

Міністерство освіти і науки України
Нац університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Технології створення програмних продуктів»

за темою

«Розробка мобільного додатку «Електронний діловий
щоденник»»»

Пояснювальна записка до етапів визначення вимог до програмного продукту та
планування процесів розробки програмного продукту

Виконав:

студент 3-го курсу

групи AI-194

Кулик В.О.

Перевірив:

Блажко О. А.

Анотація

В курсовій роботі розглядається процес створення програмного продукту «Розробка мобільного додатку «Електронний діловий щоденник»» на етапах визначення вимог до програмного продукту та планування процесів розробки.

Робота виконувалась в команді з одного учасника: Кулик В.О..

Робота пов'язана з такими матеріальними потребами споживача, як зекономити час на не потрібних, не пріоритетних задачах. Аналіз вказаних потреб визначив інформаційну потребу – отримати інформацію для кращого розуміння, як краще зекономити час на задачах.

При визначені ступеня готовності існуючих програмних продуктів до вирішення інформаційної потреби проаналізовано наступні програмні продукти: Any.do, Todoist!, Do!.

Поточну версію пояснювальної записки до результатів роботи розміщено на *GitHub*-репозиторії за адресою:
https://github.com/VladislavKulik/-Software_Requirements_and_Planning

Перелік скорочень

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП – програмний продукт

UML – уніфікована мова моделювання

Зміст

	стор.
1 Вимоги до програмного продукту	6
1.1 Визначення потреб споживача	7
1.1.1 Ієрархія потреб споживача	
1.1.2 Деталізація матеріальної потреби	
1.2 Бізнес-вимоги до програмного продукту	
1.2.1 Опис проблеми споживача	
1.2.1.1 Концептуальний опис проблеми споживача	
1.2.1.2 Опис цільової групи споживача	
1.2.1.3 Метричний опис проблеми споживача	
1.2.2 Мета створення програмного продукту	
1.2.2.1 Проблемний аналіз існуючих програмних продуктів	
1.2.2.2 Мета створення програмного продукту	
1.2.3 Назва програмного продукту	
1.2.3.1 Гасло програмного продукту	
1.2.3.2 Логотип програмного продукту	
1.3 Вимоги користувача до програмного продукту	
1.3.1 Історія користувача програмного продукту	
1.3.2 Діаграма прецедентів програмного продукту	
1.3.3 Сценаріїв використання прецедентів програмного продукту	
1.4 Функціональні вимоги до програмного продукту	
1.4.1. Багаторівнева класифікація функціональних вимог	
1.4.2 Функціональний аналіз існуючих програмних продуктів	
1.5 Нефункціональні вимоги до програмного продукту	
1.5.1 Опис зовнішніх інтерфейсів	
1.5.1.1 Опис інтерфейсів користувача	
1.5.1.1.1 Опис INPUT-інтерфейсів користувача	
1.5.1.1.2 Опис OUTPUT-інтерфейсів користувача	
1.5.1.2 Опис інтерфейсу із зовнішніми пристроями	

- 1.5.1.3 Опис програмних інтерфейсів
- 1.5.1.4 Опис інтерфейсів передачі інформації
- 1.5.1.5 Опис атрибутів продуктивності
- 2 Планування процесу розробки програмного продукту
 - 2.1 Планування ітерацій розробки програмного продукту
 - 2.2 Концептуальний опис архітектури програмного продукту
 - 2.3 План розробки програмного продукту
 - 2.3.1 Оцінка трудомісткості розробки програмного продукту
 - 2.3.2 Визначення дерева робіт з розробки програмного продукту
 - 2.3.3 Графік робіт з розробки програмного продукту
 - 2.3.3.1 Таблиця з графіком робіт
 - 2.3.3.2 Діаграма Ганта
- 3 Проектування програмного продукту
 - 3.1 Концептуальне та логічне проектування структур даних програмного продукту
 - 3.1.1 Концептуальне проектування на основі *UML*-діаграми концептуальних класів
 - 3.1.2 Логічне проектування структур даних
 - 3.2 Проектування програмних класів
 - 3.3 Проектування алгоритмів роботи методів програмних класів
 - 3.4 Проектування тестових сценаріїв верифікації роботи програмних модулів
 - 3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог
 - 3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог
 - 3.4.3 Створення матриці відповідності вимог до програмного продукту
- 4 Конструювання програмного продукту
 - 4.1 Особливості конструювання структур даних

32

4.2 Особливості конструювання програмних модулів

4.2.1 Конструювання програмної структури з урахуванням спеціалізованого *Framework* для *FrontEnd*-компонент архітектури (за наявності)

4.2.2 Конструювання програмних класів (за наявності об'єктно-орієнтованого програмування)

4.2.3 Конструювання алгоритмів методів програмних класів або процедур/функцій

4.2.4 Особливості використання спеціалізованих програмних бібліотек та API (за наявності)

4.3 Модульне тестування програмних модулів

5 Верифікація програмного продукту

5.1 Тестування апаратно-програмних інтерфейсів програмного продукту

5.2 Тестування інтерфейсу користувача програмного продукту

5.3 Тестування часу реакції програмного продукту на дії користувача

6 Розгортання та валідація програмного продукту

6.1 Інструкція з встановлення системного програмного забезпечення

6.2 Інструкція з використання програмного продукту

6.3 Результати валідації програмного продукту

Висновки

1 Вимоги до програмного продукту

1.1 Визначення потреб споживача

1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено одну ієрархію потреби споживача, яку хотілося б задовольнити, використовуючи майбутній програмний продукт.

Потребою споживача є необхідність у самовираженні через подарунок, та у прояві уваги до іншої людини.

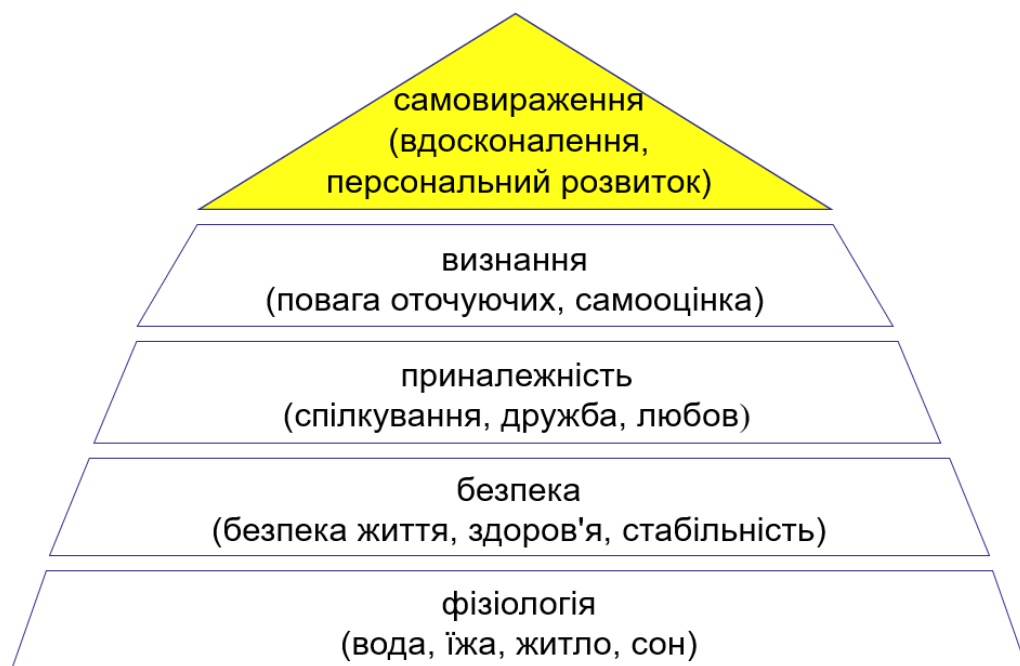


Рис. 1.1 – Ієрархія потреби споживача

1.1.2 Деталізація матеріальної потреби

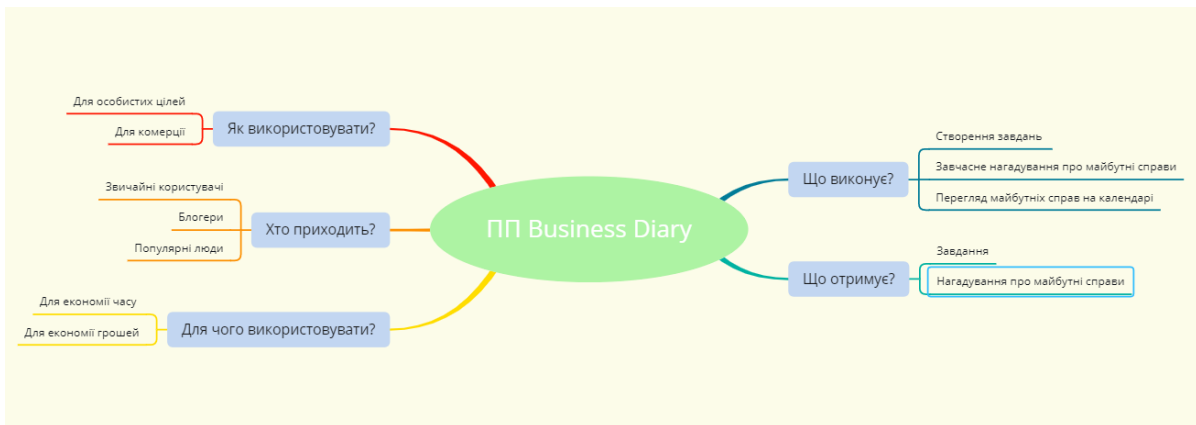


Рис. 1.2 – Mind map ПП «WishList»

1.2 Бізнес-вимоги до програмного продукту

1.2.1 Опис проблеми споживача

1.2.1.1 Концептуальний опис проблеми споживача

Діловий щоденник - відмінне рішення, яке дозволяє ефективніше вирішувати робочі завдання, управляти часом, організовувати різні процеси. Крім того, щоденник корисний і для дозвілля, пошуку натхнення, особистого вдосконалення та цікавого проведення часу.

Business Diary – це помічник, який дозволить оптимально спланувати всі справи на день, місяць, рік, записати всю необхідну інформацію, цілі, прагнення, завдання. Такий мобільний додаток чудово підвищує продуктивність, допомагає звільнити небагато часу на відпочинок, спілкування з близькими чи друзями.

Мобільний додаток з усіма планами на день допомагає визначити дуже важливі завдання на найближче майбутнє, а й зберігати важливу інформацію в одному місці. Зараз виробники в основному випускають блокноти невеликих форматів, які зручно покласти в сумку, рюкзак і завжди носити з собою.

1.2.1.2 Опис цільової групи споживачів

ЦА можна розбити на 3 групи:

- Молоді люди та студенти 18-27 років, які створюють основну масу користувачів сервісу;

- Середній вік 30-55 років, створюють середню масу користувачів;

- Літні люди 55-70 років, найменша маса користувачів сервісу.

Цільовою аудиторією будуть молоді люди в віці 18-27 років, тому, що в цьому віці люди ознайомлені із технікою та різними сервісами якісніше, частіше ніж люди похилого віку.

1.2.1.3 Метричний опис проблеми споживача

Складність вибору = $(P_a * P_p) * 100$,

де P_a – користувачі;

P_p – Завдання.

Анкетування проводилося на 11 студентах потоку AI, опитування показало, що більшість того, хто використовуватиме, це зазвичай хлопці.

1.2.2 Мета створення програмного продукту

1.2.2.1 Проблемний аналіз існуючих програмних продуктів

Критерії	Аналоги			
	Any.do	Todoist	Do!	Розроблюваний застосунок
Синхронізація із календарем	+	–	–	+
Нагадування	+	+	–	+
Представлення справ у вигляді списку	+	+	+	+
Представлення справ у вигляді календарю	+	–	–	+
Авторизація / реєстрація за допомогою соцмереж	+	+	–	+
Відсутність реклами	+	+	+	+
Повний функціонал безкоштовно	–	–	+	+
Автоматичне складання списку справ на майбутнє	–	–	–	+
Кольорові позначки	–	+	+	+

Таблиця 1.1 – Аналіз існуючих програмних продуктів і ПП, що розробляється

1.2.2.2 Мета створення програмного продукту

Метою створення ПП, є: розробка мобільного додатку для створення та відстежування завдань, контролювати свій час, завдяки чому, можливо економити час, щоб використати зекономлений час, на інші важливі справи.

1.2.3 Назва програмного продукту

«Business Diary»

1.2.3.1 Гасло програмного продукту

«Ваш час цінний, і Ми це цінуємо»

1.2.3.2 Логотип програмного продукту



Рис. 1.3 – Логотип ПП «WishList»

1.3 Вимоги користувача до програмного продукту

1.3.1 Пригодницька історія користувача програмного продукту (за бажанням членів проектної команди)

1.3.2 Історія користувача програмного продукту

Як гість я хочу:

- Мати змогу завести новий профіль
- Мати змогу редагувати свій профіль

Як користувач я хочу:

- Як користувач, Я хочу швидко створювати завдання, щоб економити час.
- Як користувач, Я хочу бачити на календарі майбутні справи.
- Як користувач, Я хочу мати можливість переглянути майбутні справи вигляді списку.
- Як користувач, Я хочу мати змогу редагувати свій профіль.
- Як користувач, Я хочу мати повний функціонал безкоштовно.

1.3.3 Діаграма прецедентів програмного продукту

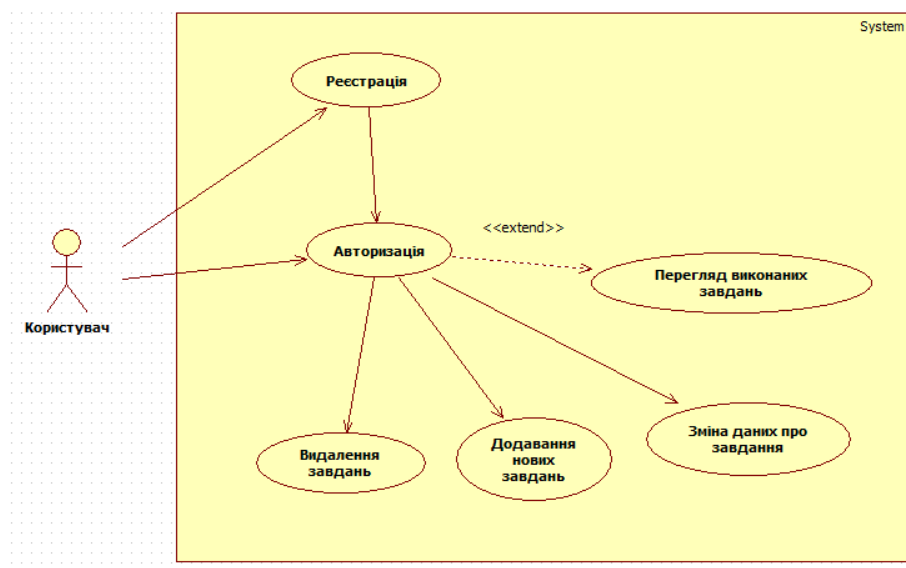


Рис. 1.4 – Діаграма прецедентів програмного продукту

1.3.4 Сценарії використання прецедентів програмного продукту

Таблиця 1.2 – 1. Прецедент ПП «Реєстрація»:

Пункт	Опис
Прецедент	Реєстрація
Передумова початку виконання прецеденту	«Відкриття мобільного додатку»
Актори як зацікавлені особи у виконанні прецеденту	Користувач
Актор-основна зацікавлена особа як ініціатор початку прецеденту	Користувач
Гарантія успіху	Реєстрація в мобільному додатку

Приклад основного успішного сценарію прецеденту «Реєстрація»:

1. При відкритті мобільного додатку, користувач побаче діалогове вікно «Реєстрації»;
2. Користувачу необхідно ввести реєстраційні дані (логін, емейл, пароль);
3. Після перевірки і підтвердження коректності даних, користувач отримує персональну сторінку.

Приклад альтернативного сценарію для успішного прикладу основного сценарію прецеденту «Реєстрація»:

1. При відкритті мобільного додатку, користувач побаче діалогове вікно «Реєстрації»;
2. Користувачу необхідно ввести реєстраційні дані (логін, емейл, пароль);
3. Після перевірки даних, виводиться повідомлення, що за цими даними не можливо зареєструватися;
4. Після виведення повідомлення про помилку, мобільний додаток повертає користувача до кроку 1.

Таблиця 1.3 – 2. Прецедент ПП «Авторизація»:

Пункт	Опис
Прецедент	Авторизація
Передумова початку виконання прецеденту	«Відкриття мобільного додатку» або «реєстрація»
Актори як зацікавлені особи у виконанні прецеденту	Користувач
Актор-основна зацікавлена особа як ініціатор початку прецеденту	Користувач
Гарантія успіху	Отримати доступ до свого профіля та всіх можливостей

Приклад основного успішного сценарію прецеденту «Авторизувати користувача»:

1. Мобільного додатку запитує у гостя параметри авторизації (емейл, пароль);
2. Гість вводить свої власні параметри (емейл, пароль) ;
3. Додаток надає авторизованому користувачу доступ до прецедентів.

Приклад альтернативного сценарію для успішного прикладу основного сценарію прецеденту «Авторизувати користувача»:

1. Додаток запитує у гостя його параметри авторизації (емейл, пароль);
2. Гість вводить свої власні параметри (емейл, пароль);
3. Додаток виявляє, що параметри (емейл, пароль) користувача не є вірними
4. Додаток видає повідомлення про помилку і повертається на виконання 1 кроку.

Таблиця 1.4 – 3. Прецедент ПП «Додавання завдання»:

Пункт	Опис
Прецедент	Додавання завдання
Передумова початку виконання прецеденту	«Успішна авторизація»
Актори як зацікавлені особи у виконанні прецеденту	Користувач
Актор-основна зацікавлена особа як ініціатор початку прецеденту	Користувач
Гарантія успіху	Додавання завдання

Приклад основного успішного сценарію прецедента «Додавання завдання»:

1. Користувач заходить в свій аккаунт та тисне на кнопку додавання завдання;
2. Через форму, користувач вводить дані для додавання завдання;
3. Додаток перевіряє на коректність даних та зберігає завдання.

Приклад альтернативного сценарію для успішного прикладу основного сценарію прецеденту «Додавання завдання»:

1. Користувач заходить в свій аккаунт та на форму додавання завдання;
2. Через форму, додає завдання на аккаунт;
3. Додаток перевіряє на коректність даних;
4. Додаток повідомляє про невдалу спробу додавання та повертає до пункту 2.

1.4 Функціональні вимоги до програмного продукту

1.4.1. Багаторівнева класифікація функціональних вимог

Таблиця 1.6 – Багаторівнева класифікація функціональних вимог

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети	Пріоритет функції
F1-Реєстрація користувача	-	30%	M
F1.1-Ввод параметрів реєстрації користувача	-		
F1.2-зберігання параметрів реєстрації	-		
F1.3-перевірка параметрів реєстрації	F1.2		
F2-Авторизація	-	30%	M
F2.1-перевірка параметрів авторизації	F1.2		
F3-Додавання завдання	F2	40%	M

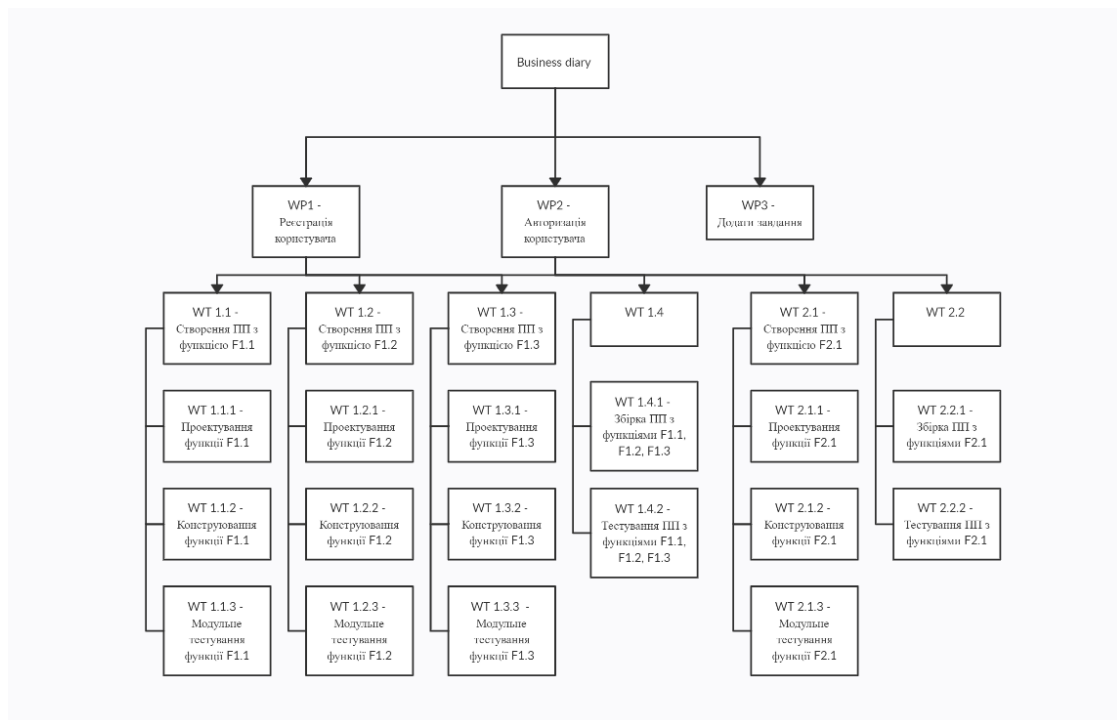


Рис. 1.5 – Ієрархічна WBS-структура багаторівневої класифікації функціональних вимог

1.4.2 Функціональний аналіз існуючих програмних продуктів

Таблиця 1.7 – Функціональний аналіз існуючих програмних продуктів

Ідентифікатор функції (назва)	ПП1	ПП2	ПП3	ПП розроблюваний
FR1 (Реєстрація користувача)	+	-	+	+
FR2 (Авторизація)	-	+	+	+
FR3 (Додати завдання)	+	-	-	+
Безкоштовне розповсюдження	+	-	-	+

1.5 Нефункціональні вимоги до програмного продукту

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейсів користувача

1.5.1.1.1 Опис INPUT-інтерфейсів користувача

Таблиця 1.8 – Опис INPUT-інтерфейсів користувача

Ідентифікатор функції (назва)	Засіб INPUT-потoku	Особливості використання
FR1 (Реєстрація користувача)	Сенсорний екран	Введення інформації через віртуальну клавіатуру
FR2 (Авторизація)	Сенсорний екран	Введення інформації через віртуальну клавіатуру
FR3 (Додати завдання)	Сенсорний екран	Введення інформації через віртуальну клавіатуру

1.5.1.1.2 Опис OUTPUT-інтерфейсів користувача

Ідентифікатор функції (назва)	Засіб OUTPUT - потоку	Особливості використання
FR1 (Реєстрація користувача)	Графічний інтерфейс	<p>Регистрация</p> <p>Имя пользователя</p> <p>Почта</p> <p>Пароль</p> <p>Зарегистрироваться</p> <p>Войти</p>
FR2 (Авторизація)	Графічний інтерфейс	<p>Логин</p> <p>Почта</p> <p>Пароль</p> <p>Войти</p> <p>Забыли пароль?</p> <p>Зарегистрироваться</p>

Рис. 1.6 – Опис OUTPUT-інтерфейсів користувача

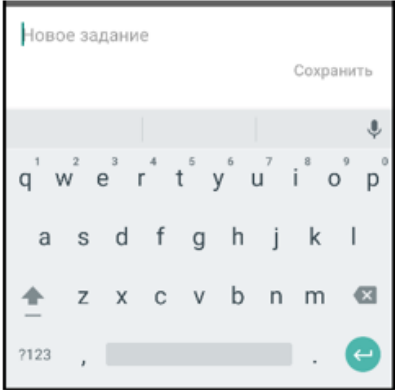
FR3 (Додати завдання)	Графічний інтерфейс	
-----------------------	---------------------	--

Рис. 1.7 – Опис OUTPUT-інтерфейсів користувача

1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

Мобільний додаток може працювати без підключення к інтернету(мобільного інтернету чи Wi-Fi).

1.5.1.3 Опис програмних інтерфейсів

Мобільний додаток може працювати на нових версіях операційної системи Android.

1.5.1.4 Опис інтерфейсів передачі інформації

1.5.1.5 Опис атрибутів продуктивності

При використанні звичайних функцій, затримка не має бути більше ніж пару секунд.

Єдині функції ПП, що можуть викликати більшу затримку:

Таблиця 1.9 – Максимальний час реакції ПП на дії користувача

Ідентифікатор функції (назва)	Максимальний час реакції ПП на дії користувачів, секунди
FR1 (Реєстрація користувача)	10
FR2 (Авторизація користувача)	8
FR3 (Додати завдання)	6

2 Планування процесу розробки програмного продукту

2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, визначено функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП. Результати представлено в таблиці 2.1

Таблиця 2.1 – приклад опису функцій з наданням унікальних ієрархічних ідентифікаторів

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети	Пріоритет функції
F1-Реєстрація користувача	-	30%	M
F1.1-Ввод параметрів реєстрації користувача	-		
F1.2-зберігання параметрів реєстрації	-		
F1.3-перевірка параметрів реєстрації	F1.2		
F2-Авторизація	-	30%	M
F2.1-перевірка параметрів авторизації	F1.2		
F3-Додавання завдання	F2	40%	M

2.2 Концептуальний опис архітектури програмного продукту

Android – це набір відкритого програмного забезпечення для мобільних пристроїв від компанії Google, до складу якого входить операційна система та комплект базових міжплатформних програм.

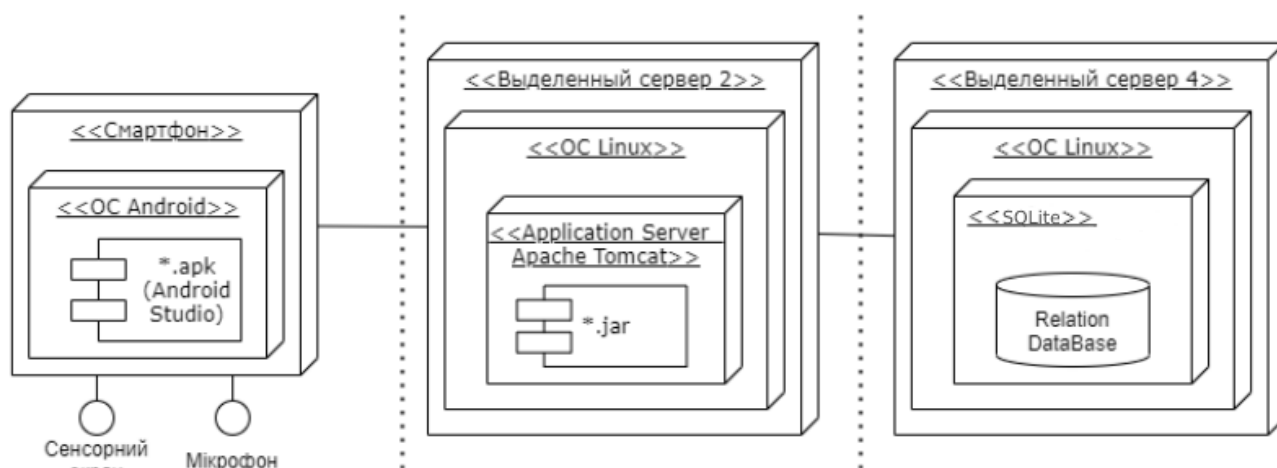


Рис. 2.1 – Концептуальний опис архітектури програмного продукту

2.3 План розробки програмного продукту

2.3.1 Оцінка трудомісткості розробки програмного продукту

Таблиця 2.2 – Оцінка трудомісткості розробки ПП

Прецедент	Кількість кроків сценарію	Ваговий коефіцієнт
Користувач	9	3
Реєстрація	4	10
Авторизація	2	5
Видалення завдання	3	5
Додавання завдання	3	5
Редагування даних завдання	3	4

$$UC = 8 \cdot 3 + 3 \cdot 8 + 2 \cdot 3 = 56$$

2.3.2 Визначення дерева робіт з розробки програмного продукту

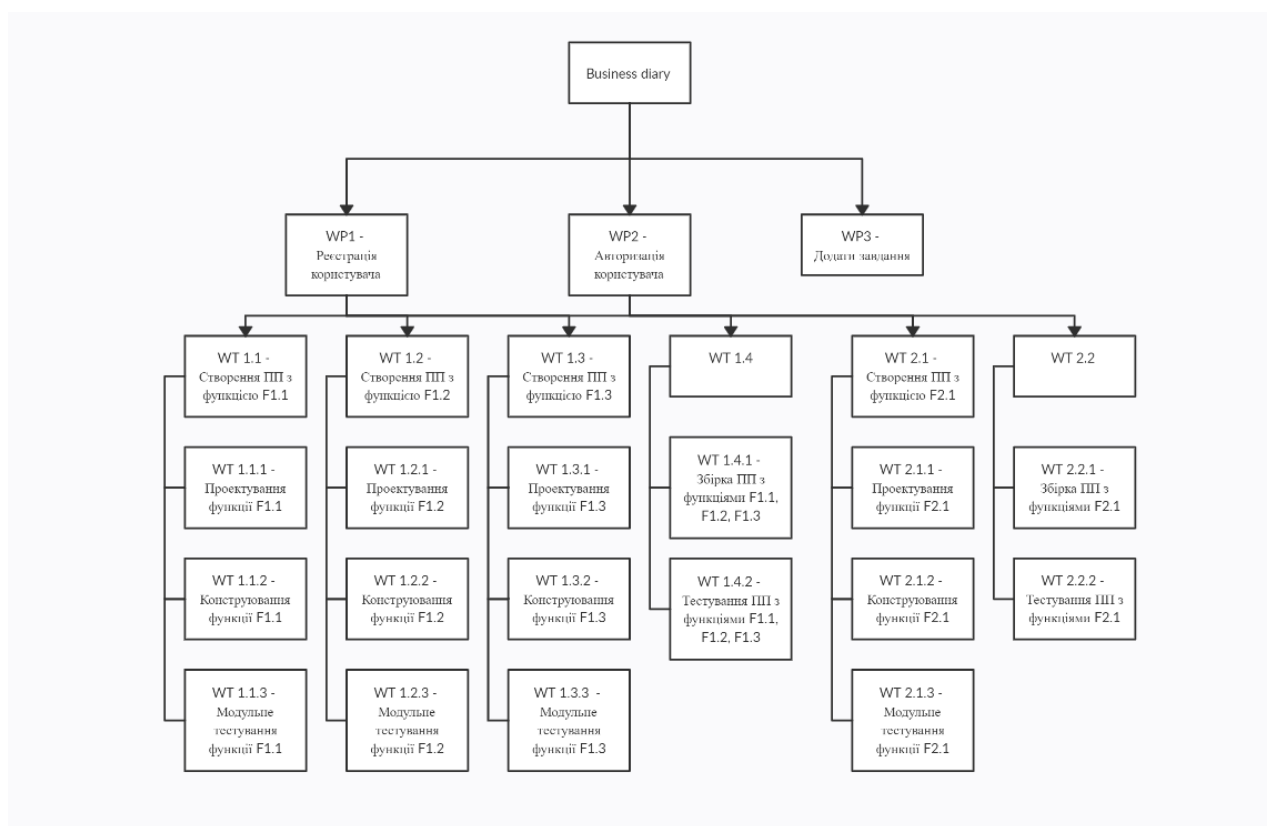


Рис. 2.2 – Дерево робіт з розробки ПП

2.3.3 Графік робіт з розробки програмного продукту

2.3.3.1 Таблиця з графіком робіт

Таблиця 2.3 – таблиця з графіком робіт

WST	Дата початку	Дні	Дата завершення	Виконавець
1.1.1	01.10.2021	2	02.10.2021	Кулик В.О.
1.2.2	02.10.2021	2	04.10.2021	Кулик В.О.
1.3.3	04.10.2021	3	07.10.2021	Кулик В.О.
1.2.1	07.10.2021	1	08.10.2021	Кулик В.О.
1.2.2	08.10.2021	3	11.10.2021	Кулик В.О.
1.2.3	11.10.2021	3	14.10.2021	Кулик В.О.
1.3.1	14.10.2021	4	18.10.2021	Кулик В.О.
1.3.2	18.10.2021	2	20.10.2021	Кулик В.О.
1.3.3	20.10.2021	3	23.10.2021	Кулик В.О.
1.4.1	23.10.2021	2	25.10.2021	Кулик В.О.
1.4.2	25.10.2021	3	28.10.2021	Кулик В.О.
2.1.1	28.10.2021	4	01.11.2021	Кулик В.О.
2.1.2	01.11.2021	5	06.11.2021	Кулик В.О.
2.1.3	06.11.2021	5	11.11.2021	Кулик В.О.
2.2.1	11.11.2021	5	16.11.2021	Кулик В.О.
2.2.2	16.11.2021	5	21.11.2021	Кулик В.О.
5.1.1	21.11.2021	5	26.11.2021	Кулик В.О.
5.1.2	26.11.2021	5	01.12.2021	Кулик В.О.
5.1.3	01.12.2021	4	05.12.2021	Кулик В.О.
5.2.1	05.12.2021	1	06.12.2021	Кулик В.О.
5.2.2	06.12.2021	3	09.12.2021	Кулик В.О.

2.3.3.2 Діаграма Ганта

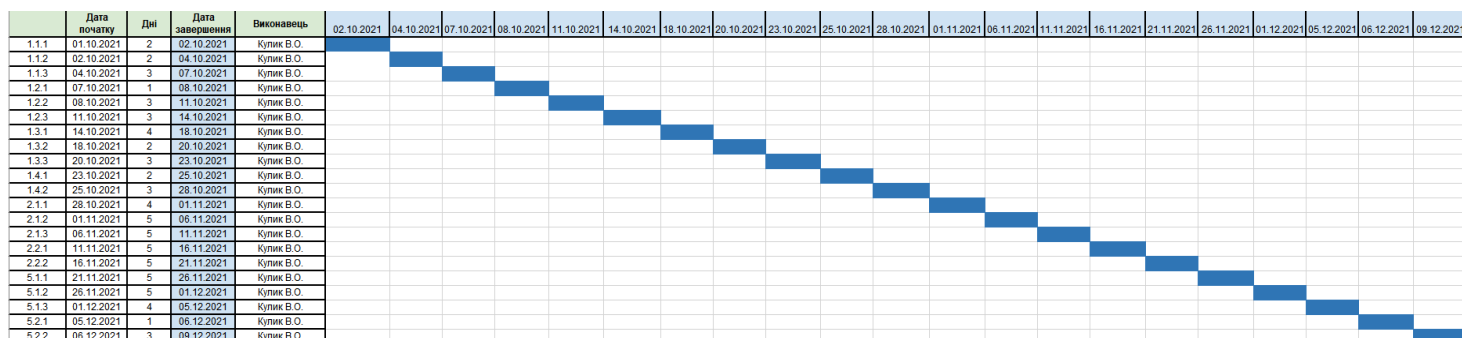


Рис 2.3 – Діаграма Ганта

3 Проектування програмного продукту

3.1 Концептуальне та логічне проектування структур даних програмного продукту

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

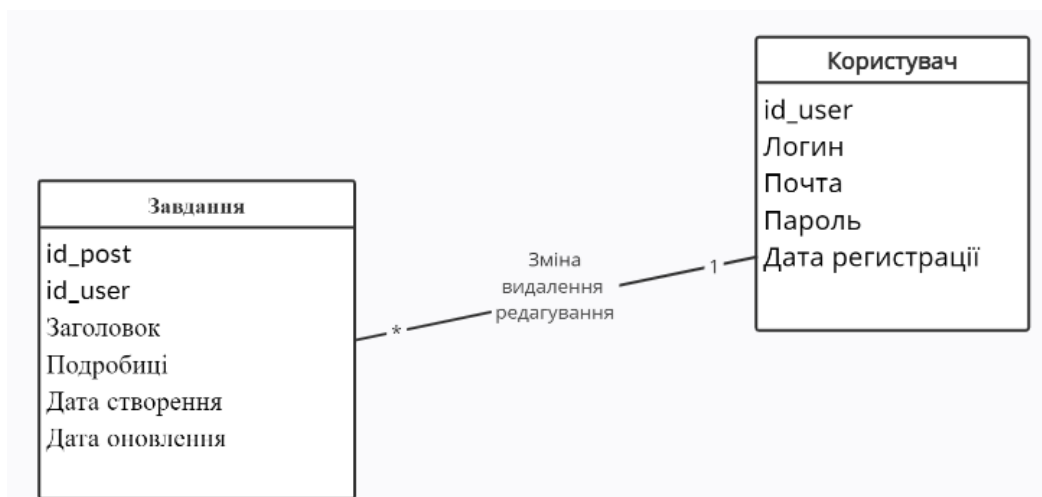


Рис. 3.1.1.1 – Концептуальне проектування

3.1.2 Логічне проектування структур даних

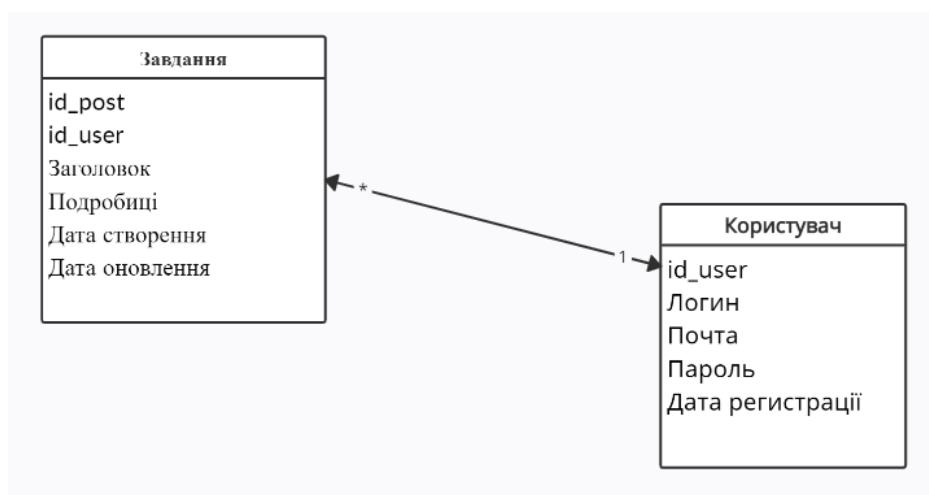


Рис. 3.1.2.1 – Логічне проектування структур даних

3.2 Проектування програмних класів

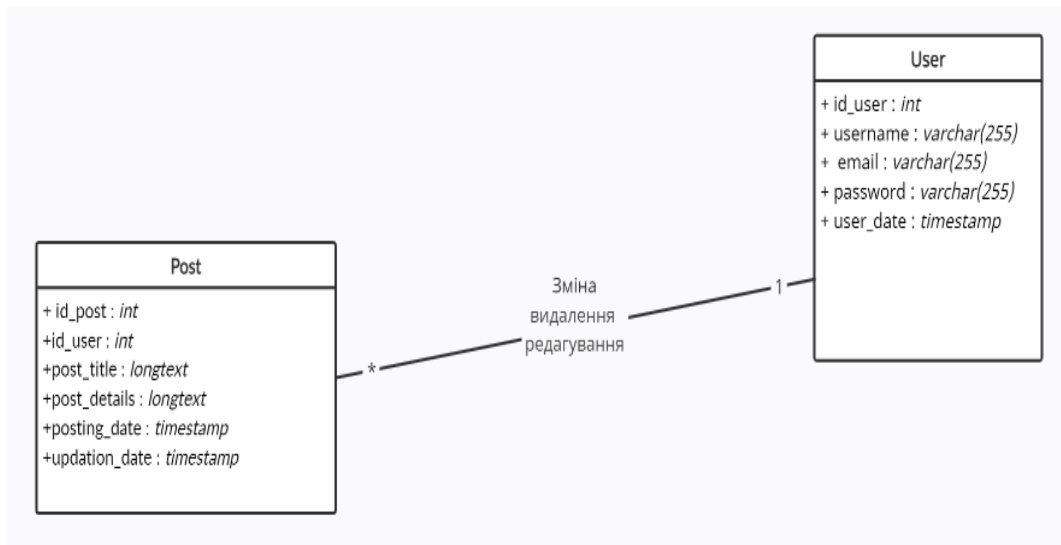


Рис. 3.2 – Проектування програмних класів

3.3 Проектування алгоритмів роботи методів програмних класів

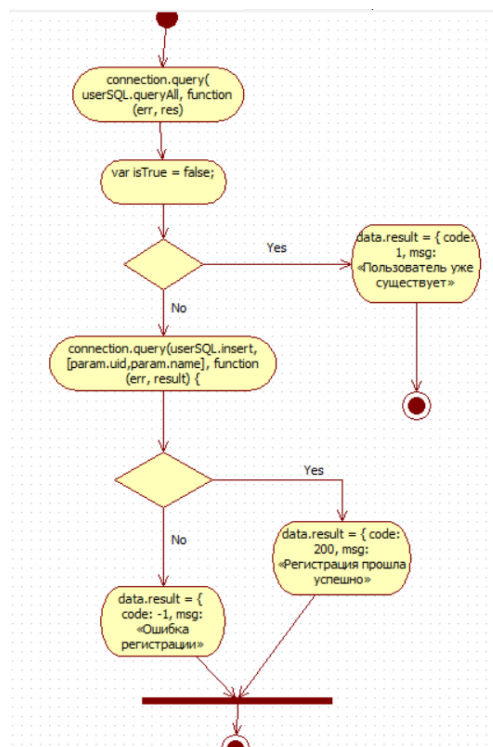


Рис. 3.3.1 – UML-діаграма активності «registration_user(), save_user(), authorization_user()»

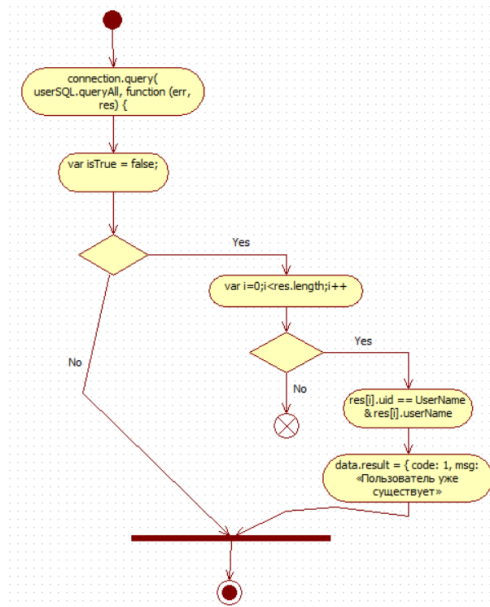


Рис. 3.3.2 – UML-діаграма активності «perevir_reestr_user()»

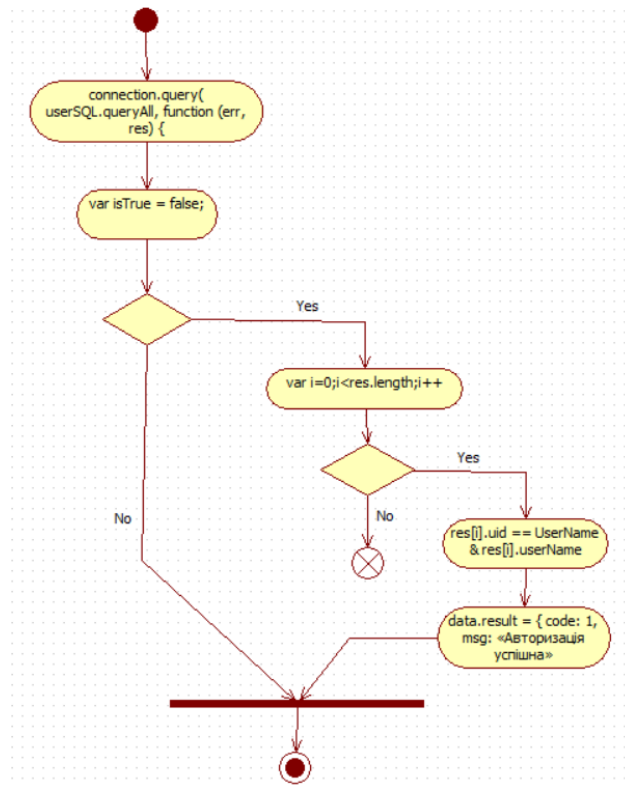


Рис. 3.3.3 – UML-діаграма активності «check_registration_user()»

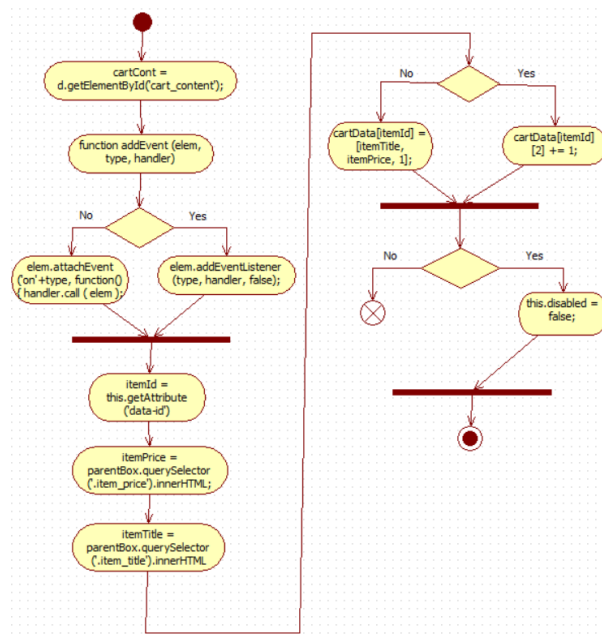


Рис. 3.3.4 – UML-діаграма активності «Add_post()»

3.4 Проектування тестових сценаріїв верифікації роботи програмних модулів

3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог

Таблиця 3.4.1.1 – Верифікація функціональних вимог

FR ID	Test Case ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR1	TC1	Input = (empty)	Повідомлення: «Назва не може бути пустим»
FR1	TC2	Input = (текст, що перевищує 100 символів)	Повідомлення: «Не можна перевищувати допустиму величину тексту в 100 знаків»
FR1	TC3	Input = (Нова інформація)	Повідомлення «Дані оновлено»

FR ID	Test Case ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR1	TC4	Input = (Некоректне введення)	Повідомлення: «Помилка з введенням, будь ласка, перезавантажте сторінку та повторіть спробу»
FR1	TC5	Input = (Введення коректного тексту)	Повідомлення: «Успішне додавання»

3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог

Таблиця 5.2.2 – Опис тестових сценаріїв для інтерфейсу користувача

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача
NFR1	TC6	<ol style="list-style-type: none"> В поле «Назва» та «Дата» і «Час» ввести будь-яке значення Натиснути кнопку «Зберегти» 	
NFR2	TC7	<ol style="list-style-type: none"> Натиснути на «Нова задача», для того, щоб відкрилось вікно «Дій» Натиснути кнопку «Завершити» «Нова задача» стане завершеним 	

Продовження таблиці 5.2.2

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача
NFR3	TC8	<ol style="list-style-type: none"> 1) Натиснути на «Нова задача», для того, щоб відкрилось вікно «Дій» 2) Натиснути кнопку «Редагувати» 3) Відредагувати дані 4) Натиснути кнопку «Зберегти» 	
NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача
NFR4	TC9	<ol style="list-style-type: none"> 1) Натиснути на «Нова задача», для того, щоб відкрилось вікно «Дій» 2) Натиснути кнопку «Видалити» 3) «Нова задача» буде видаленим 	

3.4.3 Створення матриці відповідності вимог до програмного продукту

FR ID	NFR ID	TC ID
FR1	NFR 1	TC 1
FR1	NFR 1	TC 2
FR1	NFR 1	TC 3
FR1	NFR 1	TC 4
FR1	NFR 1	TC 5
FR1	NFR 1	TC 6
FR1	NFR 1	TC 7
FR1	NFR 1	TC 8
FR5	NFR 1	TC 9
FR2	NFR 2	TC 10
FR3	NFR 3	TC 11

Рис. 3.4.3.1 – Матриця відповідності вимог

4 Конструювання програмного продукту

4.1 Особливості конструювання структур даних

Реляційна база даних

```
public class Note {
    public static final String TABLE_NAME = "notes";

    public static final String COLUMN_ID = "id";
    public static final String COLUMN_STATUS = "status";
    public static final String COLUMN_NOTE = "note";
    public static final String COLUMN_DATE = "date";
    public static final String COLUMN_TIMESTAMP = "timestamp";

    private int id;
    private int status;
    private String note;
    private String date;
    private String timestamp;

    public static final String CREATE_TABLE =
        "CREATE TABLE " + TABLE_NAME + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_STATUS + " INTEGER,"
        + COLUMN_NOTE + " TEXT,"
        + COLUMN_DATE + " DATE,"
        + COLUMN_TIMESTAMP + " DATETIME DEFAULT
CURRENT_TIMESTAMP"
        + ")";

    public Note(int id, int status, String note, String date, String timestamp) {
        this.id = id;
        this.status = status;
        this.note = note;
        this.date = date;
        this.timestamp = timestamp;
    }

    public int getId() {
        return id;
    }

    public int getStatus() {
        return status;
    }
}
```



```
public String getNote() {  
    return note;  
}  
  
public String getDate() {  
    return date;  
}  
public void setNote(String note) {  
    this.note = note;  
}  
  
public void setDate(String date) {  
    this.date = date;  
}  
  
public String getTimestamp() {  
    return timestamp;  
}  
  
public void setId(int id) {  
    this.id = id;  
}  
  
public void setStatus(int status) {  
    this.status = status;  
}  
  
public void setTimestamp(String timestamp) {  
    this.timestamp = timestamp;  
}  
}
```

```

var input = document.querySelector( selectors: "input[type = 'text']");
var ul = document.querySelector( selectors: "ul");
var container = document.querySelector( selectors: "div");
var lists = document.querySelectorAll( selectors: "li");
var spans = document.getElementsByTagName( qualifiedName: "span");
var pencil = document.querySelector( selectors: "#pencil");
var saveBtn = document.querySelector( selectors: ".save");
var clearBtn = document.querySelector( selectors: ".clear");
var tipsBtn = document.querySelector( selectors: ".tipBtn");
var closeBtn = document.querySelector( selectors: ".closeBtn");
var overlay = document.getElementById( elementId: "overlay")

```

Рис. 4.1.1 – Виконання CRUD-операцій доступу до даних

4.2 Особливості конструювання програмних модулів

```

function deleteTodo(){
  for(let span of spans){
    span.addEventListener ("click",function (){
      span.parentElement.remove();
      event.stopPropagation();
    });
  }
}

//function to load todo if list is found in local storage.
function loadTodo(){
  if(localStorage.getItem( key: 'todoList')){
    ul.innerHTML = localStorage.getItem( key: 'todoList');
    deleteTodo();
  }
}

//event listener for input to add new todo to the list.
input.addEventListener( type: "keypress", listener: function(keyPressed :Event ){
  if(keyPressed.which === 13){
    //creating lists and span when enter is clicked
    var li = document.createElement( tagName: "li");
    var spanElement = document.createElement( tagName: "span");
    var icon = document.createElement( tagName: "i");

    var newTodo = this.value;
    this.value = " ";

    icon.classList.add('fas', 'fa-trash-alt');
    spanElement.appendChild(icon);
    ul.appendChild(li).append(spanElement,newTodo);

    deleteTodo();
  }
});

```

Рис. 4.2.1 – Конструювання програмних модулів NodeDao

4.2.1 Конструювання програмної структури з урахуванням спеціалізованого *Framework* для *FrontEnd*-компонент архітектури (за наявності)

Немає

4.2.2 Конструювання програмних класів (за наявності об'єктно-орієнтованого програмування)

```
public class LoginActivity extends AppCompatActivity {
    EditText editTextEmail, editTextPassword;
    Button buttonLogin;
    TextView textViewRegister;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        editTextEmail = findViewById(R.id.editTextEmail);
        editTextPassword = findViewById(R.id.editTextPassword);
        buttonLogin = findViewById(R.id.buttonLogin);

        textViewRegister = findViewById(R.id.textViewRegister);

        textViewRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext, LoginActivity.this, RegisterActivity.class));
            }
        });

        buttonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
```

Рис. 4.2.3.1 – Конструювання програмних класів

```
public class CalendarActivity extends AppCompatActivity {
    List<Task> tasks = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calendar);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        CalendarView calendarView = findViewById(R.id.calendarView);
        calendarView.setHeaderColor(R.color.colorAccent);
        calendarView.setHeaderVisibility(View.INVISIBLE);
        getSavedTasks();

        Button list = findViewById(R.id.buttonList);
        list.setOnClickListener(view -> {
            Intent calendar = new Intent( packageContext, this, MainActivity.class);
            startActivity(calendar);
        });
    }

    private void getSavedTasks() {
        class GetSavedTasks extends AsyncTask<Void, Void, List<Task>> {
            CalendarView calendarView = findViewById(R.id.calendarView);
```

Рис. 4.2.3.2 – Конструювання програмних класів

```

public class MainActivity extends BaseActivity implements CreateTaskBottomSheetFragment.setRefreshListener {

    @BindView(R.id.taskRecycler)
    RecyclerView taskRecycler;
    @BindView(R.id.addTask)
    FloatingActionButton addTask;
    TaskAdapter taskAdapter;
    List<Task> tasks = new ArrayList<>();
    @BindView(R.id.empty_notes_view)
    TextView emptyList;
    @BindView(R.id.buttonCalendar)
    Button calendar;
    private TextView mHomeTextGreeting;
    private TextView mHomeTextGreetingNameUser;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        mHomeTextGreeting = findViewById(R.id.homeTextGreeting);
        mHomeTextGreetingNameUser = findViewById(R.id.homeTextGreetingNameUser);

        ButterKnife.bind(this);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        ComponentName receiver = new ComponentName(this, AlarmBroadcastReceiver.class);
        PackageManager pm = getPackageManager();
        pm.setComponentEnabledSetting(receiver, PackageManager.COMPONENT_ENABLED_STATE_ENABLED, PackageManager.DONT_KILL_APP);
    }
}

```

Рис. 4.2.3.3 – Конструювання програмних класів

```

public class TaskAdapter extends RecyclerView.Adapter<TaskAdapter.TaskViewHolder> {

    private MainActivity context;
    private LayoutInflater inflater;
    private List<Task> taskList;
    public SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "EE dd MMM yyyy", new Locale( language: "ru"));
    public SimpleDateFormat inputDateFormat = new SimpleDateFormat( pattern: "dd-MM-yyyy", new Locale( language: "ru"));
    Date date = null;
    String outputDateString = null;
    CreateTaskBottomSheetFragment.setRefreshListener setRefreshListener;

    public TaskAdapter(MainActivity context, List<Task> taskList, CreateTaskBottomSheetFragment.setRefreshListener setRefreshListener) {
        this.context = context;
        this.taskList = taskList;
        this.setRefreshListener = setRefreshListener;
        this.inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @NonNull
    @Override
    public TaskViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int viewType) {
        View view = inflater.inflate(R.layout.item_task, viewGroup, attachToRoot false);
        return new TaskViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull TaskViewHolder holder, int position) {
        Task task = taskList.get(position);
        holder.title.setText(task.getTaskTitle());
    }
}

```

Рис. 4.2.3.4 – Конструювання програмних класів

4.2.3 Конструювання алгоритмів методів програмних класів або процедур/функцій

Немає

4.2.4 Особливості використання спеціалізованих програмних бібліотек та API (за наявності)

Немає

4.3 Модульне тестування програмних модулів

Таблиця 4.3.1 – Модульне тестування програмних модулів

FR ID	Test Case ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR1	TC1	Input = (empty)	Повідомлення: «Ім'я та опис не може бути пустим»
FR1	TC2	Input = (текст, що перевищує 100 символів)	Повідомлення: «Не можна перевищувати допустиму величину тексту в 100 знаків»
FR1	TC3	Input = (Нова інформація)	Повідомлення «Дані оновлено»

FR ID	Test Case ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR1	TC4	Input = (Некоректне введення)	Повідомлення: «Помилка з введенням, будь ласка, перезавантажте сторінку та повторіть спробу»
FR1	TC5	Input = (Введення коректного тексту)	Повідомлення: «Успішне додавання»

FR1	TC4	Input = (empty)	Повідомлення: «Назва не може бути пустим»
-----	-----	-----------------	---

```
public boolean validateFields() {  
    if (addTaskTitle.getText().toString().equalsIgnoreCase(" ")) {  
        Toast.makeText(activity, getString(R.string.hint_enter_text), Toast.LENGTH_SHORT).show();  
        return false;  
    }  
}
```

Test Results	594 ms
SearchServiceImpTest	594 ms
getResourceByTitleTest4()	594 ms

Рис. 4.3.1 – Test case №1

FR1	TC5	Input = (текст, що перевищує 100 символів)	Повідомлення: «Не можна перевищувати допустиму величину тексту в 100 знаків»
-----	-----	--	--

```

} else if (taskDate.getText().toString().equalsIgnoreCase( anotherString: "")) {
    Toast.makeText(activity, getString(R.string.hint_text_max), Toast.LENGTH_SHORT).show();
    return false;
}

```

✓ Test Results	566 ms
✓ SearchServiceImplTest	566 ms
✓ getResourceByTitleTest5()	566 ms

Рис. 4.3.2 – Test case №2

FR1	TC6	Input = (Нова інформація)	Повідомлення: «Дані оновлено»
-----	-----	---------------------------	-------------------------------

```

@Override
protected void onPostExecute(Void aVoid) {
    super.onPostExecute(aVoid);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        createAnAlarm();
    }
    setRefreshListener.refresh();
    Toast.makeText(getActivity(), getString(R.string.hint_updated), Toast.LENGTH_SHORT).show();
    dismiss();
}

```

✓ Test Results	903 ms
✓ SearchServiceImplTest	903 ms
✓ getResourceByTitleTest5()	903 ms

Рис. 4.3.3 – Test case №3

FR1	TC7	Input = (Некоректне введення)	Повідомлення: «Помилка з введення, будь ласка, перезавантажте сторінку та повторіть спробу»
-----	-----	-------------------------------	---

```

} else if (taskTime.getText().toString().equalsIgnoreCase( anotherString: "")) {
    Toast.makeText(activity, getString(R.string.hint_error), Toast.LENGTH_SHORT).show();
    return false;
}

```

✓ Test Results	630 ms
✓ SearchServiceImplTest	630 ms
✓ getNodesTest()	630 ms

Рис. 4.3.4 – Test case №4

FR1	TC8	Input = (Введення коректного тексту)	Повідомлення: «Успішне додавання»
-----	-----	--------------------------------------	-----------------------------------



Рис. 4.3.5 – Test case №5

5 Верифікація програмного продукту

5.1 Тестування апаратно-програмних інтерфейсів програмного продукту

Таблиця 5.1.1 – Опис умов перевірки апаратно-програмних інтерфейсів

ПП.

Тип умови (Hard/Soft)	Опис вимог	Опис реальних умов
Hard	CPU: frequency \geq 2 Mhz Memory: size \geq 2Gb	Qualcomm SDM660 Snapdragon 660 (14 nm) Memory: 32GB
Soft	OS: Android 9, 10, 11, 12	OS: Android 12

5.2 Тестування інтерфейсу користувача програмного продукту

Таблиця 5.2.2 – Опис тестових сценаріїв для інтерфейсу користувача

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача
NFR1	TC9	1) В поле «Назва» та «Дата» і «Час» ввести будь-яке значення 2) Натиснути кнопку «Зберегти»	

Продовження таблиці 5.2.2

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача
NFR2	TC10	<ol style="list-style-type: none"> 1) Натиснути на «Нова задача», для того, щоб відкрилось вікно «Дій» 2) Натиснути кнопку «Завершити» 3) «Нова задача» стане завершеним 	<div>Действия</div> <div>Завершить</div> <div>Редактировать</div> <div>Удалить</div>
NFR3	TC11	<ol style="list-style-type: none"> 1) Натиснути на «Нова задача», для того, щоб відкрилось вікно «Дій» 2) Натиснути кнопку «Редагувати» 3) Відредагувати дані 4) Натиснути кнопку «Зберегти» 	<div>Действия</div> <div>Завершить</div> <div>Редактировать</div> <div>Удалить</div> <div> <div>Новое задание</div> <div>Заполните данные ниже, чтобы добавить задачу.</div> <div> <div>Название</div> <div>New task</div> </div> <div> <div>Дата</div> <div>10-2-2022</div> <div>Время</div> <div>17:14</div> </div> <div>Сохранить</div> </div>
NFR4	TC12	<ol style="list-style-type: none"> 1) Натиснути на «Нова задача», для того, щоб відкрилось вікно «Дій» 2) Натиснути кнопку «Видалити» 3) «Нова задача» буде видаленим 	<div>Действия</div> <div>Завершить</div> <div>Редактировать</div> <div>Удалить</div>

5.3 Тестування часу реакції програмного продукту на дії користувача

Після проведення двох експериментів з часом виконання функції отримано два результати: 0,9 сек, 0.7 сек. Середнє значення = $(0.9+0.7)/2 = 0.8$

$$\delta = ((0.9-0.7) / 0.8) * 100\% = 25\%$$

Такий результат, вказує на необхідність продовження експериментів, поки δ не стане менше порогового значення $< 15\%$

Таблиця 5.3.3 – Результат тестування часу реакції ПП на дії користувача

FR (назва)	Максимальний час реакції ПП на дії користувачів, секунди	Кількість проведених тестів	Результат тестування, секунди	Відносна похибка вимірювань, %
FR1	2с	5	20с	7%
FR2	1,5с	4	30с	5%
FR3	1,2с	6	45с	8%
FR4	1с	8	24с	9%

6 Розгортання та валідація програмного продукту

6.1 Інструкція з встановлення системного програмного забезпечення

Отримати оригінал:

```
git clone https://github.com/VladislavKulik/todolist
```

Отримати оригінал в папку custom:

```
git clone https://github.com/VladislavKulik/todolist custom
```

Перевірити оновлення (при умові, що скачано в папку eternity):

```
git remote -v update
```

Отримати оновлення:

```
git pull
```

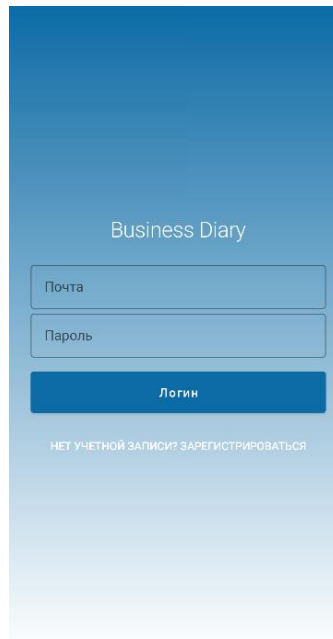
Розроблений ПП може працювати на будь-якому мобільному пристрою на якому встановлений OS Android.

Послідовність кроків для забезпечення роботи ПП:

- 1 Для клієнта потрібно, скачати інсталяційний файл з інтернету;
- 2 Для встановлення ПП потрібно відкрити інсталяційний файл та натиснути встановити;
- 3 Запустити натиснувши на ярлик на робочому столі.

6.2 Інструкція з використання програмного продукту

ПП надає можливість користувачу ролі «Гість» вести параметри реєстрації (ім'я користувача та його пароль), як показано на рисунку 2.



Business Diary

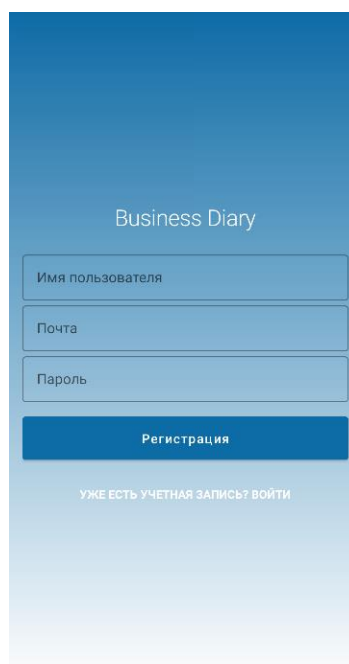
Почта

Пароль

Логин

НЕТ УЧЕТНОЙ ЗАПИСИ? ЗАРЕГИСТРИРОВАТЬСЯ

Рис. 6.2.1 – Екранна форма авторизації



Business Diary

Имя пользователя

Почта

Пароль

Регистрация

УЖЕ ЕСТЬ УЧЕТНАЯ ЗАПИСЬ? ВОЙТИ

Рис. 6.2.2 – Екранна форма реєстрації

ПП надає можливість лише користувачу ролі «Користувач» створювати завдання, як показано на рисунку 3.

Новое задание

Заполните данные ниже, чтобы добавить задачу.

Название

Дата

Время

Сохранить

Рис. 6.2.3 – Екранна форма введення завдання

При цьому необхідно пам'ятати про обмеження тексту: не менше 1 та не більше 100 букв або символів.

Для того щоб внести завдання, потрібно натиснути кнопку «Зберегти», як показано на рисунку 4.

Новое задание

Заполните данные ниже, чтобы добавить задачу.

Название
New task

Дата
20-4-2022

Время
05:08

Сохранить

Рис. 6.2.4 – Заповнена екранна форма введення завдання

ПП надає можливість користувачу ролі «Користувач» переглянути майбутні задачі в календарі, як показано на рисунку 5.

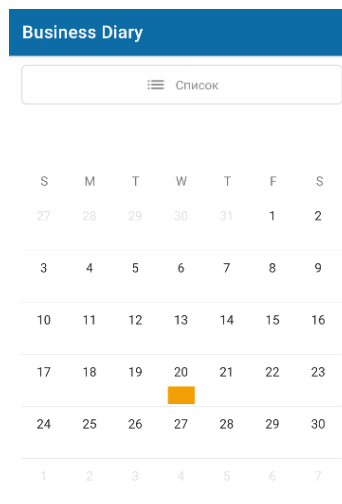


Рис. 6.2.5 – Екранна форма календарь

ПП надає можливість користувачу ролі «Користувач» ознайомитися із інтерфейсом, натиснувши кнопку «Допомога», як показано на рисунку 6 та 7.

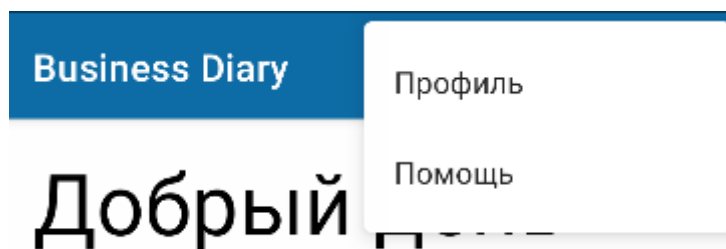


Рис. 6.2.6 – Екранна форма кнопки «Допомога»

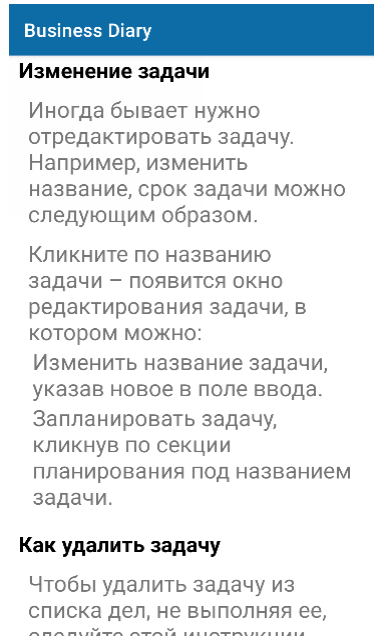


Рис. 6.2.7 – Екранна форма виведення значення кнопки «Допомога»

ПП надає можливість користувачу ролі «Користувач» редагувати завдання, як показано на рисунку 8 та 9.

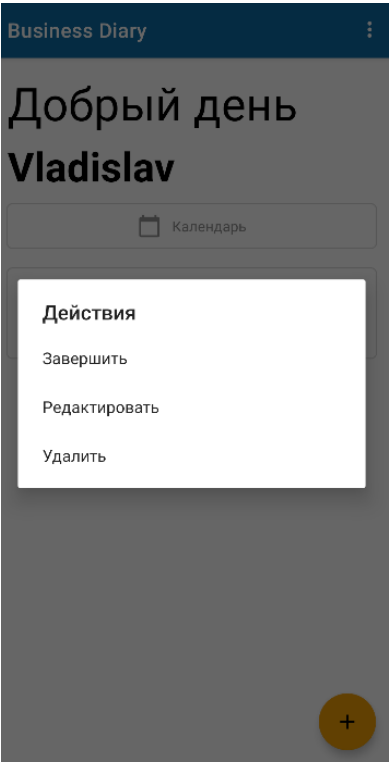


Рис. 6.2.8 – Екранна форма дій з завданням

Новое задание

Заполните данные ниже, чтобы добавить задачу.

Название

New task

Дата

20-4-2022

Время

05:08

Сохранить

Рис. 6.2.9 – Екранна форма редагування завдання

ПП надає можливість користувачу ролі «Користувач» видаляти завдання по одному, натиснувши кнопку «Видалити», як показано на рисунку 10 та 11.

Действия

Завершить

Редактировать

Удалить

Рис. 6.2.10 – Екранна форма дій з завданням

Добрый день
Vladislav

 Календарь

ср
20 New task
05:08
апр.

Удалено



Рис. 6.2.11 – Демонстрація успішного видалення завдання

Добрый день
Vladislav

 Календарь

ср
20 New task
05:08
апр.

Выполнено



Рис. 6.2.12 – Демонстрація успішного виконання завдання

6.3 Результати валідації програмного продукту

Метою створення ПП, є: розробка мобільного додатку для створення та відстежування завдань, контролювати свій час, завдяки чому, можливо економити час, щоб використати зекономлений час, на інші важливі справи.

1. Експеримент. Авторизація та створення Business Diary (повний цикл створення)

Раніше потрібно було зробити приблизно до ~80 не потрібних та зайвих натискань, враховуючи реєстрацію в мобільному додатку та інтуїтивно не зрозумілого інтерфейсу без підказок.

Існуючі ПП, кроки:

- 1) Пошук мобільного додатку приблизно ~10-15 натискань натискань по екрану мобільного телефона;
- 2) Реєстрація в мобільному додатку до ~30 натискань по екрану мобільного телефона;
- 3) Ознайомлення з інтуїтивно не зрозумілим та надто нагромадженим інтерфейсом ~25 натискань натискань по екрану мобільного телефона;
- 4) Створення та збереження завдання ~10+ натискань по екрану мобільного телефона;

Створений ПП, кроки:

- 1) Пошук мобільного додатку приблизно ~10-15 натискань по екрану мобільного телефона;
- 2) Ознайомлення з інтуїтивно зрозумілим інтерфейсом ~5 натискань по екрану мобільного телефона (або 1, якщо перейти до розділу «Допомога»);
- 3) Створення та збереження завдання ~5-10+ натискань по екрану мобільного телефона;

Як, можна побачити з експерименту, пункт 2 взагалі не потрібен, а всі інші

пункт можна оптимізувати.

Тому, було створено зручний мобільний додаток, для того, щоб швидко виконувати дії. Створений ПП дозволяє в 2-2,5 рази швидше створювати завдання та економити час, для інших корисних речей.

2. Експеримент. Створення Business Diary (Цикл створення Business Diary)

Існуючі ПП, кроки:

Веб-сервіс №1

- 1) Пошук сервісу приблизно ~10-15 натискань по екрану мобільного телефона;
- 2) Створення та збереження завдання приблизно ~20-30 натискань по екрану мобільного телефона;

Створенний ПП, кроки:

- 1) Пошук мобільного додатку приблизно ~10-15 натискань по екрану мобільного телефона;
- 2) Створення та збереження завдання приблизно ~5 натискань по екрану мобільного телефона;

Можна зробити висновок на основі 2 експерименту, що розроблений мобільний додаток Business Diary, економить час, має більший функціонал, потребує менше натискань та часу на вивчення інтерфейсу.

Висновки

В результаті виконання етапів курсової роботи було створено програмне забезпечення процесу керуванням та відстеженням майбутніх справ користувачів.

Приклад використання програмного забезпечення у відповідності з інструкцією користувача представлено у відеозапису, доступним за посиланням «https://drive.google.com/file/d/1Fw89Vx1mKezq_-gOZN_S66nVHyzJE2x2/view?usp=sharing».

Створене програмне забезпечення досягло наступної мети його споживача: розробка мобільного додатку для створення та відстежування завдань, контролювати свій час, завдяки чому, можливо економити час, щоб використати зекономлений час, на інші важливі справи

Доказом цього є наступні факти:

- 1) Зменшення кількості натискань по екрану мобільного телефона для додавання завдань;
- 2) Спрощення моніторингу майбутніх справ;
- 3) Спрощення інтерфейсу керування.

Вказані факти перетворили програмне забезпечення на програмний продукт.

В процесі створення програмного продукту виникли такі труднощі (організаційні, проблеми відсутності досвіду, знань, потрібних в різних етапах):

- 1) Брак досвіду та знань;
- 2) Брак часу.

Через вищеописані непередбачені труднощі, а також через обмежений час на створення програмного продукту, залишаються нереалізованими такі прецеденти або їх окремі кроки роботи:

- 1) Авторизація/реєстрація за допомогою соцмереж (1 крок);
- 2) Синхронізація з календарем (2 крок);

Зазначені недоробки планується реалізувати в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.