Галицький фаховий коледж імені В’ячеслава Чорновола
відділення комп’ютерних технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділенням
комп’ютерних технологій
Наталія СТЕФУРАК /_____/
підпис
«___» _____ 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи
освітньо-професійного ступеня «фаховий молодший бакалавр»
зі спеціальності 122 «Комп’ютерні науки»
на тему: «Мобільний додаток «Motivator» для контролю використання власного часу»

Студент групи КН-41 Володимир ПОРЦІНА _____
(підпис)

Керівник роботи Олександра ЧУБЕЙ _____
(підпис)

Консультанти:
з техніко-економічного
обґрунтування Любов МЕЛЕНЧУК _____
(підпис)

нормоконтролер Наталія КУЛЬЧИНСЬКА _____
(підпис)

Тернопіль – 2024

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних технологій

циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ

Завідувач відділенням

комп’ютерних технологій

Наталія СТЕФУРАК /_____/

підпис

«___» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу

на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»

студенту Порціні Володимиру Олеговичу

(прізвище, ім’я та по-батькові студента)

1. Тема роботи: Мобільний додаток «Motivator» для контролю використання власного часу

затверджено наказом по коледжу від “27” листопада 2023 р., №_____

2. Термін здачі студентом завершеної роботи “___” _____ 202_ р.
3. Вихідні дані до роботи Дослідження додатків для контролю використання особистого часу
4. Перелік питань, які повинні бути розроблені в роботі:

а) основна частина: Аналіз предметної області та постановка задання, Проектування системи, Реалізація та тестування органайзеру

б) техпічно економічне обгрунтування: аналіз ринку, розрахунок , обгрунтування необхідності розробки.

5. Перелік графічного матеріалу: діаграма станів, діаграма варіантів використання, діаграма класів
6. Консультанти проєкту: Меленчук Л. І.

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
З техніко-економічного обґрунтування	Меленчук Л. І. (вчена ступінь, звання П.І.Б. консультанта)		

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми кваліфікаційної роботи	23.11.2023	01.12.2023
2.	Аналіз програмних рішень	01.04.2024	07.04.2024
3.	Опрацювання теоретичних матеріалів, написання розділу роботи	08.04.2024	09.04.2024
4.	Формалізація вимог. Аналіз технології реалізації, Написання розділу роботи	09.04.2024	16.04.2024
5.	Проєктування та реалізація графічного інтерфейсу реалізація основних функцій програми	20.04.2024	10.05.2024
6.	Тестування програмного засобу	10.05.2024	16.05.2024
7.	Створення відповідного розділу роботи	16.05.2024	22.05.2024
8.	Обгрунтування вартості роботи	05.06.2024	10.06.2024
9.	Оформлення пояснювальної записки	14.06.2024	14.06.2024
10.	Попередній захист кваліфікаційної роботи	17.06.2024	17.06.2024
11.	Підготовка до захисту та виправлення помилок	17.06.2024	24.06.2024
12.	Захист кваліфікаційної роботи	26.06.2024	26.06.2024

Дата видачі “___” _____ 202_ р. Керівник _____/ Чубей О.

Завдання прийняв до виконання _____/ Порціна В.

Реферат

Кваліфікаційна робота. Порціна Володимир Олегович, Галицький фаховий коледж імені В’ячеслава Чорновола відділення комп’ютерних технологій. Спеціальність 122 «Комп’ютерні науки», 2024 розробка мобільного додатку для контролю використання особистого часу на мові програмування Java з використанням Android Studio. 59 сторінок, 28 малюнків, 2 додатки.

Об’єкт дослідження – органайзери для контролю використання власного часу.

Метою кваліфікаційної роботи є розробка android додатку для контролю використання особистого часу.

Для реалізації даного додатку було використано програмне забезпечення Android Studio для написання та перегляду результатів коду. Мову програмування було обрано java, база даних використовувалася SQLite.

ANDROID ДОДАТОК ДЛЯ КОНТРОЛЮ ВИКОРИСТАННЯ ВЛАСНОГО ЧАСУ, JAVA, SQLITE, ANDROID STUDIO

Abstract

Qualification work. Volodymyr Olegovich Portsina, Vyacheslav Chornovol Halych Vocational College, Department of Computer Technologies. Specialty 122 "Computer Science", 2024 development of a mobile application for monitoring the use of personal time in the Java programming language using Android Studio. 59 pages, 28 figures, 2 appendices.

The object of the research is organizers to control the use of one's own time.

The method of qualifying work is the development of an android application for controlling the use of personal time.

To implement this application, Android Studio software was used to write and view the code results. Java was chosen as the programming language, and SQLite was used as the database.

ANDROID APP FOR MONITORING THE USE OF OWN TIME, JAVA, SQLITE, ANDROID STUDIO

Зміст

					КР. КН 24.560.13.000 ПЗ		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Порціна В. О.*				*Літ.*	*Арк.*	*Аркушів*
Перев.		*Чубей О. О.*						
Рецензет.		*Сиротюк О.Б.*				ГФКімВЧ.ВКТ.ЦК ІтаКД		
Н. Контр.		*Кульчинська Н.З.*				Гр. КН – 41		
Зав. від.		*Стефурак Н.А.*						

ВСТУП

В сучасному суспільстві, що постійно розвивається та прискорюється, управління і контроль власного часу є досить значним фактором для досягнення власних цілей та мрій. Через це було прийняте рішення, проаналізувати та створити органайзер власного часу, заради збільшення продуктивності, вчасного виконання задач та досягненям балансу між професійним та особистим життям.

Головною метою створення програмного забезпечення контролю часу є надання користувачам інструментів для ефективного планування, відстеження та управління їхнім часом.

В створені проєкту програмного забезпечення контролю часу ставить за мету розробити інтуїтивний, функціональний та ефективний інструмент, який буде відповідати потребам різних користувачів у керуванні їхнім часом.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАННЯ

1.1 Актуальність використання мобільних додатків

Мобільний додаток — програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних застосунків встановлені на самому пристрої або можуть бути завантажені на нього з онлайн-магазинів мобільних застосунків, таких як App Store, Google Play, Windows Phone Store та інших, безкоштовно або за плату.

Спочатку мобільні застосунки пропонувалися як інструменти для контролю та оперування загальних потоків інформації, включаючи електронну пошту, календар, контакти, фондовий ринок та інформацію про погоду. Згодом, попит та наявність інструментів для розробників зумовили швидке поширення застосунків таких як GPS, мобільні ігри, служби позіціонування, відстеження замовлень та квитків, тощо. Як і у випадку з іншим програмним забезпеченням, вибухова кількість та різноманіття мобільних застосунків призвела до виникнення великої кількості пізнавальних ресурсів про них — з відгуками, рекомендаціями та оглядами, а саме: блоги, журнали та спеціальні служби виявлення застосунків в Інтернеті.

У 2014 році державні регуляційні служби України почали створювати та регулювати мобільні застосунки, зокрема ті, які мають відношення до медичної індустрії. Деякі компанії навіть почали пропонувати застосунки, як альтернативний метод надання інформації, на противагу офіційним вебсайтам. Так, наприклад, у 2017 році більш ніж 60 % трафіку на сайти українських мікрофінансових компаній було отримано з мобільних пристроїв.

1.2 Актуальність використання органайзеру.

Органайзер — невелика книга, що служить для організації персональної інформації, такої як важливі події, контакти та ін. Сучасні органайзери існують у вигляді програм для комп'ютерів та мобільних телефонів. Органайзер є

персональним інструментом і може містити сторінки з корисною інформацією, наприклад, адреси, телефоні номери, тощо.

Наприкинці 20-го століття органайзери з паперу почал замінюватися електроними пристроями такими як, онлайн-органайзери, програмне забезпечення для керування особистою інформацією. Значно цей процес прискорився з появою, комп’ютерів, планшетів, телефонів, smart-годинників та великою кількістю програмних забезпечень.

Електронний органайзер - електронний пристрій для планування завдань і зберігання важливої інформації. Являє собою невеликий комп'ютер розміром з калькулятор з автономним живленням, часто з вбудованим щоденником та іншими функціями, як-от адресна книга і календар, що замінюють персональні органайзери на паперовому носії. Зазвичай він має невелику буквено-цифрову клавіатуру і рідкокристалічний екран з одним, двома або трьома рядками. Електронний щоденник або органайзер був винайдений індійським бізнесменом Сатьяном Пітрода в 1975 році, який вважається одним із піонерів портативних комп'ютерів завдяки винаходу електронного щоденника.

Вони були дуже популярні, особливо серед бізнесменів у 1990-х роках, але поступово вийшли з ужитку через появу кишенькових комп'ютерів у 1990-х, персональних цифрових помічників у 2000-х і смартфонів у 2010-х, кожен з яких має ширший набір функцій.

1.3 Аналіз існуючих рішень

Існує безліч безліч існуючих рішень які дозволяють ефективно розпроділяти та контролювати власний час. Для аналізу було вибрано декілька популярних серед користувачів програмних забезпечень:

- Google Календар.
- Toggl.
- Forest.

Google Calendar — безкоштовний вебзастосунок для тайм-менеджменту розроблений Google. Став доступним 13 квітня 2006, і вийшов зі стадії бета в липні 2009. Меню та функціонал Google Календар зображено на рисунку 1.1

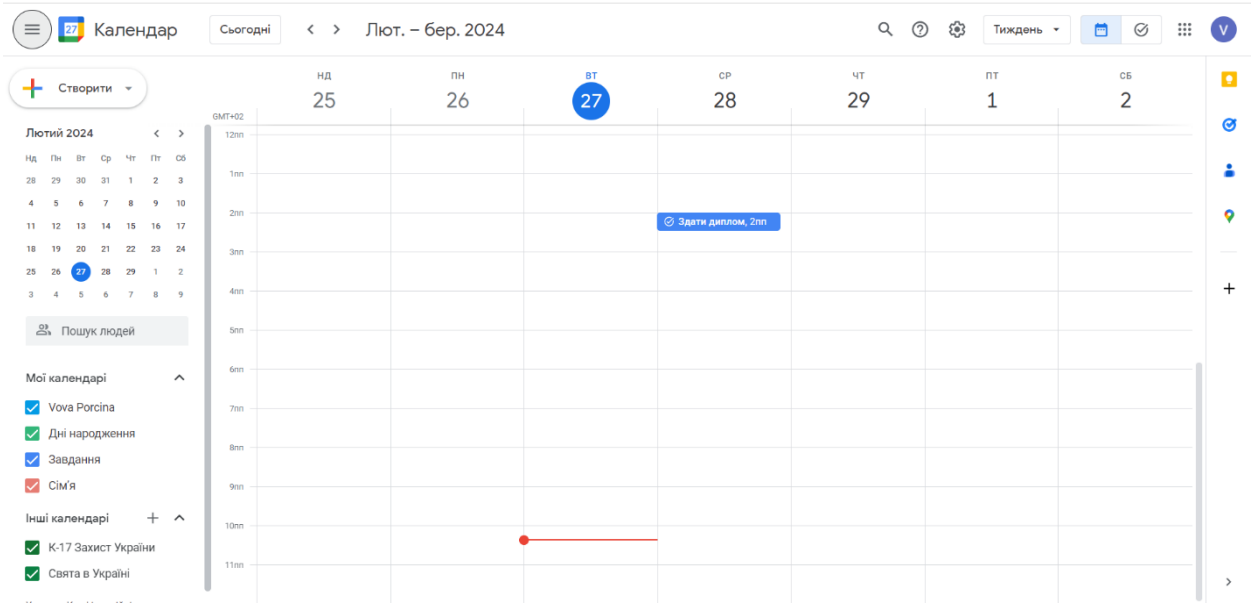

Рисунок 1.1 – Меню та функціонал Google Календар

Загалом Google Календар є досить хорошим інструментом для планування власного часу. Він доступний безкоштовно і є одним з інструментів створених компанією Google що робить його популярним серед користувачів всього світу.

До переваг даного програмного забезпечення можна віднести наступне:

– доступність: Google Календар доступний для користувачів абсолютно безкоштовно та інтегрований у багато інших сервісів Google.

– Спільний доступ: Google Календар дозволяє ділитися власним календарем з користувачами та надавати різні рівні доступу, що є корисним для групового планування подій і зустрічей.

– Багатоплатформені: Google Календар доступний не тільки на комп'ютерах а також на смартфонах для платформ Android та iOS, що дозволяє переглядати та оновлювати календар в любий момент.

Недоліки Google Календар :

– Обмежені функції: У безкоштовній версії Google Календаря для бізнесу обмежена кількість функцій порівняно з платною версією.

– Маленький функціонал: Порівняно з іншими програмними забезпеченнями функціонал Google Календар є досить маленьким і підійде для користувачів, які шукають базовий і простий у використанні календар.

– Необхідність інтернет-з'єднання: Google Календар підтримується хмарним сервісом, тому для доступу до нього потрібне інтернет-з'єднання. Це може бути не зручно в умовах відсутності інтернету.

В цілому Google Календар є досить хорошим сервісом для невибагливих користувачів але він не є надто ефективним для керування завданнями порівняно з іншими спеціалізованими інструментами.

Toggl — це застосунок для відстеження часу. Час можна відстежувати за допомогою кнопки старт/стоп. Toggl був створений у 2006 році. Спочатку сервіс був розроблений для внутрішнього використання. Співзасновники були розробниками програмного забезпечення і намагалися з'ясувати, скільки часу вони витратили на кожного з клієнтів. Меню та функціонал Toggl зображено на Рисунку 1.2.

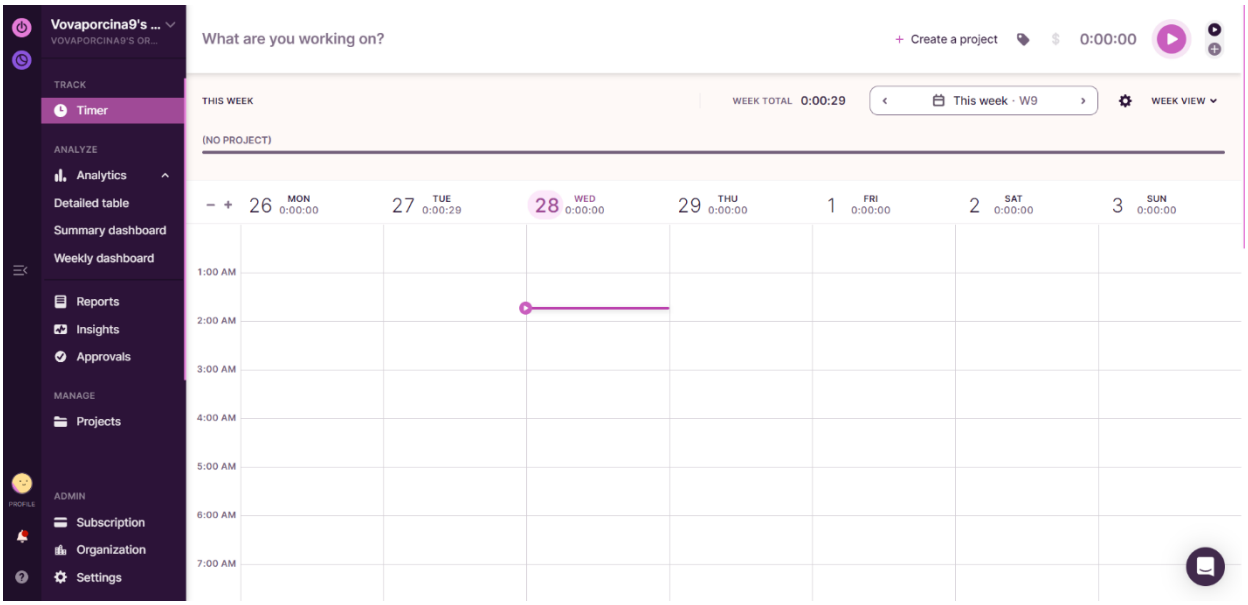

Рисунок 1.2 – Меню та функціонал Toggl

Загалом даний інструмент більше підійде для працівників та бізнесів щоб допомогти зберегти точний облік робочого часу та оптимізувати продуктивність.

Серед переваг Toggl можна виділити:

– Простота інтерфейсу: Toggl має інтуїтивний інтерфейс, що дозволяє легко користуватися як початківцям так і досвідченим користувачам.

– Звітність: Toggle надає точні звіти про використання часу, що допоможе провести статистику власної продуктивності.

– Мобільні застосунки: Існують додатки Toggl для Android та iOS, що дозволяють відстежувати час виконання завдання в любий момент.

Серед недоліків Toggl:

– Обмежена безкоштовна версія: Безкоштовна версія Toggle має значні обмеження в кількості проектів та функціоналі що може відштовхувати користувачів.

– Постійна необхідність відстежувати час: Під час виконання завдання користувач може забути зупинити відстеження часу та потім повернутися до роботи, дані моменти призводять до не точної статистики виконання завдання та використання часу.

– Необхідність інтернет з’єднання: Використання Toggl вимагає стабільного інтернету для відстеження часу, що може негативно вплинути на роботу програми в умовах поганого з’єднання.

Загалом Toggl досить потужний інструмент для контролю власного часу з багатьма функціями але може не підходити багатьом користувачам через свої обмеження та дороговартність підписку.

Forest - це програма для підвищення продуктивності та зменшення відволікання від виконання завдання на смартфони чи інші пристрої. Під час використання Forest, користувачі "вирощують" віртуальні дерева. Якщо вони залишають додаток, щоб перевірити соціальні мережі чи інші відволікаючі додатки, їх віртуальне дерево засихає.

Головне меню програмного забезпечення Forest представлено на Рисунку 1.3.

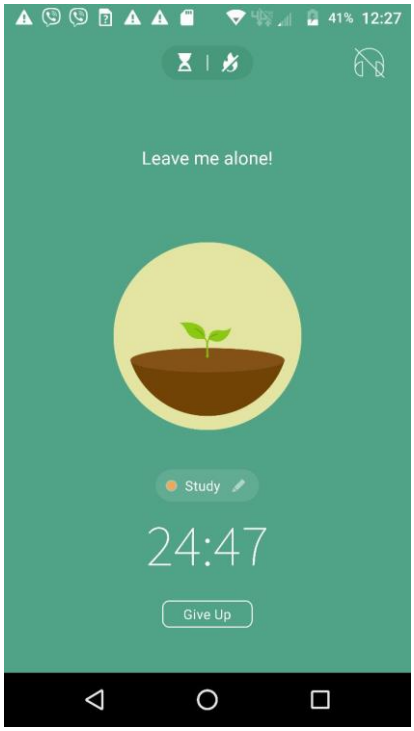

Рисунок 1.3 – Головне меню програми Forest

Forest досить простий та цікавий інструмент який допоможе користувачу навчитися правильно та ефективно використовувати свій час не відволікаючись від роботи на соціальні подразники по типу соціальних мереж.

Серед плюсів даного програмного забезпечення можу виділити:

– Гейміфікація продуктивності: Forest використовує ігрову механіку для стимулювання продуктивності, що робить процес керування часом цікавішим і захоплюючим.

– Блокування відволікань: Функція блокування відволікаючих додатків допомагає зберегти концентрацію під час робочих сесій.

– Доступність: Forest доступний як мобільний додаток для різних платформ, що дозволяє користувачам використовувати його на різних пристроях та в різних ситуаціях.

Недоліки Forest :

– Синхронізація із календарями: Forest може не мати повної інтеграції з календарними програмами, що може бути недоліком для тих, хто хоче планувати час на основі календаря.

– Недостатні можливості: У порівнянні з іншими програмами для управління часом, відчуваєтися обмеженість функцій та можливостей в додатку Forest.

Загалом Forest є досить корисний але через обмеженість його функціоналу а зокрема тільки відстежування часу, дане програмне забезпечення підійде не для кожного користувача.

1.4 Постановка задачі

Задачою даного проекту є проєктування, розробка мобільного застосунку, призначення якого є зручне створення задач чи проектів та відстеження використання часу.

Проаналізувавши готові проекти схожої тематики, були виділені основні, вимоги для успішного виконання проекту.

Завданнями при створенні мобільного органайзера має бути:

– реалізувати зручний інтерфейс;
– не перевантажувати застосунок не потрібним функціоналом;
– надавати можливість для перегляду статистики;
– геймифікація;

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Аналіз варіантів використання органайзера

Діаграма варіантів використання – це представлення взаємодії користувача з системою, що демонструє зв’язок між користувачем та різними варіантами використання, у яких бере участь користувач.

Діаграма варіантів використання здатна розрізняти різні типи користувачів системи та різні варіанти використання та часто супроводжується іншими типами діаграм.

У той час як сам варіант використання може детально вивчити будь-яку можливість, діаграма варіанта використання здатна допомогти надати огляд системи більш високому рівні. Вже було сказано, що «корисні моделі це принципи вашої системи».

Завдяки спрощеному характеру, схеми використання є гарним засобом комунікації для зацікавлених сторін.

Мета діаграми використання – показати динамічний аспект системи. Додаткові схеми та документація можуть використовуватись для забезпечення повного функціонального та технічного розуміння системи. Вони забезпечують спрощене та графічне уявлення того, що насправді має робити система.

Елементи схеми є:

а) рамки системи – прямокутник з назвою на верху та еліпсами (прецедентами) усередині;

б) користувач – стилізований людський персонаж, що позначає набір користувацьких ролей (людина, зовнішня сутність, клас, інша система), взаємодіє з будь-якої сутністю. Актори не можуть взаємодіяти між собою (через відключення обробки/дослідження звітів);

в) прецедент – позначений еліпс, що представляє систематичне виконувану дію (може включати можливі варіанти), що призводить до результатів, що спостерігаються дійовими особами. Заголовок може бути або ім'ям, або описом (з точки

зору дійової особи) того, що система робить (а не як). Під час сценарію користувачі систематично обмінюються повідомленнями. Декілька різних сценаріїв можуть бути зв’язані з одним прецендентом. Діаграма варіантів використання, що описує можливі дії користувача в системі зображена на рисунку 2.1.

Рисунок 2.1 – Діаграма варіантів використання

Дана діаграма в повній мірі відображає можливості використання функцій органайзера.

2.2 Проектування внутрішньої будови системи

У розробці програмного забезпечення діаграма класів уніфікованої мови моделювання (UML) є тип діаграми статичної структури, яка описує структуру системи, зображаючи класи системи, їх атрибути, операції та відносини між об'єктами. Основним будівельним елементом об’єктно-орієнтованого моделювання є діаграма класів.

Діаграма класів - це інструмент моделювання програмного забезпечення, який відображає структуру системи у вигляді класів та їх взаємозв'язків. У діаграмі показуються класи програми, їх атрибути та методи, а також взаємодії між класами, такі як асоціація, агрегація, композиція та

успадкування. Діаграма класів допомагає розробникам розуміти структуру системи та легше керувати розвитком програмного забезпечення.

На схемі класи подаються вікнами, що містять три відсіки:

– Верхній відсік містить назву класу. Надруковано жирним шрифтом і по центру, перша заголовна літера.

– Середній відсік містить атрибути класу. Вирівнюються вони по лівому краю, перша буква була.

– У нижньому відсіку містяться операції, які клас може виконувати.

Розглянемо детальніше основні взаємозв'язки між класами в об'єктно-орієнтованому програмуванні.

Асоціація є із фундаментальним взаємозв'язком в об'єктно-орієнтованому програмуванні, який вказує на взаємодію між двома класами. Вона описує, як один клас використовує або залежить від іншого.

Асоціації поділяються на чотири типи: односпрямована, двонаправлена, агрегаційна та рефлексивна. Найбільш поширеними є двонаправлені та односпрямовані асоціації.

Агрегація є одним з типів асоціацій між класами в об'єктно-орієнтованому програмуванні, що вказує на те, що один клас є частиною іншого, але може існувати самостійно від нього.

У контексті агрегації, об'єкт одного класу є "частиною" або "складовою" іншого класу, але він не пов'язаний з життєвим циклом останнього. Це означає, що коли видаляється батьківський об'єкт, "дитячі" об'єкти можуть залишитися незмінними і існувати далі.У UML він графічно представлений порожнистим ромбом на вміщуючому класі, що пов’язує з вміщеним класом.

Композиція є одним з видів асоціацій між класами в об'єктно-орієнтованому програмуванні, що вказує на те, що один клас є частиною іншого класу і не може існувати без нього.

У контексті композиції, об'єкт одного класу є "складовою" або "частиною" іншого класу, і його життєвий цикл пов'язаний з життєвим циклом батьківського класу. Це означає, що коли батьківський об'єкт видаляється або перестає існувати, всі його "дитячі" об'єкти також автоматично видаляються.

Успадкування є ключовим концептом в об'єктно-орієнтованому програмуванні, який дозволяє одному класу підкласу або нащадку успадковувати властивості та методи іншого класу базового класу або батька.

Цей механізм дозволяє створювати ієрархію класів, де класи-нащадки можуть розширювати або змінювати функціональність, що вже присутня у базовому класі, а також додавати нові властивості та методи. Наслідування використовується для реалізації відношень "є-один" IS-A, де клас-нащадок є більш конкретною версією базового класу.

Успадкування, композиція та агрегація - це ключові концепти, що допомагають в створенні структурованих та гнучких програм. Діаграма класів допомагає візуалізувати взаємозв'язки між класами, а також структуру та функціональність.

Діаграма класів є важливим інструментом для проектування програмного забезпечення, оскільки створення діаграми дозволяє розробникам візуалізувати структуру програми.

На рисунку 2.2 зображено діаграму класів мобільного додатку.

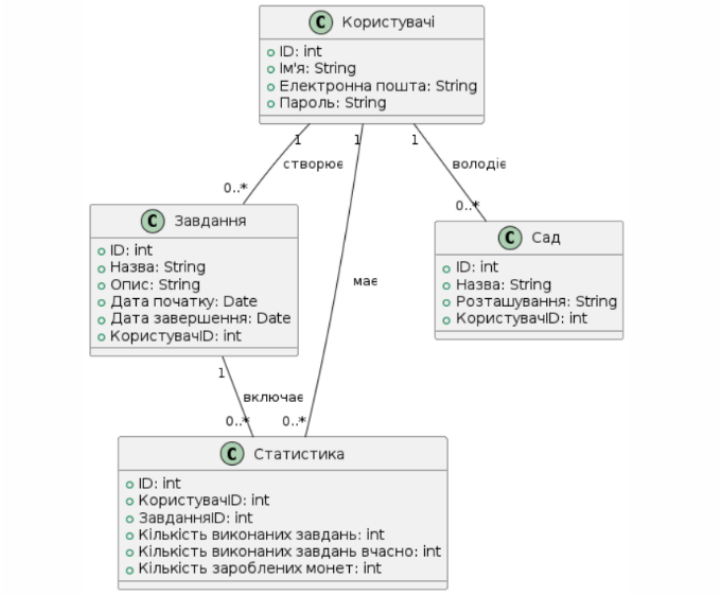

Рисунок 2.2 – Діаграма класів

Дана діаграма класів моделює структуру мобільного додатку для контролю використання власного часу. На діаграмі представлено п'ять основних класів, а також взаємозв'язки між класами.

Загальна структура діаграми допомагає зрозуміти, як класи взаємодіють між собою в мобільному додатку та як виконуються різні функції для забезпечення коректної роботи.

2.3 Діаграма станів

Діаграма станів є однією з ключових UML-діаграм, яка моделює стани об'єкту або системи та переходи між цими станами. Вона дозволяє візуалізувати поведінку системи у відповідь на зовнішні події або внутрішні умови.

Основна ідея діаграми станів полягає в тому, щоб показати всі можливі стани, в яких може перебувати об'єкт або система, а також переходи між цими станами відповідно до певних умов. Кожен стан представлений у вигляді прямокутника зі своїм ім'ям, а переходи між станами відображаються стрілками.

Діаграма станів особливо корисна для моделювання систем, які мають складну логіку переходів між станами або для опису реакції системи на різні події. Вона допомагає розібратися в логіці системи та визначити потрібні дії для кожного стану та переходу на рисунку 2.3 зображену діаграму станів для даного додатку.

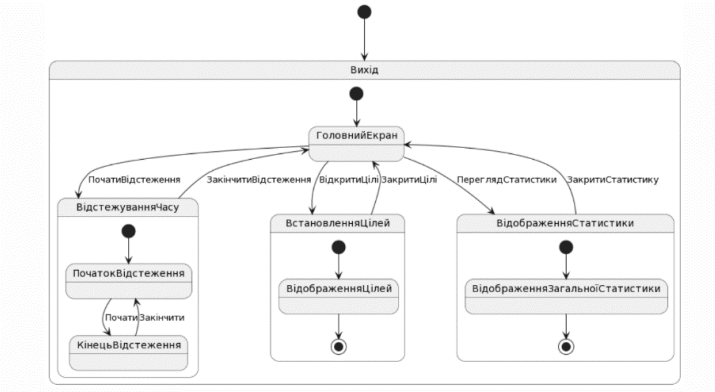

Рисунок 2.3 – Діаграма станів

2.4 Проектування інтерфейсу

Кожна програма створюється з урахуванням великої кількості користувачів, яким потрібна програма для спілкування з ними у зручний і зрозумілий спосіб. Ці інструменти є інтерфейсом користувача. За допомогою інтерфейсу користувач керує програмою, отримує повідомлення від програми та відповідає на її запити тощо. При проектуванні інтерфейсу вихідним рішенням є базові критерії вибору типу елементів управління інтерфейсом, які повинні враховувати специфіку відповідної предметної області.

При проектуванні інтерфейсу користувача необхідно враховувати характеристики передбачуваних кінцевих користувачів програмного забезпечення, що розробляється. Специфікація типу інтерфейсу користувача визначає його синтаксис.

Очевидно, що одним із факторів, що впливає на ефективність використання програмного забезпечення, є простота використання інтерфейсу.Графічний користувацький інтерфейс – це тип інтерфейсу користувача, в якому елементи представлені користувачеві на екрані у вигляді зображень.

Дизайн графічного інтерфейсу є дуже важливою частиною розробки програмного забезпечення. Цей етап покликаний зробити програму простою та зручною у використанні, і від цього також залежить подальша взаємодія користувача з програмою. Якщо цьому етапу не приділити належної уваги, то репутація програми, навіть з відмінною функціональністю, почне падати, як тільки користувачі вперше з нею познайомляться.

Android Studio надає програмістам набір візуальних компонентів для створення графічних інтерфейсів користувача. Вони насправді нічим не відрізняються від тих, що використовуються в інших програмах, які використовують графічний інтерфейс. Тому розробникам, навіть новачкам, неважко почати створювати інтерфейс користувача.

Наявність лише одного робочого екрана зазвичай використовується лише в простих програмах, яким не потрібно виконувати багато функцій. Однак у нашому випадку неможливо використовувати лише одну форму, все одно потрібно створити кілька екранних форм. Це дасть право групувати та структурувати функції програми, що дозволить користувачам легко орієнтуватися в програмі та виконувати необхідні функції.

Вибір кольору є одним з найважливіших критеріїв при розробці графічного інтерфейсу. Кольори мають дивовижну властивість: вони можуть зробити просте унікальним, а прекрасне – потворним.

Вибір шрифту також відіграє важливу роль у проектуванні та розробці інтерфейсу користувача. Вибравши гарний шрифт, ви зможете домогтися високої читабельності та сприйняття тексту.

Після детального вибору шрифтів і колірних схем слід більш важливий крок - розміщення і групування елементів на екрані. Необхідно продумати, якою повинна бути структура кожної форми, що вона повинна робити і які елементи міститиме.

Добре структурована форма екрана визначає простоту функціональності та зручність використання.

Структуру програми можна відобразити у вигляді моделі, максимально наближеної до вигляду на мобільному пристрої. На малюнку 2.4 показано вигляд головного меню та вибір необхідних вікон.

Рисунок 2.4 – Макет головного

На малюнку 2.5 зображено меню з вибором необхідного вікна.

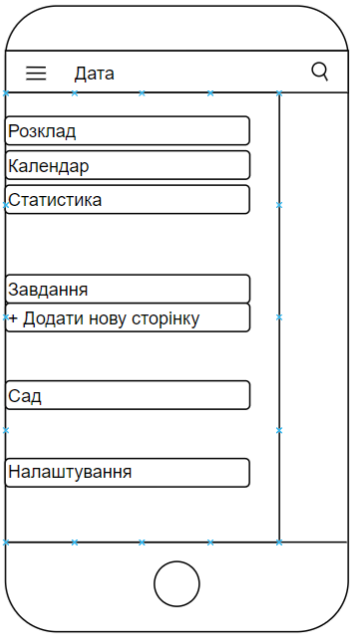

Рисунок 2.5 – Макет вибору потрібного

Вікна із відображенням календаря та вибору вікон, це одні з головних вікон додатку, інформація показана на них не є надлишковою, що робить інтерфейс набагато простішим та інтуітивнішим. Для того щоб додати видалити, чи редагувати завдання, переглянути статистику тощо користувач повинен натиснути на відповідну кнопку, після чого відкриється відповідне вікно на рисунках 2.6, 2.7.

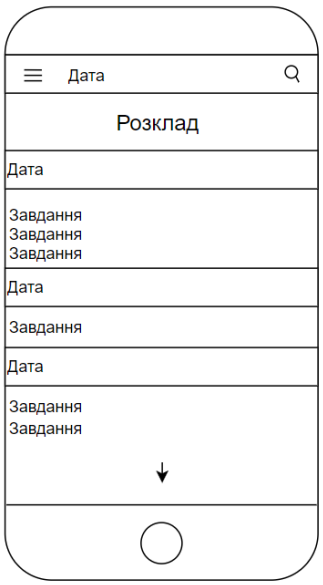

Рисунок 2.6 – Макет меню

Меню розкладу дозволяє переглядати на котрі дати заплановані завдання на конкретну дату данним меню можна керувати за допомогою свайпів зображених стрілочкою на макеті.

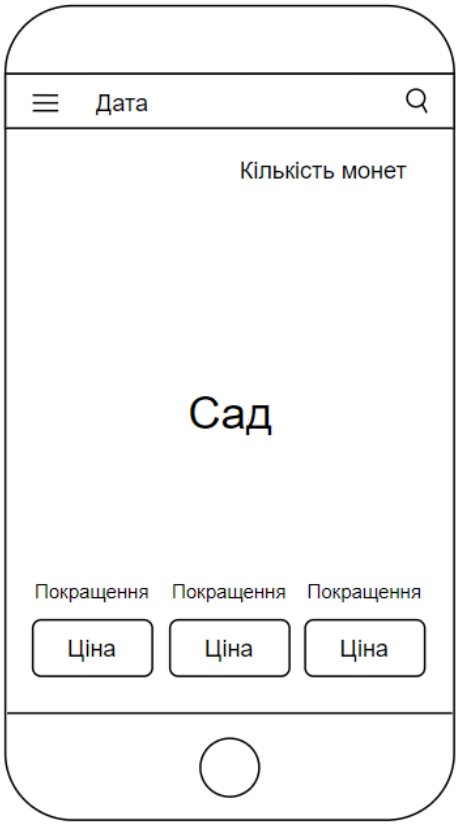

Рисунок 2.7 – Макет вікна «Сад»

В данному вікні користувач може покращувати свій сад за монети які можна получити за вчасне виконання завдання. Данна гейміфікація проекту дозволить більше змотивувати користувачів та відображати стабільність виконання завдань не тільки статистично.

Наступне вікно зображене на рисунку 2.8 є вікно статистики. В данному вікні користувач може переглядати загальну статистику по всіх завдання (скільки часу витрачено, кількість виконаних завдань, тощо). Це допоможе користувачу краще відслідковувати свою ефективність та проводити певну аналітику.

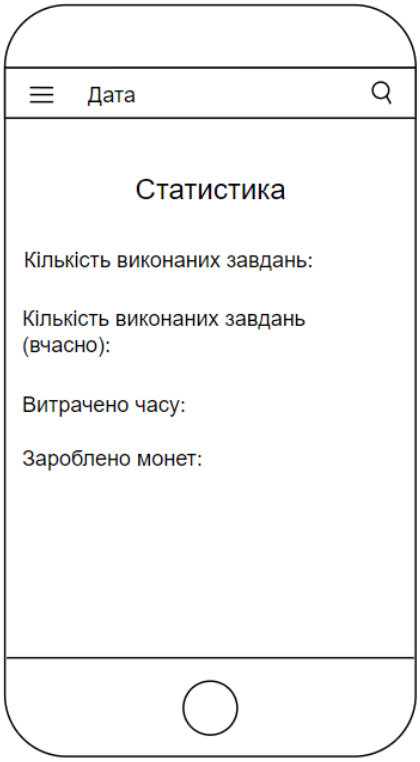

Рисунок 2.8 – Макет вікна

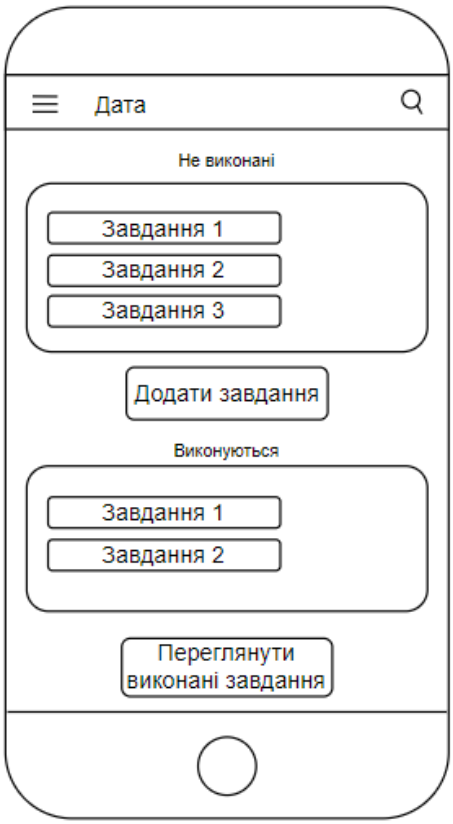

Рисунок 2.9 – Макет вікна завдання

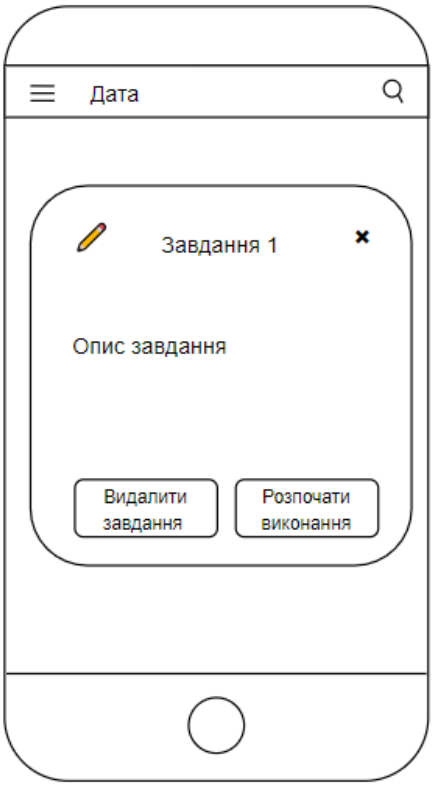

Рисунок 2.10 – Макет форми при натискані на завдання

На рисунках 2.9 та 2.10 зображено вікно завдання та форма для взаємодії користувача з завданнями на якій можна переглянути завдання та опис відредагувати завдання, розпочати виконання чи видалити завдання.

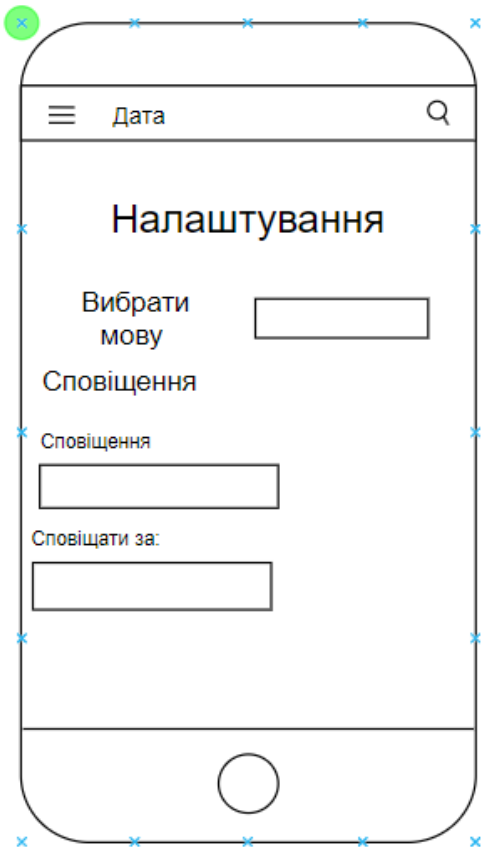

Рисунок 2.11 – Макет вікна налаштування

На рисунку 2.11 зображено вікно налаштувань що забезпечує налаштування додатку.

3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ОРГАНАЙЗЕРУ

3.1 Обгрунтування технологій та засобів реалізації мобільного органайзеру

3.1.1 Мова програмування та операційна система

Розробка мобільного додатку відбувалася за допомогою мови програмування Java. Також прямим конкурентом Java для розробки android-додатків є Kotlin. Хоча Kotlin є більш сучасною мовою програмування, перевагу віддав Java через наступні пункти.

Java – є однією з найпопулярніших в світі мов програмування, що означає велику кількість framework-ів чи бібліотек, які можуть значно спростити розробку програми.

Велика кількість форумів де досвідчені програмісти спілкуються між собою, новачки переймають досвід, пізнають щось нове та можуть найти відповідь на запитання які цікавлять

Також синтаксис Java не є досить складним, що дозволяє не витрачати багато часу на вивчення, а одразу приступити до виконання завдання та створення програмного забезпечення.

Одним з ключових пунктів розробки на Java стало незалежність данної мови від операційної системи чи апаратного забезпечення, що дозволяє використовувати програми незалежно від поєднання апаратного забезпечення чи операційної системи, програми повині працювати однаково.

Android

Розробка додатку для android була обрана через велику популярність даної операційної системи яка працює яка використовується як на бюджет пристроях так і на флагманах.

Велика кількість користувачів по всьому світу. Розробка додатків для android дозволяє залучити велику кількість користувачів, що може стати

досить хорошою рекламою додатку, а якщо включити в додаток рекламу так ще й монетизацію.

Також велика кількість інструментів для розробки додатків для даної операційної системи, які постійно підтримуються та оновлюються. Наприклад android studio.

У цілому розробка додатків для Android є досить цікавою, є можливість залучити велику кількість аудиторії.

3.1.2 Середовище розробки

При виборі середовища для розробки потрібно оцінити його його функціональну структуру, наявність інструментів розробки, графічний інтерфейс, які операційні системи підтримує та чи підтримується обрана мова програмування.

Переглянувши та проаналізувавши велику кількість існуючих середовищ на ринку. Було обрано Android Studio.

Android Studio - це офіційне інтегроване середовище розробки (IDE) для платформи Android, розроблене компанією Google. Це потужний інструмент для створення мобільних додатків на мовах програмування Java та Kotlin.

Функції які пропонує Android Studio для зменьшення часу розробки програми та підвищення продуктивності :

- Графічний редактор інтерфейсу користувача.
- Емулятор пристроїв.
- Підтримка мов програмування Java та Kotlin.
- Редактор коду з автодоповненням.
- Дебагер.

Android Studio це потужне інтегроване середовище розробки, яке забезпечує повний набір інструментів для розробки мобільних додатків для платформи Android.

З його допомогою було створено програму яку можна розмістити на різних платформах.

3.2. Реалізація функціоналу мобільного застосунку

а. Завантаження інструментів розробки.

б. Підключення необхідних бібліотек.

в. Налаштування середовища.

г. Розробка програми.

3.2.1 Завантаження необхідного забезпечення для розробки

Для початку розробки завантажуємо необхідне середовище розробки додатку яке було обрано вище. Середовище «Android-studio» є безкоштовним, тому середовище можна встановити без проблем рисунок 3.1.

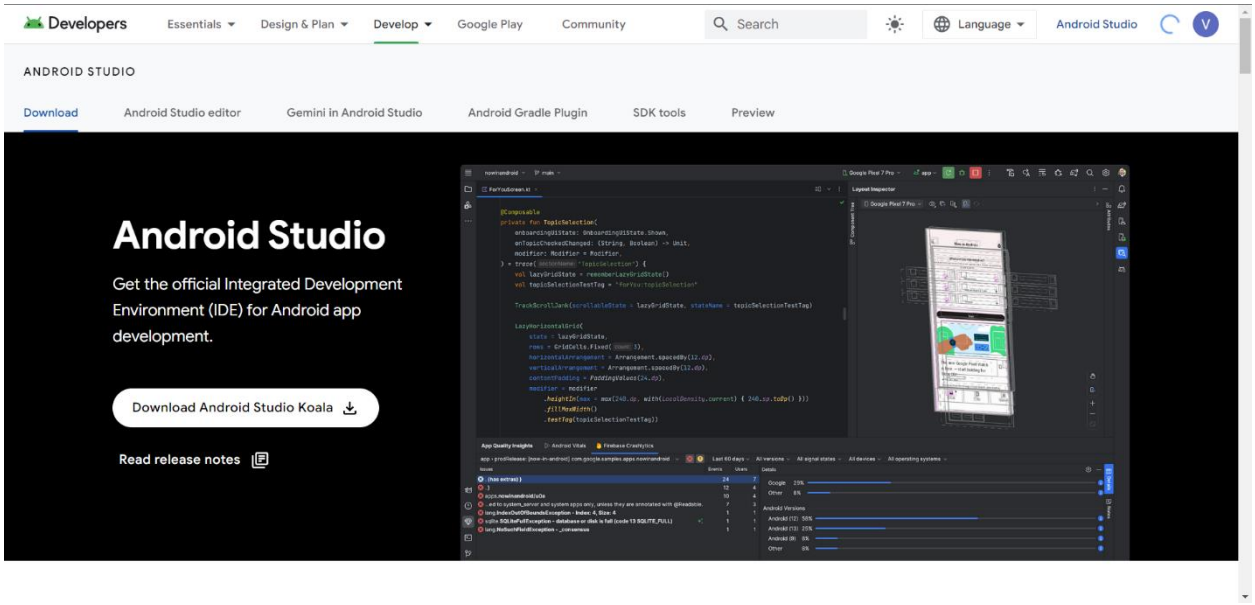

Рисунок 3.1 – Сторінка завантаження середовища «Android-studio»

Під час інсталяції автоматично завантаджуються всі потрібні матеріали які знадобляться при розробці.

При створенні проекту для розробки програми потрібно враховувати фактори для коректної роботи програми. Наприклад мінімальну підтримувану версію операційної системи тощо.

Для спрощення програмне середовище «Android studio» надає статистику найбільш використовуваних операційних систем Android також вибрати пристрій для якого буде розроблятися програма рисунок 3.2.

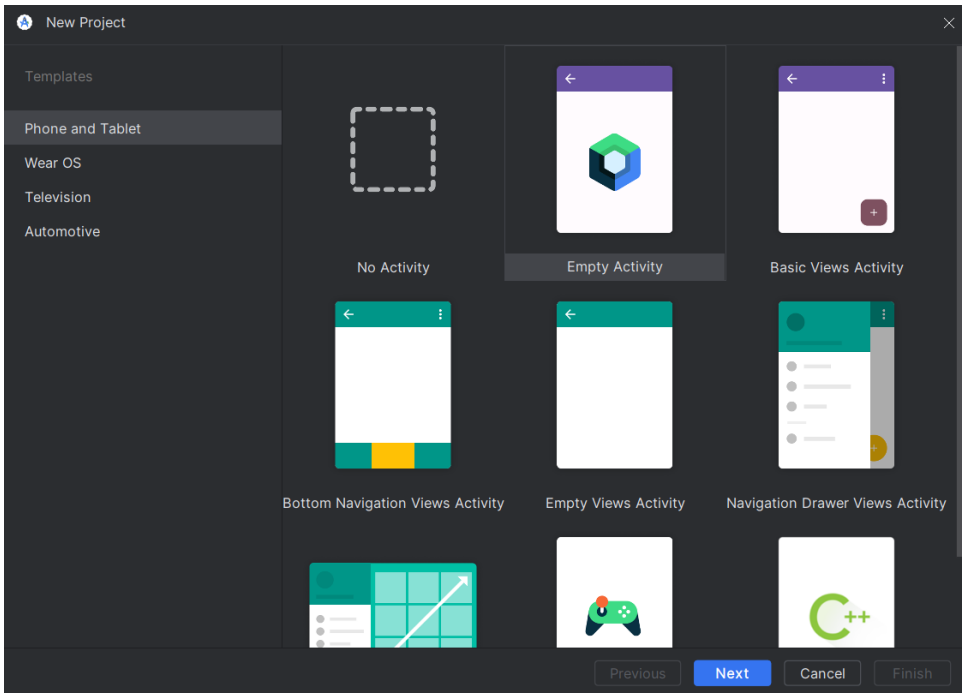

Рисунок 3.2 – вибір пристрою для розробки додатку

Після того як обрали пристрій та макет для початку розробки додатку переходимо до налаштувань самого проекту рисунок 3.3

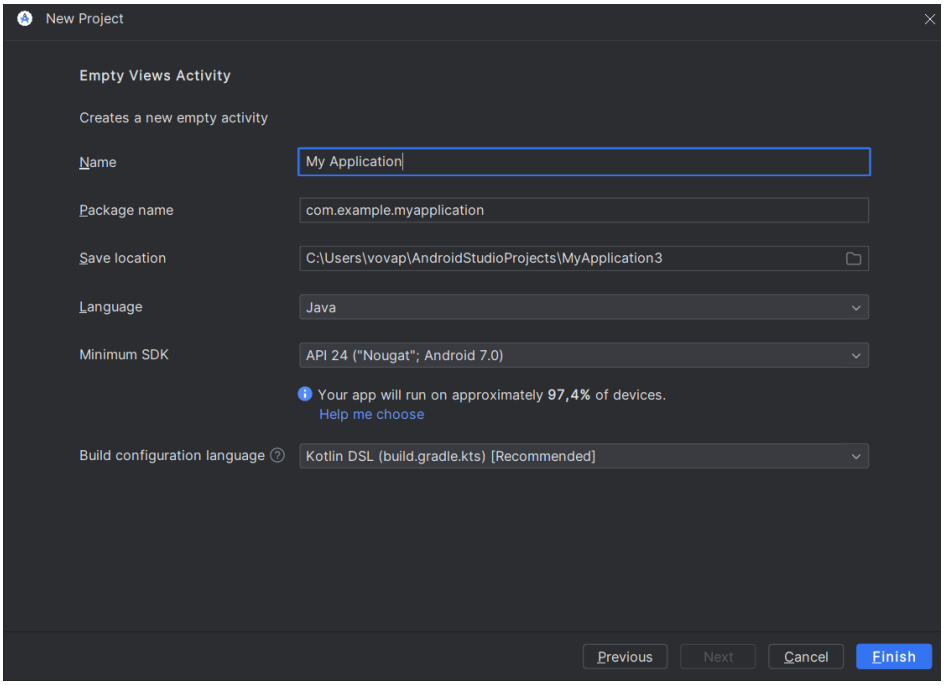

Рисунок 3.3 – Створення проєкту

В даному діалоговому вікні вводимо назву проєкту, вибираємо шлях до даного проекту, мову програмування на якій буде виконуватися проект тобто вищезгадувану Java, операційну систему на якій буде працювати додаток, якщо операційна система буде нижче за вказану додаток працювати не буде.

3.2.2 Реалізація інтерфейсу користувача

При написанні програмного коду увага завжди приділяється його стандартизації. Це означає, що ви пишете свій код відповідно до певних правил та інструкцій щодо проектування конкретних блоків, класів, змінних тощо. Тому створення програми стає трудомістким процесом.

Початком роботи є розміщення функціональних елементів на формах відповідно до заздалегідь розроблених макетів. Існує багато об'єктів, які можна розмістити на формі.

Розглянемо лише ті, які потрібні для реалізації програми:

а) TextView – це найчастіше використовуваний елемент у будь-якій програмі. Він може відображати будь-яку інформацію.

б) Button – елемент кнопка, містить в собі великий набір функцій. Кнопки дозволяють реалізовувати складні користувацькі інтерфейси. Кнопки дають змогу виконувати певні функції залежно від потреб користувача.

в) ImageView – вся інформація, представлена у вигляді графіки, реалізована за допомогою цього компонента.

г) ListViev – це елемент, який вражає своїми можливостями – зміна властивості призводить до суттєвих змін можливостей елемента. З його допомогою можна не лише відображати задані елементи, а й створювати комбо-списки зі своїми властивостями.

Після того, як екранні форми було сформовано, подальшою роботу буде програмування кнопок та основного функціоналу.

Основними функціями в цьому додатку є обробка виконання певних дій. Розглянемо функції додавання видалення дій в додатку (лістинг 3.1, 3.2, 3.3).

Лістинг 3.1 Функція додавання завдання.

```
public void onClick(View v) {
                                                String    title    =
titleEditText.getText().toString().trim();
                                             String    description    =
descriptionEditText.getText().toString().trim();
                                               String    assignee    =
assigneeEditText.getText().toString().trim();
                                                String    reward    =
rewardEditText.getText().toString().trim();
                                                String    userId    =
userIdEditText.getText().toString().trim();

                // Perform validation if necessary

                // Print or process the data as needed
                String message = "Title: " + title + "\n"
                                 + "Description: " + description +
"\n"
                                + "Assignee: " + assignee + "\n"
                                + "Reward: " + reward + "\n"
                                + "User ID: " + userId;
                   Toast.makeText(TaskFormActivity.this, message,
Toast.LENGTH_SHORT).show();

                // Clear the EditText fields after submission
                clearFields();
            };
```

Лістинг 3.2 – Виконати завдання

```
public boolean updateTaskStatus(int userId) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COL_STATUS, 2);

        int rowsAffected = db.update(TABLE_NAME, values,
```

```
                COL_USER_ID + " = ?",
                new String[]{String.valueOf(userId)});
        return rowsAffected > 0;
    }
```

Лістинг 3.3 – Покращення саду

```
  SQLiteDatabase db = this.getWritableDatabase();
        int currentGardenId = getCurrentGardenId(db, userId);
        db.execSQL("UPDATE users SET coins = coins - ?, garden_id
= ? WHERE id = ?",
                    new  Object[]{priceDifference,  newGardenId,
userId});
        if (currentGardenId != newGardenId) {
            String imageUrl = getGardenImageUrl(db, newGardenId);
            imageView.setImageUrl(imageUrl);
        }
        db.close();
    }
public void updateCoinsAndGarden(int userId, int newGardenId, int
priceDifference) {
       private  int  getCurrentGardenId(SQLiteDatabase  db,  int
userId) {
        int gardenId = -1;
        String query = "SELECT garden_id FROM users WHERE id=?";
        Cursor     cursor     =     db.rawQuery(query,     new
String[]{String.valueOf(userId)});
        if (cursor.moveToFirst()) {
            gardenId                                            =
cursor.getInt(cursor.getColumnIndex("garden_id"));
        }
        cursor.close();
        return gardenId;
    }
```

3.3 Розробка графічного інтерфейс

Розробка мобільних додатків передбачає інтеграцію функціональності з графічним інтерфейсом користувача, який включає створення інтерактивних екранів за певними схемами та шаблонами. Ось основні екрани мобільного органайзеру:

– Головний екран (рисунок 3.4).

– Вибір потрібного вікна (рисунок 3.5).

– Вікно розкладу де показано до якого числа потрібно виконати завдання рисунок (3.6).

– Вікно списку всіх завдань (рисунок 3.7).

– Вікно додавання завдань рисунок (3.8).

– Вікно статистики рисунок (3.9).

– Вікно саду рисунок (3.10).

Для початку розглянемо вікно головного екрану на ньому відображається календар з сьогоднішньою датою рисунок (3.4).

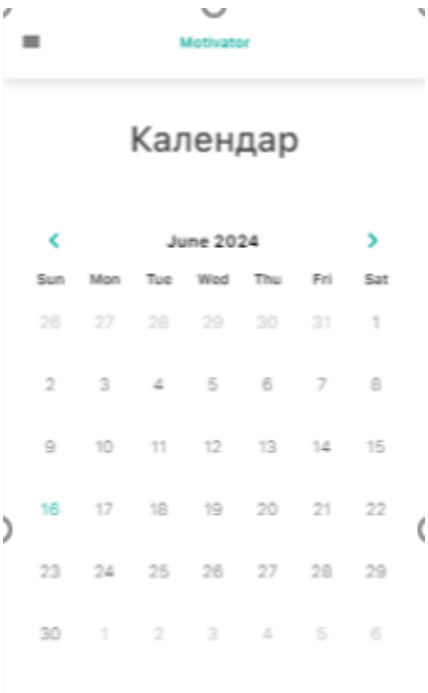

Рисунок 3.4 – Головний екран програми

В цьому меню користувач немає багато можливостей окрім того як переглянути дату. Для того щоб перейти на інше вікно користувачу потрібно натиснути на кнопку menu в лівому верхньому куті після чого в нього відкриється список доступних вікон на які він може перейти рисунок 3.5.

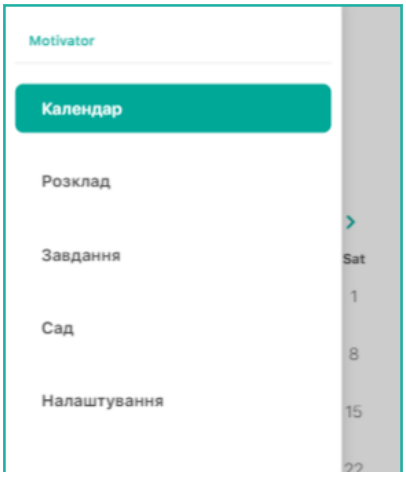

Рисунок 3.5 – Вибір потрібного вікна

В цьому вікні menu користувач для того щоб обрати необхідне вікно повинен просто обрати його. Наступне вікно йде розклад де користувач може переглядати повний список завдань які йому потрібно виконати до певного терміну. Термін користувач обирає сам рисунок 3.6.

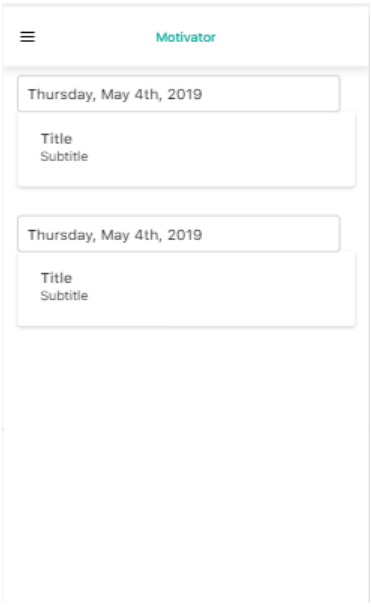

Рисунок 3.6 – Вікно розкладу

Далі переходимо до діалогово вікна завдання на якому відображається список завдань які не виконано в ньому користувач може добавити завдання, редагувати завдання та видаляти (рисунок 3.7).

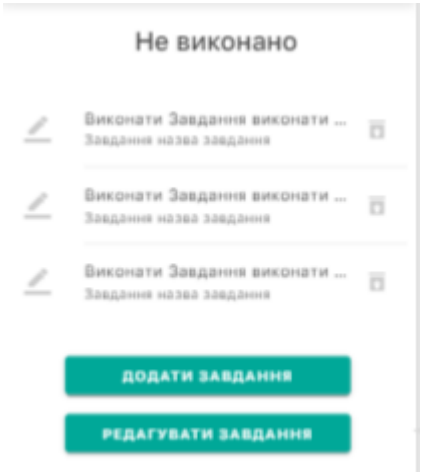

Рисунок 3.7 – Вікно завдань

Для того щоб виконати любі дії з завданнями користувачеві потрібно натиснути на потрібну кнопку. Якщо користувач натисне на «Додати завдання» відкриється відповідне вікно на якому користувач зможе ввести необхідні дані для додавання завдання (рисунок 3.8).

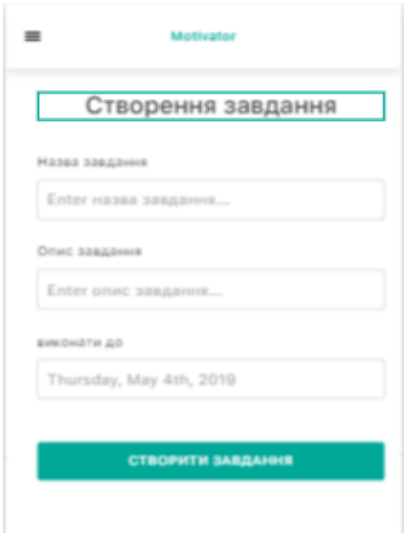

Рисунок 3.8 Вікно додавання завдань

Після виконання додавання програма повтерається до вікна завдань для того щоб продовжити роботу з самими завданнями. Наступний крок це видалення. Для того щоб користувач видалив завдання з списку йому достатньо натиснути на іконку корзини після чого завдання видалиться.

Наступне вікно це перегляд статистики виконаних завдань. Користувач може переглядати статистику за весь час користування програмою це йому допоможе проаналізувати якість виконання завдань рисунок 3.10.

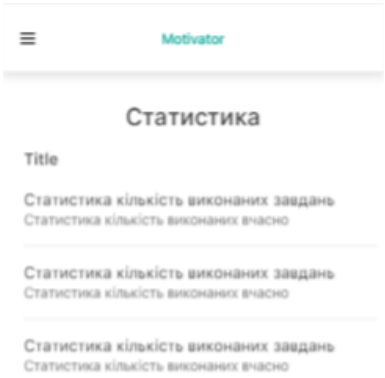

Рисунок 3.10 – Вікно статистики

Останнє вікно це вікно саду на якому користувач вирощує своє дерево для того щоб покращити свій сад користувач повинен заробити валюту яку можна получити за виконання завдань рисунок 3.11.

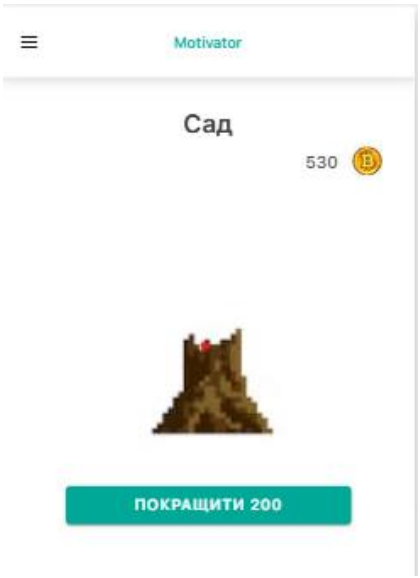

Рисунок 3.11 – Вікно саду

3.4 Тестування

Тестування програмного забезпечення — це процес перевірки відповідності встановленим вимогам до продукту та реалізованим функціональним можливостям, спостереження за поведінкою додатків у штучно створених ситуаціях і обмежений набір тестів, обраних певним чином Тестування програмного забезпечення забезпечує об’єктивний і незалежний погляд на програмне забезпечення, що дозволяє компаніям оцінювати та розуміти ризики, пов’язані з впровадженням програмного забезпечення. Методи тестування включають, але не обмежуються:

– Аналіз вимог до продукту щодо повноти та точності в усіх контекстах, включаючи галузеві та бізнес-перспективи, доцільність впровадження та прибутковість, зручність використання, продуктивність, безпеку та коректність інфраструктури.

– Аналіз архітектури продукту та загального дизайну.

– Співпрацюйте з розробниками продуктів, щоб покращити методи кодування, шаблони проектування та тести, які можна написати як частину коду на основі різноманітних методів, зокрема: В.

– Граничні умови тощо.

– Запустіть програму перевірки роботи.

– Огляд інфраструктури розгортання та пов’язаних сценаріїв та засобів автоматизації.

– Бере участь у виробничій діяльності з використанням методів моніторингу та моніторингу.

– Тестування програмного забезпечення надає користувачам об'єктивну незалежну інформацію про якість і ризик дефектів програмного забезпечення.

Помилки програмного забезпечення виникають через такий процес: Програміст зіткнувся з помилкою (дефект, збою) у вихідному коді програмного забезпечення.

Якщо припустити цю помилку, система може видавати неправильні результати та в деяких ситуаціях виходити з ладу.

Не всі помилки програмного забезпечення викликані помилками кодування.

Однією з найпоширеніших причин дорогих помилок є відсутність вимог або нерозпізнані вимоги, через які розробники не помічають помилок.

Прогалини у вимогах часто є нефункціональними вимогами, такими як тестування, продуктивність, масштабованість, зручність обслуговування та інші вимоги безпеки.

Існують різні підходи до тестування програмного забезпечення.

Статичне тестування називається проходженням, проходженням або перевіркою, тоді як виконання запрограмованого коду для конкретного тестового випадку називається динамічним тестуванням.

Статичне тестування починається на ранніх стадіях життєвого циклу програмного забезпечення і тому є частиною процесу перевірки.

Цей тип тестування може не потребувати комп'ютера, як, наприклад, перевірка вимог.

Динамічні тести виконуються під час запуску програми.

Динамічні тести можна розпочати до того, як програма буде завершена на 100%, щоб протестувати окремі розділи коду, і їх можна застосувати до окремих функцій або модулів.

Загальні методи - використовувати заглушку/драйвер або робити це під час налагодження.

Статичне тестування включає перевірку, але динамічне також включає перевірку.

Пасивне тестування допомагає змінити поведінку системи без взаємодії з програмним продуктом.

На відміну від активного тестування, тестер не надає тестових даних, а перевіряє системні журнали та трасування таблиця 3.1.

Таблиця 3.1 – Тестування додатку

№	Тест кейс	Очікуваний результат	Отриманий результат
1	Невірні дані при додавані даних	При введені невірних данних система повідомляє користувача про це	При введені не вірних даних система повідомляє про помилку
2	Пусті поля при вводі даних	При спробі створення запису з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення запису з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
3	Створення відмітки з пустими полями	При спробі створення відмітки з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення відмітки з пустими полями система повідомляє користувача про тщо ці поля необхідно заповнити

Результати тестування показують, що система повністю працездатна та готова до використання в реальних ситуаціях.

Для повної перевірки вам слід перевірити свою програму на іншому пристрої.З цієї причини тести проводяться на слабкому мобільному телефоні з операційною системою Android 6.0. Програма успішно відкрилася, працездатність перевірена. На рисунку 3.15 зображен тестування вікно розкладу.

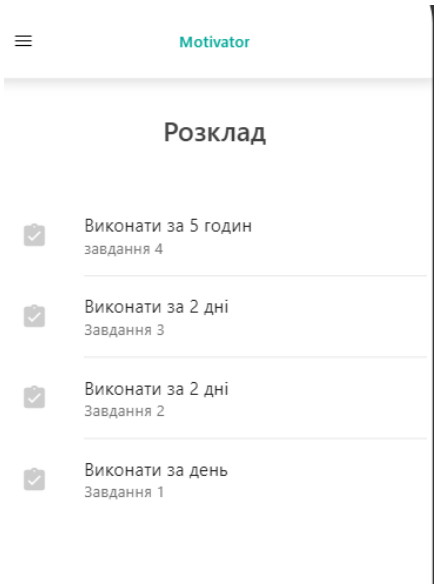

Рисунок 3.15 – Вікно розкладу

Наступним кроком тестуємо вікно саду яке успішно пройшло перевірку рисунок 3.16

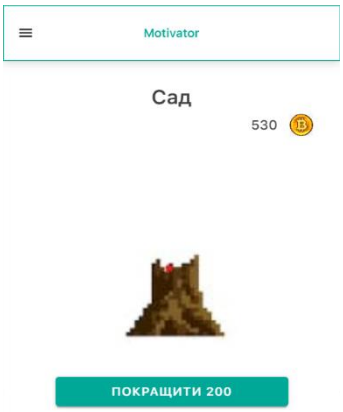

Рисунок 3.1 – Вікно саду

Наступним кроком тестуємо додавання завданя та вікно завдань рисунок 3.17 та рисунок 3.18.

Вікно додавання завдань успішно пройшло тестуваннння ніяких проблем не виникло

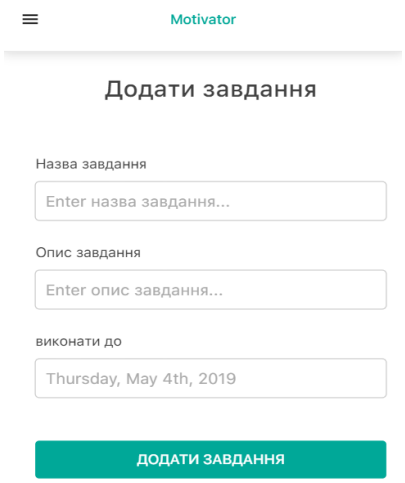

Рисунок 3.18 – Вікно додавання

Вікно додавання завданя також пройшло успішно тестування.

4.ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Аналіз ринку.

Результати роботи є додатком розміром 3,8 МБ. Наразі на ринку доступні подібні продукти як з програмним, так і з апаратним забезпеченням.

Ця програма є її вдосконаленою версією та реалізована повністю програмно. Замість технічної частини продуктів, які пропонуються на ринку, цей додаток використовує пристрій користувача.Для роботи програми потрібна версія телефону Android.Для використання мобільного додатку версія Android не має значення, але має бути Android 5.0 або вище. Програма не сумісна зі старими версіями і може не працювати. Екрани пристроїв можуть мати довільні діагональні лінії.

Основні потреби наших клієнтів - найкраща ціна та хороша якість продукції. Крім того, продукт має бути простим для встановлення та використання. Для ширшого та кращого охоплення краще використовувати програмне забезпечення онлайн. На українському ринку багато конкурентів.Однак іноземні компанії створюють дуже хороші додатки, тому існує велика конкуренція. Головною відмінністю створеного мобільного додатку є наявність у програмі української мови, що наразі дуже важливо.

Ми зупинимося на програмі todolist, яка зараз дуже популярна на українському ринку. Сам додаток дуже легко налаштувати, але в ньому відсутня українська мова і деякі функції платні.

4.2 Розрахунок

Усі необхідні дані подано в додатку Б.

Зарплата проектувальникам розраховується на двох осіб.

Потрібен розрахунок заробітної платні для двох працівників: програміст та тестувальник (таблиця 4.2).

Середня заробітна плата молодшого програміста становить – 18200 грн в місяць.

Податок фізичних осіб: Податок для фізичних осіб в Україні становить 18% від ЗП в данному випадку становить 3276 грн.

Військовий збір: Військовий збір становить 1.5% від ЗП а тобто 273 Єдиний соціальний внесок. Відрахування: 3276 грн. + 273 грн = 3549 грн.

Від повної суми програміст отримає – 14651 грн. (18200 – 3465).

Середня заробітна плата молодшого тестувальника становить – 16800 грн в місяць.

Податок фізичних осіб: Податок для фізичних осіб в Україні становить 18% від ЗП в данному випадку становить 3024 грн.

Військовий збір: Військовий збір становить 1.5% від ЗП а тобто 252

Єдиний соціальний внесок. Відрахування: 3024 грн. + 252 грн. = 3276 грн.

Від повної суми програміст отримає – 13524 грн. (16800 – 3276).

Таблиця 4.1 Розрахунок заробітньої плати.

№	Посада	Оклад грн/міс	Відрахування грн/міс	Кількість		Сума з/п грн
				чол	міс	
1	Програміст	18200	3465	1	4	58604
2	Тестувальник	16800	3276	1	1	13524
Усього заробітньої плати						72128

Контрагенні роботи – 72128 * 12% = 8655 грн.

Витрат на відрядження немає

Інші непрямі витрати становлять 72128 * 42% = 30293 грн.

Загальна сума витрат становить 72128 + 8655 + 30239 =111076 грн.

4.3 Обґрунтування необхідності розробки

Розробка проекту полягає у тому, що кінцевий продукт призначений для широкого кола користувачів, незалежно від віку. Основна мета проекту -

спростити, зробити більш ефективним та цікавим організацію робочого дня. Проект спрямований на підвищення ефективності управління графіком та завданнями користувачів, надаючи їм можливість краще організувати свій час. Користувачі додатку можуть сприяти зменшенню вирубки лісів та захисту навколишнього середовища, обмежуючи використання паперу. Ефективний тайм-менеджмент є ключовим аспектом для кожної людини. Багато успішних осіб планують свої дні заздалегідь за чітким розкладом, що допомагає їм краще сконцентруватися на завданнях, розподіляти час між ними та зробити їх виконання продуктивним та приємним.

При правильному управлінні, програма має значний потенціал для розвитку, може бути успішною та прибутковою для розробників та інвесторів. Крім того, програма є дуже корисною для користувачів, покращуючи їх досвід і надаючи цінні інструменти та ресурси для задоволення їх потреб. За допомогою правильних процесів і стратегій програма може охопити широку аудиторію та закріпитися на ринку.

Це призведе до довгострокового успіху та сталості. Крім того, її корисність для потенційних користувачів підвищить лояльність і задоволеність клієнтів, що є ключовими факторами для довгострокового зростання і розвитку.

ВИСНОВКИ

У рамках моєї дипломної роботи був створений мобільний додаток, який слугує органайзером щоденних завдань та спеціальних завдань.

Після розгляду основних правил та рекомендацій щодо розробки та створення мобільних додатків, а також огляду подібних рішень, які існують на даний момент, органайзер був розроблений та реалізований відповідно до вимог замовника.

Зокрема, реалізовано такі функції:

- Переглянути список завдань та термін їх виконання.
- Перегляд, збереження та редагування даних.
- Можливість переглядати статистику.
- Гейміфікація.

Для реалізації мобільного додатку використано мову програмування Java та середовище розробки Android Studio.

Ось переваги мобільного органайзера, створеного для відстеження випадків:

- Не використовує системні ресурси.
- Простий та інтуїтивно зрозумілий інтерфейс.
- Немає зайвого функціоналу.

Тестування програми підтвердило адаптивність до різних специфікацій пристроїв.За потреби ви можете додати функціональні можливості до своєї програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Типи мобільних додатків. Smile-ukraine.com: вебсайт. URL: https://smileukraine.com/ua/mobile-apps/mobile-apps-types (дата звернення: 23.11.2023).

2. Розробка мобільних додатків. Webcase.com.ua: вебсайт. URL: https://webcase.com.ua/uk/razrabotka-mobilnyh-prilozhenij/ (дата звернення: 27.11.2023).

3. Мобільний інтерфейс. Wezom.com.ua: вебсайт. URL: https://wezom.com.ua/blog/dizajn-interfejsov-mobilnyh-prilozhenij (дата звернення: 01.12.2023).

4. Мобільні додатки органайзери. Zhyvyaktyvno.org: вебсайт. URL: https://zhyvyaktyvno.org/news/mobln-dodatki-organajzeri (дата звернення: 12.01.2024).

5. Діаграма варіантів використання. Studopedia.ru: вебсайт. URL: https://studopedia.ru/19_284009_diagrama-variantiv-vikoristannya-USEcase-diagram.html (дата звернення: 15.11.2023).

6. Розробка UML діаграм. Studfile.net: вебсайт. URL: https://studfile.net/preview/5200239/page:6/ (дата звернення: 30.11.2023).

7. Керівництво по мові програмування Java. Habr.com: вебсайт. URL: https://habr.com/ru/hub/java/ (дата звернення: 03.04.2022).

8.Android Studio. Metanit.com: вебсайт. URL: https://metanit.com/java/android/ (дата звернення: 06.04.2024).

ДОДАТКИ

Додаток А

Лістинг А1 – DBHelper

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "tasks_database";
    private static final int DATABASE_VERSION = 1;

    private static final String TABLE_NAME = "tasks";
    private static final String COL_ID = "id";
    private static final String COL_STATUS = "status";
    private static final String COL_USER_ID = "user_id";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String createTableQuery = "CREATE TABLE " + TABLE_NAME +
" ("
                + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + COL_STATUS + " INTEGER, "
                + COL_USER_ID + " INTEGER)";
        db.execSQL(createTableQuery);
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public boolean updateTaskStatus(int userId) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COL_STATUS, 2);

        int rowsAffected = db.update(TABLE_NAME, values,
                COL_USER_ID + " = ?",
                new String[]{String.valueOf(userId)});

        db.close();
        return rowsAffected > 0;
    }

    public void updateCoinsAndGarden(int userId, int newGardenId,
int priceDifference) {
        SQLiteDatabase db = this.getWritableDatabase();
        int currentGardenId = getCurrentGardenId(db, userId);
        db.execSQL("UPDATE users SET coins = coins - ?, garden_id
= ? WHERE id = ?",
                new    Object[]{priceDifference,    newGardenId,
userId});
        if (currentGardenId != newGardenId) {
            String imageUrl = getGardenImageUrl(db, newGardenId);
            // Тут потрібно реалізувати логіку оновлення UI для
зображення саду
            // Наприклад, через зміну ресурсу або завантаження
зображення
        }
```

```
        db.close();
    }

    private int getCurrentGardenId(SQLiteDatabase db, int userId)
{
```

Лістинг А2 – Додаванння завдання

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class TaskFormActivity extends AppCompatActivity {

    private    EditText    titleEditText,    descriptionEditText,
assigneeEditText, rewardEditText, userIdEditText;

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

        // Ініціалізація елементів розмітки
        titleEditText = findViewById(R.id.titleEditText);
        descriptionEditText                                    =
findViewById(R.id.descriptionEditText);
        assigneeEditText = findViewById(R.id.assigneeEditText);
        rewardEditText = findViewById(R.id.rewardEditText);
        userIdEditText = findViewById(R.id.userIdEditText);
        Button submitButton = findViewById(R.id.submitButton);

        // Обробник натискання на кнопку "Submit"
```

```
        submitButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Отримання введених даних з полів вводу
                String                  title                  =
titleEditText.getText().toString().trim();
                String               description               =
descriptionEditText.getText().toString().trim();
                String                 assignee                =
assigneeEditText.getText().toString().trim();
                String                  reward                 =
rewardEditText.getText().toString().trim();
                String                  userId                 =
userIdEditText.getText().toString().trim();

                // Виконання валідації, якщо потрібно

                // Формування повідомлення для відображення
                String message = "Title: " + title + "\n"
                                + "Description: " + description +
"\n"
                                + "Assignee: " + assignee + "\n"
                                + "Reward: " + reward + "\n"
                                + "User ID: " + userId;

                // Відображення повідомлення Toast зі збереженими
даними
                Toast.makeText(TaskFormActivity.this,    message,
Toast.LENGTH_SHORT).show();

                // Очищення полів вводу після відправки
                clearFields();
            }
        });
```

```
    }

    // Метод для очищення полів вводу
    private void clearFields() {
        titleEditText.setText("");
        descriptionEditText.setText("");
        assigneeEditText.setText("");
        rewardEditText.setText("");
        userIdEditText.setText("");
    }
}
```

Лістинг А3 Видалення завдання import android.os.Bundle;

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class TaskFormActivity extends AppCompatActivity {

    private EditText titleEditText, descriptionEditText,
assigneeEditText, rewardEditText, userIdEditText, taskIdEditText;
    private DBHelper dbHelper;

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

        // Ініціалізація елементів розмітки
        titleEditText = findViewById(R.id.titleEditText);
        descriptionEditText =
findViewById(R.id.descriptionEditText);
        assigneeEditText = findViewById(R.id.assigneeEditText);
```

```
        rewardEditText = findViewById(R.id.rewardEditText);
        userIdEditText = findViewById(R.id.userIdEditText);
        taskIdEditText  =  findViewById(R.id.taskIdEditText);  //
додане поле для введення ID завдання
        Button submitButton = findViewById(R.id.submitButton);
        Button deleteButton = findViewById(R.id.deleteButton); //
додана кнопка "Видалити"

        dbHelper = new DBHelper(this);

        // Обробник натискання на кнопку "Submit"
        submitButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Отримання введених даних з полів вводу
                String                  title                  =
titleEditText.getText().toString().trim();
                String               description               =
descriptionEditText.getText().toString().trim();
                String                 assignee                =
assigneeEditText.getText().toString().trim();
                String                  reward                 =
rewardEditText.getText().toString().trim();
                String                  userId                 =
userIdEditText.getText().toString().trim();

                // Виконання валідації, якщо потрібно

                // Формування повідомлення для відображення
                String message = "Title: " + title + "\n"
                                + "Description: " + description +
"\n"
                                + "Assignee: " + assignee + "\n"
                                + "Reward: " + reward + "\n"
```

```
                                + "User ID: " + userId;

                // Відображення повідомлення Toast зі збереженими
даними
                Toast.makeText(TaskFormActivity.this,    message,
Toast.LENGTH_SHORT).show();

                // Очищення полів вводу після відправки
                clearFields();
            }
        });

        // Обробник натискання на кнопку "Видалити"
        deleteButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String               taskIdString               =
taskIdEditText.getText().toString().trim();
                if (!taskIdString.isEmpty()) {
                    long taskId = Long.parseLong(taskIdString);
                    boolean result = dbHelper.deleteTask(taskId);

                    if (result) {
                        Toast.makeText(TaskFormActivity.this,
"Task deleted successfully", Toast.LENGTH_SHORT).show();
                        clearFields();
                    } else {
                        Toast.makeText(TaskFormActivity.this,
"Task not found", Toast.LENGTH_SHORT).show();
                    }
                } else {
                    Toast.makeText(TaskFormActivity.this, "Please
enter task ID", Toast.LENGTH_SHORT).show();
                }
```

```
            }
        });
    }

    // Метод для очищення полів вводу
    private void clearFields() {
        titleEditText.setText("");
        descriptionEditText.setText("");
        assigneeEditText.setText("");
        rewardEditText.setText("");
        userIdEditText.setText("");
        taskIdEditText.setText("");
    }
}
```

Лістинг А4 – Оновлення даних

```
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class TaskFormActivity extends AppCompatActivity {

    private EditText titleEditText, descriptionEditText,
assigneeEditText, rewardEditText, userIdEditText, taskIdEditText;
    private DBHelper dbHelper;
    private ListView tasksListView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_task_form);

        // Ініціалізація елементів розмітки
        titleEditText = findViewById(R.id.titleEditText);
        descriptionEditText =
findViewById(R.id.descriptionEditText);
        assigneeEditText = findViewById(R.id.assigneeEditText);
        rewardEditText = findViewById(R.id.rewardEditText);
        userIdEditText = findViewById(R.id.userIdEditText);
        taskIdEditText  =  findViewById(R.id.taskIdEditText);  //
додане поле для введення ID завдання
        Button submitButton = findViewById(R.id.submitButton);
        Button deleteButton = findViewById(R.id.deleteButton); //
додана кнопка "Видалити"
        Button viewTasksButton =
findViewById(R.id.viewTasksButton); // додана кнопка "Переглянути
завдання"
        tasksListView  =  findViewById(R.id.tasksListView);  //
доданий ListView

        dbHelper = new DBHelper(this);

        // Обробник натискання на кнопку "Submit"
        submitButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Отримання введених даних з полів вводу
                String title =
titleEditText.getText().toString().trim();
                String description =
descriptionEditText.getText().toString().trim();
                String assignee =
assigneeEditText.getText().toString().trim();
```

```
                String                  reward                  =
rewardEditText.getText().toString().trim();
                String                  userId                  =
userIdEditText.getText().toString().trim();

                // Виконання валідації, якщо потрібно

                // Формування повідомлення для відображення
                String message = "Title: " + title + "\n"
                        + "Description: " + description +
"\n"
                        + "Assignee: " + assignee + "\n"
                        + "Reward: " + reward + "\n"
                        + "User ID: " + userId;

                // Відображення повідомлення Toast зі збереженими
даними
                Toast.makeText(TaskFormActivity.this,    message,
Toast.LENGTH_SHORT).show();

                // Очищення полів вводу після відправки
                clearFields();
            }
        });

        // Обробник натискання на кнопку "Видалити"
        deleteButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String              taskIdString              =
taskIdEditText.getText().toString().trim();
                if (!taskIdString.isEmpty()) {
                    long taskId = Long.parseLong(taskIdString);
                    boolean result = dbHelper.deleteTask(taskId);
```

```
                    if (result) {
                        Toast.makeText(TaskFormActivity.this,
"Task deleted successfully", Toast.LENGTH_SHORT).show();
                        clearFields();
                        loadTasks(); // Оновлення списку завдань
після видалення
                    } else {
                        Toast.makeText(TaskFormActivity.this,
"Task not found", Toast.LENGTH_SHORT).show();
                    }
                } else {
                    Toast.makeText(TaskFormActivity.this, "Please
enter task ID", Toast.LENGTH_SHORT).show();
                }
            }
        });

        // Обробник натискання на кнопку "Переглянути завдання"
        viewTasksButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loadTasks();
            }
        });

        // Завантаження завдань при запуску активності
        loadTasks();
    }

    // Метод для завантаження та відображення всіх завдань
    private void loadTasks() {
        Cursor cursor = dbHelper.getAllTasks();
```

```
        String[] from = new String[]{DBHelper.COLUMN_TITLE,
DBHelper.COLUMN_DESCRIPTION, DBHelper.COLUMN_ASSIGNEE,
DBHelper.COLUMN_REWARD, DBHelper.COLUMN_USER_ID};
        int[] to = new int[]{R.id.taskTitle, R.id.taskDescription,
R.id.taskAssignee, R.id.taskReward, R.id.taskUserId};

        SimpleCursorAdapter adapter = new
SimpleCursorAdapter(this, R.layout.task_item, cursor, from, to,
0);
        tasksListView.setAdapter(adapter);
    }

    // Метод для очищення полів вводу
    private void clearFields() {
        titleEditText.setText("");
        descriptionEditText.setText("");
        assigneeEditText.setText("");
        rewardEditText.setText("");
        userIdEditText.setText("");
        taskIdEditText.setText("");
    }
}
```

Додаток Б

Назва	Відсоток вирахування
Податок фізичних осіб	18%
Військовий збір	3%
Контрагенські роботи	12%