SWEN BULLETINS

SWEN3000 – iOS Development

Abstract

This report describes the development of a Bulletin Board application. The application is targeted at the students enrolled in the BSc. Software Engineering programme offered by The University of the West Indies-Chinese Institute of Technology (UWI-CIIT).

CodeMetric Technologies

codemetricswen@gmail.com

iOS Development: Report

Lecturer:
Pro. Thomas Canhao Xu

Group Name:
CodeMetric Technologies

Group Leader:
Scott Allen

Other Members:
Khadeja Clarke
Mischka Neill

Table of Tasks/Responsibilities

Student ID No.		Student Name	Tasks
UWI	GIST		
620099245	180923	Allen, Scott Nicholas	Creating a New Bulletin
			User Authentication
620098443	180908	Clarke, Khadeja Renee	Viewing All Bulletins
620100135	180913	Neill, Mischka Zoila	Viewing a Specific Bulletin

Introduction

Purpose of this Document

The main purpose of this document is to provide a detailed explanation of this project from start to finish.

Intended Audience

This document is intended for the lecturers of The University of the West Indies (UWI) and Global Institute of Software Technology (GIST).

Scope

This document should provide the intended audience with details about the project's lifecycle.

Background Information

Purpose of this Project

To fulfil the requirements of the course SWEN3000 - iOS Application Development.

Design Description

Overview

A native iOS application built using the Swift language.

Detailed Description

A bulletin board application which allows for the creation, viewing, edition, sharing and deletion of bulletins stored in a Firebase Realtime Database.

Goals

Creating a new bulletin
Create a new bulletin entry in the Firebase Realtime Database.

Viewing all bulletins
Retrieving all records from the Firebase Realtime Database.

Viewing a specific bulletin
Retrieving a single specified record from the Firebase Realtime Database.

Editing a bulletin
Make changes to the bulletin being currently viewed, in the Firebase Realtime Database.

Sharing a bulletin
Share the bulletin being viewed with other applications.

Deleting a bulletin
Remove the bulletin currently being viewed, from the Firebase Realtime Database.

Implementation

Viewing All Bulletins

A UICollectionView was used to display all the bulletins.

onViewDidLoad() initiates the layout i.e.:

1. the background colour which is an image of a corkboard, and
2. the toolbar which has a Compose UIBarButtonItem, positioned between a Fixed Space UIBarButtonItem and a Flexible Space UIBarButtonItem on either side. On click of the compose button, the user will be redirected to the storyboard for the creation of a new bulletin.

onViewWillAppear() reads from the Firebase database. If there are bulletins, it adds them to the array used to keep a local record of all the bulletins. It is from this array that the view is populated.

The UICollectionView was supported by two extensions: UICollectionViewDataSource and UICollectionViewDelegateFlowLayout.

Data Source
This extension was not declared explicitly as being of type "UICollectionViewDataSource" because in later versions of Swift, it is redundant to do so. However, the functionality it provides remains the same. For every item in the array, a UICollectionViewCell is created. In the case of our application, that

was a UIImageView. The image used was constant throughout—an image of a post it with a thumb tack. The properties of the cell were always that the background was clear and the image was generated with a UILabel. The value on the label is always the title of the bulletin at the same index as that of the cell.

On click of any of the cells, the user will be redirected to the storyboard for the viewing of a specific bulletin.

Flow Layout Delegate

This extension allows for the customisation of the cells in terms of size and row count. We customised it so that each row would have at most two cells, distributed evenly in each row.

Figure 1
Mock-up of UI when there are no bulletins

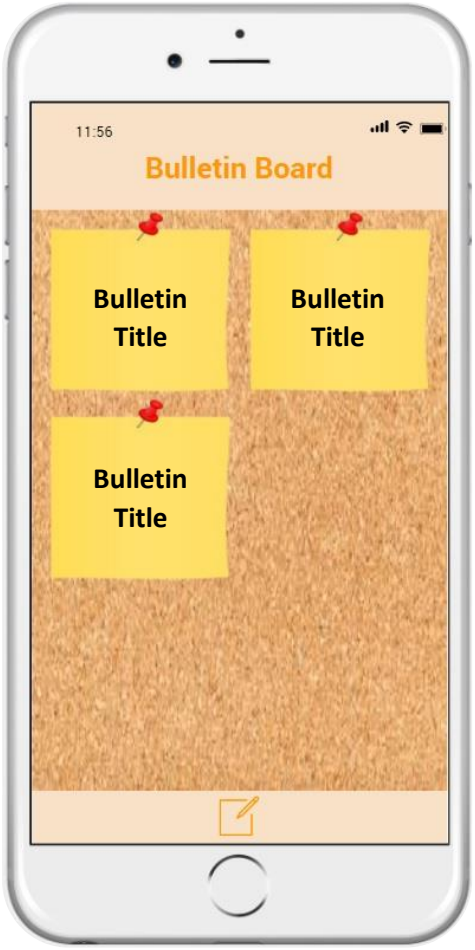


Figure 2
Mock-up of UI when there are bulletins

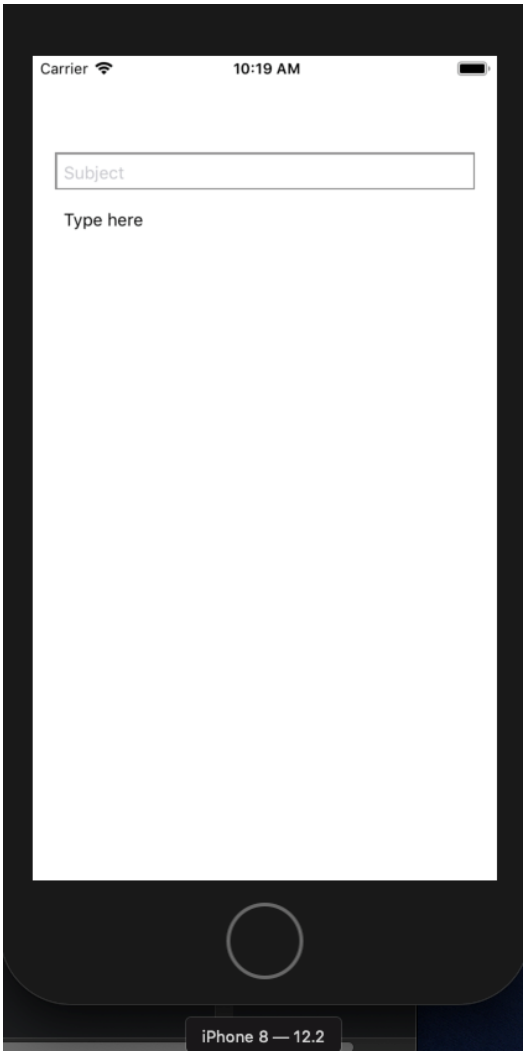
Creating a New Bulletin

Firebase

The information for bulletins is stored on firebase database. Once the database is modified by creating a bulletin, the app will be notified by an observer that passes a snapshot of the data.

Create a bulletin

When the user touches the new post button, the create bulletin screen is launched. From this page the user must enter a subject and a body for the bulletin. The current user is retrieved from firebase auth feature.



Pushing to the database

Once the done button is tapped on the navigation bar, the data is extracted from the fields and passed to the ‘createAndPush’ function. This function uses a reference to the bulletins path of the firebase database and uses childByAutoId to create a new bulletin with a unique identifier.

Viewing A Specific Bulletin

Setup

To begin, we imported Firebase to facilitate the storing of bulletins. The SinglePostView was used to add functionality to its associated storyboard. The UILabels were referenced, a database reference was created, and other necessary variables were declared that would be used throughout the code following.

onViewWillAppear()

Each time the view is loaded, the ID of the bulletin is found in the database and the values are loaded onto the screen.

Actions

From this view, the user can perform four actions, represented by buttons:

1. Edit a Bulletin
   When the user selects the “edit” button, a dialog asks them if they wish to update the body of the post.

   If they do, they can edit the body from within the pop-up dialog window and the function pictured below shows how the bulletin is updated.

   On update, the time stamp the update is made is saved to overwrite the date of the bulletin’s creation. A post dictionary is then created to store the values of the updated bulletin. The bulletin with the current id is then found in the database and the values of that bulletin are updated.

   The values on the view in the storyboard are also updated with the changed information.

2. Delete a Bulletin
   The delete button will first ask the user if he/she is sure he/she wishes to delete the bulletin. If the user clicks cancel, it will close the dialog and cancel the process. If the user clicks delete, the bulletin is cleared from the screen and also removed from the database.

3. Share a Bulletin
   When the share button on the story board is clicked, the author, title, date and body of the bulletin currently in view are put together and pushed to an external application of choice that can accept textual values.

4. Navigate Backwards to All Bulletins
   The back button will change the view from the single post view storyboard to the main view with all bulletins. This was implemented from the storyboard.

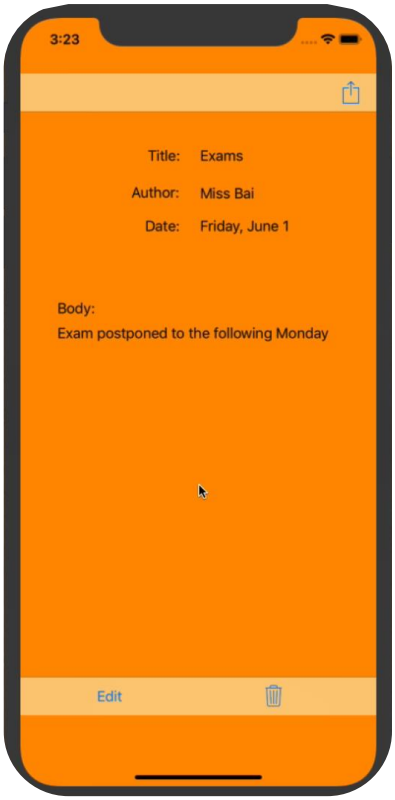


Figure 3
Storyboard for Viewing a Specific Bulletin

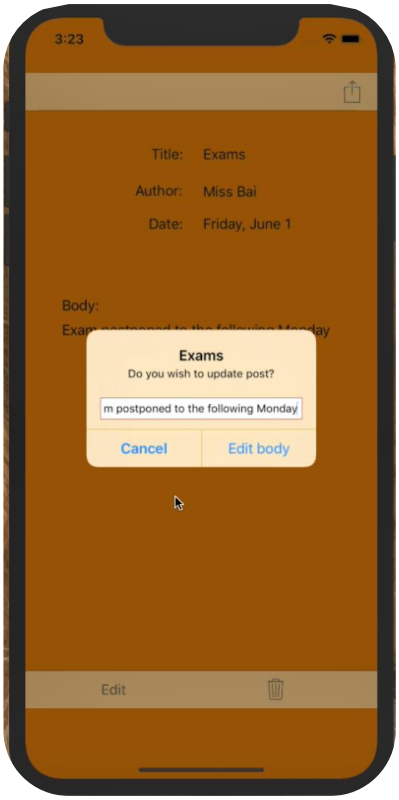


Figure 4
Dialog for Editing a Bulletin

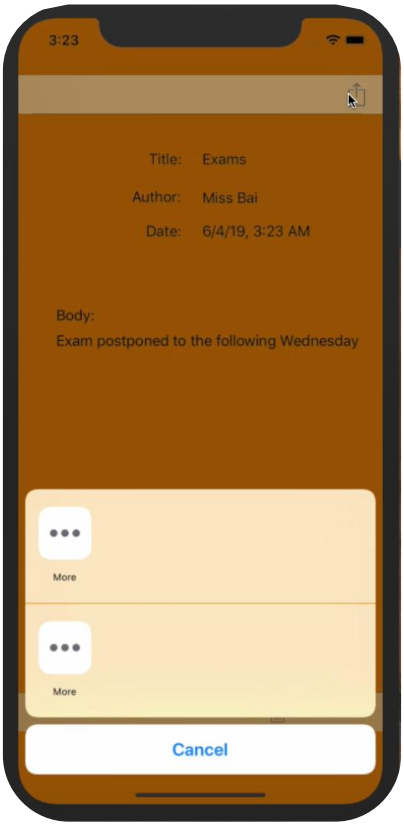


Figure 5
Sharing a Bulletin

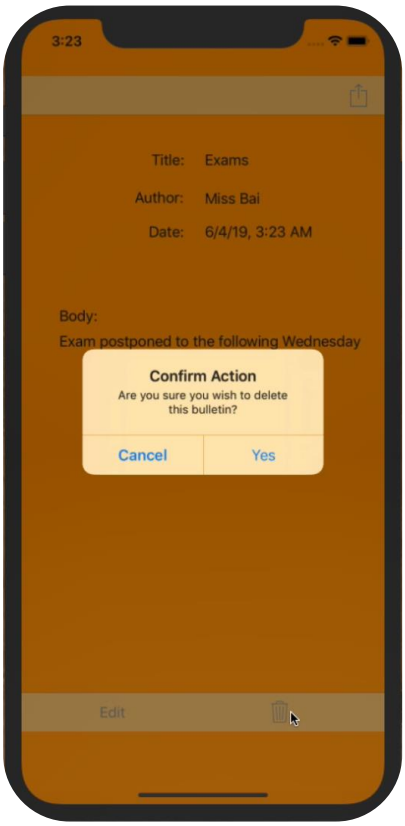


Figure 6
Alert Dialog for Deleting a Bulletin

Challenges

The Project in its Entirety

- Due to VPN restrictions at the time of development, it took longer than anticipated using trial and error to get functional code.

- Besides the Apple Developers website, there were very few other useful resources that could help when we came across issues.

Viewing All Bulletins

“Despite having followed the directive from various sources, the ID of the bulletin clicked would always be nil when trying to access it from the SinglePostView even after it had been passed. So, the hard-coded value had to remain in order to see the SinglePostView in action.”

— Khadeja

Creating a New Bulletin

“For developing applications to be used in China, it has become clear that firebase is not a suitable option for storing and pushing data. It was especially difficult to test the create bulletin feature because it required that I connect to a VPN each time I needed to test.”

— Scott

Viewing A Specific Bulletin

“Due to not having access to a Mac or iOS system, at the time when I was working on my section of the project—when I did have access—my teammate had not yet finished her section (view all bulletins). Because of this, I did not have an ID passed to me and had to hard code the ID of a bulletin in the database that would be in the ‘view all bulletins’ storyboard had it been functional at the time.”

— Mischka