Software Engineering Capstone

Task 3

App Documentation

Capstone Proposal Project Name:	Student Planner Mobile Application
Student Name:	Vittone

Table of Contents

APPLICATION DESIGN AND TESTING

This document discusses the design and testing of the Student Planner Android Mobile Application. The design documents include structural and user interface designs of the app. The structural designs include an entity relationship diagram and a class diagram. These designs show the entity relationships and explain how the classes work with each other. The user interface designs include a low fidelity wireframe, a high fidelity wireframe, and a storyboard of the application. The UI designs show a visual layout of the app and how the app should flow. Furthermore, the testing section of this document explains the unit tests that have been performed to ensure that this application works as it is expected to. This section includes an introduction, the test plan, specifications, procedures, and results. In addition, there is a link to where the mobile app is hosted, a link to the GitLab repository, and an image of the GitLab branch history.

DESIGN DOCUMENTS

Structural Designs

The structural designs include an entity relationship diagram and a class design. These designs show how the application is structured and how the components interact with one another.

ER Diagram:

The entity relationship diagram symbolizes how the entities relate to each other. In this application there are two entities: class and assignment. This diagram shows the attributes of each entity. The class entity includes courseID, courseName, courseInstructor, courseStartDate, courseEndDate, and courseDescription. The courseID is an integer, courseName is a string, courseStartDate is a string, courseEndDate is a string, and courseDescription is a string. The courseID is also the primary key. The assignment entity includes assignmentID, assignmentName, assignmentDueDate, assignmentDescription, and courseID. The assignmentID is an integer, assignmentName is a string, assignmentDueDate is a string, assignmentDescription is a string, and courseID is an integer. The assignmentID is the primary key. Moreover, the class and assignment entity have a one to many relationship. This means that for one class, there are multiple assignments.

Class Design:

The class design demonstrates how the classes of the application work together. This design uses Crow's Foot Notation to symbolize the relationships between the classes. The text below describes the relationships between the classes. This information is also reflected in the class diagram included in this document.

- The MainActivity class has a one-to-one relationship with the CourseList class. There is only one MainActivity class that shows one CourseList class.

- The CourseAdapter class has a one-to-one relationship with the CourseList class. The one CourseAdapter is used to display the one list of courses on the CourseList's recyclerview.

- The CourseList class has a one-to-many relationship with the CourseDetails class. There is one list of courses that can show one or many courses. The courses displayed here each have their own course details.

- The AssignmentAdapter has a one-to-many relationship with CourseDetails. For each course, there is a course detail page. This page uses a recyclerview that is filtered to show all that course's associated assignments. There is one assignment adapter that is used to display each course's associated assignments.

- The AssignmentAdapter has a one-to-many relationship with the AssignmentDetails class. The one assignment adapter is used to display a list consisting of multiple assignments.

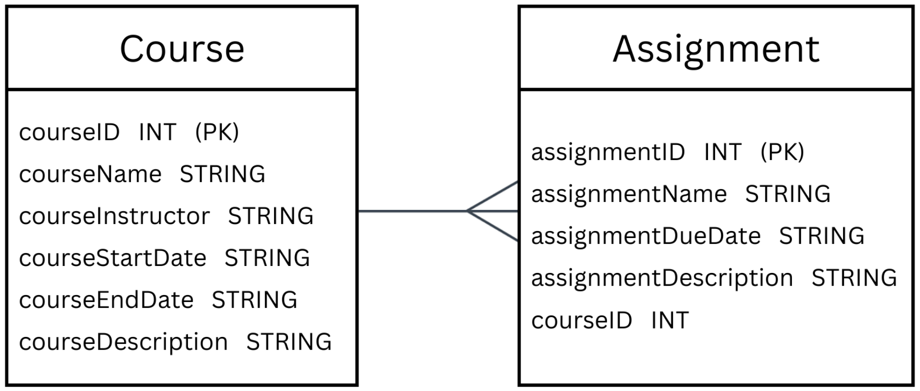
- The MyReceiver class has a one-to-many relationship with the AssignmentDetails class. The receiver is able to be used by multiple assignments to set alerts for their due dates.

- The MyReceiver class has a one-to-many relationship with the CourseDetails class. The receiver is able to be used by multiple courses to set alerts for their start date and end dates.

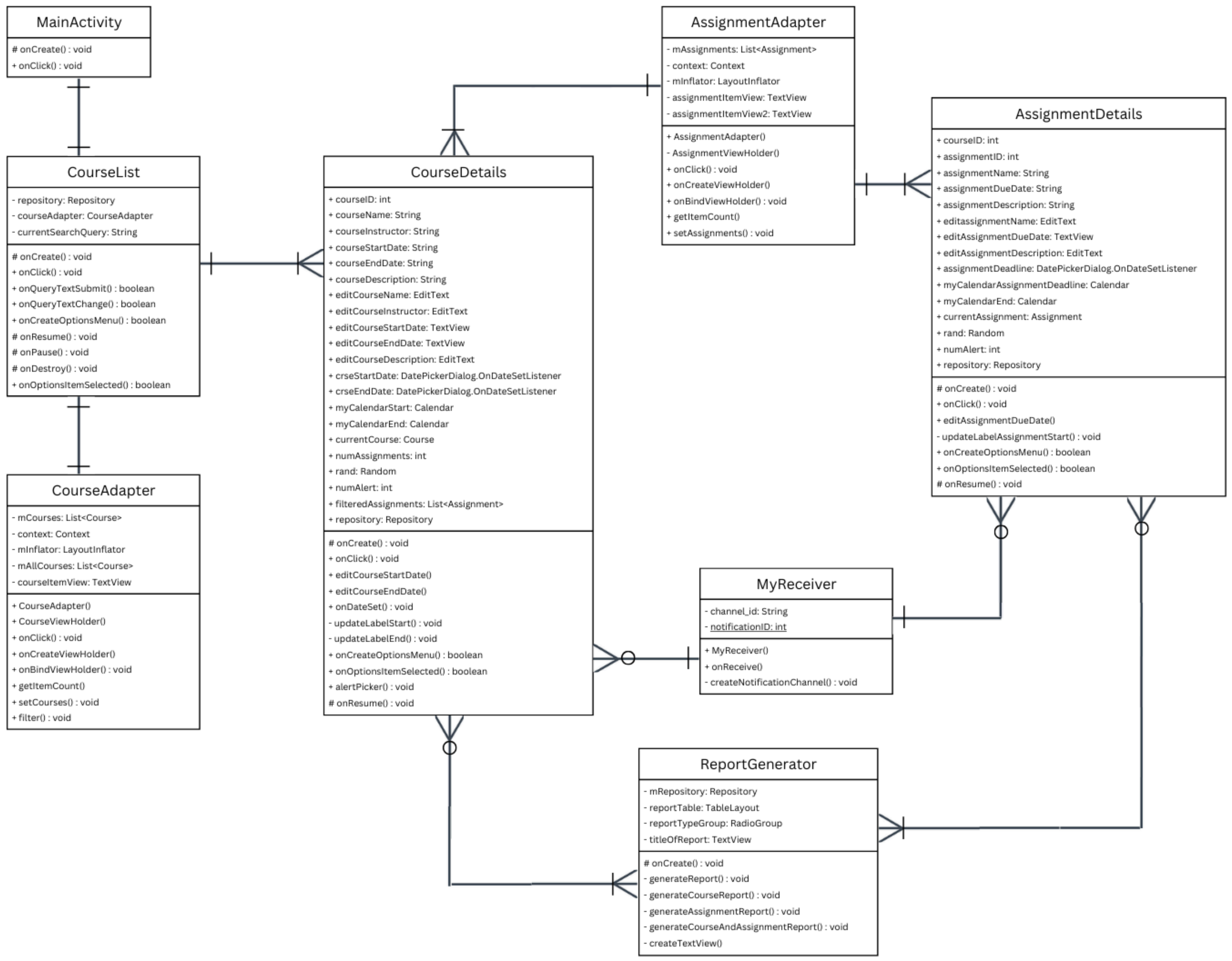
- The ReportGenerator class has a many-to-many relationship with the CourseDetails class. Multiple class details can be used to generate multiple reports. For instance, the "Class" report option will include all of the classes and their details. Another report option is "Class and Assignment" report. This will include many classes and their details.

- The ReportGenerator class has a many-to-many relationship with the AssignmentDetails class. Multiple assignment details can be used to generate multiple reports. For instance, the "Assignment" report option will include all of the assignments and their details. Another report option is "Class and Assignment" report. This will include many assignments and their details.

Entity Relationship Diagram



Class Design


MainActivity
# onCreate() : void
+ onClick() : void
CourseList
- repository: Repository
- courseAdapter: CourseAdapter
- currentSearchQuery: String
# onCreate() : void
+ onClick() : void
+ onQueryTextSubmit() : boolean
+ onQueryTextChange() : boolean
+ onCreateOptionsMenu() : boolean
# onResume() : void
# onPause() : void
# onDestroy() : void
+ onOptionsItemSelected() : boolean
CourseAdapter
- mCourses: List<Course>
- context: Context
- mInflator: LayoutInflator
- mAllCourses: List<Course>
- courseItemView: TextView
+ CourseAdapter()
+ CourseViewHolder()
+ onClick() : void
+ onCreateViewHolder()
+ onBindViewHolder() : void
+ getItemCount()
+ setCourses() : void
+ filter() : void
CourseDetails
+ courseID: int
+ courseName: String
+ courseInstructor: String
+ courseStartDate: String
+ courseEndDate: String
+ courseDescription: String
+ editCourseName: EditText
+ editCourseInstructor: EditText
+ editCourseStartDate: TextView
+ editCourseEndDate: TextView
+ editCourseDescription: EditText
+ crseStartDate: DatePickerDialog.OnDateSetListener
+ crseEndDate: DatePickerDialog.OnDateSetListener
+ myCalendarStart: Calendar
+ myCalendarEnd: Calendar
+ currentCourse: Course
+ numAssignments: int
+ rand: Random
+ numAlert: int
+ filteredAssignments: List<Assignment>
+ repository: Repository
# onCreate() : void
+ onClick() : void
+ editCourseStartDate()
+ editCourseEndDate()
+ onDateSet() : void
- updateLabelStart() : void
- updateLabelEnd() : void
+ onCreateOptionsMenu() : boolean
+ onOptionsItemSelected() : boolean
+ alertPicker() : void
# onResume() : void
AssignmentAdapter
- mAssignments: List<Assignment>
- context: Context
- mInflator: LayoutInflator
- assignmentItemView: TextView
- assignmentItemView2: TextView
+ AssignmentAdapter()
- AssignmentViewHolder()
+ onClick() : void
+ onCreateViewHolder()
+ onBindViewHolder() : void
+ getItemCount()
+ setAssignments() : void
AssignmentDetails
+ courseID: int
+ assignmentID: int
+ assignmentName: String
+ assignmentDueDate: String
+ assignmentDescription: String
+ editassignmentName: EditText
+ editAssignmentDueDate: TextView
+ editAssignmentDescription: EditText
+ assignmentDeadline: DatePickerDialog.OnDateSetListener
+ myCalendarAssignmentDeadline: Calendar
+ myCalendarEnd: Calendar
+ currentAssignment: Assignment
+ rand: Random
+ numAlert: int
+ repository: Repository
# onCreate() : void
+ onClick() : void
+ editAssignmentDueDate()
- updateLabelAssignmentStart() : void
+ onCreateOptionsMenu() : boolean
+ onOptionsItemSelected() : boolean
# onResume() : void
MyReceiver
- channel_id: String
- notificationID: int
+ MyReceiver()
+ onReceive()
- createNotificationChannel() : void
ReportGenerator
- mRepository: Repository
- reportTable: TableLayout
- reportTypeGroup: RadioGroup
- titleOfReport: TextView
# onCreate() : void
- generateReport() : void
- generateCourseReport() : void
- generateAssignmentReport() : void
- generateCourseAndAssignmentReport() : void
- createTextView()

User Interface Designs

The user interface designs include a low fidelity wireframe, a high fidelity wireframe, and a storyboard. These designs illustrate the user interface related aspects of the app. They are used to visualize the app's appearance and usability.

Low Fidelity Wireframe:

The low fidelity wireframe shows the basic layout of the user interface. There is a low fidelity wireframe for the Main Activity, Course List, Course Details, Assignment Details, and Report Generator page. It includes blocks and texts that represent the placements of elements on the app. This is a simple visual that helps explain where text, buttons, recyclerviews, and other elements will appear on the application.

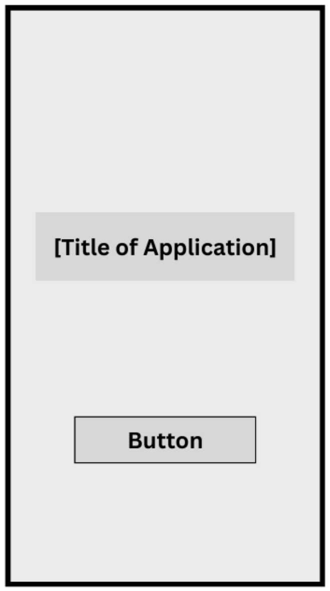
High Fidelity Wireframe:

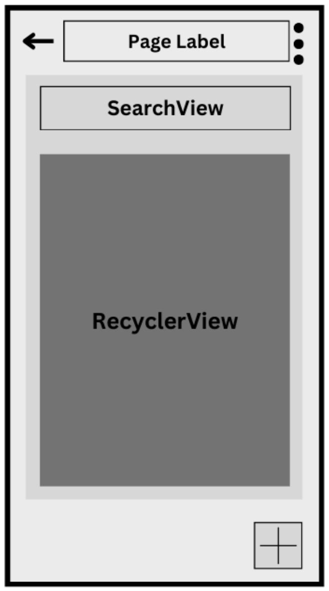
The high fidelity wireframe is a detailed mockup of the application's layout. There is a high fidelity wireframe for the Main Activity, Course List, Course Details, Assignment Details, and Report Generator page. Unlike the low fidelity wireframe, this design reflects how the final version of the app will most likely look. High fidelity wireframes include color, fonts, graphics, and typography. This is a highly detailed visual that represents how the finished product will look.

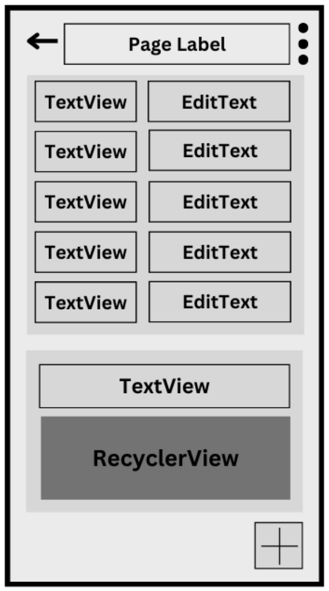
Storyboard:

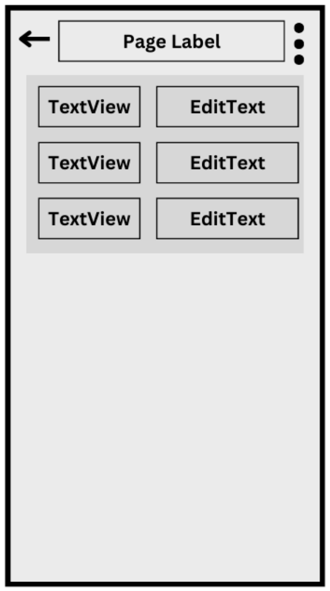
The storyboard layout demonstrates the flow of the app. This is a simple storyboard that shows the process of adding a new class, a new assignment, and generating a report. The first step is entering the app. This is done by clicking the button on the main activity page that says "Enter". To add a class, the user should then click on the plus sign button on the class list page. Once they do this, they will be able to enter their class information and select "Save Class". To add an assignment to that class, the user will need to navigate back to the class details page and click the plus sign button. Then they will need to enter their assignment information, click "Save Assignment" and then click "Save Class". Finally to generate a report, the user will need to open the menu on the class list page and select "Generate Report". They will be taken to the report generated page where they will be able to select the type of report they want to generate. In summary, the storyboard is a UI design that illustrates the flow of the app. It is important to note that this storyboard does not show all possible actions of the app. It demonstrates a very simple process of adding a single class and assignment and then generating one report.

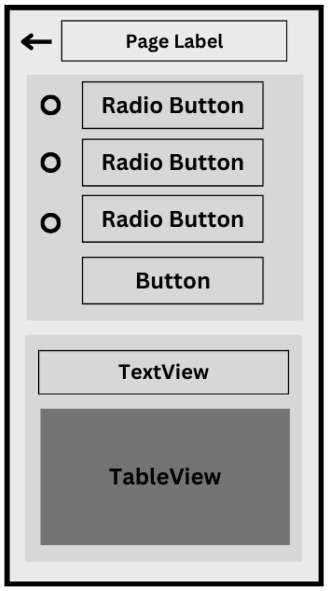
Low Fidelity Wireframe











MainActivity

- The app launches to this page upon startup
- Title of the application is shown
- The button is used to enter the app and navigate to the CourseList page

CourseList

- This page shows the list of classes on the recyclerview
- Uses a search function to allow users to search through the list of classes
- The floating action button is used to add a class

CourseDetails

- This page allows users to enter the class information
- The associated assignments will be listed on the recyclerview
- The floating action button is used to add an assignment

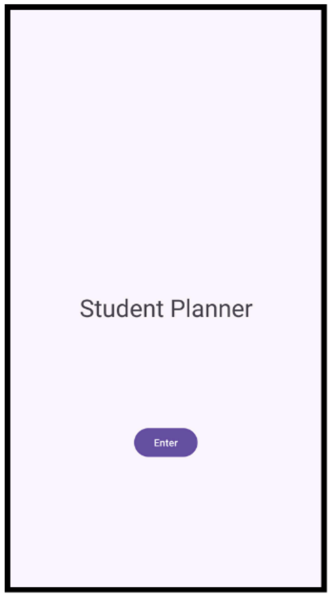
AssignmentDetails

- This page allows users to enter the assignment information

ReportGenerator

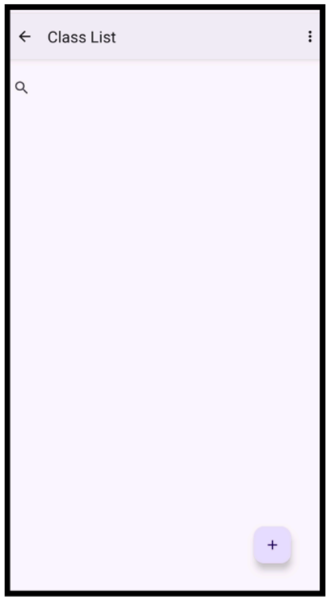
- This page allows users to generate various types of reports based on the class and assignment information they have entered

High Fidelity Wireframe



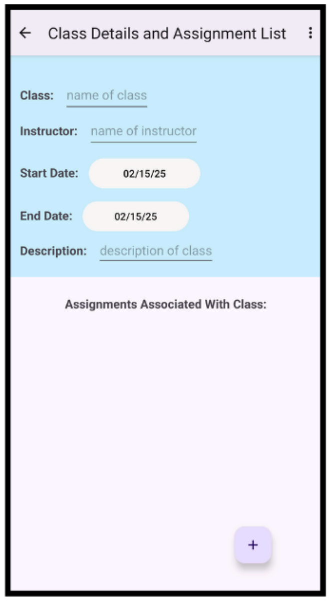
MainActivity

- The app launches to this page upon startup
- Title of the application is shown
- The button is used to enter the app and navigate to the CourseList page



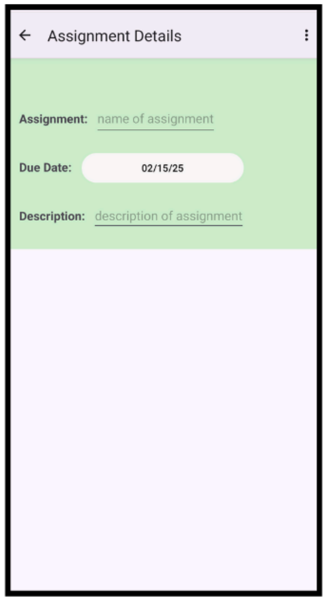
CourseList

- This page shows the list of classes on the recyclerview
- Uses a search function to allow users to search through the list of classes
- The floating action button is used to add a class


Class Details and Assignment List
Class: name of class
Instructor: name of instructor
Start Date: 02/15/25
End Date: 02/15/25
Description: description of class
Assignments Associated With Class:
+

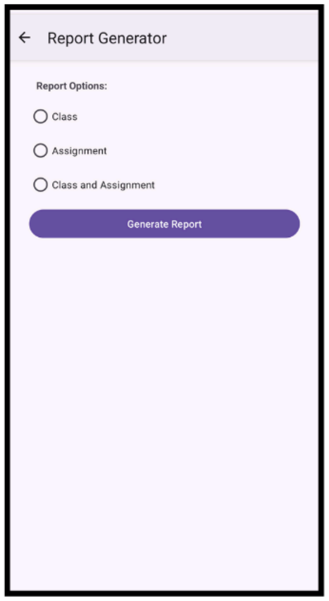
CourseDetails

- This page allows users to enter the class information
- The associated assignments will be listed on the recyclerview
- The floating action button is used to add an assignment


Assignment Details
Assignment: name of assignment
Due Date: 02/15/25
Description: description of assignment

AssignmentDetails

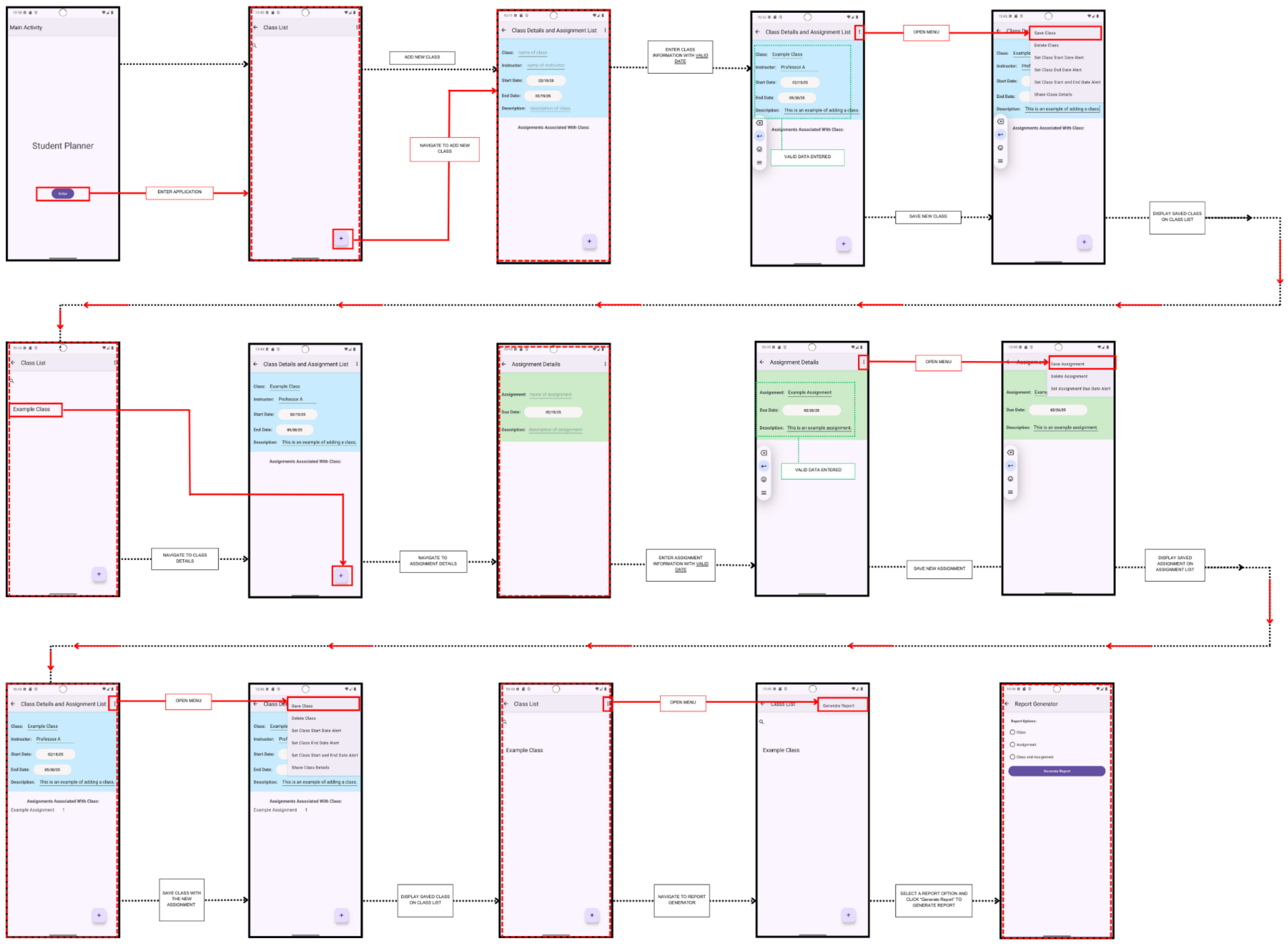
- This page allows users to enter the assignment information



ReportGenerator

- This page allows users to generate various types of reports based on the class and assignment information they have entered
- The table will appear once an option is chosen and the button is clicked

Storyboard


Main Activity
Student Planner
Enter
ENTER APPLICATION
Class List
ADD NEW CLASS
NAVIGATE TO ADD NEW CLASS
Class Details and Assignment List
Class:
Instructor:
Start Date:
End Date:
Description:
Assignments Associated With Class:
ENTER CLASS INFORMATION WITH VALID DATE
Example Class
Professor A
This is an example of adding a class.
VALID DATA ENTERED
OPEN MENU
Save Class
Delete Class
Set Class Start Date Alert
Set Class End Date Alert
Set Class Start and End Date Alert
Share Class Details
SAVE NEW CLASS
DISPLAY SAVED CLASS ON CLASS LIST
NAVIGATE TO CLASS DETAILS
NAVIGATE TO ASSIGNMENT DETAILS
Assignment Details
Assignment:
Due Date:
ENTER ASSIGNMENT INFORMATION WITH VALID DATE
Example Assignment
This is an example assignment.
Save Assignment
Delete Assignment
Set Assignment Due Date Alert
SAVE NEW ASSIGNMENT
DISPLAY SAVED ASSIGNMENT ON ASSIGNMENT LIST
SAVE CLASS WITH THE NEW ASSIGNMENT
DISPLAY SAVED CLASS ON CLASS LIST
Generate Report
NAVIGATE TO REPORT GENERATOR
SELECT A REPORT OPTION AND CLICK "Generate Report" TO GENERATE REPORT
Report Generator
Report Options:
Class
Assignment
Class and Assignment
Generate Report

UNIT TEST PLAN

Introduction

This Android mobile application will be tested with a series of unit tests. The following sections explain the purpose, overview, test plan, specifications, procedures, and results of the unit tests.

Purpose

Unit tests will be used to test the Student Planner mobile application. Unit tests are a type of software test that checks if an individual part of the code is working correctly. The purpose of these unit tests are to verify functions in the CourseDetails and AssignmentDetails work as intended. More specifically, these unit tests focus on testing aspects concerning date formation and date validation. The results from the performed unit tests conclude that all functions that were tested are working correctly. There were a total of eight unit tests conducted and all of them passed. The application is able to successfully handle date formatting and determine if a date inputted is valid or invalid. There was no remediation that was required. All of the unit tests passed and no further modification was needed. If one of the tests had failed and remediation was necessary the failed test would first be documented, revised, and then tested again. This process would repeat until that unit test passes.

Overview

There are two unit test classes that contain the unit tests performed. The unit test classes include CourseDetailsTest and AssignmentDetailsTest.

CourseDetailsTest:

- testDateFormatValidation()
- testInvalidDateFormat()
- testEndDateIsAfterStartDate()
- testEndDateIsBeforeStartDate()

The CourseDetails activity requires dates to be inputted for the class start date and the class end date. The dates should be formatted to “mm/dd/yy”. The testDateFormatValidation() unit test and the testInvalidDateFormat() unit test are used to verify that the inputted date is formatted to “mm/dd/yy”. When the testDateFormatValidation() test is run, it is expected that the inputted date is valid because it matches the desired format. When the testInvalidDateFormat() is run, it is expected to flag the inputted date as invalid because it does not match the required format. Furthermore, in the CourseDetails activity, the class end date should always be after the start date. The testEndDateIsAfterStartDate() unit test and the testEndDateIsBeforeStartDate() unit test are used to verify that the inputted dates meet this criteria. The testEndDateIsAfterStartDate() test verifies that the dates inputted are valid because the class end date is after the start date. When the testEndDateIsBeforeStartDate() is run, it is determined the inputted class start and end dates are invalid because the end date is before the start date. Together, these four tests help ensure that the functions in the CourseDetails activity involving dates behave as expected. There were no errors that occurred within these unit tests. These tests were conducted by using the JUnit 4 framework and by following the Test Plan that is outlined in the next section.

AssignmentDetailsTest:

- testDateFormatValidation()
- testInvalidDateFormat()
- assignmentDueDateIsValid()
- assignmentDueDateIsNotValid()

The AssignmentDetails activity requires a due date to be inputted for the assignment. This date should format to “mm/dd/yy”. The testDateFormatValidation() unit test and the testInvalidDateFormat() unit test are used to verify that the inputted due date is formatted correctly. When the testDateFormatValidation() test is run, it is expected that the inputted date is considered valid because it matches the “mm/dd/yy” format. When the testInvalidDateFormat() is run, it is expected to be considered invalid because the inputted date does not match the required format. Moreover, the assignment’s due date must be set between the associated class’s start and end date. This means that if the associated class starts on 01/20/25 and ends 05/28/25, the assignment due date must be set between these dates. The assignmentDueDateIsValid() unit test and assignmentDueDateIsNotValid() unit test is used to verify that the inputted due date meets this requirement. The assignmentDueDateIsValid() confirms that the due date is valid because it falls within the class’s start and end date. The assignmentDueDateIsNotValid() determines that if an assignment’s due date is set to a date outside of its associated class’s start and end date, the due date will be considered invalid. Altogether, these four unit tests make sure that the functions concerning the assignment’s due date work correctly. There were no errors that occurred within these unit tests. These tests were conducted by using the JUnit 4 framework and by following the Test Plan that is outlined in the next section.

Test Plan

Items

The following items are required to complete the tests:

- **Hardware:** Laptop
  - A laptop or another device that is capable of running Android Studio
- **Integrated Development Environment:** Android Studio
  - Android Studio is the development environment that is necessary to run the tests because the app is an Android application
- **Framework:** JUnit 4 Testing Framework
  - JUnit 4 is a testing framework for the Java programming language. In this case, it is used for the unit tests.
- **Source Code:** The application containing the Student Planner source code is necessary to complete the tests. This source code also includes the unit tests that will be ran in this test plan.

Features

The tests conducted test the following features:

1. Date Validation for Class Dates
   - **testDateFormatValidation()**: Verifies that the inputted date is formatted to the desired "MM/dd/yy" format
   - **testInvalidDateFormat()**: Verifies that if a date is not formatted to the "MM/dd/yy" format, the app will identify that this is invalid
   - **testEndDateIsAfterStartDate()**: Verifies that the class's end date is after the class's start date
   - **testEndDateIsBeforeStartDate()**: Verifies that if the class's end date is before the class's start date, the app will identify that this is invalid

2. Date Validation for Assignment Dates
   - **testDateFormatValidation()**: Verifies that the inputted date is formatted to the desired "MM/dd/yy" format
   - **testInvalidDateFormat()**: Verifies that if a date is not formatted to the "MM/dd/yy" format, the app will identify that this is invalid
   - **assignmentDueDateIsValid()**: Verifies that the assignment's due date is set between the associated class's start and end date
   - **assignmentDueDateIsNotValid()**: Verifies that is the assignment's due date is not set between the associated class's start and end date, the app will identify that this is invalid

Deliverables

- **Unit Test Scripts:** The unit tests performed are included in the source code of the application. They are located in CourseDetailsTest.java and AssignmentDetailsTest.java
- **Test Results:** Each test performed will produce a pass or fail test result. These results will be used to determine if aspects of the code function as expected or if they need to be improved.
- **Documentation:** The test procedures and results will be documented. This documentation can be used in future references, to evaluate what needs to be fixed, and keep track of which of the app's functionalities have been tested.

Tasks

1. **Test Preparation:**
    - Set up the testing environment with JUnit 4 in Android Studio
    - Add JUnit dependencies to the Gradle file
    - Add testing configurations to the Gradle file
    - Write the appropriate unit tests to test features of the app
    - Import the necessary import statements to the class
2. **Test Execution:**
    - Run each unit test
3. **Test Analyzation:**
    - Use the pass / fail criteria to determine if each unit test passes or fails
    - Modify the unit test code if necessary
    - Document the tests, test results, and test changes

Needs

The following hardware and software needs were required to perform the unit tests:

- **Hardware:**
    - A device that is able to run the latest version of Android Studio
- **Integrated Development Environment:**
    - Android Studio
- **Framework:**
    - JUnit 4
- **Import Statements:**
    - import static org.junit.Assert.*;
    - import org.junit.Test;

    - import java.text.ParseException;
    - import java.text.SimpleDateFormat;
    - import java.util.Date;
    - import java.util.Locale;
- **Gradle Configurations:**
    - testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
- **Gradle Dependencies:**
    - testImplementation libs.junit
    - testImplementation libs.core
    - testImplementation libs.ext.junit
- **Source Code:**
    - Access to the application
    - Access to the unit tests

Pass/Fail Criteria

To determine the success of the tests, a pass / fail result was used.

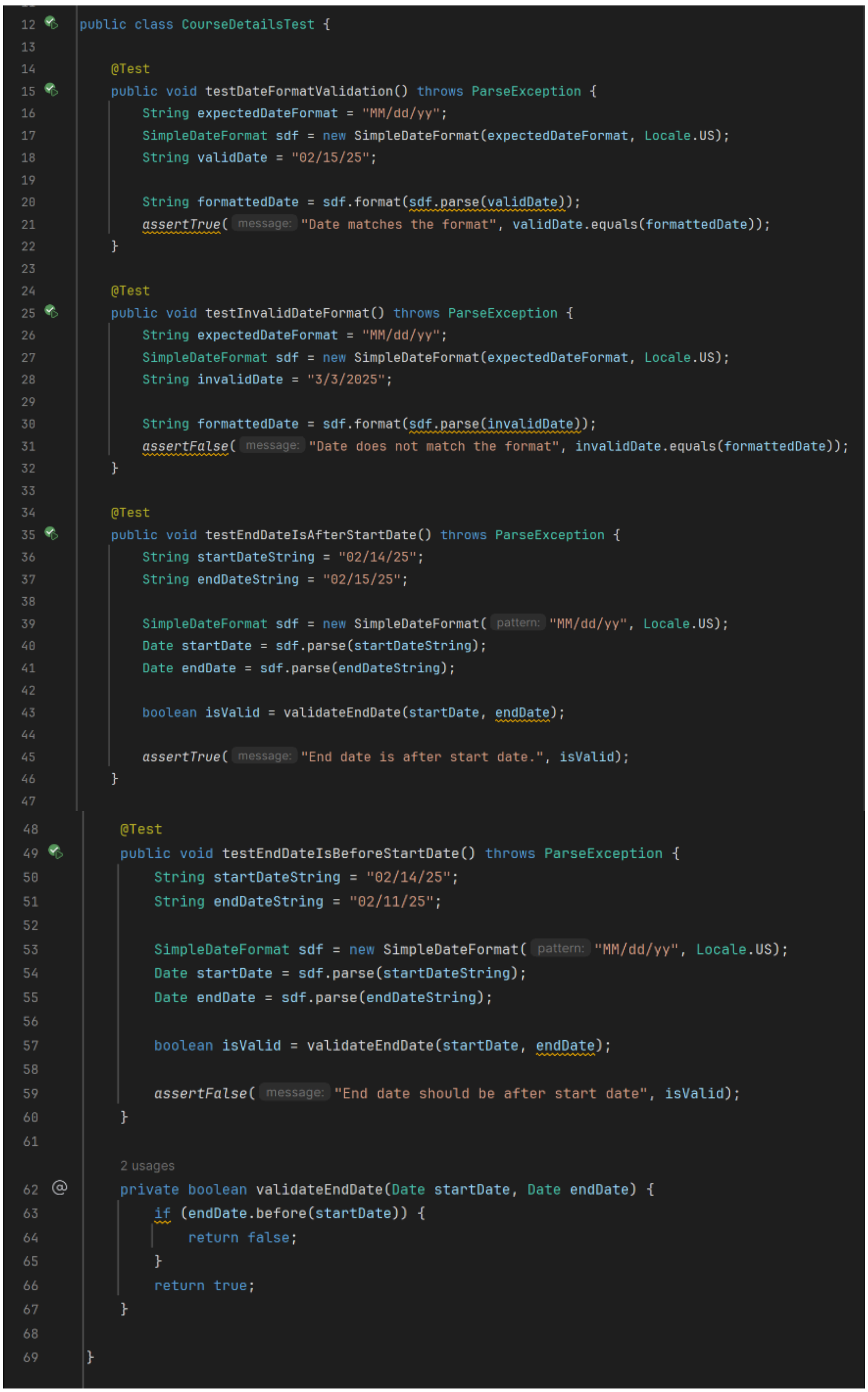
- **Pass:** The test was considered a "pass" if the test outcome matched the expected behavior. For instance, the testDateFormatValidation() verifies that the inputted date is formatted to "mm/dd/yy". When this test is run, if the imputed date matches the desired date format, the test would be marked as a "pass". The unit tests that receive a "pass" are then documented and saved.

- **Fail:** The test was considered a "fail" if the test outcome did not match the expected behavior. For example, the assignmentDueDateIsValid() unit test is used to verify that the assignment's due date is set between the associated class's start and end date. When this test was run, if it allowed the assignment's due date to be set outside of the associated class's start and end date, it would be marked as a "fail". The unit tests that receive a "fail" are documented, revised, and then tested again until they pass.

Specifications

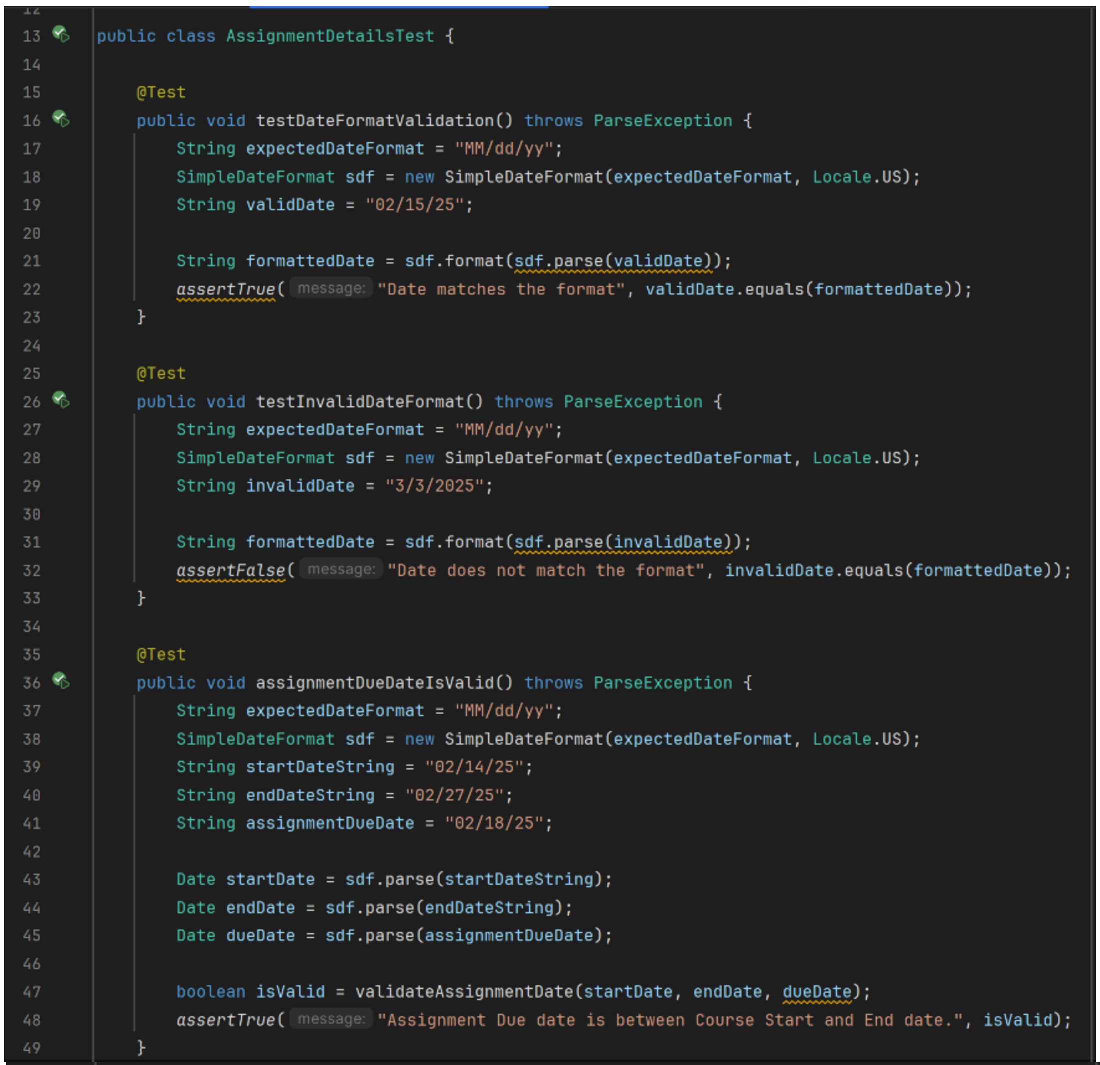
The following screenshots display the testing code that was used in the unit tests. The class details test class includes four unit tests: testDateFormatValidation(), testInvalidDateFormat(), testEndDateIsAfterStartDate(), and testEndDateIsBeforeStartDate(). The assignment details class includes four unit tests: testDateFormatValidation() ,testInvalidDateFormat(), assignmentDueDateIsValid(), and assignmentDueDateIsNotValid().

Class Details Test Code:

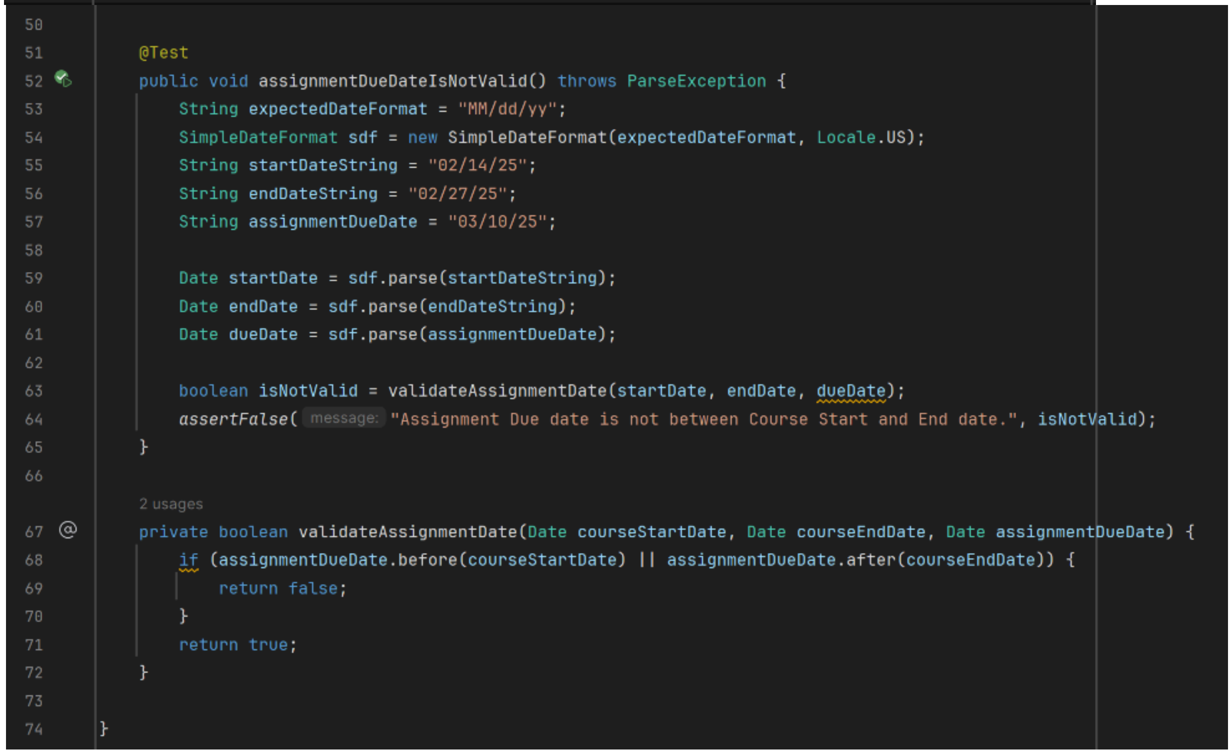


```java
public class CourseDetailsTest {

    @Test
    public void testDateFormatValidation() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String validDate = "02/15/25";

        String formattedDate = sdf.format(sdf.parse(validDate));
        assertTrue( message: "Date matches the format", validDate.equals(formattedDate));
    }

    @Test
    public void testInvalidDateFormat() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String invalidDate = "3/3/2025";

        String formattedDate = sdf.format(sdf.parse(invalidDate));
        assertFalse( message: "Date does not match the format", invalidDate.equals(formattedDate));
    }

    @Test
    public void testEndDateIsAfterStartDate() throws ParseException {
        String startDateString = "02/14/25";
        String endDateString = "02/15/25";

        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MM/dd/yy", Locale.US);
        Date startDate = sdf.parse(startDateString);
        Date endDate = sdf.parse(endDateString);

        boolean isValid = validateEndDate(startDate, endDate);

        assertTrue( message: "End date is after start date.", isValid);
    }

    @Test
    public void testEndDateIsBeforeStartDate() throws ParseException {
        String startDateString = "02/14/25";
        String endDateString = "02/11/25";

        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MM/dd/yy", Locale.US);
        Date startDate = sdf.parse(startDateString);
        Date endDate = sdf.parse(endDateString);

        boolean isValid = validateEndDate(startDate, endDate);

        assertFalse( message: "End date should be after start date", isValid);
    }

    2 usages
    private boolean validateEndDate(Date startDate, Date endDate) {
        if (endDate.before(startDate)) {
            return false;
        }
        return true;
    }

}
```

Assignment Details Test Code:



```
public class AssignmentDetailsTest {

    @Test
    public void testDateFormatValidation() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String validDate = "02/15/25";

        String formattedDate = sdf.format(sdf.parse(validDate));
        assertTrue( message: "Date matches the format", validDate.equals(formattedDate));
    }

    @Test
    public void testInvalidDateFormat() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String invalidDate = "3/3/2025";

        String formattedDate = sdf.format(sdf.parse(invalidDate));
        assertFalse( message: "Date does not match the format", invalidDate.equals(formattedDate));
    }

    @Test
    public void assignmentDueDateIsValid() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String startDateString = "02/14/25";
        String endDateString = "02/27/25";
        String assignmentDueDate = "02/18/25";

        Date startDate = sdf.parse(startDateString);
        Date endDate = sdf.parse(endDateString);
        Date dueDate = sdf.parse(assignmentDueDate);

        boolean isValid = validateAssignmentDate(startDate, endDate, dueDate);
        assertTrue( message: "Assignment Due date is between Course Start and End date.", isValid);
    }
```



```

    @Test
    public void assignmentDueDateIsNotValid() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String startDateString = "02/14/25";
        String endDateString = "02/27/25";
        String assignmentDueDate = "03/10/25";

        Date startDate = sdf.parse(startDateString);
        Date endDate = sdf.parse(endDateString);
        Date dueDate = sdf.parse(assignmentDueDate);

        boolean isNotValid = validateAssignmentDate(startDate, endDate, dueDate);
        assertFalse( message: "Assignment Due date is not between Course Start and End date.", isNotValid);
    }

    2 usages
    private boolean validateAssignmentDate(Date courseStartDate, Date courseEndDate, Date assignmentDueDate) {
        if (assignmentDueDate.before(courseStartDate) || assignmentDueDate.after(courseEndDate)) {
            return false;
        }
        return true;
    }

}
```

Procedures

1. **Test Preparation:**
    a. Install Android Studio or update Android Studio to the most recent version
    b. Open Android Studio
    c. Open the Student Planner application
    d. Add JUnit libraries to Gradle build
        i. The libraries may already be built into the project. If they are not, add the following:
            1. **Gradle Configurations:**
                a. testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
            2. **Gradle Dependencies:**
                a. testImplementation libs.junit
                b. testImplementation libs.core
                c. testImplementation libs.ext.junit
    e. Sync the project to ensure Gradle runs correctly

2. **Writing The Tests:**
    a. Determine which features should be tested
        i. In this test plan, the functions involving date format validation and date verification will be tested. The unit tests will check that inputted dates are formatted to "mm/dd/yy". The unit tests will test that class and assignment dates are valid according to specific requirements.
    b. Create the unit tests classes and determine the expected behavior of each test
        i. CourseDetailsTests Unit Tests:
            1. **testDateFormatValidation()**: Verifies that the inputted date is formatted to the desired "MM/dd/yy" format
            2. **testInvalidDateFormat()**: Verifies that if a date is not formatted to the "MM/dd/yy" format, the app will identify that this is invalid
            3. **testEndDateIsAfterStartDate()**: Verifies that the class's end date is after the class's start date
            4. **testEndDateIsBeforeStartDate()**: Verifies that if the class's end date is before the class's start date, the app will identify that this is invalid
        ii. AssignmentDetailsTest Unit Tests:
            1. **testDateFormatValidation()**: Verifies that the inputted date is formatted to the desired "MM/dd/yy" format
            2. **testInvalidDateFormat()**: Verifies that if a date is not formatted to the "MM/dd/yy" format, the app will identify that this is invalid
            3. **assignmentDueDateIsValid()**: Verifies that the assignment's due date is set between the associated class's start and end date

4. **assignmentDueDateIsNotValid()**: Verifies that is the assignment's due date is not set between the associated class's start and end date, the app will identify that this is invalid

c. Import the necessary import statements. The tests will not run without these.
   i. import static org.junit.Assert.*;
   ii. import org.junit.Test;
   iii. import java.text.ParseException;
   iv. import java.text.SimpleDateFormat;
   v. import java.util.Date;
   vi. import java.util.Locale;

3. **Executing The Tests:**
   a. Run each of the unit tests using JUnit 4.
   b. This step is where test results will be produced.
      i. If the unit test produces the expected behavior, this means that the unit test has passed. On the other hand, if the unit test produces an error or unexpected behavior, the unit test has failed.
         1. If the unit test passes, move on to step 5
         2. If the unit test fails, move on to step 4

4. **Iterations:**
   a. Iterations were not used in this unit test plan because none of the unit tests failed. They all passed on the first time they ran. However, if a test were to fail, these are the steps that should be taken to fix it:
      i. First, document the error that occurred. Documenting the failed test can help solve why the test failed.
      ii. Second, analyze the error. The test could have failed due to missing libraries, import statements, or other dependencies. It could also fail because the test has incorrect logic.
      iii. Next, once the root cause of the error is determined, modify the code to fix the issue.
      iv. Next, after the code has been updated, re-run the unit test.
      v. Finally, if the test fails again, go back to the first step of iterations. If the test passes, move forward to step 5.

5. **Documenting Test Results:**
   a. At this point, all of the tests should run with no error and be considered passing.
   b. Document the following for each of the unit tests:
      i. The unit test name
      ii. The code
      iii. Any fails that occurred and how they were fixed

iv. The result

c. Android Studio has a built in tool that allows test results to be exported in HTML format. This export will automatically be added to the project's files as well.

Results

Course Details Test Results:

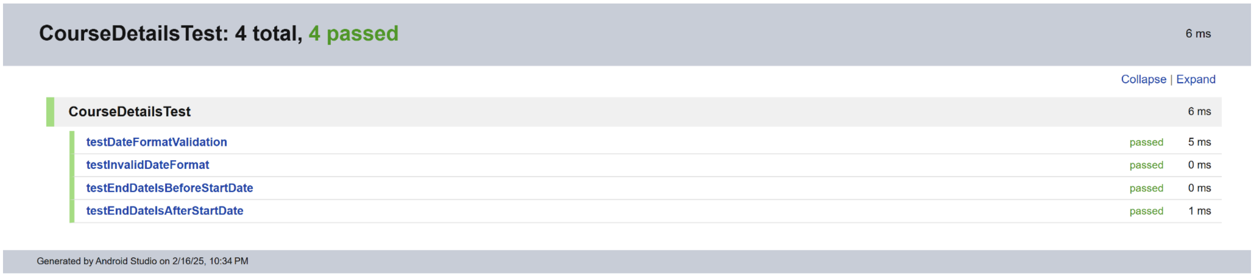
1. **PASS : testDateFormatValidation()**
   - This unit test verifies that the inputted date is formatted to the desired "MM/dd/yy" format. The input is "02/15/25" and the expected outcome is that this inputted string is recognized as matching the desired date format. The test was run and the input was successfully recognized as matching the "MM/dd/yy" format. This unit test passed.

2. **PASS : testInvalidDateFormat()**:
   - This unit test verifies that if a date is not formatted to the "MM/dd/yy" format, the app will identify that this is invalid. The input is "3/3/2025" and the expected outcome is that this inputted string is recognized as an invalid input because it does not match the desired date format. The test was run and the input was successfully recognized as invalid because the string did not match the "MM/dd/yy" format. This unit test passed.

3. **PASS : testEndDateIsAfterStartDate()**:
   - This unit test verifies that the class's end date is after the class's start date. The start date was inputted as "02/14/25" and the end date was inputted as "02/15/25". The test was run and successfully verified that the class end date is after the start date. This unit test passed.

4. **PASS : testEndDateIsBeforeStartDate()**:
   - This unit test verifies that if the class's end date is before the class's start date, the app will identify that this as an invalid input. The start date was inputted as "02/14/25" and the end date was inputted as "02/11/25". The test was run and successfully verified that the class end date is before the start date, which makes this input invalid. This unit test passed.

The results of the unit tests in the CourseDetailsTest class are provided in the screenshots below. The first screenshot is an exported HTML of the results. The second screenshot is of Android Studio. All four tests were run and all four tests passed.

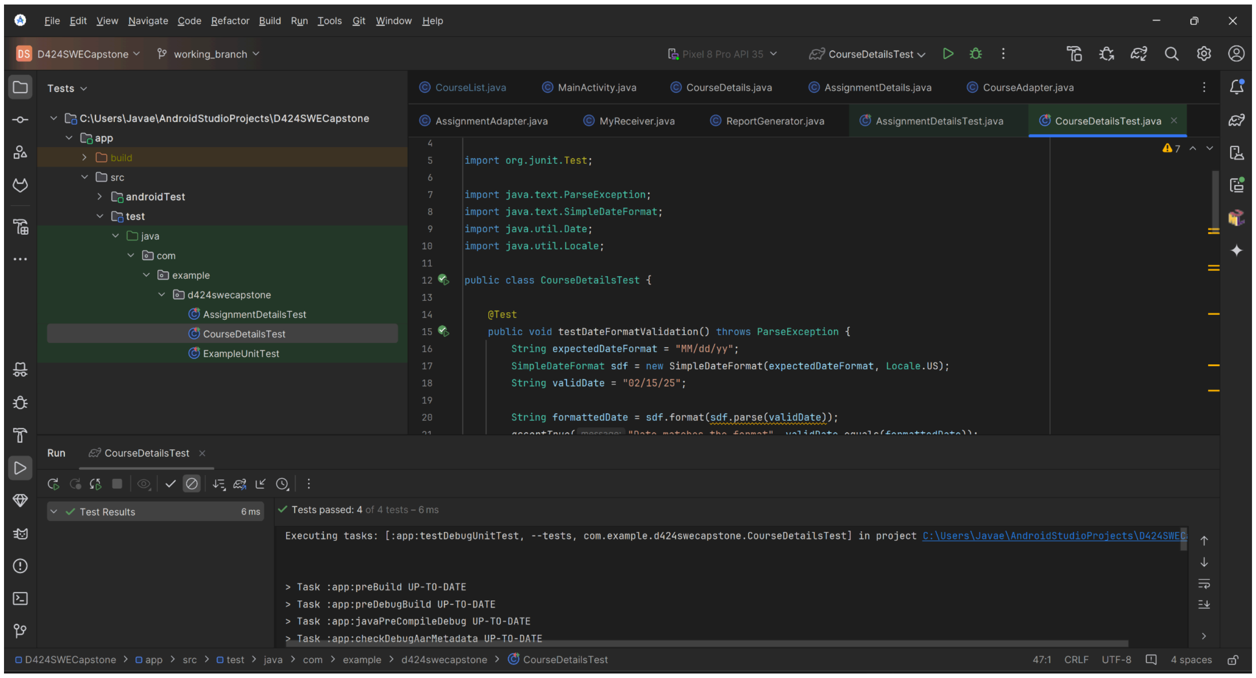


CourseDetailsTest: 4 total, 4 passed 6 ms

Collapse | Expand

CourseDetailsTest		6 ms
testDateFormatValidation	passed	5 ms
testInvalidDateFormat	passed	0 ms
testEndDateIsBeforeStartDate	passed	0 ms
testEndDateIsAfterStartDate	passed	1 ms

Generated by Android Studio on 2/16/25, 10:34 PM



```java
import org.junit.Test;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class CourseDetailsTest {

    @Test
    public void testDateFormatValidation() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String validDate = "02/15/25";

        String formattedDate = sdf.format(sdf.parse(validDate));
```

Tests passed: 4 of 4 tests – 6 ms

```
Executing tasks: [:app:testDebugUnitTest, --tests, com.example.d424swecapstone.CourseDetailsTest] in project C:\Users\Javae\AndroidStudioProjects\D424SWEC

> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
```

Assignment Details Test Results:

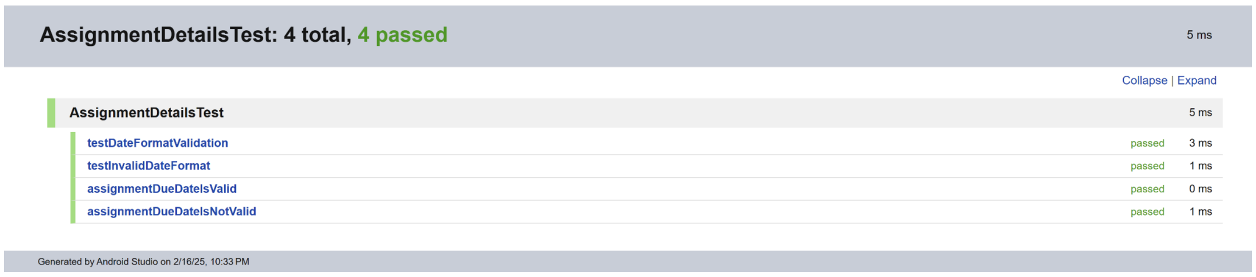
1. **PASS : testDateFormatValidation()**
   - This unit test verifies that the inputted date is formatted to the desired "MM/dd/yy" format. The input is "02/15/25" and the expected outcome is that this inputted string is recognized as matching the desired date format. The test was run and the input was successfully recognized as matching the "MM/dd/yy" format. This unit test passed.

2. **PASS : testInvalidDateFormat():**
    - This unit test verifies that if a date is not formatted to the “MM/dd/yy” format, the app will identify that this is invalid. The input is “3/3/2025” and the expected outcome is that this inputted string is recognized as an invalid input because it does not match the desired date format. The test was run and the input was successfully recognized as invalid because the string did not match the “MM/dd/yy” format. This unit test passed.

3. **PASS : assignmentDueDateIsValid()**:
    - This unit test verifies that the assignment’s due date is set between the associated class’s start and end date. The inputted assignment due date is “02/18/25”. The associated class’s start date is inputted as “02/14/25” and the end date is inputted as “02/27/25”. The test was run and the assignment’s due date was recognized as a valid input because it was in between the class’s start and end date. This unit test passed.

4. **PASS : assignmentDueDateIsNotValid()**:
    - This unit test verifies that if the assignment’s due date is not set between the associated class’s start and end date, the app will identify this as an invalid input. The inputted assignment due date is “03/10/25”. The associated class’s start date is inputted as “02/14/25” and the end date is inputted as “02/27/25”. The test was run and the assignment’s due date was recognized as an invalid input because it was set outside the class’s start and end date. This unit test passed.

The results of the unit tests in the AssignmentDetailsTest class are provided in the screenshots below. The first screenshot is an exported HTML of the results. The second screenshot is of Android Studio. All four tests were run and all four tests passed.

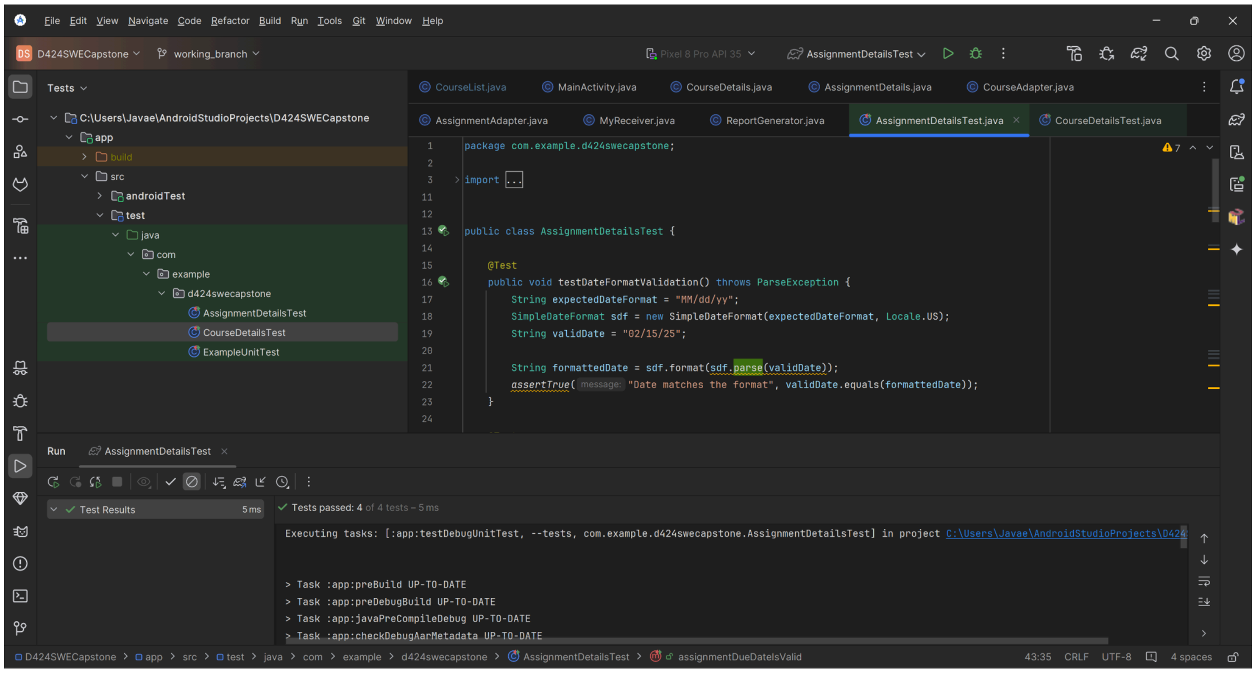


AssignmentDetailsTest: 4 total, 4 passed — 5 ms

Collapse | Expand

AssignmentDetailsTest		5 ms
testDateFormatValidation	passed	3 ms
testInvalidDateFormat	passed	1 ms
assignmentDueDateIsValid	passed	0 ms
assignmentDueDateIsNotValid	passed	1 ms

Generated by Android Studio on 2/16/25, 10:33 PM



```java
package com.example.d424swecapstone;

import ...


public class AssignmentDetailsTest {

    @Test
    public void testDateFormatValidation() throws ParseException {
        String expectedDateFormat = "MM/dd/yy";
        SimpleDateFormat sdf = new SimpleDateFormat(expectedDateFormat, Locale.US);
        String validDate = "02/15/25";

        String formattedDate = sdf.format(sdf.parse(validDate));
        assertTrue( message: "Date matches the format", validDate.equals(formattedDate));
    }

```

Tests passed: 4 of 4 tests – 5 ms

```
Executing tasks: [:app:testDebugUnitTest, --tests, com.example.d424swecapstone.AssignmentDetailsTest] in project C:\Users\Javae\AndroidStudioProjects\D424


> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
```

USER GUIDES

Installation and Maintenance

Installation Instructions:

1. **For Users:**
    a. Request the APK from the developer
    b. The APK file will be emailed to you
    c. Enable “Install from Unknown Sources”
        i. Settings → Security → Enable install from unknown sources
    d. Find the APK file on your Android device and tap it to begin the installation process

2. **For testers:**
    a. An invite link will be sent to you

        i. Current invite link:
    b. Sign up with your email
    c. Click the link
    d. Download the APK from the most current release
    e. If you are using an Android device:
        i. Enable “Install from Unknown Sources”
            1. Settings → Security → Enable install from unknown sources
        ii. Find the APK file on your Android device and tap it to begin the installation process
    f. If you are using a computer:
        i. Install an emulator or use an Android Virtual Device on Android Studio
        ii. Drag and drop the APK file into the emulator

<u>Maintenance Instructions:</u>

1. **Updates:**
    a. Check for updates
    b. If an update is available, select update to install the latest version of the app

2. **If the app is running slow or not working as expected, you can try the solutions below:**
    a. Clear App Cache
        i. Go to settings on your Android device
        ii. Select apps
        iii. Select the Student Planner app
        iv. Select storage
        v. Select clear cache
    b. Reinstall the app
        i. Go to settings on your Android device
        ii. Select apps
        iii. Select the Student Planner app
        iv. Select uninstall app
        v. Redownload the APK

3. **Further questions:**
    a. If you have any further questions or concerns, please contact the developer.

Application Operation Directions

<u>Getting Started:</u>

1. Upon launching the app, the user is taken to the main start page. Here there is a button that says, “Enter”, which will allow the user to enter the app.

2. After clicking the “Enter” button, the user is taken to the class list page. This page displays a list view of all the user’s classes. This page will be empty until the user adds a class. Once the user creates and saves a class, it will be shown on the class list page.

<u>Classes</u>

Viewing Classes:

There are two pages where users are able to view their class information. The first is the class list page and the second is the class details page.

1. Class List Page: To get to the class list page, the user will need to launch the app and then click the “Start Planning” button. This will then take the user to the class list page where they will be able to view a list of their classes. Important Notice: If no class has been added, this page will remain empty. Once a class has been added and saved, it will populate on this page.

2. Class Details Page: To get to the class details page, the user will need to navigate to the class list page and then click on the class that they wish to see additional details about. Once they have clicked on the class, they will be taken to that class’s details page. This page will display a detailed view of the class’s information including the class’s title, hotel, start date, end date, and assignments (if there are associated assignments). This view can also be used to add and update the class information.

Adding, Saving, Editing, and Deleting Classes:

- <u>Adding Classes:</u>
  - To add a new class, the user will need to click the button with the plus sign on the bottom right-hand side of the page. This will take the user to the class details page. The user will be able to enter the following information for their class: class title, hotel name, class start date, and class end date. Once this information has been entered the user will be able to save the class and it will appear on the class list page. Users are able to add as many classes as they desire.

- <u>Saving A New Class:</u>
  - Before saving a new class, the user should follow the instructions above to add details to the new class. After entering the class information in the appropriate text fields, the user

should click on the three dots on the upper right-hand side of the screen to open a menu. On the menu, select the “Save Class” option. This will successfully save the new class.

- <u>Saving An Existing Class After Editing:</u>
    - To save an existing class after its original details have been edited, the user will need to navigate to the class list page and click on the class they wish to edit. This will take them to the class’s detail page where they will be able to edit the class’s information. There are three dots on the upper right-hand side of the screen. Click the three dots and a menu will appear. In this menu, click the option that says “Save Class”. This will successfully save the new edit(s) to that class’s information.

- <u>Editing Classes:</u>
    - To edit an existing class, the user will need to navigate to the class list page and click on the class they wish to edit. This will take them to the class’s detail page where they will be able to edit the class’s information. To edit the class’s details, the user should simply click on the field that they wish to change. After editing, they can then save the new edit(s) by clicking the “Save Class” option in the menu. The user is able to update as many classes as they desire.

- <u>Deleting Classes:</u>
    - To delete a class, the user will need to navigate to the class list page and click on the class they wish to delete. This will take them to the class’s detail page where there will be three dots on the upper right-hand side of the screen. Click the three dots and a menu will appear. In this menu, click the option that says “Delete Class”. This will successfully delete the class and a message, “[name of class] was deleted”, will appear at the bottom of the screen. The user will automatically be taken back to the class list page. The user is able to delete as many classes as they desire. *Important Notice: A class cannot be deleted if there is an assignment(s) associated with it. If the user still wishes to delete this class, please delete the associated assignment(s) first and then proceed with deleting the class.*

Class Date Information:

- Class start date and end date are formatted to “MM/dd/yy”. For example, this means that if the user selects Thursday, January 16th, 2025 as their class start date, it will be formatted as 01/16/25.

- The app includes validation that prevents the class end date from being set to a date before the class start date. This means that the user will not be able to save a class that has the class end date before the class start date.

Class Alerts / Notifications:

To create an alert for a class’s start / end date, the user will need to click on the class they wish to set this alert for. This will take them to the class’s detail page where there will be three dots on the upper right-hand side of the screen. Click the three dots and a menu will appear. In this menu, there are 3 different alert options.

- <u>Set Class Start Date Alert:</u>
  - The “Set Class Start Date Alert” will set an alert that will notify the user of only the class start date. Upon selecting this option, the user will receive a notification on the class’s start date that says, “Class Starting Today: [name of class]”.

- <u>Set Class End Date Alert:</u>
  - The “Set Class Start Date Alert” will set an alert that will notify the user of only the class end date. Upon selecting this option, the user will receive a notification on the class’s end date that says, “Class Ending Today: [name of class]”.

- <u>Set Class Start and End Date Alert:</u>
  - The “Set Class Start Date and End Date Alert” will set an alert that will notify the user of both the class start date and the class end date. Upon selecting this option, the user will receive a notification on the class’s start date that says, “Class Starting Today: [name of class]” and a notification on the class’s end date that says, “Class Ending Today: [name of class]”.

**Tip: If the notifications are not appearing, make sure the app’s notifications are enabled. To do this, navigate to the app settings, select “Notifications”, and switch the button to the right. This will turn on notifications for the app and allow the user to be successfully alerted.*

Sharing Class Details:

- To share a class, the user will need to click on the class they wish to share. This will take them to the details class page where there will be three dots on the upper right-hand side of the screen. Click the three dots and a menu will appear. In this menu, select “Share Class Details”. This option will populate the sharing features with all of the class details including the class title, hotel name, class start date, class end date, assignment title, and assignment date. If there are no assignments associated with the class being shared, only the class details will show up to be shared. The class details can be shared via a sharing feature such as e-mail, clipboard or SMS.

<u>Assignments</u>

Viewing Assignments:
There are two pages where users are able to view their class’s assignment information. The first is the class details page and the second is the assignment details page.

1. Class Details Page: To get to the class details page, the user will need to navigate to the class list page and then click on the class that they wish to see additional details about. Once they have clicked on the class, they will be taken to that class’s details page. The assignments will be listed

after the class's title, hotel, start date, and end date. Important Notice: If there are no assignments associated with the selected class, the assignment list will be empty. Once an assignment has been added to a class, it will appear on the list of assignments.

2. Assignment Details Page: To get to the assignment detail page, navigate to the class details page and then click on an associated assignment to view its details. After clicking on the assignment, the next page will display a detailed view of the assignment including its title and date. This view can also be used to add and update the assignment information.

Adding, Saving, Editing, and Deleting Assignments:

- Adding Assignments:
  - Before adding a new assignment, the user will first need to navigate to the class list page and then select the class that they wish to add an assignment to. This will take them to the class's details page. To add a new assignment to the class, the user will need to click the button with the plus sign on the bottom right-hand side of the page. This will take the user to the assignment details page. The user will be able to enter the following information for their assignment: assignment title and assignment date. Once this information has been entered the user will be able to save the assignment and it will appear on the class's details page under the "Assignments Associated With Class: " section. Users are able to add as many assignments as they desire. Important notice: The date of the assignment must be set between the dates of the associated assignment.

- Saving A New Assignment:
  - Before saving a new assignment, the user should follow the instructions above to add details to the new assignment. After entering the assignment information in the appropriate text fields, the user should click on the three dots on the upper right-hand side of the screen to open a menu. On the menu, select the "Save Assignment" option. This will successfully save the new assignment and the user will automatically be taken back to the class's details page where the new assignment will appear under the "Assignments Associated With Class: " section. Important Notice: Before saving a new assignment, the user needs to have a preexisting class or needs to create and save a new class. If there is no class, the assignment will not be able to save to a class.

- Saving An Existing Assignment After Editing:
  - To save an existing assignment after its original details have been edited, the user will need to navigate to the class list page and click on the class that is associated with the assignment they wish to edit. This will take them to the class's detail page where they will be able to see the list of associated assignments. The user needs to then click on the assignment they wish to edit. This takes the user to that assignment's details page where they will be able to make changes to the assignment's information. After the edits have been made, there are three dots on the upper right-hand side of the screen. Click the three

dots and a menu will appear. In this menu, click the option that says "Save Assignment". This will successfully save the new edit(s) to that assignment's information.

- Editing Assignments:
    - To edit an existing assignment, the user will need to navigate to the class list page and click on the class that is associated with the assignment they wish to edit. This will take them to the class's detail page where they will be able to see the list of associated assignments. The user needs to then click on the assignment they wish to edit. This takes the user to that assignment's details page where they will be able to make changes to the assignment's information. To edit the assignment's details, the user should simply click on the field that they wish to change. After editing, they can then save the new edit(s) by clicking the "Save Assignment" option in the menu. The user is able to update as many assignments as they desire.

- Deleting Assignments:
    - To delete an assignment, the user will need to navigate to the class list page and click on the class that is associated with the assignment they wish to delete. This will take them to the class's detail page where they will be able to see the list of associated assignments. The user needs to then click on the assignment they are wanting to delete. After reaching the assignment's details page, click the three dots and a menu will appear. In this menu, click the option that says "Delete Assignment". This will successfully delete the assignment and a message, "[name of assignment] has been deleted", will appear at the bottom of the screen. The user will automatically be taken back to the class's details page. The user is able to delete as many assignments as they desire.

Assignment Date Information:

- Assignment dates are formatted to "MM/dd/yy". For example, this means that if the user selects Thursday, January 16th, 2025 as the date of their assignment, it will be formatted as 01/16/25.

- The app includes validation that requires the assignment date to be set between its associated class's start and end date. This means that if the class's start date is 01/16/25 and the class's end date is 01/28/25, the selected assignment date must be set to a date sometime between 01/16/25 to 01/28/25.

Assignment Alert / Notification

- To create an alert for an assignment's date, the user will need to navigate to that assignment's details page. To get to this page, the user will need to navigate to the class list page and click on the class that is associated with the assignment they wish to set an alert for. This will take them to the class's detail page where they will be able to see the list of associated assignments. The user needs to then click on the assignment they are wanting to set the alert for. This will take the user to the assignment's detail page where they will see three dots on the upper right-hand side of the screen. Click the three dots and a menu will appear. In this menu, select the option that says, "Set

Assignment Due Date Alert". This will successfully set the assignment date alert and the user will receive a notification on the date of the assignment that says "Assignment: [name of assignment] is today".

**Tip: If the notifications are not appearing, make sure the app's notifications are enabled. To do this, navigate to the app settings, select "Notifications", and switch the button to the right. This will turn on notifications for the app and allow the user to be successfully alerted.*

Search Function

Using the Search Function

- The class list page has a search function that allows users to search through the list of classes that are displayed on this page. To use the search function, click on the search icon above the class list. This will open up a search bar that allows users to type in. Begin typing in the name of the class you wish to find and the search view will automatically update to display a list of classes that match the text entered into the search bar. Once you are done using the search function, simply click the "x" on the right hand side of the search bar to close the search function. Once it has been closed, all of the classes will display on the class list.

Reports

Generating Reports:

The app is able to generate three different types of reports. The first report option is "Class Report". This option will generate a report that includes the name, instructor, start date, end date, and description of each class. The second report option is "Assignment Report". This option will generate a report that includes the name, due date, and description of each assignment. The third report option is "Class and Assignment Report. This option will generate a report that includes each class name, name of the assignments associated with the class, and the due date of each associated assignment.

1. <u>Class Report:</u>
    a. To generate a class report, first navigate to the class list page. On the upper, right hand side there are three dots. Click the three dots to open the menu. Once the menu is open, select the "Generate Report" option. Once you navigate to the Report Generator page, select the first radio button "Class". Then click the button that says "Generate Report". The report will appear below in a table layout. It will include the name, instructor, start date, end date, and description of each class. The title of the report can be seen above the table. There is also a timestamp below the table that documents when the report was generated.

2. Assignment Report:
    a. To generate an assignment report, first navigate to the class list page. On the upper, right hand side there are three dots. Click the three dots to open the menu. Once the menu is open, select the “Generate Report” option. Once you navigate to the Report Generator page, select the second radio button “Assignment”. Then click the button that says “Generate Report”. The report will appear below in a table layout. It will include the name, due date, and description of each assignment. The title of the report can be seen above the table. There is also a timestamp below the table that documents when the report was generated.

3. Class and Assignment Report:
    a. To generate a class and assignment report, first navigate to the class list page. On the upper, right hand side there are three dots. Click the three dots to open the menu. Once the menu is open, select the “Generate Report” option. Once you navigate to the Report Generator page, select the third radio button “Class and Assignment”. Then click the button that says “Generate Report”. The report will appear below in a table layout. It will include each class name, name of the assignments associated with the class, and the due date of each associated assignment. The title of the report can be seen above the table. There is also a timestamp below the table that documents when the report was generated.