UNIVERSITI TUNKU ABDUL RAHMAN

ASSIGNMENT 3 (20%)

**UECS3263 iOS APPLICATION DEVELOPMENT**

BACHELOR OF SCIENCE (HONOURS) SOFTWARE ENGINEERING

|  |  |
| --- | --- |
| Name (as stated in Student Card) | Student ID |
| Gervin Fung Da Xuen | 18UEB01655 |
| Programme | Submission Date |
| Software Engineering | 26/7/2021 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | | **Total Marks** | **Marks Given** |
| User Interface Components – CO2 (5 marks) | |  |  |
|  | Entry of thoughts using UITextView | 1 |  |
|  | Multiple scenes | 1 |  |
|  | Display date picker on a second scene | 1 |  |
|  | Display of list of moods on a second scene | 1 |  |
|  | Display of selected date and mood in the first scene | 1 |  |
| App Construction and Execution – CO3 (10 marks) | |  |  |
|  | Requirements implemented | 4 |  |
|  | Code design (including code readability) | 4 |  |
|  | Executes successfully | 1 |  |
|  | User guide documentation | 1 |  |
| Design Documentation – CO1 (5 marks) | |  |  |
|  | Diagram showing layout of user interface components (labelled with associated IBOutlets and ViewControllers) | 1 |  |
|  | Design decisions and app logic flow | 4 |  |
| Total | | 20 |  |

# Documentation

1. This is a Journal Application. The purpose of creating this application is to help people on writing journal at a specific date with a specific emotion. This application was created by assuming the users would like to write journal for what had already happened, hence allowing users to choose an earlier date. Likewise, this application was created with the assumption that users would like to write something to their future-self to read. In short, this application provides flexibility for users to write their journal by allowing them to choose any date they prefer.
2. **Main scene (ViewController)**

Graphical user interface, text, application, chat or text message

Description automatically generated

# Figure 1.1) Main screen

1. **Pick a Date screen**

Users can choose any date they prefer and the default date is the latest date, user can reselect any date if they had selected the wrong date by clicking *“Pick a Date”* button at the main screen

|  |  |
| --- | --- |
| Graphical user interface, text  Description automatically generated with medium confidence | Graphical user interface, text, application  Description automatically generated |

Figure 2.1) Enter Pick a Date scene (Left)

Figure 2.2) Randomly choose a Date (Right)

# Returning to main screen after selecting a date

# Returning to Main screen by clicking *“Confirm Date Button”*. The Main screen date was update to “26/8/2022”. Now *“Pick an Emoji”* button will be clicked

# Graphical user interface, application Description automatically generated

# Figure 3.1) Update of Main screen after choosing Date

# Pick an Emoji Screen

|  |  |
| --- | --- |
| Graphical user interface, text, application  Description automatically generated | Graphical user interface, text, application  Description automatically generated |

# Figure 4.1) Enter Pick Emoji Screen (Left)

# Figure 4.2) Choose an emoji (Cheerful in this case) (Right)

# If user would like to search specific emoji, they can use the search bar given above the table-view as shown by figure below

# Graphical user interface, text, application, Teams Description automatically generated

# Figure 4.3) Use search bar to find available emoji

# 5.0) Return to Main screen after choosing emoji

# After choosing emoji, user can return to home screen by clicking *“Confirm Emotion/Emoji”* button. The Figure on the left side show the update of Emoji chosen as well as typing new text on TextView. The Figure on the right side will show an alert when user click the “Save” button

|  |  |
| --- | --- |
| Graphical user interface, text, application, chat or text message  Description automatically generated | Graphical user interface, text, application  Description automatically generated |

# Figure 5.1) Update of Main screen after choosing Emoji and typing new text on TextView

# Figure 5.2) Showing an alert when user click “Save” button

**User Interface Diagram**

@IBOutlet weak var emojiLabel: UILabel

@IBOutlet weak var emojiSearchBar: UISearchBar!

**Graphical user interface, text, application

Description automatically generated**

@IBOutlet weak var confirmEmojiBtn: UIButton!

@IBOutlet weak var emojiTableView: UITableView!

@IBOutlet weak var saveBtn: UIButton!

@IBOutlet weak var pickADate: UILabel!

@IBOutlet weak var emojiLabel: UILabel!

**Graphical user interface, text, application, chat or text message

Description automatically generated**

@IBOutlet weak var thoughtTextArea: UITextView!

@IBOutlet weak var pickDateBtn: UIButton!

@IBOutlet weak var pickADate: UILabel!

@IBOutlet weak var chosenDateLabel: UILabel!

**Graphical user interface, text, application

Description automatically generated**

@IBOutlet weak var datePicker: UIDatePicker!

@IBOutlet weak var confirmDateBtn: UIButton!

**Design Decisions and App Logic Flow**

The application is designed by assuming that it will be used any users, that means it can be used by teenagers, adult and elderlies. Therefore, it has a very simple, minimal and straight-forward interface so that elderlies can also easily distinguish the buttons and labels. Also, the “Pick an Emoji” button is colored with red color and placed at the top-left of the application because red color can easily capture their attention and users will first noticed the element that are on the top left, hence allow users to pick an emotion first. Next, the “Pick a Date” button is colored with blue to provide calmness after choosing an emotion/emoji. Finally, the save button is of green color and is placed near the center of the application to accomplish 2 purposes

1. To capture their attention by allow them to easily the save button is at the center
2. To allow them to easily click the save button as it’s near to all the buttons and textView, so users don’t have to move their finger a lot, hence fulfilling the **Path of Least Resistance.**

For the EmojiUIView, a search bar is provided to allow user to type the emotions that they are experiencing so that they don’t have to scroll to search for the emoji they want. According to one of the Shneiderman’s Eight Golden Rules for Interface Design, which is **Offer Informative Feedbacks**, application should provide appropriate feedback to user to let them know that a certain action has been carried out accordingly. Hence, when they click the emoji/emotion that match their desired emoji, the label of “How do you feel?” will be changed to “I feel <emoji emotion>” to provide feedback that their emoji is saved. When they click confirm button, they will be directed to Main Page.

For the DatePickerUIView, only a DatePicker will be shown to allow users to scroll the date that they desired. Similar to picking an emoji, once the found the date that they want, the default date label will change from “Date chosen: <latest date>” to “Date chosen: <chosen date by user>” to provide informative feedbacks as well. When they click confirm button, they will be directed to Main Page

For the thought entry part, a prompt text will appear to indicate that the TextView can be edited and they can write whatever they want. Lastly, user canclick “*Save”* button to save their content. An alert dialog will popup to inform that the content has been saved to provide informative feedback

The diagram below had shown the flow of the program clearly.

**Task-flow of the application**

Diagram

Description automatically generated

## **Code Listing**

Main Page

//

//  ViewController.swift

//  asm3

//

//  Created by Guest User on 7/24/21.

//  Copyright © 2021 Guest. All rights reserved.

//

import UIKit

final *class* ViewController: UIViewController, UITextViewDelegate {

    private static let DEFAULT\_TEXT\_VIEW\_TEXT = "What is on your mind?"

    @IBOutlet weak var emojiLabel: UILabel!

    @IBOutlet weak var pickADate: UILabel!

    @IBOutlet weak var pickEmojiBtn: UIButton!

    @IBOutlet weak var pickDateBtn: UIButton!

    @IBOutlet weak var thoughtTextArea: UITextView!

    @IBOutlet weak var saveBtn: UIButton!

    override *func* viewDidLoad() {

        super.viewDidLoad()

        self.emojiLabel.text = EmojiUIView.DEFAULT\_EMOJI\_TEXT

        self.pickEmojiBtn.layer.cornerRadius = 5.0

        self.pickDateBtn.layer.cornerRadius = 5.0

        self.saveBtn.layer.cornerRadius = 5.0

        self.thoughtTextArea.delegate = self

        self.thoughtTextArea.layer.cornerRadius = 10.0

        self.thoughtTextArea.text = ViewController.DEFAULT\_TEXT\_VIEW\_TEXT

        self.thoughtTextArea.textColor = UIColor.lightGray

        self.pickADate.text = DateUtil.getSingletonInstance().formStringFromDate(date: Date())

    }

    private *func* showSavedAlertDialog() {

        let alertController = UIAlertController(title: "Journal Saved", message: "Your thoughts and emotion at date: \(self.pickADate.text!) is saved!", preferredStyle: UIAlertController.Style.alert)

        let cancelAction = UIAlertAction(title: "OK", style: UIAlertAction.Style.cancel, handler: nil)

        alertController.addAction(cancelAction)

        self.present(alertController, animated: true, completion: nil)

    }

    @IBAction *func* saveBtnClicked(\_ *sender*: UIButton) {

        self.thoughtTextArea.resignFirstResponder()

        self.showSavedAlertDialog()

    }

    internal *func* textViewDidBeginEditing(\_ *textView*: UITextView) {

        if textView.textColor == UIColor.lightGray {

            textView.text = nil

            textView.textColor = UIColor.black

        }

    }

    internal *func* textViewDidEndEditing(\_ *textView*: UITextView) {

        if textView.text.isEmpty {

            textView.text = ViewController.DEFAULT\_TEXT\_VIEW\_TEXT

            textView.textColor = UIColor.lightGray

        }

    }

    override *func* prepare(for *segue*: UIStoryboardSegue, sender: *Any*?) {

        if (segue.identifier == "emoticonIdentifier") {

            let emojiVC = segue.destination as! EmojiUIView

            emojiVC.updateEmoji(emojiText: self.emojiLabel.text!)

        } else if (segue.identifier == "dataPickerIdentifier") {

            let datePickerVC = segue.destination as! DatePickerUIView

            datePickerVC.updateDate(date: self.pickADate.text!)

        }

    }

}

final *class* DateUtil {

    private static let dateUtil = DateUtil()

    private let dateFormatter: DateFormatter

    private *init*() {

        self.dateFormatter = DateFormatter()

        self.dateFormatter.dateFormat = "dd/MM/yyyy"

    }

    public *func* formStringFromDate(date: Date) -> *String* {

        return self.dateFormatter.string(from: date)

    }

    public *func* formDateFromString(string: *String*) -> Date {

        return self.dateFormatter.date(from: string)!

    }

    public static *func* getSingletonInstance() -> DateUtil {

        return DateUtil.dateUtil

    }

}

DatePickerUIView page

//

//  DatePickerUIView.swift

//  asm3

//

//  Created by Guest User on 7/24/21.

//  Copyright © 2021 Guest. All rights reserved.

//

import UIKit

final *class* DatePickerUIView : UIViewController {

    private static let DATE\_CHOSEN = "Date Chosen: "

    @IBOutlet weak var datePicker: UIDatePicker!

    @IBOutlet weak var confirmDateBtn: UIButton!

    @IBOutlet weak var chosenDateLabel: UILabel!

    private var date: *String* = ""

    override *func* viewDidLoad() {

        super.viewDidLoad()

        let dates = DateUtil.getSingletonInstance().formDateFromString(string: self.date)

        self.datePicker.setDate(dates, animated:true)

        self.chosenDateLabel.text = DatePickerUIView.DATE\_CHOSEN + self.date

        self.chosenDateLabel.textAlignment = .center

        self.confirmDateBtn.layer.cornerRadius = 5.0;

    }

    @IBAction *func* chooseDate(\_ *sender*: UIDatePicker) {

        let dateTxt = DatePickerUIView.DATE\_CHOSEN + DateUtil.getSingletonInstance().formStringFromDate(date: datePicker.date)

        self.chosenDateLabel.text = dateTxt

    }

    @IBAction *func* confirmDateBtn(\_ *sender*: UIButton) {

        let firstVC = presentingViewController as! ViewController

        firstVC.pickADate.text = DateUtil.getSingletonInstance().formStringFromDate(date: datePicker.date)

        self.dismiss(animated: true, completion: nil)

    }

    public *func* updateDate(date: *String*) -> *Void* {

        self.date = date

    }

}

EmojiUIView page

//

//  EmojiUIView.swift

//  asm3

//

//  Created by Guest User on 7/24/21.

//  Copyright © 2021 Guest. All rights reserved.

//

import UIKit

final *class* EmojiUIView : UIViewController, UISearchBarDelegate, UITableViewDataSource, UITableViewDelegate {

    public static let DEFAULT\_EMOJI\_TEXT = "How do you feel?"

    private static let CELL\_IDENTIFIER: *String* = "cell"

    private static let IMMUTABLE\_EMOTIONS\_COLLECTION: *Array*<*String*> = EmojiUIView.readPropertyList()

    private var emojiText: *String* = ""

    private var filteredEmotionCollection: *Array*<*String*> = EmojiUIView.IMMUTABLE\_EMOTIONS\_COLLECTION;

    @IBOutlet weak var emojiTableView: UITableView!

    @IBOutlet weak var emojiSearchBar: UISearchBar!

    @IBOutlet weak var emojiLabel: UILabel!

    @IBOutlet weak var confirmEmojiBtn: UIButton!

    override *func* viewDidLoad() {

        super.viewDidLoad()

        self.emojiTableView.dataSource = self

        self.emojiTableView.delegate = self

        self.emojiSearchBar.delegate = self

        self.emojiTableView.register(UITableViewCell.self, forCellReuseIdentifier: EmojiUIView.CELL\_IDENTIFIER)

        self.emojiLabel.text = self.emojiText

        self.emojiLabel.textAlignment = .center

        self.confirmEmojiBtn.layer.cornerRadius = 5.0;

    }

    @IBAction *func* confirmEmojiBtn(\_ *sender*: UIButton) {

        let firstVC = presentingViewController as! ViewController

        firstVC.emojiLabel.text = self.emojiText

        self.dismiss(animated: true, completion: nil)

    }

    public *func* updateEmoji(emojiText: *String*) {

        self.emojiText = emojiText

    }

    private static *func* readPropertyList() -> *Array*<*String*> {

        if let filePath = Bundle.main.path(forResource: "emoticonsPList", ofType: "plist") {

            if let plistData = FileManager.default.contents( atPath: filePath) {

                do {

                    let plistObject = try PropertyListSerialization.propertyList(from: plistData, options: PropertyListSerialization.ReadOptions(), format: nil)

                    let emotionsInList = plistObject as? *Array*<*String*>

                    if let emotionsInList = emotionsInList {

                        return emotionsInList

                    }

                } catch {

                    print("Error serializing data from property list")

                }

            } else {

                print("Error reading data from property list file")

            }

        } else {

            print("Property list file does not exist")

        }

        return []

    }

    internal *func* numberOfSections(in *tableView*: UITableView) -> *Int* {

        return 1

    }

    internal *func* tableView(\_ *tableView*: UITableView, numberOfRowsInSection *section*: *Int*) -> *Int* {

        return self.filteredEmotionCollection.count

    }

    internal *func* tableView(\_ *tableView*: UITableView, cellForRowAt *indexPath*: IndexPath) -> UITableViewCell {

        let cell: UITableViewCell! = tableView.dequeueReusableCell(withIdentifier: EmojiUIView.CELL\_IDENTIFIER, for: indexPath)

        if cell == nil {

            let newCell = UITableViewCell(style: UITableViewCell.CellStyle.default, reuseIdentifier: EmojiUIView.CELL\_IDENTIFIER)

            newCell.textLabel!.text = self.filteredEmotionCollection[indexPath.row]

            return newCell

        }

        cell.textLabel?.text = self.filteredEmotionCollection[indexPath.row]

        return cell

    }

    internal *func* tableView(\_ *tableView*: UITableView, didSelectRowAt *indexPath*: IndexPath) {

        self.emojiLabel.text = "I feel \(self.filteredEmotionCollection[indexPath.row])"

        self.emojiText = self.emojiLabel.text!

    }

    internal *func* searchBar(\_ *searchBar*: UISearchBar, textDidChange *searchText*: *String*) {

        if searchText != "" {

            let lowerCasedSearchText = searchText.lowercased()

            self.filteredEmotionCollection = EmojiUIView.IMMUTABLE\_EMOTIONS\_COLLECTION.filter { (emoji) -> *Bool* in

*String*(emoji.dropFirst(2)).lowercased().contains(lowerCasedSearchText)

            }

        }

        else {

            self.filteredEmotionCollection = EmojiUIView.IMMUTABLE\_EMOTIONS\_COLLECTION

        }

        self.emojiTableView.reloadData()

    }

    internal *func* searchBarSearchButtonClicked(\_ *searchBar*: UISearchBar) {

        searchBar.resignFirstResponder()

    }

}