МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інтелектуальна система планування навантаження з урахуванням специфіки діяльності людини»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-наукова програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м.н-71 Антипенко Богдан Анатолійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» травня 2019 р.

Науковий керівник _____ к.т.н., доц., Марченко А. В.
(підпис)

Голова комісії _____ Шифрін Д. М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2019 р.

ЗАВДАННЯ
на кваліфікаційну роботу магістра студентові

Антипенко Богдан Анатолійович
(прізвище, ім'я, по батькові)

1 Тема проекту *Інтелектуальна система планування навантаження з урахуванням специфіки діяльності людини*

затверджена наказом по університету від «05» лютого 2019 р. №0237-III

2 Термін здачі студентом закінченого проекту «_13__» ___травня___ 2019 р.

3 Вхідні дані до проекту Таблиці енерговитрат людини від виконання активності, формули розрахунку навантаження людини.

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Завдання, Реферат, Вступ, Аналіз предметної області, Постановка задачі та методи дослідження, Моделювання інформаційної системи, Реалізація інформаційної системи, Висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Презентація (20 слайдів).

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз проблеми перевантаження та планування задач	До 15.11.18	
2	Порівняльний аналіз аналогів ІС	До 01.12.18	
3	Аналіз вимог	До 15.12.18	
4	Планування проекту	До 30.12.18	
5	Моделювання інформаційної системи	До 30.01.19	
6	Проектування інтерфейсу продукту	До 12.02.19	
7	Розробка ІС	До 15.04.19	
8	Тестування продукту	До 01.05.19	
9	Оформлення документації по проекту	До 13.05.19	

Магістрант _____ Антипенко Б.А.

Керівник роботи _____ к.т.н., доц. Марченко А.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інтелектуальна система планування навантаження з урахуванням специфіки діяльності людини».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 109 сторінок, у тому числі 56 сторінок основного тексту, 3 сторінки списку використаних джерел, 43 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці інтелектуальної системи планування навантаження з урахуванням специфіки діяльності людини.

У роботі розглянуто проблему перенавантаження людини, формування розкладу людини, проведено огляд традиційних та сучасних аналогів розробленої системи, машинних методів класифікації, створено евристичний алгоритм планування завданнь.

Результатом проведеної роботи є реалізована інтелектуальна система. Експериментальні дослідження полягають у перевірці адекватності розробленого алгоритму формування навантаження людини з урахуванням специфіки її діяльності, шляхом тестування системи, викристовуючи її для оргнізації розкладу та планування навантаження тестувальників.

Наукова новизна полягає у доповнені існуючої інформаційної технології формування навантаження людини в частині врахування специфіки діяльності людини.

Ключові слова: інформаційна система, інтелектуальна система, аналіз, планування, класифікація, навантаження.

ЗМІСТ

ВСТУП

Зважаючи на посітйно зростаючий рівень навантаження людини професійними, соціальними задачами доцільним є розроблення інформаційної інтелектуальної системи для формування оптимального з точки зору енергетичних витрат людини графіку задач.

Використання технологій інтелектуального аналізу дозволить сформувати об'єктивно оптимальний розклад навантаження, враховуючи специфічні особливості діяльності кожної людини. Реалізація доповненого алгоритму в архітектурі android-додатку враховує сучасні тенденції тотального впровадження інформаційних технологій в усі сфери людського буденного та професійного життя.

Об’єктом дослідження виступає процес розподілу навантаження людини. Предмет дослідження - алгоритм оптимального розподілу різних видів навантаження з урахуванням специфіки діяльності людини.

Метою дослідження є розробка інтелектуальної системи планування навантаження людини з урахуванням специфіки її діяльності.

Теоретичні дослідження роботи зосереджені на аналізі методів формування розкладу навантаження людини, які були використані при розробці програмних аналогів. Основна мета теоретичного дослідження полягає у розробленні алгоритму формування навантаження людини на основі даних літературних джерел тематики дослідження.

Експериментальні дослідження полягають у перевірці адекватності розробленого алгоритму формування навантаження людини з урахуванням специфіки її діяльності, шляхом тестування системи, викристовуючи її для організації розкладу та планування навантаження тестувальників.

Наукова новизна полягає у доповненні існуючої інформаційної технології формування навантаження людини в частині врахування специфіки діяльності людини.

Практична значимість полягає у розробленні інтелектуальної системи автоматизованого формування навантаження людини з урахуванням специфіки її діяльності.

Використання розробленої системи дозволить оптимальним чином розподілити задачі людини для контролю енергетичних витрат та уникнення "вигорання".

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність дослідження

Керування часом, тайм-менеджмент (від англ. time management) — сукупність методик оптимальної організації часу для виконання поточних задач, проектів та календарних подій. Типовими підходами в керуванні часом є постановка пріоритетів, розбиття великих завдань та проектів на окремі дії та делегування іншим людям. До керування часом належать також методи впливу на мотивацію та контролю результатів. По темі менеджменту часу часто проводяться психологічні тренінги. Головними допоміжними інструментами для керування часом є особистий календар, список поточних завдань та список проектів. Механізми для керування часом (календар та список задач з можливістю їх пріоритизації та категоризації) реалізовані в комп'ютерних програмах таких як Microsoft Outlook, iCal а також у сучасних мобільних додатках[14].

Планування надає такі переваги :

1. Планування дисциплінує.
2. Планування спрощує робочий процес.
3. Планування робить людину більш ефективною.
4. Планування знижує рівень стресу.
5. Планування розвиває пунктуальність.
6. Планування вивільняє вільний час.
7. Планування впорядковує справи.
8. Планування звільняє мозок.
9. Планування сприяє натхненню.
10. Планування сприяє досягненню цілей [15].

Актуальність теми дослідження пов’язана з нераціональним плануванням свого часу людиною та відповідно її перенавантаженням.

Перевантаження - це термін, що використовується для опису виконання занадто великої роботи. Це може означати перевантаження себе занадто великою кількістю роботи або ж занадто важкою роботою. Перевантаження може бути як від фізичної, так і від розумової діяльності, тому людина може бути виснажена як фізично, так і емоційно.

Ознаками перенавантаження є :

1. Втома - проявляється багатьма способами, включаючи відсутність енергії, втрату сну, постійне виснаження.
2. Тривога - виникає внаслідок перевантаження в результаті стресу, відсутності сну і психічного стану
3. Безсоння - перевантаження може викликати безсоння внаслідок стресу
4. Гнів - перевантаження, як правило, проявляється через надмірну дратівливість, напруженість або вибухи гніву.
5. Депресія - перевантаження може викликати почуття безнадійності і страху, що призводить до того, що людина відчуває себе в пастці депресії.
6. Апатія - перевантаження може викликати почуття безнадійності, песимізму та нерухомості.

Боротися з перевантаженням важче, ніж його попередити. Людина має слідкувати за своїм психічним та фізичним здоров’ям. Кожен повинен мати достатній час, щоб відновити себе та підвищити свою продуктивність[1].

Одним із способів контролю перевантаження є складання списку справ або особистого розкладу [2].

1.2 Огляд існуючих аналогів

Традиційно люди використовують для планування свого часу такі додатки як MS Outlook, Google Calendar. Дані додатки виступають як місце для зберігання задач

та відповідно часу їх проведення. Людина самостійно обирає час виконання задач та контролює їх кількість,

Доповнюючи традиційні інструменти планування задач створюються додатки, що на основі методів штучного інтелекту розширюють основний функціонал попередніх. Сучасні системи здатні самостійно формувати розклад людини, запам’ятовувати звички, аналізувати попередні задачі, що дає змогу назвати тип даних додатків особистим асистентом. Суміжним типом додатків є системи тайм-менеджменту для команд, що додатково надають змогу контролювати співробітників та команду вцілому.

Далі представлений опис додатків, пов’язаних за тематикою з керуванням часом та задачами, що використовують у своєму функціоналі методи штучного інтелекту.

1) Timeful

Додаток, розроблений під керівництвом вчених в області штучного інтелекту Йоава Шохама та Якоба Бенка та відомого вченого в області психології та поведінкової економіки Дена Аріелі, що функціонує як традиційний календар, проте також рекомендує оптимальні слоти для конкретних подій.

Якщо перед людиною стоїть задача піти в продуктовий магазин, та вона має вільний час між 18:00. і о 20:00 у понеділок, додаток може запропонувати їй саме цей вільний час. Запропонований час може бути прийнятий, у такому випадку він відображатиметься у календарі, змінено або відхилено, у такому випадку з'явиться альтернативний запропонований часовий інтервал.

Коли ви вперше починаєте роботу над додатком, Timeful робить пропозиції, що базуються на середніх показниках, наприклад, що статистично, люди найбільш продуктивні вранці і відповідно вирішують справи вранці. У той час як користувач вводить більше даних у систему, особисті моделі домінують у рекомендаціях та система дізнається, що людина прийняла і відкинула, і з часом зробить кращі пропозиції[3].

Додаток має зручний інтерфейс для перегляду добового списку задач (рис.1.1) та контролю діяльності впродовж місяця(1.2).

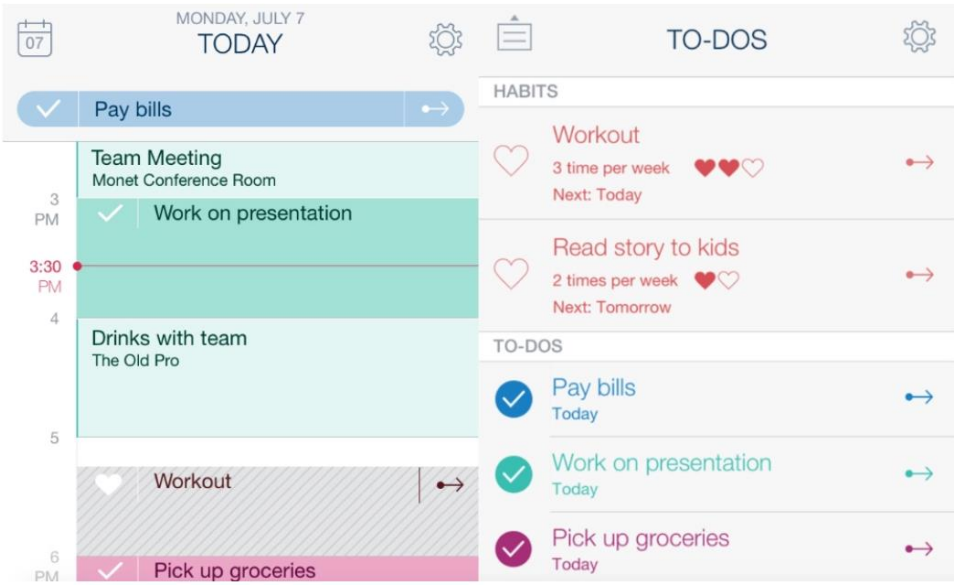


Рисунок 1.1 – Вікно додатку Timeful (Перегляд добового списку справ)

Основною функцією Timeful є «розумні» пропозиції планування: вказавши тимчасові рамки для виконання завдання, використовуючи алгоритми оптимізації та поведінкової науки, Timeful пропонує інтервали для завдань, щоб забезпечити його виконання[6].

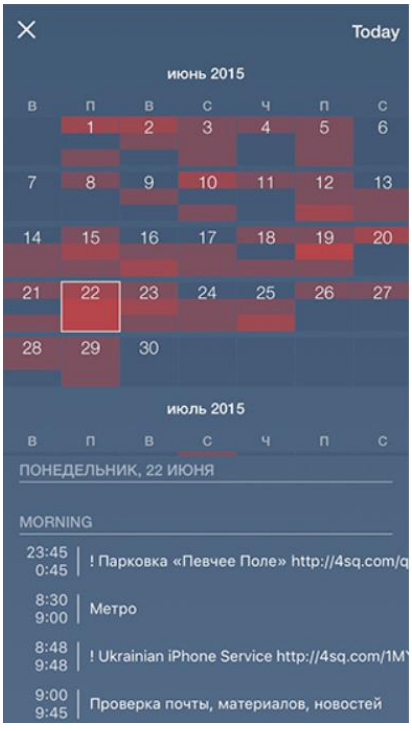


Рисунок 1.2 – Вікно додатку Timeful (Відображення місячного навантаження)

2) Selfplanner

SELFPLANNER - розгорнутий веб-інтелектуальний додаток-календар, який допомагає користувачу планувати в часі та просторі свої індивідуальні завдання. На відміну від інших асистентів, які концентруються на автоматизації планування зустрічей, SELFPLANNER підкреслює планування індивідуальних завдань і подій.

SELFPLANNER підтримує прості, переривчасті та гнучкі періодичні завдання, довільні часові проміжки, обмеження над частинами перериваного завдання, двійкові обмеження між завданнями, перевагами часових проміжків над іншими, посилання на місцеположення, класи розташувань, часові пояси тощо.

SELFPLANNER інтегрується з Календарем Google і додатком на основі Карт Google (рис.1.3). Він запроваджує інноваційний спосіб визначення часових проміжків на основі визначених користувачем правил. Основою системи є Squeaky Wheel Optimization алгоритм поєднаний з ефективною евристикою. SELFPLANNER є кроком на шляху до наступного покоління інтелектуальних додатків календаря. [10].

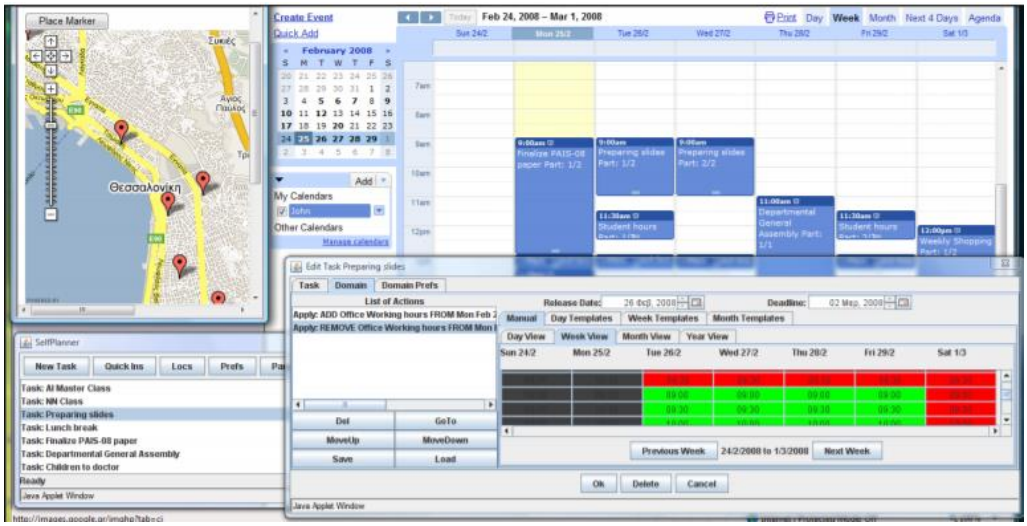


Рисунок 1.3– Вікно додатку SELFPLANNER

3) Time

Time - додаток, що дозволяє більш нав'язливо контролювати завдання під рукою з кольоровим таймером, який змінюється від зеленого до жовтого та до червоного, коли виділений час для даного завдання закінчується.

На перший погляд програма не виглядає дуже складною, хоча приховує технології штучного інтелекту. Чим більше користувач використовуєте Time, тим розумнішим він стає. Підсумковий перегляд покаже всі завдання, що були зроблені, а технологія Time надасть пропозиції щодо продуктивності на основі вашого попереднього використання.

Аналіз діяльності користувача починається після того, як виконується певне завдання кілька разів. Алгоритми використовують лінгвістичне розпізнавання, щоб зв'язати назви завдань разом, навіть якщо вони не написані однаково. Наприклад, “code app” and “work on app” можна вважати тим самим елементом[7].

AI програми починає працювати з часом, дізнаючись, як ви працюєте, і пропонуєте індивідуальні пропозиції щодо продуктивності. Користувач може отримати візуалізацію того, скільки часу він працював та додавав кожному завданню. Це допоможе краще зрозуміти завдання, які були виконані швидко, і ті, які потребували додаткового часу[8]. Якщо користувач завжди відставав від своїх цілей у часі, програма може запропонувати додати певну кількість додаткових хвилин до завдання наступного разу (рис. 1.4).

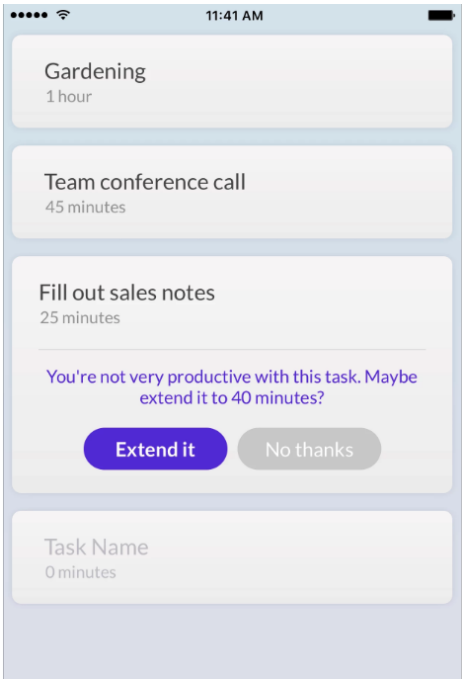


Рисунок 1.4 – Додаток Time, пропозиція збільшити час

4) Todoist

Todoist - планувальник завдань, що має функцію штучного інтелекту, що називається Smart Schedule.

Smart Schedule використовує інтелектуальне моделювання, щоб допомогти користувачу легко спланувати свої завдання на день і тиждень. Вона вивчає особисті звички та враховує моделі всіх користувачів Todoist, щоб передбачити найкращі терміни виконання завдань.

Це означає, що завдання, які планує користувач, можуть бути швидко перепланові в масовому порядку, тоді як нові та позапланові завдання можуть бути легко призначені для кращих термінів.

При пошуку ідеальних термінів, Smart Schedule враховує:

- Звички - Smart Schedule ознайомиться з звичками та відповідно запропонує дати. Усі особисті дані обробляються автоматично алгоритмом інтелектуального розкладу
- Терміновість завдання
- Робочі дні в порівнянні з вихідними - Smart Schedule дізнається, які типи завдань можна виконувати у вихідні дні, і які завжди повинні бути заплановані протягом тижня
- Баланс - Smart Schedule спробує збалансувати навантаження на завдання протягом наступних 7 днів відповідно, та не перевантажувати одни день в порівнянні з іншим (рис. 1.5).
- Щоденні та щотижневі цілі - Todoist дозволяє встановлювати та відстежувати цілі за кількістю завдань, які потрібно виконувати кожен день і тиждень. Розумний графік рекомендуватиме терміни, які допоможуть вам досягти конкретних цілей.

Користувач завжди матимете змогу приймати, редагувати або відхиляти пропозиції Smart Schedule. Як і для всіх функцій, що працюють на основі AI, які з часом «навчаються», прогнозування дат смарт-розкладу ставатиме більш точними, чим більше користуються Todoist [16].

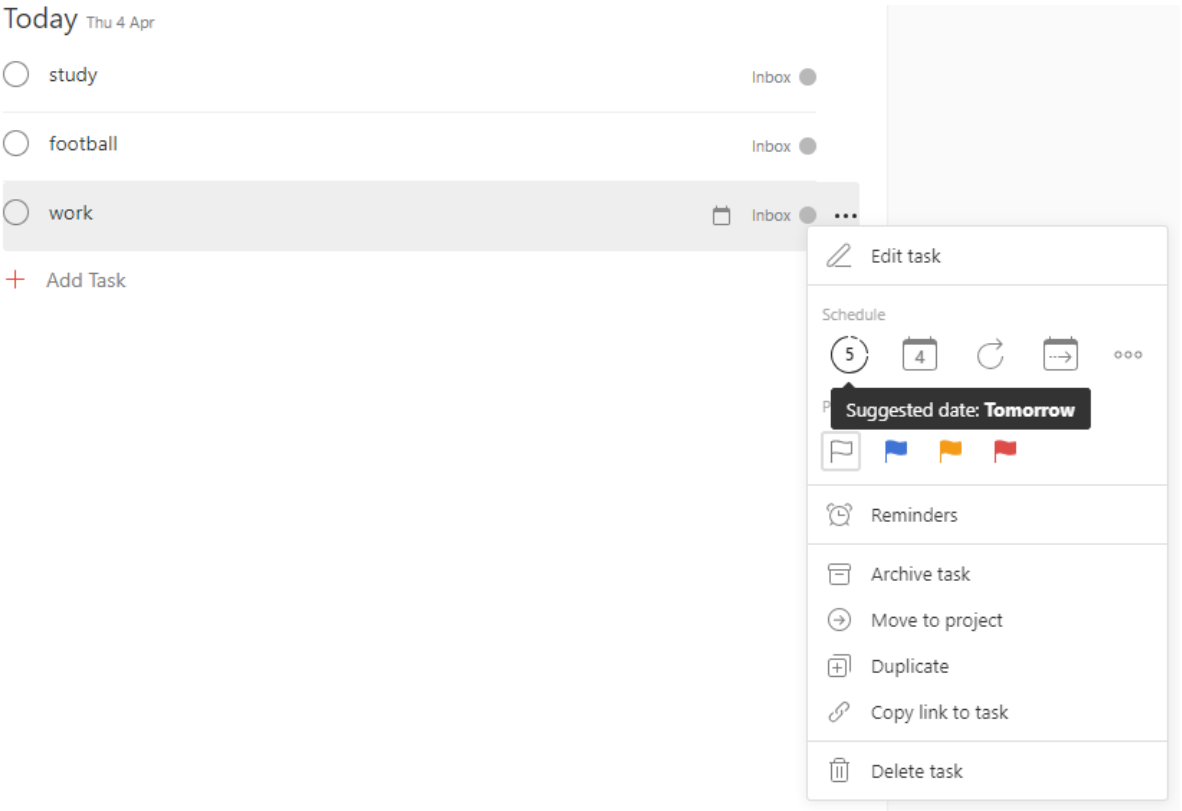


Рисунок 1.5 – Додаток todoist, пропозиція змінити день виконання

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою даного проекту є розробка інтелектуальної системи планування навантаження людини з урахуванням специфіки її діяльності.

Для досягнення такої комплексної мети та розробки системи були визначені задачі, які потрібно реалізувати:

- Огляд існуючих досліджень за темою навантаження людини
- Огляд існуючих досліджень за темою створення планування розкладів та задач
- Огляд існуючих методів штучного інтелекту, що застосовуються для планування
- Моделювання системи: проектування бази даних, створення алгоритмів, проектування інтерфейсу
- Розробка системи
- Тестування
- Введення в користування

2.2 Методи дослідження

Теоретичні дослідження роботи зосереджені на аналізі методів формування навантаження людини, які були використані при розробці програмних аналогів. Основна мета теоретичного дослідження полягає у розробленні алгоритму формування навантаження людини на основі даних літературних джерел тематики дослідження.

Друга мета теоретичного дослідження полягає у систематизації та аналізі відомої інформації про формування правил розрахунку коефіцієнтів навантаження для різних видів людської активності. Останнє має бути враховано при доповненні існуючих алгоритмів формування розкладу.

Доцільним також є огляд та аналіз прийомів тайм-менеджменту з метою пошуку можливих залежностей енергетичних втрат людини при різних видах активностей та рекомендацій для найбільш оптимальної комбінації цих видів.

Експериментальні дослідження полягають у перевірці адекватності розробленого алгоритму формування навантаження людини з урахуванням специфіки її діяльності. Результати експериментального дослідження оформлені у вигляді таблиць з відповідями оцінки роботи системи.

Вибір методу визначення класу навантаження

Задачею системи є визначення класу навантаження окремого дня людини, тобто поставлена задача класифікації. Задача відноситься до розділу навчання з вчителем адже при першому вході користувача в систему створюється навчаюча вибірка з числових значень добової кількості калорій та відповідних п’яти класів.

Було проаналізовано наступні основні методи вирішення задачі класифікації є:

1) Наївний байєсів класифікатор

Метод класифікації, заснований на наївному байєсівської класифікаторі, є алгоритмом навчання з учителем, в якому застосовується теорема Байеса із суворим (наївним) припущенням про незалежність між кожними парами ознак [11]. Припущення про незалежність дозволяє позбутися від складної схеми оцінки параметрів класифікатора. Це дозволяє застосовувати алгоритм на великих вибірках. також класифікація виявляється досить точною: недостатньою для високоточних систем класифікації, однак задовільною для грубої оцінки та порівняння з іншими алгоритмами. Незважаючи на їхні надмірно спрощені припущення, наївні класифікатори Байеса працювали досить добре у багатьох реальних ситуаціях, класифікація документів та фільтрація спаму.

2) Дерево ухвалення рішень

Дерево ухвалення рішень (також можуть називатися деревами класифікацій або регресійними деревами) — використовується в галузі статистики та аналізу даних для прогнозних моделей. Структура дерева містить такі елементи: «листя» і «гілки». На ребрах («гілках») дерева ухвалення рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах — атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення. Подібні дерева рішень широко використовуються в інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка прогнозує значення цільової змінної на основі декількох змінних на вході[4].

3) K-найближчих сусідів

Метод K-найближчого сусіда - один з методів вирішення задачі класифікації. Передбачається, що вже є якась кількість об'єктів з точною класифікацією (тобто для кожного них точно відомо, якого класу він належить). Потрібно виробити правило, що дозволяє віднести новий об'єкт до одного з можливих класів (тобто самі класи відомі заздалегідь) [9].

В основі k-NN лежить таке правило: об'єкт вважається належним того класу, до якого належить більшість його найближчих сусідів. Під «сусідами» тут розуміються об'єкти, близькі до досліджуваного в тому чи іншому сенсі. В якості метрики найчастіше обирається евклідова метрика через її простоту та зрозумілість.

До недоліків метричних алгоритмів можна віднести зберігання всієї навчальної вибірки[5].

Оскільки вхідними даними для класифікації добового навантаження є числові значення, вибірка поповнюється даними – оцінками користувачів та необхідно приближувати дані класифікації до відгуків користувача, то було вирішено використовувати метод K-найближчого сусіда для задачі класифікації.

Метод контролю енергетичних витрат

Планування розкладу людини буде обмежене контролем за навантаженням рівнянням Харріса-Бенедикта (також називається принципом Харріса-Бенедикта) - методом, що використовується для оцінки рівня базального метаболізму людини (BMR). Оцінене значення BMR може бути помножено на число, що відповідає рівню активності індивіда (специфіки діяльності); отримане число - приблизний добовий прийом та витрата кілокалорій для підтримки поточної маси тіла[12], рівняння представлені на рис. 2.1.

Sex	Units	Calculation
Men	Metric	BMR = 66.5 + (13.75 × weight in kg) + (5.003 × height in cm) – (6.755 × age in years)
	Imperial	BMR = 66 + (6.2 × weight in pounds) + (12.7 × height in inches) – (6.76 × age in years)
Women	Metric	BMR = 655.1 + (9.563 × weight in kg) + (1.850 × height in cm) – (4.676 × age in years)
	Imperial	BMR = 655.1 + (4.35 × weight in pounds) + (4.7 × height in inches) - (4.7 × age in years)

Рисунок 2.1 - Рівняння Харріса-Бенедикта

Хоча оригінальний документ не намагається перевести BMR у загальні витрати енергії (TEE), результат BMR може бути помножений на коефіцієнт, який апроксимує рівень фізичної активності людини для оцінки їх TEE. Наступна таблиця, зображена на рис. 2.2 дає змогу наблизити щоденний TEE на основі специфіки діяльності людини [13].

Lifestyle	Example	PAL	Calculation
Sedentary or light activity	Office worker getting little or no exercise	1.53	BMR x 1.53
Active or moderately active	Construction worker or person running one hour daily	1.76	BMR x 1.76
Vigorously active	Agricultural worker (non mechanized) or person swimming two hours daily	2.25	BMR x 2.25

Рисунок 2.2 - Загальні витрати енергії залежно від специфіки діяльності

2.3 Інструменти реалізації

1) Вибір типу продукту

Сучасні it-продукти можна розподілити за категоріями – web-сайти, web-додатки, desktop-додатки, мобільні додатки. Оскільки основною ознакою розроблюваної системи є постійна доступність, то найкращим варіантом є використання смартфону для доступу. Web-сайти та web-додатки вимагають доступу до мережі, що може бути не постійним, а також сповільнює роботу в порівнянні з offline мобільним додатком. Тому було вирішено реалізувати систему у вигляді мобільного додатку.

2) Вибір платформи – Android, IOS,

На сьогоднішній день iOS і Android є двома основними мобільними операційними системами [17].

Ці дві системи було порівняно за наступними критеріями:

1. Кількість користувачів і середніх користувачів кожної платформи

a) Частка ринку

Відомо, що єдиний виробник виробляє пристрої для iOS - Apple. Але є тисячі малих і великих компаній, які роблять пристрої для Android. Ця конкуренція знижує ціни, що призводить до збільшення частки ринку дешевими телефонами Android. Statcounter випустила графік, що відображає цю ситуацію на ринку (рис. 2.3). В усьому світі близько 75% людей використовують Android, і лише 19% використовують iOS.

Рисунок 2.3 – Частка виробників на ринку

b) Демографічні дані користувачів

Нижче наведено карту, яка надасть вам більше інформації про налаштування платформи по всьому світу (рис 2.4).

Рисунок 2.4 – Демографічна карта платформ

Ця карта показує, що багато країн Європи, Південної Америки, Азії та Африки віддають перевагу Android. Деякі країни з вищими доходами, включаючи США, деякі європейські країни та Австралію, віддають перевагу iOS. Ці регіональні переваги можна частково пояснити низькою вартістю деяких мобільних телефонів Android.

У середньому люди, які віддають перевагу iOS, молодші за людей, які віддають перевагу Android, мають вищий рівень освіти і заробляють більше грошей.

2. Складність розробки

a) Устаткування

Розробник з комп'ютером - може створити програму для Android. Linux, Windows, і навіть пристрої Mac можуть робити цю роботу. Розробка програми iOS вимагає від розробника Mac.

b) Сумісність

Немає сенсу вибирати старі технології для сучасних проектів. Нові мови програмування Kotlin і Swift замінюють Java і Objective-C.

Ось коли на перше місце виникають питання сумісності. Котлін на 100 відсотків сумісний з Java. Ця повна сумісність означає, що ви можете використовувати всі численні фреймворки та бібліотеки для Java в проекті Котліна. Більше того, ви можете переключитися з однієї мови програмування на іншу з рядка в лінію

З Swift речі не виглядають так добре. Objective-C та Swift не є повністю сумісними. Це створює безліч проблем і ускладнює розвиток. Крім того, кожна версія коду має різну сумісність, тому один фреймворк може бути більш сумісним з Swift 2, ніж з Swift 3.

Стисле порівняння платформ приведене на рис. 2.5.

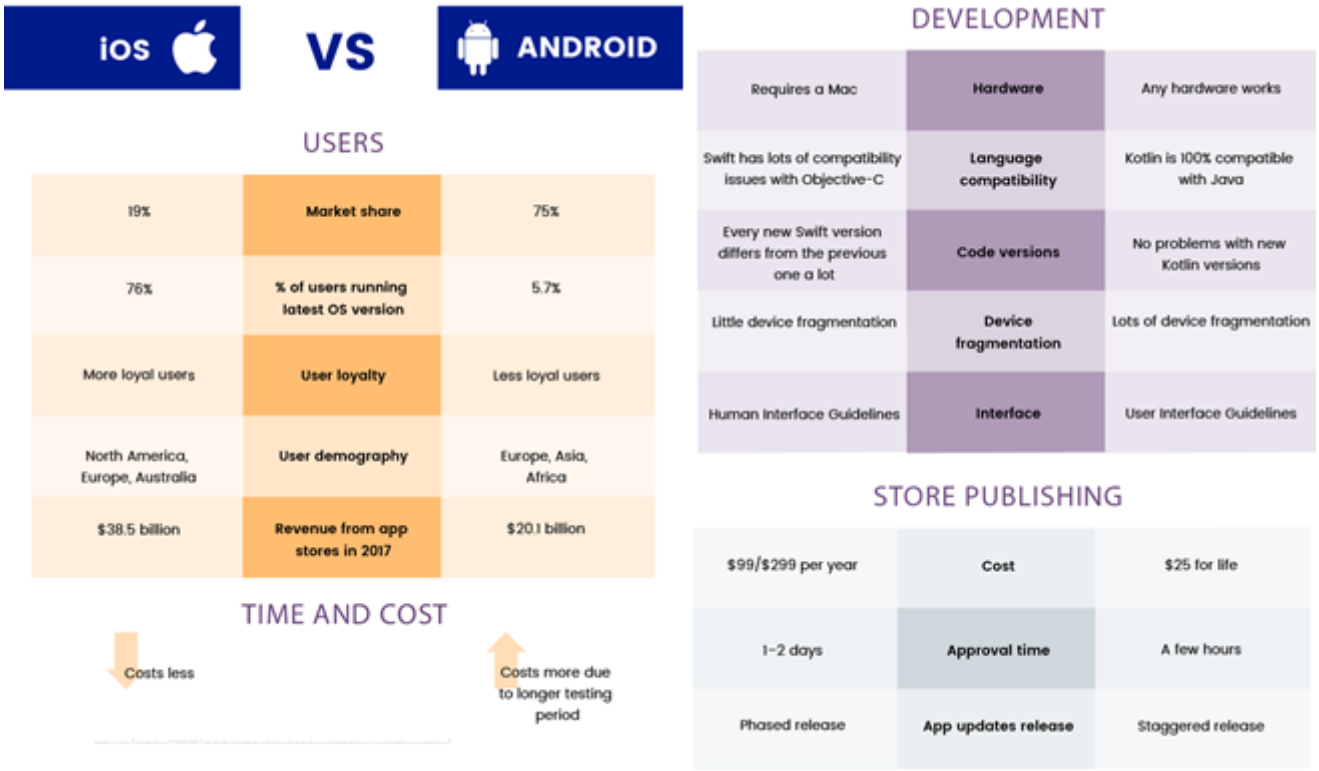


Рисунок 2.5 – Порівняння платформ

Висновок:

Оскільки Android є більш поширеною платформою, не потребує додаткового hardware та розробник має навички java програмування, Android було обрано платформою розробки.

3) Вибір технологій збереження інформації

В Android-пристроях інформація може зберігатися в:

- SharedPreferences

SharedPreferences потрібно використовувати для збереження примітивних даних у парах ключ-значення. SharedPreferences складаєтья з ключа, який повинен бути String, і відповіного значення для цього ключа, яке може бути одним з типів: boolean, float, int, long або string. Внутрішньо платформа Android зберігає спільні налаштування програми у файлі xml у приватному каталозі. Програма може мати кілька файлів спільних налаштувань. В ідеалі SharedPreferences використовуються для збереження налаштувань програми [18].

- Внутрішнє сховище

Метод зберігання даних внутрішньої пам'яті спеціально призначений для тих ситуацій, коли потрібно зберігати дані до файлової системи пристрою, але щоб користувачі не читали ці дані. Дані, що зберігаються за допомогою методу внутрішнього зберігання, є повністю приватними для програми та видаляються з пристрою, коли програма видаляється.

- Зовнішнє сховище

І навпаки, існують інші випадки, коли користувач може переглядати файли та дані, збережені додатком. Щоб зберегти файли на зовнішньому сховищі пристрою, програма повинна запросити дозвіл WRITE_EXTERNAL_STORAGE. Якщо потрібно лише прочитати з зовнішнього сховища без запису, запит на дозвіл READ_EXTERNAL_STORAGE. Дозвіл WRITE_EXTERNAL_STORAGE надає доступ для читання / запису. Однак, починаючи з Android 4.4, ви можете записати в приватну папку зовнішньої пам’яті без запиту WRITE_EXTERNAL_STORAGE. Папку "приватна" можна читати іншими програмами та користувачем, однак дані, що зберігаються в цих папках, не скануються сканером. Ця папка app_private розташована в каталозі Android / даних, а також видаляється після видалення програми.

Починаючи з Android 7.0, програми можуть запитувати доступ до певного каталогу, а не вимагати доступу до всього зовнішнього сховища. Таким чином, ваша програма може, наприклад, запитувати доступ до каталогу лише фотографій або до каталогу документів. Це називається доступом до каталогу областей.

- Підключення до мережі

Підключення до мережі не є методом зберігання даних, але може бути способом збереження даних для конкретного користувача, якщо пристрій підключено до Інтернету, використовуючи деяку аутентифікацію. Ви повинні балансувати між завантаженням даних кожен раз, коли додаток має потребу в ній, або з одноразовою синхронізацією даних, що в кінцевому підсумку призведе до іншого варіанту зберігання, згаданого вище.

- Бази даних SQLite

SQLite - компактна вбудована СУБД [19]. Слово «вбудована» (embedded) означає, що SQLite не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а являє собою бібліотеку, з якої програма компонується, і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID-функції досягаються в тому числі за рахунок створення файлу журналу.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

Висновок:

Оскільки система буде використовувати структуровані дані та управляти ними, для зберігання інформації було обрано бази даних SQLite.

4) Вибір середовища реалізації - Android studio

Найпопулярнішим середовищем реалізації android-додатків є Android studio. Він є офіційним IDE (інтегрованим середовищем розробки) для Android, що робить його вибором номер один для більшості розробників[20].

Переваги використання Android studio:

- Проста інтеграція

Після того, як Google представив Android Studio, компанія почала зосереджувати свою підтримку на цій платформі. Користувач отримує просту інтеграцію між інструментами потрібними для розробки мобільних додатків

- Краще проектування та розповсюдження

Інструменти Android, такі як Android Studio, мають краще розповсюдження робочого простору, що допомагає зробити дружнє середовище розробки додатків. Наприклад, розповсюдження робочої області Android Studio є більш природним, ніж робоча область Eclipse, платформа, яка зазвичай отримує погані відгуки про дизайн та інтерфейс.

- Все в одному

Офіційні інструменти Android надають необхідну інформацію, необхідну для мислення, розробки та тестування вашої програми, перш ніж завантажувати її в Google Play: відладчики, емулятори, навчальні посібники або широку документацію, щоб покращити процес розробки та оптимізувати час створення проекту.

- Оновлення

Кожні оновлення та нові (або фіксовані) функції, які Android оголошує для розробки додатків, будуть показані спочатку через їхні офіційні інструменти, яким є Android Studio.

- Розробники Android мають кілька варіантів для створення своїх проектів. Кожен з них має свої плюси і мінуси, однак, рекомендується використовувати офіційні інструменти Android для того, щоб мати кращу підтримку і ефективність для створення проектів [21].

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Аналіз варіантів використання системи

Діаграма прецедентів — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

В розроблюваній системі єдиним можливим актором є користувач додатку. Варіантами використання даної системи є її 4 основні функції. Дві з них, а саме Calendar та to-do list, тобто використання сисеми, як звичайного календаря або ж записника справ є звичайними та поширинеми, мають достатньо аналогів.

Дві інші – “Task Scheduling” та “Analyze scheduling spending” є особливими прецедентами, що виділяють систему з поміж її аналогів. Система здатна аналізувати навантаження людини та планувати її завдання в оптимальний з точки зору навантаження час.

Діаграма варіантів використання представлена на рис. 3.1.

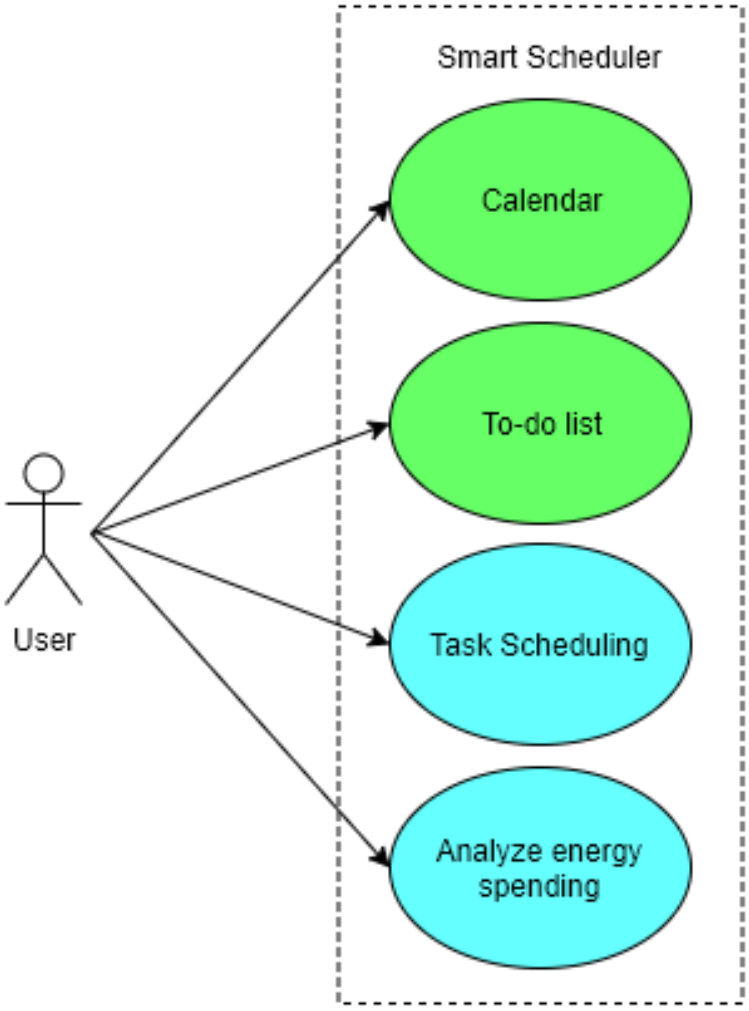


Рисунок 3.1 – Даграма варіантів використання

3.2 Архітектура системи

Архітектура інформаційної системи - це концепція, що визначає модель, структуру, виконувані функції й взаємозв’язок компонентів інформаційної системи [24]. На даному етапі розроблена інтелектуальна система має просту архітектуру, що складається з двох взаємодіючих компонентів – мобільного додатку, що включає в себе сторонні бібліотеки та sqlite бази даних. Схема архітектури системи зображена на рис. 3.2.

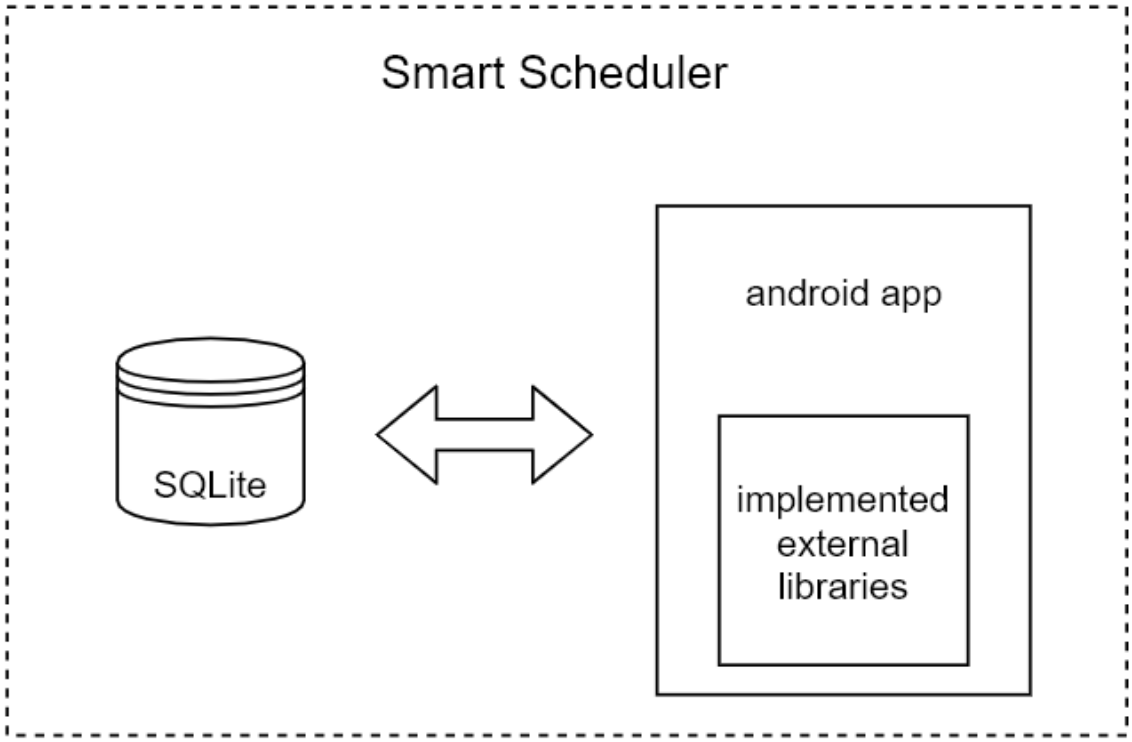


Рисунок 3.2 – Схема архітектури системи

3.3 Моделювання даних

База даних (англ. database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування [25]. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

3.3.1 Концептуальне моделювання

Першим кроком було проведене концептуальне проектування бази даних. Концептуальна модель - це модель, представлена безліччю понять і зв'язків між ними, що визначають смислову структуру розглянутої предметної області або її

конкретного об'єкта [26]. Концептуальна модель включає високорівневі конструкції даних та може бути не нормалізованою.

Основною одиницею розробленої системи є завдання або ж задача. Саме заавдання мають зберагтися в базі даних. Користувач буде виконувати операції додавання, редагування та видалення над ними, а система буде аналізувати за допомогою них навантаження людини, тому була виділена сутність “task”.

Для класифікації навантаження дюдини необхідно знати та відповідно зберігати її параметри, за якими розраховується її середнє навантаження, тому була виділена сутність “parameters”.

Оскільки було вирішено, що користувач матиме змогу оцінювати сумарне навантаження саме дня, а не завдання та для подальшої підтримки швидкодії додатку, виділено сутність “analysis”.

Таким чином на етапі концептуального моделювання виділено три сутності “task”, “parameters”, “analysis”.

3.3.2 Логічне моделювання

Логічні моделі даних подають абстрактну структуру області інформації. Вони часто мають схематичний характер і найтиповіше використовуються у бізнес-процесах, які прагнуть захопити речі, що мають важливе для організації значення, та як вони відносяться одна до одної. Одного разу перевірена та схвалена, логічна модель даних може стати основою фізичної моделі даних і сформувати дизайн бази даних.

Логічні моделі даних повинні засновуватися на структурах, визначених у попередній концептуальній моделі даних, оскільки вона описує семантику інформаційного контексту, яку логічна модель повинна також відображати.

Оскільки логічна модель передбачає реалізацію на конкретній обчислювальній системі, вміст логічної моделі даних коригується для досягнення певної ефективності [27].

Логічна модель даних включає сутності (таблиці), атрибути (колонки / поля) та відношення (ключі) та є нормалізованою.

На етапі логічного моделювання було вирішено нормалізувати сутність “task” та виділити сутності “type” та “sort” для подальшого групування та аналізу. Для кожної сутності були виділені атрибути та визначено взаємозв’зок між сутностями. Атрибути таблиць та їхній опис представлено в табл. 3.1 – 3.5.

Таблиця 3.1 – Опис атрибутів таблиці “sort”

Назва	**Опис**
_id	Первинний ключ таблиці sort, який однозначно ідентифікує запис; зв’язаний з зовнішнім ключем в таблиці type
name	Назва сорту, було визначено три сорти: фізичну, розумову, активність без витарт енергії.

Таблиця 3.2 – Опис атрибутів таблиці type

Назва	**Опис**
_id	Первинний ключ таблиці Packager, який однозначно ідентифікує запис; зв’язаний з зовнішнім ключем в таблиці task
name	Назва типу активності
energy	Витрата калорій за годину за даним типом
sort_id	Містить Id запису з таблиці sort; зовнішній ключ

Таблиця 3.3 – Опис атрибутів таблиці task

Назва	**Опис**
_id	Первинний ключ таблиці task, який однозначно ідентифікує запис
name	Назва завдання
duration	Продовжуваність завдання
energy	Витрачена енергія в калоріях

Продовження таблиці 3.3

day	День дати виконання
month	Місяць дати виконання
year	Рік дати виконання
hour	Година старту завдання
minute	Хвилина старту завдання
key_id_type	Містить Id запису з таблиці type; зовнішній ключ

Таблиця 3.4 – Опис атрибутів таблиці parameters

Назва	**Опис**
_id	Первинний ключ таблиці parameters, який однозначно ідентифікує запис; зв’язаний з зовнішнім ключем в таблиці Package, Leaves, ProjectUsers
male	Стать користуача
age	Вік користувача
weight	Вага користувача
height	Зріст користувача
KOEF	Коефіцієнт, що відображує життєву активність

Таблиця 3.5 – Опис атрибутів таблиці analysis

Назва	**Опис**
_id	Первинний ключ таблиці analysis, який однозначно ідентифікує запис;
year	Рік
month	Місяць
day	День
total_energy	Сумарна енергія
mark_total_energy	Оцінка користувача

Логічна схема бази даних зображена на рис. 3.3.

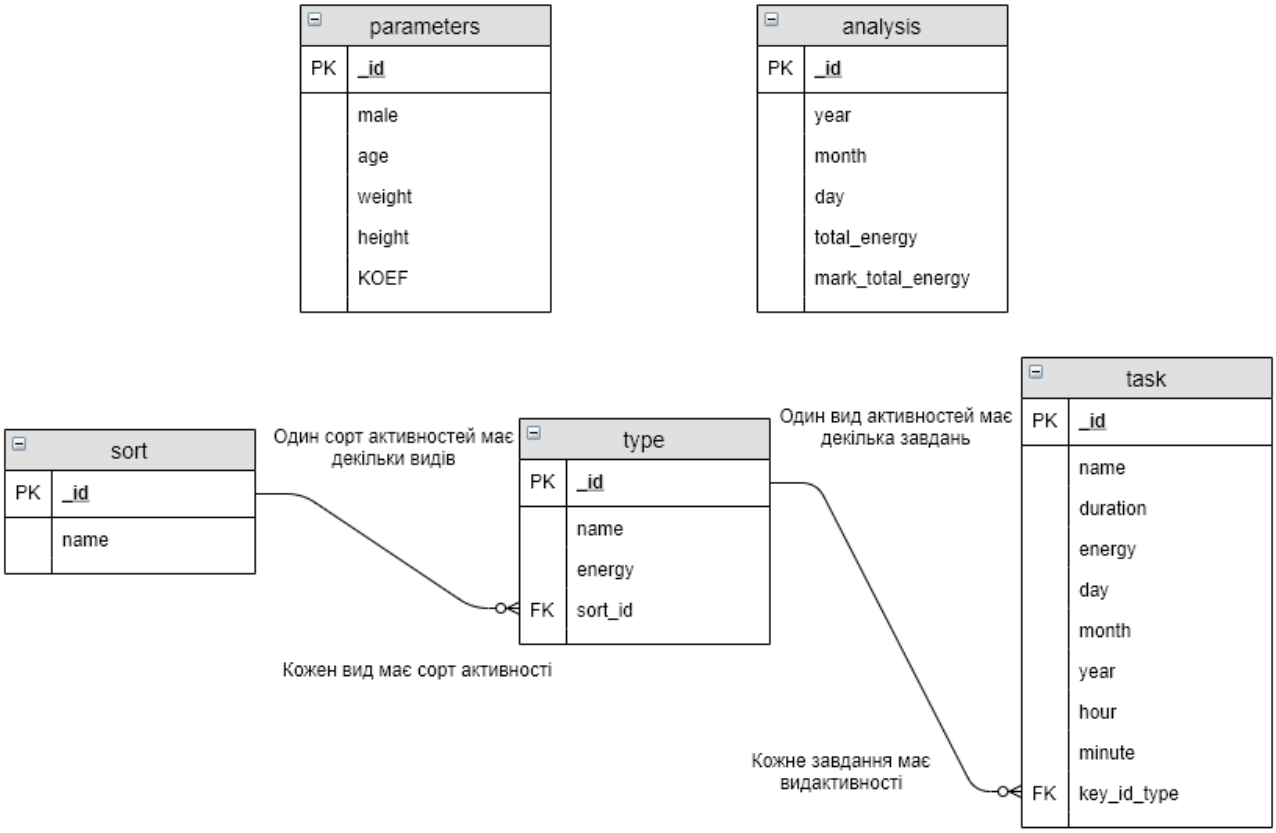


Рисунок 3.3 – Логічна схема бази даних

3.3.3 Фізичне моделювання

Фізична модель даних — подання дизайну даних як реалізованого чи призначеного для реалізації у системі керування базами даних. У життєвому циклі проекту вона типово походить від логічної моделі даних, хоча вона може бути зворотно розроблена з даної реалізації бази даних. Завершена фізична модель даних включатиме всі артефакти бази даних, необхідні для створення відношень між таблицями чи для досягнення мети продуктивності, як-от індексів, визначень обмежень, зв'язаних і секціонованих таблиць або кластерів [28].

SQLite навідміну від більшості SQL движків використовує динамічну типізацію. Кожне значення, що зберігається в базі даних SQLite (або маніпулюється

механізмом бази даних), має один з таких класів зберігання, що преставлені в табл. 3.6.

Таблиця 3.6 – Таблиця типів даних в SQLite

Назва	**Опис**
NULL	The value is a NULL value.
INTEGER	Значенням є ціле число, яке зберігається в 1, 2, 3, 4, 6 або 8 байтах залежно від величини значення.
REAL	Значення є значенням з плаваючою точкою, яке зберігається як 8-байтове число з плаваючою комою.
TEXT	Значенням є текстовий рядок, який зберігається за допомогою кодування баз даних
BLOB	бінарний набір даних

Для полів таблиць бази даних були обрані типи представлені в таблицях. Були використані типи даних INTEGER, REAL, TEXT. Були застосовані ненульові значення та унікальність значень. Поля таблиць, тип їхніх значень та обмеження представлена в табл. 3.7 – 3.11.

Таблиця 3.7 – Опис атрибутів таблиці “sort”

Назва	**Опис**
_id	Integer, primary key
name	text NOT NULL UNIQUE

Таблиця 3.8 – Опис атрибутів таблиці type

Назва	**Опис**
_id	Integer, primary key
name	text NOT NULL UNIQUE
energy	Integer
sort_id	Integer NOT NULL FOREIGN KEY

Таблиця 3.9 – Опис атрибутів таблиці task

Назва	**Опис**
_id	Integer, primary key
name	text NOT NULL
duration	Integer NOT NULL
energy	Integer
day	Integer NOT NULL
month	Integer NOT NULL
year	Integer NOT NULL
hour	Integer NOT NULL
minute	Integer NOT NULL
key_id_type	Integer NOT NULL FOREIGN KEY

Таблиця 3.10 – Опис атрибутів таблиці parameters

Назва	**Опис**
_id	Integer, primary key
male	Integer NOT NULL
age	Integer NOT NULL
weight	Integer NOT NULL
height	Integer NOT NULL
KOEF	REAL NOT NULL

Таблиця 3.11 – Опис атрибутів таблиці analysis

Назва	**Опис**
_id	Integer, primary key
year	Integer NOT NULL
month	Integer NOT NULL
day	Integer NOT NULL

Продовження таблиці 3.11

total_energy	Integer
mark_total_energy	Integer

Додатково було створено унікальні індекси на поля “name” в таблицях sort та type а також індекс на поле “mark_total_energy” в таблиці analysis для оптимізації виконання запитів.

3.4 Розробка алгоритмів

Система виділяється з-поміж інших аналогів двома функціями, а саме Scheduling (планування завдань) та Analysis (класифікація навантаження).

3.4.1 Алгоритм планування завдань

Планування завдань є основною функцією розробленої системи. Було вирішено розробити евристичний алгоритм головною ціллю якого є мінімізація різниці між середнім значенням витрат енергії розрахованим за параметрами користувача та факичними витратами. Таким чином для кожного дня виконується умова представлена нижче.

$$\left|E_{avarage} - E_{fact}\right| \rightarrow min, \qquad (3.1)$$

де $E_{avarage}$ – середнє добове навантаження,

E_{fact}– фактичне сумарне навантаження дня.

Дана мінімізація може бути досягнена якщо завдання планувати на найменш завантажений день з можливих.

Не менш важливою частиною даного алгоритму є обрання часу початку події. Оскільки всі завдання проведені користувачем зберігаються в базі дааних, то можливо знайти найчастіше вживаний час для кожного завдання з певним ім’ям. Якщо користувач найчастіше ходить на роботу на 9 годину ранку, то система має

запропонувати йому саме 9 годину ранку для завдання з назвою “work”, що і означає врахування специфіки діяльності людини при плануванні окремих завдань.

Таким чином функція планування завдання працює за наступним алгоритмом.

1. Кориситувач вводить параметри завдання, а саме назву, продовжуваність, сорт та тип завдання, дати між якими має відбутися завдання
2. Відбувається пошук днів з заданого проміжку, в яких є вільний проміжок часу, що дорівнює продовжуваності задачі.
3. З результуючого набору днів обирається найменш навантажений
4. Початок завдання призначається на час першого вільного проміжку часу, що відповідає продовженості завдання
5. Проводиться пошук найчастіше використовуваного початку завдання, якщо таке завдання зустрічалося раніше
6. Якщо тайм фрейм з найбільш використовуваним початком є вільним, то початок даного завдання прирівнюється до найчастіше вживаного, якщо ні – залишається

Алгоритм планування завдань представлений на рис. 3.4.

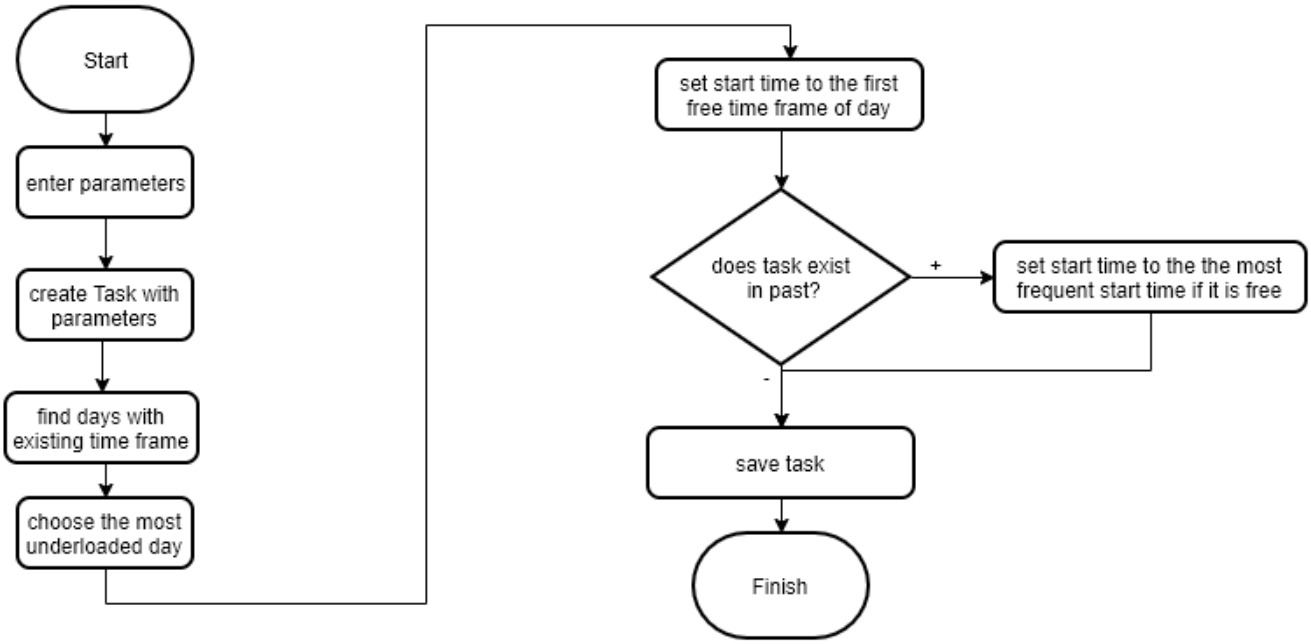


Рисунок 3.4– Алгоритм планування завдань

3.4.2 Алгоритм класифікації навантаження

Другою основною функцією системи є класифікація добового наванатження користувача. Ця фуннкція наочно надає користувачу можливість зрозуміти які дні потрібно навантажити, а які навпаки розвантажити.

Середньо-добове навантаження людини розраховується за формулою Харріса-Бенедикта. Це значення, окрім параметрів людин таких як стать, вага, зріст, вік, також залежить від типу активності людини, що відображається у відповідному коефіцієнті. Таким чином «неактивному» типу відповідає коефіцієнт 1,53, «активному» - 1,76, «спортивному» – 2,25.

Було вирішено класифікувати навантаження людини за п’ятьма класами: ненавантажений, недовантажений, нормально навантажений, перевантажений, занадто перевантажений.

Для початкової класифікації створюється стартовий набір даних наступним чином (на прикладі активного типу):

1. Після введення особистих параметрів користувачем розраховується Середньо-добове навантаження людини
2. Розраховуються значення навантажень для неактивного та спортивного типу.
3. Розраховуються проміжні значення між трьома точками та їх відзеркалення від крайніх точок, та відзеркалення крайніх проміжних точок. Таким чином створюється 5 відрізків, що відповідають класам навантаження, приклад представлений на рис. 3.5.

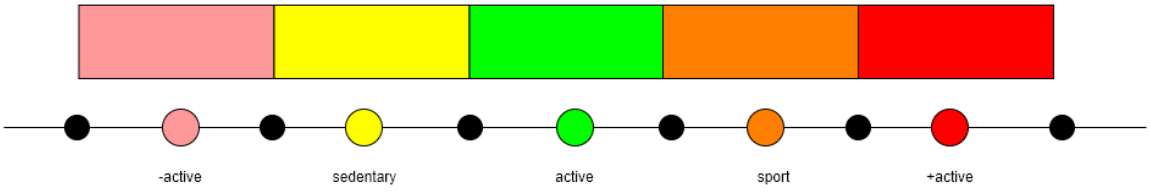


Рисунок 3.5 – Створення меж класів навантаження

4. П’ять створених відрізків діляться на 10 точок та для кожної точки призначається відповідний клас. Дані значення записуються до бази даних до таблиці «analysis».

Алгоритм створення стартового датасету представлений на рис. 3.6.

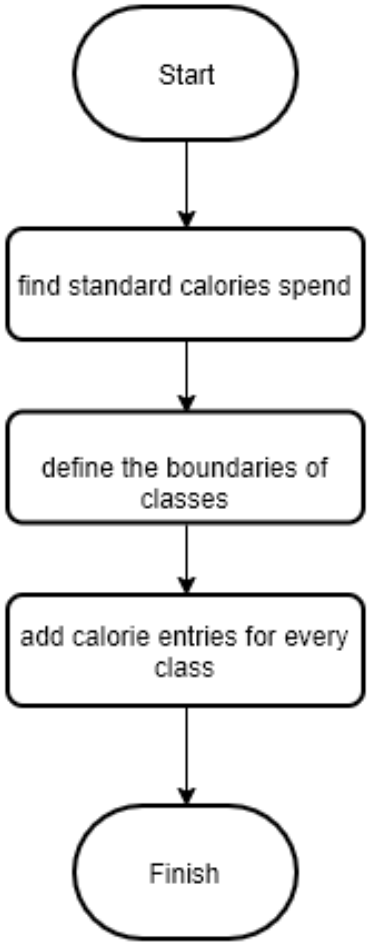


Рисунок 3.6 – Алгоритм створення стартового датасету

Всього стартовий датасет складається з 50 значень, що є мінімальним значенням для класифікації. На основі цих даних система класифікує добові навантаження людини у відповідності до п’яти класів.

Методом класифікації було обрано метод k найближчих сусідів. Класифікація кожного дня проходить за наступним алгоритмом:

1. Розраховується сумарне навантаження дня

2. Сортуються всі оцінені дні датасету за евклідовою відстанню між їхніми навантаження та навантаженням дня, що треба класифікувати
3. Розглядаються перші k відсортованих днів та відповідно обирається клас, що найбільше зустрічався

Алгоритм методу k найближчих сусідів представлений на рис. 3.7.

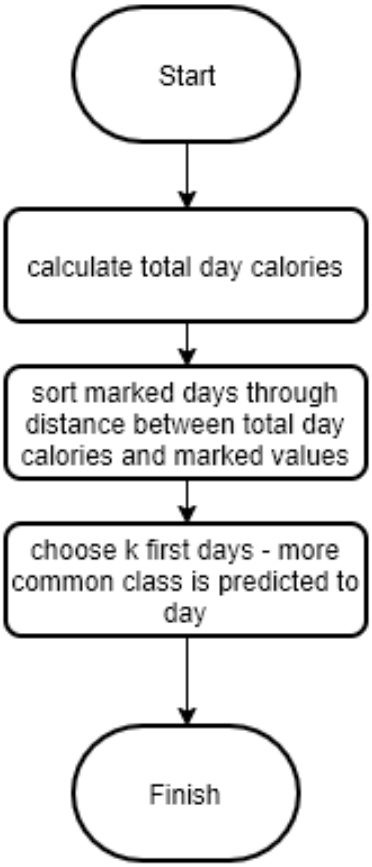


Рисунок 3.7 – Алгоритм методу k найближчих сусідів

Значення k разраховується як квадратний корінь з кількості даних датасету, як один з поширених підходів його розрахунку [29].

Оскільки середньо-добове навантаження розраховується за формулою та енерговитрати кожного завдання є табличними значеннями та можуть неточно відображати навантаженість користувача, було вирішено поповнювати датасет оцінками користувача, тим самим робити додаток ближчим та пристосованішим до конкретного користувача. Таким чином користувач має змогу оцінити кожен

попередній день оцінкою, що відповідає класам навантаження, визначених раніше та прямо впливає на роботу додатку, а саме класифікацію.

Користувач може змінювати свої параметри, такі як вік, вага, зріст, тип життєдіяльсті тому було вирішено при кожній зміні парамерів перші 50 записів таблиці бази даних «analysis» видаляються та заповнюються відповідно новими розрахованими значеннями.

3.5 Визначення енерговитрат активності

Враховуючи проведення подальших досліджень та вдосконалення створеної системи було вирішено розподілити усі завдання на 3 види: фізичної, розумової, діяльності без витрат енергії. Групування за даним параметром надає можливість аналізувати навантаження людини, та використовувати ці дані для кращого планування завдань для конкретного користуача

Розглянувши дослідження науковців медичної сфери про енерговитрати від конкретного виду діяльності, було вирішено використати дані Гарвардської медичної школи [30] для заповнення таблиці type бази даних. Енерговитрати пердставлені в табл. 3.12 – 3.14.

Таблиця 3.12 – Енерговитрати активностей виду “Physical”

Назва	**Калорії/годину**
Auto repair	224
Cooking	186
Child-care	260
Planting	298
Volleyball	224
Gymnastics	298
Walking 2.5 mlh	298

Продовження таблиці 3.12

Dancing	446
Swimmong	446
Soccer	520
Tennis	520
Weight Lifting	224
Running 5.2 mlh	670

Таблиця 3.13 – Енерговитрати активностей виду “Brain”

Назва	**Калорії/годину**
Computer work	102
Sitting in class	130
Light office work	112

Таблиця 3.14 – Енерговитрати активностей виду “No calories”

Назва	**Калорії/годину**
Sleeping	0
Etc.	0

3.6 Проектування UI

Для реалізації функціоналу було вирішено розробити 6 вікон, через які користувач взаємодіє з додатком. Опис кожного вікна представлений в таблиці 3.15.

Таблиця 3.15 – Опис вікон додатку

Назва	**Опис**
Sleeping window	Користувач зберігає час початку та час закінчення сну
User`s parameters window	Користувач вводить свої параметри: стать, вік, зріст, вагу, рівень активності та зберігає дані. На основі цих даних розраховуються дані TDEE, BMR, середніх добових енерговитрат
Main window	Головне вікно відображає календар, кожен день якого має колір в залежності від його навантаженості
Task scheduling	Користувач заповнює параметри завдання, що потрібно планувати та зберігає його, якщо запропонована дата йому підходить
Day activity window	Користувач переглядає активності дня, сумарну кількість енерговитрат, та різницю між фактичним та середньо-добовими енерговитратими , отримує інформацію про клас добового навантаження, має змогу оцінити навантаження дня
Task window	Користувач додає, редагує та видаляє завдання

При першому запуску додатку першим запускається “Sleeping window”, при кожному наступному запуску стартовим вікном є “Main window”.

На рис. 3.8. представлена схема зв’язків між вікнами розробленого додатку.

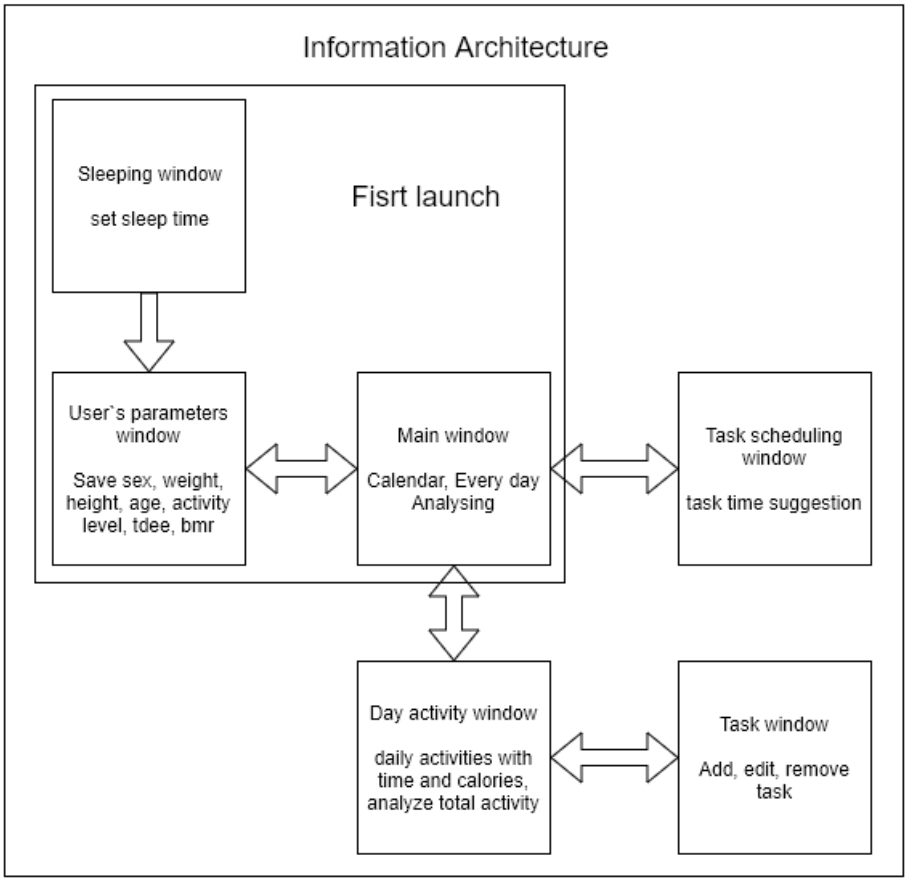


Рисунок 3.8 – Схема зв’язків між вікнами

4 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Реалізація бази даних

База даних створюється програмно при першому запуску додатку на пристрої за допомогою класу DBHelper, що наслідує стандартний клас для роботи з базами даних SQLite – SQLiteDatabase.

Наявність таблиць, поля та типи полів кожної можливо перевірити шляхом відкриття бази даних в додатку “DB browser for SQLite”, приклад представлено на рис. 4.1.

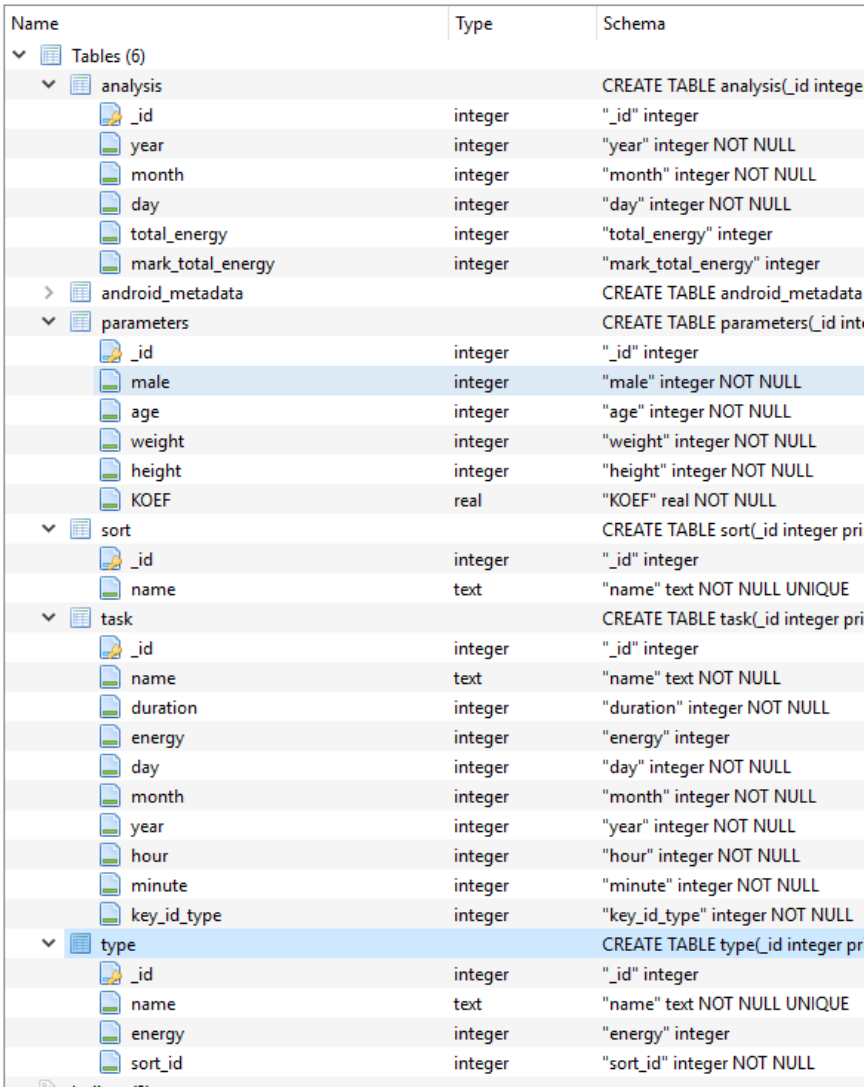


Рисунок 4.1 – Перегляд бази даних через додаток “DB browser for SQLite”

Також додаток “DB browser for SQLite” надає можливість переглянути створені індекси, що представлено на рис. 4.2.

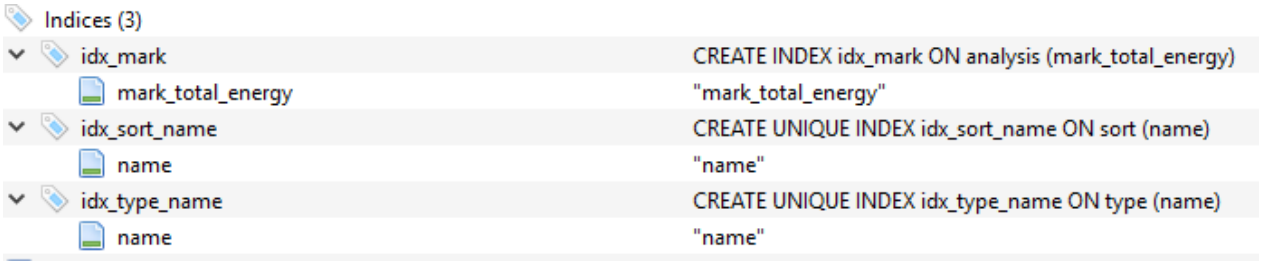


Рисунок 4.2 – Перегляд індексів бази даних через додаток “DB browser for SQLite”

Одразу після створення бази даних заповнюються її таблиці sort та type табличними значеннями активностей та відповідних енерговитрат.

Для роботи додатку з таблицями бази даних були створені відповідні класи-моделі, перелік яких зображений на рис. 4.3.

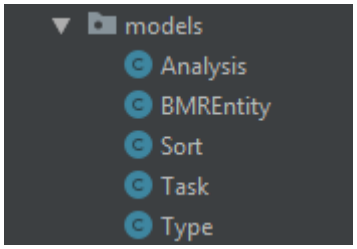
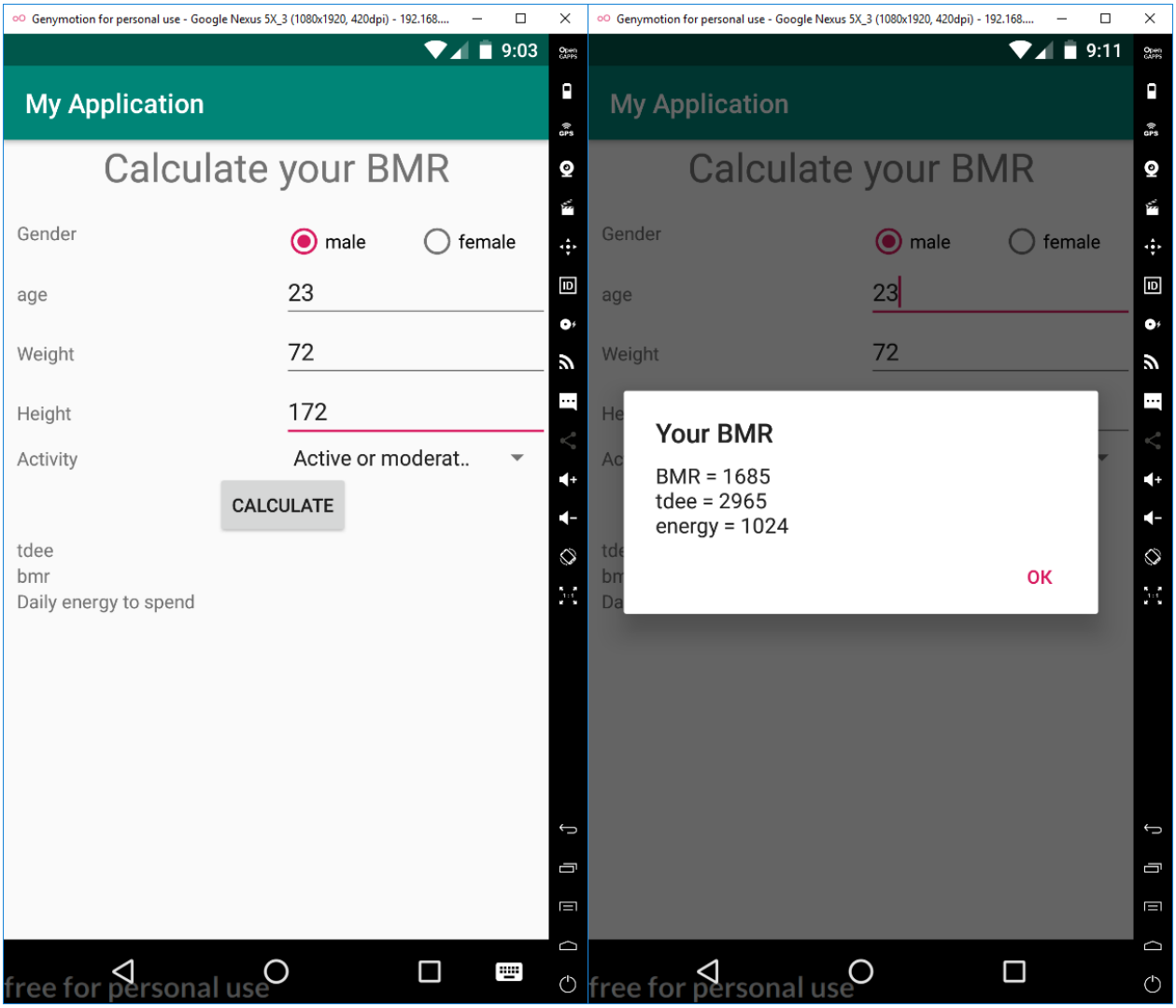


Рисунок 4.3 – Класи-моделі

4.2 Робота додатку

При першому запуску додатку користувач обов'яково має заповнити свої параметри, а саме стать, вік, зріст, вагу, тип життєвої активності у вікні, представленому на рис. 4.4(а).

Після цього користувачу необхідно натиснути на кнопку "Calculate" та розрахувати відповідно TDEE, BMR, добову енергію для витрат, про що користувач отримає сповіщення, представлене на рис. 4.4(б).



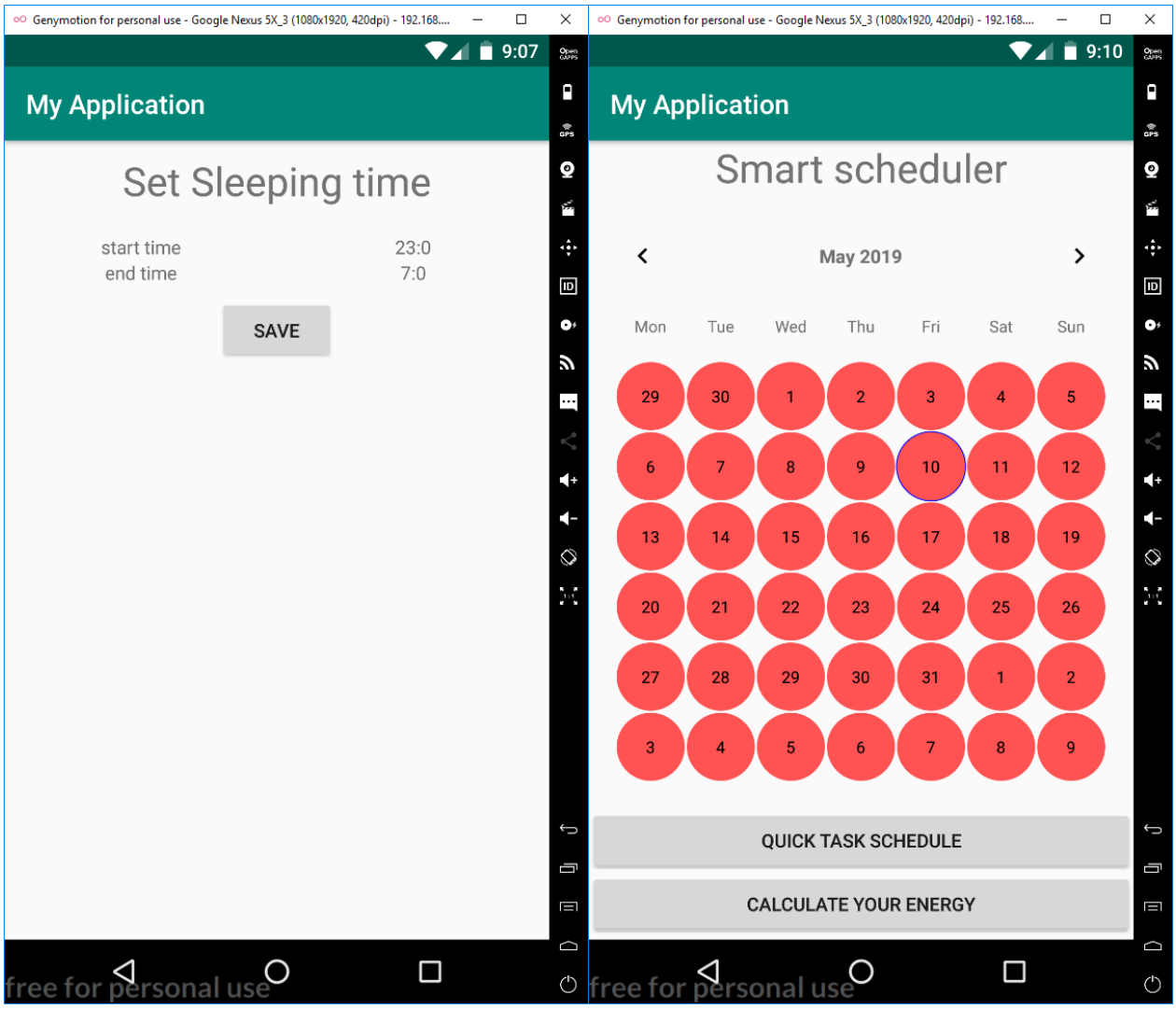
а) б)

Рисунок 4.4 – Стартове вікно (а) та сповіщення з розрахованими параметрами TDEE, BMR, середніх добових енерговитрат (б)

Наступним обов’язковим вікном є визначення годин сну користувча, задля швидкого заповнення цієї активності на весь рік до бази даних, зображене на рис. 4.5(а).

Після цього відкривається головне вікно програме, що є стартовим при кожному наступному запуску додатку, зображене на рис. 4.5(б). Центральним є елемент бібліотеки com.github.prolificinteractive:material-calendarview:2.0.1 - MaterialCalendarView, що має додаткові функціїї в порівнянні зі стандартним CalendarView. Головною причиною використання цієї бібліотеки є можливість впровадженя спеціальних міток для заголовка, робочих днів або окремих днів за допомогою decorator API [22]. За допомогою використаня decorator API було встановлено кастомізовані селектори та фони, а саме різнокольорові круги для кожного дня в залежності від навантаженості дня, та синє коло в якості селектора. DayViewDecorator - це інтерфейс, який має лише два методи, які необхідно реалізувати: shouldDecorate (CalendarDay) і decorate (DayViewFacade). Метод shouldDecorate () викликається для кожної дати в календарі, щоб визначити, чи слід застосовувати декоратор до цієї дати. Метод decorate () викликається лише один раз, щоб зібрати налаштування, що використовуються для цього декоратора [23].

Після першого входження до додатку всі дні є світло-червоного кольору, що означає недовантаженість, так як окрім сну ніяких активностей не було додано.

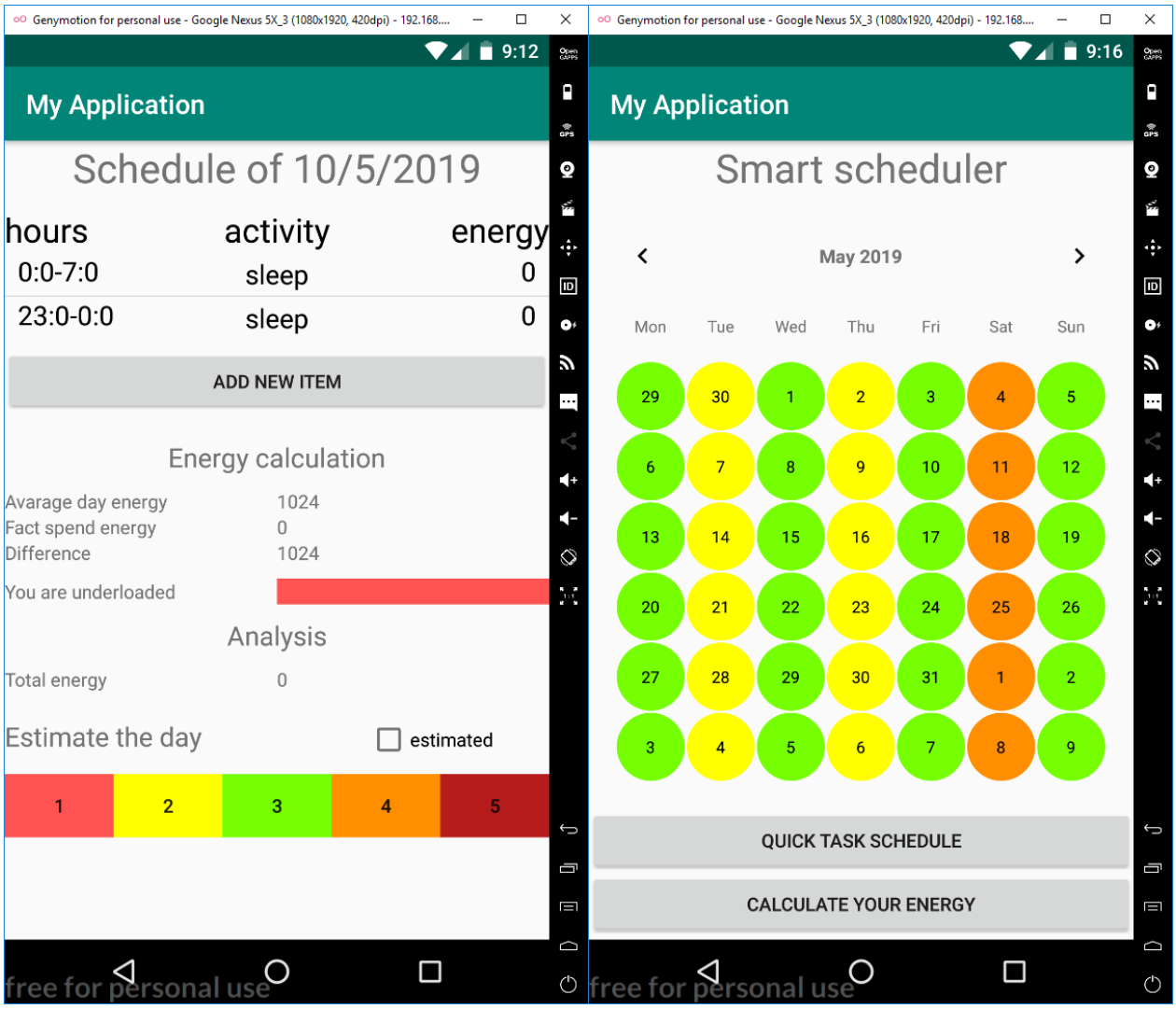


а) б)

Рисунок 4.5 – Вікно визначення годин сну (а) та головне вікно (б)

Користувач може перейти на будь-який день та побачити список активностей в обраний день, вікно активностей дня зображене на рис. 4.6(а). При першому вході в додаток лише сон є доданим. У вікні користувач також бачить середню кількість калорій для витрат, фактично витрачені калорії та різницю між цими значеннями.

Для реалістичного приведення прикладу роботи додатку база даних була заповнена тестовими даними. При вході в додаток головне вікно має вигляд зображений на рис. 4.6(б).



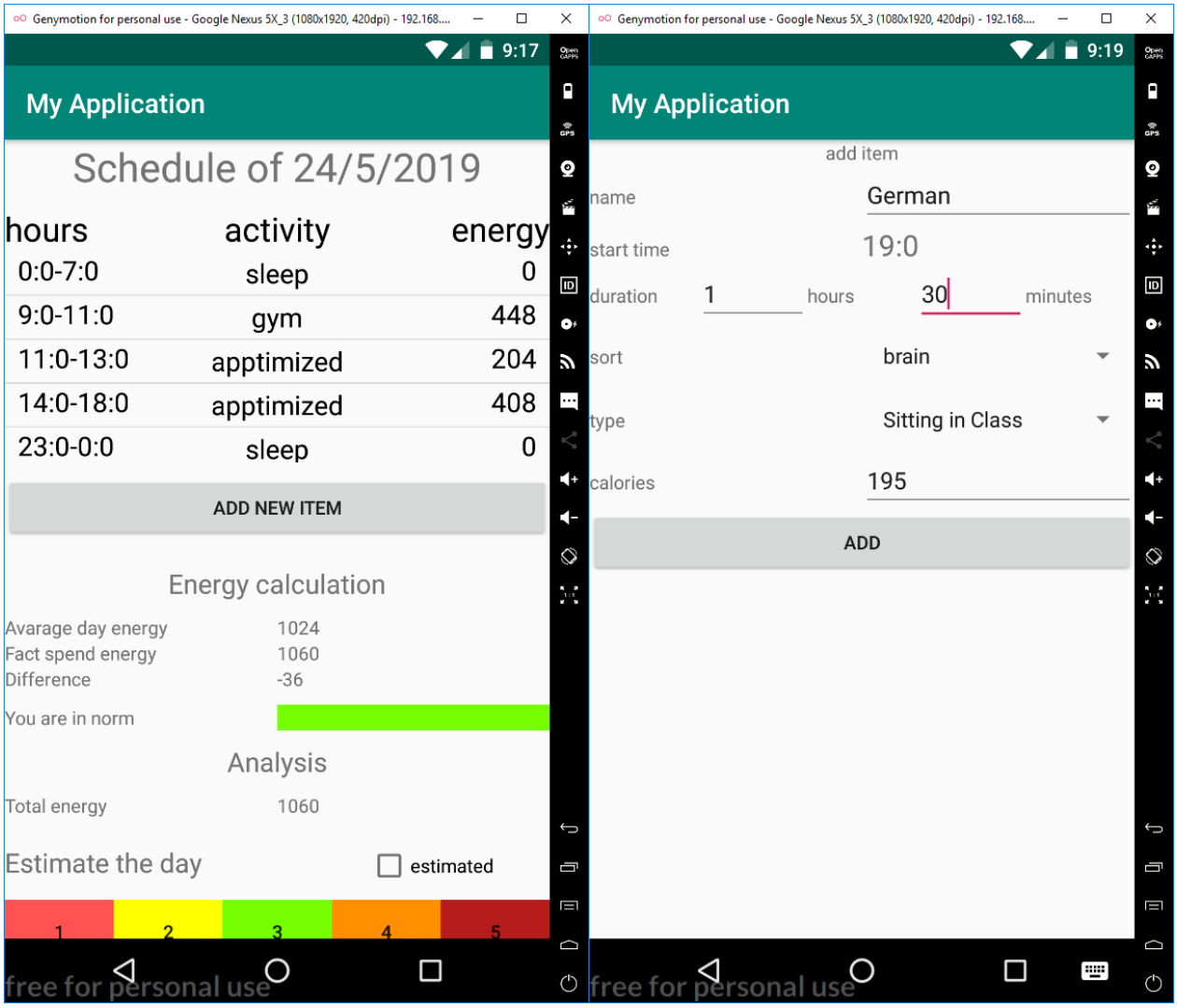
а) б)

Рисунок 4.6 – Вікно активностей дня (а) та головне вікно після заповнення бази даних тестовими даними (б)

Для розмітки вікна з активностями дня був використаний ScrollView, що надає можливість при великому наповненні листа активностей прокручувати все наповнення вікна та не втрачати елементів. Приклад використання зображений на рис. 4.7(а).

З вікна активностей дня можна перейти до вікна активності, зображеного на рис. 4.7(б). У ньому користувач вводить назву активності, її початок, продовжуваність, сорт та тип. Поле для калорій розраховується динамічно при зміні чисел в полях для годин та хвилин, завдяки взаємопов’язаним слухачам подій даних

події, а саме слухачам addTextChangedListener, а також кількості калорій для обраного типу, що є збереженим у базі даних.



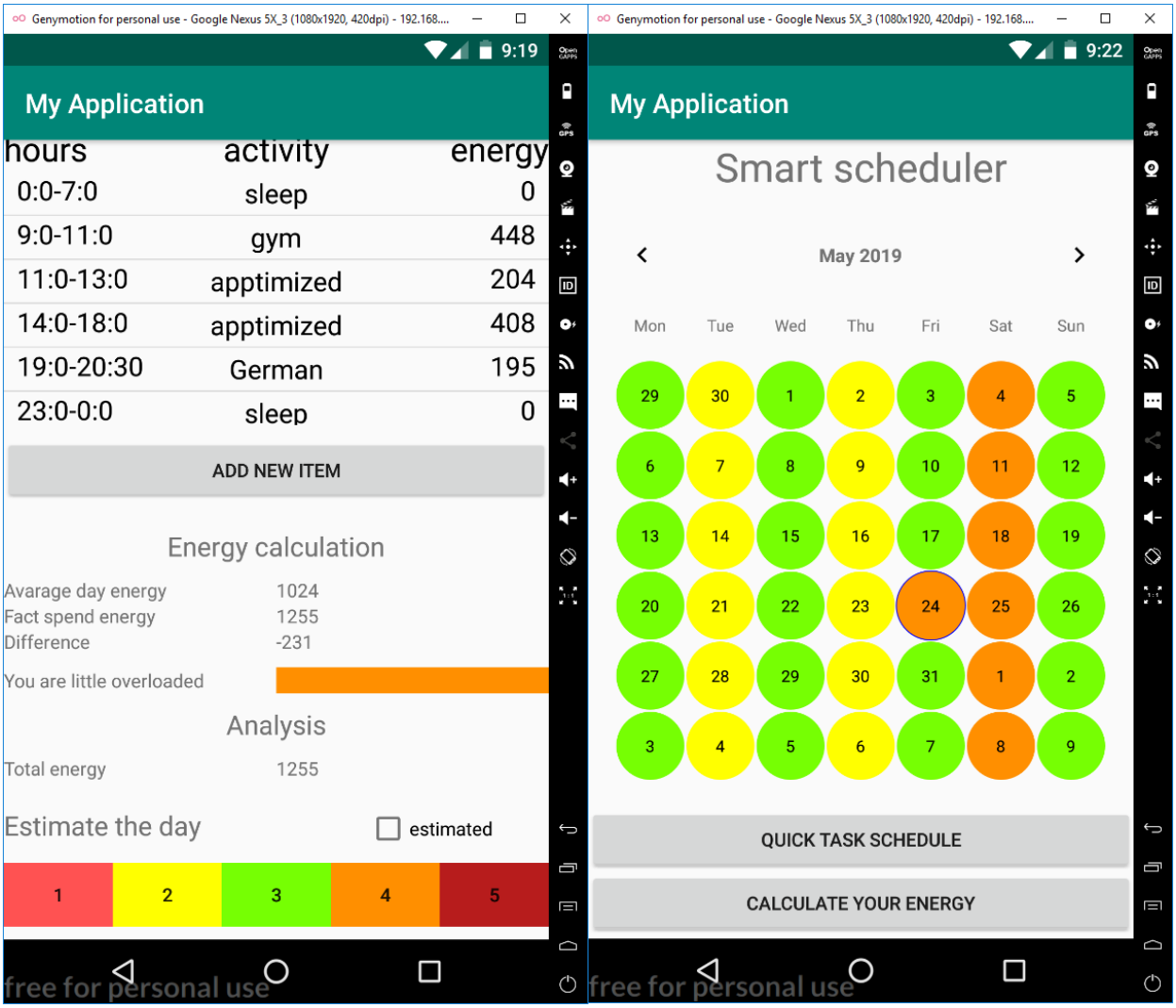
а) б)

Рисунок 4.7 – Вікно активностей дня (а) та вікно активності (б)

Після додавання активності додаток повертається до вікна активності дня. Лист активностей є оновленим, та відповідно до цього оновленим є значення фактично витраченої енергії та різниці її з середнім значенням, оновлене вікно зображене на рис. 4.8(а). Також змінюється підказка системи про навантаженість користувача в цей день. Відповідно на даному прикладі повідомляється про легке перенавантаження та колір змінюється на помаранчевий.

Відповідно після повернення на головне вікно додатку редагований день змінив свій колір також на помаранчевий, що дає зрозуміти користувачеві доцільність

перенести подію на інший, менш навантажений день, оновлене головне вікно зображене на рис. 4.8(б).



а) б)

Рисунок 4.8 – Оновлені вікно активностей дня (а) та головне вікно (б)

Кожну акивність можна редагувати або видаляти у вікні активності. При редагуванні або додаванні активності контролюється перекриття завдань, тобто в один момент часу може виконуватись лише одна подія. Якщо завланована подія перекриває вже існуюче завданя користувач буде повідомлений про це сповіщенням, зображеним на рис.4.9.

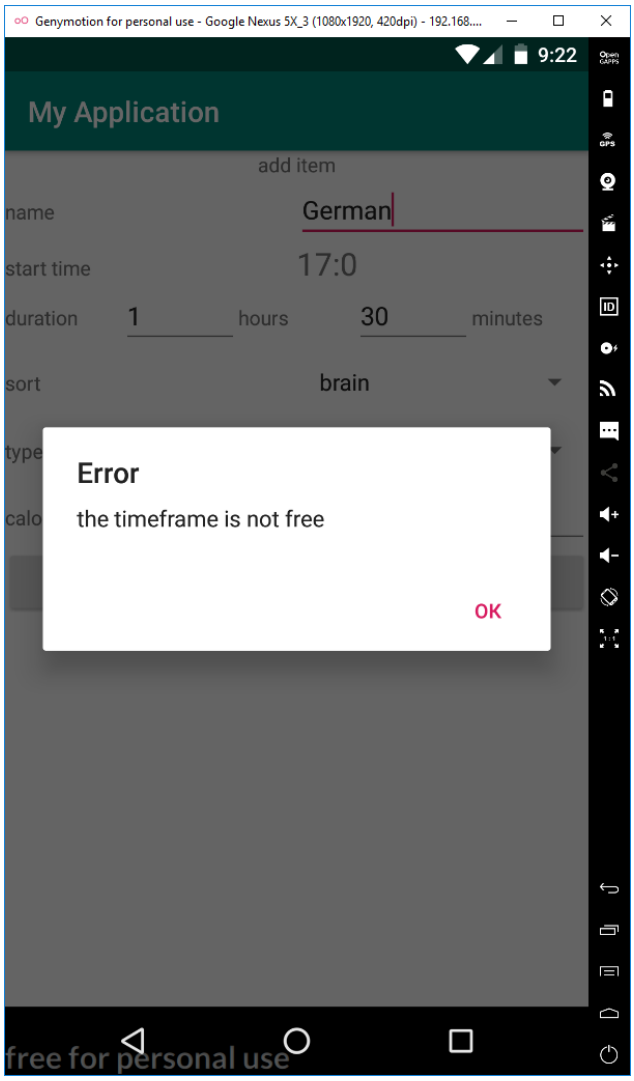
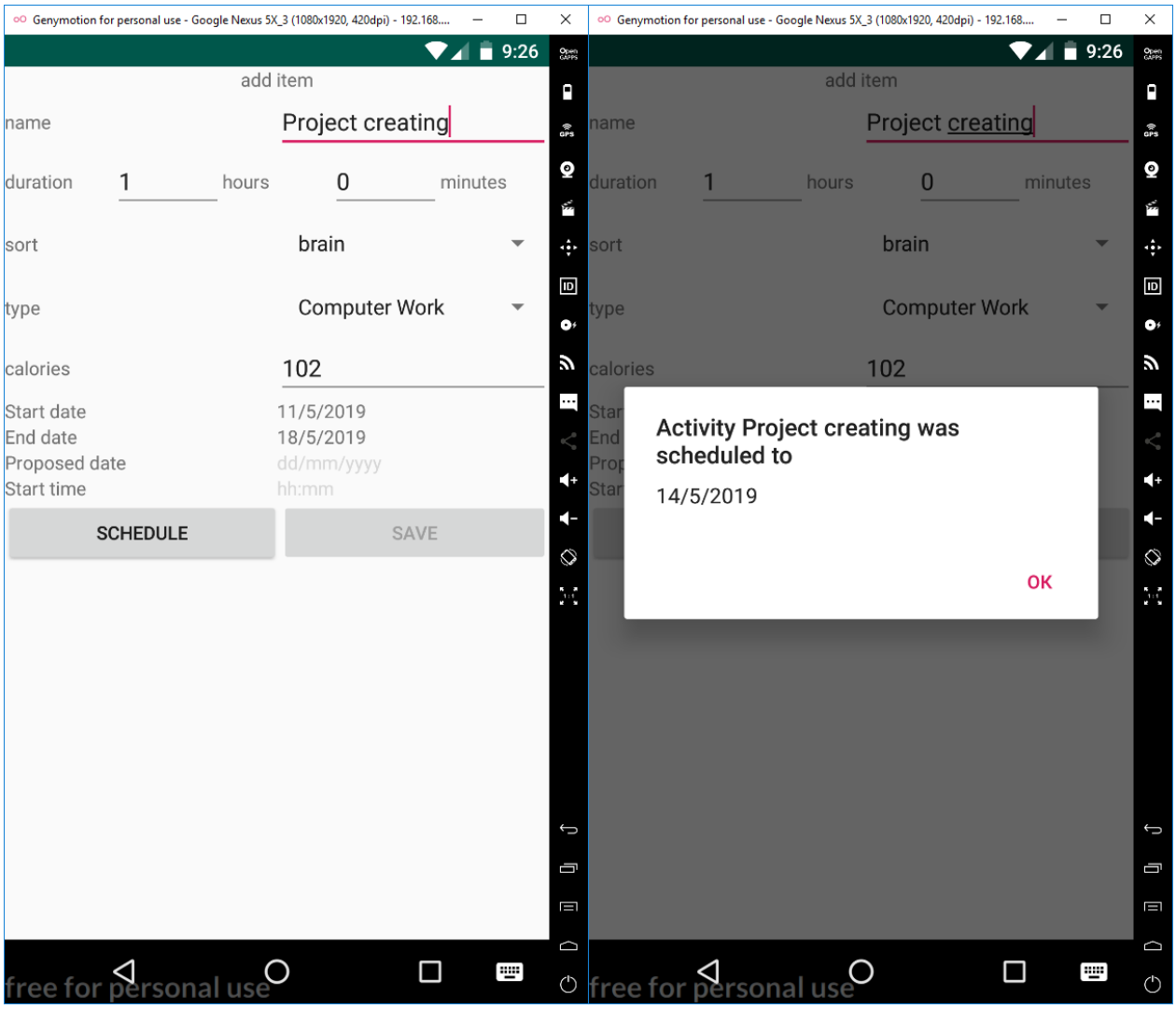


Рисунок 4.9 – Сповіщення про зайнятість тайм-фрейму

З головного вікна додатку користувач має можливість перейти до вікна планувння завдання, зображеного на рис. 4.10(а). Користувач має ввести параметри завдання: назву продовженість сорт та тип, проміжок дат між якими має бути записане задання.

При натисненні на кнопку “Schedule” додаток запропонує дату та час виконання завдання, за допомогою розробленого евристичного алгоритму, показавши сповіщення, зображене на рис. 4.10(б).



а) б)

Рисунок 4.10 – Вікно планування задачі (а) та сповіщення з запропонованою датою (б)

Якщо користувач погодиться з часом та датою, завдання додається в список завдань дня, вікно якого зображене на рис. 4.11 та відповідно оновлюються поля підрахунку калорій.

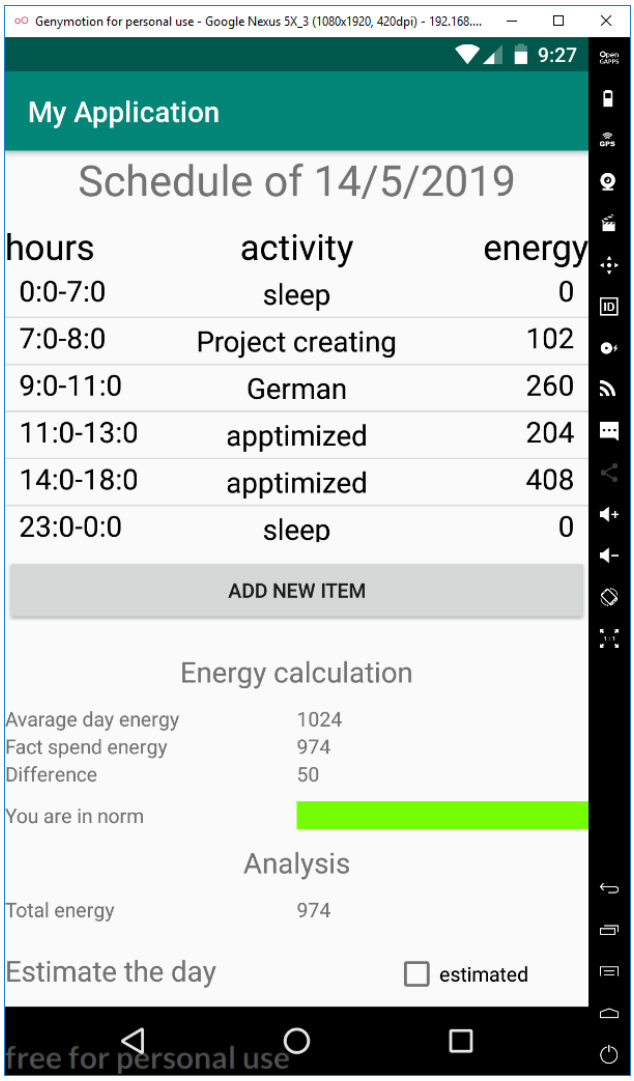


Рисунок 4.11 – Оновлене вікно активностей дня

Якщо завдаання виконувалось раніше, то система запропонує найчастіше використовуваний час початку завдання, якщо він вільнй у найменш навантаженому дні. Вікно з запропопонованим найчастіше використовуваним початком завдання зображений на рис. 4.12(а).

Якщо користувач погодиться з часом та датою, завдання додається в список завдань дня, вікно якого зображене на рис. 4.12(б) та відповідно оновлюються поля підрахунку калорій .

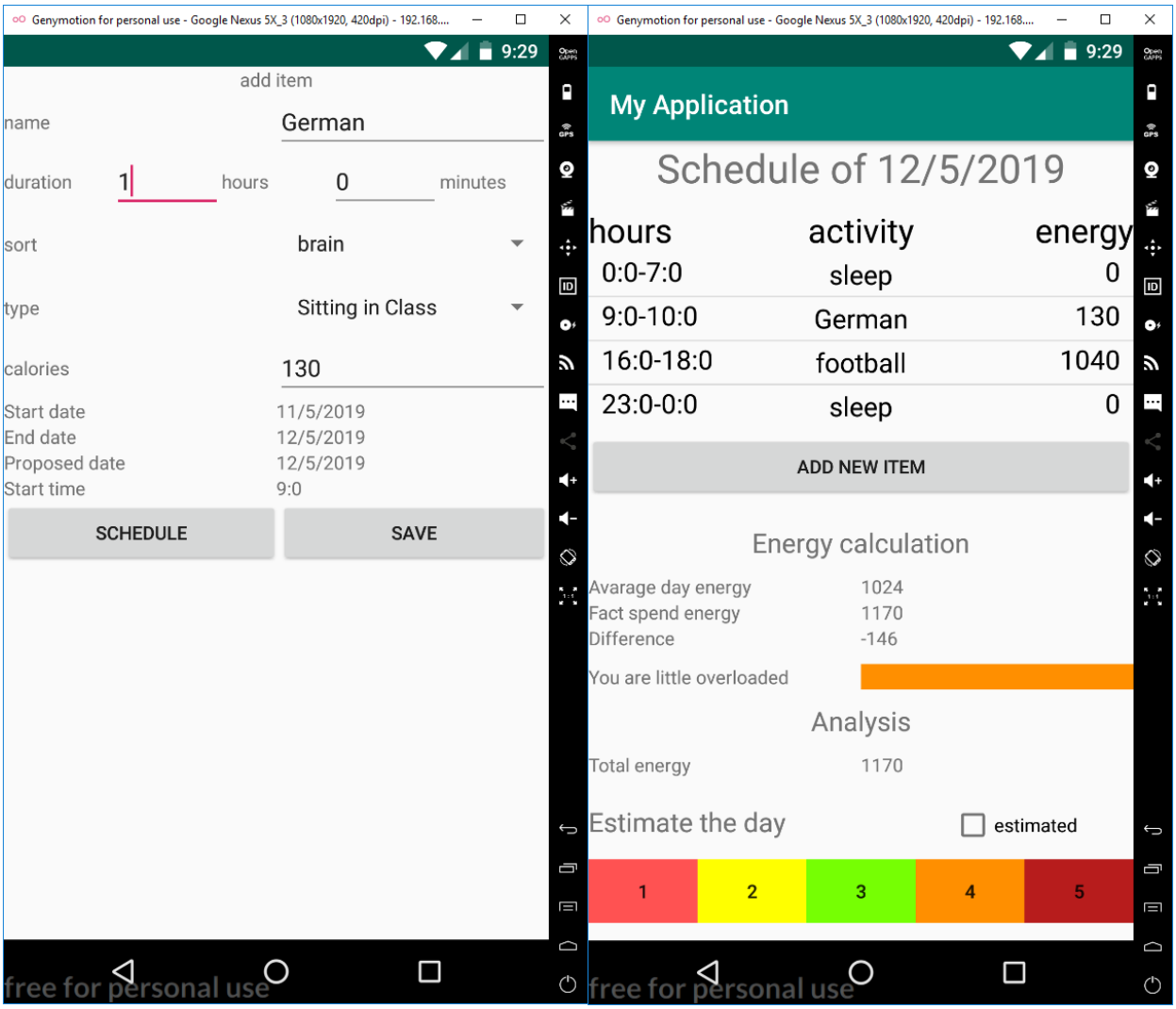


Рисунок 4.12 – Вікно з запропопонованим найчастіше використовуваним початком завдання (а) та оновлене вікно активностей дня(б)

У вікні перегляду завдань дня користувач має можливість оцінити навантаженість дня в залежності від своїх відчуттів, натиснувши на одну із п’яти кнопок відповідного кольору. Користувач буде сповіщений, що його оцінка збережена та надалі буде використовуватись для аналізу навантаження днів. Приклад зображений на рис. 4.13(а). Дана оцінка зааписується до бази даних в таблицю “analysis”.

Чекбокс “Estimated” стає позначеним та зафарбованим у колір оцінки у вікні завдань дня, зображеного на рис. 4.13(б).

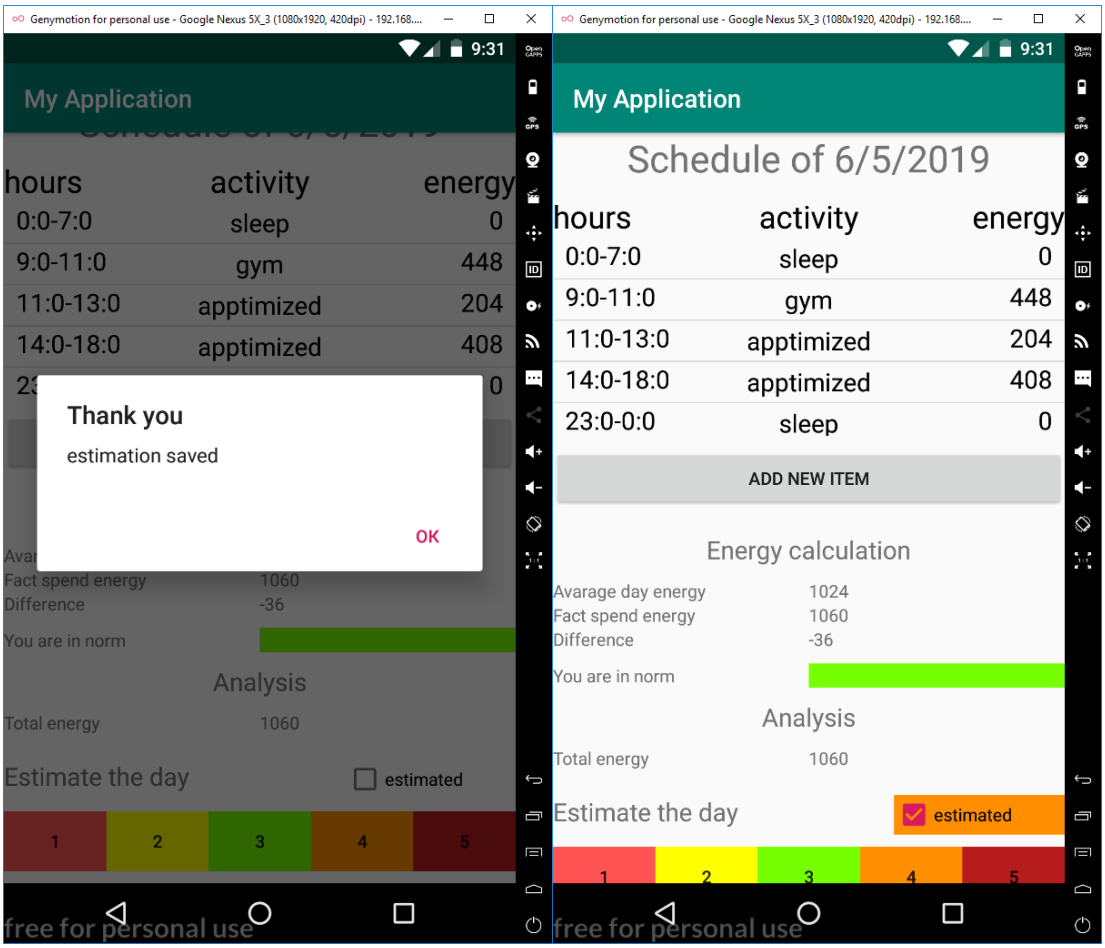


Рисунок 4.13 – Сповіщення про оцінку (а) та вікно активностей дня зі збереженою оцінкою(б)

Якщо користувач певну кількість разів оцінить день з подібним навантаженням як перевантажений, то додаток проаналізувавши дані оцінки за допомогою методу k найближчих сусідів, буде розцінювати дні з таким навантаженням як перевантажені, а не нормально навантажені, та відповідно перефарбує в помаранчевий колір такі дні. Наведений приклад зображений на рис. 4.14(а). Користувач розумітиме, що йому краще зменшувати кількість або ж продовжуваність певних завдань.

Головне вікно з прикладами всіх п’яти можливих класів навантажень та відповідними кольорами зображене на рис. 4.14(б).

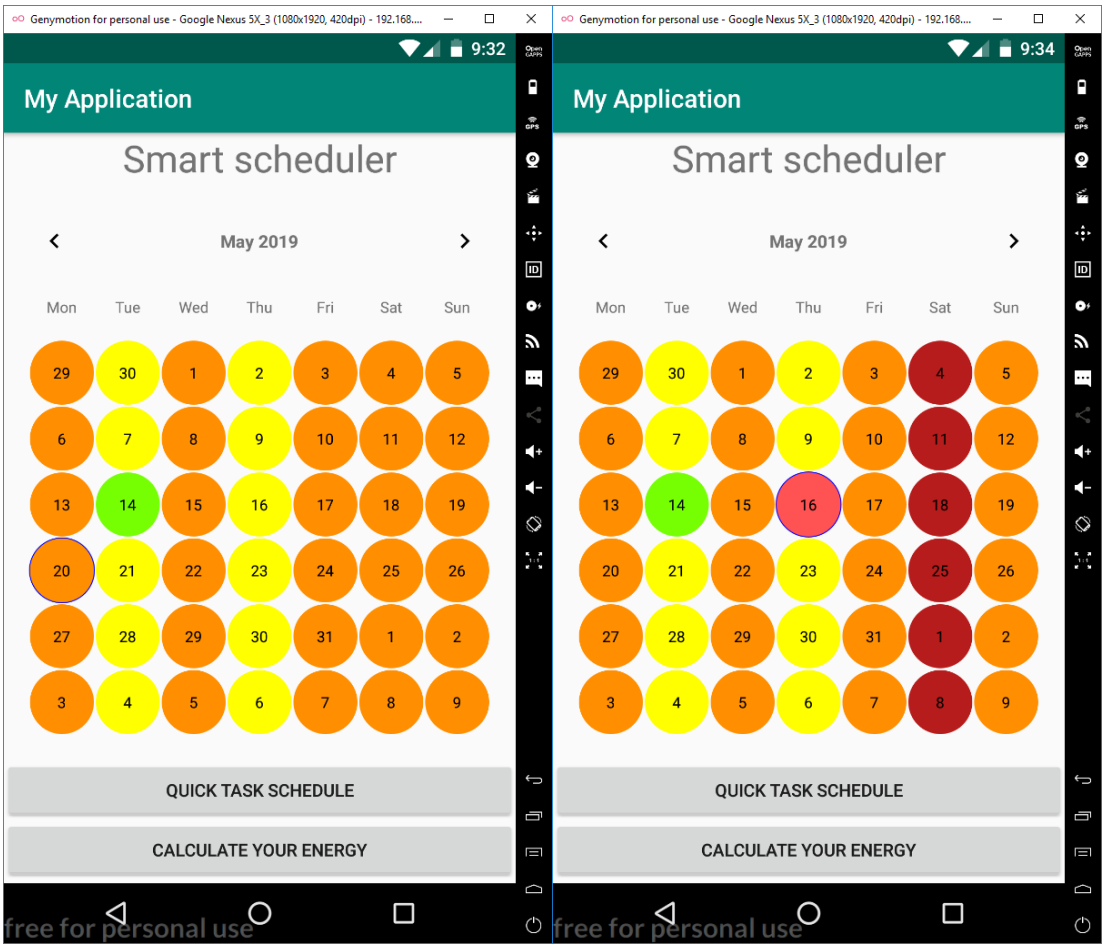


Рисунок 4.14 – Головне вікно додатку після оцінок користувача (а) та з прикладами всіх п’яти можливих класів навантажень (б)

4.3 Тестування додатку

Для тестування системи було обрано 20 персон, завданням яких було корситуватися тиждень додатком “Smart scheduler” та використовувати функції планування завдань та аналізу навантаження, оцінюючи енерговитрати кожного дня корситування.

Після тестового періоду кожен тестувальник заповнив форму з короткими питаннями – відповідями, виділив основні позитивні та сторони, що мають бути поліпшені.

Таблиці 4.1 – 4.4 представляють короткі питання та відповіді.

Таблиця 4.1 – Відповіді на питання: «На скільки відрізняється планування завдань з допомогою додатку від особистого планування?»

Гірше	**Однаково**	**Краще**
10%	60%	30%

Таблиця 4.2 – Відповіді на питання: «Чи робив календар Вас більш продуктивним?»

Так	**Ні**
80%	20%

Таблиця 4.3 – Відповіді на питання: «На скільки система правильно оцінює добове навантаження?»

Зовсім неправильно	**У половині випадків**	**Правильно**
20%	40%	40%

Таблиця 4.4 – Відповіді на питання: «Чи відповідають табличні значення енерговитрат вашим відчуттям?»

Зовсім не відповідають	**У половині випадків**	**Відповідають**
30%	40%	30%

На питання без варіантів відповідей про позитвні сторони та ті, що мають бути поліпшені були дані наступні відповіді:

Позитивні:

- Додаток запам’ятовує найчастіше вживаний час завдання, що в більшості випадків надає бажаний час при плануванні задань
- Додаток переважно визначає потрібну дату при плануванні завдань в розрізі енерговитарт
- Головне вікно додатку наочно та зрозуміло показує навантаження людини, на основі чого користувач може перенести завдання

Поліпшення:

- Користувач потребує можливість додати тип активностей та змінювати значення енерговитрат, адже може виміряти середнє значення наприклад фітнес-трекером
- Інтеграція з поширеними додатками, як Google календар.
- Оформлення більш яскравого інтерефейсу вікон.
- Створення нагадувань про оцінку попереднього дня
- Можливість дублювання завдань на декілька днів

ВИСНОВКИ

Результати аналізу предметної області дослідження та існуючих аналогів дозволили обгрунтувати актуальність роботи, а також сформовати мету - розробка інтелектуальної системи планування навантаження людини з урахуванням специфіки її діяльності.

Були проведені аналіз традиційних та сучасних аналогів інструментів планування часу, огляд існуючих досліджень на тему навантаження людини та виміру енергетичних витрат під час виконання активностей.

Android додаток було обрано як тип продукту системи, інструментом реалізації – Android Studio.

Ророоблена інтелектуальна система виконує спеціальні функції аналізу навантаження людини та планування завдання в оптимальний з точки зору навантаження час, а також звичайні для аналогів функції календаря та todo-list.

Наукова новизна дослідження полягає у доповнені існуючої інформаційної технології формування навантаження людини в частині врахування специфіки діяльності людини.

Практична значимість полягає у розробленні інтелектуальної системи автоматизованого формування навантаження людини. Використання розробленої системи дозволить оптимальним чином розподілити задачі людини для контролю енергетичних витрат, уникнення її втоми та "вигорання".

Проведення подальших досліджень та розробки полягає в трьох напрямках, основною ідеєю якого є використання часу, як одиниці виміру навантаження:

1) розробка альтернативного методу аналізу навантаження людини, шо полягає у проведенні аналізу навантаження за параметрами оцінки часу виконання задач – сумарного, фізичної та розумової діяльності
2) удосконалення евристичного алгоритму планування задань, а саме врахування історії її виконання – часу та відповідної оцінки

3) використання біоритмів людини при аналізі навантаження та планування задач

Результати дипломного проектування апробовані на конференції студентів, аспірантів та співробітників «ІМА-2019», опубліковані тези доповідей у збірнику матеріалів конференції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to overworking. Режим доступу: https://www.statustoday.com/what-is-overworking/

2. 4 Time Management Tips For The Chronically Overworked. Режим доступу: https://www.americanexpress.com/en-us/business/trends-and-insights/articles/4-time-management-tips-for-a-247-work-life/

3. his New App Wants to Optimize Your Schedule by Running It Through an Algorithm. Режим доступу: https://www.entrepreneur.com/article/236094/

4. Дерево_ухвалення_рішень. Режим доступу: https://uk.wikipedia.org/wiki Дерево_ухвалення_рішень

5. А.А. Бородинова , В.В. Мясников “Сравнение алгоритмов классификации в задаче распознавания объектов на радарных изображениях базы MSTAR”, 2017

6. Meet Timeful, the intelligent ‘Time Assistant’. Режим доступу: https://bgr.com/2014/07/31/timeful-time-assistant-app-for-iphone//

7. Time’s AI-powered time tracking app will help you stop procrastinating. Режим доступу: https://techcrunch.com/2017/01/09/times-ai-powered-time-tracking-app-will-help-you-stop-procrastinating/

8. 6 Artificial Intelligence Apps That Will Help You Reach Your Goals. Режим доступу: https://www.makeuseof.com/tag/artificial-intelligence-apps-reach-goals

9. Brett Lantz. Machine Learning with R. Pack Publishing. Birmongham-Mumbai, 2013.

10. Refanidis, I., & Alexiadis, A. (2011). Deployment and evaluation of selfplanner, an automated individual task management system. Computational intelligence, 27(1), 41-59.

11. H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

12. Harris JA, Benedict FG (1918). "A Biometric Study of Human Basal Metabolism"

13. ENERGY REQUIREMENTS OF ADULTS. Режим доступу: http://www.fao.org/docrep/007/y5686e/y5686e07.htm#bm07.2

14. Керування часом. Режим доступу: https://uk.wikipedia.org/wiki/Керування_часом

15. 10 причин планировать свой день. Режим доступу: http://www.vitamarg.com/article/causes/7541-10-prichin-planirovat-svoj-den

16. Introducing Smart Schedule, a more intelligent way to plan your day. Режим доступу: https://doist.com/blog/todoist-smart-schedule/

17. iOS vs Android Development: Which One is Best for Your App? Режим доступу: https://rubygarage.org/blog/ios-vs-android-development

18. Какую технику хранения данных Android использовать? Режим доступу: http://qaru.site/questions/180398/which-android-data-storage-technique-to-use

19. SQLite. Режим доступу: https://ru.wikipedia.org/wiki/SQLite

20. Best Android developer tools for getting started or levelling up your dev skills. Режим доступу: https://www.androidauthority.com/best-android-developer-tools-671650/

21. Advantages of using Android official tools. Режим доступу: http://androiddeveloper.galileo.edu/2017/09/25/advantages-using-android-official-tools/

22. Material Calendar View. Режим доступу: https://github.com/prolificinteractive/material-calendarview

23. Decorators. Режим доступу: https://github.com/prolificinteractive/material-calendarview/wiki/Decorators

24. Архітектура інформаційної системи. Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20151208095132/170352/index.html

25. ISO/IEC 2382:2015, Information technology — Vocabulary — Part 1

26. Концептуальная модель. Режим доступу: https://ru.wikipedia.org/wiki/Концептуальная_модель

27. Логічна модель даних. Режим доступу: https://uk.wikipedia.org/wiki/Логічна_модель_даних#Концептуальна,_логічна_та_фізична_модель_даних

28. Фізична модель. Режим доступу: https://uk.wikipedia.org/wiki/Фізична_модель_даних

29. Lall U., Sharma A. A nearest neighbor bootstrap for resampling hydrologic time series //Water Resources Research. – 1996. – Т. 32. – №. 3. – С. 679-693.

30. Calories burned in 30 minutes for people of three different weights. Режим доступу: https://www.health.harvard.edu/newsweek/Calories-burned-in-30-minutes-of-leisure-and-routine-activities.htm

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є Інтелектуальна система планування навантаження людини з урахуванням специфіки її діяльності.

Результати деталізації методом SMART розміщені у табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Інтелектуальна система планування навантаження людини з урахуванням специфіки її діяльності
Measurable (вимірювана)	Результатом роботи проекту є оцінка замовника.
Achievable (досяжна)	Реалізації системи здійснюється за допомогою середовища розробки Android Studio.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були оговорені замовником проекту. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Основним інструментом для планування змісту структури робіт служить WBS діаграма − графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. Побудуємо структуру WBS, у якій детально опишемо роботи, які потрібно виконати на кожному етапі створення проекту. Виконаємо декомпозицію робіт для даного проекту. Діаграма WBS зображена на рис. А.1.

Планування структури організації, для впровадження готового проекту (OBS). Після побудови WBS розробимо організаційну структуру виконавців OBS. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Діаграма OBS зображена на рис. А.2. Список виконавців, що функціонують в проекті знаходиться в табл. А.2.

Таблиця А.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Антипенко Б.А.	Виконує розробку основного функціоналу проекту та інтерфейс користувача
Проектувальник	Антипенко Б.А.	Проектує базу даних, інтерфейс, моделює використання програми.
Тестувальник	Будник О.С.	Відповідає за тестування функціоналу та дизайну додатку, перевірку моделі на адекватність.
Косультант проекту	Марченко А.В.	Консультує на всіх етапах проекту.
Менеджер проекту	Антипенко Б.А.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

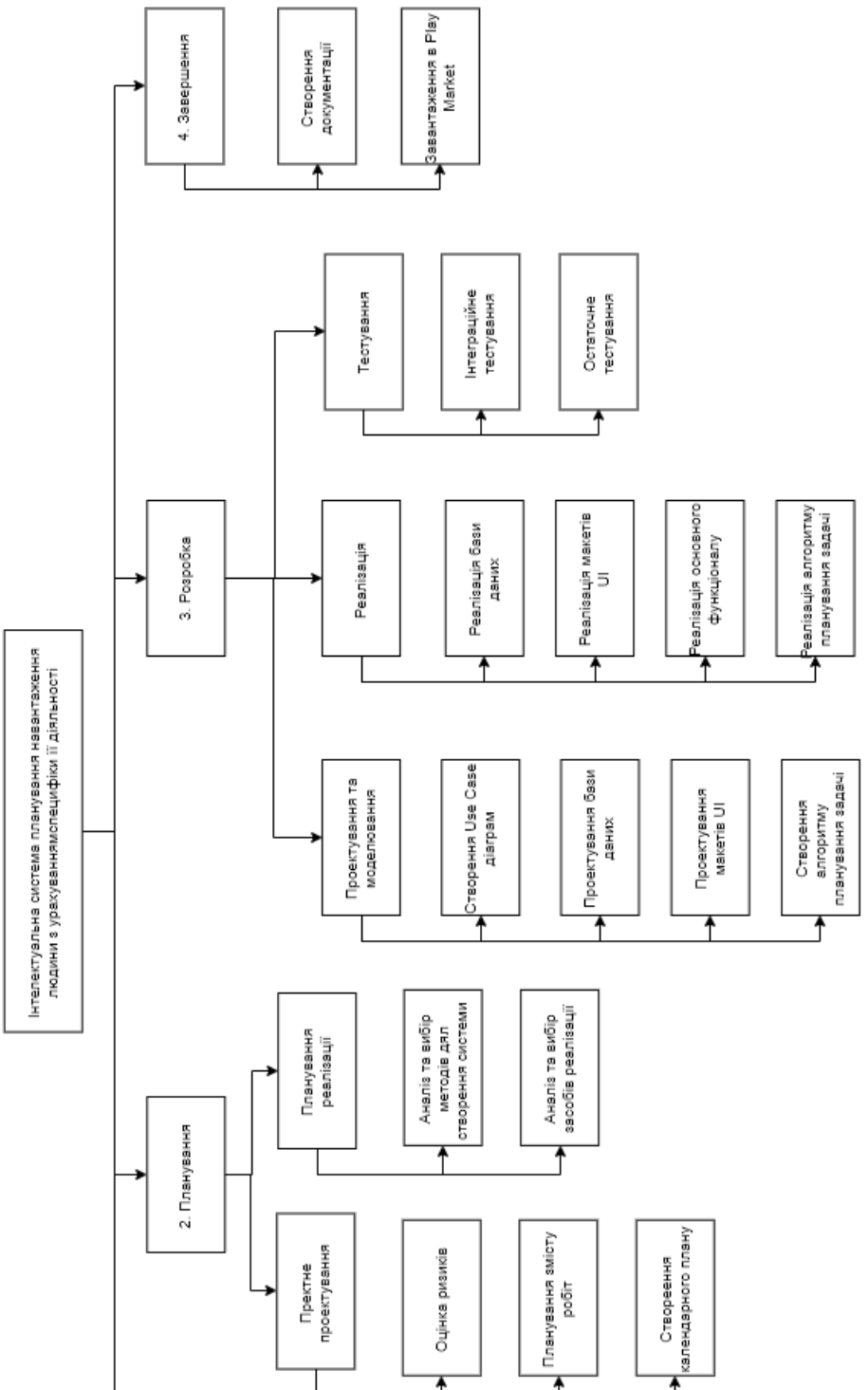


Рисунок А.1 – WBS. Структура робіт проекту

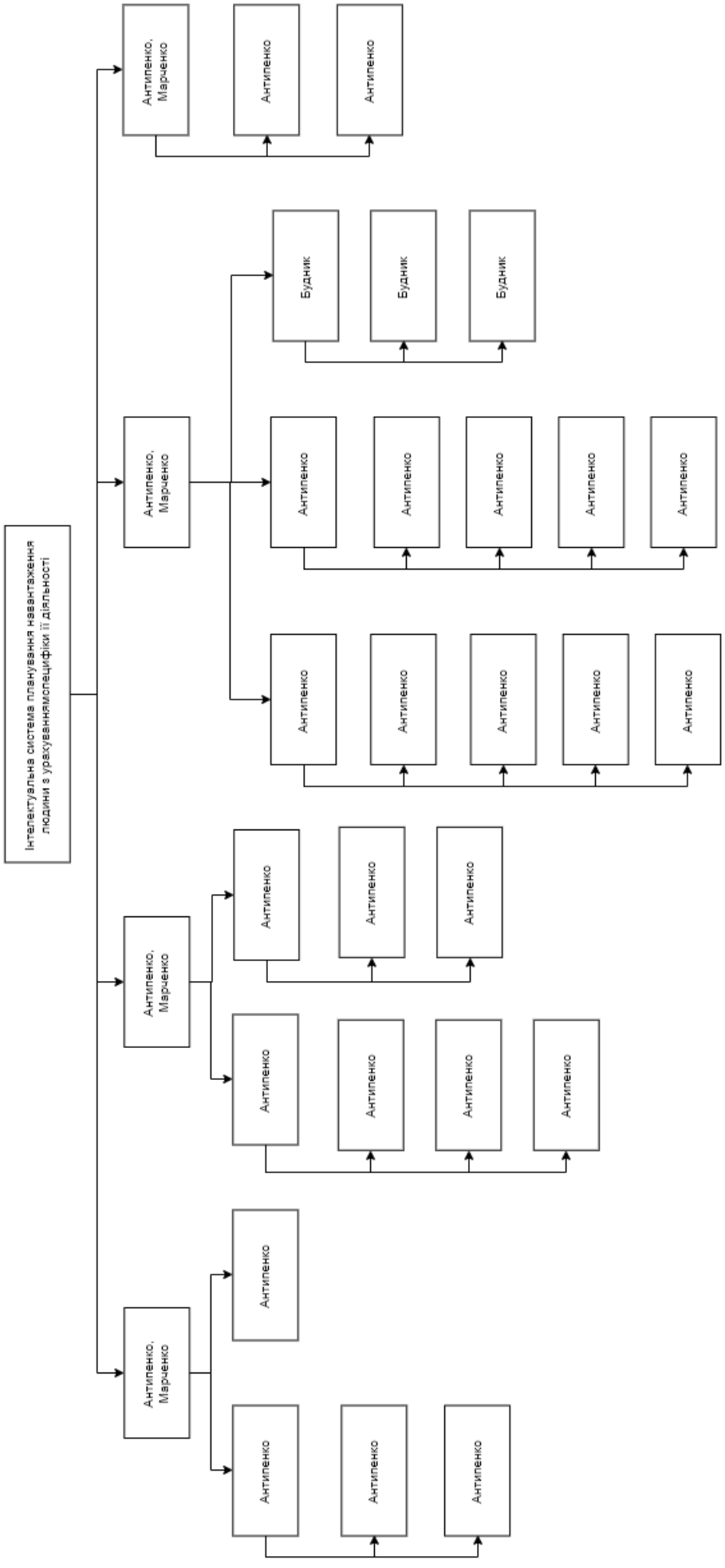


Рисунок А.2 – Організаційна структура проекту (OBS)

Діаграма Ганта. Далі побудуємо календарний план виконання дипломного проекту. Найпоширеніший формат графіка в будь-якій галузі — діаграма Ганта. Цей графік дозволяє менеджерам проекту і всій команді розробників візуалізувати графіки часу і взаємозв'язок між окремими завданнями та етапами роботи над проектом. Тривалість виконання робіт зазначена в днях, але фактична тривалість виконання робіт приблизно дорівнює 2-3 години на день. Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, з урахуванням вихідних та святкових днів, побудовано календарний графік. Діаграма Ганта та список робіт діаграми Ганта зображені на рис. А.3-А.7.

Название задачи	Длительно	Начало	Окончание
Ініціювання	**50 дней**	**Пн 03.09.18**	**Пт 09.11.18**
Аналіз предмеої області	**30 дней**	**Пн 03.09.18**	**Пт 12.10.18**
Аналіз проблеми перенавантаження людини	10 дней	Пн 03.09.18	Пт 14.09.18
Аналіз методів планування задач	20 дней	Пн 17.09.18	Пт 12.10.18
Аналіз систем аналогів	20 дней	Пн 15.10.18	Пт 09.11.18

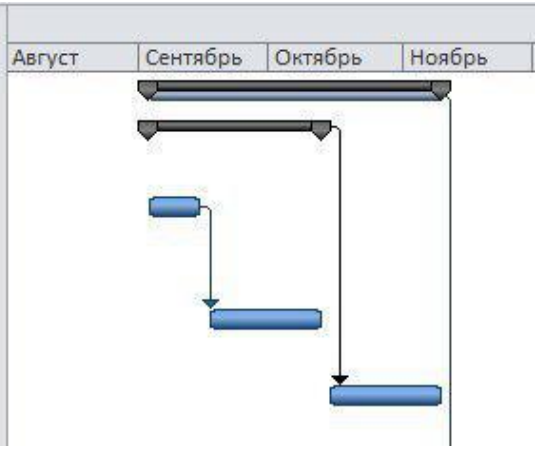


Рисунок А.3 − Діаграма Ганта

Название задачи	Длительно	Начало	Окончание
Планування	**29 дней**	Пн 12.11.18	**Чт 20.12.18**
Проектне проектування	**9 дней**	**Пн 12.11.18**	**Чт 22.11.18**
Оцінка ризиків	3 дней	Пн 12.11.18	Ср 14.11.18
Планування змісту робіт	3 дней	Чт 15.11.18	Пн 19.11.18
Створення календарного плану	3 дней	Вт 20.11.18	Чт 22.11.18
Планування реалізації	**20 дней**	**Пт 23.11.18**	**Чт 20.12.18**
Аналіз та вибір методів для створення системи	10 дней	Пт 23.11.18	Чт 06.12.18
Аналіз та вибір засобів реалізації	10 дней	Пт 07.12.18	Чт 20.12.18

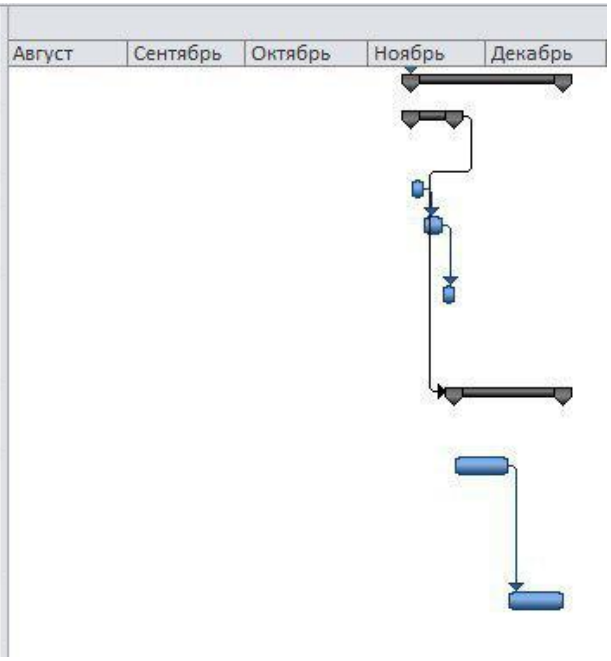


Рисунок А.4 − Продовження діаграми Ганта

Название задачи	Длительно	Начало	Окончание
Розробка	**95 дней**	**Чт 10.01.19**	**Ср 22.05.19**
Проектування та моделювання	**40 дней**	**Чт 10.01.19**	**Ср 06.03.19**
Створення Use Case діаграм	10 дней	Чт 10.01.19	Ср 23.01.19
Проектування бази даних	10 дней	Чт 24.01.19	Ср 06.02.19
Проектування макетів UI	10 дней	Чт 07.02.19	Ср 20.02.19
Створення алгоритму планування задачі	10 дней	Чт 21.02.19	Ср 06.03.19

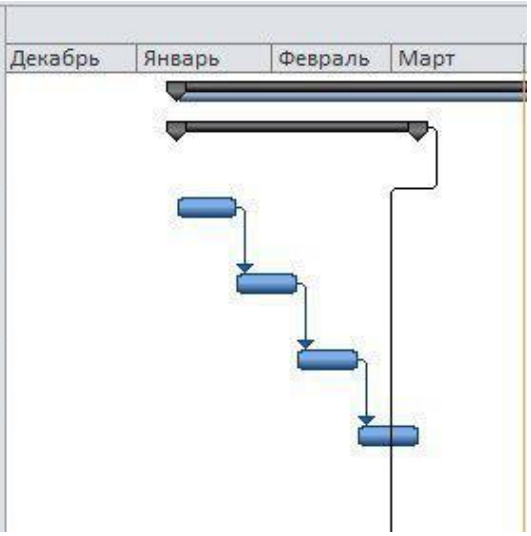
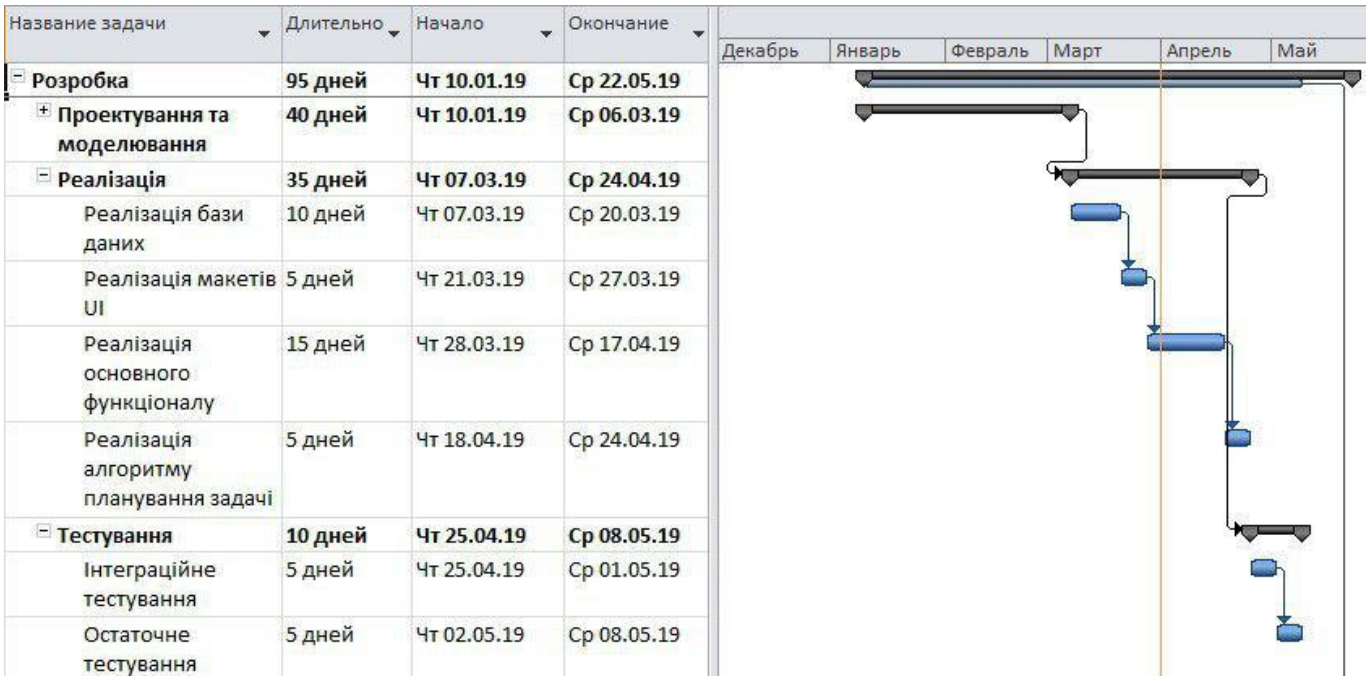
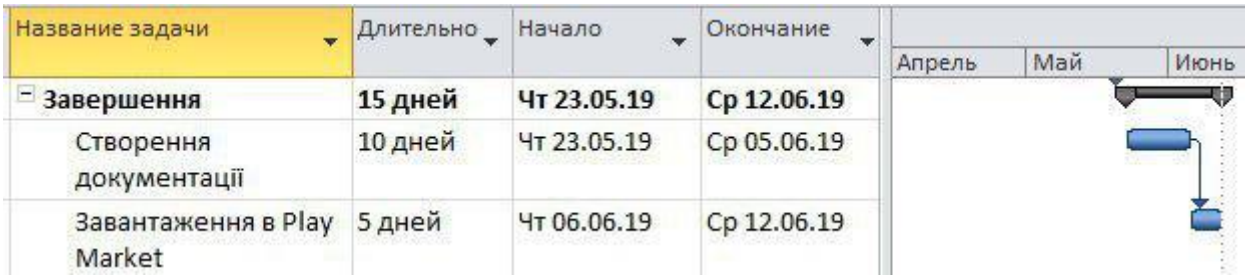


Рисунок А.5 − Продовження діаграми Ганта



Название задачи	Длительно	Начало	Окончание
Розробка	**95 дней**	**Чт 10.01.19**	**Ср 22.05.19**
Проектування та моделювання	**40 дней**	**Чт 10.01.19**	**Ср 06.03.19**
Реалізація	**35 дней**	**Чт 07.03.19**	**Ср 24.04.19**
Реалізація бази даних	10 дней	Чт 07.03.19	Ср 20.03.19
Реалізація макетів UI	5 дней	Чт 21.03.19	Ср 27.03.19
Реалізація основного функціоналу	15 дней	Чт 28.03.19	Ср 17.04.19
Реалізація алгоритму планування задачі	5 дней	Чт 18.04.19	Ср 24.04.19
Тестування	**10 дней**	**Чт 25.04.19**	**Ср 08.05.19**
Інтеграційне тестування	5 дней	Чт 25.04.19	Ср 01.05.19
Остаточне тестування	5 дней	Чт 02.05.19	Ср 08.05.19

Рисунок А.6 − Продовження діаграми Ганта



Название задачи	Длительно	Начало	Окончание
Завершення	**15 дней**	**Чт 23.05.19**	**Ср 12.06.19**
Створення документації	10 дней	Чт 23.05.19	Ср 05.06.19
Завантаження в Play Market	5 дней	Чт 06.06.19	Ср 12.06.19

Рисунок А.7 − Продовження діаграми Ганта

Аналіз ризиків. Виконаємо якісну і кількісну оцінку ризиків роботи. При якісній оцінці визначимо ризики, що потребують швидкого реагування. Така оцінка визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У табл. А.5 знаходиться класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики — це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним. Оцінюємо ризики за показниками, що знаходяться в табл. А.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на рис. А.8.

Таблиця А.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

Ймовірність виникнення				
	3			RS_5, RS_9
	2	RS_13	RS_4, RS_6	RS_3, RS_7, RS_14
	1		RS_1, RS_8, RS_11, RS_12	RS_2, RS_10, RS_15
		1	2	3
		Вплив ризику		

Рисунок А.8 − Матриця ймовірності виникнення ризиків та впливу ризику

- зелений колір – прийнятні ризики;
- жовтий колір – виправданні ризики;
- червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в табл. А.4.

Таблиця А.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1,8,11,12,13
2	Виправдані	3≤R≤4	2,4,6,10,15
3	Недопустимі	6≤R≤9	3,5,7,9,14

Таблиця А.5 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	2	1. Налагодити гарні відносини між розробником та керівником. 2. Дотримуватися ділового етикету спілкування. 3. Створити комфортні умови для співпраці	Попередження	При виявлення непорозуміння потрібно вияснити, що саме стало причиною непорозуміння обсудити її та створити здорову атмосферу в колективі.
RS_2	Відкритий	Поява альтернативного продукту	Низька	Високий	3	1. Провести попереднє дослідження альтернативних продуктів. 2. Вибрати унікальну стратегію створення проекту.	Прийняття	
RS_3	Відкритий	Нечітке завдання на розробку	Середня	Високий	6	1. Ясно і однозначно скласти ТЗ разом із замовником, обговоривши усі види вимог. 2. Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів. 3. Періодичний контроль замовником етапів роботи.	Попередження	При виявленні невідповідностей деяких характеристик продукту заявленим вимовам потрібно уважно та чітко окреслити те, що було виконано невірно (після розмови із замовником) та зробити правки.

Продовження таблиці А.5 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	4. План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Низька кваліфікація розробників проекту	Середня	Середній	4	1. Підвищити кваліфікацію персоналу. 2. Переглянути онлайн-ресурси для підвищення рівня знань.	Пом'якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн-уроки.
RS_5	Відкритий	Неоптимальний розподіл часу	Висока	Високий	9	Провести аналіз актуальності найважливіших процесів та робіт. Звернути особливу увагу на правильність розподілу часу. Правильно визначити пріоритети виконання робіт. Чітко дотримуватися календарного плану.	Пом'якшення	Змінити порядок пріоритетів робіт. Знайти способи оптимізації роботи із вже існуючою розстановкою. Обговорити варіанти внесення правок до термінів реалізації із замовником.
RS_6	Відкритий	Часте внесення змін у ТЗ	Середня	Середній	4	1. Виділити всі необхідні параметри проекту. 2. Чітко описати вимоги до проекту. 3. Обговорити всі технічні засоби виконання проекту та умови реалізації.	Перенос	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.
RS_7	Відкритий	Вибір неефективної технології розробки	Середня	Високий	6	1. Проаналізувати методи та засоби, для виконання проекту.	Пом'якшення	Виділити час та ресурсі на пошуки покращення обраної технології.

Продовження таблиці А.5 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_8	Відкритий	Не вірна оцінка масштабів проекту	Низька	Середній	2	Провести детальний аналіз проекту. Визначити основні етапу проекту, розподілити час на їх виконання. Проаналізувати масштаби проекту на основі додаткових джерел.	Пом'якшення	Переоцінка масштабів проекту. Перебудова стратегії реалізації проекту.
RS_9	Відкритий	Помилки проектування	Висока	Високий	9	На етапі проектування тісно співпрацювати із замовником та на певних етапах демонструвати поточні результати.	Пом'якшення	Здійснювати проміжний контроль результатів в ході виконання проекту.
RS_10	Відкритий	Збої в роботі програмного забезпечення	Низька	Високий	3	1. Підготувати резерв програмних засобів. 2. Залучити спеціаліста для усунення збоїв.	Попередження	Замінити програмне забезпечення.
RS_11	Відкритий	Відсутність резервних копій даних	Низька	Середній	2	1. Налаштувати автоматичне збереження даних. 2. Зберігати дані на різних носіях інформації.	Попередження	Робити копію даних після кожного виконаного етапу.
RS_12	Відкритий	Реалізація непотрібної функціональності	Низька	Низький	1	Попередити замовника про можливість додаткового функціоналу.	Використання	Обговорити вигоди і збитки від можливих змін проекту.

Продовження таблиці А.5 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_13	Відкритий	Невиконання моніторингу проекту	Середня	Низький	2	Здійснювати проміжний контроль результатів в ході виконання проекту. Здійснювати моніторинг проекту працівниками.	Перенос	Здійснювати моніторинг проекту замовником. Надання проміжних результатів виконання проекту після кожного етапу.
RS_14	Відкритий	Виникнення проблем із програмним забезпеченням користувачів	Середня	Високий	6	Розробка проекту з врахуванням вимог до програмного забезпечення користувачів проекту. Модифікація проекту з врахуванням різних версій програмного забезпечення, яке буде застосовуватися.	Прийняття	
RS_15	Відкритий	Зміна вимог замовника в процесі розробки проекту	Низька	Високий	3	Узгодити всі питання на початкових етапах, щоб мінімізувати кількість змін під час розробки.	Пом’якшення	Переоцінка проекту, кожного разу, коли вимоги змінюються.

ДОДАТОК Б
КОД РОЗРОБЛЕНОЇ СИСТЕМИ

Б.1 Класи моделей та контексту системи

Class Task

```
public class Task {
    int id;
    String name;
    int kalor;
    GregorianCalendar dateStart;
    int duration;
    GregorianCalendar dateFinish;
    int type_id;
    int date_id;



    Task(int id1, String name1, int a, GregorianCalendar s, int d, int type_id1){

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy MMM dd HH:mm:ss");

        id = id1;
        name=name1;
        kalor=a;
        dateStart=s;
        duration=d;
        dateFinish= (GregorianCalendar) s.clone();
        dateFinish.add(Calendar.MINUTE,duration);

        type_id = type_id1;
    }

    Task( String name1, int a, GregorianCalendar s, int d, int type_id1){

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy MMM dd HH:mm:ss");


        name=name1;
        kalor=a;
        dateStart=s;
        duration=d;
        dateFinish= (GregorianCalendar) s.clone();
        dateFinish.add(Calendar.MINUTE,duration);

        type_id = type_id1;
    }

    Task(int id1,String name1, int a, int d){

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy MMM dd HH:mm:ss");

        id = id1;
        name=name1;
        kalor=a;
        dateStart=new GregorianCalendar(2019, 1, 23, 20, 0, 0);
        duration=d;
        dateFinish= new GregorianCalendar(2019, 1, 23, 20, 0, 0);}
```

```
    @Override
    public String toString() {
        return      this.name      +      "      "      +      this.dateStart.get(Calendar.HOUR_OF_DAY)+"-
"+this.dateFinish.get(Calendar.HOUR_OF_DAY)+"   energy - "+this.kalor+" kalories";
    }

    public static Task getTsskNyId(int id, DBHelper dbHelper){

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "_id =?";
        String[] selectionArgs = {String.valueOf(id)};
        Cursor cursor = database.query(DBHelper.TABLE_TASK, null, selection, selectionArgs, null, null,
null);
        int count = cursor.getCount();
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;

        Task task = null;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
            int energyIndex = cursor.getColumnIndex(DBHelper.KEY_ENERGY_TOTAL);
            int dayIndex = cursor.getColumnIndex(DBHelper.KEY_DAY);
            int monthlIndex = cursor.getColumnIndex(DBHelper.KEY_MONTH);
            int yearIndex = cursor.getColumnIndex(DBHelper.KEY_YEAR);
            int hourIndex = cursor.getColumnIndex(DBHelper.KEY_HOUR);
            int minIndex = cursor.getColumnIndex(DBHelper.KEY_MINUTE);
            int durationIndex = cursor.getColumnIndex(DBHelper.KEY_DURATION);
            int typeIndex = cursor.getColumnIndex(DBHelper.KEY_ID_TYPE);
            do {

                //String     result     =     cursor.getString(nameIndex)     +     "                    "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                 task = new Task(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        Integer.parseInt(cursor.getString(energyIndex)),
                        new GregorianCalendar(
                                Integer.parseInt(cursor.getString(yearIndex)),
                                Integer.parseInt(cursor.getString(monthlIndex)),
                                Integer.parseInt(cursor.getString(dayIndex)),
                                Integer.parseInt(cursor.getString(hourIndex)),
                                Integer.parseInt(cursor.getString(minIndex)),
                                0),
                        Integer.parseInt(cursor.getString(durationIndex)),
                         cursor.getInt(typeIndex));
                day1.add(task);

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");


        cursor.close();

        dbHelper.close();

        return task;
    }
}
```

Class Sort

```
public class Sort {
    int id;
    String name;

    Sort(int id1, String name1){
        id=id1;
        name=name1;
    }

    Sort(String name1){
        name=name1;
    }

    @Override
    public String toString() {
        return this.name;
    }

    public static Sort getSortNyId(int id, DBHelper dbHelper){

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "_id =?";
        String[] selectionArgs = {String.valueOf(id)};
        Cursor cursor = database.query(DBHelper.TABLE_SORT, null, selection, selectionArgs, null, null,
null);
        int count = cursor.getCount();
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;

        Sort sort = null;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.SORT_KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.SORT_KEY_NAME);
            do {

                //String    result    =    cursor.getString(nameIndex)    +    "                  "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                sort = new Sort(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex));


            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");


        cursor.close();

        dbHelper.close();

        return sort;
    }
}
```

Class Type

```
public class Type {
    int id;
```

```
    String name;
    int energy;
    int sort_id;

    Type(int id1, String name1, int energy1, int sort_id1){
        id=id1;
        name=name1;
        energy=energy1;
        sort_id=sort_id1;
    }

    Type(String name1, int energy1, int sort_id1){
        name=name1;
        energy=energy1;
        sort_id=sort_id1;
    }

    @Override
    public String toString() {
        return this.name;
    }

    public static Type getTypeNyId(int id, DBHelper dbHelper){

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "_id =?";
        String[] selectionArgs = {String.valueOf(id)};
        Cursor cursor = database.query(DBHelper.TABLE_TYPE, null, selection, selectionArgs, null, null,
null);
        int count = cursor.getCount();
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;

        Type type = null;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.TYPE_KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.TYPE_NAME);
            int energyIndex = cursor.getColumnIndex(DBHelper.TYPE_ENERGY);
            int sortIdIndex = cursor.getColumnIndex(DBHelper.TYPE_SORT_ID);
            do {

                //String    result    =    cursor.getString(nameIndex)    +    "                    "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                type = new Type(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        cursor.getInt(energyIndex),
                        cursor.getInt(sortIdIndex));

                System.out.println("TYPE");
                System.out.println( cursor.getInt(idIndex));
                System.out.println( cursor.getString(nameIndex));
                System.out.println(  cursor.getInt(energyIndex));
                System.out.println( cursor.getInt(sortIdIndex));
            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");


        cursor.close();

        dbHelper.close();
```

```
        return type;
    }
}
```

Class Analysis

```
public class Analysis {

    static ArrayList<CalendarDay> lightRedDays ;
    static ArrayList<CalendarDay> lightYellowDays  ;;
    static ArrayList<CalendarDay> greenDays  ;;
    static ArrayList<CalendarDay> yellowDays  ;
    static ArrayList<CalendarDay> redDays  ;

    static Analysis[] array;


    int id;

    int year;
    int month;
    int day;

    int total_energy;
    int mark_total;

    Analysis(int year1, int month1, int day1, int total_rnrrgy1){
        id = 0;
        year = year1;
        month = month1;
        day = day1;
        total_energy = total_rnrrgy1;
        mark_total=0;
    }

    Analysis(int year1, int month1, int day1, int total_rnrrgy1, int mark){
        id = 0;
        year = year1;
        month = month1;
        day = day1;
        total_energy = total_rnrrgy1;
        mark_total=mark;
    }

    static protected void setPrimary(DBHelper dbHelper){

        LocalDate start = LocalDate.of(2019,1,1);
        LocalDate fin = LocalDate.of(2020,1,1);
        List<LocalDate> localDates = getDatesBetweenUsingJava8(start,fin);
        SQLiteDatabase database = dbHelper.getWritableDatabase();

        int i =0;
        ContentValues contentValues = new ContentValues();


                   for (LocalDate date : localDates) {
                contentValues.put(DBHelper.ANALYSIS_YEAR, date.getYear());
                contentValues.put(DBHelper.ANALYSIS_MONTH, date.getMonthValue());
                contentValues.put(DBHelper.ANALYSIS_DAY, date.getDayOfMonth());
                contentValues.put(DBHelper.ANALYSIS_TOTAL_ENERGY, 0);
                //contentValues.put(DBHelper.ANALYSIS_MARK_TOTAL_ENERGY, 0);



                database.insert(DBHelper.TABLE_ANALYSIS, null, contentValues);
                       System.out.println(i++);
            }
```

```
        dbHelper.close();
    }



    public static Analysis getAnalysisDay(DBHelper dbHelper, CalendarDay date ){


        int year = date.getYear();
        System.out.println("YEAR"+year);
        int month = date.getMonth();
        System.out.println("MONTH"+month);
        int day = date.getDay();
        System.out.println("DAY"+day);

        Analysis analysis = null;

        ArrayList<CalendarDay> days = new ArrayList<>();

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "day =? and month =? and year =?";
        String[] selectionArgs = {String.valueOf(day), String.valueOf(month), String.valueOf(year)};
        Cursor cursor = database.query(DBHelper.TABLE_ANALYSIS, null, selection, selectionArgs, null,
null, null);
        int count = cursor.getCount();
        System.out.println("Count"+count);
        int i=0;


        if (cursor.moveToFirst()) {

            int totalEnergyIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_TOTAL_ENERGY);
            int totalMarkIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_MARK_TOTAL_ENERGY);



            do {

                //String     result     =     cursor.getString(nameIndex)     +     "                  "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                analysis                                          =                                          new
Analysis(year,month,day,cursor.getInt(totalEnergyIndex),cursor.getInt(totalMarkIndex));
                System.out.println("TOTALENERTY  "+cursor.getInt(totalEnergyIndex));


            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();
        dbHelper.close();
        return analysis;

    }



    public static List<LocalDate> getDatesBetweenUsingJava8(LocalDate startDate, LocalDate endDate) {

        long numOfDaysBetween = ChronoUnit.DAYS.between(startDate, endDate);
        return IntStream.iterate(0, i -> i + 1)
                .limit(numOfDaysBetween)
                .mapToObj(i -> startDate.plusDays(i))
                .collect(Collectors.toList());
```

```
    }

    public static void getDays(DBHelper dbHelper ){

        System.out.println("getDays");
         lightRedDays = new ArrayList<>();
         lightYellowDays  = new ArrayList<>();;
         greenDays  = new ArrayList<>();;
        yellowDays  = new ArrayList<>();;
         redDays  = new ArrayList<>();;


        ArrayList<CalendarDay> days = new ArrayList<>();

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "mark_total_energy is not null";
        //String[] selectionArgs = {String.valueOf(redEnergy2), String.valueOf(redEnergy)};
        Cursor cursor = database.query(DBHelper.TABLE_ANALYSIS, null, selection, null, null, null, null);
        int count = cursor.getCount();
        array = new Analysis[count];
        System.out.println("Count mark_total_energy is not null "+count);
        int i=0;


        if (cursor.moveToFirst()) {


            int dayIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_DAY);
            int monthlIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_MONTH);
            int yearIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_YEAR);
            int energyIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_TOTAL_ENERGY);
            int markIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_MARK_TOTAL_ENERGY);

            do {

                //String     result     =     cursor.getString(nameIndex)     +     "                    "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                Analysis analysis = new Analysis(
                        cursor.getInt(yearIndex),
                        cursor.getInt(monthlIndex),
                        cursor.getInt(dayIndex),
                        cursor.getInt(energyIndex),
                        cursor.getInt(markIndex)
                );
                System.out.println(analysis.total_energy +"   " + analysis.mark_total);
            array[i]=analysis;
            i++;

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();


        if (count!=0) {

            contentValues = new ContentValues();

            selection = "year =?";
            String[] selectionArgs = {String.valueOf(2019)};
            cursor = database.query(DBHelper.TABLE_ANALYSIS, null, selection, selectionArgs, null, null,
null);
            count = cursor.getCount();
```

```
            System.out.println("Count" + count);
            i = 0;


            if (cursor.moveToFirst()) {

                int dayIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_DAY);
                int monthlIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_MONTH);
                int yearIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_YEAR);
                int energyIndex = cursor.getColumnIndex(DBHelper.ANALYSIS_TOTAL_ENERGY);


                do {

                    CalendarDay day = CalendarDay.from(
                            cursor.getInt(yearIndex),
                            cursor.getInt(monthlIndex),
                            cursor.getInt(dayIndex));
                    System.out.println("Every day   " + day.getYear() + " " + day.getMonth() + " " +
day.getDay());

                    //String    result    =    cursor.getString(nameIndex)    +    "                "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                    Analysis analysis = new Analysis(
                            cursor.getInt(yearIndex),
                            cursor.getInt(monthlIndex),
                            cursor.getInt(dayIndex),
                            cursor.getInt(energyIndex)
                    );

                    System.out.println("analysis energy = " + analysis.total_energy);

                    selectionSort(array, analysis);

                    int lightRed = 0;
                    int light_yellow = 0;
                    int green = 0;
                    int yellow = 0;
                    int red = 0;

                    for (int k = 0; k < 5; k++) {
                        if (array[k].mark_total == 1) lightRed++;
                        else if (array[k].mark_total == 2) light_yellow++;
                        else if (array[k].mark_total == 3) green++;
                        else if (array[k].mark_total == 4) yellow++;
                        else red++;
                    }
                    System.out.println("lightRed = "+lightRed);
                    System.out.println("light_yellow = "+light_yellow);
                    System.out.println("green = "+green);
                    System.out.println("yellow = "+yellow);
                    System.out.println("red = "+red);

                    int[] marks = new int[]{lightRed, light_yellow, green, yellow, red};
                    selectionSort(marks);
                    System.out.println("marks");



                    System.out.println("first mark");

                    for (int y:marks
                         ) {
                        System.out.println(y);
                    }
                    System.out.println(marks[4]);
                    System.out.println(lightRed);
```

```
                    if (marks[4] == lightRed) {
                        lightRedDays.add(CalendarDay.from(analysis.year,                analysis.month,
analysis.day));
                    } else if (marks[4] == light_yellow) {
                        lightYellowDays.add(CalendarDay.from(analysis.year,             analysis.month,
analysis.day));
                    } else if (marks[4] == green) {
                        greenDays.add(CalendarDay.from(analysis.year, analysis.month, analysis.day));
                    } else if (marks[4] == yellow) {
                        yellowDays.add(CalendarDay.from(analysis.year, analysis.month, analysis.day));
                    } else {
                        redDays.add(CalendarDay.from(analysis.year, analysis.month, analysis.day));
                    }


                } while (cursor.moveToNext());
            } else
                Log.d("mLog", "0 rows");

            cursor.close();
        }
        dbHelper.close();




    }


    public static void selectionSort(Analysis[] arr, Analysis classy){
    /*По очереди будем просматривать все подмножества
      элементов массива (0 - последний, 1-последний,
      2-последний,...)*/
        for (int i = 0; i < arr.length; i++) {
        /*Предполагаем, что первый элемент (в каждом
           подмножестве элементов) является минимальным */
            Analysis min = arr[i];
            int min_distance= Math.abs(classy.total_energy-arr[i].total_energy);
            int min_i = i;
            //System.out.println("min_i "+min_i);
        /*В оставшейся части подмножества ищем элемент,
           который меньше предположенного минимума*/
            for (int j = i+1; j < arr.length; j++) {
                int distance1  =  Math.abs(classy.total_energy-arr[j].total_energy);
                //System.out.println("current  " +distance1);
                //System.out.println("distance1 = " + distance1);
                //System.out.println("distance2 = " + distance2);
                //Если находим, запоминаем его индекс
                if (distance1 < min_distance) {
                    min = arr[j];
                    min_i = j;
                    min_distance = distance1;
                    // System.out.println("min = " + distance1);
                }
            }
        /*Если нашелся элемент, меньший, чем на текущей позиции,
          меняем их местами*/
            if (i != min_i) {
                Analysis tmp = arr[i];
                arr[i] = arr[min_i];
                arr[min_i] = tmp;
                //System.out.println(" Array arr["+i+"] = "+arr[i].total_energy);
            }
        }

    }
```

```
    public static void selectionSort(int[] arr){
    /*По очереди будем просматривать все подмножества
      элементов массива (0 - последний, 1-последний,
      2-последний,...)*/
        for (int i = 0; i < arr.length; i++) {
        /*Предполагаем, что первый элемент (в каждом
           подмножестве элементов) является минимальным */

            int min= arr[i];
            int min_i = i;
            //System.out.println("min_i "+min_i);
        /*В оставшейся части подмножества ищем элемент,
           который меньше предположенного минимума*/
            for (int j = i+1; j < arr.length; j++) {
                //Если находим, запоминаем его индекс
                if (arr[j] < arr[i]) {
                    min = arr[j];
                    min_i = j;
                    // System.out.println("min = " + distance1);
                }
            }
        /*Если нашелся элемент, меньший, чем на текущей позиции,
          меняем их местами*/
            if (i != min_i) {
                int tmp = arr[i];
                arr[i] = arr[min_i];
                arr[min_i] = tmp;
            }
        }

    }
```

Class BMREntity

```
public class BMREntity {

    int id ;
    int male ;
    int age ;
    int weight ;
    int height ;
    double koef;

    BMREntity(int id1 ,
            int male1 ,
            int age1 ,
            int weight1 ,
            int height1 ,
              double koef1){

        id=id1;
        male=male1;
        age=age1;
        weight=weight1;
        height=height1;
        koef=koef1;
    }

}
```

Б.2 Класи обробки Activity

Class CircleDecorator

```
public class CircleDecorator  implements DayViewDecorator {

    private HashSet<CalendarDay> dates;
    private Drawable drawable;

    public CircleDecorator(Context context, int resId, Collection<CalendarDay> dates) {
        drawable = ContextCompat.getDrawable(context, resId);
        this.dates = new HashSet<>(dates);
    }

    @Override
    public boolean shouldDecorate(CalendarDay day) {
        return dates.contains(day);
    }

    @Override
    public void decorate(DayViewFacade view) {
        view.addSpan(new ForegroundColorSpan(Color.BLACK));
        view.setBackgroundDrawable(drawable);
    }
}
```

Class ScheduleActivity

```
package com.example.myapplication;


public class ScheduleActivity extends AppCompatActivity {

    ArrayList<Task> dayTasks;

    int DIALOG_DATE_Start = 1;
    int DIALOG_DATE_Finish = 2;
    int DIALOG_DATE_Proposed = 4;

    int myYearStart;
    int myMonthStart;
    int myDayStart;

    int myYearFinish;
    int myMonthFinish;
    int myDayFinish;

    int myYearCurrent;
    int myMonthCurrent;
    int myDayCurrent;

    TextView startDate;
    TextView finishDate;

    Button schedule;
    Button save;

    TextView name;
    TextView duration;
    EditText duratiomMinute;
    TextView energy;

    Spinner sortSpinner;
    Spinner typeSpinner;
```

```
static  Sort sort;
static Type type;

ArrayList<Sort> sorts;
ArrayList<Type> types;

static DBHelper dbHelper;

int TotalDayActivity;

TextView proposedTime;
TextView proposedDate;

int day;
int month;
int year;

int DIALOG_TIME = 3;
int myHour = 12;
int myMinute = 0;

int hpurProposed;
int minuteProposed;
boolean timeProposed=false;

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


    startDate = (TextView) findViewById(R.id.startDate);
    finishDate = (TextView) findViewById(R.id.finishDate);

    proposedTime = (TextView) findViewById(R.id.proposedTime);
    proposedDate = (TextView) findViewById(R.id.proposedDate);

    Calendar c = Calendar.getInstance();
     myYearCurrent = c.get(Calendar.YEAR);
     myMonthCurrent = c.get(Calendar.MONTH);
     myDayCurrent = c.get(Calendar.DAY_OF_MONTH);

     myYearStart=myYearCurrent;
     myMonthStart=myMonthCurrent;
     myDayStart=myDayCurrent;

    startDate.setText(myDayStart + "/" + (myMonthStart+1) + "/" + myYearStart);

    c.add(Calendar.DAY_OF_MONTH, 7);
    myYearFinish = c.get(Calendar.YEAR);
    myMonthFinish = c.get(Calendar.MONTH);
    myDayFinish = c.get(Calendar.DAY_OF_MONTH);

    finishDate.setText(myDayFinish + "/" + (myMonthFinish+1) + "/" + myYearFinish);


     dbHelper = new DBHelper(this);



    name = findViewById(R.id.nameEdit);
    duration = findViewById(R.id.duration);
    duratiomMinute=findViewById(R.id.durationMinute);
    energy = findViewById(R.id.energyEdit);

    schedule = findViewById(R.id.scheduleButton);
    save = findViewById(R.id.saveButton);
```

```
        schedule.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (duration.getText().toString().equals("")
                )
                {
                    duration.setText("0");
                }
                if (duratiomMinute.getText().toString().equals("")
                )
                {
                    duratiomMinute.setText("0");
                }

                int hours = Integer.parseInt(duration.getText().toString());

                int minutes = Integer.parseInt(duratiomMinute.getText().toString());
                int durationInMinutes = hours*60+minutes;


                LocalDate start = LocalDate.of(myYearStart,myMonthStart,myDayStart);
                LocalDate finish = LocalDate.of(myYearFinish,myMonthFinish,myDayFinish);

                ArrayList <ArrayList<Task>> desiredDays = new ArrayList<ArrayList<Task>>();


                List<LocalDate> localDates = getDatesBetweenUsingJava8(start,finish);
                for (LocalDate date:localDates
                ) {
                    desiredDays.add(getList(date));
                }

                String nameString =  String.valueOf(name.getText());
                Log.d("m.log", "namestring  "+nameString);
                int energyInt = Integer.parseInt(String.valueOf(energy.getText()));
                Log.d("m.log", "energyInt  "+energyInt);
                int durationInt =  Integer.parseInt(String.valueOf(duration.getText()));
                Log.d("m.log", "durationInt  "+durationInt);
                Task task4 = new Task(0,nameString, energyInt, durationInMinutes);
                GregorianCalendar show = insert5(desiredDays,task4);
                if (show==null){
                    AlertDialog.Builder builder = new AlertDialog.Builder(schedule.getContext());
                    GregorianCalendar finalShow = show;
                    builder.setTitle("Activity "+task4.name+" cant be scheduled, ")
                            .setMessage("no free time window")
                            .setCancelable(false)
                            .setNegativeButton("OK",
                                    new DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();
                                        }
                                    });
                    AlertDialog alert = builder.create();
                    alert.show();
                    return;
                }
                String                                          date                                                =
show.get(Calendar.DAY_OF_MONTH)+"/"+(show.get(Calendar.MONTH)+1)+"/"+show.get(Calendar.YEAR);


                findTimeTask(nameString);
                System.out.println("timeProposed "+timeProposed);
                if (timeProposed){
                    int yearOld = show.get(Calendar.YEAR);
                    int monthOld = show.get(Calendar.MONTH);
                    int dayOld = show.get(Calendar.DAY_OF_MONTH);
                    GregorianCalendar                   newDate                   =                   new
GregorianCalendar(yearOld,monthOld,dayOld,hpurProposed,minuteProposed);
```

```
                    Task taskForInsert = new Task(
                            name.getText().toString(),
                            Integer.parseInt(energy.getText().toString()),
                            newDate,
                            Integer.parseInt(duration.getText().toString()),
                            type.id);

                    year = yearOld;
                    month = monthOld;
                    day = dayOld;

                    System.out.println("proposed day "+day);


                    dayTasks=getDayTask();

                    boolean insertable = allowToInsertTaskInDay(dayTasks,taskForInsert);
                    System.out.println("Insertable "+insertable);
                    if (insertable){
                        show=newDate;
                    }
                }


                AlertDialog.Builder builder = new AlertDialog.Builder(schedule.getContext());
                GregorianCalendar finalShow = show;
                builder.setTitle("Activity "+task4.name+" was scheduled to")
                        .setMessage(date)
                        .setCancelable(false)
                        .setNegativeButton("OK",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int id) {
                                        dialog.cancel();
                                        /*DBHelper dbHelper = new DBHelper(getBaseContext());

                                        SQLiteDatabase database = dbHelper.getWritableDatabase();

                                        ContentValues contentValues = new ContentValues();

                                        int year = task4.dateStart.get(Calendar.YEAR);
                                        int month = task4.dateStart.get(Calendar.MONTH)+1;
                                        Log.d("monthScheduled"," "+month);
                                        int day = task4.dateStart.get(Calendar.DAY_OF_MONTH);

                                        int totalDay = getTotalDayActivity(year,month,day);

                                        String strSQL = "UPDATE "+DBHelper.TABLE_ANALYSIS+" SET
total_energy = "+totalDay+" WHERE year = "+ year+" and month = "+month+" and day = "+ day;

                                        database.execSQL(strSQL);

                                        dbHelper.close();

                                        Intent intent = new Intent();
                                        setResult(RESULT_OK,intent);
                                        finish();*/
                                        String time = finalShow.get(Calendar.HOUR_OF_DAY)+":"+
finalShow.get(Calendar.MINUTE);
                                        proposedTime.setText(time);
                                        String date2 =
finalShow.get(Calendar.DAY_OF_MONTH)+"/"+(finalShow.get(Calendar.MONTH)+1)+"/"+
finalShow.get(Calendar.YEAR);
                                        proposedDate.setText(date2);

                                        year = (finalShow.get(Calendar.YEAR));
                                        month = (finalShow.get(Calendar.MONTH));
                                        day = (finalShow.get(Calendar.DAY_OF_MONTH));
```

```
                                    myHour = finalShow.get(Calendar.HOUR_OF_DAY);
                                    myMinute = finalShow.get(Calendar.MINUTE);
                                    System.out.println("mmmmmmmmmmmmmmmmmmm  "+month);
                                }
                            });
                AlertDialog alert = builder.create();
                alert.show();
              /*      schedule.setEnabled(false);
                startDate.setEnabled(false);
                finishDate.setEnabled(false);*/
                proposedTime.setEnabled(true);
                proposedDate.setEnabled(true);
                save.setEnabled(true);



            }
        });

        save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                if (duration.getText().toString().equals("")
                )
                {
                    duration.setText("0");
                }
                if (duratiomMinute.getText().toString().equals("")
                )
                {
                    duratiomMinute.setText("0");
                }

                int hours = Integer.parseInt(duration.getText().toString());

                int minutes = Integer.parseInt(duratiomMinute.getText().toString());
                int durationInMinutes = hours*60+minutes;

                dayTasks=getDayTask();

                SQLiteDatabase database = dbHelper.getWritableDatabase();

                ContentValues contentValues = new ContentValues();


                Task taskForInsert = new Task(
                        name.getText().toString(),
                        Integer.parseInt(energy.getText().toString()),
                        new GregorianCalendar(
                                (year),
                                (month),
                                (day),
                                (myHour),
                                myMinute),
                        durationInMinutes,
                        type.id);

                boolean insertable = allowToInsertTaskInDay(dayTasks,taskForInsert);
                if (insertable){
                    contentValues.put(DBHelper.KEY_NAME, name.getText().toString());
                    contentValues.put(DBHelper.KEY_ENERGY_TOTAL,
Integer.parseInt(energy.getText().toString()));
                    contentValues.put(DBHelper.KEY_DAY, day);
                    contentValues.put(DBHelper.KEY_MONTH, month+1);
                    contentValues.put(DBHelper.KEY_YEAR, year);
                    contentValues.put(DBHelper.KEY_HOUR, myHour);
```

```
                    contentValues.put(DBHelper.KEY_MINUTE, myMinute);
                    contentValues.put(DBHelper.KEY_DURATION, durationInMinutes);
                    contentValues.put(DBHelper.KEY_ID_TYPE, type.id);

                    database.insert(DBHelper.TABLE_TASK, null, contentValues);
                    dbHelper.close();


                    DBHelper dbHelper = new DBHelper(getBaseContext());

                     database = dbHelper.getWritableDatabase();

                     contentValues = new ContentValues();



                    int totalDay = getTotalDayActivity(year,month,day);
                    System.out.println("TOTALDAYSCHEDULING "+totalDay);
                    System.out.println("year "+year);
                    System.out.println("month "+month);
                    System.out.println("day "+day);

                    String strSQL = "UPDATE "+DBHelper.TABLE_ANALYSIS+" SET total_energy = "+totalDay+"
WHERE year = "+ year+" and month = "+(month+1)+" and day = "+ day;

                    database.execSQL(strSQL);

                    dbHelper.close();


                    Intent intent = new Intent();
                    setResult(RESULT_OK,intent);
                    finish();



                }

                else{
                    AlertDialog.Builder builder = new AlertDialog.Builder(save.getContext());
                    builder.setTitle("Error")
                            .setMessage("the timeframe is not free")
                            .setCancelable(false)
                            .setNegativeButton("OK",
                                    new DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();
                                        }
                                    });
                    AlertDialog alert = builder.create();
                    alert.show();

                }
            }
        });



        sortSpinner=findViewById(R.id.spinnerSort);
        typeSpinner=findViewById(R.id.spinnerType);

        sorts=getSortList();

        ArrayAdapter<String>          adapter          =          new          ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, getSortNames());
        sortSpinner.setAdapter(adapter);
```

```
        sortSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            public    void    onItemSelected(AdapterView<?>    parent,    View    itemSelected,    int
selectedItemPosition, long selectedId) {
                Log.d("selectedItemPosition",""+selectedItemPosition);

                String name = sortSpinner.getSelectedItem().toString();
                Log.d("Log","111111111111111");
                sort = getSortFromSortName(name,sorts);
                Log.d("Log","222222222222222");
                Log.d("sort", sort.name);
                types=getTypeList();
                Log.d("Log","3333333333333");
                getTypeNames();
                Log.d("Log","444444444");

                ArrayAdapter<String>    adapter2    =    new    ArrayAdapter<String>(getBaseContext(),
android.R.layout.simple_list_item_1,  getTypeNames());
                Log.d("Log","5555555555");
                typeSpinner.setAdapter(adapter2);




                typeSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
                    public   void   onItemSelected(AdapterView<?>   parent,   View   itemSelected,   int
selectedItemPosition, long selectedId) {

                        Log.d("", "onItemSelected: "+typeSpinner.getSelectedItem().toString());
                        type=getTypeFromTypeName(typeSpinner.getSelectedItem().toString(),types);
                        int dur = Integer.parseInt(duration.getText().toString());
                        int durM = Integer.parseInt(duratiomMinute.getText().toString());
                        Log.d("dur", " = "+dur);
                        Log.d("type.energy", " = "+type.energy);

                        energy.setText(String.valueOf(type.energy*dur+durM*type.energy/60));


                    }
                    public void onNothingSelected(AdapterView<?> parent) {
                    }
                });



            }
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });

        /*duration.addTextChangedListener(new TextWatcher() {

            @Override
            public void afterTextChanged(Editable s) {}

            @Override
            public void beforeTextChanged(CharSequence s, int start,
                                          int count, int after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start,
                                      int before, int count) {
                if(s.length() != 0)
```

```
        {
            int dur = Integer.parseInt(duration.getText().toString());

            energy.setText(String.valueOf(type.energy*dur));
        }
        else  energy.setText(String.valueOf(0));
    }
});*/

duration.addTextChangedListener(new TextWatcher() {

    @Override
    public void afterTextChanged(Editable s) {}

    @Override
    public void beforeTextChanged(CharSequence s, int start,
                                  int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start,
                              int before, int count) {
        int durMinutes = 0;
        try { durMinutes = Integer.parseInt(duratiomMinute.getText().toString());}
        catch (Exception e){
        }
        int calMin = type.energy*durMinutes/60;
        if(s.length() != 0)
        {
            int dur = Integer.parseInt(duration.getText().toString());

            energy.setText(String.valueOf(type.energy*dur+calMin));
        }
        else  energy.setText(String.valueOf(calMin));
    }
});

duratiomMinute.addTextChangedListener(new TextWatcher() {

    @Override
    public void afterTextChanged(Editable s) {}

    @Override
    public void beforeTextChanged(CharSequence s, int start,
                                  int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start,
                              int before, int count) {
        int durHours = 0;
        try { durHours = Integer.parseInt(duration.getText().toString());}
        catch (Exception e){
        }
        int calHours = type.energy*durHours;

        //energyEdit.setText(String.valueOf(calHours));

        if(s.length() != 0)
        {
            int dur = Integer.parseInt(duratiomMinute.getText().toString());

            int additionalCal = dur*type.energy/60;

            int totalCal = calHours+additionalCal;

            energy.setText(String.valueOf(totalCal));
        }
```

```
                else  energy.setText(String.valueOf(calHours));
            }
        });

    }



    public void onclick(View view) {
        if (view.getId()==R.id.startDate) {
            showDialog(DIALOG_DATE_Start);
        }
        if (view.getId()==R.id.finishDate) {
            showDialog(DIALOG_DATE_Finish);
        }
        if (view.getId()==R.id.proposedTime) {
            showDialog(DIALOG_TIME);
        }
        if (view.getId()==R.id.proposedDate) {
            showDialog(DIALOG_DATE_Proposed);
            System.out.println("proposed day picker"+year);
            System.out.println("proposed day picker"+month);
            System.out.println("proposed day picker"+day);
        }
    }


    protected Dialog onCreateDialog(int id) {
        if (id == DIALOG_DATE_Start) {
            DatePickerDialog tpd = new DatePickerDialog(this, myCallBack, myYearStart, myMonthStart,
myDayStart);
            tpd.getDatePicker().setFirstDayOfWeek(2);

            return tpd;
        }
        if (id == DIALOG_DATE_Finish) {
            DatePickerDialog tpd = new DatePickerDialog(this, myCallBack2, myYearFinish, myMonthFinish,
myDayFinish);
            tpd.getDatePicker().setFirstDayOfWeek(2);
            return tpd;
        }
        if (id == DIALOG_TIME) {
            TimePickerDialog tpd = new TimePickerDialog(this, myCallBack3, myHour, myMinute, true);
            return tpd;
        }
        if (id == DIALOG_DATE_Proposed) {

            DatePickerDialog tpd = new DatePickerDialog(this, myCallBack4, year, month, day);
            tpd.getDatePicker().setFirstDayOfWeek(2);

            return tpd;
        }
        return super.onCreateDialog(id);
    }

    TimePickerDialog.OnTimeSetListener myCallBack3 = new TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            myHour = hourOfDay;
            myMinute = minute;
            proposedTime.setText( myHour + ":" + myMinute );
        }
    };

    DatePickerDialog.OnDateSetListener myCallBack = new DatePickerDialog.OnDateSetListener() {

        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            myYearStart = year;
```

```
        myMonthStart = monthOfYear;
        myDayStart = dayOfMonth;
        startDate.setText(myDayStart + "/" + (myMonthStart+1) + "/" + myYearStart);
    }
};

DatePickerDialog.OnDateSetListener myCallBack2 = new DatePickerDialog.OnDateSetListener() {

    public void onDateSet(DatePicker view, int year, int monthOfYear,
                          int dayOfMonth) {
        myYearFinish = year;
        myMonthFinish = monthOfYear;
        myDayFinish = dayOfMonth;
        finishDate.setText(myDayFinish + "/" + (myMonthFinish+1) + "/" + myYearFinish);
    }
};

DatePickerDialog.OnDateSetListener myCallBack4 = new DatePickerDialog.OnDateSetListener() {

    public void onDateSet(DatePicker view, int year1, int monthOfYear1,
                          int dayOfMonth1) {
        year = year1;
        month = monthOfYear1;
        day = dayOfMonth1;
        proposedDate.setText(day + "/" + (month+1) + "/" + year);
    }
};



public static GregorianCalendar  insert5 (ArrayList <ArrayList<Task>> desiredDays, Task task4){


    ArrayList<Task>[] arrayLists= new ArrayList[desiredDays.size()];
    Log.d("m.log", "size  "+desiredDays.size());
    int countArray = 0;
    for (ArrayList<Task> arrayList:desiredDays
         ) {
        arrayLists[countArray]=arrayList;
        countArray++;
    }

    //find days with frames
    int countDaysWithFrames=0;
    for (int i=0; i<arrayLists.length;i++){
        if (duration(arrayLists[i],task4)) countDaysWithFrames++;
    }
    if (countDaysWithFrames==0) {
        System.out.println("cant insert");
        return null;
    }
    else {

        ArrayList[] arrayListsFrames = new ArrayList[countDaysWithFrames];
        countDaysWithFrames = 0;
        for (int i = 0; i < arrayLists.length; i++) {
            if (duration(arrayLists[i], task4)) {
                arrayListsFrames[countDaysWithFrames] = arrayLists[i];
                countDaysWithFrames++;
            }
        }

        for (int i = 0; i < arrayListsFrames.length; i++) {
    /*Предполагаем, что первый элемент (в каждом
       подмножестве элементов) является минимальным */
            ArrayList min = arrayListsFrames[i];
            int min_i = i;
    /*В оставшейся части подмножества ищем элемент,
```

```
           который меньше предположенного минимума*/
                for (int j = i + 1; j < arrayListsFrames.length; j++) {
                    //Если находим, запоминаем его индекс
                    if (energyFactDay(arrayListsFrames[j]) < energyFactDay(arrayListsFrames[min_i])) {
                        min = arrayListsFrames[j];
                        min_i = j;
                    }
                }
            /*Если нашелся элемент, меньший, чем на текущей позиции,
              меняем их местами*/
                if (i != min_i) {
                    ArrayList tmp = arrayListsFrames[i];
                    arrayListsFrames[i] = arrayListsFrames[min_i];
                    arrayListsFrames[min_i] = tmp;
                }
            }

            ArrayList<Task> insertDay=arrayListsFrames[0];
            Log.d("m.log", "task4.dateStart  "+task4.dateStart);
            task4.dateStart=getFirstDuration(insertDay,task4);
            Log.d("m.log", "getFirstDuration  "+task4.dateStart);
            task4 = new Task(0,task4.name, task4.kalor, task4.dateStart, task4.duration,type.id);
            insertDay.add(task4);
            //addTaskInDay(task4,task4.dateStart);
            insertDay.sort(new Comparator<Task>() {
                public int compare(Task one, Task other) {
                    return one.dateStart.compareTo(other.dateStart);
                }
            });

            SimpleDateFormat sdf = new SimpleDateFormat("MMM dd ");
            System.out.println("*************Insert      "+task4.name+"   "+task4.duration+"   hours   "
+task4.kalor+" calories "+" in "+sdf.format(insertDay.get(0).dateStart.getTime()));
            //writeDay(insertDay);


        }

        /*String dateYear= String.valueOf(task4.dateStart.get(Calendar.YEAR));
        String dateYear= String.valueOf(task4.dateStart.get(Calendar.YEAR));
        String dateYear= String.valueOf(task4.dateStart.get(Calendar.YEAR));*/

        String date = String.valueOf(task4.dateStart.getTime());
        Log.d("m.log", "date  "+date);
        return task4.dateStart;
    }

    public static boolean duration (ArrayList<Task> tasks, Task task4){

        //System.out.println(tasks.size());
        boolean exist=false;
        Log.d("m.log", "tasks.size()  "+tasks.size());
        for (int i=1; i<tasks.size();i++)
        {
            Log.d("m.log",                              "tasks.get(i).dateStart.get(Calendar.HOUR_OF_DAY
"+tasks.get(i).dateStart.get(Calendar.HOUR_OF_DAY));
            Log.d("m.log", "ttask4.duration  "+task4.duration);
            Log.d("m.log",    "tasks.get(i-1).dateFinish.get(Calendar.HOUR_OF_DAY)       "+tasks.get(i-
1).dateFinish.get(Calendar.HOUR_OF_DAY));
            /*if         (tasks.get(i).dateStart.get(Calendar.HOUR_OF_DAY)-task4.duration>=tasks.get(i-
1).dateFinish.get(Calendar.HOUR_OF_DAY))
            {//System.out.println("Между    "+tasks.get(i-1).name+"    и    "+tasks.get(i).name+"    есть
"+task4.duration+" hours" );
                exist=true;
            }*/

            GregorianCalendar gregorianCalendar = (GregorianCalendar) tasks.get(i-1).dateFinish.clone();
            gregorianCalendar.add(Calendar.MINUTE,task4.duration);
```

```
            if
(tasks.get(i).dateStart.compareTo(gregorianCalendar)==0||tasks.get(i).dateStart.compareTo(gregorianCale
ndar)==1)
            {//System.out.println("Между    "+tasks.get(i-1).name+"    и    "+tasks.get(i).name+"    есть
"+task4.duration+" hours" );
                exist=true;
            }
            //else   System.out.println("Между   "+tasks.get(i-1).name+"   и   "+tasks.get(i).name+"   нет
"+task4.duration+" hours" );
        }
        return exist;
    }

    public static GregorianCalendar getFirstDuration (ArrayList<Task> tasks, Task task4){

        //System.out.println("getFirstDuration");
        //System.out.println(tasks.size());
        GregorianCalendar exist=null;
        for (int i=1; i<tasks.size();i++)
        {
            GregorianCalendar gregorianCalendar = (GregorianCalendar) tasks.get(i-1).dateFinish.clone();
            gregorianCalendar.add(Calendar.MINUTE,task4.duration);
            if
(tasks.get(i).dateStart.compareTo(gregorianCalendar)==0||tasks.get(i).dateStart.compareTo(gregorianCale
ndar)==1)
            {//System.out.println("Между    "+tasks.get(i-1).name+"    и    "+tasks.get(i).name+"    есть
"+task4.duration+" hours" );
                exist=tasks.get(i-1).dateFinish;
                //System.out.println(exist.get(Calendar.HOUR_OF_DAY));
                return exist;
            }
            //else   System.out.println("Между   "+tasks.get(i-1).name+"   и   "+tasks.get(i).name+"   нет
"+task4.duration+" hours" );
        }

        return exist;
    }

    public static int energyFactDay(ArrayList<Task> tasks){
        int energy=0;
        for (Task t:tasks
        ) {
            energy+=t.kalor;
        }
        return  energy;
    }


    public static List<LocalDate> getDatesBetweenUsingJava8(LocalDate startDate, LocalDate endDate) {

        endDate=endDate.plusDays(1);
        long numOfDaysBetween = ChronoUnit.DAYS.between(startDate, endDate);
        return IntStream.iterate(0, i -> i + 1)
                .limit(numOfDaysBetween)
                .mapToObj(i -> startDate.plusDays(i))
                .collect(Collectors.toList());
    }


    public ArrayList<Task> getList(LocalDate date) {

        int year = date.getYear();
        Log.d("m.log", "year  "+year);
        int month = date.getMonthValue()+1;
        Log.d("m.log", "month  "+month);
        int day = date.getDayOfMonth();
        Log.d("m.log", "day  "+day);
```

```
        DBHelper dbHelper = new DBHelper(this);

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "day =? and month =? and year =?";
        String[] selectionArgs = {String.valueOf(day),String.valueOf(month),String.valueOf(year)};
        Cursor cursor = database.query(DBHelper.TABLE_TASK, null, selection, selectionArgs, null, null,
null);
        int count = cursor.getCount();
        Log.d("m.log", "cursor.getCount()  "+count);
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
            int emailIndex = cursor.getColumnIndex(DBHelper.KEY_ENERGY_TOTAL);
            int dayIndex = cursor.getColumnIndex(DBHelper.KEY_DAY);
            int monthlIndex = cursor.getColumnIndex(DBHelper.KEY_MONTH);
            int yearIndex = cursor.getColumnIndex(DBHelper.KEY_YEAR);
            int hourIndex = cursor.getColumnIndex(DBHelper.KEY_HOUR);
            int minIndex = cursor.getColumnIndex(DBHelper.KEY_MINUTE);
            int durationIndex = cursor.getColumnIndex(DBHelper.KEY_DURATION);
            int typeIndex = cursor.getColumnIndex(DBHelper.KEY_ID_TYPE);
            do {
                Log.d("mLog", "ID = " + cursor.getInt(idIndex) +
                        ", name = " + cursor.getString(nameIndex) +
                        ", email = " + cursor.getString(emailIndex) +
                        ", day = " + cursor.getString(dayIndex) +
                        ", month = " + cursor.getString(monthlIndex) +
                        ", year = " + cursor.getString(yearIndex) +
                        ", hour = " + cursor.getString(hourIndex) +
                        ", min = " + cursor.getString(minIndex) +
                        ", duration = " + cursor.getString(durationIndex));
                //String     result     =     cursor.getString(nameIndex)     +     "                    "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                Task task = new Task(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        Integer.parseInt(cursor.getString(emailIndex)),
                        new GregorianCalendar(
                                Integer.parseInt(cursor.getString(yearIndex)),
                                Integer.parseInt(cursor.getString(monthlIndex))-1,
                                Integer.parseInt(cursor.getString(dayIndex)),
                                Integer.parseInt(cursor.getString(hourIndex)),
                                Integer.parseInt(cursor.getString(minIndex)),
                                0),
                        Integer.parseInt(cursor.getString(durationIndex)),
                        cursor.getInt(typeIndex));
                day1.add(task);

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();

        day1.sort(new Comparator<Task>() {
            public int compare(Task one, Task other) {
                return one.dateStart.compareTo(other.dateStart);
            }
        });
        dbHelper.close();
```

```
        return day1;
    }

    public static void addTaskInDay(Task task, GregorianCalendar calendar){



        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        contentValues.put(DBHelper.KEY_NAME, task.name);
        contentValues.put(DBHelper.KEY_ENERGY_TOTAL, task.kalor);
        contentValues.put(DBHelper.KEY_DAY, calendar.get(Calendar.DAY_OF_MONTH));
        contentValues.put(DBHelper.KEY_MONTH, calendar.get(Calendar.MONTH)+1);
        contentValues.put(DBHelper.KEY_YEAR, calendar.get(Calendar.YEAR));
        contentValues.put(DBHelper.KEY_HOUR, calendar.get(Calendar.HOUR_OF_DAY));
        contentValues.put(DBHelper.KEY_MINUTE, calendar.get(Calendar.MINUTE));
        contentValues.put(DBHelper.KEY_DURATION, task.duration);
        contentValues.put(DBHelper.KEY_ID_TYPE, type.id);

        database.insert(DBHelper.TABLE_TASK, null, contentValues);
        dbHelper.close();


    }



    public ArrayList<Sort> getSortList() {



        DBHelper dbHelper = new DBHelper(this);

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();


        Cursor cursor = database.query(DBHelper.TABLE_SORT, null, null, null, null, null, null);
        int count = cursor.getCount();
        ArrayList<Sort> day1 = new ArrayList<Sort>();
        int i=0;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.SORT_KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.SORT_KEY_NAME);

            do {
                Log.d("mLog1", "ID = " + cursor.getInt(idIndex) +
                        ", name = " + cursor.getString(nameIndex));
                //String     result     =     cursor.getString(nameIndex)     +     "                    "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                Sort task = new Sort(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex));

                day1.add(task);

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");
```

```
        cursor.close();

        dbHelper.close();

        return day1;
    }
    public ArrayList<String> getSortNames(){

        ArrayList<String> sortNames = new ArrayList<>();
        ArrayList<Sort> list = getSortList();
        for (Sort sort:list
        ) {
            sortNames.add(sort.toString());
        }
        return sortNames;
    }
    public Sort getSortFromSortName(String name, ArrayList<Sort> list ){

        Sort sort = null;
        for (Sort sort2:list
        ) {
            if (sort2.name.equals(name)) sort=sort2;
        }
        Log.d("getSortFromSortName", sort.name);
        return sort;
    }

    public ArrayList<Type> getTypeList() {



        DBHelper dbHelper = new DBHelper(this);

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "sort_id =?";
        String[] selectionArgs = {String.valueOf(sort.id)};
        Cursor cursor = database.query(DBHelper.TABLE_TYPE, null, selection, selectionArgs, null, null,
"name");
        int count = cursor.getCount();
        ArrayList<Type> day1 = new ArrayList<Type>();
        int i=0;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.TYPE_KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
            int energyIndex = cursor.getColumnIndex(DBHelper.TYPE_ENERGY);
            int sortIndex = cursor.getColumnIndex(DBHelper.TYPE_SORT_ID);
            do {


                Type task = new Type(
                        Integer.parseInt(cursor.getString(idIndex)),
                        cursor.getString(nameIndex),
                        Integer.parseInt(cursor.getString(energyIndex)),
                        Integer.parseInt(cursor.getString(sortIndex)));
                day1.add(task);

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();

        dbHelper.close();
```

```
        return day1;
    }
    public ArrayList<String> getTypeNames(){

        ArrayList<String> typeNames = new ArrayList<>();
        ArrayList<Type> list = getTypeList();
        for (Type type:list
        ) {

            typeNames.add(type.toString());
            Log.d("getTypeNames1", type.name);
        }
        return typeNames;
    }
    public Type getTypeFromTypeName(String name, ArrayList<Type> list ){

        Type type = null;
        for (Type type2:list
        ) {
            if (type2.name.equals(name)) type=type2;
        }
        return type;
    }


    public int getTotalDayActivity(int year, int month, int day) {


        System.out.println("getTotalDayActivity "+year+" "+month+" "+day);
        DBHelper dbHelper = new DBHelper(this);

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "day =? and month =? and year =?";
        String[] selectionArgs = {String.valueOf(day), String.valueOf(month+1), String.valueOf(year)};
        Cursor cursor = database.query(DBHelper.TABLE_TASK, null, selection, selectionArgs, null, null,
null);
        int count = cursor.getCount();
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
            int energyIndex = cursor.getColumnIndex(DBHelper.KEY_ENERGY_TOTAL);
            int dayIndex = cursor.getColumnIndex(DBHelper.KEY_DAY);
            int monthlIndex = cursor.getColumnIndex(DBHelper.KEY_MONTH);
            int yearIndex = cursor.getColumnIndex(DBHelper.KEY_YEAR);
            int hourIndex = cursor.getColumnIndex(DBHelper.KEY_HOUR);
            int minIndex = cursor.getColumnIndex(DBHelper.KEY_MINUTE);
            int durationIndex = cursor.getColumnIndex(DBHelper.KEY_DURATION);
            int typeIndex = cursor.getColumnIndex(DBHelper.KEY_ID_TYPE);
            do {
                Log.d("mLog", "ID = " + cursor.getInt(idIndex) +
                        ", name = " + cursor.getString(nameIndex) +
                        ", email = " + cursor.getString(energyIndex) +
                        ", day = " + cursor.getString(dayIndex) +
                        ", month = " + cursor.getString(monthlIndex) +
                        ", year = " + cursor.getString(yearIndex) +
                        ", hour = " + cursor.getString(hourIndex) +
                        ", min = " + cursor.getString(minIndex) +
                        ", duration = " + cursor.getString(durationIndex));
                //String     result     =     cursor.getString(nameIndex)     +     "                  "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;
```

```
                Task task = new Task(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        Integer.parseInt(cursor.getString(energyIndex)),
                        new GregorianCalendar(
                                Integer.parseInt(cursor.getString(yearIndex)),
                                Integer.parseInt(cursor.getString(monthlIndex)),
                                Integer.parseInt(cursor.getString(dayIndex)),
                                Integer.parseInt(cursor.getString(hourIndex)),
                                Integer.parseInt(cursor.getString(minIndex)),
                                0),
                        Integer.parseInt(cursor.getString(durationIndex)),
                        cursor.getInt(typeIndex));;
                day1.add(task);

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();

        day1.sort(new Comparator<Task>() {
            public int compare(Task one, Task other) {
                return one.dateStart.compareTo(other.dateStart);
            }
        });

        TotalDayActivity = 0;

        dbHelper.close();

        String [] taskString = new String[count];
        int k=0;
        for (Task t : day1
        ) {
            TotalDayActivity+=t.kalor;
            taskString[k]=t.toString();
            k++;
        }

        return TotalDayActivity;
    }

    public  ArrayList<Task> getDayTask(){

        System.out.println("getDayTask");
        DBHelper dbHelper = new DBHelper(this);

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "day =? and month =? and year =?";
        String[] selectionArgs = {String.valueOf(day), String.valueOf(month+1), String.valueOf(year)};
        Cursor cursor = database.query(DBHelper.TABLE_TASK, null, selection, selectionArgs, null, null,
null);
        int count = cursor.getCount();
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;


        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
            int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
            int energyIndex = cursor.getColumnIndex(DBHelper.KEY_ENERGY_TOTAL);
            int dayIndex = cursor.getColumnIndex(DBHelper.KEY_DAY);
            int monthlIndex = cursor.getColumnIndex(DBHelper.KEY_MONTH);
```

```
            int yearIndex = cursor.getColumnIndex(DBHelper.KEY_YEAR);
            int hourIndex = cursor.getColumnIndex(DBHelper.KEY_HOUR);
            int minIndex = cursor.getColumnIndex(DBHelper.KEY_MINUTE);
            int durationIndex = cursor.getColumnIndex(DBHelper.KEY_DURATION);
            int typeIndex = cursor.getColumnIndex(DBHelper.KEY_ID_TYPE);
            do {
                Log.d("mLog", "ID = " + cursor.getInt(idIndex) +
                        ", name = " + cursor.getString(nameIndex) +
                        ", email = " + cursor.getString(energyIndex) +
                        ", day = " + cursor.getString(dayIndex) +
                        ", month = " + cursor.getString(monthlIndex) +
                        ", year = " + cursor.getString(yearIndex) +
                        ", hour = " + cursor.getString(hourIndex) +
                        ", min = " + cursor.getString(minIndex) +
                        ", duration = " + cursor.getString(durationIndex));
                //String    result    =    cursor.getString(nameIndex)    +    "                "+
cursor.getString(hourIndex)+":"+cursor.getString(minIndex) ;

                Task task = new Task(
                        cursor.getInt(idIndex),
                        cursor.getString(nameIndex),
                        Integer.parseInt(cursor.getString(energyIndex)),
                        new GregorianCalendar(
                                Integer.parseInt(cursor.getString(yearIndex)),
                                Integer.parseInt(cursor.getString(monthlIndex))-1,
                                Integer.parseInt(cursor.getString(dayIndex)),
                                Integer.parseInt(cursor.getString(hourIndex)),
                                Integer.parseInt(cursor.getString(minIndex)),
                                0),
                        Integer.parseInt(cursor.getString(durationIndex)),
                        cursor.getInt(typeIndex));
                day1.add(task);

            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();

        day1.sort(new Comparator<Task>() {
            public int compare(Task one, Task other) {
                return one.dateStart.compareTo(other.dateStart);
            }
        });


        dbHelper.close();

        return day1;
    }

    public boolean allowToInsertTaskInDay(ArrayList<Task> tasks, Task task){
        System.out.println("allowToInsertTaskInDay");

        /*for (Task taskInDay: tasks
        ) {
            Log.d("insert in day", taskInDay.name);
            if
(task.dateStart.get(Calendar.HOUR_OF_DAY)>taskInDay.dateStart.get(Calendar.HOUR_OF_DAY)&&task.dateStart
.get(Calendar.HOUR_OF_DAY)<taskInDay.dateStart.get(Calendar.HOUR_OF_DAY)+taskInDay.duration)
            {
                Log.d("insert in day", "1");
                return false;
            }

            else                                                                                    if
(task.dateStart.get(Calendar.HOUR_OF_DAY)+task.duration>taskInDay.dateStart.get(Calendar.HOUR_OF_DAY)&&
```

```
task.dateStart.get(Calendar.HOUR_OF_DAY)+task.duration<taskInDay.dateStart.get(Calendar.HOUR_OF_DAY)+ta
skInDay.duration) {
                Log.d("insert in day", "2");
                return false;
            }
            else                                                                                    if
(task.dateStart.get(Calendar.HOUR_OF_DAY)<=taskInDay.dateStart.get(Calendar.HOUR_OF_DAY)&&task.dateStar
t.get(Calendar.HOUR_OF_DAY)+task.duration>=taskInDay.dateStart.get(Calendar.HOUR_OF_DAY)+taskInDay.dura
tion)
            {
                Log.d("insert in day", "3");
                return false;
            }

        }*/
        for (Task taskInDay: tasks
        ) {
            Log.d("insert in day", taskInDay.name);
            if
(task.dateStart.compareTo(taskInDay.dateStart)==1&&task.dateStart.compareTo(taskInDay.dateFinish)==-1)
            {
                Log.d("insert in day", "1");
                return false;
            }

            else                                                                                    if
(task.dateFinish.compareTo(taskInDay.dateStart)==1&&task.dateFinish.compareTo(taskInDay.dateFinish)==-
1) {
                Log.d("insert in day", "2");
                return false;
            }
            else                                                                                    if
((task.dateStart.compareTo(taskInDay.dateStart)==0||task.dateStart.compareTo(taskInDay.dateStart)==-1)


&&(task.dateFinish.compareTo(taskInDay.dateFinish)==0||task.dateFinish.compareTo(taskInDay.dateFinish)=
=1))
            {
                Log.d("insert in day", "3");
                return false;
            }

        }
        return true;
    }

    public  void findTimeTask(String name){

        DBHelper dbHelper = new DBHelper(this);

        SQLiteDatabase database = dbHelper.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        String selection = "name =?";
        String[] selectionArgs = {name};
        String[] columns = {"hour", "minute"};
        Cursor cursor = database.query(true, DBHelper.TABLE_TASK, columns, selection, selectionArgs,
null, null, null, null);
        int count = cursor.getCount();
        ArrayList<Task> day1 = new ArrayList<Task>();
        int i=0;

        int h=0;
        int m=0;
        int c=0;
        boolean some=false;
```

```
        if (cursor.moveToFirst()) {


            do {

            System.out.println("Time "+cursor.getInt(0));
            System.out.println("Time "+cursor.getInt(1));
            int hoursTemp = cursor.getInt(0);
            int minutesTemp = cursor.getInt(1);

                SQLiteDatabase database1 = dbHelper.getWritableDatabase();

                ContentValues contentValues1 = new ContentValues();

                String selection1 = "name =? and hour =? and minute =?";
                String[]      selectionArgs1      =      {name      ,      String.valueOf(hoursTemp),
String.valueOf(minutesTemp)};
                String[] columns1 = {"hour", "minute"};
                Cursor   cursor1   =   database.query(   DBHelper.TABLE_TASK,   columns1,   selection1,
selectionArgs1, null, null, null, null);
                int count1 = cursor1.getCount();

                if (count1>c)
                {
                    c= count1;
                    h = hoursTemp;
                    m=minutesTemp;
                    some = true;
                }

                cursor1.close();

                System.out.println("count time  "+count1);


            } while (cursor.moveToNext());
        } else
            Log.d("mLog","0 rows");

        cursor.close();

        dbHelper.close();


        if (some){
            hpurProposed=h;
            minuteProposed=m;
            timeProposed=true;
        }
        else timeProposed=false;

        System.out.println("proposedHour " +hpurProposed);
        System.out.println("proposedHour " +minuteProposed);

    }

}
```

Class CustomListAdapter

```
public class CustomListAdapter extends ArrayAdapter<Task> {

    ArrayList<Task> products;
    Context context;
    int resource;

    public CustomListAdapter(Context context, int resource, ArrayList<Task> products) {
        super(context, resource, products);
        this.products = products;
        this.context = context;
        this.resource = resource;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        if (convertView == null){
            LayoutInflater layoutInflater = (LayoutInflater) getContext()
                    .getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
            convertView = layoutInflater.inflate(R.layout.custom_list_layout, null, true);

        }
        Task product = getItem(position);



        TextView txtName = (TextView) convertView.findViewById(R.id.txtName);
        txtName.setText(product.name);

        TextView txtPrice = (TextView) convertView.findViewById(R.id.txtTime);
        String                                          time                                                =
product.dateStart.get(Calendar.HOUR_OF_DAY)+":"+product.dateStart.get(Calendar.MINUTE)+"-
"+product.dateFinish.get(Calendar.HOUR_OF_DAY)+":"+product.dateFinish.get(Calendar.MINUTE);
        txtPrice.setText(time);

        TextView txtPoints = (TextView) convertView.findViewById(R.id.txtEnergy);
        txtPoints.setText(String.valueOf(product.kalor));

        return convertView;
    }
}
```