International Journal of Global Research in Science & Technology
ISSN: 2455-3832, Volume No.-6, Issue No-1, Jan-Dec 2020
http://ijgrst.com/index.php/journal/index

Mobile iOS Application Development with Case Study: ReadMeLater

Rikky Malviya[1], Pooran Kumar[2], Pooja Sharma[3], and Nitin Jain[4]

[1,2] B-Tech. Scholar, [3,4] Assistant Professor, Global Institute of Technology, Rajasthan, India

Abstract

This paper is on the mobile application developed in iOS using Swift. The idea of this application is to be able to simplify the process of reading online content more efficiently. Let's say you want to read an article but you don't like ads and mostly you don't want to keep the tabs open in the browser every time you want to visit the article. This seems like a very small step but it becomes frustrating when you want to read novels and so, as they have countless chapters and you want them to be stored offline and read at discretion. This also eliminates the problem which arrives due to having an unstable internet connection.

Keywords: Offline Storage, Read Mode, Text-to-Speech, Web Archive.

I. INTRODUCTION

It is an iOS application which enables the user to save webpages for offline reading. As easy as it sounds, it has various applications and challenges attached to it. For now the main goal of the application is to make the iOS app open the web link using Webkit[1] and store in local storage via WebArchive[2], Userdefaults[3] and Core Data[4]. The application is designed very simply to be user friendly by only keeping an important part of the concept for now and later adding additional features into the application without cramming too much information without legal(Copyright)[5] planning. Currently there are some alternatives to this application but are either paid or don't work properly. We want to make this application free by only using Open Source[6] packages and furthermore there is an extensive case study done for this project to curate the application for the intended people and make their experience better. The application really shines when there is no internet connection or you want to save articles to refer for later use. The saved pages are available for view even when the original site is down or the application is uninstalled as the data is stored in File Storage[7]. We will be following the Waterfall Model of Development for the app to create a working app with required features and later add more features using the feedback the previous app provides.

II. LITERATURE REVIEW

A. Apple XCode

Xcode[8] is Apple's integrated development environment (IDE) for macOS, used to develop applications for apple products like iOS, tvOS, macOS, iPadOS, and watchOS. It was first released in 2003; the latest stable release is version 12.5(12E262), released on April 26, 2021, and is available via the Official Apple XCode website and macOS app store. We are using XCode Version 12.3 for the project.

Now Apple XCode 13 has been released with many features like XCode Cloud that help in collaboration with peers on platforms like Github, Gitlab, and Bitbucket. Now we can directly use features like initiate, comment and review and merge pull requests directly from XCode.

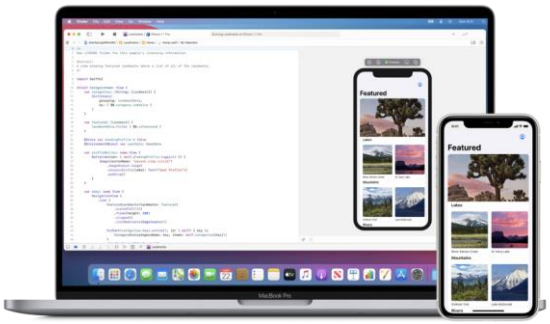

Fig. 1: Apple XCode

B. Framework: WebKit, UIKit

WebKit is goto Web Engine for iOS and macOS app development whenever a Browser -like application is needed to be developed.

UIKit

UIKit[9] is a lightweight and modular front-end framework for developing fast and powerful web interfaces.

WebKit

WebKit is a fast and Open-Source[6] Web Engine developed by Apple's Developer and is used by many popular applications on macOS, iOS, and Linux like app store, safari, etc.

C. Dependency Manager: Cocoapad, Swift Package Manager

What is a Dependency Manager?

It is an automated fashion of resolving packages and their intermediary library used in development. For instance if you try to traditionally install a package "A" which imports classes of package "B"; you won't be able to use it as only package "A" will be installed. That's why Dependency Manager is used to automatically install additional packages. There are Cocoapod[10], and Swift Package Manager[11] for managing packages in iOS app development when using Swift[12] language.

D. Programming Language: Swift 5.3

Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. and the Open-Source Contributor. First released on 2 June, at Apple's 2014 Worldwide Developers Conference (WWDC), Swift was developed as a replacement for Objective-C, as Objective-C hasn't made much progress since the 1980s and can't provide modern features required to create modern iOS, and macOS applications. The current stable version is 5.4.1 released on 25 May, 2021. Swift combines powerful type inference and pattern matching with a modern, lightweight syntax, allowing complex ideas to be expressed in better manner. As a result, code is easy to read, write and maintain. We are using Swift 5.3 version for the project.

E. User Interface: SwiftUI

SwiftUI[13] helps you build responsive, and awesome-looking apps across all Apple platforms with the power of Swift— SwiftUI makes the code responsive so it enables different views according to the platform. SwiftUI, also offers various tools and APIs to provide better user experiences to all users, on any Apple products.

F. HTML Parser: Fuzi

Fuzi[15] is a fast & lightweight XML & HTML parser in Swift with XPath & CSS support. Fuzi is based on a Swift port of Mattt Thompson's Ono, using most of its low level implementations with moderate class & interface redesign following standard Swift conventions, along with several bug fixes.

III. METHODOLOGY

We used the Waterfall Methodology for this project. The main reason to use Iteration Methodology is to first make MVP(Minimum Value Product) and then use the feedback provided by it to make a better version of the application.

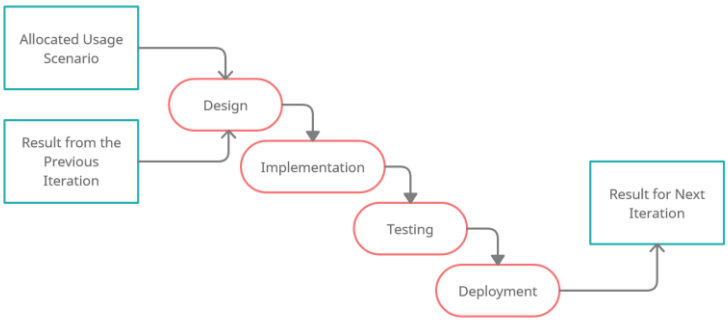

Fig. 2: Iteration Workflow

A. DFD and CFD

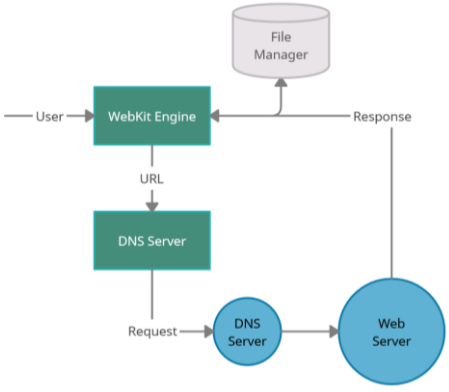

Fig. 3: Data Flow Diagram

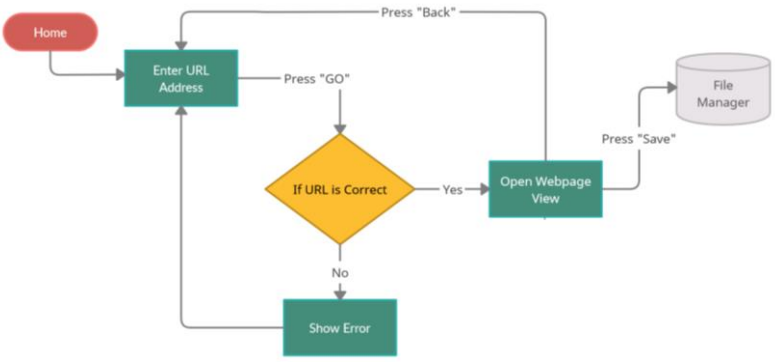

Fig. 4: Control Flow Diagram for Saving Web Pages

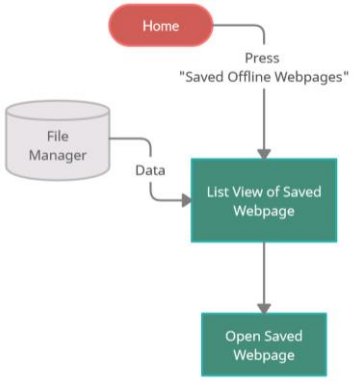

Fig. 5: Control Flow Diagram for viewing Saved Web Pages

B. App Component

1. Home Pages

Paste the URL on the input tab and press “GO”.

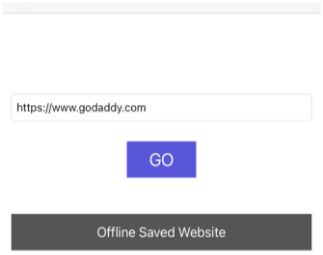

Fig. 6: Home Page

2. WebKit Web Viewer

We are using the WebKit Engine to view the webpage. Now there is an option to either go back to the home page and enter a different url or save the current webpage.

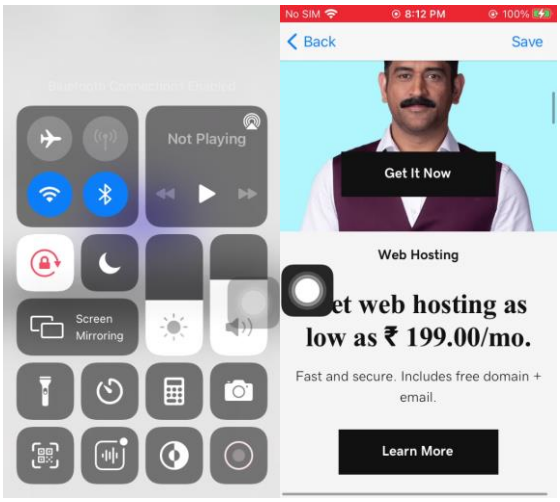

Fig. 7: Web View

3. *Save*

On the Web View you can Click on Save Button to save the Web Page. Saving the current page will download the page locally and add it to the Offline Saved Website page which the user can navigate from the home page.

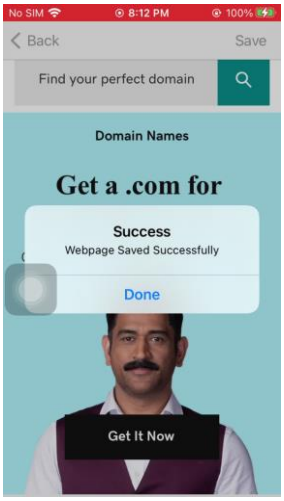

Fig. 8: Successfully Saved

4. *Offline Saved WebSite*

Firstly Disconnect the mobile from the internet. Then click on Offline Saved Website Button from the Home Page. In the Offline Saved Website you will be able to see the list view of all the saved websites and articles.

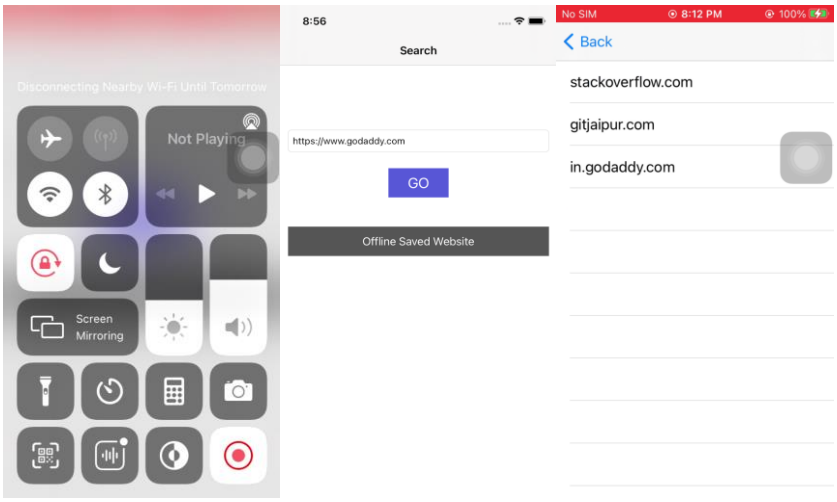

Fig. 9: Offline Saved Website List

5. *Saved Pages*

In the list you can now open any of the websites and read it whenever you want. The webpage saved persists even when you kill the app. And this is because the webpage is stored as an html document with all attached files in the File Manager with the in-built feature provided by Swift.

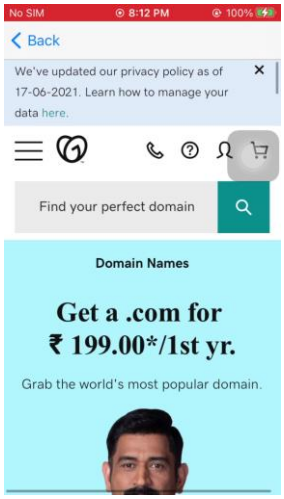

Fig. 10: Saved Web Page

IV. CASE STUDY AND SURVEY

A. Problem Statement

Most Often we find ourselves in a situation where the internet is not stable or we find an article but don't have enough time to view the article and want to save them offline. so, we can read it in the airplane or at our discretion. Some people also like to multitask and do their chores while listening to music or in this case listening to the article. But is this feature popular and how many users may want this feature. The problem with Text-to-Speech is that it doesn't offer much control.

B. Drawback:

As case study pointed out, the default Text-to-Speech feature only works per document and we want it to work

like a playlist with features such as queue, speed of speech.

C. Solution

To resolve the above problem we will be using a Podcast API that sends text to a Server then takes mp3 format audio of that text and thus controlling the mp3 format and implementing the above task becomes easy.

D. Survey Conclusion

We found out that this application is very niche based and not all people found it useful as they mainly good portion of people don't read articles; which is not at all bad news as we also found out that people who work in professional industry often do require to read lot of articles and may want the possible solution that our application may provide.

People also added to what other features can be added like:

1. Rating of an Article.
2. Suggestion Feed of Related Article.
3. To be able to share users saved lists to other users.
4. Subject of Interest for Recommended Feed.

V. DISCUSSION AND FUTURE WORK

A. Better User Interface and User Experience

Everyone knows how important it is to provide great User Experience, for the user to keep using the application. From the user's perspective only two things matter about the application: how it looks and how it feels. Users don't care how great of a technology or what type of complex structure is used to give the final product. They only care about look and feel. That is the reason why one of the main goals in the future is to make the application provide a great user interface(UI) and user experience(UX).

B. Additional Features

Future Goals:

1. Adding Translation Feature for the saved article when online.
2. Able to create a podcast out of articles to be able to listen using some human-like voices rather than default robotic voices.
3. Scroll on a saved article to be able to follow where the text-to-speech is at.
4. Give recommendations or related articles with the saved pages when online.
5. Checking Website Certification to determine if web pages can be saved offline or not.
6. Creating an Android, and Web Application and being able to sync data between them.

C. Future Challenges

There are three challenges our team foresight in completion of project:

1. Translation Feature to be implemented as all the good translation APIs are paid and free Translation API is not good enough.
2. As not being an expert, we can't really tell if we are breaking any law when saving online web pages offline as it can generate copyright infringement. We have done some research and found out that it's

actually ok to save websites available on public domain. So, we have to make security changes so users aren't able to store websites that have strict rules or are out of public domain. We can achieve this by checking the Website Certificates but as of now we don't know how to do it.

3. Having a Human-like Voice for Text-to-Speech.

VI. CONCLUSION

We have created the iOS application which provides the user to save a website or article for offline storage and be able to read them at discretion or at an uncomfortable time when the internet is not available. We are very happy that we are able to contribute to the idea of how we consume online content and may possibly allow people to be given alternative options for how they choose to read an article. Currently the application provides basic features and has much more to improve where in the field of User Interface and User Experience and possibly implement additional features.

ACKNOWLEDGMENT

It brings us great pleasure in writing this paper. Firstly, We would additionally like to show our sincere gratitude toward our guide "Ms. Pooja Sharma" and "Mr. Nitin Jain" and all the supporting personnel of the Department of Computer Science & Engineering, (GIT, Jaipur) who provided us with their experience and expertise. We would also like to thank Mr. S. S. Shekhawat for allowing us to move forward with this idea for our project. Also, a great and well-deserving thanks to Apple and Open Source Community for creating and maintaining these high-end technologies.

REFERENCES

[1] https://developer.apple.com/documentation/webkit

[2] https://developer.apple.com/documentation/webkit/webarchive

[3] https://developer.apple.com/documentation/foundation/userdefaults

[4] https://developer.apple.com/documentation/coredata

[5] https://copyright.gov.in/

[6] https://www.oreilly.com/radar/topics/open-source/

[7] https://developer.apple.com/documentation/foundation/filemanager

[8] https://apps.apple.com/us/app/xcode/id497799835?mt=12

[9] https://getuikit.com/docs/introduction

[10] https://cocoapods.org/

[11] https://swift.org/package-manager/

[12] https://swift.org/blog/swift-5-3-released/

[13] https://developer.apple.com/xcode/swiftui/

[14] https://devopedia.org/dependency-manager

[15] http://cezheng.github.io/Fuzi/