Mobile Systems and Applications Assignment Two.

Student Name: Connor Haines

SID: 17025931

SCHOOL OF COMPUTING AND MATHEMATICS

*An assignment submitted for the degree of   
Bachelor of Science (BSc) in Computer Science.*

January 2020 – University of South Wales

INTRODUCTION.

This assignments task is clear and simple, you must create a small iOS application that enables the user to make a to-do list and manage it within the application, as well as mark the tasks as done when they are completed. Knowledge of iOS design principles will also be displayed and utilised within the application and report.

This document will guide you through the planning and design stage, gradually leading into the implementation stage and finally will prove the work with testing.

PLANNING AND DESIGN

When designing this application, the first thing that was to be done was to look at the iOS design principles. The apple website clearly states the three primary themes that they think “differentiates iOS from other platforms” (Themes - iOS - Human Interface Guidelines - Apple Developer, 2020). The first of these three things is Clarity, so the application must have text that is legible at every size while icons are always clear. There should be appropriate colouring and negative space and interface elements should subtly highlight important content whilst continuing to convey interactivity.   
  
The second of these is Deference. This is to do with fluid motion and a crisp UI to make it clear to the user what every piece of content does. Finally, the third is Depth. This relates to the visual layers and hierarchy of realistic motion to facilitate understanding. However, these are just the Design Themes. The Design Principles go less in depth but there are more ideas to discuss. The 6 Design Principles are:

* ***Aesthetic Integrity***
* ***Consistency***
* ***Direct Manipulation***
* ***Feedback***
* ***Metaphors***
* ***User Control***

The main principles that should really be focused upon are the three top ones, an applications appearance can have a huge impact on whether or not a user decides to download it. This is evident in an online thread on popular forum website *Reddit,* where the users talk about applications they purposely don’t use because one or more parts of the application is not aesthetically pleasing. Consistency is key and an aim for this project is to keep everything clean and relatively the same. Also, Direct Manipulation is obviously a big part of an application such as this where the user needs to input their own information.

A screenshot of a cell phone

Description automatically generatedThe first layout designed was the actual task creation screen. This was where the user would be once they came from the *TableView* and decided to add a task. As per the design brief the application needed a Title, Due Date and a *UIImageView* and so these were the first things implemented. For the *UIImageView* there is a placeholder image, that indicates that the user can choose their own image to be used there which uses the design principle of Direct Manipulation as well as the theme of Deference. Moving on to the Title and Due Date, these both consisted of a simple Label and a *TextView* which allows the users to input the task title and the date for when the task was meant to be completed by. Once the user had inputted the information they wanted to be added to the to-do list, nothing would happen until they clicked either the cancel, or the save button. Once this button was clicked the information would be captured, the user would be taken back to the *ListView* and the newly added information would be there, how this is done will be talked about in the implementation stage. Additionally, improvements were made on the initial design by changing the navigation bar details so XCode preferred big navigation bars and the text would be slightly bigger making it more legible for any user.

A screenshot of a cell phone

Description automatically generated Secondly, the next design that will be discussed is the *TableView.* This is where the user will initially load into when the application starts up. From this area the user can access all of the functions of the app such as add, edit, delete, sort the table, mark tasks as done and as an advanced feature they can search the table for a specific task. By default, the Tasks will be sorted by Nearest Date of Completion. The design for this was simple and again, looking at the principle of Consistency, is in line with the way the rest of the application looks. It is sleek, clean and intuitive. The user can swipe through the *TableView* and the use of big navigation bars is useful here as when you pull the *TableView* down the navigation bar and the text gets bigger and thus easier to read.

A screenshot of a cell phone

Description automatically generated

Another design that needs to be talked about is the editing a task design. This page is essentially the same as the add page however it is differentiated by the fact that the information is already filled out, giving the user a chance to check the information before they change the information. As it is the same as the add page, it will be a relatively simple layout with only the things the user needs, so the page doesn’t feel cluttered.

A screenshot of a cell phone

Description automatically generated

Finally, the last ‘page’ that will be looked at is when the user decides to remove the data from the *TableView* or potentially rearrange the data to better suit their needs, they will be able to choose the edit button in the corner of the page and they will see a small animation that will then allow them to move the tasks order around or click on the small universally known delete button to remove it.

IMPLEMENTATION

This area of the report will unfortunately be slightly affected due to circumstances that nobody could control (COVID-19). Therefore, all of the features that were implemented correctly will be at the beginning of the report and explained quite briefly and any features that were unable to be implemented because of this issue will be explained with pseudocode to support.

A screenshot of a cell phone

Description automatically generatedStarting with the add task page, from the top left downwards. The cancel button is simple, it is used as part of a function in the *TaskViewController* and luckily XCode has built in ‘Nav Bar Buttons’ that have premade choices in the ‘System Item’ tab that allow you to select ‘Cancel’. The function that this a part of listens for a *TapGesture*, if it is not the Save button, then it is the Cancel button and it runs this function:

A screenshot of text

Description automatically generated

On the other hand, if the Save Button is clicked is will respond with this function:

With some extra code elsewhere.

A screenshot of a cell phone

Description automatically generated Next, is the UIImageView. When creating the design for the Add Task page, CTRL+Drag function makes it easy to connect the Main.Storyboard to the TaskViewController. After they are linked, the ability to pick an image from the user’s camera roll is implemented the following way:

A screenshot of a cell phone

Description automatically generatedA black sign with white text

Description automatically generatedUsing the same CTRL + Drag technique for the TextView and DatePicker, the connection was easily made. Once they are inputted and the Save Button is clicked, the task is then saved as a *let* and added to the TasksArray (talked about further later).

A picture containing drawing

Description automatically generatedThe UIDatePicker was surprisingly easy to implement. It works by again using a GestureRecognizer and the following code to bring up a DatePicker and allowing the user to choose a date from there.

Moving on to the TableView, the first thing to look at would be the edit button. Like the cancel button on the previous page this is already built into the Swift language as a dedicated Nav Bar Button with prebuilt features. This code is prebuilt into swift here:A screenshot of a cell phone

Description automatically generated

A close up of a device

Description automatically generated

The add button on the top right hand side of the page is a dedicated Nav Bar Button like the last button however, this code is not prebuilt into swift and its main job is to move the page to the AddTask page, it was implemented using a segue and is presented Modally as you can see here:

FEATURES UNABLE TO BE IMPLEMENTED

Unfortunately, these features were planned, designed and mostly started but because of lack of access to certain hardware the completion of the features was not possible. This section is to explain what these features would have done, had they been completed.

Search Bar

The search bar was to be used at the top of the *TableView*, the user should have been able to type in a string, and if the string existed in the *Collection* of tasks it would have created a filtered collection that brought up only tasks that included the searched for string. This would make it easier for users to find specific tasks for editing and/or deletion. For this to be implemented, a connection between the *Search Bar* and the *TaskTableViewController* would have to be made and you would have to check for a t*extDidChange* in the *Search Bar*. During that *textDidChange* it would be possible to search if the filtered collection included the string the user searched for. Had I been able to complete this feature the code would have looked like this:

A screenshot of a cell phone

Description automatically generated

This would have been an additional feature that makes the users experience easier as if they have lots of activities in their to-do list, or that have forgotten the time limit they set their selves on a particular task they would not have to search the entire collection and they would be able to type in the task name and be shown the relevant task immediately.

Sort By Date

This is a simple feature, and it was simply to sort the tasks in the TableView by the date that they were supposed to be completed. This should be run every time the Table View was open and the code should look like this:

tasksArray.sort(by: {$0.taskDate < $1.taskDate});

Mark As Complete

The plan for this feature was to be able to swipe right on any given task and then it can be added to a new dynamically created ‘Completed’ section where it can be seen as the task is done and then deleted. The way in which this would be implemented would be by toggling that the task is done, and then removing said task from the ‘To-Do’ collection and reload the table view. Then you would add the task to a new collection containing only ‘Completed’ tasks and then you would reload the table view again. This could use UIContextualAction for extra help.

TESTING

All tests below were planned, any tests uncompleted were tests that I could not undertake because of the circumstances.

|  |  |  |  |
| --- | --- | --- | --- |
| TEST NUMBER | TEST DESC. | TEST EXPECTATIONS | TEST RESULTS |
| 1 | Load Up The Application. | The application should load successfully and show TableView. | The application loaded successfully and showed TableView. |
| 2 | Click Edit Button with No Tasks. | Nothing should happen, app shouldn’t crash. | Application doesn’t crash, nothing happens. |
| 3 | Click Add Button. | Should take user to the Add Test Page. | Loads up Add Test Page in a modal. |
| 4 | Click Cancel Button. | Should dismiss the modal and go back to TableView. | Dismisses the modal and takes user back to TableView. |
| 5 | Click Image Picker. | Should bring up the Users photo albums. | Brings up the users photo albums for them to choose from. |
| 6 | Fill in the Add Task page and Save. | All information should be saved to Collections and task name should be displayed on TableView | Information saved to relevant collections and task name is displayed on TableView. |
| 7 | Click task in TableView. | Should bring up all information in the edit page including image, name and date. | Brings up all information in the edit page with image, name and date included. |
| 8 | Make a change and save. | Should update the task in the array | Updates the task in the array. |
| 9 | Add task thats date is earlier than the other task | Should be visible in the TableView above the other task. |  |
| 10 | Click edit and move the task above the other one. | Should be able to move the task above the other one, regardless of date (priority). | The task is able to move above the other one, regardless of date. |
| 11 | Close the application fully and open again. | The data persistence should keep the tasks in the TableView. | The tasks remained in the TableView thanks to data persistence. |
| 12 | Swipe right to complete task. | Task should move from the ‘to-do’ section to the ‘completed’ section |  |
| 13 | Search for a task. | As the user is typing it should bring up relevant results. |  |

BIBLIOGRAPHY

Developer.apple.com. 2020. Themes - Ios - Human Interface Guidelines - Apple Developer. [online] Available at: <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/> (Accessed 09 March 2020).

body\_without\_organs. (2017) *Are there iOS apps you don’t use because the icon is ugly?.* Available at: <https://www.reddit.com/r/apple/comments/5wcbux/are_there_ios_apps_you_dont_use_because_the_icon/> (Accessed 17 March 2020).