**C868 – Software Capstone Project Summary**



**Task 2 – Section C**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | CDUTermTracker- Term Tracking  Application for CharDennis University |
| **Student Name:** | Shannon Marie Peck |

**Table of Contents**

Table of Contents

[Table of Contents 2](#_Toc105171368)

[Task 2 Part C – C868 Software Development Capstone 3](#_Toc105171369)

[Application Design 3](#_Toc105171370)

[Design Document 3](#_Toc105171371)

[Class Design 3](#_Toc105171372)

[UI Design 5](#_Toc105171373)

[Application Testing 7](#_Toc105171374)

[Unit Test Plan 7](#_Toc105171375)

[Introduction 7](#_Toc105171376)

[Overview 7](#_Toc105171377)

[Test Plan 7](#_Toc105171378)

[Pass/Fail Criteria 10](#_Toc105171379)

[Specifications 10](#_Toc105171380)

[Procedures 11](#_Toc105171381)

[Results 12](#_Toc105171382)

[C4. Source Code 12](#_Toc105171383)

[User Guide 13](#_Toc105171384)

[Application Maintenance Guide Introduction 13](#_Toc105171385)

[User Guide: End User Setup and Installation 17](#_Toc105171387)

Task 2 Part C – C868 Software Development Capstone

# Application Design

## Design Document

The following sections contain CDUTermTracker application’s design documents. These artifacts were used by developers during creation of the application to provide a programmatic design framework and user-friendly User Interface. The UML diagram displays the classes to be used and their associated methods and field; the low-fidelity wireframe provides a preliminary design plan; and the more detailed prototype contains a more detailed and accurate representation of the final product.

### Class Design

The class diagrams below display the classes used in C# to create the term tracker. The design method used in this project is the “Model-View-View Model” object-oriented software architecture. As such, each component of the MVVM architecture is split into its own class diagram to more easily view relationships and components of each class.

The MVVM architecture splits data from display of the data. The Models handle backend manipulation of SQL database tables (ObjectiveAssessment, PerformanceAssessment, and Term tables). The View Models serve as the interface between the front end Views, where the user enters and manipulates data, and the Models which update the database. It controls the flow of data, and what is displayed. The views control how the data looks to the user.

The UML diagram displays each class and its fields, properties, and viariables. Inheritence is indicated using lines between classes, with a direction arrow from the derived to the parent class. Performance Assessments and Objective Assessments both inherit from Assment (each are a type of assessment).

The SQL database contains: Objective Assessment, Performance Assessment, Course, and Term tables. The primary keys of Term are used to form a one-to-many relationship with each Term’s course. Likewise, the primary keys of Course are used to form a one-to-one relationship with a Course’s Objective Assessment and Performance Assessment.

#### Models

Graphical user interface

Description automatically generated

#### View Models

A picture containing text

Description automatically generated

#### Views

Graphical user interface, application

Description automatically generated

### UI Design

#### Low-fidelity wireframe

A low-fidelity wireframe of the prototype’s entry page (Terms View), Course View, and Assessment View can be found in this section. This documentation provided an initial estimation of what the layout would look like, to guide the development team.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

#### Prototype

A high-fidelity wireframe was designed following implementation of initial functionality. This design focused more on user experience. This prototype includes the screens used to enter information, and error messages.

A screen shot of a calculator

Description automatically generated Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

Graphical user interface, text

Description automatically generated

# Application Testing

## Unit Test Plan

### Introduction

The testing plan for this project focused on unit tests on the methods within the code to ensure that the logic works, and the output results are as expected.

#### Purpose

Testing ensures that methods within the application’s classes function as expected and finds errors if any are present. Unit testing is performed on these methods to ensure that the smallest bits of code perform as expected; this means that the development team will identify problems earlier in the code to prevent larger problems down the road.

### Overview

When a new Course is created, Instructor information is added by the user including name, phone number, and email address. Validation is run on the phone number to ensure it is a valid, properly formatted phone number. If validation fails, an error message to “Please enter a valid phone number” is sent to the user. This validation also runs when a Course is updated, and when a user shares Course notes externally. The database call is not performed until the phone number is validated, to ensure that unsafe code is not being inserted into this database field. If the phone number is validated, the Add Course, Update Course, or Share notes method continues.

### Test Plan

#### Items

Unit testing utilizes Visual Studio’s built-in Visual Studio Unit Test Project. This project contains the PhoneNumberValidationTest class, with tests that run in Visual Studio.

#### Features

The PhoneNumberValidationTest unit test is testing the IsValidPhone method within the Validation class. IsValidPhone() accepts a user-entered string, the phone number, and returns a Boolean. This Validation method runs when a new Course is added, when an existing Course is updated, and when course notes are shared.

#### Deliverables

The tests produce output to Visual Studio’s testing console. Visual Studio runs each unit test, and outputs its success or failure relative to the expected output defined in the PhoneNumberVaidationTest class. Additional details can be found in the Test Detail Summary (located on the left of the image below), for debugging in the event of failed tests.

A screenshot of a computer

Description automatically generated with medium confidence

#### Tasks

1. This application utilizes Visual Studio’s built-in Unit Test Project and Test Explorer.
   1. Tests should be in a project within the TermTracker solution, named: TermTracker.UnitTests
   2. Test methods are found within individual classes in that project

A screenshot of a computer

Description automatically generated with medium confidence

1. Under the “Test” tab, select “Run all tests”
2. Allow the tests to run
   1. Visual Studio’s Test Explorer will open, and run all test methods within TermTracker.UnitTests
3. The results of this project’s unit testing are found below
   1. All tests pass for the IsValidPhone() validation method

A screenshot of a computer

Description automatically generated with medium confidence

#### Needs

* Visual Studio 2019
* Application code, with the following structure:
  + TermTracker project
    - Models folder
      * Validation class
        + IsValidPhone() method that accepts a string
* Unit Test Project within the TermTracker solution
  + This is a built-in project type found in Visual Studio
  + This project must contain:
    - A class
      * This class contains test methods
* Visual Studio Test Explorer
  + This is also built-in to visual studio

### Pass/Fail Criteria

All tests listed in Visual Studio’s Test Explorer should have green check marks once run (see image above). This indicates that test has passed with the expected outcome as defined in its test method.

### Specifications

The following is a sample of the code used in unit testing for this project.

Text

Description automatically generated

### Procedures

1. Unit tests were written prior to code, with a “test first” strategy.
2. Within the TermTracker solution, create a Unit Test project.
3. Within the TermTracker.UnitTests project, create a PhoneNumberValidationTest class.
4. Within the PhoneNumberValidationTest class, determine acceptable phone number formats and write tests expected to pass the phone number validation tests.
5. Within the PhoneNumberValidationTest class, determine non-acceptable phone number formats and write tests to expected to fail the phone number validation tests.
6. Write the Validation.IsValidPhone() method within the TestTracker project.
7. Initial tests had one failing test, so the Regex used in the IsValidPhone() method was re-evaluated to find the cause of the error. This code was corrected.
8. The test was run again, and the all tests passed
9. Development moved on to the next method in the project

### Results

A screenshot of a computer

Description automatically generated with medium confidence

# C4. Source Code

Source code is submitted in a directory titled “TermTracker.zip” This .zip folder was submitted alongside this documentation.

Note: this application is not a web app, so there is no link to the live version of this project. The project can be downloaded onto an Android phone or opened in an Android emulator using the CDUTermTracker.apk file, also submitted alongside this documentation.

# User Guide

## Application Maintenance Guide Introduction

This User Guide documents how to set up and run CDUTermTracker as a **developer** for the purpose of maintenance. For instructions on installation and setup as an end user, see the User Guide: End User Setup and Installation section.

### Installation

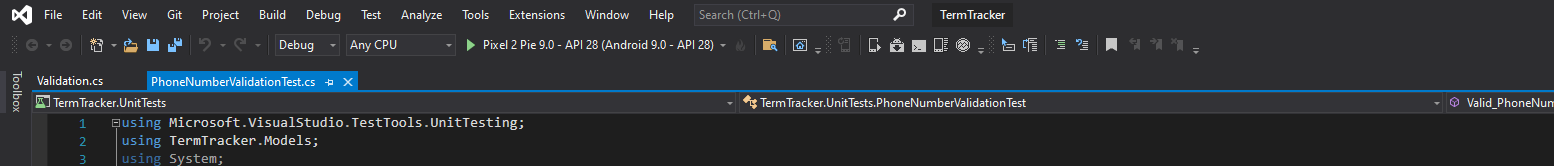
The following section documents how to set up your local development environment for the purpose of maintaining the CDUTermTracker application.

#### System Requirements

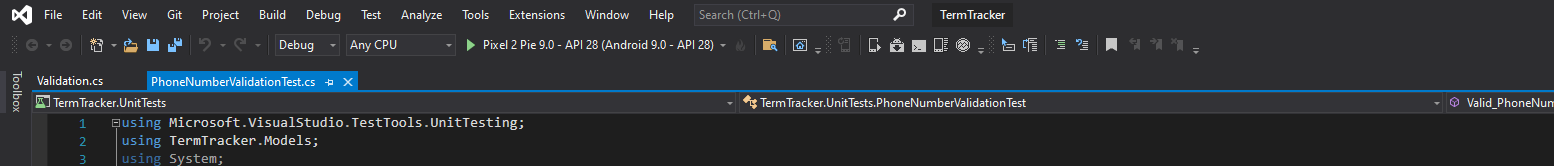
* CDUTermTracker.zip
  + Contains the source files for the application
* Windows 10
* Visual Studio 2019
* Visual Studio NuGet packages:
  + NETStandard.Library
    - v.2.0.3
  + SQLite-net-pcl
    - v.1.7.335
  + Xam.Plugins.Notifier
    - v. 3.0.1
  + Xamarin.Essentials
    - v1.5.3.2
  + Xamarin.Forms
    - v. 4.7.0.1142

#### Installation Steps

1. Extract the CDUTermTracker.zip to the desired location on your local computer.
2. Navigate to the resulting folder (likely titled TermTracker unless renamed during the extraction process).
3. Within the folder, right click the .sln file (TermTracker.sln) and “Open with” Visual Studio 2019.
   1. You can also double-click this file, but if you have multiple versions of Visual Studio installed, right-clicking will ensure the correct version of Visual Studio is used.
4. Navigate to Project 🡪 Manage NuGet packages to ensure the correct versions (listed in System Requirements) are installed and in use.
5. Ensure the Solution Configuration dropdown is set to “Debug”



1. Ensure the Android dropdown is set to “Pixel 2 Pie 9.0 – API 28 (Android 9.0 – API 28)



* 1. If this option is not available in the dropdown, right click the Project entitled “TermTracker.Android” and navigate to Android Manifest. Set the Minimum Android Version **and** Target Android Version to Android 9.0 (API Level 28 – Pie)

1. To run the program in the Android Emulator, click the green arrow (above) after the correct Android version is set.
   1. An Android phone emulator will open (a Pixel 2), and after a few seconds the TermTracker application will launch. If this is the first run, the start page will not contain any terms because the local database will be empty.

Graphical user interface, application

Description automatically generated

* 1. See User Guide below for “How-to” use the application

1. To make changes to the application, access classes and methods within the TermTracker project in Visual Studio’s solution explorer
   1. This is where the bulk of the project’s logic is contained
2. To make changes to tests, access classes within the TermTracker.UnitTests project in Visual Studio’s solution explorer

## User Guide: End User Setup and Installation

### Introduction

This User Guide documents how to set up and run CDUTermTracker as a **user.** For maintenance and to access the application as a developer, please see the Application Maintenance guide.

The purpose of this guide is to aid the user in setup and use of CDUTermTracker application.

#### Installation and Using the Application

* Install the application by downloading the CDUTermTracker.apk file onto your Android device.
* Run the application by tapping the TermTracker icon.

Graphical user interface, application

Description automatically generated

* The application will open to the “All Terms” display, from which you can view all terms that have you entered into your application.

#### Terms

##### Create a new term

* Click on the “Add Term” button in the header

Shape, square

Description automatically generated

* Enter Term Name, Term Start Date, and Term End date
  + The Term End Date must be after the Term Start Date

Graphical user interface, application

Description automatically generated

* Click “Add Term” when you are satisfied with the information
* When you add a term, the application will automatically navigate you back to your startup screen where you can “View All Terms”

Graphical user interface

Description automatically generated with medium confidence

##### Viewing and Editing Terms

* You can navigate to a detailed view of any term by tapping that term
* From the detailed view, you can add courses
  + Each Term can contain no more than 6 courses
  + See below for How to add courses
* You can also edit the term, by clicking “Edit Term” in the header bar

A picture containing shape

Description automatically generated

* + This will take you to a screen that allows you to edit all details previously entered about that term
* You can delete a term by tapping “Delete Term” (to the right of “Edit Term” above).
  + When you delete a term, all of its associated courses and assessments will also be deleted

#### Courses

Terms contain courses; each term can contain no more than 6 courses.

##### Adding a Course

* To add a Course, navigate to the term into which it occurs. Select “Add Course” in the term’s header bar (see image above, “Add Course” is to the left of “Edit Term”).
* From the Add Course screen, you can enter:
  + The course’s name
  + The course’s start and end dates
  + The course due date (this is automatically set to the term’s end date)
  + The course’s status
  + The course instructor’s name, phone number, and email
  + You can toggle on/off start and end date notifications for the course
  + You can enter optional notes

Graphical user interface, text

Description automatically generated

* All of the fields above are required, except the Notes field
  + All of the above fields can be later edited, including the Notes field
* Click “Add Course” when you are satisfied with the course details.

Graphical user interface, text, application

Description automatically generated

* When a course is added, you are automatically navigated back to its Term’s details page, where all courses for a term are listed

##### Viewing Course details

* You can view a course’s details page by tapping on that course while in a Term’s details page
* You can also search for a course from the startup/All Term menu

A picture containing graphical user interface

Description automatically generated Graphical user interface, application

Description automatically generated

##### Editing or Deleting a Course

* While on a Course’s detailed view, click “Edit Course” in the header

Graphical user interface, text, application

Description automatically generated

* + This will take you to a screen to edit all of the course details you previously entered, except the Notes field. See below for how to edit notes.
  + Click “Update Course” when you are satisfied with your changes.

Graphical user interface, application

Description automatically generated

* You can also delete a course by clicking “Delete Course” in the header, to the right of the “Edit Course” option
  + Deleting a course will also delete the assessments associated with that course

##### Editing and sharing course notes

* From the detailed Course view, you can edit or share a course’s notes
* When you tap “Share notes”, a field to enter the phone number with which you will share the notes will appear

Graphical user interface, application

Description automatically generated

* + Click “Send” to send your notes
* To edit notes, click “Edit Notes” to the right of “Share Notes”
  + This will take you to a screen to edit your course notes.

Shape

Description automatically generated

* + Click “Update Notes” to save your changes

#### Assessments

* When you add a course, application automatically creates two assessments for you
  + An Objective Assesment and a Performance Assessment
* To view these assessments, click “View Assessments” from the detailed course view

Graphical user interface, text, application

Description automatically generated

* You’ll notice that the Performance Assessment’s name is “Performance Assessment” and its due date is set to the Course’s due date
  + You’ll also notice that the Objective Assessment’s name is “Objective Assessment”, its due date is set to the Course’s due date, and its scheduled date is set to midnight on the course’s due date

Text, letter

Description automatically generated

* You can tap on either assessment to navigate to that assessment’s details page

Graphical user interface, text, application

Description automatically generated

* You cannot delete assessments, as each course requires both assessments be completed.

##### Editing an Assessment

You can edit both Objective and Performance Assessments. Because Objective Assessments have more details than Performance Assessments, this guide will focus on Objective Assessments. You can edit Performance Assessments in the same way.

* Click “Edit Assessment” from the assessment’s detailed page (pictured above)
* From the edit screen, you can edit all of an assessment’s details. Pictured below is the edit screen for an objective assessment.

Graphical user interface, text, application, email

Description automatically generated

* + A performance assessment’s edit screen will look similar to this. However, performance assessments do not have PreAssessment Scores or scheduled dates (and therefore no scheduled date notification).
* If you have not yet entered your objective assessment’s scheduled date, you’ll have a red notification that reads “Exam not yet scheduled”.

Graphical user interface, text, application, email

Description automatically generated

* You can toggle scheduled date and due date notifications from this screen.
* Click “Update Assessment” when you are satisfied with your changes.
* When you update an assessment, the application will automatically navigate back to that assessment’s details page.

#### Reports

You can view a report of upcoming objective assessments from the Report’s page.

Graphical user interface

Description automatically generated with medium confidence

* You can enter a name for your report, and a start date and an end date during which to search for scheduled objective assessments

Graphical user interface, text, application

Description automatically generated

* Click “Generate Report” to view the report’s results
* The application will display any objective assessments which have scheduled dates that fall within that interval, or a “Report generated 0 records” message
  + You’ll also see a timestamp that indicates when the report was created

A picture containing text

Description automatically generated