Title

Home Assignment 2 - University Management App

Description

This is the second bonus point homework assignment, covering the creation, extending and testing of a University Management App.

Scenario

You have been hired by a university to develop a University Management App that helps in academic administration. The university is experiencing difficulties in managing their courses and they need a simple but effective solution. The application should allow students to enroll in and drop subjects and teachers to create and manage subjects, while also providing data persistence through a JSON file. You will also have to conduct testing of the code. Below you can find system requirements and instructions you should follow.

Non-Functional Requirements

- The application must be developed using Avalonia and following the MVVM architecture.
- The application must implement Student and Teacher roles with access control (each can only access their functionality).
- Ensure simple and intuitive user interface.

Functional Requirements

System:

- An entry point is a login screen that lets the user login as a certain role with a simple validation logic (comparing provided login and password of student/teacher to valid ones and picking the right account).
- The application should provide system's data persistence using a JSON file. The JSON file should be loaded when starting the application and saved on closing the application or using a save and load button.

Student:

- View available subjects
- View enrolled subjects
- View subject's details (Name, teacher, description etc.)
- Enroll in available subjects
- Drop from enrolled subjects

Teacher:

- View subjects they teach
- Create a new subjects
- Edit subject's details
- Delete their subject

Data Models

The data models shown below are a loose suggestion – not a requirement.

Student:

Field Name	Data Type	Description
id	int	Unique identifier for the student.
name	string	Full name of the student.
username	string	Username for authentication.
password	string	Password for authentication.
enrolledSubjects	List<int>	List of subject IDs the student is enrolled in.

Teacher:

Field Name	Data Type	Description
id	int	Unique identifier for the teacher.
name	string	Full name of the teacher.
username	string	Username for authentication.
password	string	Password for authentication.
subjects	List<int>	List of subjects IDs the teacher has created.

Subject:

Field Name	Data Type	Description
id	int	Unique identifier for the subject.
name	string	Subject name.
description	string	Brief details about the subject.
teacherId	int	ID of the teacher who created the subject.
studentsEnrolled	List<int>	List of student IDs enrolled in the subject.

Use Cases

I. Student Enrollment in a Subject

- Actors: Student
- Preconditions: You are logged in as a Student.

Steps:

1. The student navigates to the "Available Subjects" page.
2. The student selects a subject and clicks "Enroll".
3. The system updates the student's list of enrolled subjects in "Enrolled Subjects" page.
4. The student receives confirmation of successful enrollment in a form of text or pop-up.

II. Student Dropping a Subject

- Actors: Student
- Preconditions: You are logged in as a Student and enrolled in at least one subject.

Steps:

1. The student navigates to the "Enrolled Subjects" page.
2. The student selects a subject and clicks "Drop".
3. The system removes the subject from the student's enrolled list.
4. The student receives confirmation of successful removal in a form of text or pop-up.
5. The dropped subject appears in "Available Subjects" page.

III. Teacher Creating a Subject

- Actors: Teacher
- Preconditions: You are logged in as a Teacher.

Steps:

1. The teacher navigates to "My Subjects" and selects "Create new subject".
2. The teacher enters subject details (name, description, etc.).
3. The teacher clicks "Save".
4. The system adds the subject to "My Subjects" for a teacher and "Available Subjects" for students.
5. The teacher receives confirmation of successful creation in a form of text or pop-up.

IV. Teacher Deleting a Subject

Actors: Teacher

Preconditions: You are logged in as a Teacher.

Steps:

1. The teacher navigates to "My Subjects".
2. The teacher selects a subject and clicks "Delete".
3. The system removes the subject from "My Subjects" for a teacher and "Available Subjects" for students.
4. The teacher receives confirmation of successful deletion in a form of text or pop-up.

Testing Requirements

A. Unit Testing

- You must implement at least three different unit tests using xUnit. These should focus on core business logic (e.g., enrollment rules, data persistence, access control). You may choose your test cases or adapt them from the Functional Testing list below.

B. Functional Testing

- You must conduct manual testing by verifying that all key functionalities work as expected. At a minimum, perform the following test cases and document your results in a .txt file or README.md:

Student Functionality Tests:

- Enrollment Test: Verify that a student can enroll in a subject and that it appears in their enrolled subjects list.
- Drop Subject Test: Verify that a student can drop a subject, and it reappears in the available subjects list.

Teacher Functionality Tests

- Create Subject Test: Verify that a teacher can create a subject and it appears in both "My Subjects" and "Available Subjects".
- Delete Subject Test: Verify that deleting a subject removes it from both "My Subjects" and "Available Subjects".

System-Level Tests

- Login Test: Ensure that login validation works for both Student and Teacher roles.
- Data Persistence Test: Verify that all changes are saved to the JSON file and persist after opening the application again.

C. User Interface Testing

- Headless Testing allows you to test UI-related logic, such as ViewModels or command bindings, in a separate, automated environment. User Interface Testing in Avalonia is done with a Avalonia.Headless.XUnit package and the process is explained here:

  https://docs.avaloniaui.net/docs/concepts/headless/headless-xunit

  Perform and describe UI Tests on Use Cases given above.

Grading

The bonus points will be awarded in accordance to the following requirements:

- For one bonus point, you must make an application that works without much setup and that covers most of the requirements. For one bonus point you also have to complete at least one test from each of sections: A), B) and C) of Testing Requirements.

- For two bonus points, you must accomplish what is required for one bonus point, as well as complete all of the Testing Requirements. Additionally, you must implement at least one of the following features:

- Implement a role of "System Admin" that is able to oversee the system and act as a Teacher or Student.
- Implement a search functionality for subjects - allow students and teachers to search the subject by name.
- Implement a "password hashing" as a security measure to logging into accounts.