Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія управління проектами»

Виконала: студентка 2-го курсу, групи 3КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Остапчук І.О.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. КН
Озеранський В.С.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., доц. каф. АПТ
Барабан М.В.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«08. 12 »_2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 122 – Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
_____
(підпис)
"29" 08 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Остапчук Ірині Олександрівні

1 Тема роботи: «Інформаційна технологія управління проектами».

Керівник роботи: Озеранський Володимир Сергійович, к.т.н., доцент.

затверджені наказом вищого навчального закладу «18» 09 20 року №247

2 Строк подання студентом роботи 13.11.23

3 Вихідні дані до роботи: Скадн команди – до 8 чол; тривалість тестування – 8 тижнів; Мова програмування – об'єктно-орієнтована;

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ, обгрунтування актуальності створення інформаційної технології управління проектами; проектування інформаційної технології управління проектами; реалізація та тестування інформаційної технології управління проектами; економічна частина, висновки, перелік використаних джерел, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму функціонування інформаційної технології управління проектами; структура інформаційної технології управління проектами; приклад роботи програмного засобу.

6 Консультанти розділів проекту (роботи)

Консультанти розділів роботи в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Озеранський В.С., к.т.н., доцент каф. КН	29.08.23	10.11.23
4	Кавецький В.В., к.е.н., доц. каф. ЕПВМ	16.10.23	29.10.23

7 Дата видачі завдання 29.08.23

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обгрунтування актуальності створення інформаційної технології управління проектами	01.09.23 – 07.09.23	
2	Проектування інформаційної технології управління проектами;	08.09.23 – 24.09.23	
3	Реалізація та тестування інформаційної технології управління проектами	25.09.23 – 15.10.23	
4	Підготовка економічної частини	16.10.23 – 29.10.23	
5	Апробація та/або впровадження результатів дослідження	30.10.23 – 5.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	6.11.23 – 11.11.23	

Студентка _____ (підпис) Остапчук І.О (прізвище та ініціали)

Керівник роботи _____ (підпис) Озеранський (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Остапчук І.О. Інформаційна технологія управління проектами. Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 110с.

На укр. мові. Бібліогр.: 18 назв; рис.: 15; табл. 5.

У даній магістерській кваліфікаційній роботі був реалізован фреймворк для управління проектами. Проаналізовано та здійснено порівняння сучасних технологій управління проектами, шо використовуються в ІТ галузі при створенні програмних продуктів. На підставі цього було запропоновано інформаційну технологію, що дозволяє покращити ефективність управління командами розробників. Запропоновано удосконалені принципи управління проектами та удосконалені івенти для управління командою.

При реалізації фреймворку використано сучасну технологію Lean, як основу для покращення.

Ключові слова: інформаційна технологія, Agile, Scrum, Kanban, управління проектами.

ABSTRACT

Ostapchuk I.O. Information technology of project management. Master's thesis on specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2023. 110 p.

In Ukrainian speech Bibliography: 18 titles; Fig.: 15; table 5.

A project management framework was implemented in this master's thesis. Analysis and comparison of modern project management technologies used in the IT industry in the creation of software products were carried out. Based on this, information technology was proposed, which allows to improve the efficiency of management of development teams. Improved principles of project management and improved events for team management are proposed.

When implementing the framework, modern Lean technology was used as a basis for improvement.

Keywords: information technology, Agile, Scrum, Kanban, Lean, project management.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Будь-яка IT компанія коли-небудь задавалися питанням: «Чому важливий менеджмент проектів?»

Якщо компанія ніколи раніше не працювала з менеджером проекту, може виникнути спокуса відмовитися від витрат і керувати проектом самостійно. Зрештою, управління проектами може бути серйозною фінансовою інвестицією, то чому б просто не зробити це самостійно? Але це було б помилкою.

Хороший менеджер проекту – це клей, який скріплює проект і забезпечує якість та досягнення цілей вчасно та в рамках бюджету. За даними Інституту управління проектами (PMI), організації, які недооцінюють управління проектами, повідомляють, що в середньому на 50% більше їхніх проектів зазнають невдачі. П'ятдесят відсотків.

Невдалі проекти можуть швидко зруйнувати ваші ініціативи та затримати або навіть запобігти зростанню бізнесу.

Управління проектом – це більше, ніж просто відстеження термінів і встановлення бюджету. Хороший менеджер проекту контролює проект від початку до кінця, гарантуючи, що ініціативи та цілі стратегічно узгоджені, а проект має підтримку зацікавлених сторін.

Головний інструмент в руках менеджера проекту – це технологія розробки програмного забеспечення.

Технологія розробки ПЗ - це система (фреймворк), яка визначає порядок виконання завдань, методи оцінки та контролю.

Підходи до розробки програмного забезпечення визначають успіх проекту, адже без правильно підібраною технології складно досягти стабільності в роботі продукту, безпеки і стійкості функціональних особливостей. Керівники проекту намагаються знайти оптимальний варіант з безлічі.

Нажаль, не існує ідеальної технології, яка б вирішувала усі задачі під час процесу розробки програмного забеспечення. Кожна має свою вузконаправленність, і може підходити для одного проекту, а для іншого – ні.

Тому створення інформаційнох технології управління проектами сприятиме поліпшеню налаштування процесів при розробці ПЗ.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Розробка спеціалізованих засобів штучного інтелекту на основі інтелектуального аналізу даних та машинного навчання».

Мета та завдання дослідження. Метою даної магістерської кваліфікаційної роботи є розширення функціональних можливостей програмного забезпечення для управління проектами.

Для досягнення поставленної мети необхідно розв'язати такі завдання:

1. здійснити аналіз предметної області управління проектами;
2. здійснити аналіз існуючих технологій управління проектами та їх характеристик;
3. розробити інформаційну технологію управління проектами;
4. виконати програмну реалізацію інформаційної технології управління проектами;
5. провести тестування розробленого програмного забезпечення та проаналізувати отримані результати.
6. здійснити економічне обгрунтування доцільності розробки нової інформаційної технології

Об'єкт дослідження – процес управління проектами.

Предмет дослідження є програмні засоби управління проектами.

Методи дослідження. У роботі використані наступні методи наукових досліджень: аналізу структури інформаційної технології, теорії методів керування командами для інформаційної технології управління IT проектами, методи експерименту для проведення тестування та аналізу результатів.

Наукова новизна одержаних результатів полягає в наступному:

Розроблено інформаційну технологію управління проектами, яка відрізняється від існуючих удосконаленою інформаційною моделлю управління проектами, що

забезпечує розширення функціональних можливостей програмного забезпечення для управління проектами.

Практичне значення одержаних результатів полягає в наступному:

- розроблено алгоритм управління проектами;
- розроблено програмне забезпечення для управління проектами.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю валідації недоліків сучасних технологій, експериментальнимим тестуванням в існуючих командах рохробників.

Особистий внесок магістранта. Результати даної магістерської кваліфікаційної роботи отримані самостійно.

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» (м. Вінниця, Україна, 2023 р.). Результати дослідження впроваджено в роботу ТОВ «ІТІ».

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано 1 тезу доповіді на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» [1].

1 ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ ПРОЕКТАМИ

1.1 Аналіз предметної області управління проектами

Управління проектами – це використання специфічних знань, навичок, інструментів і прийомів для надання людям чогось цінного. Розробка програмного забезпечення для покращення бізнес-процесу, будівництво будівлі, надання допомоги після стихійного лиха, розширення збуту на новий географічний ринок – усе це приклади проектів

Усі проекти – це тимчасова спроба створити цінність за допомогою унікального продукту, послуги чи результату. Усі проекти мають початок і кінець. У них є команда, бюджет, графік і набір очікувань, яким команда повинна відповідати. Кожен проект унікальний і відрізняється від рутинних операцій – поточної діяльності організації – тому що проекти досягають завершення, коли мета досягнута.

Змінний характер роботи через технологічні досягнення, глобалізацію та інші фактори означає, що все частіше робота організовується навколо проектів, коли команди об’єднуються на основі навичок, необхідних для виконання конкретних завдань.

Керують цими проектами менеджери – люди, яких навмисно або через обставини просять переконатися, що команда проекту досягає своїх цілей. Менеджери проекту використовують багато різних інструментів, прийомів і підходів, щоб задовольнити потреби проекту.

Деякі проекти необхідні для швидкого вирішення проблем з розумінням того, що покращення будуть зроблені протягом певного періоду часу. Інші проекти мають більшу тривалість і/або виробляють продукт або інші результати, які не потребуватимуть суттєвих покращень за межами запланованого обслуговування, наприклад, шосе.

Інші будуть поєднанням обох цих типів проектів. Проектні менеджери використовують різноманітні навички та знання, щоб залучити та мотивувати інших

для досягнення цілей проекту. Також вони мають вплив на успіху проектів і дуже затребувані, щоб допомогти організаціям досягти своїх цілей.

Протягом історії людства управління проектами завжди практикувалося неформально, але воно почало з’являтися як окрема професія в середині 20-го століття, коли група далекоглядних людей з аерокосмічної, інженерної, фармацевтичної та телекомунікаційної сфер усвідомили, що світ змінюється. потрібні нові інструменти. Мотивовані потребою вирішити проблеми планування та ресурсів, пов’язаних із все більш складними проектами, вони зібралися, щоб розпочати встановлення та стандартизацію інструментів для нової професії. А в 1969 році зародився Інститут управління проектами (PMI – Project Management Institute, рис.1.1) [1].

Рисунок 1.1 – Офіційний логотип Інституту управління проектами

Сьогодні ми живемо в епоху проектної економіки (The Project Economy), де проекти є рушійною силою того, як виконується робота, реалізуються зміни та надається вартість.

У The Project Economy світове зростання рівня управління проектами доводить його цінність як:

- як визнана та стратегічна організаційна компетентність;
- як предмет навчання та виховання;
- як кар'єрний шлях.

Зараз загальновизнано, що базові знання з управління проектами можуть бути корисними для людей з різними ролями в широкому діапазоні починань. Навички

управління проектами можуть допомогти молодому студенту, який працює над науковим проектом, досягти успіху, або керівнику компанії врегулювати особисті суперечки. Ці навички можуть допомогти медсестрі впорядкувати зміни, щоб покращити час реагування пацієнтів у своїй палаті. Вони можуть допомогти IT-фахівцям розробляти інноваційне програмне забезпечення в рекордно короткі терміни або допомогти державній установі покращити послуги, які вони надають, у більш економічний спосіб [1, 2].

1.2 Технологія розробки програмного забеспечення

Перед стартом кожного проекту для менеджера наступним кроком буде з’ясувати, які технології управління проектами підходять цьому проекту і команді розробників.

Ландшафт технологій управління проектами може здатися дещо приголомшливим.

Незалежно від того, чи маєте менеджер офіційну сертифікацію з управління проектами, або навчається стати менеджером проектів з досвіду, на вибір є абсолютний вибір технологій проектів. І вони часто мають свої правила, списки, принципи та нескінченні скорочення.

Технологія управління проектами – це набір принципів і практик, які допоможуть вам організувати свої проекти для забезпечення їх оптимальної ефективності [3].

По суті, це фреймворк, який допомагає менеджеру найкращим чином керувати проектом.

Управління проектами дуже важливе для організацій і команд, але для того, щоб воно було дійсно ефективним, потрібно переконатися, що правильно зіставлена технологія управління проектами з типом команди, проектом, організацією та цілями.

Немає двох абсолютно однакових проектів (навіть якщо компанія використовує такі зручні функції, як шаблони проектів, щоб повторити свої минулі успіхи).

І якщо врахувати різні цілі, ключові показники ефективності (KPI) та методи виробництва не лише різних типів команд, але й різних типів галузей, має сенс, що не існує універсального підходу до управління проектом. Те, що найкраще працює для одного типу команди, може стати абсолютним кошмаром для іншого.

Наприклад, багато розробників програмного забезпечення почали виявляти, що традиційні методи управління проектами заважають, а не допомагають, їх робочим процесам і негативно впливають на їхню продуктивність і результати.

В результаті команди програмного забезпечення почали розробляти новий тип технології управління проектами, яка була розроблена для вирішення їхніх конкретних проблем [4, 5].

Невдовзі інші команди та галузі почали адаптувати ці нові методи управління проектами відповідно до своїх унікальних потреб і проблем. І так далі й далі, коли різні технології управління проектами переробляються та адаптуються для різних галузей та налаштовуються відповідно до конкретних випадків використання.

Існує багато факторів, які впливають на те, яка технологія управління проектами підходить для вашого проекту, команди та організації:

- Вартість і бюджет: з яким бюджетом ви працюєте? Чи можна це змінити, якщо це необхідно, чи важливо, щоб воно залишалося в цих наперед визначених межах?
- Розмір команди: скільки людей задіяно? Скільки зацікавлених сторін? Чи є ваша команда відносно компактною та самоорганізованою, чи більш розгалуженою, з потребою в більш суворому делегуванні?
- Здатність йти на ризик: чи це величезний проект із великим впливом, яким потрібно ретельно керувати, щоб досягти дуже серйозних результатів? Або це проект меншого масштабу з трохи більше простору для гри?
- Гнучкість: чи є можливість змінювати масштаби проекту під час процесу? А як щодо готового продукту?
- Хронометраж: скільки часу відведено на виконання? Вам потрібен швидкий оборот, чи важливіше, щоб у вас був прекрасний результат, незалежно від того, скільки часу це займе?

Співпраця між клієнтами/зацікавленими сторонами: наскільки клієнт/зацікавлена сторона має потребу або хоче брати участь у процесі? Наскільки ви потребуєте – чи хочете – щоб вони були залучені [8]?

1.3 Аналіз актуальних технологій управління проектами

Технологія Waterfall— це лінійний підхід до управління проектом, коли вимоги зацікавлених сторін і замовників збираються на початку проекту, а потім створюється послідовний план проекту, щоб задовольнити ці вимоги. Модель водоспаду названа так тому, що кожна фаза проекту переходить каскадом у наступну, плавно спускаючись вниз, як водоспад (рис. 1.2).

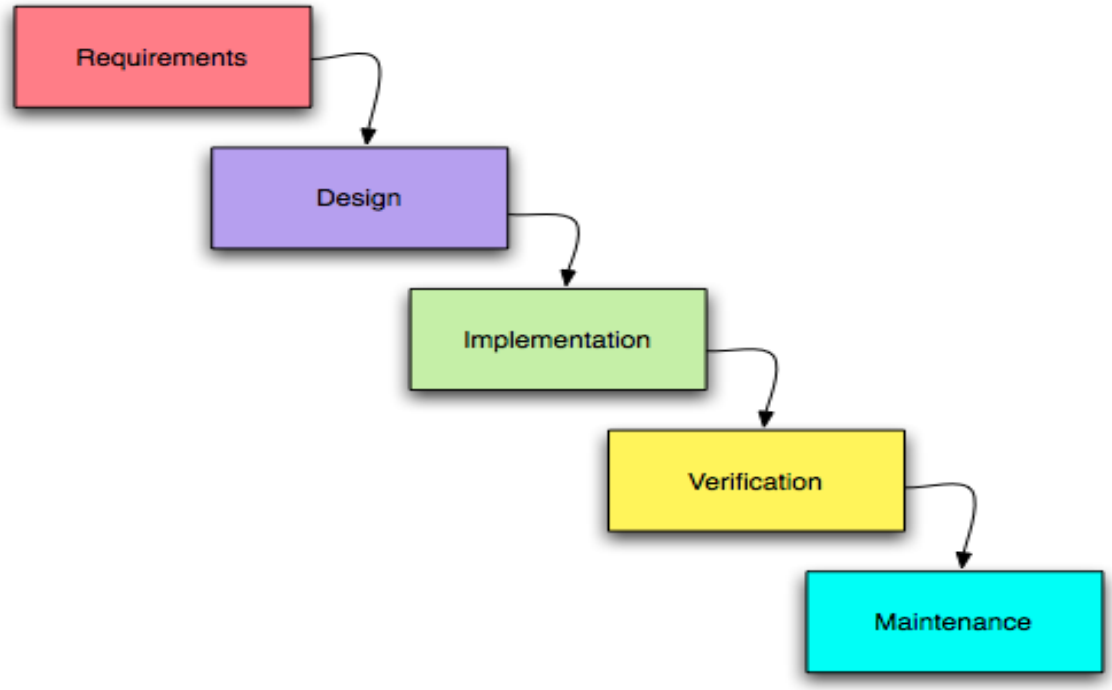


Рисунок 1.2 – Стадії каскадної технології Waterfall

Це ретельна, структурована технологія, яка існує вже давно, тому що вона працює. Деякі галузі, які регулярно використовують модель водоспаду, включають будівництво, IT та розробку програмного забезпечення. Як приклад, життєвий цикл розробки програмного забезпечення каскаду, або водоспад SDLC, широко використовується для управління проектами програмної інженерії.

Водоспадний підхід має принаймні п’ять-сім фаз, які слідують у строгому лінійному порядку, де фаза не може розпочатися, доки не буде завершена попередня фаза. Конкретні назви сходинок водоспаду відрізняються, але спочатку їх винахідник Вінстон В. Ройс визначив у такий спосіб:

Вимоги: ключовим аспектом технології водоспаду є те, що всі вимоги замовника збираються на початку проекту, що дозволяє планувати кожну наступну фазу без подальшого листування з клієнтом, поки продукт не буде завершено. Передбачається, що всі вимоги можуть бути зібрані на цьому етапі управління водоспадом.

Проектування: фазу проектування процесу водоспаду найкраще розділити на дві підфази: логічний дизайн і фізичний дизайн. Підфаза логічного проектування — це коли обговорюються та теоретично обговорюються можливі рішення. Підфаза фізичного проектування – це коли ці теоретичні ідеї та схеми перетворюються на конкретні специфікації.

Реалізація: фаза впровадження — це коли програмісти засвоюють вимоги та специфікації попередніх етапів і створюють фактичний код.

Перевірка: на цьому етапі клієнт переглядає продукт, щоб переконатися, що він відповідає вимогам, викладеним на початку проекту водоспаду. Це здійснюється шляхом відпуску готової продукції замовнику.

Технічне обслуговування: Клієнт регулярно використовує продукт на етапі технічного обслуговування, виявляючи помилки, неадекватні функції та інші помилки, що виникли під час виробництва. Виробнича група застосовує ці виправлення за необхідності, доки клієнт не буде задоволений[8].

Agile – це ітеративний підхід до управління проектами, який зосереджується на розбивці великих проектів на більш керовані завдання, які виконуються за короткі ітерації протягом життєвого циклу проекту. Команди, які використовують технологію Agile, можуть виконувати роботу швидше, адаптуватися до мінливих вимог проекту та оптимізувати свій робочий процес (рис. 1.3).

Як випливає з назви, Agile дозволяє командам бути краще оснащеними, щоб швидко змінювати напрямок і фокус. Компанії програмного забезпечення та

маркетингові агентства особливо знають про тенденцію до змін від зацікавлених сторін проекту щотижня. Технологія Agile дозволяє командам переоцінювати роботу, яку вони виконують, і коригувати з певними кроками, щоб переконатися, що в міру того, як змінюється робота та клієнтський ландшафт, змінюється і фокус для команди.

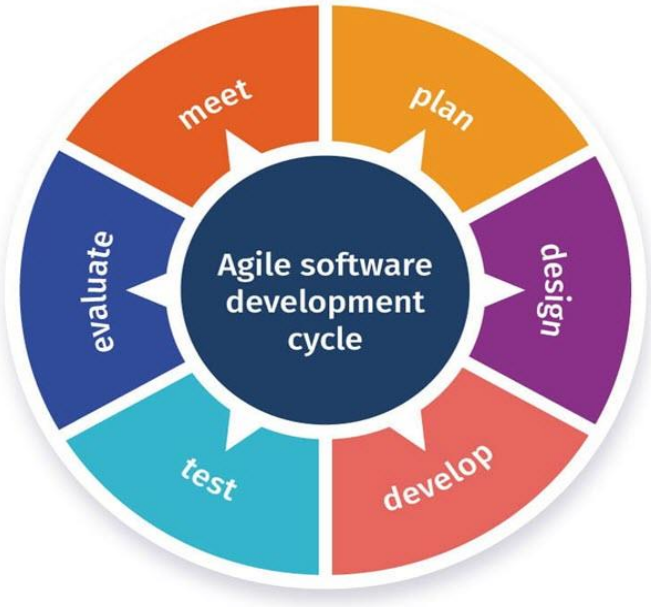


Рисунок 1.3 – Життєвий цикл Agile розробки

Якщо ви новачок в управлінні проектами Agile, на перший погляд, це може виглядати як складна і складна в управлінні система. Але, усвідомлюєте ви це чи ні, ви вже робите багато речей, яких вимагає Agile. За допомогою кількох налаштувань ви будете на шляху до коротших циклів розробки та менших, більш частих випусків продуктів.

Хто використовує Agile управління проектами?

Спочатку створений для розробки програмного забезпечення, Agile підхід до управління проектами швидко адаптується не тільки IT-командами. Маркетологи, університети, військові та навіть автомобільна промисловість також розглядають технологію Agile та інші структури Agile для доставки інноваційних продуктів у невизначених умовах. Багато організацій можуть отримати вигоду від управління проектами Agile, і його легко налаштувати та використовувати.

У світі програмного забезпечення, коли приймається рішення про створення або подальший розвиток існуючої технології, кінцевий продукт може бути важко

визначити. Agile допускає цю неоднозначність через свою гнучкість, щоб змінювати напрям проекту, коли робота рухається в майбутнє.

Хоча ви можете скористатися перевагами програмного забезпечення Agile, книг або Agile тренерів, кожна команда Agile унікальна, і розуміння основ може допомогти вам створити технологію Agile, яка працює для вас і вашої команди [9].

Agile - Маніфест окреслює 4 основні цінності та 12 керівних принципів, які служать полярною зіркою для будь-якої команди, яка використовує технологію Agile.

Чотири основні цінності Agile:

1. Люди та взаємодія важливіші за процеси та інструменти. Наскільки витонченішими стають технології, людський елемент завжди буде відігравати важливу роль у будь-якому управлінні проектами. Надто покладаючись на процеси та інструменти, ви не зможете адаптуватися до мінливих обставин.

2. Працюючий продукт важливіший за вичерпну документацію. Наскільки важливою є документація, робоче програмне забезпечення – це більше. Ця цінність полягає в тому, щоб дати розробникам саме те, що їм потрібно для виконання роботи, не перевантажуючи їх.

3. Співпраця із замовником важливіша за погодження умов контракту. Ваші клієнти є одним із ваших найпотужніших активів. Незалежно від того, чи є внутрішні або зовнішні клієнти, залучення їх протягом усього процесу може допомогти забезпечити більш ефективне задоволення кінцевого продукту їхнім потребам.

4. Готовність до змін важливіша за проходження початкового плану. Це значення є одним із найбільших відхилень від традиційного управління проектами. Історично зміни вважалися витратами, яких слід уникати. Agile дозволяє безперервно змінювати протягом життя будь-якого проекту. Кожен спринт дає можливість перегляду та корекції курсу [10].

Технології Agile можуть бути такими ж різноманітними та унікальними, як і кожна окрема команда, але 12 принципів Agile завжди повинні керувати вашими рішеннями та розробкою продукту:

– Наш найвищий пріоритет – задовольнити клієнта шляхом своєчасної та безперервної доставки програмного забезпечення (або іншого, що ви поставляєте);

– Вітаємо зміни вимог, навіть на пізніх стадіях розробки. Гнучкі процеси використовують зміни для конкурентної переваги клієнта;

– Розділяйте проект на часові відрізки, від кількох тижнів до кількох місяців, віддаючи перевагу коротшим термінам;

– Члени координаційної групи повинні працювати разом щодня протягом усього проекту;

– Створюйте проекти навколо мотивованих людей. Дайте їм необхідне середовище та підтримку та довірте їм виконання роботи;

– Бесіда віч-на-віч є найефективнішим і ефективним методом передачі інформації різним командам і всередині них;

– Кінцевий продукт є основним показником прогресу;

– Гнучкі процеси сприяють сталому розвитку. Усі зацікавлені сторони повинні мати можливість підтримувати постійний темп на невизначений термін;

– Постійна увага до технічної досконалості та гарний дизайн покращує маневреність;

– Простота – мистецтво максимізації обсягу невиконаної роботи – має важливе значення;

– Найкращі архітектури, вимоги та дизайн виходять із самоорганізуючих команд.

Через регулярні проміжки часу команда обмірковує, як стати ефективнішою, а потім відповідно налаштовує та коригує свою поведінку.

Scrum – це технологія гнучкої розробки, яка використовується при розробці програмного забезпечення на основі ітераційних та інкрементних процесів. Scrum – це адаптивна, швидка, гнучка та ефективна гнучка структура, яка призначена для надання цінності клієнту протягом усього періоду розробки проекту. Основна мета Scrum – задовольнити потреби клієнта через середовище прозорості спілкування, колективної відповідальності та постійного прогресу (рис. 1.4). Розробка починається із загального уявлення про те, що потрібно побудувати, розробки переліку характеристик, упорядкованих за пріоритетом (відставання продукту), які хоче отримати власник продукту [9].

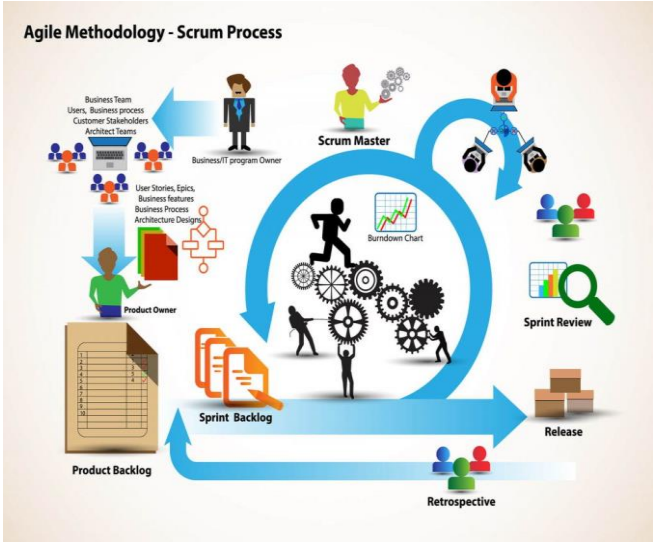


Рисунок 1.4 – Графічне відображення Scrum процесів

Scrum – це саме еволюція Agile-менеджменту. Технологія Scrum заснована на наборі чітко визначених практик і ролей, які повинні бути задіяні в процесі розробки програмного забезпечення. Це гнучка технологія, яка винагороджує застосування 12 принципів agile в контексті, узгодженому всіма членами команди продукту.

Scrum виконується у вигляді тимчасових блоків, які є короткими та періодичними, які називаються спринтами, які зазвичай тривають від 2 до 4 тижнів, що є терміном для зворотного зв'язку та роздумів. Кожен спринт є сутністю сам по собі, тобто він забезпечує повний результат, варіацію кінцевого продукту, який повинен бути доставлений клієнту з найменшими зусиллями, коли його запитують.

Відправною точкою процесу є перелік цілей/вимог, які складають план проекту. Саме клієнт проекту визначає пріоритетність цих цілей, враховуючи баланс вартості та вартості, таким чином визначаються ітерації та наступні поставки.

З одного боку, ринок вимагає якості, швидкої доставки при нижчих витратах, для чого компанія повинна бути дуже швидкою та гнучкою в розробці продуктів, щоб досягти коротких циклів розробки, які можуть задовольнити попит клієнтів, не підриваючи якість результату. . Це дуже проста в застосуванні технологія і дуже популярна завдяки швидким результатам.

Технологія Scrum використовується в основному для розробки програмного забезпечення, але інші сектори також користуються її перевагами, впроваджуючи цю

технологію в свої організаційні моделі, такі як продажі, маркетинг, відділи персоналу тощо.

У Scrum команда зосереджена на створенні якісного програмного забезпечення. Scrum Product Owner зосереджується на тому, щоб визначити, які характеристики повинен мати продукт для створення (що створювати, що ні і в якому порядку) і подолати будь-які перешкоди, які можуть перешкодити команді розробників.

Скрам-команда складається з таких ролей:

- Scrum-майстер: людина, яка керує командою, направляючи її до дотримання правил і процесів технології. Scrum-майстер керує зменшенням перешкод проекту та працює з Власником продукту, щоб максимізувати рентабельність інвестицій. Скрам-майстер відповідає за те, щоб тримати Scrum в актуальному стані, забезпечуючи коучинг, наставництво та тренінги для команд у разі потреби.
- Власник продукту (PO): є представником зацікавлених сторін і клієнтів, які використовують програмне забезпечення. Вони зосереджуються на бізнес-частині та відповідають за рентабельність інвестицій проекту. Вони передають бачення проекту команді, перевіряють Story, які потрібно включити в Беклог продукту, і регулярно визначають їх пріоритети.
- Команда: Група професіоналів з необхідними технічними знаннями, які спільно розробляють проект, виконуючи Story, які вони зобов’язуються на початку кожного спринту.

Kanban – це візуальна система для керування роботою під час її проходження через процес. Kanban візуалізує як процес (робочий процес), так і фактичну роботу, що проходить через цей процес. Мета Kanban – визначити потенційні вузькі місця у вашому процесі та виправити їх, щоб робота могла проходити через це економічно ефективно з оптимальною швидкістю або пропускною здатністю (рис. 1.5).

Рисунок 1.5 – Класична візуалізація Канбан-дошки

Технологія Канбан дотримується набору принципів і практик для управління проектами та покращення потоку роботи. Це еволюційний, не руйнівний метод, який сприяє поступовому вдосконаленню процесів організації. Якщо ви дотримуєтеся цих принципів і практик, ви зможете успішно використовувати Kanban для максимізації переваг для вашого бізнес-процесу – покращення потоку, скорочення часу циклу, підвищення цінності для клієнта з більшою передбачуваністю – усе це має вирішальне значення для будь-якого бізнесу [10].

Нижче наведено чотири основоположні принципи та шість основних практик технології Канбан:

4 Основні принципи:

- Почніть з того, що ви робите зараз: метод Kanban (надалі іменований просто Kanban) наголошує, що не потрібно відразу вносити жодних змін у ваші існуючі налаштування/процеси. Kanban необхідно застосовувати безпосередньо до поточного робочого процесу. Будь-які необхідні зміни можуть відбуватися поступово протягом певного періоду часу у зручному для команди темпі;

- Погодьтеся на поступові, еволюційні зміни: Kanban заохочує вас вносити невеликі поступові зміни, а не вносити радикальні зміни, які можуть призвести до опору в команді та організації;

- Спочатку поважайте поточні ролі, обов'язки та посади: на відміну від інших методів, Kanban не нав'язує жодних організаційних змін сам по собі. Таким чином, не потрібно вносити зміни до ваших існуючих ролей і функцій, які можуть працювати добре. Команда спільно визначить та впровадить усі необхідні зміни. Ці три принципи допомагають організаціям подолати типовий емоційний опір і страх перед змінами, які зазвичай супроводжують будь-які ініціативи щодо змін в організації;

- Заохочуйте акти лідерства на всіх рівнях: Kanban заохочує постійне вдосконалення на всіх рівнях організації і говорить про те, що дії лідерства не повинні походити лише від керівників вищого рівня. Люди на всіх рівнях можуть надавати ідеї та демонструвати лідерство для впровадження змін, щоб постійно вдосконалювати спосіб надання своїх продуктів і послуг.

6 основних практик методу Канбан:

- Візуалізуйте хід роботи: це фундаментальний перший крок до прийняття та впровадження методу Канбан. Вам потрібно візуалізувати – на фізичній дошці або електронній дошці Kanban, етапи процесу, які ви зараз використовуєте для надання своєї роботи чи послуг. Залежно від складності вашого процесу та вашої робочої суміші (різних типів робочих елементів, над якими ви працюєте та надаючи), ваша дошка Kanban може бути дуже простою або дуже складною. Після того, як ви візуалізуєте свій процес, ви можете візуалізувати поточну роботу, яку виконуєте ви та ваша команда. Це може бути у вигляді наклейок або карток різних кольорів, які позначають різні класи обслуговування, або можуть бути просто різним типом робочих предметів. (У SwiftKanban кольори означають різні типи робочих елементів!) Якщо ви вважаєте, що це може бути корисно, ваша дошка Kanban може мати різні доріжки для плавання, по одній для кожного класу обслуговування або для кожного типу робочого елемента. Однак спочатку, щоб все було просто, ви також можете мати лише одну доріжку для плавання, щоб керувати всією своєю роботою – і згодом робити будь-який редизайн дошки;

- Limit WIP (Work in Progress): Обмеження незавершеного виробництва (WIP) є основоположним для впровадження Kanban – «системи витягування». Обмежуючи WIP, ви заохочуєте свою команду спочатку завершити роботу, перш ніж братися за нову роботу. Таким чином, роботи, які зараз виконуються, мають бути завершені та відзначені як виконані. Це створює потенціал у системі, тому команда може залучати нову роботу. Спочатку може бути нелегко визначити, якими мають бути ваші обмеження WIP. Насправді, ви можете почати без обмежень WIP. Великий Дон Рейнертсен пропонує (він зробив це на одній із конференцій Lean Kanban), що ви можете почати без обмежень WIP і просто спостерігати за початковою роботою, яка виконується, коли ваша команда починає використовувати Kanban. Коли у вас буде достатньо даних, визначте обмеження WIP для кожного етапу робочого процесу (кожного стовпця вашої дошки Kanban) як рівні половині середнього WIP;

- Як правило, багато команд починають з ліміту WIP в 1–1,5 рази більше, ніж кількість людей, які працюють на певному етапі. Обмеження WIP та встановлення лімітів WIP на кожній колонці ради не тільки допомагає членам команди спочатку закінчити те, що вони роблять, перш ніж братися за нові справи, але також повідомляє клієнту та іншим зацікавленим сторонам, що існує обмежена здатність виконувати роботу для будь-якого команда – і вони повинні ретельно спланувати, яку роботу вони просять виконати команду;

- Керування потоком: Керування та покращення потоку є основою вашої системи Kanban після того, як ви впровадили перші 2 практики. Система Kanban допомагає вам керувати потоком, висвітлюючи різні етапи робочого процесу та статус роботи на кожному етапі. Залежно від того, наскільки чітко визначено робочий процес і встановлені ліміти WIP, ви спостерігатимете або плавний потік у межах WIP, або роботу, яка накопичується, коли щось затримується і починає затримувати ємність. Все це впливає на те, наскільки швидко робота проходить від початку до кінця робочого процесу (деякі люди називають це потоком створення цінностей). Kanban допомагає вашій команді аналізувати систему та вносити зміни, щоб покращити процес, щоб скоротити час, необхідний для виконання кожної частини роботи;

- Ключовим аспектом цього процесу спостереження за вашою роботою та вирішенням/усуненням вузьких місць є перегляд проміжних етапів очікування (проміжних етапів Готово) і перегляду, як довго робочі елементи залишаються на цих «стадіях передачі». Як ви дізнаєтеся, скорочення часу, витраченого на цих етапах очікування, є ключем до скорочення часу циклу. У міру того, як ви покращуєте потік, робота вашої команди стає більш гладкою та передбачуваною. Оскільки це стає більш передбачуваним, вам стає легше давати надійні зобов'язання перед клієнтом щодо того, коли ви закінчите роботу, яку для нього виконуєте. Покращення вашої здатності надійно прогнозувати час завершення є важливою частиною впровадження системи Kanban.

Зробіть політику процесу чіткою: як частина візуалізації вашого процесу, має сенс також чітко визначити та візуалізувати ваші політики (правила процесу або вказівки) щодо того, як ви виконуєте роботу, яку ви виконуєте. Формулюючи чіткі вказівки щодо процесу, ви створюєте загальну основу для розуміння всіх учасників, як виконувати будь-який тип роботи в системі. Правила можуть бути на рівні дошки, на рівні доріжок для плавання та для кожної колонки. Вони можуть бути контрольним списком кроків, які необхідно виконати для кожного типу робочого елемента, критеріїв входу-виходу для кожного стовпця або взагалі чогось, що допомагає членам команди добре керувати потоком роботи на дошці. Приклади явних політик включають визначення того, коли завдання завершено, опис окремих доріжок чи колон, хто коли тягне тощо. Політики мають бути чітко визначені та відображені зазвичай у верхній частині [11].

1.4 Життєвий цикл розробки програмного забеспечення

Планування проекту буде неможливим, якщо немає життєвого циклу проекту, про який можна говорити. Це стосується ряду заходів, які необхідно виконати або виконати для досягнення цілей проекту. Зрозуміло, проекти відрізняються за розміром, спрямованістю та складністю. Проте всі вони мають один і той же життєвий цикл [12].

Життєвий цикл проекту включає п'ять фаз. Це також часто називають традиційним управлінням проектом, яке має п’ять основних етапів процесу:

1. Ініціювання

Це передбачає визначення характеру та обсягу проекту. Для цього необхідно вивчити бізнес-середовище та зрозуміти, як воно працює. Деякі з ключових заходів на цьому етапі:

– аналіз вимог бізнесу;

– оцінка історичних та поточних даних про діяльність бізнесу, включаючи фінансові звіти та бюджети;

– визначення зацікавлених сторін та аналіз їх ролі та впливу;

– визначення потреб зацікавлених сторін;

– визначення цілей проекту.

Саме на цьому етапі часто проводяться техніко-економічні обґрунтування. Це чудові інструменти для визначення можливих варіантів, які можуть вирішити наявні проблеми та допомогти досягти мети проекту.

Також часто на цьому етапі обирають і призначають керівника проекту, а також членів проектної групи та інших робочих груп.

2. Планування

Переходимо до більш детальної фази проекту. План проекту або блок-схема готується для планування часу, графіка, витрат і розподілу ресурсів для виконання заходів у проекті. Це передбачає врахування вартості супутніх ризиків під час реалізації заходів проекту. Також на цьому етапі команда проекту отримає остаточне схвалення для продовження проекту. Заходи, що виконуються на етапі планування, включають:

– створення команди планування;

– визначення результатів проекту, включаючи цілі якості та заходи контролю;

– визначення заходів, які необхідно виконати;

– розробка структури розбивки робіт та відображення їх взаємозв’язків;

– отримання кошторису витрат на матеріали, обладнання, оплату праці та інші витрати;

– складання бюджету проекту;

– розробка графіка виконання заходів у розкладі робіт;

– ідентифікація потенційних загроз, проблем або ризиків і формулювання відповідних заходів, якщо ці загрози, проблеми або ризики виникнуть під час реалізації проекту.

У проекті немає фіксованої кількості заходів, які необхідно виконати. Деякі проекти можуть мати лише кілька завдань, тоді як інші проекти складаються з довгого списку заходів. У цьому немає нічого поганого, якщо ви стежите за метою.

3. Виконання або впровадження

Це етап впровадження, на якому виконуються або виконуються ключові дії проекту, щоб отримати результати проекту, визначені раніше. Іншими словами, план проекту зараз запущений.

Тут проводяться такі заходи:

– розподіл ресурсів на відповідні заходи або фази проекту;

– координація з ключовими зацікавленими сторонами;

– виконання завдань, зазначених у плані;

– звітування про хід проекту на регулярних зустрічах.

4. Моніторинг і контроль

На кожному кроці проекту необхідно відстежувати його прогрес. Це корисно, тому проблеми можна виявити та вирішити, щоб мінімізувати ризики. Важливу роль на цьому етапі відіграє зворотній зв'язок. Можуть бути відхилення від базової лінії або цілі, встановленої раніше. Якщо їх неможливо виправити, щоб повернути речі до початкового плану, їх слід задокументувати як відхилення.

Цей етап вимагає:

– вимірювання діяльності в міру їх проведення;

– слідкувати за змінними проекту та постійно порівнювати їх із планом;

– вживання відповідних дій для усунення проблем і вирішення проблем;

– регулярна звітність перед зацікавленими сторонами;

– документування ходу та оновлення плану, якщо є;

– огляд результатів проекту відповідно до базової лінії або цілей.

5. Закриття або завершення

На цьому етапі проект офіційно оголошується завершеним. Це відбудеться лише тоді, коли зацікавлені сторони приймуть і будуть задоволені кінцевим результатом або результатом.

Воно включає:

– випуск або доставка остаточних результатів;

– документація та архівування файлів проекту та інших відповідних документів, які використовуються та створюються протягом усього проекту;

– проведення огляду після впровадження, де обговорюються набуті уроки з метою їх застосування в майбутніх проектах.

Також необхідно офіційно повідомити всіх зацікавлених сторін про закриття проекту [13;14].

1.5 Аналіз переваг та недоліків існуючих програм-аналогів

Worksection – це програма для управління проектами (рис. 1.6). Адміністратор чи інший учасник може відслідковувати хід процесів, стан завдань та закріплені за командою робочі завдання. Функціонал та гнучкість дозволяють налаштувати свій проект відповідно до точного його стану [6].

Багато налаштувань зрозумілі, але при довгій роботі з подібними сервісами, тому потрібен час для адаптації. Існує декілька методологій, втілених у додатку, наприклад Agile з відслідковуванням чи Kanban-дошки, але доволі мало інструментів для впровадження ощадливого виробництва (рис. 1.7).

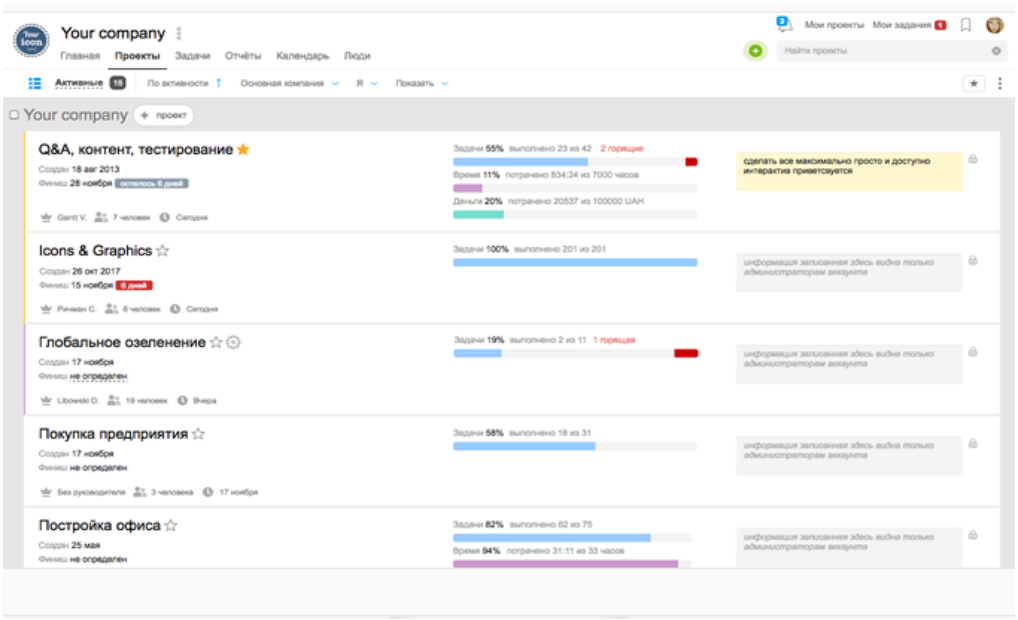


Рисунок 1.6 – Вигляд інтерфейсу Worksection для ПК

Щоб користуващись цим програмним забезпеченням необхідна підписка щомісяця чи щороку.

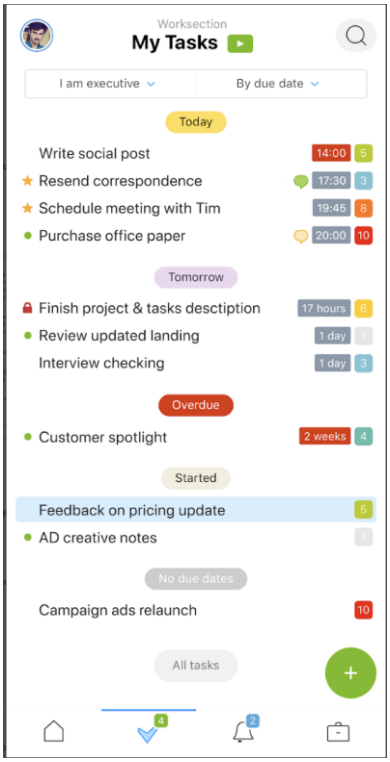


Рисунок 1.7 – Вигляд інтерфейсу мобільної версії Worksection

Існують ще кілька мобільних додатків для управління проектами, наприклад LEAN (рис. 1.8) та LeanApp. Обидва додатки зручні з точки зору мобільності, адже можна відслідковувати певні моменти одразу з телефону.

Основними недоліками є незручність інтерфейсу та мала кількість функцій. Наприклад, для роботи у LEAN потрібно мати чітке уявлення про проект чи мати консультації щодо внесення коректних даних, адже додаток не пояснює призначення багатьох функцій.

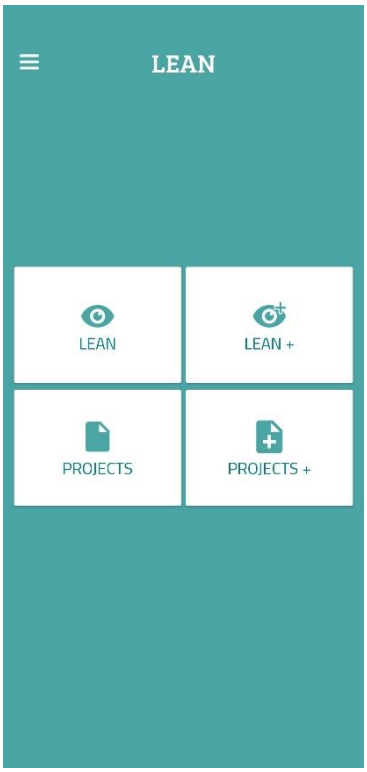


Рисунок 1.8 – Вигляд інтерфейсу мобільного додатку LEAN

Те саме стосується й іншого додатку – LeanApp (рис. 1.9). Він зручний, але іноді інтуїтивно не зрозумілий. Дозволяє швидко заповнювати форми, існує підтримка декількох системних підходів проектування, наприклад Канбан.

Але додаток не є безкоштовним, негнучкий інтерфейс й недостатня кількість функціонально потрібних речей затримують робочий процес команди.

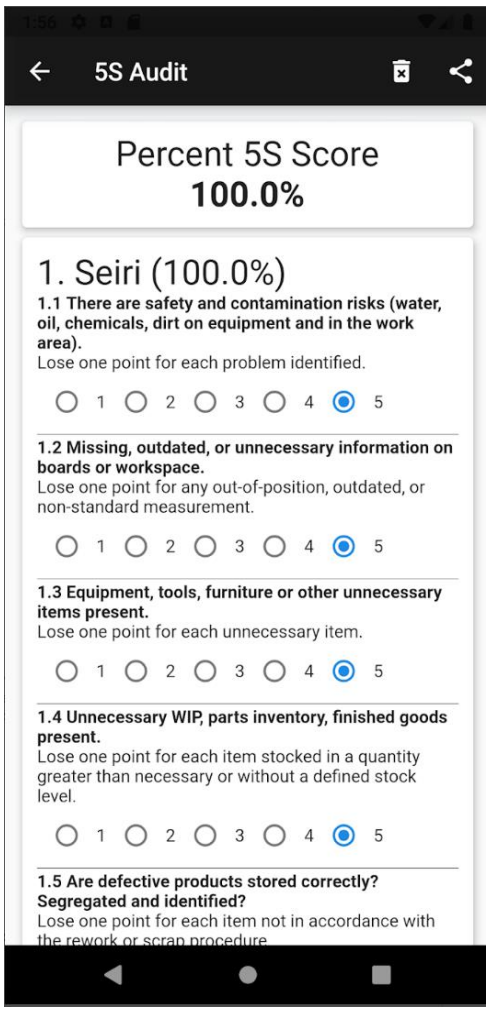


Рисунок 1.9 – Вигляд інтерфейсу мобільного додатку LeanApp

1.6 Висновок до розділу 1

В даному розділі проаналізовано предметну область управління проектами, розглянуті популярні технолігії розробки ПЗ та розібрано типовий життєвий цикл розробки. Виходячи з проведеного аналізу гнучких методів розробки програмного забезпечення можна зробити такі висновки:

- жодна з сучасних гнучких технологій не є ідеальною;
- не існує технології, яка зможе бути єфективною на проектах різних типів.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ ПРОЕКТАМИ

2.1 Вибір технології як основу для покращення

Аналіз предметної області управлення IT проектами вказує на те, що в галузі розробки программного забеспечення уснують багато різноманітних фреймворків управління. Не існує фреймворка який підходив би кожній команді розробників та будь-якому проекту.

Порівняльна характеристика технологій подана у табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика технологій Scrum та Kanban

Назва технології	Плюси	Мінуси
Scrum	– Scrum може допомогти командам швидко й ефективно завершити результати проекту – Scrum забезпечує ефективне використання часу та грошей – Великі проекти поділяються на легко керовані спринти – Розробки кодуються та перевіряються під час огляду спринту – Добре працює для швидко розвиваються проектів – Команда отримує чітку видимість через зустрічі Scrum – Scrum, будучи гнучким, сприймає відгуки клієнтів і зацікавлених сторін – Короткі спринти набагато легше вносять зміни на основі зворотного зв’язку	– Scrum часто призводить до скорочення обсягу через відсутність певної кінцевої дати – Шанси на провал проекту високі, якщо люди не дуже віддані чи співпрацюють – Застосування фреймворку Scrum у великих командах є складним завданням – Фреймворк може бути успішним лише з досвідченими членами команди – Щоденні зустрічі іноді розчаровують членів команди – Якщо хтось із членів команди покине проект у середині проекту, це може мати величезний негативний вплив на проект – Якість важко реалізувати, поки команда не пройде агресивний процес тестування

	─ Індивідуальні зусилля кожного члена команди видно під час щоденних зустрічей Scrum	
Kanban	─ Простота використання: Канбан – це дуже простий і зрозумілий підхід, що робить його практичним для ефективного застосування керівництва компанії. Вам не потрібно бути експертом, щоб працювати з підходом Kanban. ─ Сприяє постійним і стійким удосконаленням різних функцій компанії: Канбан-підхід складається не тільки з ручних інструкцій або карток, а й візуалізації результатів процесу, що полегшує аналіз роботи. Це також може виділити інші потенційно невизначені області, на яких необхідно зосередитися. ─ Адаптивність: Канбан заохочує до максимальної адаптивності, що неймовірно для більш масштабних підприємств, які потребують постійних змін. ─ Співпраця: Kanban сприяє розвитку співпраці та змушує всю команду працювати разом, щоб досягти ідеальних результатів. ─ Низькі накладні витрати: Нагляд за використанням канбан-дошки, карток та аналіз результатів є легшим у порівнянні з більшістю технологій. ─ Зменшує витрати та втрати: Канбан виділяється тим, що виділяє проблеми процесу та їх вирішення. Система Kanban покращує потік і управління, безпосередньо допомагаючи	– Не вписується в динамічне середовище: підхід Канбан передбачає стабільні та послідовні плани. – Неможливість ітерації: створення програмного забезпечення на ітераціях є основою для більшості процесів розробки, що не є невід'ємною частиною Kanban. – Відсутність часу: з кожною фазою не закріплені часові рамки, що може бути невигідним для замовника або самої компанії.

	компанії задіяти існуючі системи компанії, наприклад, «точно вчасно» (JIT) і зробити на замовлення тощо, що зменшує витрати на перевезення або зберігання.	
Waterfall	– Наявність інструкцій та правил по всьому процесу. Робота починається з докладного аналізу вимог замовника та того, як буде реалізовано проект. Плани, етапи та процеси затверджуються заздалегідь, фіксуються у документах та питань не викликають. Виконавцю потрібно просто їм слідувати. – Визначеність у термінах та бюджеті. Вартість товару та терміни здачі проекту розраховані та затверджені на самому початку і не змінюються в процесі. – Відсутність додаткових витрат на комунікацію у команді. Навіть якщо прийде новий розробник або тестувальник, зрозуміти завдання і почати роботу вийде швидко: для всіх процесів є описані правила.	– Відсутність гнучкості. Неможливо передбачити усі проблеми у проекті заздалегідь. Через жорстку послідовність етапів недоліки стануть відомі тільки в кінці проекту, доведеться робити додаткові ітерації і починати роботу наново, а це нові витрати і зайві робочі години. – Замовник не допускається до розробки та тестування. Він не може коментувати макети чи прототипи і бачить результат лише наприкінці проекту. Якщо змінилися вимоги або умови, це неможливо заздалегідь врахувати. – Проблеми спливають лише під час тестування. Зробити частину роботи і відразу протестувати чи поєднати розробку та тестування, щоб знайти вразливість, не можна. Тестування починається після закінчення розробки, тому часто недоліки виявляються надто пізно.

Scrum частіше використовують на нових проектах, де розробка буде проводитись з самого початку і до виходу в Live, в той час як Kanban популярний на саппорт проектах, де сам продукт вже створений і приносить прибуток, але існує необхідність додавати новий функціонал. Kanban також часто використовують в командах дизайнерів та бізнес аналітиків. Waterfall найменш популярний серез

проаналізованих фреймворків, тому що не підходить для швидко змінних бізнес сфер. Проте часто використовуються в медичній та воєнній сферах, тому що вимоги зафіксовані заздалегідь і не будуть змінюватись під час розробки.

Основою для розробки інформаційної технології управління проектами буде технологія Scrum [8;9].

2.2 Практичний досвід використання Scrum технологій відчизняними компаніями

Світ переживає дуже складний час. Епідемія короновірусу повністю змінила підхід к робочим процесам не тільки в IT галузі, а і в будь-якому іншому бізнесі. Локдауни та обемеження спонукають до повної зупинки усіх робочих процессів, або до переходу на віддалену роботу. Навчання в освітніх закладах стало онлайн, усюди можлива доставка на дім будь-яких товарів. І навіть ті профессії, які раніше сприймалися лише як оффлайнові, також змогли перейти в онлайн. І після такого людство точно не стане таким як раніше!

Сфера розробки ПЗ вже давно звикла до ремоуту. Розробники з України можуть спокійно працювати в команді з європейцями та американцями, та разом створювати програмні продути для східних країн, не покидаючи при цьому рідної оселі.

IT-сфера в Україні – одна з найдинамічніших і найперспективніших. Її вже давно називають локомотивом розвитку української економіки. За даними сайту DOU.UA, зараз в українській IT-індустрії працює понад 190 000 фахівців. А компаній близко 4000. І більшість з них працює с закордонними замовниками, на європейських та американських стартапах в режимі віддаленної роботи.

До епідемії розробники, дизайнери, бізнес аналітики та тестувальники все одно знаходились в одному офісі, працювали командами в одній кімнаті та постійно спілкувались. А зараз усього цього немає, кожен учасник команди теперь повинен працювати з дому. І це створює досить багато проблем, тому що жодна з технологій не адаптована під повний ремоут. Навіть в офіційних маніфестах по технологіям

Scrum (Scrum Guide) та Kanban (Kanban Manifesto) вказано, що команди мають територіально знаходитись в одній кімнаті для максимальної продуктивності.

Під час аналізу предметної області управління IT проектами, були розглянуті найпопулярніші Agile технології управління проектами по розробці програмного забеспечення. Всі ці технології чітко сформовані та задокументовані, перевірені часом і не підлягають змінам. Проте, дуже мала кількість відчизняних IT компаній, при використанні данних фреймворків, дотримуються усіх принципів Agile. Співпрацючи з різними українськими підприємствами та ознайомившись з їх робочими процессами, стало зрозуміло що в більшості випадків необхідно нехтувати багатьмя законами фреймворків задля покращення робочих процессів. Кожен окремий проект унікальний, склад команд завжди різний, рівень розробників також не однаковий. Тому і зміни в процессах постійно різні.

Проте компанії на співбесідах з кандидатами все одно кажуть, що працють по технології Scrum. Але це не відповідає дійсності, і працюють вони за неіснуючою технологією без назви, яка чимось нагадує Scrum.

2.3 Розробка робочих процесів поліпшеної технології управління проектами на основі фреймворку Scrum

Scrum як технологія позиціонує себе як ітеративна і циклічна, тобто усі процеси повторюються з визначенним інтервалом. Ці процеси можуть тривати нескінченно, до моменту завершення проекту. При роботі за цією технологією команда матиме 5 кроків, які також називають Scrum-фази.

1. Створення Беклогу продукту.

Це створення списку функціоналу, який необхідно виконати команді розробників на цьому проекті.

На цьому кроці функціонал декомпозується та візуалізуються як стікери на дошці. Кожен такий стікер ще називають тікетами.

Загалом уснує декілька типів тікетів:

- епік (epic) - велике завдання, на вирішення якої команді потрібно кілька спринтів;

- історія (story) - частина великого завдання (епіка), яку команда може вирішити за 1 спринт;
- завдання (task) - технічне завдання, яке робить один із членів команди;
- під-завдання (sub-task) - частина історії/завдання, яка описує мінімальний обсяг роботи члена команди;
- баг (bug) - завдання, яке описує помилку в системі.

Історії користувачів перетворюються з великих елементів і стають меншими, які можна помістити в беклог продукту. Епіки також можуть бути включені до бэклогу продукту, але не можуть бути включені до відставання спринту без перетворення його в історію користувача.

Типовий приклад історії користувача: як адміністратор, я хочу додавати, змінювати та видаляти завдання для користувачів на веб-сайті.

2. Планування спринту

Тривалість спринту дуже важлива, щоб історій користувачів було якомога менше. Типова середня тривалість спринту триває близько 2 тижнів. Якщо тривалість спринту невелика, перевага полягає в тому, що можна отримати більше відгуків від клієнтів, а більшість помилок і помилок можна усунути раніше. Якщо тривалість спринту велика, це дозволяє розробнику працювати ретельно.

3. Створення беклогу спринту.

На даному етапі команда scrum має вибрати важливі історії користувачів і перетворити їх на менші завдання. Вони повинні спланувати, як виконати завдання. Крім того, важливо визначити пріоритети необхідних завдань.

По завершенню спринта, він може отримати два статуси:

- Спринт успішний: весь розроблений функціонал розроблений, протестований, продемонстрований замовнику та виконує свою бізнес ціль;
- Спринт провальний: функціонал не був розроблений до кінця в задані часові рамки, дефектний або не виконує свою бізнес ціль. Тоді усі розробки за спринт необхідно видалити та почати розробку спочатку.

4. Робота над спринтом

Фактичні історії користувачів переміщуються у вигляді невеликих завдань у відставання спринту, де починається фактична робота. Тут починається реалізація програмного додатка, наприклад, розробка веб-сайту.

Для початку використовується дошка завдань, яку також називають дошкою Канбан, з великою кількістю карток. На картках вказуються такі відомості про завдання, як доручена особа, деталі роботи, термін виконання або тривалість тощо. Дошка завдань складається з стовпців «Завдання», «Робота виконується», а потім стовпці «Тестування» та «Виконана робота». Картки можна переміщати зліва направо в порядку бажаного та залежно від завершення.

На цьому кроці важливі scrum мітинги, оскільки вони проводяться для відстеження статусу прогресу та того, хто який статус виконує. Тому щоденно на початку робочого дня проводять DSM .

DSM або Daily Scrum Meeting – це 15-хвилинний мітинг для Scrum команди. Зустрічі, як правило, проводяться в одному місці й в один і той же час щодня. В ідеалі щоденні збори Scrum проводяться вранці, оскільки це допомагає встановити контекст для роботи майбутнього дня. Основною ціллю мітингу є засинхронізуватися. DSM не використовується як нарада для вирішення проблем. Питання, які піднімаються, розглядаються окремо, щом не затримувати учасників команди, кого ця проблема не стосується. Під час щоденної сутички кожен член команди відповідає на наступні три запитання:

- Що ви робили вчора?
- Що ви будете робити сьогодні?
- Чи є якісь перешкоди на вашому шляху?

Зосереджуючись на тому, що кожна людина досягла вчора і досягне сьогодні, команда отримує відмінне розуміння того, яка робота була виконана, а яка робота залишилася. DSM – це не зустріч з оновленням статусу, на якій бос збирає інформацію про те, хто відстає від графіка. Скоріше, це зустріч, на якій члени команди беруть зобов’язання один перед одним.

Нажаль, більшість учасників цих мітингів надають перевагу спілкуванню без увімкненної веб-камери, що руйнує командну єдність і відштовхує членів команди один від одного.

5. Тестування та демонстрація продукту

Виконані завдання мають бути реалізовані як робочий продукт з випробуванням повного життєвого циклу. Вартість тестування може бути мінімізована за допомогою додавання Quality Assurance Engineer (тестувальника ПЗ) або меншої кількості історій, однак перше є найкращим рішенням. Кожен виконаний спринт повинен бути продемонстрований замовнику для того, щоб він погодився і його погляд на повне рішення

6. Ретроспектива та планування наступного спринту

Результатом цього кроку є обговорення того, що пройшло добре, а що можна покращити для наступного рівня. Крім того, вам потрібно обговорити отримані уроки та підводні камені будь-яких конкретних питань чи проблем. Потім слід розпочати планування наступного спринту на основі знань, які ми маємо про поточні процеси та минулі проекти.

Склад команди та ролі мають величезний вплив на процесси та їх формування. Класична скрам команда складається з Product Owner, Scrum Master та розробників. Але наразі команда, зазвичай, включає ще керівників проектів, тестувальників, дизайнерів UI/UX, бізнес аналітиків та DevOps. Scrum Guide не розраховний на сучасний склад команди, тому процеси для них не адаптовані. Також Scrum Guide передбачає, що усі члени команди досвідченні розробники, у яких низька вірогідність помилки. Але в реальності це далеко не так. Сучасна різноролева та різнорівнева команда при класничному скрамі не зможе працювати в повну силу.

2.4 Висновок до розділу 2

В даному розділі проаналізовані робочі процеси технології Scrum, описано особливості моделювання інформаційнох технології управління проектами, розглянуто адаптації технології відчизняними компаніями. Також в результаті порівняльної характеристики вибрано технологію для подальшого вдосконалення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ ПРОЕКТАМИ

3.1 Використання клієнт-серверної архітектури для управління проектами

Модель клієнт-сервер – це програмний комплекс, у якому функціональні частини мають таку взаємодію: надходить запит клієнта, на який реагує сервер та дає свою відповідь. Клієнт виконує зазвичай активну роботу, яка складається із формування запитів, в той час як сервер пасивно відповідає на них. Ролі іноді змінюються, в залежності від поставленої задачі, наприклад, обидві функції можуть виконуватись одночасно, або ж обидві частини реалізовані окремо [7].

Зазвичай, архітектура клієнт-сервер має бути розподілена на три основних частини: два фізичні модулі, сервер бази даних, де зберігаються дані, клієнтська частина, зазвичай інтерфейс, а обов'язки обробки даних розподіляються між клієнтською та серверною частиною. Предметом суперечок іноді постає саме розподіл цих зобов'зань. Для цього виділяють групи для вирішення різних підзадач:

- Взаємодія з користувачем, відображення даних;
- Бізнес-логіка та прикладні функції;
- Управління ресурсами.

Ці три групи є загальними для клієнт-серверної архітектури, але можуть існувати сервери або вузли, які виконують специфічні завдання. Користувацький інтерфейс реалізується окремо, на клієнтських пристроях, з яких клієнт може взаємодіяти з додатком. Іноді додаток може давати результат обробки даних одразу у клієнтській стороні.

Бізнес-логіка являє собою сукупність правил і принципів для поведінки об'єктів, що є синонімом предметної області [12]. Часто цей рівень присутній у багаторівневих додатках та взаємодіє як з серверами, так і з інтерфейсом користувача. Рівень може бути представлений у вигляді аналітики, діаграм, бізнес-правил або алгоритмів та графів, вимогою ж до цього рівня є збереження даних. Це означає, що

при некоректній роботі додатку чи однієї з його частин, дані будуть збережені у базі чи іншому середовищі для подальшої роботи з ними (рис. 3.1).

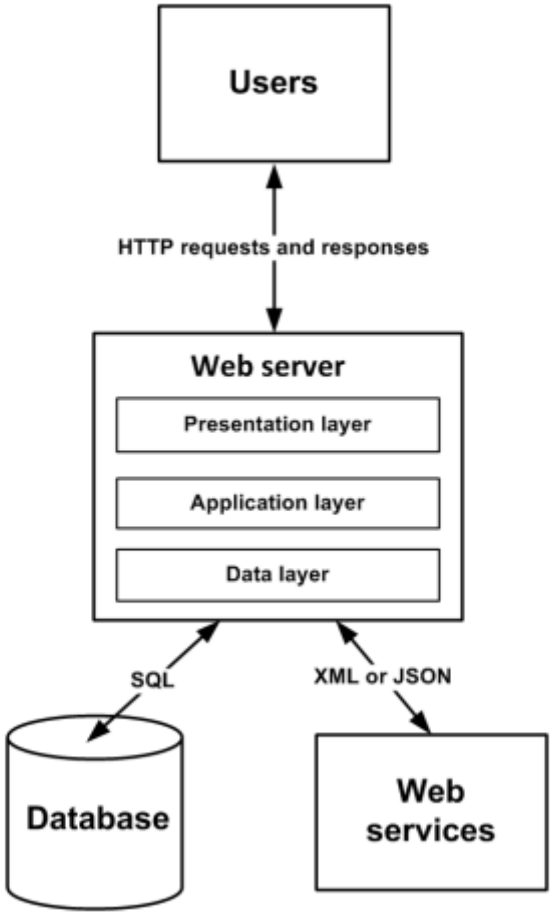


Рисунок 3.1 – Зображення клієнт-серверної архітектури

Для збереження та обробки даних зазвичай використовують реляційні бази даних, що допомагає зробити їх незалежними. Дані не залежать від додатків у двосторонньому зв’язку, тобто зміни додатку не мають впливу на дані, а дані, в свою чергу, на коректне спрацювання додатку. Використання реляційних БД може вплинути на розподіл рівнів обробки даних та їхнього зберігання, тобто виконувати ці процеси незалежно один від одного. Проте, є велика кількість додатків і сервісів, для яких використання даного типу баз даних не є доцільним. Найчастіше даними в них є об’єкти, а не прості типи, які зручно зберігати в базах. Це добре ілюструють мультимедійні системи, які працюють з відео- та аудіоматеріалами та використовують для цього значно доцільніші потоки даних, аніж реляційне

зберігання. У цих випадках потрібно розглядати об’єктно-орієнтовані бази даних, які підтримують зберігання, обробку та конкретні операції для предметної області.

3.2 Розробка структури серверної частини програмного забезпечення для управління проектами

Програмне забезпечення включає в себе кілька модулів, а саме: модуль API Telegram, модуль бібліотек, модуль баз даних, модуль отримання інформації та модуль сортування категорій.

Telegram міститиме у собі функції, які взаємодіють з відкритою системою Telegram, зокрема ботом, у який надсилатимуть дані задля їхнього подальшого опрацювання.

Модуль бібліотек описує усі під’єднані до додатку бібліотеки. Такі бібліотеки потрібно під’єднати до даної розробки:

- бібліотеки для бази даних;
- бібліотеки для підключення до телеграму;
- бібліотеки для тестування;
- бібліотека додатку з усіма складовими.

Модуль баз даних включає в себе взаємодію даних користувачів, як із боту, так і адміністратора, з базами даних, зокрема NoSQL база даних Firebase Realtime, у якій відбуватиметься логування усіх структур та подій в режимі реального часу із хмарним середовищем Firebase. Для зберігання та оновлення даних під’єднується внутрішня база даних Room.

У модулі отримання інформації дані, що надходять взаємодіють з модулем API Telegram, модулем баз даних та модулем програми. Це проміжний модуль, у якому дані відображуватимуться у додатку в полях адміністратора. Адміністратор зможе обрати або змінити категорію, обрати або змінити статус ідеї, відповісти на повідомлення. Існують такі категорії:

- Безпека праці (Security);
- Система якості (Quality);

- 5S;
- Продуктивність\ефективність (Effectivity).

Усі категорії окремо сортуються по чотирьом напрямкам їхньої подальшої участі, тобто адміністратор вирішує, що потрібно робити із ними далі:

- Відхилити (Reject);
- Відправити до архіву (Archive);
- Відправити на доопрацювання ідеї (To be Done);
- Завершити (Complete).

Після перегляду інформації та обрання категорії, адміністратор може надати відповідь, яка відправиться у бот користувачу.

Між модулями існує певна взаємодія. Так, бібліотеки під’єднуються до усього проекту, усі модулі мають доступ до бібліотеки в разі потреби. API використовує основний програмний модуль, усі функції передачі даних із клієнтської сторони проходять на опрацювання через цей модуль та завдяки ньому. У додаток дані підгружаються завдяки базам даних, відображуються у відповідних полях адміністратора, та направляються назад після опрацювання також у бази даних. Дані мають бути незалежними від роботи додатку, і навпаки. Модуль сортування та отримання інформації взаємодіють спочатку з базами даних, логуються та відправляються назад у бази.

Додатково дані збиратимуться на клієнтській стороні бота окремо по категоріям задля полегшеного виведення у додаток. Також їх легше сортувати та заносити у базу даних після попередньої обробки.

На рис. 3.2 зображено структурну схему описаних модулів серверної частини програмного забезпечення для управління проектами.

Рисунок 3.2 – Структура серверної частини програмного забезпечення для управління проектами

Use case diagrams (діаграми прецедентів) – модель функціонування системи, яка є узагальненою. Діграма містить два компоненти:

1) Actor (учасник) – множина ролей, логічно пов’язаних одна з одною та які виконуються при взаємодії з прецедентами чи сутностями. Актором може бути людина, система, роль людини в цій системі, клас чи підсистема.

2) Use case (прецедент) – опис кожного аспекта поведінки при проходженні процесу взаємодії з додатком з точки зору актора. Прецедент вказує, що саме виконується, але не яким чином.

На діаграмі зображено акторів (або одного) та закінчену послідовність дій, яка проходить від початку взаємодії з програмою, до кінця. У даному додатку діаграма містить у собі два актори, це адміністратор та клієнт, та їхні прецеденти. Адміністратор отримує дані, переглядає їх, сортує та при бажанні може надати відповідь (фідбек). Можливе створення нової ідеї зі сторони адміністратора або ж

зміна категорії, змісту чи статусу в подальшому, тобто взаємодія з даними. Клієнт у свою чергу при взаємодії з додатком обирає бажану категорію для ідеї, формує ідею чи побажання з можливістю опису та фотографіями, та надсилає до адміністратора на розгляд.

На рис. 3.3 зображено діаграму прецедентів програмного забезпечення для управління проектами.

Рисунок 3.3 – Діаграма прецедентів програмного забезпечення для управління проектами

3.3 Розробка UML-діаграм програмного забезпечення для управління проектами

Unified Modeling Language (UML) – уніфікована мова для моделювання, тобто створення моделі, яка описуватиме об’єкт.[8] Те, що вона універсальна, означає єдність розуміння символів та позначок різноманітних систем, предметних областей та у різних організаціях, для людей з різним рівнем знань та компетентності, сфери. Ця мова описує об’єкти в одному загальноприйнятому форматі з єдиним синтаксисом.

Деякі варіанти використання мови:

- Проектування;
- Реверс-інжиніринг;
- Отримання текстової інформації та документування.

UML-діаграми при проектуванні допомагають моделювати стуктуру великих проектів, де нарховуються як макро-, так і мікродеталі, та завдяки яким можливо створити макет проекту. Згодом по цьому макету створюватиметься код проекту.

При реверсивному інжинірингу UML-діаграми будуються на основі вже існуючого коду, тобто навпаки. Допомагає, наприклад, у ситуаціях написанного коду із існуючою неповною документацією, або ж за повної її відсутності.

Так само із моделей часто витягують текстову інформацію для генерації більш зручного її подання – документацій. Графіка й текстові файли доповнюють один одного, надають повніше уявлення моделі [8].

Як усі мови, мова UML-діаграм також має певний синтаксис, який розпізнають у різних країнах та різних структурних проектах. Загальний рівень містить чотири базових позначення:

- Фігури;
- Лінії;
- Позначки;
- Надписи.

Така нотація є стандартом створення архітектурних рішень та бізнес-систем. Існує всього дванадцять типів діаграм:

- 4 типи дають представлення статичного стану додатку;
- 5 типів представляють поведінкові моменти;
- 3 представляють фізичні моменти та стани системи (діаграми реалізації).

Деякі з цих видів є специфічними для широкого використання, а найдоступнішими з інших є:

- Діаграма класів (Class diagram);
- Діаграма прецедентів (Use-case diagram);
- Діаграма послідовностей (Sequence diagram);

- Діаграма розгортування (Deployment diagram);
- Діаграма взаємодії (Collaboration diagram);
- Діаграма активностей (Activity diagram);
- Діаграма об’єктів (Object diagram);
- Діаграма станів (Statechart diagram).

Діаграма класів (Class diagram), як видно з назви, містить класи. Клас (class) – це деяка категорія, яка об’єднує речі, що містять спільні атрибути й операції. Діаграма являє собою набір статичних та декларативних елементів моделі. Дана діаграма дає найбільш розгорнуте та повне уявлення про взаємозв’язок у додатку, його функціональність, а також інформація про окремі класи. Найчастіше додаток генерують саме з діаграми класів.

Діаграму активностей (Activity diagram) також часто реалізують на практиці. Діаграма описує динамічні аспекти поведінки у моделі, відображається за допомогою блок-схеми, а та являє собою бізнес-процеси, бізнес-логіку, логіку процедур й потоки (перехід між діяльностями). Насправді, вона є алгоритмом поведінки системи (логіка системи) чи взаємодія одразу декількох систем.

У нашому випадку актор (клієнт), який має свою ідею, запускає бот у Telegram, обирає категорію, під яку найбільше підходить його ідея, якщо існують підкатегорії, обирає їх також, надсилає. Дані приходять у додаток, де їх має можливість переглянути адміністратор, сортує та обирає наступний стан ідеї. При потребі, можливе коригування даних чи категорій.

На рис. 3.4 зображено діаграму активності програмного забезпечення для управління проектами.

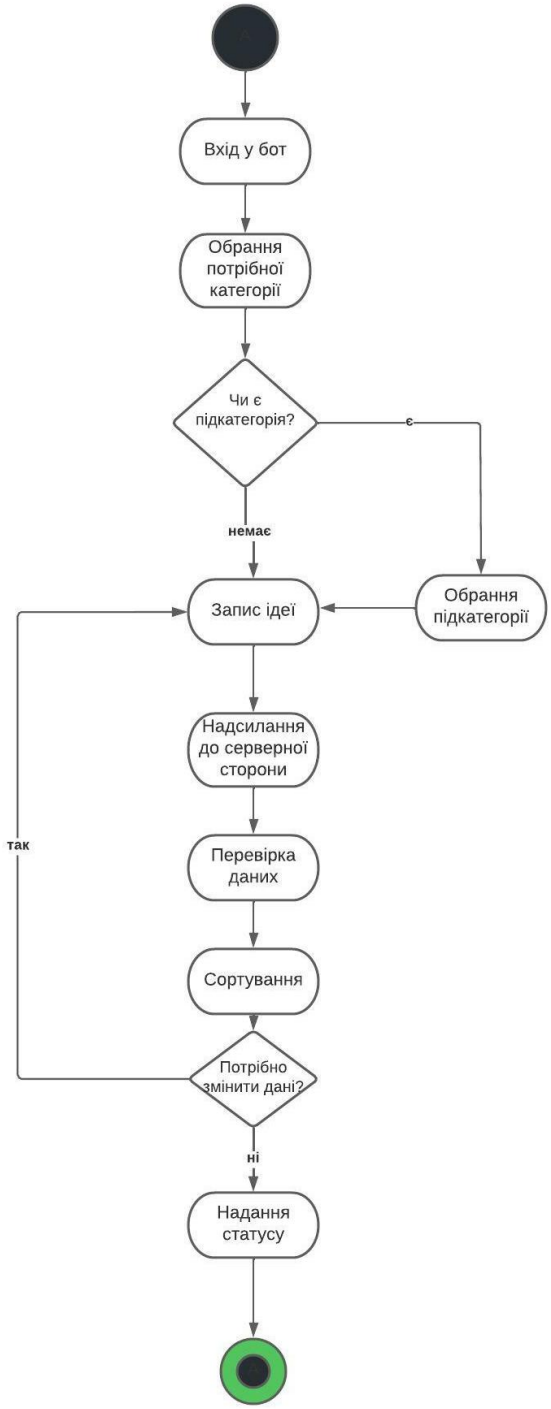


Рисунок 3.4 – Діаграма діяльності програмного забезпечення для управління проектами

Діаграму послідновностей (Sequence Diagram) використовують для опису поведінки системи чи її окремих компонентів. Діаграма показує взаємодію у динамічному сенсі, тобто у часі. Інформація на такій діаграмі приймає вигляд повідомлень, а взаємодія об’єктів має на увазі обмін цими повідомленнями.

Діаграма розгортування (Deployment Diagram) графічно відображає інфраструктуру, яка використовуватиметься для розгортування додатку. Це може бути топологія системи, розподіл вузлів та компонентів по цих вузлах, а також усі взаємозв’язки між ними – для передачі даних між вузлами. За допомогою діаграми можливо спроектувати компоненти раціонально, залежності між цими компонентами та навіть продуктивність системи. Також можливо вирішити задачі безпеки.

Діаграма станів (Statechart diagram) являє собою абстрактний підклас, який відображає окрему ситуацію, протягом якої виконується умова, та її динамічні аспекти поведінки. Стан може задаватись набором конкретних значень, атрибутів об’єктів чи класів. Зміна окремих значень відображає також зміну стану заданого класу. Слід зауважити, що не всякий атрибут класу чи об’єкту підходить для характеристики його стану. Важливі лише такі властивості елементів системи, що мають відношення до динамічних змін об’єкту чи класу.

На рис. 3.5 зображено діаграму станів програмного забезпечення для управління проектами.

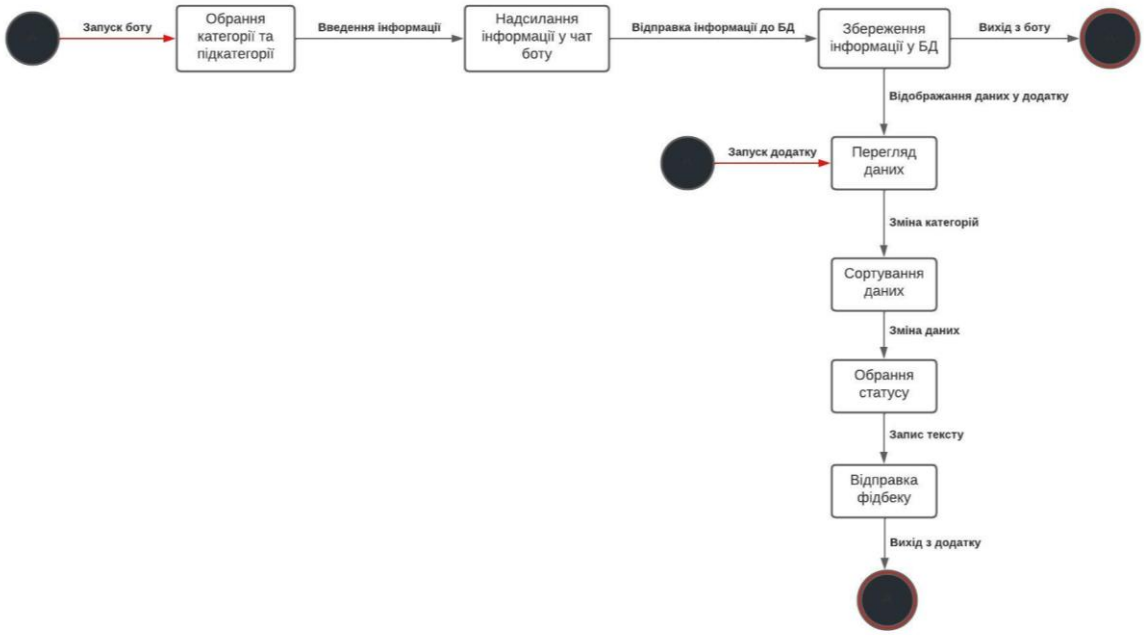


Рисунок 3.5 – Діаграма станів програмного забезпечення для управління проектами

Діаграма відображує процес проходження станів з двох сторін: клієнтської та серверної. На початку роботи користувач запускає бот у Telegram та запускає бот. Після цього обирає одну з п’яти категорій та при наявності підкатегорій. Бот надсилає форму відправки даних та порядок запису, після чого користувач згідно з інструкціями записує усі дані, включно з фотографіями (при їхній наявності). Дані потрапляють до бази даних, звідки відображаються у додатку зі сторони адміністратора. Дані зберігаються окремо та можуть очікувати на подальшу обробку незалежно від додатку чи боту. Робота адміністратора починається із запуску додатку. Можливо обрати спосіб відображення існуючих даних: уже відсортовані та переглянуті можливо змінити, а нові – переглянути. Дані, що надійшли нещодавно, зберігаються окремо та очікують сортування. Після перегляду на обрання деякої категорії, адміністратор може написати фідбек у вигляді тексту, що потім потрапить у бот, відкоригувати існуючі дані або завершити роботу з додатком. Дані зберігаються у модулі бази даних до подальшого іхнього відображення.

3.4 Обґрунтування вибору мови програмування та середовища розробки

Для аналізу і порівняння було обрано три мови програмування: Kotlin, Java та JavaScript

Kotlin – це одна з найновіших мов програмування, що характеризується відносно доступним та лаконічним синтаксисом. З’явившись лише 6 років тому ця мова вже встигла стати однією з найпопулярніших та найбільш орієнтованих на зручне написання коду[10]. Що казати, якщо навіть компанія Google одразу після появи цієї мови вирішила додати свої інструменти в Android Studio 3.0, офіційний інструмент для розробки додатків на Android. Цій мові характерна швидка компіляція та адаптивність до інших мов програмування, що дозволяє фахівцям зі знанням Kotlin швидко опанувати і інші мови написання коду, такі як Java, оскільки вони мають багато спільних рис. Але все ж головною відмінністю на користь Kotlin грає його стисліть та легкодоступність порівнянно з будь-якою іншою мовою програмування.

Java – мабуть, одна з найпопулярніших мов програмування на базі Android на даний момент, адже вона є доволі зручною та багатофункціональною. Заснована ще у далекому 1995 році, Java не раз демонструвала свої надзвичайні здібності та можливість програмувати на будь-якій комп'ютерній архітектурі. Про релевантність даної мови каже той факт, що не дивлячись на начебто її застарілість, оновлення досі випускаються та цикл життя підтримується уже більш ніж 25 років [10]. До плюсів цієї мови можна віднести автоматичне керування пам'яттю, наявність класів для виконання базових HTTP запитів а також уніфікований доступ до баз жанних. Однак великим мінусом є той факт, що ця мова може бути важкою через свій синтаксис. Всі перелічені можливості присутні також і в Kotlin, але з урахуванням спрощенного методу написання коду.

JavaScript – як не дивно, але хоч це і інша мова програмування, що швидко розвивається але з назви можна здогадатись, що вона є нащадком Java. Як вже зазначалось раніше, мова-предок мала великий недолік, а саме занадто складний синтаксис, тому, для вирішення цієї проблеми, було створено підвид, а саме JavaScript [9]. Мова володіє динамічною типізацією, автоматичним керуванням пам'яттю, прототипне програмування, спрощений порівняно із предком синтаксис та широким застосуванням серед розробників веб-сторінок, а саме їх інтерактивної частини. Незважаючи на всі доповнення порівняно зі старшою версією, в даної також є певні недоліки. Через урізання синтаксису з метою його спрощеність зникла також і крос-платформеність рівня Java. Звісно, базові мови підтримуються, але якщо брати в порівняння старшого предка доволі очевидним стає результат, при якому ця мова зазнала значних урізань. До того ж, як і більш старша версія, поріг входження у JavaScript являється доволі високим, оскільки незважаючи на присутність класів та їх наслідування, модель запропонована у даній мові суттєво відрізняється від вже звичних аналогів (табл. 3.1).

Таблиця 3.1 – Порівняльна характеристика мов програмування.

Мова/ Критерій	Статична типізація	Динамічна типізація	Виконання на серв.частині	Легке підключання до БД	Крос-платформеність
Kotlin	+	-	+	+	+
JavaScript	-	+	+	-	-
Java	+	-	_	+	-

Для виконання бакалаврської кваліфікаційної роботи було обрано мову програмування Kotlin,в зв'язку з тим, що вона дозволяє набагато доступніше візуалізувати результат роботи з базами даних. Окрім того, необхідно також зазначити, що ця мова підтримує крос-платформеність, що дозволяє не зосереджуватись на одному конкретному типу пристроїв при написанні коду. В доповнення до вищезазначених факторів також можна сказати, що в нинішній час ця мова набуває широкого розмаху і буде актуальною протягом довгого періоду часу, що дозволить використовувати виконану роботу в подальшій перспективі.

Головним ресурсом для створення проекту було обрано платформу Android Studio. Вона є у вільному доступі та дозволяє створювати необхідні маніпуляції з кодом та програмою в цілому без застосування додаткових засобів [10]. На користь обраного додатку говорить той факт, що він із легкістю компілює проекти написані як мовою Kotlin, так і Java, що дозволяє вільно продовжити роботу з проектом навіть при бажанні зміни мови для більшого функціонування.

Одними з головних переваг Android Studio над конкурентами можна виділити оптимізацію і, як результат, швидкість компіляції. Також необхідно зазначити, що ця програма є однією з небагатьох доступних у вільному доступі та дозволяє розраховувати на додаткові можливості без необхідності доплат. Всі необхідні інструменти уже встановленні та можуть використовуватись при будь яких умовах. Окрім цього, як уже було зазначено, швидкість компіляції в данній програмі вражає, що говорить про хорошу оптимізацію та розвиток додатку у майбутньому. Також редактор має гнучке налаштування клавіш, шрифту, кольорової схеми тощо [11].

Не менш важливим є сам аспект написання коду, і програма Android Studio чудово себе зарекомендувала як зручний та приємний для роботи додаток. Всі необхідні дані підсвічуються, що робить пошук необхідної стрічки швидким та зручним. Всі рекомендації щодо спростування коду одразу ж надходять при написанні, без необхідності повторної переробки великої кількості написаного матеріалу. До того ж, окрім уже названних вище можливостей, у програмі зрвучно реалізовано перехід між вікнами виконання завдань та папками зі змістом усіх матеріалів. Це дозволяє вірно користуватись програмою навіть на інтуітивному рівні без базових знань у сфері програмування

Таким чином, Android Studio можгна вважати найбільш доступним, модульним та функціональним інструментом створення програмного забезпечення (рис. 3.6).

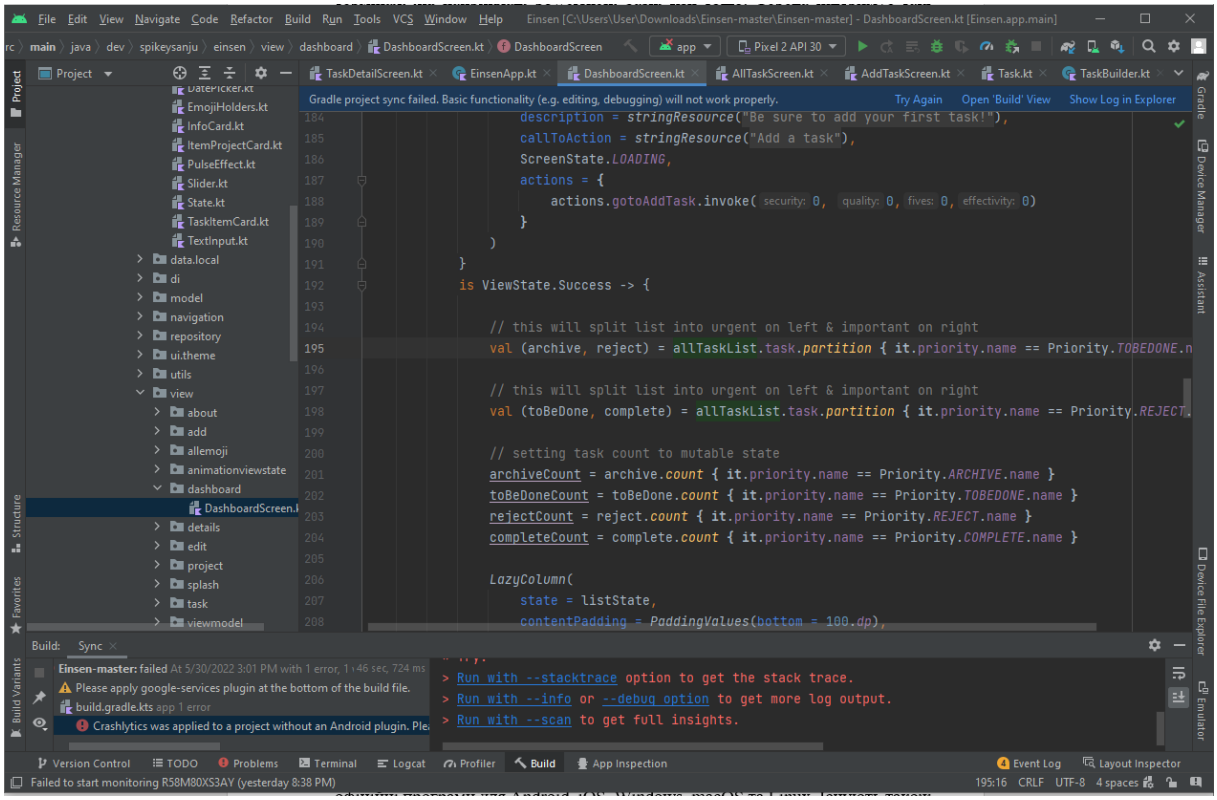


Рисунок 3.6 – Вікно середовища розробки Android Studio

Система SQLite в Android ніколи не була зручною у використанні, потребувала багатьох строчок коду перед написанням самої логіки мови. Google запропонувала своє рішення, яке одразу стало найпоширенішим – Room. Він є обгорткою для

SQLiteOpenHelper, тобто ORM (Object-relational mapping) між класами Java та SQLite. [12] Room входить до Android Jetpack (раніше Android Architecture Components) та з’явився разом з Котліном у 2017 році на Google I/O [11]. Раніше альтернативої були Realm, ORMLite, GreenDao, а також інші застарівші бібліотеки. Room є високорівневим інтерфейсом для низькорівневих прив'язок SQLite, які безпосередньо вбудовані в систему Android. Система виконує найбільшу частину роботи під час компіляції: створює API-інтерфейс над вбудованим SQLite API. Це дозволяє забути про роботу з такими додатками, як Cursor чи ContentResolver, які пропонувалися іншими розробниками для спрощення роботи.

Firebase є збіркою платформ для допомоги в розробці мобільних застосунків. Firebase Inc. У 2014 році придбав Google. Містить у собі найпоширеніші включення: Firebase Analytics, Firebase Cloud Messaging, Firebase Auth, Realtime Database, Cloud Firestore, Firebase Storage, Firebase Hosting [13].

Firebase Analytics – рішення для початкової оцінки застосунку за допомогою користувачів. Є безкоштовним та легке в застосуванні.

Firebase Cloud Messaging (FCM) раніше був Google Cloud Messaging (GCM). FCM крос-платформний, завдяки чому набув широкого застосування у мобільній розробці. Допомагає у реалізації повідомлень, нотифікацій для вебу, iOS, Android. Є безкоштовним.

Firebase Auth – служба для автентифікації користувачів та взаємодіючи лише із кодом на клієнтській стороні. Підтримує швидку авторизацію за домогою логіну і паролю з GitHub, Google, Twitter і Facebook. До того ж, включає в себе систему управління користувачами, що допомагає розробникам додатків вмикати автентифікацію електронною поштою, що зберігається у Firebase.

Крім того, Firebase впроваджує базу даних в режимі реального часу (Realtime Database) разом з бекендом до неї. Служба надає API розробникам, за допомогою якого можливо синхронізувати надходження даних з клієнту та зберігати у Firebase [14]. Доступні служби та бібліотеки для роботи з Swift, JavaScript, Objective-C, Android, iOS, Java, а база даних доступна для роботи з React, Backbone.js, Ember.js через REST API. У свою чергу, він використовує протокол подій сервера-інтерфейса

створення HTTP-з'єднань. Допомагає у створенні ' отриманні push-повідомлень. Сервери підтримують відповідні правила безбеки, які розробники використовуватимуть для захисту даних у Realtime Database.

Firebase Storage є платформою для забезпечення обміну файлами: їхнє завантаження та вивантаження у Firebase навіть незалежно від якості з'єднання з мережею. Може працювати із відео-, аудіо-, забраженнями або іншими файлами, створеними користувачем. Має підтримку Google Cloud Storage.

Firebase Hosting та Functions, що являють собою вебхостинг: статичний та динамічний. Із статичного це JavaScript-файли, HTML-, CSS-, можливі інші файли, та динамічні Node.js через Cloud Functions. За допомогою HTTPS протоколу служба має змогу передавати файли мережею доставки контенту (CDN).

3.5 Програмна реалізація інтерфейсу інформаційної технології управління проектами

Реалізовуємо серверну та клієнтську частини програмного забезпечення для управління проектами згідно розробленої у пункті 3.2 схеми, а саме мобільний додаток із використанням боту та баз даних. Відповідно, система складатиметься із описаних вище основних модулів.

Перший модуль – модуль бібліотек.

Модуль необхідний для короткної роботи додатку, баз даних та з'єднання додатку з ботом для відображення даних. Під'єднані бібліотеки Firebase Analytics, Room, Kotlin, Android Jetpack, Hilt у файлі підключень проекту:

```
dependencies {
        classpath("com.android.tools.build:gradle:7.2.1")
        classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:${kotlinVersion}")
        classpath("com.google.dagger:hilt-android-gradle-plugin:${hiltVersion}")
        classpath("com.google.gms:google-services:${googleServiceVersion}")
        classpath("com.google.firebase:firebase-crashlytics-
gradle:${crashlyticsVersion}") }
```

Та у файлі підключень модуля:

```
    dependencies {

        implementation("androidx.core:core-ktx:${rootProject.extra["ktxCoreVersion"]}")

implementation("androidx.appcompat:appcompat:${rootProject.extra["materialVersion"]}")

implementation("com.google.android.material:material:${rootProject.extra["materialVersion"]}
")
        implementation("androidx.compose.ui:ui:${rootProject.extra["composeVersion"]}")
        implementation("androidx.compose.animation:animation-
graphics:${rootProject.extra["composeAnimation"]}")

implementation("androidx.compose.material:material:${rootProject.extra["composeVersion"]}")
        implementation("androidx.compose.ui:ui-
tooling:${rootProject.extra["composeVersion"]}")
        implementation("androidx.lifecycle:lifecycle-runtime-
ktx:${rootProject.extra["lifeCycleVersion"]}")
        implementation("androidx.activity:activity-
compose:${rootProject.extra["composeActivityVersion"]}")
        implementation("androidx.hilt:hilt-work:1.0.0")
        testImplementation("junit:junit:4.13.2")
        androidTestImplementation("androidx.test.ext:junit:1.1.3")
        androidTestImplementation("androidx.test.espresso:espresso-
core:${rootProject.extra["expressoVersion"]}")
        androidTestImplementation("androidx.compose.ui:ui-test-
junit4:${rootProject.extra["composeVersion"]}")

        // Firebase
        implementation(platform("com.google.firebase:firebase-bom:29.0.0"))
        implementation("com.google.firebase:firebase-crashlytics-ktx")
        implementation("com.google.firebase:firebase-analytics-ktx")

        // compose navigation
        implementation("androidx.navigation:navigation-
compose:${rootProject.extra["composeNavigationVersion"]}")
        implementation("androidx.hilt:hilt-navigation-
compose:${rootProject.extra["hiltComposeNavVersion"]}")

        // Preferences DataStore
```

```
            implementation("androidx.datastore:datastore-
preferences:${rootProject.extra["dataStoreVersion"]}")

            // Room
            implementation("androidx.room:room-runtime:${rootProject.extra["roomVersion"]}")
            kapt("org.xerial:sqlite-jdbc:${rootProject.extra["jdbcVersion"]}")
            kapt("androidx.room:room-compiler:${rootProject.extra["roomVersion"]}")
            implementation("androidx.room:room-ktx:${rootProject.extra["roomVersion"]}")

            // System UI Controller
            implementation("com.google.accompanist:accompanist-
systemuicontroller:${rootProject.extra["systemUIControllerVersion"]}")

            // Coroutines
            implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
core:${rootProject.extra["coroutinesVersion"]}")
            implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
android:${rootProject.extra["coroutinesVersion"]}")

            // Dagger Hilt
            implementation("com.google.dagger:hilt-
android:${rootProject.extra["hiltVersion"]}")
            kapt("com.google.dagger:hilt-android-
compiler:${rootProject.extra["hiltVersion"]}")
            implementation("androidx.hilt:hilt-lifecycle-
viewmodel:${rootProject.extra["hiltAndroidXVersion"]}")
            kapt("androidx.hilt:hilt-compiler:${rootProject.extra["hiltCompilerVersion"]}")
            implementation("androidx.hilt:hilt-navigation-
compose:${rootProject.extra["hiltComposeVersion"]}")
            implementation("androidx.hilt:hilt-
common:${rootProject.extra["hiltCompilerVersion"]}")
            kapt("com.google.dagger:hilt-compiler:${rootProject.extra["hiltVersion"]}")

            // KotlinX Serialization
            implementation("org.jetbrains.kotlinx:kotlinx-serialization-
json:${rootProject.extra["kotlinSerializationVersion"]}")

            // Compose Navigation Animation
            implementation("com.google.accompanist:accompanist-navigation-
animation:${rootProject.extra["navigationAnimation"]}")
```

```
            implementation("com.google.accompanist:accompanist-navigation-
material:${rootProject.extra["navigationAnimation"]}")

            // Lottie
            implementation("com.airbnb.android:${rootProject.extra["lottieAnimation"]}")

            // Square Logcat
           implementation("com.squareup.logcat:logcat:${rootProject.extra["logcatVersion"]}")

            // Worker + Coroutine
            implementation("androidx.work:work-runtime-
ktx:${rootProject.extra["workerVersion"]}")}
```

Другий модуль – модуль API Telegram.[17] Модуль поєднує у собі функції бібліотеки ботів Telegram, їхні функції. Кожен чат-бот має унікальний ключ-підключення до свого чату й функцій для забезпечення доступу його керування. У даному додатку буде використовуватись створений бот із заданим токеном:

```
       fun main() {
            val token = '5591722517:AAEd9X1_NB3eacAFC9W6MZQbN8v-kDZVKtM'
            val username = "<BOT USERNAME>"
            val bot = Bot.createPolling(username, token)

            bot.chain("/start") { msg -> bot.sendMessage(msg.chat.id, "Hi! What is your name?")
}
                .then { msg -> bot.sendMessage(msg.chat.id, "Nice to meet you, ${msg.text}! Send
something to me") }
                .then { msg -> bot.sendMessage(msg.chat.id, "Fine! See you soon") }
                .build()

            bot.chain(
                label = "location_chain",
                predicate = { msg -> msg.location != null },
                action = { msg ->
                    bot.sendMessage(
                        msg.chat.id,
                        "Fine, u've sent me a location. Is this where you want to be?(yes|no)"
                    )
                })
```

```
            .then("answer_choice") { msg ->
                when (msg.text) {
                    "yes" -> bot.jumpToAndFire("save location", msg)
                    "no" -> bot.jumpToAndFire("cancel ", msg)
                    else -> {
                        bot.sendMessage(msg.chat.id, "Oops, I don't understand you. Just
answer yes or no?")
                        bot.jumpTo("answer_choice", msg)
                    }
                }
            }
            .then("save_location", isTerminal = true) { msg ->
                bot.sendMessage(msg.chat.id, "Fine!")
            }
            .then("cancel", isTerminal = true) { msg ->
                bot.sendMessage(msg.chat.id, "Ok! See you next time")
            }
            .build()

        bot.start()
    }
```

Третій модуль – Модуль баз даних.

Тут під’єднуються Firebase та Room.[16] Дані надсилаються з боту та зберігаються у Firebase, з якої можливо буде переглянути надіслані дані у додатку. Дані відображуються у відповідних категоріях, які додаток заповнює у полях. У базі даних Room можливо зберігати об’єкти, а також надсилати запити для їх редагування, додавання чи видалення. Під’єднання Room та створення DAO (Data Access Object):

```
import androidx.room.Database
import androidx.room.RoomDatabase
import model.task.Task

@Database(entities = [Task::class], version = 1, exportSchema = true)
abstract class EinsenDatabase : RoomDatabase() {
    abstract fun getTaskDao(): TaskDao
}
```

DAO створюється для доступу в Room та забезпечення виконання запитів. Запити, які використовуються у додатку:

```
@Dao
interface TaskDao {

    @Query("SELECT * FROM task")
    fun getAllTask(): Flow<List<Task>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertTask(task: Task)

    @Query("DELETE FROM task where id =:id")
    suspend fun deleteTaskByID(id: Int)

    @Update(onConflict = OnConflictStrategy.REPLACE)
    suspend fun updateTask(task: Task)

    @Query("SELECT * FROM task where id=:id")
    fun findByID(id: Int): Flow<Task>

    @Query("SELECT * FROM task where id=:id")
    fun findTaskByID(id: Int): Task

    @Query("UPDATE task set isCompleted= :isCompleted where id=:id")
    suspend fun updateTaskStatus(id: Int, isCompleted: Boolean)

    @Query("SELECT * FROM task WHERE priority=:priority")
    fun getTaskByPriority(priority: String): Flow<List<Task>>

    @Query("SELECT COUNT(id) FROM task WHERE priority=:priority")
    fun getTaskByPriorityCount(priority: String): Flow<Int>
}
```

Видалення, зміна, додавання нового запису та збереження інформації після змін. Також у кожного завдання є своє унікальне айді, завдяки якому й можливий доступ до кожного завдання.

Далі – модуль отримання інформації.

У модулі взаємодіють бази даних та додаток. Після отримання даних з боту, вони з’являються для перегляду у відповідній формі: полях. Адміністратор може одразу перейти до сортування, а може виправити або внести корективи в існуючу інформацію. Усі завдання потрапляють до адміністратору таким чином:

```
// get all task
fun getAllTask() = viewModelScope.launch(Dispatchers.IO) {
    repo.getAllTask().distinctUntilChanged().collect { result ->
        try {
            if (result.isNullOrEmpty()) {
                _viewState.value = ViewState.Empty
            } else {
                _viewState.value = ViewState.Success(result)
            }
        } catch (e: Exception) {
            _viewState.value = ViewState.Error(e)
        }
    }
}

// get all list of emoji from Json
fun getAllEmoji(context: Context, searchQuery: String) = viewModelScope.launch {
    try {
        // read JSON file
        val myJson = context.assets.open("emoji.json").bufferedReader().use {
            it.readText()
        }

        // format JSON
        val format = Json {
            ignoreUnknownKeys = true
            prettyPrint = true
            isLenient = true
        }

        // decode emoji list from json
        val                              decodedEmoji                                =
format.decodeFromString<List<EmojiItem>>(myJson).distinct()
```

```
                // filter the emoji based on Aliases
                val filteredEmojiAliases = decodedEmoji.filter { emojiAliases ->
                    emojiAliases.aliases.any {
                        it.contains(searchQuery, ignoreCase = true)
                    } || emojiAliases.category.contains(searchQuery, ignoreCase = true)
                }.distinct()

                if (searchQuery.isNullOrEmpty()) {
                    _emojiViewState.value = EmojiViewState.Success(decodedEmoji)
                } else {
                    if (filteredEmojiAliases.isNullOrEmpty()) {
                        _emojiViewState.value = EmojiViewState.Empty
                    } else {
                        _emojiViewState.value                                    =
EmojiViewState.Success(filteredEmojiAliases)
                    }
                }
            } catch (e: Exception) {
                _emojiViewState.value = EmojiViewState.Error(exception = e)
            }}

        // insert source
        fun insertTask(task: Task) = viewModelScope.launch {
            repo.insert(task)
        }

        // delete source
        fun deleteTaskByID(id: Int) = viewModelScope.launch {
            repo.delete(id)
        }

        // update status
        fun updateStatus(id: Int, isCompleted: Boolean) = viewModelScope.launch {
            repo.updateStatus(id, isCompleted)
        }

        // find task by id
        fun findTaskByID(id: Int) = viewModelScope.launch(Dispatchers.IO) {
            repo.find(id).distinctUntilChanged().collect { result ->
                try {
                    if (result.title.isEmpty()) {
```

```
                _singleViewState.value = SingleViewState.Empty
            } else {
                _singleViewState.value = SingleViewState.Success(result)
            }
        } catch (e: Exception) {
            _viewState.value = ViewState.Error(e)
        }
    }}

// get all task
fun getTaskByPriority(priority: String) = viewModelScope.launch(Dispatchers.IO) {
    repo.getTaskByPriority(priority).distinctUntilChanged().collect { result ->
        try {
            if (result.isNullOrEmpty()) {
                _viewState.value = ViewState.Empty
            } else {
                _viewState.value = ViewState.Success(result)
            }
        } catch (e: Exception) {
            _viewState.value = ViewState.Error(e)
        }
    }}

// update status
fun currentEmoji(emoji: String) = viewModelScope.launch {
    try {
        if (emoji.isEmpty()) {
            _currentEmoji.value = "Select an emoji"
        } else {
            _currentEmoji.value = emoji
        }
    } catch (e: Exception) {
        _currentEmoji.value = "⚠"
    }}

fun firebaseLogEvent(event: String, bundle: Bundle) {
    firebaseAnalytics.logEvent(
        event,
        bundle
    )
}
```

Останній модуль – сортування отриманої інформації.

Після потрапляння даних до додатку та відображення, настає час користувача взаємодіяти з цими даними. Він може оцінити за категоріями, обрати подальший статус завдання чи відредактувати існуючі завдання. Код, що додає завдання:

```
fun AddTaskScreen(
    modifier: Modifier,
    viewModel: MainViewModel,
    actions: MainActions,
    defaultUrgency: Int,
    defaultImportance: Int,
    defaultSecurity: Int,
    defaultQuality: Int,
    defaultFiveS: Int,
    defaultEffectivity: Int
) {

    // component state
    val listState = rememberLazyListState()
    val scope = rememberCoroutineScope()
    val context = LocalContext.current
    val scaffoldState = rememberScaffoldState()

    // slider points
    val points = listOf("0", "1", "2", "3", "4", "5")

    // task value state
    var taskState by remember {
        mutableStateOf(
            task {
                title = ""
                telephone = ""
                description = ""
                category = ""
                emoji = ""
                urgency = defaultUrgency
                importance = defaultImportance
                security = defaultSecurity
                quality = defaultQuality
```

```
                fives = defaultFiveS
                effectivity = defaultEffectivity
                feedback = ""
                priority = Priority.IMPORTANT
                due = ""
                isCompleted = false
            }
        )
    }

    LaunchedEffect(
        key1 = Unit,
        block = {
            // log event to firebase
            val addTaskComposable = bundleOf(
                FirebaseAnalytics.Param.SCREEN_NAME to "Add Task Screen",
                FirebaseAnalytics.Param.SCREEN_CLASS to "AddTaskScreen.kt"
            )

            viewModel.firebaseLogEvent("add_task_screen", addTaskComposable)
        }
    )

    // get current emoji
    viewModel.currentEmoji.collectAsState().value.apply {
        taskState = taskState.copy(emoji = this)
    }

    // Add Task Screen content
    Scaffold(
        scaffoldState = scaffoldState,
        topBar = {
            TopAppBar(
                title = {
                    Text(
                        text = stringResource(id = R.string.text_addTask),
                        style = AppTheme.typography.h2,
                        textAlign = TextAlign.Start,
                        color = AppTheme.colors.text,
                        modifier = modifier.padding(start =
AppTheme.dimensions.paddingLarge)
```

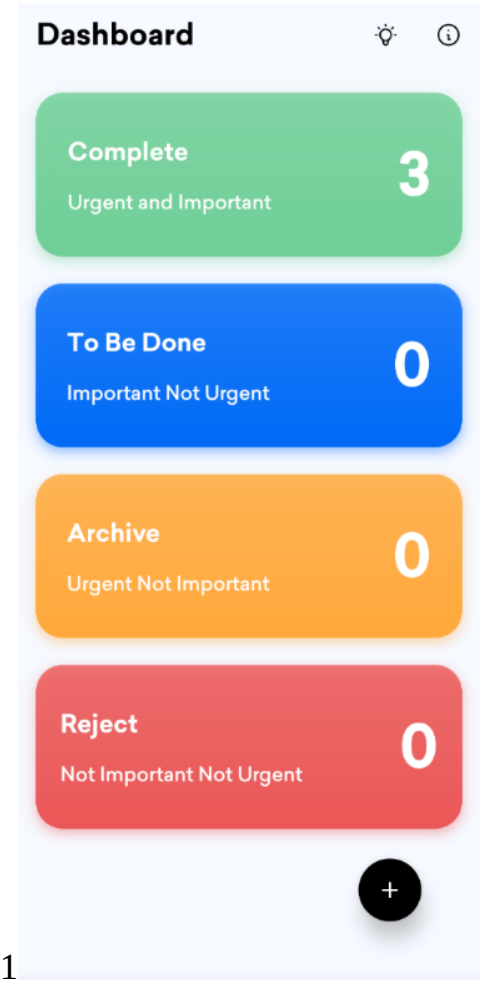
```
                )
            },
            navigationIcon = {
                IconButton(onClick = { actions.upPress.invoke() }) {
                    Icon(
                        painter = painterResource(id = R.drawable.ic_back),
                        contentDescription = stringResource(R.string.back_button),
                        tint = AppTheme.colors.text
                    )
                }
            },
            backgroundColor = AppTheme.colors.background, elevation = 0.dp
        )
    }

) {

    LazyColumn(
        state = listState,
        contentPadding = PaddingValues(bottom = AppTheme.dimensions.paddingXXL),
        modifier = modifier
            .background(
                AppTheme.colors.background
            )
            .fillMaxSize()
    )
```

3.6 Тестування програмного забезпечення для управління проектами та аналіз результатів

Для тестування програмного забезпечення, потрібно перевірити усі модулі, їхню коректну роботу та відображення. При запуску додатку відображається екран з чотирма категоріями – статусом завдань (рис. 3.7).



1

Рисунок 3.7 – Головний екран додатку «Magic Box»

При виборі одної з категорій відбудеться перехід до загальної кількості завдань у цій категорії. При натисканні кнопки з плюсом можна додати завдання із тими самими умовами (рис. 3.8).

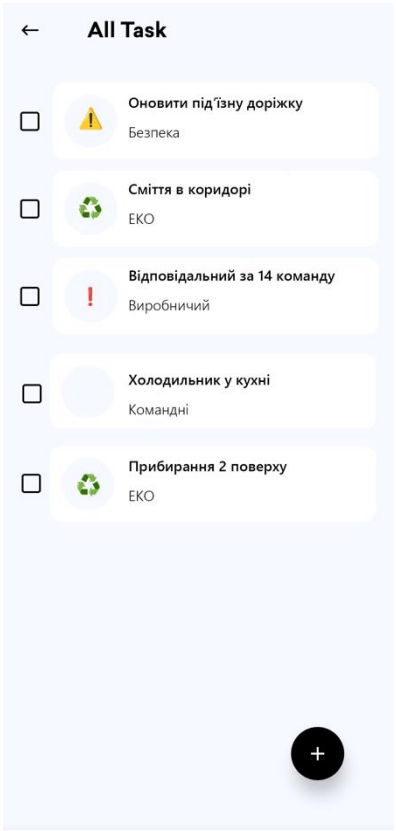


Рисунок 3.8 – Екран завдань додатку «Magic Box»

Далі обираємо будь-яке завдання, щоб переглянути його зміст (рис. 3.9).

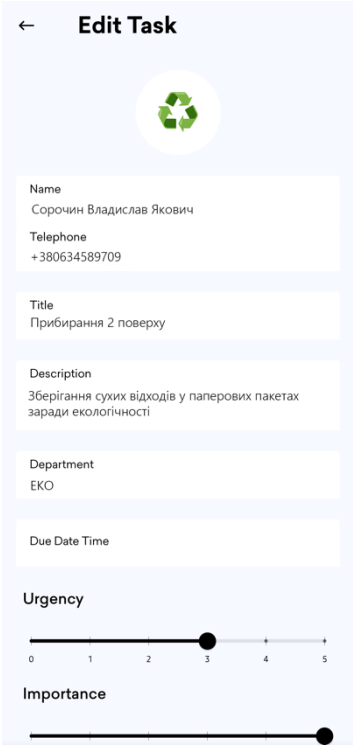


Рисунок 3.9 – Екран перегляду даних завдання

Після перегляду адміністратор може відредактувати існуючі дані, поставити оцінку або змінити її, відмітити як виконане чи видалити. При натискуванні іконки «поділитись», текстова інформація щодо завдання може бути відправлена у різні месенджери чи електронну пошту (рис. 3.10).

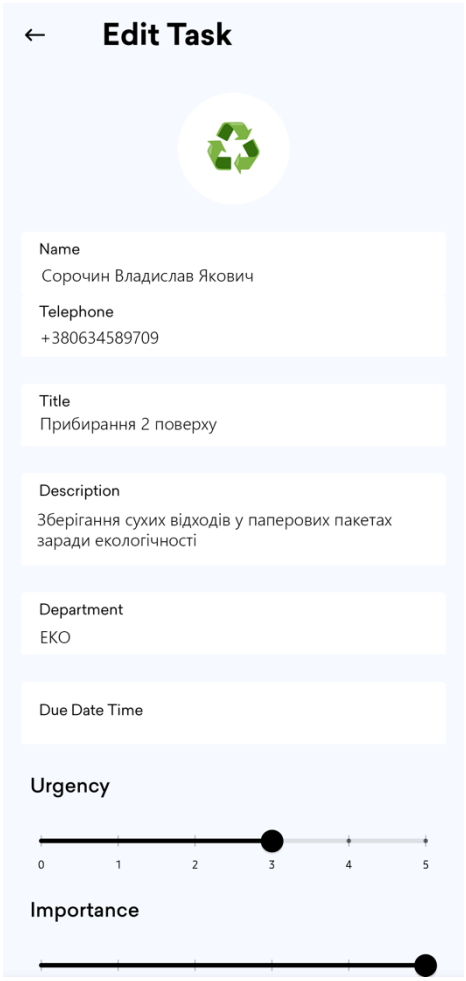


Рисунок 3.10 – Екран зміни даних завдання з полями

Текстові поля можна редагувати. Варто відмітити, що зверху є поле для емоджі, що надає завданням окремого швидкого розпізнання. Це зручно в ситуаціях, коли потрібно знайти завдання серед багатьох не читаючи тексту, а також зменшує когнітивне навантаження мозку. Останнім, але не менш важливим пунктом є налаштування дат: при встановленні дати, автоматично прийде повідомлення про її наближення, тобто нагадування до виконання (рис. 3.11).

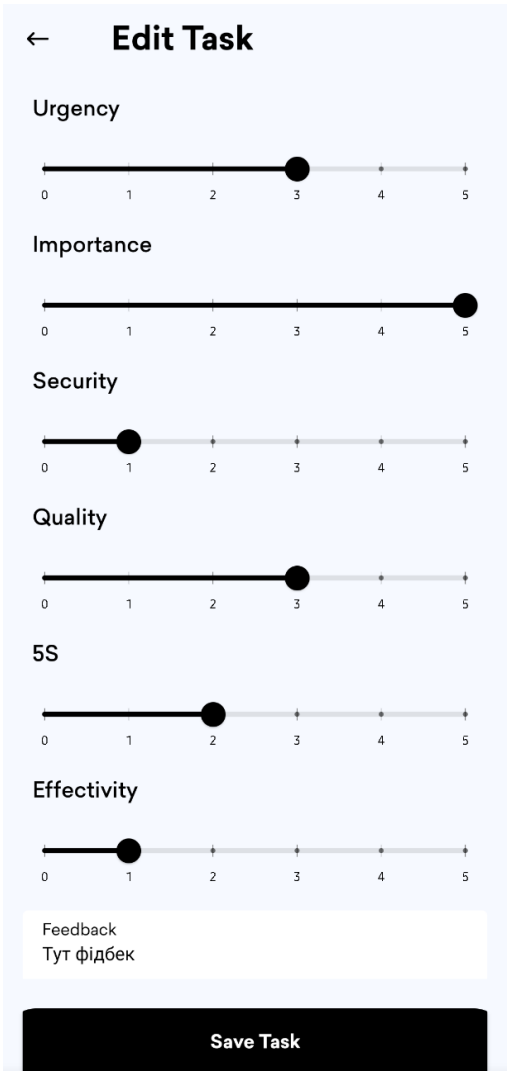


Рисунок 3.11 – Екран зміни даних завдання з оцінкою категорій

Нижче розташовані повзунки із категоріями, які потрібно відмічати оцінкою 0-5 балів. Важливість та швидкість потрібні для допомоги вирахування категорії завдання: чи потрібно зробити його якнайшвидше, що можна відкласти, а за що варто взагалі не братися. У самому низу розташоване поля для фідбеку (рис. 3.12). Примітки можна під’єднати до боту та відправити до людини, яка надіслала завдання. При настисканні «зберегти завдання» одразу екран перемикається на попередній. Якщо завдання переглянуте, можна натиснути зелену кнопку «Виконано», яка зміниться на «Невиконано». Цю дію так само можна зробити з екрану усіх завдань. Тоді чекбокс зміниться на відмічений, а текстові поля стануть закресленими.

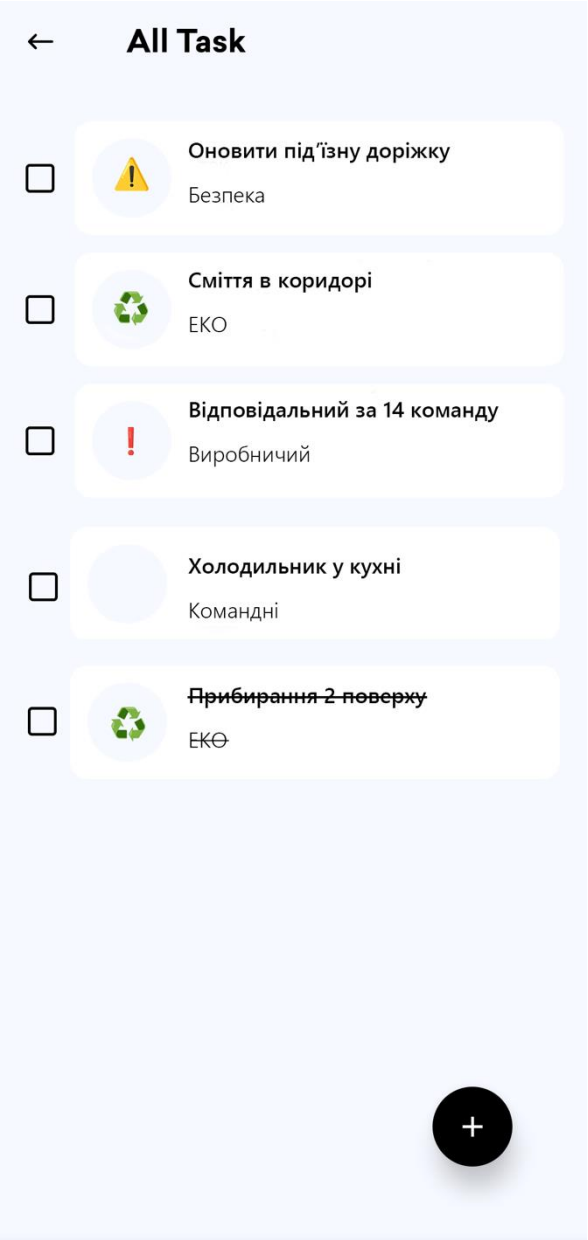


Рисунок 3.12 – Екран усіх завдань із одним виконаним

Такі самі завдання, як на рис. 3.8 можна додати за допомогою чорної кнопки.

Усі завдання можна сортувати за іншими категоріями, тобто проектами. Один проект може включати в себе декілька завдань з одією позначкою. Проекти можна виділяти для себе емоджі (рис. 3.13).

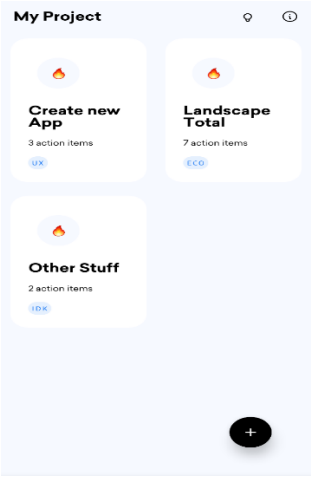


Рисунок 3.13 – Екран проектів додатку «Magic Box»

Додаток має два режими дизайну – денний та нічний. Різне оформлення потрібне для кращого сприйняття продукту користувачем, наприклад вночі краще для очей буде нічний режим із меншим контрастом та темними кольорами. Змінити режим можливо за допомогою кнопки із зображенням лампочки у верхньому меню (рис. 3.14).

Рисунок 3.14 – Екран перегляду даних завдання у нічному режимі

При натисканні кнопки із символом інформації у правому верхньому кутку поруч із лапочкою, можна побачити інформаційне вікно додатку (рис. 3.15).

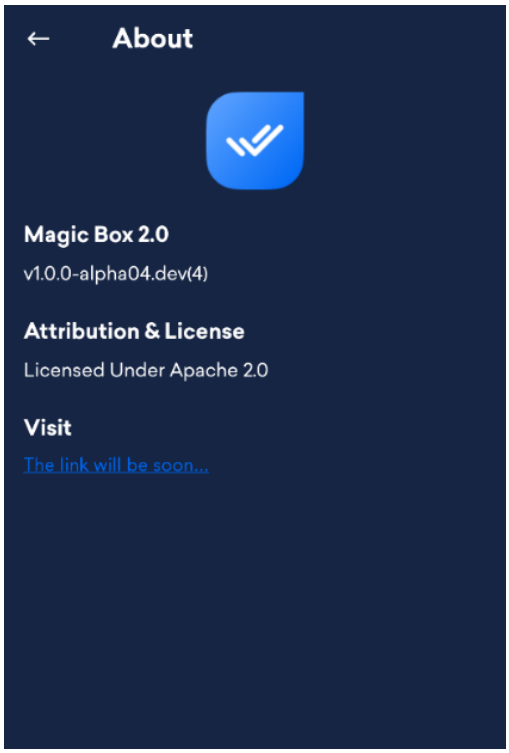


Рисунок 3.15 – Інформаційний екран у нічному режимі

Під час тестування додатку було підтверджено його коректну роботу та відображення складових частин. Виправлені неточності у даних під час опису складових додатку. В результаті, мобільний застосунок працює справно та виконує поставлену задачу.

Проведемо порівняння функціональних можливостей програм для управління проектами (табл. 3.2)

Таблиця 3.2 – порівняння функціональних можливостей програм для управління проектами

	Додавання завдань	Управління виконаням завдань	Наявність нічного режиму	Групування завдань	Можливість надсилання завданнь у месенджер та назад

Створена програма «Magic Box»	+	+	+	+	+
Worksection	+	+	-	-	-
Lean	+	+	-	-	-
LeanApp	+	+	+	-	-

З табл. 3.2 видно, що новий програмний продукт володіє додатковим функціоналом: наявність нічного режиму, що є зручним для використання в мобільних пристроях: Групування завдань по категоріях, це корисно, тому що один проект може включати в себе декілька завдань з одією позначкою; Можливість надсилання завданнь у месенджер і назад (наприклад у Telegram), це зручно, тому що користувачі можіть відповідати напряму з месенджера в дане програмне забезпечення.

3.7 Висновок до розділу 3

В даному розділі було розроблено структуру програмного забезпечення для управління проектами. Визначено основні модулі зі сторони адміністраторської та клієнтської частин. Було побудовано структурну схему взаємодії модулів клієнтської частини та побудовано діаграму прецедентів програмногозабезпечення. Також було створено діаграму станів та діаграму активності\діяльності програмного забезпечення для управління проектами. Обгрунтовано вибір мови програмування та середовища розробки. Визначено мову для розробки додатку – об'єктно-орієнтовану мову програмування Kotlin. Обраними програмними засобами було реалізовано інформаційну технології управління проектами у вигляді телеграм боту. Додаток протестовано та проаналізовано результати його роботи.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота «Інформаційна технологія управління проектами» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія управління проектами» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [19].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	5
2. Ринкові переваги (наявність аналогів)	2	3	3
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	3	4	5
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	39	44	45
Середньоарифметична сума балів $СБ_c$	42,7		

За результатами розрахунків, наведених в табл. 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [19].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія управління проектами» становить 42,7 бала, що, відповідно до табл. 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [20]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} \,, \tag{4.2}$$

де $I_{нi}$ та $I_{на}$ – чисельні значення конкретного *i*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \quad (4.3)$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до табл. 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Тактова частота процесора не нижче	Ггц	2,7	1,8	1,5	0,25
Оперативна пам'ять не менше	Мб	8	4	2	0,3
Об'єм диску	Гб	100	20	5	0,15
Мережевий канал	бал	2	8	4	0,1
Швидкість мережевого каналу	Мб/с	100	56	1,95	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1{,}5 \cdot 0{,}25 + 2 \cdot 0{,}3 + 5 \cdot 0{,}15 + 4 \cdot 0{,}1 + 1{,}95 \cdot 0{,}2 = 2{,}52.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,52 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія управління проектами», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [19]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 20500{,}00 \cdot 44 / 22 = 41000{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту розробки та дослідження інформаційної технології управління проектами	20500,00	931,82	44	41000,00
Консультант (Project Manager аналітик)	18250,00	829,55	11	9125,00
Інженер-програміст 1-ї категорії	17000,00	772,73	42	32454,55
Технік	7300,00	331,82	42	13936,36
Всього				96515,91

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія управління проектами» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{4.5}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{4.6}$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [19];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}35 / (22 \cdot 8) = 56{,}53$ *грн.*

$З_{р1} = 56{,}53 \cdot 10{,}00 = 565{,}31$ *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення обладнання для проведення досліджень	10,00	2	1,10	56,53	565,31
Інсталяція програмного забезпечення розробки програмного забезпечення	6,50	3	1,35	69,38	450,97
Встановлення цифрових обчислювальних систем	4,20	4	1,50	77,09	323,77

Відлагодження програмних модулів формування завдань	5,75	5	1,70	87,37	502,36
Підготовка тестового дослідження	3,20	4	1,50	77,09	246,68
Формування бази даних результатів випробування системи	12,00	3	1,35	69,38	832,55
Всього					2921,64

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (96515,91 + 2921,64) · 11 / 100% = 10938,13 грн.

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (96515,91 + 2921,64 + 10938,13) · 22 / 100% = 24282,65 грн.

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія управління проектами».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (4.9)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 3,0 \cdot 225,00 \cdot 1,05 - 0 \cdot 0 = 708,75$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	225,00	3,0	0	0	708,75
Папір для заміток (А5)	116,00	3,0	0	0	365,40
Начиння канцелярське	195,00	3,0	0	0	614,25
Органайзер офісний	183,00	3,0	0	0	576,45
Картридж для принтера	950,00	2,0	0	0	1995,00
Диск оптичний	25,00	6,0	0	0	157,50

USB-пам'ять Microtech 64 GB	199,00	2,0	0	0	417,90
Тека для паперів BBC-QT	89,90	3,0	0	0	283,19
Всього					5118,44

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія управління проектами», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 1 \cdot 1499{,}00 \cdot 1{,}05 = 1573{,}95$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішній жорсткий диск 2.5" 1TB Seagate (STGD2000200)	1	1499,00	1573,95
Концентратор Defender SEPTIMA SLIM (83505)	1	720,00	756,00
Кабель для передачі даних USB to COM 1.0m Patron (CAB-PN-USB-COM)	1	369,90	388,40
Всього			2718,35

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 7999,00 \cdot 2 \cdot 1,05 = 16797,90$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Смартфон Xiaomi Redmi 9 Pro	2	7999,00	16797,90
Всього			16797,90

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 4890,00 \cdot 1 \cdot 1,05 = 5134,50$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище програмування PyCharm	1	4890,00	5134,50
Всього			5134,50

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (42899,00 \cdot 2) / (2 \cdot 12) = 3574,92$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер Core I5-9600	42899,00	2	2	3574,92
Робоче місце розробника ПЗ	8570,00	5	2	285,67
Пристрої виводу інформації	9799,00	4	2	408,29
Оргтехніка	9240,00	4	2	385,00
Приміщення лабораторії розробки та дослідження інформаційної технології	356000,00	20	2	2966,67
ОС Windows 11	5800,00	2	2	483,33
Прикладний пакет Microsoft Office 2021	5100,00	2	2	425,00
Обладнання передачі цифрових даних даних	9650,00	5	2	321,67
Всього				8850,54

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Це = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,40 \cdot 320,0 \cdot 7,50 \cdot 0,95 / 0,97 = 960,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер Core I5-9600	0,40	320,0	960,00
Робоче місце розробника ПЗ	0,08	320,0	192,00
Пристрої виводу інформації	0,12	5,0	4,50
Оргтехніка	0,32	2,5	6,00
Обладнання передачі цифрових даних даних	0,15	320,0	360,00
Всього			1522,50

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія управління проектами» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-

правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (4.15)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (96515,91 + 2921,64) · 20 / 100% = 19887,51 грн.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (4.16)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$В_{сп}$ = (96515,91 + 2921,64) · 35 / 100% = 34803,14 грн.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (4.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

$І_в$ = (96515,91 + 2921,64) · 55 / 100% = 54690,65 грн.

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.18)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 105%.

$В_{нзв}$ = (96515,91 + 2921,64) · 105 / 100% = 104409,42 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія управління проектами» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.19)$$

$В_{заг}$= 96515,91 +2921,64 +10938,13 +24282,65 +5118,44 +2718,35 +16797,90 + 5134,50 + 8850,54 +1522,50 +19887,51 +34803,14 +54690,65 +104409,42 = =388591,27 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.20)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

$$ЗВ = 388591,27 / 0,95 = 409043,45 \text{ грн.}$$

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія управління проектами» передбачають комерціалізацію протягом 4-х років реалізації на ринку і відповідають ситуації «Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання споживачем».

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	200	450	550	450

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 1800 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 65800,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 3687,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [19]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.21)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 35\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (3687,00 \cdot 1800,00 + 69487,00 \cdot 200) \cdot 0,83 \cdot 0,35 \cdot (1\text{-}0,18/100\%) = 4891404,14$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (3687,00 \cdot 1800,00 + 69487,00 \cdot 650) \cdot 0,83 \cdot 0,35 \cdot (1\text{-}0,18/100\%) = 12340028,36$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (3687,00 \cdot 1800,00 + 69487,00 \cdot 1200) \cdot 0,83 \cdot 0,35 \cdot (1\text{-}0,18/100\%) = 21443902,41$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (3687,00 \cdot 1800,00 + 69487,00 \cdot 1650) \cdot 0,83 \cdot 0,35 \cdot (1\text{-}0,18/100\%) = 28892526,63$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.22)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,14;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 4891404,14/(1+0,14)^1+12340028,36/(1+0,14)^2+21443902,41/(1+0,14)^3+$$
$$+28892526,63/(1+0,14)^4=4290705,39+9495251,12+14474023,32+17106695,18=$$
$$=45366675,02 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.23}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 409043,45 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 409043,45 = 818086,89 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.24}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 45366675,02 грн;

PV – теперішня вартість початкових інвестицій, 818086,89 грн.

$E_{абс} = ПП - PV$ = 45366675,02 - 818086,89 = 44548588,12 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.25)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 44548588,12 грн;

PV – теперішня вартість початкових інвестицій, 818086,89 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+44548588,12/818086,89)$^{1/4}$= 1,73.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.26)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,09;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,35.

$\tau_{мін}$ = 0,09+0,35 = 0,44 < 1,73 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія управління проектами» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.27)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,73 = 0,58 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновок до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія управління проектами» становить 42,7 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,52 рази.

Також термін окупності становить 0,58 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія управління проектами».

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію управління проектами.

В першому розділі проаналізовано предметну область управління проектами, розглянуті популярні технології розробки ПЗ та типовий життєвий цикл розробки. Виходячи з проведеного аналізу гнучких методів розробки програмного забезпечення можна зробити висновки, щл жодна з сучасних гнучких технологій не є ідеальною і не існує технології, яка зможе бути єфективною на проектах різних типів.

В другому розділі проаналізовані робочі процеси технології Scrum, описано особливості управління ІТ проектів, розглянуто адаптації технології відчизняними компаніями. Також в результаті порівняльної характеристики вибрано технологію для вдосконалення.

В третьому розділі було розроблено структуру програмного забезпечення для управління проектами. Визначено основні модулі зі сторони серверної та клієнтської частин. Було побудовано структурну схему взаємодії модулів клієнтської частини та побудовано діаграму прецедентів програмногозабезпечення. Також було створено діаграму станів та діаграму діяльності програмного забезпечення для управління проектами. Обгрунтовано вибір мови програмування та середовища розробки. Визначено мову для розробки додатку – об'єктно-орієнтовану мову програмування Kotlin. Обраними програмними засобами було реалізовано інформаційну технології управління проектами у вигляді телеграм боту. Додаток протестовано та проаналізовано результати його роботи. Тестування додатку підтвердило, програмний продукт володіє додатковим функціоналом: наявність нічного режиму, що є зручним для використання в мобільних пристроях: Групування завдань по категоріях, це корисно, тому що один проект може включати в себе декілька завдань з одією позначкою; Можливість надсилання завданнь у месенджер і назад (наприклад у Telegram), це зручно, тому що користувачі можуть відповідати напряму з месенджера в дане програмне забезпечення. Результати дослідження впроваджено в роботу ТОВ «ІТІ».

Отже всі задачі дослідження виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. І.О. Остапчук, В.С. Озеранський. Особливості управління ІТ-проектами. Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» − [Електронний ресурс]. – https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/19619/16236.
2. Вплив agile методологій на розробку ПЗ: [Електронний ресурс]. – Режим доступу: https://www.ccsenet.org/journal/index.php/cis/article/view/44383/
3. Agile маніфест: [Електронний ресурс]. – Режим доступу: https://agilemanifesto.org/
4. 12 принципів Agile: [Електронний ресурс]. – Режим доступу: https://agilemanifesto.org/principles.html
5. Найкращі інструменти для спільної роботи у 2021 році: [Електронний ресурс]. – Режим доступу: https://blog.jetbrains.com/space/2021/07/16/best-collaboration-tools/
6. Scrum управління проектами: [Електронний ресурс]. – Режим доступу: https://www.digite.com/agile/scrum-methodology/
7. Scrum Guide: [Електронний ресурс]. – Режим доступу: https://scrumguides.org/
8. Все про управління ІТ проектами: [Електронний ресурс]. – Режим доступу: https://www.cleverism.com/everything-about-project-management/
9. ІТ в Україні: куди ми рухаємося? [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/columns/future-of-it-ukraine/
10. Що таке Канбан: [Електронний ресурс]. – Режим доступу: https://www.digite.com/kanban/what-is-kanban/
11. Що таке Project Management: [Електронний ресурс]. – Режим доступу: https://www.pmi.org/about/learn-about-pmi/what-is-project-management
12. Чому Project Management важливий для сучасних проектів: [Електронний ресурс]. – Режим доступу: https://www.lucidchart.com/blog/why-is-project-management-important/
13. Плюси та мінуси Scrum: [Електронний ресурс]. – Режим доступу: https://www.simplilearn.com/scrum-project-management-article

14. Що не так з сучасним Agile: [Електронний ресурс]. – Режим доступу: https://www.workamajig.com/blog/project-management-methodologies
15. Створення чат-бота у Telegram: [Електронний ресурс]. – Режим доступу: https://sendpulse.ua/knowledge-base/chatbot/telegram/create-telegram-chatbot.
16. Управління проєктами: навч. посібник / за ред. О.В Ульянченка та П.Ф. Цигікала. — Харків: ХНАУ ім. В.В. Докучаєва, 2010. — 522 с.
17. Як на писати Telegram-бота на Ruby: [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/telegram-bot-ruby/.
18. Крижановський Є.М. Моделювання бізнес-процесів та управління ІТ-проєктами / Є.М. Крижановський, А.Р. Ящолт, С.О. Жуков, О.М. Козачко — Вінниця: ВНТУ, 2018. — 91 с.
19. Atomic Kotlin / Bruce Eckel,., 2021. – 431 с.
20. Android Studio [Електронний ресурс] – Режим доступу: https://developer.android.com/studio.
21. Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/.
22. Firebase Cloud Storage [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/products/firestore
23. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
24. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія управління проектами

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93,6% Схожість 6,4%

Аналіз звіту подібності (відмітити потрібне):

✓ 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 виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 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.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Остапчук І.О.

Керівник роботи Озеранський В.С.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

MainApp.kt

```
import android.app.Application
import androidx.hilt.work.HiltWorkerFactory
import androidx.work.Configuration
import com.google.firebase.analytics.FirebaseAnalytics
import dagger.hilt.android.HiltAndroidApp
import logcat.AndroidLogcatLogger
import logcat.LogPriority
import javax.inject.Inject

@HiltAndroidApp
class EinsenApp : Application(), Configuration.Provider {

    @Inject
    lateinit var mFirebaseAnalytics: FirebaseAnalytics

    @Inject
    lateinit var workerFactory: HiltWorkerFactory

    override fun getWorkManagerConfiguration() =
        Configuration.Builder()
            .setWorkerFactory(workerFactory)
            .build()

    override fun onCreate() {
        super.onCreate()
        // Log all priorities in debug builds, no-op in release builds.
        AndroidLogcatLogger.installOnDebuggableApp(this, minPriority = LogPriority.VERBOSE)

        // Obtain the FirebaseAnalytics instance.
        mFirebaseAnalytics = FirebaseAnalytics.getInstance(this)
    }
}
```

NavGraph.kt

```
import androidx.compose.runtime.Composable
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.hilt.navigation.HiltViewModelFactory
import androidx.hilt.navigation.compose.hiltViewModel
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import androidx.navigation.NavType
import androidx.navigation.navArgument
import com.google.accompanist.navigation.animation.AnimatedNavHost
import com.google.accompanist.navigation.animation.composable
import com.google.accompanist.navigation.animation.rememberAnimatedNavController
import com.google.accompanist.navigation.material.ModalBottomSheetLayout
import com.google.accompanist.navigation.material.bottomSheet
import com.google.accompanist.navigation.material.rememberBottomSheetNavigator
```

```
/**
 * Single source for Navigation Routes of this app.
 */
object EndPoints {
    const val ID = "id"
    const val PRIORITY = "priority"
    const val URL = "url"
    const val TITLE = "title"
    const val EMOJI = "emoji"
}

object QueryParams {
    const val URGENCY = "urgency"
    const val IMPORTANCE = "importance"
    const val SECURITY= "security"
    const val EFFECTIVITY = "effectivity"
    const val FIVES = "fives"
    const val QUALITY = "quality"
}

object EinsenModifier {
    val modifier: Modifier = Modifier
}

@Composable
fun NavGraph(toggleTheme: () -> Unit) {
    val bottomSheetNavigator = rememberBottomSheetNavigator()
    val navController = rememberAnimatedNavController(bottomSheetNavigator)
    val actions = remember(navController) { MainActions(navController) }

    ModalBottomSheetLayout(bottomSheetNavigator) {

        AnimatedNavHost(navController, startDestination = Screen.Dashboard.route) {

            /**
             * Navigates to [SplashScreen].
             */
            composable(Screen.Splash.route) {
                SplashScreen(EinsenModifier.modifier, actions)
            }

            /**
             * Navigates to [Project].
             */
            composable(
                Screen.Project.route
            ) {
                val viewModel = hiltViewModel<MainViewModel>(it)
                ProjectScreen(EinsenModifier.modifier, viewModel, actions, toggleTheme)
            }

            /**
             * Navigates to [Dashboard].
             */
            composable(
                Screen.Dashboard.route
            ) {
                val viewModel: MainViewModel = viewModel(
```

```
            factory = HiltViewModelFactory(LocalContext.current, it)
        )
        viewModel.getAllTask()
        DashboardScreen(
            EinsenModifier.modifier,
            viewModel,
            actions,
            toggleTheme
        )
    }

    /**
     * Navigates to [AddTask].
     */
    composable(
        "${Screen.AddTask.route}?urgency={urgency}&importance={importance}",
        arguments = listOf(
            navArgument(QueryParams.URGENCY) {
                type = NavType.IntType
                defaultValue = 0
            },
            navArgument(QueryParams.IMPORTANCE) {
                type = NavType.IntType
                defaultValue = 0
            },
            navArgument(QueryParams.FIVES) {
                type = NavType.IntType
                defaultValue = 0
            },navArgument(QueryParams.SECURITY) {
                type = NavType.IntType
                defaultValue = 0
            },navArgument(QueryParams.QUALITY) {
                type = NavType.IntType
                defaultValue = 0
            },navArgument(QueryParams.EFFECTIVITY) {
                type = NavType.IntType
                defaultValue = 0
            }
        )
    ) {
        val viewModel = hiltViewModel<MainViewModel>(it)
        navController.currentBackStackEntry?.savedStateHandle?.getLiveData<String>(EndPoints.EMOJI)
            ?.observeForever { result ->
                viewModel.currentEmoji(result)
            }

        val defaultUrgency = it.arguments?.getInt(QueryParams.URGENCY) ?: 0
        val defaultImportance = it.arguments?.getInt(QueryParams.IMPORTANCE) ?: 0
        val defaultEffectivity = it.arguments?.getInt(QueryParams.EFFECTIVITY) ?: 0
        val defaultFiveS= it.arguments?.getInt(QueryParams.FIVES) ?: 0
        val defaultQuality = it.arguments?.getInt(QueryParams.QUALITY) ?: 0
        val defaultSecurity = it.arguments?.getInt(QueryParams.SECURITY) ?: 0

        AddTaskScreen(
            EinsenModifier.modifier,
            viewModel,
            actions,
            defaultUrgency,
            defaultImportance,
```

```
            defaultSecurity,
            defaultQuality,
            defaultFiveS,
            defaultEffectivity
        )
    }

    /**
     * Navigates to [AllTask].
     * @param priority
     */
    composable(
        "${Screen.AllTask.route}/{priority}",
        arguments = listOf(navArgument(EndPoints.PRIORITY) { type = NavType.StringType })
    ) {
        val viewModel = hiltViewModel<MainViewModel>(it)

        val priority = it.arguments?.getString(EndPoints.PRIORITY)
            ?: throw IllegalStateException("'Priority' shouldn't be null")

        viewModel.getTaskByPriority(priority = priority)

        val defaultUrgencyImportance = getUrgencyImportanceFromPriority(priority)

        AllTaskScreen(
            EinsenModifier.modifier,
            viewModel,
            actions,
            defaultUrgencyImportance.first,
            defaultUrgencyImportance.second
        )
    }

    /**
     * Navigates to [TaskDetails].
     * @param [id]
     */
    composable(
        "${Screen.TaskDetails.route}/{id}",
        arguments = listOf(navArgument(EndPoints.ID) { type = NavType.IntType })
    ) {
        val viewModel = hiltViewModel<MainViewModel>(it)
        val taskID = it.arguments?.getInt(EndPoints.ID)
            ?: throw IllegalStateException("'Task ID' shouldn't be null")

        viewModel.findTaskByID(taskID)
        TaskDetailsScreen(EinsenModifier.modifier, viewModel, actions)
    }

    /**
     * Navigates to [EditTask].
     * @param id
     */
    composable(
        "${Screen.EditTask.route}/{id}",
        arguments = listOf(navArgument(EndPoints.ID) { type = NavType.IntType })
    ) {
        val viewModel = hiltViewModel<MainViewModel>(it)
        val taskID = it.arguments?.getInt(EndPoints.ID)
```

```
            ?: throw IllegalStateException("'Task ID' shouldn't be null")

        navController.currentBackStackEntry?.savedStateHandle?.getLiveData<String>(EndPoints.EMOJI)
            ?.observeForever { result ->
                viewModel.currentEmoji(result)
            }
        viewModel.findTaskByID(taskID)
        EditTaskScreen(EinsenModifier.modifier, viewModel, actions)
    }

    /**
     * Navigates to [About].
     */
    composable(
        Screen.About.route
    ) {
        val viewModel = hiltViewModel<MainViewModel>(it)
        AboutScreen(EinsenModifier.modifier, viewModel, actions)
    }

    /**
     * Navigates to [WebView].
     * @param title
     * @param url
     */
    composable(
        "${Screen.WebView.route}/{title}/{url}",
        arguments = listOf(
            navArgument(EndPoints.TITLE) { type = NavType.StringType },
            navArgument(EndPoints.URL) { type = NavType.StringType }
        )
    ) {
        val viewModel = hiltViewModel<MainViewModel>(it)
        val url = it.arguments?.getString(EndPoints.URL)
            ?: throw IllegalStateException("'URL' shouldn't be null")
        val title = it.arguments?.getString(EndPoints.TITLE)
            ?: throw java.lang.IllegalStateException("'Title' shouldn't be null")
        WebViewScreen(
            modifier = EinsenModifier.modifier,
            title = title,
            url = url,
            actions = actions,
            viewModel
        )
    }

    /**
     * Navigates to [AllEmoji].
     * @param viewModel
     * @param actions
     * @param onSelect
     */
    bottomSheet(route = Screen.AllEmoji.route) {
        val viewModel = hiltViewModel<MainViewModel>(it)
        AllEmojiScreen(
            EinsenModifier.modifier,
            viewModel,
            actions,
            onSelect = { selectedEmoji ->
```

```
                    navController.previousBackStackEntry?.savedStateHandle?.set(
                        EndPoints.EMOJI,
                        selectedEmoji
                    ).apply {
                        actions.popBackStack.invoke()
                    }
                }
            )
        }
    }
}

/**
 * A class to define Navigation Route to All Flows of this app with the help of [NavController]
 * @param navController
 */
class MainActions(navController: NavController) {

    val upPress: () -> Unit = {
        navController.navigateUp()
    }

    val popBackStack: () -> Unit = {
        navController.popBackStack()
    }

    val gotoDashboard: () -> Unit = {
        navController.navigate(Screen.Dashboard.route)
    }

    val gotoAddTask: (urgency: Int?, importance: Int?) -> Unit = { urgency, importance ->
        navController.navigate("${Screen.AddTask.route}?urgency=$urgency&importance=$importance")
    }

    val gotoAllTask: (priority: Priority) -> Unit = { priority ->
        navController.navigate("${Screen.AllTask.route}/${priority.name}")
    }

    val gotoTaskDetails: (id: Int) -> Unit = { id ->
        navController.navigate("${Screen.TaskDetails.route}/$id")
    }

    val gotoEditTask: (id: Int) -> Unit = { id ->
        navController.navigate("${Screen.EditTask.route}/$id")
    }

    val gotoAbout: () -> Unit = {
        navController.navigate(Screen.About.route)
    }

    val gotoWebView: (title: String, url: String) -> Unit = { title, url ->
        navController.navigate("${Screen.WebView.route}/$title/$url")
    }

    val gotoAllEmoji: () -> Unit = {
        navController.navigate(Screen.AllEmoji.route)
    }
}
```

ProjectScreen.kt

```
import androidx.compose.foundation.background
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.lazy.GridCells
import androidx.compose.foundation.lazy.LazyVerticalGrid
import androidx.compose.material.FloatingActionButton
import androidx.compose.material.FloatingActionButtonDefaults
import androidx.compose.material.Icon
import androidx.compose.material.IconButton
import androidx.compose.material.Scaffold
import androidx.compose.material.Text
import androidx.compose.material.TopAppBar
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Add
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.core.os.bundleOf

@Composable
fun ProjectScreen(
    modifier: Modifier,
    viewModel: MainViewModel,
    actions: MainActions,
    toggleTheme: () -> Unit
) {

    val gridSize by remember { mutableStateOf(150.dp) }

    Scaffold(
        topBar = {
            TopAppBar(
                title = {
                    Text(
                        text = stringResource(id = R.string.text_project),
                        textAlign = TextAlign.Start,
                        style = AppTheme.typography.h2,
                        color = AppTheme.colors.text
                    )
                },
                actions = {

                    // log event to firebase
                    val bundle = bundleOf(
                        "theme_switch" to isSystemInDarkTheme()
                    )

                    IconButton(
```

```
                onClick = {
                    toggleTheme().run {
                        viewModel.firebaseLogEvent("project_theme_switch", bundle)
                    }
                }
            ) {

                Icon(
                    painter = when (isSystemInDarkTheme()) {
                        true -> painterResource(id = R.drawable.ic_bulb_on)
                        false -> painterResource(id = R.drawable.ic_bulb_off)
                    },
                    contentDescription = stringResource(R.string.text_bulb_turn_on),
                    tint = AppTheme.colors.primary
                )
            }

            Spacer(modifier = modifier.width(AppTheme.dimensions.paddingMedium))

            IconButton(
                onClick = {
                    actions.gotoAbout.invoke().run {
                        // log event to firebase
                        val aboutBundle = bundleOf(
                            "about_button" to "Clicked about button from Project"
                        )
                        viewModel.firebaseLogEvent("project_about_button", aboutBundle)
                    }
                }
            ) {
                Icon(
                    painter = painterResource(id = R.drawable.ic_about),
                    contentDescription = stringResource(R.string.text_bulb_turn_on),
                    tint = AppTheme.colors.primary
                )
            }
        },
        backgroundColor = AppTheme.colors.background, elevation = 0.dp
    )
},
floatingActionButton = {

    FloatingActionButton(
        modifier = modifier.padding(AppTheme.dimensions.paddingXXXL),
        onClick = {
            actions.gotoAddTask.invoke(0, 0).run {
                // log event to firebase
                val addTaskBundle = bundleOf(
                    "add_project" to "Clicked Add Project button from Workspace"
                )

                viewModel.firebaseLogEvent("project_add_workspace_button", addTaskBundle)
            }
        },
        backgroundColor = AppTheme.colors.primary,
        contentColor = AppTheme.colors.text,
        elevation = FloatingActionButtonDefaults.elevation(12.dp)
    ) {
        Icon(
```

```
                Icons.Filled.Add,
                contentDescription = stringResource(id = R.string.text_addTask),
                tint = AppTheme.colors.white
            )
        }
    }
) {

    // workspace dashboard layout goes here
    LazyVerticalGrid(
        cells = GridCells.Adaptive(minSize = gridSize),
        modifier
            .fillMaxSize()
            .background(AppTheme.colors.background)
    ) {

        item {
            ItemWorkspaceCard(
                modifier = modifier,
                title = "Create new App",
                itemCount = "3 action items",
                emoji = "\uD83D\uDD25",
                tag = "UX",
                onCardClick = {}
            ) {
            }
        }

        item {
            ItemWorkspaceCard(
                modifier = modifier,
                title = "Landscape Total",
                itemCount = "7 action items",
                emoji = "\uD83D\uDD25",
                tag = "ECO",
                onCardClick = {}
            ) {
            }
        }

        item {
            ItemWorkspaceCard(
                modifier = modifier,
                title = "Other Stuff",
                itemCount = "2 action items",
                emoji = "\uD83D\uDD25",
                tag = "IDK",
                onCardClick = {}
            ) {
            }
        }
    }
}
```

AnimationViewState.kt

```
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
```

```
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.wrapContentSize
import androidx.compose.material.Button
import androidx.compose.material.ButtonDefaults
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import com.airbnb.lottie.compose.LottieAnimation
import com.airbnb.lottie.compose.LottieCompositionSpec
import com.airbnb.lottie.compose.LottieConstants
import com.airbnb.lottie.compose.animateLottieCompositionAsState
import com.airbnb.lottie.compose.rememberLottieComposition

@Composable
fun AnimationViewState(
    modifier: Modifier = Modifier,
    title: String,
    description: String,
    callToAction: String,
    screenState: ScreenState,
    actions: () -> Unit
) {
    Column(
        modifier = modifier
            .fillMaxSize()
            .background(AppTheme.colors.background)
            .wrapContentSize(Alignment.Center),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        when (screenState) {
            ScreenState.ERROR -> {
                LottieAnimationPlaceHolder(
                    modifier,
                    title,
                    description,
                    callToAction,
                    actions,
                    R.raw.error_state
                )
            }
            ScreenState.EMPTY -> {
                LottieAnimationPlaceHolder(
                    modifier,
                    title,
                    description,
                    callToAction,
                    actions,
                    R.raw.empty_state
```

```
            )
        }
        ScreenState.LOADING -> {
            LottieAnimation(modifier, lottie = R.raw.loading_state)
        }
    }
  }
}

@Composable
fun LottieAnimationPlaceHolder(
  modifier: Modifier,
  title: String,
  description: String,
  callToAction: String,
  actions: () -> Unit,
  lottie: Int,
) {

  // lottie animation
  LottieAnimation(modifier, lottie)

  // title, description & CTA button
  Text(
    text = title,
    modifier = modifier.fillMaxWidth(),
    style = AppTheme.typography.h2,
    textAlign = TextAlign.Center,
    color = AppTheme.colors.text
  )
  Spacer(modifier = modifier.height(AppTheme.dimensions.paddingMedium))
  Text(
    text = description,
    modifier = modifier
      .fillMaxWidth()
      .padding(
        start = AppTheme.dimensions.paddingXXL,
        end = AppTheme.dimensions.paddingXXL
      ),
    style = AppTheme.typography.body,
    maxLines = 3,
    textAlign = TextAlign.Center,
    color = AppTheme.colors.text.copy(.7F)
  )
  Spacer(modifier = modifier.height(AppTheme.dimensions.paddingXXL))
  Button(
    onClick = { actions() },
    colors = ButtonDefaults.buttonColors(
      backgroundColor = AppTheme.colors.primary,
      contentColor = AppTheme.colors.white
    )
  ) {
    Text(text = callToAction, style = AppTheme.typography.button)
  }
}
```

Додаток В (обов'язковий)

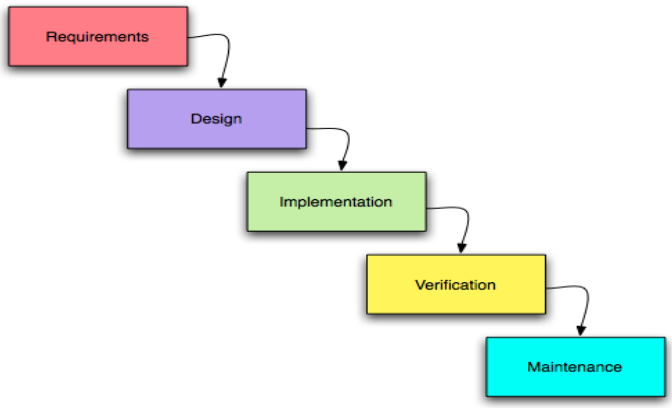
ІЛЮСТРАТИВНА ЧАСТИНА

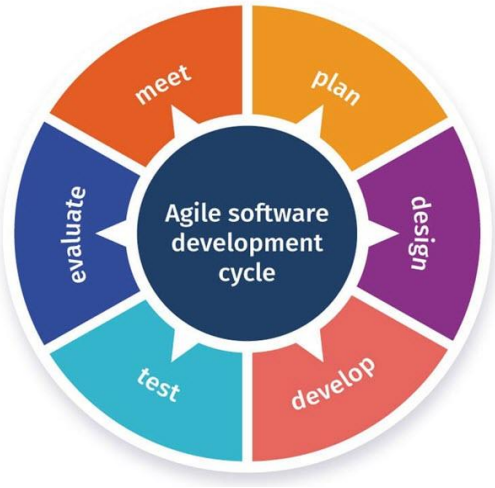
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ УПРАВЛІННЯ ПРОЕКТАМИ

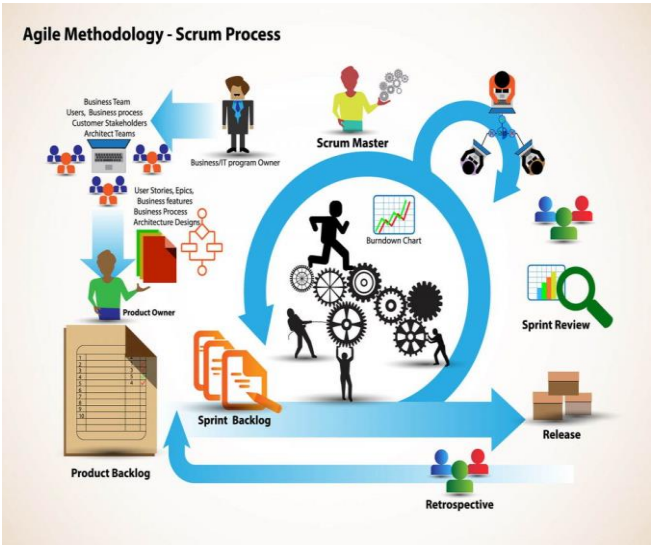
Виконала: студентка 2 курсу, групи 3КН-22м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Остапчук І.О.
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Озеранський В.С.
(прізвище та ініціали)
«7» 11. 2023 р.

Вінниця ВНТУ – 2023 рік

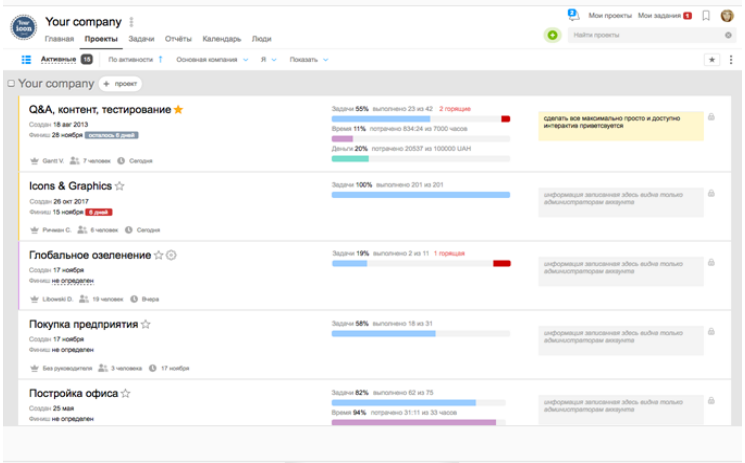
Існуючі технології управління проектами

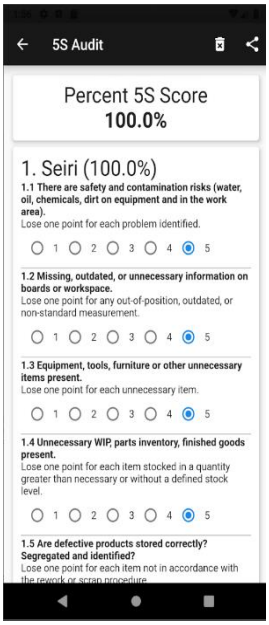




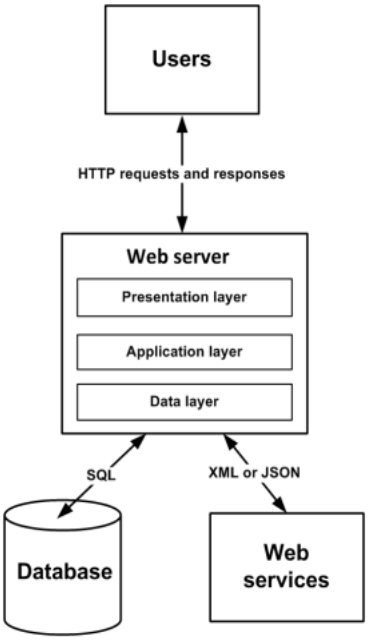


Програми аналоги для управління проектами



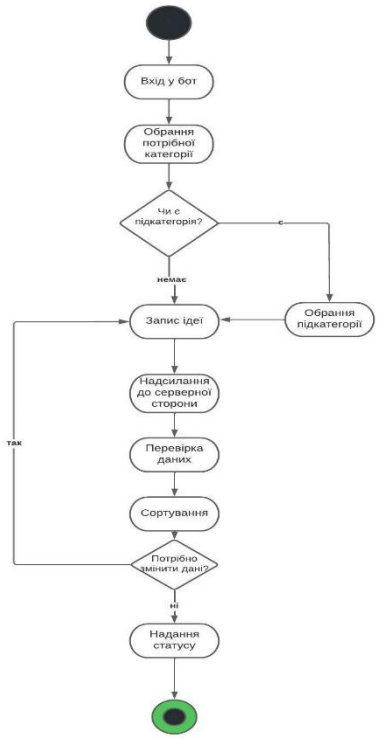


Використання клієнт-серверної архітектури для управління проектами

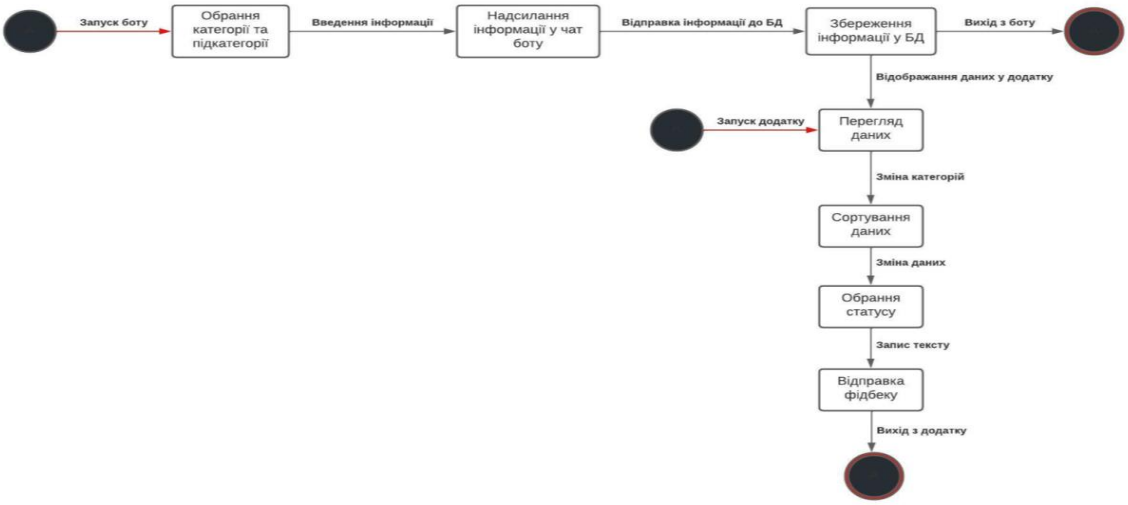


Діаграма прецедентів програмного забезпечення для управління проектами

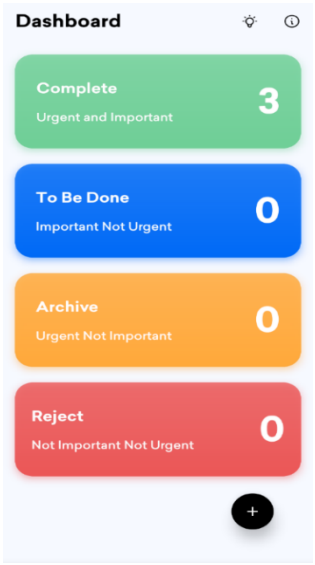
Діаграма діяльності програмного забезпечення для управління проектами

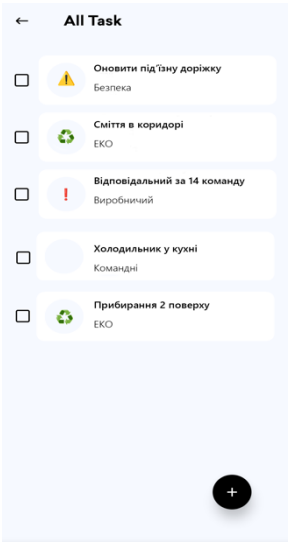


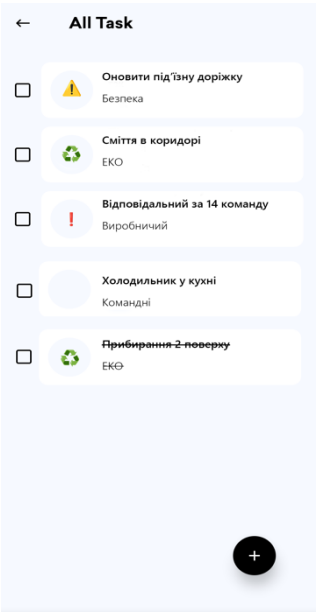
Діаграма станів програмного забезпечення для управління проектами

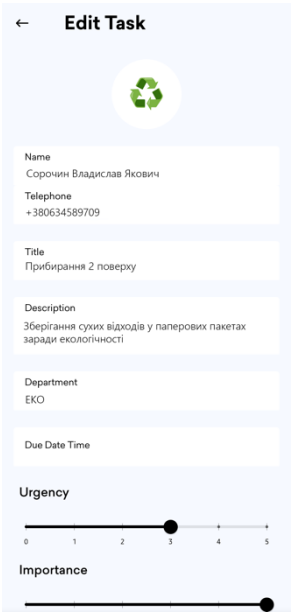


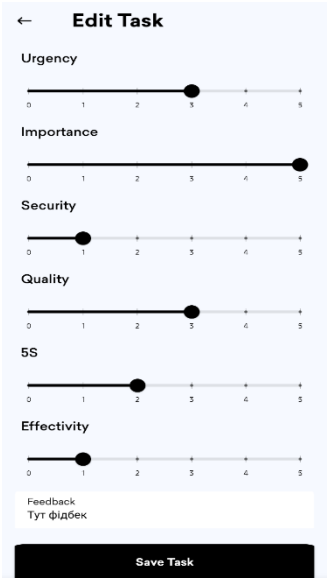
Інтерфейс програмного забезпечення для управління проектами

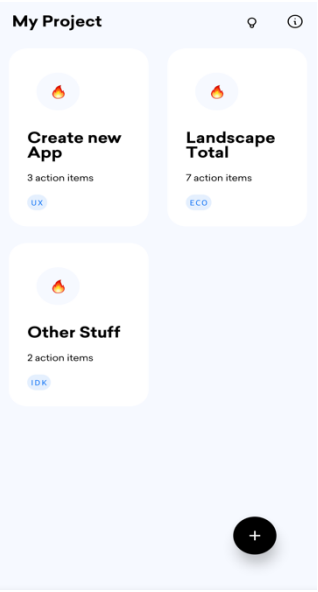


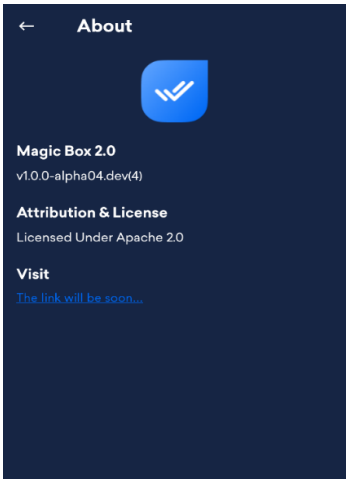


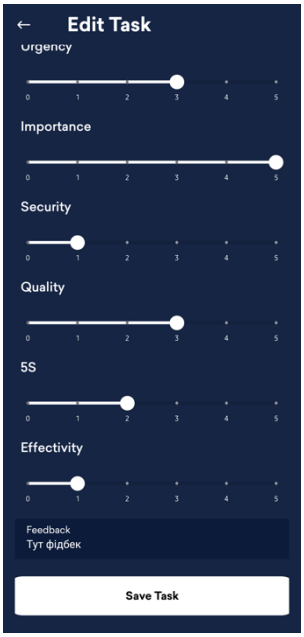












Додаток Г (довідниковий)

Інструкція користувача

При запуску додатку відображається екран з чотирма категоріями – статусом завдань (рис. Г.1).

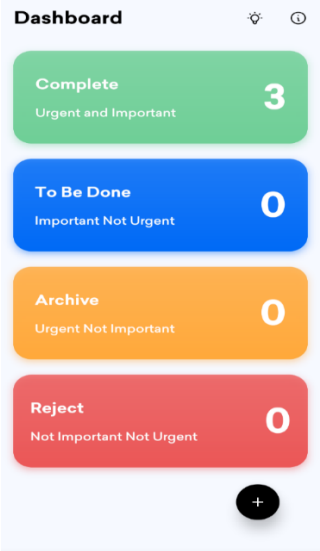


Рисунок Г.1 – Головний екран додатку «Magic Box»

При виборі одної з категорій відбудеться перехід до загальної кількості завдань у цій категорії. При натисканні кнопки з плюсом можна додати завдання із тими самими умовами (рис. Г.2).

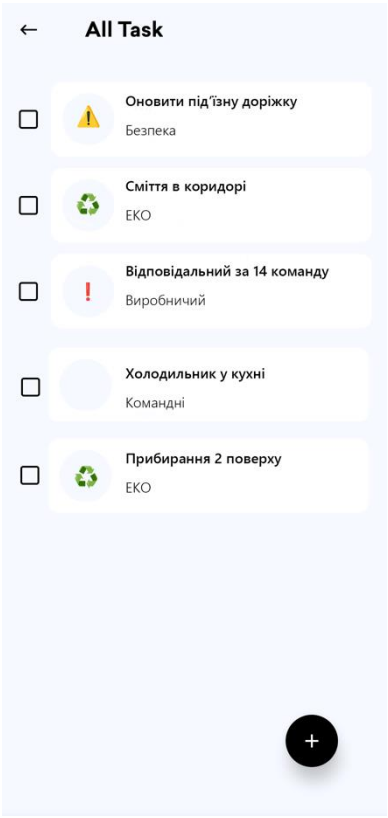


Рисунок Г.2 – Екран завдань додатку «Magic Box»

Далі обираємо будь-яке завдання, щоб переглянути його зміст (рис. Г.3).

Рисунок Г.3 – Екран перегляду даних завдання

Після перегляду адміністратор може відредактувати існуючі дані, поставити оцінку або змінити її, відмітити як виконане чи видалити. При натискуванні іконки «поділитись», текстова інформація щодо завдання може бути відправлена у різні месенджери чи електронну пошту (рис. Г.4).

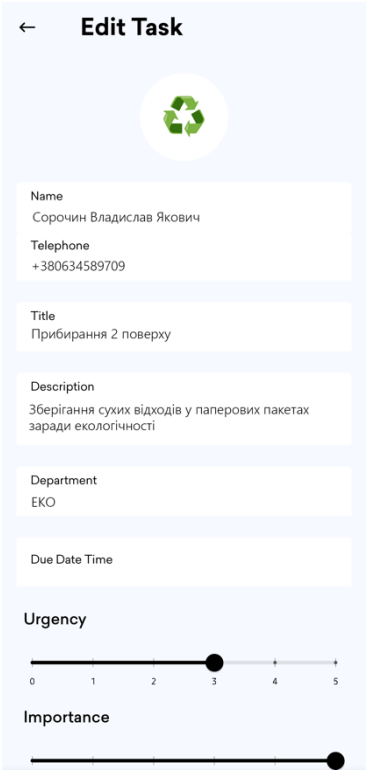


Рисунок Г.4 – Екран зміни даних завдання з полями

Текстові поля можна редагувати. Варто відмітити, що зверху є поле для емоджі, що надає завданням окремого швидкого розпізнання. Це зручно в ситуаціях, коли потрібно знайти завдання серед багатьох не читаючи тексту, а також зменшує когнітивне навантаження мозку. Останнім, але не менш важливим пунктом є налаштування дат: при встановленні дати, автоматично прийде повідомлення про її наближення, тобто нагадування до виконання (рис. Г.5).

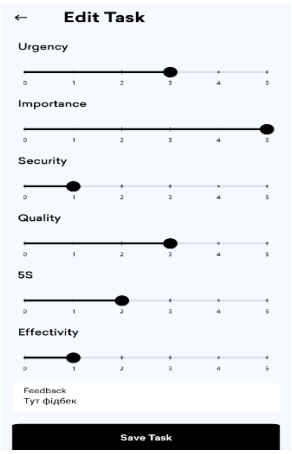


Рисунок Г.5 – Екран зміни даних завдання з оцінкою категорій

Нижче розташовані повзунки із категоріями, які потрібно відмічати оцінкою 0-5 балів. Важливість та швидкість потрібні для допомоги вирахування категорії завдання: чи потрібно зробити його якнайшвидше, що можна відкласти, а за що варто взагалі не братися. У самому низу розташоване поля для фідбеку (рис. Г.6). Примітки можна під’єднати до боту та відправити до людини, яка надіслала завдання. При настисканні «зберегти завдання» одразу екран перемикається на попередній. Якщо завдання переглянуте, можна натиснути зелену кнопку «Виконано», яка зміниться на «Невиконано». Цю дію так само можна зробити з екрану усіх завдань. Тоді чекбокс зміниться на відмічений, а текстові поля стануть закресленими.

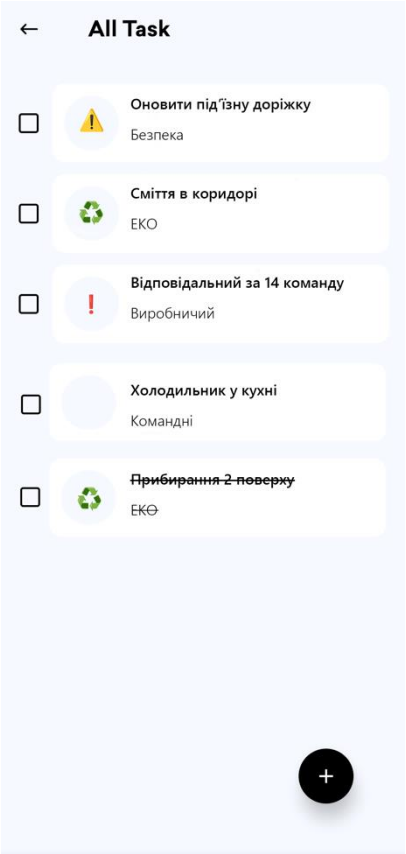


Рисунок Г.6 – Екран усіх завдань із одним виконаним

Такі самі завдання, як на рис. Г.2 можна додати за допомогою чорної кнопки.

Усі завдання можна сортувати за іншими категоріями, тобто проектами. Один проект може включати в себе декілька завдань з одією позначкою. Проекти можна виділяти для себе емоджі (рис. Г.7).

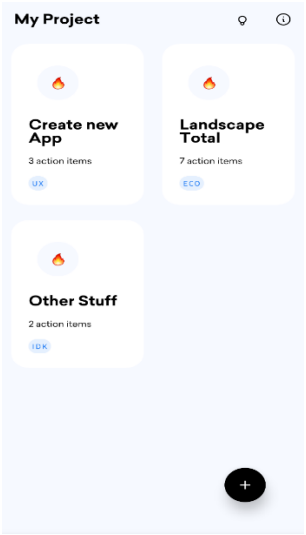


Рисунок Г.7 – Екран проектів додатку «Magic Box»

Додаток має два режими дизайну – денний та нічний. Різне оформлення потрібне для кращого сприйняття продукту користувачем, наприклад вночі краще для очей буде нічний режим із меншим контрастом та темними кольорами. Змінити режим можливо за допомогою кнопки із зображенням лампочки у верхньому меню (рис. Г.8).

Рисунок Г.8 – Екран перегляду даних завдання у нічному режимі

При натисканні кнопки із символом інформації у правому верхньому кутку поруч із лапочкою, можна побачити інформаційне вікно додатку (рис. Г.9).

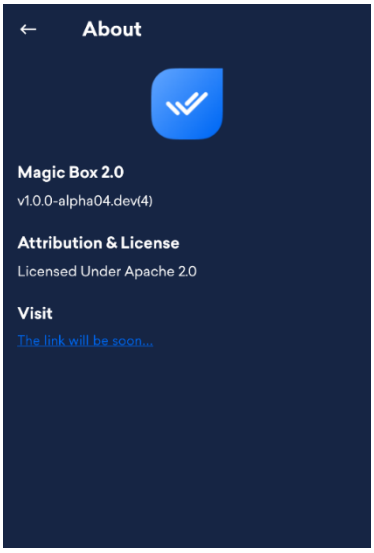


Рисунок Г.9 – Інформаційний екран у нічному режимі

Додаток Д (довідниковий)

Довідка про впровадження

МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ ____ від "__" _____ 20__ р.

Д О В І Д К А

Дана Остапчук Ірині Олександрівні в тому, що результати, одержані нею в процесі виконання магістерської кваліфікаційної роботи, а саме алгоритми та програмні засоби управління проектами, планується використати в розробках ТОВ «ІТІ».

Зам. директора ТОВ «ІТІ» Бодяк В.М.

ТОВ "ІТІ"