**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**

*Create a professional looking Table of contents that includes your main and subheadings and the related page numbers. Use the automatic TOC generating function of Word or other word processing packages to make the process easier.*

*The headings that follow are only examples of what might be included. You will need to create headings that are appropriate for your application and process.*

***Remember that this needs to be a professionally formatted document with detailed information about your project that is easily accessible.***

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

#### 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

Here you go into more detail about the test(s) and how it related to the overall project. You should include if a similar method was used in other parts of the application or why this was unique for a certain aspect of the code. Then, go into detail about what functions were tested, how the tests were conducted, and how errors were dealt with.

### Test Plan

#### Items

What is required to complete the test(s)?

#### Features

List the function/features that are part of each test.

#### Deliverables

List what the test(s) would produce. For example, documentation or code notations.

#### Tasks

List the tasks required to complete the testing and provide the outcomes you identified.

#### Needs

Describe what was needed to be running or what support items had to be in place to perform the test? Specify versions if appropriate and other technical requirements. If a testing package and/or library was employed, be sure to identify it/them.

### Pass/Fail Criteria

Describe the criteria you used to determine the success of the test and what the protocol was for a positive result. Also describe what the recourse was if the test failed including remediation strategies and documentation requirements.

### Specifications

Provide sample code that represents what testing code was used. Screenshots are acceptable.



### Procedures

Provide a detailed list of the steps you used to complete the testing process. Be sure to mention if iterations were/are part of the process used and when pass/fail results were provided.

### Results

Here you will describe and provide examples of the testing results. If you were using a testing package include a screenshot of the interface. Screenshot work best.



# C4. Source Code

All source code for the application must be submitted with your documentation. Remember, this must be a fully functional application and the evaluator will need the code and other functional pieces to make the application launch and work as described. For web-based applications, a live website is advantageous and must be accessible to the evaluator, but you still must submit the complete code. Use a separate file for this section and identify the name.

# Link to live version

CDUTermTracker is a mobile app; there is no link to the live version. See the “User Guide” below for app documentation, including setup and maintenance.

# User Guide

*Note: This may be included as a separate document if you desire.*

## Introduction

Provide a description of the content you’re providing in the User Guide. This guide will include how to install, log into, sign up, and use all of the functions of the application. The steps need to be clearly defined and fully tested so the process works flawlessly for the evaluator.

## Installation and Using the Application

This procedural information should follow the basic rules of such technical references. While some procedures may provide for personal judgment yours should be clear and concise. Here are other rules to remember:

* Provide step-by-step sequences in the correct order.
* Follow the timing and sequencing of the actual operations.
* Provide visual stepping stones by using bullets or labeling steps.
* Strive to be concise. Avoid lengthy paragraphs but include enough detail so false assumptions are not made.
* Use common terms and jargon appropriate for the audience (someone with basic IT background).
* Explain why steps are completed or what they will yield as well as "How to" instructions.
* Test the instructions to ensure they match the actual product.
* Format the material for ease of reading and use graphic aids to clarify point/steps.
* Write in the present tense and the active voice.

## *Login and Signup (An example*)

1. *Click the "Log in" button in the top right corner of the app.*

**

1. *If you already have an account, log in with your account name and password. If you need an account, click on the link below that states “Need an account?”*
2. *If you need to create an account, choose a unique username and password. By default, the password requires at least 6 characters. This function could be changed to address new password requirements.*

## *Classes*

### *Create a New Class*

1. *Once logged in, click on the link at the top labeled “Classes”. This will enable you to create a new class of students.*

**

1. *Click on “+ Add Class”.*

**

1. *Enter a class name and its description. The class name must be unique.*
2. *Click “Add Class” to add the class, otherwise click “Cancel” or outside of the modal to cancel adding the class.*



## *Reports*

1. *To access the reporting feature, from the Schedule module, click on “Generate Report” near the top right of the page.*

**

1. *By default, all events are generated and displayed.*