UNIVERSITI TUNKU ABDUL RAHMAN

ASSIGNMENT 6 (10%)

**UECS3263 iOS APPLICATION DEVELOPMENT**

BACHELOR OF SCIENCE (HONOURS) SOFTWARE ENGINEERING

|  |  |
| --- | --- |
| Name (as stated in Student Card) | Student ID |
| Gervin Fung Da Xuen | 1801655 |
| Programme | Submission Date |
| Software Engineering | 8/9/2021 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | | **Total Marks** | **Marks Given** |
| User Interface Design and Audio Files – CO2 (3 marks) | |  |  |
|  | Allows audio file selection | 1 |  |
|  | Display selected audio file name | 1 |  |
|  | Audio files included | 1 |  |
| App Construction and Execution – CO3 (4 marks) | |  |  |
|  | Requirements implemented | 3 |  |
|  | Executes successfully | 1 |  |
| App Design – CO1 (3 marks) | | | |
|  | App logic design (including screenshot of simulator) | 3 |  |
| Total | | 10 |  |

Table of Contents

[**1.** **Screenshot** 3](#_Toc81962617)

[**2.** **Documentation** 5](#_Toc81962618)

[**Application Preferences** 5](#_Toc81962619)

[**Application Localization** 6](#_Toc81962620)

[**3.** **App Taskflow** 8](#_Toc81962621)

[**4.** **Code Listing** 9](#_Toc81962622)

# **Screenshot**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
| Figure 1.0 Bahasa Malaysia | Figure 1.1 English | Figure 1.2 Mandarin | |

The 3 Figures above shows that 3 languages are used for Localization. This application contains 6 songs and user can click next to change to the next song on the list. If the last song is reached, and user click next, they will be directed back to the first song. Likewise, if the user click previous song when they had reached the first song, the will be directed back to the last song. This is done by changing the index as shown by figure below.



Figure 1.3: Update index function

By default when user open the application, no music will be played, the song will be played when either one of the conditions below are fulfilled.

1. User click “play” button to play current song
2. User click “next” button to change to next song
3. User click “prev” button to change to previous song
4. User click on any songs showed on the tableview or the list.

Whenever a song is played, the index of the song will be saved and the reason of doing so will be explain in the next section.

Also, if there is index saved in the Application Preferences, it will use to index to load all the necessary images and songs.

# **Documentation**

The documentation includes 2 sections:

1. Application Preferences
2. Application Localization

## **Application Preferences**

Application Preferences used for this assignment is fairly simple, it only stores the index of the latest song/music chosen as shown by the figure below.



Figure 2.0: Application Preferences of Assignment 6

Instead of saving the song’s name, the song/music path as well as the image path, both 3 attributes contain the same property, that is both can be access from the same index as the attributes are stored in 2 arrays with the same order as shown by the figure below.

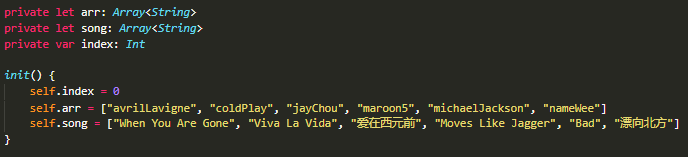


Figure 2.1: Array used to store path and song name

Hence this simplifies the process of retrieving and saving a lot as only 1 attribute are required to be saved and the 3 attributes required can be derived from 1 attribute, the index.

## **Application Localization**

This Application use Localization in 2 ways which are setting the text on Storyboard statically *and* changing text dynamically upon user interaction when user click play/pause and the text change accordingly.

Basically, what the figure shown below does is hard coding the value to each property so that whenever localization changes, it will display the corresponding text. However, this behavior is very limited and is applicable to Storyboard only and thus cannot update itself dynamically. As such, another solution is also adopted.

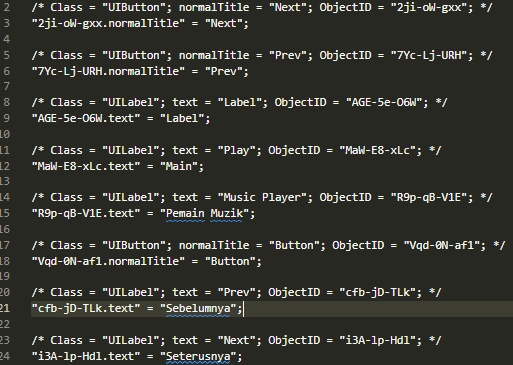


Figure 2.2: Main Storyboard localization Bahasa Malaysia



Figure 2.3: Localizable.strings for Bahasa Malaysia

The Figure 1.3 above shows the setup for changing the text on the application dynamically. Hence whenever there is a string with key “Play”, it will return the value of “Main” for Bahasa Malaysia. The figure below will show the extension made to the String data for this method to work

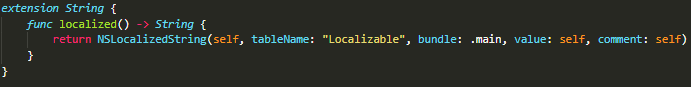


Figure 2.4: Function named *localized*

Basically, self stands for the key or the string that called the function named above and whenever the key is found with value paired together in the Localizable file, it will return the value, otherwise it will return the key itself. Hence this function will work even without Localizable file for Storyboard and can update itself whenever the key is stored in the Localizable file for String.

# **App Taskflow**

Diagram

Description automatically generated

Figure 3.0: Taskflow of the Application

# **Code Listing**

1. import UIKit
2. import AVFoundation
3. final *class* Util {
4. private let arr: *Array*<*String*>
5. private let song: *Array*<*String*>
6. private var index: *Int*
7. *init*() {
8. self.index = 0
9. self.arr = ["avrilLavigne", "coldPlay", "jayChou", "maroon5", "michaelJackson", "nameWee"]
10. self.song = ["When You Are Gone", "Viva La Vida", "爱在西元前", "Moves Like Jagger", "Bad", "漂向北方"]
11. }
13. public *func* updateIndex(index: *Int*) {
14. self.index = index
15. }
16. public *func* incrementCount() {
17. if (self.index == self.arr.count - 1) {
18. self.index = 0
19. } else {
20. self.index += 1
21. }
22. }
23. public *func* decrementCount() {
24. if (self.index == 0) {
25. self.index = self.arr.count - 1
26. } else {
27. self.index -= 1
28. }
29. }
31. public *func* getIndex() -> *Int* {
32. return self.index
33. }
35. public *func* getImageFromIndex(index: *Int*) -> *String* {
36. return "util/\(self.arr[index]).jpg"
37. }
38. public *func* getImagePath() -> *String* {
39. return "util/\(self.arr[self.index]).jpg"
40. }
42. public *func* getSongPathFromIndex(index: *Int*) -> *String* {
43. return "\(self.arr[index])"
44. }
46. public *func* getSongPath() -> *String* {
47. return self.arr[self.index]
48. }
50. public *func* getSongName(index: *Int*) -> *String* {
51. return self.song[index]
52. }
53. public *func* getPlayButtonPath() -> *String* {
54. return "util/play.jpg"
55. }
56. public *func* getPauseButtonPath() -> *String* {
57. return "util/pause.jpg"
58. }
59. public *func* getPrevButtonPath() -> *String* {
60. return "util/prev.jpg"
61. }
62. public *func* getNextButtonPath() -> *String* {
63. return "util/next.jpg"
64. }
66. public *func* getCount() -> *Int* {
67. return arr.count
68. }
69. }
70. final *class* ViewController: UIViewController {
71. private static let IDENTIFIER = "TABLE\_CELL"
72. private static let INDEX = "INDEX"
73. private let util: Util = Util()
74. private var pressPlay = false
75. private var audioPlayer: AVAudioPlayer?
76. @IBOutlet weak var musicImg: UIImageView!
77. @IBOutlet weak var playButton: UIButton!
78. @IBOutlet weak var prevButton: UIButton!
79. @IBOutlet weak var nextButton: UIButton!
80. @IBOutlet weak var tableView: UITableView!
81. @IBOutlet weak var musicLabel: UILabel!
82. @IBOutlet weak var playLabel: UILabel!
84. override *func* viewDidLoad() {
85. super.viewDidLoad()
87. tableView.rowHeight = 60
88. tableView.dataSource = self
89. tableView.delegate = self
91. playButton.setImage(UIImage(named: self.util.getPlayButtonPath()), for: .normal)
92. prevButton.setImage(UIImage(named: self.util.getPrevButtonPath()), for: .normal)
93. nextButton.setImage(UIImage(named: self.util.getNextButtonPath()), for: .normal)
95. let defaults = UserDefaults.standard
96. if let index: *Int* = defaults.integer(forKey: ViewController.INDEX) {
97. pressPlay = true
98. self.setUp(index: index)
99. } else {
100. self.setUp(index: 0)
101. }
102. }
104. private *func* setUp(index: *Int*) {
105. musicImg.image = UIImage(named: self.util.getImageFromIndex(index: index))
106. musicLabel.text = self.util.getSongName(index: index)
107. let audioSelected = Bundle.main.path(forResource: self.util.getSongPathFromIndex(index: index), ofType: "mp3")
108. do {
109. self.audioPlayer = try AVAudioPlayer(contentsOf: URL(fileURLWithPath: audioSelected!))
110. } catch{
111. print(error)
112. }
113. }
114. @IBAction *func* playButtonPressed(\_ *sender*: *Any*) {
115. self.start(play: !pressPlay)
116. }
117. @IBAction *func* nextButtonPressed(\_ *sender*: *Any*) {
118. self.util.incrementCount()
119. musicImg.image = UIImage(named: self.util.getImagePath())
120. musicLabel.text = self.util.getSongName(index: self.util.getIndex())
121. self.start(play: true)
122. }
123. @IBAction *func* prevButtonPressed(\_ *sender*: *Any*) {
124. self.util.decrementCount()
125. musicImg.image = UIImage(named: self.util.getImagePath())
126. musicLabel.text = self.util.getSongName(index: self.util.getIndex())
127. self.start(play: true)
128. }
130. private *func* start(play: *Bool*) {
131. pressPlay = play
132. playLabel.text = (pressPlay ? "Pause" : "Play").localized()
133. let img = pressPlay ? UIImage(named: self.util.getPauseButtonPath()) : UIImage(named: self.util.getPlayButtonPath())
134. playButton.setImage(img, for: .normal)
135. self.playMusic(index: self.util.getIndex())
136. }
138. private *func* playMusic(index: *Int*) {
140. if let audioPlayer = audioPlayer, !pressPlay {
141. audioPlayer.stop()
142. } else {
143. let urlMusic = Bundle.main.path(forResource: self.util.getSongPathFromIndex(index: index), ofType: "mp3")
145. do {
146. try AVAudioSession.sharedInstance().setMode(.default)
147. try AVAudioSession.sharedInstance().setActive(true, options: .notifyOthersOnDeactivation)
149. guard let urlMusic = urlMusic else {
150. return
151. }
153. self.audioPlayer = try AVAudioPlayer(contentsOf: URL(fileURLWithPath: urlMusic))
155. guard let audioPlayer = self.audioPlayer else {
156. return
157. }
159. let defaults = UserDefaults.standard
160. defaults.setValue(index, forKey: ViewController.INDEX)
161. audioPlayer.play()
163. } catch {
164. print("Error playing")
165. }
166. }
167. }
168. }
169. *extension* ViewController: UITableViewDataSource {
170. *func* tableView(\_ *tableView*: UITableView, numberOfRowsInSection *section*: *Int*) -> *Int* {
171. return self.util.getCount()
172. }
174. *func* tableView(\_ *tableView*: UITableView, cellForRowAt *indexPath*: IndexPath) -> UITableViewCell {
175. let cell: UITableViewCell! = tableView.dequeueReusableCell(withIdentifier: ViewController.IDENTIFIER)
177. if cell == nil {
178. let newCell = UITableViewCell(style: UITableViewCell.CellStyle.default, reuseIdentifier: ViewController.IDENTIFIER)
179. newCell.imageView?.image = UIImage(named: self.util.getImageFromIndex(index: indexPath.row))
180. newCell.textLabel!.text = self.util.getSongName(index: indexPath.row)
181. return newCell
182. }
184. cell!.imageView?.image = UIImage(named: self.util.getImageFromIndex(index: indexPath.row))
185. cell!.textLabel!.text = self.util.getSongName(index: indexPath.row)
186. return cell!
187. }
188. }
189. *extension* ViewController: UITableViewDelegate {
190. *func* tableView(\_ *tableView*: UITableView, didSelectRowAt *indexPath*: IndexPath) {
191. musicLabel.text = self.util.getSongName(index: indexPath.row)
192. self.pressPlay = true
193. musicImg.image = UIImage(named: self.util.getImageFromIndex(index: indexPath.row))
194. self.util.updateIndex(index: indexPath.row)
195. tableView.deselectRow(at: indexPath, animated: true)
196. self.start(play: true)
197. }
198. }
199. *extension* *String* {
200. *func* localized() -> *String* {
201. return NSLocalizedString(self, tableName: "Localizable", bundle: .main, value: self, comment: self)
202. }
203. }