

# D424 – Software Engineering

## Task 3



**Capstone Proposal Project Name:** Mobile Application 2.0

**Student Name:** Spencer Vedenoff

## ***Table of Contents***

### ***1. Introduction***

- ***Purpose of the Documentation***
- ***Overview of Features and Project Scope***

### ***2. Application Design and Testing***

- ***Class/Design Diagram***
- ***Unit Test Plan***
  - ***Features Tested***
  - ***Test Plan and Items***
- ***Unit Test Results***
  - ***Pass/Fail Criteria***
  - ***Screenshots of Results***
- ***Summary of Changes Based on Testing***

### ***3. User Guide***

- ***Setting Up and Running the Application***
  - ***Installing the APK on Android Devices***
  - ***Using the Android Emulator***
- ***Application Usage from a User Perspective***
  - ***Adding Terms and Courses***
  - ***Generating Reports***
  - ***Searching for Courses***
  - ***Setting Notifications***

### ***4. Hosting and Code Repository***

- ***Link to Hosted Application***
- ***Link to GitLab Repository***
- ***Version Information***

### ***5. Troubleshooting***

- ***Common Issues During Installation***
  - ***"App Not Installed" Error***
  - ***App Crashes on Launch***
- ***Steps to Resolve Issues***

## **Introduction**

This document provides comprehensive information on the design, testing, and usage of my .Net Maui Mobile Application. The application was built with a focus on modularity, usability, and robustness, leveraging a SQLite database to manage and interact with terms, courses, assessments, instructors, and related data.

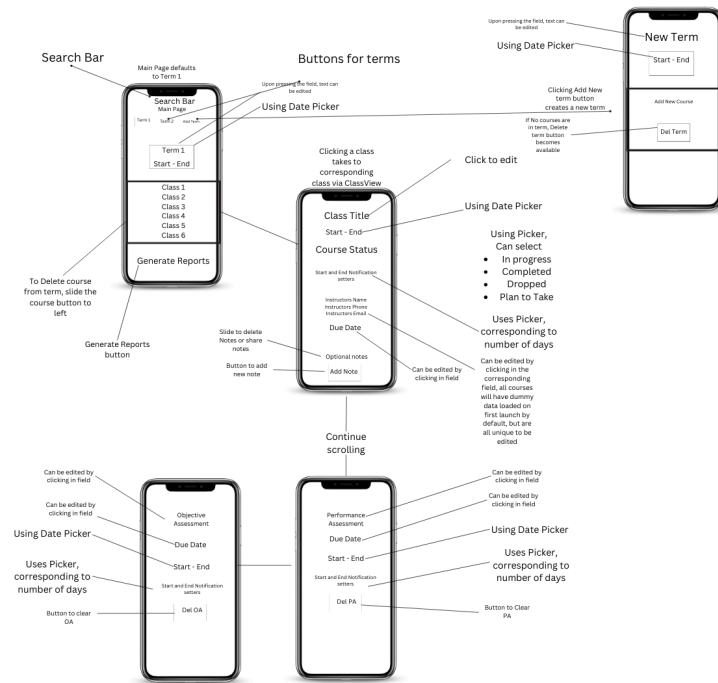
The documentation includes the following:

- A detailed overview of the application's design, including class and design diagrams.
- A user guide for setting up, running, and using the application.
- An outline of the testing process, including unit tests and their results.
- Links to the hosted application (if applicable) and the GitLab repository containing the source code.

This document aims to serve both developers (myself) and end-users (evaluators) by providing clear instructions for using and maintaining the application, as well as validating its functionality through thorough testing.

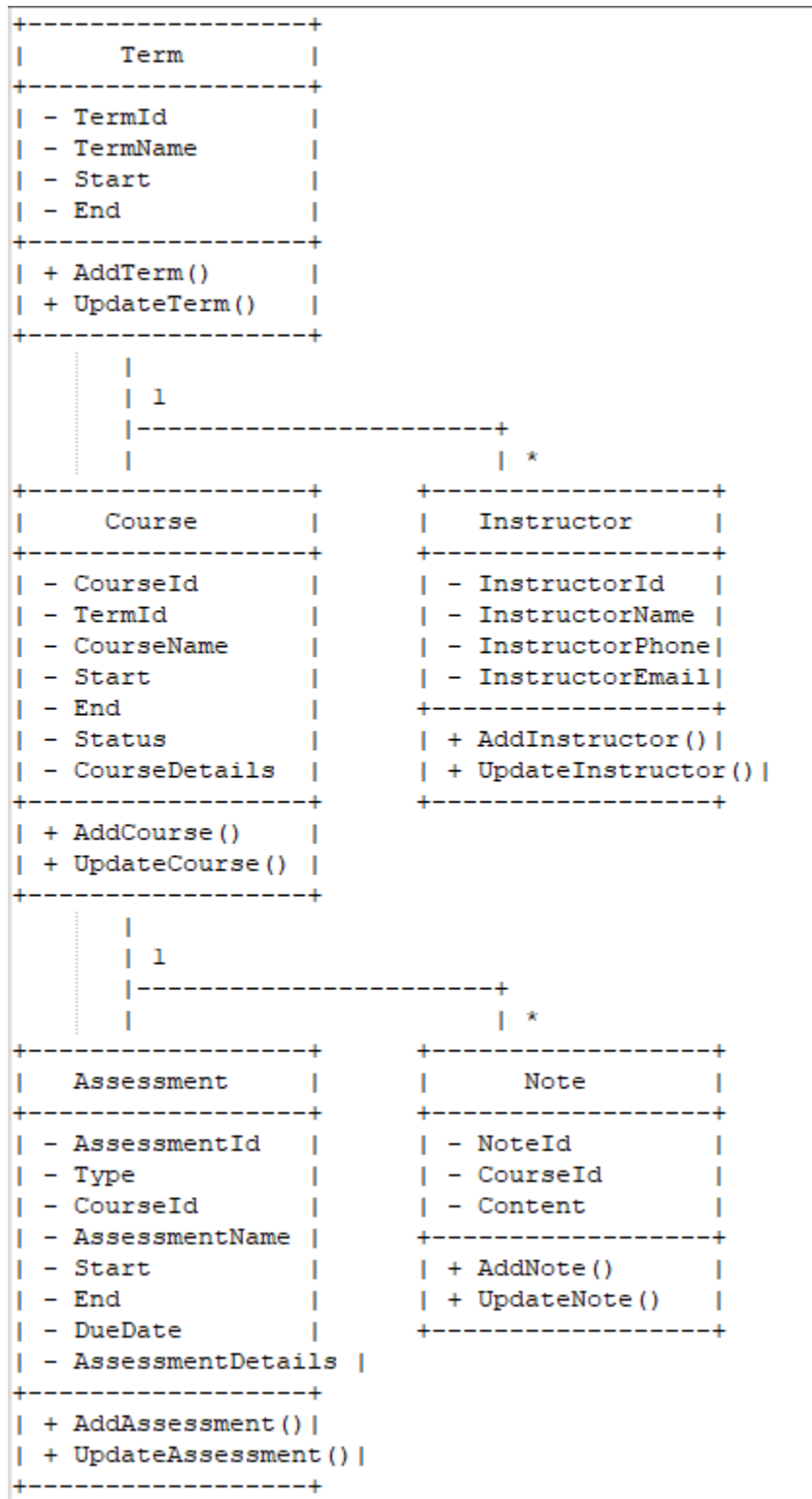
## Application Design and Testing

### Design Diagram:



This is my design diagram for my project, more or less I have utilized the wireframe that I made for C971 and updated it to reflect the needed functionality regarding generating reports and search functionality. Each and every button is labeled with a brief explanation of how it works and corresponds with the rest of the app.

## Class Diagram:



I have generated the above listed Class diagram for my application, this is more or less the same design utilized from my C971 application, however I want to expand on the explanation here to ensure understanding of my design choices and how it aligns with the rubric for D424.

Firstly, My application demonstrates **encapsulation** by grouping related data and behavior within classes. For example, the Course class encapsulates all properties related to a course, such as CourseId, CourseName, and TermId, while methods in DB.cs handle database operations like retrieving or updating course information. By centralizing functionality within these classes, the application ensures data consistency and hides implementation details, exposing only what is necessary for interaction. This design keeps the system modular and maintains data integrity.

Secondly, In my application, **Inheritance** is demonstrated through the hierarchical structure of courses and terms, where each course is conceptually a child of its parent term. The ClassView page dynamically navigates to specific content based on the selected course, utilizing the shared relationship between courses and terms. Furthermore, Performance Assessments and Objective Assessments act as specialized types under the broader Assessment class via the Type property. This design reflects real-world dependencies, where terms serve as the parent context for courses, and assessments inherit shared attributes and behaviors from their parent class.

Third, **Polymorphism** is exhibited in my application through the dynamic handling of different courses and their associated content within shared functionality. For example, the ClassView page adapts to the selected course, presenting specific details while using a unified navigation system. Additionally, database queries dynamically adjust based on user input, such as filtering courses by name in the search functionality. This allows the application to handle varied inputs and behaviors seamlessly, providing flexibility and a user-friendly experience.

## Unit Test Plan

### Introduction

#### Purpose

The purpose of these tests is to validate the existing functionality of `OnGenerateReportClicked` and `OnSearchTextChanged` using the exact code you've written. The goal is to ensure the following:

1. The **report generation** retrieves and formats course data correctly for a given term.
2. The **search functionality** dynamically updates results based on user input.

The results will verify that the methods handle edge cases gracefully without introducing new code or dependencies.

---

# Overview

## Tested Features

- **OnGenerateReportClicked:**
  - Ensures accurate report generation for courses associated with a term.
  - Confirms handling of cases with no courses for a term.
- **OnSearchTextChanged:**
  - Validates dynamic search updates for course names.
  - Confirms case-insensitivity and graceful handling of invalid input.

## Relation to the Project

- These methods are critical for user experience, ensuring accurate reporting and robust search functionality.
- The methods use parameterized queries for security, which the tests confirm as working.

## Error Handling

- Simulated edge cases like empty search text and terms with no courses.
- 

# Test Plan

## Items

- SQLite database (**MyData.db**).
- Visual Studio test suite.

## Features

1. **Generate Report:**
  - Query courses for a selected term.
  - Format the report output.
2. **Search:**
  - Filter courses by name dynamically.
  - Handle empty or invalid inputs gracefully.

## Deliverables

- Test cases for both methods.
- Documentation of pass/fail results.

## Tasks

1. Create tests for `OnGenerateReportClicked` to verify:
  - Report formatting.
  - Handling of terms with no courses.
2. Create tests for `OnSearchTextChanged` to validate:
  - Search functionality for valid inputs.
  - Graceful handling of invalid inputs.

## Needs

- Working SQLite database file with sample data.

## Pass/Fail Criteria

- **Pass:**
  - The method executes without exceptions.
  - Outputs match expected results for inputs.
- **Fail:**
  - Errors occur during execution.
  - Results don't match expectations.



```

0 references
private async void OnGenerateReportClicked(object sender, EventArgs e)
{
    try
    {
        using var db = new SQLiteConnection(MainPage.databasePath);

        // Ensure a term is selected
        if (termSelected == null)
        {
            await DisplayAlert("Error", "Please select a term to generate the report.", "OK");
            return;
        }

        // Query to get courses for the selected term
        var query = "SELECT * FROM Courses WHERE TermId = ?";
        var result = db.Query<Course>(query, termSelected.TermId);

        if (result.Count == 0)
        {
            await DisplayAlert("Report", $"No courses found for the term '{termSelected.TermName}'.", "OK");
            return;
        }

        // Format the report
        var report = new StringBuilder($"Course Report for Term: {termSelected.TermName}\n\n");
        foreach (var course in result)
        {
            report.AppendLine($"Title: {course.CourseName}");
            report.AppendLine($"Start: {course.Start}");
            report.AppendLine($"End: {course.End}");
            report.AppendLine($"Status: {course.Status}\n");
        }

        // Display the report
        await DisplayAlert("Report", report.ToString(), "OK");
    }
    catch (Exception ex)
    {
        await DisplayAlert("Error", $"Failed to generate report: {ex.Message}", "OK");
    }
}

```

Code used for Generating Reports ^

```

// This method contains a parameterized query to ensure secure search functionality to DB
0 references
private void OnSearchTextChanged(object sender, TextChangedEventArgs e)
{
    var searchText = e.NewTextValue?.ToLower() ?? string.Empty;

    using var db = new SQLiteConnection(MainPage.databasePath);

    // Parameterized query to prevent SQL injection
    var query = "SELECT * FROM Courses WHERE LOWER(CourseName) LIKE ?";

    try
    {
        // Pass the parameter to the query
        var courses = db.Query<Course>(query, $"%{searchText}%");

        // Update the CollectionView
        if (courses.Count > 0)
        {
            SearchResults.IsVisible = true;
            SearchResults.ItemsSource = courses;
        }
        else
        {
            SearchResults.IsVisible = false;
        }
    }
    catch (SQLiteException ex)
    {
        Console.WriteLine($"Error querying database: {ex.Message}");
        SearchResults.IsVisible = false;
    }
}

```

Code used for Search Functionality ^

## Results

### Expected Results

1. **OnGenerateReportClicked:**
  - For populated terms: Outputs a detailed course report.
  - For empty terms: Outputs "No courses found for the selected term."

Result Passed:

## Report

Course Report for Term: Term 1

Title: Python Programming  
Start: 11/21/2024 3:48:44 PM  
End: 3/21/2025 12:00:00 AM  
Status: In Progress

Title: Software 1  
Start: 11/21/2024 3:48:44 PM  
End: 3/21/2025 3:48:44 PM  
Status: In Progress

Title: Software 2  
Start: 11/21/2024 3:48:44 PM  
End: 3/21/2025 3:48:44 PM  
Status: In Progress

Title: Mobile App Development  
Start: 11/21/2024 3:48:44 PM  
End: 3/21/2025 3:48:44 PM  
Status: In Progress

Title: Data Foundations  
Start: 11/21/2024 3:48:44 PM  
End: 3/21/2025 3:48:44 PM  
Status: In Progress

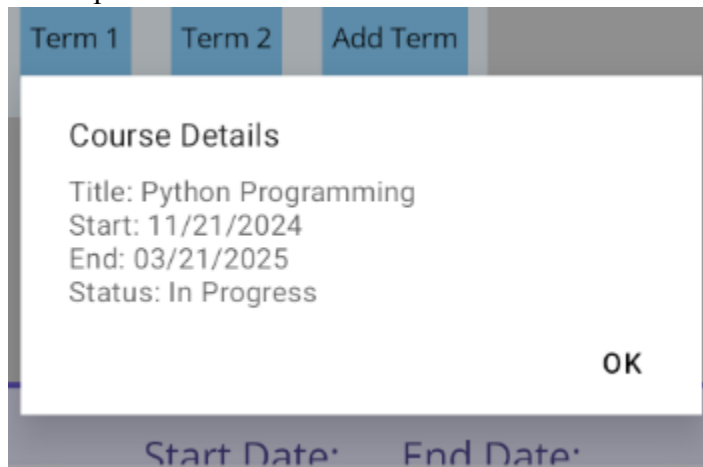
Title: Data Applications  
Start: 11/21/2024 3:48:44 PM  
End: 3/21/2025 3:48:44 PM  
Status: In Progress

OK

## 2. **OnSearchTextChanged:**

- Matches courses based on input.
- Displays "No courses match the search term" if no results.

Result passed:



**Link to hosted application:**

<https://github.com/SpencerVedenoff/D424/releases/tag/v1.0.0>

**Link to GitLab Repo:**

[https://gitlab.com/wgu-gitlab-environment/student-repos/SpencerVedenoff/d424-software-engineering-capstone/-/tree/working/D424?ref\\_type=heads](https://gitlab.com/wgu-gitlab-environment/student-repos/SpencerVedenoff/d424-software-engineering-capstone/-/tree/working/D424?ref_type=heads)

---

## User Setup Guide: Installing the APK

This guide will help you install and run the application on an Android device or emulator.

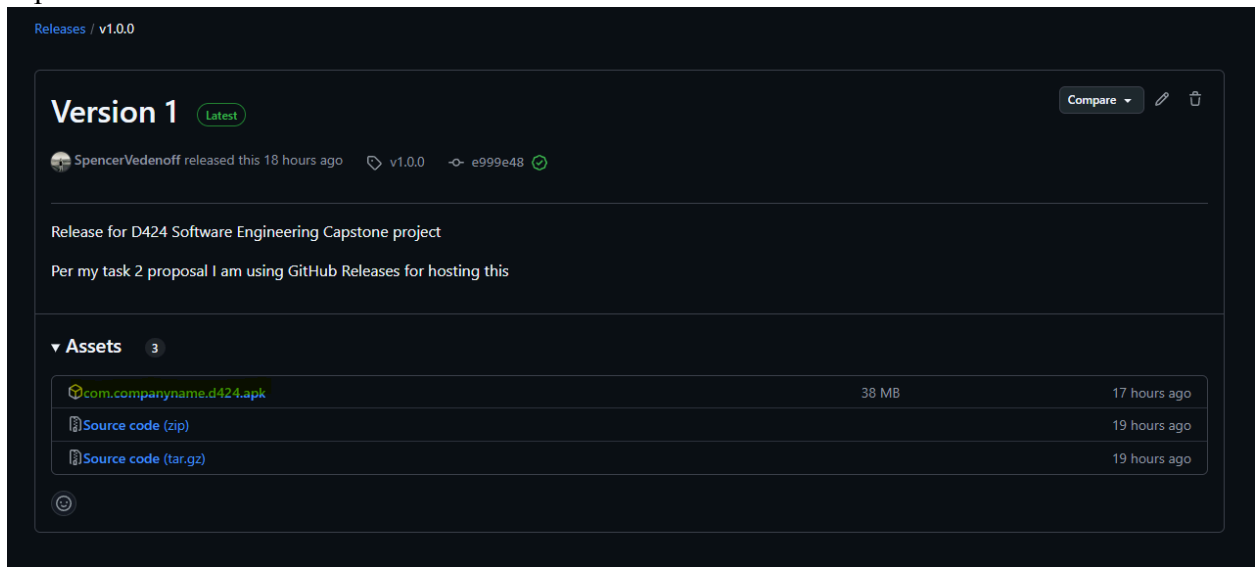
---

### For Android Devices

#### Step 1: Download the APK

1. Open the GitHub Release page for the application on your Android device's browser.  
<https://github.com/SpencerVedenoff/D424/releases/tag/v1.0.0>
2. Locate the APK file under the **Assets** section (e.g., `com.companyname.d424.apk`).

3. Tap on the APK file to download it.



## Step 2: Enable Installation from Unknown Sources

1. Go to your Android device **Settings**.
2. Navigate to:
  - **Apps > Special Access > Install Unknown Apps** (on newer Android versions)
  - OR
  - **Security > Unknown Sources** (on older Android versions).
3. Enable the permission for the browser or file manager you used to download the APK.

## Step 3: Install the APK

1. Once the APK is downloaded, open the **Downloads** folder or your notification bar.
2. Tap on the APK file.
3. Review the permissions required and tap **Install**.
4. Wait for the installation to complete.

## Step 4: Open the App

1. After installation, tap **Open** to launch the app immediately.
2. Alternatively, find the app icon in your app drawer and tap to open it.

---

## For Android Emulators

### Step 1: Download the APK

1. Open the GitHub Release page in your computer's browser.

2. Download the APK file to a location on your computer.

### **Step 2: Set Up Your Emulator**

1. Open your Android emulator (I personally used Android Studio).
2. Ensure the emulator is running and functional.

### **Step 3: Install the APK**

1. Drag and drop the APK file onto the emulator window.  
OR
2. Use the following steps:
  - Open the emulator's **File Manager**.
  - Navigate to the folder where the APK was downloaded.
  - Tap on the APK file to install it.

### **Step 4: Launch the App**

1. After installation, find the app icon in the emulator's app drawer.
2. Tap to open and run the application.


### **Step 5: Main Page Functionality**

1. Now on first load the application will load with dummy data for testing purposes. However this will save locally to ensure any edits are kept after this initial boot.
2. From here you can add terms, navigate terms, go to course pages for their corresponding terms, search for all courses in the search bar to filter a specific report for a course, or at the bottom of the page generate a report:

4:02



## Software Engineering Capstone

 Search for a course...

Term 1

Term 2

Add Term

*Term 1*

Start Date:

11/21/2024

End Date:

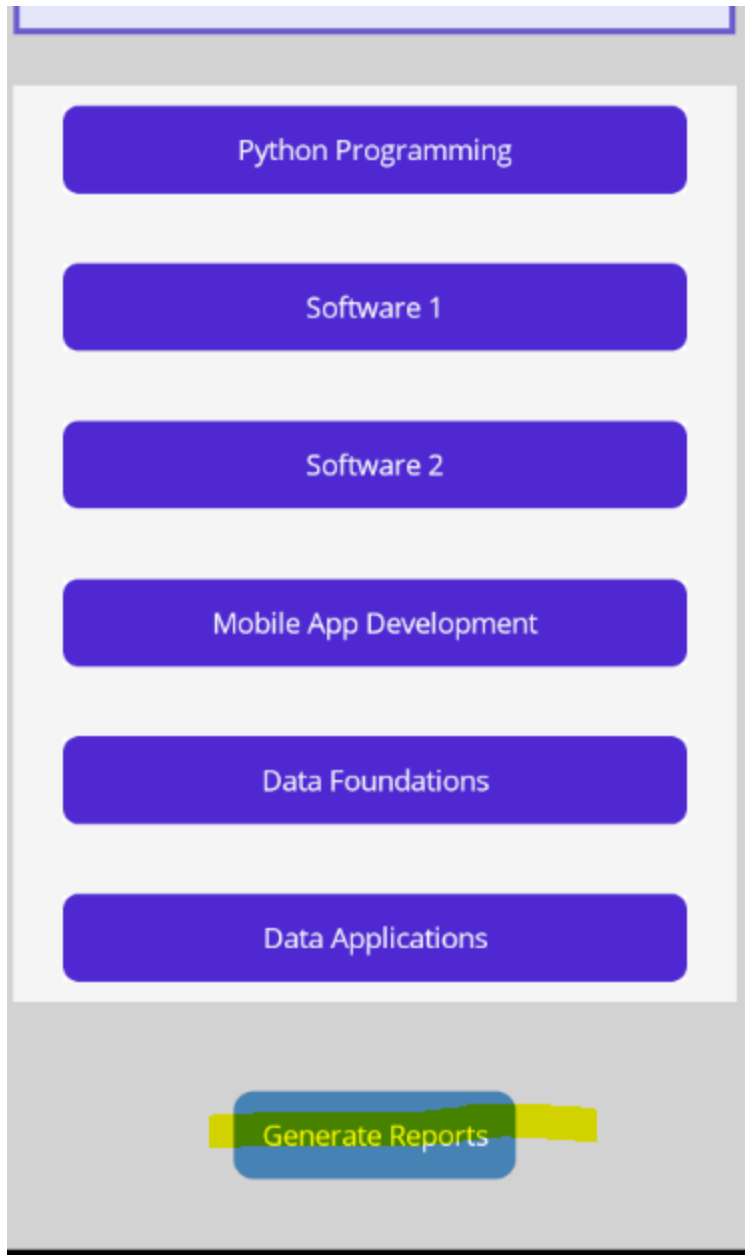
5/21/2025

Python Programming

Software 1

Software 2

Mobile App Development



### Step 6: Adding a new Term

1. Clicking “Add Term” will create a new step iterated 1 number more than the last term. this will also create an empty course with the button “Add Course” under this.
2. All Details in this course can be edited as needed, name, start date, end date, status, instructor info, notifications, assessments and notes.



3. To delete a course simply slide the newly added course to the left and press “Delete”:

## Software Engineering Capstone

The screenshot displays the 'Software Engineering Capstone' app interface. At the top is a search bar with a magnifying glass icon and the text 'Search for a course...'. Below this is a row of four blue buttons: 'Term 1', 'Term 2', 'Term 3', and 'Add Term'. The 'Term 3' button is highlighted with a blue underline. Below the buttons, the text 'Term 3' is displayed in a large, blue, italicized font. Underneath this, there is a light blue box containing two date fields: 'Start Date:' with the value '11/21/2024' and 'End Date:' with the value '1/20/2025'. At the bottom, there is a row of two buttons: a blue 'NewCourse' button and a red 'Delete' button. Below these is a large blue button labeled 'Add Course'.

4. This will default the term back to being empty (without courses) and the “Delete Term” button will reappear.

Note: If you desire to delete a newly added term, you will need to delete all courses added from that newly added term first before you can delete the term as courses are tied to that term as to not lose track of any data.

**Step 7 Setting notifications:**

1. To set notification either for course start or end dates, simply navigate to the course in question and set the start/end date using the datepicker
2. Then set either the corresponding notify before selector to the number of days prior to that date you want the notification to occur.
3. Once that date passes you should receive both a Display alert notifying you of the set notification and a system notification:

4:13



## ← Course Overview

# Python Programming

Start Date: 11/23/2024

End Date: 3/21/2025

Status:

Python Programming Starting Soon

Python Programming is starting soon!

OK

Phone:

555-123-4567

Email:

anika.patel@strimeuniversity.edu

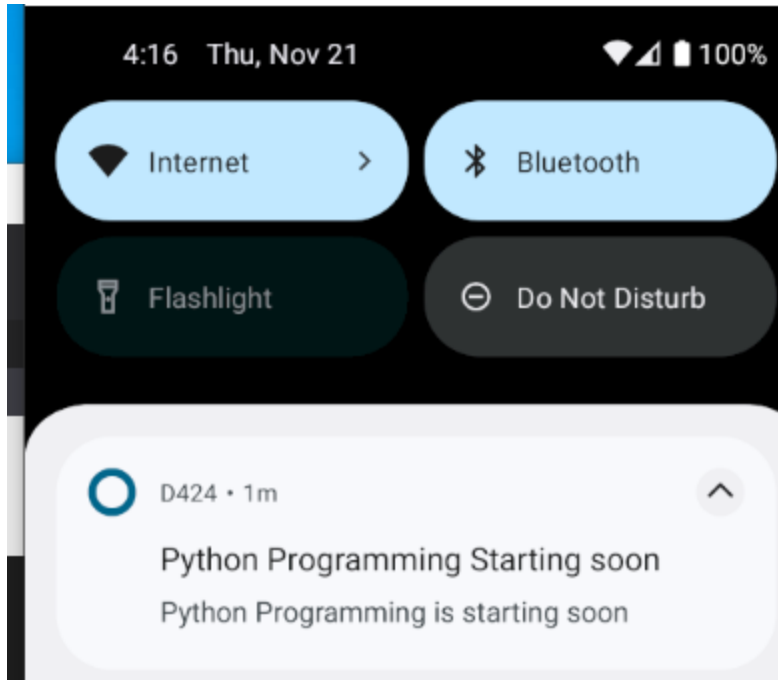
## Notifications

Notify Before Start:

2

Notify Before End:

0



### Step 8 Search and Generate Reports:

1. As noted above in the unit tests for these, the search functionality is as simple as clicking the search box and then typing in the class you want to find. On a blank search bar it will return all courses, but as you type it will narrow down based upon the criteria:



2. Generate reports is as simple as clicking on the generate reports button

---

## Troubleshooting

(Found these little snippets of info on the web, thought they would be good to include them, source: <https://mobileinternist.com/cant-install-apk-file-android>)

### **Problem: "App Not Installed" Error**

- Ensure you have enabled "Install Unknown Apps" in your Android settings.
- Check if the APK is compatible with your Android version.

### **Problem: The App Crashes on Launch**

- Verify that your device or emulator meets the system requirements (e.g., Android version compatibility).
- Ensure the APK file isn't corrupted (redownload if necessary).