IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

Enhancing App Upgrade Experience in iOS Applications

Madhamsetty Charitha[1], Merin Meleet[2]

Department of ISE, RV College of Engineering, Bengaluru, India[1]

Assistant Professor, Department of ISE, RV College of Engineering, Bengaluru, India[2]

Abstract: Updates for the apps are frequently released nowadays. Therefore, users must have a very smooth experience of updating the app. The aim is to develop an independent SDK that can be easily integrated with any iOS applications. This paper proposes "App Upgrade SDK" that is developed for iOS applications which ensures that the user is notified if there is an update available in the app store and directs the user to the app store if he/she decides to update. In this SDK, three APIs are integrated – one is for checking updates and presenting the user with an update screen, the second is for handling the blocker logic, and the third is for presenting the new features after the app is opened for the first time after updating. These APIs are integrated with app UI using MVVM architecture. MVVM is used to ensure the scalability of the SDK. Fetching the latest version and minimum OS compatibility for the latest version is a challenge. AppUpgrade SDK provides the easiest way to fetch the latest version of the app from Appstore every 24 hours. AppUpgrade SDK is also made customizable according to the developer's needs. This AppUpgrade SDK can be directly integrated with any iOS application as a development pod.

Keywords: AppUpgrade SDK, Software Development Kit (SDK), Protocol, Delegate, View Controller, iOS applications, Model-View-View model (MVVM) architecture.

I. INTRODUCTION

Since the advent of smartphones, the two most widely used mobile operating systems in the world (Android and iOS) have developed a catalogue of millions of applications for each use imaginable. However, building a mobile application is not a once-off procedure. App updates are now necessary to keep them relevant and effective for a long time due to rapidly evolving technology and audience demands. According to a Sensor Tower poll, the average time between updates for popular apps is between 3 and 45 days [10]. So, enhancing app upgrade experience is very much necessary to ensure easy workflow for the user to update the app.

An outdated app may be more vulnerable to security threats. Also, latest version provides more features to the user. Therefore, app developers need to ensure that there are easy and innovative steps involved in updating the app. App Upgrade is the service that provides easy workflow for the users to update the app. This service can either be developed as an API or SDK. In the proposed work, an App Upgrade SDK is developed as SDK is easier to integrate with any app independently. One of the main challenges is to fetch latest version and minimum supported OS version for the latest version of the app. AppUpgrade SDK makes this task easier. The proposed work focuses on iOS applications. AppUpgrade SDK fetches the latest version of the app from the app store every 24 hours. It notifies the user by presenting an update screen every 24 hours if there is an update available. IT also blocks the user from using the app if the user hasn't updated it for a certain number of days. AppUpgrade SDK also provides an option for the developers to present all the features after the app is opened for the first time after updating. The developer must provide all the new features in the app store while releasing the update. The developer has an option to set the following parameters while configuring the SDK.

X - number of days after which the user should be notified of the update from its release date.
Y - number of days after which the user should be blocked from using the app from its release date.
Z - number of outdated versions after the user should be notified of the update (For Example: If the current version of the app is 20.1 and the value of Z is 3, then the user is not notified of the update until 20.4 version is released.)
Update URL – The developer can also provide the update URL i.e., the URL to which the user is directed for updating the app. By default, this URL would be iOS AppStore URL. Fig.2 shows the workflow of AppUpgrade SDK.

To achieve the objectives. MVVM architecture is used. Fig.1 shows the MVVM architecture.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

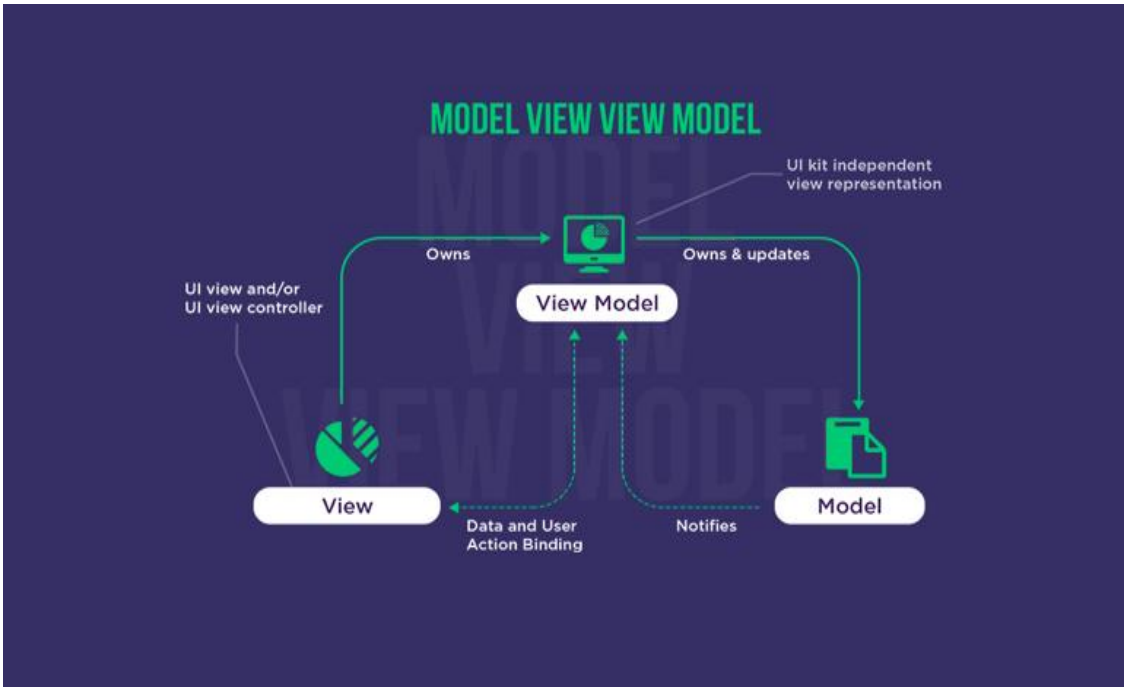

Fig.1. MVVM Architecture. In MVVM, View Model interacts with View and Model, but View (UI) and model (business logic) are independent of each other [15].

II. RELATED WORK

A detailed survey is made on the work done on app upgrade experience. An investigation conducted by the authors in 2020 at San Francisco[1] on the issue of enforcing security critical updates of distributed apps. After completing a survey to learn more about the relationship between released updates and actual use of those updates, they discovered that over half of all users continued to use an outdated app version even seven days after the fix was released. They raised concern for a potential sluggish update procedure on Android and other mobile platforms, although careful generalisations of their preliminary findings and further research are required. The authors of [2] detail recent experiences on the development of an official IEEE mobile application on both Android OS and Apple iOS. They concentrated on video playback because both iOS and Android have built-in standard methods for playing video into their development environments. They were greatly aided in managing some of the difficulties mentioned in "Mobile app backdrop," such as handling various network conditions and having an offline mode, by the existing mechanisms for video playback. In order to do real development for the iOS platform, they got an agreement with Apple.

The writers of [3] proposed a comprehensive methodology to investigate the causes and effects of app updates, and empirically validated it by observing user feedback, update notes, and app rankings for 20 iOS applications. According to the findings, customers' strong opinions will push focus organisations to deliver significant updates regularly and to take into account their valuable comments. The same type of survey was carried out in [4], and the authors came to the conclusion that major updates are caused by users' evolving requirements. The major benefit of app upgrades is an updated version of the app, though some users have reported decreased performance as a result of increased load. Through a case study, the authors of [5] analysed third-party app stores and devised a statistic called "Pulling rate" to calculate the impact of update events on downloads. According to the findings, the number of downloads will nearly triple ten days following the upgrade.

The writers of [6] begin by providing a summary of the many development methodologies used in the realm of e-Commerce Andriod and iOS applications. Their study focuses on analysing how well-built e-Business mobile apps perform in terms of interoperability, access to mobile functionalities, and enhanced visuals. A unique method is presented by the authors of [7] for automatically identifying the portions of Android apps that have been impacted by updates to the underlying Android mobile operating system and statistically analysing the impact of the update. An operating system upgrade has little overall impact, according to preliminary analysis.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

III. METHODOLOGY

The objectives are formulated as two separate tasks i.e. development of AppUpgrade SDK and testing the SDK by integrating with any sample app. Figure 2 shows the detailed design of the SDK

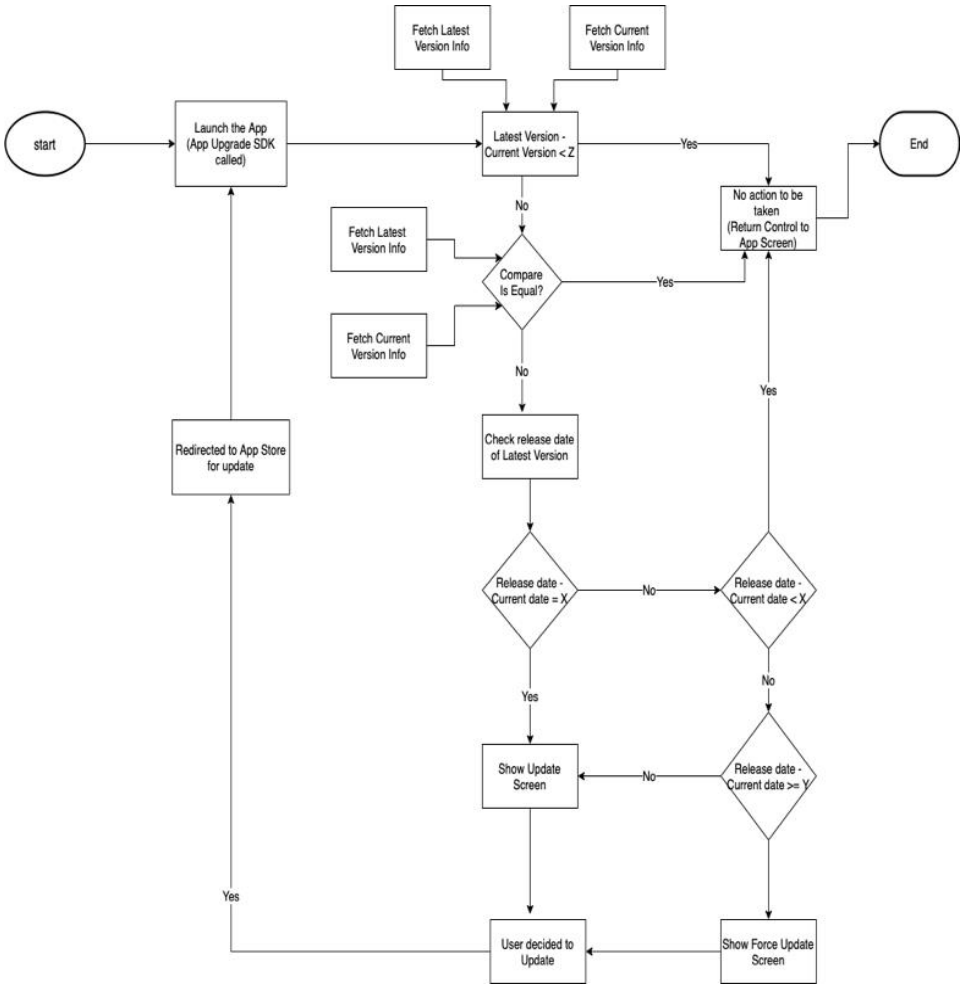

Fig.2. Workflow of AppUpgrade SDK. All the three APIs are integrated to perform the functions shown in the figure.

A. Development of AppUpgrade SDK:

1) Fetching latest version and current version: Latest version is fetched from App Store. App's current version is fetched using iTunes API and bundle id [17].

2) Developing "Update Controller" API : "Upgrade Controller" functionalities are

- Checks if the current version is z versions outdated.
- Checks if the number of days from the release of the new version exceeded x days. This is done by comparing current date and release date. If yes, displays an update screen for every 24 hours.
- Allows the user to directly jump on to the App Store if the user decides to update.

3) Developing "Force Update" API : This API when called

- Checks if the number of days from the release of the new version exceeds y days. (x and y values are fetched from policy provider - daysAfterPolicyTrigger, daysAfterForceUpdate). If yes, block the user from using the app.
- Checks if the OS version is not compatible by fetching the minimum OS support version from the App store using AppStoreConnect API. If yes, don't force the update.

4) Developing "WhatsNew" API : An API when called checks if current version is the latest version and it is getting stored for the first time. If yes, it fetches new features from the app store (must be provided by the developer while releasing the latest version) and displays them.

5) Integration of APIs, Controller, User Interface and Database : All the components are integrated using MVVM architecture.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

B. Integration of AppUpgradeSDK into sample app:

The AppUpgradeSDK is integrated into sample app as a development pod. SDK is consumed as a pod for internal apps (apps used in developer's organization). SDK provides a framework for external customers.

C. Protocols: The protocols used in the development of AppUpgrade SDK are:

1) Configuration Provider: This protocol mainly exposes four settings that can be configured by the application:

- bundleid - The bundleId represents the app's unique identifier that developer can register, modify, and delete.
- app version - Current version of the app. It is fetched from iTunes API.
- primaryColor - Optional UIColor object used as the primary button color for all the screens shown by the application.
- updateURL - Optional appStore URL pointing to the current application. This is used as the action URL when user opts to update the application. By default this URL is taken from the iTunesAPI.

2) Policies Provider: Protocol that defines the admin policy on when user should be informed on the new available version and to block the usage of an older version of the application. As of now there is no functionality to block the usage. And user is informed of the update if update is available immediately.

3) Storage Provider: AppUpgradeSDK also provides an option to save any data with the help of StorageProvider. This is an optional protocol. By default, it saves the latest version information fetched from iTunesAPI and the date when it was last saved. The keys used for them are : VersionInformation and InfoLastFetchedDate respectively. Applications can provide an object that conforms to StorageProvider protocol if they have a custom storage requirement. It mainly provides the following three APIs which are used to set, get and clear any data from storage.

D. Delegates: The delegates used in the development of AppUpgrade SDK are:

1) Upgrade Delegate: AppUpgradeSDK also provides an option to the host app to know the status of the Update Check. This is also an optional protocol. Application can provide an object that conforms to Upgrade Delegate protocol if they have requirement.

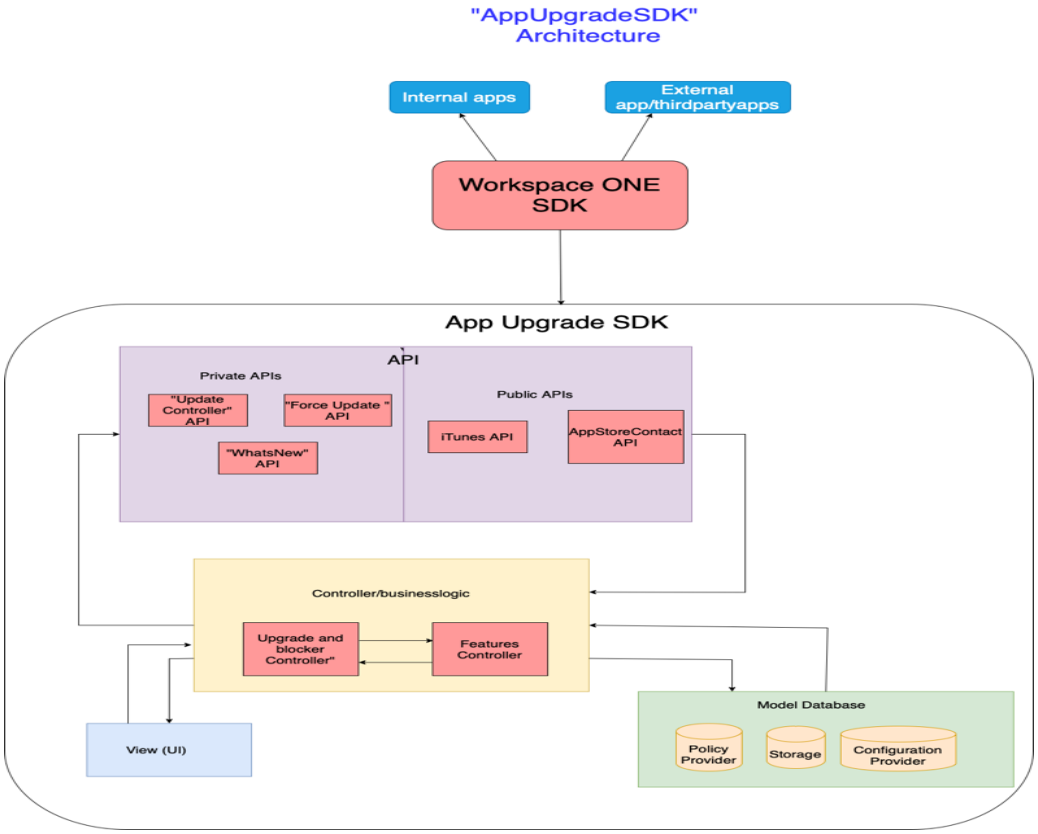

Fig.3. Detailed design of the AppUpgrade SDK methodology followed during the development. It has two public APIs which are used in the development and three private APIs which are developed using public APIs.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022
DOI: 10.17148/IJARCCE.2022.11782

2) Custom Screen Delegate: Developers can choose between default screen and custom screen for update screen. If they choose to provide a custom screen, then they have to implement this protocol so that they can provide their own Custom Update Screen in showUpdateScreen function. Also the App Upgrade SDK sends the Default Update Screen object which allows them to make some changes to the default update screen if it is required.

3) Feature Delegate : AppUpgradeSDK also provides an option to the host app to know the status of the What's New. This is also an optional protocol. Application can provide an object that conforms to Feature Delegate protocol if they have a requirement. It provides the following API which can be used to get the status of the SDK.

E. View Controllers: The view controllers used in the development of AppUpgrade SDK.

1) Upgrade Controller : It provides the required API to check for the availability of new version of the application and to show it to the user. Policy Provider and Storage Provider are optional. It requires:

- Configuration Provider object - provides the required configuration for the AppUpgradeSDK.
- Custom Screen Delegate object - Current app screen visible when checkUpdate API is invoked.

2) Feature Controller : It provides the required API to view the new features added in the new version of the application and to show them to the user. Configuration Provider and StorageProvider are optional protocols for this feature. Applications can provide an object that conforms to them if they have a custom configuration or storage requirement. For this feature, SDK user can configure 'primaryColor' and 'appVersion' with the help of Configuration Provider. This usage requires minimum of two parameters.

- listOfFeatures - Developer needs to provide the list of the features. It includes title and description of the feature where description is optional to provide.
- presenter - Current app screen when viewFeatures API is invoked. AppUpgrade SDK's What's new screen will be presented modally on top of this screen. Application is responsible for making sure that the presenter is the top most screen available.

All the protocols are used for storage. All the delegates are the building blocks of APIs. View controllers handle the logic in the user interface.

IV. RESULTS AND ANALYSIS

The following were the results obtained on successful development of AppUpgradeSDK and integration with a sample app.

a. Current version and latest version are fetched accurately.
b. Update screen is presented if there is an update available in the app store every 24 hours.
c. Blocker screen is presented if the user has not updated for a certain number of days.
d. The minimum supported OS version is fetched appropriately. Also, the user is not mandated to update if there is OS version incompatibility with the latest version of the app.
e. New features of the latest version of the app are fetched appropriately. New features are presented to the user when the app is opened for the first time after updating.

A. Presentation of Update Screen:

Fig.4 shows the update which is presented for every 24 hours if an update is available. The app is configured with a version less than the current version to check if SDK presented the update screen. Update screen is presented with all the version information within seconds after opening the app. It means that the current version and latest version are fetched and compared within seconds. This process is performed every 24 hours. When the update button is clicked, the user is directed to the app store where the user can update the app. The user need not search for the app in the app store, it is directly opened once clicked on the update button.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

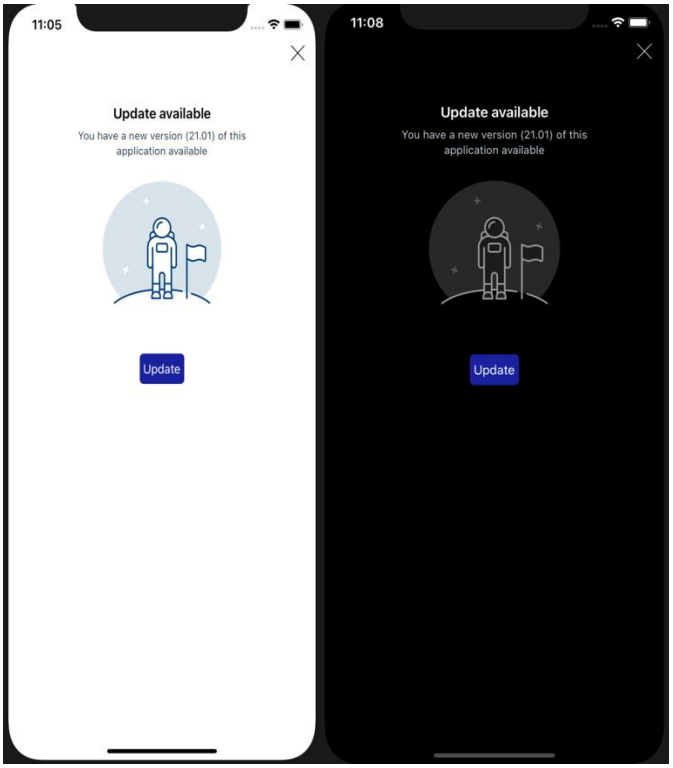

Fig.4. Update Screen in light and dark modes. Update screen contain an update button which on clicked directs the user to the app store. It also contains close button at the top right corner which means that the screen is dismissible.

B. Presentation of Blocker Screen:

Fig.5 shows the blocker screen both in light mode and dark mode. Blocker screen is presented since the app is let un updated for 2 hours (configured by the policy provider with value of y as 2 hours for testing purpose). Blocker screen does not have a close button to dismiss the screen. This makes the user not to use the app further until it is updated. So that user is mandated to update the app.

C. Presentation of WhatsNew Screen:

Fig.6 shows the "WhatsNew" Screen both in light and dark mode. It contains all the features in the latest version of the app when the app is opened for the first time after opening the app. The new features of the latest version of the app are fetched from the app store. For testing purposes, the new features are provided in the storage itself.

The AppUpgrade SDK has definitely shown good results. It is able to function appropriately in different situations. This has added a new feature which has enhanced the app upgrade experience in iOS applications. Before this feature was integrated, the process of releasing the new updates and letting the user know about the updates was a cumbersome task. Now, AppUpgrade SDK, which can be integrated with any iOS application independently, has definitely made the process of updating the app easier and can be customized according to developer's needs. The time taken for fetching the latest version, minimum supported OS version and new features is very small(1 to 2 seconds). AppUpgrade SDK presented the update screen and blocker screen appropriately during different situations. Also, when integrated with sample app, it is triggered at the appropriate time while the application is running. The SDK has not affected the performance of the app in any way as it is made to run in the background.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

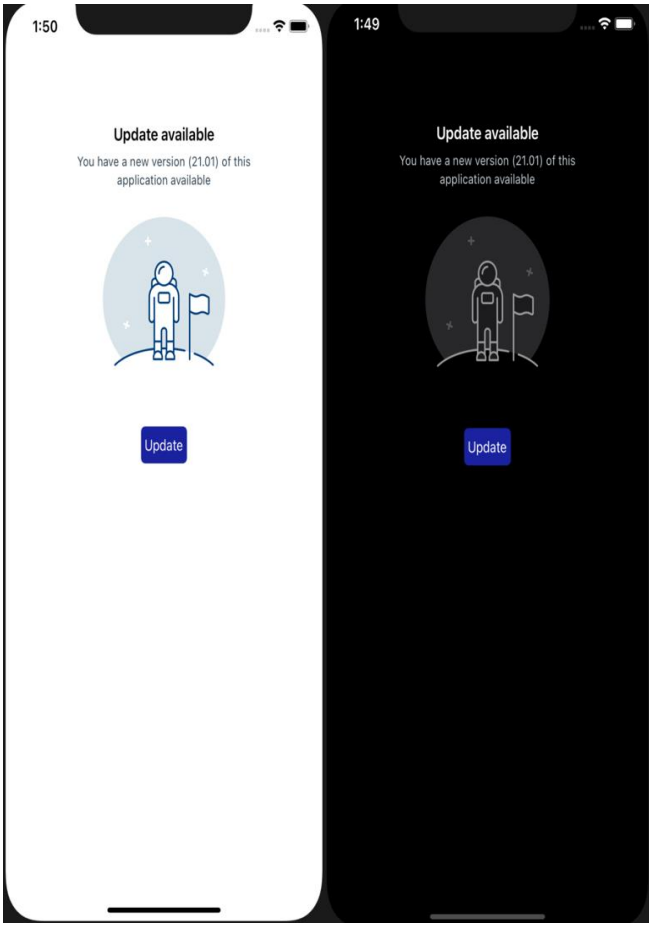

Fig.5. Blocker screen in dark and light modes. Blocker screen does not contain close button which means that it is not dismissible. User must update the app if the blocker screen is presented otherwise it is not possible for the user to use the app.

V. CONCLUSION

AppUpgrade SDK is a new feature added to iOS applications which eases the process of updating an app. It can be integrated with any iOS application very easily. So, the AppUpgrade SDK is scalable, flexible and portable. The objectives that are achieved are fetching current and latest version in the fastest way, presenting update screen and blocker screen appropriately and presenting all the new features of the latest version of the app appropriately. AppUpgrade SDK is designed in a way that the developer can configure it according to his/her requirements. This way SDK has provided an easy way to update the app for the users and also provided flexibility to the developers.

Further, AppUpgrade SDK can be developed for android applications also. It can be made more customizable by providing an option for the developers to select the presenter. When there is OS version incompatibility, users can be warned about this. Rather than fetching the minimum supported OS version from the public API, a more secure and fast way can be found.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

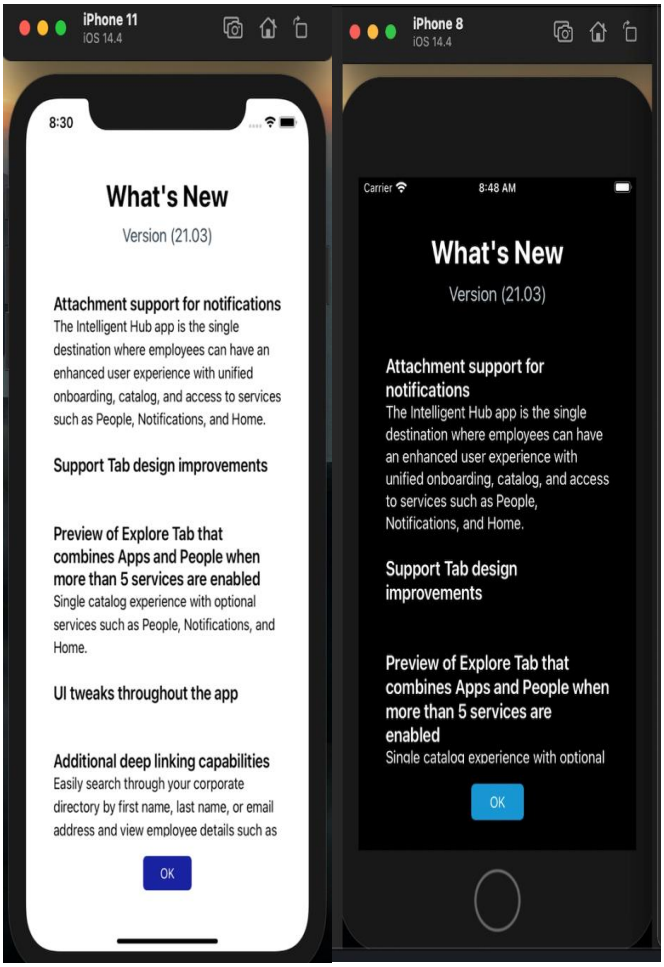

Fig.6. What's New screen in light and dark modes. It contains all the new features that are added in the latest version of the app and this screen is presented when the app is opened for the first time after updating.

ACKNOWLEDGMENT

This work would not have been possible without the contribution of **Prof. Merin Meleet,** Assistant Professor – RV College of Engineering and **Dr. B.M. Sagar,** Head of the Department, Department of Information Science and Engineering – RV College of Engineering. I am very thankful to all the guides and mentors for all the help they provided throughout the course of this work.

REFERENCES

[1] Andreas Moller; Stefan Diewald; Luis Roalter; Florian Michahelles, "Update Behaviour in App Markets and Security Implications: A Case Study in Google Play", 3rd International Workshop on Research in the Large, September 2020.

[2] Kim W. Tracy, "Mobile Application Development Experiences on Apple's iOS and Android iOS", IEEE Potentials, 2021, DOI: 10.1109/MPOT.2021.2182571.

[3] Hengqi Tian; Jing Zhao, "Antecedents and Consequences of App Update: An Integrated Research Framework", Conference on Mobile HCI, August 2019, DOI: 10.1007/978-3-319-99936-4_6.

[4] W. Liu; Y. Zou; Y. Yang; W. Cheng; G. Zhang, "How app update affects app download in iOS Appstore", International Conference on Computer and Communications, August 2018.

[5] Yongkang Xing, "Research and Analysis of the Development Approaches in E-Business Mobile Applications", International conference on Computer Science and Education Technology (CSET 2018).

[6] Ronald Valledor; Gomesaria Pal, "Good Mobile App Based eCommerce Application", Conference on Environmental Engineering, Science and Sustainability, April 2019, DOI: 10.17605?OSF.IO/6BSW5.

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11782

[7] Guowei Yang; Jeffrey Jones; Austin Moninger; Meiru Che, "How Do Android Operating System Updates Impact Apps?", Vol.3, International Conference on Mobile Software Engineering and Systems, September 2017.

[8] Feorderer.J.; Heinzl.A, "Product updates: attracting new consumers versus alienating existing ones" Thirty Eighth International Conference on Information Systems 2017.

[9] Boudreau, K.J, "Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation.", International Conference on Mobile HCI, September 2018.

[10] Comino, S; Manenti, F.M; Mariuzzo, "Updates management in mobile applications. Itunes Vs Google Play", Conference on Technical Management and Analysis" Malaysia 37(4), 354–356 (2015).

[11] Aidean Sharghi; Jacob S. Laurel; Boqing Gong, "Query-Focused Update Evaluation: Dataset Evaluation, and A Memory Network BasedApproach", IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[12] Varun Luthra; Jayanta Basak, "Joint Pulling rate Approach on finding the impact of Updates on Apps", International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP), 2008.

[13] Kaiyang Zhou;Yu Qiao; Tao Xiang, "Deep-Learning Methods to Recognize the Area of Impact of Updates"; The Thirty - Second AAAI Conference on Artificial Intelligence(AAAI-18), 2018.

[14] Prof.Santanu Chaudhury; K.A.N. Jyothi, "Fresh Apps: Mostly Updated Apps on Google Play Store", International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP), 2008.

[15] A.H Bihari, "Introduction to Model-View-Viewmodel Architecture", https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/

[16] Arun Kumari, B.H Arun Kumar, "Fetching Latest version from iTunes API", https://www.programmableweb.com/api/itunes-app-store-search-rest-api.

[17] Guzman, E; El-Haliby, M.; Bruegge, B, "Ensemble Methods for App Review Classification: An Approach for Software Evolution" pp. 771–776 (2015), Journal of Informational Science and Technology.

[18] Apple developer docs, "AppStoreConnect API", https://developer.apple.com/app-store-connect/

[19] Apple Inc, "Swift Documentation", https://www.swift.org/documentation/

[20] Apple Inc, "API Documentation", https://developer.apple.com/documentation/docc/api-reference-syntax