**Final Requirements Document**

**Professor:** Michael Landreth

**Course:** CST-452

**Student:** Patrick Garcia

**Date:**  6/28/2022

**Title:** Final Requirements Document

The following are Updated requirements for this app.

* *R1*: Two-factor authentication will be used to ensure students' accounts are kept secure.
  + *Outcome*: Not Met
    - *Justification:* Did not have the time necessary to implement a two factor authentication system.
    - *Future Versions*: When time is longer a constraint, a Two-Factor Authentication system will be implemented. This can be easily accomplished using the built in tools of .NET. Particularly, .NET Identity, can be used to provide two factor authentication through either sms or email.
* *R2:* Inputted data will be cleansed to prevent any injection style attacks. Validation rules and C#’s string builders will be useful for this purpose.
  + *Outcome:* Met
    - *Justification:* The application utilizes C#’s *IDataErrorInfo* interface, as well as the *INotifyPropertyChanged* interface. These interfaces allows the app to raise events when the view models properties are changed, through user input involving TextBox controls. When a user enters text into a textbox field, the app raises a *PropertyChanged* event, and uses a *PropertyChangedEventHandler* to handle the event. The app then takes the model's changed property and compares it to the validation rules set through the *IDataErrorInfo* interface. If errors are found, a dock panel is created under the TexBox to display a custom error message. The threat of SQL injection attacks has been eliminated, due to the lack of a database. User Course data is simply stored within a JSON file stored locally within the project folder.
* *R3:* The application must boast quick page load times, and must populate student data using a database quickly. SQL database will be used for this purpose
  + *Outcome*:Partially Met
    - *Justification*: The application does boast quick page load times. This is largely due to the DataBinding capabilities of the Windows Presentation Framework (WPF). Using a Model View ViewModel (MVVM) approach, a view model is created and bound to a window’s UI elements. Whenever changes are made to the model, the UI elements are automatically refreshed with updated data. This results in fast page loading times, however the Requirement was not met using a SQL database. Instead, a simple JSON file is generated to store course information. Due to the simplicity of the data being saved, a SQL database was no longer necessary.
* *R4:* Using SASS, HTML and CSS, text contrast will be increased, colors will be used effectively to indicate actions, text will be properly spaced, and controls will be large and easily identifiable, all in compliance with Web Content Accessibility Guidelines (WCAG).
  + *Outcome:* Met
    - *Justification*: The application has successfully utilized CSS and html elements to provide a clear and accessible UI for the application. Controls have been made larger than surrounding elements to increase readability. Colors are used for proper contrast, mainly to decorate navigational elements that help users interact with the app. The color RED was used throughout to indicate errors in user input fields. Message boxes were also used to convey information pertaining to system processes to the user.
* *R5*: Users will be able gain all needed course information from a single screen.
  + *Outcome:* Met
    - *Justification:* The application has been streamlined, allowing for a single central dashboard where all relevant information pertaining to a course may be found. Users can do most operations of the application from this screen. Course information, and a full list of assignments can be found, as well as an assignment details page located within the window itself. This page displays assignments in a list, and when selected, will display the details of the selected assignment. Users can also create new Courses, and Create, Delete, and Mark assignments as Complete from this page.
* *R6*: App should be accessible without an internet connection.
  + *Outcome:* Met
    - *Justification*: The application does not require any internet connection for operation. It does not need to interact with a database. It is entirely self contained, and can run on its own.
* *R7:* Application routes should be protected, to ensure users only have access to pages that pertain to them.
  + *Outcome:* Not Met
    - *Justification*: The application does not utilize any authentication capabilities. There are no *Log in* functions. The app also does not have any routes to protect. It is a single dashboard where students can find course and assignment information. There are no admin functions that must be protected. Future iterations may include authorization. In these iterations, a user settings page will be added that will be protected from unauthorized users. Features will also be implemented to lock Course information as the user desires. Requiring a password or key to access.

The following are Updated Use-case for this app.

* UC1: Start new Course
  + *Actor:* User of the application, most likely a student.
    - The User has started a new course. They wish to organize the course and its associated assignments within the Organize app. The user opens *Organized.exe*. The user selects the *Add Course* button in the top right corner of their dashboard. The user provides valid data to the *AddCourse* dialog. The Course is created, and added to the list. The User selects the new course, and uses the *Add Assignment* button in the assignmentDetailsPage to add any relevant assignment details for the course.
* UC2: Finish a Course
  + *Actor*: User of the application.
    - The User has finished a course. They wish to delete the course from the Organized app to clear space for future courses. The user opens the application and finds the desired course in the course list. The User selects the course, and then selects *Delete Course* button in the assignmentDetailsPage.
* UC3: Finish a Course without delete
  + *Actor*: User of the application.
    - The User has finished a course. They do not wish to delete the course from the Organized app. The user opens the application and finds the desired course in the course list. The User selects the course, and then selects *Complete Course* button in the assignmentDetailsPage
* UC4: Finish an Assignment without delete
  + *Actor*: User of the application.
    - The User has finished an assignment. They do not wish to delete the assignment from the Organized app. The user opens the application and finds the desired course in the course list. The User selects the course, and then selects the desired assignment. The User then clicks the *Complete Assignment* button in top-right of assignmentDetailsPage.
* UC5: Finish an assignment
  + *Actor*: User of the application.
    - The User has finished an assignment. They wish to delete the assignment from the Organized app to clear space for future assignments. The user opens the application and finds the desired course in the course list. The User selects the course, and then selects the desired assignment. The User then click *Delete Assignment* button in top-right of assignmentDetailsPage.