники проекту перевіряють діаграму мережі для областей провисання або плавання, а також ідентифікують ресурсні конфлікти. Наприклад, іноді можна видалити перерозподіл через затримку виконання некритичних завдань, що не приводить до результату у загальній затримці графіка. В інший час вам доведеться відкласти дату завершення проекту, щоб зменшити або видалити додаткові кошти. Навантаження ресурсів можна додатково спостерігати за допомогою ресурсних гістограм (рисунок 1.3) щоб відобразити коливання завантаження ресурсів за часом. Гістограма може бути дуже корисно для визначення потреб або проблеми персоналу.

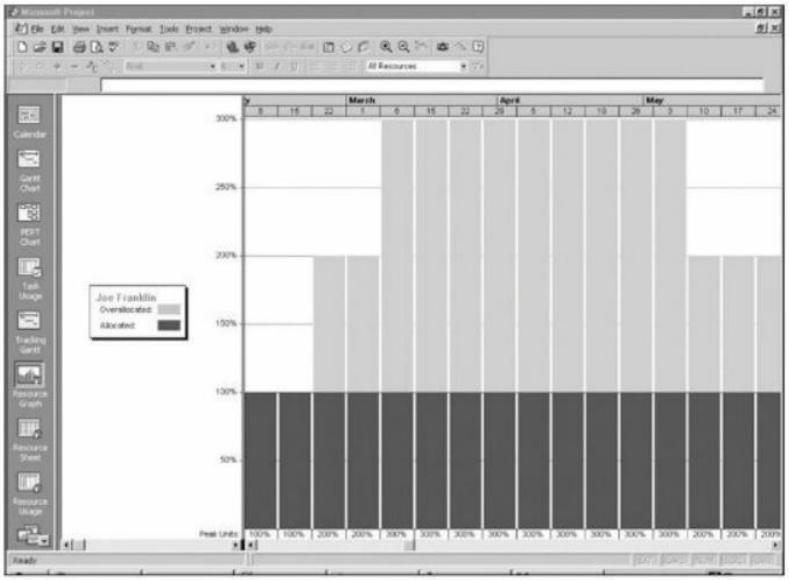


Рисунок 1.3 – Гістограма що показує перевантажені ресурси[5]

У контексті розробки програмного забезпечення, дослідження Dingsoyr та Moe [6] додають у вигляді рішень економічні та методологічні аспекти управління завданнями. Це дозволяє забезпечити оптимальний розвиток проекту з урахуванням обмежених ресурсів. З огляду на актуальність Agile та Scrum важливо адаптивно та гнучко розподіляти завдання для успіху в цих методологіях.

1.2 Визначення вимог системи командного навантаження IT проекту. Постановка задачі

Система командного навантаження в IT-проекті може мати різноманітні вимоги, орієнтовані на ефективне управління завданнями та ресурсами команди для досягнення успішних результатів. Вимоги до системи командного навантаження в IT-проекті можуть варіюватися в залежності від конкретних потреб команди та характеру проекту.

Аналіз вимог є кроком, який покладає основу для подальшого розподілу командного навантаження в IT-проекті. Цей процес допомагає

зрозуміти, які завдання необхідно виконати. Можна визначити, що ретельний аналіз вимог проекту є критичним для досягнення конкурентної переваги[3]. Цей аналіз включає в себе розуміння вимог стейкхолдерів, тобто всіх зацікавлених сторін проекту. Зважаючи на це, команді проекту слід здійснити ретельний огляд потреб кожного стейкхолдера, зокрема врахувати їх очікування, бажання та функціональні вимоги до проекту.

Ефективне розподілення навантаження у команді проекту має суттєве значення для досягнення високої продуктивності та результативності. Розуміння навантаження кожного члена команди, їхніх сильних сторін та можливостей дозволяє керувати ресурсами більш ефективно, уникати перевантажень і забезпечити баланс у роботі всієї команди.[4].

Основні вимоги полягають у чіткості завдань, а саме система повинна забезпечувати чітке формулювання завдань для кожного учасника команди, учаснику може автоматично призначатись завдання, яке буде найбільш оптимальне. Також система повинна бути гнучкою та адаптивною [7]. Вимога до системи полягає у її здатності адаптуватися до змін в процесі роботи, враховуючи вплив нових вимог, пріоритетів чи змін у складі команди. У команді має бути ефективний розподіл завдань у відповідності до навичок, досвіду та потенціалу учасників команди, на проекті має бути оптимальне використання ресурсів команди, таких як час, таланти та знання, для максимізації продуктивності.

Система повинна забезпечувати моніторинг виконання завдань та створення звітів про прогрес для ефективного контролю і прийняття рішень, деякі сучасні програмні додатки (наприклад Jira, Microsoft Project, Trello, Projectlibre) мають функціонал до цього. Але у цих додатках немає налаштувань для вибору завдань, наприклад якщо не буде чіткого лідера на проекті, як це часто буває зі студентами, або коли розробникам завдання пропонується згідно оцінки їхнього емоційного стану. Тому розроблена система розподілу навантаження має підтримувати цей функціонал. Система розподілу навантаження може не просто перекривати функціонал існуючих

програм, а доповнювати їх шляхом впровадження додаткового аналізу.

Система має підтримувати Agile, Scrum проектну методологію.

1.3 Врахування обмежень та ризиків

Важливо також враховувати обмеження та ризики, які можуть вплинути на розподіл командного навантаження. Це включає в себе розгляд можливих перешкод, обмежень у ресурсах та технічних складностей.

Серед можливих проектних ризиків є: наявність обмежень у доступних ресурсах, таких як людські, фінансові або матеріальні що може обмежувати можливості для розподілу завдань. Наприклад, якщо в команді обмежена кількість спеціалістів у певній області, це може ускладнити розподіл завдань, які потребують цих навичок.

У випадку обмеження часу на виконання проекту, необхідно обдумати, як кожне завдання впливає на загальний графік та як ефективно розподілити навантаження так, щоб відповідати строкам.

Також можуть з’явитись неочікувані ризики, які можуть виникнути протягом проекту, такі як зміни у вимогах, проблеми з ресурсами або технічні проблеми, можуть вплинути на планований розподіл завдань. Через це може виникнути необхідність у перерозподілені завдання або змінити пріоритети в ході проекту.

Навантаження має бути збалансованим, адже деякі учасники команди можуть мати вищий обсяг роботи або бути більш завантаженими, що може призвести до дисбалансу у розподілі завдань.

Іноді у команді можуть виникати конфлікти між членами команди або несумісність інтересів, що може ускладнити розподіл завдань та викликати затримки. На такі ситуації слід звернути особливу увагу, адже розлад команди у проекті може зруйнувати усю роботу. Конфлікти — це спосіб життя в структурі проекту, і вони, як правило, можуть виникнути на будь-якому рівні організації, як правило, в результаті суперечливих цілей.[2]

Необхідно ретельно враховувати усі ці ризики для забезпечення успішного завершення проекту.

1.4 Аналіз ресурсів та навичок команди

Окремо важливим аспектом є аналіз доступних ресурсів та навичок учасників команди. Експертів з управління проектами з різних галузей попросили визначити 10 найважливіших навичок і компетенцій для ефективних менеджерів проектів[1].(рисунок 1.4)

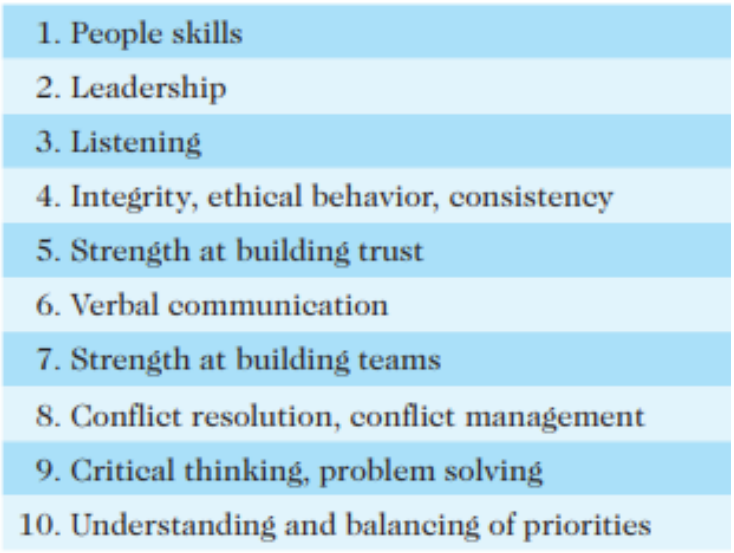


Рисунок 1.4 – 10 найважливіших якостей менеджера [1]

Щодо розробників, то IT-керівники перерахували «десять найпопулярніших навичок», з якими вони планували найняти в 2015 році.[1] Програмування та розробка додатків залишилися на першому місці, в основному через збільшення потреби в програмістах з досвідом мобільної розробки та досвідом створення безпечних додатків. Навички управління проектами продовжують складати список, оскільки ці навички є вирішальними для визначення пріоритетів потреб бізнесу та впровадження ефективних рішень. Рисунок 1.5 показує результати останнього опитування,

а також відсоток респондентів, які назвали ці навички корисними. Навіть якщо ви вирішите залишитися на технічній посаді, вам усе одно потрібні знання та навички з управління проектами, щоб допомогти вашій команді та організації досягти успіху.[1]

Skill	Percentage of Respondents
Programming and application development	48
Project management	35
Help desk/technical support	30
Security/compliance governance	28
Web development	28
Database administration	26
Business intelligence/analytics	24
Mobile application and device management	24
Networking	22
Big data	20

Рисунок 1.5 – 10 найгарячіших навичок для розробника[1]

Розподіл командного навантаження в IT-проектах також вимагає застосування методів прийняття рішень. Під час аналізу потреб, команді проекту слід розглядати різні варіанти розподілу завдань та обирати оптимальний. Модель системи розподілу навантаження може враховувати навички учасників команди під час розподілу завдань.

1.5 Застосування методів прийняття рішень

Методи прийняття рішень, такі як аналіз SWOT (сильні сторони, слабкі сторони, можливості, загрози), можуть бути використані для цього.

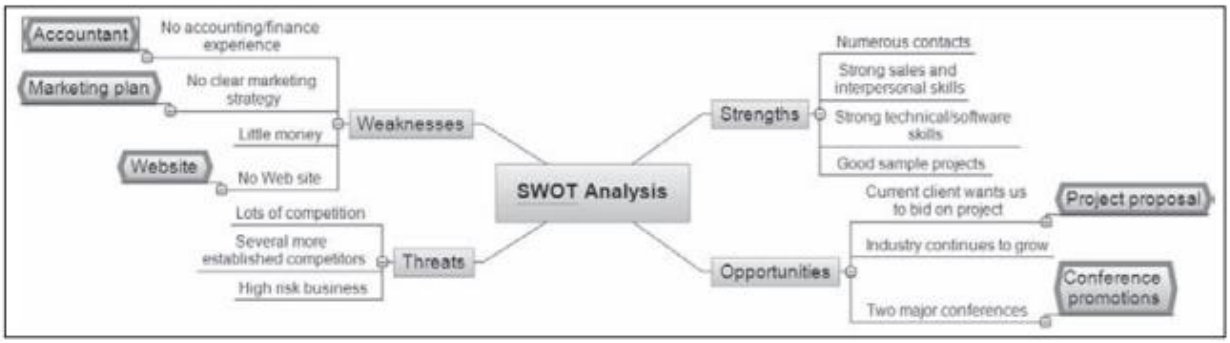


Рисунок 1.6 – Аналіз SWOT[1]

Деяким людям подобається виконувати SWOT-аналіз за допомогою інтелектуального мапування, техніки, яка використовує гілки, що виходять від основної ідеї, щоб структурувати думки та ідеї. Людський мозок не працює лінійно. У людей виникає багато незв'язаних ідей. Зафіксувавши ці ідеї у форматі візуальної мапи, ви часто можете створити більше ідей, ніж створюючи списки. Ви можете створювати інтелект-карти вручну, використовуючи наліпки, за допомогою програмного забезпечення для презентацій, наприклад Microsoft PowerPoint, або за допомогою програмного забезпечення для мапування.[1]

У системі розподілу навантаження SWOT-аналіз дозволить зробити вибірку завдань більш комфортною для учасника проекту.

1.6 Врахування індивідуальних особливостей команди

Під час розподілу командного навантаження важливо враховувати індивідуальні особливості кожного члена команди. Різні спеціалізації, досвід і навички можуть впливати на ефективність та продуктивність виконання завдань. Дослідження навичок, досвіду та компетенцій кожного учасника команди визначило їхню придатність для виконання певних завдань.

Керівники проектів, як правило, відомі тим, що мають багато делегованих повноважень, але дуже мало формальних повноважень. Тому

вони повинні виконувати роботу за допомогою міжособистісного впливу. Існує п'ять таких міжособистісних впливів[2]:

1 законна влада (можливість отримати підтримку, оскільки персонал проекту вважає, що керівник проекту офіційно уповноважений видавати накази.);

2 влада винагороди (можливість отримати підтримку, оскільки персонал проекту сприймає керівника проекту як здатного прямо чи опосередковано видавати цінні організаційні винагороди, тобто зарплату, просування по службі, бонуси, майбутні робочі завдання);

3 влада покарання (можливість отримати підтримку, оскільки персонал проекту вважає, що керівник проекту здатний прямо чи опосередковано застосовувати покарання, яких вони хочуть уникнути. Сила покарання зазвичай походить з того самого джерела, що й сила винагороди, причому одна є необхідною умовою для іншої);

4 експертна влада (здатність отримати підтримку, оскільки персонал сприймає керівника проекту як володаря спеціальних знань або досвіду, це функціональний персонал вважає важливим);

5 референтна влада (здатність отримати підтримку, оскільки персонал проекту відчуває особисту привабливість для керівника проекту або його проекту).

Експертна та референтна влада є прикладами особистої влади, яка походить від особистих якостей або характеристик, до яких приваблюють членів команди. Законні повноваження, винагороди та покарання часто називають прикладами посадової влади, яка безпосередньо пов’язана з позицією людини в організації.

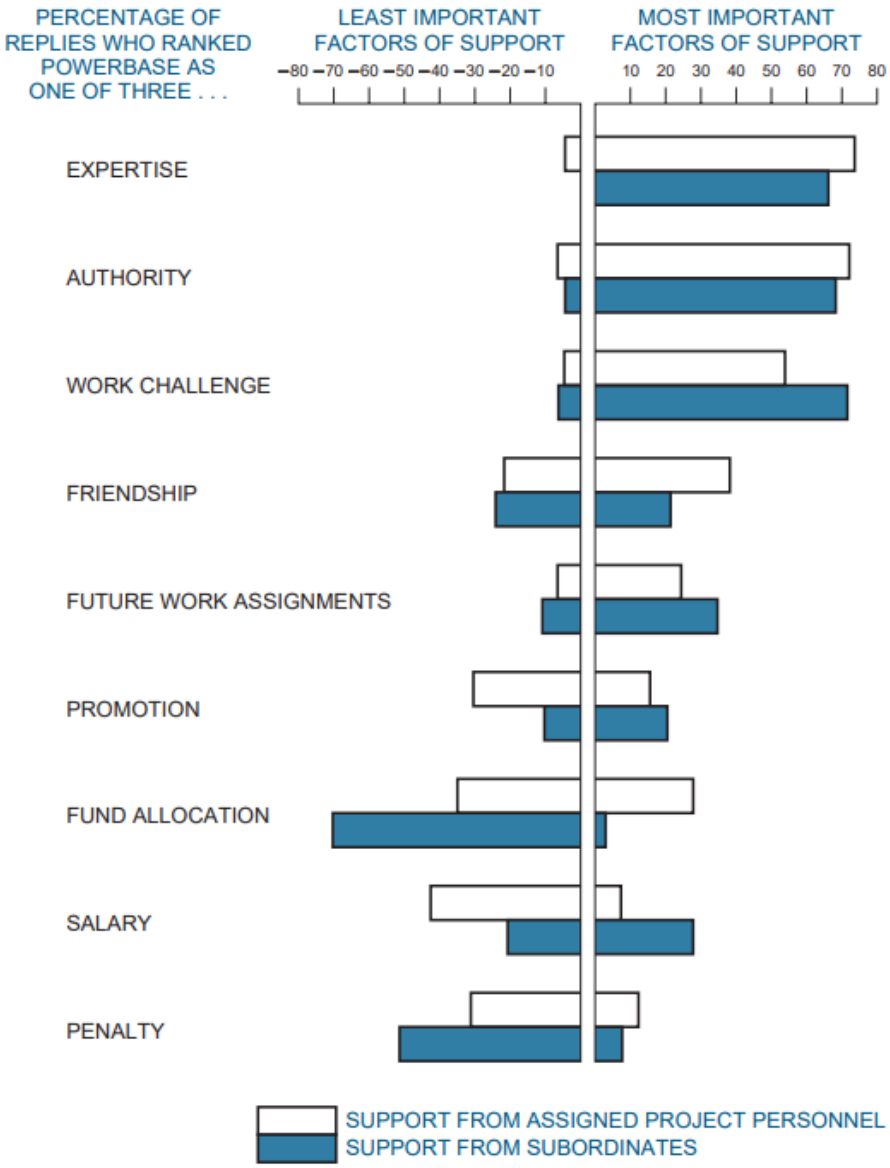


Рисунок 1.7 – Значення факторів підтримки та особливостей[2]

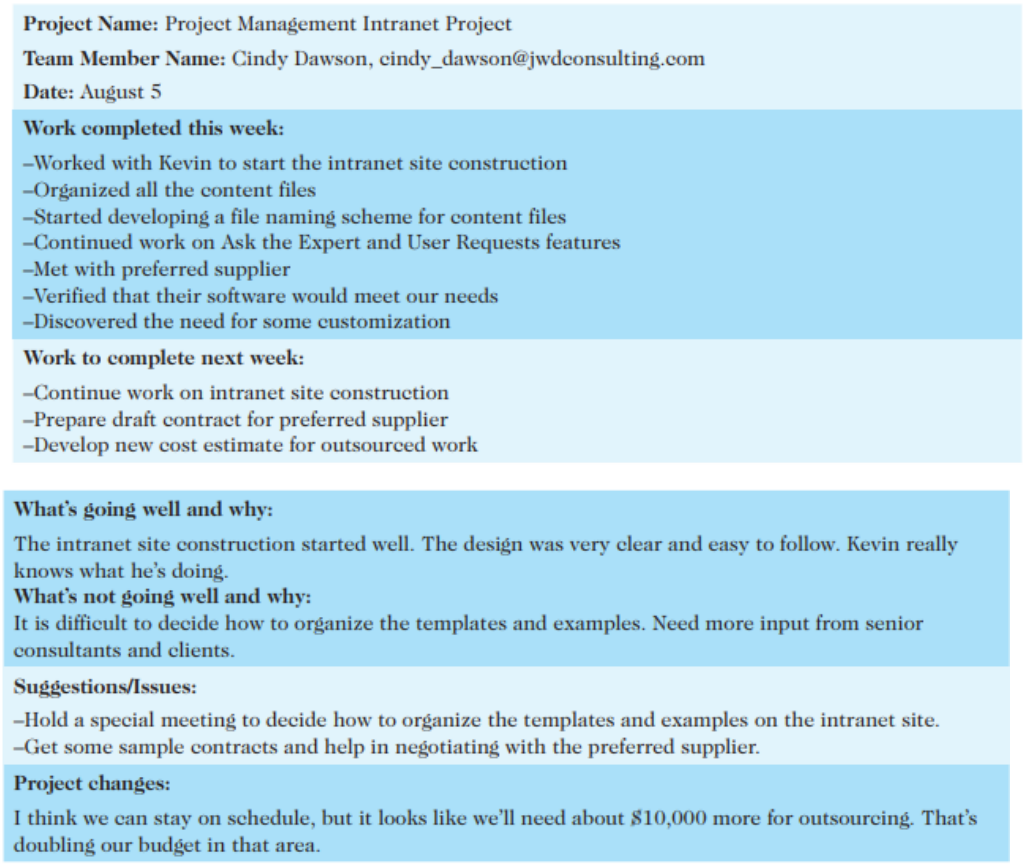
Цей аналіз сприяв більш ефективному призначенню завдань, уникненню перевантаження або недооцінки можливостей окремих членів команди.

1.7 Визначення механізмів контролю та звітності

Важливим етапом є визначення механізмів контролю та звітності. Команда повинна мати чіткий спосіб відслідковувати прогрес та

відстежувати виконання завдань.

Моніторинг і контроль — це процес вимірювання прогресу в досягненні цілей проекту, моніторингу відхилень від поточного плану та вжиття коригувальних дій для узгодження прогресу з поточним планом.[1] Моніторинг і контроль здійснюються протягом усього життєвого циклу проекту та охоплюють 9 із 10 сфер знань про управління проектом.



Project Name: Project Management Intranet Project

Team Member Name: Cindy Dawson, cindy_dawson@jwdconsulting.com

Date: August 5

Work completed this week:

–Worked with Kevin to start the intranet site construction
–Organized all the content files
–Started developing a file naming scheme for content files
–Continued work on Ask the Expert and User Requests features
–Met with preferred supplier
–Verified that their software would meet our needs
–Discovered the need for some customization

Work to complete next week:

–Continue work on intranet site construction
–Prepare draft contract for preferred supplier
–Develop new cost estimate for outsourced work

What's going well and why:

The intranet site construction started well. The design was very clear and easy to follow. Kevin really knows what he's doing.

What's not going well and why:

It is difficult to decide how to organize the templates and examples. Need more input from senior consultants and clients.

Suggestions/Issues:

–Hold a special meeting to decide how to organize the templates and examples on the intranet site.
–Get some sample contracts and help in negotiating with the preferred supplier.

Project changes:

I think we can stay on schedule, but it looks like we'll need about $10,000 more for outsourcing. That's doubling our budget in that area.

Рисунок 1.8 – приклад звіту [1]

Крім того, необхідно встановити механізми звітності, щоб інформувати замовника або стейкхолдерів про стан проекту. Система може реалізовувати моніторинг у реальному часі, або формувати звіт за проміжок часу. Більший функціонал дозволить відправляти звіт електронною поштою, що зробить процес контролю більш зручним.

2 РОЗРОБКА МОДЕЛІ СИСТЕМИ РОЗПОДІЛУ КОМАНДНОГО НАВАНТАЖЕННЯ ІТ ПРОЕКТУ

2.1 Вибір методів створення системи розподілу завдань в команді ІТ проекту

Для реалізації моделі системи навантаження необхідно зробити підтримку проектної методології. Оскільки кожен проект унікальний, команди проекту завжди намагаються зробити те, чого раніше не робили. Щоб досягти успіху в унікальних і нових видах діяльності, проектні команди повинні добре планувати. Треба мати на увазі, що найбільше часу та грошей зазвичай витрачається на виконання[1]. Хорошою практикою для організацій є визначення того, як управління проектами працюватиме найкраще в їхніх власних організаціях.

Життєвий цикл розробки систем (SDLC) є основою для опису фаз розробки інформаційних систем. Деякі популярні моделі SDLC включають водоспадну модель, спіральну модель, модель поетапного збирання, модель прототипування та модель швидкої розробки додатків (RAD). Ці моделі життєвого циклу є прикладами передбачуваного життєвого циклу, що означає, що обсяг проекту можна чітко сформулювати, а графік і вартість можна точно передбачити. Команда проекту проводить значну частину проекту, намагаючись уточнити вимоги до всієї системи а потім створити дизайн. Користувачі часто не можуть побачити будь-яких відчутних результатів з точки зору робочого програмного забезпечення протягом тривалого періоду. Нижче наведено короткий опис кількох прогнозних моделей SDLC:[1]

Модель життєвого циклу водоспаду має чітко визначені лінійні етапи системного аналізу, проектування, будівництва, тестування та підтримки. Ця модель життєвого циклу передбачає, що вимоги

залишатимуться стабільними після їх визначення. Модель життєвого циклу водоспаду використовується, коли необхідно жорстко контролювати ризик і коли зміни слід обмежити після визначення вимог. Водоспадний підхід використовується в багатьох великомасштабних системних проектах, де складність і вартість настільки високі, що більш жорсткі кроки підходу допомагають забезпечити ретельне завершення всіх результатів.

Спіральну модель життєвого циклу було розроблено на основі уточнень моделі водоспаду у застосуванні до великих урядових проектів програмного забезпечення. Він визнає той факт, що більшість програмного забезпечення розробляється з використанням ітераційного або спірального підходу, а не лінійного. Команда проекту відкрита для змін і переглядів пізніше в життєвому циклі проекту та повертається до фази вимог, щоб більш ретельно прояснити та спроектувати зміни. Цей підхід підходить для проектів, у які можна внести зміни з розумним збільшенням вартості або прийнятними часовими затримками. [1]

На рисунку 2.1 зображені відмінності між моделями життєвого циклу водоспаду та спіралі.

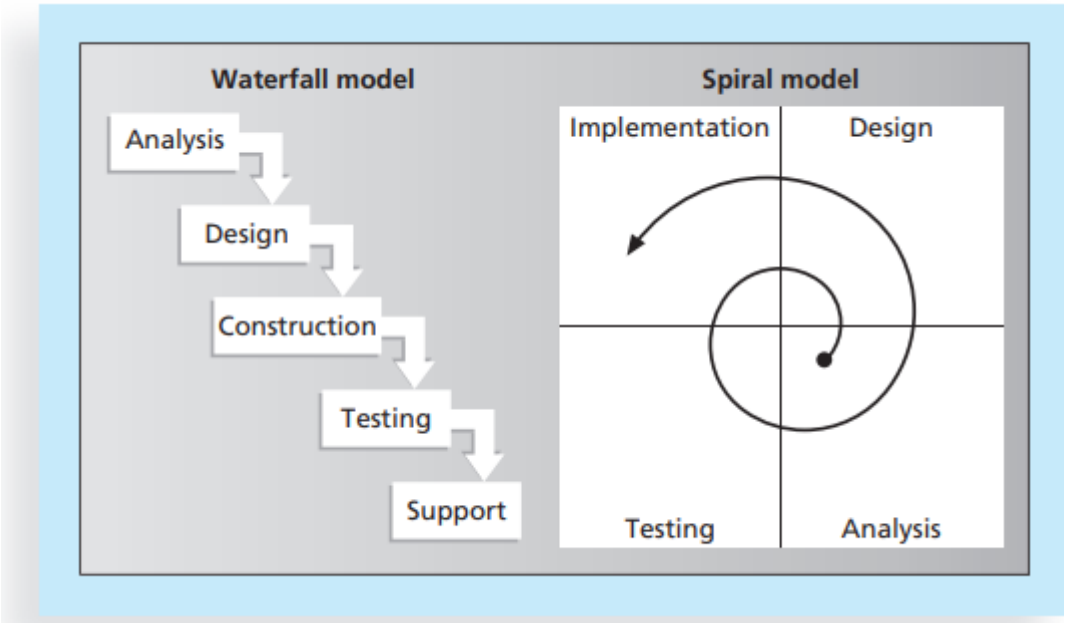


Рисунок 2.1 – Спіральна та водоспадна моделі[1]

Недостатками цих моделей є те що не можна динамічно змінювати процес розробки завдань, у цьому випадку невдале розподілення ресурсів на початку проекту може погано проявитися протягом усього проекту. Тому у якості наступної методології можна розглянути Agile Scrum.

Scrum — це легкий фреймворк, який допомагає людям, командам і організаціям генерувати цінність за допомогою адаптивних рішень для складних проблем. У двох словах, Scrum потребує Scrum Master для створення середовища, де:[7]

1 власник продукту замовляє роботу над складною проблемою в Backlog продукту;

2 команда Scrum перетворює відібрану роботу на приріст вартості під час спринту;

3 команда Scrum та її зацікавлені сторони перевіряють результати та коригують для наступного спринту;

4 повторіть.

Scrum простий і побудований на основі колективного розуму людей, які його використовують. Замість того, щоб надавати людям детальні інструкції, правила Scrum керують їхніми відносинами та взаємодією. У межах цієї структури можна використовувати різні процеси, техніки та методи. Scrum обертається навколо існуючих практик або робить їх непотрібними. Scrum робить видимою відносну ефективність поточного менеджменту, середовища та методів роботи, щоб можна було внести покращення. Артефакти Scrum і прогрес у досягненні узгоджених цілей необхідно перевіряти часто і ретельно, щоб виявити потенційно небажані відхилення або проблеми. Щоб допомогти з перевіркою, Scrum надає каденцію у формі п’яти подій. Перевірка дозволяє адаптуватися. Огляд без адаптації вважається безглуздим. Події Scrum створені, щоб спровокувати зміни. [7]

Головною перевагою цього методу є те що завдання можна завжди

змінювати та перерозподіли, що у випадку незбалансованого командного навантаження на проекті може швидко виявитись та оптимізуватись.

У своїй роботі я розробив веб-додаток, у який будуть заноситись учасники, їх завдання, а потім за допомогою спеціального алгоритму користувач буде отримувати список рекомендованих завдань. Для реалізації я використовував мову програмування Java, Javascript.

Java — об’єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.[9]

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.[10]

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об’єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об’єктно-орієнтованих програм. [9]

Основною перевагою даної мови програмування є те що мова об’єктно-орієнтована, написані програми виходять дуже гнучкі, відмовостійкі та здатні витримати великі навантаження. Також реалізація “прибиральника сміття” [7 с. 53] є головною перевагою у порівнянні з C++, бо це робить процес вивільнення пам’яті автоматичним, що у першу чергу прибере відтік пам’яті та зробить програмний застосунок більш стабільним.

У якості мови для реалізації веб-інтерфейсу можна використати мову програмування Javascript та фреймворк React, який на сьогодні є дуже популярним.

JavaScript спочатку був створений, щоб «оживити веб-сторінки».

Програми цією мовою називаються скриптами. Їх можна записати прямо в HTML веб-сторінки та запускати автоматично під час завантаження сторінки. Сценарії надаються та виконуються як звичайний текст. Для їх запуску не потрібна спеціальна підготовка чи компіляція. Сучасний JavaScript є «безпечною» мовою програмування. Він не надає низькорівневого доступу до пам'яті чи ЦП, оскільки спочатку був створений для браузерів, яким це не потрібно. Можливості JavaScript значною мірою залежать від середовища, у якому він працює. Наприклад, Node.js підтримує функції, які дозволяють JavaScript читати/записувати довільні файли, виконувати мережеві запити тощо. У браузері JavaScript може робити все, що стосується маніпулювання веб-сторінками, взаємодії з користувачем і веб-сервером.[10]

Наприклад, JavaScript у веб-переглядачі може:

1 додавати новий HTML на сторінку, змінювати наявний вміст та стилі;

2 реагувати на дії користувача, запускати на клацання мишкою, рух вказівника, натискання клавіш;

3 надсилати запити по мережі на віддалені сервери, завантажувати та завантажувати файли (так звані технології AJAX і COMET);

4 отримувати та встановлювати файли cookie, ставити запитання відвідувачу, показувати повідомлення;

5 запам’ятовувати дані на стороні клієнта (“локальне сховище”).

React був розроблений із самого початку для поступового впровадження, щоб ви можете використовувати “стільки, скільки вам потрібно”. Незалежно від того, чи хочете ви спробувати React, додати трохи інтерактивності до простої HTML-сторінки чи запустити складну програму на основі React, посилання в цьому розділі допоможуть вам почати роботу. У Реакті використовується підхід за допомогою компонентів. Компонентам часто потрібно змінити те, що відображається на екрані в результаті взаємодії. Введення у форму має оновити поле введення, натискання «Далі» на каруселі зображень має змінити зображення, яке відображається, натискання кнопки поміщає завдання в чергу. Компоненти повинні

запам’ятовувати поточне вхідне значення, поточне зображення, кошик для покупок. У React такий тип пам’яті, що залежить від компонента, називається станом. Ви можете додати стан до компонента за допомогою хука useState. Хуки — це спеціальні функції, які дозволяють вашим компонентам використовувати функції React (стан — одна з цих функцій). Хук useState дозволяє оголосити змінну стану. Він приймає початковий стан і повертає пару значень: поточний стан і функцію встановлення стану, яка дозволяє оновлювати його.

2.2 Алгоритм системи розподілу командного навантаження IT проекту

Розподіл командного навантаження є однією з ключових складових успішного виконання IT-проектів. Вибір правильної моделі розподілу завдань і відповідальностей визначає, як команда буде спільно працювати над проектом.

Розроблена модель була представлена у статті “Модель системи розподілу командного навантаження IT проекту” мого авторства[8]. Ця модель розподілу навантаження може враховувати емоційний аспект під час навантаження проекту, але дана модель може працювати лише в інтерактивній моделі, що на даному етапі мені підходить. Розглянута модель сприяє зниженню конфліктів, покращенню співпраці та підвищенню результативності проекту. Основою є врахування емоційного стану кожного учасника команди та його впливу на продуктивність. Для математичного формалізму можемо використовувати певні показники емоційного фону (наприклад, шкала від 1 до 10) для кожного учасника та обчислювати середнє значення емоційного стану команди. Формула для розрахунку середнього емоційного фону команди може мати вигляд:

$$\text{СЕФ} = \frac{\text{Сума емоційного фону усіх учасників}}{\text{Кількість учасників}} \quad (2.1)$$

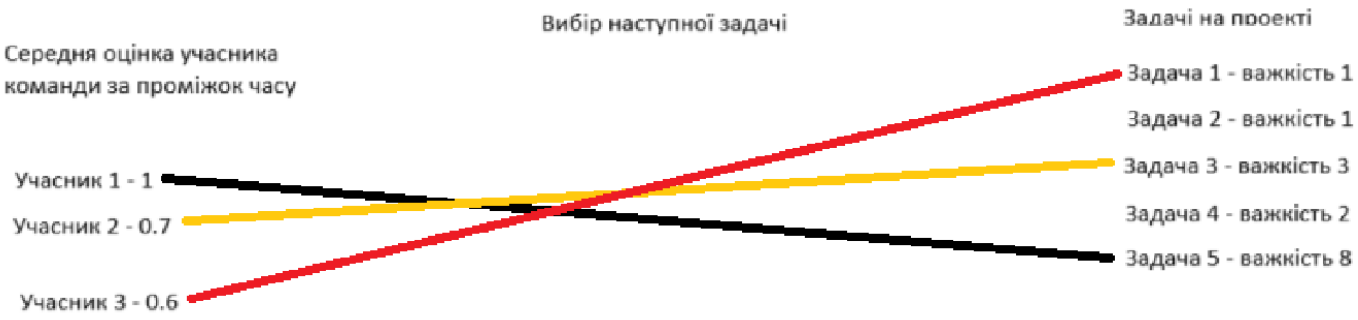


Рисунок 2.2 – Розподіл завдань за психологічним аспектом

$$РП = Базова\ продуктивність \bullet (\frac{СЕФ}{10}) \quad (2.2)$$

Алгоритм розподілу завдань:

Крок 1 збирається емоційний стан розробників. Перед початком робочого дня розробники відправляють оцінку свого емоційного стану від 1 до 10;

Крок 2 збираються дані щодо існуючих завдань. Важливо відмітити що завдання мають містити оцінку нагрузки (одиниці виміру SP). У методології Scrum в якості цих одиниць можна взяти ті сторіпоінти які призначені завданням;

Крок 3 обираються ті розробники які зараз вільні та сортуються за емоційним станом на сьогоднішній день (від найменшого до найбільшого);

Крок 4 сортуються вільні завдання (від найменшого до найбільшого)

Крок 5 для кожного відсортованого розробника призначити у якості вільного завдання номер відсортованого завдання, який йде за номером розробника + завдання яке йде на одиницю більше. Якщо існує кілька завдань з однаковою кількістю навантаження, то береться 2 довільних.

Крок 6 вивести розробникам їх завдання.

2.3 Моделювання інформаційної системи розподілу командного навантаження

Сама система складається з двох частин: ядра та веб-інтерфейсу. Веб інтерфейс приймає дії від користувача у зручному вигляді, перетворює ці дії на http запити та посилає їх на ендпоінти ядра (рисунок 2.3).

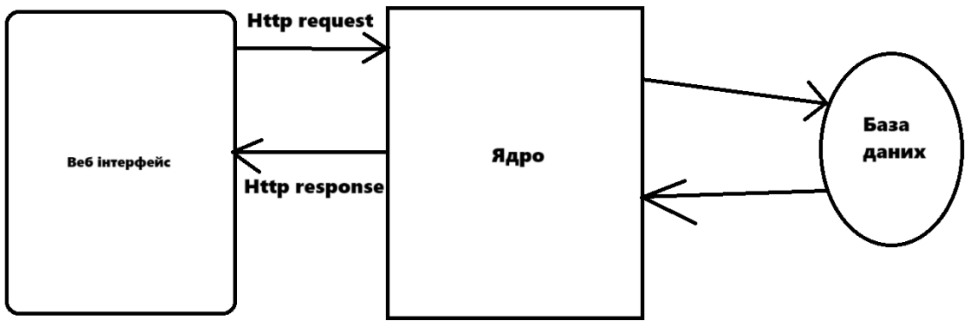


Рисунок 2.3 – Схема інформаційної системи

Нехай нашою системою розподілу навантаження будуть користуватись розробники та менеджери проектів. Розробники будуть раз на добу давати оцінку свого емоційного стану, у свою чергу цей емоційний стан буде йти до бази і потім алгоритми будуть вибирати для розробника завдання. Менеджери проектів будуть періодично дивитись або отримувати статистику щодо командного навантаження. Ядро має приймати запити від веб інтерфейсу, перевіряти їх на коректність та повертати відповідь. Також може бути розроблена інтеграція з відомими додатками для ведення проекту (наприклад Jira).

2.4 Моніторинг та аналіз ефективності розподілу навантаження

В системі можна переглядати ефективність розподілу, кількість учасників, які вони завдання роблять і як швидко справляються. За потреби

можна надіслати звіт на електронну пошту. Також можна дивитись емоційний фон учасника протягом необхідного проміжку часу, якщо необхідно, то менеджер може направити до розробника психолога щоб покращити емоційний стан. Увесь реалізований функціонал дозволить регулярно керувати проектом, переглядати всі завдання, збирати статистику протягом проміжку часу. Якщо у якого-сь розробника часто буде занизький емоційний фон, то можна призначити йому тренінги (рисунок 2.4) або консультації, після яких розробник покращить свій емоційний фон та йому стануть доступні завдання з більшим рівнем складності.

Рисунок 2.4 – Механізм реагування на емоційний стан розробника

3 РОЗРОБКА ПРОТОТИПУ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПОДІЛУ КОМАНДНОГО НАВАНТАЖЕННЯ ІТ ПРОЄКТУ

3.1 Розробка інтерфейсу і функціоналу інформаційної системи розподілу командного навантаження ІТ проекту.

Основний функціонал ядра можна поділити на:

1 аутентифікація та авторизація користувача;

2 отримання оцінок емоційного стану від розробників;

3 видача розробнику наступних завдань;

4 видача менеджеру статистики щодо виконання завдань та поточного стану розробників;

5 видача менеджеру емоційного фону розробників.

Основний функціонал інтерфейсу – це зробити функціонал ядра “user friendly”. Ядро написане на мові програмування Java з використанням Spring Framework. Графічний інтерфейс на Javascript з використанням React.

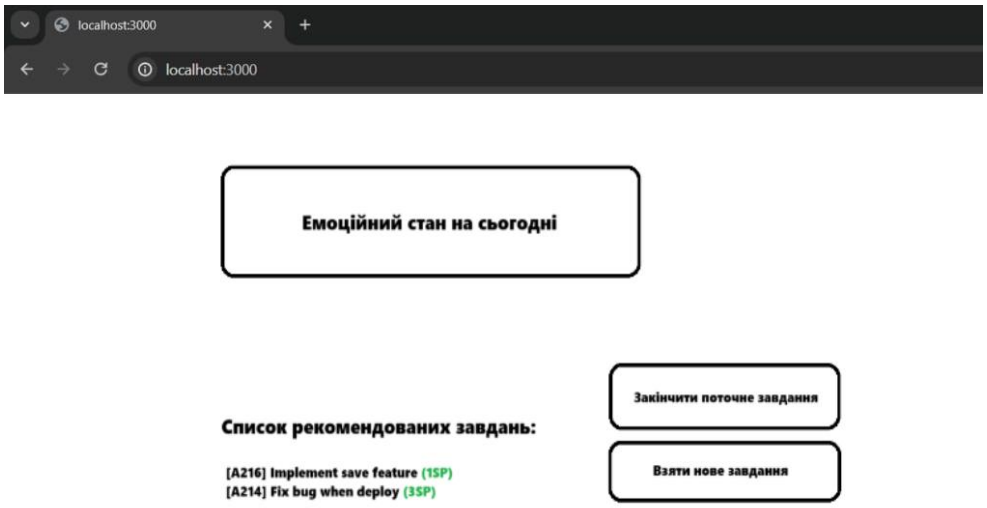


Рисунок 3.1 – Графічний інтерфейс для розробників

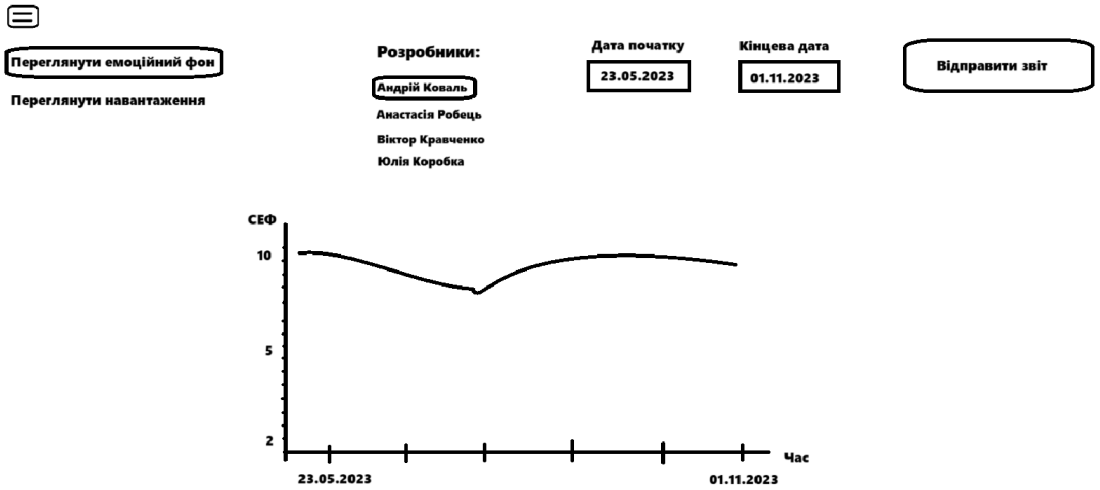


Рисунок 3.2 – Інтерфейс перегляду емоційного фону для менеджера

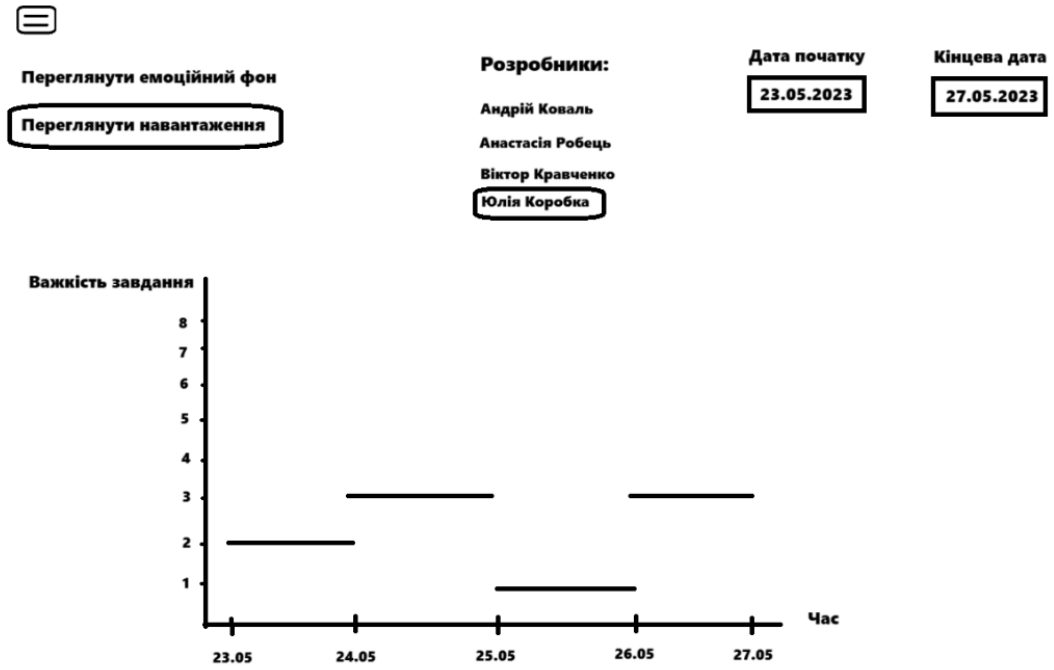


Рисунок 3.3 – Інтерфейс перегляду навантаження розробника

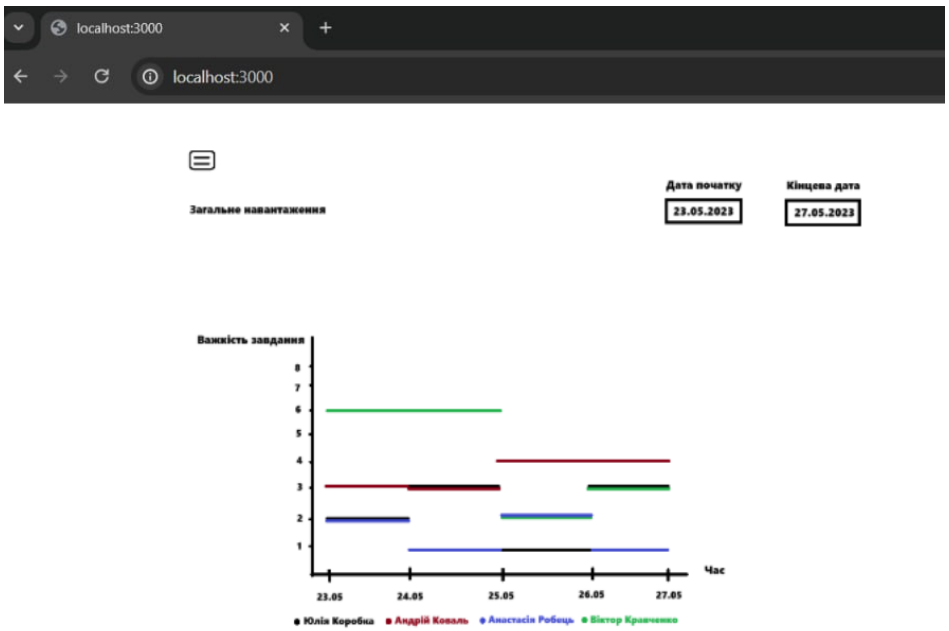


Рисунок 3.4 – Інтерфейс перегляду навантаження на проекті

3.2 Формування баз даних інформаційної системи розподілу командного навантаження IT проекту.

Базу даних можна використовувати будь-яку, але я обрав PostgreSQL, так як вона безкоштовна, швидка і мені подобається.

Сутності бази:

1 користувач;

2 завдання;

3 емоційна оцінка.

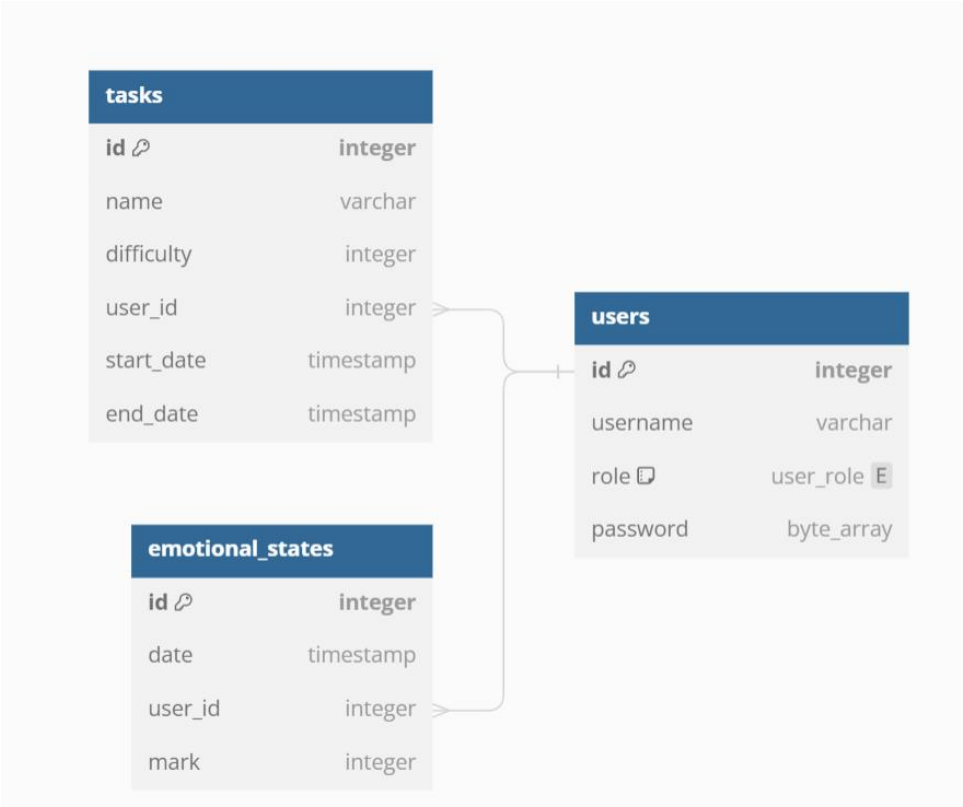


Рисунок 3.5 – Структура бази даних

Користувачі мають айді, ім’я, пароль та роль, роль у свою чергу представлена у вигляді перерахування, що в свою чергу під капотом створює новий тип даних у базі. Емоційний стан має дату, оцінку та користувача що її дав. Завдання має назву, важкість, айді користувача що його робить, дату початку і дату завершення. Таким чином у програмному додатку можна вираховувати навантаження маючи поточні завдання та емоційний стан користувача. Так як завданнями займаються лише розробники то менеджери потрібні лише для системи безпеки.

4 ОЦІНКА ЕФЕКТИВНОСТІ ТА ПРАКТИЧНІ РЕКОМЕНДАЦІЇ З ВДОСКОНАЛЕННЯ СИСТЕМИ РОЗПОДІЛУ КОМАНДНОГО НАВАНТАЖЕННЯ В ІТ-ПРОЕКТАХ

4.1 Збір даних для апробації та оцінки ефективності інформаційної системи.

Для збору даних використовується система логування, формування звітів та програмне тестування.

Під час програмного тестування перевіряється чи всі компоненти програми добре працюють.

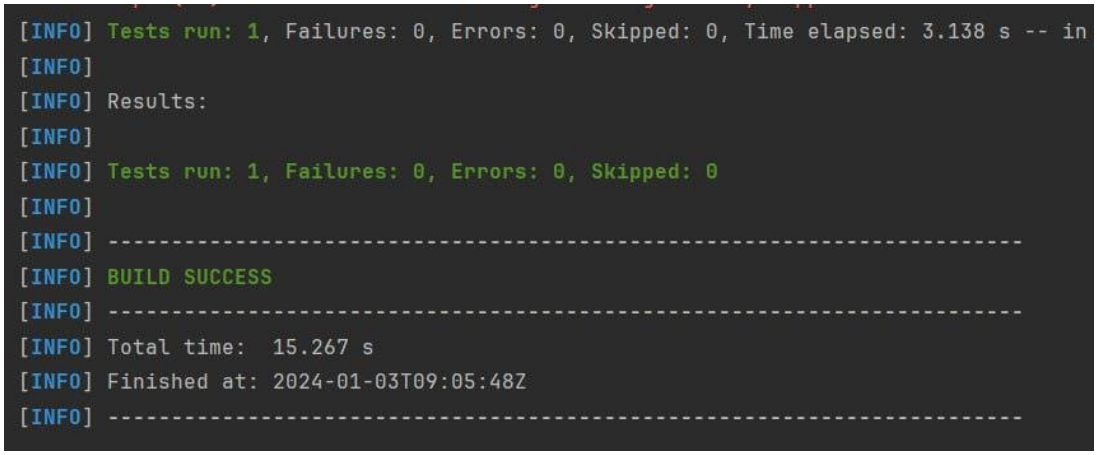


Рисунок 4.1 – Результати проходження програмного тестування

Програмне тестування відбувалось за допомогою тестових фреймворків Junit, Mockito. Таким чином можна визначити ефективність роботи системи.

Результати виконання		23.05.2023 - 27.05.2023	
Працівник:		Юлія Коробка	
Список завдань	Займано часу (в днях)	Вага завдання	
[A314] Implement authentic feature	1	2	
[A318] Fix bug with http client	1	3	
[A302] Fix view on dev env	1	1	
[A320] Add change report feature	1	3	

Рисунок 4.2 – Статистика по розробнику за проміжок часу

На основі статистики щодо виконання завдань розробниками можна порівняти ефективність системи розподілу навантаження. Авжеж існує багато зовнішніх факторів які можуть впливати на продуктивність якогось з розробників, тому ця система у якості покращення продуктивності саме розробників не працює, але якщо при умові що розробник не залежить від зовнішніх факторів та може бездоганно виконувати великий об’єм роботи, то ця система може допомогти вибрати оптимальні для розробника завдання, що в свою чергу може оптимізувати навантаження на проекті.

Щодо покращення системи можна виділити можливість автоматичної синхронізації завдань між проектом який знаходиться десь на інших ресурсах для управління проектом. Тобто якщо проект ведеться наприклад в Jira, то можна за допомогою стороннього API реалізувати підтягування завдань та зміни їх статусу безпосередньо у системі розподілу навантажень без ручного редагування вмісту бази системи. Також є сенс зробити підтримку декількох проектів у одного менеджера.

4.2 Тестування системи розподілу командного навантаження на прикладі IT проекту "Атлас біорізноманіття".

Щоб протестувати систему на реальному проекті треба пройти інструкцію користувача та працювати на проекті. Для перевірки працездатності системи потрібно було знайти команду, яка мала виконувати

проект за моєю методологією. Звернувшись до свого друга, який зараз навчається в Університеті Ренн 1 (Франція), ми домовилися з ним, що я буду курирувати їх команду під час проекту. Щоб не використовувати справжні імена та назви у своїй доповіді, я буду використовувати змінні. Назва проекту – Атлас біорізноманіття.

Команда складалася з 5 людей (О, К, Г, А, С). Завданням проекту було написати програмний додаток для збору і зображення інформації про біорізноманіття в регіонах. Потрібно було створити сервер на основі технологій Java та Spring або будь-якого іншого серверного фреймворку, програмний інтерфейс на основі нативних методів, таких як ReactNative, AndroidNative або Flutter. Після створення програмної частини, команді потрібно було презентувати роботу у вигляді відео презентації з поясненням детальних аспектів роботи кожного члена команди.

Оскільки я почав працювати з командою з самого початку їх проекту, я спостерігав, як працює моя методика протягом всіх стадій розробки. Перш за все команда ознайомилася з інструкцією користувача, після чого всім було створено свій обліковий запис для внесення перших даних свого емоційного стану. Кожен член команди зазначив свій емоційний стан за шкалою від 1 до 10. Після цього команда почала формувати основні завдання та оцінку їх середньої складності:

1 написання веб-інтерфейсу (важкість 33 SP);

2 написання сервера (важкість 13 SP);

3 презентація проекту (важкість 6 SP);

4 збір інформації про деталі розробки проекту (важкість 6 SP).

Дані великі завдання були розподілені між учасниками та згодом дуже важкі завдання з найбільшою кількістю (написання веб-інтерфейсу та сервера) були розбиті на менш важкі завдання.

Далі команда мала можливість сама обрати собі завдання. З самого початку розробки у команді виникли проблеми з вибором архітектури і мов програмування, які будуть використовуватися під час проекту. Студенти О та

К хотіли працювати над веб-інтерфейсом з використанням технології Flutter, оскільки вони мають досвід у цьому напрямі. Студент Г хотів працювати з сервером, використовуючи Java, та наполягав на використанні JavaNative для розробки веб-інтерфейсу щоб зменшити час, необхідний для розробки, через використання спільних класів у сервері та веб-інтерфейсі. Студенти А та С не хотіли займатися проектом через навантаження іншими проектами. Розробка ускладнювалася відсутністю керівництва командою, всі мали однаковий вплив.

Кожен член команди зазначив свій емоційний стан за шкалою від 1 до 10. Студенти О, К, Г визначили свій стан цифрою 10, що дало їм можливість виконувати завдання з високим рівнем складності. Студенти А та С зазначили свій емоційний рівень як 2 та 3, що автоматично призначило їм завдання з низьким рівнем складності.

З самого початку потрібно було вирішити проблеми з тими, хто хотів працювати, щоб робота розпочалася. Для вирішення цієї проблеми було розкладено на такі пункти:

Що пропонує сторона Flutter.

Недоліки цього методу:

1 неможливість використання тих самих класів, що і на сервері;

2 необхідність написання коду з повтором функціональності на іншій мові;

3 незнання принципів роботи мови Dart та фреймворку Flutter студентами Г, А, С.

Переваги цього методу – це досвід роботи з цією технологією у студентів О, К.

Що пропонує сторона Java.

Недоліки цього методу:

1 складність в написанні мобільного додатку, через більш низький рівень мови Java;

2 небажання працювати з цією технологією студентів О та К;

Переваги цього методу:

1 зрозумілість коду для студента Г;

2 повторне використання Java класів з серверу.

Для вирішення конфлікту були взяті до уваги побажання людей, які хочуть працювати. Отже, якщо студенти О та К згодні виконувати свою роботу та писати код з використанням класів з серверу і мають високий емоційний рівень, вони самі повинні вирішувати, яку технологію обрати. Варіант із використанням вже опрацьованої технології для них є найкращим.

Що стосується студента Г та його незнання технологій Flutter та Dart, було укладено домовленості, що він працюватиме лише з частиною серверу, оскільки в нього вже максимально допустиме навантаження. Для взаємодії між сервером і веб-інтерфейсом були заплановані робочі дні для налаштування взаємодії між ними.

Після вирішення недомовленостей між вмотивованими розробниками, настав час поговорити зі студентами А та С, які з особистих причин не могли приділяти увагу важливим аспектам проекту. Для них було автоматично обране завдання, основною частиною якого була підготовка до презентації та відстеження прогресу. Вони мали фільмувати прогрес наприкінці кожного робочого дня та вести облік виконаної роботи кожним членом команди, щоб підготувати детальну презентацію проекту наприкінці роботи. Такий крок дозволив укріпити емоційний стан студентів оскільки вони мали регулярну роботу невеликої складності.

Розподіливши обов’язки, студенти О та К почали малювати інтерфейс майбутнього додатку, а студент Г розпочав створення UML-діаграм класів. Написавши перші прототипи, вони почали обговорювати та виправляти помилки в проекті.

Перші зміни в емоційному стані сталися через після виявлення наступних трьох проблем:

1 для створення повного дерева опису тварин потрібно було б скласти безліч класів, які б описували кожен рівень таксономії;

2 необхідно узгодити всі дані, які сервер буде відправляти клієнту та дані, які клієнт буде відправляти серверу;

3 необхідне використання нових технологій, які студенти О та К не використовували у своєму досвіді.

Для вирішення проблеми 1 було прийнято рішення скоротити дерево таксономії до 4 пунктів, щоб зменшити кількість роботи сервера та веб-інтерфейсу, але щоб їх було достатньо для зображення більшості відомих живих організмів. Цей рішення зменшувало складність роботи, що було гарним кроком для підтримки емоційного стану вмотивованих розробників.

Для вирішення проблеми 2. Оскільки робота була лише на початковому етапі, за основу були взяті принципи CRUD, що, як результат, дало б однакову структуру для веб частини обміну даними та сервера і зменшило б потенційні відмінності у функціональності. Це рішення збільшило кількість роботи, але дало можливість працювати вдома і в особистому темпі кожного розробника.

Для вирішення проблеми 3. Студенти О та К мали конфлікт у виборі нових технологій для виконання завдання, оскільки студент К не хотів витрачати час на дослідження специфічних варіантів та тестування їх роботи. Як результат, було прийнято рішення винести цю частину завдання як окреме та виконувати ту роботу, яка буде для кожного не напруженою. Студент О буде займатися тестуванням та знаходженням нових технологій, студент К буде займатися реалізацією базових віджетів та дизайном. Як результат такої роботи команда працювала в своєму темпі кожного дня проходячи тестування емоційного стану.

Наступна велика зміна емоційного стану відбулася у момент коли потрібно було починати написання взаємодії сервера та веб-інтерфейса. На цьому етапі розробники О, К, Г повинні взаємодіяти між собою та створювати однотипні запити. Але для обрання єдиного стандарту запитів потрібно було надати виконання цього завдання конкретному працівнику. Оскільки студент Г реалізовував базу даних та реалізував інтерфейс доступу

до неї, було прийняте рішення надати керування цим аспектом йому. Цей хід створив чітку залежність та прозорість у виборі типів запитів та їх параметрів, оскільки зараз студенти О та К займалися чітко поставленими задачами та відповідали на запити студента Г. Таке рішення привело то підвищення емоційного стану та показало правильність прийнятих рішень. Після того, як команда завершила проект, ми можемо спостерігати інтерфейс перегляду навантаження на проекті (рисунок 4.3).

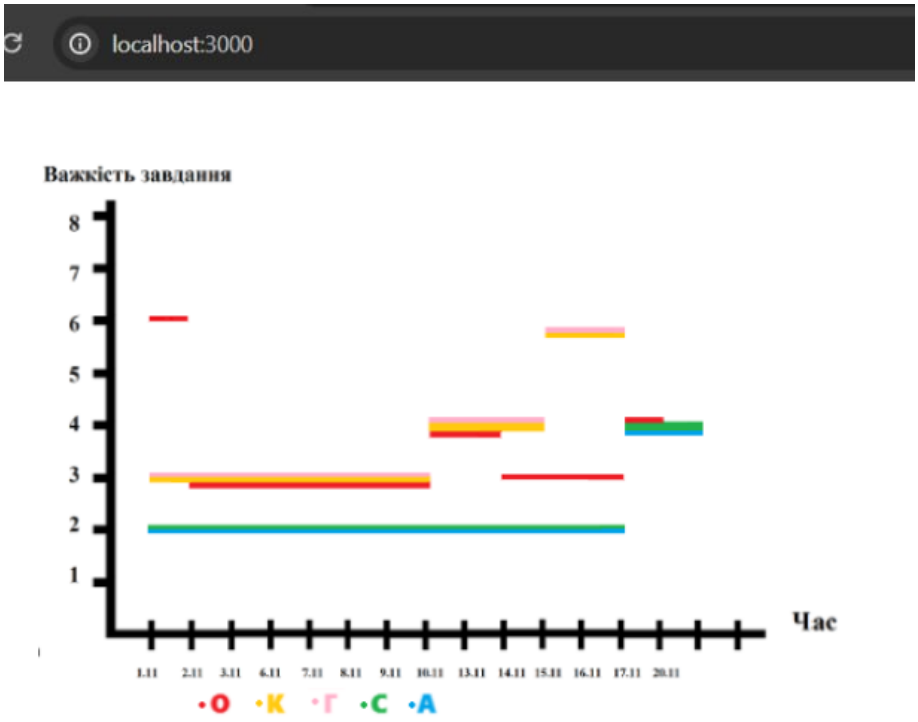


Рисунок 4.3 – Графік навантаження команди під час виконання проекту.

Після проведеної роботи, команда відзначила, що метод розподілу завдань, який я запропонував, дозволяє вирішувати одну з найважливіших проблем командного проекту без чіткого лідера. Оскільки постійний моніторинг психологічного стану дозволяє чітко розуміти присутність проблем у проекті та є сигналом, коли всім потрібно зібратися разом для обговорення проблем та пошуку спільного рішення.

4.3 Розробка керівництва користувачу системи розподілу завдань.

Для запуску системи необхідно мати Docker і docker-compose. Для перевірки можна ввести команди (рисунок 4.4).

```
C:\Users\Entis>docker -v
Docker version 24.0.6, build ed223bc

C:\Users\Entis>docker-compose -v
Docker Compose version v2.23.0-desktop.1
```

Рисунок 4.4 – Перевірка на коректність встановленого Docker

Далі необхідно перейти до директорії проекту, і запустити команду docker-compose up. Якщо усе коректно запуститься, перейдіть за адресою localhost:3000, має з’явитись вікно (рисунок 4.5).

Рисунок 4.5 – Вікно авторизації

Далі уведіть у вікна логін та пароль стандартного менеджера (логін: default пароль: password). Після цього з’явиться вікно для вводу нового логіну та паролю (рисунок 4.6).

Рисунок 4.6 – Вікно для вводу даних менеджера у перший раз

Після вводу даних сторінка перейде на персональний кабінет менеджера, де можна додавати розробників, змінювати їх дані. Також можна додавати завдання з інструментів для управління проектом. Для цього треба натиснути кнопку у кутку і обрати пункт “Зміна завдань”. Після налаштування облікових записів розробників та їх завдань, можна переходити у кабінет розробника та приступати до роботи. Загалом інтерфейс дуже простий та зрозумілий.

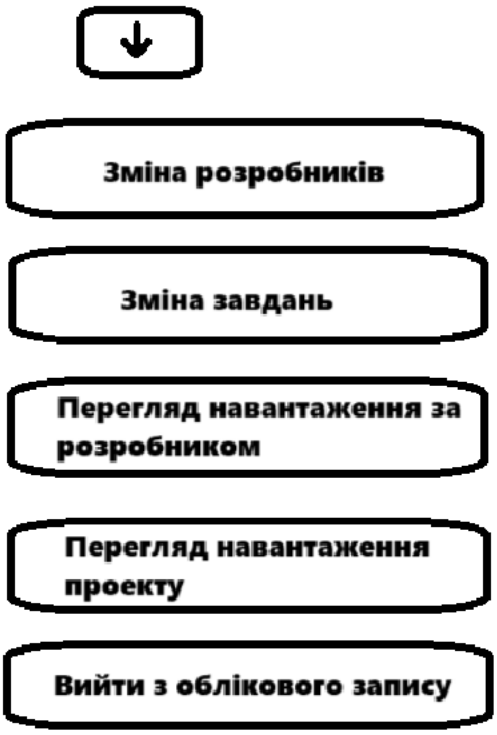


Рисунок 4.7 – Меню менеджера

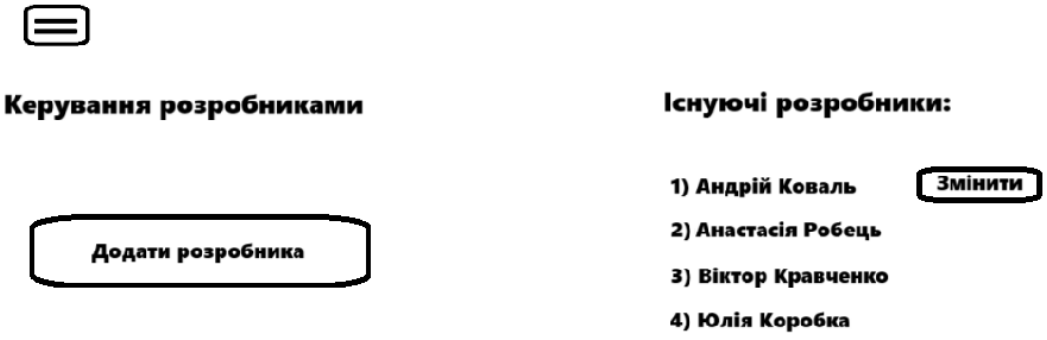


Рисунок 4.8 – Меню керування розробниками

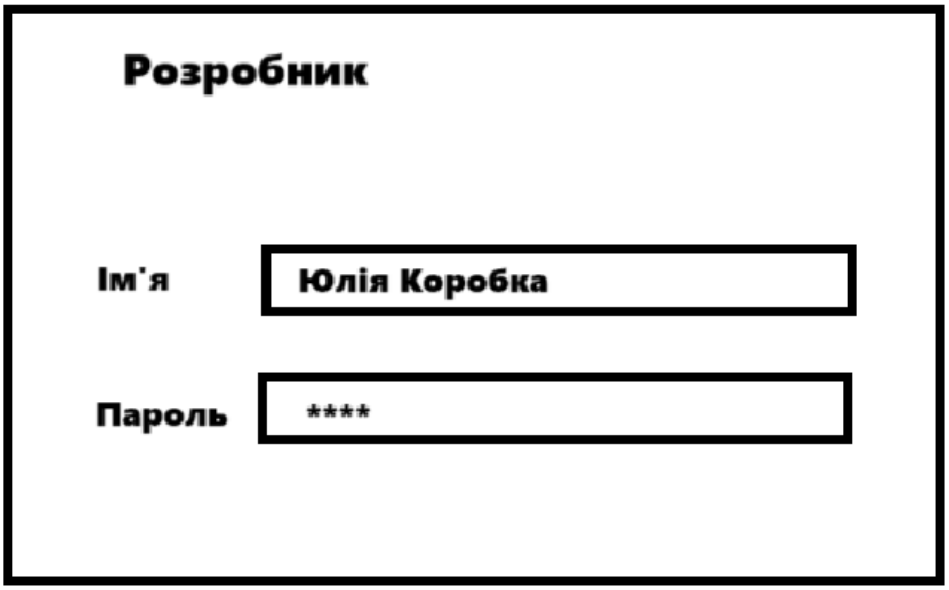


Рисунок 4.9 – Вікно додавання або зміни розробника

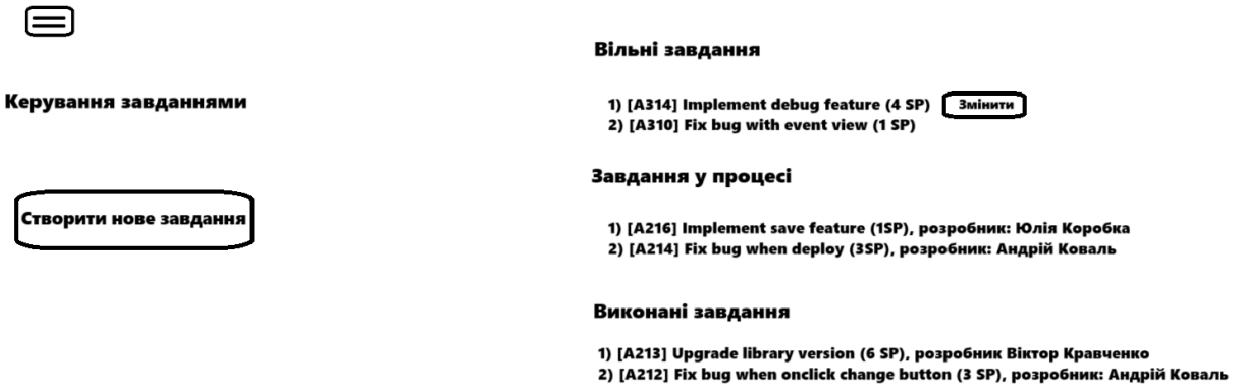


Рисунок 4.10 – Сторінка зміни та перегляду завдань

ВИСНОВКИ

У результаті виконання роботи було розроблено метод розподілу командного навантаження з урахуванням психологічного фактору.

Були розглянуті сучасні методології управління IT проектами як Agile Scrum, Waterfall. Можна зробити висновок, що для реалізації моделі системи розподілення командного навантаження Agile Scrum більш підходить, бо у системі використаний аспект який доречний саме до цієї методології, крім того ця методологія дозволяє часто змінювати завдання, що в свою чергу буде постійно перевіряти та перерозподіляти завдання між розробниками.

Встановлено, що ця модель може використовуватись лише у методологіях, де є гнучкий вибір завдань, адже психологічний фактор в учасників постійно змінюється. Модель розподілу командного навантаження повинна бути індивідуалізованою для кожного проекту, враховуючи його особливості та завдання.

Прототип системи може використовуватись паралельно з головним програмним забезпеченням для ведення проекту, такими як Jira, Trello, Microsoft Project, Project Libre, тому можна зробити висновок, що він є універсальним. Результати аналізу та моніторингу допомагають вчасно виявляти можливі проблеми та здійснювати корекцію. Головною перевагою моделі це те, що вона дозволяє впровадити аспект, який офіційно не впроваджується у всіх гнучких методологій розробки. За результатом роботи додатково було розроблено прототип системи. Він успішно протестований на проекті “Атлас біорізноманіття”, де дуже гнучко адаптував розподіл завдань у команді де немає чіткого лідера. За результатом проект був зроблений успішно, а система сприяла його успішному закінченню. Прототип системи може бути налаштований на швидкі зміни у проекті, що у наш час може часто траплятись, тож замовник може реагувати на нові потреби та пропозиції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Schwalbe, K. Information Technology Project Management. Cengage Learning, 2021. 343 p.

2. Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley, 2017. 1072 p.

3. Pinto, J. K. Project management: Achieving competitive advantage. International Journal of Project Management, 2016, т. 34, p. 447.

4. Saldaña, M. J. G., & López, R. M. C. Project team management and performance in information technology. International Journal of Project Management, 2019, т. 37, с. 511.

5. Schwalbe, K. An Introduction to Project Management. Cengage Learning, 2018. 515 с.

6. Dingsoyr, T., & Moe, N. B. Research challenges in the development and management of large-scale agile software development. Information and Software Technology, 2013, т. 55, с. 315.

7. Schwaber K. The Scrum Guide, Creative Commons, 2020

8. Пальвальов О. Модель системи розподілу навантаження IT проекту. 5th International scientific and practical conference. MDPC. 2024. Pp. 202-206. // URL: https://sci-conf.com.ua/v-mizhnarodna-naukovo-praktichna-konferentsiya-current-challenges-of-science-and-education-15-17-01-2024-berlin-nimechchina-arhiv/. (дата звернення 21.01.2024)

9. Oracle about Java // URL: https://www.java.com/en/download/help/whatis_java.html (дата звернення 20.12.2023)

10. Еккель Б. Філософія Java Пресс. 1998. № 1. С.16–33.

11. Мови програмування Java // URL: http://y66819tz.beget.tech/java/ (дата звернення 20.12.2023).

12. An introduction to Javascript // URL: https://javascript.info/intro (дата звернення 21.12.2023)