SEED-GO

SWEN3004 – WEB & MOBILE APPLICATION DEVELOPMENT II

Abstract

This report describes the development of a content grabber for the SEED Moodle Platform. 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

WEB & MOBILE APPLICATION DEVELOPMENT II: 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	Courses
			Web Service
620098443	180908	Clarke, Khadeja Renee	User Authentication
			Viewing an Announcement
			Web Interface
620100135	180913	Neill, Mischka Zoila	Viewing all Announcements
			Barcode Scanner

Table of Contents

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 SWEN3004 – Web & Mobile Application Development II.

Design Description

Overview

A native Android application built using Java and Kotlin, that employs an SQLite database and communicates with a web service.

Detailed Description

A content grabber for the SEED Moodle platform that scrapes only content relevant to the students enrolled in the BSc Software Engineering (Mobile Application Technologies) offered by The UWI-CIIT. It should, on the successful scan of a barcode, send all data scraped to a Flask web service that will be displayed on a web interface.

Goals

User Authentication
Allow valid users to login.

Viewing all Announcements
Retrieving and displaying all announcements relevant to the app's target users.

Viewing a specific Announcement
Retrieving and displaying the specified announcement.

Viewing all Courses
Retrieving and displaying all courses the current user is enrolled in.

Viewing a specific Course
Retrieving and displaying course material for a specified course.

Communicating with a Web Service
On successful scan of a barcode, send all announcements and courses as JSON to the web service.

Implementation

User Authentication

Login

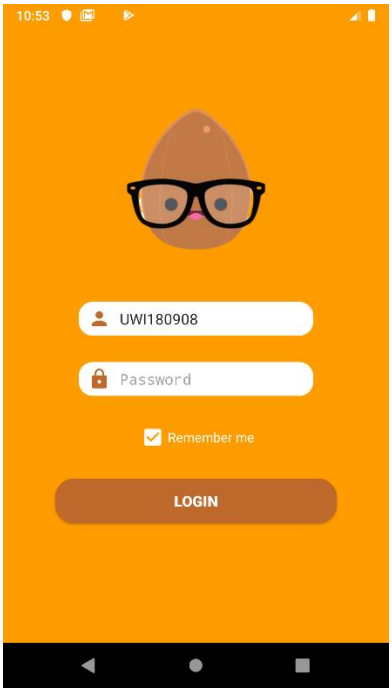
The application takes the values extracted from the EditText widgets and sends them to SEED for authentication. There is a CheckBox widget on the Login Form of the application that works with the application's SharedPreferences. If the CheckBox widget is checked, then the user's username is stored in SharedPreferences. This allows for the user needing only to input his/her password the next time the application is launched. If left unchecked, nothing is stored.

Field validation takes place on all input fields.

While the application is signing in, a circular ProgressBar appears to indicate that there is some processing taking place in the background.

A Snackbar is used to indicate any success or failure messages.

SEED uses session cookies. On a successful login, these cookies are stored and used throughout the application to allow for navigation across webpages.



Logout

A user may logout from the application by selecting it from the options menu in the toolbar. An AlertDialog will appear to handle user decisions. If the user does indeed desire to logout, on click of the positive button, he/she will be redirected to the Login Screen. The session cookies will also be deleted. On click of the negative button, the AlertDialog will simply close.

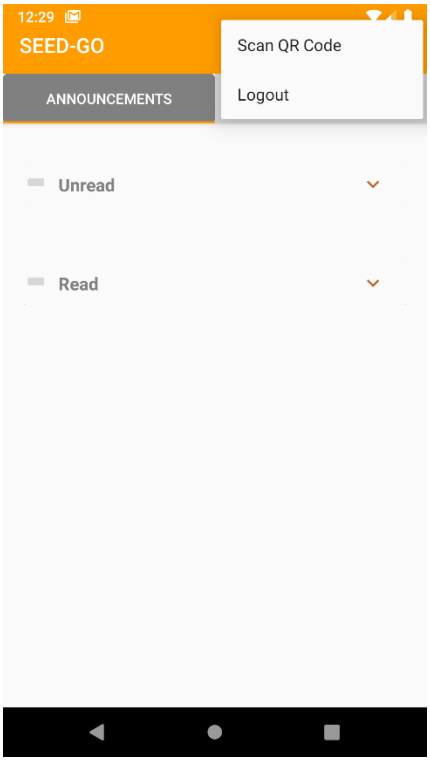


Figure 2 Logout in Options Menu

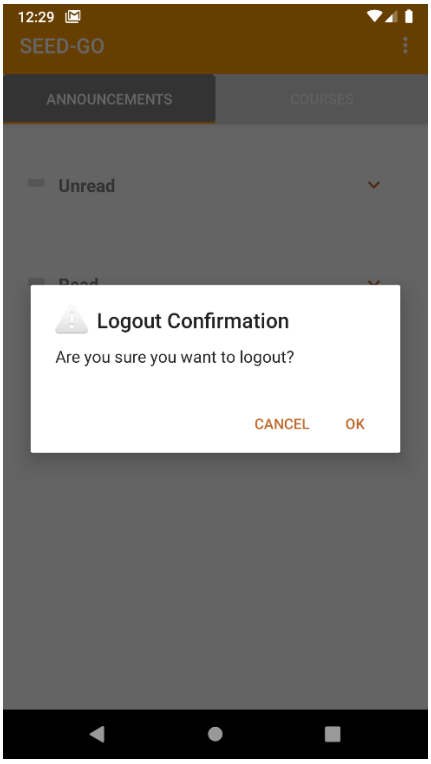


Figure 3 Logout Confirmation Dialog

Announcements

Viewing All Announcements

An announcement pulled from SEED will contain its title, date of creation, its ID and the link to read more about the announcement.

Once the view is accessed, the different lists that respectively add to the 'read announcements' recycler view and the 'unread announcements' recycler view are cleared, widgets are instantiated, and Jsoup Tasks that scrape the webpage are called to action.

The Jsoup process is done in the background of the application. Within this class we deal with all processing of the webpage's contents.

The elements of the webpage are then broken down and analyzed to find the relevant announcements to UWI students. Once those announcements are found, the announcement title, creation date, ID and link are then stored to a new announcement object.

The announcements are then sorted into read or unread announcement lists. This is facilitated by an SQLite database which stores the ID of all announcements previously read.

If the database is empty, all UWI announcements on the page are added to the unread announcements list. If the database has contents then the database is checked to see whether each announcement is seen or unseen.

To check if the announcement is seen, the ID of the announcement from the webpage is checked against each ID in the database. If it is not found in the database it is added to unread, if it is in the database it will be added to the read announcements list.

The lists of announcements should be loaded after the Jsoup tasks are ran.

When the 'read more' button is clicked, if the announcement is in the unread announcements list is then added to the read announcements list. If it's already in the read announcements list nothing is done with the announcement.

An expandable card view was used for the recycler views so as to hide the unwanted lists of announcements when not needed, whether it be the already read announcements or the unread announcements.

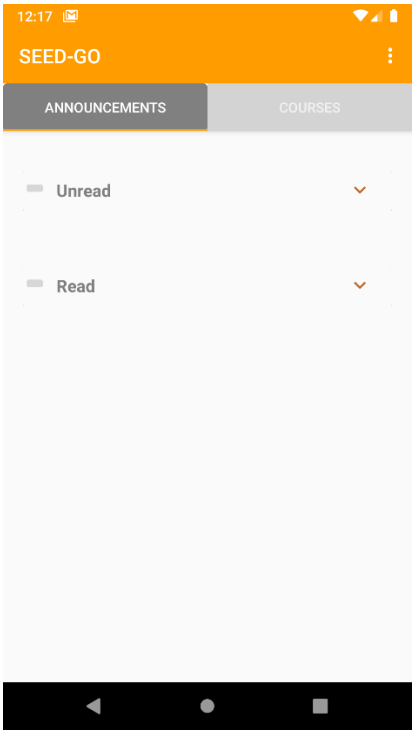


Figure 4 Viewing All Announcements: Collapsed

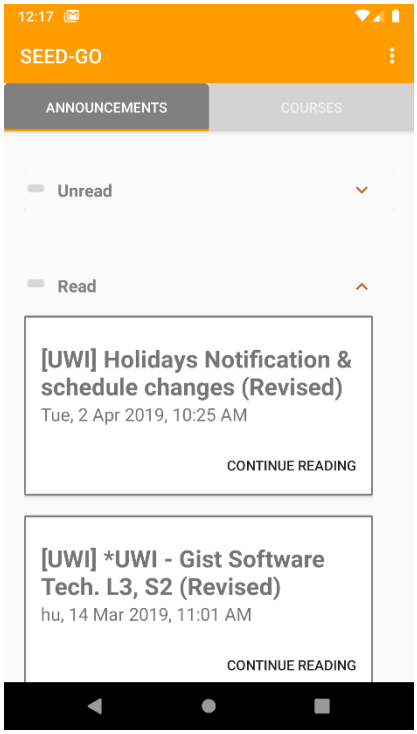


Figure 5 Viewing All Announcements: Expanded

Viewing A Specific Announcements

On click of the “Continue Reading” in the view holder of the relevant announcement, the user is redirected to the full announcement. An announcement consists of (from top to bottom):

1. A title
2. A date
3. An author
4. A body
5. A(n) attachment(s) (optional)

There exist two special cases for announcements:

1. Announcements with tables
2. Announcements with attachments

Announcements with tables

The Android TableLayout widget is used to display the content of the HTML table. This is created dynamically because not all announcements have tables. In order to determine whether an announcement has a table, we check for the existence of the HTML table tag.

A limitation of the TableLayout widget is that it does not support rowspan. So, we had to create our own implementation. For a cell that has the rowspan HTML attribute, we make note of its address, its value, and the size of the rowspan. It is important for us to know the address of the cell because rowspan is applied to cells in subsequent consecutive rows in the same column. The size of the rowspan is crucial in determining the number of times the value in the cell is to be duplicated. Say, for example, if we know that the cell at A1 has a rowspan of 2, then we know that the value in cell A2 must be the same as cell A1.

Announcements with attachments

Announcements are displayed in an ExpandableCardView. This is a third-party library found on GitHub. The implementation of the ExpandableCardView is such that it takes an inner view. The inner view used in the case of the attachments was a Custom ListView. A Custom ListView was used instead of a standard ListView to facilitate placing a download button beside the file name. We wanted to allow users to download any attachments linked to an announcement. Placing a download button indicates clearly that a file is available for download. Had we simply used the standard ListView, it would not have been obvious to the user that a file is available for download.

DownloadManager was used to allow the user to download the respective file. However, in order to do so, the application must have storage permissions. Given that the user grants such permissions to the application, all downloaded files are saved to the application folder SEED-GO.

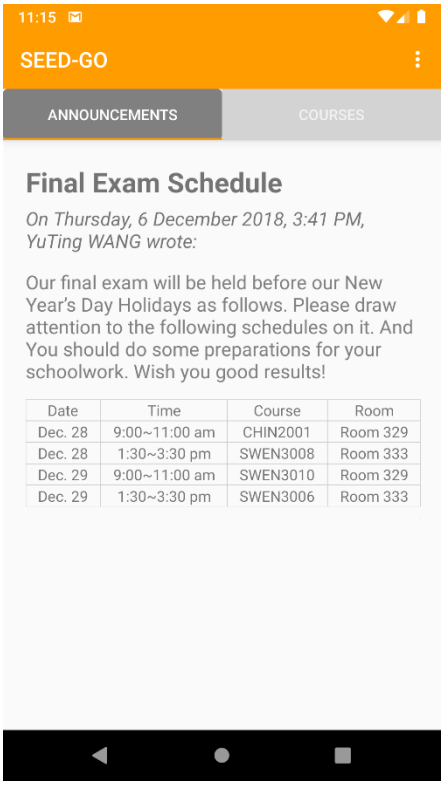


Figure 5 An announcement with a table

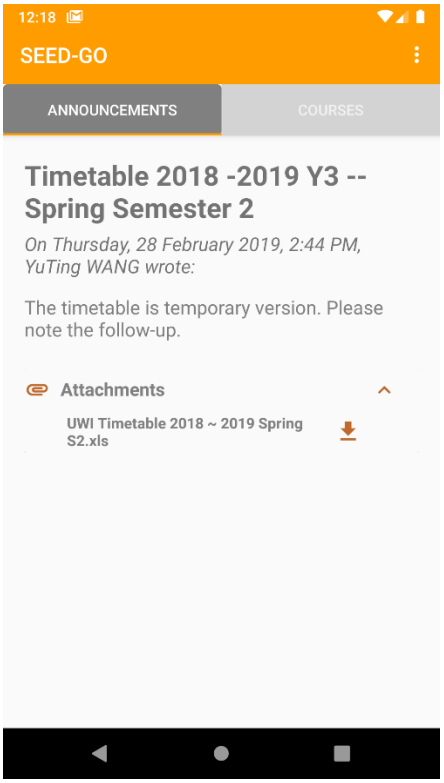


Figure 6 An announcement with an attachment

Courses

Viewing All Courses

Once the user logs into the application, an AsyncTask commences to retrieve the current information for each course specific to the seed account used to log in. If the user navigates to the course tab before this is complete, a progress bar is shown. Once the information for a course is ready to be displayed, it is displayed to the user in a MaterialCardView. A course is defined by the following attributes:

1. A title
2. A list of topics
3. A list of files
4. A list of assignments

Viewing A Specific Course

To display the different types of data within a course, the kotlin-groupie library was used to create a custom expandable course card that displayed information within expandable and collapsible groups. By default, each course card is collapsed. On click of a course card, that

card will expand to reveal information about its available topics, the files associated with those topics and assignments. On click of a currently expanded course card will collapse that course card again.

While a course is expanded, a download option is available beside each file under a topic. On click of the download icon will and create a relevant download folder and queue that file for downloading.

Communicating with Web Service

Web Server

The server was created using the python flask framework. A RESTful web service that uses the methods below:

- POST
  When the QR code is scanned, the android application should send JSON data on course and announcement information.
- GET
  The GET requests returns a JSON of course data or announcement data, depending on the URL.

Web Interface

The web interface was created using HTML/SCSS/Javascript. Like the mobile application, it uses a tab layout with a tab for Announcements and a tab for Courses. For Announcements, like the application, it separates Read Announcements from Unread Announcements with each announcement being displayed on a card. For Courses, it shows the name of the course, which when clicked, launches a popup overlay that will hold all course materials.

JSON Parsing

When the barcode is scanned, the announcements from the announcements fragment and the courses from the courses fragment are sent to the server via Volley.

When announcementsServer() is called after scanning the barcode, a connection is made with the server and both read and unread announcements lists are formatted from objects (gson to Json) objects to string.

When coursesServer() is called after scanning the barcode, a connection is made with the server and both read and unread announcements lists are formatted from objects (gson to Json) objects to string.