**Mobile Computing – iOS Fall’23**

**Assignment03**

**30 Points**

**Please follow the following instructions to complete this assignment.**

1. Open Xcode from the launchpad of your Mac. Clone your private GitHub repository.
2. In that repo, create a new Xcode project. Select the iOS platform and click on the App template.
3. Click on next, which will prompt you to choose options for the project.
4. Provide product name as **LastnameQuizApp**, “**edu.nwmissouri.fall23.cs44643**” for organization identifier, “**Storyboard**” as interface and “**Swift**” as the language.
5. Click on next and select an appropriate location to save your app and click on Create. A project directory will be loaded.
6. From the project navigator click on “Main.storyboard” file, a blank mobile screen will be loaded, where the required fields for an app need to be added.

**View**

Table 1: UI elements configuration

|  |  |  |
| --- | --- | --- |
| **UI element** | **Purpose** | **Outlet/action name** |
| 2 UILabel elements | To display the title “Quiz Time™️” | titleLBL |
| To display the title “Question/Description” | questionLBL |
| 1 UITextField | To check the answer | answerTF |
| 1 UIImageView | To display a picture | pictureIV |
| 1 UITextView | To show picture’s description or question | descTV |
| 1 UISegementControl | To display quiz categories | categorySC  selectCategory: |
| 1 UIStackView | To group UI elements | playQuizSV |
| 3 UIButton elements | 👍- Will be highlighted for correct answer  👎- Will be highlighted for incorrect answer | optionsBtnCLCTN |
| ✅ - To check whether the answer is correct or incorrect | checkBTN  checkAnswer: |

*Note: Names that are ending with a colon (****:****) are actions.*

1. Open library (cmd+shift+l). Search for “label”, add (drag and drop) it to the storyboard. Give text as Quiz Time™️. In Xcode menu bar, see Edit > Emoji & Symbols to find special Unicode characters.
   1. Enable user interaction for the label.

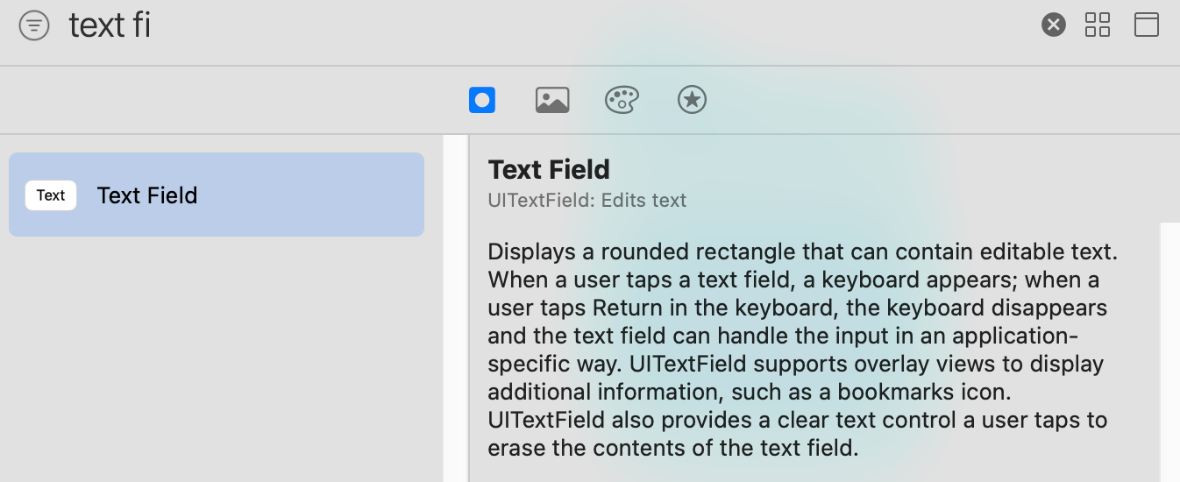


Figure Xcode Library

1. Add a segmented control to the storyboard to display categories. There should be a total of 5 categories, namely, General, Swift, Gadgets, and 2 other categories of your choice.
2. Add an image view to the storyboard. Set its default image property to face.smiling in its Attributes Inspector.
   1. Enable user interaction for the image view.
3. Add a label to the storyboard. Change its text to Question/Description. Align it center.
4. Add a text view to the storyboard. Uncheck its Editable property under the Behavior group in its Attributes Inspector. Also, change its alignment to Justified.
5. Add a text field to the storyboard. Give its placeholder value as Type your answer.
6. Add a filled button to the storyboard. Change the title of the button to ✅.
7. Add 2 tinted buttons to the app and give like (👍) and dislike (👎) emojis as their titles, respectively.
8. Now, all the required UI elements for the app are added to the storyboard.

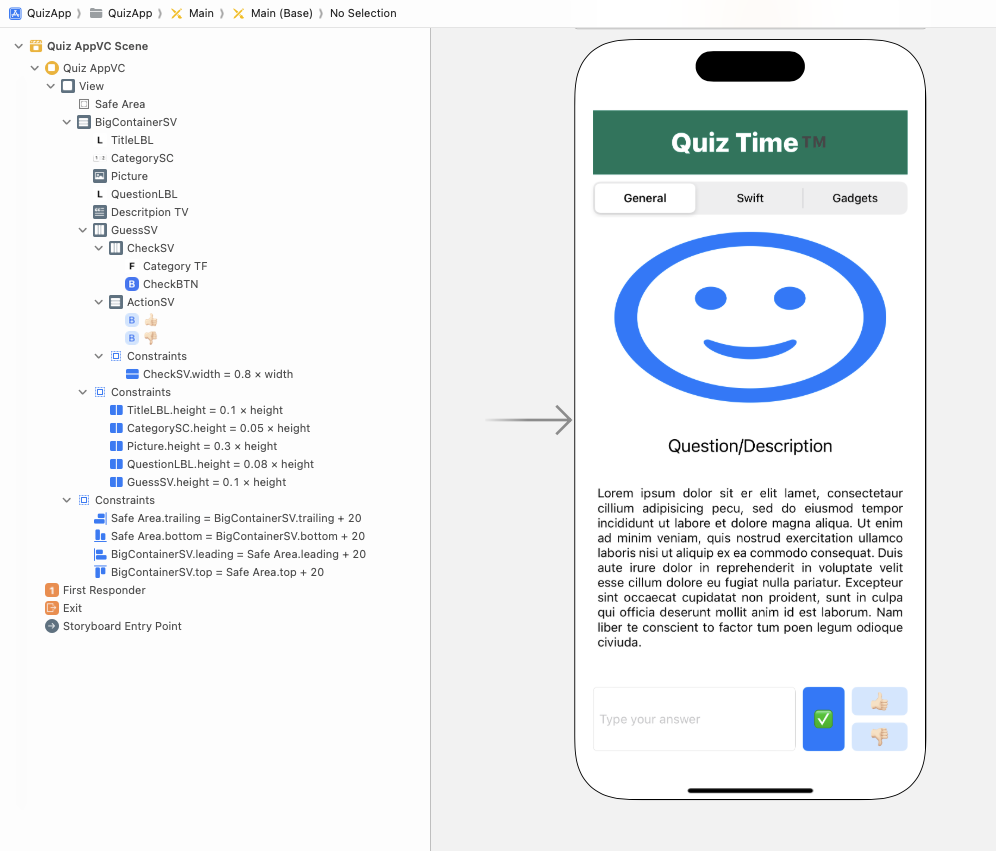


Figure 2 The View

1. Apply auto layout to the app by adding constraints to the UI elements as shown in the *Figure 2*.
   1. Hint: Follow the view hierarchy in the *Figure 2*. (i.e., document outline on the left side) to add and organize stack views and embed elements inside them.

Table 1 QuizAppVC’s properties for stack views

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stack View** | **Axis** | **Alignment** | **Distribution** | **Spacing** |
| BigContainerSV | Vertical | Fill | Fill | Standard |
| CheckSV | Horizontal | Fill | Fill | Standard |
| ActionSV | Vertical | Fill | Fill Equally | Standard |
| GuessSV | Horizontal | Fill | Fill | Standard |

**Model**

1. Download the supplied Model.swift file and add it to the project.
2. The struct UtilityConstants acts as our app data model. It contains database for different categories, for each category, pictures names, their description, related questions and possible answers are store using appropriate data structures. A sample for categories, namely, general, swift, and gadgets are provided. Similarly, add 2 more other appropriate categories of your choice.
3. Along with the categories, it also has some type constants for app requirements.
4. The struct Picture has name, description, and question properties. Name is the title of the image that put in your assets.
5. The struct Category is the struct that is used to create categories using its two initializers.
6. Provide a total of 5 different categories, for instance, general, swift, gadgets and 2 more other categories of your choice. Provide at least 5 pictures with their individual details for each category. Add all the pictures to your app’s Assets folder in XCode.
   1. Note: Provide high resolution files for each picture that you pick like @2x and @3x versions of that picture.

**Controller**

1. Create a Cocoa Touch Class “QuizAppVC” that is a sub class of UIViewController and assign it as class to Main.storyboard file.
2. Create outlet/action items as specified in *Table 1*.
3. This application is about displaying pictures along with their description and a quiz question based on the category chosen. For example, if user selects category general, then the pictures must be displayed related to that category using the pictureIV.
4. If the user double taps on the image, then show the description related to that picture in the descTV.
   1. If a picture has no description, display the default description message “Oops! We haven’t found anything related to this item.”
5. In case of triple tap on the image, a question related to the picture will be displayed in the descTV.
   1. If a picture has no quiz, display the default question message “Not a fun quiz 😕.”
6. User can navigate back and forth using swipe gestures i.e., left swipe for previous image and right swipe for next image. The pictureIV, & descTV should be updated w.r.t the action gesture performed.
7. Use the enumeration in the Model.swift to group the swipe and tap gestures that will be applied to the pictureIV.
8. If the user long presses the titleLBL button, app will be taken back to its initial state.

**Functionality**

1. At first, display the screen with default settings.
   1. The categorySC is unselected.
   2. The pictureIV is displaying default image face.smiling.
   3. The descTV displays guidelines on how to use the app.
   4. The playQuizSV containing the UI elements related to answering a quiz question is hidden.
2. User interaction should only be enabled for the pictureIV when user selects a category.
3. When the user selects a category, pictures in that category must be displayed using the pictureIV. Change the selected category tint color to red with alpha 0.5.
4. User can switch between the description and quiz using double and triple taps on the image, respectively.
5. When the user swipes left or right, refresh the app’s UI by updating the picture (i.e., pictureIV) and its description and quiz.
6. Provide a sound with system id 1104 whenever the user double or triple taps on the image.
   1. Use AudioServicesPlaySystemSound()method from AVKit for the sound.
7. When displaying the description, the playQuizSV should be hidden and the questionLBL text is “Description”.
8. When displaying a quiz question, the playQuizSV should be visible for user interaction and the questionLBL text is “Quiz Time ⏱️”.
   1. Display the below text along with the quiz question for the picture.

“- Type the possible answer in the text field.

- Tap the “✅” button to verify it.”

* 1. When the users enter a possible answer in the text field and taps on the ✅ button, display the appropriate messages in the descTV for the correct answer, incorrect answer, and for empty string. (Use Array class contains(\_:)method to check whether an array contains an element are not).
  2. If the user clicks on the ✅ button with empty string the text field, display an error message “Provide an answer in the text field.” in the descTV along with question.
  3. If the user enters a correct answer, play a sound with system id 1025. Change the tintColor of “👍” button to systemGreen and display the correct message from the UtilityConstants in the descTV along with question.
  4. In case of an incorrect answer, play a sound with system id 1053. Change the tintColor of “👎” button to systemRed and display the incorrect message from the UtilityConstants in the descTV along with question.

1. Long pressing the titleLBL button (i.e., minimum half-a-second) will simply reset the app to its initial state using an alert.

**Submission:**Push your entire Xcode project to your private GitHub repo and submit your private repository link on the Canvas via Text Entry.