C868 – Software Capstone Project Summary

Task 2 – Section C

Capstone Proposal Project Name: Semester Scheduling Mobile Application

Student Name: Derrick A. Koehn

Table of Contents

Application Design and Testing

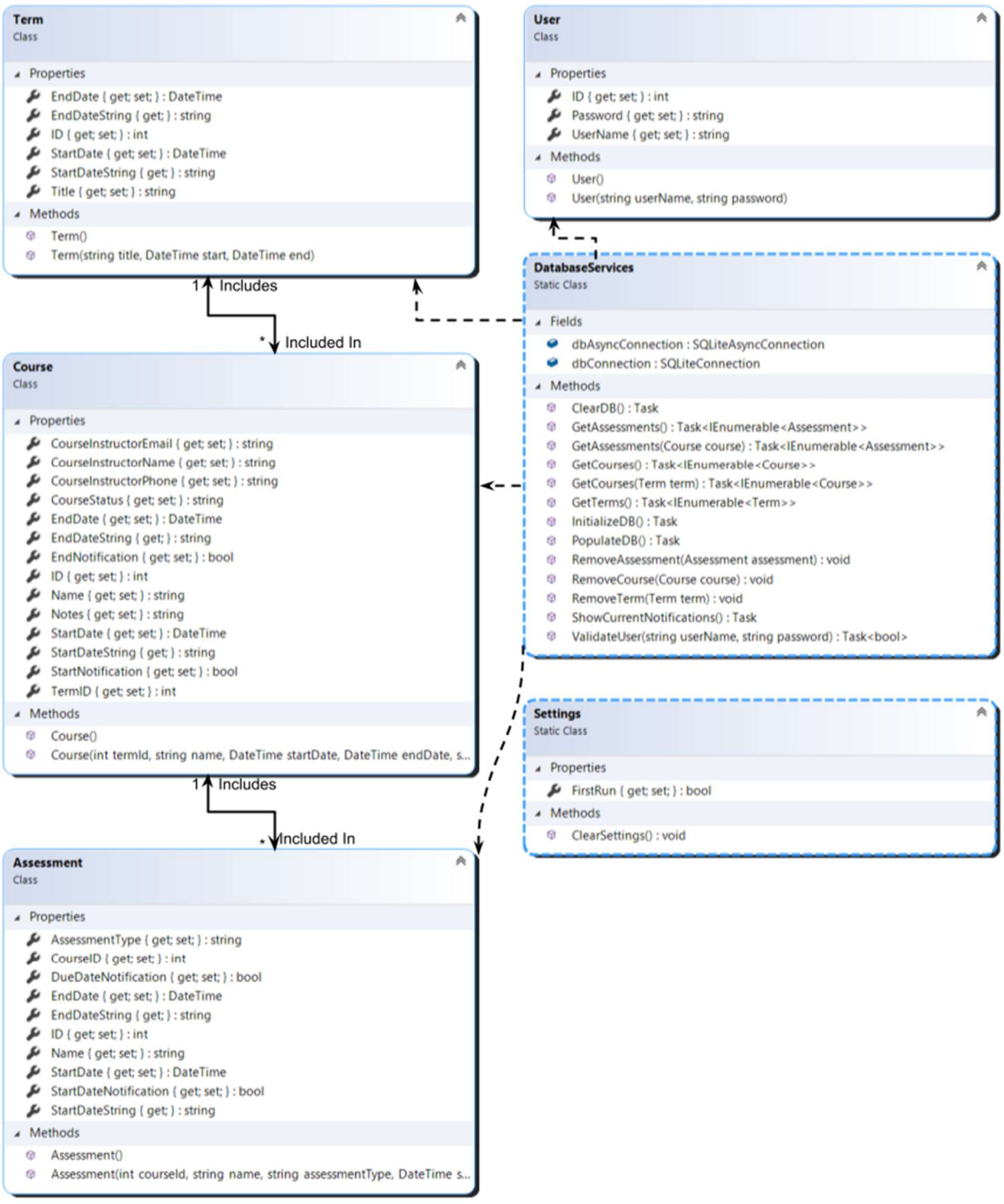
Design Document

Class Design

On the following page is a diagram showing the custom classes that I created for this project. Depicted are four non-static classes: Term, Course, Assessment, and User. Instances of these non-static classes are used throughout the application and are stored within the application's internal SQLite database. Also depicted are two static classes: DatabaseServices and Settings. As static classes, their fields and functions are referenced throughout the application to support important features, but there are no object instances. These static classes, outlined with a blue dashed line in the diagram below, are not a part of the database schema.

Each of the four non-static classes, Term, Course, Assessment, and User, corresponds to a table in the application's SQLite database. The graphic below shows the database relationships between the tables. Terms have a one-to-many relationship with Courses. Course objects' *TermID* field is a foreign key corresponding to the associated Term object's *ID* field. Courses have a one-to-many relationship with Assessments. Assessment objects' *CourseID* field corresponds to the associated Course object's *ID* field. A User object is also stored in the relational database, but it is only used during the authentication process and is not linked to any other tables via a foreign key.

The two static classes, DatabaseServices and Settings, contain fields and methods that support application features. The DatabaseServices class initializes the application's database, and its methods handle tasks related to database queries. The graphic below includes dashed arrows from the DatabaseServices class to the classes on which it is dependent. The Settings class enables a feature that detects whether the application is being run for the first time.


Term
Class
Properties
EndDate { get; set; } : DateTime
EndDateString { get; } : string
ID { get; set; } : int
StartDate { get; set; } : DateTime
StartDateString { get; } : string
Title { get; set; } : string
Methods
Term()
Term(string title, DateTime start, DateTime end)
User
Class
Properties
ID { get; set; } : int
Password { get; set; } : string
UserName { get; set; } : string
Methods
User()
User(string userName, string password)
1 Includes
* Included In
DatabaseServices
Static Class
Fields
dbAsyncConnection : SQLiteAsyncConnection
dbConnection : SQLiteConnection
Methods
ClearDB() : Task
GetAssessments() : Task<IEnumerable<Assessment>>
GetAssessments(Course course) : Task<IEnumerable<Assessment>>
GetCourses() : Task<IEnumerable<Course>>
GetCourses(Term term) : Task<IEnumerable<Course>>
GetTerms() : Task<IEnumerable<Term>>
InitializeDB() : Task
PopulateDB() : Task
RemoveAssessment(Assessment assessment) : void
RemoveCourse(Course course) : void
RemoveTerm(Term term) : void
ShowCurrentNotifications() : Task
ValidateUser(string userName, string password) : Task<bool>
Course
Class
Properties
CourseInstructorEmail { get; set; } : string
CourseInstructorName { get; set; } : string
CourseInstructorPhone { get; set; } : string
CourseStatus { get; set; } : string
EndDate { get; set; } : DateTime
EndDateString { get; } : string
EndNotification { get; set; } : bool
ID { get; set; } : int
Name { get; set; } : string
Notes { get; set; } : string
StartDate { get; set; } : DateTime
StartDateString { get; } : string
StartNotification { get; set; } : bool
TermID { get; set; } : int
Methods
Course()
Course(int termId, string name, DateTime startDate, DateTime endDate, s...
Settings
Static Class
Properties
FirstRun { get; set; } : bool
Methods
ClearSettings() : void
1 Includes
* Included In
Assessment
Class
Properties
AssessmentType { get; set; } : string
CourseID { get; set; } : int
DueDateNotification { get; set; } : bool
EndDate { get; set; } : DateTime
EndDateString { get; } : string
ID { get; set; } : int
Name { get; set; } : string
StartDate { get; set; } : DateTime
StartDateNotification { get; set; } : bool
StartDateString { get; } : string
Methods
Assessment()
Assessment(int courseId, string name, string assessmentType, DateTime s...

UI Design

The user interface for the application includes eight screens. When a student launches the application, they will be greeted by a login screen. The first time the application is run, this screen will prompt the user to create a username and a password. After the initial setup, and for all subsequent launches of the application this screen will require the user to enter their username and password before it allows authenticated users to access other application features or stored data.

The first screen that will appear after user authentication is a “Term Manager” screen on which saved terms will be displayed in a vertical list once terms have been added by the student. Near the top of the screen, there will be a button that, when tapped, will display an “Add Term” screen that will allow the application’s user to input and save information for a new school term. Once information, including a term title and the term’s start and end date, has been entered into the displayed form, the user will, by tapping the “Save” button, save a new term in the application’s internal database. Such saved terms will then be displayed in the list on the “Term Manager” screen.

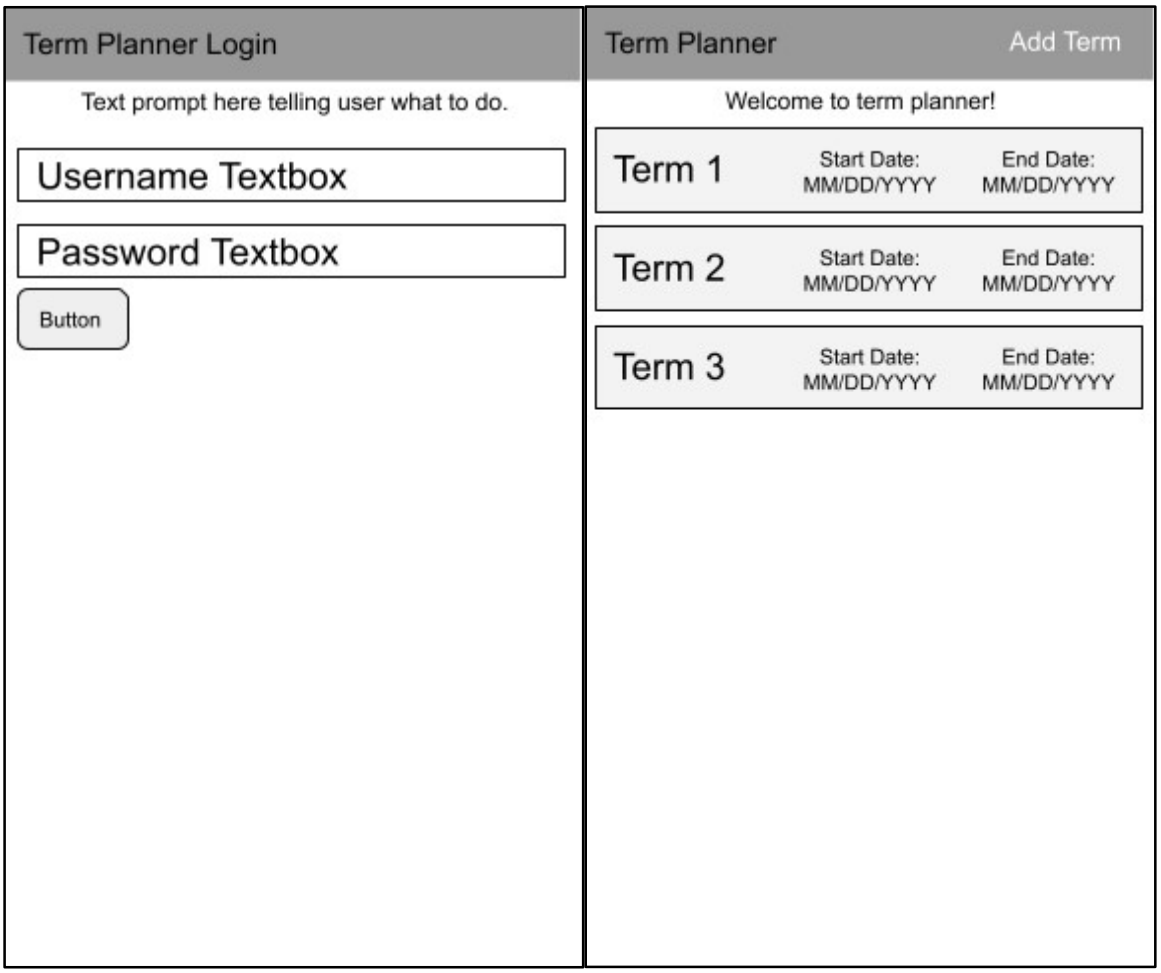
Tapping a term displayed on the “Term Manager” screen will open a “Term Details” screen which will include a form enabling the user to update the term’s title, start date, and end date, as well as a button to delete the term entirely. This screen will also include a list of courses for the term, once courses have been added by the student. Near the top of the screen, there will be a button that, when tapped, will display an “Add Course” screen which will allow the application’s user to input and save information for a new course associated with the term. Once information, including a course’s name, the course’s start date and end date, and whether the user wishes to receive push notifications when the course begins or ends, has been entered into the

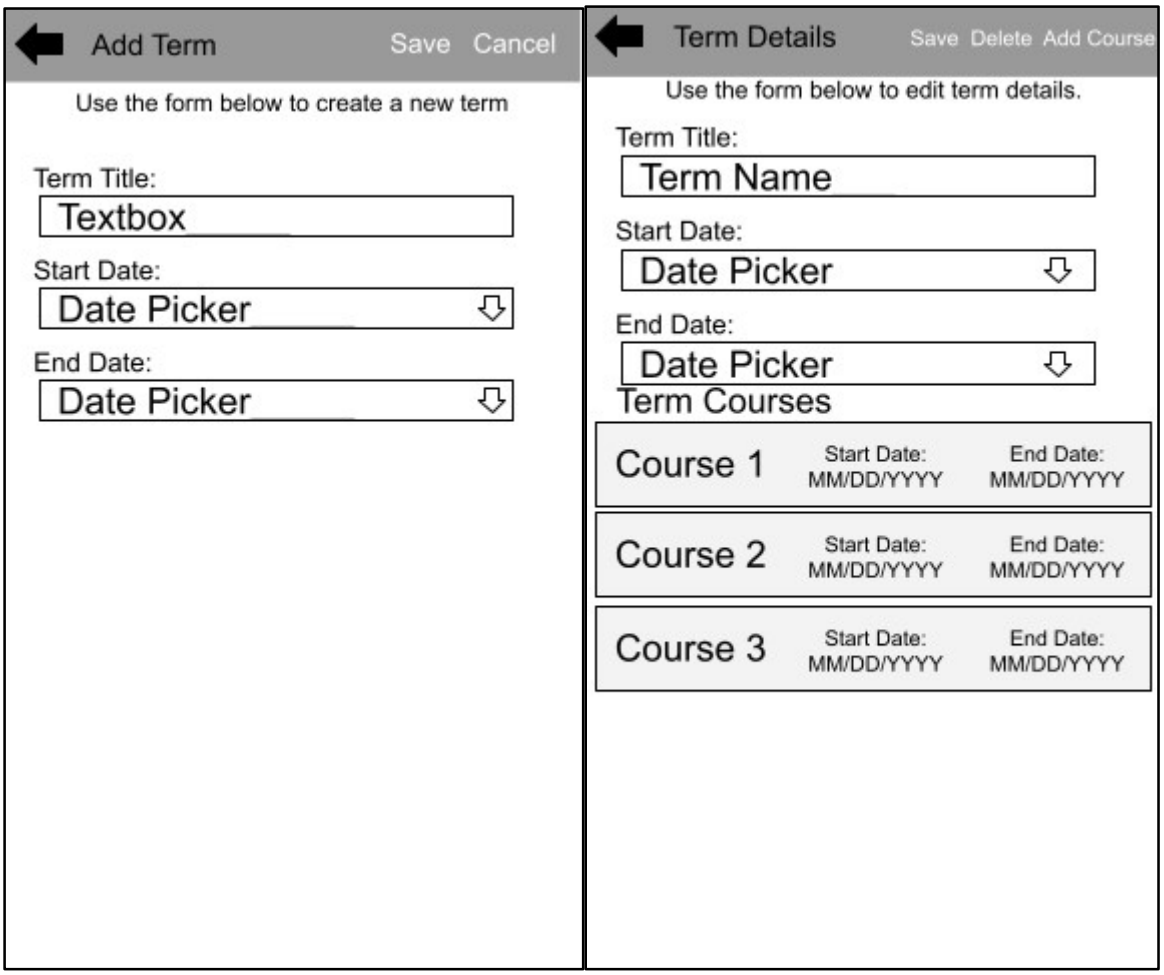
displayed form, the user will, by tapping the “Save” button, save a new course in the application’s internal database. Such saved terms will then be displayed in the list on the respective term’s “Term Details” screen. The “Term Details” screen will also allow users to quickly find specific courses in the list by entering search terms into a displayed search tool.

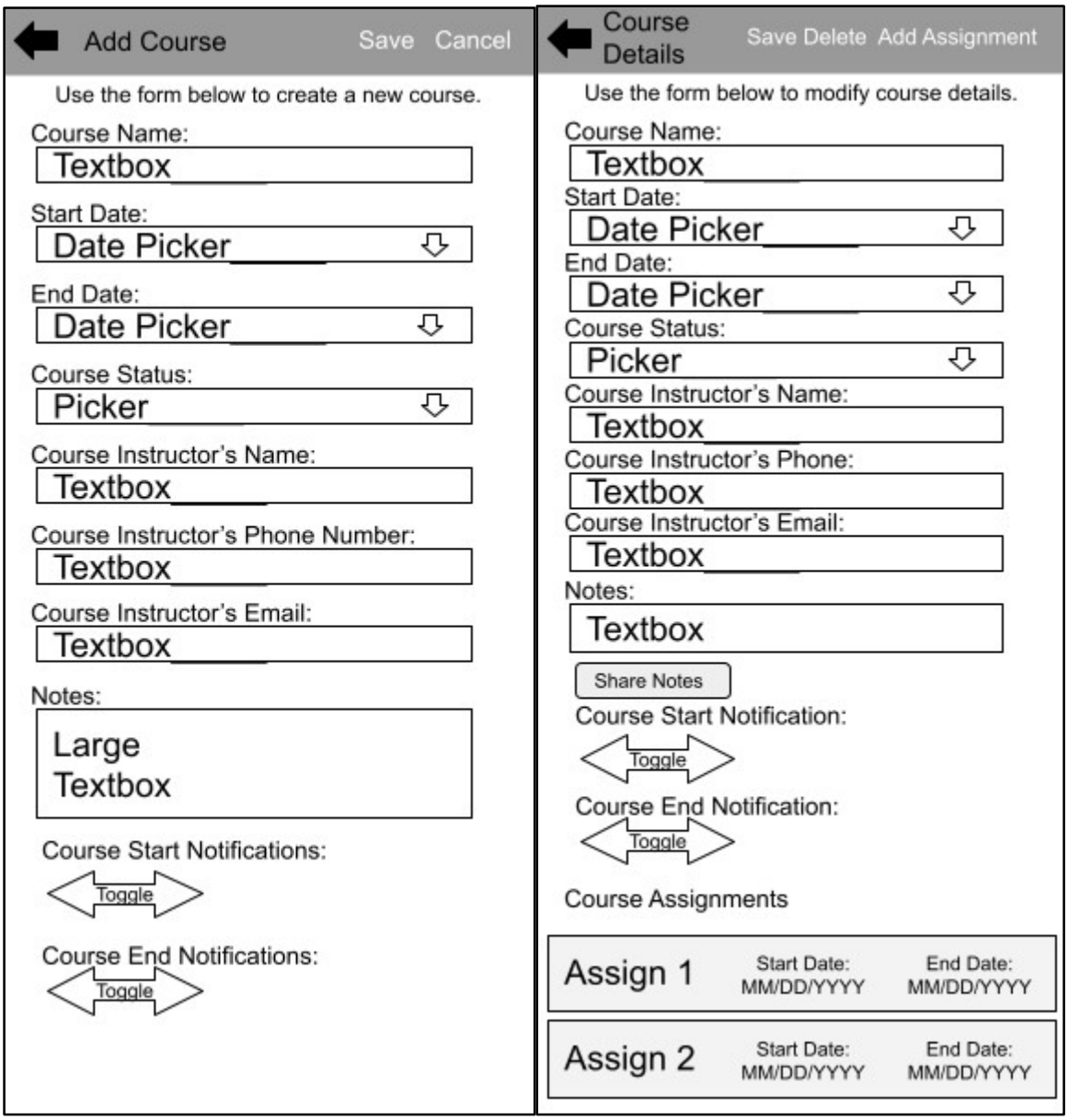
Tapping a course displayed on the “Term Details” screen will open a “Course Details” screen which will include a form enabling the user to update the course’s name, start date, end date, and notification settings as well as a button to delete the course entirely. This screen will also include a list of assignments for the course, once assignments have been added by the student. Near the top of the screen, there will be a button that, when tapped, will display an “Add Assignment” screen which will allow the application’s user to input and save information for a new assignment associated with the term. Once information, including the assignment’s name, start date, and due date, and whether the user wishes to receive push notifications when the assignment begins or ends, has been entered into the displayed form, the user will, by tapping the “Save” button, save a new assignment in the application’s internal database. Such saved assignments will then be displayed in the list on the respective course’s “Course Details” screen.

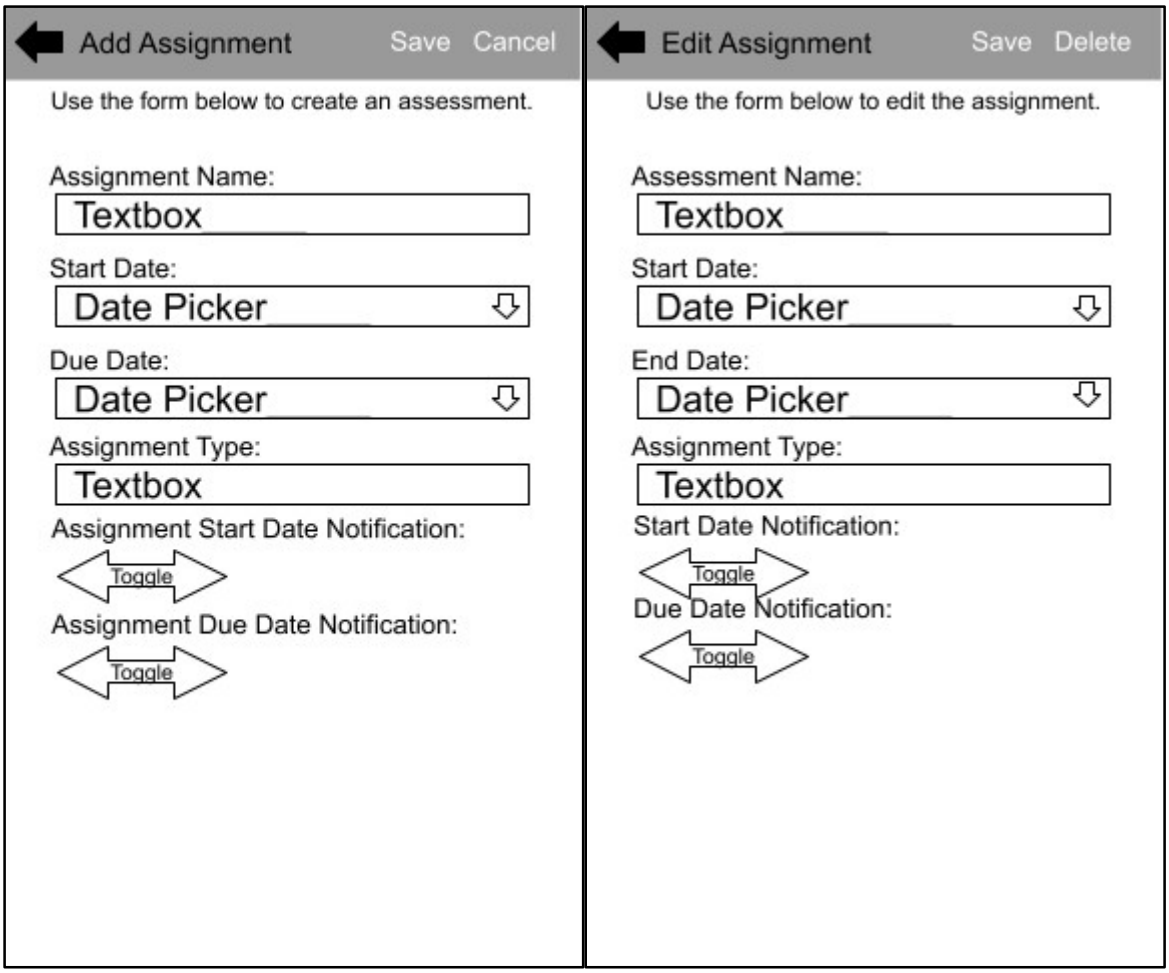
Tapping an assignment displayed on the “Course Details” screen will open an “Edit Assignment” screen which will include a form enabling the user to update the assignment’s name, start date, due date, and notification settings as well as a button to delete the course entirely.

What follows is a collection of eight low-fidelity wireframe images that depict the basic user interface for each of the eight screens included in the application.


Term Planner Login
Text prompt here telling user what to do.
Username Textbox
Password Textbox
Button
Term Planner
Add Term
Welcome to term planner!
Term 1
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY
Term 2
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY
Term 3
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY


Add Term
Save
Cancel
Use the form below to create a new term
Term Title:
Textbox
Start Date:
Date Picker
End Date:
Date Picker
Term Details
Save
Delete
Add Course
Use the form below to edit term details.
Term Title:
Term Name
Start Date:
Date Picker
End Date:
Date Picker
Term Courses
Course 1
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY
Course 2
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY
Course 3
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY


Add Course
Save
Cancel
Use the form below to create a new course.
Course Name:
Textbox
Start Date:
Date Picker
End Date:
Date Picker
Course Status:
Picker
Course Instructor's Name:
Textbox
Course Instructor's Phone Number:
Textbox
Course Instructor's Email:
Textbox
Notes:
Large Textbox
Course Start Notifications:
Toggle
Course End Notifications:
Toggle
Course Details
Save
Delete
Add Assignment
Use the form below to modify course details.
Course Name:
Textbox
Start Date:
Date Picker
End Date:
Date Picker
Course Status:
Picker
Course Instructor's Name:
Textbox
Course Instructor's Phone:
Textbox
Course Instructor's Email:
Textbox
Notes:
Textbox
Share Notes
Course Start Notification:
Toggle
Course End Notification:
Toggle
Course Assignments
Assign 1
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY
Assign 2
Start Date: MM/DD/YYYY
End Date: MM/DD/YYYY


Add Assignment
Save
Cancel
Use the form below to create an assessment.
Assignment Name:
Textbox
Start Date:
Date Picker
Due Date:
Date Picker
Assignment Type:
Textbox
Assignment Start Date Notification:
Toggle
Assignment Due Date Notification:
Toggle
Edit Assignment
Save
Delete
Use the form below to edit the assignment.
Assessment Name:
Textbox
Start Date:
Date Picker
End Date:
Date Picker
Assignment Type:
Textbox
Start Date Notification:
Toggle
Due Date Notification:
Toggle

Unit Test Plan

Introduction

Purpose

The semester scheduling mobile application is made up of many separate parts which work together to enable important functionalities and features. Each of these parts needed to be tested to ensure that it is working as it should. This testing was vitally important to ensure that the application meets or exceeds the needs of the customer.

Overview

For the testing phase of this project, I chose to use manual unit testing rather than automated unit testing. Every application requirement identified during the requirements phase of the project has been manually tested to ensure that it is adequately reflected in the completed application.

First, I created a test plan for each requirement. Such a test plan describes what application components are needed to carry out the test, what function or feature will be tested, what deliverables the test will produce, what tasks must be performed to complete the test, what environment must be set up to accommodate the test, and what criteria will be used to determine whether the test should be considered passed or failed.

Then, using those test plans as a guide, I carried out a unit test for each requirement. If the feature or function passed the test, I moved on to the next requirement. If the test was not passed, I made necessary changes to the application's source code before trying the test again to verify that the changes were adequate. This process of testing and improving the application was repeated until every test was passed successfully. One such test plan is included below.

Test Plan

Requirement Being Tested:

The application must include functionality that prevents users from creating a password that is less than six characters in length.

Items Required For Test:

To complete this test, the tester will need a compilable copy of the application's source code and project file.

Testing Environment:

Testing will be carried out using an Android Pie 9.0 device simulator built into Visual Studio 2019. Visual Studio's "debug" feature will be used to compile the project into an executable file and deploy it to the aforementioned device simulator. This will all be installed on a Windows 10 PC.

Test Tasks:

1. The tester will launch the application by clicking the green arrow in the debugging section of the toolbar at the top of the Visual Studio 2019 integrated development environment.
2. Once the application login screen appears in the simulator, the tester will enter a username and an invalid password that is less than six characters in length.
3. The tester will press the "Save" button.

Pass/Fail Criteria:

The following statements must both be true for the test to be passed.

A. Clicking the "Save" button while the password field contains an invalid password causes an alert notification asking the user to create a longer password to be displayed.

B. Clicking the "Save" button while the password field contains an invalid password does not

cause a navigation event, such as displaying a different application page. The test will be considered failed if either statement is not true. If the test is failed, necessary changes should be made to the source code, and the test should be repeated.
Expected Test Results/Expected Deliverables: The validation code successfully prevents a user from creating a password that is less than six characters in length and shows a notification that prompts the user to create a longer password.

Specifications

Pictured below is the code, located in the C# code-behind file for the login page, which provides the password validation functionality that the above test plan was designed to test.

The first image shows the code which is executed when the “Save” button is clicked during the test. This code calls a ValidateInput() method which returns true or false, depending upon whether the user has input valid credentials.

```
private async void submitButton_Clicked(object sender, EventArgs e)
{
    // If the application is being run for the first time, clicking submitButton will attempt to create
    // a new username and password. Otherwise it will be used to authenticate the user.
    if (Settings.FirstRun)
    {
        // The ValidateInput() method is called to check whether the user has entered a valid
        // username and password.
        if (ValidateInput())
        {
            //if the input is valid, the app will proceed to create a new user from the inputs.
            await CreateNewUser();
            await Navigation.PopAsync();
        }

    }
```

The second image shows code within the ValidateInput() method. This is the code that is responsible for displaying notifications if the password field does not contain a valid password.

```
private bool ValidateInput()
{
    if (string.IsNullOrWhiteSpace(userNameTextBox.Text)) // Was a username not provided?
    {
        DisplayAlert("Missing Username", "Please enter a valid username.", "OK");
        return false;
    }
    if (string.IsNullOrWhiteSpace(passwordTextBox.Text)) // Was a password not provided?
    {
        DisplayAlert("Missing Password", "Please enter a valid password.", "OK");
        return false;
    }
    if (passwordTextBox.Text.Length < 6) // Is the password less than six characters long?
    {
        DisplayAlert("Invalid Password", "Please enter password that is at least " +
            "six characters in length.", "OK");
        return false;
    }
    else
    {
        return true;
    }
}
```

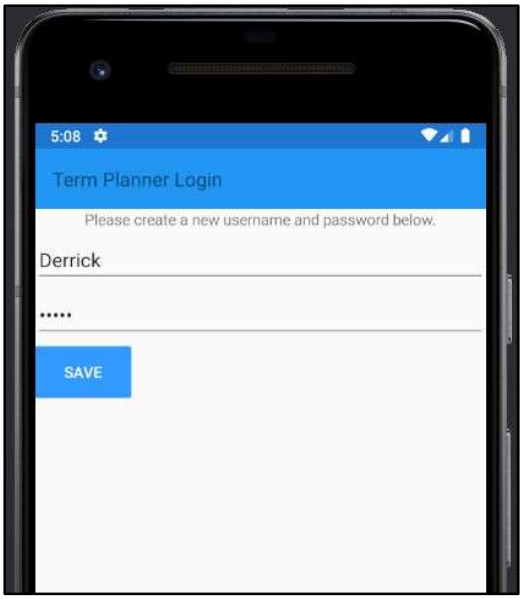
Procedures

For each requirement in the requirements document created during the requirements phase of the project, I created a corresponding unit test plan. Such a test plan describes what application components are needed to carry out each test, what function or feature will be tested, what deliverables the test will produce, what tasks must be performed to complete the test, what environment must be set up to accommodate the test, and what criteria will be used to determine whether the test should be considered passed or failed.

Then, using those test plans as a guide, I carried out a manual unit test for each requirement. If the feature or function passed the test, I moved on to the next requirement. If the test was not passed, I made necessary changes to the application's source code before trying the

test again to verify that the changes were adequate. This process of testing and improving the application was repeated until every test was passed successfully. Following are the steps I used to complete that process for the password validation test plan detailed in previous pages.

The first step once the environment was in order, was to click the green arrow in the debugging section of the toolbar at the top of the Visual Studio 2019 integrated development environment. This prompts Visual Studio 2019 to compile the application and deploy it to the simulator.



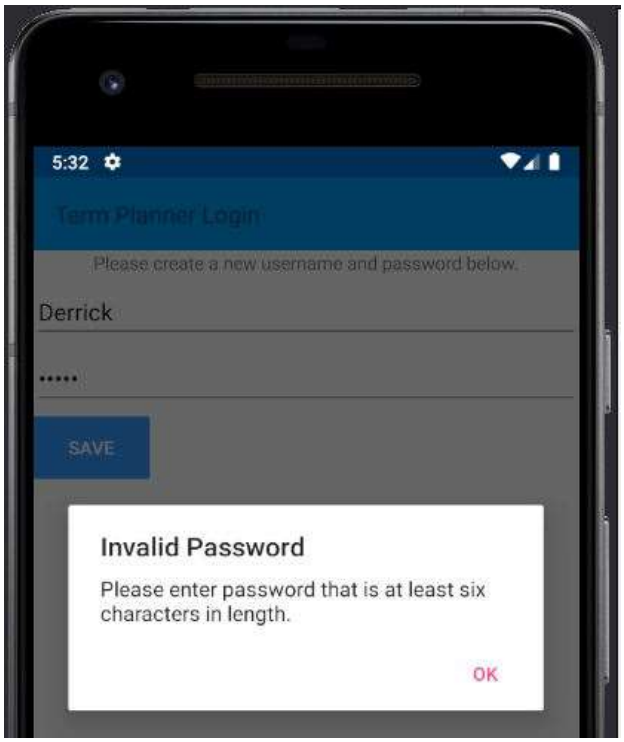
Once the application's login page was visible in the simulator, I entered a valid username and a password that was less than six characters in length.

Then, I pressed the "Save" button. The application displayed an alert asking me to enter a password that is at least six characters in length. As expected, the application did not execute further code or navigate to any other application page.

Results

The criteria for whether the unit test should be considered passed or failed were as follows: Clicking the "Save" button while the password field contains an invalid password

should display an alert prompting the user to enter a longer password. Further, clicking the save button while the password field contains an invalid password should not cause the application to navigate to a different screen. The below image depicts the application after I clicked the “Save” button while the password field contained a password that was less than six characters in length.



The application performed as expected and the test was passed. Because the test was passed, this unit test was removed from my list of remaining tests, and I moved on to the next unit test. If the test had failed, I would have explored what was preventing the application from performing as expected and made necessary changes to the source code before repeating the test.

Source Code

A compressed .zip archive containing all the source code files and the Visual Studio solution file has been submitted separately.

User Guides

Introduction

This user guide includes two sections. The first section, titled Maintenance Guide, describes how to set up an integrated development environment capable of editing source code, and debugging the application in a simulated Android environment. The second section, titled End User Guide, provides detailed instructions for end users, describing how to use each of the main functionalities included in the mobile application.

Maintenance Guide

To view, modify, and debug source code for the mobile application, you may wish to open its solution file and source code in an integrated development environment. This can be accomplished in several easy steps.

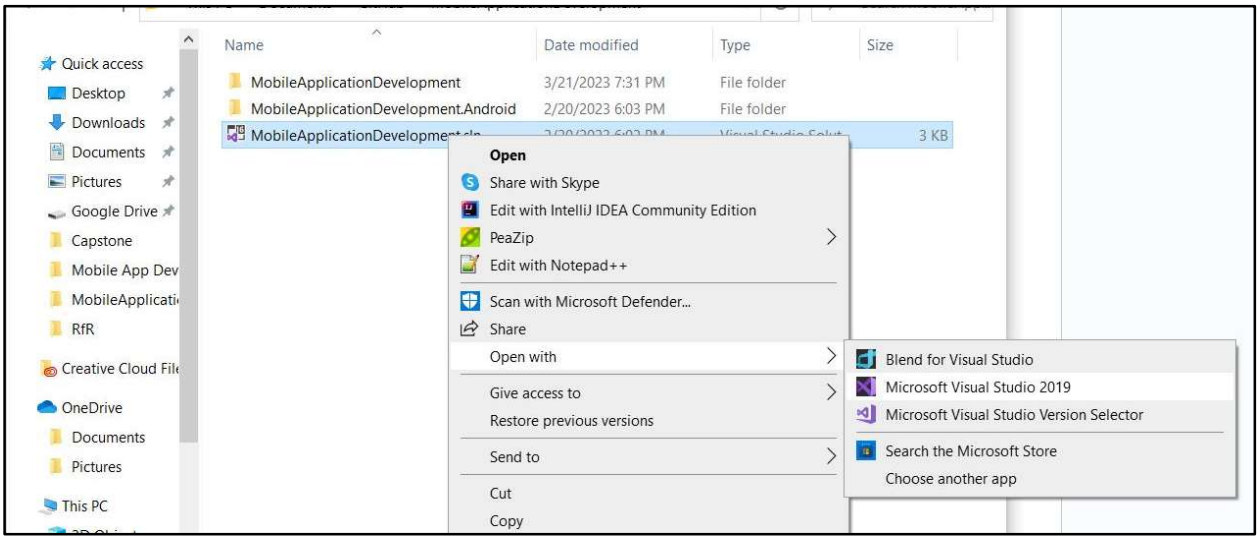
Setting Up The Environment

First, you will need Visual Studio 2019 and its Xamarin extension installed on your PC. If you do not already have those programs installed, you may find Microsoft's detailed instructions for installing Xamarin and Visual Studio 2019 helpful. These detailed instructions can be found at the following link: https://learn.microsoft.com/en-us/xamarin/get-started/installation/?pivots=windows-vs2019

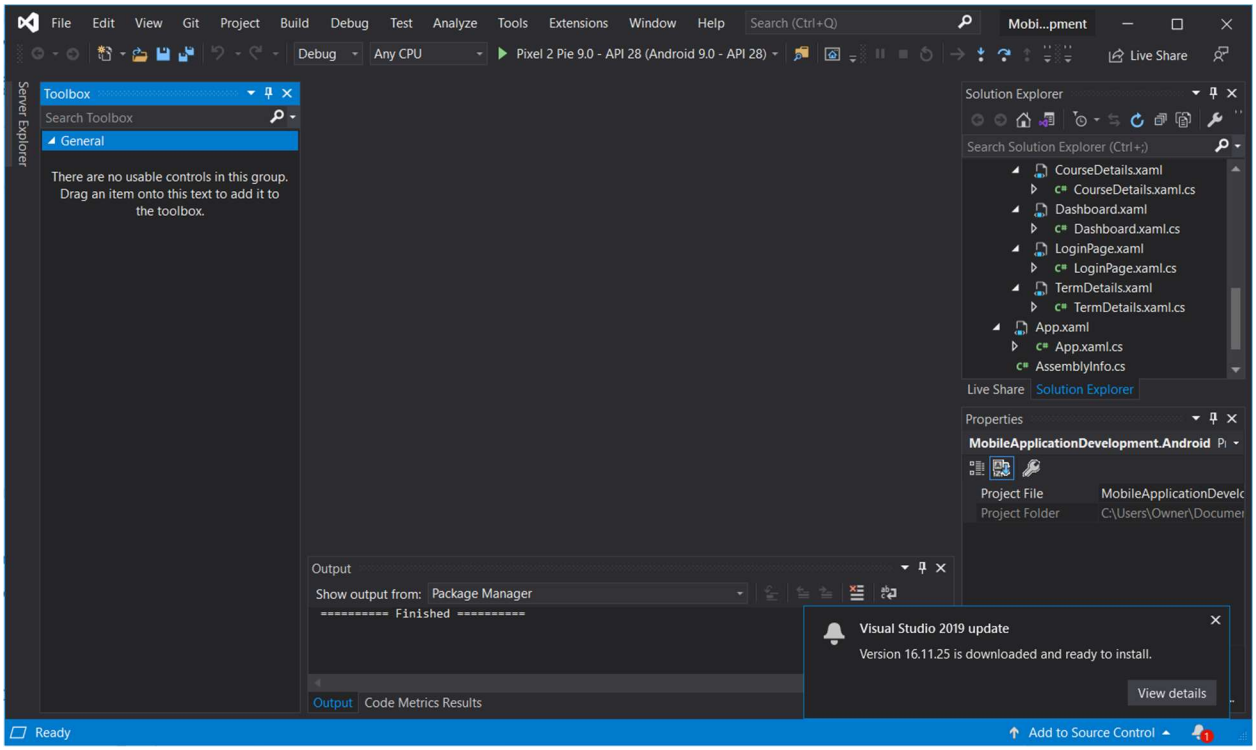
Opening The Project

Once the integrated development environment has been installed, decompress the .zip archive named MobileAppSourceCode.zip. In the folder created by decompressing the archive, navigate to and open the solution file named MobileApplicationDevelopment.sln using Visual

Studio 2019. See the screenshot below.



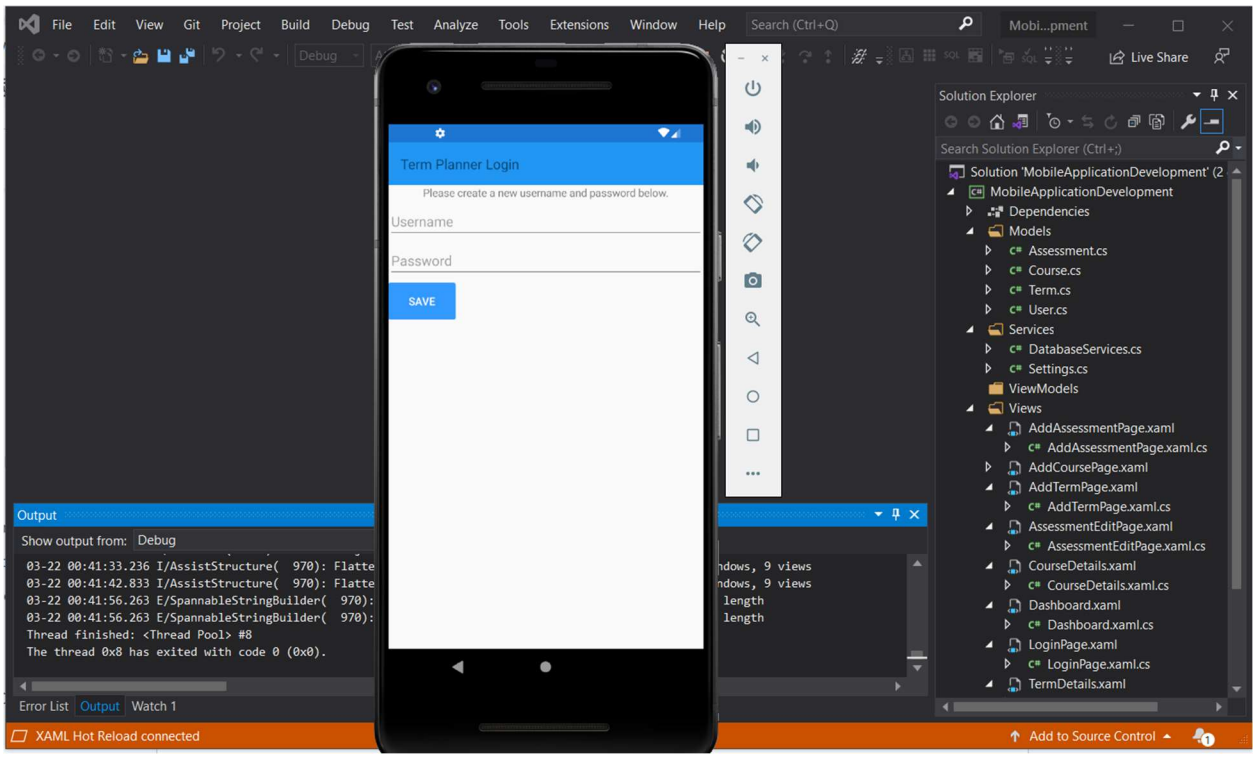
This will open the application's source code project in Visual Studio 2019. Once the project has loaded successfully, the Visual Studio application will look like the following screenshot. Note that the following image shows Visual Studio 2019 in dark mode. Color configurations may vary.



Running The Application

Once the source code project has been opened in Visual Studio 2019, you can run the application in a simulated Android environment by clicking the green arrow in the debug menu located within the toolbar near the top of the Visual Studio 2019 screen.

After you click the green arrow, an Android device simulator will appear, and the mobile application will be displayed on its virtual screen. You can then interact with the mobile application as you would on a mobile phone.

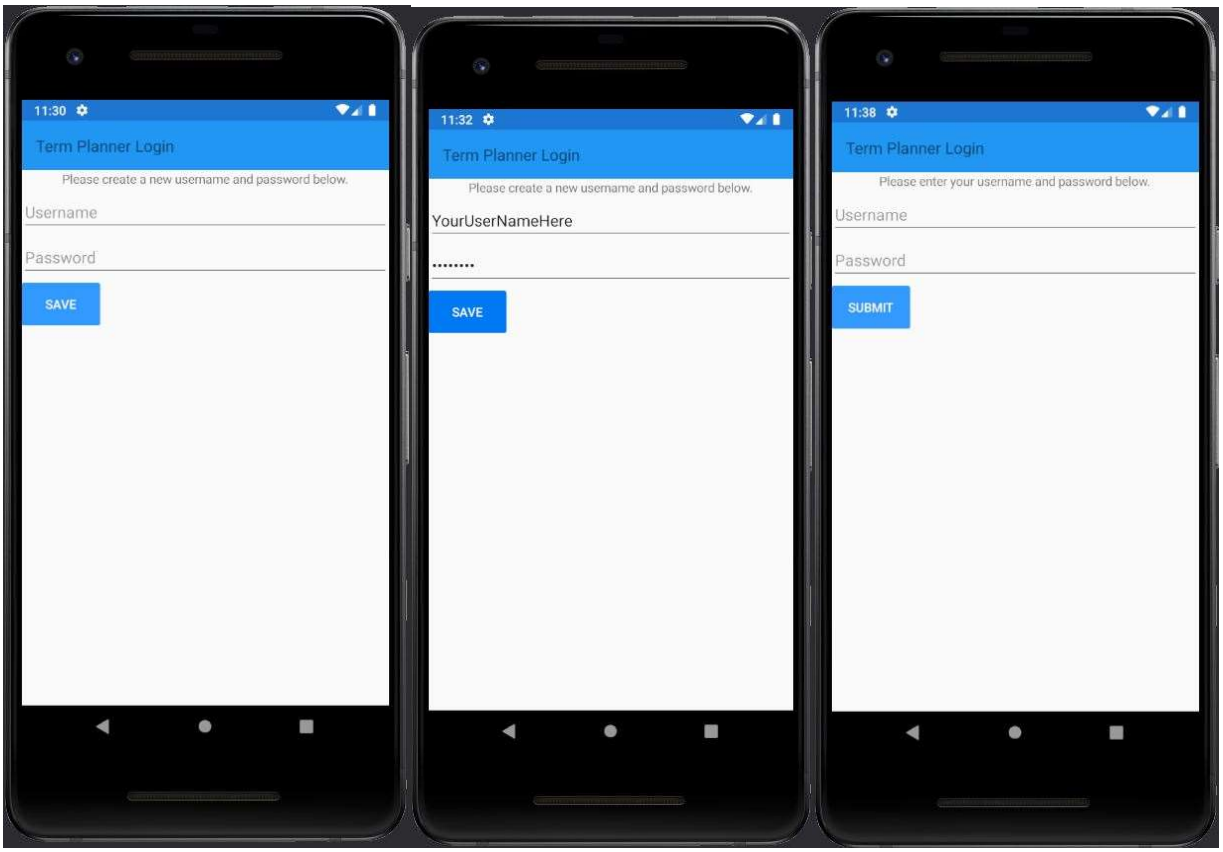


End User Guide

The semester scheduling mobile app provides students with an easy way to keep track of all their coursework in one place. This section of the user guide provides detailed instructions for end users, describing how to use each of the main features of the mobile application.

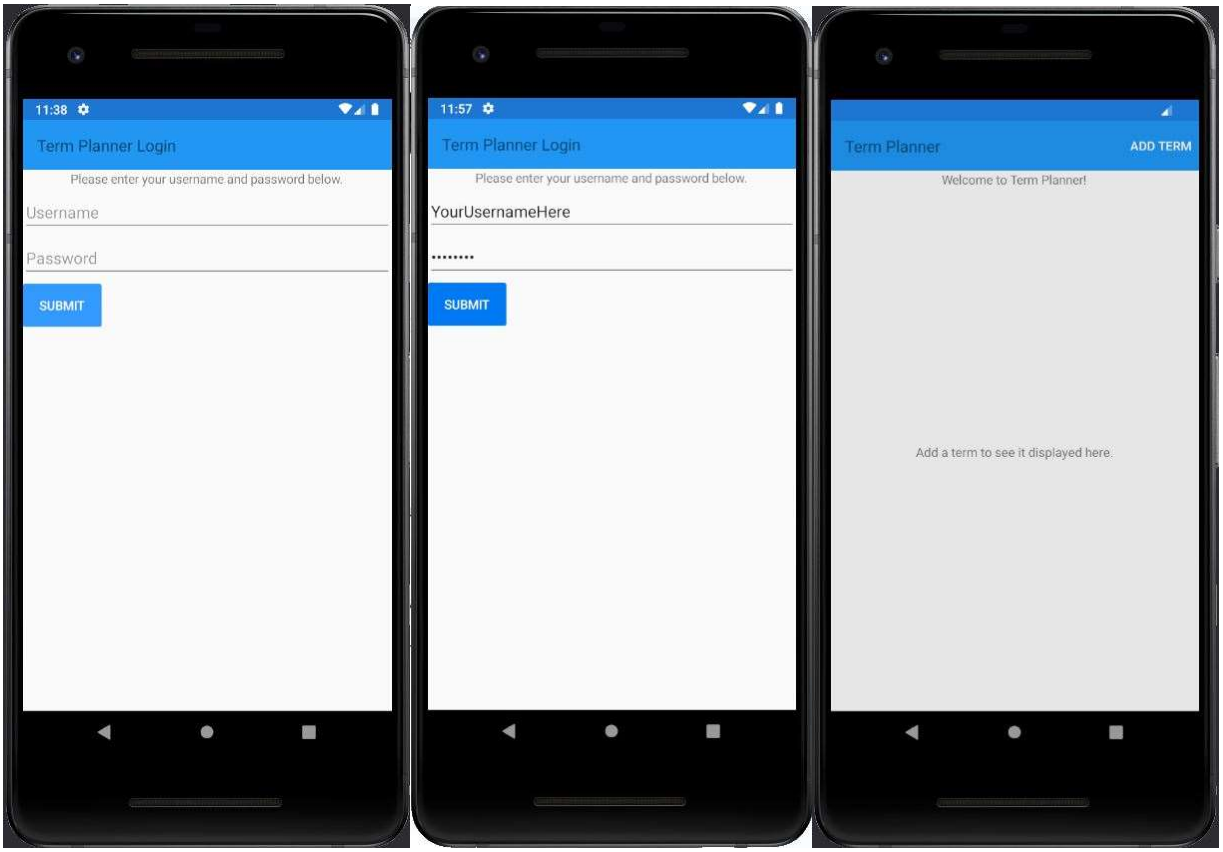
Creating Login Credentials

When the mobile application is run for the first time on a device, it will display a page prompting you to create a new username and password. See the first image below. This username and password will be used to password-protect your semester schedule information. To create your login credentials, enter a username and a password at least six characters long in the provided form. Then, click the “Save” button to save your credentials. You will then be redirected to the login page, shown in the third image below, where you will sign into the application for the first time.



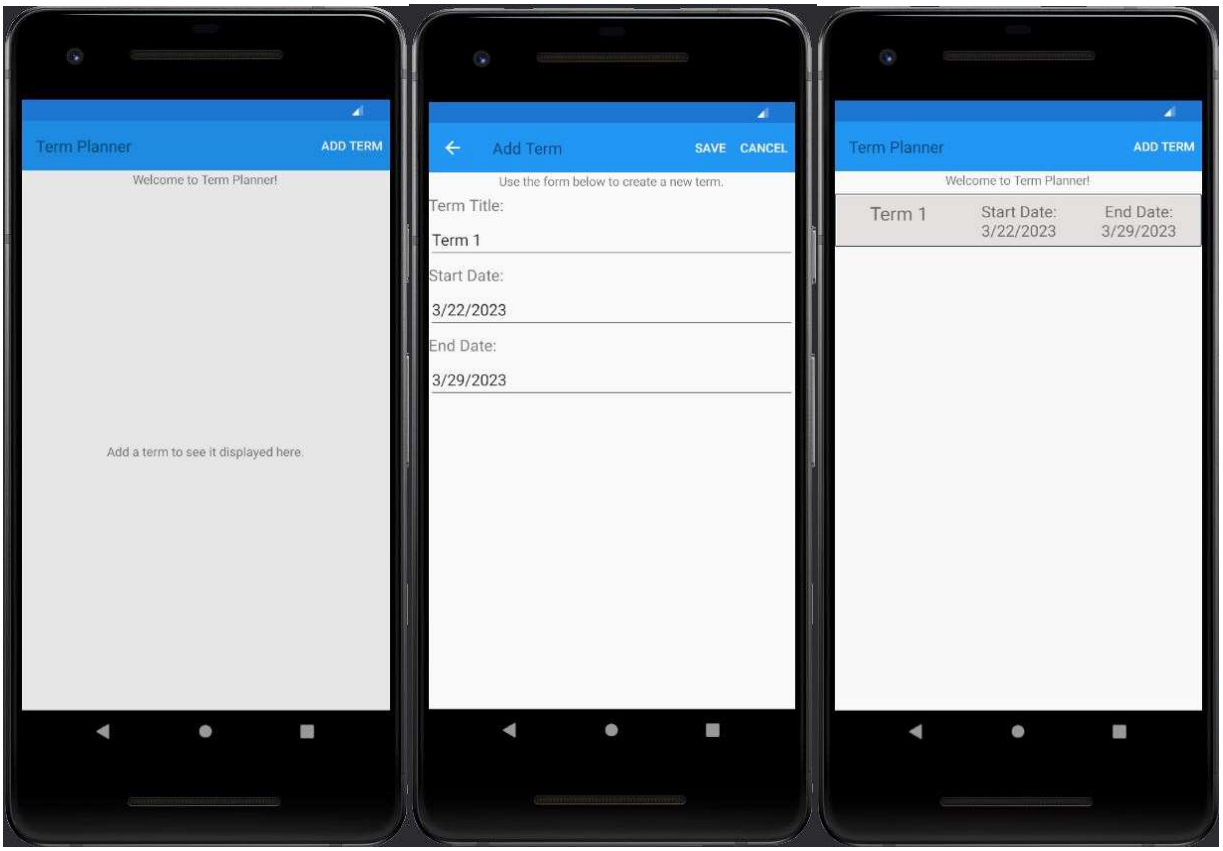
Signing In

After you have created username and password credentials, you will be required to use them to sign into the application. To sign in, enter your username and password in the form provided on the login page. Then, click the blue “Submit” button. The application will verify whether the provided username and password match the credentials you previously created. If they match, you will be redirected to a “Term Planner” dashboard page. You will be required to log in to the application each time it is launched before you can access information stored within it.



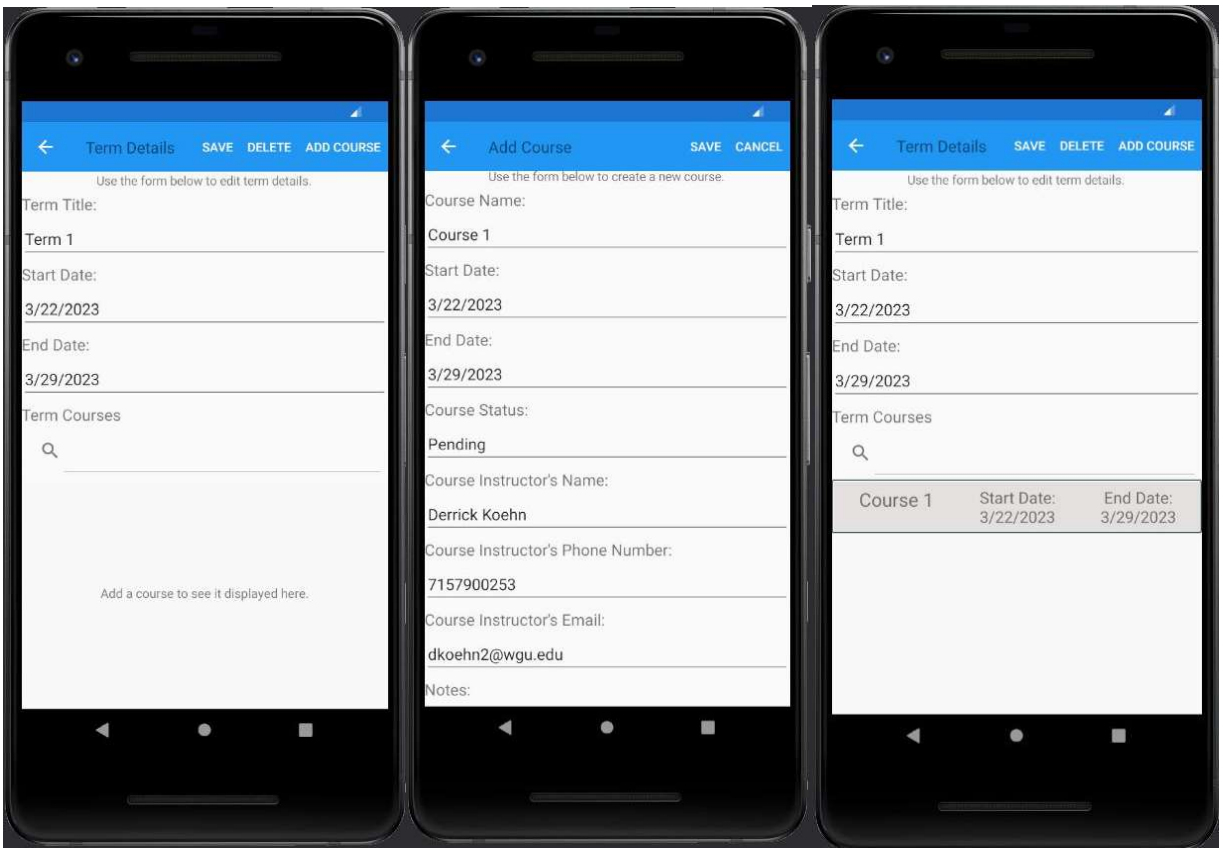
Adding A Term

To add a new school semester, referred to as a "term" throughout this document, to the application is very simple. First, log in to the application. Then, on the "Term Planner" dashboard page, tap the "Add Term" menu item near the top of the screen. You will be redirected to the "Add Term" page. Enter the term's title and its start and end dates in the provided form. The provided end date must be later than the provided start date. Then, click the "Save" menu item near the top of the screen. This will save a new term in the application's internal database. You will be redirected back to the "Term Planner" page which will display the term you just created.



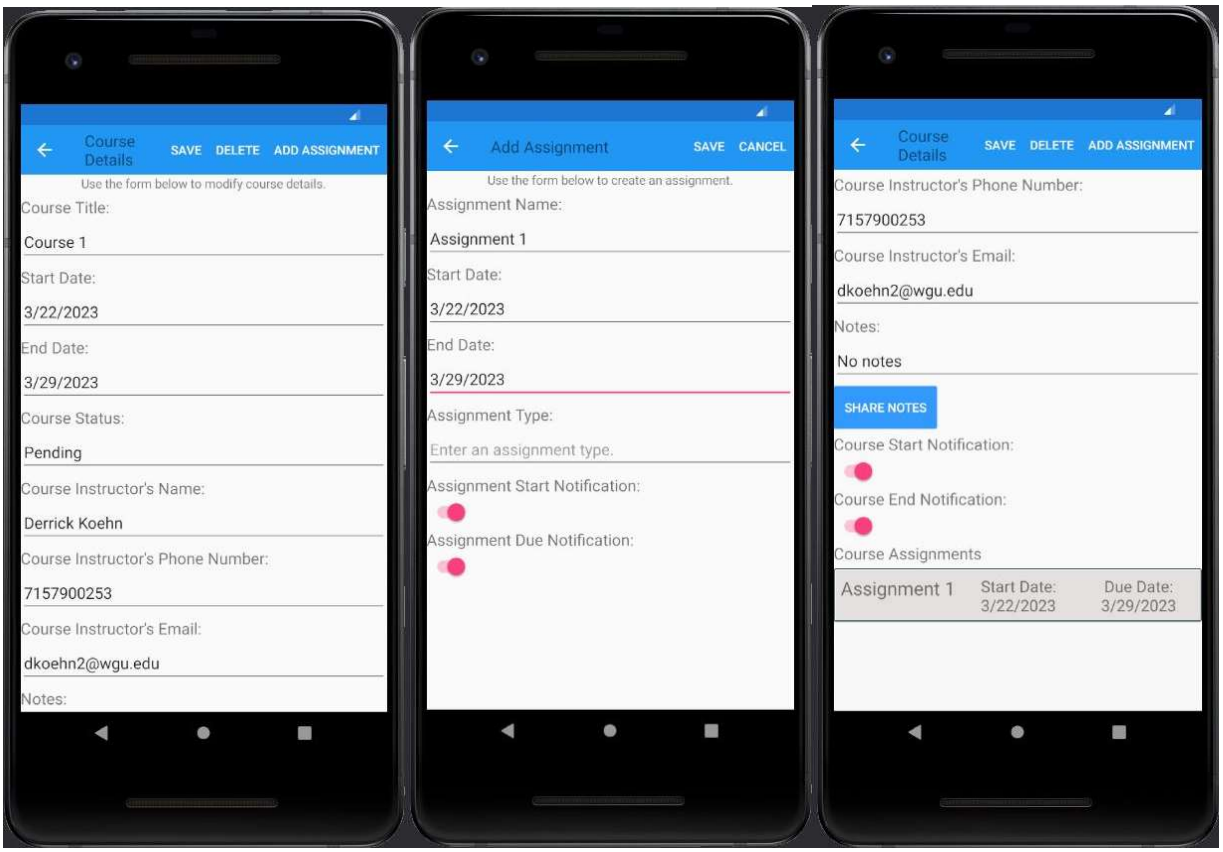
Adding A Course

Adding a new course to the application can be accomplished in several easy steps. Each course takes place during a term, to add a course, you must first have created a term. To create a term, use the instructions on the previous page. Once you have created a term to which you would like to add a course, tap on that term on the "Term Planner" dashboard. You will be redirected to a "Term Details" page populated with the term's details. Tap the "Add Course" menu item near the top of this page. You will be redirected to an "Add Course" page. Enter the new course's name and other attributes, including whether you wish to be notified on the course's start or end date, into the provided form. Then, tap the "Save" menu item near the top of the page. This will save the new course in the application's internal database. You will be redirected back to the "Term Details" page, which will display the course you just created.



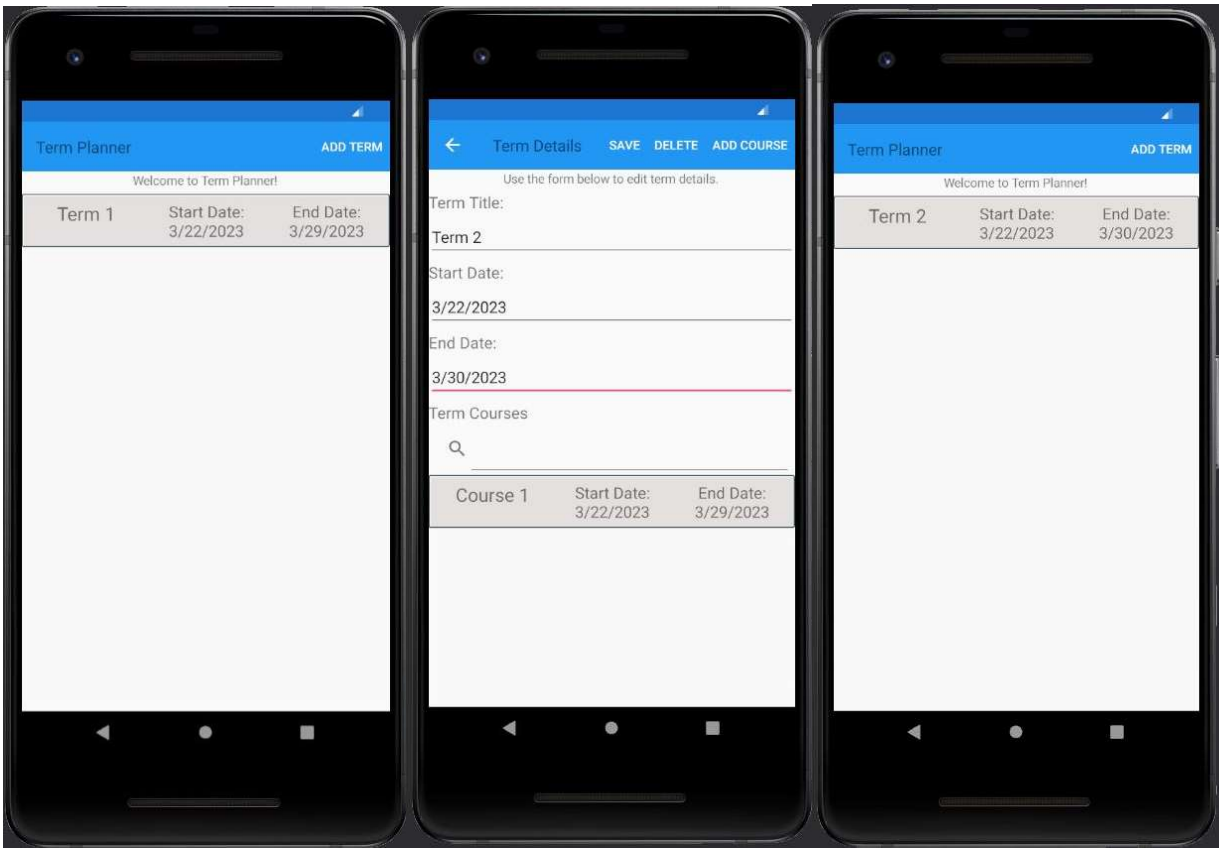
Adding An Assignment

Adding a new assignment to the application can be accomplished in several easy steps. Assignments are each a part of a course, so to add an assignment, you must first create a course. To create a course, use the instructions on the previous page. Once you have created a course, tap on that course on the "Term Details" page. You will be redirected to a "Course Details" page populated with the course's details. Tap the "Add Assignment" menu item near the top of this page. You will be redirected to an "Add Assignment" page. Enter the new assignment's name and other attributes, including whether you wish to be notified on its start or due date, into the provided form. Then, tap the "Save" menu item near the top of the page. This will save the new assignment in the application's internal database. You will be redirected to the "Course Details" page, which will display the assignment you just created near the bottom of the page.



Editing A Term

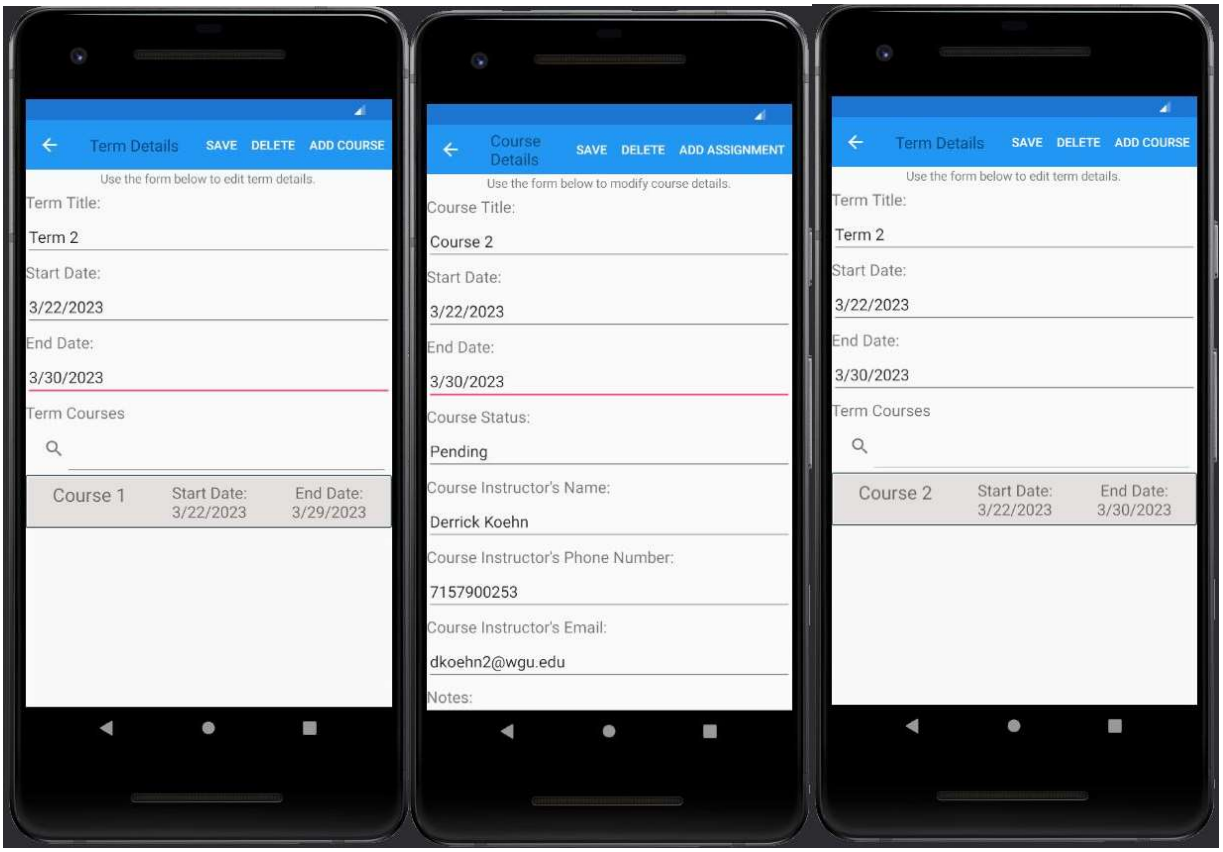
After you have added a term to the mobile application, you can update any of its details. To update a term's details, first, tap the term you want to update on the "Term Planner" dashboard page. You will be redirected to a "Term Details" page featuring a form populated with the selected term's details. After making desired changes to the information in the form, tap the "Save" menu item near the top of the screen. This will update the term's details in the application's internal database. You will be redirected back to the "Term Planner" page which will display the term you just modified.



Editing A Course

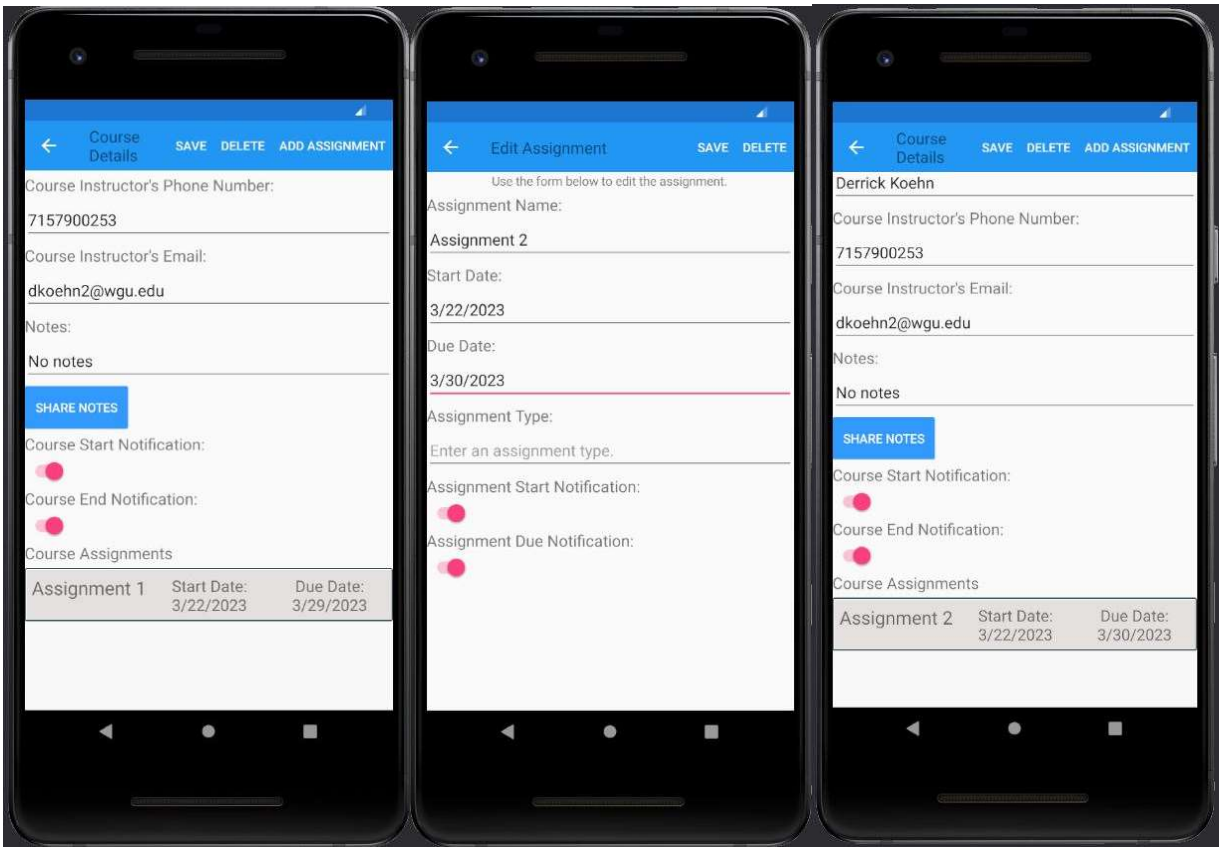
After you have added a course to the mobile application, you can update any of its details. To update a course's details, first, tap the course you want to update on the "Term

Details" page of the term it is a part of. You will be redirected to a "Course Details" page featuring a form populated with the selected course's details. After making desired changes to the information in the form, tap the "Save" menu item near the top of the screen. This will update the course's details in the application's internal database. You will be redirected back to the "Term Details" page which will display the course you just modified.



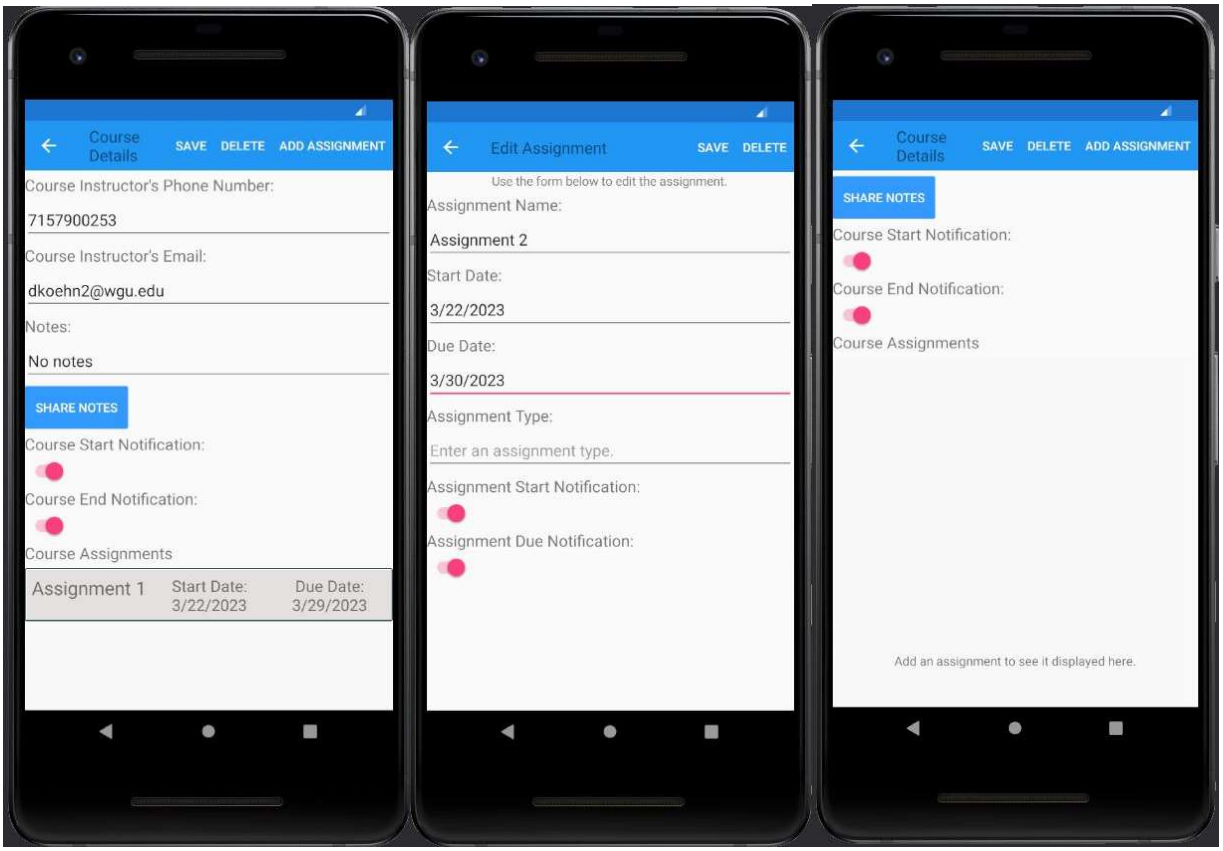
Editing An Assignment

After you have added an assignment to the mobile application, you can update any of its details. To update an assignment's details, first, tap the assignment you want to update on the "Course Details" page of the course it is a part of. You will be redirected to an "Edit Assignment" page featuring a form populated with the selected assignment's details. After making desired changes to the information in the form, tap the "Save" menu item near the top of the screen. This will update the assignment's details in the application's internal database. You will be redirected back to the "Course Details" page which will display the assignment you modified.



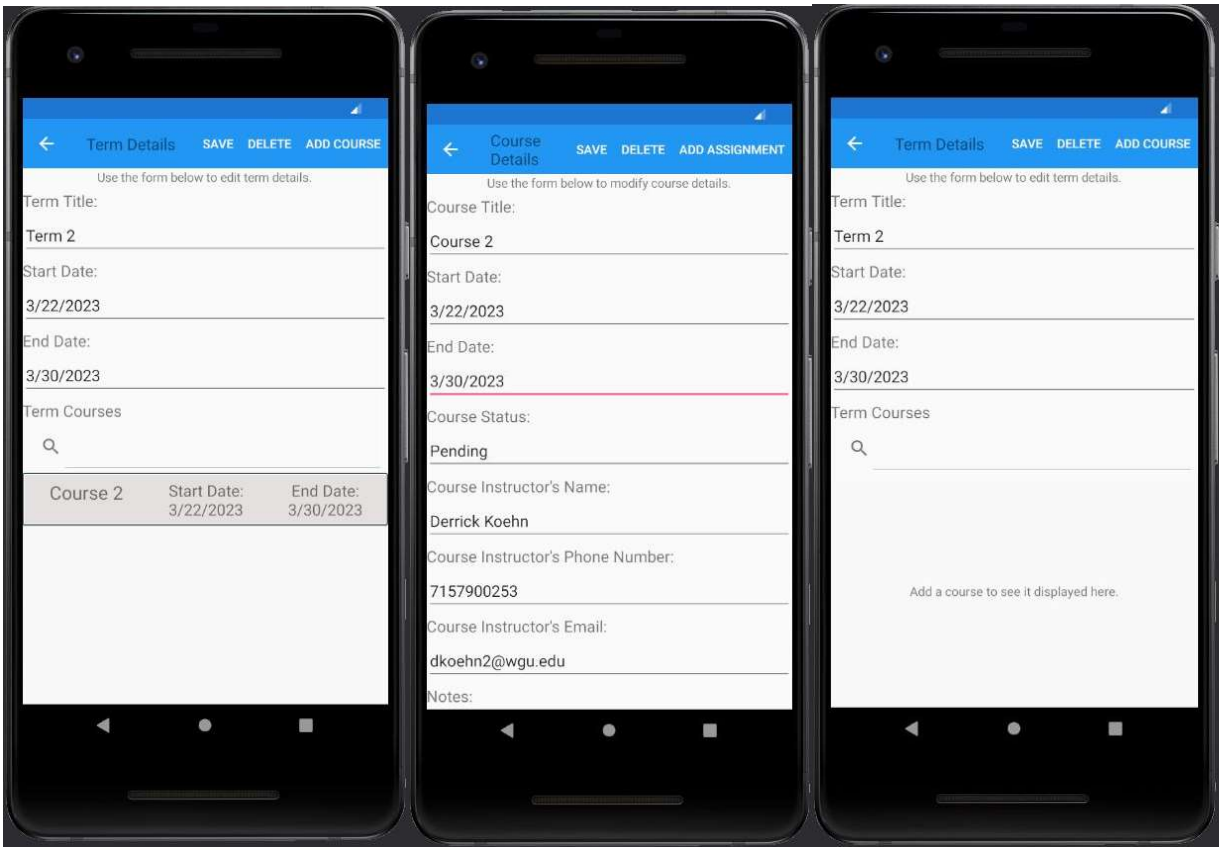
Deleting An Assignment

Any assignment saved in the application can be deleted. To delete an assignment, first, tap the assignment you want to delete on the "Course Details" page of the course it is a part of. You will be redirected to an "Edit Assignment" page featuring a form populated with the selected assignment's details. Tap the "Delete" menu item near the top of the screen. This will remove the assignment from the application's internal database. You will be redirected back to the "Course Details" page which will no longer display the assignment.



Deleting A Course

Any course saved in the application can be deleted. To delete a course, first, tap the course you want to delete on the “Term Details” page of the term it is a part of. You will be redirected to a “Course Details” page featuring a form populated with the selected course’s details. Tap the “Delete” menu item near the top of the screen. This will remove the course and any assignments associated with the course from the application’s internal database. You will be redirected back to the “Term Details” page which will no longer display the course.



Deleting A Term

Any term saved in the application can be deleted. To delete a term, first, tap the term you want to delete on the “Term Planner” page. You will be redirected to a “Term Details” page featuring a form populated with the selected term’s details. Tap the “Delete” menu item near the top of the screen. This will remove the term and any courses or assignments associated with the term from the application’s internal database. You will be redirected back to the “Term Planner” page which will no longer display the term.