**D424 – Software Engineering**

**Task 3**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Academic Tracking Application |
| **Student Name:** | Blake Geard |

**Table of Contents**

1. **Introduction**
   * Purpose of the App…............................. Page \_2\_\_
2. **Application Design and Testing**
   * Class Design ............................................ Page \_4\_\_
   * UI Design ................................................. Page \_5\_\_
3. **Test Plan**
   * Introduction ............................................. Page \_6\_\_
   * Purpose .................................................... Page \_6\_\_
   * Overview .................................................. Page \_7\_\_
   * Items ........................................................ Page \_7\_\_
   * Features ................................................... Page \_7\_\_
   * Deliverables .............................................. Page \_8\_\_
   * Tasks ....................................................... Page \_8\_\_
   * Needs ...................................................... Page \_9\_\_
   * Pass/Fail Criteria ..................................... Page \_9\_\_
   * Procedures .............................................. Page \_9\_\_
   * Results .................................................... Page \_9\_\_
4. **User Guide for Maintenance Purposes**
   * Setup and Installation ................................ Page \_10\_\_
   * Testing the Application ................................ Page \_11\_\_
   * Troubleshooting ......................................... Page \_11\_\_
   * Maintenance and Updates ........................... Page \_11\_\_
5. **User Guide for Running the Application**
   * Overview of Features .................................. Page \_16\_\_
   * Navigating the Home Screen ......................... Page \_17\_\_
   * Adding and Managing Courses ...................... Page \_20\_\_
   * Viewing Progress and Notifications ............... Page \_21\_\_

# Application Design and Testing

## Class Design

A screenshot of a computer

Description automatically generated

The diagram above represents the database structure and relationships for the academic tracking application built using .NET MAUI. The design captures the core entities in the application, such as Terms, Courses, Assessments, and Users, along with their relationships. Each entity corresponds to a class in the application, with fields representing the attributes of the class. Inheritance from the BaseModel class, which is used for polymorphism in the code, is reflected by including shared fields (e.g., Id) in all tables. The relationships between entities (e.g., one Term can have many Courses) are modeled with foreign keys. This structure ensures that the application is scalable and adheres to good database design principles.

## UI Design

The following section showcases the low-fidelity wireframe designs for the Academic Tracking Application. These wireframes represent the initial conceptualization of the app’s user interface (UI) and layout. They outline the placement of key elements, such as navigation menus, progress dashboards, course management tools, and notification areas, without focusing on visual aesthetics like colors or typography.

The purpose of these wireframes is to provide a clear and functional structure for the application, enabling feedback on usability and layout before progressing to high-fidelity designs. These early designs serve as a foundation for iterative improvements, ensuring the final product meets user needs effectively.

A screenshot of a login form

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Test Plan**

# 

## Introduction

### Purpose

The purpose of this test plan is to outline the testing methods used for verifying the functionality of the AddAsync method in the DatabaseService class. The tests focus on ensuring that the method behaves as expected when provided with valid and invalid data. The tests check that the system correctly handles the addition of courses and that it throws the appropriate exception when a required field, like the Title, is empty. Both tests yielded the expected results, confirming that the DatabaseService class is functioning as intended. vb

### Overview

## These unit tests are part of a larger effort to verify the core business logic of adding a course in the DatabaseService class. This logic is critical to the application as it ensures that courses can be added correctly and that invalid input (such as an empty title) is rejected. Similar testing strategies may be used for other service methods within the application. These tests focused on validating two key aspects: successful addition of a course and proper error handling for invalid input.

## Tested Functions:

## AddAsync method in DatabaseService class for adding a course with valid data.

## Exception handling in AddAsync when the course Title is empty.

## Test Process:

## Test 1: Validates that a course with valid data can be successfully added.

## Test 2: Ensures that the system throws an ArgumentException with the correct message when the course Title is empty.

## Error Handling:

## Errors were handled by verifying that the correct exception (ArgumentException) is thrown with the correct message when invalid data is provided.

## Test Plan

### Items

### DatabaseService Class: To be tested for both valid and invalid data handling.

### Course Object: Instances of the Course class are created with various inputs to test both normal and edge cases.

### Features

* AddAsync Method: Used in both tests to simulate adding a course.
* Argument Exception Handling: Ensures proper validation when the course Title is empty.

### Deliverables

### Test results documentation.

### Updated codebase with tests in place to verify both valid and invalid course additions.

### A possible refactor of the AddAsync method to ensure better error handling if needed.

### Tasks

### Create unit tests for AddAsync method in DatabaseService.

### Test 1: Add a course with valid data.

### Test 2: Add a course with an empty title and check that an exception is thrown.

### Run the tests and verify their outcomes.

### Document any errors found during testing and adjust the implementation or tests accordingly.

### Ensure all tests pass successfully.

### Needs

### Visual Studio: To write and run tests.

### .NET SDK: Ensure the correct version is installed for compatibility with NUnit.

### NUnit Framework: For unit testing.

### Pass/Fail Criteria

## Test 1: Pass if a course with valid data is added successfully and returns the expected result (1).

## Test 2: Pass if an ArgumentException is thrown with the expected message "Course title cannot be empty." when an empty title is provided.

## Remediation if Failed:

## If Test 1 fails, verify that the AddAsync method returns the expected result and check that the logic for adding the course is correct.

## If Test 2 fails, ensure that the AddAsync method properly handles the empty Title case and throws the appropriate exception with the correct message.

A computer screen with text on it

Description automatically generated

## A computer screen shot of a program Description automatically generated

## Procedures

**Set Up:**

The test suite was set up in Visual Studio with the NUnit framework.

The DatabaseService instance was initialized for each test method using the SetUp attribute.

**Test Execution:**

The tests were run using Visual Studio's test runner and via command-line interface.

Assertions were made for both valid and invalid inputs.

**Iteration:**

The tests were iterated multiple times to ensure that they consistently pass when the expected conditions are met.

**Pass/Fail Result:**

Each test was marked as pass/fail based on whether the expected behavior was observed (valid course addition or correct exception)..

## Results

Test 1:Passed successfully, confirming that the course was added and the method returned the expected result.

Test 2: Passed as expected, confirming that an ArgumentException was thrown when the course title was empty.

Since we got both of the expected results, no changes were made after completing the tests.

A screenshot of a computer

Description automatically generated

**User Guide for Maintenance Purposes**

## Introduction

## This guide provides instructions for developers or system administrators to set up, configure, and maintain the application. It includes steps for running the application on an Android emulator, building the application, and troubleshooting.

## Setup and Installation

## Prerequisites:

## Operating System: Windows/Linux/MacOS

## Development Tools: Visual Studio 2022 or later (or another compatible IDE)

## .NET SDK: Version 8.0 (or the version used for the project)

## Android Emulator/Simulator (for testing mobile functionality)

Step 1: Clone the Repository

Clone the repository from GitLab or GitHub:

* git clone https://gitlab.com/wgu-gitlab-environment/student-repos/bgeard/d424-software-engineering-capstone/-/commits/working?ref\_type=heads

Navigate into the project folder:

* cd mobile

Step 2: Running the Application on an Android Emulator

To run the application on an Android emulator, follow these steps:

Set up Android Emulator:

* Ensure you have Android Studio installed (or Visual Studio with the Mobile workload).
* Open Android Studio and navigate to AVD Manager (Android Virtual Device Manager).
* Create a new virtual device (e.g., Pixel 4) with the desired Android version.
* Start the emulator once it's created.

Run the Application:

* Open the solution in Visual Studio (or another preferred IDE).
* Ensure the Android emulator is running.
* In Visual Studio, select the Android emulator as the target device from the top toolbar.
* Press Run or use the following command:
* dotnet maui
* The app should launch on the Android emulator.

Step 3: Testing the Application

To ensure everything is running correctly:

1. Run unit tests to check for application stability:

* dotnet test

1. Manually test the core features (e.g., login, adding courses, generating reports) by interacting with the app on the Android emulator.

Step 4: Troubleshooting

Common issues and their fixes:

* App not launching on Android emulator: Ensure the emulator is running and configured correctly. If the app doesn't launch, try restarting the emulator and rebuilding the solution.
* Unit tests failing: Review test results and address any issues with the code or configurations.
* Slow emulator performance: If the emulator is running slowly, consider using hardware acceleration or a more lightweight virtual device.

Step 5: Maintenance and Updates

Updating Dependencies:

* If needed, update project dependencies:
* dotnet restore

Handling Backups:

* Ensure regular backups are taken for any critical application data, especially if using a local or embedded database.

**Conclusion**

This guide provides all the necessary steps for setting up, running, and maintaining the application. Developers should follow these instructions for smooth deployment, testing, and ongoing maintenance. Running the app on an Android emulator will help simulate the mobile experience during development.

**User Guide for Running the Application**

**Introduction**

Welcome to the Academic Tracking Application. This guide provides a comprehensive walkthrough of the application's features, enabling you to navigate, manage, and track your academic terms, courses, and assessments with ease.

**Login Page**

**How to Log In**

1. Enter your username in the respective field.
2. Enter your password.
3. Click the "Login" button to access your account.
4. Upon successful login, you will be redirected to the Term List Page, where you can manage your academic terms.

A screenshot of a login screen

Description automatically generated

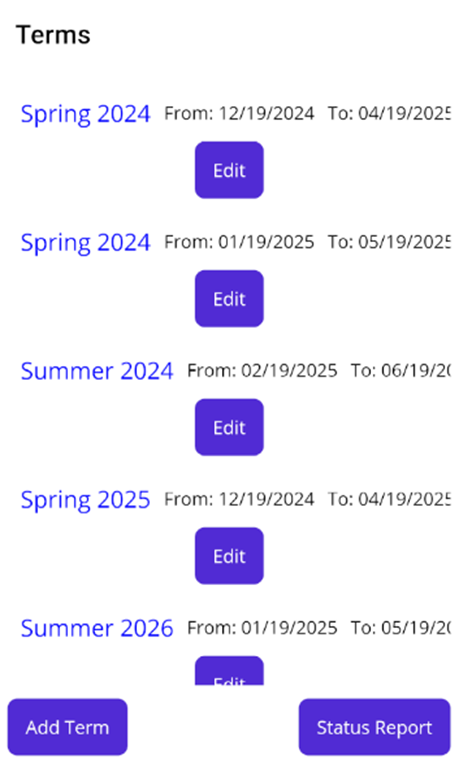
**Term List Page**

**Overview**

The Term List Page displays all the academic terms you have created.

**Key Features:**

* View Term Details: Click on any term to access its detailed view, which includes the list of associated courses.
* Edit Term: To modify a term, click the Edit button located below the term's name.
* Add New Term: Click the Add Term button at the bottom of the page to create a new academic term.
* View Status Report: Click the Status Report button to view your courses categorized by their status (Not Started, In Progress, Completed).



**Add Term Page**

**How to Add a New Term**

1. Enter the Title of the term in the provided field.
2. Select the Start Date and End Date for the term.
3. Once the fields are filled, click Save to add the term, or click the Back button to return to the Term List without saving.

A screenshot of a cell phone

Description automatically generated

**Status Report Page**

**How to Generate a Status Report**

1. Select the Status: Click on the dropdown menu labeled "Select Status" and choose the status you wish to filter by (Not Started, In Progress, Completed).
2. Generate Report: Click the Show Courses button to display all courses with the selected status.
3. The system will show a list of courses matching your selected status.
4. Return to Term List: Click the Back button to return to the Term List page.

A screenshot of a course status report

Description automatically generated

**Edit Term Page**

**How to Edit an Existing Term**

1. Enter Term Information:

* Term Title: Update the title of the term as needed.
* Start Date: Modify the term's start date if necessary.
* End Date: Update the end date of the term if required.

1. Save Changes:

* After making your desired changes, click the Save button to save the updated term information.

1. Delete Term:

* If you wish to remove the term, click the Delete button. A confirmation prompt will appear to ensure you want to delete the term.

1. Return to Term List:

* Click the Back button to return to the Term List Page without saving any changes.

A screenshot of a phone

Description automatically generated

**Term Detail Page**

This page will display a list of all courses for the selected term.

Course Details: Click on any course to view its details.

Navigation:

* + Back Button: Takes you back to the Term List page.
  + Add Course: Click the Add Course button to add a new course to the term.
  + Search Courses: Click the Search Courses button to find courses by instructor name.

A screenshot of a computer

Description automatically generated

**Course View Page**

**Course Details and Management**

1. Editable Fields: All course information (e.g., title, dates, instructor) is editable. Update the details as necessary.
2. Enable Notifications: Check the box to receive a notification one day before your course starts.
3. Save Changes: Click the Save button to save any changes made to the course.
4. Cancel Changes: Click Cancel to discard any changes and return to the Term Detail page.
5. Delete Course: Click the Delete Course button to remove the course from the term.
6. Share Notes: Use the Share Notes button to share class notes via email or text.
7. Assessments: Click the Assessments button to view the assessment page for this course.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

**Assessments Page**

**Viewing Assessments**

1. On this page, select either Performance or Objective assessments to view the corresponding assessments for the selected course.
2. Clicking on an assessment will take you to the Assessment Detail Page.
3. Navigate Back: Use the Back button to return to the previous page.

A screenshot of a phone

Description automatically generated

**Assessment Detail Page**

**Managing Assessment Information**

1. View Assessment Details: The assessment title, start date, and end date will be displayed.
2. Enable Notifications: Check the box to receive notifications 24 hours before the assessment starts.
3. Save Assessment: Click the Save Assessment button to save or update the assessment information.
4. Delete Assessment: If necessary, click the Delete Assessment button to remove the assessment.
5. Navigate Back: Click Back to return to the previous page.

A screenshot of a phone

Description automatically generated

**Search Course Page**

**Searching for Courses by Instructor**

1. Enter Instructor Name: Type the instructor's name in the search field.
2. Search: Click the Search button to display all courses associated with that instructor.
3. Return: Click Back to return to the Term Detail page.

**Gitlab Repo**

<https://gitlab.com/wgu-gitlab-environment/student-repos/bgeard/d424-software-engineering-capstone/-/tree/working?ref_type=heads>

A screenshot of a computer

Description automatically generated

**Panopto Recording**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b64218ff-761c-459e-8cbf-b238006780b5>

**Sources**

No sources used.