Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного додатку для покращення тайм менеджменту користувача»

Виконала: студентка ___IV___ курсу
групи ___2ПІ–18б(д/ф)_____
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Богач Катерина Сергіївна
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Бабюк Н.П.
(прізвище та ініціали)

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

Допущено до захисту

Зав. кафедри _____

«____» _____ 2022 р.

ВНТУ – 2022

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
25 березня 2022 року

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

Богач Катерині Сергіївні

1. Тема роботи: «Розробка програмного додатку для покращення тайм менеджменту користувача»
керівник роботи: к.т.н., заступник декана по організаційній роботі Бабюк Н. П. затверджені наказом вищого навчального закладу від "24" березня 2022 року № 66 .
2. Строк подання студентом роботи _____13.06.2022р._____
3. Вихідні дані до роботи : дані про актуальність засобу для підвищення тайм-менеджменту користувача, зручність у використанні у будь який момент часу та у будь яких сферах життя, як у роботі, навчанні, так і у повсякденному житті. Програмний додаток розроблений за допомогою мови програмування Java.
4.Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити): вступ; аналіз та постановка задачі; розробка архітектури та алгоритмів програмного продукту; розробка програмного продукту; тестування програми, висновки; перелік посилань, додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): Демонстрація роботи аналогів, діаграми варіантів використання, структурна схема компонентів додатку, інтерфейс головного вікна та основних модулів додатку, підбір кольорової гамми, загальний алгоритм роботи та тестування додатку.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–4	к.т.н., доцент кафедри ПЗ Бабюк Н. П.	**26.03.2022**	**13.06.2022**

7. Дата видачі завдання_____**26.03.2022 р.**_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обґрунтування актуальності розробки додатку та постановка задачі	26.03.22-14.04.22	Вик.
2	Розробка архітектури та алгоритмів роботи додатку, проектування інтерфейсу додатку	15.04.22-25.04.22	Вик.
3	Вибір середовища та мови розробки	26.04.22-12.05.22	Вик.
4	Розробка програмного продукту	13.05.22-30.05.22	Вик.
5	Тестування роботи додатку	31.05.22-03.06.22	Вик.
6	Оформлення матеріалів до захисту БДР	03.06.22-13.06.22	Вик.

Студентка _____ Богач К.С.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Бабюк Н.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

В бакалаврській дипломній роботі розроблено мобільний додаток для нотаток. Зазначена програма створена для підвищення продуктивності виконання поставлених завдань користувачем, через їх візуалізацію. Створення нотатків із вказанням статусу завдання, дату та часу дедлайну, сильно спрощує користувачу флоу планування процесів. Портативність додатку дозволяє користувачу у будь-який момент часу відредагувати свій список завдань.

Створений програмний додаток написаний на мові програмування Java, характеризується зручністю у використанні, портативністю, зручним та інтуїтивно зрозумілим UI, а також швидкістю роботи, що забезпечує всі вимоги, які ставить користувач щодо роботи програмного додатку.

ANNOTATION

A mobile application for notes has been developed in the bachelor's thesis. The specified program is created for increase of productivity of performance of the set tasks by the user, through their visualization. Creating notes indicating the status of the task, the date and time of the deadline, greatly simplifies the flow of the user planning processes. The portability of the application allows the user to edit their to-do list at any time.

The created software application is written in the Java programming language, is characterized by ease of use, portability, user-friendly and intuitive UI, as well as speed, which provides all the requirements that the user places on the operation of the software application.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження.

Технології ніколи не стоять на місці, а за останні декілька років стрімко розвиваються. Все більше високотехнологічних девайсів стають доступнішими, а сервіси та додатки до них популярнішими. На сьогоднішній день важко уявити своє життя без мобільних додатків, отож великий попит продукує пропозицію. Сфера розробки мобільних додатків сильно розвинулася за останні декілька років, так як попит на них зростає з кожним місяцем.

Кожного дня, люди займаються удосконаленням всього, а для цього, потрібно слідкувати за своєю продуктивністю, щоб нічого не проґавити і завжди мати можливість відредагувати список своїх справ, та запланувати нові.

Використання різних паперових блокнотів, зошитів або чогось подібного не завжди зручно, а інколи ці підручні засоби не завжди під є рукою. Також доволі гостро стоїть питання зменшення використання паперової продукції, для збереження лісів, які гектарами вирубують на потреби паперової промисловості.

Потреба у менеджменті часу є найважливішим фактором у повсякденному житті кожної сучасної людини, незалежно від галузі роботи, навчання чи навіть звичного життя. Часто підручні гаджети мають вбудовані засоби для відстежування, регулювання та організації потоку завдань, але вони обмежені у функціях для створення та відображення завдань. Отже розробка програмного додатку для покращення тайм менеджменту користувача “TaskCheck” є актуальною.

Зв’язок роботи з науковими програмами, планами, темами.

Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження.

Метою бакалаврської дипломної роботи є підвищення якості планування часу користувача шляхом розробки та подальшого користування додатком для покращення тайм-менеджменту користувача.

Основними задачами дослідження є:

- Аналіз стану галузі вивчення мов програмування;
- Порівняльний аналіз аналогів;
- Аналіз методів розв’язання задачі;
- Постановка задач для мобільного додатку для покращення тайм менеджменту користувача;
- Розробку структури і алгоритмів програмного продукту;
- Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу;
- Розробка мобільного додатку для покращення тайм менеджменту користувача;
- Тестування програми.

Об’єкт дослідження – процес отримання, обробки, збереження та відображення даних у програмному додатку.

Предмет дослідження – методи і засоби отримання, обробки, збереження та відображення даних у мобільному додатку.

Методи дослідження.

У процесі досліджень використовувались: теорія структуризації реляційних БД, налаштування роботи сервера. Було досліджено принципи поєднання клієнтської та серверної частини проектів.

Новизна отриманих результатів.

Програмний додаток має модифікований метод створення та відображення записів завдань, який відрізняється від досліджуваних аналогів зрозумілістю, та обмеженою кількістю функціоналу, що дає більшу практичність додатку, і таким чином сприяє підвищенню якості використання додатку.

Практична цінність отриманих результатів.

Практична цінність досягнених результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень розроблено програмні засоби, що дозволяють створювати персональний планувальник цілей,

часу їх виконання а також встановлення статусу кожному із запланованих завдань.

Особистий внесок здобувача.

Усі наукові та прикладні результати, викладені у бакалаврській дипломній роботі, отримані автором особисто.

Технічне завдання наведено в додатку А.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз сучасного стану питання

З кожним днем технології все більше розвиваються і впроваджуються в наше повсякденне життя. Сьогодні багато процесів та завдань автоматизуються, а їх звичне виконання замінюють різні ґаджети: ноутбуки, планшети, телефони, розумні приставки тощо. Очевидно, що навіть планування дня, постановка задач, планування зустрічей та інше, мало хто виконує записуючи це собі у блокнот чи на листі паперу. Це пояснюється великою кількістю переваг у використанні різних додатків на мобільні пристрої, які завжди знаходяться під рукою.

Можна виділити декілька основних переваг цифрових носіїв:

- фізична захищеність даних від різних зовнішніх несприятливих факторів;
- простота та порівняна надійність у використанні;
- мінімізація втрати за рахунок знаходження віх даних в одному місці.

При використанні мобільного додатку з’являється ще декілька суттєвих переваг, які неможливо не помітити:

- доступ до своїх даних у будь який момент часу;
- захищеність від використання інформації іншими користувачами;
- можливість створювати та зберігати інформацію навіть без доступу в інтернет.

Зазвичай, сервіси для менеджменту досить спеціалізовані та містять в собі купу функціоналу, який ускладнює користувачу розуміння програми, і для того щоб користуватися цією програмою, потрібно потратити досить багато часу на її опанування. Велика кількість непотрібного функціоналу відволікає, та ускладнює процес створення персональних поміток, завдань чи цілей. Також більшість додатків доволі ресурсозатратні тому не підходять для більшості бюджетних ґаджетів.

1.2 Порівняльний аналіз аналогів

Для визначення актуальності розробки мобільного додатку, потрібно провести аналіз аналогів. Провівши порівняння було прийнято рішення про доцільність розробки власного додатку.

Оскільки тема менеджменту у сучасному світі досить актуальна, існує велика кількість додатків розроблених з метою вирішити певні проблеми користувачів.

Складність та наповненість різним функціоналом варіюється відповідно до певних категорій користувачів. Але спеціалісти із тайм-менеджменту сходяться на одній думці - простий та інтуїтивно зрозумілий інтерфейс повинен скоротити час планування та збільшити час для досягнення цілей. Мінімальна кількість основних функцій повинна підвищити наочність і простоту використання програми.

Отже, розглянемо декілька популярних мобільних додатків для створення нотатків та завдань.

Mobile for Jira (рис 1.1) – мобільна версія Jira Software. Вона входить до сімейства продуктів, розроблених з метою спростити управління робочим процесом для різних команд.

Спочатку система Jira створювалася як рішення для відстеження завдань та помилок. Але сьогодні Jira - це потужний інструмент управління роботою, що підходить для різних випадків, від управління вимогами і сценаріями тестування до agile-розробки програмного забезпечення [1].

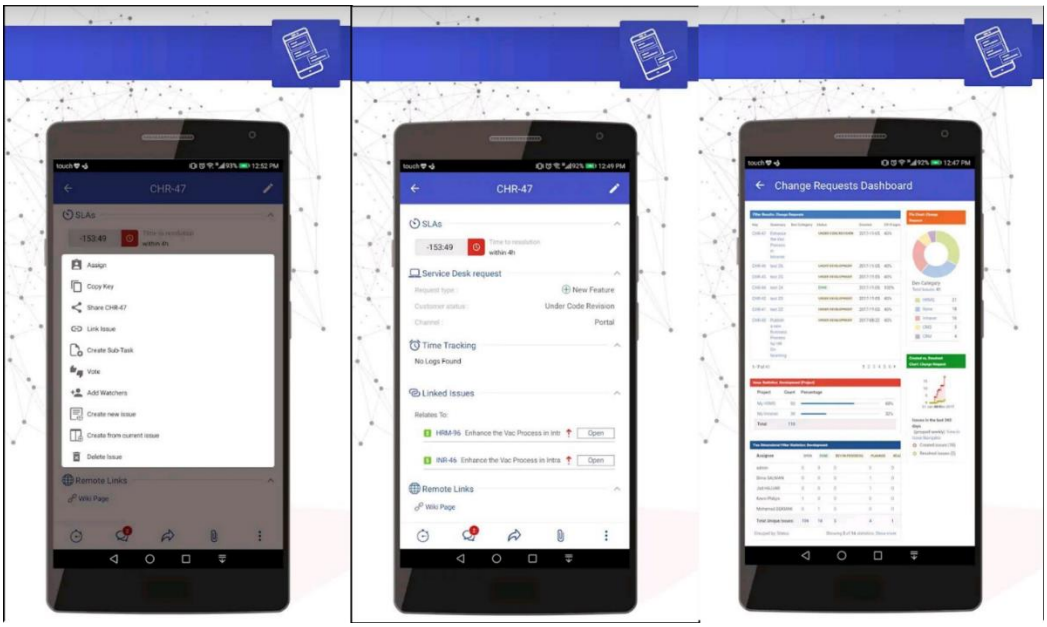


Рисунок 1.1 – Скріншоти додатку Mobile for Jira

Проект 365 (рис. 1.2) – Мобільний додаток для створення завдань, команди, а також для її спілкування. Також у застосунку присутня можливість створення діаграми Ганта [2].

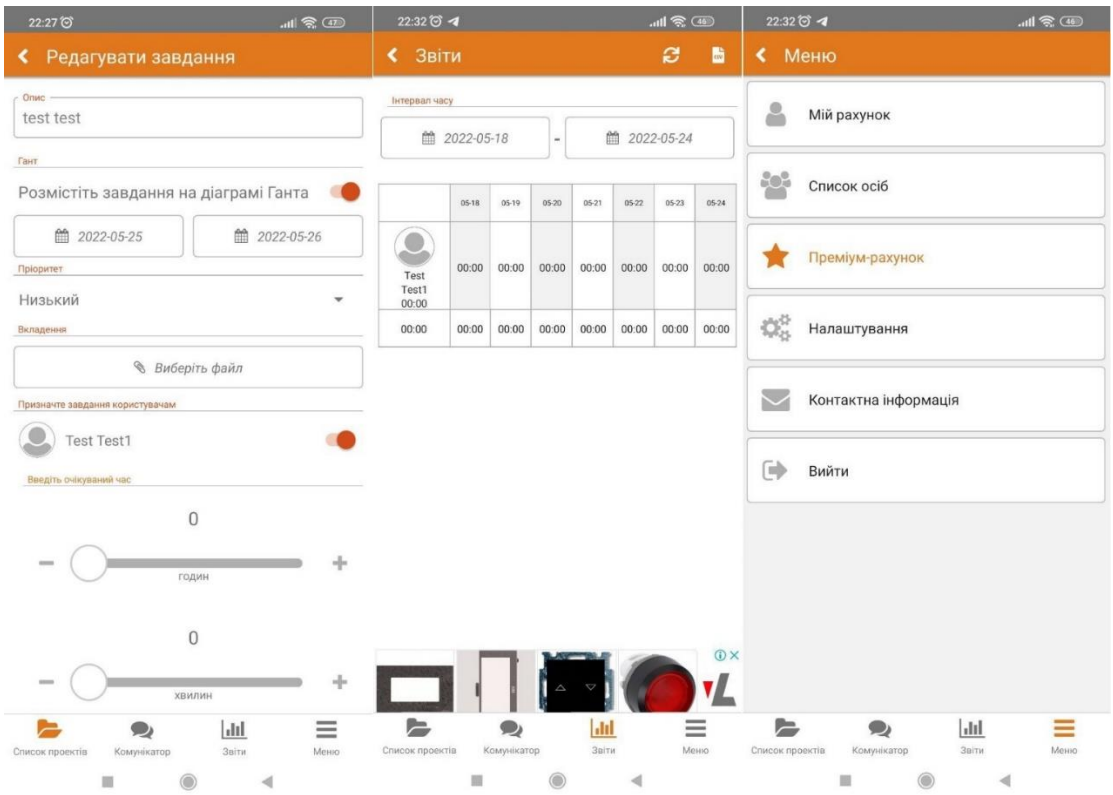


Рисунок 1.2 – Скріншоти додатку Проект 365

Trello (рис. 1.3) – це мобільна версія додатку Trello. Цей додаток є візуальним інструментом, який дає вашій команді можливість керувати будь-яким типом проекту, робочого процесу або відстеження завдань. Додайте файли, контрольні списки або навіть автоматизуйте: налаштуйте все, щоб ваша команда працювала найкраще. Просто зареєструйтеся, створіть дошку і готово.

Картки зберігають всю необхідну вашій команді інформацію організованою та в одному місці. Призначайте учасників, додавайте терміни, залишайте коментарі тощо[3].

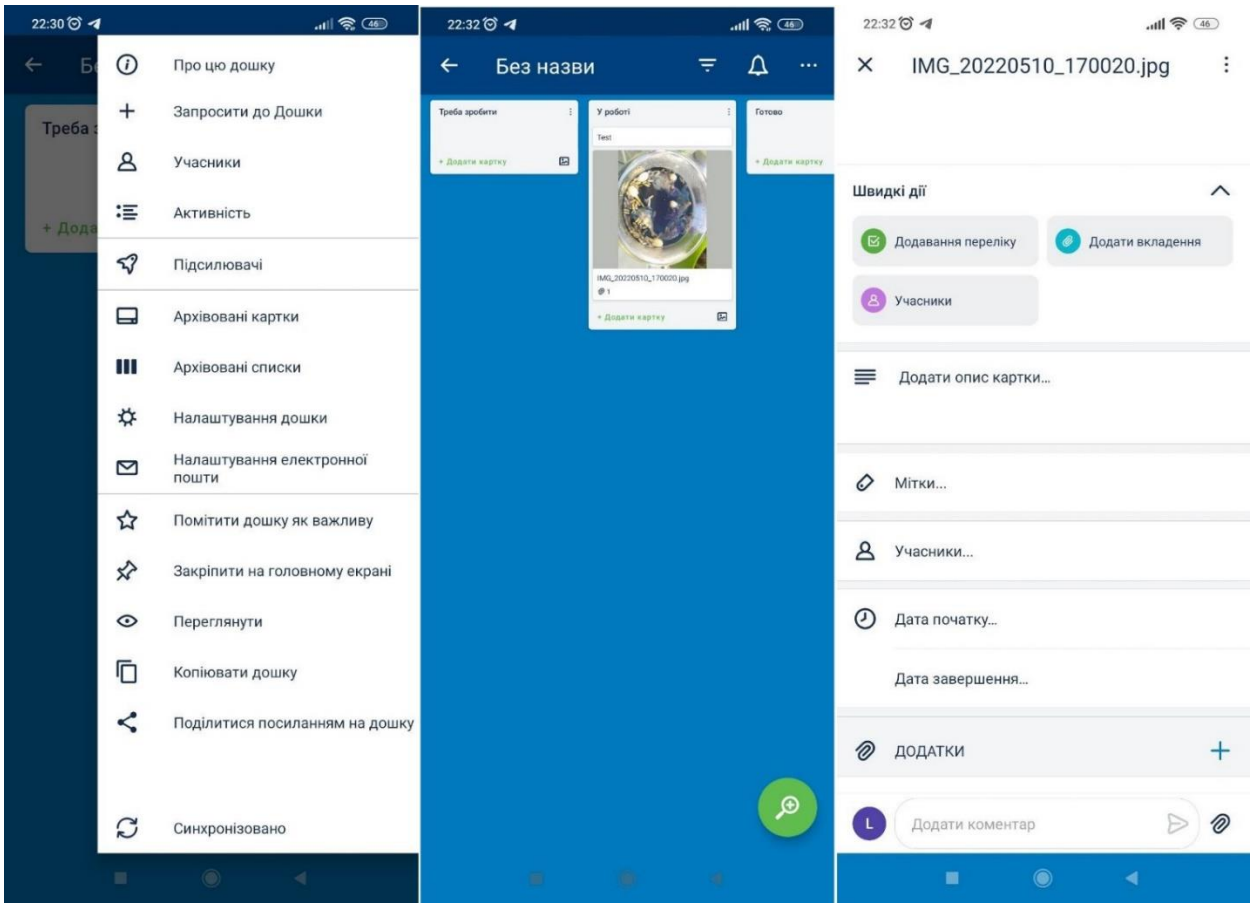


Рисунок 1.3 – Скріншот додатку Trello

Отже, провівши порівняння додатків наведених вище, було створено таблицю порівняння аналогів табл. 1.1.

Критерій	Trello	Проект 365	Mobile for Jira	TaskCheck

Орієнтований на створення нотаток	-	+	-	+
Простий у використанні	-	+/-	-	+
Повністю безкоштовний	-	-	-	+
Відсутність вбудованої реклами	+	-	+	+
Швидкість підготовки до початку роботи	-	-	-	+
Мобільний додаток	+	+	+	+

Перший критерій вказує, на що орієнтований додаток, тобто на які функції покладено основну увагу. Якщо додаток створено для відмінних від наших цілей, то навіть його другорядні функції не можуть повністю задовольнити наші потреби.

Другий і третій пункт дуже важливий для користувачів подібних додатків. Повна безкоштовність додатку знімає відчуття у користувача, що він користується чимось неповноцінним. Також за звичай дуже часто саме важливий функціонал стає доступний лише при внесенні певної суми. Простота у використані розширює кількість користувачів, адже простим додатком можуть користуватися як діти, так і люди літнього віку. Зазвичай така категорія людей доволі важко сприймає потребу навчатися чомусь новому, для виконання звичних речей, тому відсутність бар'єру у вигляді складного або надто великого функціоналу робить додаток більш привабливим.

Четвертий пункт є одним із важливих аспектів підбору додатку, адже ніхто не хоче бачити приховану рекламу під час використання додатку. Наявність вбудованої реклами дуже сильно знижує думку про додаток, тому її відсутність є дуже привабливим аспектом.

П'ятий пункт також є не менш важливим, адже якщо процес підготовки додатку до роботи(різного виду реєстрації, створення облікових записів, підтвердження електронної адреси тощо) стомлюють і знижують зацікавленість

додатком, адже є велика вірогідність того, що цей додаток не підійде користувачу, а потрачений час вже неможливо повернути.

Шостий пункт є не менш важливим за попередні, адже завдяки тому що додаток є мобільним, ним можна користуватися у будь-який момент і в будь-якій ситуації, адже мобільний телефон завжди поруч.

Отже, відповідно до таблиці порівняння характеристик, розробка власного програмного продукту є доцільною. В результаті розробки буде отримано продукт, який покриває всі недоліки існуючих рішень і забезпечить більш простий та ефективний спосіб створення задач та цілей для покращення тайм менеджменту користувача.

1.3 Аналіз методів вирішення поставленої задачі

На сьогодні, існує багато методів, які допоможуть вирішити розв’язання поставленої задачі. Навіть одну і ту саму задачу можна вирішити декількома способами. Можна застосовувати різні мови програмування, фреймворки або навіть різні бібліотеки. Це все впливає на те яким способом і як в кінцевому рахунку буде виглядати програмний продукт. Тому важливо визначити різні методи вирішення поставленої задачі.

Першим варіантом може бути веб-версія додатку, яка буде зберігатися на сторонньому сервері, а користувач матиме доступ до своїх даних із будь якого пристрою. Але в такому варіанті виникає проблема захищеності а також доступності. Проблема захищеності полягає у тому, що всю інформацію із веб додатку можуть викрасти шахраї, або дізнавшись коди доступу, інформація може потрапити до сторонніх людей. Проблема доступності полягає в тому, що не завжди є доступ до інтернету, тому в такі моменти, користувач не зможу отримати доступу до своїх даних.

Програмний додаток для покращення тайм менеджменту користувача може бути офлайн мобільний додаток. Це допоможе користувачу отримати

доступ до своїх даних на відкинути можливість викрадення особистих даних через шахрайство в мережі Internet.

Простий та інтуїтивно зрозумілий інтерфейс має збільшити цікавість користувачів різних груп. Мінімальна кількість базових функцій має збільшити попит користувачів для використання зазначеного додатку у звичайному житті.

1.4 Постановка задач на проектування

Бакалаврська дипломна робота присвячена розробці додатку для покращення тайм менеджменту користувача: його алгоритмів і інтерфейсу. Для його інтерфейсу було висунуто такі вимоги:

- Простий і зручний в користуванні;
- Гарний дизайн, який викликає позитивні емоції користувача;

Для алгоритмів додатку було висунуто такі вимоги:

- Визначити найбільш ефективний алгоритм зчитування інформації;
- Визначити найбільш ефективний алгоритм збереження інформації;
- Розробити алгоритм зчитування інформації;
- Розробити алгоритм збереження інформації;
- Розробити алгоритм видалення інформації;
- Розробити програмний продукт, призначений для вирішення проблеми;
- Провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

1.4 Висновок

У першому розділі було розглянуто питання тайм менеджменту користувача на сьогоднішній день відповідно до технологічного розвитку. Також, було проаналізовано варіанти вирішення цього питання за рахунок порівняння аналогів між собою та додатком, що розробляється. В результаті

порівняння було виявлено доцільність розробки бакалаврської дипломної роботи та можливі методи по вирішенню питань. Також було поставлено завдання, які буде необхідно розробити у програмному продукті.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Розробка структури інтерфейсу для мобільного додатку TaskCheck

Одним із важливих аспектів розробки, є розробка інтерфейсу додатку. Для того, щоб користувачу хотілося використовувати додаток, потрібно, щоб інтерфейс був зручним та зрозумілим, а також мав приємні для ока кольори.

UX дизайн мобільних додатків проектує їхні функції та властивості. Скільки кроків потрібно користувачеві, щоб досягти мети? Які додаткові можливості будуть у нього на цьому шляху (прочитати відгуки, характеристики, перевірити фото, поставити оцінку), чи потрібне підключення до інтернету для виконання тих чи інших завдань [4].

Іноді кольорами легше описати стан, чи викликати якісь почуття аніж звичайним текстом. Тож цілком очевидно, що кольори у мобільному додатку відіграють не менше значення, ніж функціонал додатку. Часто буває так, що незважаючи на свій функціонал та потрібність, додаток не може втримати користувача, через невдало підібрані кольори. Це може зумовлюватися занадто яскравими відтінками, які сильно відбивають бажання у користуванні додатком або невдале їх поєднання. Тому для досягнення кращого ефекту, дизайнер має правильно підібрати кольорову гаму для додатку, щоб той став більш привабливим для користувача [5].

Процес створення дизайну додатку розпочинається підчас розробки дизайн-макету, прототипу, на якому зазначаються розташування та вибір всіх елементів, а також підбирається кольорова гама та інше.

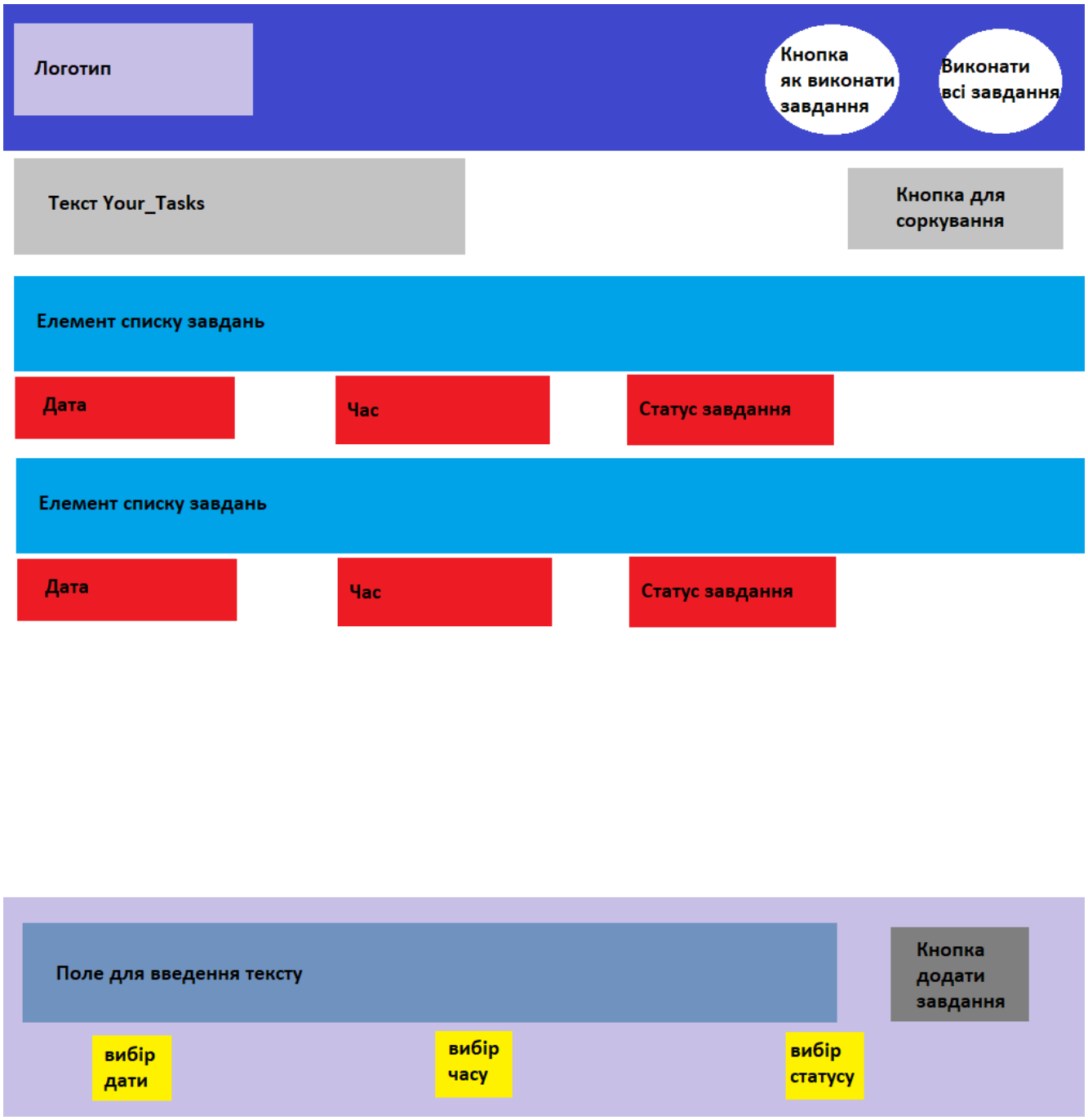


Рисунок 2.1 – Структура сторінки створення нотаток

Зображена структура мобільного додатку для покращення тайм менеджменту користувача, дозволяє досить швидко створити нове або “Виконати” завдання (рис 2.1). Спокійна кольорова гама не відволікає користувача на якісь сторонні речі, а фокусує увагу саме на головному.

2.2 Візуальне моделювання програмного додатку

Будь-яке програмне забезпечення можна описати у вигляді діаграм. Перед розробкою цілком доцільно спочатку розробити діаграми, а потім відповідно до них і створити програмний продукт.

Діаграми послідовності UML – це діаграми взаємодії, які детально описують, як виконуються операції. Вони фіксують взаємодію між об’єктами в контексті співпраці. Діаграми послідовності – це час, вони візуально показують порядок взаємодії, використовуючи вертикальну вісь діаграми, щоб відобразити час, які повідомлення надсилаються і коли [5].

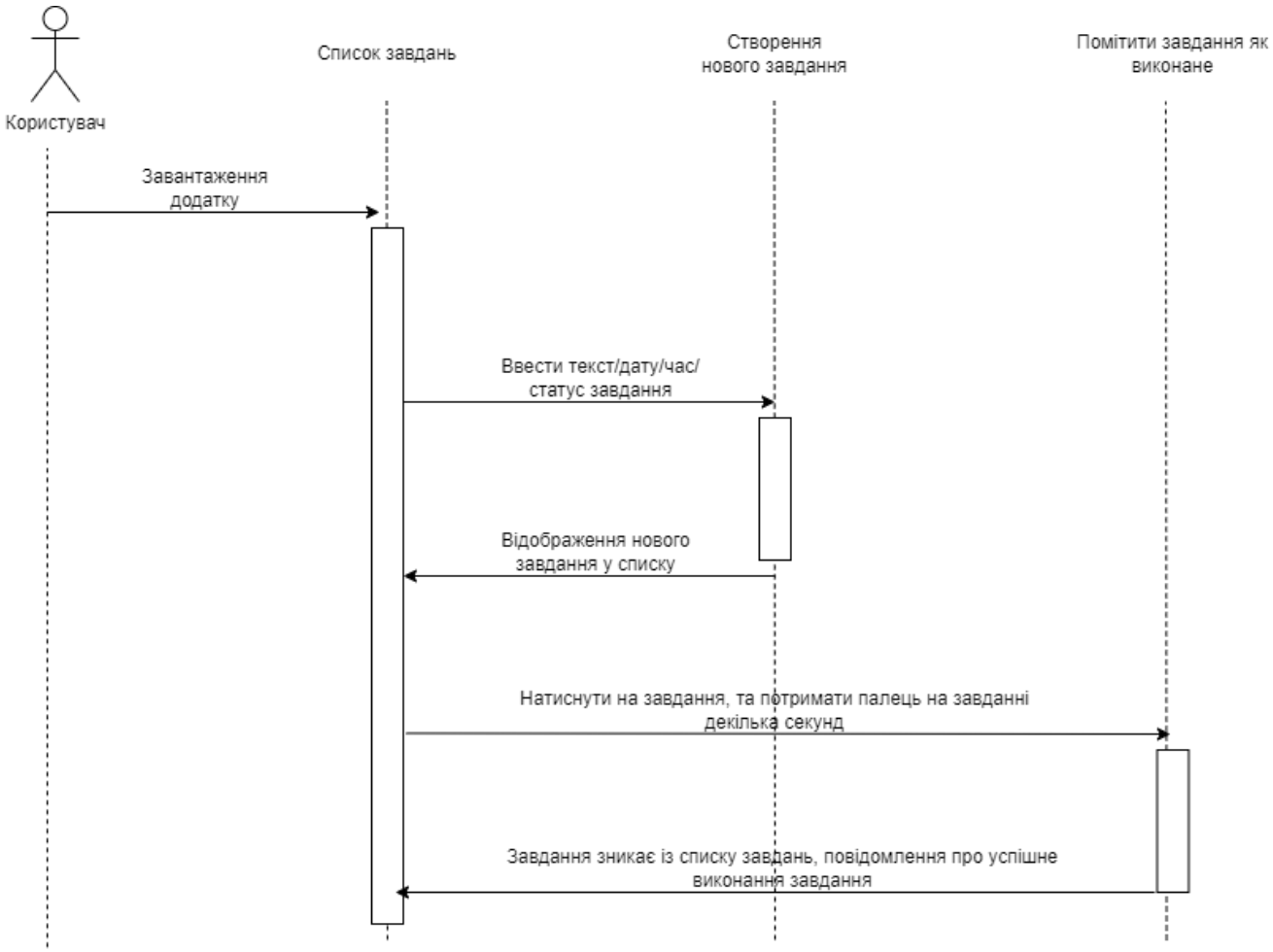


Рисунок 2.2 – Діаграма послідовності додатку

Зважаючи на те, що час – ключовий параметр у розроблюваному мобільному додатку тайм менеджменту користувача, розробка діаграми послідовності може візуально зобразити роботу головної функції додатку – створення завдань та цілей (рис.2.2).

Діаграма діяльності є ще однією важливою діаграмою поведінки в діаграмі UML для опису динамічних аспектів системи. Діаграма діяльності, по суті, є розширеною версією блок-схеми, яка моделює перехід від однієї діяльності до іншої.

Діаграми діяльності описують, як діяльність координується для надання послуги, яка може бути на різних рівнях абстракції. Як правило, подія повинна бути досягнута деякими операціями, особливо якщо операція призначена для досягнення ряду різних речей, які потребують координації, або як події в одному випадку використання пов’язані одна з одною, зокрема, випадки використання, коли дії можуть перетинатися і вимагати координації. Діаграма також підходить для моделювання того, як набір варіантів використання координується для представлення робочих процесів бізнесу [5].

Тому зважаючи на це, було прийнято рішення розробити діаграму діяльності, щоб описати роботу додатку, та створити більш чітке розуміння, яким має бути створюваний додаток (рис. 2.3).

Діаграма пояснює певну послідовність подій, яку має виконувати користувач: після відкриття додатку можна створити нову нотатку, редагуючи основні параметри: введення тексту, вибір дати, часу, і призначення статусу нотатки. Новостворена нотатка відображається в додатку. При повторному запуску чи продовженні користування додатком можна надалі працювати із нотатками – помітити завдання, як виконане чи помітити всі завдання, як виконані.

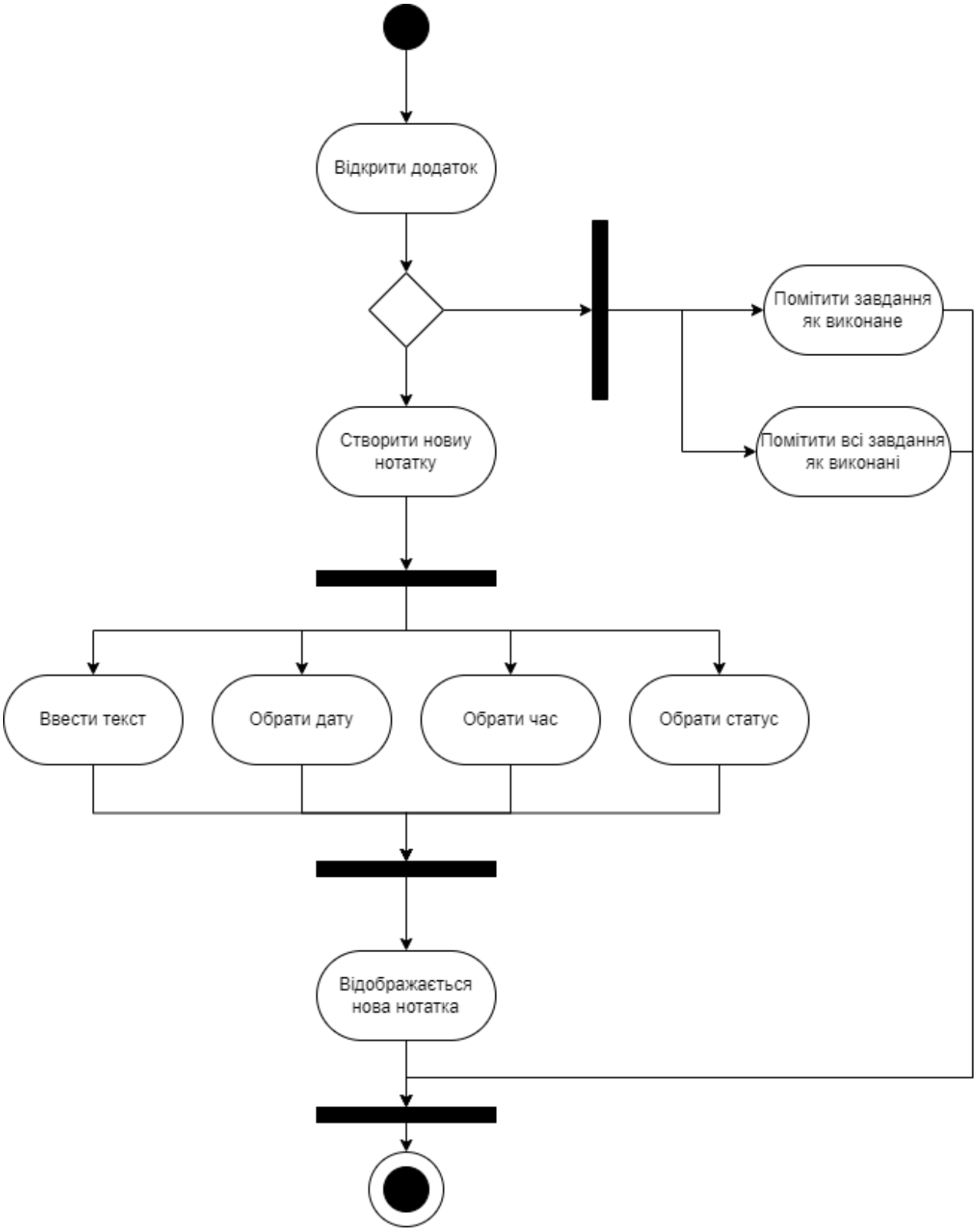


Рисунок 2.3 – Діаграма діяльності

2.3 Розробка алгоритму створення та відображення записів завдань

У сучасному світі цифрових технологій, програмування є основою для роботи різноманітних комп'ютерів, гаджетів та іншого електронного обладнання. А вміння швидко і правильно скласти блок-схему алгоритму є основою цієї науки. Така схема є графічною моделлю процесів, які має виконувати обладнання. Він складається з окремих функціональних блоків, які виконують різні функції [7].

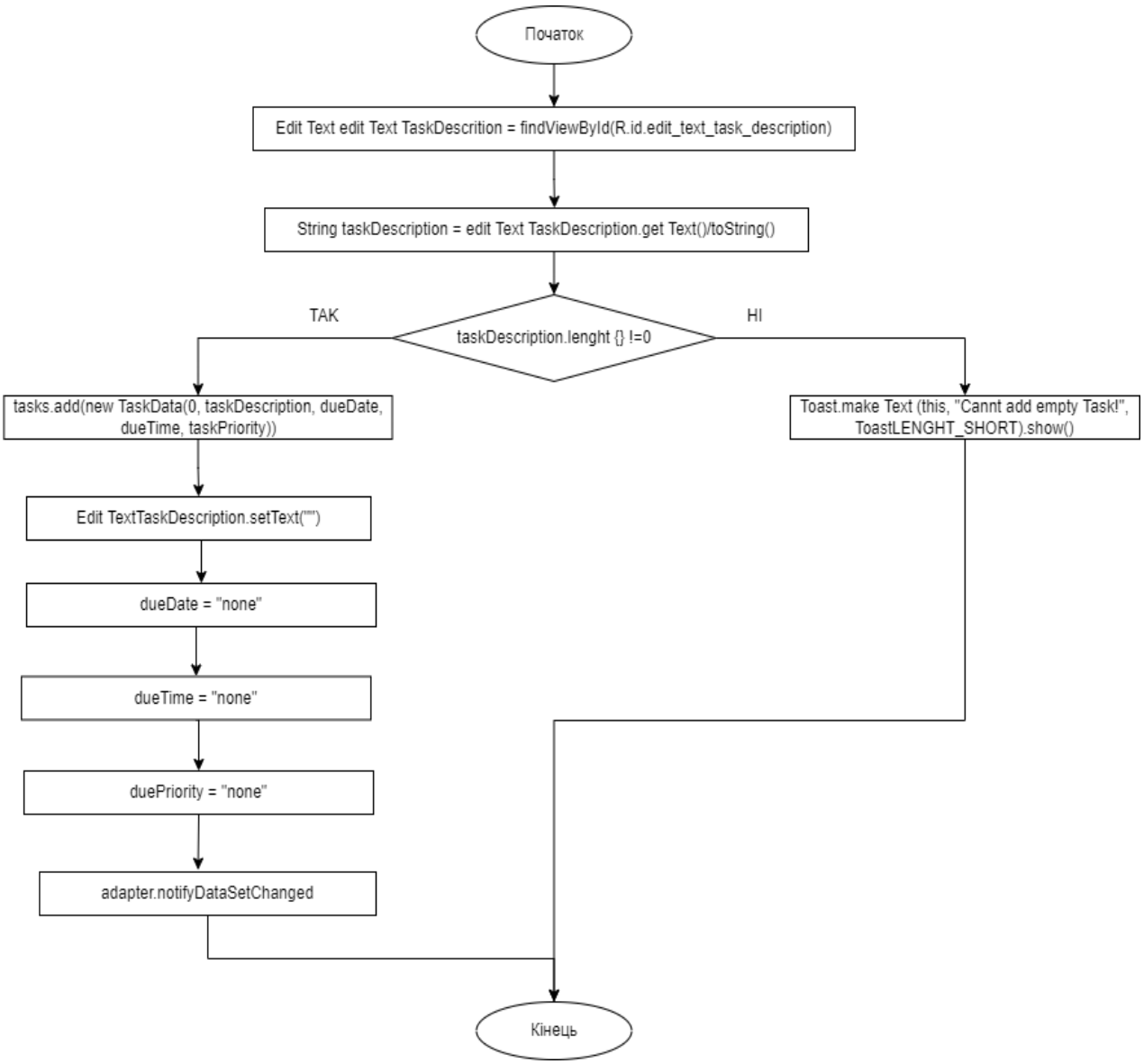


Рисунок 2.4 – Блок-схема алгоритму додавання нових завдань

Основною функцією додатка є створення завдань. Створення нового завдання, а також недопущення створення завдання без визначеної дати, часу або статусу є функцією додатка, що потребує більшого уточнення, тому було розроблено блок-схему алгоритму створення нового завдання (рис. 2.4).

2.4 Висновок

У другому розділі було проаналізовано інтерфейс програмного продукту, та на основі аналізу, було розроблена структура та прототип мобільного додатку. Обраний інтерфейс буде зручний у використані та інтуїтивно зрозумілий, а мала кількість елементів не буде відволікати увагу користувача, отже він буде братися за основу при розроблені фінальної версії мобільного додатку. Також було розроблено діаграми діяльності та послідовності, що допоможе краще зрозуміти, яким має бути фінальний програмний продукт. Також було створено блок-схему алгоритму додавання нових завдань, адже це основна функція додатку що розробляється.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ДЛЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації

При розробці будь-якого мобільного додатку визначну роль займає вибір мови програмування, на якій додаток буде створюватися, адже від прийнятого рішення буде залежати досить багато, починаючи від середовища розробки, закінчуючи складністю реалізації тих чи інших функцій.

Java — це мова програмування та обчислювальна платформа, вперше випущена Sun Microsystems у 1995 році. Вона розвинулась із скромних початків до забезпечення великої частки сучасного цифрового світу, забезпечуючи надійну платформу, на якій побудовано багато послуг і програм. Нові, інноваційні продукти та цифрові послуги, розроблені для майбутнього, також продовжують покладатися на Java [8].

У той час як більшість сучасних програм Java об’єднують середовище виконання Java та програму разом, все ще існує багато програм і навіть деякі веб-сайти, які не працюватимуть, якщо у вас не встановлено Java для настільного комп’ютера [8].

Java використовується для написання програм для різних платформ, які запускають JRE, і підтримує програми, які запускаються на одному пристрої, наприклад на настільному комп’ютері або мобільному телефоні. Java також можна використовувати для розробки програм, які працюють розподіленим способом. Це означає, що одна і та ж програма може бути розподілена між серверами або клієнтами в мережі і може виконуватися синхронно. Java також можна використовувати для написання програмних модулів або applets як частини веб-сторінок [9].

Отож, якщо підвести підсумки вищеописаного, то можна виділити такі види додатків, для яких доцільно використовувати мову програмування Java:

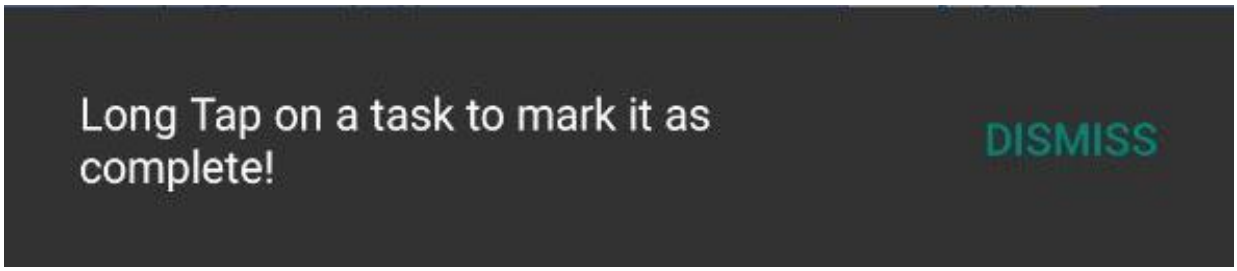
- GUI програми;
- Веб-сервери та сервери програм;

- Додатки проміжного програмного забезпечення;
- Веб-додатки;
- Мобільні додатки;
- Вбудовані системи;
- Корпоративні програми.

3.2 Розробка програмного модуля для відображення snackbar

Після першого завантаження додатку, користувачу потрібно якось зрозуміти, як саме потрібно використовувати додаток. За відсутності короткого пояснення, як саме працює та чи інша функція додатку, користувачу буде важко на осліп дізнаватися цю інформацію.

Тому, проаналізувавши дану проблему, було прийнято рішення розробити snackbar, який буде з’являтися при першому запуску додатку користувачем, і давати інформацію, як саме можна помітити завдання, як виконане (рис. 3.1).



Рисуок 3.1 – Snackbar повідомлення

Тому зрозумівши потребу у розробці snackbar, його важливість для користувачів, які вперше бачать додаток, було прийнято рішення розробити модуль, для відображення snackbar (рис. 3.2).

```
78    private void displayIntroSnackbar() {
79        final Snackbar snackbar = Snackbar.make(findViewById(R.id.parent_view), R.string.howto_mark_complete, Snackbar.LENGTH_LONG);
80        snackbar.setAction("Dismiss", v -> snackbar.dismiss()).show();
81    }
```

Рисунок 3.2 – Модуль відображення Snackbar

Основною його перевагою є те, що він не буде переривати роботу користувача і забезпечує високий рівень юзабіліті.

3.3 Розробка програмного модуля для додавання завдання

Головним модулем мобільного додатку, можна вважати модуль для створення завдань. Завдяки ньому, користувачі можуть створювати завдання та обирати дату та час коли потрібно виконати заплановану дію.

Також можна обрати статус запланованого завдання термінове, швидке завдання (завдання, що не займе багато часу) та регулярне (те завдання, яке потрібно виконувати регулярно) (рис. 3.3).

```
99      /**
100      * Adds a task to the current list and notifies recyclerView of dataset change
101      */
102     public void addTask(View view) {
103         EditText editTextTaskDescription = findViewById(R.id.edit_text_task_description);
104         String taskDescription = editTextTaskDescription.getText().toString();
105
106         // If non-empty string is received, add it to the task list and notify adapter
107         if (taskDescription.length() != 0) {
108             tasks.add(new TaskData(0, taskDescription, dueDate, dueTime, taskPriority));
109             editTextTaskDescription.setText("");    // Resets Task Input field
110
111             // Resets the optional task attributes
112             dueDate = "none";
113             dueTime = "none";
114             taskPriority = "none";
115             adapter.notifyDataSetChanged();
116         } else {
117             // Errors out with a toast.
118             Toast.makeText(this, "Cannot add empty Task!", Toast.LENGTH_SHORT).show();
119         }
120     }
```

Рисунок 3.3 – Модуль створення завдань

Також варто розглянути модуль, для помітки завдання як виконане. В додатку це реалізоване таким чином, що завдання помічається як виконане, якщо користувач натискає на завдання і утримує палець на ньому протягом декількох секунд (рис. 3.4, рис. 3.5)

```
54          // Long click deletes the task from the list and displays a toast
55          holder.itemView.setOnLongClickListener(view -> {
56              MainActivity.tasks.remove(position);
57              notifyDataSetChanged();
58              Toast.makeText(holder.itemView.getContext(), "Task completed!", Toast.LENGTH_SHORT).show();
59              return true;
60          });
61      }
62
```

Рисунок 3.4 – Відображення toast про виконання завдання

```
63      @Override
64      public int getItemCount() {
65          return MainActivity.tasks.size();
66      }
67
68      public static class ViewHolder extends RecyclerView.ViewHolder {
69
70          private final TextView taskDescriptionTextView;
71          private final TextView taskDueDateTextView;
72          private final TextView taskDueTimeTextView;
73          private final TextView taskPriority;
74
75          public ViewHolder(@NonNull View itemView) {
76              super(itemView);
77
78              taskDescriptionTextView = itemView.findViewById(R.id.text_view_single_task_description);
79              taskDueTimeTextView = itemView.findViewById(R.id.task_due_time);
80              taskDueDateTextView = itemView.findViewById(R.id.task_due_date);
81              taskPriority = itemView.findViewById(R.id.task_priority);
82          }
83
84          public TextView getTextView() {
85              return taskDescriptionTextView;
86          }
87
88          public TextView getTaskDueDateTextView() {
89              return taskDueDateTextView;
90          }
91
92          public TextView getTaskDueTimeTextView() {
93              return taskDueTimeTextView;
94          }
95
96          public TextView getTaskPriority() {
97              return taskPriority;
```

Рисунок 3.5 – Приховання завдань, при довгому натисненні

Для того, щоб створені завдання не зникали після того, як користувач вийде з програми, потрібно, щоб всі зроблені ним зміни збереглися, інакше ніякого сенсу у створені завдань не буде. Тому для забезпечення цієї функції було створено модуль для збереження інформації перед закриттям додатку (рис. 3.6).

```
83      /**
84       * Used for saving the data in the app to SharedPreferences before the app closes.
85       */
86      @Override
87      protected void onPause() {
88          super.onPause();
89
90          // Indicates that the app has been launched for the first time, so no SnackBar from the next time
91          getPreferences(Context.MODE_PRIVATE).edit().putBoolean("display_intro", false).apply();
92
93          // Save tasks data to the database
94          if (tasks.size() > 0) {
95              database.taskDao().insertAll(tasks);
96          }
97      }
98
```

Рисунок 3.6 – Блок для збереження інформації

3.3 Розробка програмного модуля для роботи Picker та сортування

Для того, щоб користувач міг обрати дату час та статус завдання потрібно реалізувати такі блоки:

1. DatePicker – для того, щоб користувач міг у зручній формі обрати дату коли потрібно виконати завдання (рис. 3.4);
2. TimePicker – для того щоб користувач міг обрати час, о котрій потрібно виконати завдання у зручній формі (рис. 3.5);
3. Popup task priority – спливаюче вікно, в якому користувач може обрати один із трьох статусів завдань(пріоритетів) (рис. 3.6).

```
162     public void showDatePickerDialog(View view) {
163         DialogFragment fragment = new DatePickerFragment();
164         fragment.show(getSupportFragmentManager(), "datePicker");
165     }
```

Рисунок 3.4 – Програмний модуль DatePicker

```
170     public void showTimePickerDialog(View view) {
171         DialogFragment fragment = new TimePickerFragment();
172         fragment.show(getSupportFragmentManager(), "timePicker");
173     }
```

Рисунок 3.5 – Програмний модуль TimePicker

```
178     public void showPriorityMenu(View view) {
179         PopupMenu popup = new PopupMenu(this, view);
180         popup.setOnMenuItemClickListener(this);
181         getMenuInflater().inflate(R.menu.priority_menu, popup.getMenu());
182         popup.show();
183     }
```

Рисунок 3.6 – Програмний модуль Popup task priority

Також доволі зручним і потрібним є можливість сортування. Можливість відсортувати свої завдання по даті, або по пріоритетності є досить важлива функція, тому було прийнято рішення реалізувати її у мобільному додатку (рис. 3.7).

```
174
175     /**
176      * Instantiates and displays a new Popup menu to set task priority
177      */
178     public void showPriorityMenu(View view) {
179         PopupMenu popup = new PopupMenu(this, view);
180         popup.setOnMenuItemClickListener(this);
181         getMenuInflater().inflate(R.menu.priority_menu, popup.getMenu());
182         popup.show();
183     }
184
185     /**
186      * Instantiates and displays a Popup menu to sort lists by either due time or priority
187      */
188     public void showSortByPopupMenu(View view) {
189         PopupMenu popupMenu = new PopupMenu(this, view);
190         popupMenu.setOnMenuItemClickListener(this);
191         getMenuInflater().inflate(R.menu.sort_menu, popupMenu.getMenu());
192         popupMenu.show();
193     }
```

Рисунок 3.7 – Реалізація програмних модулів сортування

3.4 Розробка програмного модуля для обробника подій

Для того, щоб попередньо попередньо-описані модулі працювали правильно, та й взагалі працювали, потрібно розробити модуль для оброблення подій, який буде обробляти події кліку в спливаючому меню. Він застосовується як для сортування, так і для пріоритету завдань в меню налаштувань (рис. 3.8).

```
201     @SuppressLint("NonConstantResourceId")
202     @Override
203     public boolean onMenuItemClick(MenuItem menuItem) {
204         switch (menuItem.getItemId()) {
205             case R.id.action_priority_urgent:
206                 taskPriority = "Urgent";
207                 return true;
208             case R.id.action_priority_rushed:
209                 taskPriority = "Rushed";
210                 return true;
211             case R.id.action_priority_regular:
212                 taskPriority = "Regular";
213                 return true;
214             case R.id.action_sort_by_date:
215                 Helper.sortByDate(tasks);
216                 adapter.notifyDataSetChanged();
217                 return true;
218             case R.id.action_sort_by_priority:
219                 Helper.sortByPriority(tasks);
220                 adapter.notifyDataSetChanged();
221             default:
222                 return false;
223         }
224     }
```

Рисунок 3.8 – Обробник подій

3.5 Висновки

В третьому розділі бакалаврської роботи, було обґрунтовано вибір мови програмування, який використовуватиметься при розробці мобільного додатку, а також наведено головні її переваги. Також було розроблено модулі для відображення snackbar, додавання завдання, Picker та сортування а також для обробника подій.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Методи тестування програмного забезпечення

Тестування програмного забезпечення — перевірка відповідності між реальною та очікуваною поведінкою програми, що здійснюється на наборі тестів, обраному певним чином. У більш широкому сенсі, тестування - це одна з технік контролю якості, що включає активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis) [10].

Якість програмного забезпечення (Software Quality) - це сукупність параметрів програмного забезпечення, що належать до його здатності задовольняти встановлені та передбачувані потреби. (Quality management та quality assurance).

Цілю тестування є підвищити ймовірність того, що програма, призначена для тестування, буде працювати правильно за будь-яких обставин, а також, підвищити ймовірність того, що програма, призначена для тестування, буде відповідати всім описаним вимогам та надання актуальної інформації про стан продукту конкретний момент часу [11].

Існує сім принципів тестування:

1. Тестування демонструє наявність дефектів (Testing shows presence of defects). Тестування може показати, що дефекти присутні, але не може довести, що їх немає.

2. Вичерпне тестування є недосяжним (Exhaustive testing is impossible). Повне тестування з допомогою всіх комбінацій вводів і передумов фізично нездійсненно, крім виняткових випадків.

3. Ранне тестування (Early testing). Щоб знайти дефекти якомога раніше, активності з тестування повинні бути розпочаті якомога раніше в життєвому циклі розробки програмного забезпечення або системи, і повинні бути сфокусовані на певних цілях.

4. Скупчення дефектів (Defects Clustering). Зусилля тестування повинні бути зосереджені пропорційно до очікуваної, а пізніше реальної щільності дефектів по модулях.

5. Парадокс пестициду (Pesticide paradox). Якщо ті самі тести будуть проганятися багато разів, зрештою цей набір тестових сценаріїв більше не знаходитиме нових дефектів.

6. Тестування залежить від контексту (Testing is concept depending). Тестування виконується по-різному, залежно від контексту.

7. Помилка про відсутність помилок (Absence-of-errors fallacy). Виявлення та виправлення дефектів не допоможуть, якщо створена система не підходить користувачеві та не задовольняє його очікуванням та потребам.

Статус дефекту або статус помилки в життєвому циклі дефекту – це поточний стан, з якого в даний момент перебуває дефект або помилка. Метою статусу дефекту є точна передача поточного стану або прогресу дефекту чи помилки, щоб краще відстежувати та розуміти фактичний прогрес життєвого циклу дефекту [11].

Кількість станів, через які проходить дефект, варіюється від проекту до проекту. Наведена діаграма життєвого циклу охоплює всі можливі стани (рис. 4.1).

- Новий – цей статус надається дефекту, який був знайдений вперше;
- Призначено – після виявлення дефекту, він назначається на розробника, який буде виправляти знайдений дефект;
- Відкритий – розробник починає аналізувати та виправляти помилку;
- Виправлено – розробник вніс правки;
- Очікує повторного тестування – як тільки дефект виправлено, розробник «пушить» зміни та робить нову збірку для тестувальника;
- Повторне тестування – тестувальник проводить повторне тестування дефекту;

- Перевірено(закрито) – дефект протестовано, помилка не відтворюється;
- Повторне відкриття – дефект не пройшов тестування, він повертається назад до розробника для виправлення;
- Дублікат – якщо дефект повторюється двічі;
- Відхилено – якщо дефект насправді не є дефектом;
- Відкладено – помилка має низький пріоритет і не потребує першочергового виправлення;
- Не помилка – якщо це не впливає на функціональність програми, тоді помилка має статус «Не помилка».

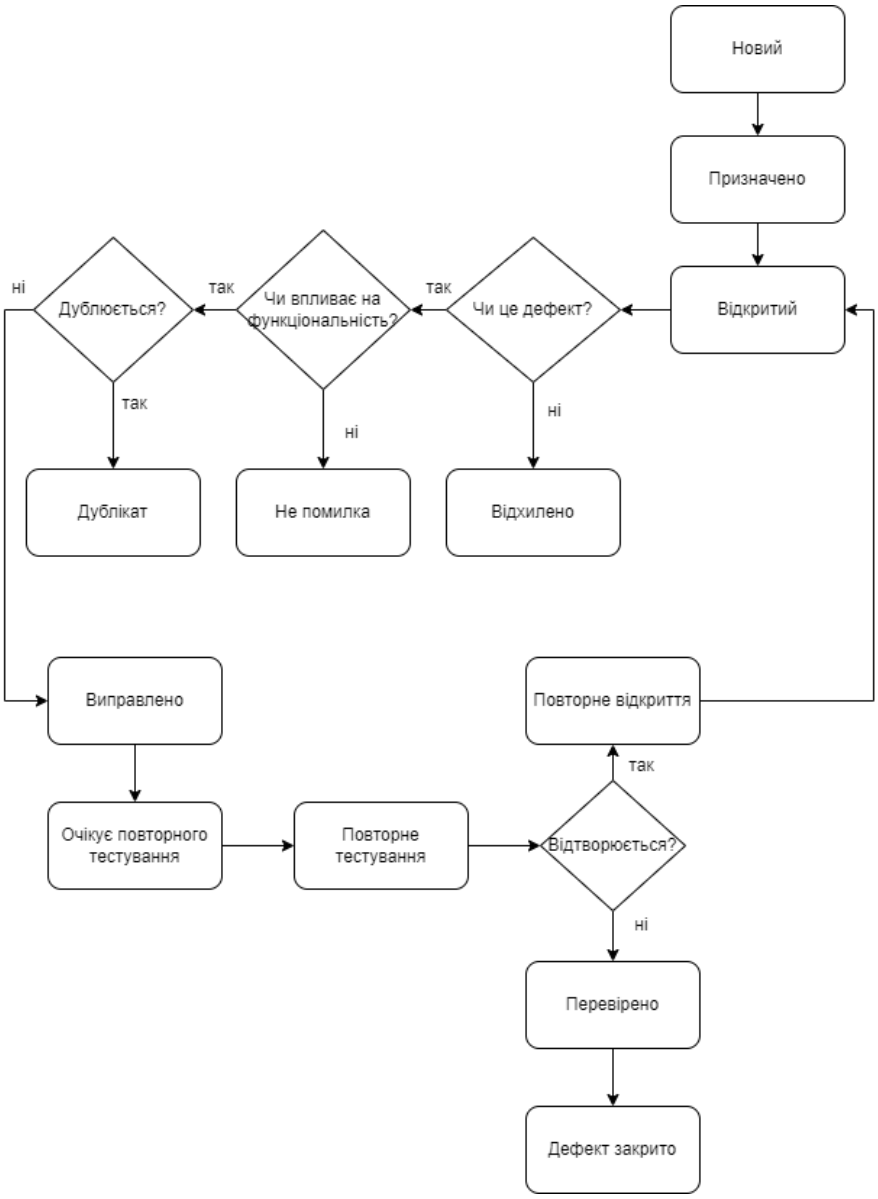


Рисунок 4.1 – Життєвий цикл дефекту

4.2 Тестування розробленого програмного продукту

Превіремо відображення snackbar із підказкою при першому відкритті додатку. Для тестування цього модулю додатку напишемо позитивний Test case (таблиця 4.1).

Таблиця 4.1 – Виконання Test case відображення snackbar

№	Назва	Кроки	Очікуваний результат	Статус
1	Відображення snackbar	1. Завантажити додаток;	Додаток успішно завантажується на мобільний пристрій	Виконано
		2.Відкрити додаток.	Додаток відкривається, snackbar із підказкою відображається	

Перевірка тестового випадку пройшла успішно, під час першого відкриття додатку відображення snackbar із підказкою, як помітити завдання, що воно виконане(рис. 4.2).

Головною функцією мобільного додатку є створення завдань, тому цей модуль потрібно перевірити якомога детальніше.

Окрім позитивних Test case, які показують як додаток реагує на валідні дані, також неменж важливою перевіркою є перевірка додатку при не валідних дій, тобто дії користувача, які не відповідають очікуванню додатку.

Тож перевіримо функцію створення нового завдання. Для тестування цього модулю додатку створено кілька тестових випадків: позитивного та негативного (таблиця 4.2).

Таблиця 4.2 – Перевірка Test cases створення нового завдання

№	Назва	Кроки	Очікуваний результат	Статус
2	Успішне створення завдання	1. Відкрити додаток;	Додаток завантажується. Головна сторінка відображається.	Виконано
		2. Ввести завдання в поле «Task Description»;	У полі «Task Description» відображається введений текст.	
		3. Натиснути на іконку дати;	Відкривається picker дати.	
		4. Вибрати дату;	Дата обирається, календар закривається.	
		5. Натиснути на іконку часу;	Відкривається time picker.	
		6. Вибрати час;	Час обирається, time picker закривається.	
		7. Натиснути на іконку пріоритету	Відкривається drop-down list із пріоритетами	
		8. Обрати пріоритет;	Пріоритет обирається, drop-down list закривається	
		9. Натиснути на кнопку «Add»;	Нова задача додається до списку задач	

Продовження таблиці 4.2 - Перевірка Test cases створення нового завдання

3	Створення завдання без дати	1. Відкрити додаток;	Додаток завантажується. Головна сторінка відображається.	Виконано
		2. Натиснути на іконку дати;	Відкривається picker дати.	
		3. Вибрати дату;	Дата обирається, календар закривається.	
		4. Натиснути на іконку часу;	Відкривається time picker.	
		5. Вибрати час;	Час обирається, time picker закривається.	
		6. Натиснути на іконку пріоритету	Відкривається drop-down list із пріоритетами	
		7. Обрати пріоритет;	Пріоритет обирається, drop-down list закривається	
		8. Натиснути на кнопку «Add»;	Повідомлення про помилку відображається	

Перевірка тестового випадку пройшла успішно.

При позитивному виконанні тестового випадку, нове завдання додається до списку завдань. При негативному виконанні – завдання не створюється, а відображається повідомлення про помилку.

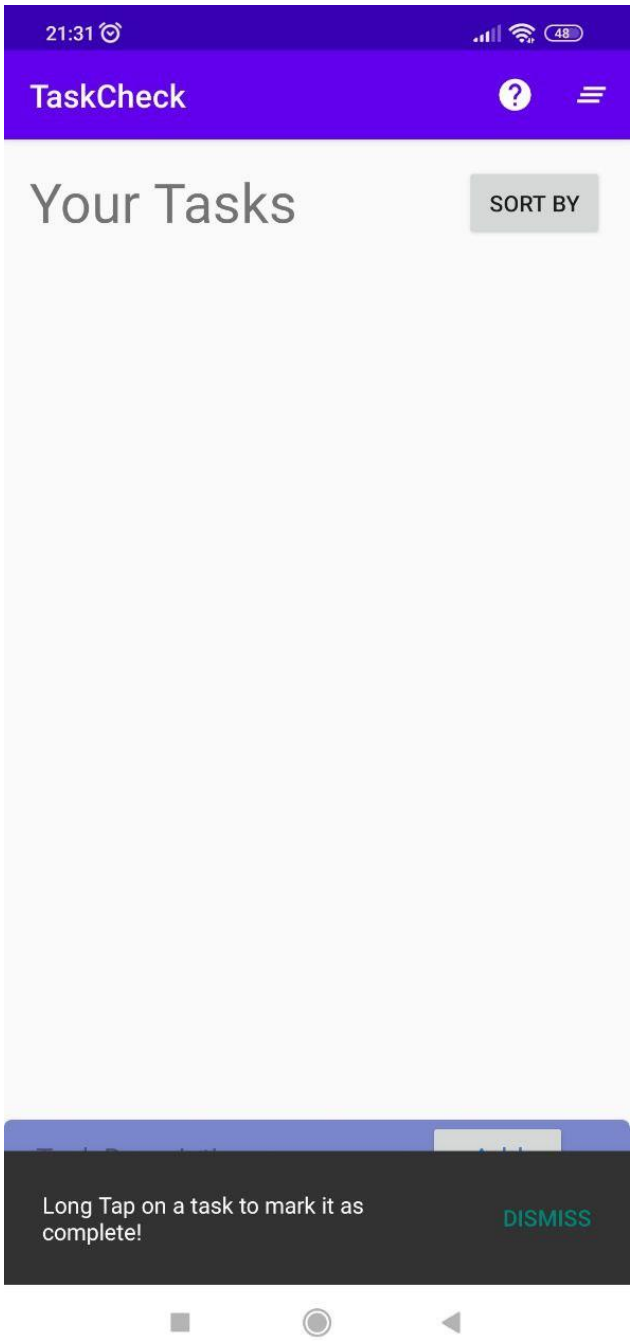


Рисунок 4.2 – Відображення snackbar із підказкою

Результати виконання тестування зображені на рисунках 4.3 та 4.4.

Не менш важливою та зручною річчю є сортування. В додатку реалізоване сортування за двома параметрами.

Перший параметр – сортування за датою. Використовуючи це сортування, користувач може відсортувати свої справи відповідно до дати, та перевірити, які завдання потрібно буде зробити найближчим часом.

Інший параметр, це сортування за пріоритетом. Такий тип сортування допоможе користувачу глянути, я які завдання є невідкладними та потребують уваги найближчим часом.

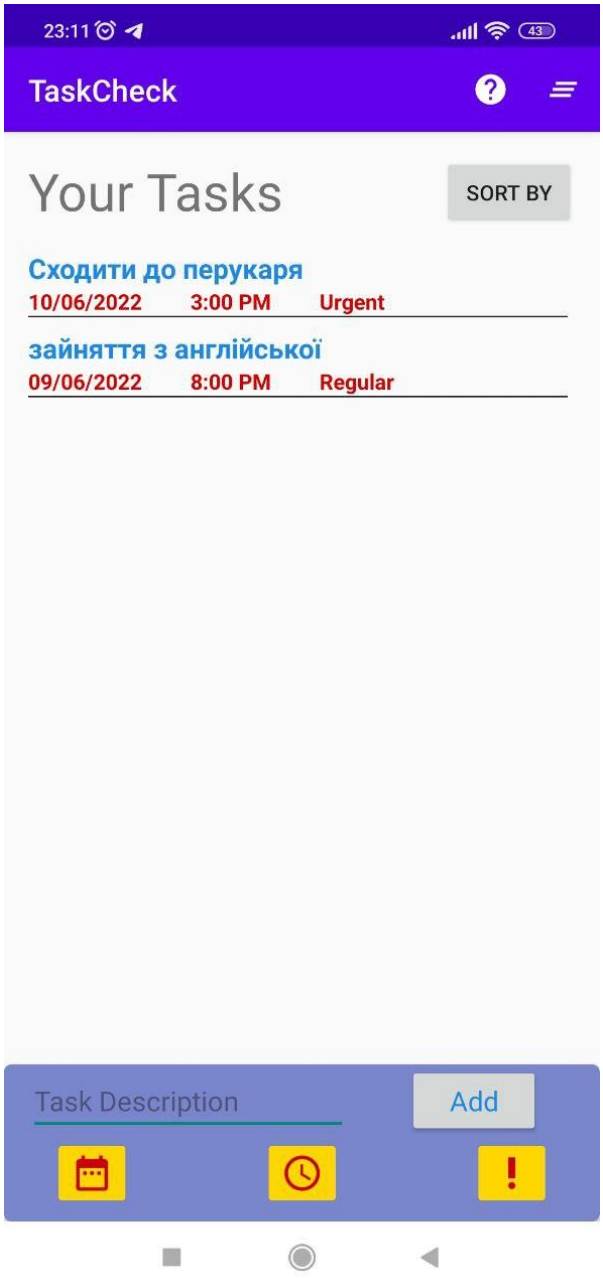


Рисунок 4.3 – Успішне додавання нового завдання

Тож перевіримо валідність функціонування сортування. Для перевірки цього функціоналу потрібно створити декілька test cases, для того щоб покрити різні тестові випадки, та запобігти не валідність роботи додатку (таблиця 4.3).

Таблиця 4.3 – Перевірка Test cases сортування завдань

№	Назва	Кроки	Очікуваний результат	Статус
4	Сортування за датою	1. Відкрити додаток;	Додаток відкривається. Головна сторінка завантажується	Виконано
		2. Натиснути на кнопку «Sort by»;	Відкривається випадний список. У випадному списку присутні «Sort by date» та «Sort by priority»	
		3. Обрати «Sort by date»;	«Sort by date» виділяється. Випадний список закривається. Завдання сортуються відповідно до дати, спочатку ті завдання які потрібно виконати раніше.	

Продовження таблиці 4.3

		4. Повторити крок 2;	Відкривається випадний список. У випадному списку присутні «Sort by date» та «Sort by priority»	
		5. Повторити крок 3.	«Sort by date» виділяється. Випадний список закривається. Завдання сортуються відповідно до дати, спочатку ті завдання які потрібно виконати останніми.	
5	Сортування за приоритетом	1. Відкрити додаток;	Додаток відкривається. Головна сторінка завантажується	Виконано

Продовження таблиці 4.3

		2. Натиснути на кнопку «Sort by»;	Відкривається випадний список. У випадному списку присутні «Sort by date» та «Sort by priority»	
		3. Обрати «Sort by priority»;	«Sort by priority» виділяється. Випадний список закривається. Завдання сортуються відповідно до пріоритету, спочатку ті завдання які потрібно виконати раніше.	
		4. Повторити крок 2;	Відкривається випадний список. У випадному списку присутні «Sort by date» та «Sort by priority»	

Продовження таблиці 4.3

		5. Повторити крок 3.	«Sort by priority» виділяється. Випадний список закривається. Завдання сортуються відповідно до пріоритету, спочатку ті завдання які потрібно виконати останніми.	

Результати виконання тестування зображені на рисунках 4.4 і 4.5.

Однією із основних функцій є «виконання» поставленої задачі.

Задля зручності, ця функція реалізована таким чином, що для того, щоб відмітити завдання як виконане, потрібно натиснути на завдання і утримувати протягом декількох секунд.

4.4 – Повідомлення про помилку при спробі додати завдання без опису

Тож перевіримо зазначений функціонал.

Для цього потрібно виконати два test cases. Один для успішного проходження flow відмітки завдання, а також неуспішного. Test cases із виконанням відображено в таблиці 4.4.

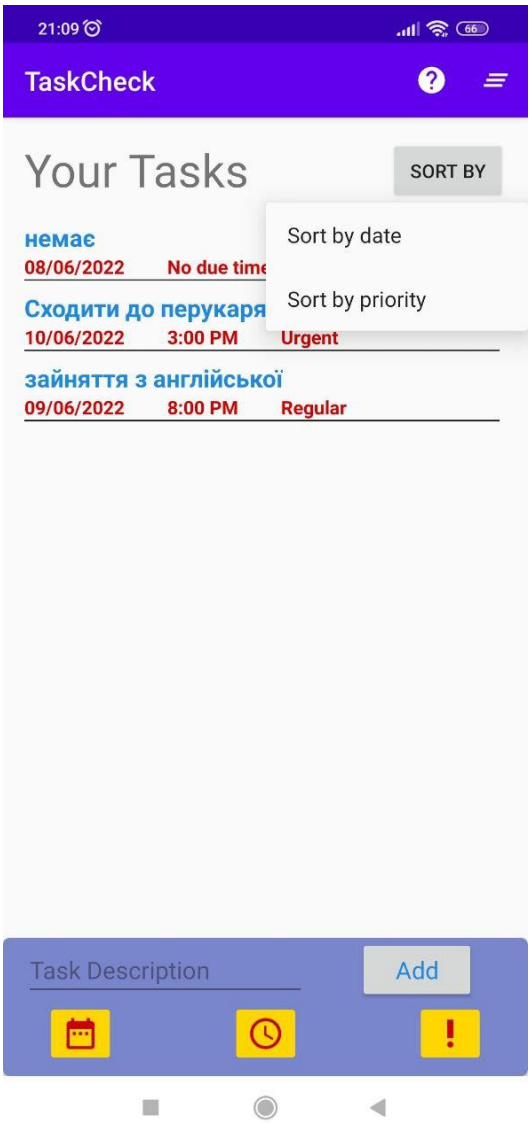


Рисунок 4.5 – Відображення випадного списку

Таблиця 4.4 – Виконання test cases завершення завдання

№	Назва	Кроки	Очікуваний результат	Статус
6	Успішне завершення завдання	1. Відкрити додаток;	Додаток відкривається, Головна сторінка завантажується, список створених завдань завантажується	Виконано
		2. Натиснути на одне із завдань;	Завдання виділяється	

Продовження таблиці 4.4

		3. Утримувати протягом декількох секунд.	Завдання зникає із списку, toast про успішне виконання відображається	
7	Неуспішне виконання завання	1. Відкрити додаток;	Додаток відкривається, Головна сторінка завантажується, список створених завдань завантажується	Виконано
		2. Натиснути на одне із завдань;	Завдання виділяється	

Доволі цікавою функцією є «Виконати все». Ця функція дозволяє користувачу помітити всі завдання, як виконані. Ця функція буде корисною в тому випадку, коли користувач створює собі невеликі завдання, які можна буде потім виконати одним рухом.

Для тестування зазначеного функціоналу було створено test case зображеного в таблиці 4.5.

Таблиця 4.5 – Виконання test case завершення всіх завдань

№	Назва	Кроки	Очікуваний результат	Статус
8	Завершення всіх завдань	1. Відкрити додаток;	Додаток відкривається, Головна сторінка завантажується, список створених завдань завантажується	

Продовження таблиці 4.5

		2. Натиснути на піктограму виконання всіх завдань	Анімація натиску відбувається. Всі завдання видаляються із списку, toast про успішне виконання з’являється	

4.3 Керівництво користувачу

Посібник користувача — це документ, який надається користувачеві, який допомагає безперешкодно використовувати певну систему, продукт чи послугу. Він також відомий як інструкція з експлуатації або посібник користувача. Такі документи охоплюють детальну інформацію щодо операцій, стандартів і рекомендацій, посібників з усунення несправностей, функцій тощо [12].

Посібники користувача зазвичай містять покрокові інструкції, які вказують користувачам на те, як використовувати ваш продукт, і можливе усунення несправностей у випадку, якщо щось піде не так. Він не обов’язково призначений для читання від початку до кінця, він повинен містити зміст та покажчик, щоб допомогти клієнтам знайти розділ, який має відношення до їхньої проблеми [12].

При першому запуску додатку користувача потрібно проінформувати про можливості додатку, тому snackbar інформує користувача, як позначити завдання як виконане. В подальшому, користувач може викликати підказку натиснувши на відповідну кнопку.

Найголовнішою функцією додатку є створення завдань, які користувач буде виконувати. Для цього у нижній частині екрану розташований блок у якому є поле для введення, та кілька pikers. Якщо користувач спробує додати завдання без тексту, то додаток проінформує його про те, що потрібно заповнити поле для успішного створення завдання.

Якщо ж користувач введе текст завдання, то він може добавити його до списку. Для більшої ефективності, створено piker для обирання дати та часу, а також випадний список із пріоритетами.

Після заповнення всіх необхідних даних, завдання з’являється у списку завдань. Після цього, користувач може відмітити завдання як виконане просто натиснувши на нього і зачекавши декілька секунд. Після цього, завдання зникне із списку завдань, та з’явиться повідомлення про успішне виконання завдання.

Також для зручності, є можливість помітити всі завдання у списку, як виконані. Для цього потрібно натиснути на кнопку, яка розташована у верхній частині екрану. Після цього, всі завдання зникнуть із списку завдань, та з’явиться повідомлення про успішне виконання завдань.

4.4 Висновки

В результаті проведення тестування, було створено вісім test cases, як позитивних так і негативних.

Додаток успішно пройшов тестування, в результаті якого дефектів виявлено не було.

Також було створено керівництво користувача.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи, було розроблено мобільний додаток для підвищення тайм менеджменту користувача.

Було проаналізовано сучасну проблему тайм менеджменту людини та зроблено порівняння аналогів, на основі якого вирішено розробити власний мобільний додаток.

Після порівняння, було розроблено блок схеми та розроблено структуру інтерфейсу.

Було проаналізовано переваги та недоліки мови програмування Java та обрано для розробки мобільного додатку.

У контексті виконання бакалаврської дипломної роботи, було вирішені наступні задачі:

- Розроблено простий і зручний в користуванні;
- Розроблено гарний дизайн, який викликає позитивні емоції користувача;
- Розроблено програмний модуль для відображення snackbar;
- Розроблено програмний модуль для додавання завдання;
- Розроблено модуль для помітки завдання як виконане;
- Розроблено модуль для збереження інформації;
- Розроблено програмний модуль для роботи Picker та сортування;
- Розроблено програмний модуль сортування.

Перед розробкою програмних модулів, було створено блок схему алгоритму додавання нових завдань, а також діаграми діяльності та послідовності.

Тестування проведено успішно, що доводить повну працездатність мобільного додатку та відповідність поставленому технічному завданню. Також було створено керівництво користувача.

При оформлені бакалаврської дипломної роботі всі вимоги були дотримані [13].

СПИСОК ЛІТЕРАТУРИ

1. Що таке Jira. [Електронний ресурс] – Режим доступу: https://www.atlassian.com/ru/software/jira/guides/use-cases/what-is-jira-used-for

2. Що таке Проект 365. [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.rgplanner&hl=ru&gl=US

3. Що таке Trello. [Електронний ресурс] – Режим доступу: https://trello.com/tour

4. Романюк О. Н. Веб-дизайн і комп'ютерна графіка / О. Н. Романюк, Д. І. Катєльніков, О. П. Косовець. – Вінниця, 2007. – 142 с

5. Діаграми послідовності. [Електронний ресурс] – https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

6. Романюк О. Н. Організація баз даних і знань. / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: УНІВЕРСУМ – Вінниця. – 2003. – 123 с.

7. Що таке блок схема алгоритму, та для чого її використовують. [Електронний ресурс] – https://ellas-cookies.com/tehnologii/109081-blok-shema-algoritma-programmy-zadachi-elementy-postroenie.html

8. Мова програмування Java. [Електронний ресурс] – https://www.java.com/en/download/help/whatis_java.html

9. Шилдт Г. Java 8. Керівництво для початківців: пер. з англ. М. Вільямс / Г. Шилдт – Санкт-Петербург: Вільямс, 2015. – 712 с.

10. Manual testing. [Електронний ресурс] – https://dou.ua/forums/topic/13389/

11. Certified Tester Foundation Level Release Notes. Version 2018 v3.1.1 – 93 с.

12. Що таке посібник користувача. Для чого його створюють. [Електронний ресурс] – https://document360.com/blog/creating-a-user-manual/

13. Семенов А.О. Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти у Вінницькому національному технічному університеті /Уклад. А.О. Семенов, Л.П. Громова, О.В. Сердюк, Т.В. Макарова, – Вінниця: ВНТУ, 2021. – 68 с.

Додатки

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"__" _____ 2022 р.

Технічне завдання

на бакалаврську дипломну роботу

«Розробка освітнього порталу для вивчення мов програмування»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

к.т.н., доцент кафедри ПЗ Бабюк Н. П

"____"_____2022 р.

Виконала:

студентка гр. 2ПІ-18б Богач К. С.

"____"_____2022 р.

Вінниця – 2022 року

1 Найменування та галузь застосування.

Бакалаврська робота: «Розробка програмного додатку для покращення тайм менеджменту користувача».

Сфера застосування – навчальна(інформаційні технології).

2 Підстава для розробки.

Підставою для розробки бакалаврської дипломної роботи є рішення засідання кафедри програмного забезпечення.

3 Мета та призначення розробки.

Мета виконання бакалаврської дипломної роботи – підвищення тайм менеджменту користувача.

Призначення роботи – розробка програмного продукту для підвищення тайм менеджменту користувача.

4 Вихідні дані для проведення НДР.

Вихідні дані для розробки є індивідуальне завдання на бакалаврську дипломну роботу на розробку програмного продукту для підвищення тайм менеджменту користувача.

5 Технічні вимоги.

Створення завдання із вибором дати часу та пріоритету; архітектура мобільного додатку; помітка завдання як виконане; сформований мобільний додаток.

6 Конструктивні вимоги.

Мобільний додаток має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

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

- пояснювальна записка до бакалаврської дипломної роботи.
- технічне завдання
- лістинг програми.

8 Вимоги до рівня уніфікації та стандартизації.

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9 Стадії та етапи розробки.

№ з/п	Назва етапів дипломної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обґрунтування актуальності розробки додатку та постановка задачі	26.03.22-14.04.22	Вик.
2	Розробка архітектури та алгоритмів роботи додатку, проектування інтерфейсу додатку	15.04.22-25.04.22	Вик.
3	Вибір середовища та мови розробки	26.04.22-12.05.22	Вик.
4	Розробка програмного продукту	13.05.22-30.05.22	Вик.
5	Тестування роботи додатку	31.05.22-03.06.22	Вик.
6	Оформлення матеріалів до захисту БДР	03.06.22-13.06.22	Вик.

10 Порядок контролю та прийняття.

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

Додаток Б – Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Розробка програмного додатку для покращення тайм менеджменту користувача

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент Бабюк Наталя Петрівна

Оригінальність	
Схожість	

Аналіз звіту подібності

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

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

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

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Богач К.С.

Керівник роботи _____ Бабюк Н.П.

Додаток В – Лістинг програми

```
package com.example.android.taskcheck;

import android.annotation.SuppressLint;
import android.content.Context;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.PopupMenu;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.DialogFragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.room.Room;

import com.google.android.material.snackbar.Snackbar;

import java.util.List;

public class MainActivity extends AppCompatActivity implements
PopupMenu.OnMenuItemClickListener {

    // Data fields
```

```
    static List<TaskData> tasks;
    static String dueDate = "none";
    static String dueTime = "none";
    static String taskPriority = "none";
    TasksListAdapater adapter;
    TaskDatabase database;

    /**
     * First method that gets called when the app is launched. All instantiations and inflations here.
     */
    @SuppressLint("WrongThread")
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Set main content view
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialise database access variable. Main Thread Queries allowed as app isn't expected to
        // perform very heavy tasks
        database = Room.databaseBuilder(getApplicationContext(), TaskDatabase.class, "task_data")
                .allowMainThreadQueries()
                .fallbackToDestructiveMigration()
                .build();

        // Fetched data from SharedPref. Indicates whether app has been launched at least one time or not
```

```
        boolean firstAppLaunch = getPreferences(Context.MODE_PRIVATE).getBoolean("display_intro", true);

        // Fetching stored task list from the database
        tasks = database.taskDao().getAll();

        // Deleting existing task list. New one will be saved upon app closure.
        database.clearAllTables();

        // Initialize and bind recyclerView
        adapter = new TasksListAdapater();
        RecyclerView recyclerView = findViewById(R.id.recycler_view_tasks);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(adapter);

        // Displays a snackbar with information on how to mark a task as completed (long click)
        // Displayed only at the first launch
        if (firstAppLaunch) {
            displayIntroSnackbar();
        }
    }

    /**
     * Displays a snackbar to the user when launching the app for the first time or when clicked on
     * the menu button. The snackbar display info on how to mark a task as completed
```

```
     */
    private void displayIntroSnackbar() {
        final Snackbar snackbar = Snackbar.make(findViewById(R.id.parent_view),
R.string.howto_mark_complete, Snackbar.LENGTH_LONG);
        snackbar.setAction("Dismiss", v -> snackbar.dismiss()).show();
    }

    /**
     * Used for saving the data in the app to SharedPreferences before the app closes.
     */
    @Override
    protected void onPause() {
        super.onPause();

        // Indicates that the app has been launched for the first time, so no SnackBar from the next time

getPreferences(Context.MODE_PRIVATE).edit().putBoolean("display_intro", false).apply();

        // Save tasks data to the database
        if (tasks.size() > 0) {
            database.taskDao().insertAll(tasks);
        }
    }

    /**
```

```
    * Adds a task to the current list and notifies recyclerView of dataset change
    */
   public void addTask(View view) {
      EditText editTextTaskDescription = findViewById(R.id.edit_text_task_description);
      String taskDescription = editTextTaskDescription.getText().toString();

      // If non-empty string is received, add it to the task list and notify adapter
      if (taskDescription.length() != 0) {
         tasks.add(new TaskData(0, taskDescription, dueDate, dueTime, taskPriority));
         editTextTaskDescription.setText("");    // Resets Task Input field

         // Resets the optional task attributes
         dueDate = "none";
         dueTime = "none";
         taskPriority = "none";
         adapter.notifyDataSetChanged();
      } else {
         // Errors out with a toast.
         Toast.makeText(this, "Cannot add empty Task!", Toast.LENGTH_SHORT).show();
      }
   }

   /**
    * Inflate the AppBar menu
    *
    * @param menu menu object
```

```
     * @return boolean value indicating success
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    /**
     * Handles AppBar menu option clicks
     *
     * @param item the clicked menu entry
     * @return status indicating success
     */
    @SuppressLint("NonConstantResourceId")
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_action_clear_all: {
                tasks.clear();
                adapter.notifyDataSetChanged();
                Toast.makeText(this, "Task list cleared!",
Toast.LENGTH_SHORT).show();
                return true;
            }
            case R.id.redisplay_intro: {
                displayIntroSnackbar();
                return true;
            }
            default:
```

```
            return super.onOptionsItemSelected(item);
        }
    }

    /**
     * Displays DatePicker for due date selection
     */
    public void showDatePickerDialog(View view) {
        DialogFragment fragment = new DatePickerFragment();
        fragment.show(getSupportFragmentManager(), "datePicker");
    }

    /**
     * Displays TimePicker for due time selection
     */
    public void showTimePickerDialog(View view) {
        DialogFragment fragment = new TimePickerFragment();
        fragment.show(getSupportFragmentManager(), "timePicker");
    }

    /**
     * Instantiates and displays a new Popup menu to set task priority
     */
    public void showPriorityMenu(View view) {
        PopupMenu popup = new PopupMenu(this, view);
        popup.setOnMenuItemClickListener(this);
        getMenuInflater().inflate(R.menu.priority_menu, popup.getMenu());
        popup.show();
    }
```

```
    /**
     * Instantiates and displays a Popup menu to sort lists by either due time or priority
     */
    public void showSortByPopupMenu(View view) {
        PopupMenu popupMenu = new PopupMenu(this, view);
        popupMenu.setOnMenuItemClickListener(this);
        getMenuInflater().inflate(R.menu.sort_menu, popupMenu.getMenu());
        popupMenu.show();
    }

    /**
     * Handles Popup menu option click events. Applicable for both sorting and task priority
     * setter menus
     *
     * @return status code indicating whether event has been handled or not
     */
    @SuppressLint("NonConstantResourceId")
    @Override
    public boolean onMenuItemClick(MenuItem menuItem) {
        switch (menuItem.getItemId()) {
            case R.id.action_priority_urgent:
                taskPriority = "Urgent";
                return true;
            case R.id.action_priority_rushed:
                taskPriority = "Rushed";
                return true;
            case R.id.action_priority_regular:
                taskPriority = "Regular";
```

```
            return true;
        case R.id.action_sort_by_date:
            Helper.sortByDate(tasks);
            adapter.notifyDataSetChanged();
            return true;
        case R.id.action_sort_by_priority:
            Helper.sortByPriority(tasks);
            adapter.notifyDataSetChanged();
        default:
            return false;
      }
   }
}

package com.example.android.taskcheck;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class TasksListAdapater extends
RecyclerView.Adapter<TasksListAdapater.ViewHolder> {

   /**
    * Boilerplate for creating views for RecyclerView
```

```
     */
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View                                view                                =
LayoutInflater.from(parent.getContext()).inflate(R.layout.single_task_view,
parent, false);
        return new ViewHolder(view);
    }

    /**
     * Boilerplate for runtime binding in RecyclerView
     */
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int
position) {

holder.getTextView().setText(MainActivity.tasks.get(position).taskDescript
ion);

        // This obviously isn't a very good pattern, but I couldn't think of any
other patterns
        // Using another function would require passing all the data like holder
and position to it,
        // ultimately defeating the purpose.
        if (MainActivity.tasks.get(position).dueDate.equals("none")) {
            holder.getTaskDueDateTextView().setText(R.string.no_due_date);
        } else {
            // Date is formatted to DD/MM/YYYY before being set
```

```
holder.getTaskDueDateTextView().setText(Helper.reverseDateString(Main
Activity.tasks.get(position).dueDate));
        }

        if (MainActivity.tasks.get(position).dueTime.equals("none")) {
            holder.getTaskDueTimeTextView().setText(R.string.no_due_time);
        } else {

holder.getTaskDueTimeTextView().setText(MainActivity.tasks.get(positio
n).dueTime);
        }

        if (MainActivity.tasks.get(position).taskPriority.equals("none")) {

holder.getTaskPriority().setText(R.string.no_task_priority_assigned);
        } else {

holder.getTaskPriority().setText(MainActivity.tasks.get(position).taskPriori
ty);
        }

        // Long click deletes the task from the list and displays a toast
        holder.itemView.setOnLongClickListener(view -> {
            MainActivity.tasks.remove(position);
            notifyDataSetChanged();
            Toast.makeText(holder.itemView.getContext(), "Task completed!",
Toast.LENGTH_SHORT).show();
            return true;
        });
```

```
    }

    @Override
    public int getItemCount() {
        return MainActivity.tasks.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {

        private final TextView taskDescriptionTextView;
        private final TextView taskDueDateTextView;
        private final TextView taskDueTimeTextView;
        private final TextView taskPriority;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);

            taskDescriptionTextView                                    =
itemView.findViewById(R.id.text_view_single_task_description);
            taskDueTimeTextView                                        =
itemView.findViewById(R.id.task_due_time);
            taskDueDateTextView                                        =
itemView.findViewById(R.id.task_due_date);
            taskPriority = itemView.findViewById(R.id.task_priority);
        }

        public TextView getTextView() {
            return taskDescriptionTextView;
        }
```

```
        public TextView getTaskDueDateTextView() {
            return taskDueDateTextView;
        }

        public TextView getTaskDueTimeTextView() {
            return taskDueTimeTextView;
        }

        public TextView getTaskPriority() {
            return taskPriority;
        }
    }
}
```

Додаток Г –Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ДЛЯ ПОКРАЩЕННЯ ТАЙМ МЕНЕДЖМЕНТУ КОРИСТУВАЧА

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська дипломна робота

На тему: Розробка програмного додатку для покращення тайм-менеджменту користувача

Виконала:
студентка групи 2ПІ-18Б Богач К.С

Науковий керівник:
к.т.н., доцент кафедри ПЗ Бабюк Н.П.

Мета, дослідження та предмет роботи

- **Метою бакалаврської дипломної роботи** є підвищення якості планування користувача шляхом розробки та користування додатком для покращення якості тайм менеджменту користувача.
- **Предмет дослідження** - алгоритми запису інформації та відображення її у мобільному додатку.
- **Об'єкт дослідження** - процес отримання, обробки, збереження та відображення даних у веб-сервісі.

Актуальність теми

Можливість планувати свій час, для покращення продуктивності

Візуалізація завдань. Структурування задач

Зменшення ризику забути важливі справи до мінімального рівня

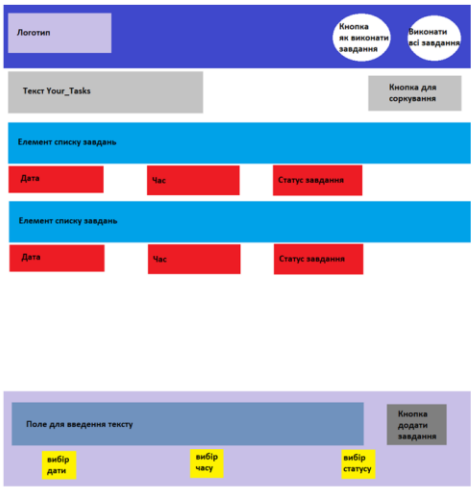
Візуалізація завдань

Критерій	Trello	Проект 365	Mobile for Jira	TaskCheck
Орієнтований на створення нотаток	-	+	-	+
Простий у використанні	-	+/-	-	+
Повністю безкоштовний	-	-	-	+
Відсутність вбудованої реклами	+	-	+	+
Швидкість підготовки до початку роботи	-	-	-	+
Мобільний додаток	+	+	+	+

Перевага мови програмування Java

- У Java простой та зрозумілий синтаксис;
- Об’єктно-оріїнтованість мови програмування;
- Велика кількість бібліотек із відкритим кодом;
- Велика спільнота Java розробників;
- Кросплатформленсть. Можливість розробляти мобільні додатки

Розробка інтерфейсу



Структура сторінки створення нотатків

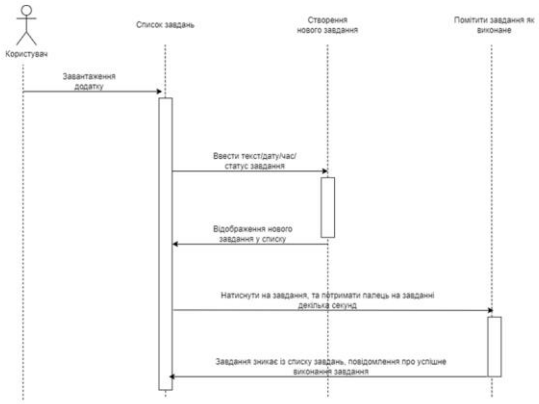
Сторінка складається з таких компонентів:

1. Верхня панель на якій розміщено дві кнопки: підказка, виконання всіх завдань

2. Блок із списком створених завдань на якій розміщено: кнопка для сортування та елементи списку завдань

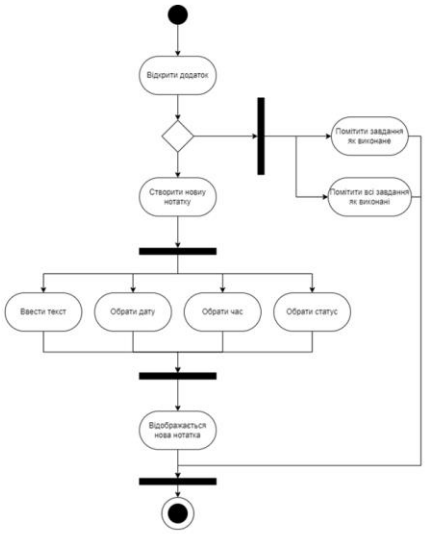
3. Нижня паналь на якій розміщено: поле для введення, кнопка для додавання завдання, picker дати, picker часу та випадаючий список приоритету

Розробка алгоритмів роботи додатку



Діаграми послідовності - це час, вони візуально показують порядок взаємодії, використовуючи вертикальну вісь діаграми, щоб відобразити час, які повідомлення надсилаються і коли . Зважаючи на те, що час - ключовий аспект у мобільному додатку тайм менеджменту користувача, розробка діаграми послідовності може візуально зобразити роботу головної функції додатку - створення завдань та цілей

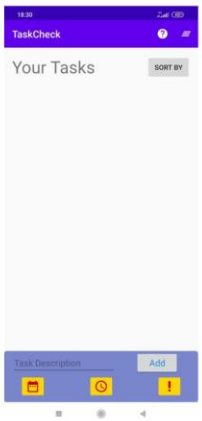
Розробка алгоритмів роботи додатку



Діаграми діяльності описують, як діяльність координується для надання послуги, яка може бути на різних рівнях абстракції. Як правило, подія повинна бути досягнута деякими операціями, особливо якщо операція призначена для досягнення ряду різних речей, які потребують координації, або як події в одному випадку використання пов'язані одна з одною. Тому зважаючи на це, було прийнято рішення розробити діаграму діяльності, щоб описати роботу додатку, та створити більш чітке розуміння, яким має бути створюваний додаток.

Тестування

1



2

3

В результаті проведення тестування, було створено вісім test cases, як позитивних так і негативних. Додаток успішно пройшов тестування, в результаті якого дефектів виявлено не було. Також було створено керівництво користувача.

Вимоги до користувача

Для коректної роботи мобільного додатку потрібно мати:

Система	**Параметри**
Девайс	Смартфон
Операційна система	Android
Версія	6
Вільне місце	20мб

Результати розробки

У контексті виконання бакалаврської дипломної роботи, було вирішені наступні задачі:

- Розроблено простий і зручний в користуванні;
- Розроблено гарний дизайн, який викликає позитивні емоції користувача;
- Розроблено програмний модуль для відображення snackbar;
- Розроблено програмний модуль для додавання завдання;
- Розроблено модуль для помітки завдання як виконане;
- Розроблено модуль для збереження інформації;
- Розроблено програмний модуль для роботи Picker та сортування;
- Розроблено програмний модуль сортування;

Дякую за увагу!