

D424 – Software Engineering

Task 3



Capstone Proposal Project Name: SemesterSync

Student Name: David Ogden

Table of Contents

<i>Application Design and Testing</i>	4
Class Design	4
UI Design	6
<i>Unit Test Plan</i>	7
Introduction	7
Purpose	7
Overview	7
Test Plan	8
Items	8
Features.....	9
Deliverables	10
Tasks	10
Needs	10
Pass/Fail Criteria.....	11
Specifications	12
Results	13
<i>User Guide – Maintenance Perspective</i>	14
Introduction	14
Setup Guide	14
<i>User Guide – Customer Perspective</i>	15
Introduction	15
<i>GitLab Repository</i>	17

Panopto Video Link..... 17

Hosted Web App Link..... 17

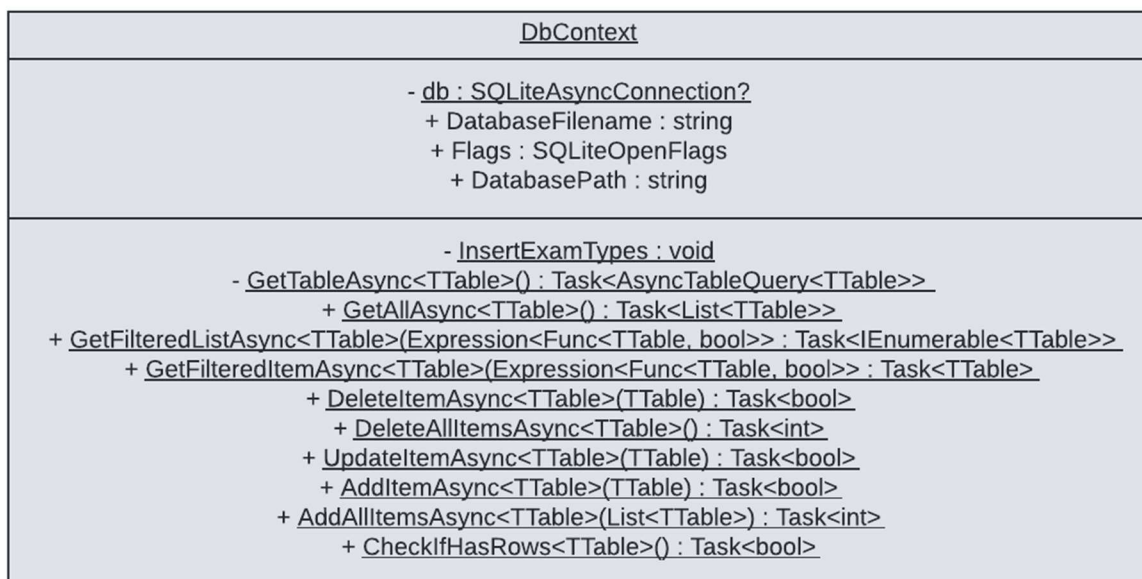
Application Design and Testing

Class Design

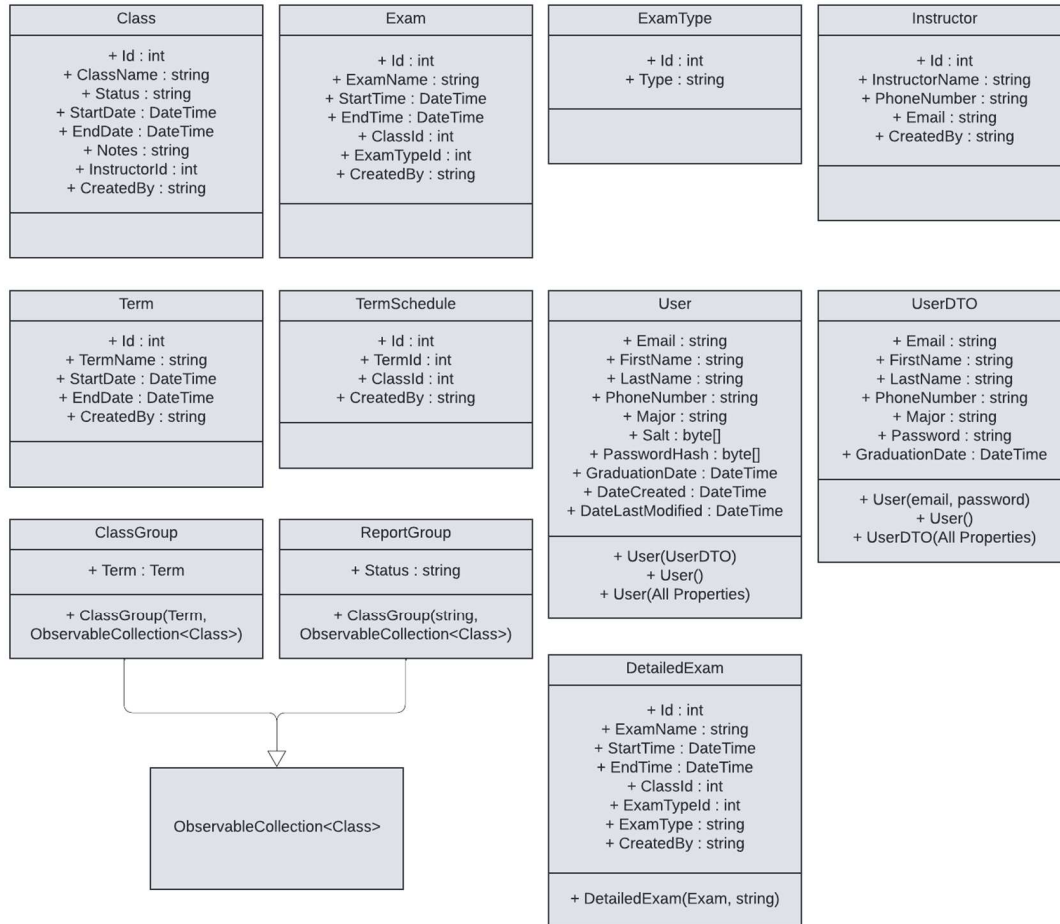
The unified modeling language (UML) class diagrams depicted below are the primary classes that makeup SemesterSync. Since C# is an object-oriented programming language, the UML diagrams accurately represent how all the objects and services are modeled.

The application is designed to track school semesters, classes, and exams for individual users. Therefore, the main classes utilized are the Term, Class, Term Schedule, Exam, User, and DbContext classes. Other classes, like ReportGroup, ClassGroup, and DetailedExam, handle group data. For Example, ClassGroup is constructed with a term and an observable collection of classes. This allows for associating classes with the term they are a part of.

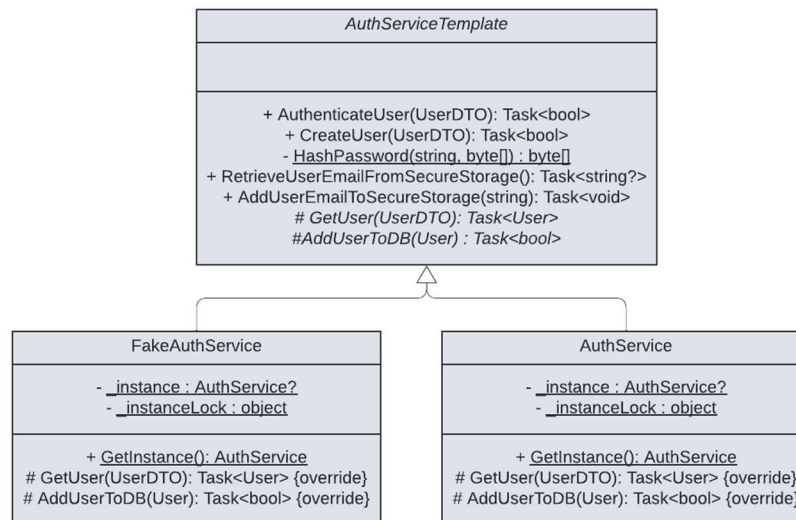
DataLibrary



ModelLibrary

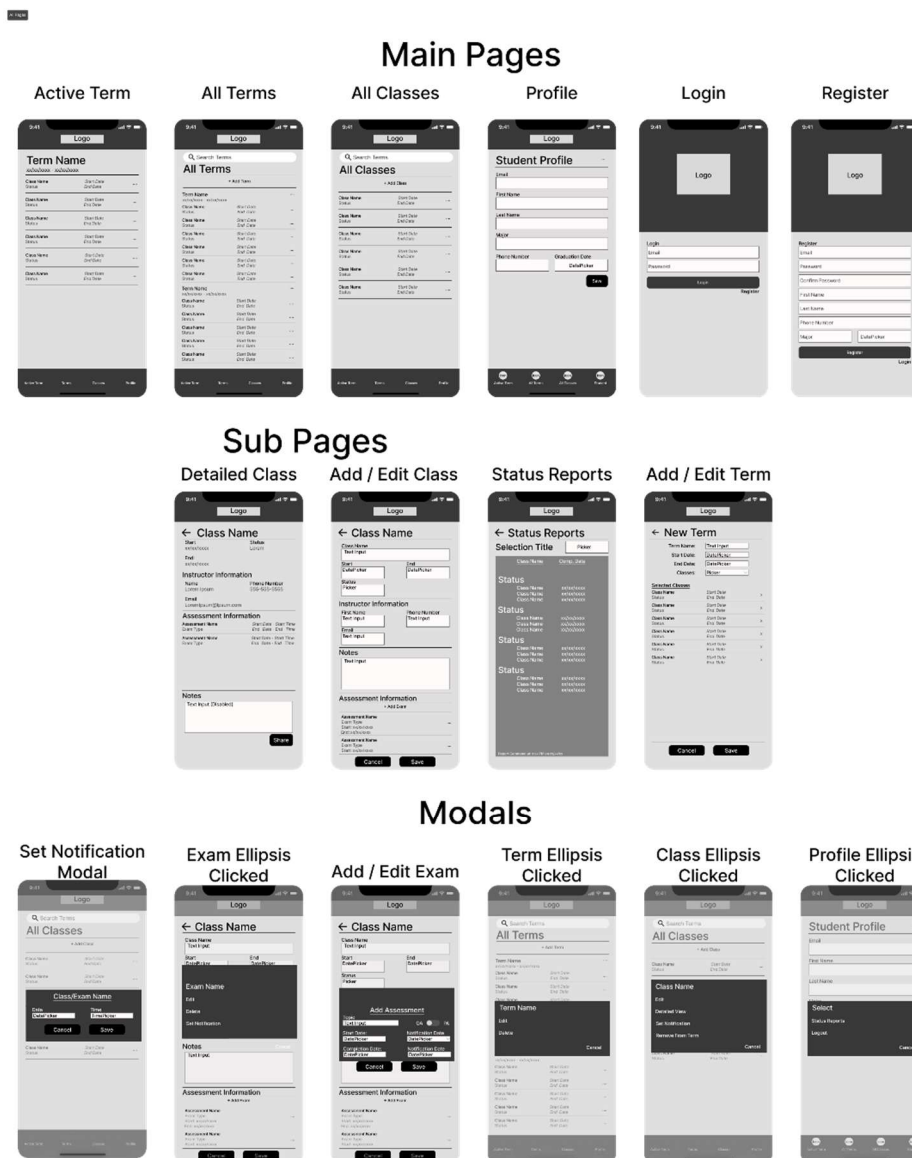


ServiceLibrary



UI Design

The user interface (UI) of SemesterSync provides a user-friendly interface that scales as the user adds more data. The entry point to SemesterSync is the login page. From there, users can navigate to the register page or login to see the entire application. Once logged in, users will be presented with the active term page. Users can navigate the primary pages using the bottom navigation bar. To navigate to sub-pages, users can click the ellipsis next to a class, term, or student profile and then select an option from the modal.



Unit Test Plan

Introduction

Purpose

The unit test plan aims to ensure that SemesterSync's codebase is reliable and will produce consistent results, regardless of the user. By including a robust set of unit tests, SemesterSync can ensure that all code is working as expected and more easily track down the source of issues in the future.

Overview

The testing plan for SemesterSync was meant to validate as many functionalities as possible. Therefore, all functionalities that could be tested with unit tests were tested. The plan was designed this way to maximize the amount of test coverage in the codebase. This will improve regression testing in the future.

The tested SemesterSync components are user authentication, the database, and the business logic in the View Models that provide data for each page.

- Database: The database was tested by creating an in-memory copy of the production database. This allowed all database queries to be tested and the database to be used for various integration tests without affecting the production database. The tests involved providing or requesting different objects from the database, ensuring the queries would respond appropriately, regardless of the object used. The only error encountered was due to running unit tests in parallel. Because the tests were running on multiple threads and only one instance of the database existed, the in-memory database connection would be closed while a

different thread was trying to execute queries, causing exceptions. To fix this issue, the testing framework, xUnit, was configured not to run tests in parallel.

- **User Authentication:** The tests for the user authentication service covered the login and registration of users, input validation, and login/registration selector functionality. To test login and registration, mock data was passed to the corresponding method, and the output was tested to produce the expected result. Input validation was tested by passing valid and invalid mock data to the corresponding method. When errors were present, the code was refactored and decoupled, ensuring methods only had a single responsibility.
- **ViewModels:** The ViewModels tests covered various cases dependent on the underlying ViewModel or business logic. Common functionalities tested from the ViewModels include testing input validation methods and methods to check if data requires an update. For example, a class (or course) would require an update if the user changed the status from “Active” to “Passed.” Errors encountered during testing include methods being coupled too tightly and poor error handling. To address the issues, the code was refactored to decouple methods and ensure edge cases were accounted for.

Test Plan

Items

- **Develop Environment:** The .NET 8.0 SDK and .NET Multi-Platform App UI Development workload are required because SemesterSync is a .NET MAUI application.

- Database: The database requires no additional setup as it is an SQLite database coupled with the application. All unit tests arrange themselves to run independently of outside factors, meaning no extra setup is required.
- Test Framework: xUnit is the testing framework that is used. No extra setup is required to use xUnit as it is already a part of SemesterSync. xUnit provides all the assertion handling.
- Source Code: The only setup required for the source code is to clone it from the remote repository. All scripts and required plug-ins are already a part of SemesterSync.

Features

- User Authentication: Test user login and registration functionality.
- Database: Test the ability to create, read, update, and delete from all tables.
- ViewModels: Test business logic that will vary depending on the underlying ViewModel.

Deliverables

- Test Scripts: A robust set of test scripts written using the xUnit testing framework as part of the source code.
- Test Results: A report that shows the results of all tests.

Tasks

- Create the SemesterSync solution and project and set up the development environment.
- Create xUnit testing project and include it as part of the SemesterSync solution
- Create test scripts to meet the requirements of the test plan. Requirements include testing user authentication, the database, and the underlying business logic for the ViewModels.
- Execute the test scripts using xUnit and record the results.
- Analyze the results, making note of all failed tests.
- Update code from the test results as needed.

Needs

- Develop Environment: The .NET 8.0 SDK and .NET Multi-Platform App UI Development workload are required because SemesterSync is a .NET MAUI application.
- Database: The database requires no additional setup as it is an SQLite database coupled with the application. All unit tests arrange themselves to run independently of outside factors, meaning no extra setup is required.

- Test Framework: xUnit is the testing framework that is used. No extra setup is required to use xUnit as it is already a part of SemesterSync. xUnit provides all the assertion handling.
- Source Code: The only setup required for the source code is to clone it from the remote repository. All scripts and required plug-ins are already a part of SemesterSync.

Pass/Fail Criteria

- User Authentication:
 - Pass: The user is successfully authenticated with email and password.
 - Pass: The user is successfully registered with an email, password, and other personal information.
 - Fail: The user is not authenticated due to incorrect credentials.
 - Fail: The user is not registered due to missing input fields.
 - Fail: The user is not registered if the provided email is already in use.
- Database:
 - Pass: All queries return the expected values indicating a successful query.
 - Fail: The database queries do not work as intended, given valid inputs
- ViewModels:
 - Pass: The ViewModel is updated appropriately, and changes appear on the corresponding page.
 - Pass: The state within the ViewModel is accurate and changes as expected after an event.
 - Fail: Corresponding pages display incorrect data.

- Fail: ViewModel state values do not match the expected value.

Specifications

Show below is an example of test scripts. All test scripts follow the Arrange, Act, then Assert pattern. All test names follow the naming convention of TestedMethodName_Scenario_ExpectedBehavior.

```
public class LoginViewModelTest
{
    // ValidateInputs Method Tests
    [Fact]
    0 references
    public async Task ValidateInputs_AllPropsHaveData_RegisterSelectedTrue_ReturnsFalse()
    {
        LoginViewModel vm = new()
        {
            RegisterSelected = true,
            FirstName = "John",
            LastName = "Doe",
            Email = "johndoe@email.com",
            Password = "password",
            ConfirmPassword = "password",
            PhoneNumber = "1234567890",
            Major = "Computer Science"
        };

        bool inputsContainError = await vm.ValidateInputs();

        Assert.False(inputsContainError);
    }

    [Fact]
    0 references
    public async Task ValidateInputs_EmailPasswordHaveData_RegisterSelectedFalse_ReturnsFalse()
    {
        LoginViewModel vm = new()
        {
            RegisterSelected = false,
            Email = "johndoe@email.com",
            Password = "password"
        };

        bool inputsContainError = await vm.ValidateInputs();

        Assert.False(inputsContainError);
    }

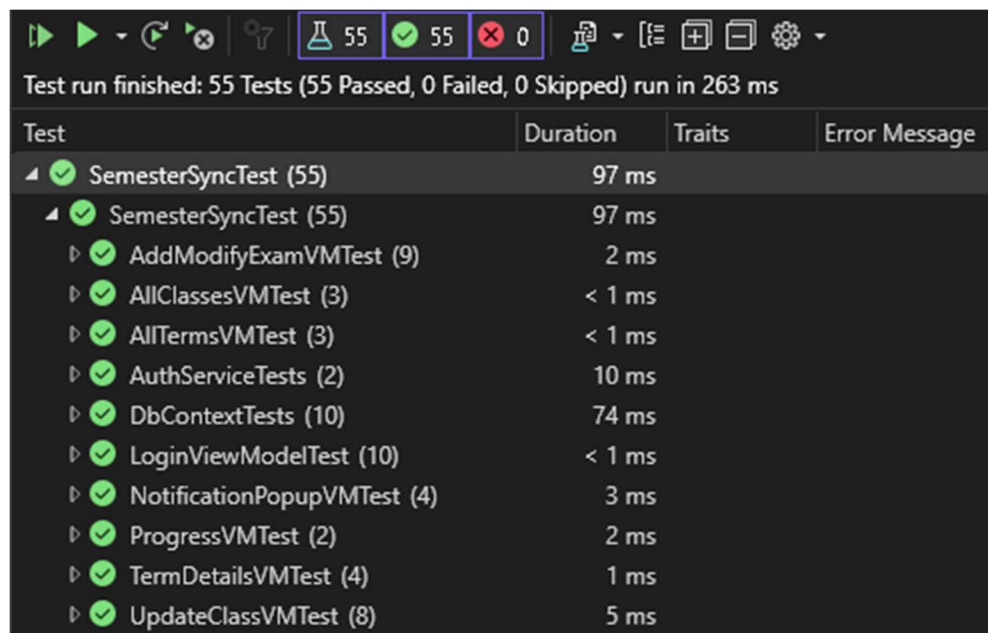
    [Fact]
    0 references
    public async Task ValidateInputs_PasswordsDontMatch_RegisterSelectedTrue_ReturnsTrue() ...

    [Fact]
    0 references
    public async Task ValidateInputs_EmailNull_RegisterSelectedFalse_ReturnsTrue() ...
}
```

Results

The test results for SemesterSync are as follows:

- User Authentication: All tests are passed when valid credentials are given to log in or register. The tested methods returned the appropriate value. All errors were handled as expected when given invalid credentials and returned appropriate values, resulting in passed tests.
- Database: Given that the test was arranged correctly, all tests passed. This validated all queries regardless of which table was being interacted with. Given incorrect data, the database methods returned the expected values, signifying an issue was present.
- ViewModels: All tests passed for the ViewModels despite being given valid or invalid data. Tests included verifying inputs, validating data updates, and various data transformations.



Test	Duration	Traits	Error Message
▲ ✓ SemesterSyncTest (55)	97 ms		
▲ ✓ SemesterSyncTest (55)	97 ms		
▶ ✓ AddModifyExamVMTest (9)	2 ms		
▶ ✓ AllClassesVMTest (3)	< 1 ms		
▶ ✓ AllTermsVMTest (3)	< 1 ms		
▶ ✓ AuthServiceTests (2)	10 ms		
▶ ✓ DbContextTests (10)	74 ms		
▶ ✓ LoginViewModelTest (10)	< 1 ms		
▶ ✓ NotificationPopupVMTest (4)	3 ms		
▶ ✓ ProgressVMTest (2)	2 ms		
▶ ✓ TermDetailsVMTest (4)	1 ms		
▶ ✓ UpdateClassVMTest (8)	5 ms		

User Guide – Maintenance Perspective

Introduction

Attached below is the developer setup guide. The purpose of this guide is to provide easily replicable steps for future developers to follow so they can maintain SemesterSync.

Prerequisites

- .NET 8.0 SDK installed
- Visual Studio 2022 or equivalent integrated development environment
- .NET Multi-platform App UI development workload installed
- Git installed

Setup Guide

1. Open Git Bash in the directory where the project code will reside in
2. Run the command: **git clone -b working https://gitlab.com/wgu-gitlab-environment/student-repos/davidogden.dev/d424-software-engineering-capstone.git** and wait for the command to finish
3. Run the command: **cd d424-software-engineering-capstone/**
4. Run the command: **start devenv .**
 - a. **** Notice the previous command has a period “.” on the end****
5. Once in Visual Studio 2022, open the Solution Explorer and double-click **SemesterSync.sln**
6. Click the “Test” tab in the top toolbar. Hover over “Configure Run Settings”. Click “Select Solution Wide runsettings File”. Navigate to the directory **d424-software-engineering-capstone/SemesterSyncTest** and select the file **xUnit.RunSettings**

7. Click the “Test” tab in the top toolbar. Select “Run All Tests”. All tests should pass; otherwise, a prerequisite was not met, or a previous step was skipped
8. Ensure “SemesterSync” is selected as the startup project
9. If a device for the Android Emulator is already downloaded, launch the emulator, and the setup is now complete. Otherwise, proceed to the next step
10. Select the “Tools” tab in the toolbar. Hover over “Android,” then select “Android Device Manager.”
11. Once in the device manager, select “+ New” and then select “Create.”
12. Back on the main page of the device manager, select “Start.” Once the emulator has loaded, the setup is completed

User Guide – Customer Perspective

Introduction

Attached below is the customer setup guide. The purpose of this guide is to provide easily replicable steps for customers to get started using SemesterSync. This guide is meant to be followed in the order in which items are presented.

Prerequisites

- Android Mobile Device

Setup Guide

Login / Register

1. From a personal Android device, navigate to the Google Play Store, search for SemesterSync, and download the application

2. Once downloaded, open the app to be shown the login page. Select the “Register” button
3. Fill out all fields in the registration form and click the button labeled “Register.”

Create New Class

1. From the All Classes page, select “+ Add Class” at the top of the page and wait for the Add Class page to appear
2. Fill out all fields for the new class
3. If the class has no exams, skip to step 9. Otherwise, click the “+ Add Exam” button
4. Fill out the form for the new exam and click save
5. If the new class form is complete, select “Save.”

Create New Term

1. From the All Terms page, select “+ Add Term” at the top of the page and wait for the Add Term page to appear
2. Fill out all fields for the new term. In the class selector, choose the class that was just created. The class should now appear below the selector
3. If the new Term form is complete, select “Save.”

Edit User Information

1. Navigate to the Profile page
2. Edit the information that needs to be updated, then select “Save.”

View Status Reports

1. Navigate to the Profile page
2. Select the ellipsis “...” button at the top right of the page
3. Select Status Reports
4. Use the selector to view reports based on all terms or a specific term

GitLab Repository

GitLab Repository Link:

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

Panopto Video Link

Name: Ogden, David – Capstone Demo

Link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8faaff82-9ffe-4444-8de3-b1cb006368bc>

Hosted Web App Link

Not applicable. SemesterSync is a mobile application that will be available on the Google Play Store.