UML CLASS DIAGRAM BUILDER

SOFTWARE DESIGN & TEST RESULTS TECHNICAL REPORT 12-16-20

CIS 375 Software Engineering I

College of Engineering and Computer Science
University of Michigan Dearborn
Dearborn, Michigan

CIS 375, Software Engineering I, Group 2

Table of Contents

1.0 Introduction

This section provides an overview of the entire design document. This document describes all data, architectural, interface and component-level design for the software.

1.1 Goals and objectives

The UML Class Diagram Builder is an online graphical tool used to aid in the design and creation of UML class diagrams. It attempts to bring creating UML class diagrams down to a level that any computer savvy user can perform. Users will be presented with a simple "get started" screen to begin. After selecting the "get started" button, users will then be displayed with the UML Class Creator Page. Actions, such as, name, variables, and functions make creating classes a simple process for all users. A hover-over tooltip window provides users with helpful information of any variable that their mouse hovers over. At the completion of the classes, users are taken to a view page screen that presents users with their completed classes with the option to print their classes.

1.2 Statement of scope

The main purpose of the UML Class Diagram Builder is to help automate the process of creating a UML class diagram. The software will consist of a number of inputs, graphically assisting the user in creating class diagrams including the following:

- Simplify the creation of a UML diagram
- Minimizing a user's time by allowing them to input their data
- Allow our program to design the diagrams for the users

1.3 Software context

UML Class Diagram Builder is being marketed as a tool to allow any user to simplify the creation of creating and designing their own class diagrams. Users will not need to have prior knowledge of class diagrams due to our tooltip window that provides users with helpful information to complete their diagram(s). UML Class Diagram builder is available free for educational use and will be distributed via the University of Michigan-Dearborn's Computer and Information Science student's webpage.

1.4 Major constraints

- Time
  - We have 1 month to plan, design, program, and document the software. We have a lot of ideas that we cannot implement due to the time constraint.
- Technology
  - Members of the development team do not have the necessary software to create such a program which leads us to open-source software.
- Knowledge
  - A team member with the lack of knowing javascript, html, or css constrains the project due to the need of learning such languages to complete the project.
- Quality
  - Lack of quality due to the software no-longer being maintained after completion of the project. Quality is also affected due to the inexperience the team has with the used languages.

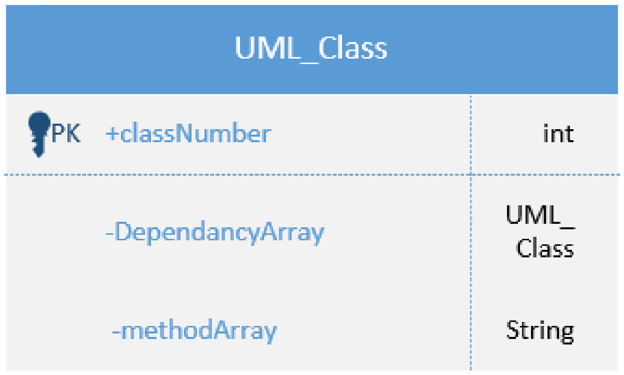
2.0 Data design

A description of all data structures including internal, global, and temporary data structures.

2.1 Internal software data structure

Data structures that are passed among components of the software are described.

- Below was our initial data structural plan. However, many changes had to be made. This internal software UML class diagram was not used. Instead our program made use of javascript variables. No other internal data structure was used as nothing was stored in the website.



UML_Class	
PK +classNumber	int
-DependancyArray	UML_ Class
-methodArray	String

2.2 Global data structure

- This website did not make use of any global data structures as no information was stored in the website.

2.3 Temporary data structure

Files created for interim use are described.

- The UML Class Diagram Builder will utilize the use of javascript variables to store temporary values.

2.4 Database description

Database(s) created as part of the application is(are) described.

- The UML Class Diagram Builder will not be utilizing a database as the software will not have the ability to "save" any of the diagrams nor will the software have the ability to store user information. All data within the software will be deleted once the user terminates the session.

3.0 Architectural and component-level design

A description of the program architecture is presented.

We will be using object-oriented architecture, a design pattern based on the division of responsibilities for an application or system into individual reusable and self-sufficient objects.

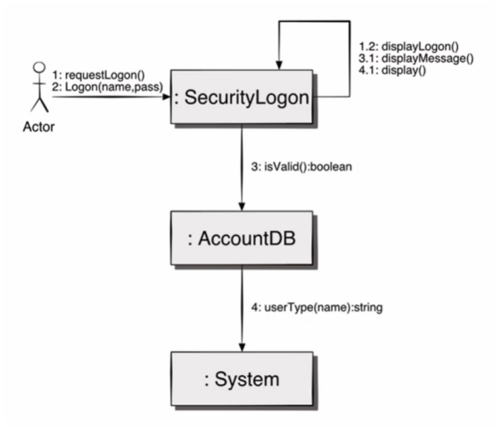
3.1 Program Structure

A detailed description of the program structure chosen for the application is presented.

3.1.1 Architecture diagram

A pictorial representation of the architecture is presented.

Object-Oriented Architecture



3.1.2 Alternatives

A discussion of other architectural styles considered is presented. Reasons for the selection of the style presented in Section 3.1.1 are provided.

Call and return architecture. This architecture provides high security for internal structures, and data is manipulated using only the appropriate methods. This style of architecture is not efficient for high performance computing, it has extensive middleware to provide access to remote objects, and it can lead to highly complex applications.

3.2 Description for Component n

A detailed description of each software component contained within the architecture is presented. Section 3.2 is repeated for each of n components.

N ID #	**(3.2.2) Component N**	**(3.2.1) Processing narrative (description)**	**(3.2.3) N inputs and outputs & algorithmic model & local data structures**	**(3.2.3) N Restrictions & performance issues & design constraints**
#001	Class textboxes	The class textboxes will hold the class's name, attributes, operations (or methods) and relationship among objects.	Users will type directly into the text boxes, their data stored as string. Their input will be displayed on the final screen as uneditable text.	Class text boxes can not be stretched or compressed

			The class textbox objects will be stored in an ArrayList.	
#002	Diagram Title textbox	The title text box will hold the title of the class diagram	Users can input text as strings into the textbox. It will be displayed above the diagram in the final page.	No special character other than ‘_’ Also class names can not start with a number.
#003	“Add” button	The Add button allows the user to add another blank class diagram to the webpage	Users will be required to click on the button to create another blank class to the webpage. The objects would be stored in an ArrayList of the UML class type.	Can not add more than 25 class objects.
#004	“+” Button	The plus button allows the user to add another function or variable in their class diagram.	Users will be required to click on the field to add another variable name or function name to the UML class. The strings will be stored in an ArrayList.	No restrictions
#005	“Save & Finish” button	The Save and Finish button will show the user's final class diagram.	Users will click the “Save & Finish” button which will display the final class diagrams. All class objects will be stored in an ArrayList.	No restrictions
#006	“Print” button	The Print button will only print the UML Class diagram	The user will click “Print” to print all the information in the class diagram.	No restrictions
#007	“-” button	The minus button allows the user to delete a function, variable, or class in their class diagram.	The user will click the minus button to delete a function, variable, or function.	Can not go back further than one field.
#008	Dependencies Text-box	Dependencies text box will allow the user to enter any class name	The text-box will take input from the user via the text-box.	No restrictions
#009	Help button	The help button is responsible for taking the user to a help page, where an FAQ and contact information is available	This object has no inputs. When clicked, it will take you to a new page. The help page will have no inputs or outputs, and consist of displayed, uneditable text stored as strings.	Only available help is through the help page. No other information can be found on the internet.
#010	Hover-over tooltips	The hover-over tooltips show helpful descriptions of everything that is hovered over.	This object has no inputs. Users will be prompted with a side-window with tips when their mouse is hovered over a button.	Can not click for more information.
#011	“Back” button	The back button takes you back to the UML Class Creator page.	The output takes you to the previous page. No input or data structures.	Can not go back before saving and finishing the UML class diagram.

3.2.1 Processing narrative (PSPEC) for component n

A processing narrative for component n is presented.

Look at table above, column 2

3.2.2 Component n interface description.

A detailed description of the input and output interfaces for the component is presented.

Look at table above, column 2

3.2.3 Sub-Component n.m processing detail

A detailed algorithmic description for each sub-component within the component n is presented. Section 3.2.3 is repeated for each of the m sub-components of component n.

No sub-components available

3.2.3.1 Interface description

A description of sub-component m inputs and outputs is presented.

No sub-components available

3.2.3.2 Algorithmic model (e.g., PDL)

The pseudocode listing for sub-component m is presented.

Algorithmic model for component n, listed in above table, column 3

3.2.3.3 Restrictions/limitations

The external environment and/or infrastructure that must exist for sub-component m to operate correctly is provided.

Restrictions/limitations for component n, listed in above table, column 4

]**3.2.3.4 Local data structures**

The data structures used within sub-component m are presented.

Local data structures for component n, listed in above table, column 3

3.2.3.5 Performance issues

Information on topics that may affect the run-time performance, security, or computational accuracy of this sub-component are presented.

Performance issues for component n, listed in above table, column 4

3.2.3.6 Design constraints

Attributes of the overall software design (including data structures, OS features, I/O, and interoperable systems) that constrain the design of this sub-component are presented.

Design constraints for component n, listed in above table, column 4

3.3 Software Interface Description

The software's interface(s) to the outside world are described.

3.3.1 External machine interfaces

Interfaces to other machines (computers or devices) are described.

There are no external machine interfaces.

3.3.2 External system interfaces

Interfaces to other systems, products, or networks are described.

There are no external system interfaces

3.3.3 Human interface

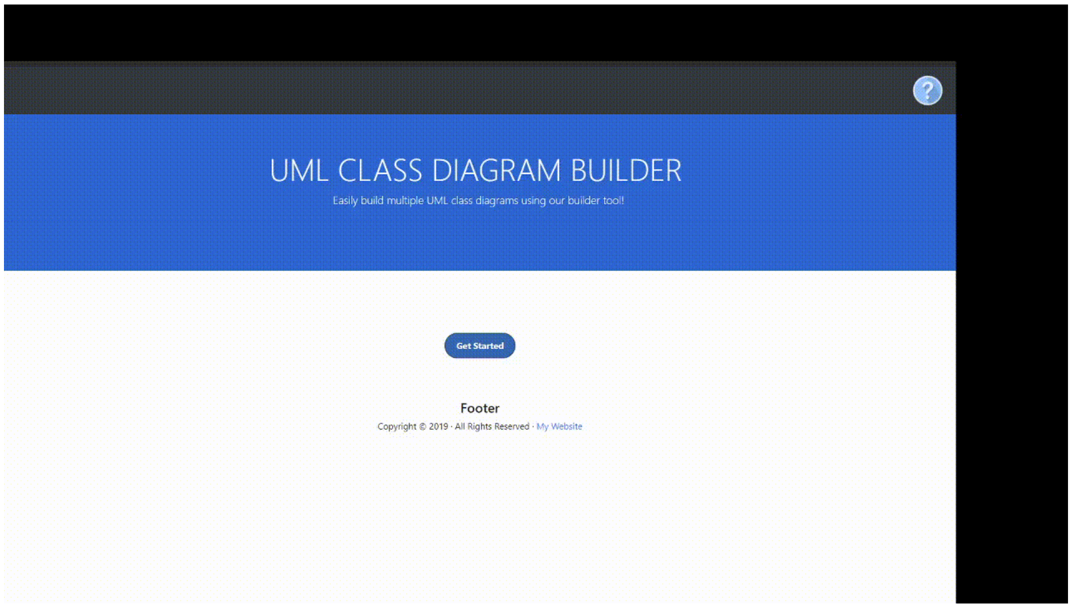
An overview of any human interfaces to be designed for the software is presented. See Section 4.0 for additional detail.

Ther user can interact with the program using a mouse, keyboard and screen.

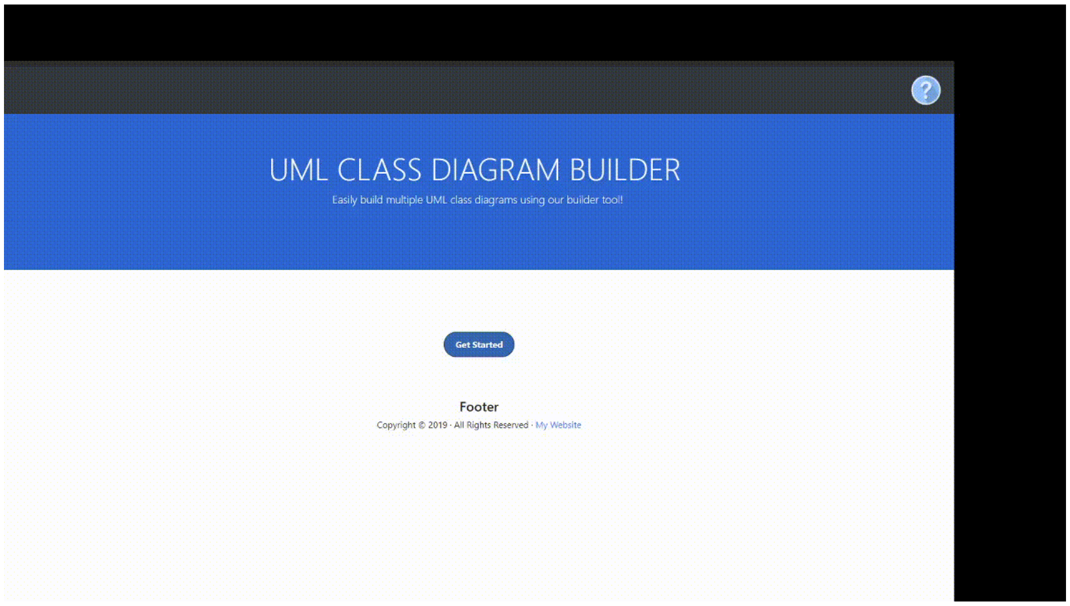
3.4 Test work products

The work products produced as a consequence of the testing procedure are identified.

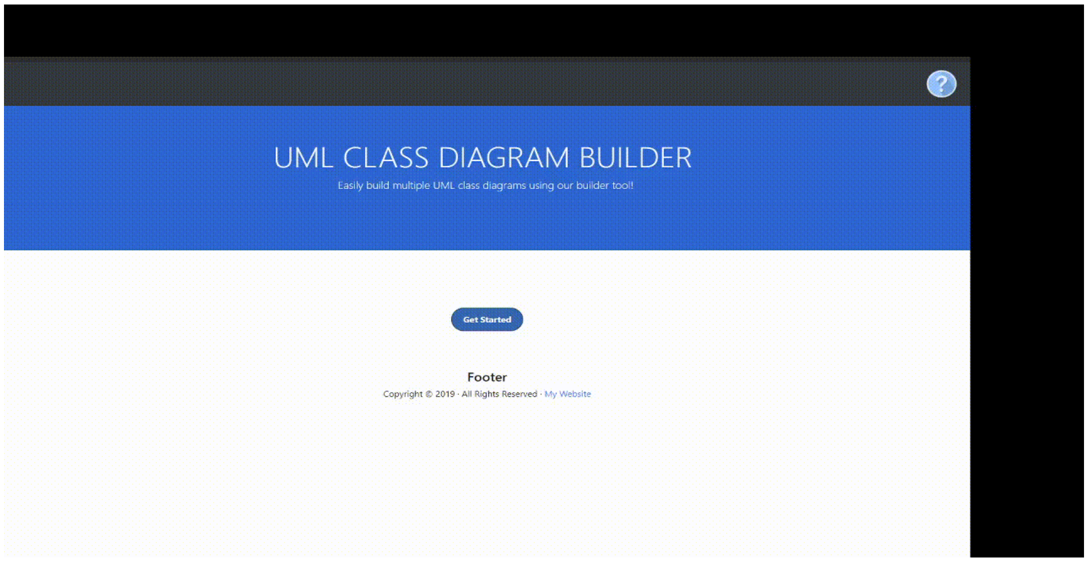
Test Number	Description of Test	Result	Pass/Fail
1	Home page test	No bugs found	Pass


UML CLASS DIAGRAM BUILDER
Easily build multiple UML class diagrams using our builder tool!
Get Started
Footer
Copyright © 2019 · All Rights Reserved · My Website

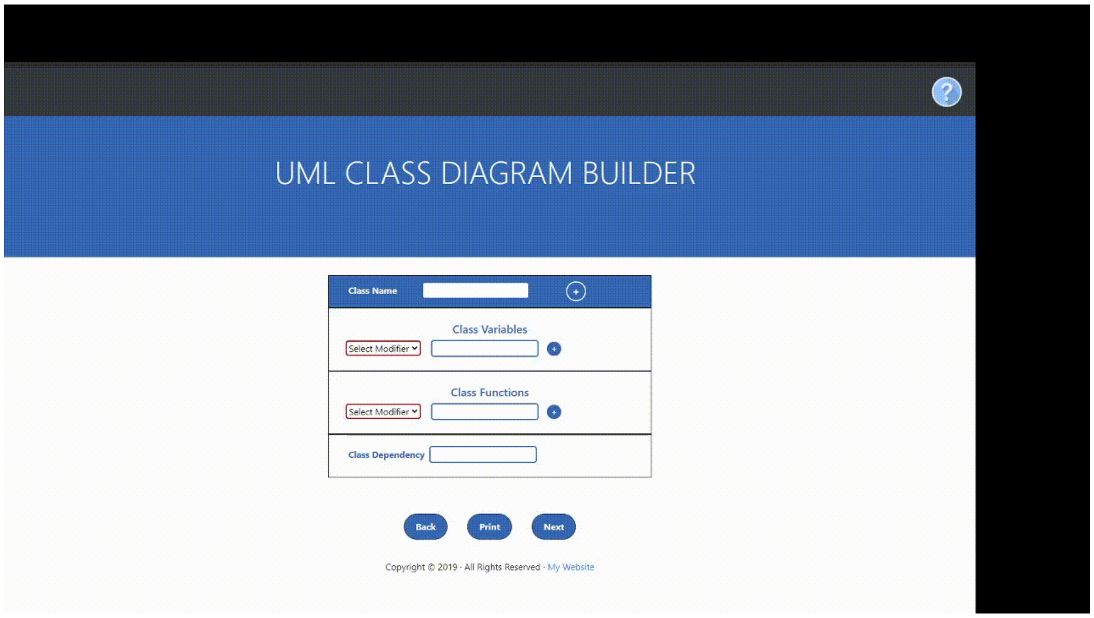
2	Help button test	Button works and takes us to the help page	Pass


UML CLASS DIAGRAM BUILDER
Easily build multiple UML class diagrams using our builder tool!
Get Started
Footer
Copyright © 2019 · All Rights Reserved · My Website

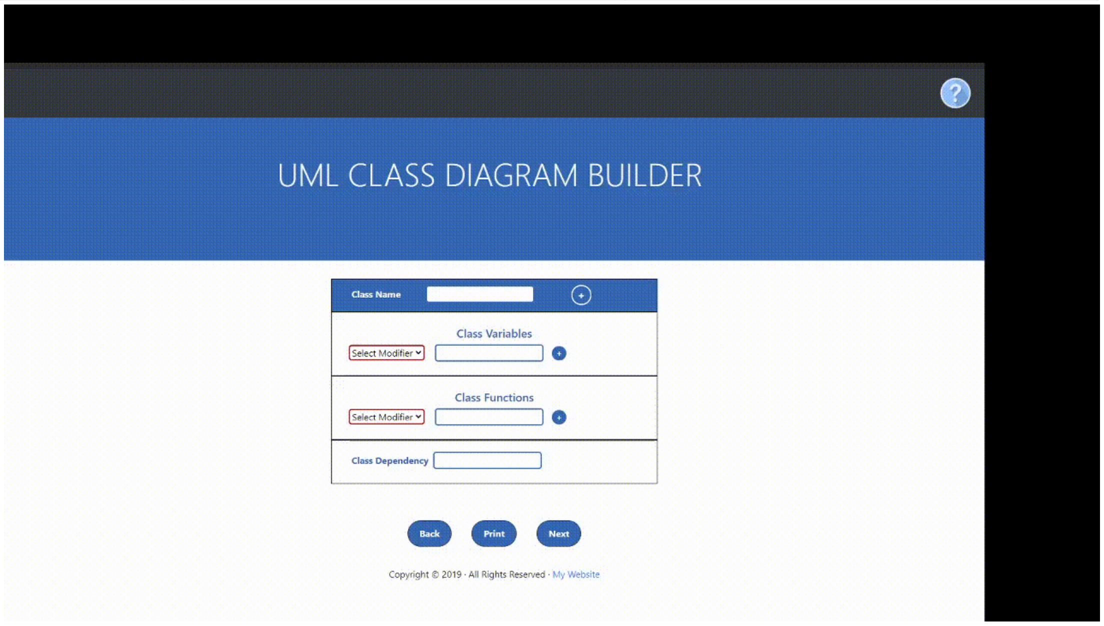
3	Start button	Button works and takes us to the build page	Pass


?
UML CLASS DIAGRAM BUILDER
Easily build multiple UML class diagrams using our builder tool!
Get Started
Footer
Copyright © 2019 · All Rights Reserved · My Website

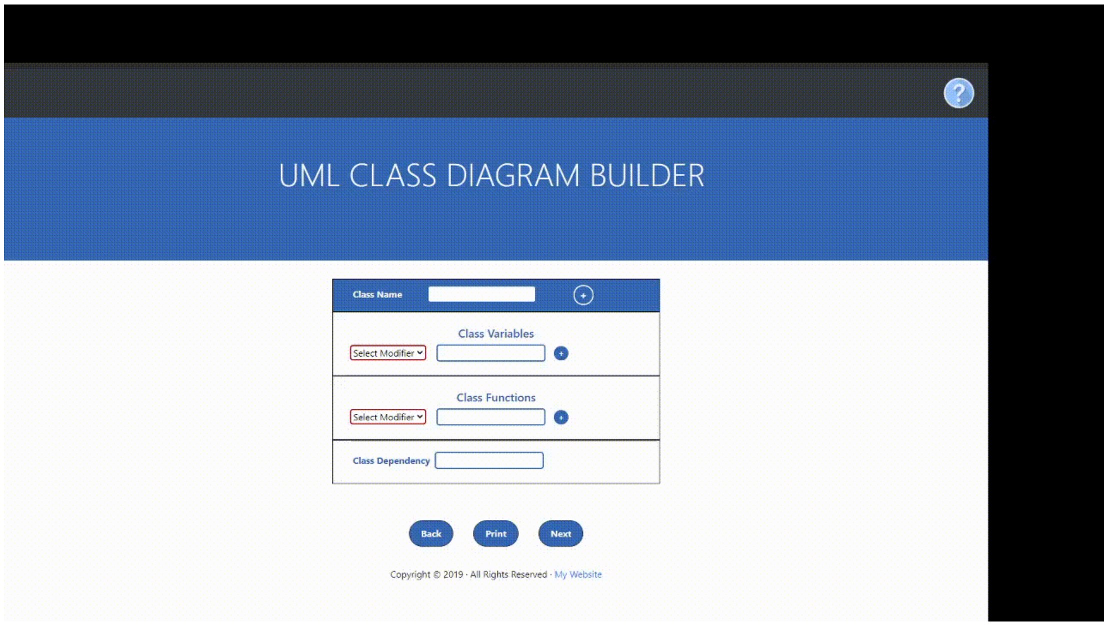
4	Class name text box	Can type	Pass



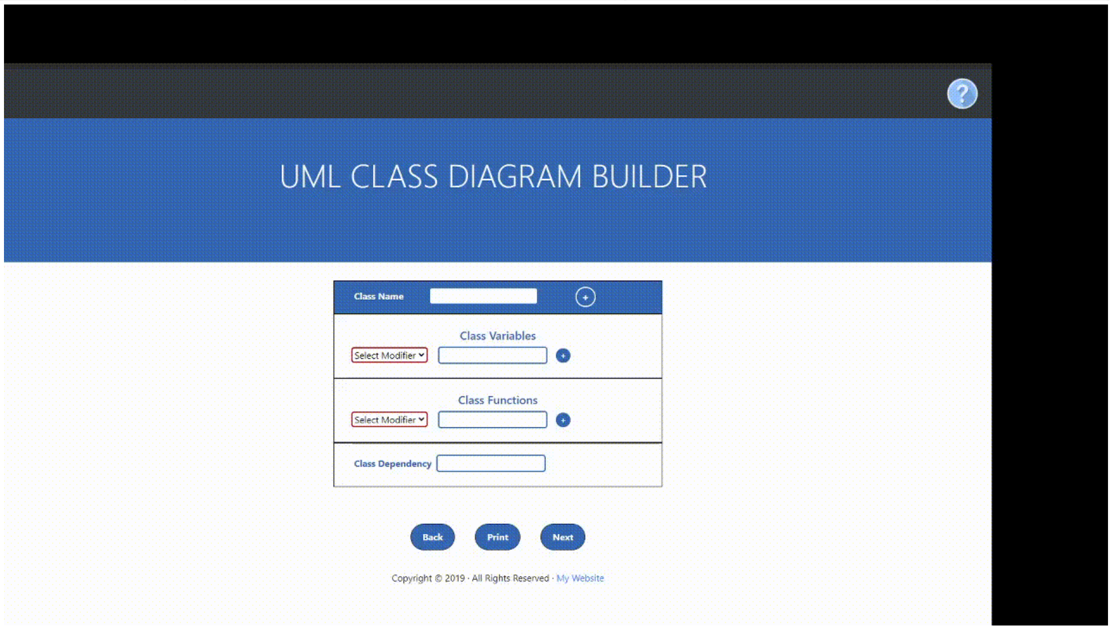
5	Variable name text box	Can type, no bugs	Pass



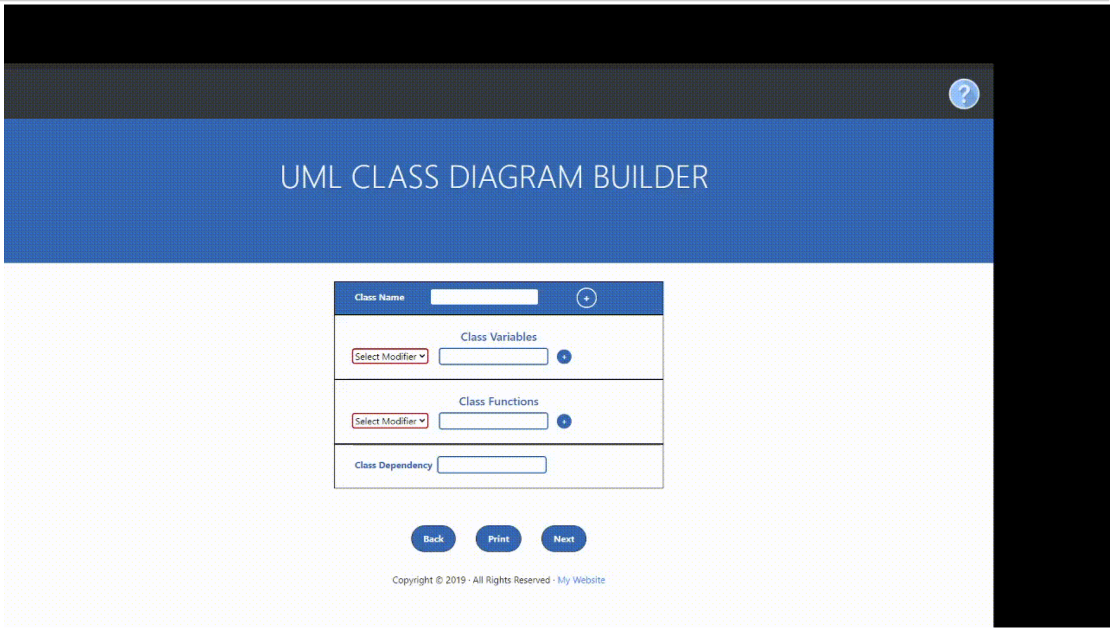
6	Add variable button	No bugs found, can add multiple text boxes	Pass



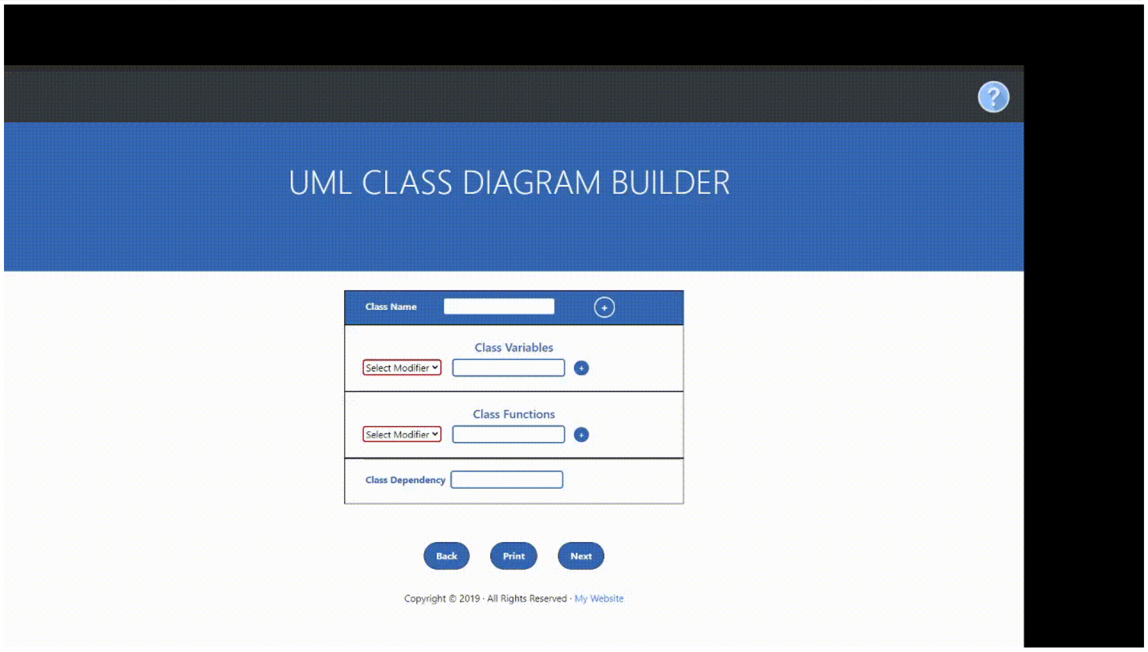
7	Function name text box	Can type, no bugs	Pass



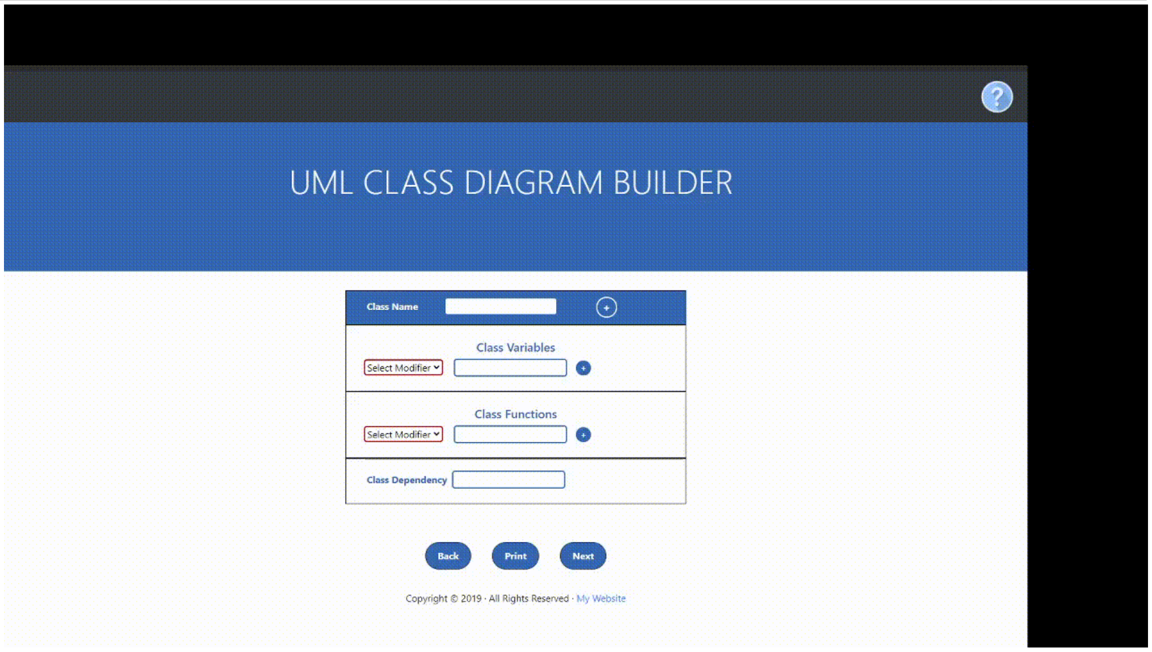
8	Add function button	No bugs found, can add multiple text boxes	Pass



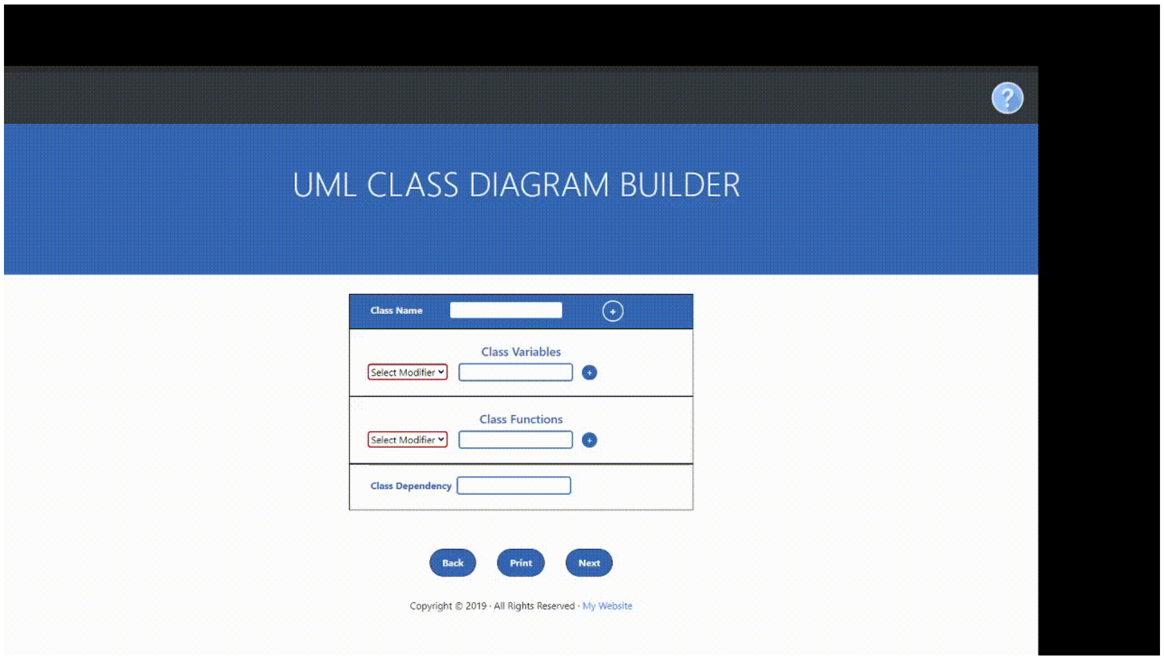
9	Dependency text box	Can type, no bugs	Pass



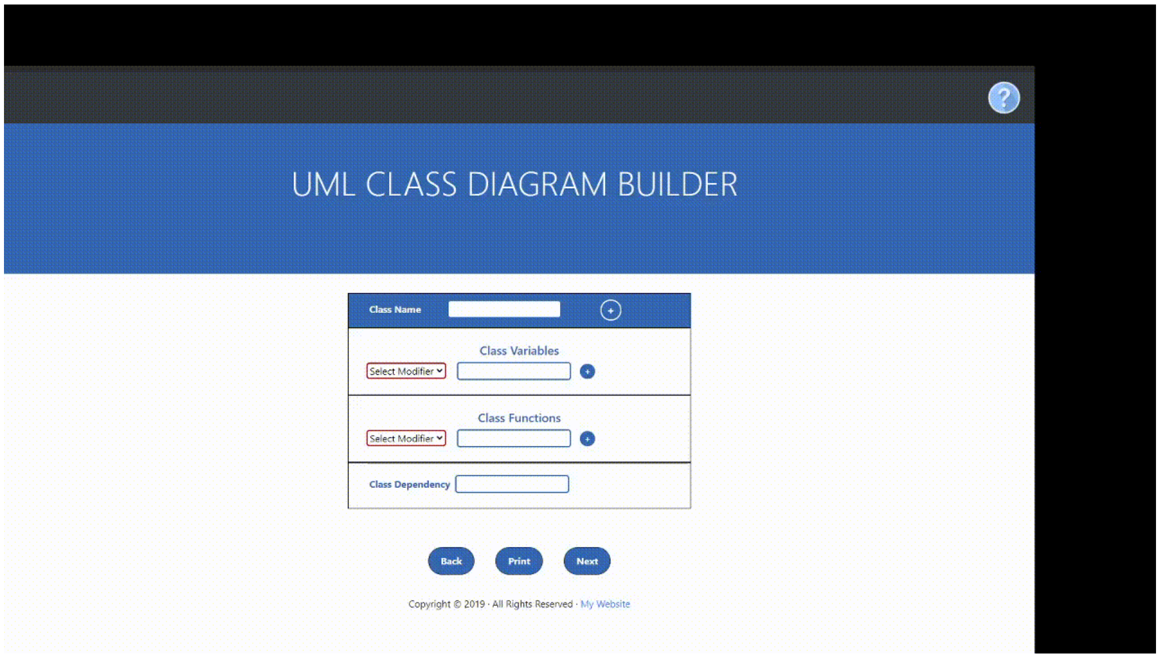
10	Add class button	No bugs found, can add multiple classes	Pass



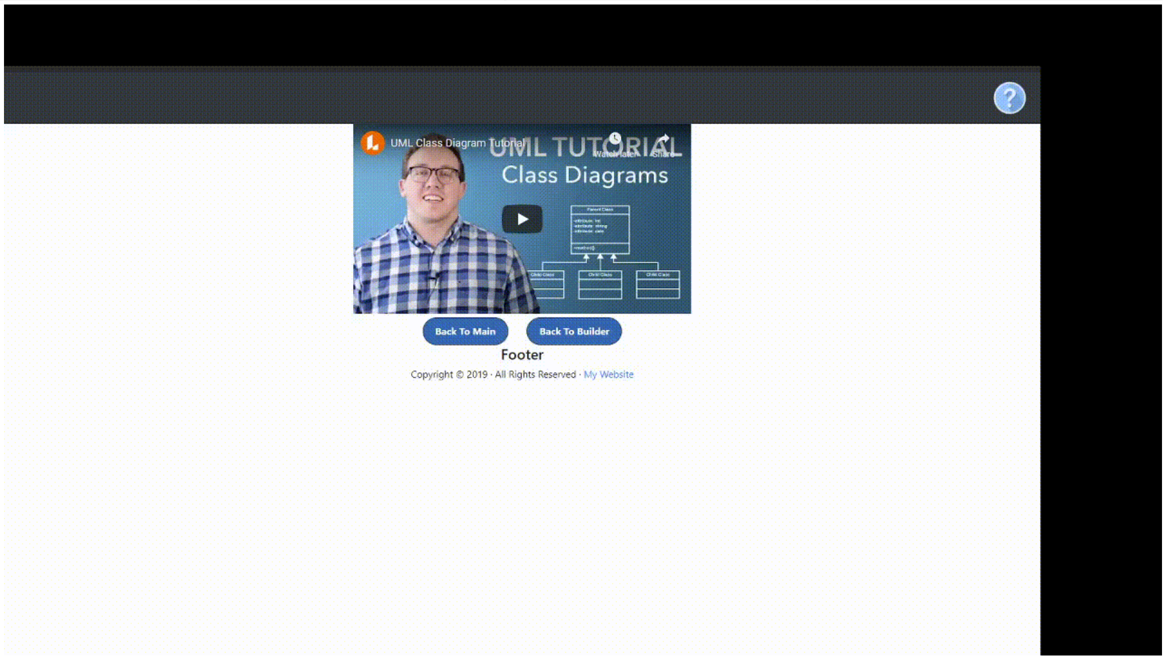
11	Print button	Button prints page, no bugs found	Pass



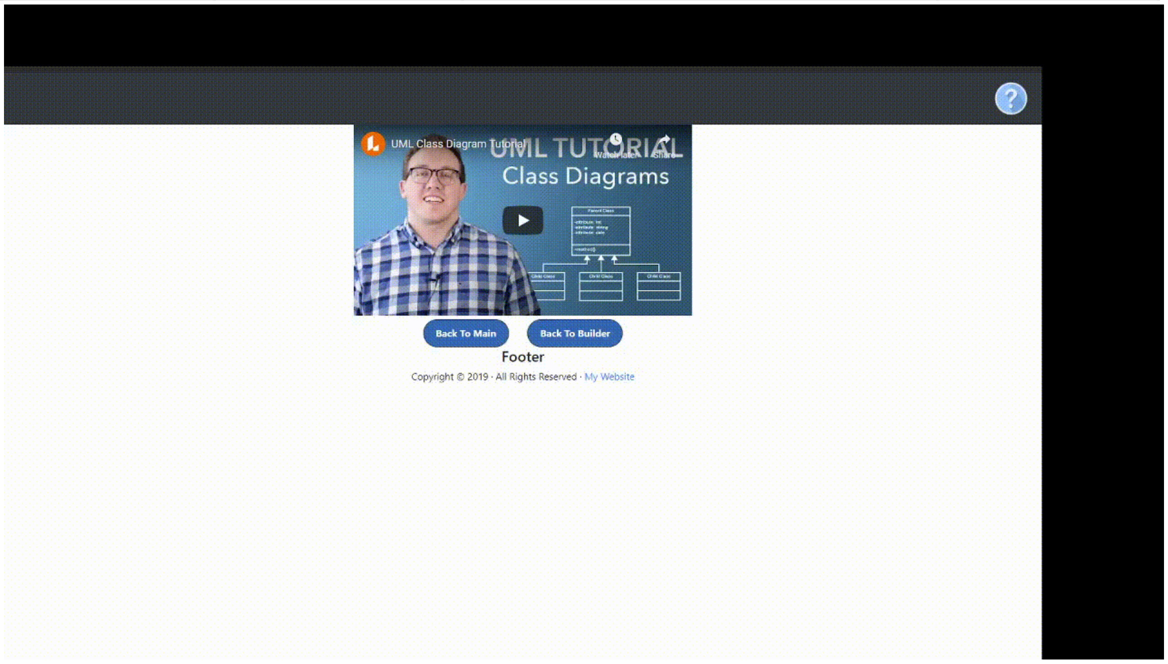
12	Finish button	Button works, takes us to the final page	Pass



13	Final page test	No bugs found	Pass


UML Class Diagram Tutorial
UML TUTORIAL
Class Diagrams
Back To Main
Back To Builder
Footer
Copyright © 2019 · All Rights Reserved · My Website

14	Restart button	Button works and takes us to the start page with local storage cleared	Pass


UML Class Diagram Tutorial
UML TUTORIAL
Class Diagrams
Back To Main
Back To Builder
Footer
Copyright © 2019 · All Rights Reserved · My Website

4.0 User interface design

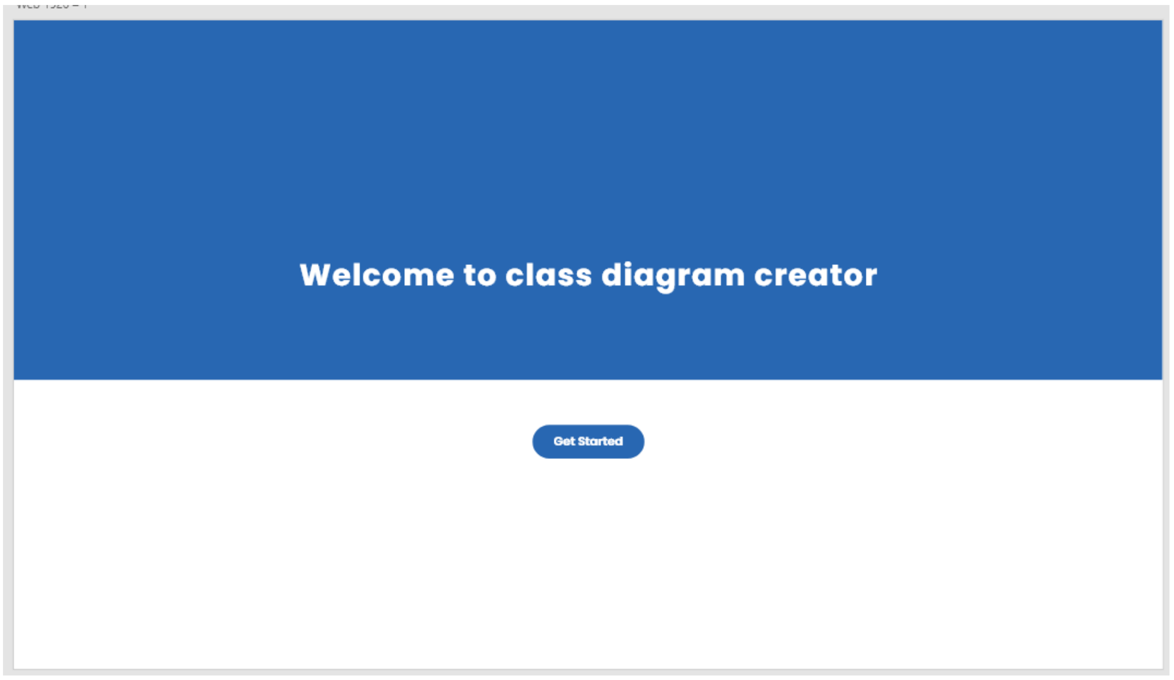
A description of the user interface design of the software is presented.

4.1 Description of the user interface

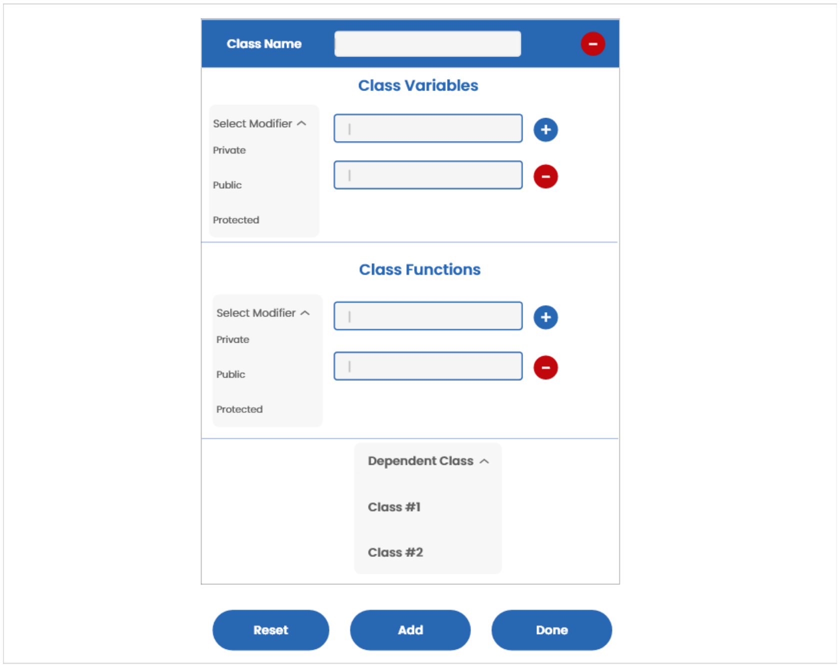
Below are some of the forms in the program. When the program is started,the welcome screen will appear. When the user clicks the get started button , it will immediately take them to the main interface.

4.1.1 Screen images

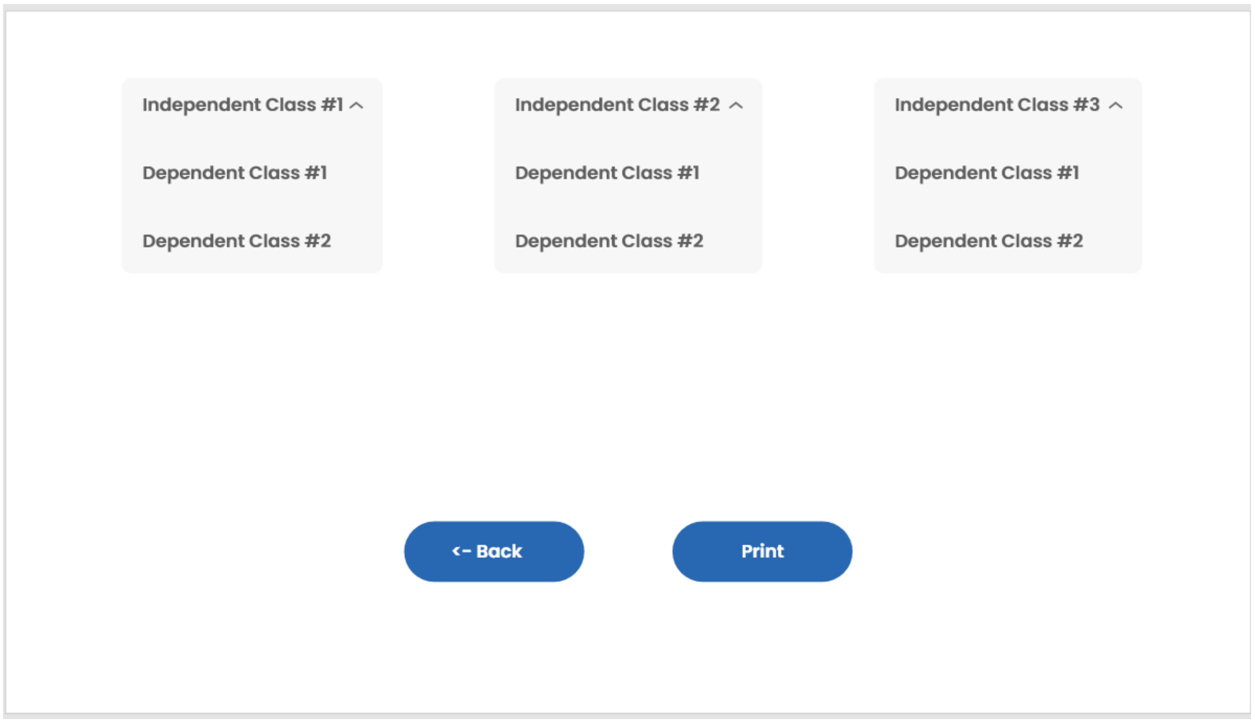
Welcome page



UML Class Creator page



View page



4.1.2 Objects and actions

All screen objects and actions are identified.

1. Welcome Screen

Get started
- When the user wants to start creating class diagrams they click the "Get Started" button to take them to that webpage.

2. UML Class Creator Page

Class Name
- Set the name of the users class diagram

Class Variables
- Users create the variables they want for their class diagrams.

Class Functions
- Users create the functions they want for their class diagrams.

Select Modifier
- A dropdown menu that allows the user to select between private, public, and protected modifier types.
- Sets the modifier type for the variables/functions in the users class diagram.

Class Dependency
- A Textbox that allows the user to enter the name of the dependent class**.**

"+"
- The plus button allows the user to add another function or variable in their class diagram and allows the user to add another UML class.

"-"
- The minus button allows the user to delete another function, variable, or class in their class diagram.

Done
- The Done button takes the user to the view page.

Print
- The print button prints the enter the diagram page.

Next
- The next button takes the user to the Final page.

3. View page

Back to Builder
- The back button takes you back to the UML Class Creator page.

Back to Main
- The back button takes you back to the main page.

4.2 Interface design rules

Conventions and standards used for designing/implementing the user interface are stated.

Interface design focuses on two areas of concern:
1. The design of interfaces between software;
2. The design of the interface between the user and the computer.
The webpage should be easy and simple to learn how to use it , readability, and easy to navigate between interfaces.

4.3 Components available

GUI components available for implementation are noted.

Since we are using Adobe Xd ,HTML, and CSS as our front-end development language and design tool, there are a lot of ready-made components that are available for us to use already. The following is a list of controls that we will be using for this software

Textbox
A textbox is a section or object on a page that allows the user to enter text and the information inputted can be used by a program.

Dropdown Menu
A dropdown menu is a menu that offers a list of options. It allows the user to choose one value from a list. When a drop-down list is inactive, it displays a single value.

Button
A button is an object that allows the user to interact with the webpage. Once a button is clicked an action will be triggered within the webpage.

4.4 UIDS description

The user interface development system is described.

For the UML class generator we're gonna develop textboxes where the user will add in the name of the class, variables, and function where that information will then be used in our backend to create the class. Also users can select if variables and functions are private, public, or protected and select what is the class dependency. If users wanted to add another class they click the Add button which then creates a blank class underneath the previous class. The + button allows the user to add a new variable or function and the - button is used to delete a class, function, or variable. When the user clicks done they are taken to a new webpage that shows a drop down menu for each of the independent classes and when these dropdown menus are selected it shows a list of their dependent class.

5.0 Restrictions, limitations, and constraints

Special design issues which impact the design or implementation of the software are noted here.

Our group is limited by our knowledge of JavaScript. We will use a website builder to create our project. This is a large constraint of the design, as we can only be as detailed as the website builder allows.

6.0 Testing Issues

6.1 Classes of tests

Black box testing is a testing technique in which the functionality of the software is tested without examination of the internal code. The test is based on the software requirements and specifications. The tester will input data into the system and verify the output in comparison to the expected output. This testing method is used to improve the system and eliminate any errors the user may face.

White box testing is a testing technique used to test the functionality of the internal structure, design, and code based on the analysis of the internal system. The tester will input data into the system and ensure the internal operations are performed according to the specifications and requirements. This testing method is used to ensure all internal components have been adequately used and deliver accurate results.

To test the system, we will start by testing the **get started** button. After pressing the button, the user should be provided with a blank class diagram to fill out. The user should also be provided with an add button, reset button, and done button.

We will test the system by selecting modifiers and entering text into the following components: class name, class variables, and class functions. We will then proceed by selecting the **add** button. This will add a new empty class for the user to fill out.

We will test the **print** button to see if the user is presented with all the classes created. This will all the user to print all the classes they've created.

We will test the **select modifier** drop down menu to assure the correct options are listed. The drop down menu should list the options: public, private, and protected. The user should be able to select one option only.

We will test the option of adding and removing variables from the class. If the user selects the + button next to the class variables, the user should be provided with an additional space to enter a variable. However, if the user selects the — button, the space will be removed.

We will test the option of adding and removing functions from the class. If the user selects the + button next to the class functions, the user should be provided with an additional space to enter a function. However, if the user selects the — button, the space will be removed.

We will test the **dependencies textbox** to make sure the user can write any class name of their choice.

We will test the **help** button located on all screens. The help button is responsible for providing the user with basic assistance. The button will take the user to a help page where an FAQ and contact information are available.

We will test the **hover-over tool** available for the class. When the user hovers over the text box, button, or any intractable feature, a simplified description or suggestion is provided to the user.

6.2 Expected software response

Expected software responses include:

- Users may create a limit of 10 classes total.
- Users will have the ability to connect multiple classes by entering their corresponding parent/child class.
- Users will have the ability to add or delete any class functions or variables.
- Users will be presented with all class diagrams they have created after pressing the print button.
- Users can choose the access modifiers for each function and variable of a class.

6.3 Performance bounds

The criteria below has been set up for the software to make sure it is user friendly and easy to use.
The system is expected to run with the bounds $0(n^2)$.
Response time for add class button function:
 Best Case Scenario: immediate
 Worst Case Scenario: 2 seconds
Response time for save & finish button function:
 Best Case Scenario: 1 second
 Worst Case Scenario: 3 seconds

6.4 Identification of critical components

1. Class Dependencies
   We will make sure classes are linked to the correct and corresponding parent class if applicable. This component will require extra testing to assure there are no mix-ups during the transfer of data.

2. Add Button
   The add button is a critical component of the system and will require extra testing. Testing of the extra button will ensure a new empty class is provided to the user and the previous class is saved and still visible to the user.

7.0 Appendices

Presents information that supplements the design specification.

7.1 Requirements traceability matrix

A matrix that traces stated components and data structures to software requirements is developed.

REQUIREMENTS TRACEABILITY MATRIX			
Software Requirements		Components & Data Structures	
Requirement ID#	Use Case	Description	Priority
#001	As a user, I want to be able to type freely into a class diagram	A freely editable textbox that resembles an empty class is available for users	Very High
#002	As a user, I want to be able to re-edit a class diagram	Completed classes stay accessible, and editable as users move on to another class	Medium
#003	As a user, I want to be able to made additional classes as I need	When a user clicks on the "Add" button, a new empty class will appear under the last class on the page Classes will be stored in an array list	High
#004	As a user, I want to be able to add and remove attributes, operations (methods), and dependents to a class	When a user clicks on the "+" or "-" button next to a class textbox, an attribute, operation, etc will appear or be removed accordingly	Medium
#005	As a user, I want to be able to click on a class and see its dependencies	A small, downwards pointing arrowhead will appear on all classes that have dependencies, and clicking on it will reveal all the dependent classes underneath the main class, in a column. Dependent classes will be stored in an array list that is linked to the parent class.	High
#006	As a user, I should be able to see a helpful description of everything I hover over	When hovering over empty textboxes, labels, and buttons, a small tooltip will appear above the object with a short description of what it does, or what to put in the field.	Low
#007	As a user, I want a help page to assist me when I am having troubles	There will be a help button on the upper right hand corner of every page that will take you to a new window with an FAQ section, a customer support number/email, and a short description of the UML class diagram builder application.	Low
#008	As a user, I want a seamless experience when making and viewing classes, as well as viewing and interacting with my final class diagram.	The application will be simple, and have appropriate icons to aid users in using the application. A user will not become "stuck" on a page and be able to navigate to any part of the application. All buttons will perform the correct tasks	Medium

7.2 Packaging and installation issues

Special considerations for software packaging and installation are presented.

The software is a program in itself, so it shouldn't have many, if any, installation problems. The software doesn't need to communicate with another software, database, or cloud storage. The software will be packaged into one .JS file, so implementation will be simple.

7.3 Design metrics to be used

A description of all design metrics to be used during the design activity is noted here.

1 function point is roughly 0.08 effort, 0.05 person months, and costs roughly $480 (data taken from previous project estimations)

Task completion rate. Completed tasks will show as a 1, and uncompleted tasks will be shown as zero. The task completion rate will be a percentage of the total number of tasks and the sum of the numbers (1 and 0).

User satisfaction (measured on a scale from 1-5, where 1 is the lowest rating, 3 is indifferent, and 5 is the highest rating)

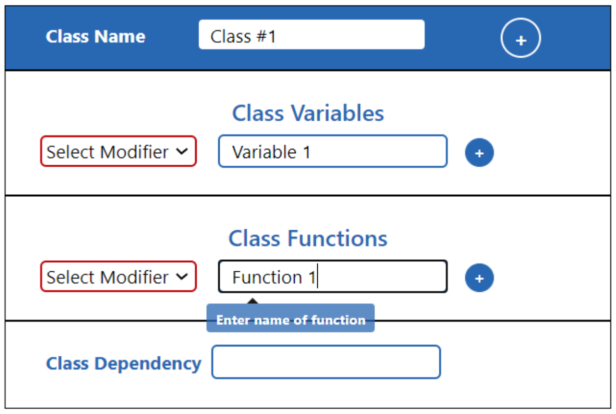
- Overall, I am satisfied with the ease of completing tasks in this scenario.
- Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.
- Overall, I am satisfied with the support information (online-help, messages, documentation) when completing the tasks in this scenario.

System Usability Scale (SUS):

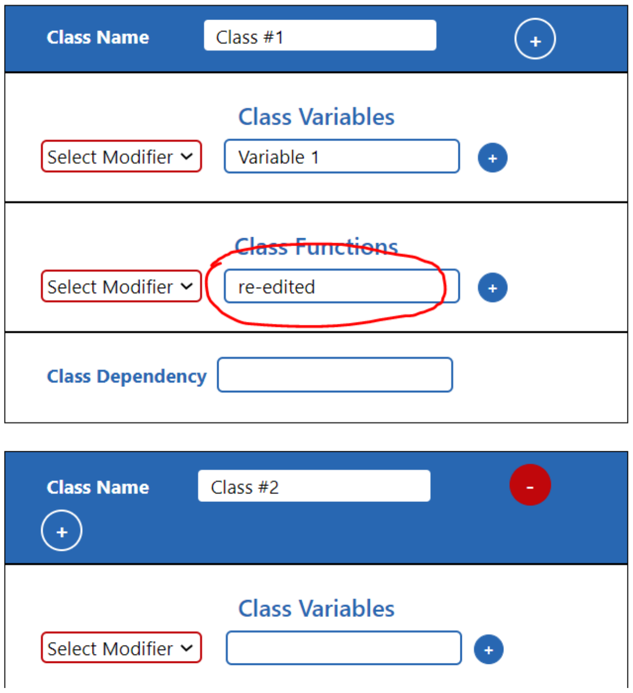
- I think that I would like to use this system frequently.
- I found the system unnecessarily complex. I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

7.4 Test Results

Requirement #001: Pass

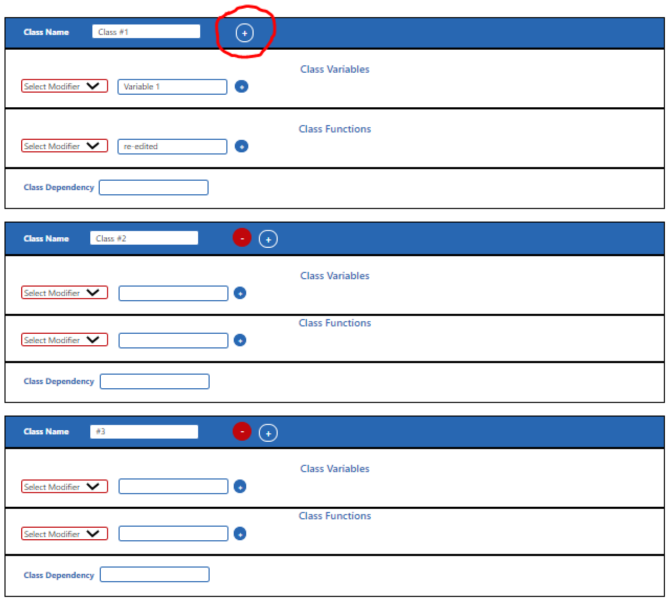


Requirement #002: Pass



Requirement #003: Pass

The button adds a new class

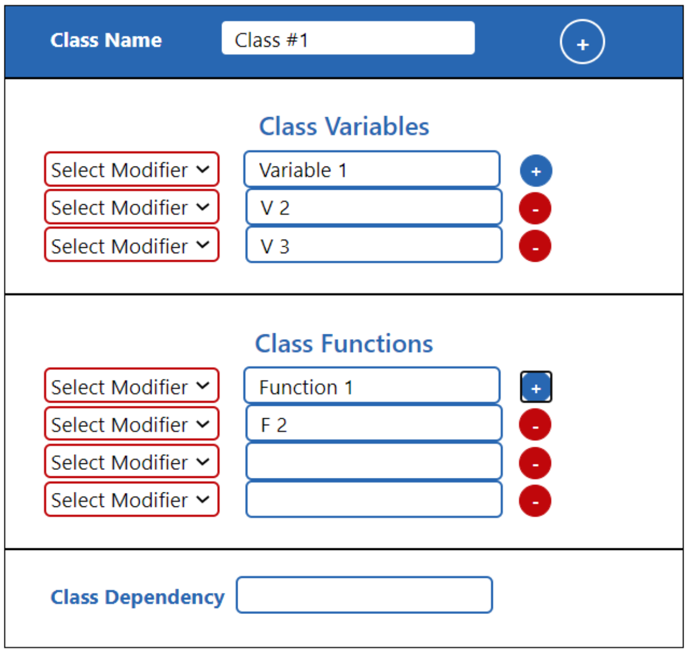


Class Name: Class #1
Variable 1
re-edited

Class Name: Class #2

Class Name: #3

Requirement #004: Pass

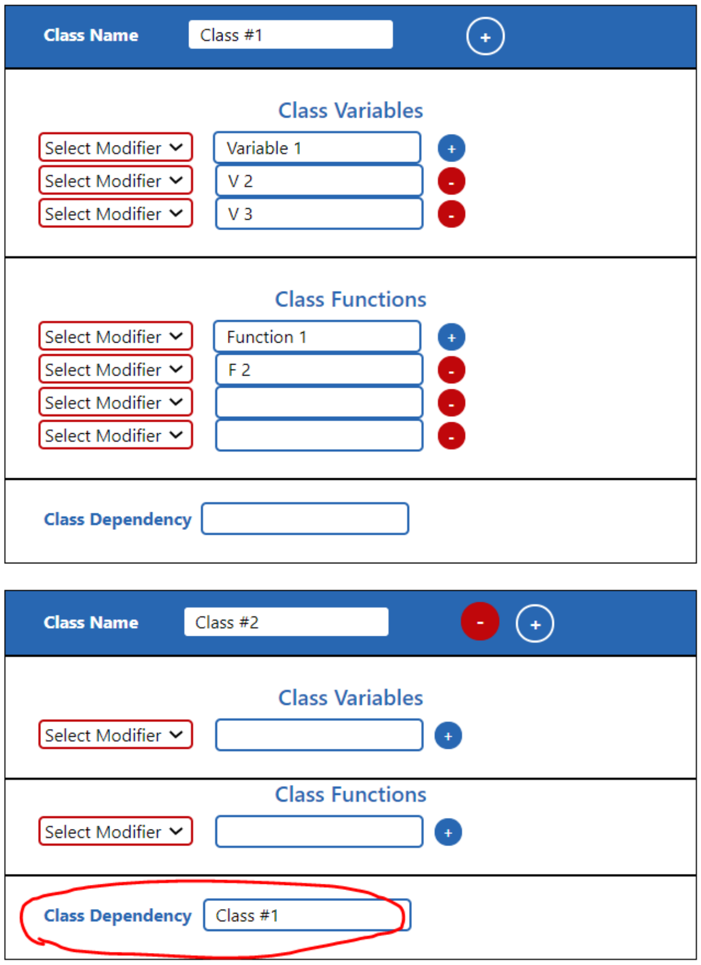


Class Name: Class #1

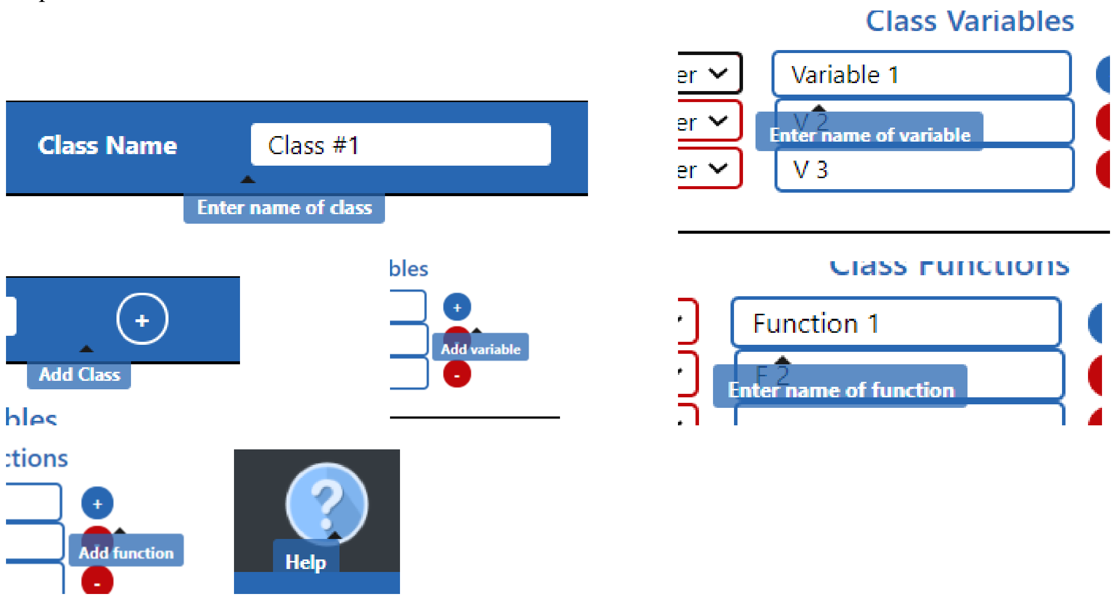
Class Variables: Variable 1, V 2, V 3

Class Functions: Function 1, F 2

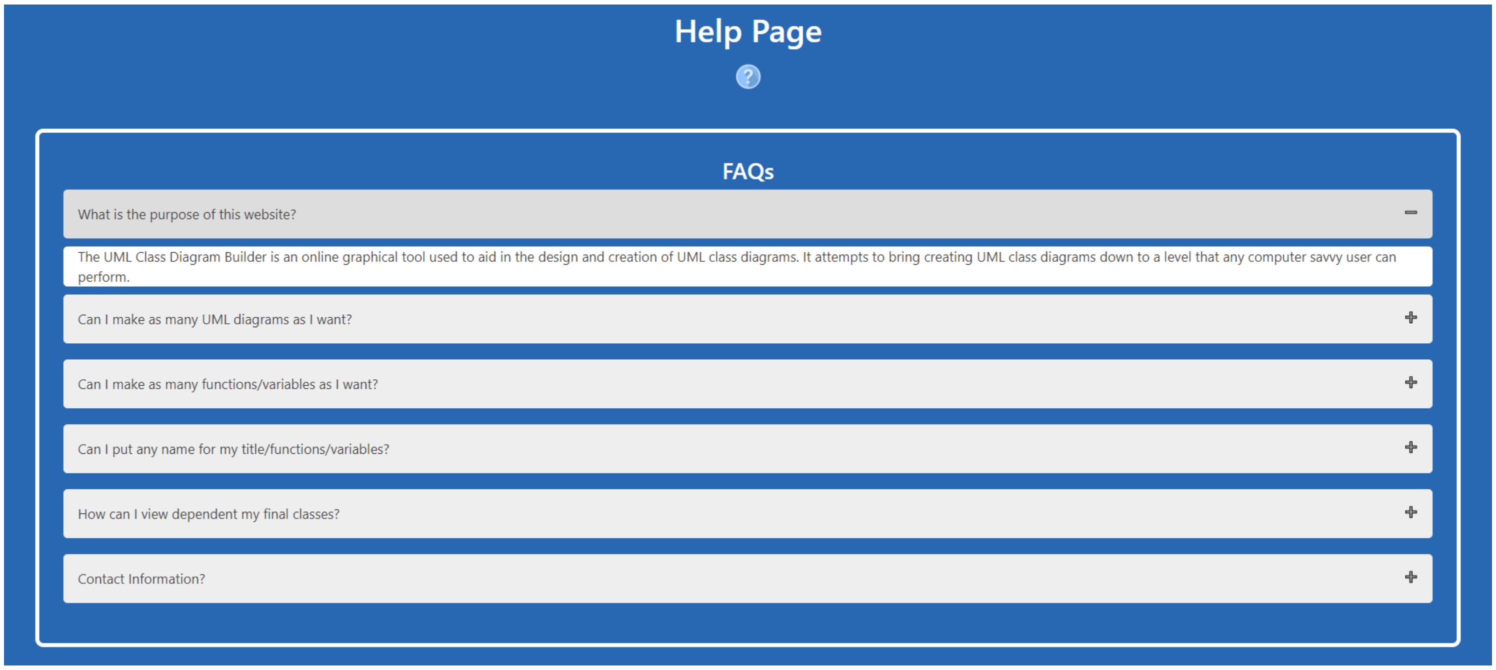
Requirement #005: Pass



Requirement #006: Pass



Requirement #007: Pass



Help Page

FAQs

What is the purpose of this website?

The UML Class Diagram Builder is an online graphical tool used to aid in the design and creation of UML class diagrams. It attempts to bring creating UML class diagrams down to a level that any computer savvy user can perform.

Can I make as many UML diagrams as I want?

Can I make as many functions/variables as I want?

Can I put any name for my title/functions/variables?

How can I view dependent my final classes?

Contact Information?

Requirement #008: Pass

All the buttons go to the correct page, and you don't get "stuck" on a page. Every Add and Minus button works perfectly, with no visual errors. The program is easy and simple to use.