Table of Contents

Contents

Task 2 Part C – C868 Software Development Capstone

Application Design and Testing

Design Document

Class Design

The classes that were used in this application were chosen to closely model the data that will be presented to the user. Classes that were implemented in this application were the following:

- **Assessment Entity** – this will model the assessments that the student will take during their academic career.
- **Course Entity** – the information for the course that the student is is/was enrolled in.
- **Course Instructor** – instructor information for the course.
- **Note Entity** – note information that will be associated with a course.
- **Term Entity** – term information.
- **User Entity** – user information that will only contain the password and username.

The diagram below shows the classes and how they will relate to each other in the application.

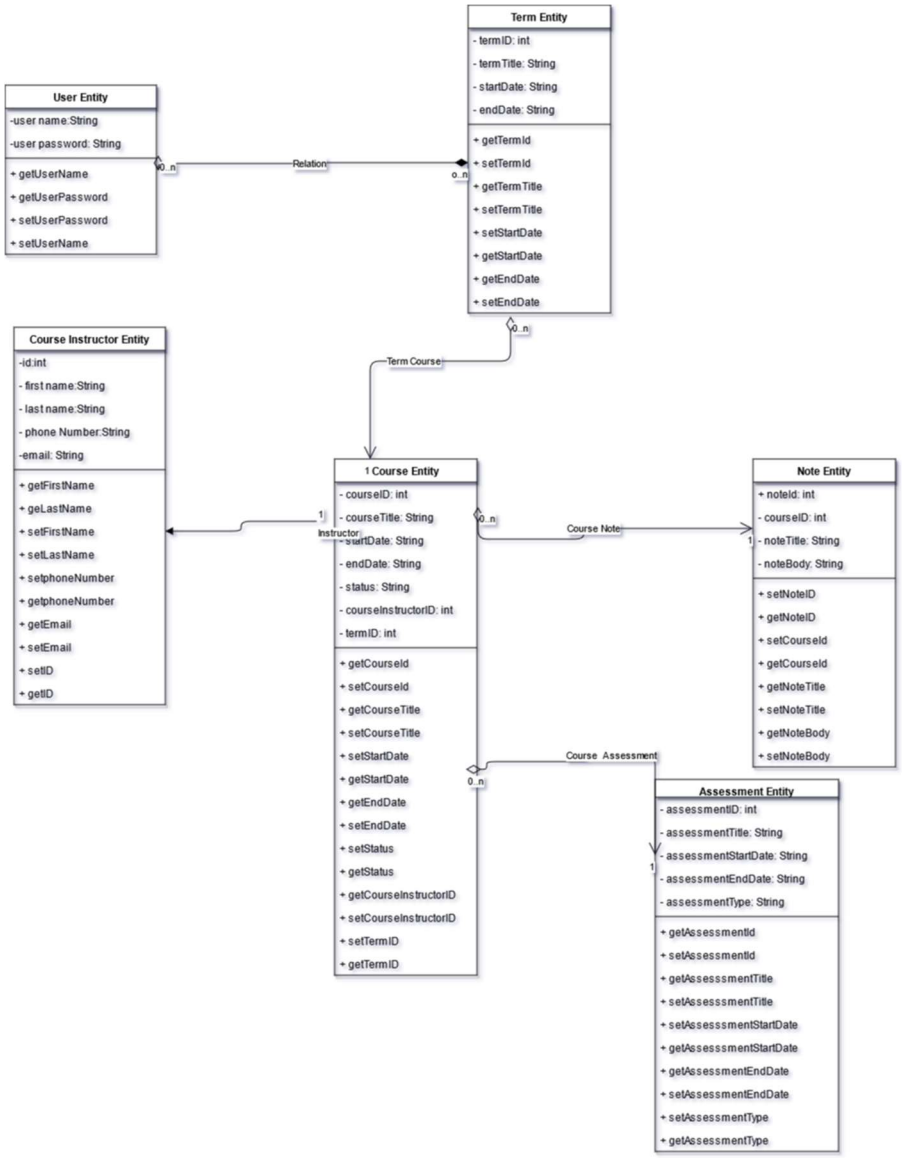
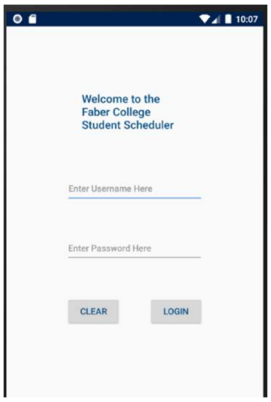


Figure 1 Class Diagram for Application Entities

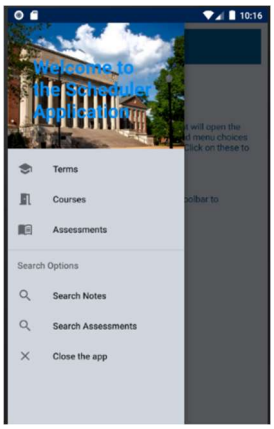
UI Design

The images that will be presented in this section will give the general impression as to what the user interface will look like once the application is completed. The screen below will be represented example of the functionality and the general flow of the application.



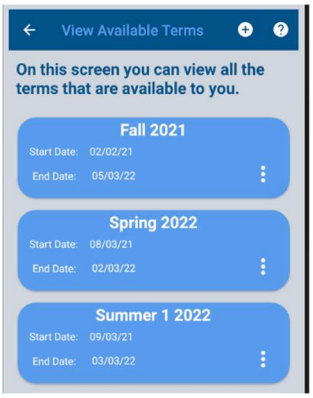
This is the design for the Login screen. This screen will allow the user to login securely to their application. Information required for this screen will be the username and their password.

Figure 2 Login Screen



This will be the main screen for the application. The user will be able to navigate to any of the major areas of the application. The major areas of the application are Terms, Course and Assessments and the search features.

Figure 3 Main Screen



This screen is the main Terms screen. The user will be able to view all the terms that are available to them. Clicking on the card will take them to additional screens that will provide more information about the term. The screen will also provide a way to add a new term to the user's schedule.

Figure 4 Main Terms Screen

Accomplished by clicking the plus sign at the top of the screen.

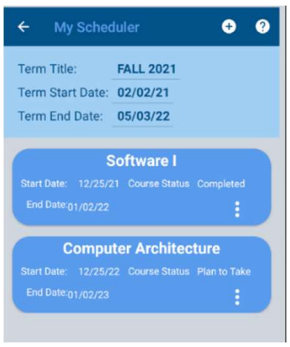


Figure 5 Term Details Screen

This screen will be displayed when the user clicks on the three dots in the card (see **Figure 4**). This screen will display a more detailed look at the term. It will display term information as well as any courses that is currently associated with the term.

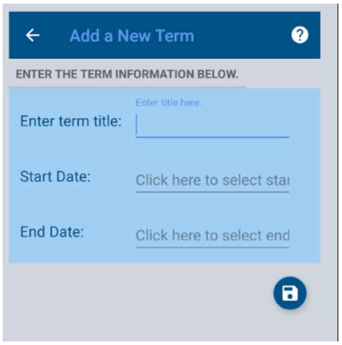


Figure 6 Add Term

This screen will allow the user to enter a new term. Information for the term will entered in the appropriate fields. Information is term title and the start and end dates

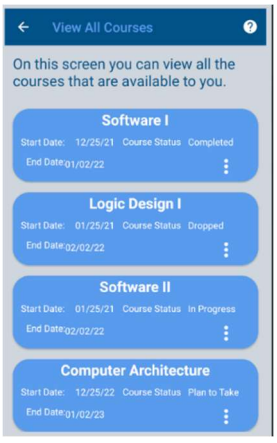


Figure 7 Courses Main Screen

This screen will be accessed from the screen in **Figure 3**. This screen will present a list of all courses that are entered by the user. It will give general information about the course. For example, it gives a title and the dates of the course.

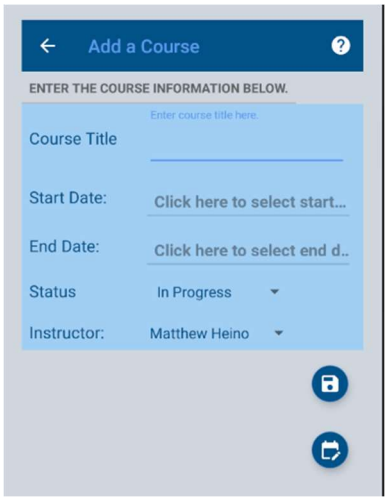


Figure 8 Add Course

This screen will allow the user to add a new course. This screen will also serve as the basis for viewing information about the course and will provide the ability to update the course. The only difference is the "mode" of the screen. It will either be update or add.

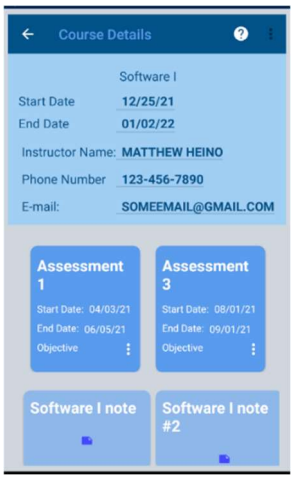


Figure 9 Course Details Screen

The Course Details Screen will show information about the course. It will show any assessments (dark blue card) and notes (light blue card) that are available to the user. Clicking on the three dots for the assessment will bring up more detailed information about the assessment. Clicking on the folded page will bring up more information about the note.

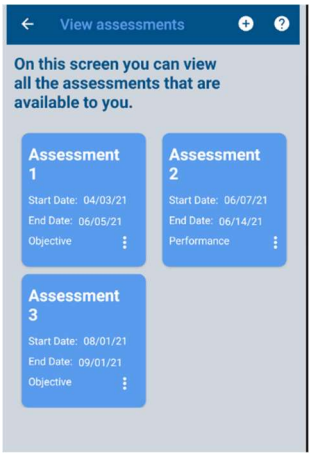


Figure 10 Assessment Main Screen

The screen will provide the user with all the assessments that are entered by the user. It will provide a quick way to look at any assessment.

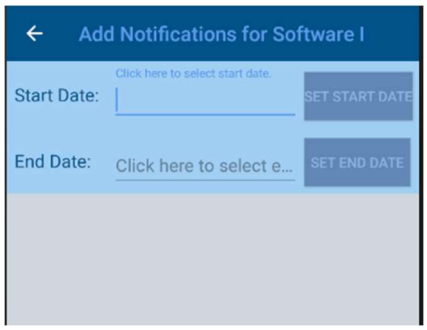


Figure 11 Notification Screen

The Add Notification screen will be used by the user to add notifications for the assessment. The user will be able to choose start and end dates for the assessment.

Unit Test Plan

Introduction

Purpose

The testing procedure will focus on the Date utility (DateUtil.java) class that was created to validate the dates that are entered by the user. The user will be allowed to enter dates in one of

two ways. The first way will be through the edit text field that is associated with either a start or and end date. This date will be checked to make sure that it confirms with the date format that is currently accepted by the application.

Overview

The ***DateUtil*** will provide all the validation for the date strings that will be entered by the user. A correct date string is important because all notifications and other date dependent events rely on a correct date to function properly. Also, dates for a course must fall within the time frame of the term. The Term will not be checked as dates could vary widely and as of this documentation no set guidelines have been provided to limit the scope of these dates.

Test Plan

Items

The tests will require the following to be run. Test dates that both test for valid and invalid inputs. These will be used to test the proper functioning of the methods that are in the class.

Features

The features that will tested will be methods that are currently contained in the **DateUtil** class. The methods that will be tested will be the following:

- **isDateBetween()** – method to check to make sure the dates given fall between the accepted dates. This method is used to make sure course dates, assessment dates fall within their respective timeframes. For example, does the start and end date for a course fall within the time that is given for the term in which it is currently in.

- **isValidDateString()** – method to check to make sure that a given date string is in a valid format. The current valid format is in the following form “"MM/dd/yy”.
- **isWithinOneWeek()** – method used to filter the assessments based on whether they are to occur within one week of the *current* date.
- **isWithinOneMonth()** - method used to filter the assessments based on whether they are to occur within one month of the *current* date.

Deliverables

The tests will yield the following deliverables: a console confirmation that the tests have proved the validity of the method and that it is functioning properly. No physical deliverables will be provided upon successful completion of the test.

Tasks

The tasks that are needed to accomplish the testing of the **DateUtil** class are the following:

1. In Android Studio press on the method or class that will be tested. Press Ctrl+Shift+T to bring up the dialog box.
2. Select Create a New Test.
3. Choose methods that will be tested. In this instance the test will include all method that are found in this class.
4. Select the “test” directory. The test file will be created located in the following directory app→C196.mheino.mysceduler(test)→MyUtils→DateUtilTest. See the following Figure.

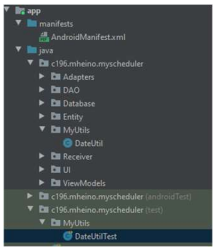


Figure 12 Test File Directory

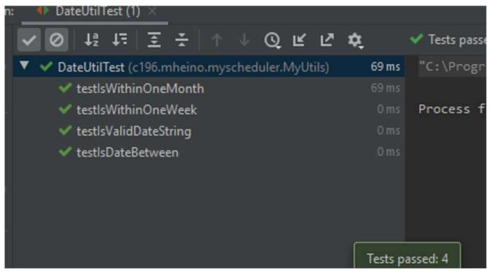
5. To run the test right click and select the run command.

Needs

In order run this test you must be running the test in Android Studio. The release of Android Studio that this test was performed in was Android Studio 4.2. No other applications are needed to perform the tests to completion.

Pass/Fail Criteria

To determine if the methods passed a unit test. The test was performed using a few different types of input date strings. The test was considered passed when the test was shown as in the screenshot below.



Specifications

The code that was used to test the ***DateUtil*** class is given below along with the class being tested for comparison purposes.

```
public class DateUtilTest extends TestCase {

    DateUtil myNewDateUtil = new DateUtil();

    String startDate = "05/05/21";
    String endDate = "08/25/21";
```

```
    String checkDate = "06/15/21";

    String inValidDateString = "005-23-21";

    public void testIsDateBetween() {
        assertTrue(myNewDateUtil.isDateBetween(checkDate, startDate, endDate));
    }

    public void testIsValidDateString() {
        assertFalse(myNewDateUtil.isValidDateString(inValidDateString));
    }

    public void testIsWithinOneWeek() {
        assertFalse(myNewDateUtil.isWithinOneWeek(checkDate));
    }

    public void testIsWithinOneMonth() {
        assertTrue(myNewDateUtil.isWithinOneMonth(checkDate));
    }
}
```

Code for testing the DateUtil Class

```
}

/** This method will check to see if a date falls within a given date of ...*/
public boolean isDateBetween(String checkDate, String startDate
        , String endDate)
{
    boolean isValid = false;

    try{
        LocalDate startDateDTF = LocalDate.parse(startDate, dtf);
        LocalDate endDateDTF = LocalDate.parse(endDate, dtf);
        LocalDate checkDateDTF = LocalDate.parse(checkDate, dtf);

        if(checkDateDTF.isAfter(startDateDTF.minusDays(1))
                && checkDateDTF.isBefore(endDateDTF.plusDays(1)))
        {

            isValid = true;
        }
    }
    catch(DateTimeParseException dpe){

        System.err.println("Invalid date!");
    }
    return isValid;
}

/** Method to determine if a given string is a valid date.  This method ...*/
public boolean isValidDateString(String checkString){

    try{

        LocalDate.parse(checkString, this.dtf);
    }
    catch(DateTimeParseException dtpe){

        return false;
    }

    return true;

} // end isValidDateString

/** Method to determine if a given date string falls within one week.  This ...*/
public boolean isWithinOneWeek(String date){

    boolean isWithinOneWeek = false;

    DateTimeFormatter dtf = DateTimeFormatter
            .ofPattern( pattern: "MM/dd/yy", Locale.ENGLISH);

    LocalDate currentDate = LocalDate.now();

    try{

        LocalDate dateDTF = LocalDate.parse(date, dtf);

        if(dateDTF.isAfter(currentDate.minusDays(1))
                && dateDTF.isBefore(currentDate.plusDays(7)))
        {

            isWithinOneWeek = true;
        }
    }
```

The DateUtil class code (Partial)

Procedures

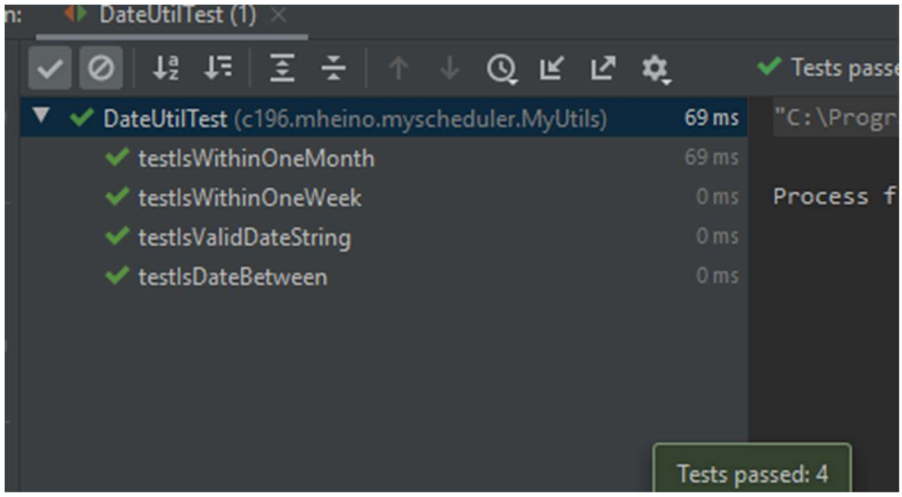
The procedure that was used to perform the tests used the following steps:

1. Devised test data that could be used to test Boolean results of the method. Since the methods return a Boolean true or false. Needed to devise both values that will elicit a proper true or false response.

2. Utilizing Android Studio created the code that would make use of the **assertTrue/assertFalse** functions. These methods will return a Boolean that is appropriate based on the given date strings.
3. Ran the tests with given data and looked at the results.
4. If the results were given as valid no further action was necessary. If there was an invalid result additional test data would be created to narrow down the problem with the method. During these tests, all test data produced the desired and expected results.

Results

The results were favorable. None of the test data produced an error in the methods that were tested. The screenshot below shows the results of the test.



C4. Source Code

The source code for this application will be in the following folder. The folder name is: **Project868**. This project folder is included with the Zip submission of the entire project.

C5. Link to Live Version

The executable version of the application can be found in the debug folder along the following file path: **Project868→Project→app→build→outputs→apk→debug→ app-debug.apk**.

The APK will be able to run if you drag and drop into the emulator. Please note that this application was tested using Android 8 using a Nexus 5x. To run the application please use the following for credentials:

Username: test

Password: test

Make sure there are no spaces at the beginning or end of the entries.

Note: The user guide is a separate document that is included with the submission. The maintenance guide is a separate short document.