Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення системи визначення та реалізації пріоритетних завдань»

Виконав: студент IV курсу
групи 1ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Галієнко А. А.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Коваленко О. О.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Крупельницький Л.В.
(прізвище та ініціали)

опущено до захисту
ав. кафедри _____
_10_» __червня_____ 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«12» травня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Галієнку Антону Арсеновичу

1. Тема роботи – «Розробка програмного забезпечення системи визначення та реалізації пріоритетних завдань».

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри

3, затверджені наказом вищого навчального закладу від 11 березня 2024 року

80.

2. Термін подання студентом роботи: 3 червня 2024 року.

3. Вихідні дані до роботи: середовище розробки – Visual Studio, мова

ограмування JavaScript; браузери – Chrome, Fire Fox Mozilla

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану

рументарію вибору пріоритетів, модулів застосунків таймменеджменту,

вління проєктами; постановка задач дослідження; розробка методу,

лі та алгоритмів роботи системи; розробка програмних модулів програми;

вання програми; висновки; список використаних джерел; додатки;

ративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий

ик бакалаврської дипломної роботи, тема; мета, об'єкт та предмет

дослідження; задачі дослідження; новизна; практична цінність одержаних результатів; результати розробки; апробація та публікації результатів роботи; фінальний слайд.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	Коваленко О. О., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз розвитку інструментів прихованого управління	14.03.2024 – 31.03.2024	вик
2	Розробка методу, моделі та алгоритмів роботи вебзастосунку	01.04.2024– 15.04.2024	вик
3	Розробка програмних модулів	16.04.2024 – 18.05.2024	вик
4	Тестування програми	19.05.2024 – 24.05.2024	вик
5	Оформлення матеріалів до захисту КДР	25.05.2024 – 30.05.2024	вик

Студент (підпис) Галієнко
(прізвище та і

Керівник бакалаврської дипломної роботи Коваленк

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 54 сторінrb формату А4, на яких є 31 рисунок, 5 таблиць, список використаних джерел містить 25 найменувань.

У бакалаврській дипломній роботі було розроблено програмні модулі для системи визначення пріоритетів.

Розроблені програмні модулі дозволяють визначати пріоритети як для загального життя, так і для робочих проєктів.

Використання застосунку сприяє розумінню ефективності використання часу, можливості участі в багатьох проєктах, оптимізації робочого часу.

Застосунок реалізовано з використанням мови програмування JavaScript.

У результаті виконання бакалаврської дипломної роботи було створено вебзастосунок для визначення пріоритетів в різних сферах життя.

Отримані в бакалаврській дипломній роботі результати можна використати для роботи з персоналом, а також як модуль у застосунках самоменеджменту й таймменеджменту.

Ключові слова: самоменеджмент, таймменеджмент, програмний застосунок, визначення пріоритетів, пріоритети життя, пріоритети робочих проєктів.

ABSTRACT

The bachelor’s thesis consists of 54 A4 pages, which have 31 figures, 5 tables, the list of sources used contains 25 items.

In the bachelor thesis, software modules for the priority setting system were developed.

The developed software modules allow you to determine priorities for both general life and work projects.

The use of the application contributes to the understanding of the efficiency of time use, the possibility of participation in many projects, and the optimization of working time.

The application is implemented using the JavaScript programming language.

As a result of completing the bachelor thesis, a web application was created for determining priorities in various spheres of life.

The results obtained in the bachelor thesis can be used for work with personnel, as well as modules in self-management and time management applications.

Keywords: self-management, time management, software application, prioritization, life priorities, work project priorities.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Серед різноманітних методів, концепцій та практик управління, пріоритетне керування подіями, часом, ресурсами, проєктами є актуальним і для прийняття окремих рішень, і для використання в таких сферах управління як менеджмент персоналу, таймменеджмент, самоменеджмент, проєктний менеджмент тощо. Пріоритети можуть бути визначені в стратегії, при виборі проєкту, для збалансування різних сфер життя [1].

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є розробка програмних модулів вибору пріоритетів, який дозволить підвищити ефективність управління часом, завданнями, вибору стратегії розвитку.

Відповідно до визначеної мети, сформовані основні завдання дослідження.

Завдання дослідження:

– аналіз підходів вибору пріоритетів;

– визначення сфер менеджменту для вибору пріоритетів;

– визначення функціоналу програмного модулю для визначення пріоритетів;

– розробити метод і модель роботи системи вибору пріоритетів для збалансування сфер життя та узгодження особистих та робочих завдань;

– розробити метод і модель відстеження життєвого циклу створення програмного продукту та узгодження пріоритетів за таким циклом;

– виконати програмну реалізацію модулів програмного забезпечення;

– провести тестування розроблених модулів.

Об'єкт дослідження – процес розробки програмного модуля вибору пріоритетних завдань.

Предмет дослідження – методи та програмні засоби вибору пріоритетних завдань.

Методи дослідження. У дослідженні було використано методи теорії множин; методу ієрархій; аналізу, синтезу, алгоритмізації; програмування, тестування.

Новизна отриманих результатів:

Подальшого розвитку набув метод інтеграції програмних модулів вибору пріоритетів серед сфер особистого життя та завдань в робочому проєкті, що, на відміну від існуючих полягає у формуванні повідомлень при виборі робочих пріоритетів відповідно до визначених пріоритетів сфер життя, що дозволяє збалансувати інформацію двох програмних модулів між собою.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що розроблений вебзастосунок може бути використаний в системах таймменеджменту, самоменеджменту, управлінння проєктами, управління персоналом.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У друкованій роботі [2], опублікованій у співавторстві, автору належать такі результати: розробка алгоритму роботи застосунку.

Апробація матеріалів бакалаврської дипломної роботи. Матеріали бакалаврської дипломної роботи доповідалися на конференції:

Публікації. Результати дослідження були опубліковані в матеріалах конференції [2].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 25 найменувань, 5 додатків. Робота містить 34 ілюстрації та 3 таблиці.

1 АНАЛІЗ РОЗВИТКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВИБОРУ ПРІОРИТЕТНИХ ЗАВДАНЬ

1.1 Предметна область використання програмних модулів вибору пріоритетних завдань

Пріоритизація завдань є складовою системи менеджменту та повинна включатись в основні функції управління, а саме – планування, організація, мотивація та контроль. Відсутність хоча б одної з визначених функцій веде до неефективного менеджменту та незбалансованих сфер діяльності та життя. Крім того, важливо визначити стратегічні цілі хоча б на рік. Не дивлячись на форсмажорні обставини в Україні та кризові моменти в житті, стратегічні цілі є основою для пріоритизації.

Одним з видів пріоритизації є таймінг – визначення термінових та нетермінових завдань. Але важливість завдань має більш широкий спектр критеріїв, які залежать від особливостей проєкту, мотивації учасників, технологічних можливостей тощо.

Конкуренція за пріоритетами може бути визначена за такими факторами:

Час;

Важливість;

Відповідність стратегії;

Підтримка репутації;

Цілісність технологічних та управлінських процесів.

Застосунок пріоритизації завдань повинен бути інтегрований з календарем виконавців і це дозволить уникнути протиріч в зайнятості учасників команди. Крім того, необхідно мати комунікації по виконанню завдань [3].

Для ІТ-фахівців та ІТ-компаній частіше всього пріоритет завдань визначається при запровадженні гнучкої методології створення програмних

продуктів, а також використання методів управління проєктами. В цих методологіях активно використовуються спеціальні інструменти для комунікацій, шкали витрат, голосування щодо витрат на виконання завдання тощо. На рис. 1.1 представлено вікно комунікацій щодо пріоритетності завдань.

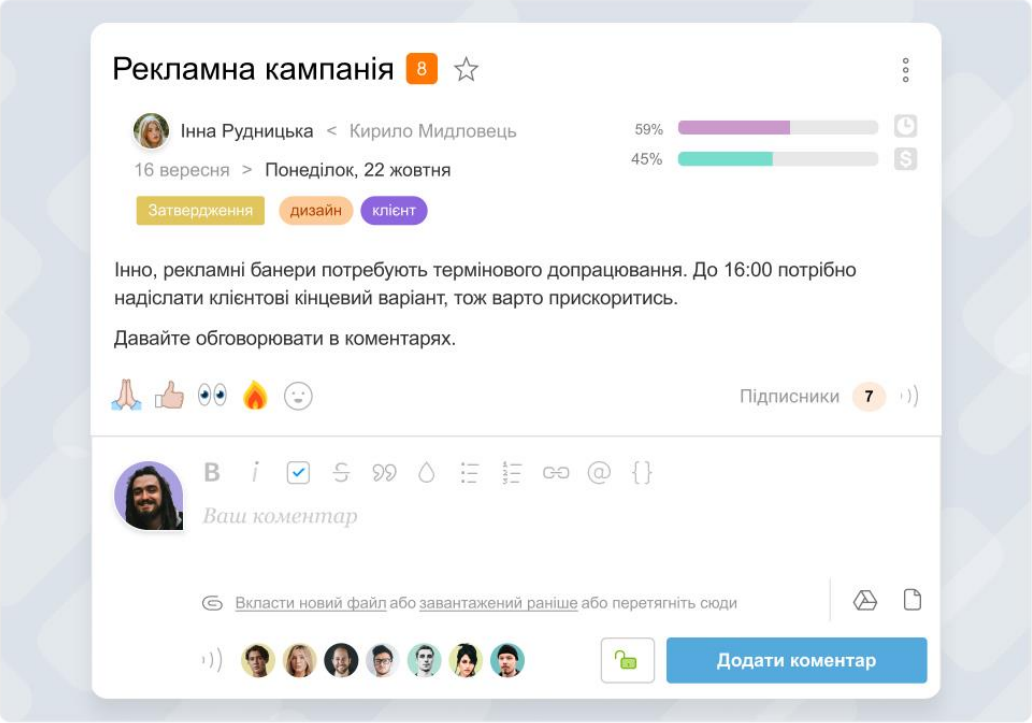


Рисунок 1.1 – Вікно комунікацій для визначення пріоритетності завдань

Також доцільно використовувати шаблони та типові завдання, минулий досвід аналогічних завдань. Все це дозволяє формувати оптимальний розклад (Рисунок 1.2).

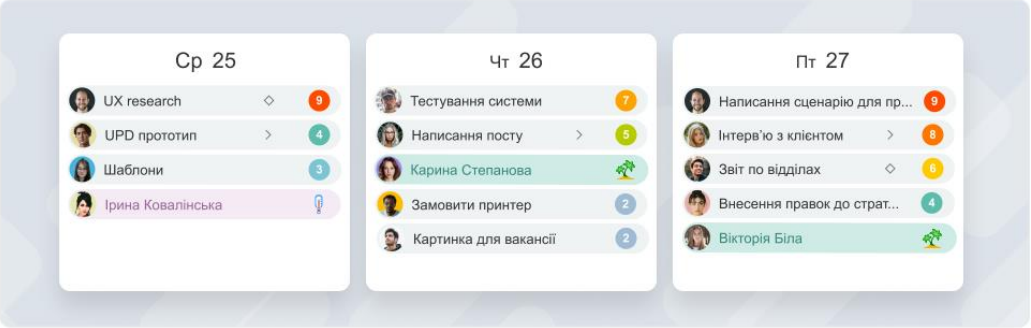


Рисунок 1.2 – Розклад команди

Серед відомих інформаційних систем управління можна виділити системи тайммменджменту, управління проєктом, управління підприємством. Всі ці інформаційні системи містять модулі пріоритизації, комунікацій для обговорення. Але головне обмеження їх в тому, що вони не пов'язані з системою особистих пріоритетів фахівця. розбити великі задачі на кілька менших. Такі застосунки психологічно зручні при умові позначання кожного виконаного завдання і формування прогресу.

Методи пріоритизації задач базуються на різних сценаріях визначення пріоритетів. Як правило, використовують методики, які були розроблені психологами, менеджерами, IT-фахівцями.

Всі задачі поділяють на групи. Це можуть бути групи за етапами розробки, групи за сферами життя, групи за часом виконання, за рівнем важливості, за видами технологічних процесів тощо. На рисунку 1.3 представлено приклад розподілу задач за групами.

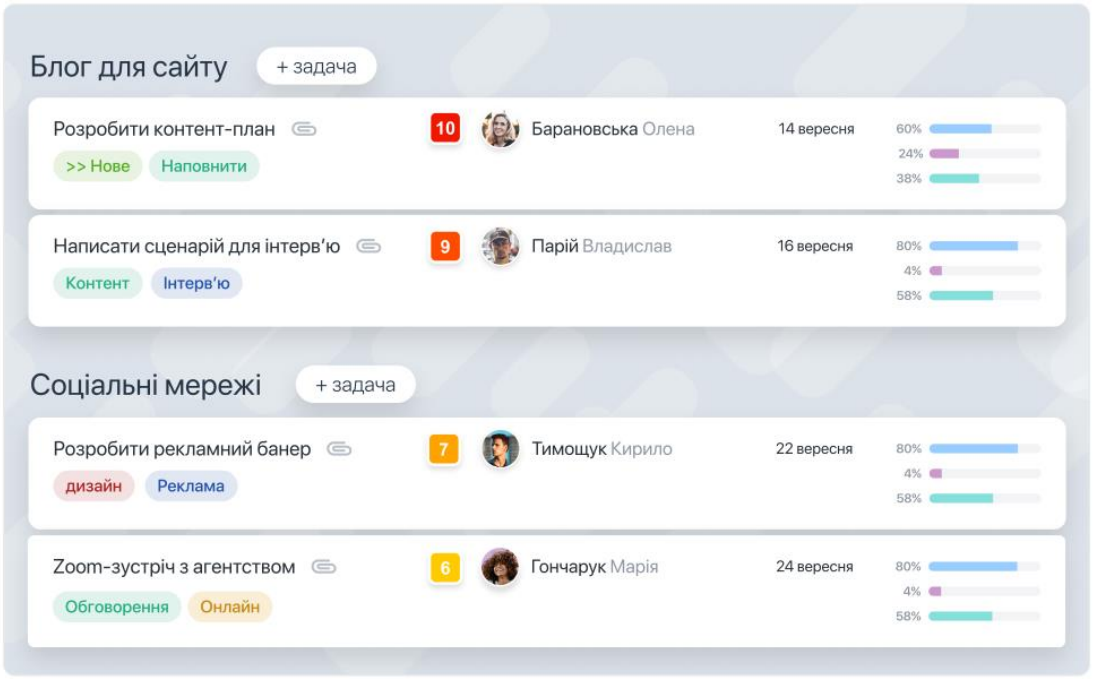


Рисунок 1.3 – Задачі за категоріями

Таке структурування дозволяє візуалізувати завдання та визначити пріоритети на даний момент.

Одним з методів пріоритизації є метод фрагментації. Він аналогічний групуванню категорій за критеріями (ознаками).

Сортування за важливістю завдань доцільно візуалізувати. Така візуалізація здійснюється шляхом використання бульбашкового алгоритму. Важливі задачі розміщуються ліворуч, а менш важливі праворуч.

Якщо виконувати пріоритизацію в межах виконання задач за день, то найкраще використовувати техніку «З'їсти жабу». Вона полягає в тому, що пріоритет мають більш складні задачі. Але саму пріоритизацію визначає користувач [1].

Техніка ABCDE відноситься до комплексної техніки управління пріоритетами та часом. Весь перелік задач розподіляють за часом. В програмних продуктах підключають календар та формують повідомлення щодо виконання завдання за датами плану. Крім самого плану, також можна врахувати наслідки невиконання (наприклад, неможливість розпочати наступну задачу) і за рівнем можливості наслідків формують різні групи, які мають назви визначених літер.

Група A містить задачі критичного шляху проєкту, без яких він не може бути виконаний. Включає задачі, без виконання яких проєкт не буде завершений. Невиконання задач групи A має серйозні наслідки для проєкту.

Група B – це категорія важливих задач, але вони не мають катастрофічних наслідків.

Група C – задачі, які доцільно виконати, але вони не мають критичного значення і можуть бути перенесені за датами..

Група D – необов'язкові задачі, які можна делегувати або виконати після закінчення проєкту.

Група E – задачі, які можна взагалі викреслити з плану або об'єднати з існуючими.

На рисунку 1.4 представлена реалізація такої техніки.

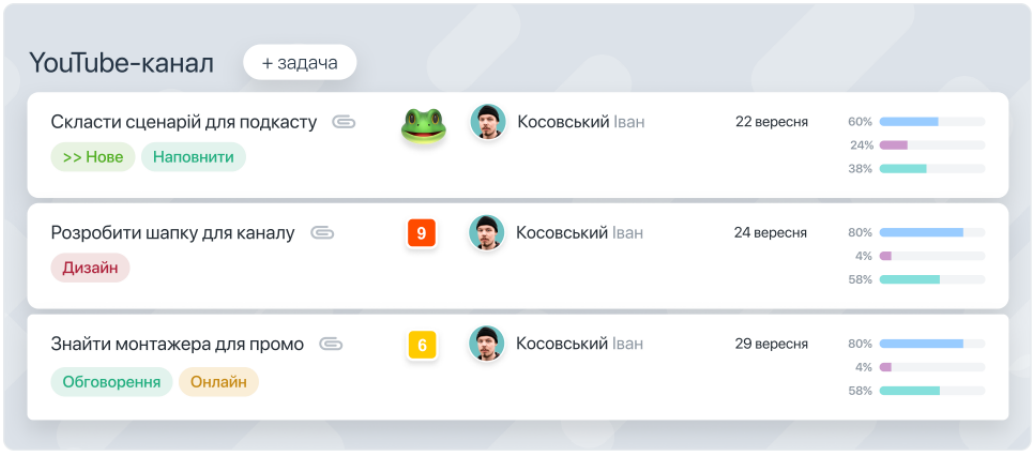


Рисунок 1.4 – Техніка ABCDE

За своїм принципом матриця прийняття рішень Ейзенхауера подібна до техніки ABCDE. За своїм принципом матриця прийняття рішень Ейзенхауера подібна до техніки ABCDE. В цій матриці використовується два критерії важливість та терміновість і групи формуються за комбінацією цих критеріїв (Рис. 1.5).

Група 1 — важливі та термінові задачі

Група 2 — менш важливі, але термінові

Група 3 — важливі, але не термінові

Група 4 — неважливі і нетермінові.

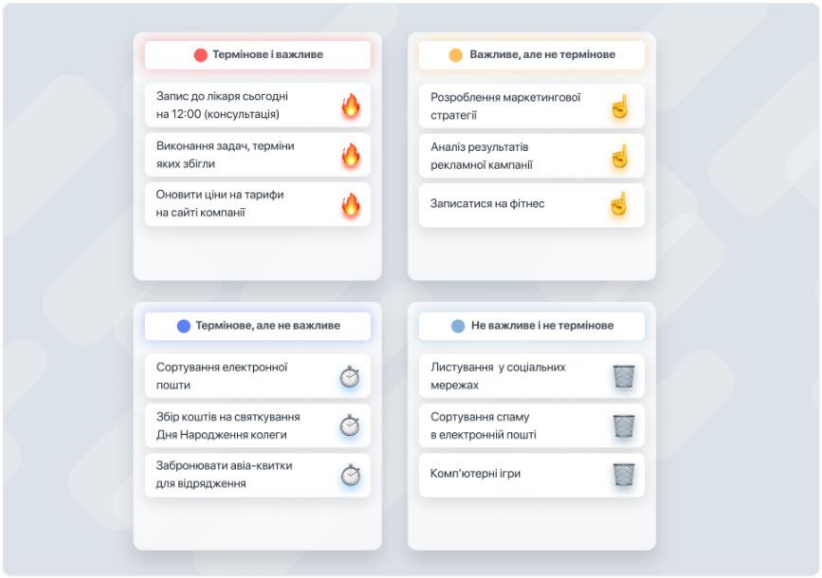


Рисунок 1.5 – Матриця Ейзенхауера

Отже, виконання завдань за критеріями та важливістю виконують поступово від першої до четвертої групи. Четверта група завдань делегується або об’єднується (видаляється).

Розглянуті методи на прикладі системи Worksection можуть бути основою для застосунку пріоритизації завдань.

1.2 Порівняльний аналіз аналогів застосунків з модулями пріоритетів

В попередньому параграфі ми вже розглянули різні методи пріоритизації на прикладі українського таскменеджера – Worksection. Він має потужний функціонал з розвинутою системою візуалізації та спеціальних чатів (Рисунок 1.6) [3].

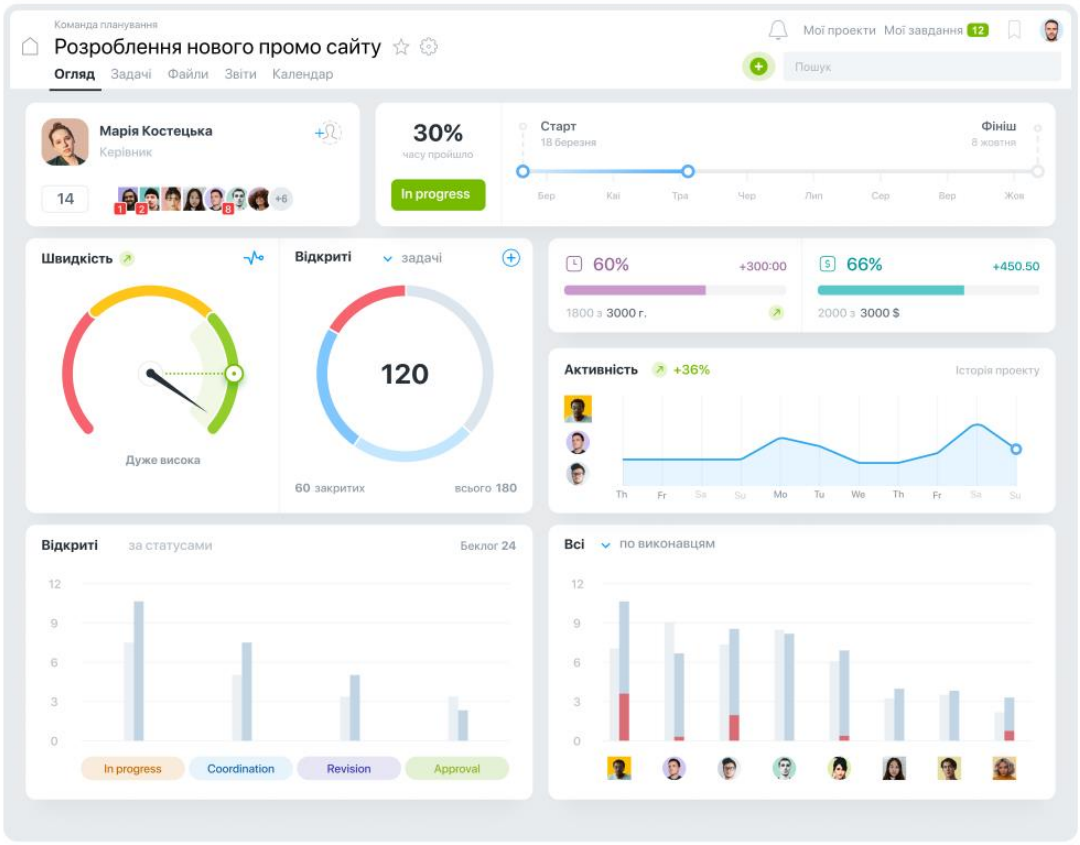


Рисунок 1.6 – Вікно роботи Worksection

Головною перевагою є розвинутий інструментарій, що містить:

спеціальну дошку для завдань;

тайм-трекер;

календар;

спеціальні інструменти управління пріоритетами.

Розглянемо ще аналогічні таскменеджери.

Trello – це візуальний інструмент керування завданнями, який використовує дошки, списки та картки для організації завдань і проєктів. Користувачі можуть створювати дошки для різних проєктів, додавати завдання як картки та переміщувати їх між списками (наприклад, «Справи», «Виконується», «Виконано»), щоб відстежувати прогрес (Рисунок 1.7) [4].

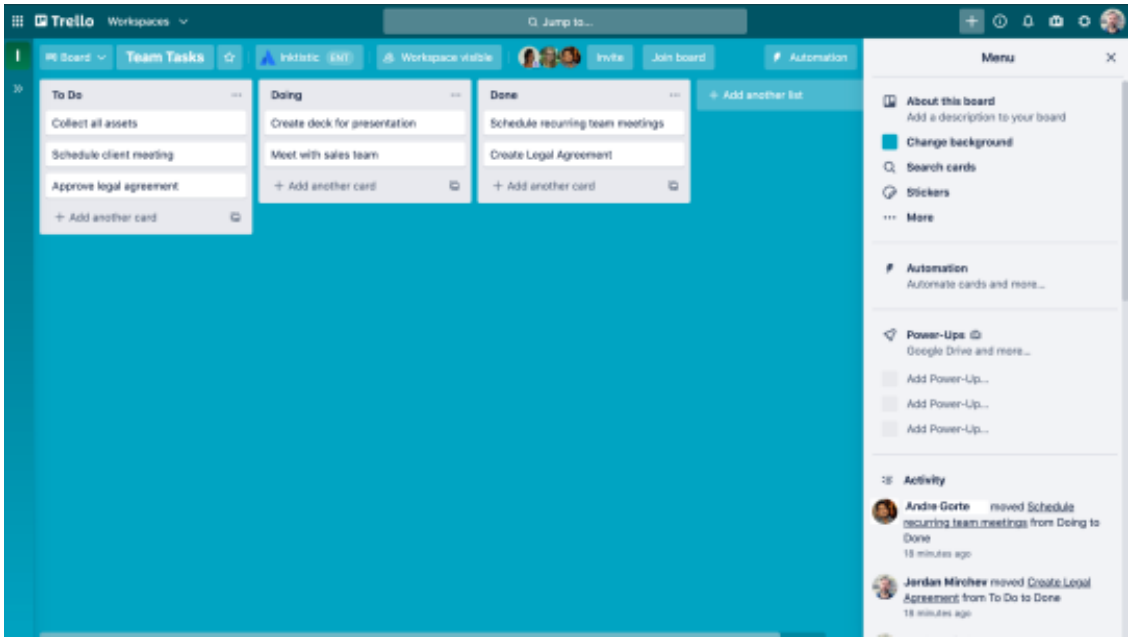


Рисунок 1.7 – Дошка справ ”Trello”

Пріоритети за важливістю та часом визначаються командою на спеціальних зборах та користувач отримує повідомлення щодо дідлайнів та делегування.

Asana – це універсальний менеджер завдань, який дозволяє командам створювати проєкти, призначати завдання, встановлювати терміни виконання та співпрацювати в одному місці. Він пропонує такі функції, як залежності

завдань, підзавдання та часові шкали, щоб допомогти командам залишатися організованими та зосередженими на критичних завданнях (Рисунок 1.8)[5].

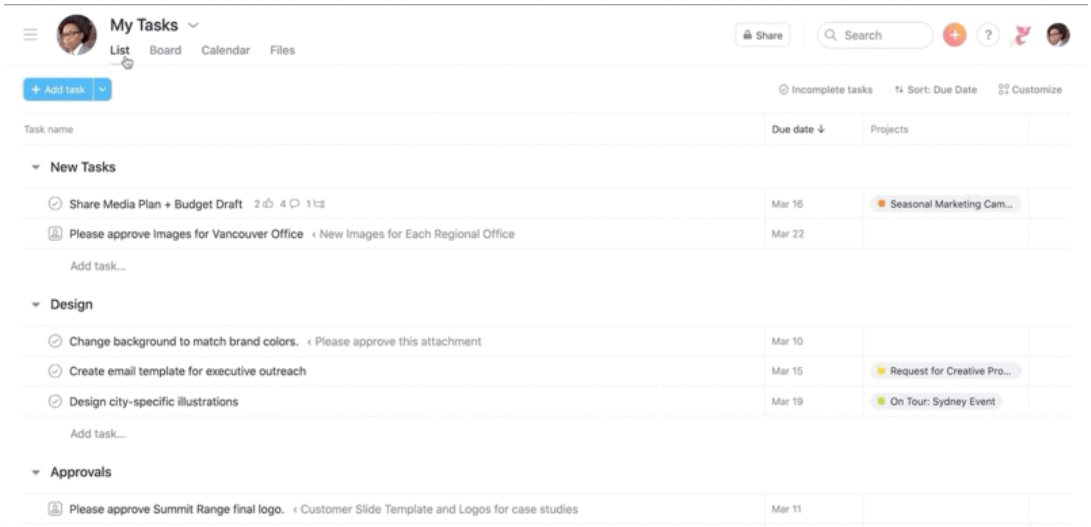


Рисунок 1.8 – Пріоритизація завдань в застосунку Asana

Microsoft To Do – це програма для керування завданнями, яка легко інтегрується з іншими програмами Microsoft Office, такими як Outlook і Teams. Користувачі можуть створювати завдання, встановлювати нагадування та отримувати доступ до своїх завдань на різних пристроях для підвищення продуктивності (рисунок 1.9) [6].

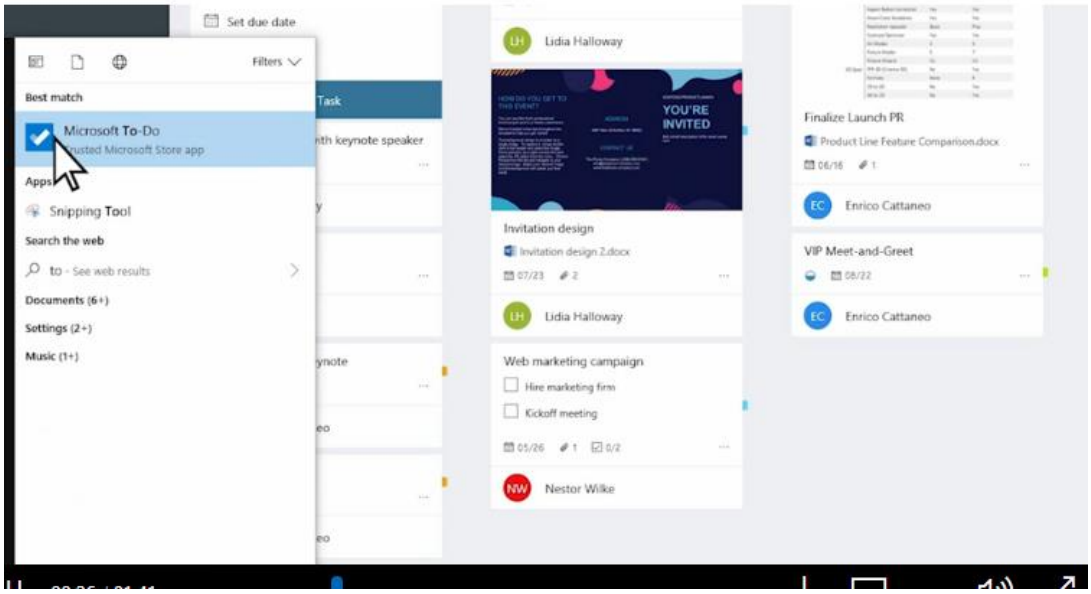


Рисунок 1.9 – Пріоритизація завдань командою в застосунку To Do

В таблиці 1.1 представлено порівняльну характеристику програмних модулів пріоритизації.

Таблиця 1.1 – Порівняльні характеристика програмних модулів пріоритизації задач

Критерії	Asana	Trello	Microsoft To Do	Worksection	Власна розробка
Підтримка пріоритизації всіх сфер життя	0	0	0	0	1
Вибір різних методів (0,1 бал за метод)	0,4	0,2	0,3	0,3	0,3
Функціонал календаря	1	1	1	1	1
Багатокористувацький інтерфейс	0	0	1	1	0
Сумарний коефіцієнт	1	3	4	4	4

Проаналізовані програмні застосунки стосуються пріоритизації задач в проєктах. Але дуже важливо виконати пріоритизацію між завданнями в різних сферах життя, для того щоб воно було збалансованим.

Застосунків, які могли б поєднувати таку пріоритизацію не існує. Задача полягає в тому, що для фахівців, які не розділяють свій день або тиждень життя на робочі та вихідні години і дні, важко пріоритизувати свої задачі та проєкти. Саме відстежування важливості та часового перебігу за сферами життя, може вплинути на рішення під’єднання ще до одного проєкту, зміни місця роботи тощо.

На рисунку 1.10 представлено колесо балансу життя, яке дозволяє визначити рівень балансу за різними сферами для подальшого аналізу [7].

Рисунок 1.11 – Колесо балансу

Але застосунків, які б поєднували пріоритети в різних сферах життя та робочі пріоритети нема. Модель для користувача така – визначити пріоритети різних сфер життя, проаналізувати час, який відповідає вирішенню різних питань і після цього приймати рішення щодо завантаження на роботі, додавання нового проєкту, чи навпаки, зайнятись оздоровленням тощо. Може бути замкнуте коло, коли людина на рівні вигорання, але фактор фінансів дуже важливий, відпустку та оздоровлення не може собі дозволити і тоді перевантажує себе роботою, і в кінці кінців все рівно попадає в лікарню. Якщо для аналізу є дані, людина все ж таки може визначити пріоритети та знайти вихід з різних ситуацій, особливо вигорання. Якщо ж менеджер проєкту дуже зацікавлений у фахівцеві, а фахівець пояснює, що йому потрібно відновлення, то може бути прийнято рішення щодо відпустки на два тижня, оплати

оздоровлення, для того, щоб проєкт отримав фахівця, який зможе дійсно працювати ефективно.

Отже, розробка програмного забезпечення пріоритизації завдань є актуальною, особливо з врахуванням всіх сфер діяльності людини та відповідності визначеним стратегічним цілям.

1.3 Постановка задачі для розробки програмного модулю управління пріоритетами завдань

Програмний модуль управління пріоритетами завдань як основа для подальшого адаптованого модулю пріоритизації в таймменеджменті, проєктному управлінні, для збалансування всіх сфер життя може бути розроблений таким чином:

1. Розробка моделей пріоритизації для різних напрямів життя.
2. Розробка моделей пріоритизації завдань виконавцем.
3. Моніторинг виконання та визначення пріоритетів менеджером проєкту.
4. Вибір технологій для реалізації модулів програмного застосунку.
5. Реалізація модуля.
6. Тестування модуля програмного застосунку.

1.4 Висновки до розділу 1

У першому розділі було розглянуто предметну область використання застосунку управління пріоритетними задачами. Детально розглянуті різні методики пріоритизації та аналоги застосунку або програм, де такий модуль використовується. Виконано постановку задачі.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ ВИЗНАЧЕННЯ ТА ВИБОРУ ПРІОРИТЕТНИХ ЗАВДАНЬ

2.1 Аналіз підходів вибору пріоритетів

Частину підходів було розглянуто в першому розділі бакалаврської роботи. Розглянемо детальніше математичний метод Сааті [8], а також інші методики вибору пріоритетів. Вибір пріоритетів в колесі балансу за рішенням користувача, не завжди буває об’єктивним. За кожною сферою можна визначити фактор оцінювання – наприклад, час на хобі, роботу, родину, важливість пріоритету – теж об’єктивність повинна бути уточнена за допомогою психологічних тестів, рівень здоров’я – чек-лист здоров’я. В нашому застосунку пропонується використовувати вже готові тести для визначення рівня різних сфер життя в колесі балансу. Наприклад, виконуючи вправу, відому як «Колесо життя», людей заохочують подумати про різні аспекти свого життя. Спочатку пропонується структура, що включає вісім сфер життя:

1. Духовність і творчість
2. Особистісне зростання
3. Здоров'я і спорт
4. Сім'я та стосунки
5. Друзі та оточення
6. Кар'єра і бізнес
7. Фінанси 8
8. Яскравість життя.

Рекомендується починати з цієї основи, але адаптувати її, щоб включити ті сфери, які вважаються особисто значущими. Колесу балансу життя бракує універсально правильної версії; її ефективність і доречність відрізняються для людей залежно від їхніх цінностей і пріоритетів. Отже, Колесо життя часто адаптується тренерами та психологами для задоволення

конкретних потреб їхньої аудиторії. Зазвичай спостерігається кілька варіацій балансу колеса життя:

- Базова модель, що зосереджується на трьох основних сферах: здоров’я, стосунки та кар’єра.

- Розширена модель, яка додає фінанси як критично важливий ресурс, що робить його концепцією чотирьох областей.

- Модель із п’яти областей, яка представляє Vibrant Life/Vacation, підкреслюючи важливість часу та його використання, що настійно рекомендується для оптимального балансу.

- Особистісна адаптація, відома як «7 сфер життя», яка включає в себе п'ять згаданих раніше сфер разом з особистим і духовним зростанням, що відзначається своєю високою ефективністю.

Хоча концепція починається з восьми сфер, люди мають право включати більше сфер, як вважають за потрібне. Однак рекомендується бути обережним проти надмірного розширення, оскільки ефективність не обов’язково зростає зі збільшенням кількості ділянок.

Мета полягає в тому, щоб ефективно підвищити якість і темпи особистого розвитку. Наступний крок передбачає оцінку задоволеності в кожній сфері за 10-бальною шкалою, де:

1-3 вказує на критичне незадоволення,

4-7 означає нормальний стан,

8-9 позначає задоволення, і 10 означає повне задоволення в даний момент.

Щоб точно оцінити прогрес і задоволення в кожній сфері, можна використати набір запитань для самонавчання, щоб сприяти глибшому розумінню та ефективній оцінці.

Менеджери з управління персоналом також використовують спеціальні психологічні тести, які дозволяють їм визначити рівень балансу в житті претендента на посаду, зробити правильні висновки.

Задоволеність різними аспектами життя оцінюється за допомогою

певної шкали, де числові значення відповідають рівням задоволеності або занепокоєння в кожній сфері.

Значення в діапазоні від 1 до 3 вказують на критичне незадоволення, що означає стан глибокого нещастя.

Оцінки від 4 до 7 балів говорять про нормальний стан речей, тоді як 8 і 9 означають задоволення.

Ідеальна оцінка 10 означає повне задоволення в даний момент.

Щоб сприяти всебічній самооцінці та сприяти розвитку, пропонується кілька рефлексивних запитань для різних сфер життя.

У сфері духовного зростання запити зосереджені на самосвідомості як духовної сутності, мірі усвідомлення життя, відчутті єдності з оточенням, реалізації творчого потенціалу, вірі в безмежні можливості, відчуття свободи та життєва відповідальність і гармонія у світі.

Для особистого зростання запитання стосуються досягнення цілей, цілеспрямованості, можливостей планування, ефективності дій для досягнення цілей і траєкторії продуктивності протягом життя.

Питання для оцінки здоров'я охоплюють стабільність здоров'я, частоту захворювань і відвідувань лікаря, дотримання принципів здорового харчування, щоденне споживання води, фізичну активність, рівень енергії протягом дня, методи відновлення та частоту, а також достатній сон.

Питання родини та стосунків стосуються самолюбства та самооцінки, якості батьківських стосунків, гармонії з супутником життя, розуміння між батьками та дитиною, дружби, стосунків з колегами/діловими партнерами та зв'язку з вищою свідомістю чи духовним переконання.

У розділі кар'єри питання для самонавчання включають задоволення від роботи, фінансове задоволення від роботи, узгодження з життєвими мріями через роботу, досягнення мети через поточну діяльність, майстерність своєї справи та знання в різних сферах бізнесу, а також оцінка успіху бізнесу.

Запити щодо фінансів стосуються достатності заробітку та достатку, зростання доходу, звички заощаджувати та накопичувати, ефективності

розподілу бюджету, різноманітності джерел доходу, фінансових цілей, як-от стати мільйонером, благодійної діяльності та внесків у світ.

Яскравість життя досліджує присутність жвавості проти одноманітності у повсякденному житті, відчуття життя проти простого існування, відкритість до нових вражень, наявність вільного часу, захоплення хобі, задоволення від дозвілля та частоту подорожі. Відповівши на ці запити, учасники візуально представляють свій рівень задоволеності в кожній сфері життя на діаграмі, яка зазвичай називається Колесо життя, яке представлене в першому розділі. Збалансованість аналізується самим користувачем, менеджером проєкту та психологом (якщо користувач до них звернувся). Менеджери підприємств також формують колесо балансу для прийняття рішення, наскільки на даний момент співробітник готовий до роботи в проєкті.

Серед математичних методів вибору пріоритетів найбільше підходить метод ієрархій Сааті. Він також потребує формулювання та узгодження визначених стратегій, для того щоб вибрати необхідну пріоритизацію. Це може бути використано як для проєкту, так і для організації, окремого фахівця тощо [8].

Метод ієрархій Сааті дозволяє порівняти попарні пріоритети, визначити їх рейтинги, а потім використати отримані результати для визначення стратегій пріоритетів виконання. На рис. 2.1 представлено загальну модель методу на прикладі вибору лідера команди за визначеними критеріями. Плюс методу ієрархій полягає в тому, що він дає можливість визначити окремі характеристики та об’єднати їх комплекс для вибору. Математично такий метод базується на таблиці пріоритетів, яка представлена нижче та за допомогою спеціальних матриць, які дозволяє визначити рейтинг для вибору.

Матриця попарних порівнянь представлена у формулі 2.1.

$$\begin{array}{c} \\ A_1 \\ \dots \\ A_n \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & \dots & A_n \end{array} \\ \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ & & \\ a_{n,1} & \dots & a_{n,n} \end{bmatrix} \end{array} \quad (2.1).$$

$A_1 \dots A_n$ – альтернативи, що порівнюються за критеріями $a_1 \dots a_n$.

Нормоване середнє геометричне по кожному ряду дозволяє сформувати одиничну матрицю вектору для вибору.

В межах розробки веб-застосунку такі розрахунки не були заплановані. Але для визначення складних пріоритетів цей метод може бути використаний як обчислювальний метод. Також можна підключити спеціальні модулі, які розраховують пріоритет за матрицями Сааті онлайн.

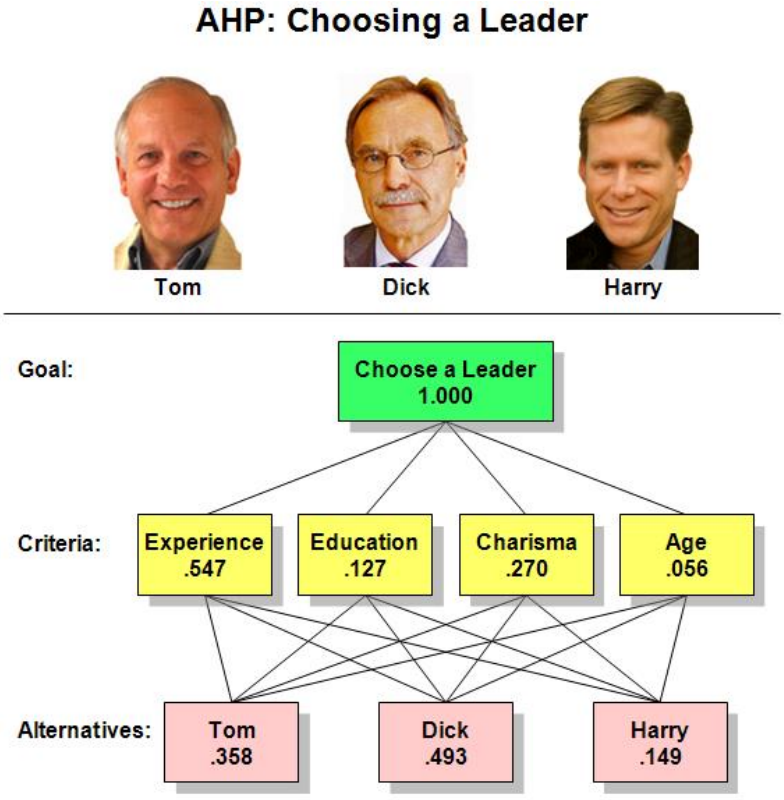


Рисунок 2.1 – Метод Сааті – візуалізація []

В таблиці 2.1 визначені рівні кількісних оцінок пріоритетів. Такий метод дозволяє обчислити пріоритети. Крім того, саму процедуру визначення пріоритетів необхідно виконувати за допомогою спеціальних модулів, а також консультації фахівців за кожним визначеним розділом в колесі балансу.

Таблиця 2.1 – Критерії переваги за методом Сааті

Рівень переваги	Визначення	Пояснення
1	Відсутність переваги	Альтернативи однакові
2	Слабка перевага	Друга не набагато більша за першу
3		
4		
5	Середня перевага	Друга в середньому більша за першу
6		
7		
8	Сильна перевага	Друга набагато більше за першу
9		

Одним з елементів стандарту SWEBOK є використання життєвого циклу створення програмного продукту. В залежності від методологій реалізації життєвого циклу, він має свої особливості.

Класичною лінійною та послідовною методологією є водоспадна методологія. Вона включає такі етапи, як збір вимог, проєктування, впровадження, тестування, розгортання та обслуговування. Модель Waterfall є простою, але жорсткою, що ускладнює внесення змін після того, як процес розпочато.

Гнучка методологія побудована на концепції Agile і може бути реалізована за різними сценаріями [9].

Так, наприклад, за методологією Scrum проєкт розділяють на невеликі частини, які називаються спринтами, причому кожен спринт зосереджується

на певному наборі функцій. Такий підхід заохочує командну співпрацю, зворотній зв’язок із клієнтами та постійне вдосконалення. Scrum є практичною реалізацією Agile і використовується для керування розробкою складного програмного забезпечення та продуктів. Він використовує ітерації фіксованої довжини, які називаються спринтами, зазвичай тривалістю два тижні. Процесом розробки керують такі ролі, як Scrum Master, Product Owner і Team Development [10].

Kanban – це ще одна гнучка методологія, зосереджена на візуалізації всього проєкту на дошках для підвищення робочого процесу та продуктивності. Це дозволяє команді бачити хід виконання всіх завдань і керувати процесом, обмежуючи обсяг незавершеної роботи, щоб уникнути перевантаження членів команди [11].

Життєвий цикл розробки програмного забезпечення (SDLC) включає в себе такі фази [12]:

1. Планування. Цей початковий етап передбачає визначення обсягу, ресурсів, вартості та часових меж проєкту. Це закладає основу для всього процесу розробки.

2. Аналіз. Вимоги збираються та ретельно аналізуються, щоб переконатися, що кінцевий продукт відповідатиме потребам користувача.

3. Проєктування. Цей етап описує архітектуру програмного забезпечення, визначаючи конкретні аспекти, такі як стек технологій, моделі даних і загальний план системи.

4. Впровадження (або кодування). На цьому безпосередньо створюється програмний продукт на основі визначених вимог і проєктної документації.

5. Тестування. Програмне забезпечення ретельно перевіряється на наявність дефектів або помилок. Тестування гарантує, що продукт відповідає оригінальним специфікаціям і не містить помилок.

6. Розгортання. Після тестування та готовності програмного забезпечення його розгортають у робочому середовищі, де кінцеві користувачі можуть почати використовувати продукт.

7. Технічне обслуговування. Після розгортання програмне забезпечення потребуватиме регулярних оновлень, виправлення помилок і, можливо, додавання нових функцій для задоволення нових потреб користувачів. Кожна методологія та етап життєвого циклу відіграють вирішальну роль у розробці програмних продуктів. Вибір правильного підходу залежить від конкретних потреб проєкту, розміру команди та бажаної гнучкості в обробці змін протягом усього процесу розробки.

Відповідно до визначеної задачі визначення пріоритетів, підхід вибору пріоритетів для користувача має такий загальний алгоритм:

1. Постановка загальної задачі вибору.
2. Дані для вибору.
3. Структуризація даних для вибору.
4. Формування критеріїв вибору.
5. Виконання дій користувачем.
6. Результати вибору пріоритетів.
7. Визначення дій в календарі.
8. Адаптація отриманих рішень користувачем.

Програмне забезпечення управління пріоритетами відповідно до життєвого циклу проєкту є одним з модулів для управління пріоритетами створення програмного продукту. Такий модуль передбачає такий функціонал:

1. Визначення методології створення програмного продукту та управління проєктом.
2. Визначення життєвого циклу управління проєктом.
3. Визначення завдань в межах життєвого циклу і таймінгу.
4. Визначення пріоритетів відповідно до таймінгу.

2.2 Розробка моделей системи пріоритетів

Розглянемо дві моделі системи пріоритетів. Перша – для користувача та прийняття рішень щодо різних сфер життя та, зокрема, роботи в проєкті. Друга – для менеджера, користувача та проєкту.

На рисунку 2.2 представлено модель управління пріоритетами для окремого користувача.

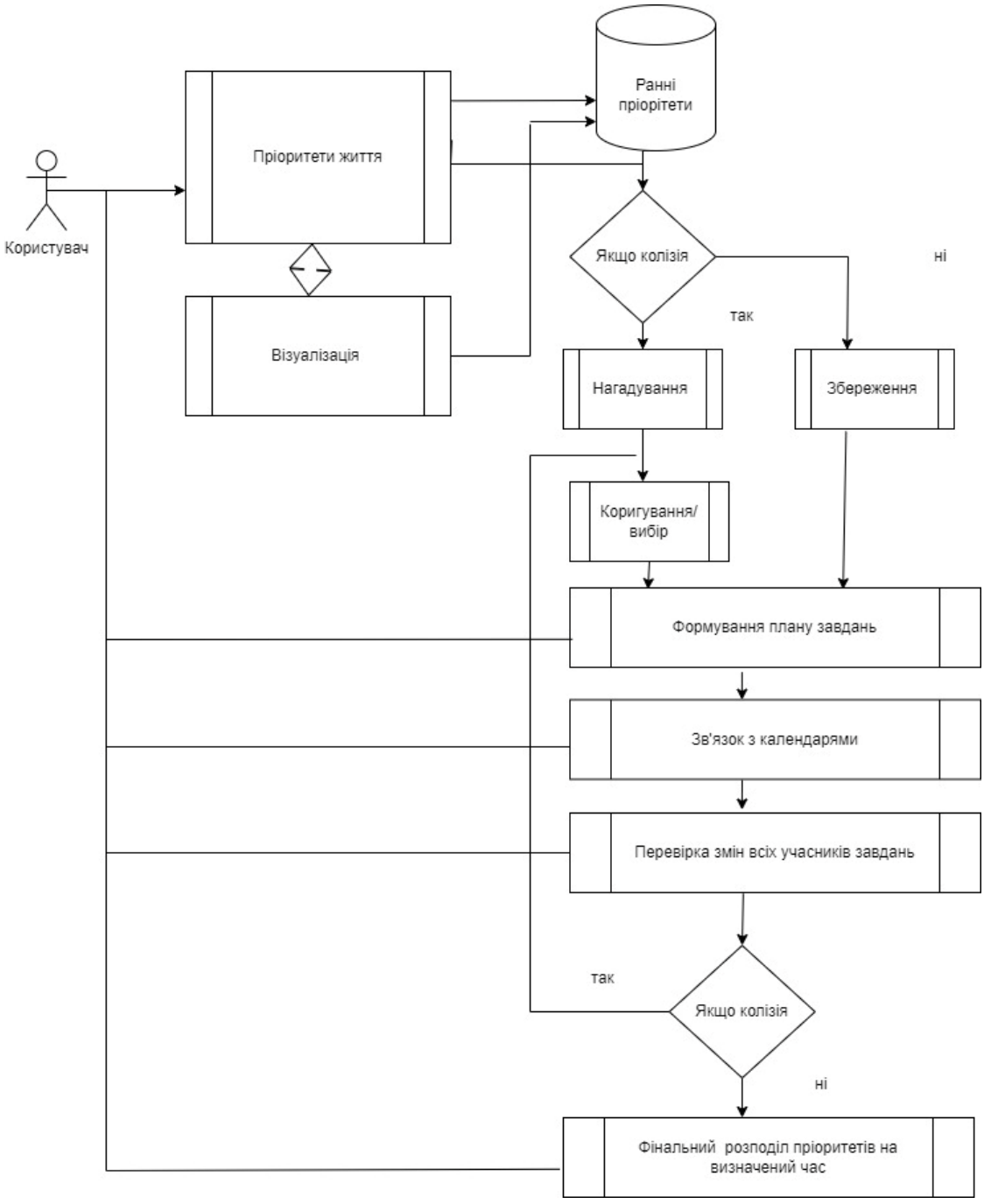


Рисунок 2.2 Модель для визначення власних пріоритетів

Загальна модель передбачає, що користувач визначає пріоритети за сферами життя. Вибір різних сфер життя та їх рейтингування залежить від багатьох факторів. Окремими програмними модулями та відповідно до консультацій фахівців, користувач визначає пріоритети. Якщо визначені вже ранні або більш важливі пріоритети і вони входять в колізію, то користувач повинен змінити пріоритет та внести зміни в календарі, рівні важливості тощо.

Візуалізація та доповнення різних сфер життя, результати балансу для конкретного користувача є основою для вибору пріоритетів. Результатами є календарний та погодинний план завдань, рекомендація для делегування тощо.

Якщо є колізії між сферами життя, фазами проєкту, зайнятістю виконавців, рівнем важливості завдань, то формується спеціальне повідомлення.

Найбільш складним питанням є визначення критеріїв для управління пріоритетами. Такі модулі повинні розроблятись тільки з експертами.

Головним критерієм для користувача в загальному визначенні пріоритетів сфер життя є поставлені цілі, можливості реалізації, наявність ресурсів здоров'я, професійної компетентності, коштів, часу.

Така модель передбачає, що користувач послідовно визначає пріоритети сфер життя, а потім список завдань на день або/і список завдань проєкту. В ідеальному випадку для користувача вибрана модель, коли робочий час визначено окремо. Але дуже часто може використовуватись фрагментарна модель для виконання робочих завдань і це потребує спеціальних повідомлень щодо крайдати (дідлайну) або позначки важливості завдання, очікування інших учасників команди проєкту.

На рис. 2.3 представлено модель управління пріоритетами менеджера з персоналу при виборі або/і запрошенні фахівця на проєкт.

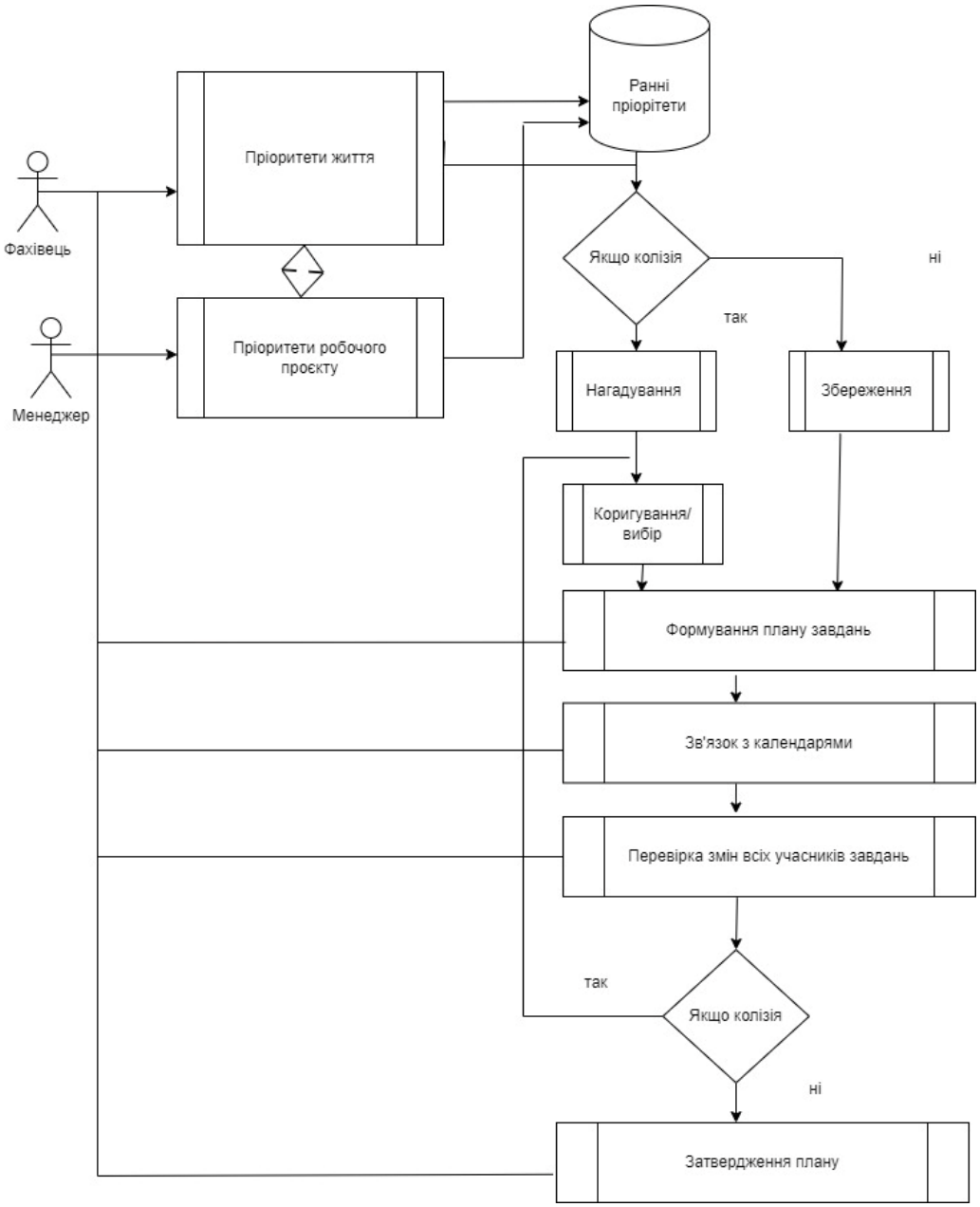


Рисунок 2.3 – Модель узгодження пріоритетів загальні та завдання проєкту

На рис. 2.4 представлена модель управління пріоритетами завдань в процесі управління проєктом.

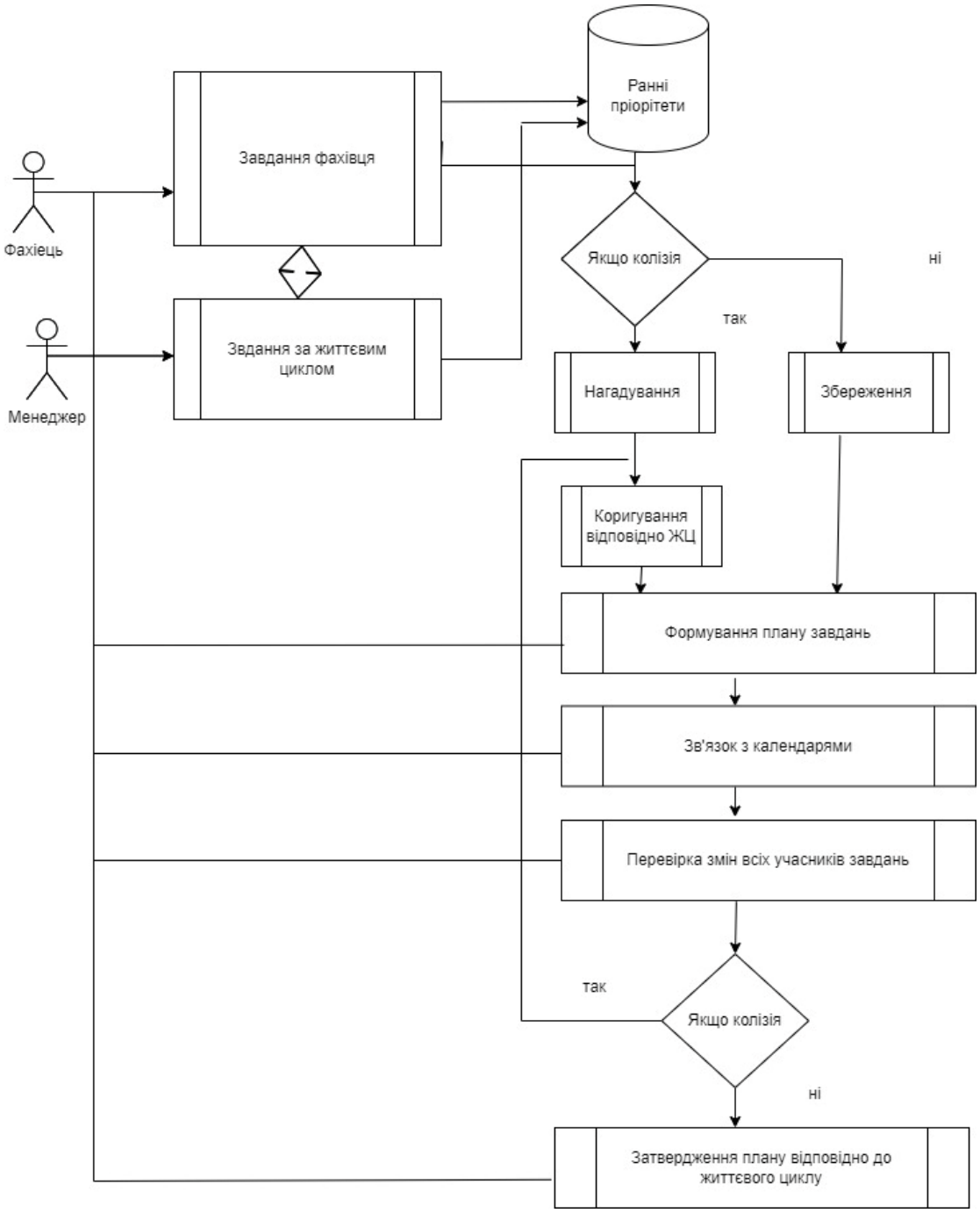


Рисунок 2.4 Завдання за життєвим циклом – пріоритети відповідно до загальних

Визначені моделі можуть бути реалізовані за різними сценаріями відповідно до критеріїв вибору пріоритетів.

Представлені моделі можуть бути використані для реалізації модуля управління пріоритетними завданнями.

2.3 Розробка методу узгодження роботи модулів програмного забезпечення вибору пріоритетів різних сфер життя

Метод узгодження роботи модулів програмного забезпечення вибору пріоритетів різних сфер життя включає в себе основні моделі та загальний алгоритм вибору пріоритетів користувачем. Такий алгоритм дозволяє визначити баланс/дисбаланс у сферах життя, сформувати головні цілі та пов’язати їх з роботою в проєкті або на робочому місці в організації. На рис. 2.5 представлено алгоритм реалізації методу узгодження за сценарієм для користувача.

Алгоритм представлено такими блоками як визначення загальних пріоритетів, узгодження з пріоритетами проєкту. Алгоритм виконано за правилами алгоритмізації [13]

Цикл визначення сфер пріоритетів, що включає в себе додавання та видалення різних сфер, вибір критеріїв – важливість, час. В нашому програмному забезпеченні ми використовували тільки один критерій за часом, а колізію – співпадіння часу. Але критерії можуть бути ускладнені. При умові адаптації системи вибору пріоритетів може бути додано критерії за даними вибору стратегій, характеристик на даний час відповідно до основних сфер життя тощо. Для цього можливо використати обчислення за методом Сааті, вибір за рекомендаціями фахівців. Запропонований алгоритм має підпрограму вибору пріоритетів проєкту. Така підпрограма може бути сформована за різними критеріями відповідно до методології управління проєктом, його життєвого циклу. Якщо вибирають водоспадну методологію, то вона передбачає послідовне виконання всіх етапів життєвого циклу. Якщо ж методологія гнучка, то в залежності від реалізації, вона має свої особливості.

Управління завданнями – Канбан, управління спринтами – Scrum тощо.

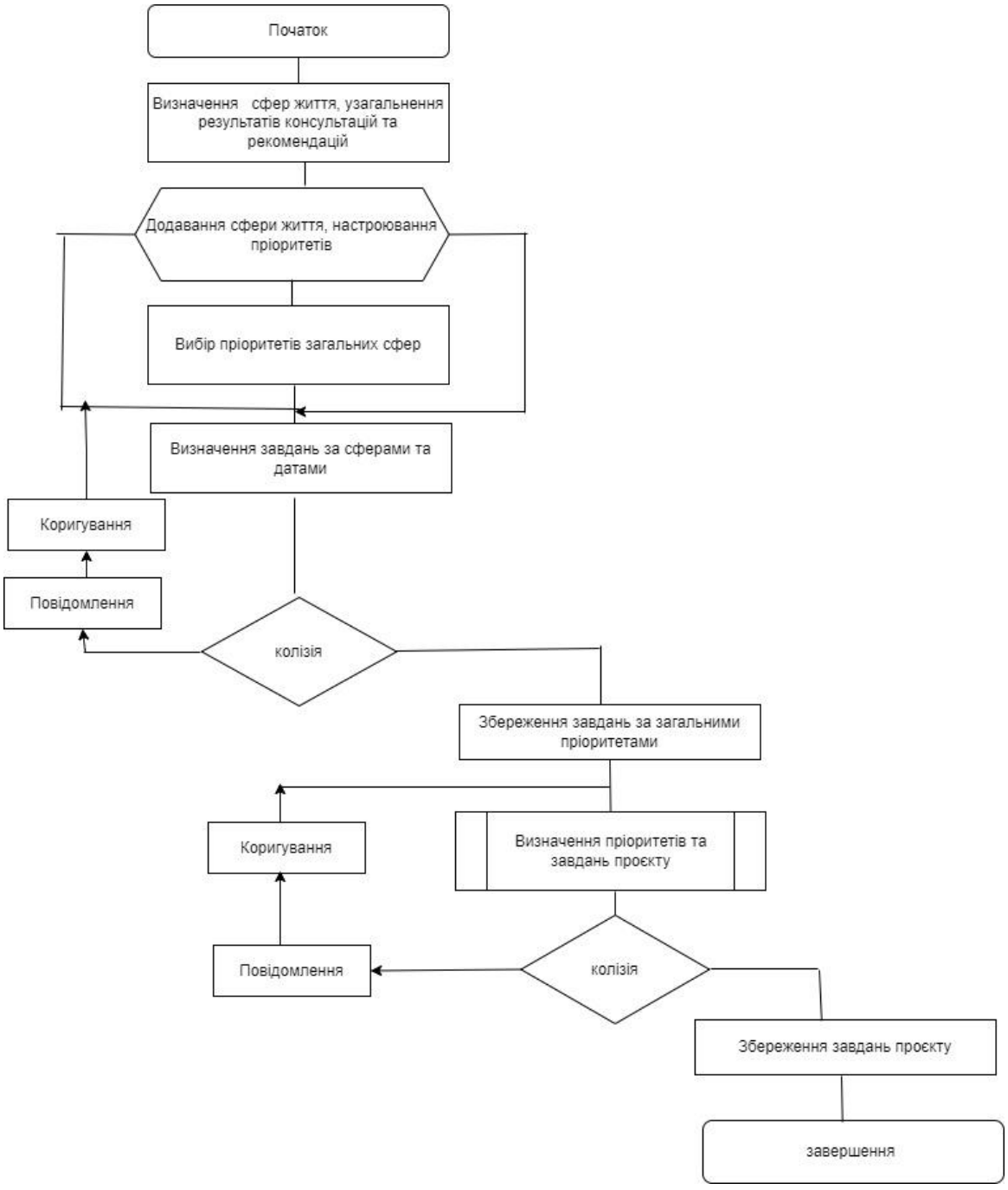


Рисунок 2.5 Алгоритм реалізації узгодження пріоритетів загальних сфер життя та проєкту

Отже, метод пріоритизації завдань повинен базуватись на визначених

критеріях та узгоджувати пріоритети виконавця та проєкту або/і виробничої діяльності.

2.4 Висновки до розділу 2

У другому розділі було проведено аналіз методів пріоритизації та запропоновано узгоджений метод пріоритизації на прикладі вибору пріоритетів різних сфер життя та роботи в проєктах або організаціях. Створено моделі та алгоритми, які є основою для програмної реалізації.

Визначені моделі та алгоритми є основою запропонованих методів узгодження пріоритетів на рівні проєкту, виконавця, менеджера.

Запропоновані алгоритми можуть бути адаптовані для різних методологій, критеріїв важливості, веб систем управління.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВИБОРУ ПРІОРИТЕТНИХ ЗАВДАНЬ

3.1 Варіантний аналіз і обґрунтування вибору мов програмування для реалізації програмного забезпечення

Веб-програмування охоплює величезну сферу, яка включає створення та підтримку веб-сайтів і вебзастосунків. Це динамічний домен, який поєднує різні технології, мови програмування та інструменти для створення та вдосконалення сторінок та порталів Інтернету [14].

Веб-програмування можна загалом розділити на дві основні категорії: інтерфейсна розробка та бекенд-розробка, кожна з яких зосереджується на різних аспектах вебзастосунків і сайтів.

Python, відомий своєю читабельністю та ефективністю, широко використовується у веб-розробці, особливо з такими фреймворками, як Django та Flask.

Ruby on Rails – фреймворк моделі-представлення-контролера (MVC), написаний на Ruby, відомий своєю простотою використання для швидкого створення складних веб-сайтів.

PHP – серверна мова сценаріїв, вбудована в HTML, яка використовується для керування динамічним вмістом, базами даних, відстеженням сеансів і навіть створення цілих сайтів електронної комерції.

Бази даних є важливим компонентом веб-програмування, зберігаючи дані, необхідні веб-сайтам і програмам. Бази даних SQL, такі як MySQL, PostgreSQL і SQLite, а також бази даних NoSQL, такі як MongoDB, є невід’ємною частиною розробки серверної частини, забезпечуючи ефективні механізми зберігання та пошуку даних.

Розробка веб-програми зазвичай передбачає розробку інтерфейсу, і серверної частини для створення повноцінної програми.

Знання середовищ веб-хостингу, хмарних служб, таких як AWS (веб-сервіси Amazon), і систем контролю версій, таких як Git дозволяють інтегрувати різні застосунки в єдину систему.

У сфері веб-розробки на вибір мов програмування впливають конкретні компоненти, які проєктуються чи створюються. Певні мови створено для розробки інтерфейсу користувача або інтерфейсу, зосереджуючись на створенні елементів, з якими користувачі взаємодіють безпосередньо, наприклад графіки та кнопок. Це включає мови розмітки, такі як HTML і CSS, які необхідні для структурування та стилізації веб-сторінок. І навпаки, мови на стороні сервера використовуються для полегшення взаємодії між стороною веб-сайтів, на яку звернений користувач, і основними базами даних і серверами. Ці мови відіграють вирішальну роль в обробці даних, якими обмінюються між інтерфейсом і сервером.

JavaScript виділяється як універсальна мова сценаріїв, яка виконується безпосередньо з вихідного коду без необхідності попереднього перетворення на машинну мову. Пов’язаний переважно з інтерфейсною розробкою, JavaScript є ключовим для додавання інтерактивних функцій до веб-сторінок, включаючи кнопки, які можна натискати, функції масштабування зображення та відтворення мультимедіа. Його корисність поширюється не тільки на інтерфейс, а й на повну розробку, що підтримується широким впровадженням на відомих веб-сайтах, таких як LinkedIn, Amazon і Facebook. Крім того, можливості JavaScript охоплюють розробку на стороні сервера через середовище виконання Node.js, що дозволяє розробляти програми в різних операційних системах [15].

Angular, фреймворк з відкритим вихідним кодом, революціонізує розробку динамічних програм, використовуючи HTML як мову шаблонів і покращуючи її для полегшення створення Rich Internet Applications (RIA). Ці програми імітують інтерактивність і візуальну насиченість настільних програм. Angular особливо ефективний у створенні односторінкових застосунків (SPA), які завантажуються один раз і динамічно оновлюють вміст

через JavaScript, тим самим підвищуючи продуктивність і взаємодію з користувачем. Angular характеризується тим, що пропонує такі функції, як модульне тестування, багаторазовий код, зв’язування даних і контролери на основі JavaScript, що дозволяє розробляти надійні та ефективні програми, сумісні з основними браузерами та мобільними платформами.

Vue, інший фреймворк JavaScript, відомий своєю легкою та адаптивною природою, що робить його придатним для покращення інтерфейсів користувача або розробки SPA з нуля. Vue найкраще підходить для таких завдань, як створення прототипів веб-сторінок, дизайн інтерфейсу користувача та розширення існуючих програм, пропонуючи доступне та ефективне рішення для розробників, які прагнуть покращити або інновувати веб-проєкти.

Vue особливо корисний для таких завдань, як створення прототипів веб-сторінок, розробка інтерфейсів користувача та розширення існуючих програмних програм. Він виділяється завдяки своїй здатності до поступового впровадження, тобто його можна легко та швидко інтегрувати в частини існуючого проєкту без необхідності капітального ремонту всієї системи. Ефективність Vue додатково підтримується його офіційними бібліотеками, які за потреби пропонують розширену функціональність.

TypeScript розширює стандартні можливості JavaScript, додаючи типи, роблячи код надійнішим, простішим у підтримці та простішим у розгортанні. Це додавання суворої типізації дозволяє розробникам визначати змінні з певними типами даних, зменшуючи типові помилки кодування.

TypeScript підтримує всі бібліотеки та API JavaScript і містить функції для виправлення помилок, що робить його надійним вибором для розробників. Незважаючи на дещо вищий рівень навчання, ніж JavaScript, TypeScript став кращим вибором для багатьох, а звіт про стан JS 2023 підкреслює його популярність і задоволеність користувачів.

Python відомий своєю універсальністю та легкістю вивчення, що робить його здатним працювати над складними програмами. Нещодавно Python став

найпопулярнішою мовою програмування, відомою завдяки її використанню в машинному навчанні, штучному інтелекті та науці про дані, а також для серверної веб-розробки [15].

Об’єктно-орієнтований підхід Python організовує код у незалежні блоки, що сприяє ефективному програмуванню. Він також характеризується величезною екосистемою модулів, бібліотек і фреймворків, які прискорюють процес розробки, надаючи попередньо написаний код для швидкого масштабування проєктів.

Django, структура Python, розроблена, щоб задовольнити швидкі вимоги редакцій, завдяки своїй можливості швидко розробляти функціональні та масштабовані вебзастосунки. Він постачається з попередньо створеними компонентами для звичайних потреб розробки, таких як автентифікація користувачів і відображення сайтів.

Flask, інший фреймворк Python, відомий як мікрофреймворк, що робить його ідеальним для простих проєктів вебзастосунків.

Ruby – це об’єктно-орієнтована мова програмування, яка надає перевагу простоті та зручності використання. Ruby служить основою для Ruby on Rails, популярного фреймворку, який спрощує розробку вебзастосунків за допомогою великої спільноти та великих ресурсів.

Jekyll, також заснований на Ruby, є простим інструментом для створення статичних веб-сайтів.

Swift – це мова програмування з відкритим вихідним кодом, призначена для розробки застосунків на платформах Apple, яка пропонує поєднання потужності, швидкості та безпеки. Вона має спрощений синтаксис для полегшення обслуговування, високої продуктивності та інтерактивного налагодження, що дозволяє зберігати сумісність для існуючих розробників Objective-C. PHP, серверна мова сценаріїв, необхідна для створення динамічних веб-сторінок, які пропонують персоналізований досвід користувача. Ця мова підтримує такі платформи, як WordPress, дає змогу

налаштовувати та відображати різноманітні веб-сайти користувачам, запроваджувати різноманітні інтерфейси.

Аналіз мов і фреймворків дозволяє зробити висновок, що програмне забезпечення визначення пріоритетних завдань може бути реалізовано на різних мовах програмування. Модулі вибору пріоритетів можуть бути реалізовані як окремий застосунок, але доцільно розробляти модулі для систем управління проєктами, систем управління пріоритетами окремої людини тощо.

3.2 Особливості архітектури та використання фреймворків JavaScript React.Js та Node.Js

Model-View-Controller (MVC) – широко визнана архітектурна структура, яку використовують розробники. Він працює за принципом, який розділяє логіку програми на три взаємопов’язані компоненти: модель, представлення та контролер. Ця структура сприяє модульному підходу до розробки застосунків, забезпечуючи ефективну організацію та взаємодію між кожною частиною [16].

Модель (Model) – містить дані застосунку.

Контролер (Controller) – реалізує механізм виконання функціоналу застосунку за подіями, які ініційовані зовнішнім джерелом та користувачем, відображаються та потребують відповідну реакцію. Така відповідь викликає метод на моделі і виконує реакцію за викликом. Результат з’являться у відображенні.

Відображення (View) – реалізує взаємодію з даними користувача, доступ до даних застосунку, відображає дані для користувача.

Для веб-застосунків цей патерн став стандартом. Для фронтенд частини в межах розроблена головна сторінка вибору сфер, додавання сфер, звертання до тестів методик пріоритизації.

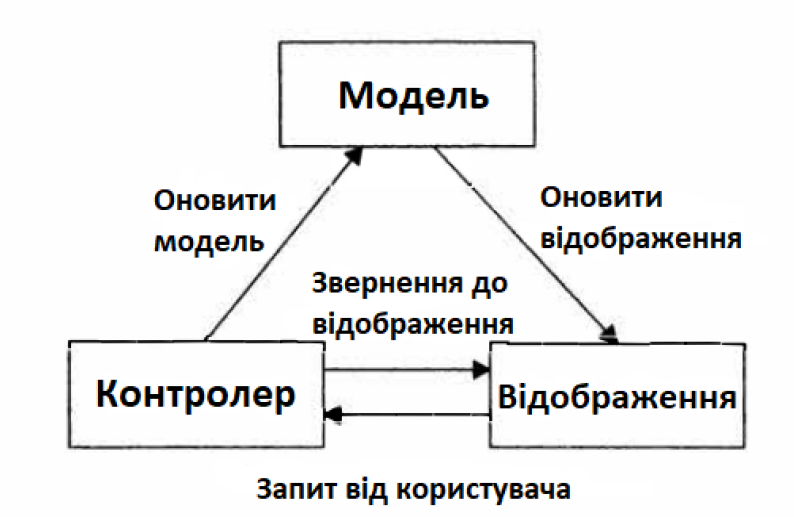


Рисунок 3.1 – Модель MVC

Модель фронтенд-бекенд відповідає моделі MVC таким чином:

Фронтенд частина відповідає за дизайн, динаміку, відображення інформації, можливості скачування інформації в різних форматах.

Бекенд-частина відповідає за логіку, статику, бази даних та реалізує контролери.

Відповідно до цих двох моделей та постановки задачі було вирішено створити програмні модулі управління пріоритетами, які можуть бути адаптовані для різних систем – управління проєктами, управління часом, працювати як окремий застосунок збалансування сфер життя тощо.

Модуль управління пріоритетами завдань відповідно до життєвого циклу виконаємо за допомогою інструментів Microsoft.net. Ця платформа часто використовується для формування програмних продуктів тайменеджменту та управління проєктами.

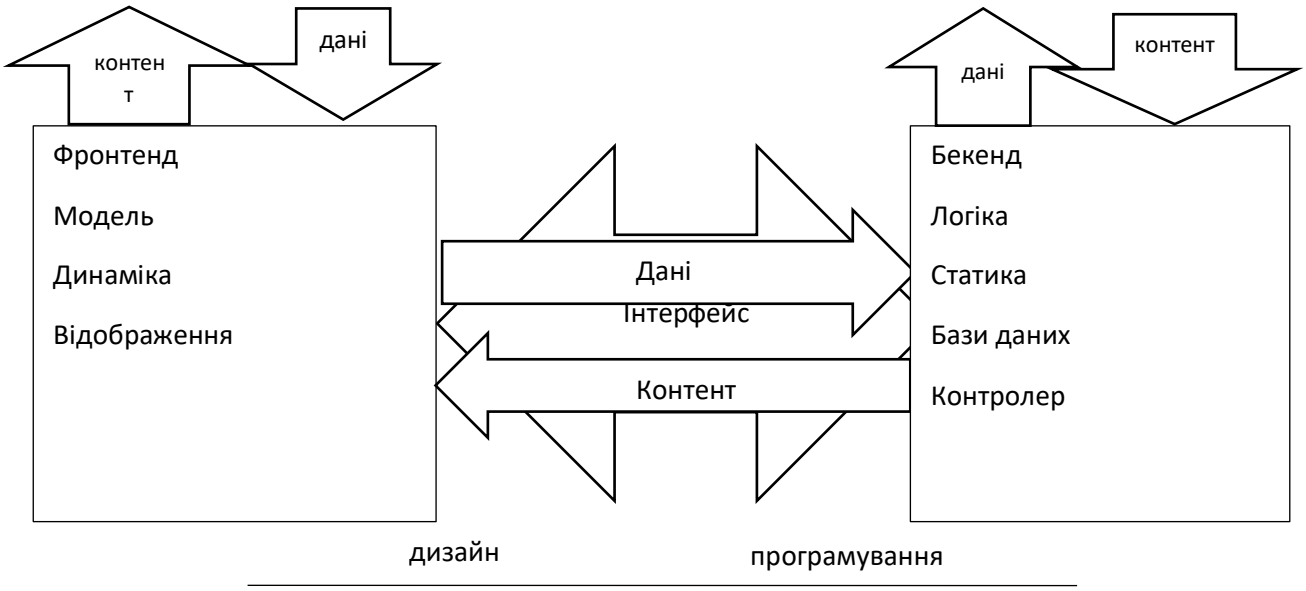


Рисунок 3.2 – Модель фронтенд-бекенд

Веб-система управління пріоритетами, її фронтенд частина була створена за допомогою фреймворку React.

Фронтенд частина дозволяє сформувати головну сторінку управління пріоритетами. В нашому випадку ми виконуємо моделювання пріоритетами за головним критерієм часом та визначаємо можливість делегування.

Користувач має можливість сформувати свій власний перелік сфер життя, ввести дані підключити рекомендації фахівців, менеджерів, результати обчислень пріоритетів і ці всі дані будуть основою для вибору пріоритетів.

Тестування розроблених програмних модулів дозволить зробити висновки щодо можливості вибору пріоритетів та формування запобіжних повідомлень для вирішення колізій у пріоритетах. В застосунку виконано розгляд колізій для вирішення питань співпадіння за часом.

React дійсно став домінуючою силою в сучасному ландшафті веб-розробки, відомий своєю розгалуженою екосистемою та величезним набором елементів інтерфейсу користувача, бібліотек управління станом і додаткових модулів, доступних для розробників. Цей фреймворк, створений на основі

JavaScript, був прийнятий глобальними гігантами, такими як Instagram, Reddit і Facebook, демонструючи його надійність і адаптивність для створення складних програм.

Однією з значних переваг React є його компонентна архітектура, яка дозволяє розробляти багаторазово використовувані компоненти інтерфейсу користувача. Ця модульність сприяє не тільки безперервному процесу розробки, але й підвищує зручність обслуговування кодових баз.

Крім того, продуктивність React у відтворенні даних у поєднанні з його масштабованим характером робить його привабливим вибором для розробників, які прагнуть створювати програми, які можуть розвиватись разом із базою користувачів.

Спрощений API фреймворку в поєднанні з його високопродуктивними можливостями структурує робочий процес розробки, робить інтеграцію різних технологій і розробку складних функціональних можливостей більш ефективними.

Підхід React до кодування, який віддає перевагу ясності та зрозумілості, уникає безладу додаткових атрибутів HTML, що призводить до чистого коду. Ця характеристика приносить значну користь розробникам, особливо якщо порівнювати React з іншими фреймворками, такими як Vue і Angular, де така чистота може бути не завжди [19].

Create React App є безцінним інструментом, пропонуючи спрощений процес налаштування для проєктів. Це особливо корисно для запуску проєктів, спрямованих на створення мінімально життєздатного продукту (MVP).

MVP зосереджується на демонстрації здійсненності концепції продукту без необхідності глибокої розробки архітектур баз даних або впровадження складних проєктів. Цей підхід ідеально підходить для підтвердження потенційного успіху проєкту на його початкових стадіях, забезпечуючи основу, яку можна поступово розширювати та вдосконалювати [17].

Екосистема React виходить за рамки веб-розробки, підтримуючи створення настільних і мобільних застосунків, статичних веб-сайтів і рішень для відтворення на стороні сервера.

Адаптивність фреймворку також прокладає шлях для інтеграції передових технологій, таких як віртуальна реальність, у вебзастосунки, розширюючи таким чином горизонти того, чого можна досягти за допомогою React.

Підсумовуючи, можна стверджувати, що React є потужною, універсальною структурою для веб-розробки, пропонуючи розробникам повний набір інструментів і бібліотек для створення динамічних, ефективних і масштабованих програм. Його компонентна архітектура, оптимізація продуктивності та чисті методи кодування в поєднанні з підтримкою активної спільноти розробників гарантують, що React є основою для створення мікросервісних веб-систем, які можуть бути доповнені новими модулями і працювати з різними готовими продуктами.

У сфері веб-розробки бекенд відіграє вирішальну роль у функціональності та роботі веб-програми або веб-сайту. По суті, це магістраль, яка підтримує інтерфейс, гарантуючи, що все, з чим користувач взаємодіє на інтерфейсі, працює безперебійно, виконуючи основні функції.

Сервер дійсно може бути реалізований на різних мовах програмування, кожна зі своїм унікальним набором бібліотек і фреймворків, призначених для оптимізації процесу розробки. Ця гнучкість дозволяє розробникам вибирати найкращі інструменти, які відповідають конкретним потребам їх проєкту, враховуючи такі фактори, як вимоги до проєкту, масштабованість і досвід команди.

Розробка серверної частини зазвичай поділяється на три основні компоненти: логіку на стороні сервера, бази даних та інтерфейси прикладного програмування (API).

Кожен компонент відіграє важливу роль у загальному механізмі веб-ресурсу.

Серверна логіка відповідає за керування реалізацією клієнт-серверної моделі. Він обробляє запити від клієнта, обробляє їх, а потім надсилає відповідну відповідь назад клієнту. Серверна логіка забезпечує правильну та ефективну роботу веб-програми.

Бази даних мають вирішальне значення для зберігання, отримання та керування даними. Вони служать основою для серверної частини, де зберігаються всі дані програми. Кожного разу, коли інформація потрібна або оновлена, серверна логіка взаємодіє з базою даних, щоб отримати або змінити дані відповідно.

Інтерфейси прикладного програмування (API) відіграють ключову роль у забезпеченні взаємодії між різними програмними компонентами. Вони дозволяють бекенду спілкуватися з інтерфейсом, надсилаючи та отримуючи дані. API також можуть уможливити інтеграцію із зовнішніми службами та системами, покращуючи функціональність програми.

Node.js стає потужним середовищем для розробки серверних вебзастосунків. Це дозволяє розробникам використовувати JavaScript, мову, яка традиційно використовується для розробки інтерфейсу, для написання коду на стороні сервера. Ця уніфікація мови в стеку може спростити процес розробки, оскільки розробники можуть використовувати одну мову програмування як для зовнішньої, так і для задньої частини.

Node.js призначений для створення масштабованих мережевих програм і відомий своєю неблокуючою архітектурою, керованою подіями.

Цей підхід дозволяє Node js обробляти багато з’єднань одночасно, що робить його чудовим вибором для застосунків, які потребують високої продуктивності та масштабованості. Використання модулів у Node.js додатково допомагає розробникам, надаючи колекцію бібліотек, які обробляють різні основні функції, зменшуючи складність написання серверних програм. Крім того, Node.js підтримує широкий спектр операційних систем і сумісний з кількома платформами хмарного хостингу, що робить його універсальним вибором для розгортання вебзастосунків. Активна спільнота

навколо Node.js надає тисячі бібліотек з відкритим кодом, що полегшує розробникам додавання функціональних можливостей і функцій до своїх програм [20].

Підсумовуючи, можна стверджувати, що серверна частина веб-програми має ключове значення для забезпечення її функціональності та продуктивності. Розділивши серверну розробку на серверну логіку, бази даних і API, розробники можуть створювати надійні та ефективні програми. Node.js є потужним інструментом для бекенд-розробки, пропонуючи неблокуючу, керовану подіями архітектуру, яка ідеально підходить для створення масштабованих вебзастосунків.

3.3 Розробка прототипу визначення пріоритетів відповідно до життєвого циклу створення програмного продукту технологіями C# та .NET

C# - універсальна об’єктно-орієнтована мова програмування, розроблена Microsoft, яка стала популярним вибором для веб-розробки, зокрема для створення динамічних веб-сайтів, програм і служб на платформі. Її інтеграція в екосистему Microsoft робить його особливо потужним для розробки вебзастосунків, які працюють на серверах Windows, хоча з появою .NET Core його використання розширилося й включає розробку в середовищах Linux і macOS[20-24].

C# зазвичай використовується разом із ASP.NET, надійною структурою для веб-розробки.

ASP.NET розширює платформу .NET інструментами та бібліотеками, призначеними спеціально для створення вебзастосунків. Ця комбінація дозволяє розробникам створювати все, від невеликих веб-сайтів до складних вебзастосунків [21].

Архітектура Model-View-Controller (MVC) в ASP.NET сприяє організованій структурі коду, полегшуючи керування складними програмами та ефективнішу розробку, тестування та підтримку коду.

Швидке створення веб-форм спрощує розробку веб-сторінок із меншим кодом.

Висока продуктивність NET Core обумовлена використанням інтегрованого середовища розробки, широким спектром інструментів як проєктування, так і адаптації для впровадження.

Entity Framework дозволяє розробникам працювати з базами даних за допомогою об’єктів .NET.

SignalR бібліотека спрощує процес додавання веб-функцій реального часу до програм. Це забезпечує зв’язок у режимі реального часу між сервером і клієнтом.

Підсумовуючи, можна зробити висновок, що платформа NET Core дозволяє швидко створити модуль управління пріоритетами для врахування етапів життєвого циклу. Такий модуль буде активно використовуватись в системі управління проєктами та часом, яка розроблена інструментами NET.

3.4 Висновки до розділу 3

Отже, аналіз різних технологій програмування показав, що найбільш ефективним рішенням є використання фреймворків мови JavaScript. Такий підхід дозволяє сформувати вебзастосунок з можливістю масштабування, визначити інструменти для створення фронтенд та бекенд частин застосунку та адаптувати в подальшому модуль програмного забезпечення управління пріоритетними завданнями для систем управління проєктами, таймменеджменту, самоменеджменту тощо.

C# для веб-розробки пропонує комплексну та потужну платформу для створення широкого спектру вебзастосунків. Його поєднання з ASP.NET, підтримка MVC і можливість розробки на кількох платформах за допомогою .NET Core роблять його привабливим вибором для багатьох розробників.

Отже, реалізація модулів виконана на мовах JavaScript та C#.

4 ТЕСТУВАННЯ ПЗ УПРАВЛІННЯ ПРІОРИТЕТАМИ ЗАВДАНЬ

4.1 Тестування програмного забезпечення управління пріоритетами

Тестування програмного забезпечення управління пріоритетами буде виконано як тестування вебзастосунків.

Серед видів тестування можна відокремити такі:

1. Функціональне тестування. Робота всіх форм та функцій.
2. Тестування зручності використання, навігації буде виконано разом з функціональним. В процесі адаптації та впровадженні модулів, інтерфейс вебзастосунку може бути удосконалений.
3. Тестування продуктивності та безпеки визначається відповідно до виконання функцій та після впровадження модулів.

Кожного разу, коли в програму вносяться оновлення або зміни, регресійне тестування перевіряє, чи новий код не вплинув негативно на наявні функції. Це часто передбачає повторне виконання попередніх тестів, щоб переконатися, що раніше розроблене та перевірене програмне забезпечення продовжує працювати після змін.

Тестування вебзастосунків – це комплексний процес, який вимагає ретельного планування та виконання. Звертаючись до кожного з цих типів тестування, ми можемо переконатися, що розроблені модулі відповідають стандартам якості, пропонують позитивну взаємодію з користувачем.

На основі визначених вимог до управління пріоритетами були створені модулі програмного забезпечення.

Перший модуль створено як вебзастосунок для формування загальних пріоритетів за сферами життя та розподілення завдань відповідно сфер життя та часу.

Другий модуль передбачає використання вебзастосунку в проєкті разом з менеджером проєкту та іншими учасниками.

Третій модуль, виконаний на мові C# є модулем для моніторингу етапів життєвого циклу проєкту і побудований як прототип пріоритетів виконання завдань відповідно до визначеної методології. Такий модуль може бути вбудований у відомі системи тайменеджменту.

Запуск менеджера завдань представлено на рис. 4.1.

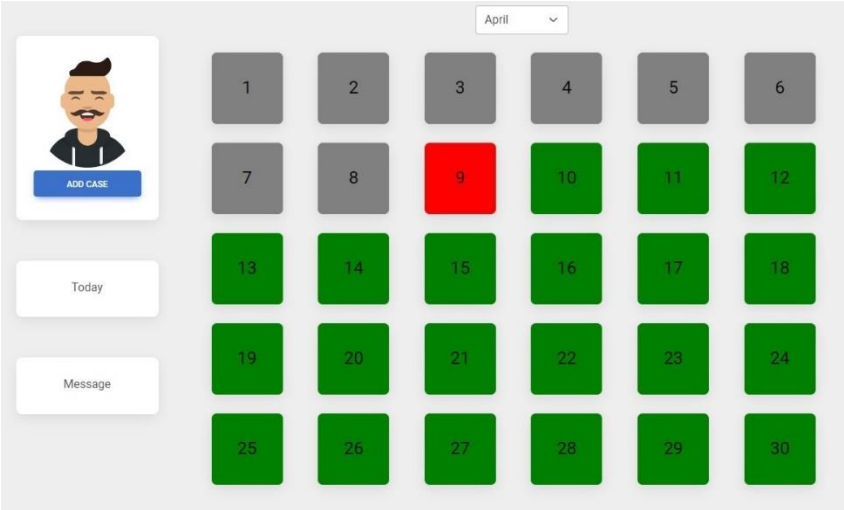


Рисунок 4.1 – Запуск менеджера завдань

Аналогічно визначаються методики для формування результатів пріоритизації (Рис. 4.2),

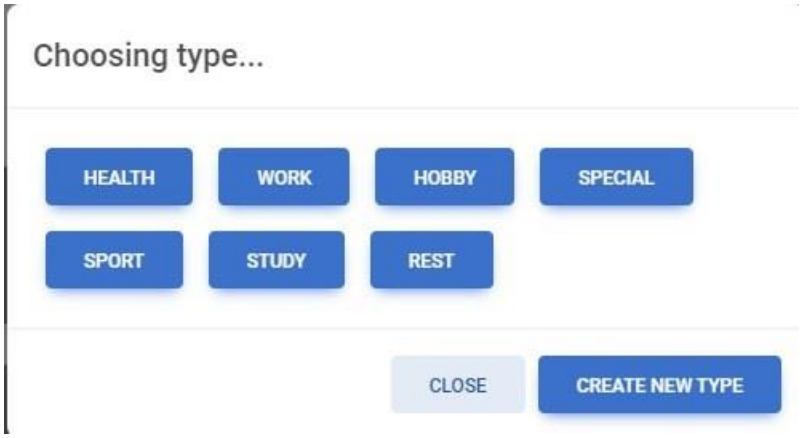


Рисунок 4.2 – Формування типів сфер життя

При подальшому розвитку доцільно сформувати баланс для типових сфер життя та тих сфер, які адаптовані до потреб користувачів.

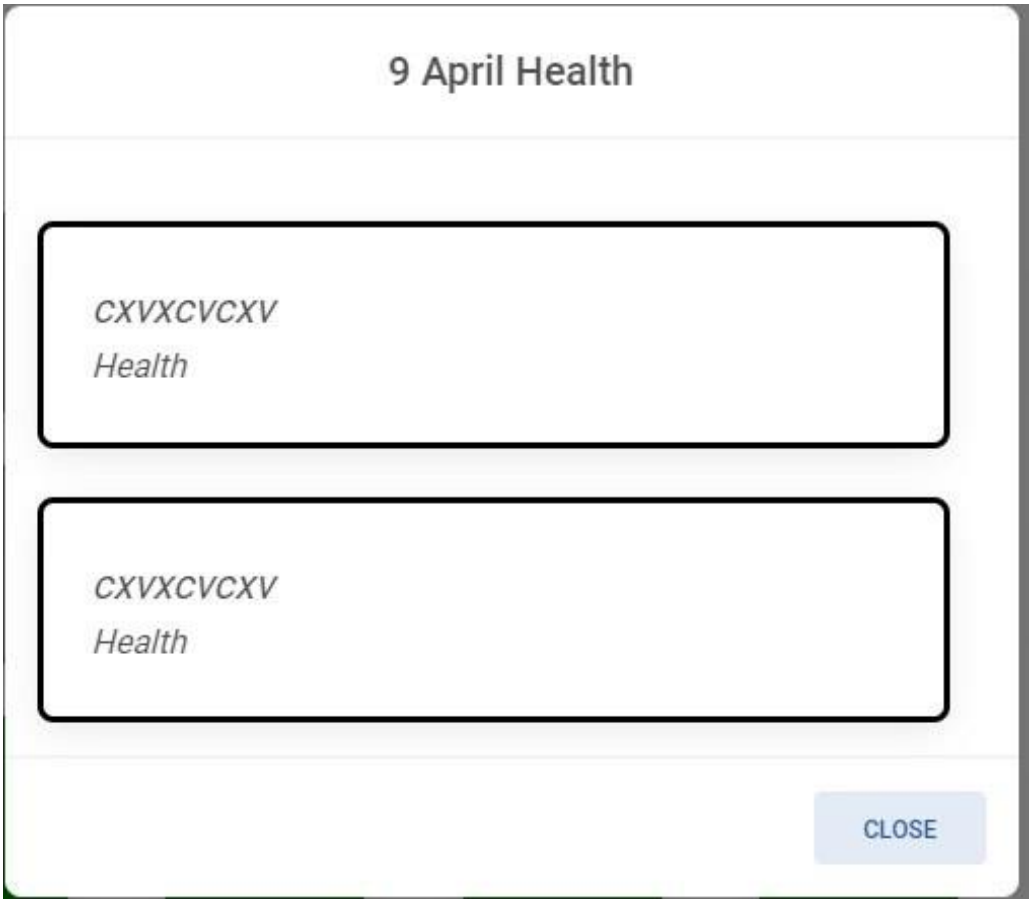


Рисунок 4.3 – Формування нового типу сфери

В першій версії функціонал дозволяє формувати різні сфери і знаходити їх частку за пріоритетами (бали виставляє користувач).

В подальших версіях бали формуються автоматично для визначених сфер, серед яких Здоров'я, Навчання – наприклад, зменшення пріоритизації для ігор, якщо ліміт перевищено тощо.

Користувач може створити новий розділ для балансу сфер життя (Рис. 4.4).

За загальною моделлю працівник спочатку проходить тестування пріоритизації за всіма сферами життя, для того щоб в нього були дані для прийняття рішень щодо професійної діяльності. За згодою користувача, таке

тестування може бути здійснено фахівцями з управління персоналом, психологом.

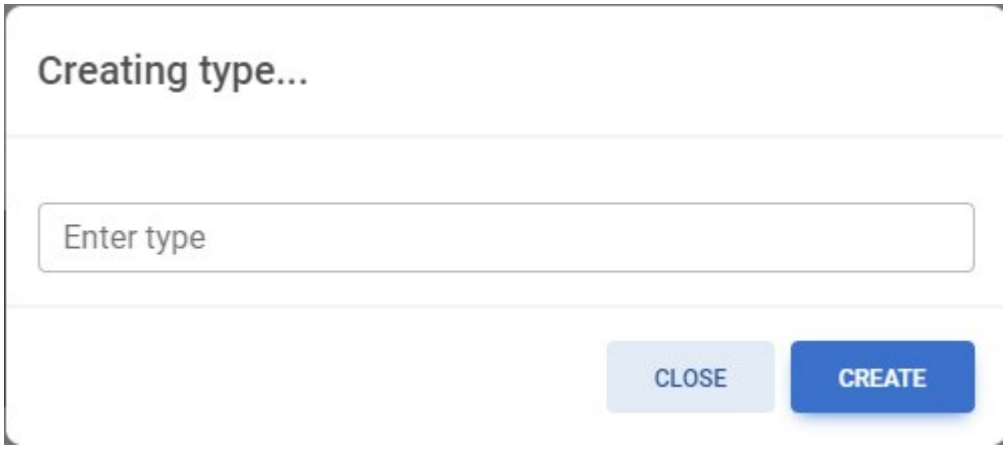


Рисунок 4.4 Створення нового розділу для сфер життя

На рис. 4.5 представлено пріоритизацію завдань та зв’язок з календарем.

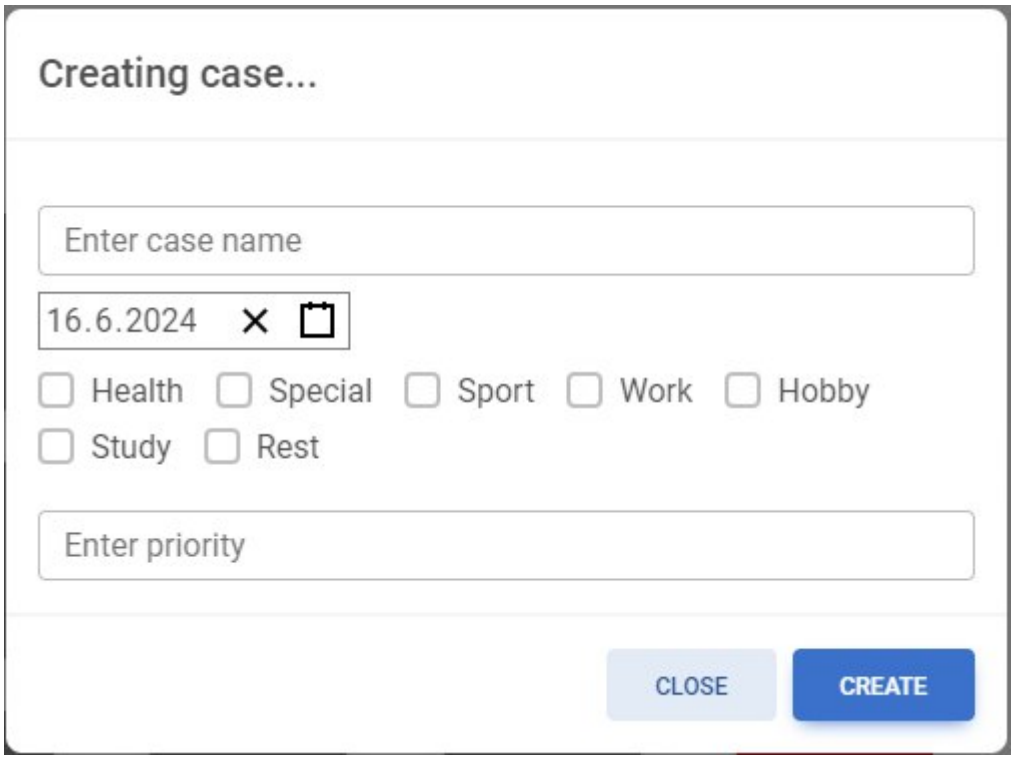


Рисунок 4.5 – Визначення пріоритизації та прив’язка до календаря

Перевіримо, як система формує повідомлення, якщо є колізія у виконанні завдань.

На рис. 4.6 представлено вікно формування повідомлення щодо раннього пріоритету по здоров'ю.

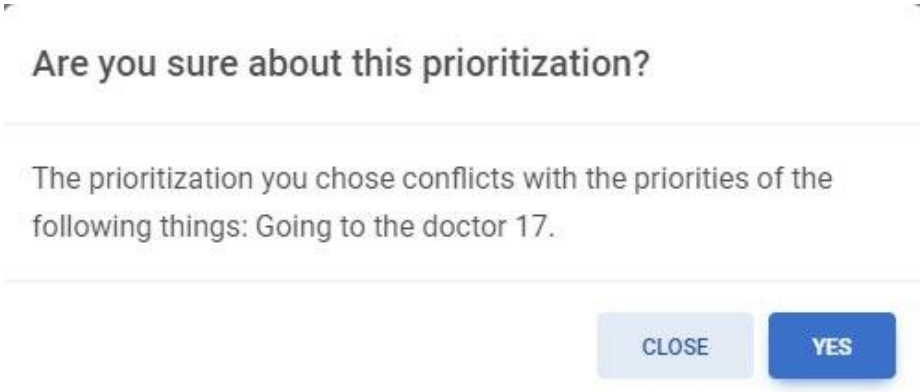


Рисунок 4.6 Повідомлення про конфлікт між встановленням нового завдання та більш раннього пріоритету (за сферою Здоров'я)

На рисунку 4.7 представлено вікно модулю узгодження завдань проєкту та пріоритизації напрямів діяльності учасників проєкту менеджером.

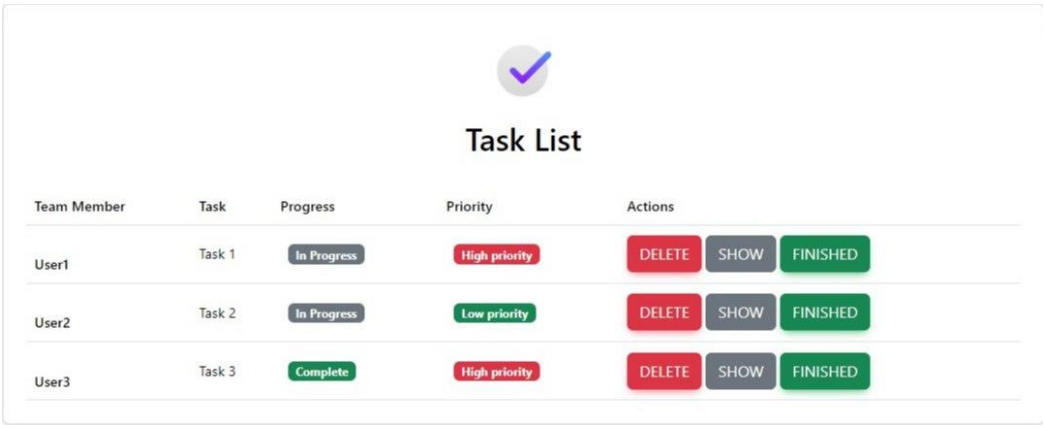


Рисунок 4.7 – Аркуш завдань за учасниками проєкту для менеджера

Свій аркуш завдань також має кожен учасник команди.

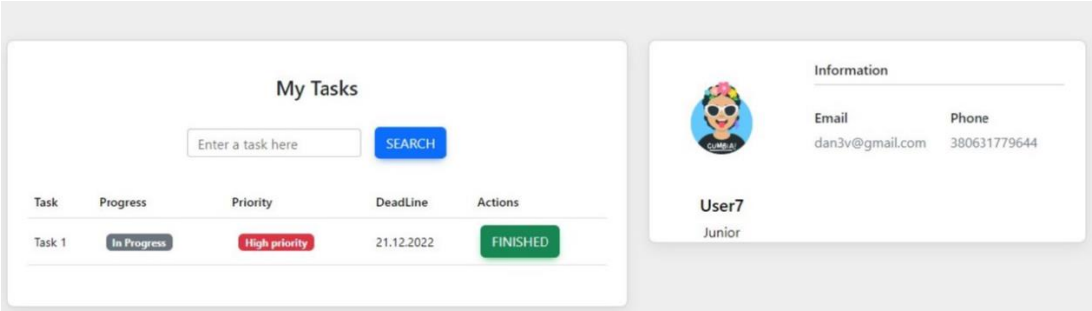


Рисунок 4.8 – Аркуш завдань учасника команди

Отже, основний функціонал роботи модулю вибору пріоритетних завдань за сферами життя та в проєкті працює відповідно до вимог та дозволяє визначити колізії в пріоритетах.

4.2 Пріоритизація завдань в проєкті відповідно до життєвого циклу створення програмного продукту

Пріоритизація завдань відповідно до життєвого циклу передбачає організацію та відстеження розробки проєктів програмного забезпечення на різних етапах їх життєвого циклу, керуючись обраною методологією управління проєктами.

Необхідно вибрати методологію та сформувати основні вимоги до її реалізації та варіанти формулювання етапів життєвого циклу.

Другий етап складається з формування таблиці життєвого циклу.

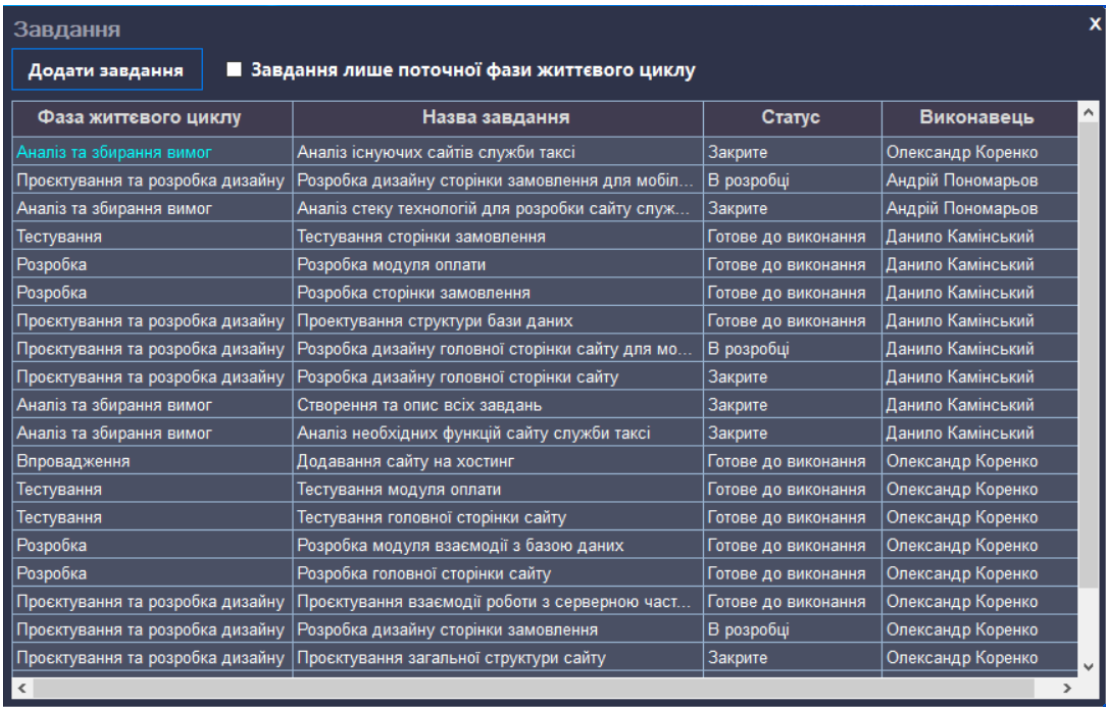
Третій етап полягає у формуванні ключових слів відповідно до кожного етапу життєвого циклу.

Так, наприклад, при формуванні етапів відповідно до гнучкої методології Скрам, проєкт розбивається на спринти. Кожен спринт має своє ключове слово та відповідність цьому спринту контролюється системою та модератором.

Закриття фази життєвого циклу передбачає виконання всіх завдань або перенесення завдань на другу фазу зі зміною ключового слова.

Специфікація виконання також може мати зміни з подальшим їх впровадженням на кожній фазі життєвого циклу.

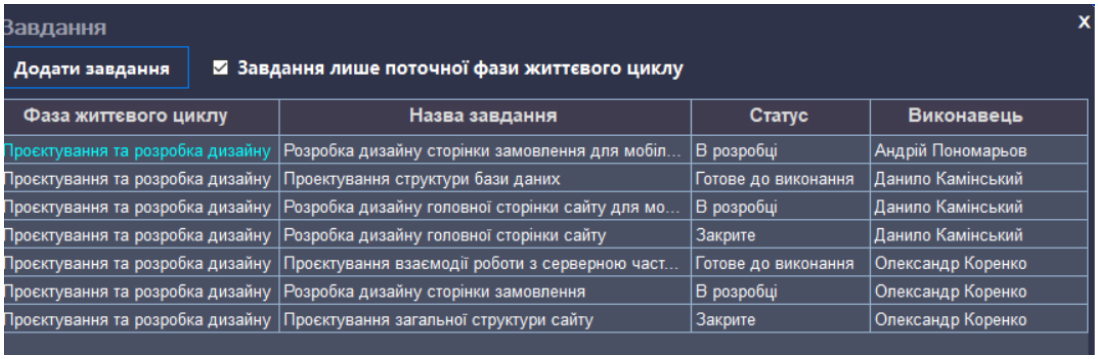
Менеджер додає завдання після вибору етапу життєвого циклу.



Фаза життєвого циклу	Назва завдання	Статус	Виконавець
Аналіз та збирання вимог	Аналіз існуючих сайтів служби таксі	Закрите	Олександр Коренко
Проєктування та розробка дизайну	Розробка дизайну сторінки замовлення для мобіл...	В розробці	Андрій Пономарьов
Аналіз та збирання вимог	Аналіз стеку технологій для розробки сайту служ...	Закрите	Андрій Пономарьов
Тестування	Тестування сторінки замовлення	Готове до виконання	Данило Камінський
Розробка	Розробка модуля оплати	Готове до виконання	Данило Камінський
Розробка	Розробка сторінки замовлення	Готове до виконання	Данило Камінський
Проєктування та розробка дизайну	Проєктування структури бази даних	Готове до виконання	Данило Камінський
Проєктування та розробка дизайну	Розробка дизайну головної сторінки сайту для мо...	В розробці	Данило Камінський
Проєктування та розробка дизайну	Розробка дизайну головної сторінки сайту	Закрите	Данило Камінський
Аналіз та збирання вимог	Створення та опис всіх завдань	Закрите	Данило Камінський
Аналіз та збирання вимог	Аналіз необхідних функцій сайту служби таксі	Закрите	Данило Камінський
Впровадження	Додавання сайту на хостинг	Готове до виконання	Олександр Коренко
Тестування	Тестування модуля оплати	Готове до виконання	Олександр Коренко
Тестування	Тестування головної сторінки сайту	Готове до виконання	Олександр Коренко
Розробка	Розробка модуля взаємодії з базою даних	Готове до виконання	Олександр Коренко
Розробка	Розробка головної сторінки сайту	Готове до виконання	Олександр Коренко
Проєктування та розробка дизайну	Проєктування взаємодії роботи з серверною част...	Готове до виконання	Олександр Коренко
Проєктування та розробка дизайну	Розробка дизайну сторінки замовлення	В розробці	Олександр Коренко
Проєктування та розробка дизайну	Проєктування загальної структури сайту	Закрите	Олександр Коренко

Рисунок 4.9 –Вікно визначення відповідності завдання та етапу життєвого циклу

На рисунку 4.10 представлено завдання поточної фази життєвого циклу



Фаза життєвого циклу	Назва завдання	Статус	Виконавець
Проєктування та розробка дизайну	Розробка дизайну сторінки замовлення для мобіл...	В розробці	Андрій Пономарьов
Проєктування та розробка дизайну	Проєктування структури бази даних	Готове до виконання	Данило Камінський
Проєктування та розробка дизайну	Розробка дизайну головної сторінки сайту для мо...	В розробці	Данило Камінський
Проєктування та розробка дизайну	Розробка дизайну головної сторінки сайту	Закрите	Данило Камінський
Проєктування та розробка дизайну	Проєктування взаємодії роботи з серверною част...	Готове до виконання	Олександр Коренко
Проєктування та розробка дизайну	Розробка дизайну сторінки замовлення	В розробці	Олександр Коренко
Проєктування та розробка дизайну	Проєктування загальної структури сайту	Закрите	Олександр Коренко

Рисунок 4.10 – Завдання поточної фази життєвого циклу

Також є можливість змінювати методологію та уточнювати фази життєвого циклу, але після заповнення довідника ключових слів.

Додавання завдання представлено на рис. 4.11.

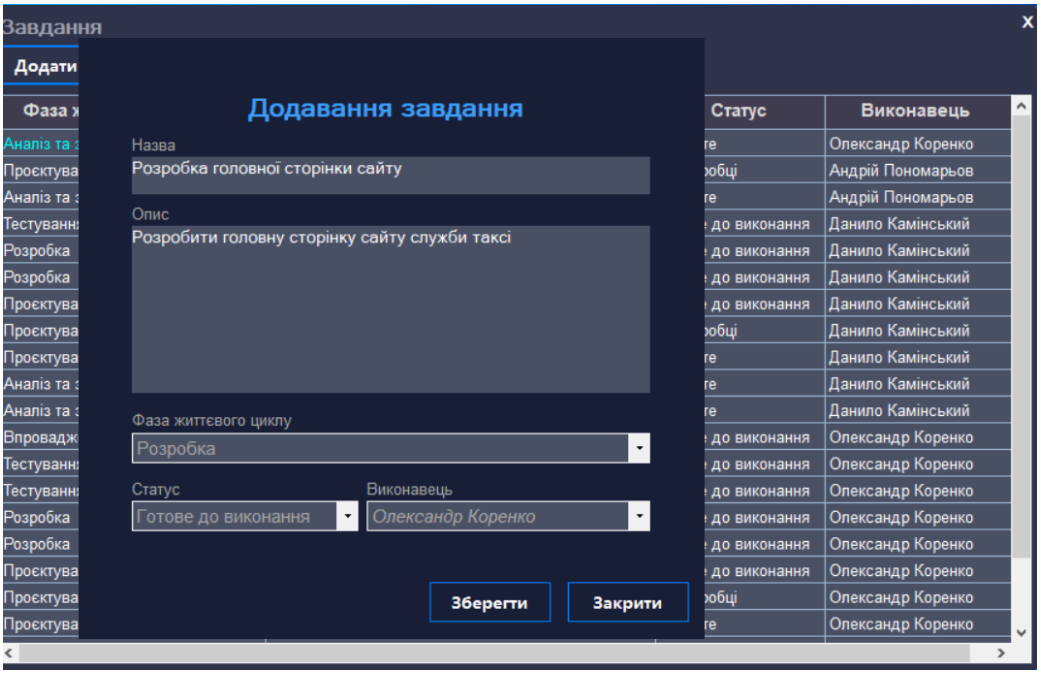


Рисунок 4.11 Додавання завдання

На рис. 4.12 визначення статусу завдання

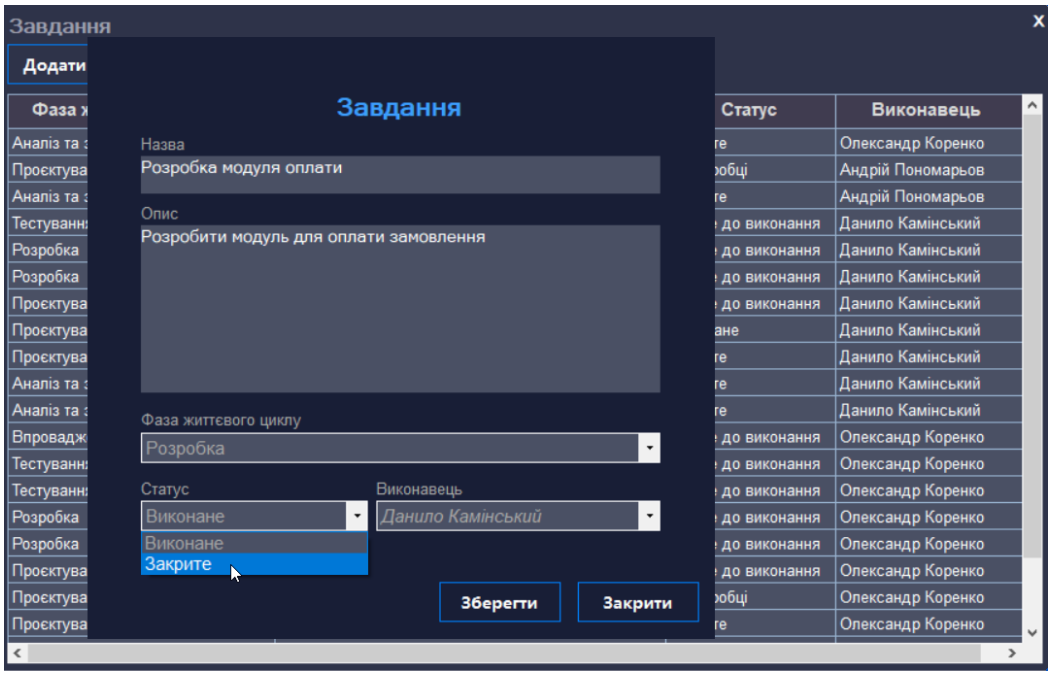


Рисунок 4.12 – Статус завдання

На рис. 4.13 представлено діаграму виконання фаз проєкту

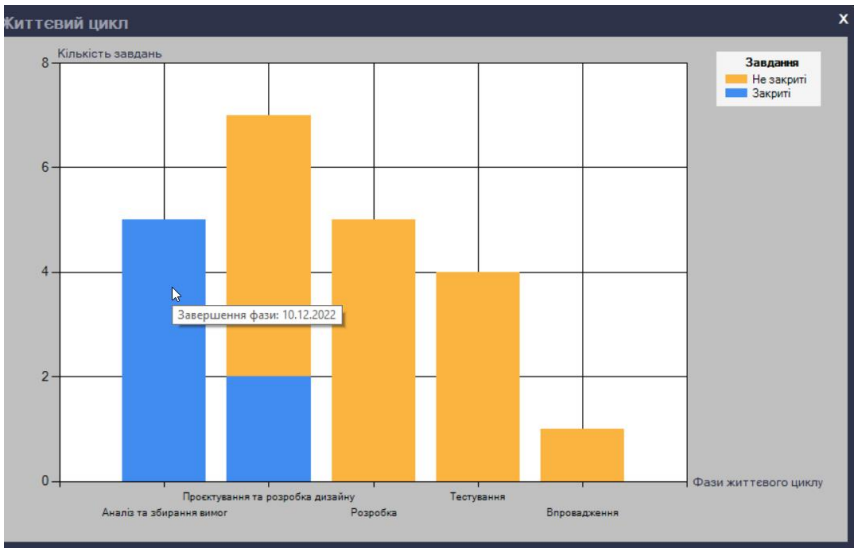


Рисунок 4.13 – Прогрес виконання фаз проєкту

Якщо є колізія (як за моделями та алгоритмами), то формується модальне вікно фази життєвого циклу.

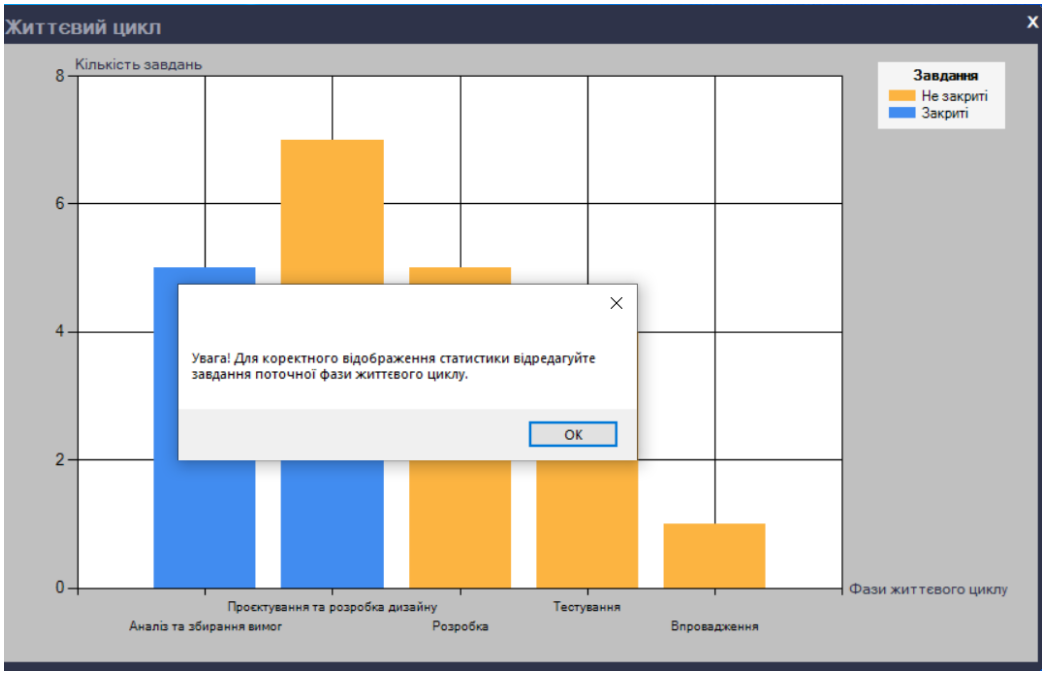


Рисунок 4.14 – Вікно повідомлення при невідповідності завдання та фази життєвого циклу.

4.3 Використання програмного модуля вибору пріоритетів

Деталі щодо мінімальної конфігурації серверного комп’ютера можна знайти в таблицях 4.1

Таблиця 4.1 – Мінімальна конфігурація:

Частота процесора	1 ГГц
Об’єм оперативної пам’яті	1 ГБ
Місце на жорсткому диску	1 ГБ
Операційна система	Windows 7/8/10/11

Рекомендована конфігурація для подальшого впровадження модулів в систему управління пріоритетами завдань представлена в таблиці 4.2 .

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	2 ГГц
Об’єм оперативної пам’яті	4 ГБ
Місце на жорсткому диску	2 ГБ
Операційна система	Windows 7/8/10/11

Програмні модулі керування пріоритетними завданнями відіграють вирішальну роль у допомозі окремим особам і організаціям ефективно керувати своїми завданнями та проєктами, визначаючи пріоритети роботи на основі різних критеріїв, таких як терміновість, важливість, терміни та доступні ресурси.

В першу чергу такі модулі можуть бути використані менеджерами проєктів та менеджерами з персоналу при виконанні розстановки завдань, підбору персоналу в команду.

Ці програмні модулі дозволяють користувачам призначати завданням рівні пріоритету, допомагаючи розрізняти те, що потребує негайної уваги, і те, що може почекати. Чітко позначаючи завдання як високого, середнього або низького пріоритету, члени команди можуть зосередити свої зусилля на виконанні в першу чергу найважливіших завдань, забезпечуючи ефективний розподіл ресурсів.

Удосконалене управління часом дозволяє краще реалізувати управління часом, визначення термінових завдань.

Удосконалення командної співпраці здійснюється за рахунок використання функцій контролю та врахування особистих пріоритетів.

Модуль відповідності життєвого циклу дозволяє притримуватись кожної фази створення програмного продукту, завдання можна організовувати в логічну послідовність, забезпечуючи оптимізацію робочих процесів. Це означає, що завдання, які залежать від виконання інших, плануються належним чином, що забезпечує нормативний робочий процес і швидше завершення проєкту. Також пріоритетність використовується для оптимізації різних ресурсів. Модулі відстеження та звітності дають уявлення про хід виконання завдань і проєктів. Менеджери можуть використовувати цю інформацію для прийняття обґрунтованих рішень щодо перерозподілу ресурсів, коригування графіків або зміни пріоритетів завдань, якщо це необхідно. Розставляючи завдання за пріоритетністю, окремі особи та команди можуть уникнути надто багато роботи одночасно. Це допомагає підтримувати збалансоване робоче навантаження та зменшує ризик вигорання.

Управління пріоритетними завданнями гарантує, що завдання, над якими працюють, відповідають ширшим цілям і завданням організації. Таке узгодження гарантує, що кожне виконане завдання сприяє досягненню стратегічних цілей організації.

У динамічному середовищі пріоритети можуть швидко змінюватися. Програмне забезпечення для керування пріоритетними завданнями дозволяє швидко змінювати пріоритети завдань, допомагаючи командам адаптуватися до змін і зосереджуватися за потреби.

Використання програмних модулів керування пріоритетними завданнями є невід'ємною частиною сучасного робочого середовища, пропонуючи структуровані та ефективні способи керування завданнями, спільної роботи над проєктами та досягнення цілей організації. Узгодження з особистими пріоритетами дозволяє підвищити мотивацію та продуктивність праці.

4.3 Висновки до розділу 4

Під час тестування системи проведено ретельний аналіз функціональності, взаємодії користувача з системою, менеджером проєкту.

Аналіз та розробка на різних платформах показали, що вибір технологій не впливає на функціональність і може бути виконаний різними інструментами.

Використання програмних модулів керування пріоритетними завданнями є невід'ємною частиною сучасного робочого середовища, пропонуючи структуровані та ефективні способи керування завданнями, спільної роботи над проєктами та досягнення цілей організації. Узгодження з особистими пріоритетами дозволяє підвищити мотивацію та продуктивність праці.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмне забезпечення для управління пріоритетами завдань.

Таке програмне забезпечення може бути використано в системах управління часом, проєктами, різними сферами життя, виробничою діяльністю, під час роботи команди ІТ-розробки тощо.

Мету розробки ПЗ модулів управління пріоритетами завдань досягнуто.

Розроблено програмні модулі для управління пріоритетами в різних сферах життя, узгодження із завданнями проєкту, управління завданнями в проєкті відповідно до фаз життєвого циклу.

Концепція управління пріоритетами повинна базуватись на теорії вибору пріоритетних завдань відносно сфер життя, важливості на поточний момент, можливостей виконання, зв’язком з визначеними етапами виконанні в проєкті, станом здоров’я, мотивації тощо.

Такі концепції потребують спеціальних розробок в області психології, управління проєктами, використання обчислювальних моделей визначення рангу пріоритетів.

Обчислювальні моделі можуть бути використані за допомогою зовнішніх відповідних застосунків або виконання обчислень за допомогою електронних таблиць.

Рекомендовані критерії для вибору пріоритетів можуть бути введені користувачем як вхідні дані для вибору.

В межах виконання бакалаврської дипломної роботи виконано розробку окремих модулів визначення пріоритетів для балансу виконання завдань.

Запропоновані програмні модулі можуть бути використані для балансування різних сфер життя пересічними особами. А також для узгодження завдання в проєкті, налагодження мотивації для команди, взаємодії між учасниками команди.

Виконаний аналіз інструментів веб-програмування показав, що можуть бути використані різні технології для отримання візуальних інструментів вибору та контролю пріоритетів.

Пріоритетні завдання визначається, розподіляються та реалізуються в часі, між учасниками команди, за важливістю, за категоріями.

Тестування програмного модуля свідчить про можливості використання його для користувача та адаптації для проєктів, менеджерів з персоналу та управління проєктами.

Бакалаврська дипломна робота містить елементи новизни у вигляді запропонованих методів узгодження пріоритетів. Запропоновані методи реалізовані у вигляді програмних модулів на основі моделей та алгоритмів узгодження особистих та виробничих пріоритетів, узгодження завдань з фазами життєвого циклу проєкту.

Практична цінність проєкту полягає у можливості використання вебзастосунків в системах управління проєктами та як окремі модулі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методики пріоритизації URL: https://worksection.com/overview.html

2. Коваленко О.О., Галієнко А.А., Власенко Д.В. Програмний застосунок вибору пріоритетних завдань. Збірник матеріалів інтернет-конференції «Інформаційне суспільство», № 89. URL: http://www.konferenciaonline.org.ua/ua

3. Worksection URL: https://worksection.com/overview.html

4. Trello URL: https://trello.com/uk

5. Asana URL: https://cloudfresh.com/ua/produkty/asana/

6. To Do URL: https://to-do.office.com/tasks/

7. Колесо життєвого балансу онлайн URL: https://coaching-way.com/uk/uk-koleso-zhittievogo-balansu-onlajn/

8. Метод аналізу ієрархій Сааті URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7%D1%83_%D1%96%D1%94%D1%80%D0%B0%D1%80%D1%85%D1%96%D0%B9

9. Smith G., Kolesnik AL, Lavrischeva K., Slabospitsky O. Improving the process of drafting families of software systems elements of agile methodologies *Programming problems.* 2010. № 2-3. P. 261—270

10. Що таке методологія Scrum і коли її варто використовувати URL: https://career.softserveinc.com/uk-ua/stories/what-is-scrum-methodology

11. Kanban in a nutshell URL: https://www.udemy.com/course/kanban-in-a-nutshell

12. Життєвий цикл розробки програмного продукту URL: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz

13. Бхаркава А. Грокаємо алгоритми: ілюстрований посібник для програмістів і допитливих Київ: Балка-Бук, 2023. – 400с.

14. Босько В.В., Константинова Л.В., Марченко К.М.,

Улічев О.С.Web-програмування. Частина 1 (frontend) : навч. посіб. – Кропивницький: ЦНТУ, 2022. – 208 с.

15. Javascript URL: https://uk.legacy.js.org/

16. Python URL: https://uk.legacy.python.org/

17. MVC URL: https://brander.ua/technologies/mvc

18. React URL: https://uk.legacy.reactjs.org/

19. Node.js URL: https://nodejs.org/en/blog/release/v13.13.0

20. Microsoft SQL Server. URL: https://www.udemy.com/course/the-complete-sql-bootcamp

21. Microsoft Azure. URL: https://azure.microsoft.com /

22. Microsoft.net. URL: https://support.microsoft.com/uk-ua

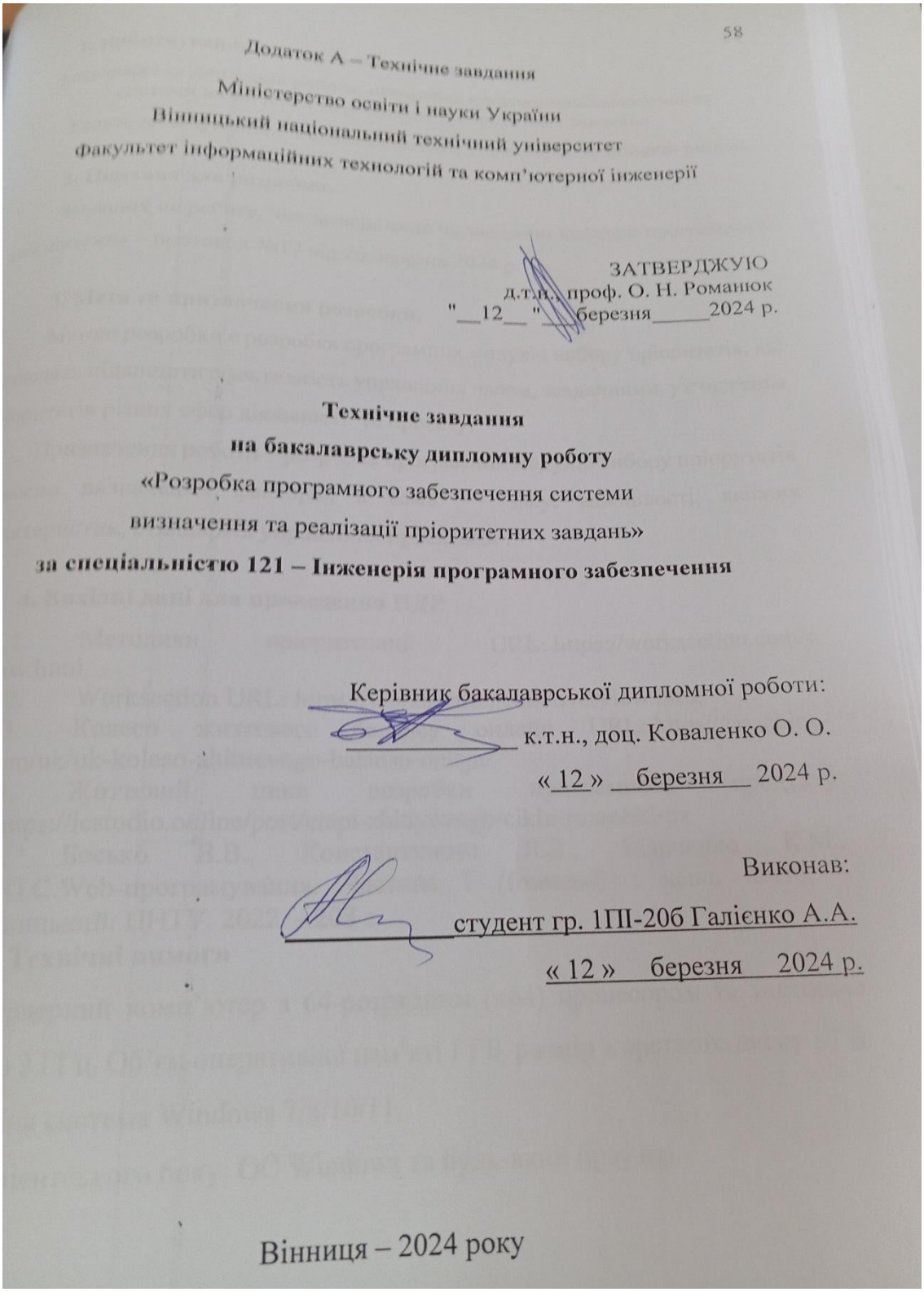
23. Jeffrey Richter CLR via C# (4th Edition) (Developer Reference) Microsoft Press; 6 edition, 2021. – 896 ст.

24. Microsoft Visual Studio. URL: http://uk.wikipedia.org/wiki/Microsoft_ Visual_Studio

25. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад.: О. Романюк, Г. Черноволик. Вінниця : ВНТУ, 2022. 52 с.

ДОДАТКИ

Додаток А – Технічне завдання



58

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"__12__" ____березня_____2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка програмного забезпечення системи
визначення та реалізації пріоритетних завдань»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. Коваленко О. О.
« 12 » березня 2024 р.

Виконав:
_____ студент гр. 1ПІ-20б Галієнко А.А.
« 12 » березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «».

Галузь застосування – таймменеджмент, управління, прийняття рішень.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 р.

3. Мета та призначення розробки.

Метою розробки є розробка програмних модулів вибору пріоритетів, які дозволять підвищити ефективність управління часом, завданнями, узгодження пріоритетів різних сфер діяльності та проєкту.

Призначення роботи – розробка програмних модулів вибору пріоритетів відносно визначених факторів, а саме - часу, важливості, якісних характеристик, стандартів управління проєктами .

4. Вихідні дані для проведення НДР

1. Методики пріоритизації URL: https://worksection.com/overview.html
2. Worksection URL: https://worksection.com/overview.html
3. Колесо життєвого балансу онлайн URL: https://coaching-way.com/uk/uk-koleso-zhittievogo-balansu-onlajn/
4. Життєвий цикл розробки програмного продукту URL: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz
5. Босько В.В., Константинова Л.В., Марченко К.М., Улічев О.С.Web-програмування. Частина 1 (frontend) : навч. посіб. – Кропивницький: ЦНТУ, 2022. – 208 с.

5. Технічні вимоги

Серверний комп’ютер з 64-розрядним (x64) процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 1 ГБ, розмір жорсткого диску 1 ГБ.

Операційна система Windows 7/8/10/11.

З клієнтського боку: ОС Windows та будь-який браузер.

6. Конструктивні вимоги.

Система має відповідати функціональним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

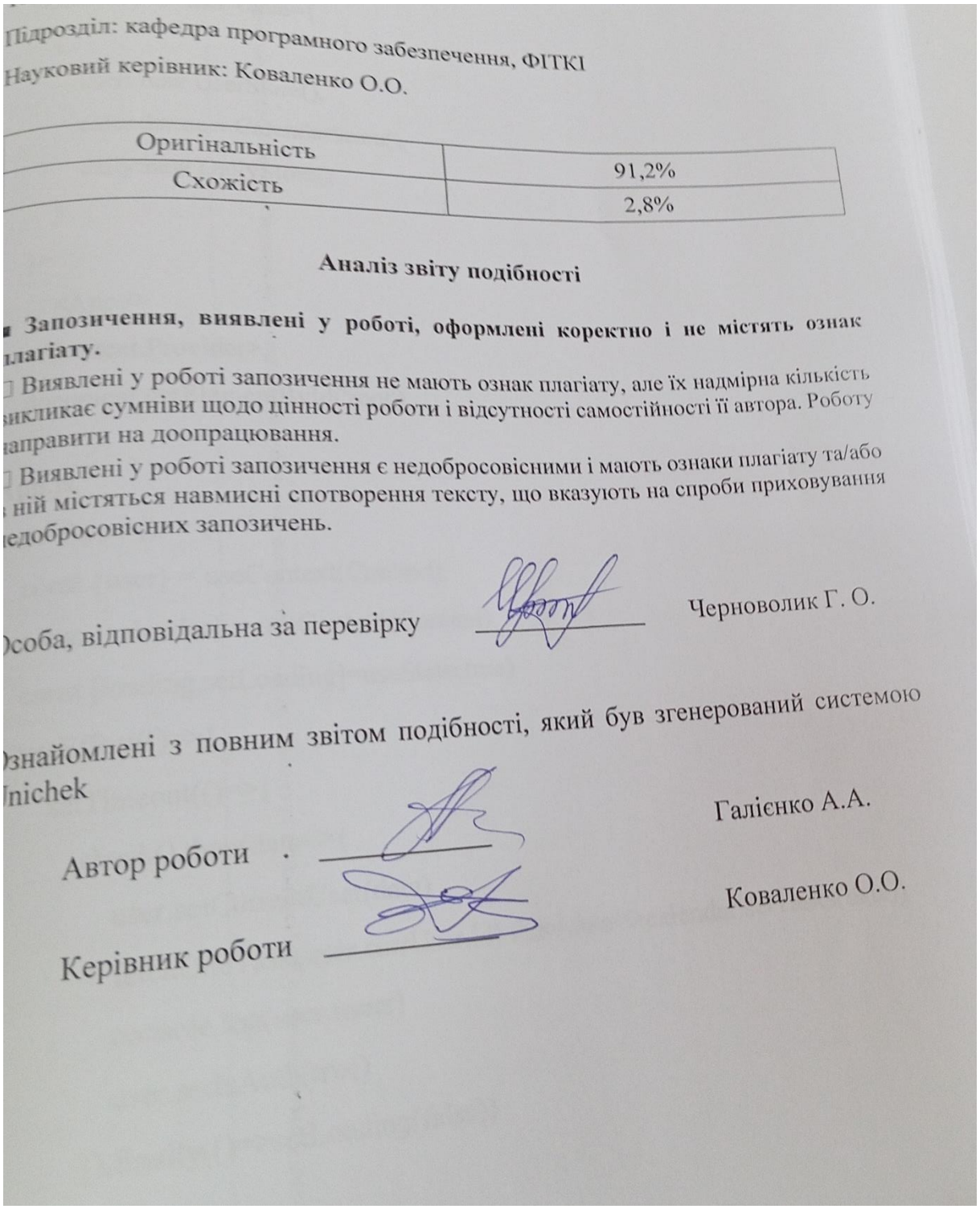
9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку систем для вибору пріоритетів та постановка задач дослідження	14.03.2024 – 31.03.2024
2	Розробка методу, моделі та алгоритмів роботи вебсистеми	01.04.2024– 15.04.2024
3	Розробка модулів вебсервісу	16.04.2024 – 18.05.2024
4	Тестування програми	19.05.2024 – 24.05.2024
5	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.2024

10. Порядок контролю та прийняття

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється корегування бакалаврської дипломної роботи.

Додаток Б – Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень



Підрозділ: кафедра програмного забезпечення, ФІТКІ
Науковий керівник: Коваленко О.О.

Оригінальність	91,2%
Схожість	2,8%

Аналіз звіту подібності

Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek

Автор роботи _____ Галієнко А.А.

Керівник роботи _____ Коваленко О.О.

Додаток В – Лістинг програми

Index.js

```
export const Context = createContext(null)
ReactDOM.render(
   <Context.Provider value={
      {
         user: new UserStore(),
         calendar:new CalendarStore(),
         daily:new DailyStore()
      }
   }>
      <App />
   </Context.Provider>,
   document.getElementById('root')
```

App.js

```
const App =()=> {
   const {user} = useContext(Context)
   const {calendar} = useContext(Context)
   const [loading,setLoading]=useState(true)
   useEffect(()=>{
      setTimeout(()=>{
         check().then(data=>{
            user.setCurrentUser(data)
            fetchOneTask(user.currUser.id).then(data=>calendar.setTasks(data))
            console.log(user.users)
            user.setIsAuth(true)
         }).finally(()=>setLoading(false))
```

```
        },2000)

    },[])
    if(loading){
        return <Spinner animation={"grow"}/>
    }
  return (
      <BrowserRouter>
        <div>
            <NavBar />
            <AppRouter />
        </div>

      </BrowserRouter>
  );
}

export default App;
```

Main.js – головна сторінка

```
const Main = observer(() => {
    // console.log(Date.now())
    let today = new Date();
    const [values, onChange] = useState(new Date());
    const [countDay, setCountDay] = useState(0);
    const [dayArray, setDayArray] = useState([]);
    const [getVisible, setGetVisible] = useState(false);
    const [getVisible1, setGetVisible1] = useState(false);
```

```
    const [typeVisible, setTypeVisible] = useState(false);
    const [testVisible, setTestVisible] = useState(false);
    const {calendar} = useContext(Context)
    const {user} = useContext(Context)

    useEffect(()=>{
        let nowDate = new Date()
        calendar.setCurrentMonth(+today.getMonth()+1)
        calendar.setCurrentMonthsName('April')
        calendar.setCurrentMonth(nowDate.getMonth())
        fetchOneTask(user.currUser.id).then(data=>calendar.setTasks(data))
        fetchUser().then(data=>user.setUser(data))
        fetchOneMessage(user.currUser.id).then(data=> {
            console.log(data)
            calendar.setMessages(data)
        }
        )
    },[])
    useEffect(()=>{
        if(calendar.currentMonthsName === "January" ||
calendar.currentMonthsName === "March" || calendar.currentMonthsName ===
"May" ||
            calendar.currentMonthsName === "July" || calendar.currentMonthsName
=== "August" || calendar.currentMonthsName === "October" ||
            calendar.currentMonthsName ==="December"){

calendar.setMonthsDays([{id:1},{id:2},{id:3},{id:4},{id:5},{id:6},{id:7},{id:8},

{id:9},{id:10},{id:11},{id:12},{id:13},{id:14},{id:15},{id:16},{id:17},{id:18},{i
d:19}
```

```
,{id:20},{id:21},{id:22},{id:23},{id:24},{id:25},{id:26},{id:27},{id:28},{id:29},{
id:30},{id:31}])
        }
        else if(calendar.currentMonthsName === "February"){

calendar.setMonthsDays([{id:1},{id:2},{id:3},{id:4},{id:5},{id:6},{id:7},{id:8},

{id:9},{id:10},{id:11},{id:12},{id:13},{id:14},{id:15},{id:16},{id:17},{id:18},{i
d:19}

,{id:20},{id:21},{id:22},{id:23},{id:24},{id:25},{id:26},{id:27},{id:28}])


        }
        else if(calendar.currentMonthsName === "April" ||
calendar.currentMonthsName === "June" ||
            calendar.currentMonthsName === "September" ||
calendar.currentMonthsName === "November"){

calendar.setMonthsDays([{id:1},{id:2},{id:3},{id:4},{id:5},{id:6},{id:7},{id:8},

{id:9},{id:10},{id:11},{id:12},{id:13},{id:14},{id:15},{id:16},{id:17},{id:18},{i
d:19}

,{id:20},{id:21},{id:22},{id:23},{id:24},{id:25},{id:26},{id:27},{id:28},{id:29},{
id:30}])
        }
    },[calendar.currentMonthsName])


    return (
        <section className="vh-100" style={{ backgroundColor: "#eee" }}>
            <MDBContainer className="py-5 h-100">
```

```
            <MDBRow className="d-flex justify-content-center align-items-
center">
                <MDBCol md={2} className="d-flex justify-content-center align-
items-center">
                    <MDBRow>
                      <MDBCol>
                        <MDBCard className="mb-6">
                          <MDBCardBody className="text-center">
                            <MDBCardImage
                              src="https://mdbcdn.b-cdn.net/img/Photos/new-
templates/bootstrap-chat/ava3.webp"
                              alt="avatar"
                              className="rounded-circle"
                              style={{ width: '150px' }}
                              fluid />
                            <p className=" mb-1">{user.currUser.email}</p>
                            <div className='d-flex flex-column'>
                              <MDBBtn className='mb-2' onClick={()=>{
                                setGetVisible(true)
                              }
                              }>
                                Add Case
                              </MDBBtn>

                            </div>


                          </MDBCardBody>
```

```
                    </MDBCard>
                </MDBCol>
                <MDBCol>
                    <MDBCard className="mb-6">
                        <MDBCardBody className="text-center">
                            <p className=" mb-1">Today</p>
                            {
                                calendar.tasks.filter(taskFilter=>{
                                    if(+taskFilter.day === +calendar.selectDay){
                                        return taskFilter
                                    }
                                }).map(task=><TodayItem task={task}/>)
                            }

                        </MDBCardBody>
                    </MDBCard>
                </MDBCol>
                <MDBCol>
                    <MDBCard className="mb-6">
                        <MDBCardBody className="text-center">
                            <p className="mb-1">Message</p>
                            {
                                calendar.messages.map(message=><MessageItem
message={message}/>)
                            }
                        </MDBCardBody>
                    </MDBCard>
                </MDBCol>
            </MDBRow>
```

```
            </MDBCol>
            <MDBCol md={8} className="d-flex justify-content-center align-items-center flex-column mx-lg-5">
              <div className ='d-flex justify-content-center align-items-center flex-row'>
                <div className="d-flex justify-content-md-start">
                  <Form.Select aria-label="Select month" onChange={(e)=>{
                    calendar.setCurrentMonthsName(e.target.value)
                    setCountDay(Number(e.target.id))
                    if(calendar.currentMonthsName === 'January'){
                      calendar.setCurrentMonth(1)
                    }
                    if(calendar.currentMonthsName === 'February'){
                      calendar.setCurrentMonth(2)
                    }
                    if(calendar.currentMonthsName === 'March'){
                      calendar.setCurrentMonth(3)
                    }
                    if(calendar.currentMonthsName === 'April'){
                      calendar.setCurrentMonth(4)
                    }
                    if(calendar.currentMonthsName === 'May'){
                      calendar.setCurrentMonth(5)
                    }
                    if(calendar.currentMonthsName === 'June'){
                      calendar.setCurrentMonth(6)
                    }
                    if(calendar.currentMonthsName === 'July'){
                      calendar.setCurrentMonth(7)
```

```
                    }
                    if(calendar.currentMonthsName === 'August'){
                        calendar.setCurrentMonth(8)
                    }
                    if(calendar.currentMonthsName === 'September'){
                        calendar.setCurrentMonth(9)
                    }
                    if(calendar.currentMonthsName === 'October'){
                        calendar.setCurrentMonth(10)
                    }
                    if(calendar.currentMonthsName === 'November'){
                        calendar.setCurrentMonth(11)
                    }
                    if(calendar.currentMonthsName === 'December'){
                        calendar.setCurrentMonth(12)
                    }
                }
                }>
                    <option >{calendar.currentMonthsName ?
calendar.currentMonthsName : 'Select month'}</option>
                    <option value="January" id ='1' onClick={()=>{
                        calendar.setCurrentMonth('1')
                        console.log(calendar.currentMonth)
                    }
                    }>January</option>
                    <option value="February"  id='2'onClick={()=>{
                        calendar.setCurrentMonth('2')
                        console.log(calendar.currentMonth)
                    }
```

```
}>February</option>
<option value="March"  id='3' onClick={()=>{
   calendar.setCurrentMonth('3')
   console.log(calendar.currentMonth)
}
}>March</option>
<option value="April"  id='4' onClick={()=>{
   calendar.setCurrentMonth('4')
   console.log(calendar.currentMonth)
}
}>April</option>
<option value="May" id='5' onClick={()=>{
   calendar.setCurrentMonth('5')
   console.log(calendar.currentMonth)
}
}>May</option>
<option value="June" id='6' onClick={()=>{
   calendar.setCurrentMonth('6')
   console.log(calendar.currentMonth)
}
}>June</option>
<option value="July" id='7' onClick={()=>{
   calendar.setCurrentMonth('7')
   console.log(calendar.currentMonth)
}
}>July</option>
<option value="August" id='8' onClick={()=>{
   calendar.setCurrentMonth('8')
```

```
            console.log(calendar.currentMonth)
        }
        }>August</option>
        <option value="September" id='9' onClick={()=>{
            calendar.setCurrentMonth('9')
            console.log(calendar.currentMonth)
        }
        }>September</option>
        <option value="October" id='10' onSelect={()=>{
            calendar.setCurrentMonth('10')
            console.log(calendar.currentMonth)
        }
        }>October</option>
        <option value="November" id='11' onClick={()=>{
            calendar.setCurrentMonth('11')
            console.log(calendar.currentMonth)
        }
        }>November</option>
        <option value="December" id='12' onClick={()=>{
            calendar.setCurrentMonth('12')
            console.log(calendar.currentMonth)
        }
        }>December</option>
    </Form.Select>
</div>
{/*<div>*/}
{/*    <DatePicker onChange={onChange} value={values} />*/}
{/*</div>*/}
```

```
                </div>

                <MDBRow md={8} className='d-flex'>
                    {
                        calendar.monthsDays.map(monthDay=><DayItem
day={monthDay.id}/>)
                    }
                </MDBRow>

            </MDBCol>
{/*                 <MDBCol md={2} className="d-flex justify-content-center
align-items-center">*/}
{/*3*/}
{/*                 </MDBCol>*/}
        </MDBRow>
        <AddCase show={getVisible} onHide={()=>setGetVisible(false)}>

        </AddCase>
        <DayCase show={calendar.dayVisible}
onHide={()=>calendar.setDayVisible(false)}>

        </DayCase>
        <ChooseType show={calendar.typeVisible}
onHide={()=>calendar.setTypeVisible(false)}>

        </ChooseType>
        <CreateType show={calendar.createVisible}
onHide={()=>calendar.setCreateVisible(false)}>

        </CreateType>
```

```
            <Test show={testVisible} onHide={()=>setTestVisible(false)}/>
        </MDBContainer>
    </section>

  );
});

export default Main;
```

Навігатор – AppRouter.js

```
const AppRouter = () => {
  const {user} = useContext(Context)
  return (
    <Switch>
      {user.isAuth && authRoutes.map(({path,Component})=>
        <Route key = {path} path={path} component={Component} exact/>
      )}
      {publicRoutes.map(({path,Component})=>
        <Route key = {path} path={path} component={Component} exact/>
      )}
      <Redirect to={MAIN_ROUTE} />
    </Switch>
  );
};

export default AppRouter;
```

DayItem.js

```
const DayItem = observer(({day}) => {
```

```
const {calendar} = useContext(Context)
const [color,setColor] = useState('green')
let nowDate = new Date()
useEffect(()=>{
    console.log(calendar.currentMonth)
    console.log(nowDate.getMonth())
    if(+calendar.currentMonth < +nowDate.getMonth()){
        setColor('grey')
    }
    else if(+calendar.currentMonth > +nowDate.getMonth()){
        setColor('green')
    }
    else if(+calendar.currentMonth === +nowDate.getMonth()){
        if(+day < +nowDate.getDate()){
            setColor('grey')
        }
        else if((+day > +nowDate.getDate())){
            setColor('green')
        }
        else{
            setColor('red')
        }

    }
},[calendar.currentMonthsName])

return (
    <MDBCol md={2} onClick={()=>{
```

```
            calendar.setSelectDay(day)
            calendar.setTypeVisible(true)


        }
        } style={{cursor:"pointer"}}>
            <MDBCard style={{fontSize:24, color:"black",
backgroundColor:`${color}`, width:100,height:100}} border={"black"}
className='mt-4 justify-content-center align-items-center'>
                {day}
            </MDBCard>
        </MDBCol>
    );
});

export default DayItem;
```

ChooseType.js – вибір Пріоритету на сьогодні

```
const ChooseType = ({show,onHide}) => {

    const {daily} = useContext(Context)
    const {calendar} = useContext(Context)
    return (
        <>
            <MDBModal
                staticBackdrop
                tabIndex="-1"
                show={show}
                onHide={onHide}
            >
```

```
            <MDBModalDialog>
              <MDBModalContent>
                <MDBModalHeader>
                  <MDBModalTitle>Choosing type...</MDBModalTitle>
                </MDBModalHeader>
                <MDBModalBody>
                  {
                    daily.type.map(type=><MDBBtn className='m-2'
onClick={()=>
                    {

                      daily.setCurrType(type.name)
                      calendar.setDayVisible(true)
                      calendar.setTypeVisible(false)
                    }
                    }
                    >{type.name}</MDBBtn>)
                  }
                </MDBModalBody>
                <MDBModalFooter>
                  <MDBBtn color="secondary" onClick={()=>{
                    onHide()
                  }}>
                    Close
                  </MDBBtn>
                  <MDBBtn onClick={()=>{
                    calendar.setCreateVisible(true)
                    calendar.setTypeVisible(false)
                  }
```

```
                    }>Create new type</MDBBtn>
                </MDBModalFooter>
            </MDBModalContent>
        </MDBModalDialog>
    </MDBModal>
  </>
 );
};
```

Створення пріоритету – CreateType.js

```
const CreateType = ({show,onHide}) => {
  const {daily} = useContext(Context)
  const {calendar} = useContext(Context)
  const [newCase, setNewCase] = useState({name:''})
  return (
    <>
      <MDBModal
        staticBackdrop
        tabIndex="-1"
        show={show}
        onHide={onHide}
      >
        <MDBModalDialog>
          <MDBModalContent>
            <MDBModalHeader>
              <MDBModalTitle>Creating type...</MDBModalTitle>
            </MDBModalHeader>
```

```
                <MDBModalBody>
                  <Form>
                    <Form.Control
                      value={newCase.name}
                      onChange={e=>setNewCase({...newCase, name:
e.target.value})}
                      className='mt-3'
                      placeholder={"Enter case name"}
                    />
                  </Form>
                </MDBModalBody>
                <MDBModalFooter>
                  <MDBBtn color="secondary" onClick={()=>{
                    onHide()
                  }}>
                    Close
                  </MDBBtn>
                  <MDBBtn onClick={()=>{
                    daily.setType(newCase)
                    calendar.setCreateVisible(false)
                    calendar.setTypeVisible(true)
                  }
                  }>Create</MDBBtn>
                </MDBModalFooter>
              </MDBModalContent>
            </MDBModalDialog>
          </MDBModal>
        </>
    );
```

```
};

export default CreateType;
```

DayCase.js – список планів на сьогодні

```
const DayCase = ({show,onHide}) => {
    const {calendar} = useContext(Context)
    const {daily} = useContext(Context)
    return (
        <>
            <MDBModal
                staticBackdrop
                tabIndex="-1"
                show={show}
                onHide={onHide}
            >
                <MDBModalDialog>
                    <MDBModalContent>
                        <MDBModalHeader className='d-flex justify-content-center
align-items-center'>
                            <MDBModalTitle className='d-flex justify-content-center
align-items-center'>{calendar.selectDay + " " + calendar.currentMonthsName + " "
+ daily.currType}</MDBModalTitle>
                        </MDBModalHeader>
                        <MDBModalBody>
                            {
                                daily.plans.filter(taskFilter=>{
                                    if(+taskFilter.day === +calendar.selectDay){
                                        if(taskFilter.type === daily.currType){
                                            return taskFilter
```

```
                        }
                    }

                }).map(task=><TaskItem task={task}/>)
            }

        </MDBModalBody>
        <MDBModalFooter>
            <MDBBtn color="secondary" onClick={()=>{
                onHide()
            }}>
                Close
            </MDBBtn>
        </MDBModalFooter>
    </MDBModalContent>
</MDBModalDialog>
</MDBModal>
</>
);
};

export default DayCase;
```

Index.js

```
const express = require('express')
require('dotenv').config()
const sequelize = require('./db.js')
const models = require('./models/models.js')
const cors = require('cors')
```

```
const router = require('./routes/index')
// const fileUpload = require('express-fileupload')
const errorHandler = require('./middleware/ErrorHandlingMiddleware')
const path = require('path')
const PORT = process.env.PORT || 5000

const app=express()
app.use(cors())
app.use(express.json())
app.use(express.static(path.resolve(__dirname,'static')))
// app.use(fileUpload({}))
app.use('/api',router)
app.use(errorHandler)

const start = async ()=>{
    try{
        await sequelize.authenticate()
        await sequelize.sync()
        app.listen(PORT,()=>console.log(`Server started on port ${PORT}`))
    }catch (e){
        console.log(e)
    }
}
start()
```

db.js

```
const {Sequelize} = require('sequelize')
module.exports = new Sequelize(
```

```
    process.env.DB_NAME,
    process.env.DB_USER,
    process.env.DB_PASSWORD,
    {
        dialect:'postgres',
        host:process.env.DB_HOST,
        port:process.env.DB_PORT
    }
)
```

Routes:

Index.js

```
const Router = require('express')
const router=new Router()
const taskRouter = require('./taskRouter')
const userRouter = require('./userRouter')
const messageRouter = require('./messageRouter')



router.use('/user',userRouter)
router.use('/task',taskRouter)
router.use('/message',messageRouter)

module.exports = router
```

taskRouter.js

```
const Router = require('express')
const router=new Router()
const taskController = require('../controllers/taskControllers')
```

```
const checkRole = require('../middleware/checkRoleMiddleware')

router.post('/',taskController.create)
router.get('/:userId',taskController.getAll)
// router.post('/update',taskController.update)

module.exports = router
```

userRouter.js

```
const Router = require('express')
const router=new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration',userController.registration)
router.post('/login',userController.login)
router.get('/auth',authMiddleware,userController.check)
router.get('/user',userController.getAll)

module.exports = router
```

ApiError.js – обробник помилок

```
class ApiError extends Error{
    constructor(status,message) {
        super();
        this.status = status
        this.message = message
    }
    static badRequest(message){
        return new ApiError(404,message)
```

```
    }
    static internal(message){
        return new ApiError(500,message)
    }
    static forbidden(message){
        return new ApiError(403,message)
    }
}
module.exports = ApiError
```

Controllers:

taskController.js

```
const {User} = require("../models/models");
const {Task} = require("../models/models");

class TaskController{
    async create(req,res){
        const {name,userId,type,hours,minute,day,months} = req.body
        const task = await Task.create({name,userId,type,hours,minute,day,months})
        return res.json(task)
    }
    async getAll(req,res) {
        const {userId} = req.params
        const task = await Task.findAll({
            where: {userId}
        })
        return res.json(task)
    }
```

```
}
module.exports = new TaskController()


   {
      public int Id { get; set; }
      public string Name { get; set; }
      public string Surname { get; set; }
      public string Login { get; set; }
      public string Password { get; set; }

      public User(string name, string surname, string login, string password)
      {
         Name = name;
         Surname = surname;
         Login = login;
         Password = password;
      }

      public bool IsEmployee()
      {
         if (this is Employee)
            return true;
         else
            return false;
      }
    }

public class Employee : User
   {
```

```
        public List<Tasks> Tasks { get; set; }
        public List<RecordPrograms> RecordsPrograms { get; set; }
        public List<WorkDay> WorkDays { get; set; }

        public Employee(string name, string surname, string login, string password)
            : base(name, surname, login, password) { }
    }

public class Manager : User
    {
        public Manager(string name, string surname, string login, string password)
            : base(name, surname, login, password) { }
    }

public class RecordPrograms
    {
        public int Id { get; set; }
        public DateTime DateTime { get; set; }
        public List<ProgramName> ProgramNames { get; set; }
        public int EmployeeId { get; set; }
        public Employee Employee { get; set; }
        public RecordPrograms() { }
        public RecordPrograms(DateTime dateTime, List<ProgramName>
programNames, int employeeId)
        {
            DateTime = dateTime;
            ProgramNames = programNames;
            EmployeeId = employeeId;
        }
    }
```

```
public class ProgramName
   {
      public int Id { get; set; }
      public string Name { get; set; }
      [NotMapped]
      public string Productivity { get; set; }
      public int RecordId { get; set; }
      public RecordPrograms RecordPrograms { get; set; }
   }


public class Tasks
   {
      public int Id { get; set; }
      public DateTime Date { get; set; }
      public int WorkingHours { get; set; }
      public int WorkingMinutes { get; set; }
      public string WorkingFunction { get; set; }
      public int EmployeeId { get; set; }
      public Employee Employee { get; set; }
      public Tasks() { }


      public Tasks(DateTime date, int hours, int minutes, string function, int
employeeId)
      {
         Date = date;
         WorkingHours = hours;
         WorkingMinutes = minutes;
         WorkingFunction = function;
         EmployeeId = employeeId;
      }
```

```
    }

public class WorkDay
    {
        public int Id { get; set; }
        public DateTime Date { get; set; }
        public int Minutes { get; set; }
        public int EmployeeId { get; set; }
        public Employee Employee { get; set; }

        public WorkDay() { }
        public WorkDay(DateTime date, int minutes, int employeeId)
        {
            Date = date;
            Minutes = minutes;
            EmployeeId = employeeId;
        }
    }

public class ApplicationContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<Employee> Employees { get; set; }
        public DbSet<Manager> Managers { get; set; }
        public DbSet<Tasks> Tasks { get; set; }
        public DbSet<WorkDay> WorkDays { get; set; }
        public DbSet<ProgramName> ProgramNames { get; set; }
        public DbSet<RecordPrograms> RecordsPrograms { get; set; }

        public ApplicationContext()
```

```
        {
            Database.EnsureCreated();
        }
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=timeMa
nagementDb;Trusted_Connection=True");
        }
    }


public class AuthorizationController
    {
        public User User { get; private set; }
        public List<User> AllUsers { get; private set; } = new List<User>();
        public AuthorizationController()
        {
            LoadAllUsers();
        }

        public async void LoadAllUsers()
        {
            using (ApplicationContext db = new ApplicationContext())
            {
                await Task.Run(() =>
                {
                    AllUsers = db.Users.ToList();
                });
            }
        }
```

```
        public bool SearchUser(string login, string password)
        {
            foreach(var user in AllUsers)
            {
                if(user.Login == login && user.Password == password)
                {
                    User = user;
                    return true;
                }
            }
            throw new Exception("Невірний логін або пароль!");
        }
    }

public class EmployeeController
    {
        public Employee Employee { get; private set; }

        public EmployeeController(Employee emp)
        {
            Employee = emp;
        }
        public void AddNewTask(int hours, int minutes, string functions)
        {
            if (!StringIsValid(functions) || (hours == 0 && minutes == 0))
                throw new Exception("Некоректно введені дані!");

            using (ApplicationContext db = new ApplicationContext())
            {
```

```
            Tasks newTask = new Tasks(DateTime.Now, hours, minutes, functions,
this.Employee.Id);
            db.Tasks.Add(newTask);
            db.SaveChanges();
        }
    }

    public async void AddTimeToWorkDay(int hours, int minutes)
    {
        await Task.Run(() =>
        {
            int allMinutes = (hours * 60) + minutes;
            if (allMinutes >= 1)
            {
                using (ApplicationContext db = new ApplicationContext())
                {
                    WorkDay workDay = db.WorkDays.FirstOrDefault(w =>
w.Date.Date == DateTime.Now.Date &&
                            w.EmployeeId == Employee.Id);
                    if (workDay != null)
                    {
                        workDay.Minutes += allMinutes;
                        db.SaveChanges();
                    }
                    else
                    {
                        WorkDay newDay = new WorkDay(DateTime.Now.Date,
allMinutes, Employee.Id);
                        db.WorkDays.Add(newDay);
                        db.SaveChanges();
                    }
```

```
            }
        }
    });
}

public async void AddNewRecordPrograms()
{
    await Task.Run(() =>
    {
        Process[] a = Process.GetProcesses();
        List<ProgramName> programNames = new List<ProgramName>();
        foreach (Process b in a)
        {
            if (StringIsValid(b.MainWindowTitle))
                programNames.Add(new ProgramName() { Name =
b.MainWindowTitle });
        }
        RecordPrograms recordPrograms = new
RecordPrograms(DateTime.Now, programNames, Employee.Id);
        using (ApplicationContext db = new ApplicationContext())
        {
            db.RecordsPrograms.Add(recordPrograms);
            db.SaveChanges();
        }
    });
}

public void ChangeLoginAndPassword(string login, string password, string
newLogin,
    string newPassword, string repeatPassword)
{
```

```
        if (!StringIsValid(login, password, newLogin, newPassword,
repeatPassword) ||
            !StringNotContainsUkrChars(login, password, newLogin, newPassword,
repeatPassword)||
            newPassword != repeatPassword)
            throw new Exception("Некоректно введені дані!");

        if (login != Employee.Login || password != Employee.Password)
            throw new Exception("Невірний логін або пароль!");

        using (ApplicationContext db = new ApplicationContext())
        {
            Employee.Login = newLogin;
            Employee.Password = newPassword;
            db.Employees.Update(Employee);
            db.SaveChanges();
        }
    }

    public static bool StringIsValid(params string[] str)
    {
        for (int i = 0; i < str.Length; i++)
        {
            if (String.IsNullOrEmpty(str[i]))
                return false;
        }
        return true;
    }
    public static bool StringNotContainsUkrChars(params string[] str)
    {
```

```
            Regex reg = new Regex(@"^([^а-яА-Я]+)$");
            for(int i = 0; i < str.Length; i++)
            {
                if (!reg.IsMatch(str[i]))
                    return false;
            }
            return true;
        }
    }

public class ManagerController
    {
        public List<Employee> AllEmployees { get; private set; }
        public string[] productivityProgramNames = new string[]
        {
            "visual studio",
            "opera",
            "chrome",
            "sublime text",
            "time management",
            "photoshop",
            "teams",
            "монитор ресурсів",
        };
        public ManagerController()
        {
            LoadAllEmployees();
        }

        private void LoadAllEmployees()
```

```
        {
            using (ApplicationContext db = new ApplicationContext())
            {
                AllEmployees = db.Employees.ToList();
            }
        }

        public List<Tasks> EmployeeTasks(string name_surname, DateTime date)
        {
            List<Tasks> tasks;
            using (ApplicationContext db = new ApplicationContext())
            {
                tasks = db.Tasks.Where(t => t.Employee.Name + " " +
t.Employee.Surname == name_surname
                                    && t.Date.Date == date.Date).ToList();
            }
            return tasks;
        }

        public List<ProgramName> EmployeeRecordsProgramsName(string
name_surname, DateTime date, out float percOfEmplProductivity)
        {
            List<ProgramName> programNames;
            int allProgramsCount = 0;
            int productivityProgramsCount = 0;
            using (ApplicationContext db = new ApplicationContext())
            {
                programNames = db.ProgramNames.Include(p=>p.RecordPrograms)
                            .Where(n => n.RecordPrograms.Employee.Name + " " +
                            n.RecordPrograms.Employee.Surname == name_surname &&
```

```
                    n.RecordPrograms.DateTime.Date == date.Date).ToList();
            }
            foreach (var program in programNames)
            {
                allProgramsCount++;
                foreach (var name in productivityProgramNames)
                {
                    if (program.Name.ToLower().Contains(name))
                    {
                        program.Productivity = "продуктивна";
                        productivityProgramsCount++;
                    }
                }
                if(string.IsNullOrEmpty(program.Productivity))
                    program.Productivity = "непродуктивна";
            }
            if (allProgramsCount != 0)
                percOfEmplProductivity = (productivityProgramsCount * 100) /
allProgramsCount;
            else percOfEmplProductivity = 0;


            return programNames;
        }


        public List<IGrouping<string, WorkDay>> EmployeeWorkDays(DateTime
firstDate, int numberOfDays)
        {
            DateTime lastDate = firstDate.Date.AddDays(numberOfDays);
            List<IGrouping<string, WorkDay>> workDays;
            using (ApplicationContext db = new ApplicationContext())
```

```
            {
                workDays = db.WorkDays.Include(d => d.Employee).AsEnumerable()
                    .Where(d => d.Date.Date >= firstDate && d.Date.Date <= lastDate)
                    .GroupBy(d => d.Employee.Name + " " +
d.Employee.Surname).ToList();
            }
            return workDays;
        }

        public void AddNewEmployee(string name, string surname, string login,
string password)
        {
            if(!StringIsValid(name, surname, login, password) ||
!StringNotContainsUkrChars(login, password))
                throw new Exception("Некоректно введені дані!");

            Employee employee = new Employee(name, surname, login, password);
            AllEmployees.Add(employee);
            using (ApplicationContext db = new ApplicationContext())
            {
                db.Employees.Add(employee);
                db.SaveChanges();
            }
        }
```

Додаток Г – Ілюстративний матеріал

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ВИЗНАЧЕННЯ ТА РЕАЛІЗАЦІЇ ПРІОРИТЕТНИХ ЗАВДАНЬ

Рисунок Г.1 – Титульний слайд

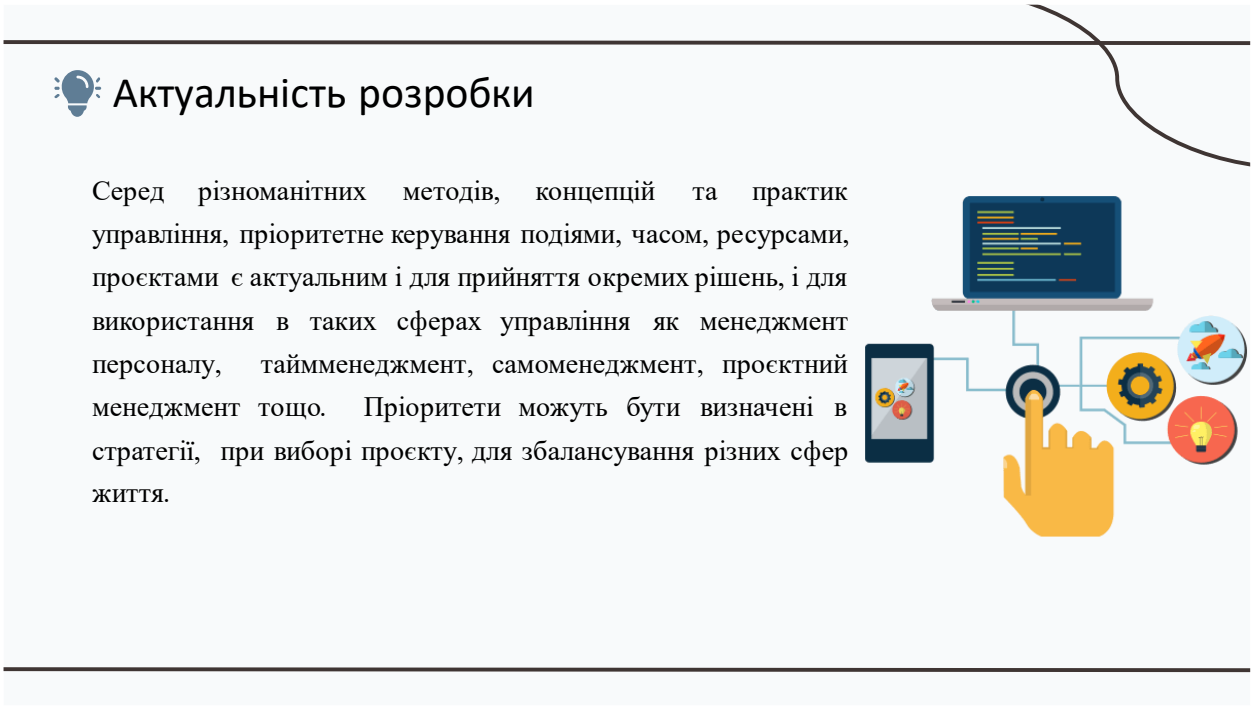


Рисунок Г.2 – Слайд «Актуальність розробки»

Мета, предмет та об’єкт дослідження

Метою роботи є розробка програмного модулю вибору пріоритетів, який дозволить підвищити ефективність управління часом, завданнями, вибору стратегії розвитку.

- Об'єкт - процес розробки програмного модуля вибору пріоритетних завдань.
- Предмет - методи та програмні засоби вибору пріоритетних завдань.

Рисунок Г.3 – Слайд «Мета, предмет та об’єкт дослідження»

Постановка задачі

- аналіз підходів вибору пріоритетів;
- визначення сфер менеджменту для вибору пріоритетів;
- визначення функціоналу програмного модулю для визначення пріоритетів;
- розробити метод і модель роботи системи вибору пріоритетів для збалансування сфер життя та узгодження особистих та робочих завдань;
- розробити метод і модель відстеження життєвого циклу створення програмного продукту та узгодження пріоритетів за таким циклом;
- виконати програмну реалізацію модулів програмного забезпечення;
- провести тестування розроблених модулів.

Рисунок Г.4 – Слайд «Постановка задачі»

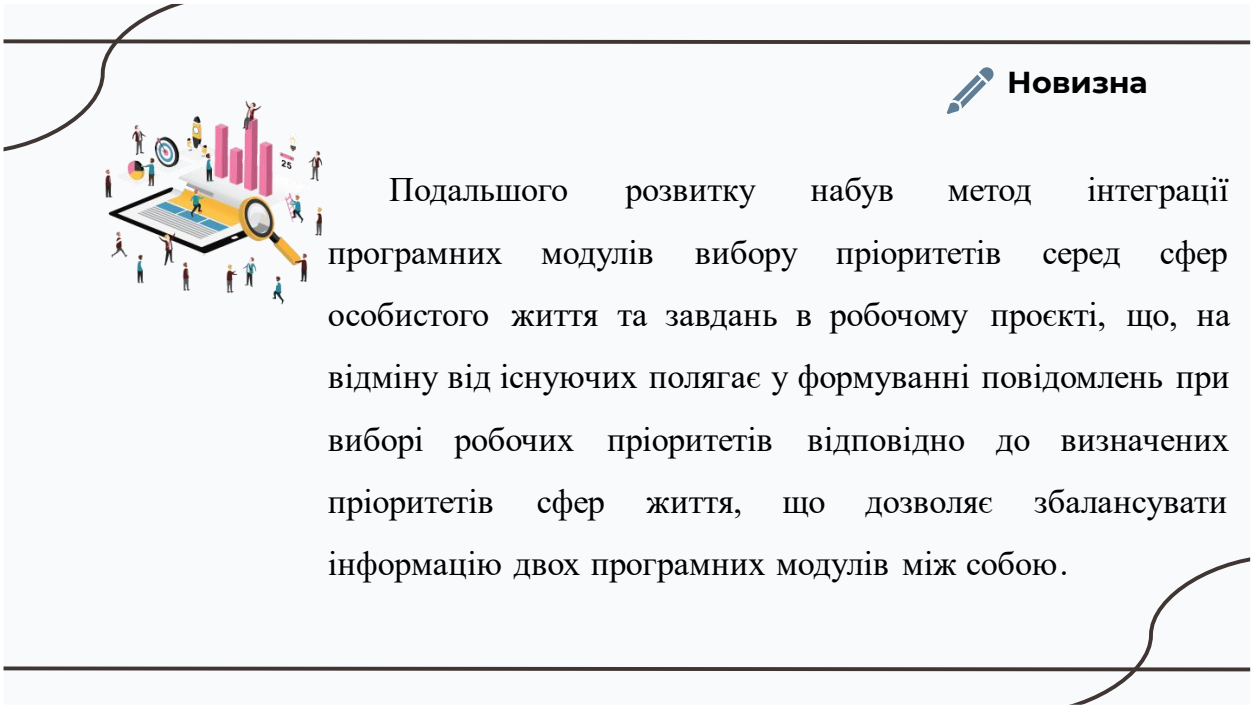


Рисунок Г.5 – Слайд «Новизна»

Рисунок Г.6 – Слайд «Практична цінність отриманих результатів»

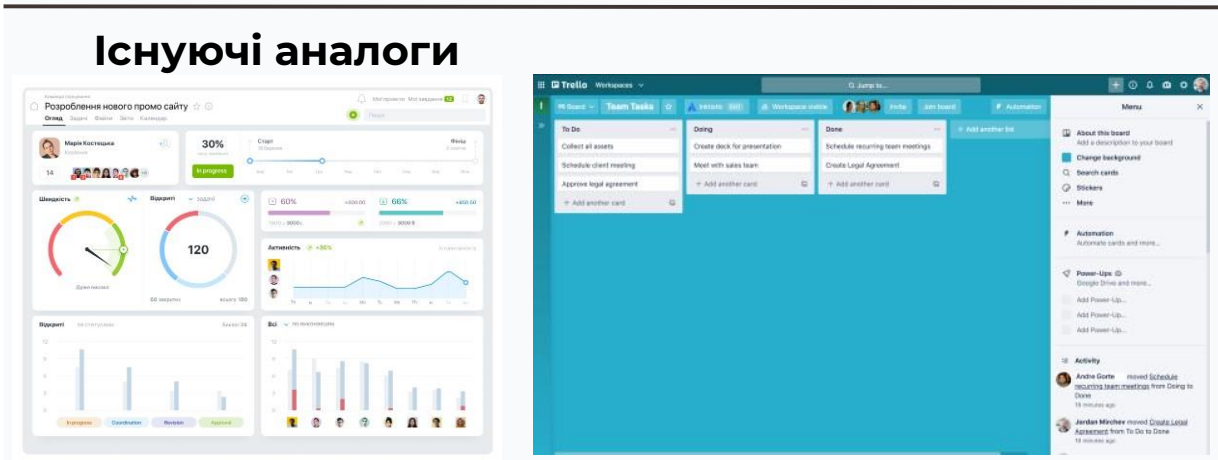


Рисунок Г.7 – Слайд «Існуючі аналоги» 1

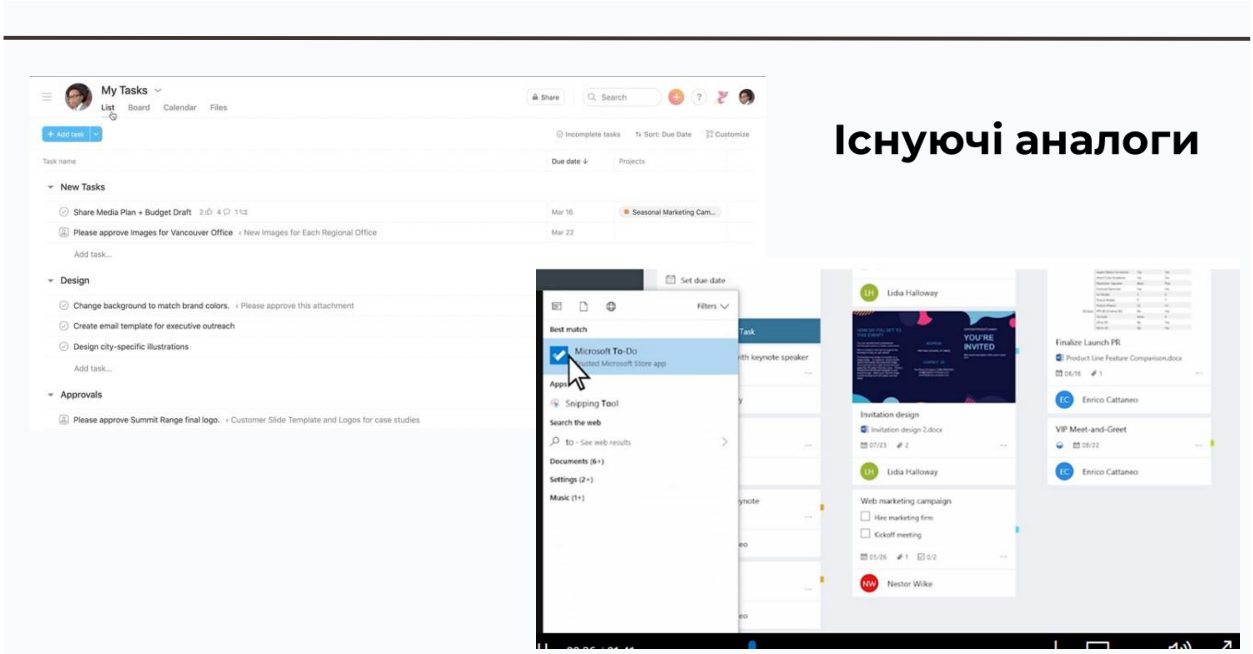


Рисунок Г.8 – Слайд «Існуючі аналоги»2

Аналіз аналогів

Критерії	Asana	Trello	Microsoft To Do	Worksection
Підтримка пріоритизації всіх сфер життя	0	0	0	0
Вибір різних методів (0,1 бал за метод)	0,4	0,2	0,3	0,3
Функціонал календаря	1	1	1	1
Багатокористувацький інтерфейс	0	0	1	1
Сумарний коефіцієнт	1	3	4	4

Рисунок Г.9 – Слайд Порівняння аналогів

Рисунок Г.10 – Слайд Колесо балансу

Рисунок Г.11 – Слайд «Метод узгодження пріоритетів Особисті сфери життя та проєкт. Моделі

Рисунок Г.12 – Слайд « Модуль та Блок-схема алгоритму роботи модулю вибору пріоритетів»

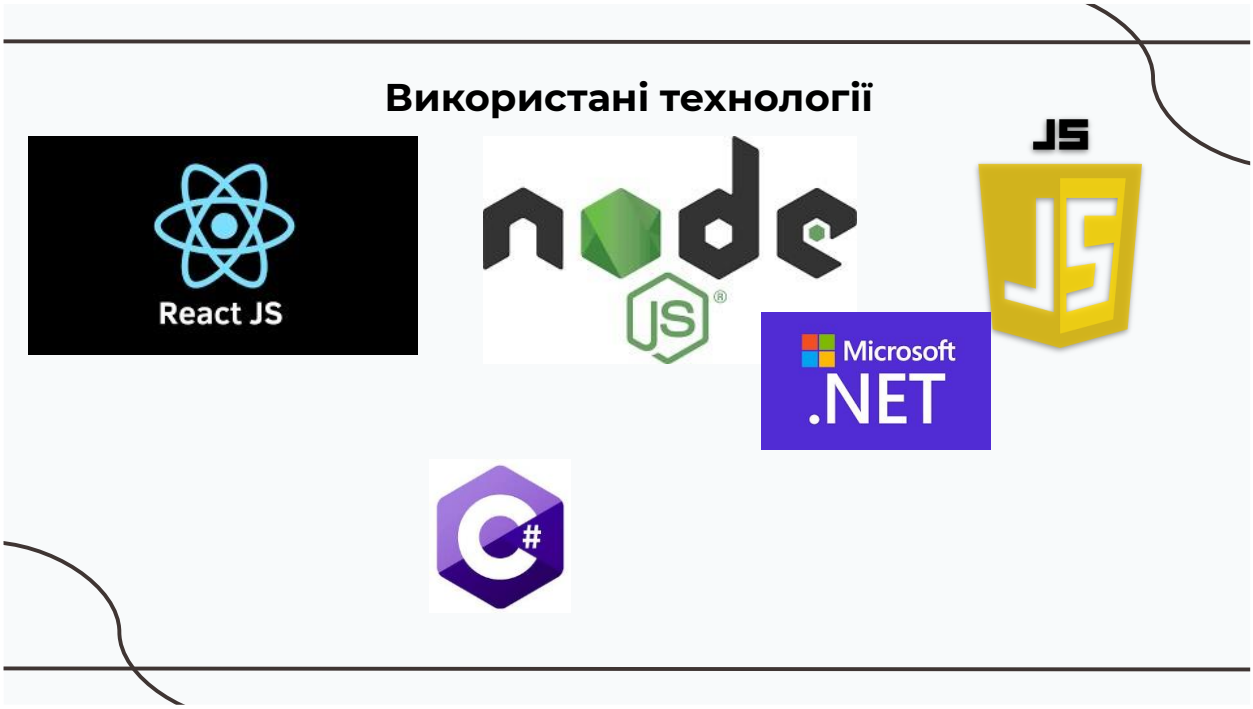


Рисунок Г.13 – Слайд « Використані технології»

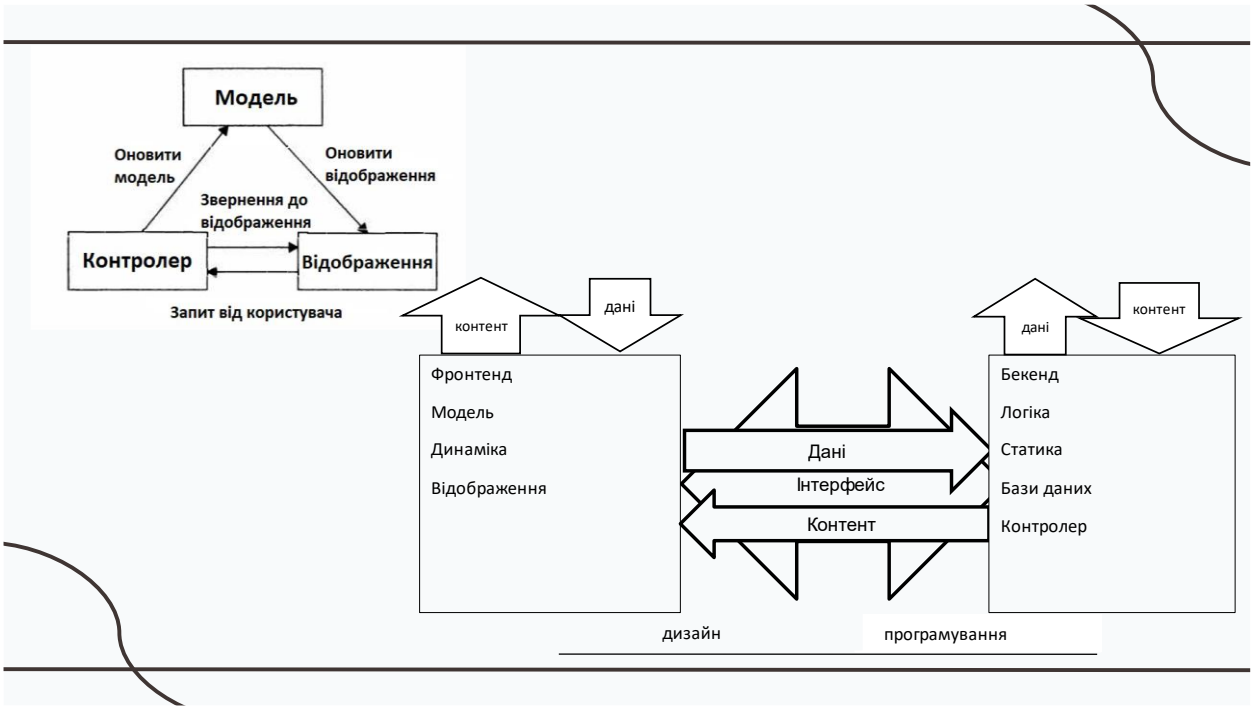


Рисунок Г.14 – Слайд «Моделі архітектури»

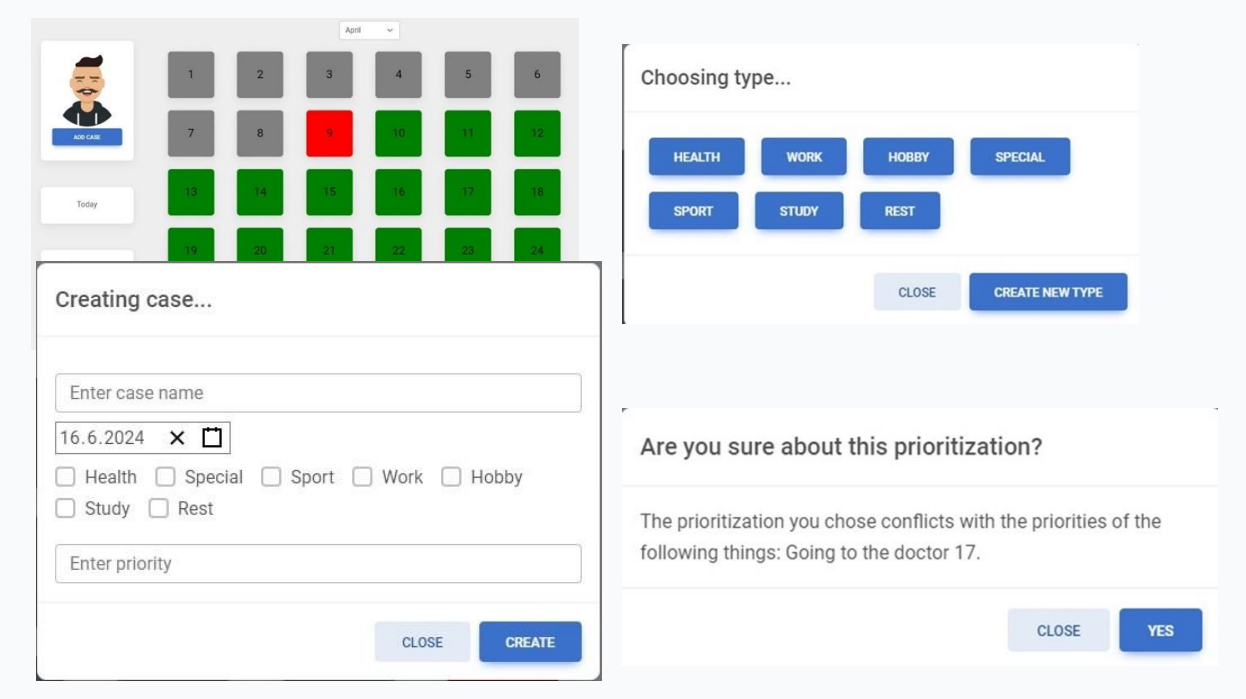


Рисунок Г.15 – Слайд « Тестування програмного модулю узгодження пріоритетів сфер життя та проєкту»

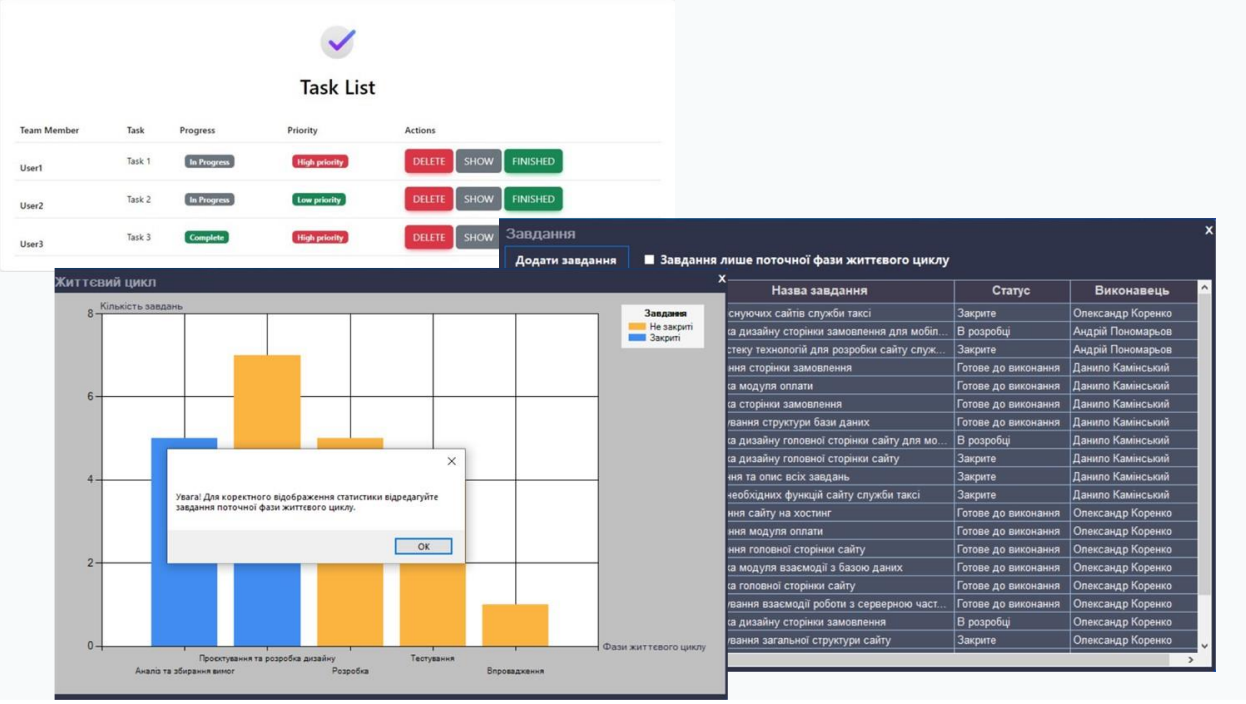


Рисунок Г.16 – Слайд « Тестування програмного модулю узгодження пріоритетів з фазами життєвого циклу»

Апробація та публікація результатів роботи

Матеріали бакалаврської дипломної роботи доповідалися на:

Інтернет-конференція «Інформаційне суспільство», випуск 89, Тернопіль, 2024.

За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференції.

Рисунок Г.17 – Слайд « Апробація та публікація результатів роботи»

Висновки

Розроблено програмні модулі для управління пріоритетами в різних сферах життя, узгодження із завданнями проєкту, управління завданнями в проєкті відповідно до фаз життєвого циклу.

Концепція управління пріоритетами повинна базуватись на теорії вибору пріоритетних завдань відносно сфер життя, важливості на поточний момент, можливостей виконання, зв'язком з визначеними етапами виконанні в проєкті, станом здоров'я, мотивації тощо.

Такі концепції потребують спеціальних розробок в області психології, управління проєктами, використання обчислювальних моделей визначення рангу пріоритетів.

Обчислювальні моделі можуть бути використані за допомогою зовнішніх відповідних застосунків або виконання обчислень за допомогою електронних таблиць.

Рекомендовані критерії для вибору пріоритетів можуть бути введені користувачем як вхідні дані для вибору.

Запропоновані програмні модулі можуть бути використані для балансування різних сфер життя пересічними

Рисунок Г.18 – Слайд «Висновки»