МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕББРАУЗЕРУ ДЛЯ ПЛАТФОРМИ IOS З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ WEBKIT»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(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 – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

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

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 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 веббраузеру для платформи IOS з використанням фреймворку WebKit

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

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023

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

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

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 07.02.2023 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.	08.02.2023	
2.	Збір вихідних даних.	10.02.2023	
3.	Обробка методичних та теоретичних	24.02.2023	
	джерел.		
4.	Розробка першого та другого розділу.	28.03.2023	
5.	Розробка третього розділу.	05.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	23.06.2023	

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 бакалавра «Розробка веббраузеру для платформи IOS з використанням фреймворку WebKit»: 46 с., 23 рис., 10 джерел, 2 додатки.

ВЕБ-ДОДАТОК, ДОСЛІДЖЕННЯ, ПЛАТФОРМА IOS, РОЗРОБКА, ФРЕЙМВОРК, SWIFT, WEBKIT.

Об’єкт дослідження – розробка веббраузера для платформи iOS з використанням фреймворку WebKit.

Мета роботи: розробити функціональний веббраузер для платформи iOS, що базується на фреймворку WebKit. Реалізувати основні можливості браузера та дослідити можливості фреймворку WebKit.

Метод дослідження – аналіз функціональних можливостей фреймворку WebKit для розробки веббраузеру на платформі IOS.

У сучасному світі веб-браузери є одним з найважливіших інструментів, що дозволяють користувачам отримувати доступ до Інтернету та його різноманітного контенту. Одним з популярних фреймворків для розробки веб-браузерів на платформі iOS є WebKit. Він передбачає використання мов програмування, таких як Swift, для створення функціональних можливостей браузера. Крім того, фреймворк WebKit надає можливості роботи з веб-додатками та розширеннями. Розробник може використовувати WebKit для реалізації плагінів, які додають додаткові функції до браузера.

Розробка веб-браузера для платформи iOS з використанням фреймворку WebKit є складним і захоплюючим процесом. Вона вимагає знань мов програмування, архітектури браузерів, роботи з мережею та веб-технологій. Завдяки потужному функціоналу WebKit, розробники можуть створювати швидкі, безпечні та функціональні веб-браузери для платформи iOS, що задовольняють потреби користувачів.

SUMMARY

Bachelor’s qualifying paper «Development of a Web Browser for the IOS Platform using the WebKit Framework»: 46 pages, 23 figures, 10 references, 2 supplements.

WEB APP, RESEARCH, IOS PLATFORM, DEVELOPMENT, FRAMEWORK, SWIFT, WEBKIT.

Object of the study is development of a web browser for the iOS platform using the WebKit framework.

Aim of the study is to develop a functional web browser for the iOS platform based on the WebKit framework. Implement the basic capabilities of the browser and explore the capabilities of the WebKit framework.

Method of research is analysis of the functionality of the WebKit framework for the development of a web browser on the IOS platform.

In today’s world, web browsers are one of the most important tools that allow users to access the Internet and its diverse content. One of the popular frameworks for developing web browsers on the iOS platform is WebKit. It involves using programming languages like Swift to build browser functionality. In addition, the WebKit framework provides the ability to work with web applications and extensions. A developer can use WebKit to implement plugins that add additional functionality to the browser.

Developing a web browser for the iOS platform using the WebKit framework is a complex and exciting process. It requires knowledge of programming languages, browser architecture, networking and web technologies. Thanks to the powerful functionality of WebKit, developers can create fast, secure and functional web browsers for the iOS platform that meet the needs of users.

ЗМІСТ

Завдання на кваліфікаційну роботу……..…..2
Реферат 4
Summary 5
Вступ..... 8
1 Постановка задачі..... 9
1.1 Мета та завдання дослідження 9
1.2 Об’єкт та предмет дослідження..... 10
1.3 Методологія дослідження 11
2 Основні теоретичні відомості 12
2.1 Актуальність теми..... 12
2.2 Наукова новизна та практична цінність роботи 12
3 Розробка веббраузеру та тестування..... 14
3.1 Огляд та порівняння існуючих веб-браузерів для платформи iOS... 14
3.1.1 Safari 14
3.1.2 Google Chrome..... 15
3.1.3 Mozilla Firefox 16
3.1.4 Opera..... 16
3.1.5 Microsoft Edge..... 17
3.2 Огляд фреймворку WebKit..... 18
3.2.1 Архітектура та компоненти WebKit..... 20
3.2.2 Можливості та функціональність WebKit..... 21
3.3 Налаштування середовища розробки Xcode 23
3.4 Обов'язкові файли проекту та їх призначення 24
3.4.1 Файл AppDelegate.swift 24
3.4.2 Файл SceneDelegate.swift..... 25
3.4.3 Файл ViewController.swift 27
3.5 Функції веббраузеру та їх реалізація 27

3.5.1 Відкриття стартової сторінки та її відображення 28
3.5.2 Відображення URL у адресній строці 28
3.5.3 Кнопки «Назад» та «Вперед» 29
3.5.4 Спливаюче вікно з додатковими функціями 30
3.5.5 Кнопка «Задати стартову сторінку» 31
3.5.6 Кнопка «Очистити усі cookie» 32
3.5.7 Кнопка «Очистити увесь cache» 32
3.5.8 Кнопка «Змінити тему додатку» 33
3.6 Тестування веббраузеру на практиці 34
Висновки 40
Перелік посилань 41
Додаток А Вміст файлу ViewController.swift 42
Додаток Б Вміст файлу SettingsController.swift 44

ВСТУП

У сучасному світі веб-браузери відіграють ключову роль у сприйнятті та взаємодії з веб-середовищем. Вони є необхідним інструментом для доступу до різноманітних веб-ресурсів, від звичайних веб-сайтів до веб-додатків та послуг.

Платформа iOS, розроблена компанією Apple, налічує мільйони активних користувачів по всьому світу. Ця платформа володіє великою базою додатків, серед яких веб-браузери займають одну з провідних позицій. Вони стають вікном у світ Інтернету для користувачів iOS-пристроїв, надаючи можливість переглядати веб-сторінки, взаємодіяти з веб-елементами та отримувати доступ до різноманітних сервісів і додатків.

Однак, розробка веб-браузера для платформи iOS вимагає певних навичок та знань, а також потужних інструментів, щоб забезпечити стабільність, швидкість та безпеку використання. У даній роботі при розробці веб-браузера для iOS, буде використовуватись фреймворк WebKit, який є основною технологією браузерів на платформі iOS.

У процесі розробки буде досліджено можливості WebKit, вивчення його API та інтегрування його функціональних можливостей в браузер. Крім того, буде досліджено основні принципи розробки для платформи iOS, зокрема використання мови програмування Swift та інших інструментів. Також буде зосереджена увага на створенні інтуїтивного інтерфейсу користувача, який дозволить легко та зручно навігувати веб-середовищем.

Результати цієї роботи можуть бути корисними як для розробників програмного забезпечення, що цікавляться розробкою веб-браузерів , так і для користувачів iOS-пристроїв, які шукають надійний та зручний браузер для своїх потреб.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Мета та завдання дослідження

Метою даного дослідження є розробка веб-браузера для платформи iOS з використанням фреймворку WebKit, який буде володіти широким функціоналом та надавати зручний та продуктивний веб-перегляд.

Для досягнення цієї мети треба виділити декілька важливих пунктів.

Вивчення особливостей платформи iOS та фреймворку WebKit. Розуміння архітектури iOS і його взаємодії з веб-браузером є необхідним для успішної розробки. Аналіз можливостей та обмежень фреймворку WebKit дозволить використовувати його потужності ефективно та оптимально.

Визначення функціональних вимог до веб-браузера. Розробка детального переліку функцій, які повинен мати веб-браузер, забезпечить зручний та функціональний веб-перегляд для користувачів.

Вибір архітектури та технологій розробки. Вибір оптимальної архітектури додатка, яка буде забезпечувати ефективну роботу веб-браузера. Визначення набору технологій, які допоможуть реалізувати необхідний функціонал та забезпечити високу продуктивність.

Реалізація основного функціоналу веб-браузера. Розробка основних функцій веб-браузера, таких як відображення веб-сторінок, навігація, закладки, пошукові системи та інші. Забезпечення інтуїтивного та зручного інтерфейсу користувача.

Тестування та оптимізація продуктивності. Виконання функціонального та негативного тестування веб-браузера для перевірки коректності роботи та виявлення можливих помилок. Оптимізація продуктивності для забезпечення швидкодії та ефективності веб-перегляду.

Це дослідження є важливим, оскільки розробка веб-браузера для платформи iOS з використанням фреймворку WebKit відкриває нові

можливості для користувачів iOS та розробників додатків. Результати цього дослідження сприятимуть поліпшенню веб-перегляду на пристроях з iOS та можуть бути використані як основа для подальшого розвитку веб-браузерів для цієї платформи.

1.2 Об’єкт та предмет дослідження

Об’єктом дослідження є процес розробки веб-браузера для платформи iOS з використанням фреймворку WebKit.

Розробка веб-браузера передбачає створення програмного забезпечення, яке забезпечує користувачам можливість переглядати веб-сторінки, виконувати пошук та виконувати інші дії, пов'язані з веб-переглядом.

Предметом дослідження є фреймворк WebKit, який входить в склад платформи iOS і забезпечує базові функції веб-браузеру, зокрема веб-рендеринг, обробку HTML, CSS та JavaScript, навігацію по сторінкам тощо.

Дослідження фреймворку WebKit передбачає аналіз його можливостей, дослідження методів використання та інтеграції з додатком, а також оптимізацію продуктивності веб-браузера, що базується на WebKit.

Дослідження спрямоване на розуміння основних аспектів розробки веб-браузера для платформи iOS з використанням фреймворку WebKit, забезпечення його стабільної та ефективної роботи, а також розробку широкого функціоналу, що задовольнятиме потреби користувачів.

Отримані результати дослідження дозволять розробникам створювати високоякісні веб-браузери для iOS, які забезпечуватимуть зручний та швидкий веб-перегляд. Користувачі iOS пристроїв також зможуть скористатися широким вибором веб-браузерів, що задовольнятимуть їх потреби та вимоги.

1.3 Методологія дослідження

Для досягнення поставленої мети та вирішення завдань дослідження використовується комплексна методологія, яка включає кілька основних етапів.

Початковим етапом дослідження є аналіз літературних джерел та вивчення наукових та технічних робіт, пов'язаних з розробкою веб-браузерів для платформи iOS та фреймворком WebKit.

Далі, проводиться детальний аналіз функціональних та технічних можливостей фреймворку WebKit, вивчення документації, аналіз вихідного коду та експериментальні дослідження, щоб краще зрозуміти можливості та обмеження фреймворку.

Після аналізу фреймворку WebKit, розпочинається проектування архітектури веб-браузера для платформи iOS. Цей етап включає визначення основних компонентів браузера, взаємодію між ними та вибір оптимальних підходів до реалізації функцій.

Після проектування архітектури, переходимо до етапу розробки, де реалізовується веб-браузер на основі фреймворку WebKit. Використовуючи мову програмування, підтримувану на платформі iOS, розробники реалізують функціонал браузера, взаємодіючи з WebKit API та іншими необхідними компонентами.

Після завершення розробки проводиться тестування веб-браузера з метою виявлення та виправлення помилок, а також оцінки його продуктивності та стабільності.

Останнім етапом дослідження є аналіз отриманих результатів та їх інтерпретація. Вивчення результатів дозволяє оцінити ефективність розробленого веб-браузера, порівняти його з існуючими рішеннями та зробити висновки про досягнення мети дослідження.

Таким чином, методологія дослідження включає аналіз літературних джерел, аналіз функціональних та технічних можливостей WebKit, проектування архітектури браузера, розробку, тестування та аналіз результатів.

2 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Актуальність теми

Актуальність теми «Розробка веб-браузера для платформи iOS з використанням фреймворку WebKit» зумовлена постійним зростанням популярності мобільних пристроїв, зокрема смартфонів та планшетів на базі iOS. Веб-браузери є ключовими додатками для доступу до Інтернету, і забезпечення швидкого та зручного веб-перегляду стає все більш важливим завданням.

Незважаючи на наявність в iOS вбудованого веб-браузера Safari, існує потреба у розробці альтернативних веб-браузерів, які могли б задовольняти особливі потреби користувачів та пропонувати нові функціональні можливості. Використання фреймворку WebKit в розробці веб-браузерів дозволяє отримати доступ до потужного движка веб-рендерингу, що реалізує сучасні стандарти Інтернету.

Ця робота має на меті розробку веб-браузера для платформи iOS з використанням фреймворку WebKit, який буде надавати широкий функціонал і забезпечувати стабільну та швидку роботу. Це дозволить користувачам iOS пристроїв мати більше варіантів веб-браузерів і знайти оптимальний інструмент для своїх потреб.

2.2 Наукова новизна та практична цінність роботи

Наукова новизна даної роботи полягає в розробці веб-браузера для платформи iOS з використанням фреймворку WebKit та дослідженні його можливостей і функціоналу. Це дозволить розширити можливості веб-перегляду на пристроях iOS та забезпечити користувачам широкий вибір веб-

браузерів з різноманітним функціоналом.

Попередні розробки в області веб-браузерів для платформи iOS переважно зосереджувалися на використанні вбудованого браузера Safari. Розробка альтернативних веб-браузерів з використанням фреймворку WebKit є новим підходом, який відкриває нові можливості для покращення веб-перегляду на iOS пристроях. Дане дослідження внесе свій внесок в цю область, дозволяючи розробникам створювати більш інноваційні та функціональні веб-браузери для iOS.

Крім того, дослідження фреймворку WebKit та його інтеграція з веб-браузером для iOS принесе практичну цінність для розробників додатків та користувачів. Розробники зможуть отримати поглиблене розуміння процесу розробки веб-браузера для iOS, використовуючи потужні можливості WebKit. Це дозволить їм створювати більш ефективні та функціональні додатки для iOS пристроїв, які забезпечуватимуть зручний та продуктивний веб-перегляд.

Для користувачів iOS пристроїв розробка веб-браузера на основі WebKit принесе багато практичних переваг. Вони матимуть доступ до більшого вибору веб-браузерів з різноманітним функціоналом, який задовольнить їхні потреби та вимоги. Більшість користувачів покладають велику вагу на зручність використання та функціонал веб-браузера, тому розробка веб-браузера з використанням WebKit може значно покращити їхній досвід веб-перегляду.

Також, практична цінність дослідження полягає в тому, що результати дослідження можуть бути використані як основа для подальшого розвитку веб-браузерів для платформи iOS. Розробники зможуть використовувати отримані знання та практичний досвід для створення нових та поліпшених версій веб-браузерів, які відповідатимуть різним потребам користувачів.

Отже, наукова новизна та практична цінність даної роботи полягають у розробці веб-браузера для платформи iOS з використанням фреймворку WebKit, що внесе свій внесок в поліпшення веб-перегляду на iOS пристроях та надасть користувачам широкий вибір веб-браузерів з різноманітним функціоналом.

3 РОЗРОБКА ВЕББРАУЗЕРУ ТА ТЕСТУВАННЯ

3.1 Огляд та порівняння існуючих веб-браузерів для платформи iOS

Веб-браузери є важливими інструментами для користувачів платформи iOS, які дозволяють швидко та зручно переглядати веб-сторінки, використовувати веб-додатки та виконувати пошукові запити. У цьому підрозділі буде наведено основні переваги та недоліки п'яти популярних веб-браузерів для iOS: Safari, Google Chrome, Mozilla Firefox, Opera та Microsoft Edge (рис. 3.1).

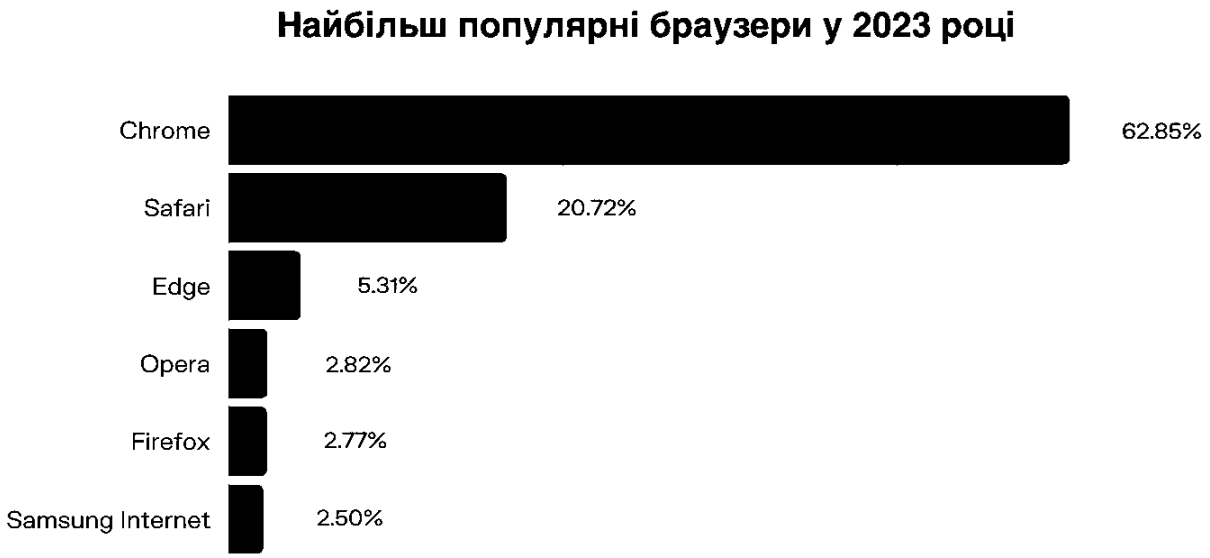

Рисунок 3.1 – Діаграма популярних браузерів

Розглянемо деякі найпопулярніші веббраузери для платформи iOS.

3.1.1 Safari

Safari є офіційним веб-браузером iOS, розробленим Apple. Він має швидкий швидкісний режим роботи, ефективне використання енергії і високий

рівень безпеки. Safari інтегрується з екосистемою Apple, що дозволяє синхронізувати закладки, паролі та історію перегляду між різними пристроями. Він також має функцію «Інтелектуальний трекер», яка допомагає блокувати стеження онлайн і покращує конфіденційність користувача.

Переваги:

- інтегрований з операційною системою iOS і має відмінну оптимізацію, що дозволяє ефективно використовувати ресурси пристрою;
- висока продуктивність та швидкість завантаження веб-сторінок;
- підтримка функцій, специфічних для платформи iOS, таких як Handoff, iCloud Tabs і AirPlay.

Недоліки:

- обмежена кастомізація та розширення, порівняно з іншими браузерами;
- відсутність підтримки деяких сучасних веб-стандартів.

3.1.2 Google Chrome

Google Chrome є одним з найпопулярніших веб-браузерів, який доступний як для iOS, так і для інших платформ. Він пропонує швидку швидкість завантаження сторінок, розширений набір функцій і синхронізацію даних з обліковим записом Google. Користувачі можуть синхронізувати закладки, історію перегляду та паролі між різними пристроями. Chrome також має функцію вбудованого перекладу сторінок, захист від шкідливих сайтів і підтримку розширень.

Переваги:

- синхронізація закладок, паролів та історії між різними пристроями через обліковий запис Google;
- великий вибір розширень та додатків з Chrome Web Store;
- швидка швидкість роботи та підтримка сучасних веб-стандартів.

Недоліки:

– високе споживання енергії та ресурсів пристрою, що може призводити до швидшого розрядження батареї;
– деякі функції, такі як Handoff і AirPlay, обмежені або відсутні.

3.1.3 Mozilla Firefox

Firefox від Mozilla є популярним веб-браузером, доступним і для iOS. Він пропонує швидкість завантаження сторінок і високий рівень безпеки. Firefox також має захист від слідкування і можливість блокувати вміст, такий як рекламу і попап-вікна. Користувачі можуть синхронізувати свої дані, включаючи закладки, паролі і історію перегляду, між різними пристроями за допомогою облікового запису Firefox.

Переваги:

– висока приватність та безпека завдяки вбудованим функціям, таким як захист від слідування та блокування шкідливих сайтів;
– великий вибір розширень та можливість кастомізації інтерфейсу;
– підтримка сучасних веб-стандартів та технологій.

Недоліки:

– відносно повільна швидкість роботи порівняно з іншими браузерами;
– деякі синхронізаційні можливості можуть бути обмежені.

3.1.4 Opera

Opera є менш популярним, але цікавим веб-браузером для iOS. Він має вбудований блокувальник реклами, VPN для безпечного перегляду, швидку швидкість завантаження сторінок і можливість зменшити використання даних. Opera також пропонує функцію «Мініатюрна вкладка», яка дозволяє

переглядати веб-сторінки у вигляді мініатюр, щоб швидко переключатися між ними.

Переваги:

– вбудовані функції, такі як блокування реклами, безкоштовний VPN і управління батареєю, що роблять перегляд веб-сторінок більш зручним та безпечним;
– швидкість роботи та висока продуктивність;
– можливість встановлювати розширення з Chrome Web Store.

Недоліки:

– деякі функції можуть бути менш стабільними та працювати некоректно;
– обмежена підтримка деяких сучасних веб-стандартів.

3.1.5 Microsoft Edge

Microsoft Edge, раніше відомий як Internet Explorer, є веб-браузером, доступним і для iOS. Він пропонує швидкість завантаження сторінок, синхронізацію даних з обліковим записом Microsoft і можливість продовжувати роботу на інших пристроях. Edge також має вбудований блокувальник реклами та функцію «Читання», яка дозволяє зручно переглядати статті без зайвих відволікань.

Переваги:

– висока швидкість завантаження сторінок та продуктивність;
– інтегровані функції, такі як Cortana (особистий асистент), перегляд PDF-файлів та режим читання;
– підтримка розширень з Microsoft Store та деяких розширень з Chrome Web Store.

Недоліки:

– відсутність підтримки деяких сучасних веб-стандартів;
– можливість використання більшої кількості ресурсів пристрою порівняно

з іншими браузерами.

На рисунку 3.2 представлено зведену таблицю порівняння популярних браузерів.

Порівняння браузерів

	CHROME	EDGE	FIREFOX	SAFARI
OPEN SOURCE	No	Partially	Yes	No
JAVASCRIPT BLOCKER	Yes	Yes	Yes	Yes, but only with the JavaScript blocker extension
NOTIFICATIONS FOR EXPOSED PASSWORDS	Yes	Yes	Yes	Yes, but IT must enable them
ENCRYPTED PASSWORD STORAGE	Yes, but it's easy to extract them with third-party tools	Yes, but it's easy to extract them with third-party tools	Yes, but it's easy to extract them with third-party tools	Yes; accessing the passwords requires encryption key
AUTO UPGRADES TO HTTPS	Yes	Yes, but IT has to configure it	Yes	Yes
ALLOWS EXTENSIONS	Yes, with limited security controls	Yes; IT can limit via Edge user experience	Yes; controls are available via addons.mozilla.org	Yes, but more focused on user control
PHISHING PROTECTION AND FRAUDULENT WEBSITE WARNINGS	Yes	Yes, with robust alter systems	Yes	Yes
DISABLE AND CONTROL TRACKING	Yes	Yes, with very granular controls	Yes	Yes, with reporting options
COLLECTS USER DATA FOR THIRD PARTIES	Yes	Yes	Yes, but IT can opt out	Yes

Рисунок 3.2 – Порівняльна таблиця популярних веббраузерів

3.2 Огляд фреймворку WebKit

WebKit – це відкритий веб-двигун, який використовується для рендерингу веб-сторінок у браузерах. Він був розроблений компанією Apple і спочатку випускався як частина їхньої операційної системи Mac OS X. Згодом

WebKit був використаний в багатьох інших браузерах, включаючи Google Chrome (до версії 27), Safari (до версії 14) і багато інших мобільних і настільних браузерів.

WebKit є одним з найпоширеніших веб-двигунів і використовується мільйонами користувачів щодня для перегляду веб-сторінок. Він забезпечує можливість відображати HTML-документи, виконувати JavaScript, обробляти CSS і виконувати багато інших веб-технологій.

Один з основних принципів WebKit – це швидкість і продуктивність. Він оптимізований для швидкого завантаження сторінок і рендерингу вмісту. WebKit використовує різні техніки, такі як асинхронне завантаження ресурсів, кешування і попереднє завантаження, щоб забезпечити швидку роботу з веб-сторінками.

WebKit також підтримує багато веб-стандартів і технологій. Він має вбудовану підтримку HTML, CSS і JavaScript, включаючи сучасні стандарти, такі як HTML5 і CSS3. Крім того, WebKit підтримує різні веб-стандарти, такі як WebGL (графіка в реальному часі в браузері), WebRTC (веб-зв'язок в реальному часі) і WebAssembly (виконання високопродуктивного коду в браузері).

WebKit також має декілька інструментів для розробників. Він має вбудований інструментарій для інспектування елементів сторінки, відладки JavaScript і профілювання продуктивності. Ці інструменти допомагають розробникам вивчати, аналізувати і відлагоджувати веб-сторінки для покращення продуктивності та відповідного функціонування.

WebKit є дуже гнучким і розширюваним. Існує багато різних проектів, які базуються на WebKit і використовують його в якості основи для створення власних веб-додатків і браузерів. Багато розширень і плагінів доступні для WebKit, що дозволяє розширити його функціональність і можливості.

Узагальнюючи, WebKit – це потужний і широко використовуваний веб-двигун, який забезпечує швидкий і ефективний рендеринг веб-сторінок. Він підтримує багато веб-стандартів і надає розробникам інструменти для створення веб-додатків. Через свою популярність і розширюваність, WebKit є

важливою складовою сучасного Інтернету.

3.2.1 Архітектура та компоненти WebKit

Архітектура WebKit базується на відкритих веб-стандартах і складається з низки компонентів (див. рис. 3.3).

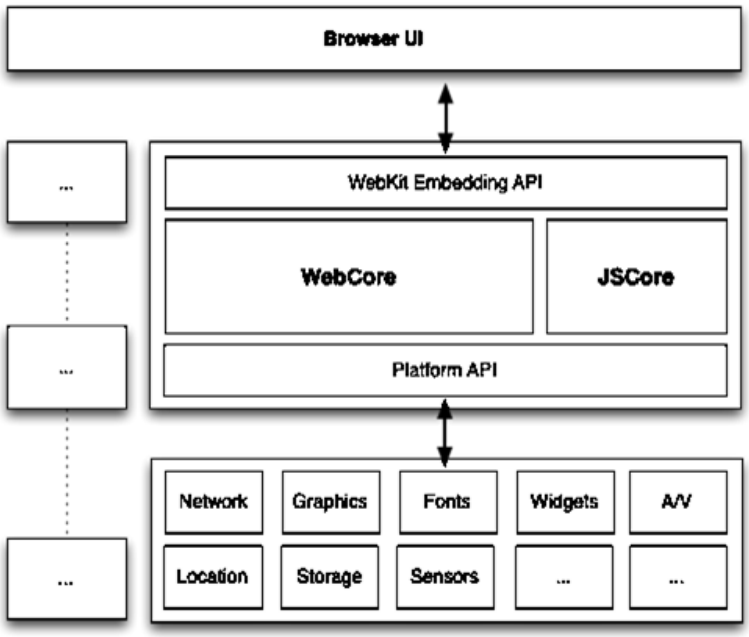

Рисунок 3.3 – Діаграма компонентів WebKit

Розглянемо, що включають основні компоненти WebKit.

WebKit Rendering Engine. Це основна частина WebKit, яка відповідає за обробку та відображення веб-сторінок. Вона включає рендерер (renderer), який перетворює HTML-код сторінки у відповідні графічні елементи, такі як текст, зображення, таблиці та інші.

JavaScript Engine. WebKit має свою власну JavaScript-машину, яка виконує JavaScript-код на веб-сторінках. Відома як JavaScriptCore або Nitro, ця машина забезпечує високу продуктивність і оптимізації виконання скриптів.

WebCore. Цей компонент відповідає за реалізацію веб-стандартів і забезпечення функціональності, такої як HTML, CSS, SVG, DOM (Document

Object Model) та інших. WebCore надає API для взаємодії між JavaScript-кодом та веб-сторінками.

Network Components. WebKit містить компоненти для мережевого взаємодії, такі як HTTP-запити, обробка файлів, управління кешем та куки. Вони відповідають за завантаження ресурсів сторінок, таких як зображення, стилі, скрипти та інші.

User Interface Components. WebKit також включає компоненти для користувацького інтерфейсу, такі як елементи керування (buttons, checkboxes), поля введення (input fields), діалогові вікна (dialog boxes) та інші. Ці компоненти використовуються для відображення веб-сторінок і взаємодії з ними.

Graphics Components. WebKit використовує різні компоненти для обробки графіки, включаючи рендеринг зображень, векторну графіку, композитинг та анімацію.

Security Components. WebKit має вбудовану систему безпеки, яка включає заходи захисту від зловживань, перехоплення даних, фішингу та інших загроз. Ці компоненти допомагають забезпечити безпеку користувачів під час перегляду веб-сторінок.

Всі ці компоненти взаємодіють між собою, утворюючи потужну систему для обробки та відображення веб-сторінок. WebKit продовжує активно розвиватися, вдосконалюючи швидкість, функціональність та безпеку для користувачів у світі веб-перегляду.

3.2.2 Можливості та функціональність WebKit

WebKit забезпечує багатий набір можливостей та функціональність для створення потужних веб-додатків для iOS. Наведемо деякі з можливостей та функцій, які надає WebKit на платформі iOS.

Відображення веб-сторінок. WebKit забезпечує потужний движок

рендерингу для відображення веб-сторінок з використанням стандартів веб-розробки, таких як HTML, CSS та JavaScript. Це дозволяє користувачам переглядати веб-сторінки безпосередньо у своїх мобільних додатках.

Підтримка HTML5. WebKit підтримує багато функцій HTML5, включаючи аудіо та відео, графіку, локальне сховище та багато іншого. Це дозволяє розробникам створювати багатофункціональні веб-додатки для iOS, які можуть використовувати розширені можливості веб-платформи.

CSS-анімації та переходи. WebKit підтримує різноманітні CSS-анімації та переходи, що дозволяють розробникам створювати вражаючі ефекти візуального оформлення веб-сторінок. Це дає змогу покращити користувацький досвід у мобільних додатках, забезпечуючи більш плавні та привабливі переходи між сторінками.

JavaScript та взаємодія зі сторінками. WebKit надає підтримку для виконання JavaScript-коду на веб-сторінках. Це дозволяє розробникам створювати веб-додатки з динамічною функціональністю, включаючи взаємодію з користувачем, обробку подій та взаємодію зі сторонніми службами API.

Геолокація та доступ до датчиків. WebKit підтримує геолокацію та доступ до різних датчиків пристрою, таких як акселерометр, гіроскоп та компас. Це дозволяє веб-додаткам отримувати інформацію про місцезнаходження користувача або використовувати дані з датчиків для створення цікавих функцій.

Інтеграція зі зображеннями та мультимедіа. WebKit дозволяє розробникам використовувати зображення та мультимедійні елементи в своїх веб-додатках. Це означає, що веб-сторінки можуть відображати фотографії, відео, аудіо та інші медіа-ресурси, дозволяючи створювати багатофункціональні додатки з багатим візуальним вмістом.

Підтримка тач-жестів. WebKit включає підтримку тач-жестів, таких як збільшення, зменшення та перетягування, що дозволяє користувачам взаємодіяти з веб-сторінками на сенсорних пристроях iOS. Це сприяє

створенню інтуїтивного та зручного інтерфейсу користувача.

WebKit на платформі iOS надає потужні можливості для розробки веб-додатків з багатим функціоналом та відмінним користувацьким досвідом. Розробники можуть використовувати WebKit для створення мобільних додатків, які інтегруються з веб-технологіями та пропонують розширені функції, що поліпшують взаємодію з користувачем і забезпечують потужні можливості веб-платформи.

3.3 Налаштування середовища розробки Xcode

Xcode – це інтегроване середовище розробки (IDE), розроблене Apple, яке дозволяє створювати програми для iOS, macOS, watchOS та tvOS. Перед початком роботи з Xcode, потрібно налаштувати та встановити це середовище розробки для платформи macOS [7].

Для встановлення потрібен комп'ютер, який працює під управлінням операційної системи macOS. Оскільки Xcode доступний лише для macOS і не може бути встановлений на інших операційних системах. Для того щоб встановити Xcode потрібно виконати три кроки.

Перший крок. Потрібно перейти в App Store на Mac та використати пошукову функцію і ввести «Xcode» для пошуку.

Другий крок. Коли Xcode був знайдений, потрібно натиснути на кнопку «Get» або «Install» для початку процесу завантаження та встановлення. Xcode досить великий файл, тому цей процес може зайняти деякий час.

Третій крок. Після завершення встановлення, знайти Xcode можна в папці «Applications» на Mac. Тепер Xcode можна запустити, щоб почати роботу. Також при першому запуску Xcode може запросити увійти за допомогою облікового запису Apple ID.

3.4 Обов'язкові файли проекту та їх призначення

При розробці iOS-проекту в Xcode є декілька обов'язкових файлів: AppDelegate.swift, SceneDelegate.swift, ViewController.swift. Це основні файли, які завжди будуть в проекті Xcode під час розробки iOS-додатка.

3.4.1 Файл AppDelegate.swift

Файл AppDelegate у проекті Xcode є одним з основних файлів, які використовуються для керування життєвим циклом додатка. Він виконує роль делегата додатка, що означає, що він отримує повідомлення про події, пов'язані з життєвим циклом додатка і може реагувати на них [1].

Розглянемо нижче основні функції AppDelegate.

Application Launch (Запуск додатка). Коли додаток запускається, AppDelegate отримує повідомлення про цю подію і може виконати певні дії, наприклад, ініціалізувати деякі об'єкти або налаштувати початковий стан додатка.

Application Termination (Закриття додатка). Коли користувач закриває додаток, AppDelegate отримує повідомлення про закриття і може виконати певні дії перед закриттям додатка. Наприклад, він може зберегти поточний стан додатка або виконати інші завершальні операції.

Background Execution (Виконання в фоновому режимі). AppDelegate може обробляти повідомлення про запуск додатка в фоновому режимі або перехоплювати повідомлення від пристрою, якщо додаток використовує функціональність фонового виконання.

Remote Notifications (Віддалені повідомлення). AppDelegate може реагувати на вхідні віддалені повідомлення, такі як повідомлення з пуш-сервера, і виконувати певні дії, наприклад, показувати сповіщення або оновлювати інтерфейс користувача.

State Restoration (Відновлення стану). AppDelegate може допомогти у відновленні стану додатка після його перезавантаження або відновлення з фонового режиму.

Вміст файлу AppDelegate.swift представлено на рисунку 3.4.

```
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 3.4 – Файл AppDelegate.swift

3.4.2 Файл SceneDelegate.swift

Файл SceneDelegate у проекті Xcode використовується для керування життєвим циклом сцен (scenes). SceneDelegate дозволяє додатку працювати з кількома сценами одночасно і керувати їхнім поведінкою. Кожна сцена може мати власний набір вікон та відповідних уявлень (views), і SceneDelegate допомагає управляти цими компонентами [2].

Розглянемо нижче основні функції SceneDelegate.

Створення сцен і вікон. SceneDelegate дозволяє створювати нові сцени і

вікна, коли додаток запускається або отримує нову сцену для обробки.

Обробка змін стану сцен. SceneDelegate слідкує за змінами стану сцен, такими як активність, видимість або фокус, і виконує відповідні дії. Наприклад, він може реагувати на зміну активної сцени і оновлювати вміст додатку відповідно.

Обробка сесій уявлень (view sessions). SceneDelegate керує сесіями уявлень для кожної сцени. Він може відстежувати, коли сцена отримує нові уявлення, змінює орієнтацію екрана або відображає сповіщення, і виконує необхідні дії для оновлення інтерфейсу користувача.

Завершення роботи з сценами. Коли сцена закривається або додаток завершує роботу, SceneDelegate відповідає за виконання останніх дій, таких як збереження стану додатку або очищення ресурсів.

Вміст файлу SceneDelegate.swift представлено на рисунку 3.5.

```
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 3.5 – Файл SceneDelegate.swift

3.4.3 Файл ViewController.swift

Файл ViewController у проекті Xcode використовується для управління користувацьким інтерфейсом додатку на основі шаблону архітектури Model-View-Controller (MVC). Він відповідає за взаємодію між моделью даних програми (Model) і її користувацьким інтерфейсом (View) [4].

Основна роль ViewController полягає у керуванні подіями та діями, які стосуються взаємодії з користувачем. Він обробляє вхідні дані від користувача, виконує необхідні операції і забезпечує відображення відповідного вмісту на екрані. Наприклад, він може реагувати на натискання кнопок, переміщення пальцем по екрану або введення тексту.

ViewController також може виконувати інші функції, такі як обробка даних, взаємодія з іншими компонентами програми (наприклад, базою даних чи API) і керування переходами між різними екранами (вікнами) програми.

Вміст файлу ViewController.swift представлено на рисунку 3.6.

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

Рисунок 3.6 – Файл ViewController.swift

3.5 Функції веббраузеру та їх реалізація

Веббраузер може мати різні функції, які реалізовані для забезпечення зручного та ефективного перегляду веб-контенту. Їх може бути велика кількість, кожна з яких буде використовуватись для поліпшення користування.

3.5.1 Відкриття стартової сторінки та її відображення

Цей код демонструє відкриття стартової сторінки у веббраузері при першому відкритті додатку (див. рис. 3.7). По замовчуванню це начальна сторінка пошуковика Google [9].

У змінній під назвою "startPageURL" задається адреса стартової сторінки. Ця змінна передається до функції "loadStartPage()", у якій змінну приймає WebView фреймворк. При завантаженні додатку викликається функція "loadStartPage()" і сторінка відкривається.

```
import UIKit
import WebKit

class ViewController: UIViewController, WKNavigationDelegate {

@IBOutlet weak var webView: WKWebView!

   override func viewDidLoad() {
      super.viewDidLoad()
      loadStartPage()
   }

func loadStartPage() {
      let url = URL(string: startPageURL)
      let request = URLRequest(url: url!)
      webView.load(request)
   }
}
```

Рисунок 3.7 – Код для відкриття стартової сторінки та її відображення

3.5.2 Відображення URL у адресній строці

Цей код демонструє відображення поточного URL відкритого сайту у адресній строці веббраузеру зверху (див. рис. 3.8).

Коли відбувається перехід на іншу сторінку, то береться значення URL на

який було здійснено перехід, він має назву "change.newValue". Після чого це значення передається до текстового поля "addressBar".

```
@IBOutlet weak var addressBar: UITextField!

var webViewURLObserver: NSKeyValueObservation?

webViewURLObserver = webView.observe(\.url, options: .new) { webview, change in
        self.addressBar.text = String(describing: change.newValue!!)
}
```

Рисунок 3.8 – Код для відображення URL у адресній строці

3.5.3 Кнопки «Назад» та «Вперед»

Цей код демонструє кнопки «Назад» та «Вперед» та зв'язок візуального інтерфейсу користувача та коду додатку (див. рис. 3.9).

Дія при натисканні на кнопку «Назад» має назву «pressBack», тобто користувач може повернутись на ту сторінку на якій був до завантаження наступної сторінки. Дія при натисканні на кнопку «Вперед» має назву «pressForward», після чого, якщо була відкрита сторінка та натиснута кнопка «Назад», то є можливість повернутись [3].

```
@IBAction func pressForward(_ sender: UIButton) {
      webView.goForward()
   }

@IBAction func pressBack(_ sender: UIButton) {
      webView.goBack()
   }
```

Рисунок 3.9 – Код для кнопок «Назад» та «Вперед»

3.5.4 Спливаюче вікно з додатковими функціями

Цей код демонструє спливаюче вікно з додатковими функціями, на які не задано кнопок у візуальному інтерфейсі (див. рис. 3.10).

Після натискання на кнопку під назвою «pressOtherFunc», відкривається спливаюче вікно з трьома новими кнопками. Такими як: «Оновити сторінку» «Зупинити завантаження сторінки» та «Повернутись на домашню сторінку». Кожна кнопка зв’язана дією з WebView і виконується моментально після натискання на себе.

```
@IBAction func pressOtherFunc(_ sender: UIButton) {

let alert = UIAlertController(title: "Додаткові функції", message: "Оберіть що саме вам потрібно", preferredStyle: .actionSheet)
alert.addAction(UIAlertAction(title: "Відмінити", style: .cancel, handler: { (action) in print ("Відмінити")
   }))

alert.addAction(UIAlertAction(title: "Оновити сторінку", style: .default, handler: { _ in
      self.webView.reload()
   }))
alert.addAction(UIAlertAction(title: "Зупинити завантаження сторінки", style: .default, handler: { _ in
      self.webView.stopLoading()
   }))
alert.addAction(UIAlertAction(title: "Повернутись на домашню сторінку", style: .default, handler: { _ in
         self.loadStartPage()
   }))

self.present(alert, animated: true)
   }
```

Рисунок 3.10 – Код для спливаючого вікна з додатковими функціями

3.5.5 Кнопка «Задати стартову сторінку»

Цей код демонструє відкриття спливаючого вікна, у якому можна задати стартову сторінку яка буде відкриватись при запуску додатку (див. рис. 3.11).

Після натискання на кнопку під назвою «pressSetStartPage», відкривається спливаюче вікно з текстовим полем. У цьому полі треба вказати URL сторінки який веббраузер буде вважати за домашню. Після того як сторінка була вказана до неї автоматично додається «https://». Цей URL передається у змінну «startPageURL».

```
var startPageURL = "https://www.google.com/"

@IBAction func pressSetStartPage(_ sender: UIBarButtonItem) {

        let alertController = UIAlertController(title: "Задати стартову сторінку", message:
"Вкажіть URL сторінки", preferredStyle: .alert)

        alertController.addTextField { textField in
          textField.placeholder = "google.com"
           textField.keyboardType = .URL
          textField.autocorrectionType = .no
        }

        let okAction = UIAlertAction(title: "Задати", style: .default) {
          action in guard let textField = alertController.textFields?.first,
          let text = textField.text else {
            print("Ви не задали сторінку")
            return
          }
          startPageURL = ("https://") + text
        }

        let cancelAction = UIAlertAction(title: "Відмінити", style: .cancel, handler: nil)

        alertController.addAction(okAction)
        alertController.addAction(cancelAction)
        present(alertController, animated: true, completion: nil)
    }
```

Рисунок 3.11 – Код для кнопки «Задати стартову сторінку»

3.5.6 Кнопка «Очистити усі cookie»

Цей код демонструє функцію, яка повністю очищає cookie від усіх сайтів, які були завантажені веббраузером (див. рис. 3.12).

Після натискання на кнопку під назвою «pressClearCookie» йде пошук по даним кожного сайту які були завантажені, коли хоч одні дані були знайдені вони одразу видаляються [10].

```
@IBAction func pressClearCookie(_ sender: UIBarButtonItem) {

    HTTPCookieStorage.shared.removeCookies(since: Date.distantPast)
    print("[WebCacheCleaner] All cookies deleted")

    WKWebsiteDataStore.default().fetchDataRecords(ofTypes:
WKWebsiteDataStore.allWebsiteDataTypes()) { records in
        records.forEach { record in
            WKWebsiteDataStore.default().removeData(ofTypes: record.dataTypes, for:
[record], completionHandler: {})
            print("[WebCacheCleaner] Record \(record) deleted")
        }
    }
}
```

Рисунок 3.12 – Код для кнопки «Очистити усі cookie»

3.5.7 Кнопка «Очистити увесь cache»

Цей код демонструє функцію, яка повністю очищає cache від усіх сайтів, які були завантажені веббраузером (див. рис. 3.13).

Після натискання на кнопку під назвою «pressClearCache» повністю видаляються увесь cache який був збережений за увесь час користування веббраузером [6].

```
@IBAction func pressClearCache(_ sender: UIBarButtonItem) {

    WKWebsiteDataStore.default().removeData(ofTypes: [WKWebsiteDataTypeDiskCache,
WKWebsiteDataTypeMemoryCache], modifiedSince: Date(timeIntervalSince1970: 0),
completionHandler:{ })

}
```

Рисунок 3.13 – Код для кнопки «Очистити увесь cache»

3.5.8 Кнопка «Змінити тему додатку»

Цей код демонструє функцію, яка змінює тему додатку (див. рис. 3.14).

Після натискання на кнопку під назвою «pressChangeTheme» виконується перевірка на те, яка тема зараз, якщо світла, то вмикається темна тема, якщо ж темна тема, то вмикається світла [8].

```
@IBAction func pressChangeTheme(_ sender: UIBarButtonItem) {

    UIApplication.shared.windows.forEach { window in
        if window.overrideUserInterfaceStyle == .dark {
            window.overrideUserInterfaceStyle = .light
        }
        else {
            window.overrideUserInterfaceStyle = .dark
        }
    }
}
```

Рисунок 3.14 – Код для кнопки «Змінити тему додатку»

3.6 Тестування веббраузеру на практиці

Веббраузер розроблявся для мобільної платформи – IOS. Тому тестування буде проводитись на емуляторі вбудований у середу розробки Xcode (див. рис. 3.15).

Він дозволяє розробникам тестувати та перевіряти свої програми безпосередньо на віртуальних iOS-пристроях, що емулюються на комп'ютері. Це дозволяє перевірити як додаток буде працювати на різних версіях iOS, на різних iPhone, а також використовувати функції сповіщення, мережеві запити, взаємодію з камерою, мікрофоном та багато інших, без необхідності використання реальних пристроїв [5].

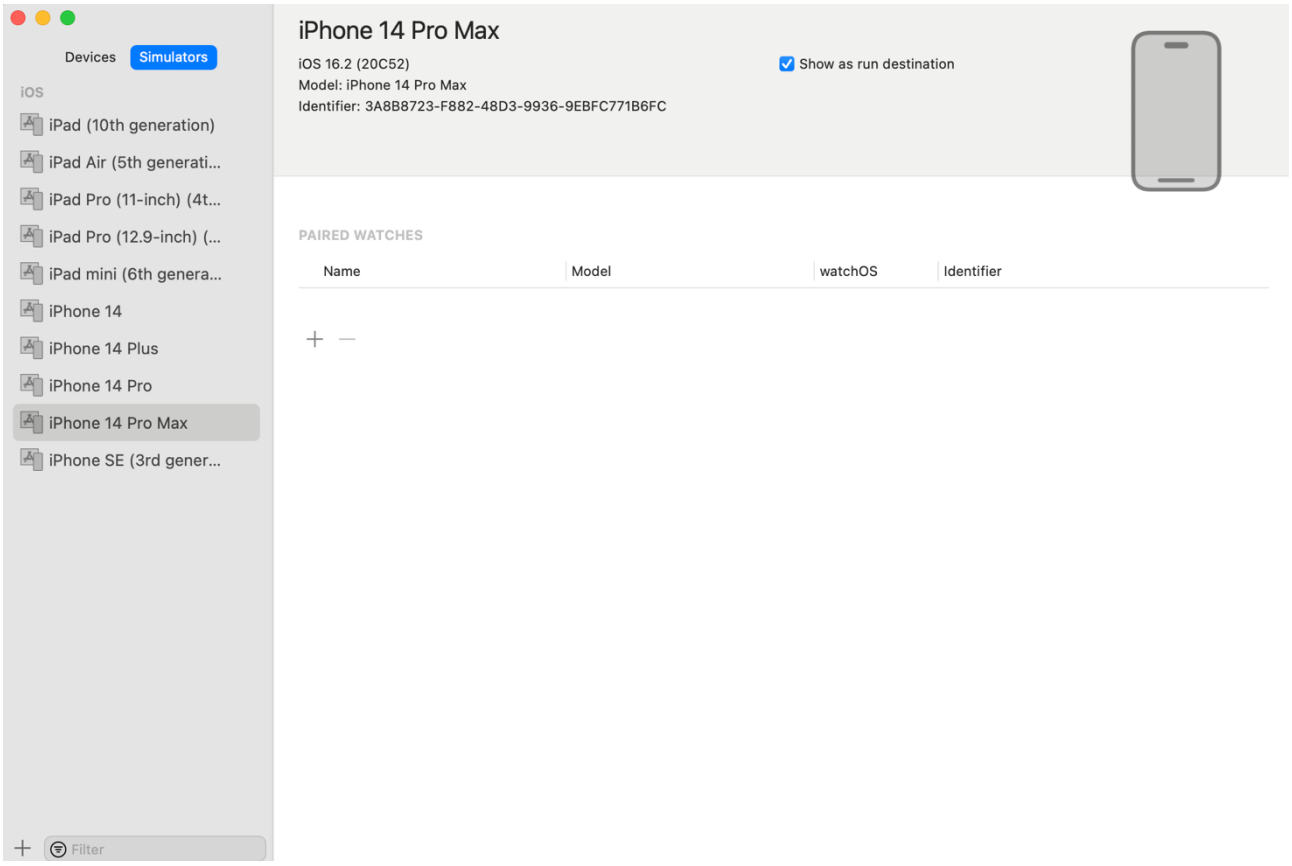

Рисунок 3.15 – Вікно вибору емулятору у Xcode

На рисунку 3.16 представлено відкриття додатку, безпосередньо – завантаження головної сторінки.

Рисунок 3.16 – Відкриття додатку та завантаження

Після завантаження, можна одразу почати користуватися браузером. На рисунку 3.17 представлена головна сторінка веббраузеру. Можна скористатись «пошуком» (див. рис. 3.18) та відкрити бажаний сайт (див. рис. 3.19).

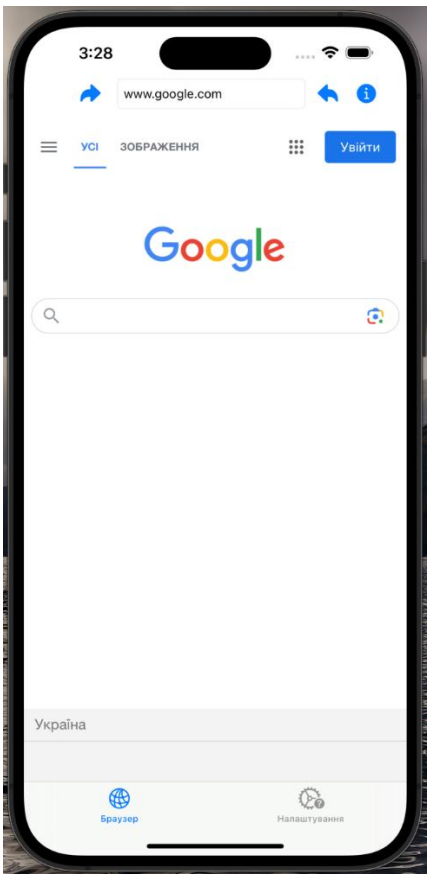

Рисунок 3.17 – Головна сторінка веббраузеру

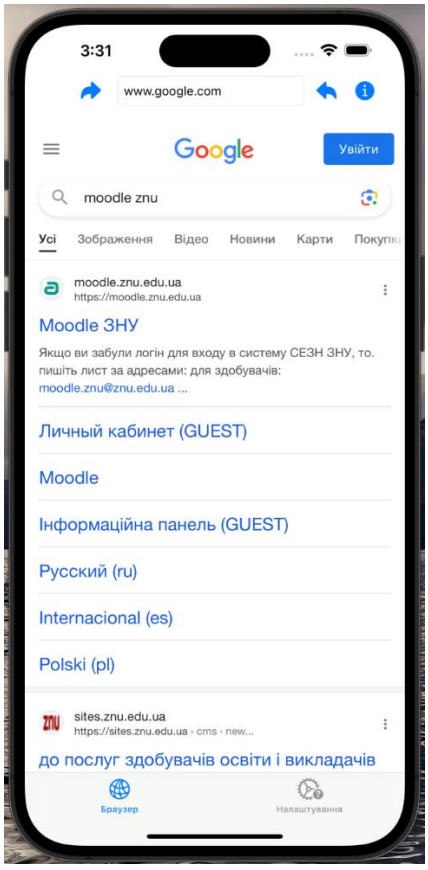

Рисунок 3.18 – Використання функції пошуку

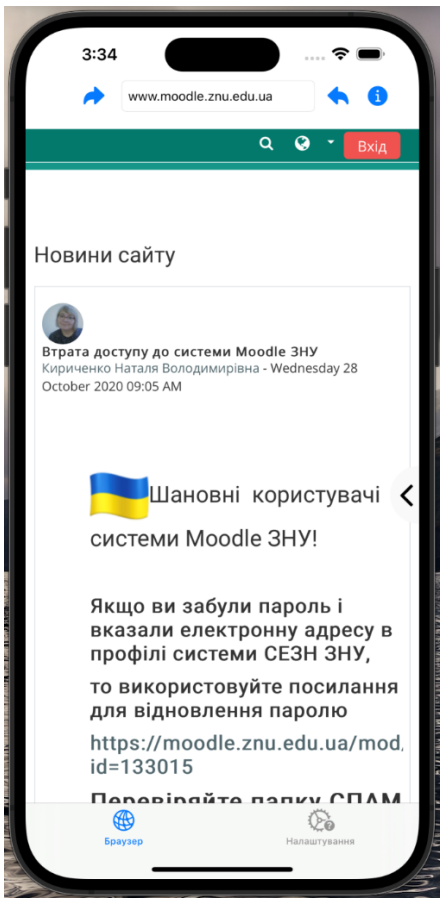

Рисунок 3.19 – Відображення будь-якого сайту

Для зручності користування передбачена можливість використання додаткових функцій веббраузера (див. рис. 3.20).

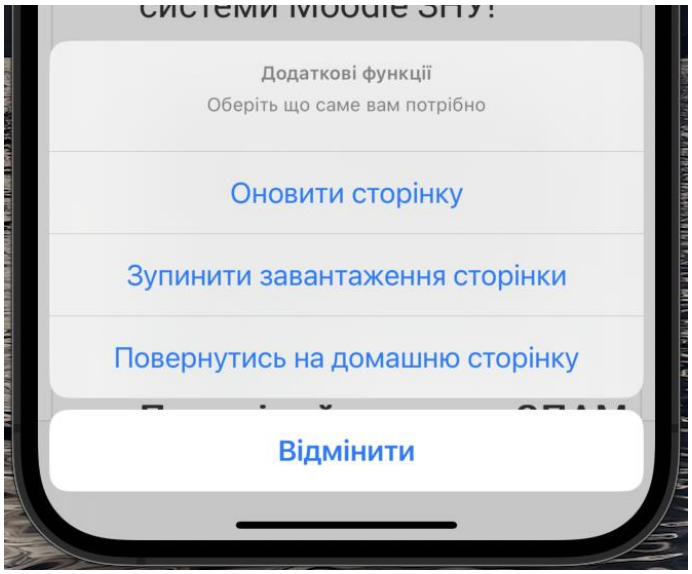

Рисунок 3.20 – Додаткові функції веббраузеру

Також передбачена можливість налаштування веббраузеру. Опціональні можливості такого налаштування представлено на рисунку 3.21.

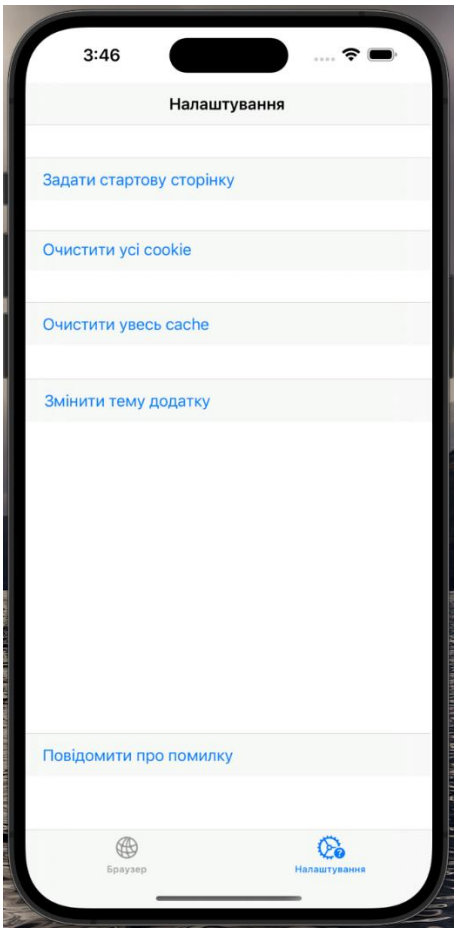

Рисунок 3.21 – Сторінка «Налаштування» веббраузеру

Для прикладу, змінили тему додатка на темну (див. рис. 3.22).

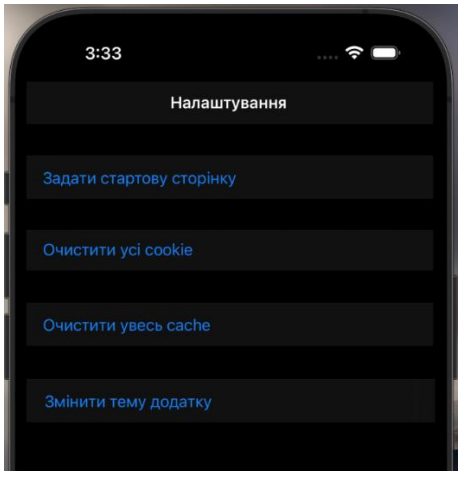

Рисунок 3.22 – Сторінка «Налаштування» у темній темі

Головна сторінка веббраузеру у темній темі виглядає відповідно (див. рис. 3.23).

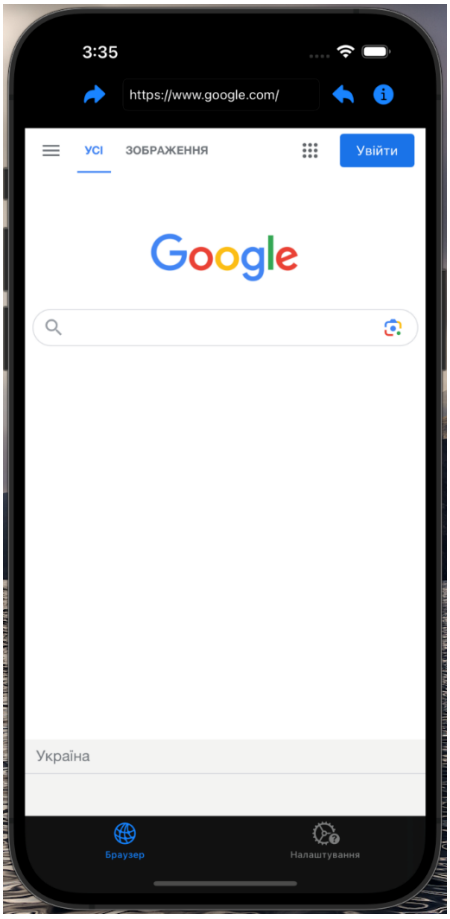

Рисунок 3.23 – Головна сторінка веббраузеру у темній темі

ВИСНОВКИ

У даній дипломній роботі була розроблена веб-браузерна програма для платформи iOS з використанням фреймворку WebKit. Метою роботи було розробити ефективний та функціональний веб-браузер, який забезпечує швидкий та безпечний доступ до веб-сторінок на пристроях iOS.

У процесі роботи були виконані наступні завдання: вивчено основні принципи функціонування веб-браузерів та архітектуру фреймворку WebKit, проведено аналіз існуючих рішень на ринку та визначено основні функціональні вимоги до розроблюваного додатку.

На основі здобутих знань та аналізу була розроблена архітектура програмного продукту. Для реалізації веб-браузера були використані основні можливості фреймворку WebKit, зокрема, його WebView-компонент. Дизайн інтерфейсу браузера був створений з урахуванням сучасних тенденцій та зручності використання.

У результаті роботи був розроблений функціональний веб-браузер для платформи iOS, який дозволяє користувачам швидко та зручно переглядати веб-сторінки, використовуючи усі можливості, що надає фреймворк WebKit.

У процесі тестування розробленого додатку було підтверджено його працездатність та відповідність функціональним вимогам. Веб-браузер показав високу продуктивність, стабільність та швидкість роботи під час перегляду різних веб-сторінок.

Отже, розроблений веб-браузер для платформи iOS на основі фреймворку WebKit є ефективним інструментом для користувачів, які шукають надійний та функціональний спосіб перегляду веб-сторінок на своїх пристроях iOS. Результати даної роботи можуть бути використані в подальших дослідженнях і розробках у сфері мобільних додатків та веб-браузерів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Neuburg M. iOS 12 Programming Fundamentals with Swift. Sebastopol : «O'Reilly Media», 2018. 566 p.
2. Maskrey M. K. Beginning iPhone Development with Swift 4. Colorado : «Parker», 2017. 564 p.
3. Gray A. Swift Swift Pocket Reference. Kanada : «O'Reilly Media», 2015. 186 p.
4. Yamacli S. Beginners Guide to iOS 12 App Development Using Swift 4. Manchester : «Manchester Academic Publishers», 2018. 216 p.
5. Feiler J. Learn Computer Science with Swift. New York : «Plattsburgh», 2018. 309 p.
6. Eguiluz M. A. Mastering iOS 14 Programming. Mumbai : «Packt Publishing», 2021. 528 p.
7. The Swift Programming Language. Swift.org. URL: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/ (дата звернення: 20.04.2023).
8. Designing for iOS. Apple Developer. URL: https://developer.apple.com/design/human-interface-guidelines/designing-for-ios (дата звернення: 25.04.2023).
9. Framework WebKit. Apple Developer. URL: https://developer.apple.com/documentation/webkit (дата звернення: 27.04.2023).
10. Web Inspector Reference. WebKit.org. URL: https://webkit.org/web-inspector/ (дата звернення: 27.04.2023).

ДОДАТОК А

Вміст файлу ViewController.swift

```
import UIKit
import WebKit

class ViewController: UIViewController, WKNavigationDelegate {

    @IBOutlet weak var addressBar: UITextField!
    @IBOutlet weak var webView: WKWebView!

    var webViewURLObserver: NSKeyValueObservation?

    override func viewDidLoad() {
        super.viewDidLoad()
        loadStartPage()

        webViewURLObserver = webView.observe(\.url, options: .new) { webview, change in
            self.addressBar.text = String(describing: change.newValue!!)
        }
    }

    func loadStartPage() {
        let url = URL(string: startPageURL)
        let request = URLRequest(url: url!)
        webView.load(request)
        addressBar.text = startPageURL
    }

    @IBAction func pressForward(_ sender: UIButton) {
        webView.goForward()
    }
```

```
        @IBAction func pressBack(_ sender: UIButton) {
            webView.goBack()
        }

        @IBAction func pressOtherFunc(_ sender: UIButton) {

        let alert = UIAlertController(title: "Додаткові функції", message: "Оберіть що саме вам потрібно", preferredStyle: .actionSheet)
        alert.addAction(UIAlertAction(title: "Відмінити", style: .cancel, handler: { (action) in print ("Відмінити")
        }))
            alert.addAction(UIAlertAction(title: "Оновити сторінку", style: .default, handler: { _ in
            self.webView.reload()
        }))
            alert.addAction(UIAlertAction(title: "Зупинити завантаження сторінки", style: .default, handler: { _ in
            self.webView.stopLoading()
        }))
            alert.addAction(UIAlertAction(title: "Повернутись на домашню сторінку", style: .default, handler: { _ in
                self.loadStartPage()
        }))

        self.present(alert, animated: true)

        }
    }
```

ДОДАТОК Б

Вміст файлу SettingsController.swift

```
var startPageURL = "https://www.google.com/"

import UIKit
import WebKit

class SettingsController: UIViewController {

   override func viewDidLoad() {
      super.viewDidLoad()
   }

   @IBAction func pressSetStartPage(_ sender: UIBarButtonItem) {

        let alertController = UIAlertController(title: "Задати стартову сторінку", message: "Вкажіть URL сторінки", preferredStyle: .alert)

        alertController.addTextField { textField in

          textField.placeholder = "google.com"
           textField.keyboardType = .URL
          textField.autocorrectionType = .no
        }

        let okAction = UIAlertAction(title: "Задати", style: .default) {
          action in guard let textField = alertController.textFields?.first,
          let text = textField.text else {
            print("Ви не задали сторінку")
            return
          }
```

```
                startPageURL = ("https://") + text
            }

            let cancelAction = UIAlertAction(title: "Відмінити", style: .cancel, handler: nil)

            alertController.addAction(okAction)
            alertController.addAction(cancelAction)
            present(alertController, animated: true, completion: nil)

        }

        @IBAction func pressClearCookie(_ sender: UIBarButtonItem) {

            HTTPCookieStorage.shared.removeCookies(since: Date.distantPast)
            print("[WebCacheCleaner] All cookies deleted")

            WKWebsiteDataStore.default().fetchDataRecords(ofTypes:
WKWebsiteDataStore.allWebsiteDataTypes()) { records in
                records.forEach { record in
                    WKWebsiteDataStore.default().removeData(ofTypes: record.dataTypes, for:
[record], completionHandler: {})
                    print("[WebCacheCleaner] Record \(record) deleted")
                }
            }
        }

        @IBAction func pressClearCache(_ sender: UIBarButtonItem) {

            WKWebsiteDataStore.default().removeData(ofTypes:
[WKWebsiteDataTypeDiskCache, WKWebsiteDataTypeMemoryCache], modifiedSince:
Date(timeIntervalSince1970: 0), completionHandler:{ })

        }

        @IBAction func pressChangeTheme(_ sender: UIBarButtonItem) {
```

```
        UIApplication.shared.windows.forEach { window in
            if window.overrideUserInterfaceStyle == .dark {
                window.overrideUserInterfaceStyle = .light
            }
            else {
                window.overrideUserInterfaceStyle = .dark
            }
        }

    }
}
```