МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ «МОТИВАЦІЯ: ЦИТАТИ ВІДОМИХ ЛЮДЕЙ» НА ПЛАТФОРМІ IOS»**

Виконав: студент 2 курсу, групи 8.1213-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
В.Д. Насонов
(ініціали та прізвище)

Керівник декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, д.т.н., професор Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Насонову Владиславу Дмитровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка мобільного додатку «Мотивація: цитати відомих людей» на платформі IOS

керівник роботи Гоменюк Сергій Іванович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка додатку та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.03.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.03.2024	
2.	Збір вихідних даних.	15.04.2024	
3.	Обробка методичних та теоретичних	27.06.2024	
	джерел.		
4.	Розробка першого та другого розділу.	03.09.2024	
5.	Розробка третього розділу.	15.11.2024	
6.	Оформлення та нормоконтроль	20.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	12.12.2024	

Студент _____ В.Д. Насонов
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.І. Гоменюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка мобільного додатку «Мотивація: цитати відомих людей» на платформі IOS»: 84 с., 34 рис., 2 табл., 12 джерел, 6 додатків.

ВЕБДОДАТОК, ДОСЛІДЖЕННЯ, ПЛАТФОРМА IOS, РОЗРОБКА, ФРЕЙМВОРК, SWIFT, USER NOTIFICATION FRAMEWORK.

Об’єкт дослідження – процес розробки мобільного додатку для підвищення мотивації користувачів за допомогою цитат відомих людей.

Мета роботи: розробка мобільного додатку для платформи iOS, що надає користувачам можливість отримувати надихаючі цитати відомих людей, стимулюючи їх до досягнення особистих цілей.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи проєктування та моделювання програмного забезпечення, методи тестування та налагодження програмного продукту, а також методи інтеграції хмарних технологій для синхронізації даних.

Створення мобільного додатку для платформи iOS, який працює як мотиватор для користувачів, надаючи їм цитати відомих людей для підвищення мотивації. Користувачі можуть обирати певну особу, цитати якої будуть надходити їм у вигляді повідомлень у встановлений проміжок часу. Прочитавши цитату, користувачі отримують натхнення та мотивацію для досягнення своїх цілей.

Ключовою особливістю додатку є можливість відкривати нові цитати відомих людей за допомогою зароблених монет. Користувачі можуть отримувати монети щодня, заходячи у додаток, або виконуючи завдання, що також сприяють мотивації. Це забезпечує додатковий стимул для регулярного використання додатку.

Цей додаток може бути використаний для розробки інших мобільних додатків, що включають елементи мотивації та взаємодії з користувачами.

SUMMARY

Master’s qualifying paper “Development of the Mobile Application "Motivation: Quotes of Famous People" for the IOS Platform”: 84 pages, 34 figures, 2 tables, 12 references, 6 supplements.

WEB-APPLICATION, RESEARCH, IOS PLATFORM, DEVELOPMENT, FRAMEWORK, SWIFT, USER NOTIFICATION FRAMEWORK.

The object of the study is the process of developing a mobile application to increase user motivation using quotes from famous people.

The aim of the study is development of a mobile application for the iOS platform, which provides users with the opportunity to receive inspirational quotes from famous people, encouraging them to achieve their personal goals.

The method of research is methods for collecting and analyzing software requirements, methods for designing and modeling software, methods for testing and debugging a software product, and methods for integrating cloud technologies for data synchronization.

Create a mobile application for the iOS platform that works as a motivator for users, providing them with quotes from famous people to increase motivation. Users can choose a certain person whose quotes will be sent to them in the form of notifications within a set period of time. After reading the quote, users get inspiration and motivation to achieve their goals.

A key feature of the app is the ability to unlock new quotes from famous people using the coins earned. Users can get coins every day by logging into the app or completing tasks that also contribute to motivation. This provides an additional incentive to use the app regularly.

This application can be used to develop other mobile applications that include elements of motivation and user interaction.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 8
1 Постановка задачі..... 11
1.1 Аналіз предметної області 11
1.2 Огляд існуючих програмних рішень..... 11
1.3 Порівняльний аналіз існуючих програмних рішень 12
1.4 Аналіз вимог до додатку 14
1.5 Визначення цільової аудиторії 15
2 Основні теоретичні відомості 16
2.1 Результат аналізу за допомогою UML-діаграми 16
2.2 Результат аналізу за допомогою ER-діаграми 19
2.3 Існуючі інструменти для вирішення подібних задач 23
2.4 Детальний опис обраного підходу 25
2.4.1 Технології та фреймворки використані у процесі розробки ... 26
2.4.2 Алгоритми та методи використані у процесі розробки 27
3 Розробка додатку та тестування 29
3.1 Детальний опис обраного підходу 29
3.2 Конфігурація середовища для розробки в Xcode 31
3.3 Необхідні файли проєкту та їх функції..... 31
3.3.1 Файл AppDelegate.swift 32
3.3.2 Файл SceneDelegate.swift..... 33
3.3.3 Файл ViewController.swift 34
3.4 Реалізація основних функціональних модулів..... 35
3.4.1 Запит дозволу на сповіщення 35
3.4.2 Персоналізоване привітання користувача..... 36

3.4.3 Інтеграція вебсторінки «Ти як?» 37
3.4.4 Логіка обміну монет на зірки 38
3.4.5 Функція «Випробуй удачу» 39
3.4.6 Завантаження та вибір цитат 40
3.4.7 Налаштування параметрів цитат 41
3.4.8 Реалізація надсилання цитат 42
3.4.9 Реалізація кнопок для керування різними функціями додатку 43
3.5 Запуск та тестування додатку на практиці 45
Висновки 65
Перелік посилань 67
Додаток А MainController.swift 68
Додаток Б QuotesController.swift 70
Додаток В SettingsController.swift 76
Додаток Г ShopController.swift 79
Додаток Д TryYourLuckController.swift 81
Додаток Е DateTimeController.swift 83

ВСТУП

Сьогодні мобільні додатки стали важливим елементом у щоденному житті, забезпечуючи зручний і швидкий доступ до різних видів контенту, зокрема мотиваційного. Все більше людей прагнуть отримувати корисну інформацію миттєво і в зручній формі, і саме ця потреба зумовлює високу популярність мотиваційних додатків, які стимулюють розвиток і зосередженість користувачів на особистих цілях.

З точки зору предметної області, розвиток мобільних додатків у сфері саморозвитку пов’язаний з глобальною тенденцією до підвищення особистої ефективності та психологічного добробуту. Раніше люди зверталися до книг, статей або мотиваційних тренінгів, які вимагали більше часу та зусиль. Однак мобільні додатки дозволяють значно швидше та легше отримувати мотивацію завдяки автоматизованим повідомленням, персоналізованим налаштуванням і постійній доступності. Крім того, сучасні мобільні додатки пропонують інтерактивні механізми взаємодії, наприклад, щоденні нагороди за активність, що посилює мотивацію користувачів до регулярного використання.

Враховуючи ці тенденції, створення додатку для платформи iOS, який надсилає користувачам цитати відомих людей, є своєчасним і відповідає попиту на сучасний та доступний засіб для щоденного натхнення. Це не лише полегшує процес отримання мотиваційних матеріалів, а й надає користувачам можливість вибирати цитати улюблених особистостей, що робить додаток персоналізованим та зручним для кожного. Все це підкреслює актуальність даного проєкту та його потенціал для широкої аудиторії користувачів.

Мета – метою роботи є створення мобільного додатку на платформі iOS, який надає користувачам доступ до мотиваційних цитат відомих людей. Цей додаток сприятиме натхненню та особистісному розвитку користувачів, допомагаючи їм підтримувати високий рівень мотивації та ефективності у повсякденному житті.

Досягнення поставленої мети передбачає реалізацію наступних **завдань**:

- проаналізувати існуючі мобільні додатки та їхні функціональні можливості в області мотивації та саморозвитку для визначення найкращих практик і актуальних тенденцій;
- дослідити інструменти та технології, зокрема Swift, Storyboard Interface, User Defaults, User Notifications Framework, для забезпечення ефективної реалізації функцій додатка;
- розробити структуру даних та архітектуру додатка, що включає систему отримання та зберігання мотиваційних цитат відомих людей;
- спроєктувати інтерфейс користувача (UI) додатка, забезпечуючи його зручність та інтуїтивність, з урахуванням рекомендацій з дизайну для платформи iOS;
- реалізувати основний функціонал додатка, включаючи механізм отримання цитат, систему нагадувань та можливість персоналізації;
- протестувати додаток на відповідність функціональним вимогам, коректність роботи на різних пристроях iOS та стабільність роботи з локальним сховищем.

Об’єкт дослідження – процес підвищення мотивації користувачів за допомогою мобільних додатків на платформі iOS.

Предмет дослідження – реалізація мотиваційного додатку, що надсилає користувачам цитати відомих людей через сповіщення, з можливістю накопичення та використання внутрішньої валюти для персоналізації цитат.

Методи дослідження – для реалізації проєкту було обрано мову програмування Swift, яка дозволяє якісно створювати додатки для платформи iOS завдяки її потужним можливостям для обробки подій, побудови інтерфейсу та роботи з даними. Swift забезпечує високу продуктивність і безпеку, що є важливими для мобільних додатків. Для створення користувацького інтерфейсу використано Storyboard Interface, що трішки спрощує розробку та дозволяє швидко компонувати елементи інтерфейсу за

допомогою користувацького редактора. Зберігання даних організовано через UserDefaults, що дозволяє ефективно зберігати інформацію про цитати та налаштування користувачів локально на пристрої. UserDefaults є зручним для збереження невеликих обсягів даних, таких як параметри користувача або дані користувача. Для надсилання сповіщень обрано User Notifications Framework, завдяки якому реалізовано автоматичну доставку цитат у встановлений час, що дозволяє користувачам отримувати мотиваційні повідомлення навіть у фоновому режимі. Це забезпечує інтерактивність та підвищує залученість користувачів до додатку.

Результати роботи можуть бути використані в освітніх ініціативах, допомагаючи студентам та учням підтримувати мотивацію до навчання. Особливо актуальною є реалізація цього додатку в період війни, коли людям необхідна підтримка та натхнення для подолання труднощів. Додаток може стати важливим інструментом для тих, хто потребує моральної підтримки, стимулюючи позитивне мислення та бажання досягати своїх цілей навіть у складних умовах. Практичне значення проєкту полягає в розробці інструменту, який допомагає людям залишатися мотивованими, досягаючи своїх цілей, а також у можливості подальшого розвитку додатку, наприклад, шляхом додавання нових функцій або інтеграції з іншими сервісами.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань та 6 додатків. Загальний обсяг роботи становить 84 сторінки. Додатки викладено на 17 сторінках. У переліку посилань 12 використаних джерел.

Перший розділ присвячений постановці задачі та визначенню основних цілей і вимог до додатку. У другому розділі наведено теоретичні відомості, що охоплюють технології та методи, використані в роботі, проілюстровано UML та ER діаграми. В третьому розділі описано реалізацію розробки додатку та тестування, включаючи етапи створення та перевірки функціональності. В додатках наведено програмну реалізацію, повний код усього проєкту.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Мотивація є не лише ключовим чинником для досягнення успіху, але й каталізатором для особистісного зростання. У сучасному світі, де люди постійно зіткнюються зі стресом, виснаженням, і відкладають важливі справи на потім, мотиваційні цитати відомих особистостей можуть стати надзвичайно цінним ресурсом. Вони не лише надихають, але й надають підтримку в ті моменти, коли ви відчуваєте сумніви або втомленість.

Такі додатки не лише надають користувачам можливість зберегти мотивацію на високому рівні, але й допомагають їм розширити свій кругозір та вивчити думки та поради видатних особистостей. Вони сприяють формуванню позитивного мислення та допомагають зосередитися на досягненні поставлених цілей.

1.2 Огляд існуючих програмних рішень

Відомі програмні рішення демонструють різноманітність підходів до стимулювання мотивації користувачів через цитати, афірмації та інші методи. Кожен з них має свої унікальні особливості, які можуть відповідати певним потребам та вподобанням користувачів.

Motivation – Daily Quotes, цей додаток надає щоденні мотивуючі цитати з можливістю налаштування сповіщень. Користувачі можуть обирати цитати певної відомої особистості, що дозволяє їм обрати саме ту персону яка на думку користувача є його ідолом для натхнення.

ThinkUp, цей додаток спрямований на саморозвиток і використовує афірмації та цитати для покращення самосприйняття та мотивації. Він надає

користувачам можливість працювати над певними аспектами свого життя через позитивне програмування свідомості.

Shine, цей додаток комбінує щоденні мотиваційні повідомлення з аудіоповідомленнями, медитаціями та іншими ресурсами для саморозвитку. Він пропонує користувачам різноманітні способи надихатися та зосереджуватися на важливих аспектах їхнього життя.

1.3 Порівняльний аналіз існуючих програмних рішень

Кожен з цих додатків має свої сильні та слабкі сторони. Порівнюючи альтернативні рішення у сфері мотиваційних додатків, можна виділити деякі наступні аспекти.

Motivation – Daily Quotes:

– функціонал: щоденні цитати, можливість налаштування сповіщень, пошук за темами;
– переваги: велика база цитат, простота використання;
– недоліки: відсутність гейміфікації та завдань для заробітку монет.

ThinkUp:

– функціонал: афірмації, мотиваційні цитати, аудіоформат;
– переваги: персоналізація, аудіопідтримка;
– недоліки: фокус на афірмаціях, а не на цитатах відомих людей.

Shine:

– функціонал: щоденні мотиваційні повідомлення, медитації, аудіоповідомлення;
– переваги: різноманітність контенту, інтеграція медитацій;
– недоліки: платні функції, складніший інтерфейс.

У таблиці 1.1 представлено зведене порівняння альтернативних рішень.

Таблиця 1.1 – Порівняння альтернативних рішень

Пункт	**Motivation – Daily Quotes**	**ThinkUp**	**Shine**
Функціонал	Щоденні цитати, налаштування сповіщень, пошук за темами	Афірмації, мотиваційні цитати, аудіоформат	Щоденні мотиваційні повідомлення, медитації, аудіоповідомлення
Персоналізація	Обмежена	Так	Так
Аудіоформат	Немає	Так	Так
Наявність гейміфікації	Немає	Немає	Немає
Можливість перегляду архіву	Так	Так	Так
Фокус на афірмаціях	Немає	Так	Немає
Різноманітність контенту	Немає	Немає	Так
Інтеграція медитацій	Немає	Немає	Так
Платні функції	Немає	Немає	Так
Складність інтерфейсу	Немає	Немає	Висока

Підсумовуючи, Motivation – Daily Quotes пропонує великий вибір цитат, що надихають, з простим і зрозумілим інтерфейсом. Однак, додаток не включає елементи гейміфікації, що може бути важливим для користувачів, які шукають інтерактивні або ігрові механізми. ThinkUp надає персоналізовані афірмації, що дозволяють налаштувати контент під індивідуальні потреби,

проте він може не задовольнити тих, хто шукає саме цитати відомих осіб. Shine, в свою чергу, відзначається широким спектром контенту, що охоплює не тільки мотиваційні цитати, але й поради для психічного здоров'я. Проте, Shine має платні функції та більш складний інтерфейс, що може відштовхувати деяких користувачів, шукаючих простоту.

1.4 Аналіз вимог до додатку

Мобільний додаток «Мотивація: цитати відомих людей» буде поєднувати сильні сторони існуючих рішень, але з додатковими унікальними функціями:

- гейміфікація: користувачі матимуть можливість заробляти монети за виконання досягнень та за щоденний вхід у додаток – ці монети можна буде використовувати для відкриття нових цитат або отримання додаткових функцій;
- налаштування сповіщень: користувачі матимуть можливість налаштовувати час отримання цитат у вибраний час дня – це дозволить їм отримувати дозу мотивації у ті моменти, коли вони найбільш необхідні;
- персоналізація: користувачі зможуть вибирати цитати від конкретних людей, які їх найбільше надихають – це створить індивідуальний досвід мотивації, враховуючи особисті вподобання та цінності кожного користувача.

Розроблений на мові програмування Swift у програмі Xcode з використанням технологій Storyboard Interface, User Defaults, User Notifications Framework, додаток поєднує в собі передові технології з інноваційним підходом до мотивації користувача. Його унікальні можливості і функціонал роблять його цікавим для широкого кола людей, які бажають досягати своїх цілей та надихатися відомими особистостями.

1.5 Визначення цільової аудиторії

Мобільний додаток «Мотивація: цитати відомих людей» буде поєднувати сильні сторони існуючих рішень, але з додатковими унікальними функціями.

Цільова аудиторія мобільного додатку «Мотивація: цитати відомих людей» охоплює широкий спектр користувачів, для яких важливо підтримувати мотивацію, отримувати натхнення та знаходити внутрішні ресурси для досягнення цілей.

Однією з ключових груп є студенти та учні. Молоді люди, які навчаються, часто стикаються зі складнощами в навчальному процесі, відчувають тиск дедлайнів і потребують натхнення для досягнення високих результатів. Регулярні мотиваційні цитати допоможуть їм залишатися зосередженими та впевненими у своїх можливостях.

Молоді фахівці також входять до цільової аудиторії. На початку кар'єри вони часто відчувають стрес, адаптуються до нових робочих умов і прагнуть знайти баланс між роботою та особистим життям. Додаток допоможе їм залишатися продуктивними та зберігати мотивацію.

Особливу увагу приділено підприємцям та стартаперам. Їхній шлях зазвичай пов'язаний із ризиками, невизначеністю та необхідністю приймати складні рішення. Мотиваційні повідомлення можуть стати джерелом впевненості, інструментом для подолання стресу та поштовхом до нових ідей.

Люди, які прагнуть особистісного розвитку, також становлять важливу аудиторію. Вони активно працюють над собою, намагаються розвивати нові навички та покращувати якість свого життя. Для цієї групи додаток стане щоденним джерелом натхнення, допомагаючи їм досягати нових вершин.

Окремим сегментом аудиторії є люди, які потребують підтримки та мотивації в умовах війни. Військові, волонтери, переселенці та ті, хто переживає важкі часи, часто шукають слова натхнення та віру в краще майбутнє. Додаток допоможе їм підтримувати психологічну стійкість.

2 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Результат аналізу за допомогою UML-діаграми

У розвитку програмного забезпечення використання стандартизованих методів та інструментів грає важливу роль у забезпеченні якості та ефективності процесу розробки. Unified Modeling Language (UML) є одним із найпоширеніших інструментів для моделювання програмних систем.

У цьому зв’язку, в даному дослідженні використовується UML-діаграма прецедентів для аналізу та визначення функціональних можливостей мобільного додатку «Мотивація: цитати відомих людей» на платформі iOS. Ця діаграма дозволяє систематизувати та уточнити вимоги користувачів, визначити ключові сценарії взаємодії та встановити базові функціональність додатку.

У подальшому аналізі буде детально розглянуто кожен прецедент, визначаючи вхідні умови, дії користувача та наслідки, що допоможе зрозуміти, як користувачі будуть використовувати додаток, та як це допоможе їм збільшити мотивацію та досягнення успіху.

Вибір відомої особи:

– опис: користувач переглядає список відомих осіб та обирає одну з них;
– передумови: відсутні;
– наслідки: обрана особа стає джерелом цитат для користувача.

Налаштування параметрів цитат:

– опис: користувач обирає параметри для отримання цитат: стартову дату, час і інтервал між повідомленнями;
– передумови: користувач надав дозвіл на отримання сповіщень і обрав відому особу;
– наслідки: додаток надсилає цитати відповідно до вибраних параметрів.

Отримання мотиваційних повідомлень:

– опис: користувач отримує мотиваційні цитати від обраної відомої особи через сповіщення;
– передумови: користувач обрав відому особу як джерело цитат; користувач налаштував параметри цитат (вибрав стартову дату, час і інтервал між цитатами); користувач надав дозвіл на отримання сповіщень;
– наслідки: користувач регулярно отримує цитати, які надихають його до досягнення цілей.

Зароблення монет:

– опис: користувач заробляє монети за виконання досягнень у додатку або за щоденне заходження;
– передумови: відсутні;
– наслідки: монети додаються до рахунку користувача.

Обмін монет на зірки:

– опис: користувач обмінює певну кількість монет на зірки;
– передумови: у користувача є достатня кількість монет;
– наслідки: зірки додаються до рахунку користувача.

Розблокування нових персон:

– опис: користувач витрачає зірки для розблокування нових відомих осіб;
– передумови: у користувача є достатня кількість зірок;
– наслідки: нові відомі особи стають доступними для вибору.

Збільшення кількості доступних цитат в день:

– опис: користувач витрачає зірки для збільшення кількості цитат, які можна отримати за день;
– передумови: у користувача є достатня кількість зірок;
– наслідки: кількість доступних цитат у день збільшується.

На рисунку 2.1 представлено UML-діаграму бази даних.

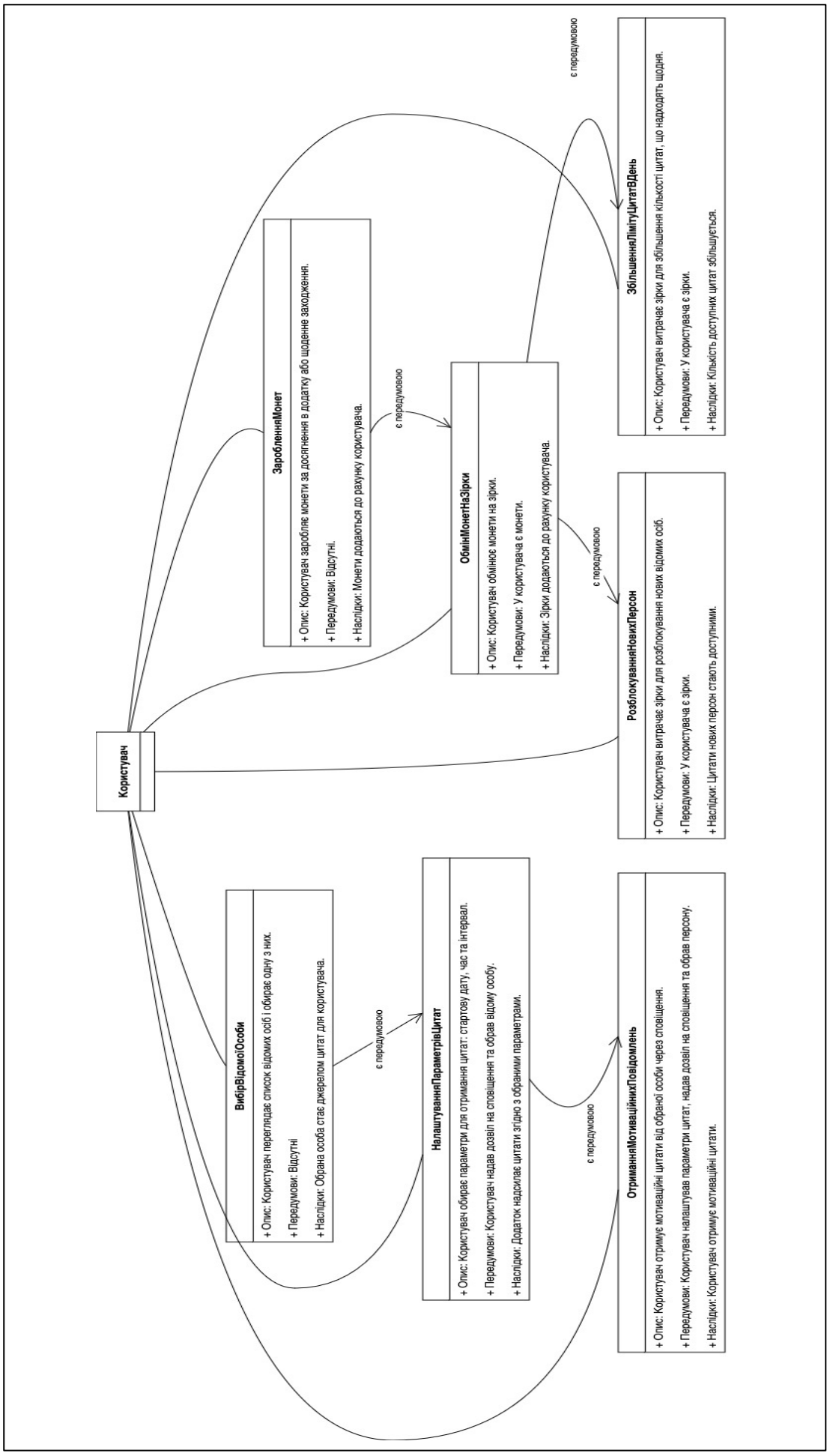


Рисунок 2.1 – UML-діаграма прецедентів

У результатах аналізу UML-діаграми прецедентів додатку «Мотивація: цитати відомих людей» відображено повну функціональність та взаємодію користувача з додатком. Ця діаграма визначає основні можливості додатку, такі як вибір відомої особи для отримання цитат, налаштування параметрів отримання цитат, отримання мотиваційних повідомлень, зароблення монет, обмін монет на зірки, а також розблокування нових персон та збільшення ліміту доступних цитат на день. Кожен прецедент чітко визначає вхідні умови, дії користувача та наслідки взаємодії з додатком. Фінальна UML-діаграма допомагає зрозуміти функціональність програми та забезпечує чіткість у розробці та виконанні кожного етапу роботи.

2.2 Результат аналізу за допомогою ER-діаграми

ER-діаграма є важливим інструментом для моделювання та аналізу бази даних. Ця діаграма відображає взаємозв’язки між сутностями (Entities) та атрибутами (Attributes), що є ключовим для розуміння структури бази даних та її функціонування.

Для цього додатку на платформі iOS, ER-діаграма створена з метою чіткого визначення сутностей та їхніх взаємозв’язків. Ця діаграма дозволяє зрозуміти структуру бази даних додатка, включаючи ключові компоненти, такі як користувачі, відомі особи, цитати, монети, зірки, параметри отримання цитат та можливості для їх розширення.

На рисунку 2.2 представлено ER-діаграму бази даних.

Stars (Зірки).

Атрибути:

– quantity (кількість зірок): кількість зірок, які має користувач.

Зв’язки:

– користувач має зірки, які можуть бути використані для розблокування нових персон або збільшення ліміту цитат на день.

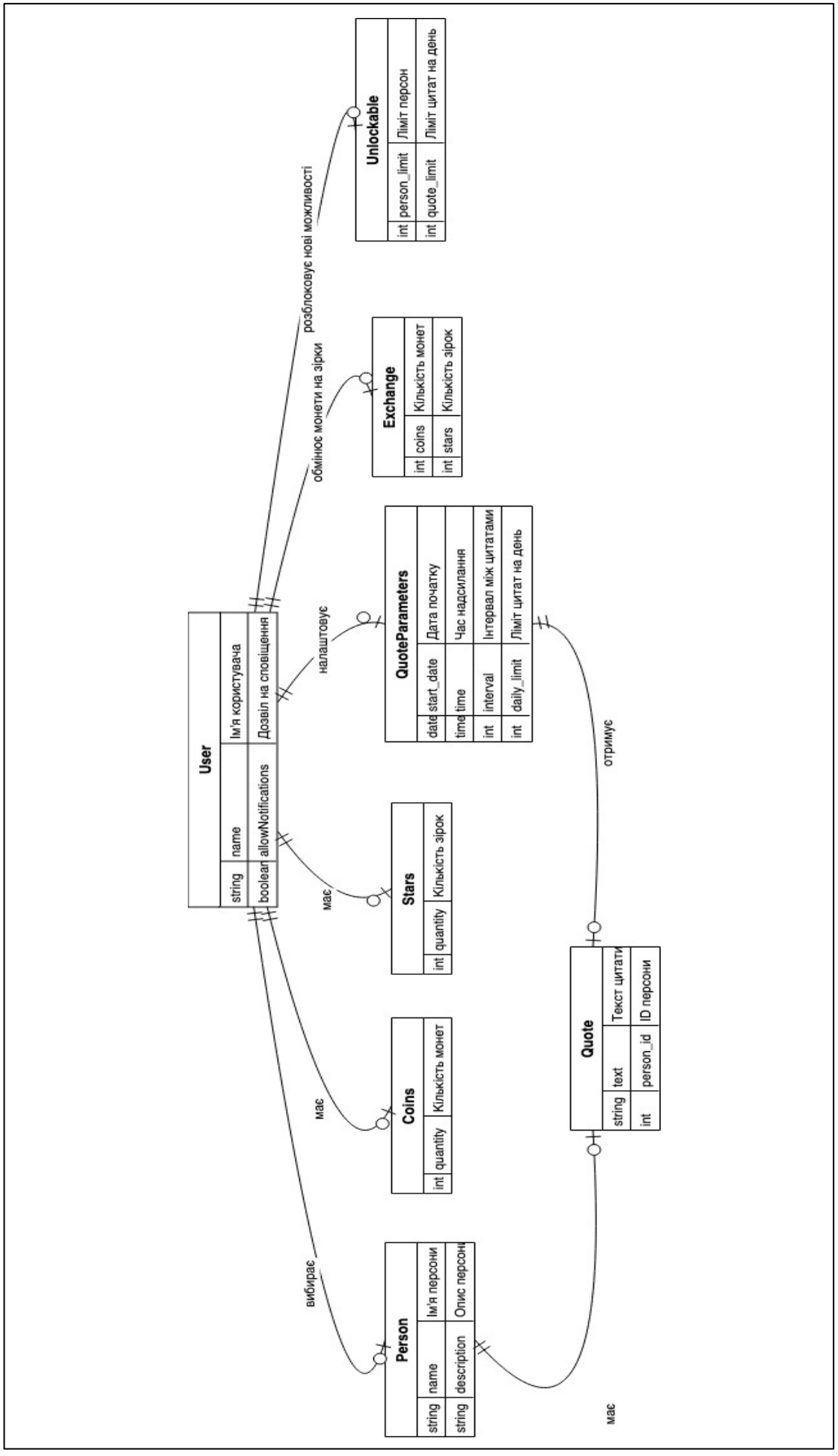


Рисунок 2.2 – ER-діаграма бази даних

User (Користувач).

Атрибути:

– name (ім’я користувача): ім’я користувача в додатку;
– allowNotifications (дозвіл на сповіщення): прапор, що визначає, чи надав користувач дозвіл на отримання сповіщень.

Зв’язки:

– користувач може вибирати одну персону;
– користувач має монети та зірки, які використовуються для різних функцій в додатку (наприклад, розблокування персон, збільшення ліміту цитат);
– користувач налаштовує параметри для отримання цитат (дату, час початку та інтервал між цитатами);
– користувач може обмінювати монети на зірки для розблокування нових можливостей;
– користувач може розблоковувати нові можливості, наприклад, збільшувати кількість доступних персон або ліміт цитат на день за допомогою зірок.

Person (Персона).

Атрибути:

– name (ім’я персони): ім’я відомої особи, цитати якої доступні для користувача;
– description (опис персони): опис відомої особи, який допомагає користувачеві зробити вибір.

Зв’язки:

– кожна персона має багато цитат, які можуть бути надані користувачеві.

Quote (Цитата).

Атрибути:

– text (текст цитати): текст мотиваційної цитати від обраної персони;
– person_id (ID персони): ідентифікатор персони, від якої надходить цитата.

Зв’язки:

– кожна цитата належить певній персони.

Coins (Монети).

Атрибути:

– quantity (кількість монет): кількість монет, які має користувач.

Зв’язки:

– користувач має монети, які використовуються для різних функцій в додатку, наприклад, для обміну на зірки.

QuoteParameters (Параметри цитат).

Атрибути:

– start_date (дата початку): дата початку отримання цитат;
– time (час надсилання): час, коли буде надсилатися наступна цитата;
– interval (інтервал між цитатами): інтервал часу між надсиланням цитат;
– daily_limit (ліміт цитат на день): ліміт кількості цитат, які користувач може отримати протягом дня.

Зв’язки:

– параметри цитат визначають, стартову дату та як часто користувач отримуватиме цитати.

Exchange (Обмін).

Атрибути:

– coins (кількість монет): кількість монет, яку користувач обмінює на зірки;
– stars (кількість зірок): кількість зірок, яку користувач отримує в результаті обміну монет.

Зв’язки:

– користувач може обмінювати монети на зірки, щоб розблоковувати нові можливості.

Unlockable (Розблоковані можливості).

Атрибути:

– person_limit (ліміт персон): кількість доступних персон, які розблоковані користувачу для отримання цитат;
– quote_limit (ліміт цитат на день): кількість цитат, які користувач може отримати в день.

Зв’язки:

– користувач може розблоковувати нові можливості (наприклад, більше персон і цитат в день) за допомогою зірок.

У підсумку, ER-діаграма для додатка «Мотивація: цитати відомих людей» на платформі iOS є ключовим інструментом, що допомагає краще розібратися у структурі бази даних та взаємозв’язках між її складовими. Вона дозволяє ефективно моделювати та аналізувати дані, необхідні для функціонування додатка, забезпечуючи його стабільну та ефективну роботу.

Правильно створена ER-діаграма допомагає уникнути помилок при розробці програми, а також сприяє покращенню її продуктивності та масштабованості. Тому важливо ретельно проробити кожну сутність, атрибут та зв’язок на діаграмі, щоб забезпечити оптимальне функціонування додатка та задоволення потреб його користувачів.

2.3 Існуючі інструменти для вирішення подібних задач

При розробці мобільних додатків існує безліч інструментів та технологій, які можна використовувати. Кожен з цих інструментів має свої власні переваги та обмеження, і вибір правильного може бути важливим для успішного завершення проєкту.

Swift з **Xcode**. Xcode – це інтегроване середовище розробки (IDE), створене компанією Apple спеціально для розробки програм для iOS, macOS, watchOS та tvOS. Це найбільш популярне та офіційне середовище для розробки на мові програмування Swift. Воно містить різноманітні інструменти, такі як інтерфейс розробки за допомогою Storyboard або SwiftUI, дебагер, компілятор Swift, аналізатор профілювання та інші. Що стосується

технологій, таких як User Defaults для роботи з базою даних, User Notifications Framework для роботи з повідомленнями, Xcode має всі необхідні інструменти для їх використання та інтеграції в додаток.

React Native та **Flutter** – це крос-платформенні фреймворки, які дозволяють розробляти мобільні додатки для iOS та Android з використанням одного коду. React Native використовує JavaScript, тоді як Flutter використовує Dart. Обидва фреймворки мають свої переваги, такі як висока продуктивність, гарний вигляд інтерфейсу користувача та можливість швидкої розробки завдяки гарнітурі готових компонентів.

NativeScript та **Xamarin** також є крос-платформеними фреймворками для розробки мобільних додатків. NativeScript використовує JavaScript, а Xamarin – мову програмування C#. Обидва фреймворки мають глибоку інтеграцію з платформами iOS та Android, що дозволяє створювати додатки, які виглядають та працюють майже так само як нативні додатки.

Вибір між цими інструментами залежить від потреб, вмінь та переваг.

Swift разом із Xcode є оптимальним вибором для розробки нативних iOS-додатків завдяки високій продуктивності та інтеграції з екосистемою Apple. Однак, якщо потрібно створити крос-платформений додаток, альтернативи, такі як React Native, Flutter, NativeScript і Xamarin, можуть бути більш ефективними. React Native використовує JavaScript для обох платформ. Flutter на Dart забезпечує плавність. Xamarin і NativeScript дають доступ до нативних API.

У таблиці 2.1 представлено зведене порівняння популярних технологій.

Таблиця 2.1 – Порівняння популярних технологій

Характеристика	**Swift з Xcode**	**React Native**	**Xamarin**
Мова програмування	Swift	JavaScript	C#
Підтримувані платформи	iOS, macOS, watchos, tvOS	iOS, Android	iOS, Android, Windows

Продовження табл. 2.1

Характеристика	Swift з Xcode	React Native	Xamarin
Тип додатків, що розробляються	Нативні додатки для платформ Apple	Крос-платформені додатки для iOS та Android	Крос-платформені додатки для iOS, Android, Windows
Інтеграція з платформою	Глибока інтеграція з платформами Apple	Обмежена інтеграція з платформами Apple та Google	Глибока інтеграція з платформами Apple, Google, Microsoft
Продуктивність	Висока	Висока	Висока
Інструменти розробки	Xcode, Storyboard, SwiftUl, дебагер, компілятор Swif	Інструменти розробки React, дебагер	Інструменти розробки Visual Studio, дебагер
Спільнота та документація	Велика, офіційна документація від Apple	Велика, активна спільнота, документація	Велика, активна спільнота, документація
Важливість для iOS розробки	Висока	Середня	Низька
Важливість для кросплатформенної розробки	Низька	Висока	Висока

2.4 Детальний опис обраного підходу

Для розробки мобільного додатку «Мотивація: цитати відомих людей» на платформі iOS було обрано підхід Agile, зокрема методологію Scrum. Цей підхід вирізняється гнучкістю та здатністю адаптуватися до змін вимог, що є

критично важливим у процесі розробки. Scrum дозволяє організувати роботу команди у короткі ітерації, або спринти, що зазвичай тривають від одного до чотирьох тижнів. Це надає можливість часто перевіряти та коригувати напрямок розвитку проєкту.

Методологія Scrum забезпечує високу прозорість процесу розробки завдяки регулярним зустрічам, таким як щоденні стендапи та спринт-огляди. Це дозволяє всім учасникам проєкту бути залученими та в курсі поточного стану роботи. Ключова перевага Scrum полягає у можливості швидко реагувати на зміни, що дозволяє команді адаптуватися до нових вимог та умов навіть на пізніх стадіях розробки.

Завдяки частим релізам та ретроспективам команда може постійно покращувати як процеси розробки, так і якість кінцевого продукту. Це мінімізує ризики і забезпечує відповідність кінцевого продукту потребам користувачів та вимогам ринку. У результаті, використання методології Scrum дозволяє ефективно управляти проєктом і створити високоякісний додаток.

2.4.1 Технології та фреймворки використані у процесі розробки

Для створення додатку використовується мова програмування Swift. Swift є основною мовою для розробки додатків на платформі iOS завдяки своїй високій продуктивності, безпечності та легкості у вивченні. Swift також забезпечує інтеграцію з сучасними бібліотеками та фреймворками, що полегшує розробку складних функціональних можливостей.

Розробка додатку здійснюється в інтегрованому середовищі розробки Xcode. Xcode є офіційним середовищем розробки для платформи iOS та надає всі необхідні інструменти для створення, тестування та налагодження додатків. Воно включає візуальні редактори для інтерфейсів, симулятори пристроїв для тестування та потужні інструменти для діагностики і оптимізації продуктивності коду.

Інтерфейс користувача буде розроблятися за допомогою Storyboard, що є візуальним інструментом для створення UI у Xcode. Це дозволяє розробникам швидше та зручніше створювати та розташовувати UI елементи, забезпечуючи візуальне редагування та перегляд дизайну додатку. Використання Storyboard спрощує управління складними інтерфейсами та допомагає зменшити кількість помилок під час розробки.

UserDefaults буде використовуватися для збереження та управління даними про валюту додатку, цитат відомих персон та параметрів налаштувань. Це простий механізм збереження даних, який підходить для зберігання налаштувань користувача та невеликих обсягів даних. UserDefaults дозволяє зберігати дані у вигляді ключ-значення, що є ефективним для налаштувань та незначних об’єктів. Водночас UserDefaults не підходить для зберігання великих обсягів даних або складних об’єктів, що вимагають високої продуктивності та масштабованості.

Для надсилання мотиваційних повідомлень користувачам буде використано User Notifications Framework. Цей фреймворк дозволяє додатку створювати, планувати та керувати локальними та віддаленими повідомленнями на пристроях користувачів. Використання цього інструменту забезпечить регулярне надходження мотиваційних цитат у встановлені проміжки часу

2.4.2 Алгоритми та методи використані у процесі розробки

У процесі розробки мобільного додатку було використано низку алгоритмів та методів.

Алгоритми для управління цитатами: рандомізація цитат, цей алгоритм рандомізації допомагає забезпечити, що користувачі отримують різноманітні та мотиваційні цитати відомих людей. Він вибирає цитати з бази даних випадковим чином.

Методи для обробки користувацьких даних: управління даними з використанням User Defaults, цей підхід забезпечує надійне зберігання та швидкий доступ до даних користувачів, таких як їх кількість валюти, обрана персона та налаштування параметрів і інші.

Методи для роботи зі сповіщеннями: керування повідомленнями через User Notifications Framework, ці методи дозволяють створювати та керувати розкладом сповіщень, що мотивують користувачів. Вони допомагають забезпечити, що цитати відомих людей надходять вчасно та ефективно мотивують користувачів.

Методи для забезпечення мотивації користувачів:

– гейміфікація досвіду користувача, ці методи дозволяють використовувати ігрові механіки, такі як наприклад гра «Випробуй удачу», для залучення та утримання користувачів; вони стимулюють до досягнення цілей та продовження використання додатку;
– алгоритми нагород за взаємодію користувачів, ці алгоритми допомагають стимулювати активність користувачів, надаючи їм монети за виконання певних дій; вони можуть включати в себе щоденні бонуси, досягнення та інші механізми мотивації;
– аналіз поведінки та персоналізація, цей підхід дозволяє аналізувати поведінку користувачів та надавати поради для досягнення їхніх цілей; він допомагає забезпечити, що мотиваційні сповіщення та завдання відповідають потребам кожного користувача.

Застосовані алгоритми та методи значно підвищують ефективність функціонування додатку, забезпечуючи користувачам постійну мотивацію та заохочення до активності.

3 РОЗРОБКА ДОДАТКУ ТА ТЕСТУВАННЯ

3.1 Детальний опис обраного підходу

Архітектура UserNotification Framework базується на відкритих вебстандартах і складається з низки компонентів (див. рис. 3.1).

Розглянемо, що включають основні компоненти UserNotification Framework.

UNUserNotificationCenter є центральною точкою для керування всіма аспектами локальних та віддалених (сповіщення на відстані) уведомлень. Він дозволяє реєструвати уведомлення, керувати дозволами користувача для отримання уведомлень, встановлювати категорії уведомлень та взаємодіяти зі сповіщеннями під час їх обробки.

UNNotificationContent представляє вміст уведомлення, включаючи заголовок, текст повідомлення, зображення, звук та інші параметри, які можуть бути відображені в уведомленні.

UNNotificationRequest представляє запит на відправлення уведомлення. Він містить в собі вміст уведомлення (UNNotificationContent) та тригер, який визначає, коли відправити уведомлення.

UNNotificationTrigger визначає тригер для відправлення уведомлення. Підкласи включають UNTimeIntervalNotificationTrigger, UNCalendarNotificationTrigger та UNLocationNotificationTrigger.

UNNotificationResponse представляє реакцію користувача на сповіщення. Він містить інформацію про користувача, яка дозволяє додатку реагувати на дії користувача зі сповіщенням, наприклад, відкриття сповіщення або натискання на кнопку в сповіщенні.

UNNotificationSettings дозволяє отримувати та встановлювати налаштування, пов’язані зі сповіщеннями, для конкретного додатку. Використовується для перевірки поточних налаштувань сповіщень та

відображення діалогових вікон налаштувань у додатку для зміни цих параметрів.

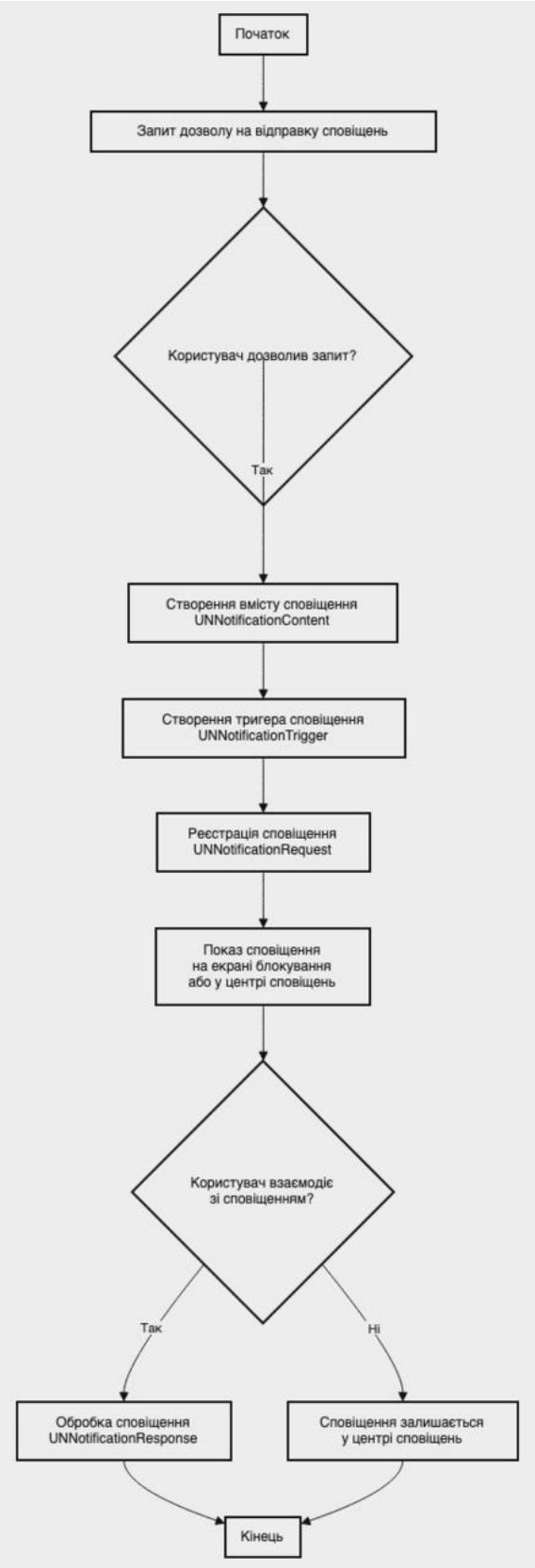


Рисунок 3.1 – Діаграма компонентів UserNotification Framework

3.2 Конфігурація середовища для розробки в Xcode

Xcode – це IDE, створене Apple, яке дає змогу розробляти програми для таких платформ, як iOS, macOS, watchOS та tvOS. Щоб почати використовувати Xcode, необхідно попередньо налаштувати та встановити це середовище розробки для macOS [9].

Для інсталяції Xcode необхідно мати комп'ютер, що працює на операційній системі macOS, оскільки ця програма доступна лише для цієї платформи і не підтримується іншими операційними системами. Процес встановлення Xcode складається з трьох основних етапів.

Першим етапом є перехід до App Store на Mac, де слід скористатися пошуковою панеллю та ввести для пошуку «Xcode».

Другим етапом є натискання на кнопку «Get» або «Install» після того, як Xcode буде знайдений. Це запустить процес завантаження та інсталяції, який може зайняти певний час, оскільки файл Xcode є досить великим.

Третій етап полягає в тому, щоб після завершення інсталяції знайти Xcode у папці «Applications» на Mac. Тепер можна відкрити програму для початку роботи. При першому запуску Xcode може також попросити ввести обліковий запис Apple ID для входу [1].

3.3 Необхідні файли проєкту та їх функції

При створенні iOS-проєкту в Xcode обов'язково повинні бути присутні такі файли: AppDelegate.swift, SceneDelegate.swift та ViewController.swift. Ці файли є основними та завжди необхідні в проєкті Xcode при створенні iOS-додатка.

3.3.1 Файл AppDelegate.swift

Файл AppDelegate в проєкті Xcode є ключовим для управління життєвим циклом додатка. Він виступає в ролі делегата додатка, що дозволяє йому отримувати повідомлення про події, пов’язані з життєвим циклом, і відповідно на них реагувати.

Вміст файлу AppDelegate.swift представлено на рисунку 3.2.

```
import UIKit
@main
class AppDelegate: UIResponder, UIApplicationDelegate {
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
return true
}
func application(_ application: UIApplication, configurationForConnecting connectingSceneSession:
UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
return UISceneConfiguration(name: "Default Configuration", sessionRole: connectingSceneSession.role)
}
func application(_ application: UIApplication, didDiscardSceneSessions sceneSessions: Set<UISceneSession>) {
}
}
```

Рисунок 3.2 – Файл AppDelegate.swift

Далі розглянемо основні функції файлу AppDelegate.

Запуск додатка (Application Launch). Під час запуску додатка AppDelegate отримує повідомлення про подію запуску і може виконувати певні дії, такі як ініціалізація об’єктів або налаштування початкового стану додатка.

Закриття додатка (Application Termination). Коли додаток закривається, AppDelegate отримує повідомлення про цю подію і може здійснити необхідні дії перед завершенням роботи додатка. Це може включати збереження поточного стану або виконання інших фінальних операцій.

Виконання в фоновому режимі (Background Execution). AppDelegate має можливість обробляти повідомлення про запуск додатка, коли він знаходиться в фоновому режимі, або ж перехоплювати повідомлення від пристрою, якщо додаток використовує функцію фонової роботи.

Віддалені повідомлення (Remote Notifications). AppDelegate може обробляти вхідні віддалені повідомлення, наприклад, повідомлення з пуш-сервера, і виконувати відповідні дії, такі як відображення сповіщень або оновлення інтерфейсу користувача.

Відновлення стану (State Restoration). AppDelegate може сприяти відновленню стану додатка після перезавантаження або повернення з фонового режиму.

3.3.2 Файл SceneDelegate.swift

У проєкті Xcode файл SceneDelegate відповідає за керування життєвим циклом сцен (scenes). Він дозволяє додатку одночасно працювати з кількома сценами та контролювати їх поведінку. Кожна сцена має свій набір вікон і відповідних уявлень (views), і SceneDelegate займається управлінням цими елементами [5].

Далі розглянемо основні функції файлу SceneDelegate.

Створення вікон і сцен. SceneDelegate відповідає за створення нових сцен і вікон під час запуску додатка або при отриманні нової сцени.

Зміни стану сцен (обробка). SceneDelegate відслідковує зміни стану сцен, наприклад, коли змінюється їхня активність, видимість або фокус, і виконує відповідні дії. Це може включати оновлення вмісту додатка при переході сцени в активний стан.

View sessions (обробка сесій уявлень). SceneDelegate займається керуванням сесіями уявлень для кожної сцени. Він відслідковує зміни, такі як нові уявлення, зміну орієнтації екрану або сповіщення, і виконує необхідні дії для оновлення інтерфейсу користувача.

Робота з завершенням сцен. Під час закриття сцени або завершення роботи додатка, SceneDelegate виконує необхідні заключні дії, наприклад, зберігає поточний стан додатка або звільняє зайняті ресурси.

Вміст файлу SceneDelegate.swift представлено на рисунку 3.3.

```
import UIKit
class SceneDelegate: UIResponder, UIWindowSceneDelegate {
var window: UIWindow?
func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions:
UIScene.ConnectionOptions) {
guard let _ = (scene as? UIWindowScene) else { return }
}
func sceneDidDisconnect(_ scene: UIScene) {
}
func sceneDidBecomeActive(_ scene: UIScene) {
}
func sceneWillResignActive(_ scene: UIScene) {
}
func sceneWillEnterForeground(_ scene: UIScene) {
}
func sceneDidEnterBackground(_ scene: UIScene) {
}
}
```

Рисунок 3.3 – Файл SceneDelegate.swift

3.3.3 Файл ViewController.swift

Файл ViewController в проєкті Xcode застосовується для керування користувацьким інтерфейсом додатка, згідно з шаблоном архітектури Model-View-Controller (MVC). Він забезпечує взаємодію між моделлю даних програми (Model) та її візуальним представленням (View).

Головна функція ViewController полягає в управлінні подіями та діями, пов’язаними з взаємодією користувача з додатком. Він обробляє введені користувачем дані, виконує необхідні дії і забезпечує відображення відповідного контенту на екрані. Наприклад, він може реагувати на або введення тексту, натискання кнопок або жестові рухи по екрану.

ViewController може виконувати й інші завдання, такі як обробка даних, взаємодія з різними компонентами додатку (наприклад, з API чи базою даних) і управління навігацією між різними екранами додатка.

Вміст файлу ViewController.swift представлено на рисунку 3.4.

```
import UIKit
class ViewController: UIViewController {
override func viewDidLoad() {
super.viewDidLoad()
}
}
```

Рисунок 3.4 – Файл ViewController.swift

3.4 Реалізація основних функціональних модулів

Розділ присвячений реалізації основних функціональних можливостей додатку, які забезпечують інтерактивність, персоналізацію та зручність використання для користувачів. У цьому розділі розглядаються різні механізми для поліпшення користувацького досвіду, що охоплюють як налаштування інтерфейсу, так і впровадження ігрових елементів, таких як монети та зірки [11].

3.4.1 Запит дозволу на сповіщення

У цьому пункті реалізовано функцію для запиту дозволу на надсилання сповіщень користувачу (див. рис. 3.5). Це необхідно для роботи з локальними сповіщеннями, які є важливою частиною функціоналу додатку.

Функція requestNotificationPermission використовує UNUserNotificationCenter для відправлення запиту на доступ до сповіщень. Вона дозволяє додатку отримати право на показ текстових повідомлень, відтворення звуків та оновлення значків на іконці. Результат запиту обробляється у вигляді повідомлень у консолі, що інформують про надання дозволу, відмову або можливі помилки. Виклик функції здійснюється в методі application(_:didFinishLaunchingWithOptions:), який запускається під час старту програми. Це гарантує, що запит на дозвіл буде виконаний одразу після запуску додатку [2].

```
func requestNotificationPermission() {
let center = UNUserNotificationCenter.current()
center.requestAuthorization(options: [.alert, .sound, .badge]) { granted, error in
if let error = error {
print("Error requesting permission: \(error)")
} else if granted {
print("Notification permission granted.")
} else {
print("Notification permission denied.")
}
}
}
@main
class AppDelegate: UIResponder, UIApplicationDelegate {
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
requestNotificationPermission()
return true
}
}
```

Рисунок 3.5 – Код для функції «Запит дозволу на сповіщення»

3.4.2 Персоналізоване привітання користувача

Код реалізує функціонал для персоналізованого привітання користувача на основі введеного імені (див. рис. 3.6).

У методі viewDidLoad перевіряється, чи є збережене ім’я користувача в User Defaults. Якщо ім’я знайдено, воно завантажується, і в елементі userNameLabel відображається привітання у форматі: «Вітаємо, [Ім’я]!». Якщо ім’я не знайдено, викликається функція showWelcomeMessage. Вона показує користувачеві діалогове вікно (UIAlertController) із текстовим полем для введення імені. Після підтвердження введеного імені воно зберігається у UserDefaults, і текст привітання на екрані оновлюється відповідно до введеного значення [12].

```
var userName: String = ""
class MainController: UIViewController {
@IBOutlet weak var userNameLabel: UILabel!
override func viewDidLoad() {
super.viewDidLoad()
if let savedName = UserDefaults.standard.string(forKey: "userName") {
userName = savedName
userNameLabel.text = "Вітаємо, \(userName)!"
} else {
showWelcomeMessage()
}
}
func showWelcomeMessage() {
DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
let alert = UIAlertController(title: "Підкажіть як ваше ім'я?",
message: "Щоб ми могли привітати вас, нам треба знати як вас звати",
preferredStyle: .alert)
alert.addTextField { textField in
textField.placeholder = "Ваше ім'я"
}
let confirmAction = UIAlertAction(title: "Ім'я вказано вірно", style: .default) { _ in
if let name = alert.textFields?.first?.text, !name.isEmpty {
userName = name
UserDefaults.standard.set(name, forKey: "userName")
UserDefaults.standard.synchronize()
self.userNameLabel.text = "Вітаємо, \(userName)!"
}
}
alert.addAction(confirmAction)
self.present(alert, animated: true)
}
}
}
```

Рисунок 3.6 – Код для функції «Персоналізоване привітання користувача»

3.4.3 Інтеграція вебсторінки «Ти як?»

У цьому пункті реалізована інтеграція вебсторінки через WKWebView, що дозволяє відображати зовнішній вебконтент прямо в додатку (див. рис. 3.7).

У методі viewDidLoad створюється об’єкт URL, що вказує на адресу вебсторінки “https://howareu.com/”, і за допомогою URLRequest відправляється запит для завантаження цієї сторінки. Після чого запит завантажується в WKWebView.

```
import UIKit
import WebKit
var startPageURL = "https://howareu.com/"
class MainController: UIViewController {
@IBOutlet weak var webView: WKWebView!
override func viewDidLoad() {
super.viewDidLoad()
let url = URL(string: startPageURL)
let request = URLRequest(url: url!)
webView.load(request)
}
}
```

Рисунок 3.7 – Код для функції «Інтеграція вебсторінки «Ти як?»»

3.4.4 Логіка обміну монет на зірки

Код реалізує функціонал обміну монет на зірки, де користувач може конвертувати 100 монет у 1 зірку, зберігаючи залишок монет (див. рис. 3.8).

Метод convertCoinsToStars перевіряє, чи є у користувача достатньо монет для обміну (мінімум 100 монет). Якщо монет достатньо, обчислюється кількість зірок, яку можна отримати (кількість монет ділимо на 100), і ця кількість додається до загального рахунку зірок. Залишок монет оновлюється за допомогою операції %, що дозволяє зберегти тільки невикористані монети. Також оновлюються лейбли на екрані для монет і зірок, і виводиться повідомлення про успішний обмін. Якщо монет недостатньо, виводиться повідомлення «Монет недостатньо для обміну». За допомогою NotificationCenter оновлюється лічильник зірок в іншій частині додатку [6].

```
@IBAction func convertCoinsToStars(_ sender: UIButton) {
if coins >= 100 {
let starsToAdd = coins / 100
stars += starsToAdd
coins %= 100
NotificationCenter.default.post(name: NSNotification.Name("updateStars"), object: nil, userInfo: ["stars":
stars])=
statusLabel.text = "Успішний обмін"
coinsCounterLabel.text = "Доступні монети: \(coins) 🪙"
starsCounterLabel.text = "Доступні зірки: \(stars) ☆"
} else {
statusLabel.text = "Монет недостатньо для обміну"
}
}
```

Рисунок 3.8 – Код для функції «Логіка обміну монет на зірки»

3.4.5 Функція «Випробуй удачу»

Ця функція дозволяє користувачу грати в гру «Випробуй удачу», де є обмежена кількість спроб для отримання різних призів (див. рис. 3.9).

При натисканні на кнопку запускається функція spinWheel. Якщо спроби ще не закінчилися, вони зменшуються на одну, і генерується випадкове число, яке визначає результат гри. В залежності від числа користувач може виграти монети або додаткову спробу. Якщо користувач виграє монети, їх кількість оновлюється через NotificationCenter, що дозволяє іншим частинам додатку отримувати актуальну інформацію. Якщо спроби закінчилися, користувач побачить повідомлення про відсутність доступних спроб [8].

```
func spinWheel() {
guard attempts > 0 else {
prizeLabel.text = "Спроб більше немає"
return
}
attempts -= 1
attemptsLabel.text = "Залишилось спроб: \(attempts)"
let randomValue = Int.random(in: 0..<100)
switch randomValue {
case 0..<5:
coins += 1000
NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins":
coins])
prizeLabel.text = "Ви виграли 1000 монет!"
case 5..<25:
prizeLabel.text = "Нічого не виграли"
case 25..<40:
attempts += 1
prizeLabel.text = "Ви отримали ще одну спробу!"
attemptsLabel.text = "Залишилось спроб: \(attempts)"
case 40..<50:
coins += 500
NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins":
coins])
prizeLabel.text = "Ви виграли 500 монет!"
case 50..<70:
coins += 250
NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins":
coins])
prizeLabel.text = "Ви виграли 250 монет!"
case 70..<100:
coins += 100
NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins":
coins])
prizeLabel.text = "Ви виграли 100 монет!"
default:
prizeLabel.text = "Помилка"
}
}
```

Рисунок 3.9 – Код для функції «Випробуй удачу»

3.4.6 Завантаження та вибір цитат

Цей розділ коду відповідає за завантаження цитат з текстового файлу та їх вибір в залежності від обраної персони (див. рис. 3.10).

Функція `loadQuotes(from:)` зчитує вміст файлу, розділяючи його на рядки, де кожен містить ім’я персони та цитату. Дані зберігаються в словнику `quotesDictionary`, де ключами є комбінації персон і цитат, а значенням – масив цитат. Вибір цитат здійснюється через три сегменти `UISegmentedControl`. При зміні вибору користувачем викликається метод `segmentChanged(_:)`, який оновлює масив `selectedQuotes`, вибираючи цитати для відповідної комбінації. Якщо цитати для вибраної комбінації не знайдено, виводиться стандартне повідомлення. Також реалізовано оновлення галочок за допомогою функції `updateLights()`, яка змінює прозорість лейблів залежно від кількості доступних цитат, візуально відображаючи прогрес користувача [10].

```
func loadQuotes(from fileName: String) -> [String: [String]] {
var quotes: [String: [String]] = [:]
guard let filePath = Bundle.main.path(forResource: fileName, ofType: "txt") else {
print("File not found")
return quotes
}
do {
let contents = try String(contentsOfFile: filePath, encoding: .utf8)
let lines = contents.split(separator: "\n")
for line in lines {
let parts = line.split(separator: ":", maxSplits: 1)
if parts.count == 2 {
let key = String(parts[0]).trimmingCharacters(in: .whitespaces)
let quote = String(parts[1]).trimmingCharacters(in: .whitespaces)
quotes[key, default: []].append(quote)
}
}
} catch {
print("Error reading file: \(error.localizedDescription)")
}
return quotes
}
@IBAction func segmentChanged(_ sender: UISegmentedControl) {
resetOtherSegments(except: sender)
```

```
if sender == personsSegment1 {
selectedQuotes = quotesDictionary["1-\(sender.selectedSegmentIndex)"] ?? ["Цитат для цієї комбінації немає."]
} else if sender == personsSegment2 {
selectedQuotes = quotesDictionary["2-\(sender.selectedSegmentIndex)"] ?? ["Цитат для цієї комбінації немає."]
} else if sender == personsSegment3 {
selectedQuotes = quotesDictionary["3-\(sender.selectedSegmentIndex)"] ?? ["Цитат для цієї комбінації немає."]
}
}
func updateLights() {
light1Label.alpha = maxQuotes >= 1 ? 1.0 : 0.3
light2Label.alpha = maxQuotes >= 2 ? 1.0 : 0.3
light3Label.alpha = maxQuotes >= 3 ? 1.0 : 0.3
light4Label.alpha = maxQuotes >= 4 ? 1.0 : 0.3
light5Label.alpha = maxQuotes >= 5 ? 1.0 : 0.3
light6Label.alpha = maxQuotes >= 6 ? 1.0 : 0.3
light7Label.alpha = maxQuotes >= 7 ? 1.0 : 0.3
light8Label.alpha = maxQuotes >= 8 ? 1.0 : 0.3
}
```

Рисунок 3.10 – Код для функції «Завантаження та вибір цитат»

3.4.7 Налаштування параметрів цитат

Цей код дозволяє користувачеві налаштувати час отримання цитат та інтервал між ними (див. рис. 3.11).

Для вибору дати та часу використовуються два елементи UIDatePicker – один для вибору дати, а інший для часу. Коли користувач натискає кнопку для збереження налаштувань, обрані значення комбінуються в одну нову дату (startTime), що визначає час отримання сповіщень. Інтервал між цитатами налаштовується за допомогою елемента UIStepper. Користувач може вибирати інтервал з кроком 1 година, починаючи з 1 години. Вибране значення відображається в текстовому полі intervalTextField. Ця функціональність дає користувачу можливість точно налаштувати час та інтервал для отримання цитат.

```
import UIKit
var startTime: Date = Calendar.current.date(bySettingHour: 17, minute: 14, second: 0, of: Date())!
var components: DateComponents?
var timeComponents: DateComponents?
var calendar = Calendar.current
class DateTimeController: UIViewController {
@IBOutlet weak var datePicker: UIDatePicker!
@IBOutlet weak var timePicker: UIDatePicker!
@IBOutlet weak var intervalTextField: UITextField!
@IBOutlet weak var intervalStepper: UIStepper!
override func viewDidLoad() {
super.viewDidLoad()
timePicker.timeZone = TimeZone(identifier: "Europe/Kiev")
intervalStepper.minimumValue = 1
intervalStepper.stepValue = 1
intervalStepper.value = interval
}
@IBAction func stepperChanged(_ sender: UIStepper) {
intervalTextField.text = "\(Int(sender.value))"
}
@IBAction func setDateTimeButton(_ sender: UIButton) {
let selectedDate = datePicker.date
let selectedTime = timePicker.date
components = Calendar.current.dateComponents([.year, .month, .day], from: selectedDate)
timeComponents = Calendar.current.dateComponents([.hour, .minute, .second], from: selectedTime)
if let components = components, let timeComponents = timeComponents {
startTime = Calendar.current.date(bySettingHour: timeComponents.hour ?? 0,
minute: timeComponents.minute ?? 0,
second: timeComponents.second ?? 0,
of: Calendar.current.date(from: components)!)!
}
interval = intervalStepper.value
print(interval)
}
}
```

Рисунок 3.11 – Код для функції «Налаштування параметрів цитат»

3.4.8 Реалізація надсилання цитат

Цей код відповідає за планування та надсилання сповіщень з цитатами (див. рис. 3.12).

Метод scheduleNotifications() очищає всі попередні сповіщення і планує нові, використовуючи UNMutableNotificationContent для створення контенту сповіщення, де задаються заголовок і випадкова цитата з масиву selectedQuotes. Час відправлення визначається через UNCalendarNotificationTrigger з інтервалом, вказаним у змінній interval, і додається до системи через UNUserNotificationCenter.current().add(request) [3].

```
private func scheduleNotifications() {
UNUserNotificationCenter.current().removeAllPendingNotificationRequests()
guard !selectedQuotes.isEmpty else { return }
for i in 0..<maxQuotes {
let content = UNMutableNotificationContent()
content.title = "Отримай порцію мотивації!"
content.body = selectedQuotes.randomElement() ?? "Цитата не знайдена."
let triggerTime = startTime.addingTimeInterval(TimeInterval(i) * interval)
let triggerDateComponents = Calendar.current.dateComponents([.year, .month, .day, .hour, .minute, .second],
from: triggerTime)
let trigger = UNCalendarNotificationTrigger(dateMatching: triggerDateComponents, repeats: false)
let request = UNNotificationRequest(identifier: "quote\(i)", content: content, trigger: trigger)
UNUserNotificationCenter.current().add(request) { error in
if let error = error {
print("Error adding notification: \(error.localizedDescription)")
}
}
}
}
```

Рисунок 3.12 – Код для функції «Реалізація надсилання цитат»

3.4.9 Реалізація кнопок для керування різними функціями додатку

У коді реалізовано кілька функцій для взаємодії користувача з додатком через кнопки (див. рис. 3.13).

Кнопка «Перевірити оновлення» порівнює поточну версію додатку з останньою доступною версією і показує відповідне повідомлення. Кнопка «Увімкнути сповіщення» відкриває налаштування додатку для управління дозволами на сповіщення через UIApplication.openSettingsURLString. Кнопка «Про додаток» відкриває UIAlertController з інформацією про додаток, такою як поточна версія і розробник. Користувач може закрити вікно натисканням кнопки «Добре». Кнопки «Повідомити про помилку» та «Залишити відгук» відкривають UIAlertController з текстовими полями для вводу, де користувач може ввести опис помилки чи відгук, який потім виводиться в консоль. Іконки соціальних мереж (Facebook, Instagram, Twitter) налаштовуються для обробки жестів натискання, що відкриває відповідні вебсайти за допомогою методу UIApplication.shared.open(url) [4].

```
let facebookTapGesture = UITapGestureRecognizer(target: self, action: #selector(facebookTapped))
facebookImageView.addGestureRecognizer(facebookTapGesture)
@objc func facebookTapped() {
if let url = URL(string: "https://www.facebook.com") {
UIApplication.shared.open(url)
}
}
@IBAction func aboutAppButtonPress(_ sender: UIBarButtonItem) {
let alertController = UIAlertController(title: "Про додаток", message: "Версія додатку 1.2.0\n\n Розробник:
Nasonov Vladyslav", preferredStyle: .alert)
let dismissAction = UIAlertAction(title: "Добре", style: .default, handler: nil)
alertController.addAction(dismissAction)
self.present(alertController, animated: true, completion: nil)
}
@IBAction func reportBugButtonPress(_ sender: UIBarButtonItem) {
let alertController = UIAlertController(title: "Опишіть помилку", message: "Докладно опишіть вашу
помилку, яку ви виявили", preferredStyle: .alert)
alertController.addTextField { (textField) in
textField.placeholder = "Опис помилки"
}
let sendAction = UIAlertAction(title: "Надіслати", style: .default) { _ in
if let errorMessage = alertController.textFields?.first?.text {
print("Помилка: \(errorMessage)")
}
}
alertController.addAction(sendAction)
self.present(alertController, animated: true, completion: nil)
}
@IBAction func leaveFeedbackButtonPress(_ sender: UIBarButtonItem) {
let alertController = UIAlertController(title: "Напишіть ваш відгук", message: "Дякуємо вам за покращення
нашого додатку", preferredStyle: .alert)
alertController.addTextField { (textField) in
textField.placeholder = "Ваш відгук"
}
let sendAction = UIAlertAction(title: "Надіслати", style: .default) { _ in
if let feedback = alertController.textFields?.first?.text {
print("Відгук: \(feedback)")
}
}
alertController.addAction(sendAction)
self.present(alertController, animated: true, completion: nil)
}
@IBAction func checkUpdateButtonPress(_ sender: UIBarButtonItem) {
let alertController: UIAlertController
if currentVersion == latestVersion {
alertController = UIAlertController(title: "Додаток оновлено до останньої версії", message: nil, preferredStyle:
.alert)
} else {
alertController = UIAlertController(title: "Доступне оновлення. Будь ласка, оновіть додаток до останньої
версії", message: nil, preferredStyle: .alert)
}
let closeAction = UIAlertAction(title: "Закрити", style: .default, handler: nil)
alertController.addAction(closeAction)
self.present(alertController, animated: true, completion: nil)
}
@IBAction func enableNotificationsButtonPress(_ sender: UIBarButtonItem) {
if let appSettingsURL = URL(string: UIApplication.openSettingsURLString) {
if UIApplication.shared.canOpenURL(appSettingsURL) {
UIApplication.shared.open(appSettingsURL, options: [:], completionHandler: nil)
}
}
}
```

Рисунок 3.13 – Код для керування різними функціями додатку

3.5 Запуск та тестування додатку на практиці

Додаток розроблявся для мобільної платформи iOS, тому тестування буде здійснюватися за допомогою емулятора, вбудованого у середовище розробки Xcode (див. рис. 3.14).

Xcode надає потужний інструмент для тестування програм на віртуальних iOS-пристроях, що дозволяє перевіряти роботу додатку без необхідності використання реальних пристроїв. Завдяки емулятору можна здійснювати тестування на різних моделях iPhone і iPad, а також на різних версіях операційної системи iOS, що є важливим аспектом для забезпечення сумісності додатку.

Емулятор Xcode дозволяє втілювати функції, які імітують реальні умови використання додатку, включаючи роботу з мережею, камерами, мікрофонами та іншими можливостями пристроїв. Це дає змогу перевіряти, як додаток взаємодіятиме з користувачем, зокрема, як будуть працювати сповіщення та інтеграція з різними функціями смартфона, такими як GPS, сенсори руху, а також мультимедійні функції.

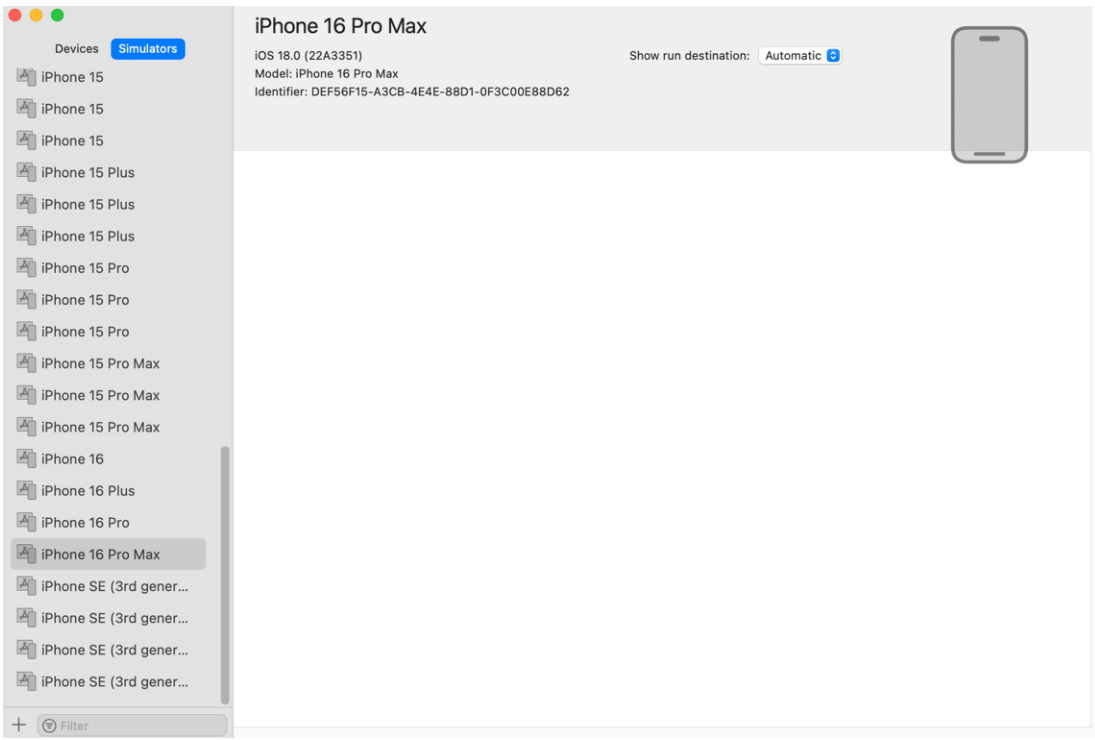


Рисунок 3.14 – Вікно вибору емулятору у Xcode

При відкритті додатка для iOS реалізовано екран завантаження, який відображається на початковому етапі завантаження програми. Це рішення дозволяє користувачам зрозуміти, що додаток обробляє необхідні ресурси і готується до роботи, запобігаючи можливому відчуттю затримки або непрацездатності [7]. На рисунку 3.15 представлений екран завантаження додатку.

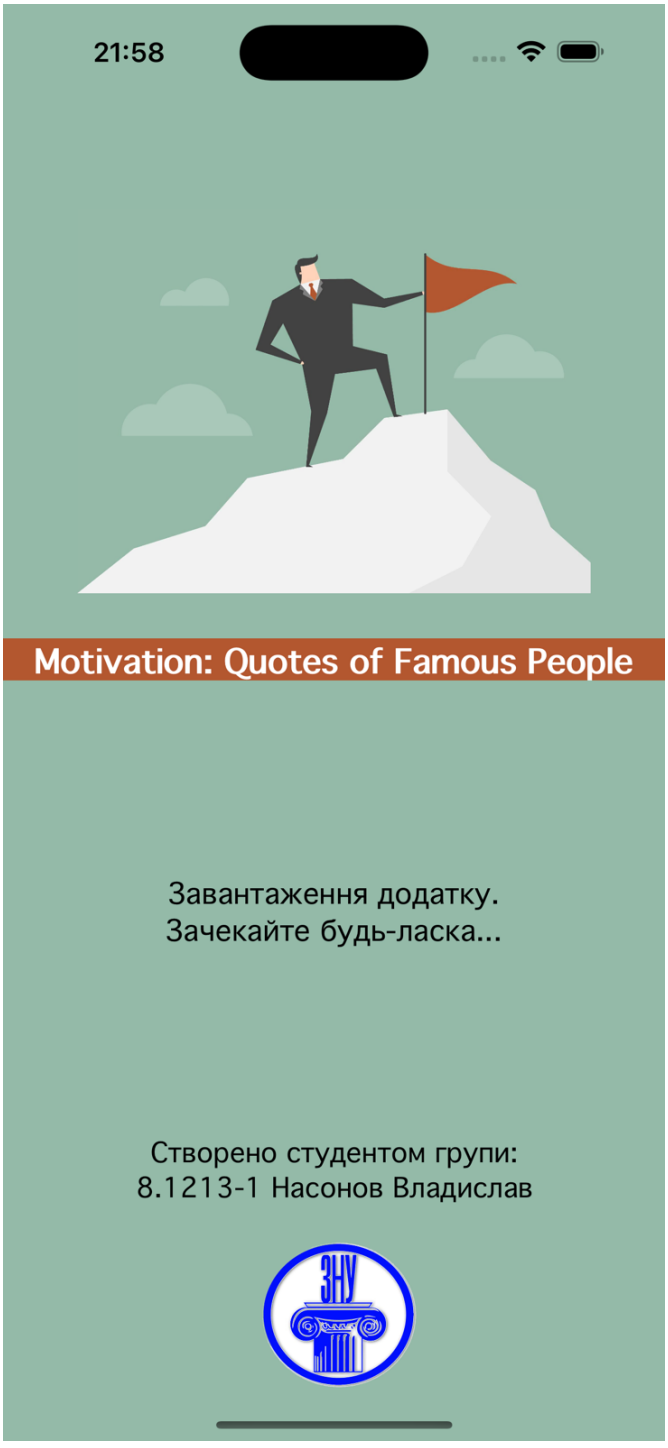


Рисунок 3.15 – Екран завантаження додатку

Після завантаження додатку користувач одразу отримує запит на дозвіл на отримання сповіщень (див. рис. 3.16). Це є критично важливим для коректної роботи додатку. Оскільки основною функцією програми є надання користувачам мотиваційних цитат у вигляді сповіщень у заданий час, без цієї можливості додаток не зможе виконати свою основну задачу. Запит на дозвіл надсилається на самому початку використання додатку.

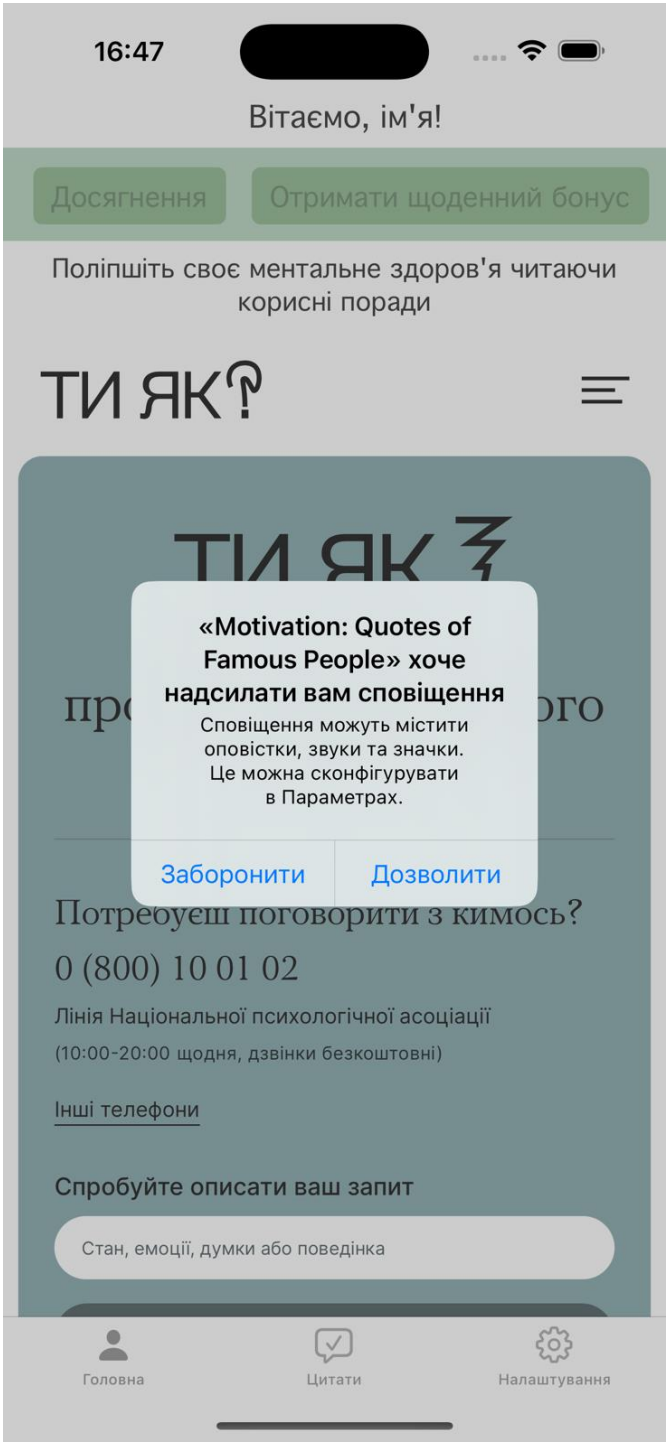


Рисунок 3.16 – Запит на дозвіл сповіщень

Також додаток запитує ім’я користувача, щоб при кожному наступному запуску привітати його персоналізованим повідомленням, створюючи відчуття індивідуального підходу та приємної взаємодії (див. рис. 3.17).

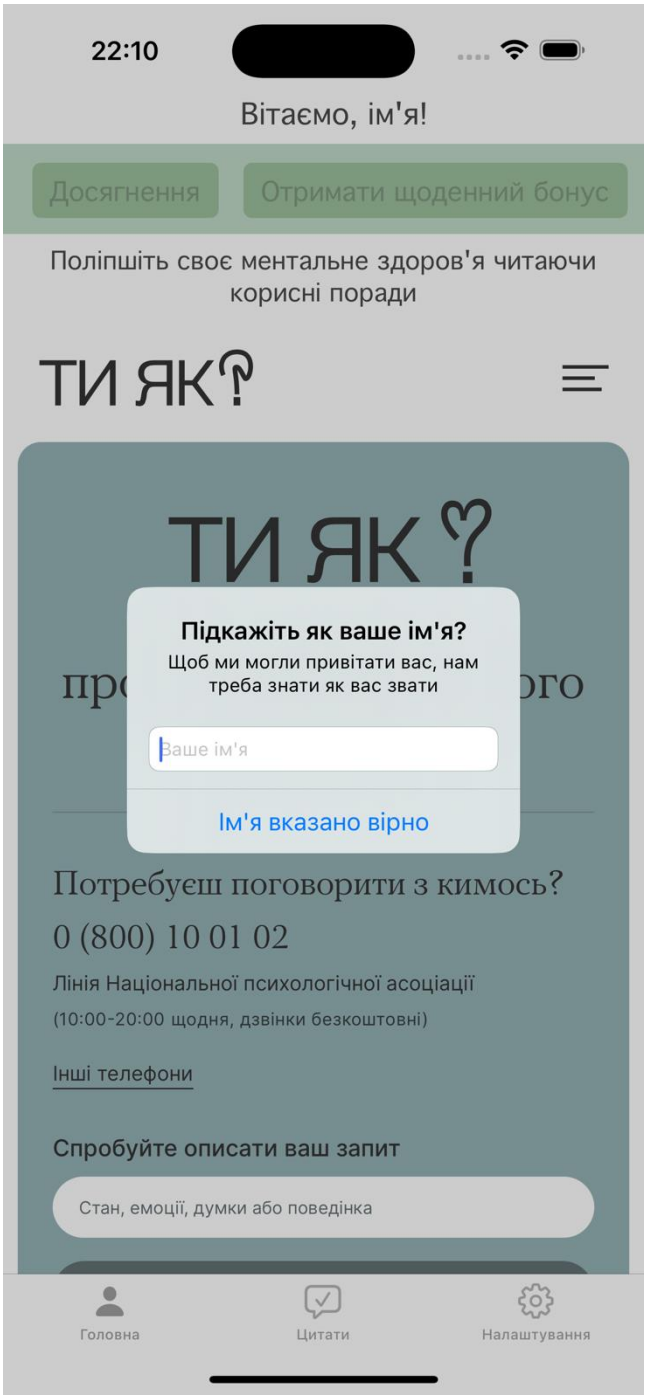


Рисунок 3.17 – Запит на ім’я користувача

Після запуску додаток готовий до використання. На головній сторінці користувач бачить привітання з його ім’ям, а також дві кнопки: для переходу

на сторінку досягнень і для отримання щоденного бонусу за вхід у додаток. Нижче розміщено вбудовану вебсторінку Всеукраїнської програми ментального здоров'я «Ти як?» за ініціативою Олени Зеленської. Цей сайт надає доступ до корисних статей і дозволяє, при необхідності, отримати ментальну підтримку від професіоналів. На рисунку 3.18 представлена повністю завантажена головна сторінка додатку.

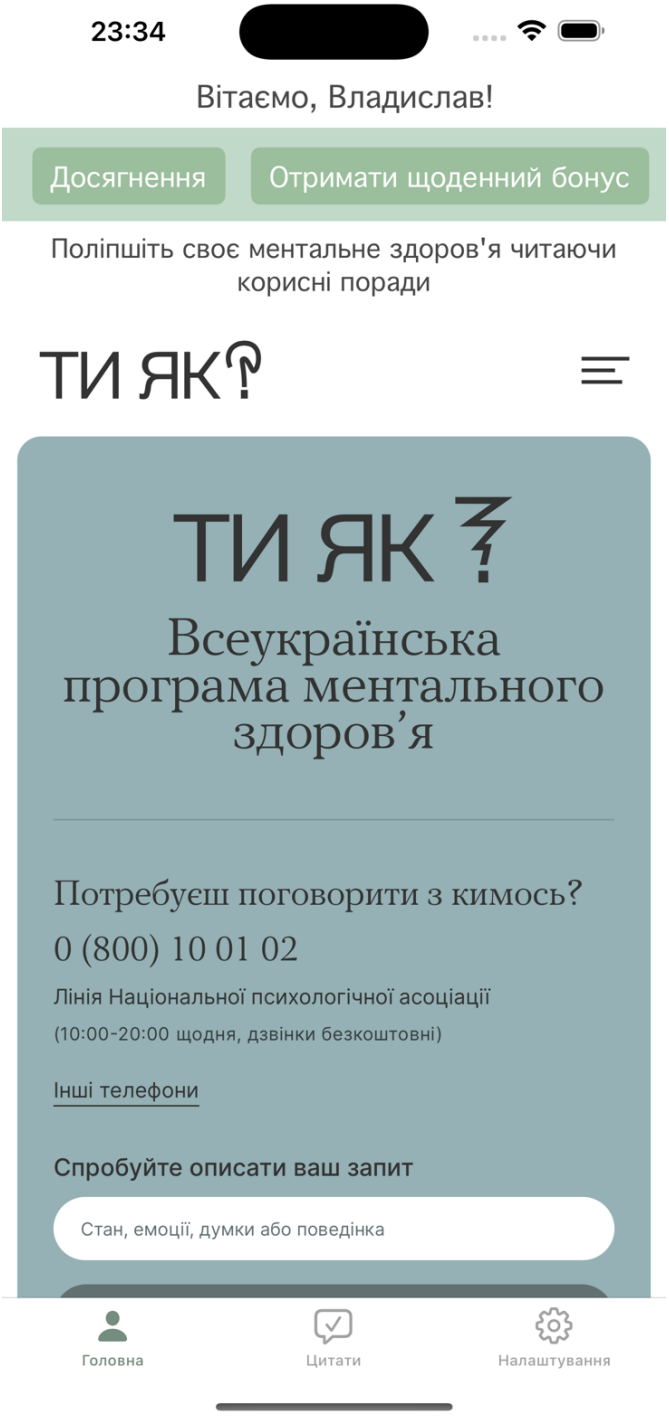


Рисунок 3.18 – Екран головної сторінки додатку

При натисканні на кнопку «Отримати щоденний бонус» користувачу нараховується невеликий бонус у валюті додатку.

При натисканні на кнопку «Досягнення» відкривається сторінка, де відображаються досягнення (див. рис. 3.19). Коли користувач виконає умови певного досягнення, з’явиться кнопка, що підтверджує його отримання, а також користувачу буде нарахована відповідна винагорода.

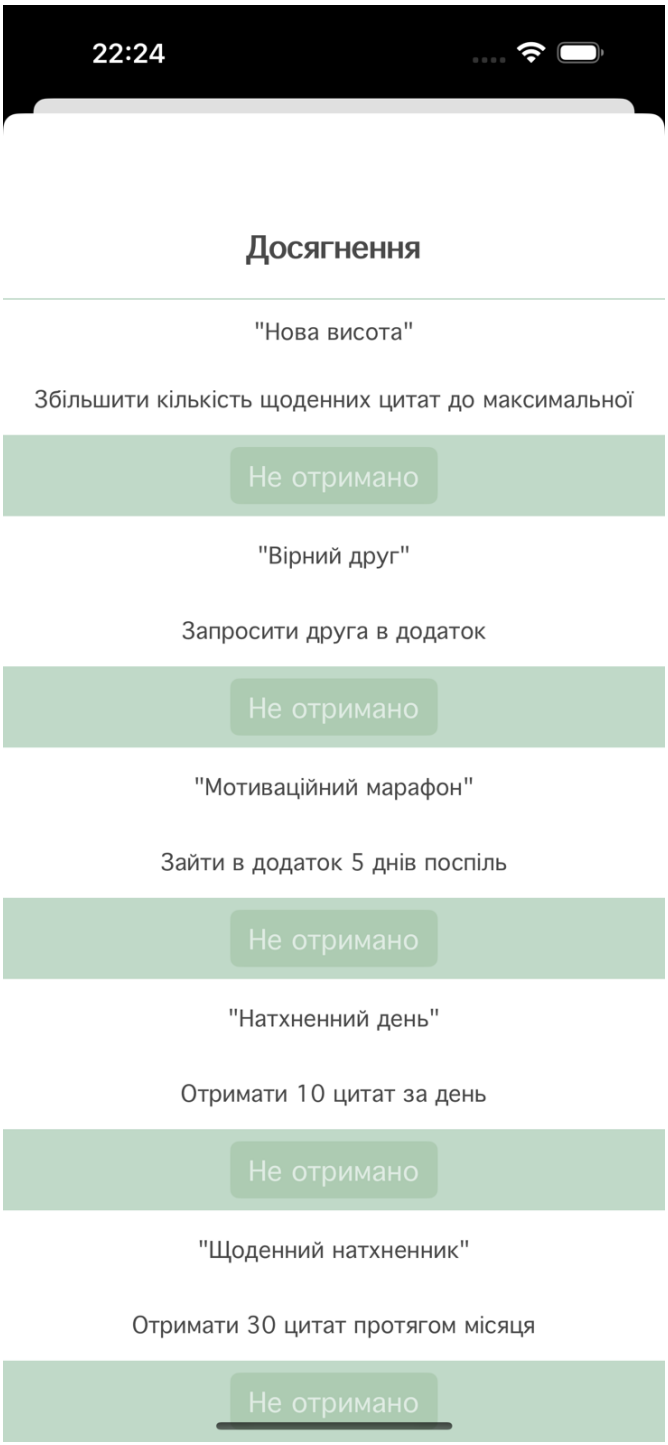


Рисунок 3.19 – Екран сторінки досягнень додатку

Якщо повернутися на головну сторінку й обрати 2-й розділ, відкриється сторінка цитат (див. рис. 3.20).



Рисунок 3.20 – Екран сторінки цитат додатку

У верхній частині сторінки відображається кількість доступних цитат на сьогодні – це кількість повідомлень з цитатами, які додаток може надіслати користувачу протягом одного дня. Кількість показана у вигляді емодзі

галочок. Нижче можна побачити преміум-валюту додатку – зірки, які використовуються для розблокування додаткових можливостей (основна валюта – монети). На рисунку 3.21 представлений екран сторінки цитат додатку з максимальними покращеннями.



Рисунок 3.21 – Екран сторінки цитат додатку з максимальними покращеннями

У мобільному додатку «Мотивація: цитати відомих людей» користувачі мають змогу повною мірою персоналізувати свій досвід, вибираючи цитати від відомих особистостей, що відповідають їхнім уподобанням і потребам. Кожен користувач може обрати одну з трьох доступних безкоштовно персон. Персона, яку обирає користувач, стає основним джерелом мотивації, надаючи цитати саме від цієї особистості. Для доступу до повного списку з дев’яти персон, що пропонуються в додатку, користувачеві необхідно накопичити зірки. Зірки можна заробити через активність в додатку, зокрема за виконання досягнень або за щоденне використання додатку. Це додає додатковий мотиваційний елемент, заохочуючи користувачів бути більш активними.

Інтерфейс додатку забезпечує зручний вибір та підтвердження персони через кнопку «Задати», яка дозволяє закріпити обрану особистість як основне джерело цитат. Кожне накопичення певної кількості зірок відкриває нові три особи для вибору, що дозволяє користувачам змінювати джерело натхнення в залежності від їхнього настрою, ситуації чи особистих цілей на той чи інший день. Це забезпечує гнучкість і варіативність у підході до мотивації, дозволяючи користувачам вибирати саме ті цитати, які найбільше відповідають їхньому внутрішньому стану або допомагають на шляху до досягнення певних цілей.

Окрім вибору персон, додаток надає можливість збільшити кількість цитат, які користувач може отримувати щодня. За замовчуванням доступно три цитати на день, однак користувач може збільшити цю кількість до восьми, використовуючи зірки, зароблені за активність в додатку. Цей механізм дозволяє більш активно налаштовувати рівень мотивації, отримуючи більше натхнення протягом дня. Такий підхід не тільки дає змогу налаштувати кількість цитат, але й дає користувачу більше контролю над його мотиваційним процесом. Вартість покращень з часом зростає, що стимулює користувачів до активності в додатку, а також заохочує їх до постійного використання, щоб накопичувати зірки для подальших налаштувань. Наприкінці сторінки розміщено кнопку для переходу до налаштувань цитат.

При натисканні на кнопку «Отримати зірки» користувач переходить на сторінку магазину монет (див. рис. 3.22). На цій сторінці користувач має можливість обміняти монети на зірки, що є необхідним для доступу до нових функцій. Для здійснення обміну достатньо натискати на відповідну кнопку. Окрім цього, на сторінці є кнопка «Випробуй удачу», при натисканні на яку користувача перенаправляє на іншу сторінку додатку

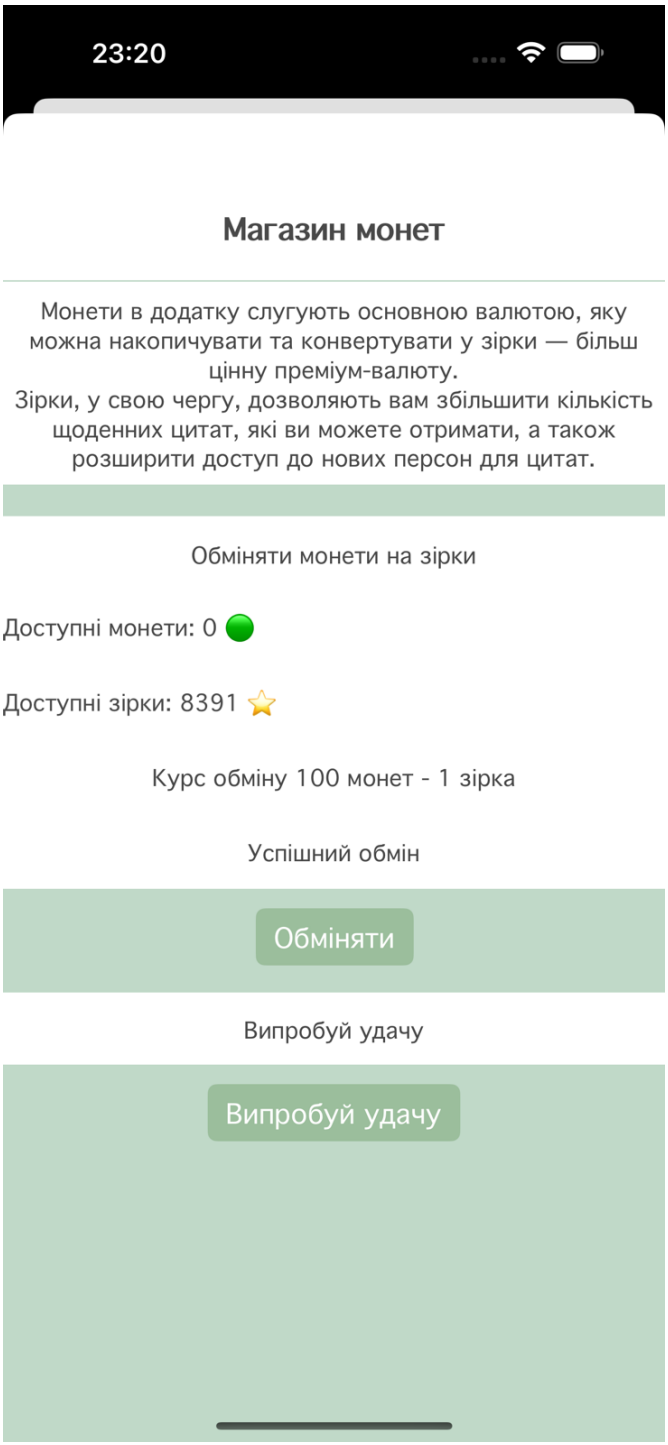


Рисунок 3.22 – Екран сторінки магазину монет додатку

На сторінці «Випробуй удачу» є лише одна кнопка «Грати». Після натискання на неї користувач отримує шанс виграти один з шести призів, кожен з яких має свій шанс на виграш. Кожного дня користувач може зробити до двох спроб, щоб погратися і спробувати виграти приз. Це обмеження дозволяє заохочувати регулярне використання додатку. На рисунку 3.23 представлений екран сторінки випробування удачі додатку.

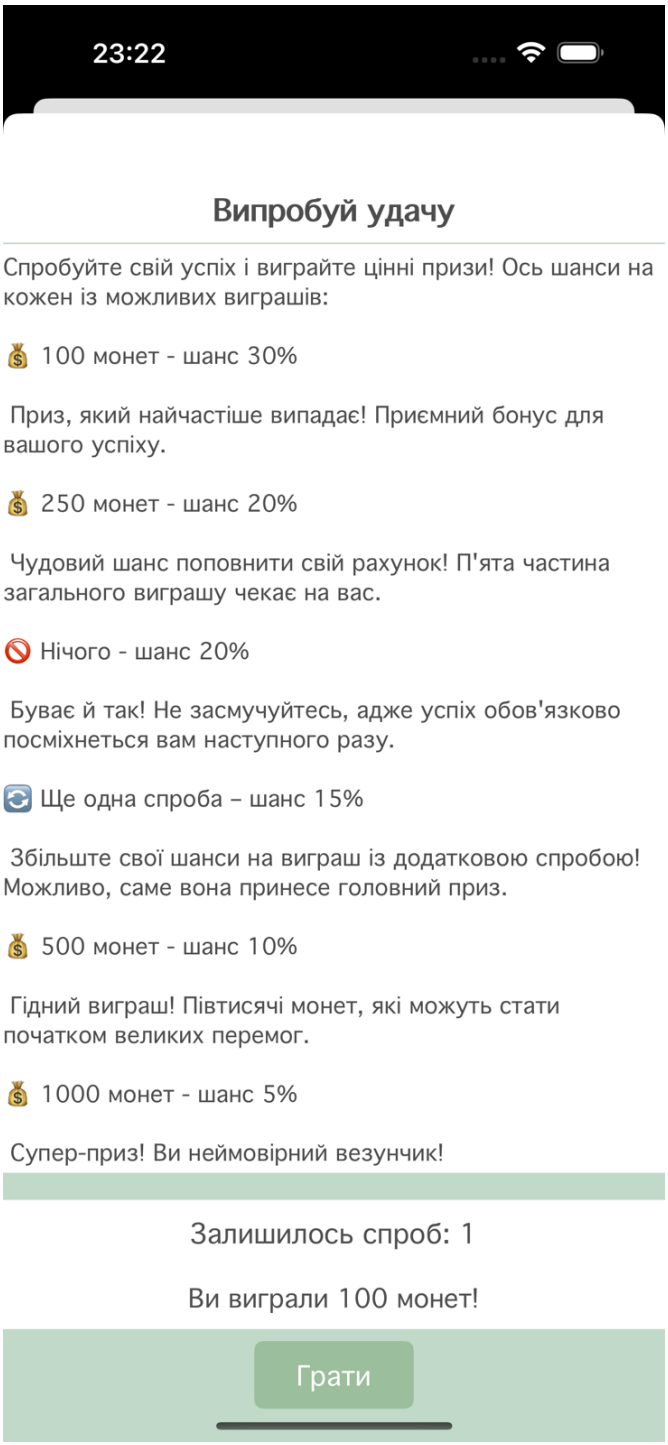


Рисунок 3.23 – Екран сторінки випробування удачі додатку

Якщо повернутися назад у 2-й розділ, то при натисканні на кнопку «Змінити параметри цитат» користувач переходить на сторінку налаштувань цитат (див. рис. 3.24). На цій сторінці користувач може вибрати параметри для отримання сповіщень. Перше налаштування дозволяє обрати стартову дату, з якої додаток почне надсилати цитати. Другий параметр – це початковий час для надсилання. Третє налаштування – це інтервал між кожною цитатою. Коли усі зміні внесені, потрібно натиснути на кнопку для збереження.

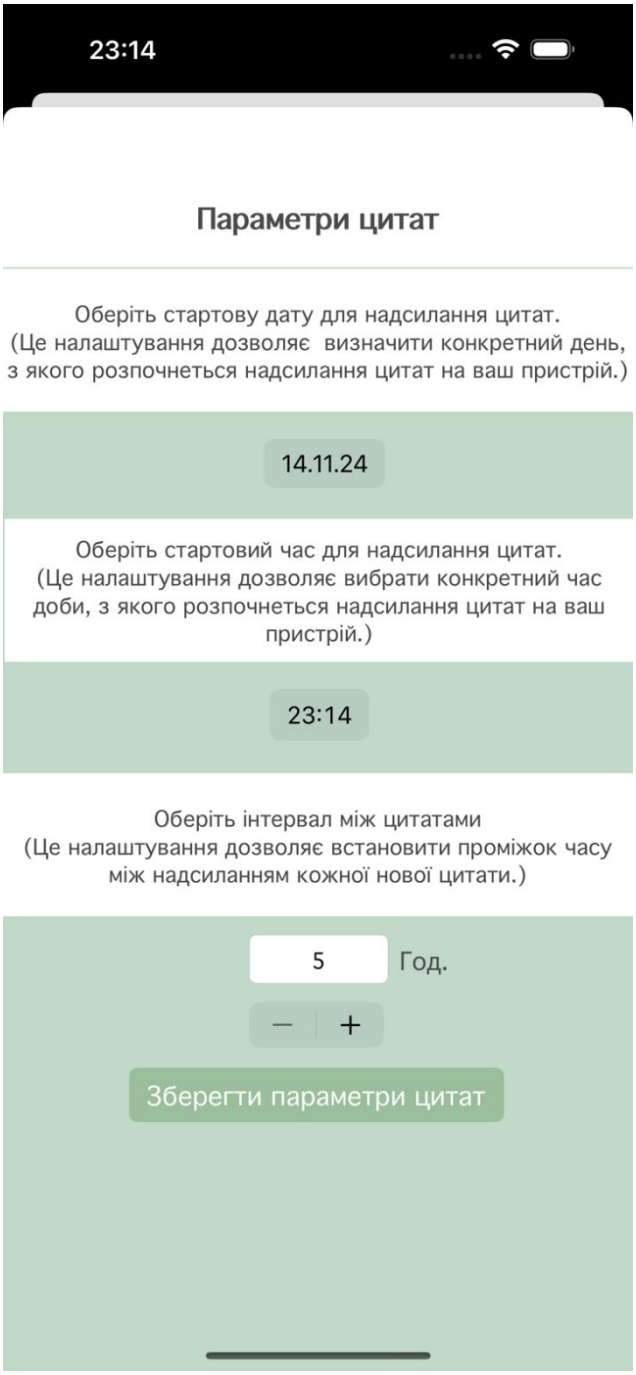


Рисунок 3.24 – Екран сторінки параметрів цитат

Після налаштування параметрів цитат користувач може повернутися в розділ цитат, де може обрати персону та натиснути кнопку «Задати». Після цього можна очікувати на сповіщення з цитатою. Сповіщення надійде навіть якщо користувач перебуває в іншому додатку або вимкне екран телефону, оскільки додаток працює у фоновому режимі і забезпечує отримання сповіщень у будь-який час (див. рис. 3.25).

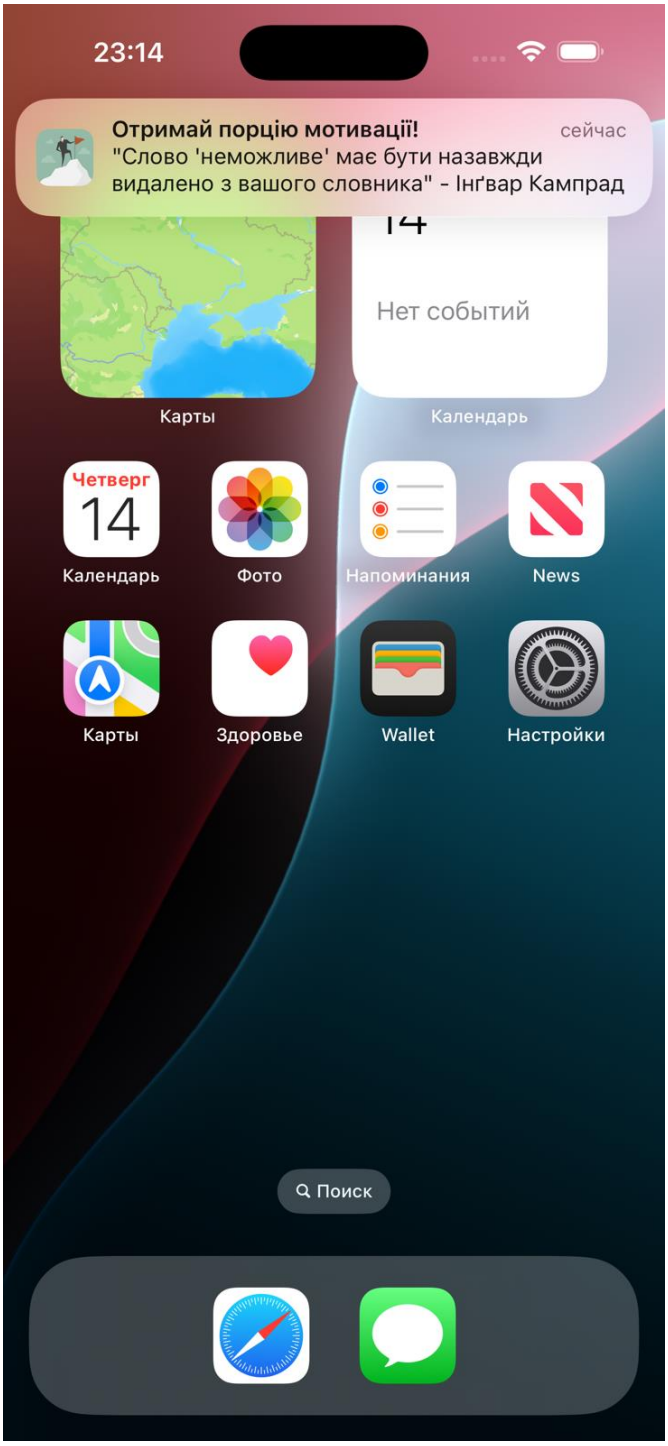


Рисунок 3.25 – Сповіщення надіслане від додатку

Останній, третій розділ додатку – «Налаштування» – призначений для керування різними параметрами та функціями додатку (див. рис. 3.26). Тут користувач може знайти кілька важливих опцій для налаштування роботи додатку відповідно до своїх потреб. На початку сторінки розміщений журнал змін (changelog), який інформує про нові функції, виправлення помилок і вдосконалення в останній версії додатку.

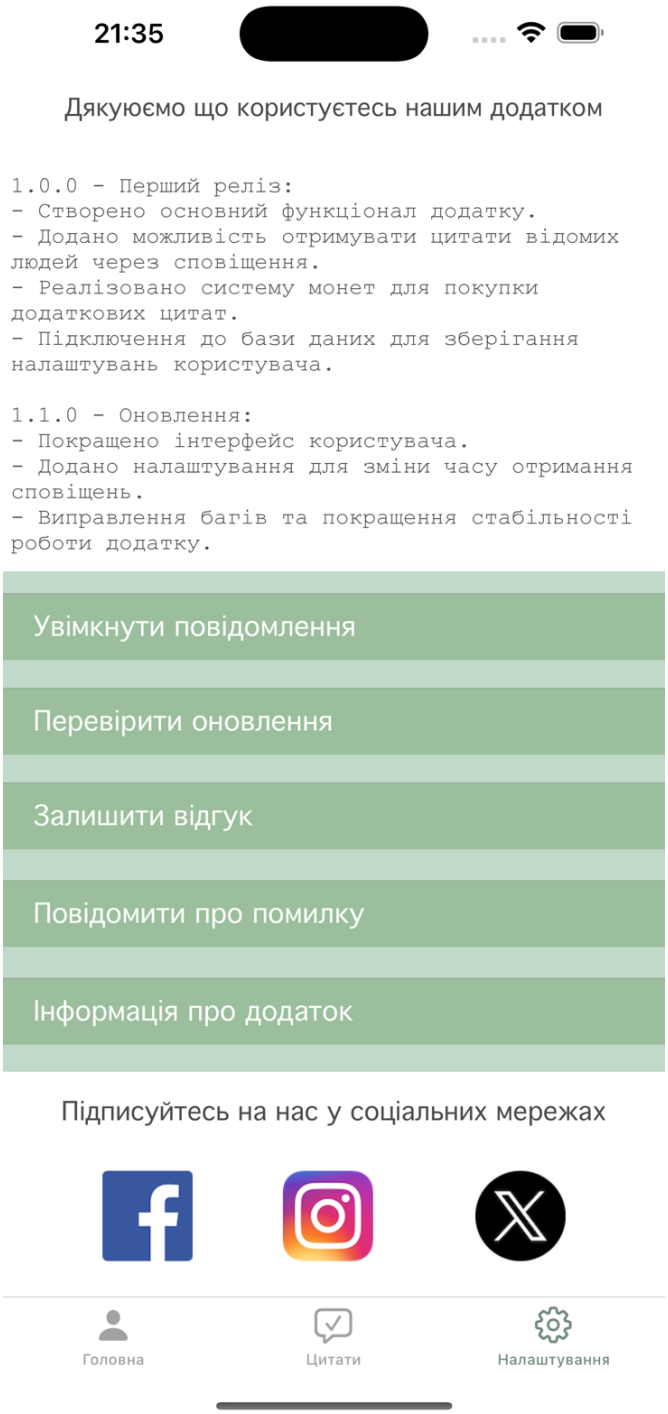


Рисунок 3.26 – Екран сторінки налаштувань додатку

Далі йде ряд кнопок, що дозволяють налаштувати додаток. Кнопка «Увімкнути повідомлення» відкриває системні налаштування iPhone для активації сповіщень, якщо користувач не зробив цього раніше (див. рис. 3.27). Це забезпечує можливість користувачеві швидко увімкнути сповіщення. Якщо сповіщення не були увімкнуті раніше, ця кнопка автоматично перенаправляє на відповідний розділ налаштувань, де користувач може дозволити додатку надсилати йому повідомлення.

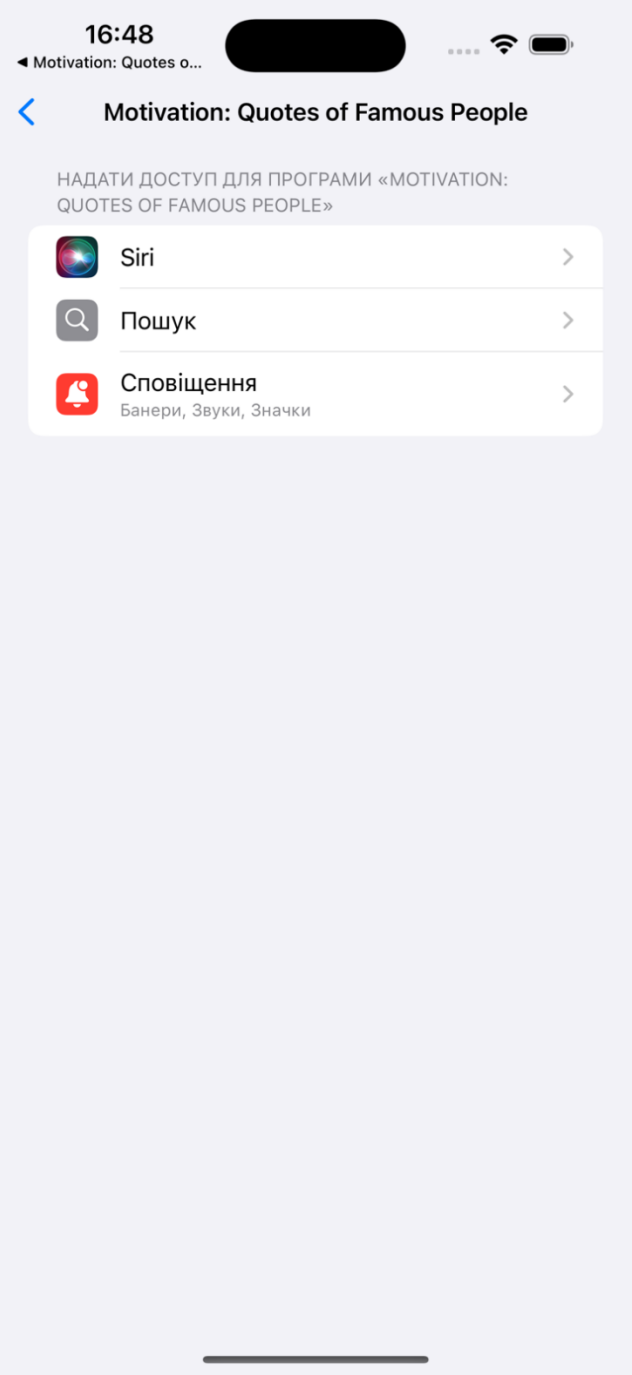


Рисунок 3.27 – Системні налаштування iPhone для додатку

Кнопка «Перевірити оновлення» дозволяє перевірити актуальність версії додатку. У разі наявності нової версії користувач буде сповіщений про доступне оновлення (див. рис. 3.28). Це дає змогу користувачеві бути в курсі останніх змін та покращень, що забезпечує кращий досвід використання додатку. Якщо нове оновлення доступне, програма інформує користувача, і той може вирішити, чи хоче він встановити його для покращення роботи додатку.

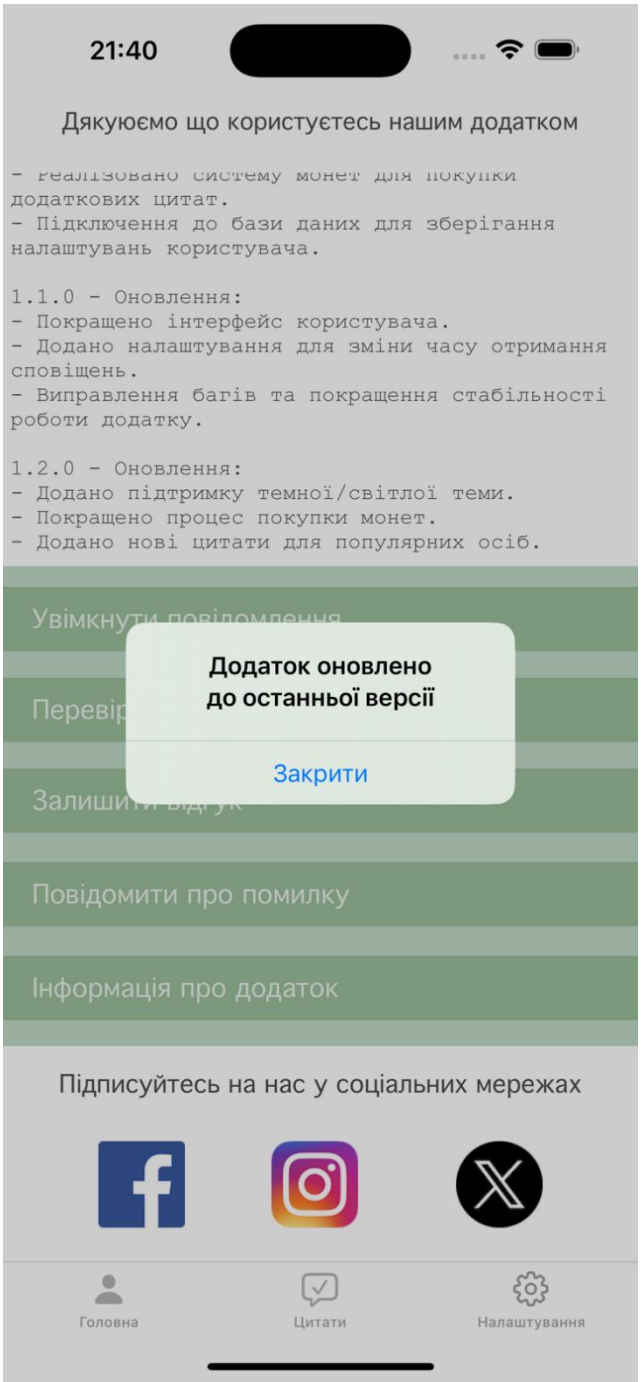


Рисунок 3.28 – Функція для перевірки актуальності версії додатку

За допомогою кнопки «Залишити відгук» користувач може написати відгук, в якому він має можливість похвалити додаток за його функціональність та зручність або, навпаки, висловити критику щодо його недоліків (див. рис. 3.29). Ця опція дозволяє користувачам ділитися своїми враженнями, що може бути корисним для розробників при покращенні програми.

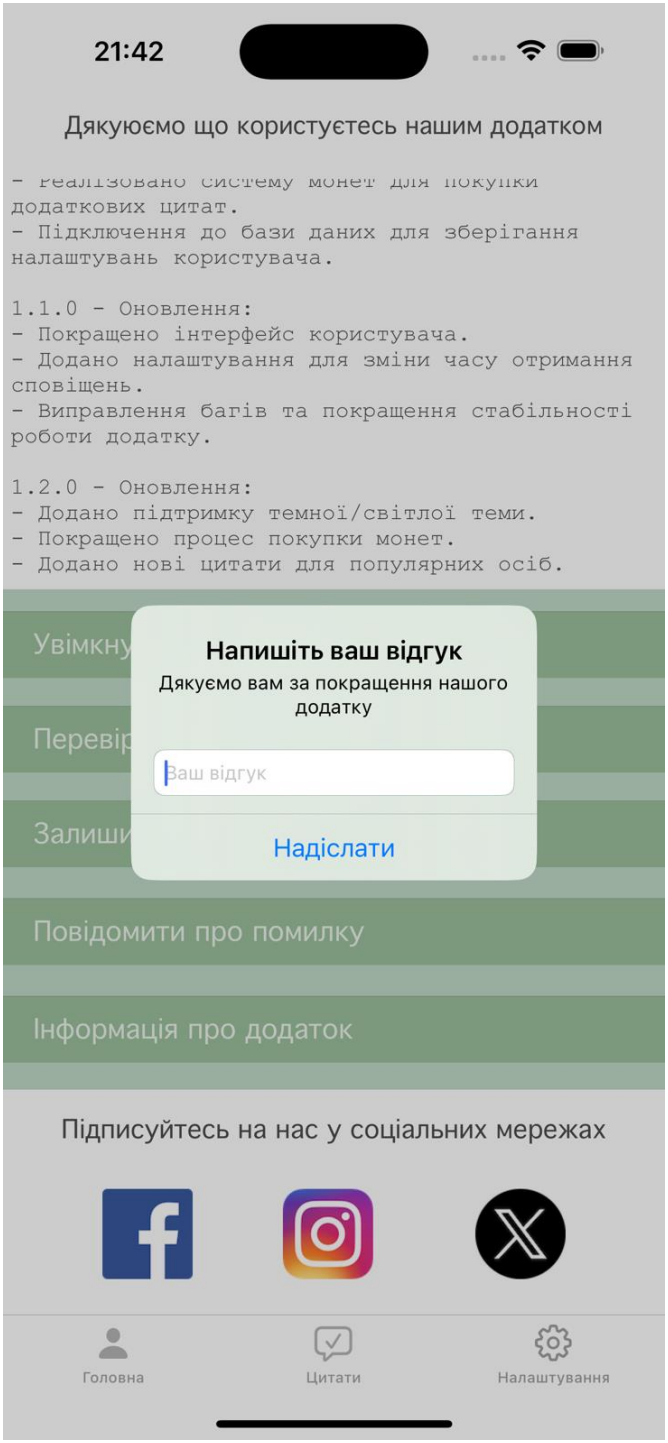


Рисунок 3.29 – Функція для написання відгуку про додаток

Наступна кнопка «Повідомити про помилку» дозволяє надіслати зворотний зв’язок розробникам щодо виявлених помилок (див. рис. 3.30). За допомогою цієї функції користувач може швидко повідомити про будь-які баги чи технічні проблеми, з якими він зіткнувся під час використання додатку. Це дозволяє розробникам оперативно реагувати на виявлені проблеми, покращувати стабільність і функціональність програми.

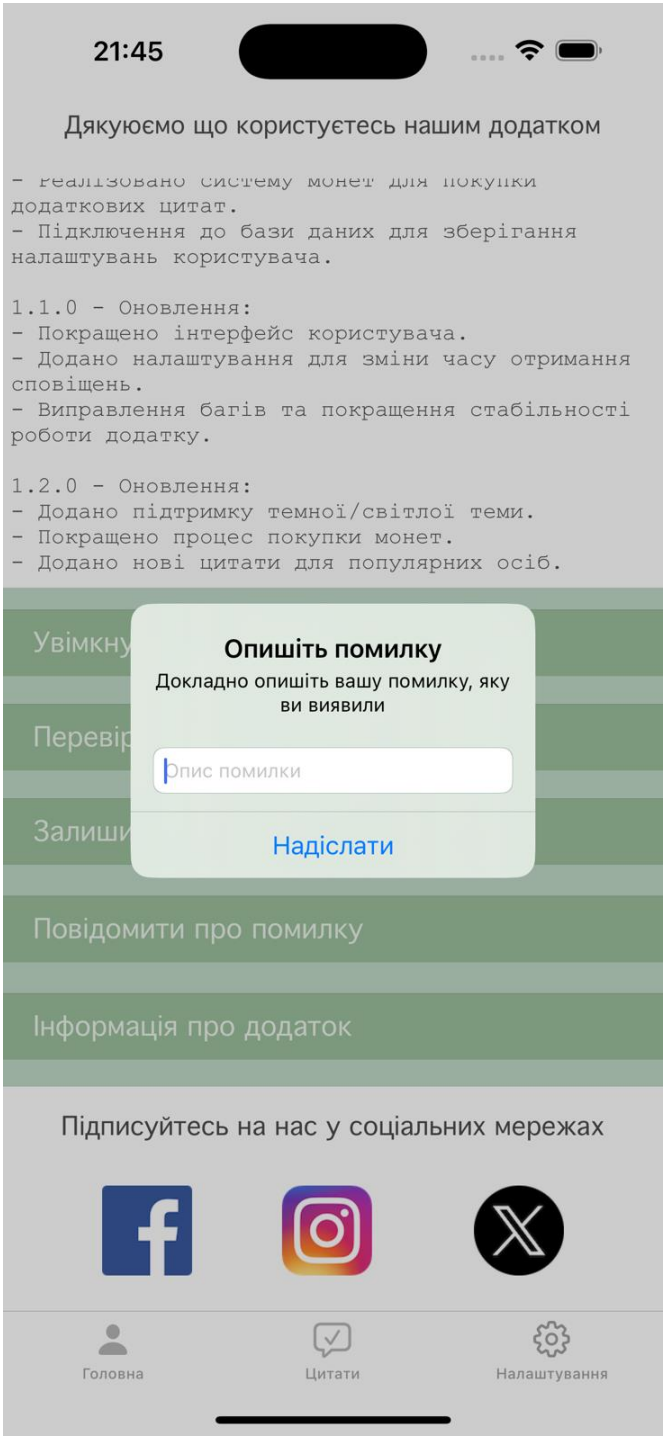


Рисунок 3.30 – Функція для написання помилки виявленої у додатку

Остання кнопка «Інформація про додаток» відкриває вікно з даними про версію додатку та іменем розробника (див. рис. 3.31). Це дозволяє користувачу дізнатися, яка версія додатку наразі встановлена на його пристрої, а також отримати контактну інформацію про розробника, що може бути корисним для звернення з питаннями чи пропозиціями. Цей розділ також може містити інші важливі відомості, такі як наприклад ліцензійні умови.

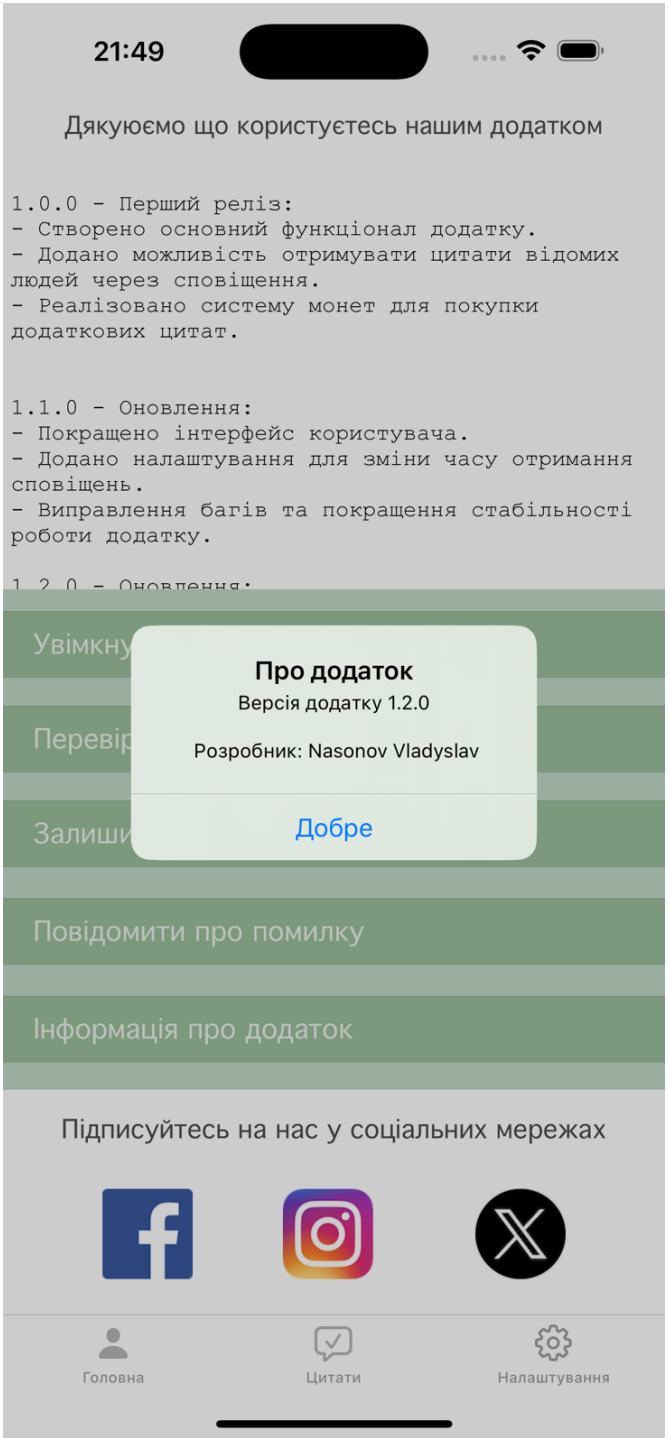


Рисунок 3.31 – Функція з даними про додаток

У кінці сторінки, під усіма кнопками, розташовано три іконки, які символізують популярні соціальні мережі. При натисканні на будь-яку з іконок користувач буде перенаправлений на відповідну особисту сторінку розробника (див. рис. 3.32). Це дозволить слідкувати за процесом розробки, отримувати актуальну інформацію про оновлення та спілкуватися з розробником у режимі реального часу.

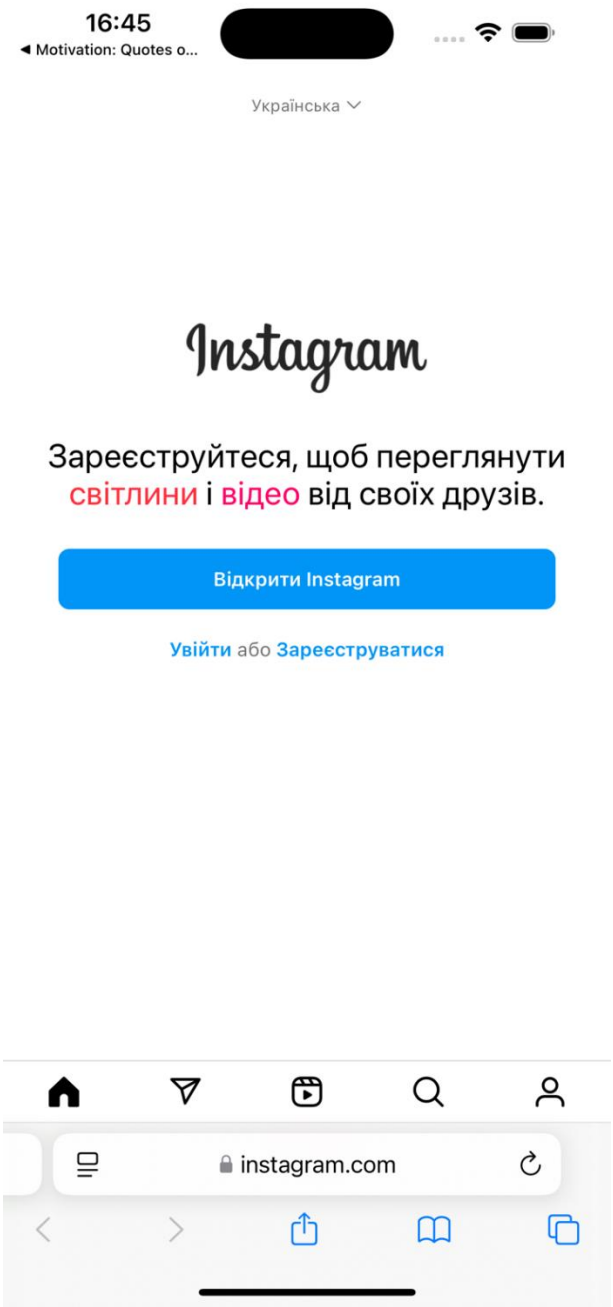


Рисунок 3.32 – Функція з соціальною сторінкою розробника додатку

ВИСНОВКИ

В результаті виконання дипломної роботи на тему «Розробка мобільного додатку «Мотивація: цитати відомих людей» на платформі iOS» було розроблено і впроваджено повноцінний мобільний додаток. Це дає користувачам можливість отримувати мотиваційні цитати від відомих людей, підвищуючи особисту мотивацію та саморозвиток. Створений додаток – це інструмент, який поєднує корисний вміст з елементами гейміфікації, що може зацікавити користувачів та заохотити їх регулярно користуватися продуктом.

Основною метою цієї дипломної роботи було створення мобільного додатку для платформи iOS, що може стати потужним мотиватором для користувачів. У додатку реалізована система персоналізованих цитат відомих осіб, які надходять користувачу у вигляді сповіщень у заданий проміжок часу. Це дозволяє користувачам отримувати натхнення на протязі дня і бути мотивованими досягати своїх цілей. Додаток дає змогу вибирати різних персон і отримувати їх цитати, що додає варіативності та персоналізації контенту.

Інтеграція гейміфікаційних елементів у вигляді монет і зірок є важливою частиною цього додатку, оскільки вона не лише мотивує користувачів бути більш активними, але й заохочує їх до щоденного використання додатку. Користувачі можуть отримувати монети за різні досягнення, наприклад, за щоденне використання додатку або виконання певних досягнень. Потім ці монети можна обміняти на зірки, що дозволяє розблокувати нових персон і збільшити кількість цитат, які додаток може надсилати користувачу щодня. Такий підхід дає можливість створити елемент змагання і стимулює користувачів до досягнення поставлених цілей.

Робота над додатком була здійснена за допомогою сучасних інструментів і технологій, зокрема мови програмування Swift і середовища розробки Xcode. Це дозволило реалізувати високу продуктивність,

стабільність роботи додатку та інтуїтивно зрозумілий користувацький інтерфейс. Для створення інтерфейсу було використано Storyboard, що дозволяє візуально розташувати елементи на екрані, забезпечуючи зручність взаємодії з додатком. Користувачі можуть легко налаштувати періодичність отримання цитат, вибирати особистості, від яких хочуть отримувати цитати, а також переглядати свої досягнення та баланс монет і зірок.

Для зберігання даних, таких як налаштування користувача, досягнення та накопичені монети і зірки, використовувався механізм UserDefaults. Це забезпечує збереження всіх важливих даних навіть після закриття додатку або перезавантаження пристрою, що підвищує зручність користування і надійність роботи додатку. Крім того, для надсилання сповіщень про нові цитати використовувався User Notifications Framework. Це дозволяє автоматично інформувати користувачів в обраний час, що сприяє регулярному залученню до додатку і підтримці мотивації.

В ході розробки додатку були враховані всі вимоги до мобільних застосунків для платформи iOS, зокрема зручність та інтуїтивно зрозумілий інтерфейс. Користувачі можуть безперешкодно налаштовувати інтервали отримання сповіщень, взаємодіяти з різними аспектами додатку та отримувати приємні візуальні й звукові ефекти при виконанні досягнень. Це дозволяє створити додаток, що має не лише функціональну, але й емоційну цінність для користувачів.

У результаті розробки додатку було досягнуто основної мети – створено інструмент, який може бути корисним для різних категорій людей, що шукають натхнення, мотивацію для досягнення особистих і професійних цілей. Подальша робота над удосконаленням додатку може включати додавання нових функцій, таких як підтримка декількох мов, інтеграція з соціальними мережами, можливість створення власних списків цитат або адаптація додатку для людей з особливими потребами. Ці доповнення сприятимуть розширенню аудиторії та підвищенню популярності додатку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Apple. The Swift Programming Language. Cupertino : «Apple Inc.», 2023. 600 p.
2. Framework WebKit. *Apple Developer*. URL: https://developer.apple.com/documentation/usernotifications (дата звернення: 27.07.2024).
3. Tidwell J. Designing Interfaces: Patterns for Effective Interaction Design. Beijing : «O'Reilly Media», 2011. 464 p.
4. Hoffman J. S. Effective Swift: 52 Specific Ways to Improve Your iOS, Mac, and Swift Code. New York : «Addison-Wesley», 2016. 344 p.
5. Hart J. iOS Design Patterns. San Francisco : «Addison-Wesley», 2015. 320 p.
6. Gary P. Pro Swift. Apress : «Pro Series», 2017. 459 p.
7. Using Storyboards. *Apple Developer*. URL: https://developer.apple.com/documentation/uikit/uistoryboard (дата звернення: 03.08.2024).
8. Li K. K. iOS Programming: The Essentials. 2nd ed. New York : «Addison-Wesley», 2015. 480 p.
9. Vincent S. Architecting iOS Applications. London : «Packt Publishing», 2016. 301 p.
10. Ward K. Functional Swift. Birmingham : «Packt Publishing», 2017. 374 p.
11. Designing for iOS. *Apple Developer*. URL: https://developer.apple.com/design/human-interface-guidelines/designing-for-ios (дата звернення: 22.09.2024).
12. User Defaults Overview. *Apple Developer*. URL: https://developer.apple.com/documentation/foundation/userdefaults (дата звернення: 14.10.2024).

ДОДАТОК А

MainController.swift

```
import UIKit
import WebKit
var startPageURL = "https://howareu.com/"
var userName: String = ""
class MainController: UIViewController {
    @IBOutlet weak var userNameLabel: UILabel!
    @IBOutlet weak var webView: WKWebView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let url = URL(string: startPageURL)
        let request = URLRequest(url: url!)
        webView.load(request)
        if let savedName = UserDefaults.standard.string(forKey: "userName") {
            userName = savedName
            userNameLabel.text = "Вітаємо, \(userName)!"
        } else {
            showWelcomeMessage()
        }
    }
    func showWelcomeMessage() {
        DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
            let alert = UIAlertController(title: "Підкажіть як ваше ім'я?", message: "Щоб ми
могли привітати вас, нам треба знати як вас звати", preferredStyle: .alert)
            alert.addTextField { textField in
                textField.placeholder = "Ваше ім'я"
            }
            let confirmAction = UIAlertAction(title: "Ім'я вказано вірно", style: .default) { _
in
                if let name = alert.textFields?.first?.text, !name.isEmpty {
                    userName = name
```

```
                UserDefaults.standard.set(name, forKey: "userName")
                UserDefaults.standard.synchronize()
                self.userNameLabel.text = "Вітаємо, \(userName)!"
            }
        }
        alert.addAction(confirmAction)
        self.present(alert, animated: true)
    }
  }
}
```

ДОДАТОК Б

QuotesController.swift

```
import UIKit
import Foundation
var maxQuotes: Int = 3
var quotesPersonCount: Int = 1
var upgradeCost: Int = 10
var upgradeCost2: Int = 100
var interval: TimeInterval = 10
var selectedQuotes: [String] = []
var quotesDictionary: [String: [String]] = [:]
class QuotesController: UIViewController, UNUserNotificationCenterDelegate {
    @IBOutlet weak var personsSegment1: UISegmentedControl!
    @IBOutlet weak var personsSegment2: UISegmentedControl!
    @IBOutlet weak var personsSegment3: UISegmentedControl!
    @IBOutlet weak var light1Label: UILabel!
    @IBOutlet weak var light2Label: UILabel!
    @IBOutlet weak var light3Label: UILabel!
    @IBOutlet weak var light4Label: UILabel!
    @IBOutlet weak var light5Label: UILabel!
    @IBOutlet weak var light6Label: UILabel!
    @IBOutlet weak var light7Label: UILabel!
    @IBOutlet weak var light8Label: UILabel!
    @IBOutlet weak var starsCounterLabel: UILabel!
    @IBOutlet weak var statusLabel: UILabel!
    @IBOutlet weak var statusLabel2: UILabel!
    override func viewDidLoad() {
        super.viewDidLoad()
        UNUserNotificationCenter.current().delegate = self
        quotesDictionary = loadQuotes(from: "quotes")
        CustomStyles.applyCustomStyle(to: personsSegment1)
        CustomStyles.applyCustomStyle(to: personsSegment2)
```

```
        CustomStyles.applyCustomStyle(to: personsSegment3)
        starsCounterLabel.text = "Доступні зірки: \(stars) ☆"
        NotificationCenter.default.addObserver(self, selector:
#selector(updateStarsLabel(_:)), name: NSNotification.Name("updateStars"), object: nil)
    }
    @objc func updateStarsLabel(_ notification: Notification) {
        if let stars = notification.userInfo?["stars"] as? Int {
            starsCounterLabel.text = "Доступні зірки: \(stars) ☆"
        }
    }
    deinit {
        NotificationCenter.default.removeObserver(self, name:
NSNotification.Name("updateCoins"), object: nil)
    }
    func updateLights() {
        light1Label.alpha = maxQuotes >= 1 ? 1.0 : 0.3
        light2Label.alpha = maxQuotes >= 2 ? 1.0 : 0.3
        light3Label.alpha = maxQuotes >= 3 ? 1.0 : 0.3
        light4Label.alpha = maxQuotes >= 4 ? 1.0 : 0.3
        light5Label.alpha = maxQuotes >= 5 ? 1.0 : 0.3
        light6Label.alpha = maxQuotes >= 6 ? 1.0 : 0.3
        light7Label.alpha = maxQuotes >= 7 ? 1.0 : 0.3
        light8Label.alpha = maxQuotes >= 8 ? 1.0 : 0.3
    }
    func loadQuotes(from fileName: String) -> [String: [String]] {
        var quotes: [String: [String]] = [:]
        guard let filePath = Bundle.main.path(forResource: fileName, ofType: "txt") else {
            print("File not found")
            return quotes
        }
        do {
            let contents = try String(contentsOfFile: filePath, encoding: .utf8)
            let lines = contents.split(separator: "\n")
            for line in lines {
                let parts = line.split(separator: ":", maxSplits: 1)
```

```
                if parts.count == 2 {
                    let key = String(parts[0]).trimmingCharacters(in: .whitespaces)
                    let quote = String(parts[1]).trimmingCharacters(in: .whitespaces)
                    quotes[key, default: []].append(quote)
                }
            }
        } catch {
            print("Error reading file: \(error.localizedDescription)")
        }
        return quotes
    }
    @IBAction func segmentChanged(_ sender: UISegmentedControl) {
        resetOtherSegments(except: sender)
        if sender == personsSegment1 {
            selectedQuotes = quotesDictionary["1-\(sender.selectedSegmentIndex)"] ??
["Цитат для цієї комбінації немає."]
        } else if sender == personsSegment2 {
            selectedQuotes = quotesDictionary["2-\(sender.selectedSegmentIndex)"] ??
["Цитат для цієї комбінації немає."]
        } else if sender == personsSegment3 {
            selectedQuotes = quotesDictionary["3-\(sender.selectedSegmentIndex)"] ??
["Цитат для цієї комбінації немає."]
        }
    }
    private func resetOtherSegments(except selectedSegment: UISegmentedControl) {
        [personsSegment1, personsSegment2, personsSegment3].forEach {
            if $0 != selectedSegment {
                $0.selectedSegmentIndex = UISegmentedControl.noSegment
            }
        }
    }
    func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive response:
UNNotificationResponse, withCompletionHandler completionHandler: @escaping () -> Void) {
        print("Notification received: \(response.notification.request.content.body)")
        completionHandler()
```

```
    }
    var lastNotificationTime: Date?
    func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification: UNNotification, withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {
        if let lastTime = lastNotificationTime, Date().timeIntervalSince(lastTime) < 1 {
            completionHandler([.alert, .sound])
            return
        }
        lastNotificationTime = Date()
        maxQuotes -= 1
        print("Notification presented: \(notification.request.content.body)")
        updateLights()
        completionHandler([.alert, .sound])
    }
    private func scheduleNotifications() {
        UNUserNotificationCenter.current().removeAllPendingNotificationRequests()
        guard !selectedQuotes.isEmpty else { return }
        for i in 0..<maxQuotes {
            let content = UNMutableNotificationContent()
            content.title = "Отримай порцію мотивації!"
            content.body = selectedQuotes.randomElement() ?? "Цитата не знайдена."
            let triggerTime = startTime.addingTimeInterval(TimeInterval(i) * interval)
            let triggerDateComponents = Calendar.current.dateComponents([.year, .month, .day, .hour, .minute, .second], from: triggerTime)
            let trigger = UNCalendarNotificationTrigger(dateMatching: triggerDateComponents, repeats: false)
            let request = UNNotificationRequest(identifier: "quote\(i)", content: content, trigger: trigger)
            UNUserNotificationCenter.current().add(request) { error in
                if let error = error {
                    print("Error adding notification: \(error.localizedDescription)")
                }
            }
        }
```

```
    }
    @IBAction func setPersonButtonPress(_ sender: Any) {
        scheduleNotifications()
    }
    @IBAction func upgradeQuotesMaxButtonPress(_ sender: UIButton) {
        if maxQuotes >= 8 {
            statusLabel.text = "Досягнуто максимальний рівень покращень"
        } else if stars >= upgradeCost {
            stars -= upgradeCost
            maxQuotes += 1
            upgradeCost = Int(Double(upgradeCost) * 3)
            statusLabel.text = "Успішно покращено. Наступне покращення коштує: \(upgradeCost)"
            print("\(maxQuotes)")
            starsCounterLabel.text = "Доступні зірки: \(stars) ☆"
            if light4Label.isHidden {
                light4Label.isHidden = false
            } else if light5Label.isHidden {
                light5Label.isHidden = false
            } else if light6Label.isHidden {
                light6Label.isHidden = false
            } else if light7Label.isHidden {
                light7Label.isHidden = false
            } else if light8Label.isHidden {
                light8Label.isHidden = false
                statusLabel.text = "Досягнуто максимальний рівень покращень"
            }
        } else {
            statusLabel.text = "Недостатньо зірок для покращення. Наступне покращення коштує: \(upgradeCost2)"
        }
    }
    @IBAction func upgradeQuotesPersonCountButtonPress(_ sender: UIButton) {
        if quotesPersonCount >= 3 {
            statusLabel2.text = "Досягнуто максимальний рівень покращень"
```

```
            } else if stars >= upgradeCost2 {
                stars -= upgradeCost2
                quotesPersonCount += 1
                upgradeCost2 = Int(Double(upgradeCost2) * 3)
                statusLabel2.text = "Успішно покращено. Наступне покращення коштує: \(upgradeCost2)"
                starsCounterLabel.text = "Доступні зірки: \(stars) ☆"
                if personsSegment2.isHidden {
                    personsSegment2.isHidden = false
                } else if personsSegment3.isHidden {
                    personsSegment3.isHidden = false
                    statusLabel2.text = "Досягнуто максимальний рівень покращень"
                }
            } else {
                statusLabel.text = "Недостатньо зірок для покращення. Наступне покращення коштує: \(upgradeCost2)"
            }
        }
    }
```

ДОДАТОК В

SettingsController.swift

```
import UIKit
let currentVersion = "1.0"
let latestVersion = "1.0"
class SettingsController: UIViewController {
    @IBOutlet weak var facebookImageView: UIImageView!
    @IBOutlet weak var instagramImageView: UIImageView!
    @IBOutlet weak var twitterImageView: UIImageView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let facebookTapGesture = UITapGestureRecognizer(target: self, action: #selector(facebookTapped))
        facebookImageView.isUserInteractionEnabled = true
        facebookImageView.addGestureRecognizer(facebookTapGesture)
        let instagramTapGesture = UITapGestureRecognizer(target: self, action: #selector(instagramTapped))
        instagramImageView.isUserInteractionEnabled = true
        instagramImageView.addGestureRecognizer(instagramTapGesture)
        let twitterTapGesture = UITapGestureRecognizer(target: self, action: #selector(twitterTapped))
        twitterImageView.isUserInteractionEnabled = true
        twitterImageView.addGestureRecognizer(twitterTapGesture)
    }
    @objc func facebookTapped() {
        if let url = URL(string: "https://www.facebook.com") {
            UIApplication.shared.open(url)
        }
    }
    @objc func instagramTapped() {
        if let url = URL(string: "https://www.instagram.com") {
            UIApplication.shared.open(url)
```

```
        }
    }
    @objc func twitterTapped() {
        if let url = URL(string: "https://x.com") {
            UIApplication.shared.open(url)
        }
    }
    @IBAction func aboutAppButtonPress(_ sender: UIBarButtonItem) {
        let alertController = UIAlertController(title: "Про додаток", message: "Версія додатку 1.2.0\n\n Розробник: Nasonov Vladyslav", preferredStyle: .alert)
        let dismissAction = UIAlertAction(title: "Добре", style: .default, handler: nil)
        alertController.addAction(dismissAction)
        self.present(alertController, animated: true, completion: nil)
    }
    @IBAction func reportBugButtonPress(_ sender: UIBarButtonItem) {
        let alertController = UIAlertController(title: "Опишіть помилку", message: "Докладно опишіть вашу помилку, яку ви виявили", preferredStyle: .alert)
        alertController.addTextField { (textField) in
            textField.placeholder = "Опис помилки"
        }
        let sendAction = UIAlertAction(title: "Надіслати", style: .default) { _ in
            if let errorMessage = alertController.textFields?.first?.text {
                print("Помилка: \(errorMessage)")
            }
        }
        alertController.addAction(sendAction)
        self.present(alertController, animated: true, completion: nil)
    }
    @IBAction func leaveFeedbackButtonPress(_ sender: UIBarButtonItem) {
        let alertController = UIAlertController(title: "Напишіть ваш відгук", message: "Дякуємо вам за покращення нашого додатку", preferredStyle: .alert)
        alertController.addTextField { (textField) in
            textField.placeholder = "Ваш відгук"
        }
        let sendAction = UIAlertAction(title: "Надіслати", style: .default) { _ in
```

```
                if let errorMessage = alertController.textFields?.first?.text {
                    print("Відгук: \(errorMessage)")
                }
            }
            alertController.addAction(sendAction)
            self.present(alertController, animated: true, completion: nil)
        }
        @IBAction func checkUpdateButtonPress(_ sender: UIBarButtonItem) {
            let alertController: UIAlertController
            if currentVersion == latestVersion {
                alertController = UIAlertController(title: "Додаток оновлено до останньої версії", message: nil, preferredStyle: .alert)
            } else {
                alertController = UIAlertController(title: "Доступне оновлення. Будь ласка, оновіть додаток до останньої версії", message: nil, preferredStyle: .alert)
            }
            let closeAction = UIAlertAction(title: "Закрити", style: .default, handler: nil)
            alertController.addAction(closeAction)
            self.present(alertController, animated: true, completion: nil)
        }
        @IBAction func enableNotificationsButtonPress(_ sender: UIBarButtonItem) {
            if let appSettingsURL = URL(string: UIApplication.openSettingsURLString) {
                if UIApplication.shared.canOpenURL(appSettingsURL) {
                    UIApplication.shared.open(appSettingsURL, options: [:], completionHandler: nil)
                }
            }
        }
    }
```

ДОДАТОК Г

ShopController.swift

```
import UIKit
var coins: Int = 100
var stars: Int = 0
class ShopController: UIViewController {
    @IBOutlet weak var coinsCounterLabel: UILabel!
    @IBOutlet weak var starsCounterLabel: UILabel!
    @IBOutlet weak var statusLabel: UILabel!
    override func viewDidLoad() {
        super.viewDidLoad()
        coinsCounterLabel.text = "Доступні монети: \(coins) 🪙"
        NotificationCenter.default.addObserver(self, selector: #selector(updateCoinsLabel(_:)), name: NSNotification.Name("updateCoins"), object: nil)
        starsCounterLabel.text = "Доступні зірки: \(stars) ☆"
    }
    @objc func updateCoinsLabel(_ notification: Notification) {
        if let coins = notification.userInfo?["coins"] as? Int {
            coinsCounterLabel.text = "Доступні монети: \(coins) 🪙"
        }
    }
    deinit {
        NotificationCenter.default.removeObserver(self, name: NSNotification.Name("updateCoins"), object: nil)
    }
    @IBAction func convertCoinsToStars(_ sender: UIButton) {
        if coins >= 100 {
            let starsToAdd = coins / 100
            stars += starsToAdd
            coins %= 100
            NotificationCenter.default.post(name: NSNotification.Name("updateStars"), object: nil, userInfo: ["stars": stars])
```

```
            statusLabel.text = "Успішний обмін"
            coinsCounterLabel.text = "Доступні монети: \(coins) "
            starsCounterLabel.text = "Доступні зірки: \(stars) "
        } else {
            statusLabel.text = "Монет недостатньо для обміну"
        }
    }
}
```

ДОДАТОК Д

TryYourLuckController.swift

```
import UIKit
var attempts = 2
class TryYourLuckController: UIViewController {
    @IBOutlet weak var attemptsLabel: UILabel!
    @IBOutlet weak var prizeLabel: UILabel!
    override func viewDidLoad() {
        super.viewDidLoad()
        attemptsLabel.text = "Залишилось спроб: \(attempts)"
    }
    @IBAction func playButtonPress(_ sender: UIButton) {
        spinWheel()
    }
    func spinWheel() {
        guard attempts > 0 else {
            prizeLabel.text = "Спроб більше немає"
            return
        }
        attempts -= 1
        attemptsLabel.text = "Залишилось спроб: \(attempts)"
        let randomValue = Int.random(in: 0..<100)
        switch randomValue {
        case 0..<5:
            coins += 1000
            NotificationCenter.default.post(name:        NSNotification.Name("updateCoins"),
object: nil, userInfo: ["coins": coins])
            prizeLabel.text = "Вы виграли 1000 монет!"
        case 5..<25:
            prizeLabel.text = "Нічого не виграли"
        case 25..<40:
            attempts += 1
```

```
                prizeLabel.text = "Ви отримали ще одну спробу!"
                attemptsLabel.text = "Залишилось спроб: \(attempts)"
            case 40..<50:
                coins += 500
                NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins": coins])
                prizeLabel.text = "Ви виграли 500 монет!"
            case 50..<70:
                coins += 250
                NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins": coins])
                prizeLabel.text = "Ви виграли 250 монет!"
            case 70..<100:
                coins += 100
                NotificationCenter.default.post(name: NSNotification.Name("updateCoins"), object: nil, userInfo: ["coins": coins])
                prizeLabel.text = "Ви виграли 100 монет!"
            default:
                prizeLabel.text = "Помилка"
            }
        }
    }
```

ДОДАТОК Е

DateTimeController.swift

```
import UIKit
var startTime: Date = Calendar.current.date(bySettingHour: 17, minute: 14, second: 0, of: Date())!
var components: DateComponents?
var timeComponents: DateComponents?
var calendar = Calendar.current
class DateTimeController: UIViewController {
    @IBOutlet weak var datePicker: UIDatePicker!
    @IBOutlet weak var timePicker: UIDatePicker!
    @IBOutlet weak var intervalTextField: UITextField!
    @IBOutlet weak var intervalStepper: UIStepper!
    override func viewDidLoad() {
        super.viewDidLoad()
        timePicker.timeZone = TimeZone(identifier: "Europe/Kiev")
        intervalStepper.minimumValue = 1
        intervalStepper.stepValue = 1
        intervalStepper.value = interval
    }
    @IBAction func stepperChanged(_ sender: UIStepper) {
        intervalTextField.text = "\(Int(sender.value))"
    }
    @IBAction func setDateTimeButton(_ sender: UIButton) {
        let selectedDate = datePicker.date
        let selectedTime = timePicker.date
        components = Calendar.current.dateComponents([.year, .month, .day], from: selectedDate)
        timeComponents = Calendar.current.dateComponents([.hour, .minute, .second], from: selectedTime)
        if let components = components, let timeComponents = timeComponents {
            startTime = Calendar.current.date(bySettingHour: timeComponents.hour ?? 0,
```

```
                                minute: timeComponents.minute ?? 0,
                                second: timeComponents.second ?? 0,
                                of: Calendar.current.date(from: components)!)!
        }
        interval = intervalStepper.value
        print(interval)
    }
}
```