CMP1127M Programming and Data Structures: Assessment 2

Table of Contents

Application Description

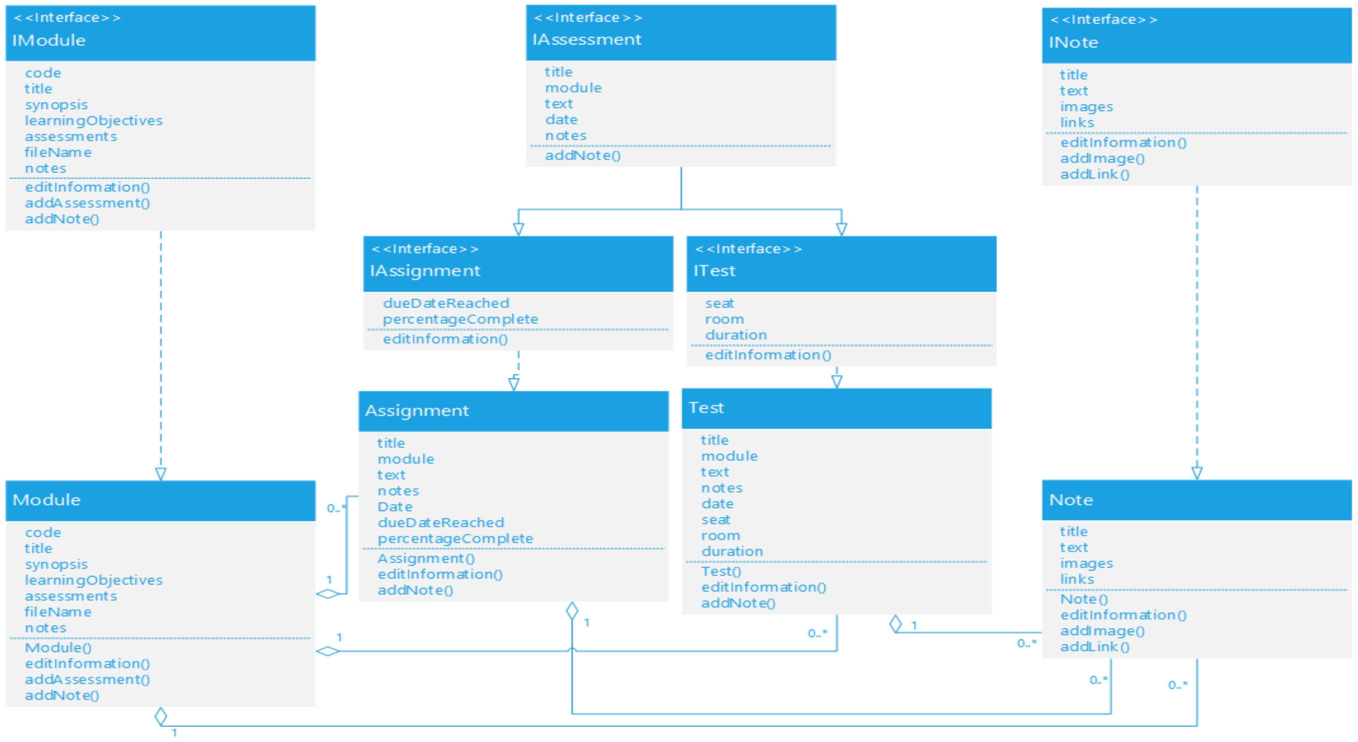
For this Assessment, I needed to design a desktop application named ‘modnote’ that functions as a ‘Module note taker’. Allowing the user to import text files with module information and add notes, which could contain text, images and links; these notes would then be added to the module, which can have many notes. The modules would also have assessments with due dates that should be shown to the user. The application I have created allows the user to create modules manually, as well as importing them from a text file; and also lets the user add their own assessments, which can either be an assignment (e.g. coursework), or a test. Notes may be created for each assessment too, allowing modules to have separate notes from its assessments notes.

The user can delete modules, assessments and notes from the application; as well as edit them, allowing the user full control over the stored data. The application shows each module in a list with a tree format, showing each module as a top-level node with its assessments shown as sub nodes; the user can then double click on any of these nodes to bring up its details and notes.

Assessments are also shown to the user in another list with a tree format, showing each assessment as a node, and these are shown in two different ways, the default way is a list of all the assessments with a due date within 2 weeks; the user can also change this to view all assessments, and sort the assessments according to due date (the amount of days left to complete). Assignments have a progress bar which allows the user to show how much they have completed, and tests have Room, seat and Duration textbox allowing the user to store have information about the test readily at hand. Notes allow the user to store text and also create links to website, or file locations and are opened by the appropriate program when clicked; notes also have images which can be added and then viewed in their full size in another window upon being clicked.

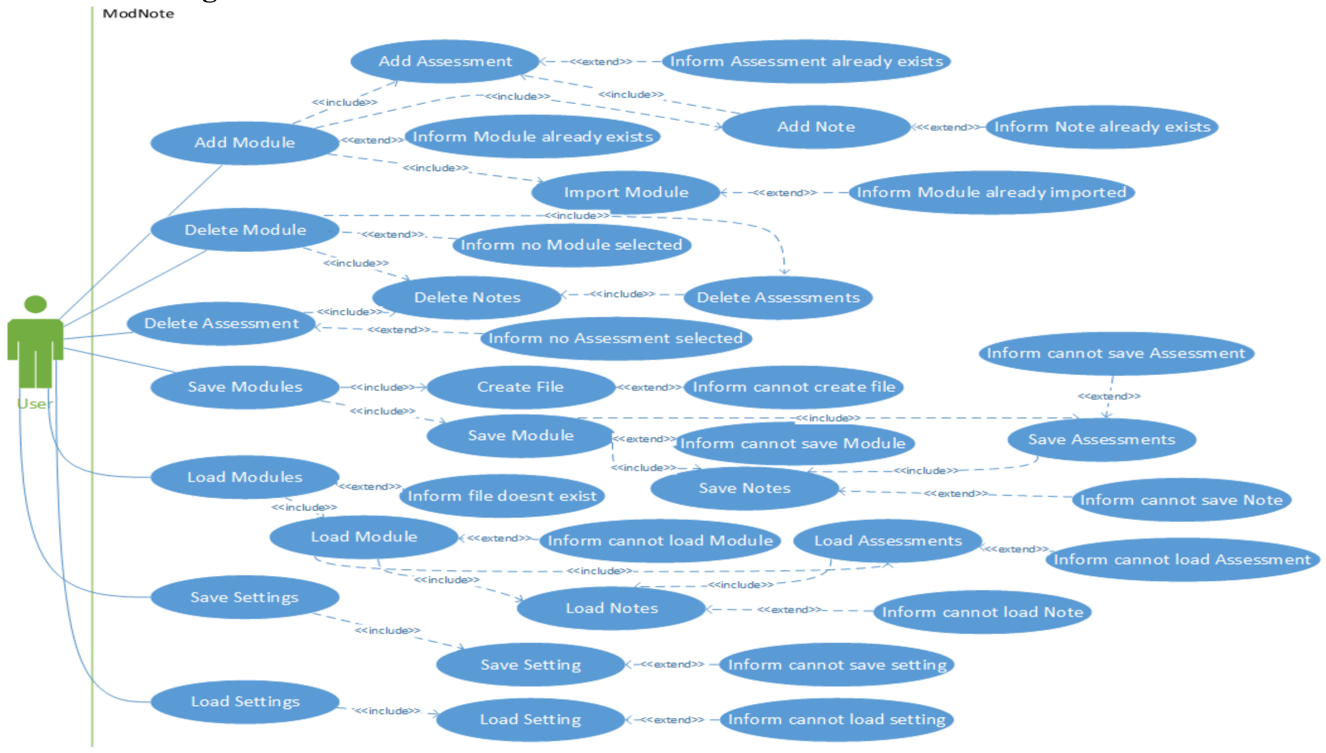
The application has settings which allow the user to select whether to auto load modules on starting the application, and choose what modules to load; on exiting the application the settings and modules are serialized into separate files xml files allowing them to be deserialized to the exact same state upon opening the application again. These saved settings and modules can also be loaded into the application at will by the user.

Class Diagrams

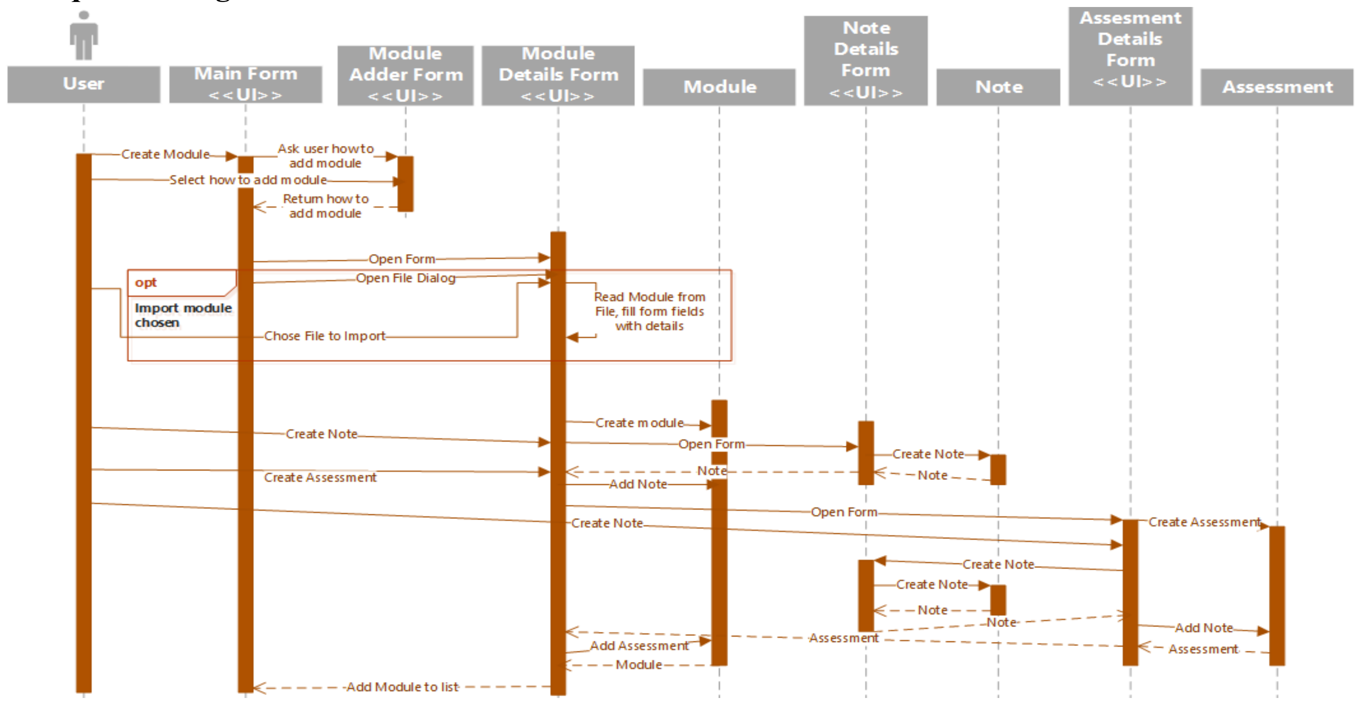


As seen in the class diagram above, I chose to create interfaces for each class, with the IAssignment and ITest interface inheriting from the base IAssessment interface. I chose to do this because using interfaces allows 'easier maintainability, makes your code base more scalable and makes code reuse much more accessible because implementation is separated from the interface. Interfaces add a plug and play like architecture' (Mccutchen, 2010) which I thought would be useful in this program allowing adding more classes (e.g. an Exam class) later to be easier and not break any existing code structure.

Use-Case Diagram



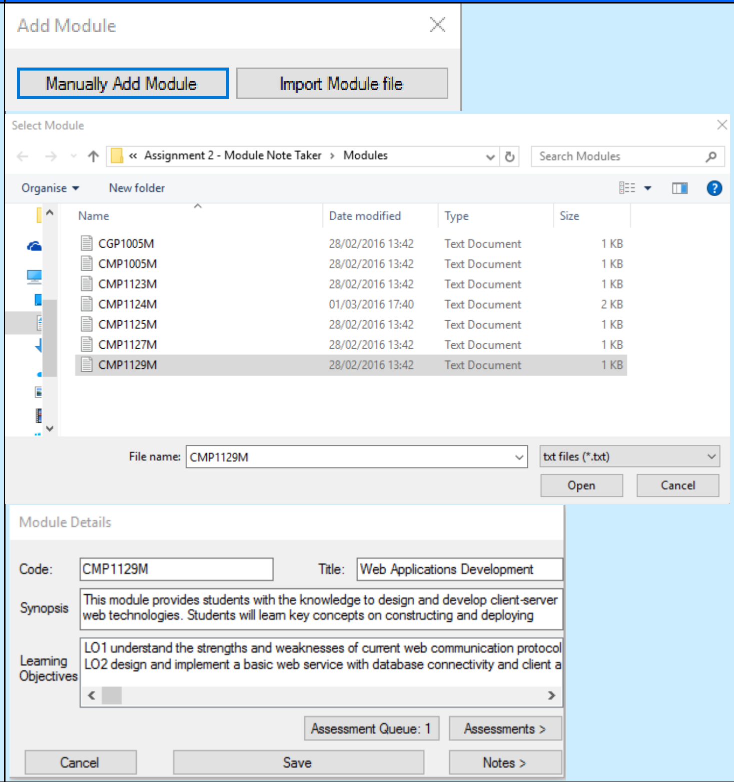
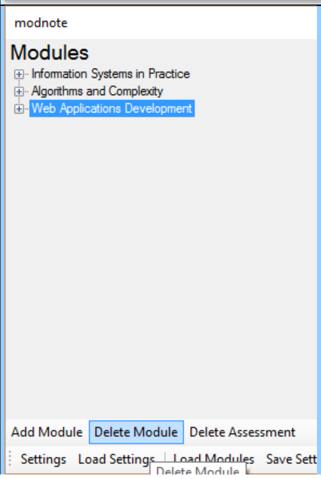
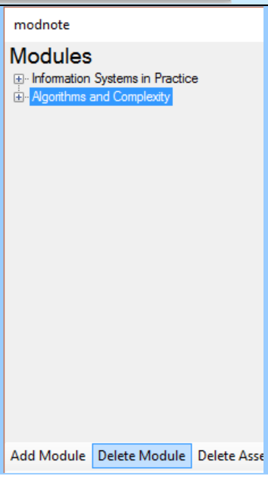
Sequence Diagram

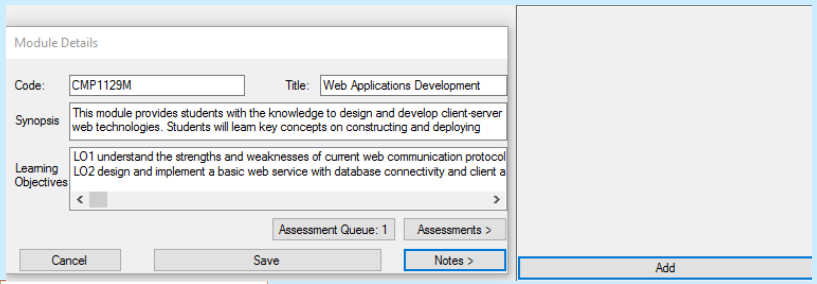
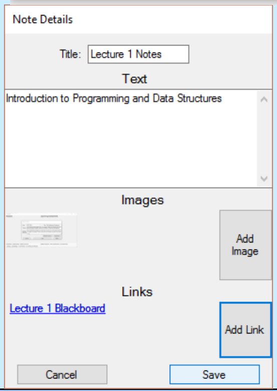
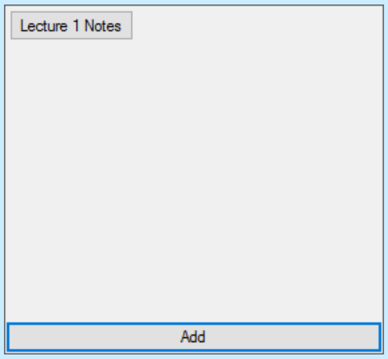
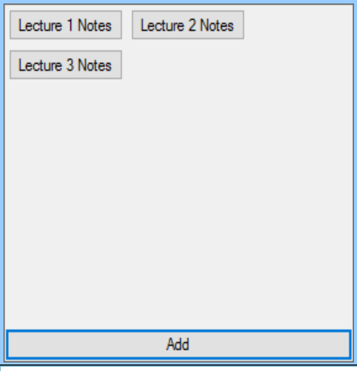
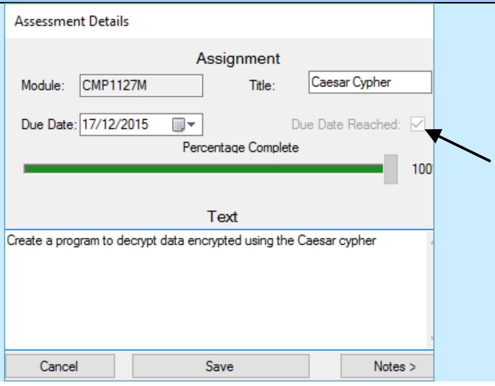


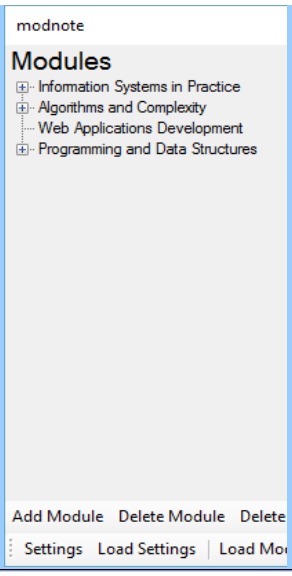
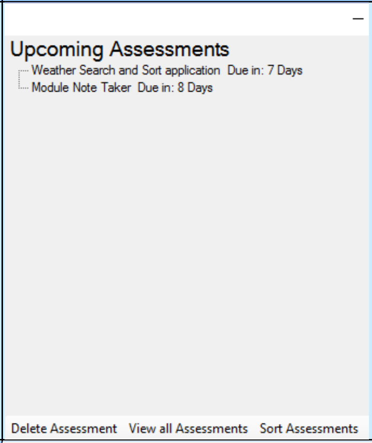
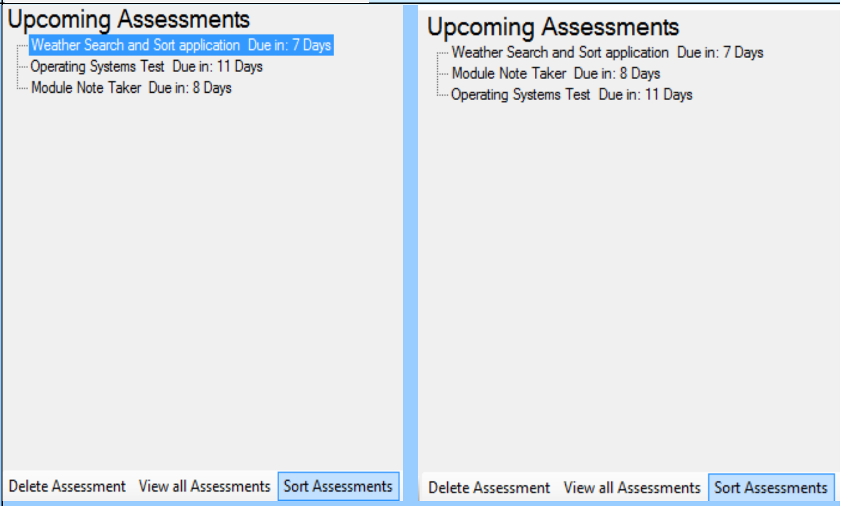
CMP1127M Programming and Data Structures: Assessment 2

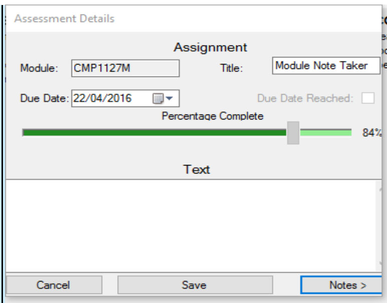
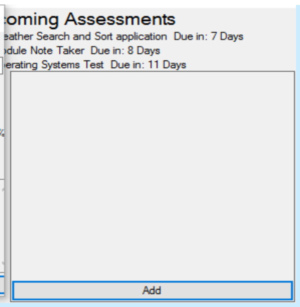
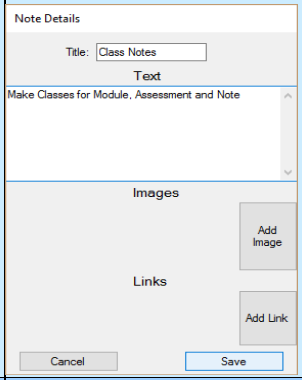
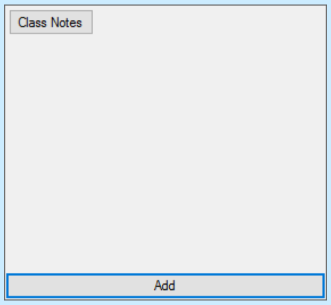
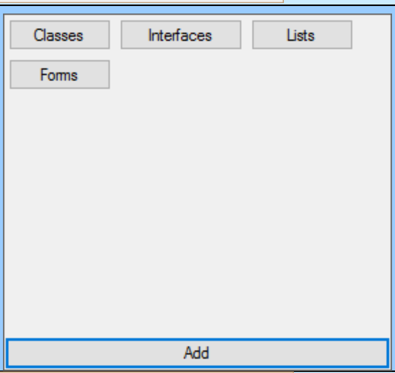
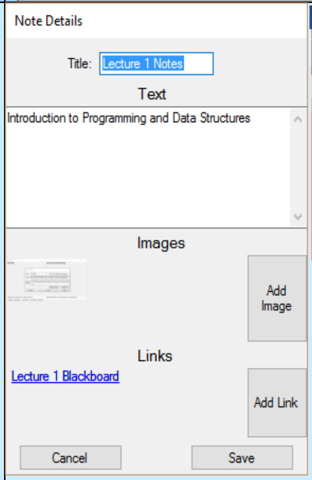
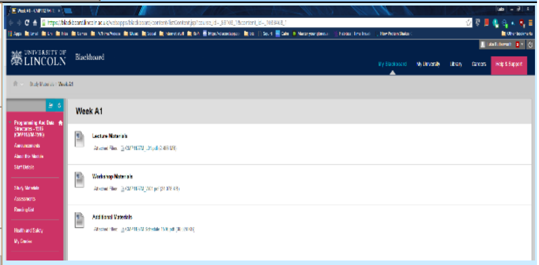
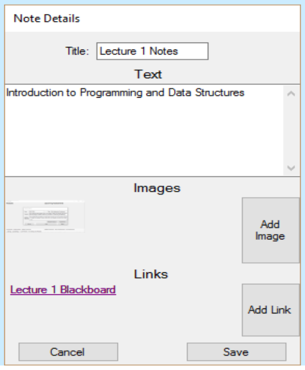
Testing

Black Box testing

Requirement	Pass/Fail	Screenshot
Import Module information from file	Pass	 Add Module Manually Add Module · Import Module file Select Module « Assignment 2 - Module Note Taker › Modules Name · Date modified · Type · Size CGP1005M · 28/02/2016 13:42 · Text Document · 1 KB CMP1005M · 28/02/2016 13:42 · Text Document · 1 KB CMP1123M · 28/02/2016 13:42 · Text Document · 1 KB CMP1124M · 01/03/2016 17:40 · Text Document · 2 KB CMP1125M · 28/02/2016 13:42 · Text Document · 1 KB CMP1127M · 28/02/2016 13:42 · Text Document · 1 KB CMP1129M · 28/02/2016 13:42 · Text Document · 1 KB File name: CMP1129M · txt files (*.txt) Module Details Code: CMP1129M · Title: Web Applications Development Synopsis: This module provides students with the knowledge to design and develop client-server web technologies. Students will learn key concepts on constructing and deploying Learning Objectives: LO1 understand the strengths and weaknesses of current web communication protocol LO2 design and implement a basic web service with database connectivity and client a Assessment Queue: 1 · Assessments > · Cancel · Save · Notes >
Modules can be deleted	Pass	 modnote Modules Information Systems in Practice Algorithms and Complexity Web Applications Development Add Module · Delete Module · Delete Assessment Settings · Load Settings · Load Modules · Save Sett Delete Module  modnote Modules Information Systems in Practice Algorithms and Complexity Add Module · Delete Module · Delete Asse

Create Note for Module	Pass	 Module Details Code: CMP1129M Title: Web Applications Development Synopsis: This module provides students with the knowledge to design and develop client-server web technologies. Students will learn key concepts on constructing and deploying Learning Objectives: LO1 understand the strengths and weaknesses of current web communication protocol LO2 design and implement a basic web service with database connectivity and client a Assessment Queue: 1 / Assessments > / Cancel / Save / Notes > / Add  Note Details Title: Lecture 1 Notes Text: Introduction to Programming and Data Structures Images / Add Image Links: Lecture 1 Blackboard / Add Link Cancel / Save  Lecture 1 Notes Add
Module can have multiple Notes	Pass	 Lecture 1 Notes / Lecture 2 Notes / Lecture 3 Notes Add
Assignments should show whether their due date has been reached	Pass	 Assessment Details Assignment Module: CMP1127M Title: Caesar Cypher Due Date: 17/12/2015 Due Date Reached: ✓ Percentage Complete: 100 Text: Create a program to decrypt data encrypted using the Caesar cypher Cancel / Save / Notes >

Modules displayed all together	Pass	 modnote Modules Information Systems in Practice Algorithms and Complexity Web Applications Development Programming and Data Structures
Assessments shown according to due date	Pass	 Upcoming Assessments Weather Search and Sort application Due in: 7 Days Module Note Taker Due in: 8 Days
Assessments can be sorted according to due date	Pass	 Upcoming Assessments Weather Search and Sort application Due in: 7 Days Operating Systems Test Due in: 11 Days Module Note Taker Due in: 8 Days Upcoming Assessments Weather Search and Sort application Due in: 7 Days Module Note Taker Due in: 8 Days Operating Systems Test Due in: 11 Days

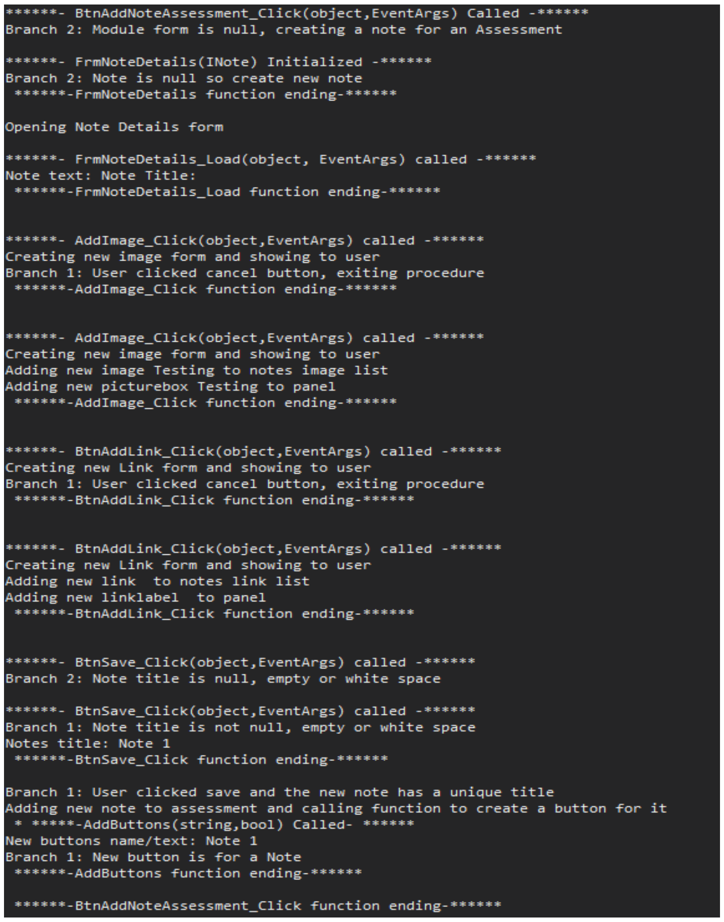
Create Note for Assessment	Pass	 Assessment Details Assignment Module: CMP1127M Title: Module Note Taker Due Date: 22/04/2016 Due Date Reached: Percentage Complete 84% Text Cancel Save Notes >  oming Assessments eather Search and Sort application Due in: 7 Days dule Note Taker Due in: 8 Days erating Systems Test Due in: 11 Days Add  Note Details Title: Class Notes Text Make Classes for Module, Assessment and Note Images Add Image Links Add Link Cancel Save  Class Notes Add
Assessments can have multiple Notes	Pass	 Classes Interfaces Lists Forms Add
Clicking a link in a Note opens the link in browser/File explorer	Pass	 Note Details Title: Lecture 1 Notes Text Introduction to Programming and Data Structures Images Add Image Links Lecture 1 Blackboard Add Link Cancel Save  UNIVERSITY OF LINCOLN Blackboard Week A1 Lecture Materials Workshop Materials Additional Materials  Note Details Title: Lecture 1 Notes Text Introduction to Programming and Data Structures Images Add Image Links Lecture 1 Blackboard Add Link Cancel Save

White Box testing

To test the method coverage, statement coverage and branch coverage of adding a note to an assessment, adding an assessment to a module, and adding a note to a module; I used Debug.Writeline() to show when a method was called, when statement was encountered and for every branch of each statement. I chose to test creating a note for a module/assessment and adding images/links and a title to the note because there are a lot of statements and branches that would allow for thorough testing. My results are shown below.

Adding a note to an assessment

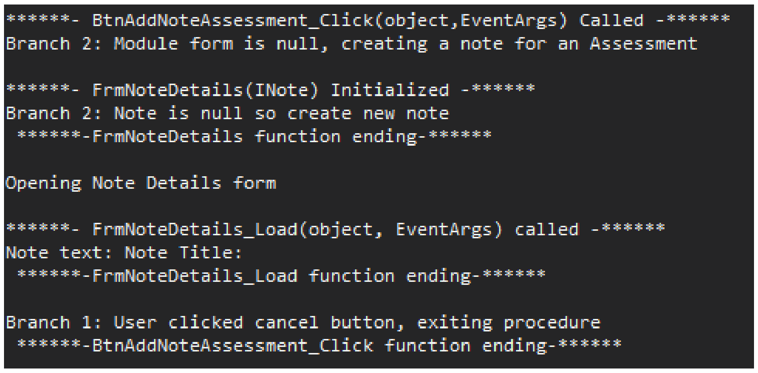
I started by choosing an assessment and clicking 'add' on the notes form, this then opened the note details form where I proceeded to give the note a title, and then test adding an Image and a Link and also cancelling mid way through both to cover all branches. After that I clicked the 'save' button and the note was added to the assessment.



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 2: Module form is null, creating a note for an Assessment

******- FrmNoteDetails(INote) Initialized -******
Branch 2: Note is null so create new note
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title:
 ******-FrmNoteDetails_Load function ending-******


******- AddImage_Click(object,EventArgs) called -******
Creating new image form and showing to user
Branch 1: User clicked cancel button, exiting procedure
 ******-AddImage_Click function ending-******


******- AddImage_Click(object,EventArgs) called -******
Creating new image form and showing to user
Adding new image Testing to notes image list
Adding new picturebox Testing to panel
 ******-AddImage_Click function ending-******


******- BtnAddLink_Click(object,EventArgs) called -******
Creating new Link form and showing to user
Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddLink_Click function ending-******


******- BtnAddLink_Click(object,EventArgs) called -******
Creating new Link form and showing to user
Adding new link  to notes link list
Adding new linklabel  to panel
 ******-BtnAddLink_Click function ending-******


******- BtnSave_Click(object,EventArgs) called -******
Branch 2: Note title is null, empty or white space

******- BtnSave_Click(object,EventArgs) called -******
Branch 1: Note title is not null, empty or white space
Notes title: Note 1
 ******-BtnSave_Click function ending-******

Branch 1: User clicked save and the new note has a unique title
Adding new note to assessment and calling function to create a button for it
 * *****-AddButtons(string,bool) Called- ******
New buttons name/text: Note 1
Branch 1: New button is for a Note
 ******-AddButtons function ending-******

 ******-BtnAddNoteAssessment_Click function ending-******
```

Cancelling adding a note to an assessment

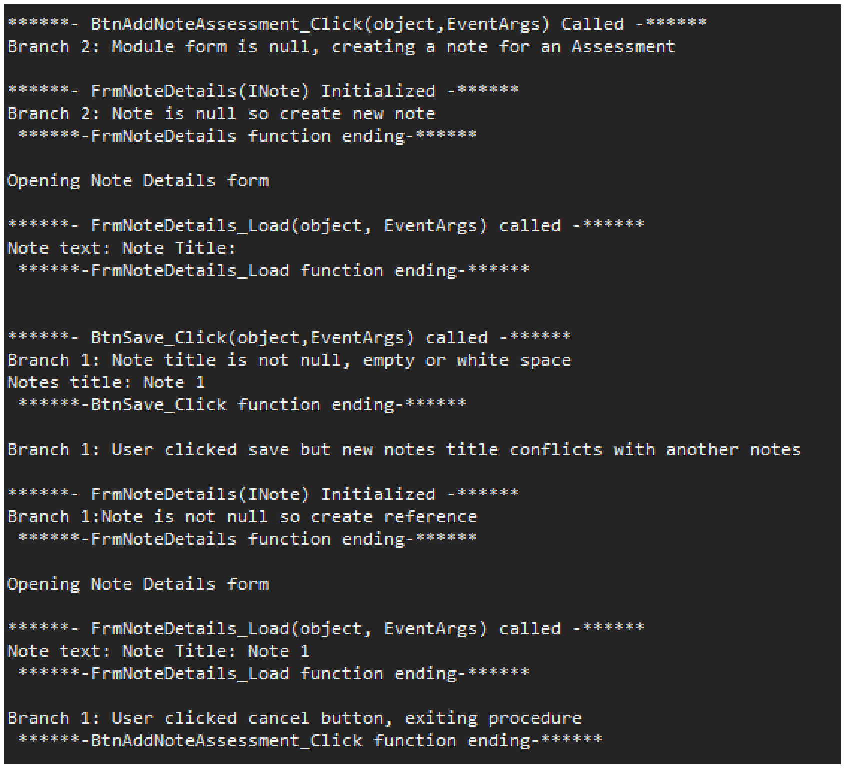
I then decided to cancel adding a note to achieve a large branch coverage.



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 2: Module form is null, creating a note for an Assessment

******- FrmNoteDetails(INote) Initialized -******
Branch 2: Note is null so create new note
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title:
 ******-FrmNoteDetails_Load function ending-******

Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```

Adding a note to an assessment with a conflicting name

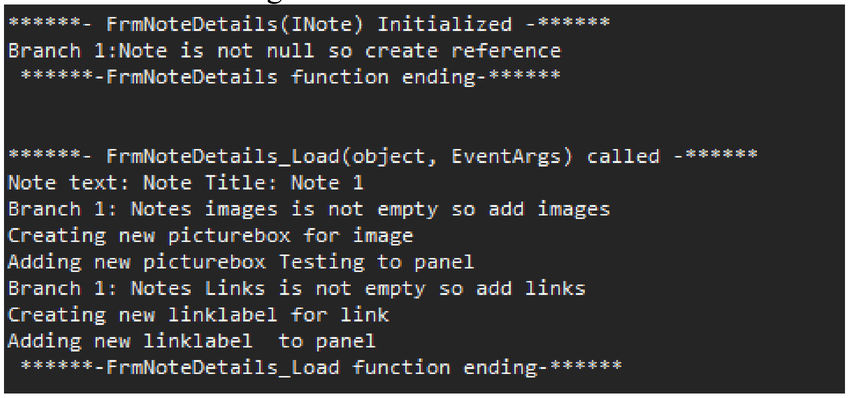
Since each note must have a name unique to that notes list I needed to test whether a statement would stop the user adding a note with the same name as another notes



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 2: Module form is null, creating a note for an Assessment

******- FrmNoteDetails(INote) Initialized -******
Branch 2: Note is null so create new note
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title:
 ******-FrmNoteDetails_Load function ending-******


******- BtnSave_Click(object,EventArgs) called -******
Branch 1: Note title is not null, empty or white space
Notes title: Note 1
 ******-BtnSave_Click function ending-******

Branch 1: User clicked save but new notes title conflicts with another notes

******- FrmNoteDetails(INote) Initialized -******
Branch 1:Note is not null so create reference
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title: Note 1
 ******-FrmNoteDetails_Load function ending-******

Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```

Opening an assessments note

I also needed to cover branches of statements to add images and links if a note was passed to the form with images and links



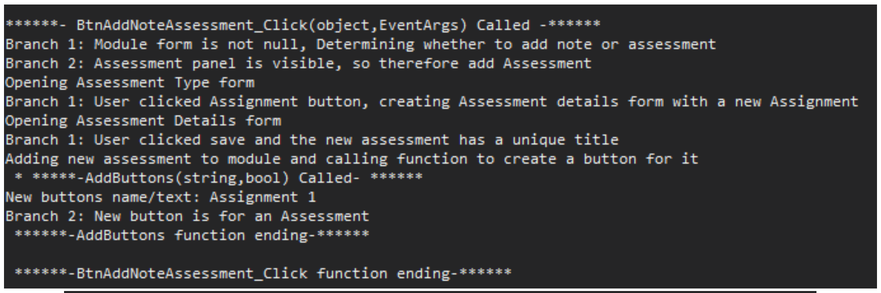
```
******- FrmNoteDetails(INote) Initialized -******
Branch 1:Note is not null so create reference
 ******-FrmNoteDetails function ending-******


******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title: Note 1
Branch 1: Notes images is not empty so add images
Creating new picturebox for image
Adding new picturebox Testing to panel
Branch 1: Notes Links is not empty so add links
Creating new linklabel for link
Adding new linklabel  to panel
 ******-FrmNoteDetails_Load function ending-******
```

Adding an assessment to a module

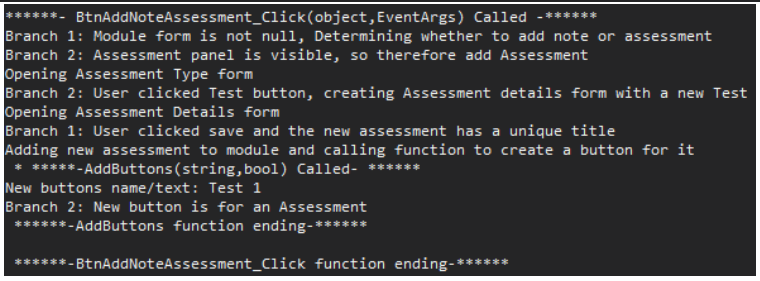
Although I covered all the methods in the previous tests, I needed to test adding an assessment to a module to cover more statements and branches.

I first chose to add an assessment, and covered adding both a test and an assignment.

Then I added a title to each assessment and clicked save.



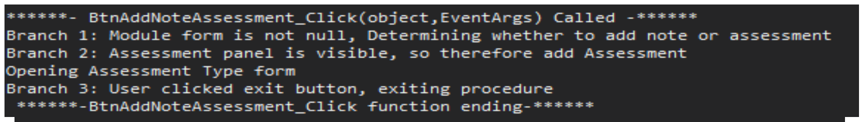
```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 2: Assessment panel is visible, so therefore add Assessment
Opening Assessment Type form
Branch 1: User clicked Assignment button, creating Assessment details form with a new Assignment
Opening Assessment Details form
Branch 1: User clicked save and the new assessment has a unique title
Adding new assessment to module and calling function to create a button for it
 * *****-AddButtons(string,bool) Called- ******
New buttons name/text: Assignment 1
Branch 2: New button is for an Assessment
 ******-AddButtons function ending-******

 ******-BtnAddNoteAssessment_Click function ending-******
```



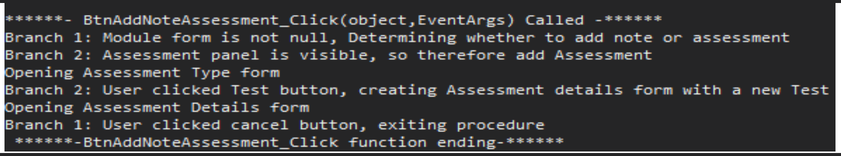
```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 2: Assessment panel is visible, so therefore add Assessment
Opening Assessment Type form
Branch 2: User clicked Test button, creating Assessment details form with a new Test
Opening Assessment Details form
Branch 1: User clicked save and the new assessment has a unique title
Adding new assessment to module and calling function to create a button for it
 * *****-AddButtons(string,bool) Called- ******
New buttons name/text: Test 1
Branch 2: New button is for an Assessment
 ******-AddButtons function ending-******

 ******-BtnAddNoteAssessment_Click function ending-******
```

Cancelling adding an assessment

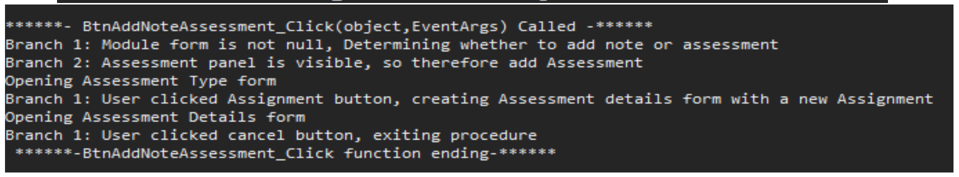
I then tested cancelling adding each type of assessment, first by closing the assessment type form, and then by clicking the 'cancel' button on each assessment details form.



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 2: Assessment panel is visible, so therefore add Assessment
Opening Assessment Type form
Branch 3: User clicked exit button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```



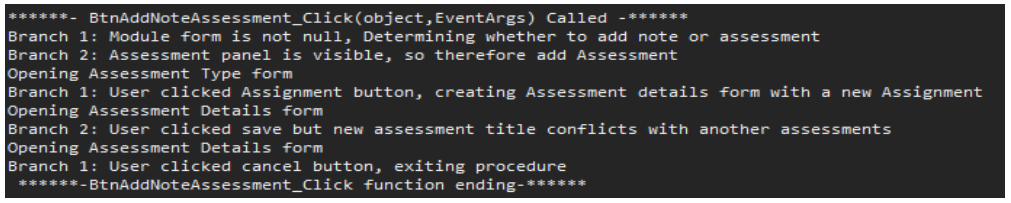
```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 2: Assessment panel is visible, so therefore add Assessment
Opening Assessment Type form
Branch 2: User clicked Test button, creating Assessment details form with a new Test
Opening Assessment Details form
Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 2: Assessment panel is visible, so therefore add Assessment
Opening Assessment Type form
Branch 1: User clicked Assignment button, creating Assessment details form with a new Assignment
Opening Assessment Details form
Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```

Adding an assessment to a module with a conflicting name

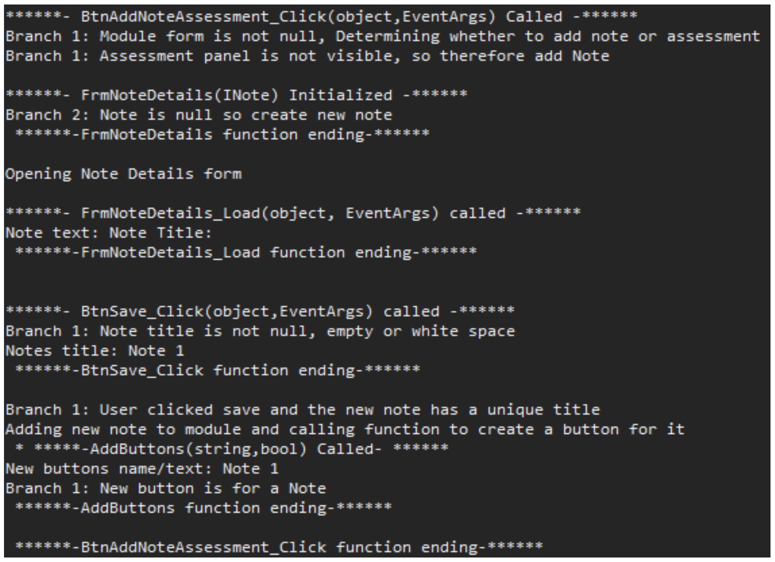
Just like notes, assessments must have titles unique to their assessment list, so I needed to test whether my logic in determining whether the title was unique or not worked.



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 2: Assessment panel is visible, so therefore add Assessment
Opening Assessment Type form
Branch 1: User clicked Assignment button, creating Assessment details form with a new Assignment
Opening Assessment Details form
Branch 2: User clicked save but new assessment title conflicts with another assessments
Opening Assessment Details form
Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```

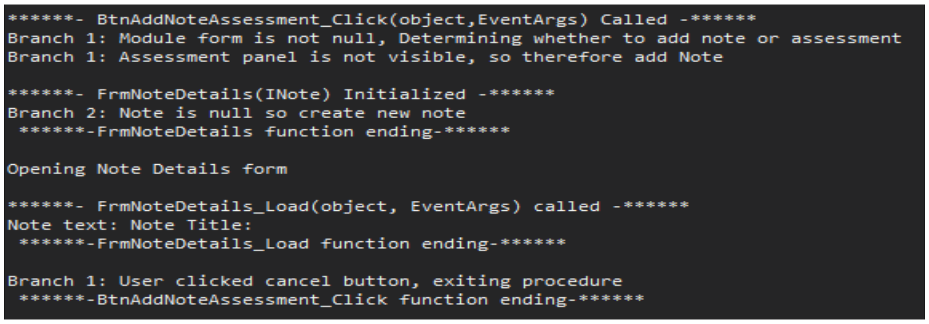
Adding a note to a Module

In order to cover every branch of each statement, I needed to add a note to a module.



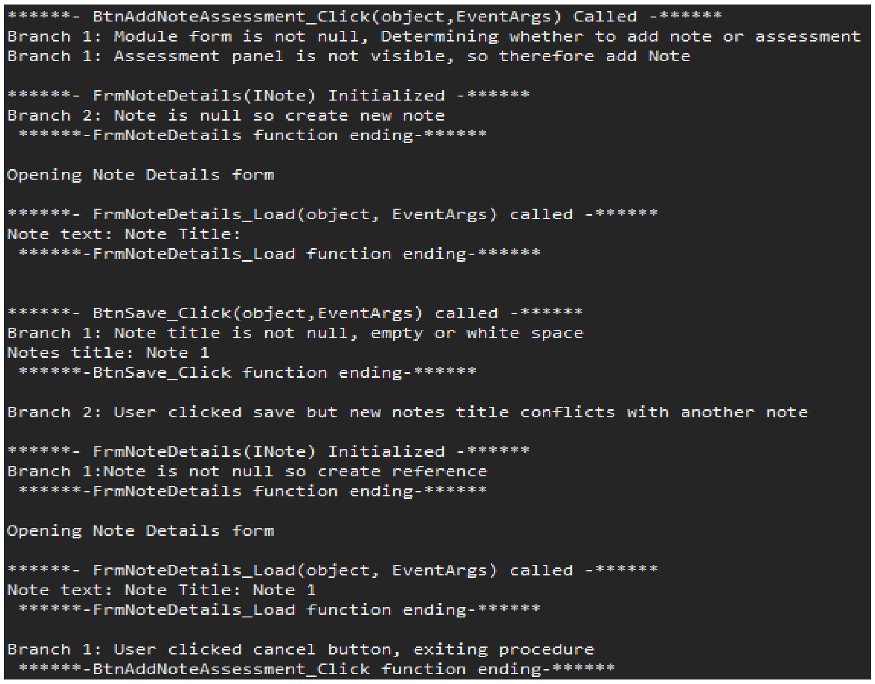
```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 1: Assessment panel is not visible, so therefore add Note

******- FrmNoteDetails(INote) Initialized -******
Branch 2: Note is null so create new note
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title:
 ******-FrmNoteDetails_Load function ending-******


******- BtnSave_Click(object,EventArgs) called -******
Branch 1: Note title is not null, empty or white space
Notes title: Note 1
 ******-BtnSave_Click function ending-******

Branch 1: User clicked save and the new note has a unique title
Adding new note to module and calling function to create a button for it
 * *****-AddButtons(string,bool) Called- ******
New buttons name/text: Note 1
Branch 1: New button is for a Note
 ******-AddButtons function ending-******

 ******-BtnAddNoteAssessment_Click function ending-******
```

Cancelling adding a note to a module



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 1: Assessment panel is not visible, so therefore add Note

******- FrmNoteDetails(INote) Initialized -******
Branch 2: Note is null so create new note
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title:
 ******-FrmNoteDetails_Load function ending-******

Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```

Adding a note to a module with a conflicting name



```
******- BtnAddNoteAssessment_Click(object,EventArgs) Called -******
Branch 1: Module form is not null, Determining whether to add note or assessment
Branch 1: Assessment panel is not visible, so therefore add Note

******- FrmNoteDetails(INote) Initialized -******
Branch 2: Note is null so create new note
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title:
 ******-FrmNoteDetails_Load function ending-******


******- BtnSave_Click(object,EventArgs) called -******
Branch 1: Note title is not null, empty or white space
Notes title: Note 1
 ******-BtnSave_Click function ending-******

Branch 2: User clicked save but new notes title conflicts with another note

******- FrmNoteDetails(INote) Initialized -******
Branch 1:Note is not null so create reference
 ******-FrmNoteDetails function ending-******

Opening Note Details form

******- FrmNoteDetails_Load(object, EventArgs) called -******
Note text: Note Title: Note 1
 ******-FrmNoteDetails_Load function ending-******

Branch 1: User clicked cancel button, exiting procedure
 ******-BtnAddNoteAssessment_Click function ending-******
```

Conclusion

To conclude the testing, the method coverage, statement coverage and branch coverage of these actions are shown below.

MEASURMENTS	RESULTS
METHOD COVERAGE	70%
STATEMENT COVERAGE	71%
BRANCH COVERAGE	68%

Video URL

https://www.youtube.com/watch?v=0VUsGcgliNA

References

Mccutchen, R. (2010, May 17). *C# Interfaces, what are they and why use them?* Retrieved from Dzone: https://dzone.com/articles/c-interfaces-what-are-they-and