UNIVERSITI TUNKU ABDUL RAHMAN

ASSIGNMENT 5 (30%)

**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 Components – CO2 (10 marks) | |  |  |
|  | Scene for display of list of items | 2 |  |
|  | Scene for display of detail of selected item | 2 |  |
|  | Scene for updating selected item | 2 |  |
|  | Scene for adding new item | 2 |  |
|  | Scene for deleting item | 2 |  |
| App Construction and Execution – CO3 (14 marks) | |  |  |
|  | NSManagedObject subclass | 1 |  |
|  | Displays list of items | 2 |  |
|  | Displays detail of selected item | 2 |  |
|  | Updates selected item | 3 |  |
|  | Adds new item | 3 |  |
|  | Deletes selected item | 3 |  |
| App Documentation – CO1 (6 marks) | | | |
|  | Entity design (Screenshot of entity in data model editor) | 1 |  |
|  | Explanation of different types of segues used in app  (why and how the segues are used) | 5 |  |
| Total | | 30 |  |

Contents

[**App Screenshots** 1](#_Toc81962477)

[**Entity Design (Screenshot of entity in data model editor)** 8](#_Toc81962478)

[**Explanation of different types of segues used in app** 9](#_Toc81962479)

[**Complete Code Listing** 13](#_Toc81962480)

# **App Screenshots**

1. When user launches the app, the table view controller, which is the main screen is shown.

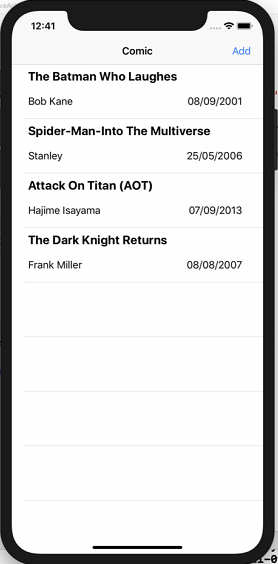


Figure 1.0: Main Screen

1. When user tap on “add” button, the user is directed to add screen to add new comic information as shown by Figure 1.1 below

|  |  |
| --- | --- |
|  |  |
| Figure 1.1: Add Comic Screen | Figure 1.2: Saved Added Comic |

When user enters the new comic book information and tap save, the app saves the new comic book information and returns to the main screen.

1. When user taps on one of the comic book information, the detail of the comic book will be shown in another screen as shown by the figure below.

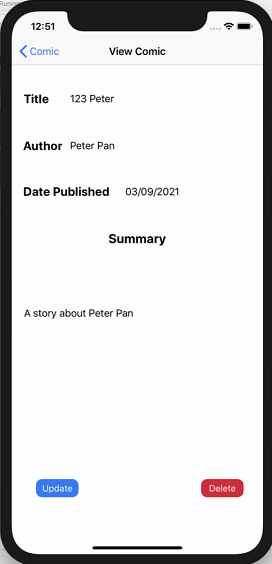


Figure 1.3: Show Comic Screen

When user click the update button, they will be direct to update screen to update the comic book information. If user click delete button, they will be direct to delete screen to confirm their deletion.

1. User do not need to change all the information because for the information that they did not update, the current value will be used.

|  |  |
| --- | --- |
|  |  |
| Figure 1.4: Edit information | Figure 1.5: Information is not updated |

Do note that the title changed from “123 Peter” to “124 Peter” in the Edit screen and the author changed from “Peter Pan” to “Peter Pan 123”. From here, user have 2 options, to either save their changes, or to go back their main screen without changing it. The figure 1.5 shown above prove that the information is not updated. Here, 2 Segues is used which will be explained later. The next section will show that what will happen when user click “save” button instead of “Back to Main” .

|  |  |
| --- | --- |
|  |  |
| Figure 1.6: Edit information | Figure 1.7: Information is updated |

1. User can also delete their comic book information as mentioned previously. Similarly, user have 2 options, to either delete the information permanently, or to go back their main screen without deleting it.

|  |  |
| --- | --- |
|  |  |
| Figure 1.8: Delete information | Figure 1.9: Information is not deleted |

|  |  |
| --- | --- |
|  |  |
| Figure 1.10: Delete information confirmation | Figure 1.11: Information is deleted |

The figure above shows that confirmation dialog will show if user choose to delete the particular comic book information. After confirm deletion, they will be direct back to main page, otherwise, there will be no changes.

# **Entity Design (Screenshot of entity in data model editor)**

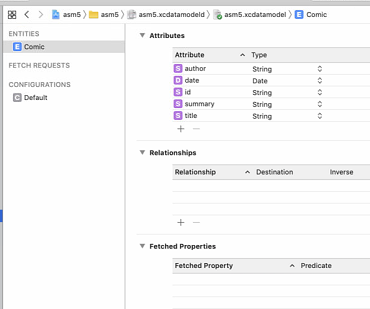


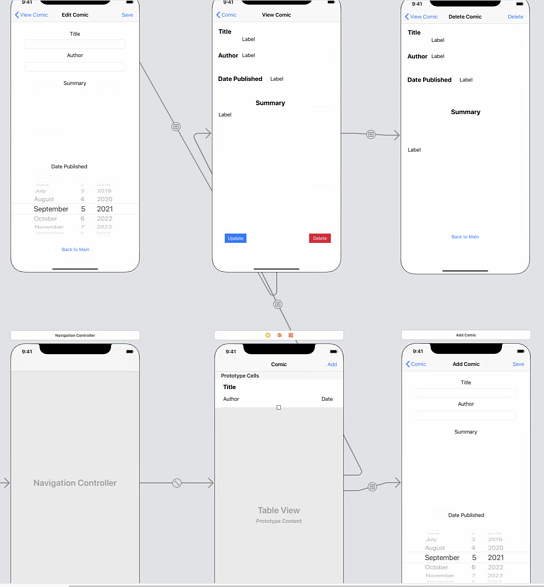
Figure 2.0: Entity Relation Diagram

The figure above shown that there are only 5 attributes, 4 of them are mutable and 1 of them is immutable, which is the id, in string generated by Universal Unique Identifier (UUID). This is to ensure each comic has a unique ID associated with it.

# **Explanation of different types of segues used in app**

There are two types of segues used in the app, which includes show segue and unwind segue.

|  |  |
| --- | --- |
|  |  |
| Figure 3.0: Segue used | Figure 3.1: Segue used |



NavigationController

ComicTableViewController

ComicDetailVCController

(AddComic)

EditComic

ViewComic

DeleteComic

Figure 3.2: Overview

There are 2 segues used – Unwind Segue and Show Segue. There are 2 Unwind Segues that were used, in the “Back to Main” button in **Edit** and **Delete** scene. This is used so that user can navigate back to main scene unlike **Add** scene, where clicking back will direct them back to the main scene, **Edit** and **Delete** scene came from **View** scene, and thus will **need another button to navigate them back to main scene** directly. On the other hand, the “save” for **Edit** and “delete” **Delete** use *popToRootViewController* to pop back to root (main) scene. It’s just a matter of using an alternative method and this allows the respective update and delete to be executed before navigating back to the main screen. Hence, unwind segue connect the segue from Update and Delete scene to Main scene via “unwindSegue” function as the function is defined in the Main scene.



Figure 3.3: unwindSegue function

The show segue is used to navigate to **Add, View, Update and Delete** scene. The reason it was used because it is very convenient. Show segue pushed a scene on top of the other scene in the navigation stack and it also allow users to slide the scene in from right to left. Lastly, it adds the navigation bar with a back button that allows user to navigate to the previous screen and user can also slide from left to right to navigate back to the previous scene. Hence, show segue is applied to

1. Main scene to navigate to Add scene via “newComic” segue
2. Main scene to navigate to View scene via “viewComic” segue and by overriding prepare function (Figure 3.4)
3. View scene to navigate to Update and Delete Scene via “editComic” segue and “deleteComic” segue by overriding prepare function (Figure 3.5)

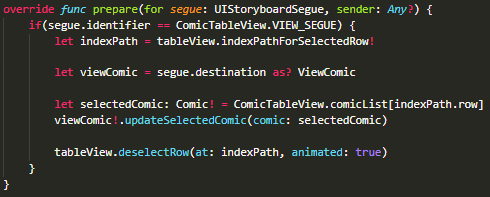


Figure 3.4 Override prepare function of Main scene

From Figure 3.4, it’s evident that the selected row of comic information is passed to View scene so that it can display all the data.

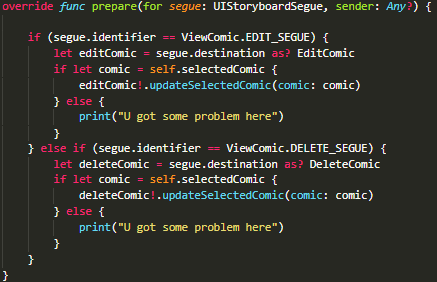


Figure 3.5 Override prepare function of View scene

From Figure 3.4, it’s evident that the row of comic information is passed to from View scene to Edit and Delete scene accordingly so that it can update or delete the chosen row of comic.

# **Complete Code Listing**

Comic.swift

import CoreData

import UIKit

@objc(Comic)

final *class* Comic: NSManagedObject {

    @NSManaged var id: *String*!

    @NSManaged var author: *String*!

    @NSManaged var title: *String*!

    @NSManaged var summary: *String*!

    @NSManaged var date: NSDate!

}

final *class* ComicUtil {

    public static *func* getContext() -> NSManagedObjectContext {

        let appDelegate = UIApplication.shared.delegate as! AppDelegate

        return appDelegate.persistentContainer.viewContext

    }

    public static *func* getFetch() -> NSFetchRequest<NSFetchRequestResult> {

        return NSFetchRequest<NSFetchRequestResult>(entityName: "Comic")

    }

}

ComicCell.swift

import UIKit

final *class* ComicCell : UITableViewCell {

    @IBOutlet weak var authorLabel: UILabel!

    @IBOutlet weak var titleLabel: UILabel!

    @IBOutlet weak var dateLabel: UILabel!

}

ComicTableView.swift

import UIKit

import CoreData

final *class* ComicTableView : UITableViewController {

    private var notLoaded = true

    private static let IDENTIFIER = "comicCellID"

    private static let VIEW\_SEGUE = "viewComic"

    private static var comicList = [Comic]()

    override *func* viewDidLoad() {

        if (notLoaded) {

            notLoaded = false

            let context = ComicUtil.getContext()

            let request = ComicUtil.getFetch()

            do {

                let results:NSArray = try context.fetch(request) as NSArray

                ComicTableView.comicList = results.map{ $0 as! Comic }

            } catch {

                print("Fetch Failed")

            }

        }

    }

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

        let comicCell = tableView.dequeueReusableCell(withIdentifier: ComicTableView.IDENTIFIER, for: indexPath) as! ComicCell

        let thisComic: Comic! = ComicTableView.comicList[indexPath.row]

        comicCell.authorLabel.text = thisComic.author

        let date: Date = thisComic.date! as Date

        comicCell.dateLabel.text = DateUtil.getSingletonInstance().formStringFromDate(date: date)

        comicCell.titleLabel.text = thisComic.title

        return comicCell

    }

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

        return ComicTableView.comicList.count

    }

    override *func* viewDidAppear(\_ *animated*: *Bool*) {

        tableView.reloadData()

    }

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

        self.performSegue(withIdentifier: ComicTableView.VIEW\_SEGUE, sender: self)

    }

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

        if(segue.identifier == ComicTableView.VIEW\_SEGUE) {

            let indexPath = tableView.indexPathForSelectedRow!

            let viewComic = segue.destination as? ViewComic

            let selectedComic: Comic! = ComicTableView.comicList[indexPath.row]

            viewComic!.updateSelectedComic(comic: selectedComic)

            tableView.deselectRow(at: indexPath, animated: true)

        }

    }

    @IBAction *func* unwindSegue(\_ *sender*: UIStoryboardSegue) {

    }

    public static *func* addComic(comic: Comic) {

        ComicTableView.comicList.append(comic)

    }

    public static *func* setComics(comics: *Array*<Comic>) {

        ComicTableView.comicList = comics

    }

}

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

    }

}

DeleteComic.swift

import UIKit

import CoreData

final *class* DeleteComic : UIViewController {

    @IBOutlet weak var titleLabel: UILabel!

    @IBOutlet weak var authorLabel: UILabel!

    @IBOutlet weak var dateLabel: UILabel!

    @IBOutlet weak var summaryLabel: UILabel!

    private var selectedComic: Comic? = nil

    override *func* viewDidLoad() {

        super.viewDidLoad()

        summaryLabel.numberOfLines = 0

        if let comic = self.selectedComic {

            titleLabel.text = comic.title

            authorLabel.text = comic.author

            summaryLabel.text = comic.summary

            let date: Date = comic.date! as! Date

            dateLabel.text = DateUtil.getSingletonInstance().formStringFromDate(date: date)

        } else {

            print("No Comic? Something is wrong, only through segue can reach here")

        }

    }

    private *func* delete() {

        let context = ComicUtil.getContext()

        let request = ComicUtil.getFetch()

        do {

            let results: NSArray = try context.fetch(request) as NSArray

            let comic = results.first(where: {($0 as! Comic).id == self.selectedComic?.id}) as! Comic

            context.delete(comic)

            ComicTableView.setComics(comics: results.filter({($0 as! Comic).id != self.selectedComic?.id}) as! *Array*<Comic>)

            try context.save()

            navigationController?.popToRootViewController(animated: true)

        } catch {

            print("Fetch Failed")

        }

    }

    @IBAction *func* deleteComic(\_ *sender*: *Any*) {

        let alertController = UIAlertController(title: "Delete Confirmation", message: "Are you sure you want to delete. This cannot be undone", preferredStyle: .alert)

        let OKAction = UIAlertAction(title: "OK", style: .default) { (action:UIAlertAction!) in

            self.delete()

        }

        alertController.addAction(OKAction)

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

        alertController.addAction(cancelAction)

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

    }

    public *func* updateSelectedComic(comic: Comic) {

        self.selectedComic = comic

    }

}

EditComic.swift

import UIKit

import CoreData

final *class* EditComic : UIViewController {

    @IBOutlet weak var titleTF: UITextField!

    @IBOutlet weak var authorTF: UITextField!

    @IBOutlet weak var datePublished: UIDatePicker!

    @IBOutlet weak var summaryTF: UITextView!

    private var selectedComic: Comic? = nil

    override *func* viewDidLoad() {

        super.viewDidLoad()

        titleTF.delegate = self

        authorTF.delegate = self

        summaryTF.delegate = self

        if let comic = self.selectedComic {

            titleTF.text = comic.title

            authorTF.text = comic.author

            summaryTF.text = comic.summary

            let date: Date = comic.date! as! Date

            self.datePublished.setDate(date, animated:true)

        } else {

            print("No Comic? Something is wrong, only through segue can reach here")

        }

    }

    @IBAction *func* saveEdit(\_ *sender*: *Any*) {

        let author: *String* = authorTF.text!

        let title: *String* = titleTF.text!

        let summary = summaryTF.text!

        if (author.count != 0 && title.count != 0 && summary.count != 0) {

            let context = ComicUtil.getContext()

            let request = ComicUtil.getFetch()

            do {

                let results:NSArray = try context.fetch(request) as NSArray

                let comic = results.first(where: {($0 as! Comic).id == self.selectedComic?.id}) as! Comic

                comic.author = authorTF.text

                comic.title = titleTF.text

                comic.date = datePublished.date as NSDate

                comic.summary = summaryTF.text

                try context.save()

                navigationController?.popToRootViewController(animated: true)

            } catch {

                print("Fetch Failed")

            }

        } else {

            let alertController = UIAlertController(title: "Warning", message: "No empty fields are allowed", 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)

        }

    }

    public *func* updateSelectedComic(comic: Comic) {

        self.selectedComic = comic

    }

}

*extension* EditComic: UITextFieldDelegate {

*func* textFieldShouldReturn(\_ *textField*: UITextField) -> *Bool* {

        textField.resignFirstResponder()

        return true

    }

}

*extension* EditComic: UITextViewDelegate {

*func* textView(\_ *textView*: UITextView, shouldChangeTextIn *range*: NSRange, replacementText *text*: *String*) -> *Bool* {

        if(text == "\n") {

            textView.resignFirstResponder()

            return false

        }

        return true

    }

}

ViewComic.swift

import UIKit

final *class* ViewComic: UIViewController {

    private static let EDIT\_SEGUE = "editComic"

    private static let DELETE\_SEGUE = "deleteComic"

    @IBOutlet weak var titleLabel: UILabel!

    @IBOutlet weak var authorLabel: UILabel!

    @IBOutlet weak var dateLabel: UILabel!

    @IBOutlet weak var summaryLabel: UILabel!

    @IBOutlet weak var updateBtn: UIButton!

    @IBOutlet weak var deleteBtn: UIButton!

    private var selectedComic: Comic? = nil

    override *func* viewDidLoad() {

        super.viewDidLoad()

        updateBtn.layer.cornerRadius = 10

        updateBtn.clipsToBounds = true

        deleteBtn.layer.cornerRadius = 10

        deleteBtn.clipsToBounds = true

        summaryLabel.numberOfLines = 0

        if let comic = self.selectedComic {

            titleLabel.text = comic.title

            authorLabel.text = comic.author

            summaryLabel.text = comic.summary

            let date: Date = comic.date! as! Date

            dateLabel.text = DateUtil.getSingletonInstance().formStringFromDate(date: date)

        } else {

            print("No Comic? Something is wrong, only through segue can reach here")

        }

    }

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

        if (segue.identifier == ViewComic.EDIT\_SEGUE) {

            let editComic = segue.destination as? EditComic

            if let comic = self.selectedComic {

                editComic!.updateSelectedComic(comic: comic)

            } else {

                print("U got some problem here")

            }

        } else if (segue.identifier == ViewComic.DELETE\_SEGUE) {

            let deleteComic = segue.destination as? DeleteComic

            if let comic = self.selectedComic {

                deleteComic!.updateSelectedComic(comic: comic)

            } else {

                print("U got some problem here")

            }

        }

    }

    public *func* updateSelectedComic(comic: Comic) {

        self.selectedComic = comic

    }

}

ComicDetailVC.swift

import UIKit

import CoreData

final *class* ComicDetailVC: UIViewController {

    @IBOutlet weak var titleTF: UITextField!

    @IBOutlet weak var authorTF: UITextField!

    @IBOutlet weak var datePublished: UIDatePicker!

    @IBOutlet weak var summaryTF: UITextView!

    override *func* viewDidLoad() {

        super.viewDidLoad()

        titleTF.delegate = self

        authorTF.delegate = self

        summaryTF.delegate = self

    }

    @IBAction *func* saveAction(\_ *sender*: *Any*) {

        let author: *String* = authorTF.text!

        let title: *String* = titleTF.text!

        let summary = summaryTF.text!

        if (author.count != 0 && title.count != 0 && summary.count != 0) {

            let context = ComicUtil.getContext()

            let entity = NSEntityDescription.entity(forEntityName: "Comic", in: context)

            let newComic = Comic(entity: entity!, insertInto: context)

            newComic.id = UUID().uuidString

            newComic.author = author

            newComic.title = title

            newComic.date = datePublished.date as NSDate

            newComic.summary = summary

            do {

                try context.save()

                ComicTableView.addComic(comic: newComic)

                navigationController?.popViewController(animated: true)

            } catch {

                print("context save error")

            }

        } else {

            let alertController = UIAlertController(title: "Warning", message: "No empty fields are allowed", 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)

        }

    }

}

*extension* ComicDetailVC: UITextFieldDelegate {

*func* textFieldShouldReturn(\_ *textField*: UITextField) -> *Bool* {

        textField.resignFirstResponder()

        return true

    }

}

*extension* ComicDetailVC: UITextViewDelegate {

*func* textView(\_ *textView*: UITextView, shouldChangeTextIn *range*: NSRange, replacementText *text*: *String*) -> *Bool* {

        if(text == "\n") {

            textView.resignFirstResponder()

            return false

        }

        return true

    }

}