Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«РОЗРОБКА iOS-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ СПРАВАМИ З ВИКОРИСТАННЯМ АЛГОРИТМУ АДАПТИВНОГО ПЛАНУВАННЯ НА ОСНОВІ ЕМОЦІЙ КОРИСТУВАЧА»**

Виконала: студентка 4-го року навчання
Спеціальності
121 Інженерія програмного забезпечення

Пізь Мар’яна Андріївна

Керівник: Вознюк Я.І.,
Старший викладач

Рецензент: _____

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК _____
«____» _____ 2025 р.

Київ – 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра інформатики

ЗАТВЕРДЖУЮ
Завідувач кафедри інформатики,
канд. фіз.-мат. наук, доцент
_____ Гороховський С.С.
(підпис)
«_____» _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентці Пізь Мар'яні Андріївні

факультету інформатики 4-го курсу бакалаврської програми

ТЕМА: Розробка iOS-застосунку для управління справами з використанням алгоритму адаптивного планування на основі емоцій користувача

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Анотація

Вступ

1 Аналіз предметної області

2 Огляд використаних технологій та інструментів

3 Технічна реалізація iOS-застосунку

Висновки

Список використаної літератури

Дата видачі «___» _____ 2024 р. Керівник Вознюк Я.І. _____
(підпис)

Завдання отримала Пізь М.А. _____
(підпис)

ГРАФІК ПІДГОТОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	**ПЕРЕЛІК РОБІТ**	**Термін виконання**	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи.	жовтень			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	жовтень – листопад			
3.	Складання плану кваліф. роботи та узгодження з науковим керівником	листопад			
4.	Постановка експерименту, аналіз отриманих результатів наукового дослідження	листопад – березень			
5.	Проміжний контроль виконання роботи	лютий			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	січень – березень			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)				

	Розділ 2 (аналітично-дослідницька частина)				
	Розділ 3 (проєктно-рекомендаційна частина)				
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	квітень – середина травня			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності,	кінець травня			
9.	Подання на зовнішню рецензію	кінець травня			
10.	Підготовка до захисту кваліфікаційної роботи на засіданні кафедри: написання доповіді та виготовлення ілюстративного матеріалу	**до _____ травня**			
11.	Попередній захист кваліфікаційної роботи на засіданні кафедри	**до _____ травня**			
12.	Подання кваліфікаційної роботи на кафедру з усіма супроводжувальними документами	**до _____ травня**			
13.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	згідно з розкладом роботи ЕК			

Графік узгоджено «___» _____ 2025 р.

Науковий керівник: Вознюк Ярослав Іванович

Виконавиця кваліфікаційної роботи: Пізь Мар’яна Андріївна

ЗМІСТ 5

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

API – Application Programming Interface;

CSV – Comma-Separated Values;

iOS – iPhone Operating System;

MVC – Model View Controller;

MVVM – Model View ViewModel;

No-SQL – Not Only SQL;

SPM – Swift Package Manager;

SQL – Structured Query Language;

VIPER – View Interactor Presenter Entity Router;

WWDC – Apple Worldwide Developers Conference.

АНОТАЦІЯ

Кваліфікаційна робота присвячена створенню цифрового інструменту, покликаного розв’язати проблему погіршення продуктивності внаслідок емоційного виснаження. За мету роботи поставлено розробити інтуїтивно зрозумілий iOS-застосунок для планування справ, що підлаштовує графік до емоцій користувача, пропонує усвідомлені паузи й аналітику емоцій у контексті виконаних завдань та підтримує українську мову. У ході роботи розглянуто наявні програмні рішення в сферах продуктивності та ментального здоров’я, проаналізовано технології, інструменти та архітектурні шаблони для сучасної iOS-розробки, а також застосовано алгоритм адаптивного планування залежно від емоцій користувача та властивостей завдань. Реалізований застосунок може бути використаний широким колом користувачів у повсякденному житті.

Ключові слова: планування, емоції, продуктивність, мобільний застосунок, iOS, Swift, SwiftUI, SwiftData.

ВСТУП

В умовах постійної нестабільності – військових конфліктів, економічних викликів та інформаційного перевантаження – люди все частіше страждають від емоційного виснаження та, як наслідок, втрати продуктивності. Також, зважаючи на стрімкий розвиток цифрових технологій, все більшої популярності набувають цифрові інструменти для організації власного часу та відстеження настрою й здоров'я.

На ринку існує безліч застосунків, що фокусуються на відслідковуванні показників здоров'я чи емоцій (DailyBean [1], Bearable [2]) або на управлінні часом (TickTick [3], Notion [4]). Проте жодне з цих рішень не розглядає ці два аспекти як єдину систему, що сприяє усвідомленому плануванню та адаптивному керуванню завданнями. Таким чином, актуальною є ніша для продукту, що організовує планування на основі поточного емоційного стану користувача.

Об'єктом дослідження є процес планування справ за допомогою цифрових інструментів.

Предмет дослідження – адаптивне планування справ з урахуванням емоційного стану користувача за допомогою мобільного застосунку.

Виходячи з актуальних проблем емоційного виснаження та потреби в усвідомленому плануванні, метою цієї роботи є розробити інтуїтивно зрозумілий і легкий у користуванні iOS-застосунок для планування справ, що підлаштовує графік до емоцій користувача, пропонує усвідомлені паузи й аналітику емоцій у контексті виконаних завдань та підтримує українську мову. Для досягнення цієї мети поставлено такі завдання:

1. Провести аналіз наявних програмних рішень у нішах планування та емоцій.
2. Обрати технології для розробки власного iOS-застосунку.
3. Реалізувати функціонал застосунку відповідно до сформованих вимог.

4. Забезпечити підтримку англійської та української мов інтерфейсу користувача.

У роботі використано такі методи дослідження, як порівняльний аналіз застосунків-конкурентів, моделювання варіянтів використання застосунку у вигляді діаграми та програмування мобільних застосунків під iOS.

Наукова новизна полягає у створенні нового iOS-застосунку, у якому реалізовано спеціально розроблений алгоритм адаптивного планування, що враховує емоційний стан користувача.

Отримані результати можуть бути застосовані в галузях особистої продуктивності, підтримки ментального здоров'я, а також братися за основу для подальших розробок систем адаптивного планування. Розроблений iOS-застосунок може бути використаний широким колом користувачів для планування їхніх щоденних справ. Застосунок також може бути частиною програм корпоративних середовищ, які прагнуть підтримувати й аналізувати емоційний стан своїх працівників.

Робота складається з трьох розділів

Перший розділ присвячено аналізу предметної області, а саме: впливу емоційного стану на продуктивність людини та огляду вже наявних програмних рішень у галузях продуктивності й відстеження емоцій.

У другому розділі здійснено обґрунтування вибору платформи, технологій та інструментів, використаних для реалізації мобільного застосунку.

У третьому розділі описано процес розробки iOS-застосунку від формування вимог та ідейної концепції до створення архітектури проєкту та реалізації функціональних можливостей користувача.

Створено iOS-застосунок, що адаптує графік користувача до його емоційного стану, надає можливість виконувати вправи на усвідомленість та переглядати аналітику емоцій в контексті виконаних завдань, а також підтримує українську мову й враховує вимоги зручності для користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вплив емоційного стану на продуктивність людини

У сучасному ритмі життя часто здається, що чим швидше ми виконуємо певні завдання, тим краще. Працедавцям вигідно наймати людей, які ефективно виконують поставлені задачі. Прийнято вважати, що для цього варто розвивати свої навички управління часом. Існує безліч технік, які допомагають людям вирішувати проблеми з їхньою продуктивністю. Наприклад, метод Помодоро підійде людям, яким важко почати робити певне завдання. Матриця Ейзенхауера допоможе зрозуміти, які задачі дійсно мають цінність і розставити пріоритети. Планування дня блоками часу допоможе уникнути багатозадачності. Принцип Парето допоможе спрямувати увагу на важливі задачі, які дійсно принесуть результат [5]. Усі ці техніки варті використання і кожен може спробувати віднайти серед них ті, які найкраще допомагатимуть йому керувати часом.

Проте навіть найбільш ефективні техніки управління часом не сильно допоможуть, якщо людина відчуває емоційне виснаження. Адже важливо не лише робити більше за коротший проміжок часу, а й мати достатньо енергії та задовільний емоційний стан у довгостроковій перспективі. Згідно із дослідженням користі усвідомленості на роботі, працівники, які займаються вправами емоційної регуляції та підтримують свій емоційний стан у порядку, мають на 20-30% вищу продуктивність, ніж ті, що використовують лише техніки управління часом [6]. Це свідчить про те, що варто не лише думати про продуктивність у звичному її розумінні, а й вчитися помічати свої емоції та дбати про ментальний стан, забезпечуючи цим собі можливість бути в ресурсі, щоб виконувати поставлені завдання.

Ще одним підтвердженням цього є дослідження впливу емоційного інтелекту на залученість та продуктивність працівників. Доведено, що працівники з високим рівнем емоційного інтелекту краще справляються зі стресовими ситуаціями, проявляють вищу адаптивність до змін, а також

ефективніше взаємодіють з колегами [7]. Забезпечення якісної комунікації є особливо важливим за умови віддаленого формату роботи, який останніми роками набув великої популярності: кожен четвертий працівник у США працює віддалено принаймні частково [8]. Впровадження програм розвитку емоційного інтелекту в корпоративному середовищі не лише покращує стосунки між працівниками, а й знижує плинність кадрів, чим підвищує загальну продуктивність команди [7] Це ще один доказ впливу емоційної складової на довгострокову продуктивність працівника та компанії в цілому.

Отже, як показують сучасні дослідження, лише в поєднанні з усвідомленістю та емоційною регуляцією техніки управління часом можуть дати людині ту саму бажану продуктивність в особистому житті та професійній діяльності.

1.2 Аналіз наявних програмних рішень у нішах емоцій та продуктивності

Дослідження вже реалізованих програмних рішень для планування часу та/або емоційної саморефлексії, що існують на ринку, дозволяє ще глибше проаналізувати предметну область. Аналіз обраних застосунків надає можливість виділити їхні функціональні можливості, сильні та слабкі сторони. На основі цих спостережень сформовано вимоги до розробки власного програмного продукту, що має на меті поєднати переваги розглянутих рішень та вирішити їхні недоліки.

1.2.1 DailyBean

DailyBean – це застосунок-щоденник для збереження подій та емоцій користувача [1]. За допомогою різноманітних емодзі користувач обирає, як пройшов день, у якому оточенні він його провів (з друзями, сім’єю чи партнером), яка погода та який емоційний стан в нього були впродовж дня.

Надану інформацію про свої дні можна переглядати в хронологічному порядку. У застосунку є розділ з аналітикою за конкретний місяць: графік зміни настрою; справи, які найчастіше виконувалися; найщасливіші та найсумніші дні. Можна налаштувати щоденні нагадування для того, щоб не забути зафіксувати інформацію про день. У користувачів є можливість створити аккаунт, проте синхронізація DailyBean між усіма пристроями доступна лише при наявності преміум-доступу до застосунку. Додатковою функцією є можливість сформувати інформацію про свій день у форматі зображення, що зручно для поширення в соціальних мережах або друзям.

Таким чином, DailyBean є мінімалістичним застосунком, який виконує роль щоденника та засобу відстеження настрою й різних дій, проте не дозволяє керувати своїм часом та планувати справи на день.

1.2.2 Bearable

Bearable – це застосунок для відслідковування стану здоров'я, самопочуття та настрою [2]. Перед початком користування обов'язково потрібно створити обліковий запис. Особливістю застосунку є висока кастомізація: користувачеві надається великий вибір того, що саме він міг би відслідковувати.

Після певного часу користування й накопичення даних, а також за умови наявності преміум-доступу, користувач отримує спостереження, як його повсякденні дії впливають на настрій, самопочуття та сон. У безкоштовній версії можна переглянути лише графіки окремих показників за останній тиждень. Як і DailyBean, Bearable має змогу надсилати нагадування, тож можна налаштувати їх навіть кілька разів на день, щоб не забувати вносити записи.

Отже, Bearable дещо схожий на DailyBean, проте дозволяє відслідковувати значно більше речей і адаптувати цей процес під свої потреби. Як і в попередньому застосунку, у Bearable також відсутні функції, пов'язані з плануванням завдань та управлінням часом.

1.2.3 TickTick

TickTick – це багатофункціональний застосунок для управління завданнями, що дозволяє формувати списки справ, створювати та відслідковувати звички, послуговуватися календарем, використовувати матрицю Ейзенхауера та техніку Помодоро [3]. Як і у Bearable, тут присутня кастомізація: користувач може обрати, які саме з цих функцій йому потрібні, і додати лише їх на панель вкладок. TickTick підтримує всі найпопулярніші операційні системи (iOS, Android, Windows, MacOS, WatchOS) та забезпечує синхронізацію даних між ними за умови наявності облікового запису.

Основний функціонал – список завдань – у TickTick можна переглядати в кількох різних режимах відображення. За замовчуванням, доступний маленький календар на місяць із можливістю вибору конкретного дня та перегляду завдань на цей день у вигляді списку на тому ж екрані нижче. Додаткові режими відображення, такі як повний календар на місяць, тиждень, три дні та день, доступні лише за умови наявності преміум-версії.

Таким чином, TickTick пропонує необхідний функціонал для керування своїми справами та часом, але ніяк не враховує при цьому емоційний стан користувача.

1.2.4 Notion

Notion – це програмне забезпечення для організації інформації, що дає можливість створювати робочий простір для ведення своїх нотаток та проєктів [4]. Цей застосунок можна використовувати як індивідуально, так і для командної роботи. Користувачі можуть записувати свої завдання у вбудований календар, а в 2024-му році був випущений окремий, більш зручний у плані синхронізації зі сторонніми сервісами, застосунок – Notion Calendar [9].

Notion притаманна висока кастомізація. Користувачі можуть створювати шаблони сторінок під різноманітні потреби та ділитися ними на просторах

інтернету. Наприклад, існують шаблони для планування задач, відстеження звичок, настрою та безліч інших. Це дозволяє розглядати Notion як значно серйозніший інструмент, ніж звичайний текстовий редактор. Проте варто відмітити, що застосунок не володіє вбудованими інструментами, що допомагають користувачам краще планувати час та поєднувати це з емоційним станом.

Отже, Notion цілком можна використовувати для базового планування та відслідковування емоцій, якщо налаштувати відповідні шаблони. Але користувач не отримає переваг поєднання планування з емоційним станом, реалізованих програмно.

1.2.5 Порівняння програмних продуктів

Проаналізовано й порівняно ряд наявних програмних рішень, що використовуються для планування часу та відслідковування емоційного стану. Кожен із проаналізованих застосунків має своє основне призначення (відстеження емоцій, планування або організація інформації) та додаткові функціональні можливості (гнучке налаштування застосунку під потреби користувача, наявність нагадувань, інтеграція з іншими сервісами, можливість створити аккаунт для синхронізації на кількох пристроях, можливість оформити платну підписку та отримати доступ до преміум-можливостей). Порівняння розглянутих застосунків наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння наявних застосунків

	DailyBean	Bearable	TickTick	Notion
Основне призначення	Ведення щоденника для записів про свої активності та настрій	Відслідковування показників здоров'я, самопочуття та настрою	Управління задачами, використання технік покращення продуктивності	Організація інформації, планування проєктів
Планування часу	Ні	Ні	Так	Так (за допомогою шаблонів)
Відстеження емоцій	Так	Так	Ні	Так (за допомогою шаблонів)
Адаптивне планування справ	Ні	Ні	Ні	Ні
Рівень кастомізації	Низький	Високий	Середній	Високий
Можливість створити аккаунт	Так, необов'язково	Так, обов'язково	Так, необов'язково	Так, обов'язково
Інтеграція з іншими сервісами	Ні	Так (Fitbit / Apple Health)	Так (Календарі, Notion, Apple Health, Reminders та ін.)	Так (Slack, Jira, Google Calendar, Figma та ін.)
Нагадування	Так	Так	Так	Так
Ціна	Безкоштовно (преміум-функції за підпискою)	Безкоштовно (преміум-функції за підпискою)	Безкоштовно (преміум-функції за підпискою)	Безкоштовно (преміум-функції за підпискою)
Підтримка української	Ні	Так	Так	Ні

Для всіх розглянутих застосунків спільними рисами є:

- можливість створити аккаунт, щоб мати змогу синхронізувати свої записи на різних пристроях;
- наявність нагадувань;
- безкоштовний доступ до базового функціоналу та преміум-можливості, що доступні за платною підпискою.

Виявлено, що жоден із застосунків не розглядає вплив емоційного стану на продуктивність та не поєднує ці два аспекти в єдину систему, що має на меті підтримувати усвідомленість користувачів про їхні емоції та допомагає планувати час відповідним чином.

1.3 Висновки до розділу 1

У першому розділі розглянуто зв’язок продуктивності з емоційним станом людини та проаналізовано наявні застосунки в цих нішах. З’ясовано важливість усвідомленості емоцій для довгострокових успіхів у роботі та особистому житті. Аналіз уже реалізованих застосунків дав зрозуміти потребу у створенні програмного рішення, який поєднає в собі найкращі практики двох сфер – планування та ментального здоров’я – і дозволить створити функціональну можливість, яка за допомогою аналізу емоцій користувача допоможе йому виконувати заплановані завдання без емоційного виснаження.

2 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ

2.1 Вибір платформи для розробки

Для застосунку, спрямованого на планування справ з урахуванням емоційного стану, важливим є надати користувачеві можливість протягом дня взаємодіяти зі своїми завданнями та фіксувати емоційний стан. Необхідно надати змогу виконувати ці дії оперативно, не змушуючи чекати можливості сісти за персональний комп'ютер. Отже, оптимальним рішенням є розробка програмного забезпечення, придатного до використання за допомогою мобільного пристрою.

Згідно з дослідженнями, у США користувачі 88% свого часу, проведеного за мобільним телефоном, витрачають на взаємодію із мобільними застосунками та лише 12% – на вебсайти [10]. Це доводить, що мобільні застосунки є більш звичним рішенням для виконання більшості завдань, ніж вебсайти, адаптовані під мобільні пристрої. Враховуючи це, прийнято рішення розробляти саме мобільний застосунок.

Як платформа для розробки обрана iOS, зважаючи на ряд причин:

- застосунок працює із доволі чутливими персональними даними: емоціями та справами. Корпорація Apple славиться суворою політикою конфіденційності, тож користувачі пристроїв Apple мають більшу схильність довіряти застосункам, що працюють із такими даними [11];
- для ефективного використання застосунку важливо багато разів на день взаємодіяти з ним, тож варто фокусуватися на групі людей, що більш звиклі до проведення часу за телефоном. Відомо, що користувачі iPhone проводять в 1.3 разів більше часу, користуючись своїми мобільними телефонами, ніж власники Android-пристроїв [12].;
- користувачі iPhone витрачають більше коштів на підписки та преміум-послуги в застосунках [13]. Враховуючи монетизацію як перспективу подальшого розвитку застосунку, це є важливим фактором.

2.2 Swift

Для розробки сучасних iOS-застосунків компанія Apple радить використовувати мову програмування Swift [14]. Вона була представлена в 2014 років як логічна наступниця Objective-C, що забезпечує вищу продуктивність, мінімізує ймовірність виникнення помилок завдяки покращеній роботі з пам’яттю та обробці опціональних значень, а також має більш чистий та сучасний синтаксис [15].

Найновіша мажорна версія Swift 6, використана для розробки проєкту, анонсована у вересні 2024 на Apple WWDC. У ній покращено роботу з багатопоточністю, взаємодію з C++, випущено новий фреймворк для тестування Swift Testing [16]. Усе це свідчить про те, що мова активно підтримується та розвивається.

Також Swift надає можливість використовувати вбудований менеджер пакетів SPM, чим спрощує інтеграцію залежностей у проєкт [17].

Отже, володіючи такими перевагами, як висока продуктивність, безпечна робота з пам’яттю та багатопоточністю, сучасний синтаксис, активна підтримка та вбудований пакетний менеджер, Swift є найбільш підходящим вибором як мова програмування для розробки iOS-застосунку.

2.3 SwiftUI

Для створення користувацького інтерфейсу в iOS Apple дає на вибір два основних фреймворки: UIKit та SwiftUI. Кожен із них має свої переваги та недоліки, і вибір більш відповідного фреймворку для конкретного проєкту може суттєво вплинути як процес та тривалість розробки, так і на якість користувацького досвіду [18].

UIKit є імперативним фреймворком, який використовується з часів появи iPhone. Цим пояснюються його сильні сторони: стабільність, вичерпна документація, велика кількість готових рішень і прикладів. UIKit також дозволяє

точно керувати поведінкою елементів – це корисно, якщо розробнику потрібно сильно адаптувати інтерфейс під власні потреби [18].

Натомість SwiftUI є декларативним фреймворком, що був представлений компанією Apple на WWDC у 2019 році. Завдяки декларативному синтаксису, у SwiftUI достатньо просто вказати, що повинен робити користувацький інтерфейс, – і фреймворк сам визначить, як це реалізувати, зважаючи на поточний стан даних. У свою чергу, UIKit вимагає детального опису, як мають змінюватися елементи залежно від зміни стану програми. Сам код на SwiftUI також є більш лаконічним та легшим для сприйняття. Окрім того, SwiftUI показує вищу продуктивність рендерингу інтерфейсу, порівняно з UIKit, особливо якщо інтерфейс має часто змінюватися залежно від стану програми [19]. Важливо відзначити ще й те, що SwiftUI підтримує всі платформи Apple - iOS, macOS, watchOS, tvOS [18].

Хоч SwiftUI поки що не має повної функціональності, яку надавав UIKit, Apple повсякчас розширює його можливості та покращує стабільність. До того ж, якщо виникає потреба використати певні елементи UIKit у SwiftUI, це можна легко зробити за допомогою протоколів UIViewRepresentable, який дозволяє загорнути будь-який UIView у View, та UIViewControllerRepresentable, який дозволяє інтегрувати в інтерфейс на SwiftUI цілий UIViewController [20].

Зважаючи на вищезгадані особливості, такі як лаконічний синтаксис, високу продуктивність рендерингу, підтримку різних платформ SwiftUI та можливість інтеграції UIKit, SwiftUI можна сміливо назвати майбутнім iOS-розробки. Саме цей фреймворк є логічним і популярним вибором для реалізації нових проєктів.

2.4 SwiftData

Донедавна для збереження даних в iOS було два основних рішення: CoreData та Realm. Перше – нативний фреймворк від Apple. Визначення типів даних та зв’язків між ними в CoreData відбувається в спеціальному Data Model editor файлі [21]. Друге рішення – це кросплатформна NoSQL-база даних від

стороннього розробника, що дає простий API та вищу продуктивність, але змушує додавати до проєкту зовнішню залежність [22].

Фреймворк SwiftData був представлений на WWDC у 2023 році та поєднав переваги CoreData і Realm. Це нативний фреймворк, проте замість окремого редактора моделей, як у CoreData, у SwiftData використовується макрос @Model одразу в коді. Це значно спрощує процес взаємодії з даними та зменшує кількість шаблонного коду, порівняно з CoreData. Варто відзначити, що SwiftData легко інтегрується до SwiftUI, а також із цим фреймворком простіше працювати в асинхронному середовищі [23].

Зважаючи на те, що застосунок розробляється лише під iOS і для користувацького інтерфейсу вже обраний фреймворк SwiftUI, найбільш доречним вибором для збереження даних є SwiftData.

2.5 SwiftLint

SwiftLint – це інструмент для статичного аналізу коду, що містить понад 200 правил, і цей список постійно зростає. Конфігурація відбувається у файлі .swiftlint.yml, що розміщується в кореневій папці проєкту: дозволяється вмикати або вимикати правила, задавати порогові значення (наприклад, максимальну кількість рядків у файлі чи довжину назви змінної), ігнорувати окремі файли або папки, а також навіть створювати власні правила за допомогою регулярних виразів. SwiftLint можна встановити як інструмент командного рядка, як залежність до проєкту, як Run Script Build Phase, що виконується при збірці проєкту, а також як Git pre-commit hook [25].

2.6 SwiftFormat

SwiftFormat – інструмент для автоматичного форматування коду, що спирається на стандартні правила відступів, розміщення дужок, лапок, розділення параметрів/аргументів функцій та багато іншого. Загалом

SwiftFormat містить понад 50 правил, і їхній список постійно розширюється відповідно до нових версій мови програмування Swift. Налаштувати правила можна через файл конфігурації .swiftformat: там можна вмикати або вимикати параметри, змінювати параметри правил, а також ігнорувати певні файли або каталоги. SwiftFormat може використовуватися локально як інструмент командного рядка, як розширення редактора коду, як Run Script Build Phase у Xcode-проєкті, а також як Git pre-commit hook [24].

2.7 Висновки до розділу 2

У цьому розділі проведено огляд основних технологій та інструментів, використаних для розробки мобільного застосунку, що має на меті допомагати користувачеві планувати справи, враховуючи його емоційний стан.

Вибір платформи iOS зумовлений високим рівнем довіри користувачів до продуктів компанії Apple, великим часом взаємодії із застосунками та перспективами монетизації. Мовою програмування обрано Swift через її активну підтримку, високу продуктивність, безпечну роботу з пам’яттю та сучасний синтаксис. Для побудови користувацького інтерфейсу вирішено використовувати SwiftUI через його сучасний декларативний підхід, лаконічний код і швидкість рендерингу елементів. Для збереження даних обрано SwiftData – нативний фреймворк, що мінімізує шаблонний код і зручно інтегрується зі SwiftUI. Для забезпечення єдиного стилю у проєкті вирішено використовувати SwiftFormat, а для виявлення потенційних помилок – SwiftLint.

Отже, обраний набір технологій та інструментів гарантує сучасне, продуктивне та якісне середовище для розробки iOS-застосунку.

3 ТЕХНІЧНА РЕАЛІЗАЦІЯ iOS-ЗАСТОСУНКУ

3.1 Формування вимог

Спираючись на аналіз наявних рішень, наведений у підрозділі 1.2, можна сформувати вимоги до застосунку, що планується розробити. Варто відзначити, що жоден із застосунків повною мірою не поєднує планування із врахуванням емоційного стану, що й стало причиною для створення нового програмного рішення для цього. Також варто врахувати переваги та уникнути недоліків розглянутих рішень.

Функціональні вимоги описують конкретні можливості, поведінку та функції, які має виконувати застосунок для користувача [26]. Їх зручно представляти у вигляді діаграми варіянтів використання (рис. 3.1).

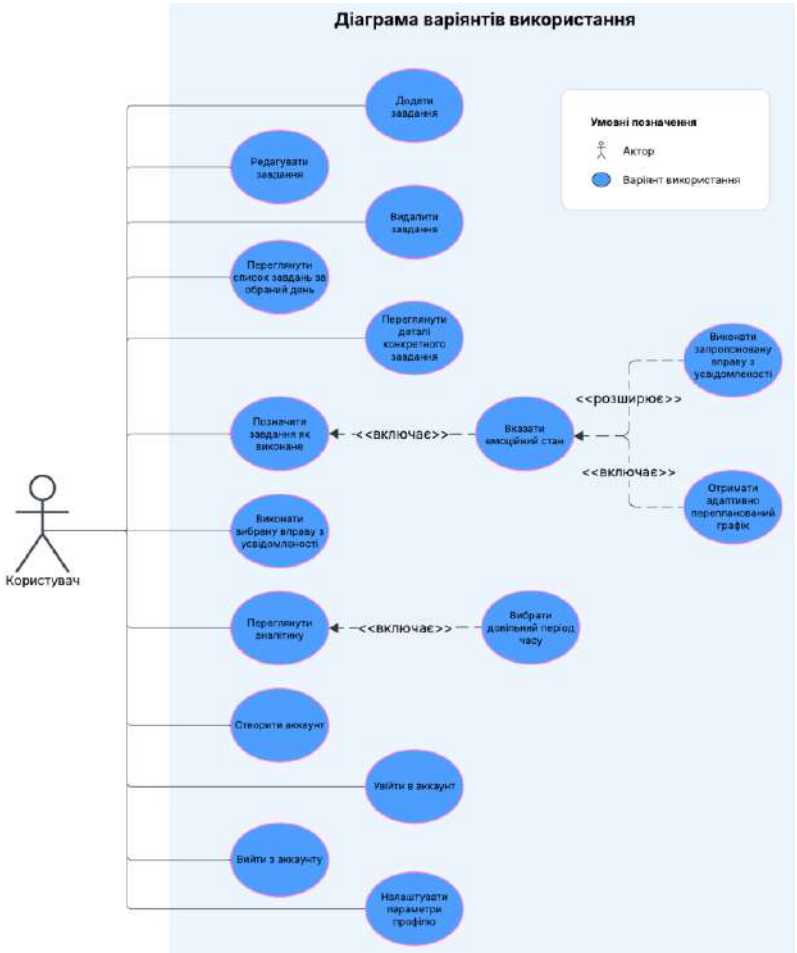

Рисунок 3.1 – Діаграма варіянтів використання застосунку

До функціональних вимог належать:

- додавання завдань;
- редагування завдань;
- видалення завдань;
- перегляд коротких відомостей про всі завдання (назва, визначений час та стан готовності), які розташовані на календарі;
- перегляд повної інформації про конкретне завдання: назва, опис, тип (гнучке чи фіксоване), категорія, складність, час початку та завершення (для фіксованих завдань) або орієнтовна тривалість (для гнучких);
- можливість позначати завдання як виконане;
- можливість позначати свій емоційний стан після виконання завдання;
- автоматичне адаптивне перепланування гнучких завдань з урахуванням обраної емоції після виконання завдання;
- виконання вправ для усвідомленості за бажанням користувача або якщо він відчуває негативні емоції після виконання завдання;
- перегляд аналітики про емоційний стан за категоріями завдань за довільний період часу;
- реєстрація;
- вхід в аккаунт;
- можливість налаштовувати параметри профілю.

Нефункціональні вимоги задають критерії роботи застосунку та визначають, як система має виконувати поставлені перед нею завдання [26]. До нефункціональних вимог належать:

- зрозумілий та інтуїтивний інтерфейс користувача;
- адаптивність до різних розмірів екранів iPhone;
- підтримка iPhone, що працюють на версіях iOS 17.0 і вище;
- підтримка локалізації (англійська та українська мови).

Отже, на основі порівняльного аналізу наявних програмних рішень та визначеної мети розробки сформовано чіткі функціональні та нефункціональні вимоги до застосунку.

3.2 Ідейна концепція застосунку

Розроблений застосунок має спростити процес планування та підвищити продуктивність користувача, одночасно враховуючи як завдання, які він хоче виконати, так і його емоційний стан, яким нехтують у традиційних інструментах планування. Тобто поставлена задача полягає в тому, щоб створити персоналізований і адаптивний планувальник часу.

Назвою застосунку обрано Moodpace (з англ. mood – настрій, pace – темп), що символізує його основну ідею: організувати день користувача в такому темпі, що відповідає його настрою, емоціям та відчуттям.

Для кольорової палітри інтерфейсу були обрані спокійні, глибокі кольори, що надають користувачеві відчуття гармонії та довіри [27]. Для фону обраний світлий бежевий колір – більш приємний оку й спокійний, ніж білий. Для більшості текстів та фонів деяких елементів використовуються два відтінки оливкового. Приглушений синій використовується як акцентний колір (наприклад, для вибраної вкладки навігаційної панелі чи позначки сьогоднішнього дня) (рис. 3.2).

Рис. 3.2 – Кольорова палітра застосунку Moodpace

У трьох із чотирьох цих кольорів виконаний і логотип застосунку. На ньому можна побачити календар, що уособлює планування, із невеликим блиском, який символізує усвідомленість (рис. 3.3).

Рис. 3.3 – Логотип застосунку Moodpace

Завдання, які користувачі виконують щодня, можна поділити на два основних типи: фіксовані – ті, які мають визначений час початку та завершення (наприклад, пара в університеті чи зустріч із менеджером на роботі) та гнучкі – ті, які треба виконати протягом дня, але не важливо, коли саме (наприклад, передивитися запис лекції чи полити кімнатні рослини). Звичайно, застосунок може міняти порядок лише гнучких завдання, залишаючи фіксовані без змін. Проте всім завданням має бути визначений певний час виконання, і всі вони мають бути відображені на інтерактивному календарі. Така реалізація відповідає підходу планування часу блоками, що допомагає зосередитися на одній задачі та утримувати на ній фокус до завершення виконання [5]. Але не потрібно розташовувати всі завдання самостійно – із цим допоможе застосунок.

Окрім типу (фіксоване чи гнучке), завдання ще мають такі властивості, як складність (дуже легка, легка, середня, складна та дуже складна) і категорія (робота, навчання, особисте, дім та інше).

Коли завдання виконане, користувач може позначити його таким, і отримати запит на вибір емоції, яку викликало в нього це завдання. Вибрати емоцію можна за допомогою емодзі: дуже сумний, сумний, нейтральний, щасливий або дуже щасливий. Навіть ця дія сама собою вже дозволяє користувачеві практикувати усвідомленість – виділити кілька секунд часу, щоб обдумати свій стан після певного завдання. Після позначення емоції в дію вступає алгоритм адаптації подальших дій користувача залежно від його емоційного стану та наступного завдання (рис. 3.4).

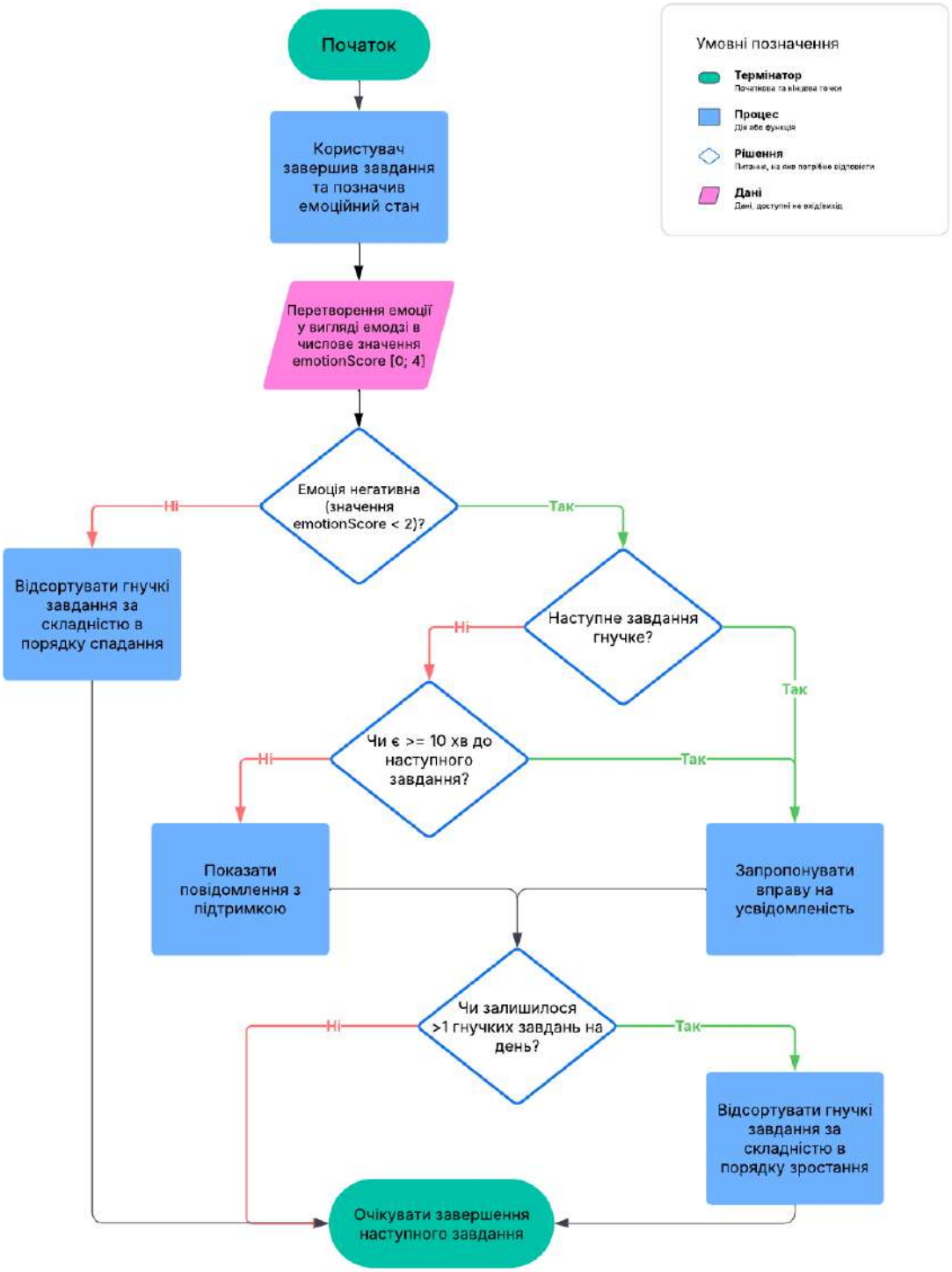

Рисунок 3.4 – Алгоритм адаптивного планування завдань залежно від емоційного стану користувача

Якщо емоція не є негативною (користувач дуже щасливий, щасливий або почувається нейтрально), то є сенс запропонувати виконати складніші завдання, тобто відсортувати гнучкі завдання за складністю в спадному порядку. Якщо ж емоція негативна (користувач сумний або дуже сумний), то сортуємо гнучкі

завдання за складністю в порядку зростання, а також аналізуємо наступне завдання. Якщо воно гнучке або фіксоване, але до його початку залишається більше 10 хв, то користувачеві пропонується виконати одну із вправ для підвищення усвідомленості або відновлення сил. Якщо ж наступне завдання фіксоване й до нього обмаль часу, показуємо підбадьорливе повідомлення.

Зібрані дані про емоції користувача можна побачити в розділі «Аналітика». За допомогою стовпчикового графіку за довільний період часу можна побачити, яка категорія завдань викликала які емоції. Також можна побачити графік зміни емоцій за днями. Це корисно для дослідження власних почуттів і покращення планування в майбутньому.

Таким чином, запропонований підхід до планування часу дає можливість не лише організовувати завдання по блоках, а й охоплює емоційну складову. Завдяки адаптивному плануванню завдань і коротким практикам усвідомленості застосунок розглядає користувача не просто виконавця справ, а розглядає його як особу зі змінним емоційним станом і підлаштовує справи, зважаючи на це.

3.3 Архітектура застосунку

Архітектура iOS-застосунку побудована на архітектурному шаблоні MVVM. Він складається з трьох компонентів:

1) model – дані та логіка, пов’язана з ними, проте жодних маніпуляцій для того, щоб представляти дані на інтерфейсі користувача;
2) view – користувацький інтерфейс; показує дані та реагує на дії користувача;
3) viewModel – посередник між model і view – володіє моделлю, перетворює її дані, щоб відображати їх користувачеві; відповідає за всю бізнес-логіку застосунку.

Великою перевагою MVVM над іншим популярним шаблоном MVC є чітке розмежування зон відповідальності, спрощення масштабування застосунку, і полегшення тестування коду. Це досягається завдяки ViewModel, що дозволяє

зменшити кількість коду у View-шарі та спростити його, а також зробити модель не залежною від інтерфейсу користувача [28]. Ще більш розділеним на окремі частини зі своїми зонами відповідальності є шаблон VIPER. Проте він значно програє простішим шаблонам (MVC, MVVM) у складності реалізації та підтримці [29]. Тож для застосунку середніх розмірів Moodpace обрано саме архітектурний шаблон MVVM.

3.4 Структура застосунку

Для забезпечення зручної навігації в коді компоненти застосунку розташовані у відповідних директоріях, згрупованих за функціональністю та призначенням (рис. 3.5).

Рисунок 3.5 – Файлова структура застосунку Moodpace

Точка входу застосунку – це структура MoodpaceApp, розміщена в директорії App. Вона ініціалізує контейнер моделей SwiftData, а також викликає початковий екран ContentView. Там уже створюється екземпляр TaskManager, який відповідає за роботу із збереженням, редагуванням та видаленням завдань. А також викликається TabBar, що забезпечує зручну навігацію між чотирма основними розділами застосунку: планування, аналітика, вправи для

усвідомленості та профіль користувача.

У директорії Model знаходяться моделі даних (TaskToDo, Day, Emotion тощо), які використовуються в проєкті. Для кращої організації деякі моделі розміщені в піддиректоріях: Manager містить AppSettingsManager і TaskManager; Service – AuthService, KeychainService, UserService, NotificationService і TaskService; Network – SignUpRequest, SignInResponse, TaskUpdateRequest тощо.

Усі View та ViewModel у проєкті згруповані за призначенням використання, щоб спростити навігацію. Таким чином, директорія UserStory містить піддиректорії: Analytics, Mindfulness, Planning, Profile, SignIn, SignUp, TabBar. Кожна з них, у свою чергу, зберігає відповідне головне View, за потреби додаткові View, а також відповідну ViewModel. Наприклад, у UserStory/Profile можна знайти ProfileView та ProfileViewModel, а в UserStory/SignUp – SignUpView та SignUpViewModel.

Окрема директорія View виділена на перевикористовувані компоненти інтерфейсу. Наприклад, у піддиректорії Common можна знайти поле для введення паролю PasswordFieldView, а у ViewModifier – модифікатор зовнішнього вигляду CustomBackButtonModifier для налаштування назви та кольору кнопки навігації Back, що наявна за замовчуванням.

Директорія Extensions містить розширення (додаткові методи чи вираховувані змінні) до вже наявних у Swift структур для зручності використання.

У Assets зберігаються візуальні компоненти застосунку: логотип, відображувані зображення чи іконки, а також набір кольорів. Це дає змогу централізовано керувати кольоровою гамою Moodpace.

Файли локалізації розміщено в каталозі Localizable. Наразі наявна підтримка англійської та української мов. За допомогою цього в застосунку всі тексти будуть такою ж мовою, яка вибрана в користувача головною на пристрої.

Така організація файлової структури застосунку полегшує навігацію та спрощує додавання нових функціональних можливостей при масштабуванні застосунку.

3.5 Автоматизована перевірка стилю та форматування коду

Для забезпечення єдиного стилю коду й запобігання потенційним помилкам вирішено використовувати такі інструменти, як SwiftFormat і SwiftLint. Маючи різні призначення, вони вдало доповнюють один одного.

3.5.1 Інтеграція SwiftLint як Run Script Phase в Xcode

SwiftLint інтегровано до проєкту як Run Script Phase в Xcode. Рішення аргументоване тим, щоб перевірка коду відбувалася при кожній збірці проєкту, а також усі знайдені помилки та попередженні візуально відображалися в середовищі розробки (рис. 3.6).

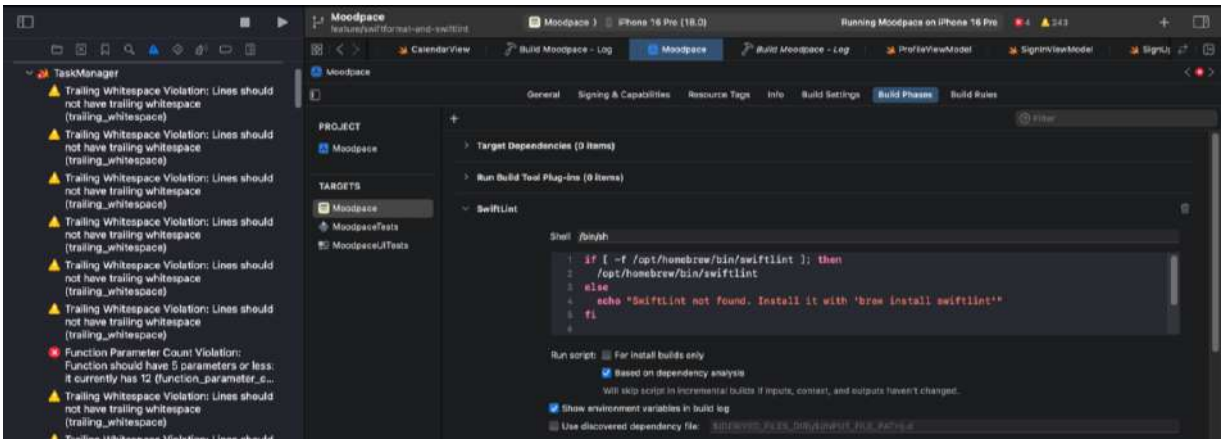

Рисунок 3.6 – Інтеграція SwiftFormat у Xcode як Run Script Phase

Для того, щоб не звертати уваги на код, розміщений у директоріях, що містять сторонні залежності, до проєкту додано .swiftlint.yml файл, у якому за допомогою параметру excluded прописано директорії, код яких не потрібно перевіряти (Pods, Packages тощо).

3.5.2 Інтеграція SwiftFormat як Git pre-commit hook

SwiftFormat інтегровано до процесу розробки як Git pre-commit hook. Тобто

форматування відбувається перед кожним комітом, що дозволяє додавати до репозиторію лише якісно оформлений код. На відміну від Run Script Phase, Git pre-commit hook не сповільнює процес збірки, тому було прийняте рішення використати його.

Для реалізації цього підходу за шляхом .git/hooks/pre-commit створено Bash-скрипт, який за допомогою утиліти git-format-staged забезпечує форматування лише тих файлів, які вже були попередньо додані до коміту (staged). Варто відзначити, що в скрипті за допомогою параметру --swiftversion явно зазначена найновіша версія мови програмування Swift – 6 (рис. 3.7).

```
#!/bin/bash
git-format-staged --formatter "swiftformat --swiftversion 6 stdin --stdinpath '{}'" "*.swift"
```

Рисунок 3.7 – Git pre-commit hook для SwiftFormat

При здійснення коміту в терміналі можна бачити, коли починається й закінчується процес форматування, а також які саме файли були відформатовані. Видно, що відформатованим був лише CalendarView.swift файл, тому що лише він був доданим до коміту (staged) (рис. 3.8).

```
(base) mapthree@Marianas-Laptop Moodpace % git status
On branch feature/swiftformat-and-swiftlint
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Moodpace/UserStory/Planning/Calendar/CalendarView.swift

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Moodpace/UserStory/Planning/Calendar/TaskCard.swift

(base) mapthree@Marianas-Laptop Moodpace % git commit -m "Make animations smoother"
Running SwiftFormat...
SwiftFormat completed successfully.
Reformatted Moodpace/UserStory/Planning/Calendar/CalendarView.swift with swiftformat --swiftversion 6 stdin --stdinpath '{}'
[feature/swiftformat-and-swiftlint 2441659] Make animations smoother
```

Рисунок 3.8 – Приклад роботи Git pre-commit hook із SwiftFormat

При поєднані двох вищезгаданих інструментів було помічено таку несумісність між ними: SwiftFormat за замовчуванням додає кому після останнього елемента в списку, а SwiftLint вважає це порушенням. Щоб уникнути цього, у .swiftformat файлі було вимкнуто це правило: --disable trailingCommas.

3.6 Функціональні можливості

Інтерфейс iOS-застосунку Moodpace організований за допомогою нижньої панелі вкладок. Як видно з рисунку 3.9, нижня панель містить чотири вкладки: планування, аналітика, усвідомленість і профіль.

Рисунок 3.9 – Панель вкладок застосунку Moodpace

У подальших пунктах розглянуто функціональні можливості відповідно до такої організації, а також окремим пунктом розглянуто можливість отримувати сповіщення.

3.6.1 Планування завдань

Відкривши вкладку «Планування», користувач одразу бачить календар. За замовчуванням вибраним днем є сьогоднішній – він позначається фоном темнішого зеленого кольору, жирним шрифтом та поміткою «Сьогодні».

У користувача є можливість переглянути минулі дні, прогорнувши горизонтально вліво, та майбутні – прогорнувши вправо. Для показу майбутніх днів використовується їхня нескінченна генерація відповідно до того, як користувач прокручує дні.

За допомогою вертикального гортання можна дивитися на години вибраного наразі дня. За замовчуванням екран прокручується до тієї години, яка вказана як початок активності користувача. Якщо цей час уже минув, фокус встановлюється на поточну годину (рис. 3.10).

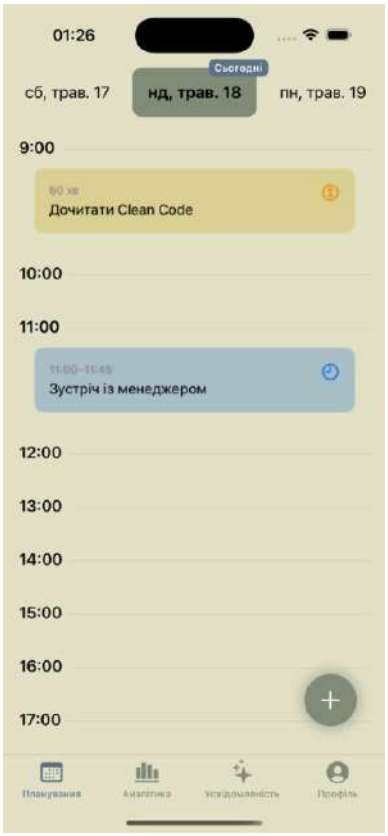

Рисунок 3.10 – Вкладка «Планування» із вибраним сьогоднішнім днем та прокруткою до години початку активності користувача

При виборі іншого дня, новий вибраний день набуває цього ж темнішого фону та жирного шрифту, а на сьогоднішньому дні залишається рамка та помітка «Сьогодні» (рис. 3.11).

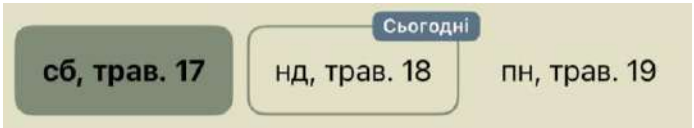

Рисунок 3.11 – Відображення вибраного та сьогоднішнього днів

Для того, щоб додати завдання, треба скористатися кнопкою «+», розміщеною справа внизу екрану (рис. 3.10). Опісля відкриється вікно для створення завдання. Характеристики завдання, які треба зазначати, залежать від його типу. Спільними для обох типів є назва, опис (необов’язково), категорія (за замовчуванням – «Інше»), складність (за замовчуванням – «Середня»).

Для гнучких завдань, які не мають часових рамок, потрібно вказати орієнтовну тривалість у хвилинах (за замовчуванням – 30 хв) (рис. 3.12).

Рисунок 3.12 – Додавання нового гнучкого завдання

Якщо ж завдання фіксоване, то замість орієнтовної тривалості користувач має вказати час початку та кінця. За замовчуванням цими значеннями є поточний час та час за годину від поточного відповідно (рис. 3.13).

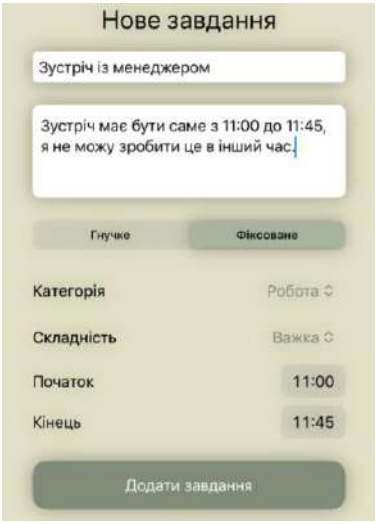

Рисунок 3.13 – Додавання нового фіксованого завдання

Кнопка «Додати завдання» залишається неактивною, доки користувач не введе назву завдання. Опис завдання може бути порожнім. Інші поля можуть залишатися такими ж, як і за замовчуванням.

Додавання завдання TaskViewModel делегує класу TaskManager, який створює або дістає відповідний день з локальної бази даних, додає до нього нове завдання та зберігає записи у локальну базу даних SwiftData.

Натиснувши на завдання в календарі, можна відкрити його деталі з можливістю редагування та видалення (рис. 3.14).

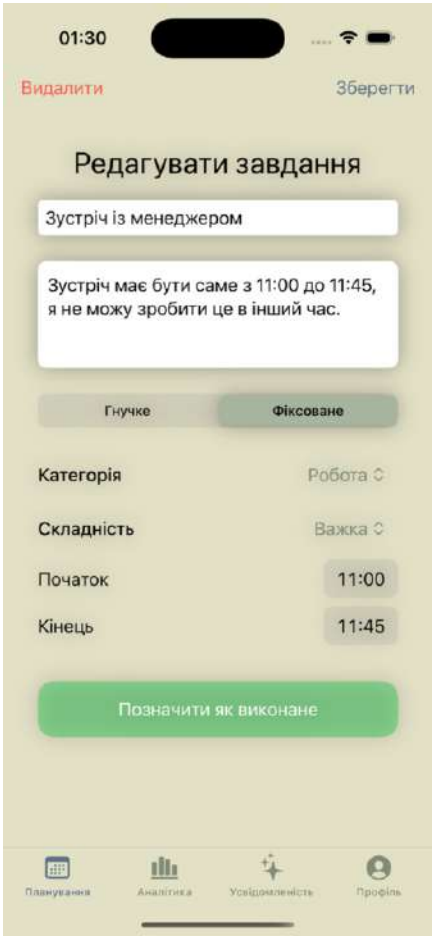

Рисунок 3.14 – Деталі завдання в режимі редагування

Таким чином, можна змінювати всі властивості й опісля натиснути кнопку «Зберегти» на верхній панелі справа, щоб активувати зміни. Також можна видалити завдання, натиснувши кнопку «Видалити» на верхній панелі зліва. З метою запобігання випадковій деструктивній дії видалення важливих завдань, ця кнопка має червоний колір, а також перед видаленням від користувача вимагається повторно підтвердити дію або ж скасувати її.

Редагування та видалення також делеговані класу TaskManager. Метод

updateTask змінює всі відповідні властивості завдання на властивості переданого TaskUpdateData, а метод deleteTask видаляє об’єкт. Після цього обидва методи зберігають контекст і оновлюють список завдань.

На цьому ж екрані деталей у режимі редагування (див. рис. 3.14) користувач може позначити завдання як виконане. Після цього він має позначити свій емоційний стан за допомогою емодзі (рис. 3.15).

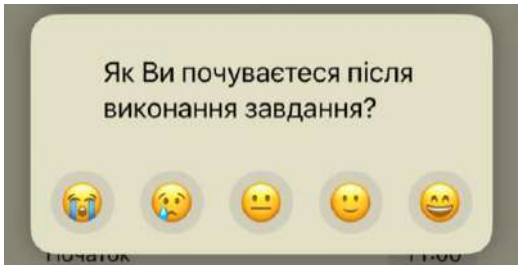

Рисунок 3.15 – Вікно вибору емоційного стану

Позначивши емоцію, користувач повертається на екран із календарем, де це завдання вже відображається як виконане. Залежно від обраної емоції подальші завдання можуть змінити свій порядок, а також, якщо емоція негативна і часу до наступного завдання достатньо, користувач отримує пропозицію виконати вправу на усвідомленість (рис. 3.16).

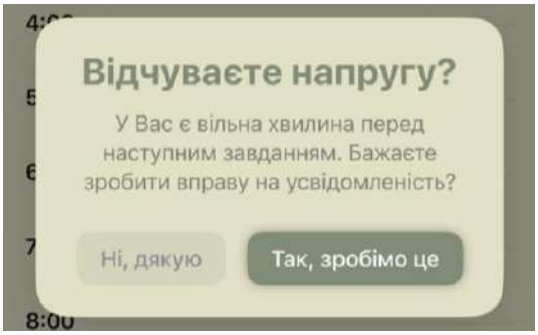

Рисунок 3.16 – Вікно з пропозицією виконати вправу на усвідомленість

Якщо користувач обирає «Так, зробімо це», то йому пропонується випадково вибрана вправа з розділу вправ на усвідомленість (докладніше про ці вправи йдеться в пункті 3.5.3).

3.6.2 Аналітика

Натиснувши на вкладку «Аналітика», користувач може переглянути статистику виконаних і невиконаних завдань, а також графік емоцій (рис. 3.17).

Рисунок 3.17 – Екран з аналітикою

У верхній частині екрану відображається, скільки завдань користувач завершив, а скільки ще потребують завершення.

Нижче можна побачити стовпчикову діаграму, що показує середній емоційний стан користувача після виконання завдань різних категорій. Кількість завдань кожної категорії зазначена над відповідним стовпчиком. Під графіком – висновок до нього, а саме: які категорії завдань викликали в користувача позитивні емоції, а які спричиняли поганий емоційний стан.

Також наявна й лінійна діаграма, яка демонструє зміну емоційного стану користувача за днями. Під цим графіком розміщений текстовий висновок із найкращими та найгіршими днями для користувача.

За замовчуванням аналітика пропонується за останній тиждень, проте за допомогою вбудованого календаря можна обрати довільний період часу.

3.6.3 Усвідомленість

Вкладка «Усвідомленість» покликана надати користувачеві доступ до інтерактивних вправ, що сприяють емоційній стабільності та усвідомленості. Інтерфейс екрану побудований у вигляді 6 карток різних кольорів з назвою та коротким описом конкретної практики (рис. 3.18).

Рисунок 3.18 – Екран з практиками усвідомленості

Усі практики реалізовані так, щоб не займати більше 10 хв у користувача й не потребувати серйозних ментальних зусиль. Натомість вони мають сприяти розумінню свого стану та допомагати відпускати негативні емоції. Загалом є 5 практик:

1) дихання 4-7-8 – це дихальна вправа, що програмно реалізована з анімацією. Користувач натискає на «Почати» і тоді спостерігає за

збільшенням і зменшенням круга. Згідно із вправою, 4 секунди має відбуватися вдих, тож круг збільшує свої розміри. Тоді 7 секунд затримки дихання круг залишається статичним. Нарешті, 8 секунд видиху він зменшується до своїх початкових розмірів. Загалом користувачеві пропонується виконати 4 таких цикли. Номер поточного циклу завжди є видимим для користувача (рис. 3.19);

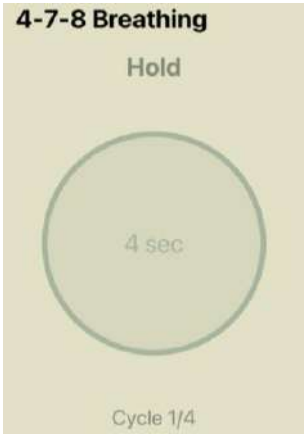

Рисунок 3.19 – Практика «Дихання 4-7-8»

2) практика вдячності – користувачеві пропонується написати 3 речі, за, які він вдячний. Після того, як усі 3 текстові поля заповнені, кнопка «Відправити» стає активною й користувач бачить toast-сповіщення про те, що його вдячність була відправлена у всесвіт (рис. 3.20);

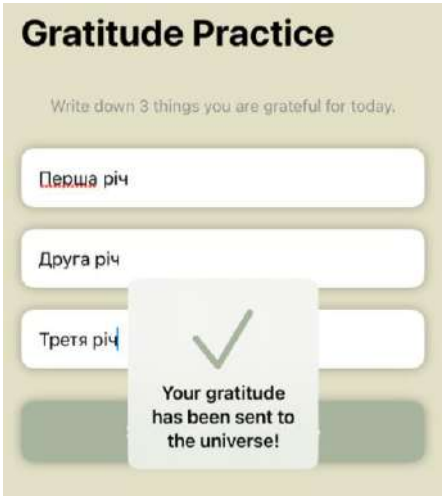

Рисунок 3.20 – Практика вдячності

3) емоційне малювання – вправа, що дозволяє висловити свої емоції через процес малювання. Користувачеві доступне поле для малювання, створене за допомогою PKCanvasView з PencilKit, а також 7 кольорів, між якими можна перемикатися. Серед доступних опцій – очистити поле та зберегти намальоване зображення до галереї пристрою, що реалізоване за допомогою UIImageWriteToSavedPhotosAlbum за умови надання користувачем дозволу на використання галереї застосунком (рис. 3.21);

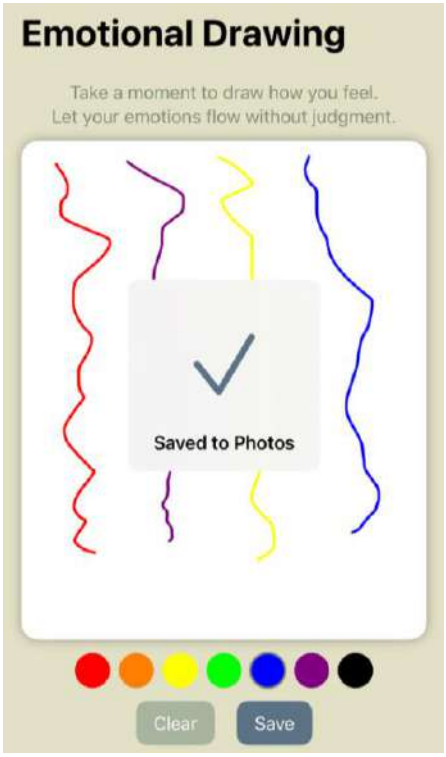

Рисунок 3.21 – Вправа «Емоційне малювання»

4) емоційний щоденник – користувачеві пропонується написати, яку емоцію він відчуває у конкретний момент часу і чому. Для кращого розуміння суті вправи наявні підказки у виглядів заповнювачів текстових полів. Як і в практиці вдячности, кнопка «Відпустити» стає активною лише після заповнення обох полів (рис. 3.22);

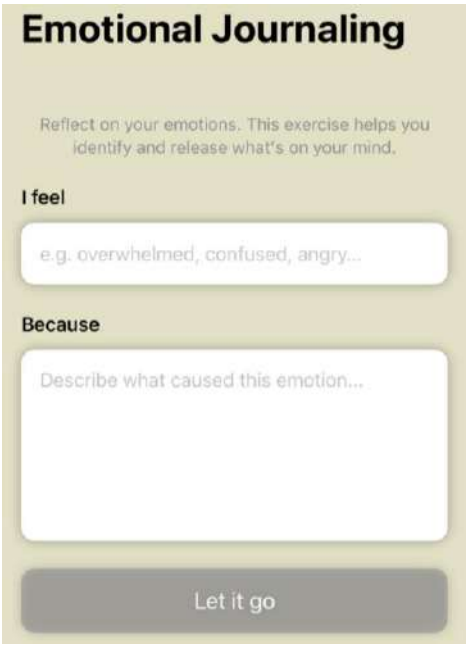

Рисунок 3.22 – Вправа «Емоційний щоденник»

5) заземлення 5-4-3-2-1 – це вправа, спрямована на те, щоб зосередитися на поточному моменті через залучення п’яти органів чуття людини. Користувачеві пропонується перемикати інтерактивні картки з інструкціями й виконувати їх (рис. 3.23).

Рисунок 3.23 – Вправа «Заземлення 5-4-3-2-1»

Вибір вправи може відбуватися як самостійно, перейшовши на вкладку «Усвідомленість» за допомогою навігаційної панелі, так і автоматично – у випадку негативної емоції після виконання певного завдання.

3.6.4 Профіль користувача

Профіль користувача дає можливість налаштовувати час початку та кінця його активного дня, що напряму впливає на розташування гнучких завдань у розкладі. Також можна налаштувати, чи бажає користувач отримувати сповіщення про початок фіксованих завдань (рис. 3.24).

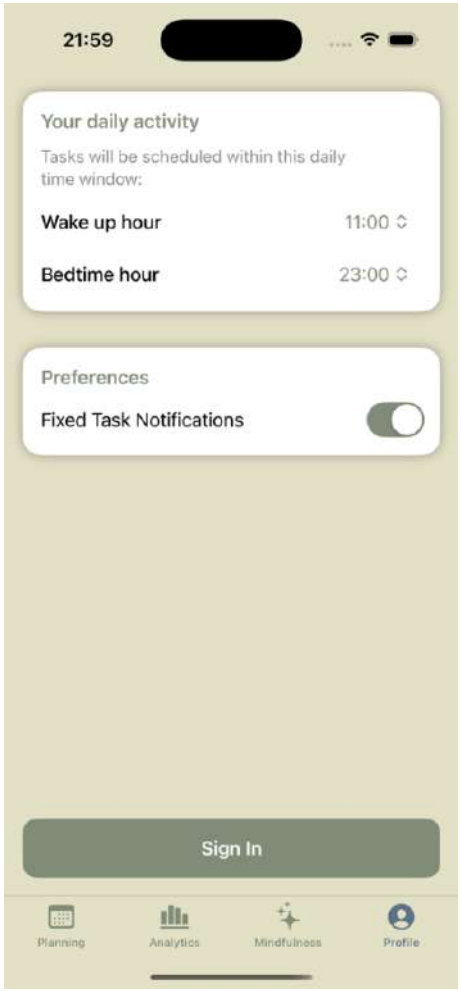

Рисунок 3.24 – Вкладка профілю користувача (неавторизований)

У нижній частині екрану розміщена кнопка «Увійти», яка дозволяє увійти до облікового запису. Авторизація не є обов’язковою, проте вона дозволяє зберігати дані на сервері та отримати свої завдання у форматі CSV на електронну пошту. Якщо ж не авторизуватися, то дані будуть зберігатися лише локально за допомогою SwiftData.

Авторизація проста – електронна адреса та пароль. Також екран авторизації дозволяє продовжити роботу із застосунком без аккаунту або ж перейти до реєстрації, якщо аккаунт відсутній (рис. 3.25).

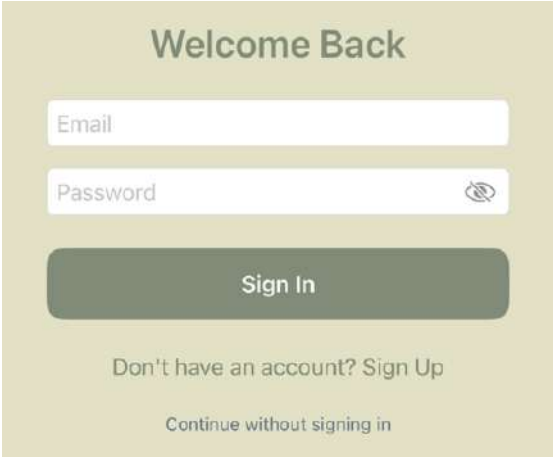

Рисунок 3.25 – Екран авторизації користувача

Для реєстрації потрібно вказати свою електронну пошту, пароль, а також ім’я користувача. Варто відмітити, що на екрані реєстрації реалізована валідація полів: пошта має бути в коректному форматі, ім’я не може бути порожнім, а пароль повинен мати не менше восьми символів, з яких один має бути великою літерою, один – цифрою, а ще один – спеціальним символом (рис. 3.26).

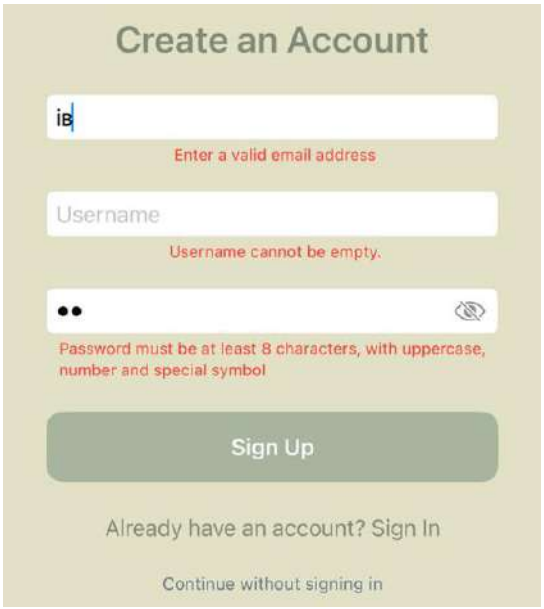

Рисунок 3.26 – Екран реєстрації користувача

Оскільки стандартний компонент TextField у SwiftUI не дозволяє показувати прихований і відкритий текстовий ввід за допомогою перемикача, для поля паролю було створене власне рішення – компонент PasswordFieldView. Він розроблений на основі ZStack та використовує @State змінну isPasswordVisible для того, щоб змінювати режим відображення при натиску на іконку. PasswordFieldView використовується для форм на екранах авторизації та реєстрації (див. рис. 3.25, 3.26).

Як і SignInViewModel, так і SignUpViewModel використовують сервіс AuthService для взаємодії із сервером за допомогою URLSession. З метою забезпечення централізованого доступу до сервісу використано шаблон проєктування Singleton, який забезпечив єдиний екземпляр класу AuthService в межах застосунку. Для коректної обробки можливих помилок також створено власний тип помилки AuthError, що охоплює такі типові ситуації, як invalidURL, invalidResponse, decodingFailed тощо.

Після успішної реєстрації автоматично здійснюється авторизація, щоб уникнути повторного введення облікових даних.

Отриманий внаслідок авторизації JWT-токен зберігається в Keychain – захищеному сховищі, що надається операційною системою [30]. Для здійснення цього реалізований сервіс KeychainService, що працює з бібліотекою KeychainAccess, яка є обгорткою над Keychain [31]. KeychainService, як і AuthService, працює за шаблоном Singleton, так як його використовують SignInViewModel, SignUpViewModel, ProfileViewModel, а також UserService.

Для авторизованого користувача на вкладці з профілем кнопка «Увійти» міняється на «Вийти», а також вгорі відображається його ім’я, електронна пошта та скільки часу він є користувачем застосунку. У авторизованого користувача з’являється додаткова функціональна можливість – отримати свої завдання як файл у форматі CSV на електронну пошту (рис. 3.27).

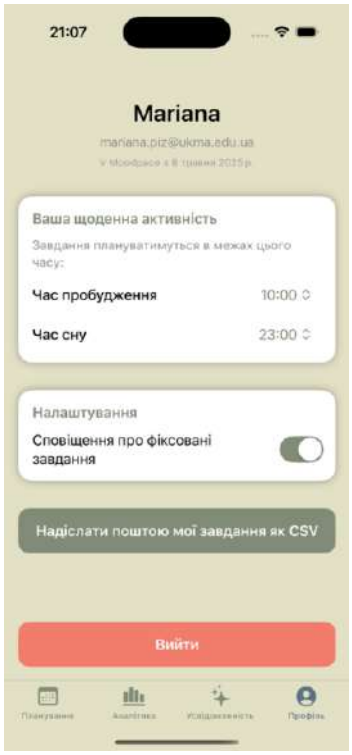

Рисунок 3.27 – Вкладка профілю користувача (авторизований)

Зважаючи на те, що вихід із аккаунту – це деструктивна дія, перш ніж здійснити її, користувач повинен повторно підтвердити свій намір або ж скасувати його за допомогою confirmationDialog. При виконанні виходу з аккаунту токен видаляється з Keychain у ProfileViewModel.

3.6.5 Сповіщення

Для того, щоб користувач не пропускав початок фіксованих завдань, прийнято рішення реалізувати механізм локальних сповіщень із використанням системного фреймворку UserNotifications [32].

Щоб уможливити отримання сповіщень із iOS-застосунку, користувач повинен одноразово надати дозвіл на це. Відповідно, одразу при першому запуску застосунку в методі application(_:didFinishLaunchingWithOptions:) класу AppDelegate відбувається запит на можливість надсилати сповіщення. Проте основною причиною реалізації та інтеграції AppDelegate в SwiftUI-середовище є те, що він стає делегатом UNUserNotificationCenterDelegate. Це дає змогу

показувати сповіщення із застосунку навіть тоді, коли він є відкритим. До структури MoodpaceApp клас AppDelegate інтегрується за допомогою @UIApplicationDelegateAdaptor, що створює й утримує його протягом життєвого циклу застосунку.

Щоб централізовано керувати всіма сповіщеннями, реалізовано клас NotificationService з відповідними методами для створення та видалення запланованого сповіщення. Прийнято рішення надсилати два сповіщення: за 15 хв до початку завдання та безпосередньо в момент початку. На рисунку 3.28 можна побачити приклад сповіщення, надісланого за 15 хв до початку завдання.

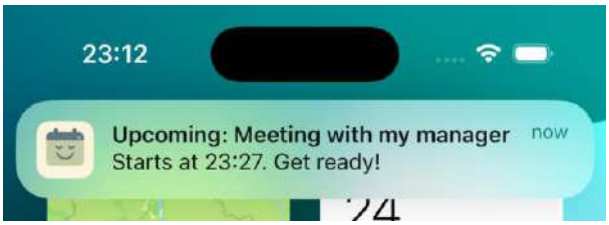

Рисунок 3.28 – Локальне сповіщення про наближення фіксованого завдання

Так як сповіщення повинні мати унікальні ідентифікатори, до UUID самого завдання додаються суфікси «-reminder» та «-start» відповідно. Попередньо вже наявні сповіщення з такими ідентифікаторами видаляються, щоб уникнути дублювання при редагуванні деталей завдання. TaskViewModel викликає відповідні методи NotificationService: scheduleNotification(for:) – при створенні чи редагуванні завдання, cancelNotification(for:) – при видаленні завдань. Це забезпечує синхронізацію сповіщень зі станом завдань.

3.7 Локалізація інтерфейсу

Починаючи з Xcode 15, рекомендованим підходом для забезпечення мультимовності застосунку є String Catalog [33]. Зважаючи на це, у проєкті використано наступний механізм локалізації: створено каталог із підкаталогами для англійської (en.lproj) та української (uk.lproj) мов (рис. 3.29).

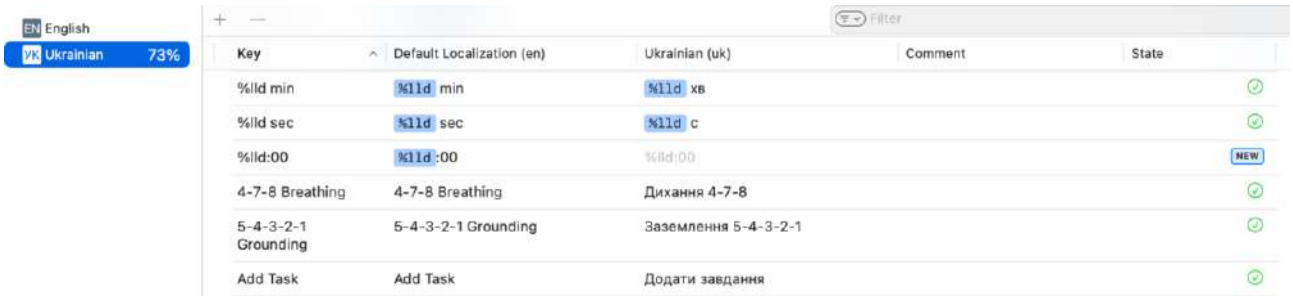

Рисунок 3.29 – String Catalog у Xcode

Для того, щоб текст можна було перекладати в підкаталогах, він має або бути рядковим літералом усередині компонента, або створений за допомогою конструктора String(localized:) [34].

Окрім, власне, текстових літералів, Xcode також підтягує для перекладу специфікатори форматування (наприклад, %@, %lld), даючи змогу локалізувати не лише статичні рядки, а й текст із динамічними параметрами.

Таким чином, при зміні пріоритетної мови в налаштуваннях пристрою, вона автоматично змінюватиметься і в застосунку.

3.8 Висновки до розділу 3

У цьому розділі розглянуто технічну реалізацію iOS-застосунку Moodpace, починаючи з формування вимог та ідейної концепції, продовжуючи вибором архітектури MVVM, організацією файлової структури, забезпечення автоматизованої перевірки стилю та форматування коду й закінчуючи описом реалізованої функціональності планування завдань, аналітики, усвідомленості, взаємодії з профілем та сповіщень. Надано скриншоти користувацького інтерфейсу. Описано процес локалізації інтерфейсу за допомогою String Catalog.

Отже, реалізований застосунок відповідає поставленим вимогам: емоційний стан інтегрований у планування справ, забезпечено мінімалістичний та зрозумілий інтерфейс користувача, надано можливість аналізувати свої емоції та налаштовувати графік.

ВИСНОВКИ

Аналіз наявних програмних рішень у сферах продуктивності та ментального здоров’я допоміг виявити відсутність таких застосунків, які б автоматично поєднували ці два аспекти. Це дало можливість зрозуміти необхідність розробки власного програмного забезпечення з фокусом на адаптивному плануванні. Зважаючи на переваги й недоліки розглянутих програмних рішень, сформовано ряд вимог до власного застосунку.

Зважаючи на ці вимоги, обрано платформу (iOS), технології (Swift, SwiftUI та SwiftData) та інструменти для розробки (SwiftLint і SwiftFormat), що дозволяють реалізувати ефективне й надійне програмне рішення з привабливим користувацьким інтерфейсом.

Продумано ідейну концепцію застосунку, підібрано доречну назву, логотип та кольорову гаму для нього. Для технічної реалізації мобільного застосунку вирішено використовувати архітектуру MVVM, згідно з якою організовано й файлову структуру проєкту. Дотримано принципів розподілу обов’язків між компонентами та їхнього перевикористання. Завдяки цьому застосунок легко піддається розширенню. Реалізовано основні функціональні можливості: планування фіксованих і гнучких завдань, аналітику емоцій відповідно до категорій завдань, вправи на усвідомленість та взаємодію із параметрами профілю. Інтерфейс застосунку локалізовано на українську мову за допомогою String Catalog.

Основною відмінністю застосунку від уже наявних стало адаптивне перепланування завдань, яке працює за розробленим алгоритмом і сортує гнучкі завдання залежно від емоції, що відчуває користувач після виконання останнього завдання. Також, зважаючи на емоцію на наявність вільного часу до наступного завдання, застосунок може пропонувати виконати вправи на усвідомленість, таким чином допомагаючи користувачеві опрацювати й відпустити його негативні емоції.

Здобуті результати можуть бути використані у сфері планування часу та підтримки ментального здоров’я, а також слугувати основою для подальшої розробки систем адаптивного планування. Розроблений застосунок може використовуватися користувачами, що зацікавлені в продуктивності не в шкоду своєму емоційному стану. Також він має потенціал бути корисним у корпоративному середовищі для аналізу того, які завдання приносять задоволення працівникам.

Подальший розвиток застосунку може бути спрямований на використання методів машинного навчання для покращення алгоритму адаптивного планування на основі аналізу історії емоційних реакцій користувача на завдання. Також доцільно розглянути можливість інтеграції з інструментами збору даних про показники здоров’я, зокрема з фреймворком Apple HealthKit, для збору фізичних показників та їхнього врахування при плануванні справ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. DailyBean [Електронний ресурс]. — Режим доступу: https://apps.apple.com/us/app/dailybean-simplest-journal/id1553223828.
2. Bearable [Електронний ресурс]. — Режим доступу: https://apps.apple.com/ua/app/bearable-symptom-tracker/id1482581097.
3. TickTick [Електронний ресурс]. — Режим доступу: https://ticktick.com.
4. Notion [Електронний ресурс]. — Режим доступу: https://www.notion.com.
5. Boogaard K. 4 Time Management Strategies to try [Електронний ресурс] / K.Boogaard // Atlassian. — 29.01.2025. — Режим доступу: https://www.atlassian.com/blog/productivity/time-management-strategies.
6. Hülsheger U. R., Alberts H. J. E. M., Feinholdt A., Lang J. W. B. Benefits of Mindfulness at Work: The Role of Mindfulness in Emotion Regulation, Emotional Exhaustion, and Job Satisfaction [Електронний ресурс] // Journal of Applied Psychology. — 2013. — Vol. 98, № 2. — С. 310-325. — Режим доступу: https://www.researchgate.net/publication/234018520_Benefits_of_Mindfulness_at_Work_The_Role_of_Mindfulness_in_Emotion_Regulation_Emotional_Exhaustion_and_Job_Satisfaction.
7. Kumar P. A., Vaishnavi M., Sinduja R. та ін. The Influence of Emotional Intelligence on Employee Engagement and Productivity [Електронний ресурс] // Advances in Cosumer Research. — 2025. — Vol. 2, Iss. 1 (Jan-Feb) 2025. — С. 86-93. — Режим доступу: https://www.researchgate.net/publication/388869691_The_Influence_of_Emotional_Intelligence_on_Employee_Engagement_and_Productivity.
8. Backlinko Team. 14 Remote Work Statistics [Електронний ресурс] // Backlinko. — 23.04.2025. — Режим доступу: https://backlinko.com/remote-work-stats.

9. Дідик Ж. Notion запустив новий додаток — Notion Calendar [Електронний ресурс] // The Instapreneurs. — 18.01.2024. — Режим доступу: https://www.theinstapreneurs.com.ua/blog-posts/notion-calendar.
10. Wurmser, Y. The Majority of Americans' Mobile Time Spent Takes Place in Apps [Електронний ресурс] // eMarketer. — 09.07.2020. — Режим доступу: https://www.emarketer.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps.
11. Apple Inc. Privacy [Електронний ресурс] // Apple. — Режим доступу: https://www.apple.com/privacy.
12. Twarogal P., Hanzel S. 136% More People Have Androids Than iPhones [Електронний ресурс] // Neontri. — 04.02.2025. — Режим доступу: https://neontri.com/blog/android-iphone-statistics-report.
13. Mustapha A. iPhone Users Spend A Lot More on Apps Than Android Users [Електронний ресурс] // GizChina. — 09.06.2024. Режим доступу: https://www.gizchina.com/2024/06/09/iphone-users-spend-a-lot-more-on-apps-than-android-users.
14. Rafalski K. Top iOS Programming Languages for App Development in 2025 [Електронний ресурс] // Netguru. — 31.03.2025. — Режим доступу: https://www.netguru.com/blog/top-ios-programming-languages.
15. Hnatiuk I. What is Swift programming language and why should you use it? [Електронний ресурс] // Blackthorn Vision. — 03.03.2024. — Режим доступу: https://blackthorn-vision.com/blog/what-is-swift-programming-language-and-why-should-you-use-it.
16. Borla H. Announcing Swift 6 [Електронний ресурс] // Swift.org. — 17.09.2024. — Режим доступу: https://www.swift.org/blog/announcing-swift-6.
17. Package Manager [Електронний ресурс] // Swift.org. — Режим доступу: https://www.swift.org/documentation/package-manager.

18. Dhruv S. SwiftUI vs. UIKit: iOS Development Comparison [Електронний ресурс] // Aalpha. — 15.04.2025. — Режим доступу: https://www.aalpha.net/blog/swiftui-vs-uikit-comparison.
19. Apple Inc. SwiftUI [Електронний ресурс] // Apple Developer. — Режим доступу: https://developer.apple.com/xcode/swiftui.
20. Apple Inc. UIKit Integration [Електронний ресурс] // Apple Developer Documentation. — Режим доступу: https://developer.apple.com/documentation/swiftui/uikit-integration.
21. Apple Inc. Core Data [Електронний ресурс] // Apple Developer Documentation. — Режим доступу: https://developer.apple.com/documentation/coredata.
22. SwiftlyNomad. CoreData vs Realm: What to Choose as a Database for iOS [Електронний ресурс] // Medium. — 13.05.2024. — Режим доступу: https://swiftlynomad.medium.com/coredata-vs-realm-what-to-choose-as-a-database-for-ios-7c1d09911d8f.
23. Shanmugam K. Core Data vs. Swift Data: A Detailed Comparison with Real-Time Solutions [Електронний ресурс] // Medium. — 25.06.2024. — Режим доступу: https://medium.com/@kalidoss.shanmugam/core-data-vs-swift-data-a-detailed-comparison-with-real-time-solutions-54ba8142c100.
24. Lockwood N. SwiftFormat [Електронний ресурс] // Github. — Режим доступу: https://github.com/nicklockwood/SwiftFormat.
25. Realm. SwiftLint [Електронний ресурс] // Github. — Режим доступу: https://github.com/realm/SwiftLint.
26. GeeksForGeeks. Functional vs. Non Functional Requirements [Електронний ресурс] // GeeksForGeeks. — Останнє оновлення: 08.01.2025. — Режим доступу: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements.
27. ColorHunt. Palette [Електронний ресурс] // ColorHunt. — Режим доступу: https://colorhunt.co/palette/5c7285818c78a7b49ee2e0c8.

28. Gronek J. MVC Swift vs MVVM on iOS: Key Differences With Examples [Електронний ресурс] // Netguru. — Останнє оновлення: 24.02.2025. — Режим доступу: https://www.netguru.com/blog/mvc-vs-mvvm-on-ios-differences-with-examples.
29. Sacchi R. Comparing MVVM and Viper architectures: When to use one or the other [Електронний ресурс] // Auth0. — 04.10.2016. — Режим доступу: https://auth0.com/blog/compare-mvvm-and-viper-architectures.
30. Ilgın Using Keychain: How to Securely Store and Access a Token [Електронний ресурс] // Medium. — 17.08.2023. — Режим доступу: https://medium.com/@ilginakgoz/using-keychain-how-to-securely-store-and-access-a-token-a0dc5bdf2d04.
31. Katsumi K. KeychainAccess [Електронний ресурс] // Github. — Режим доступу: https://github.com/kishikawakatsumi/KeychainAccess.
32. Apple Inc. Core Data [Електронний ресурс] // Apple Developer Documentation. — Режим доступу: https://developer.apple.com/documentation/usernotifications.
33. Apple Inc. Localization [Електронний ресурс] // Apple Developer Documentation. — Режим доступу: https://developer.apple.com/documentation/xcode/localization.
34. Apple Inc. Localizing and varying text with a string catalog [Електронний ресурс] // Apple Developer Documentation. — Режим доступу: https://developer.apple.com/documentation/xcode/localizing-and-varying-text-with-a-string-catalog.