Strategy as a Portfolio of Real Options

A dissertation submitted in partial fulfilment of

the requirements for the degree of

BACHELOR OF ENGINEERING in Computer Science

in

The Queen's University of Belfast

By

Ryan Neill

30th April 2019

SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING

and COMPUTER SCIENCE

CSC3002 –COMPUTER SCIENCE PROJECT

Dissertation Cover Sheet:

A signed and completed cover sheet must accompany the submission of the Software Engineering dissertation submitted for assessment.

Work submitted without a cover sheet will NOT be marked

Student Name: Ryan Neill   Student Number: 40133958

Project Title: Strategy as a Portfolio of Real Options

Supervisor: Desmond Greer

Declaration of Academic Integrity:

Before submitting your dissertation please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.

By submitting your dissertation you declare that you have completed the tutorial on plagiarism at http://www.qub.ac.uk/cite2write/introduction5.html and are aware that it is an academic offence to plagiarise. You declare that the submission is your own original work. No part of it has been submitted for any other assignment and you have acknowledged all written and electronic sources used.

Student's Signature: Ryan Neill   *Date of submission:* 30/04/2019

Abstract:

My allocated final project is called Strategy as a Portfolio of Real Options, and is an exploration of the viability of applying Black Scholes options pricing model to agile software development. The purpose of this is to produce strategic information which can aid in the project management process regarding software investment. To this end, I have designed an application in C# .NET WPF to manage projects, builds, components and calculated options. This application produces data visualisations and reports which will act as the portfolio of calculated options for the purposes of strategy.

Contents:

1.0 Introduction and Problem Area:

The purpose of this project is to apply the concept of Real Options to software investment, in the context of managing a project using the agile software development process. As this project requires an output of a portfolio of Real Options for strategic purposes, I must design a system that not only calculates options but also provides facilities for planning strategy and for reporting. As the solution will be focused on strategy and not actual software development, it will be designed with a Product Owner in mind with the expectation that they will be somewhat versed in finance and the associated terminology.

1.1 Real Options Valuation:

To apply Real Options to the agile development process we must first define and examine the concept and in what context it's traditionally used. As defined in Wikipedia [2], a Real Option is;

- "*the right—but not the obligation—to undertake certain business initiatives, such as deferring, abandoning, expanding, staging, or contracting a capital investment project. For example, the opportunity to invest in the expansion of a firm's factory, or alternatively to sell the factory, is a real call or put option, respectively*".

In other words, a Real Option is a choice that management can make with respect to an investment opportunity. For this project I intend to use Real Options to represent potential components for an existing system, as software development using Agile is an iterative process and works in terms of a build and release schedule in the context of a live system. Therefore, the context in which we will be using Real Options is in expanding an existing system, and as such, Call Options will be explored rather than Put Options.

Call and Put Options are specific kinds of Real Options which can be defined as the following;

- A Call Option[3] is *"…a financial contract between two parties, the buyer and the seller of this type of option.[1] The buyer of the call option has the right, but not the obligation, to buy an agreed quantity of a particular commodity or financial instrument (the underlying) from the seller of the option at a certain time (the expiration date) for a certain price (the strike price). The seller (or "writer") is obligated to sell the commodity or financial instrument to the buyer if the buyer so decides. The buyer pays a fee (called a premium) for this right."*

- A Put Option[4] is *"...a stock market device which gives the owner the right, but not the obligation, to sell an asset (the underlying), at a specified price (the strike), by a predetermined date (the expiry or maturity) to a given party (the seller of the put). The purchase of a put option is interpreted as a negative sentiment about the future value of the underlying stock."*

In other words, a Call Option is concerned with the acquisition of a stock or asset while a Put Option is regarding the selling of one. As the context of this project is in the development of an existing system as a live service rather than the outright selling of software ownership, Call Options will be used in strategizing the expansion of this system.

As outlined in the project list [5], this system will be using Black-Scholes Options Pricing Model [6] to calculate the calls for the Real Options, which is the world's most well-known and used model for this purpose. That said, the model makes the following assumptions;

- The option is European and can only be exercised at expiration.
- No dividends are paid out during the life of the option.
- Markets are efficient (i.e., market movements cannot be predicted).
- There are no transaction costs in buying the option.
- The risk-free rate and volatility of the underlying are known and constant.
- The returns on the underlying are normally distributed.

The formulas for the model [7] are as follows;

$$C = S_0\, e^{-qt} * N(d_1) - X\, e^{-rt} * N(d_2)$$

$$d_1 = \frac{ln(\frac{S_0}{X}) + t\,(r - q + \frac{\sigma^2}{2})}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

- S0 = underlying price (£)
- X = strike price (£)
- σ = volatility
- r = continuously compounded risk-free interest rate
- q = continuously compounded dividend yield
- t = time to expiration (% of year)

1.2 Agile Software Development Process:

As the focus of this project is strategic level management decisions, the technical aspects and roles of the Agile Development Process are of little concern, however, it is very important to understand the role of the Product Owner in the scrum process, which is defined as [8]:

- "…a project's key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team. This is key to successfully starting any agile software development project. The agile product owner does this in part through the product backlog, which is a prioritized features list for the product. The product owner is commonly a lead user of the system or someone from marketing, product management or anyone with a solid understanding of users, the market place, the competition and of future trends for the domain or type of system being developed."

In short, the Product Owner is the software owner who is responsible purely for the business decisions in the project, which is their expected area of expertise. Their level of interaction with the planning of a sprint is limited to the product backlog of components they expect to be developed for their system and their prioritisation. This means that the Product Owner has control over the components to be developed in each build of the system, which will be accounted for in this project and will be the portfolio of Real Options to be outputted.

2.0 Solution Description and Software Requirements:

2.1 Minimum System Requirements:

- Operating System: Windows 10
- CPU: 2GHz +
- RAM: 2GB +
- Disk Space Free: 1GB +
- Display: 1280 x 800

2.2 Functional Requirements:

1. The system will allow the user to create a new project.
2. The system will present a list of selectable projects from the database.
3. The system will allow the user to open a selected project from a project list.
4. The system will allow the user to edit a selected project.
5. The system will allow the user to delete a selected project.
6. The system will allow the user to create a build for a selected project.
7. The system will present a list of selectable builds from the database for a selected project.
8. The system will allow the user to delete a build for a selected project.
9. The system will allow a user to produce an excel report containing the selected project's statistics.
10. The system will allow a user to open a selected build which will navigate to a build-specific page for that selected project.
11. The system will allow the user to create a new component.
12. The system will present a list of selectable components from the database.
13. The system will allow the user to edit a selected component.
14. The system will allow the user to delete a selected component.
15. The system will allow the user to create a single option from a selected component.
16. The system will allow the user to create a combined option from a selection of selectable components.
17. The system will calculate the call values and value-to-cost ratios for both single and combined options.
18. The system will present a list of selectable options from the database.
19. The system will allow the user to edit a selected option.
20. The system will allow the user to delete a selected option, making its constituent components available for selection again.

21. The system will allow the user to delete all the current options for a build, making their constituent components available for selection again.
22. The system will allow the user to plot the current non-completed options on an OptionsPlot graph. This graph will be dynamic, updating to match changes made in the options list.
23. The system will allow the user to plot the current non-completed options on a CallValuesChart graph. This graph will be dynamic, updating to match changes made in the options list.
24. The system will allow the user to set the maximum and minimum risk values. This will be reflected dynamically in the OptionsPlot graph.
25. The system will allow the user to save the currently selected graph as a .png file to a user selected folder.
26. The system will provide a list of filters specific to the currently selected graph.
27. The system will update the currently selected graph based on the checked options in the filter list.
28. The system will allow the user to edit the currently selected build.
29. The system will present the build's release date where appropriate.
30. The system will present the sum of the development days of all the projects in the options list. This value will dynamically reflect changes made in the options list.
31. The system will present the days left from the current date until the release date where appropriate.
32. The system will present the build's current budget.
33. The system will present the total cost-to-build amounts from the options list. This value will dynamically reflect changes made in the options list.
34. The system will allow a user to produce an excel report containing the selected build's options, separating them into 3 tabs; All Options, Not Completed Options, Completed Options.
35. The system will allow the user to navigate back to the projects page.

2.3 Non-Functional Requirements:

- SQL Express should be used as the database host, as it is free, well support, fully documented and is a Microsoft product (meaning it will be compatible with a .NET application).
- Entity Framework should be used to access the database in the application code, as it is the industry standard and is compatible with LINQ.

- LINQ should be used to query the database as it is the industry standard and functions using concise code.
- The system should make use of an easy to understand user interface, with informative tooltips to clear up ambiguity of function / meaning.
- The system will be presented with a professional colour scheme, which avoid high contrast and jarring colours.
- Should be run from a single .exe file for efficient start up.
- The code should be structured such that the pages and function classes are stored in their own respective folders, in order for ease of navigation through the code.
- Visual Studio 2017 should be used to develop the system as it is the most recent version of Visual Studio and is the industry standard for C# .NET application development.
- The system should be developed using WPF as this format provides ease of use and development of user interfaces via XAML.

2.4 User Characteristics:

- The user should have a background in finance as they are expected to have the role of Product Owner in the agile development process.
- The user will be expected to understand the concept of a Real Option.
- The user will be expected to define component volatility themselves along with a component's speculated value to the system.
- The user is not expected to have a background in programming or software development, so the technological jargon will be kept to an absolute minimum.

3.0 Design

3.1 Architectural Design:

User Interface Architecture:

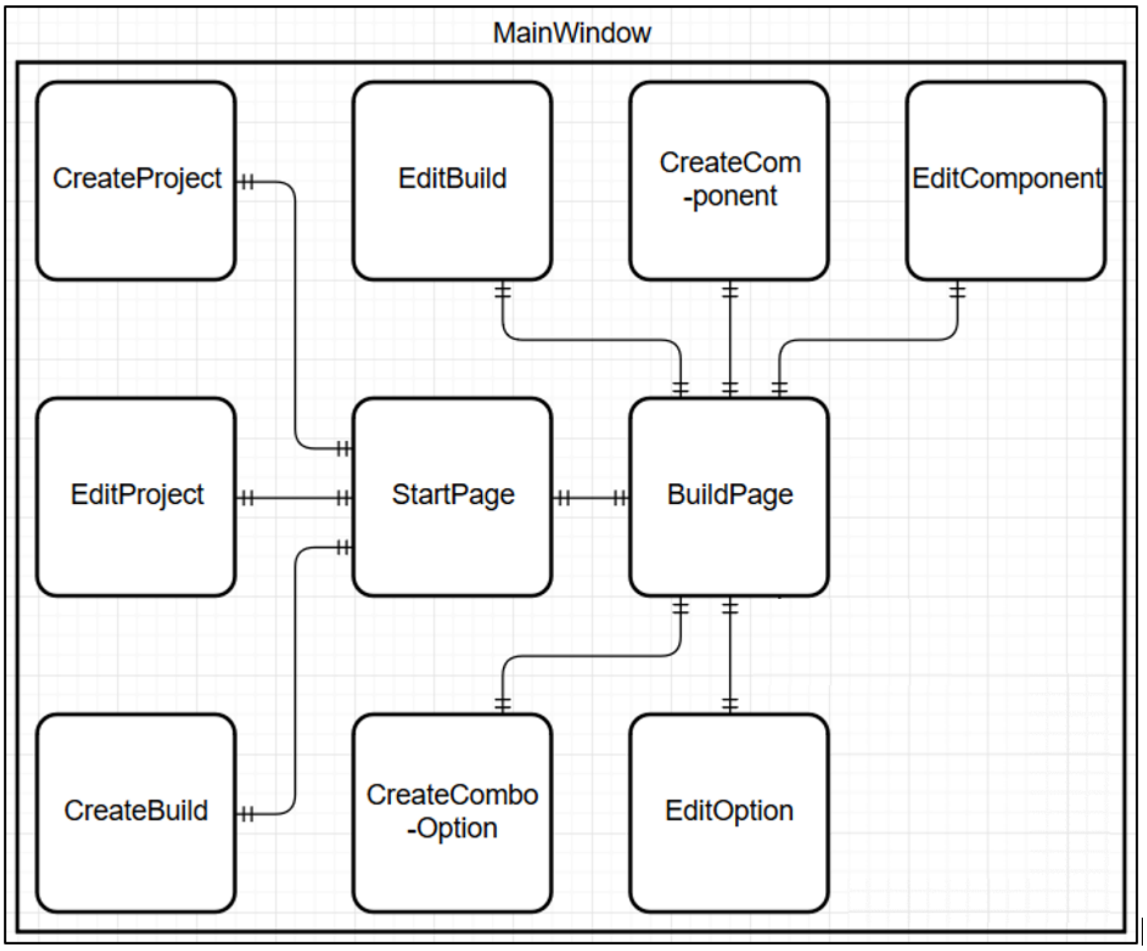
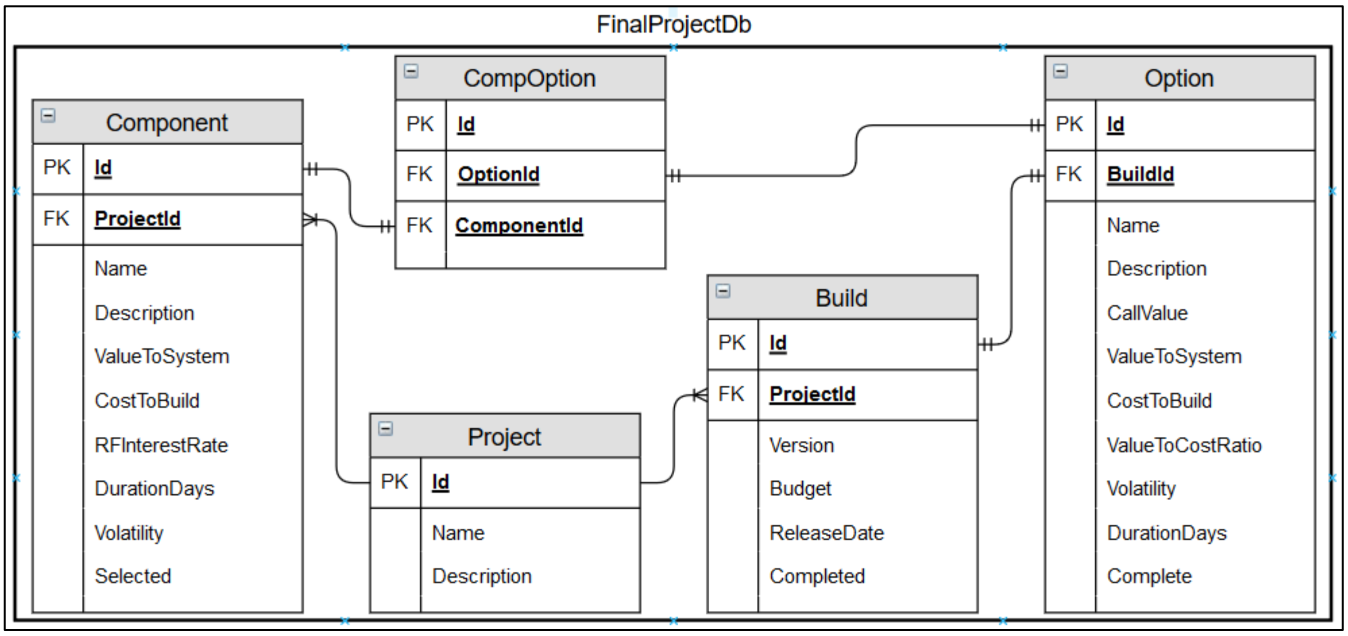
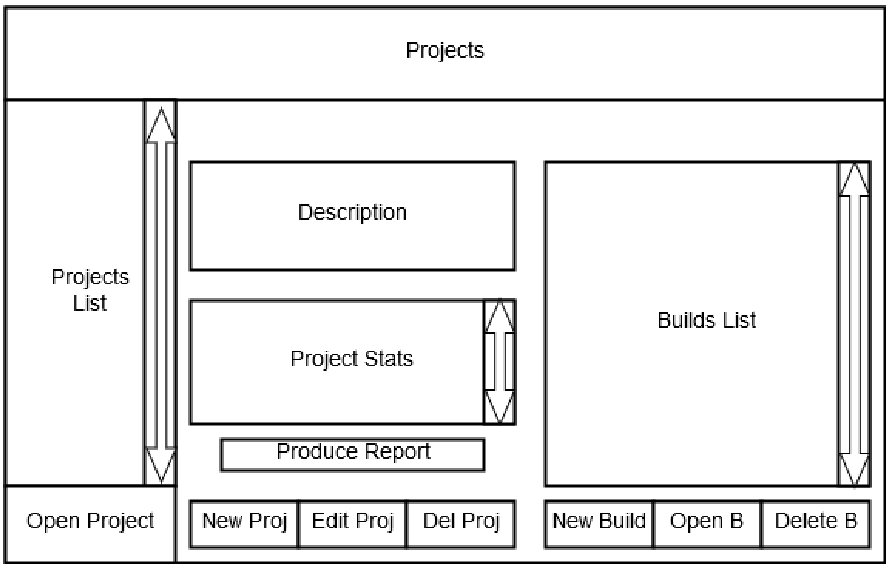


Table Design:

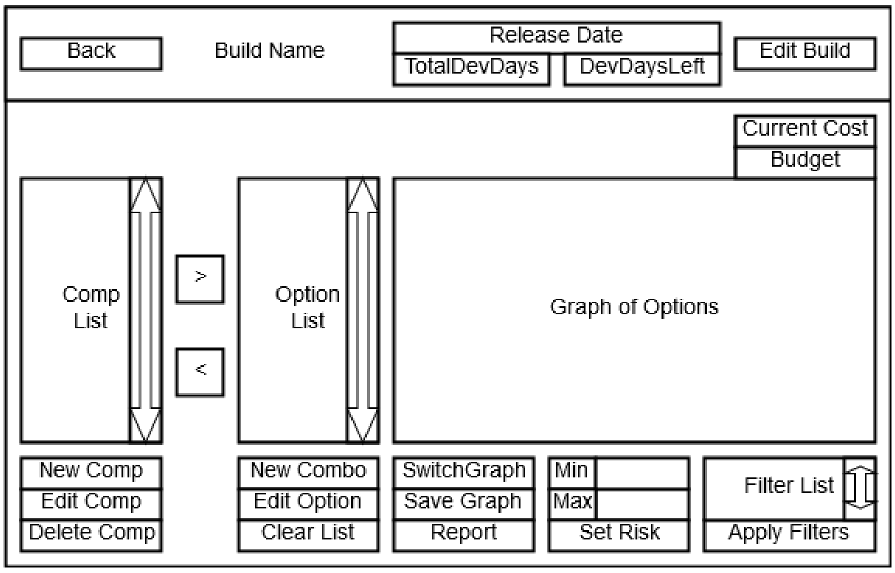


3.2 User Interface Design:

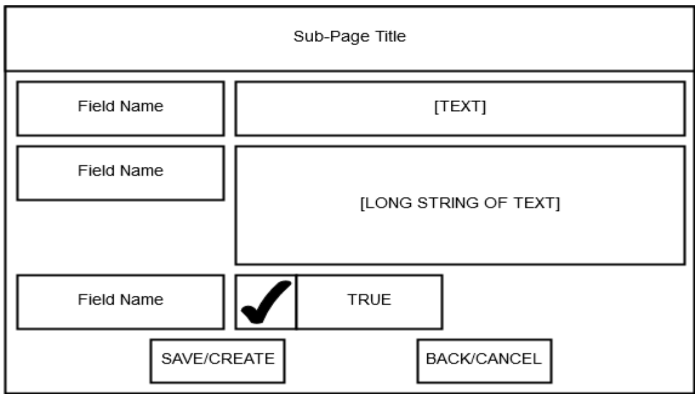
Start Page UI:



Build Page UI:

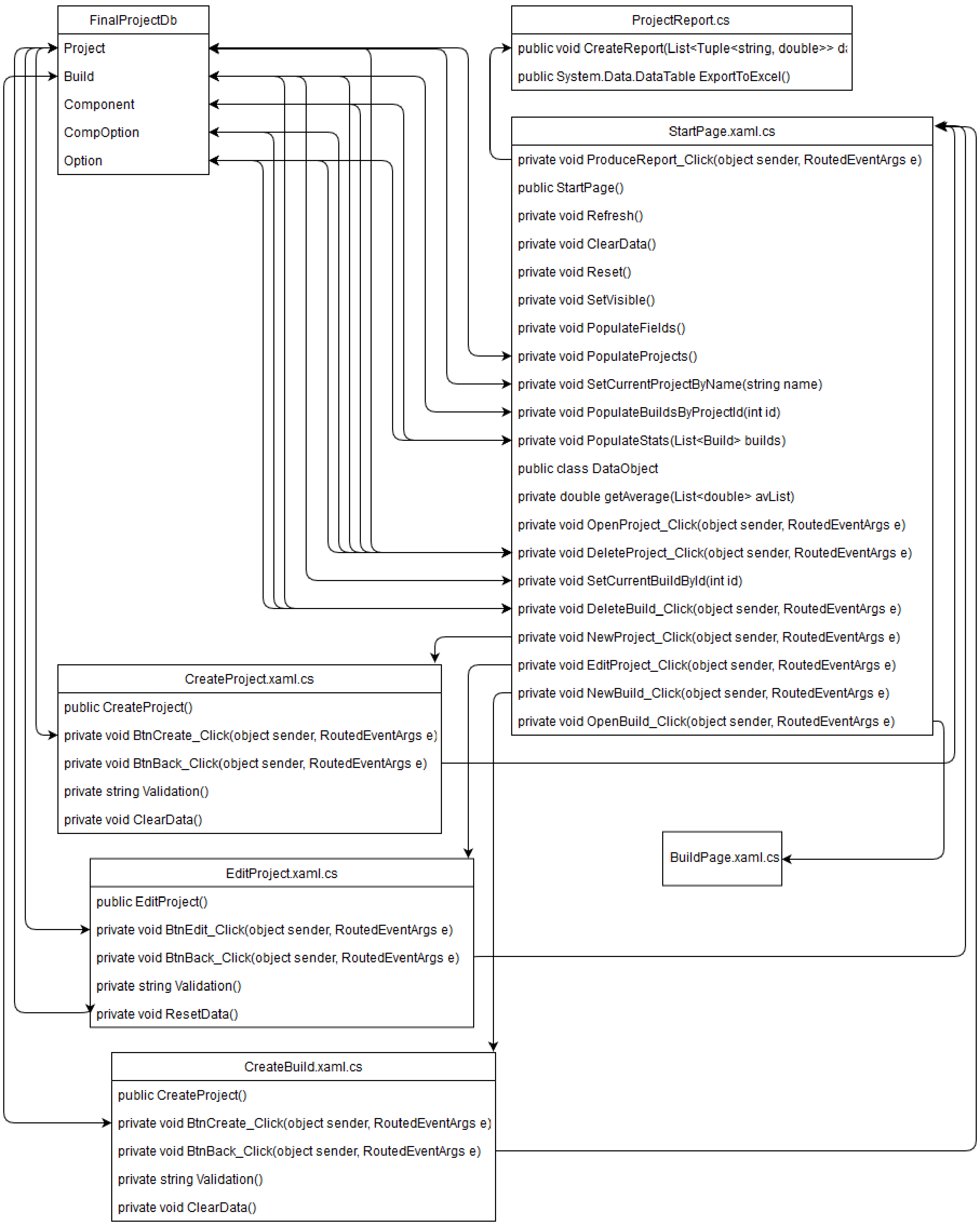


Generic Sub-Page UI Example:

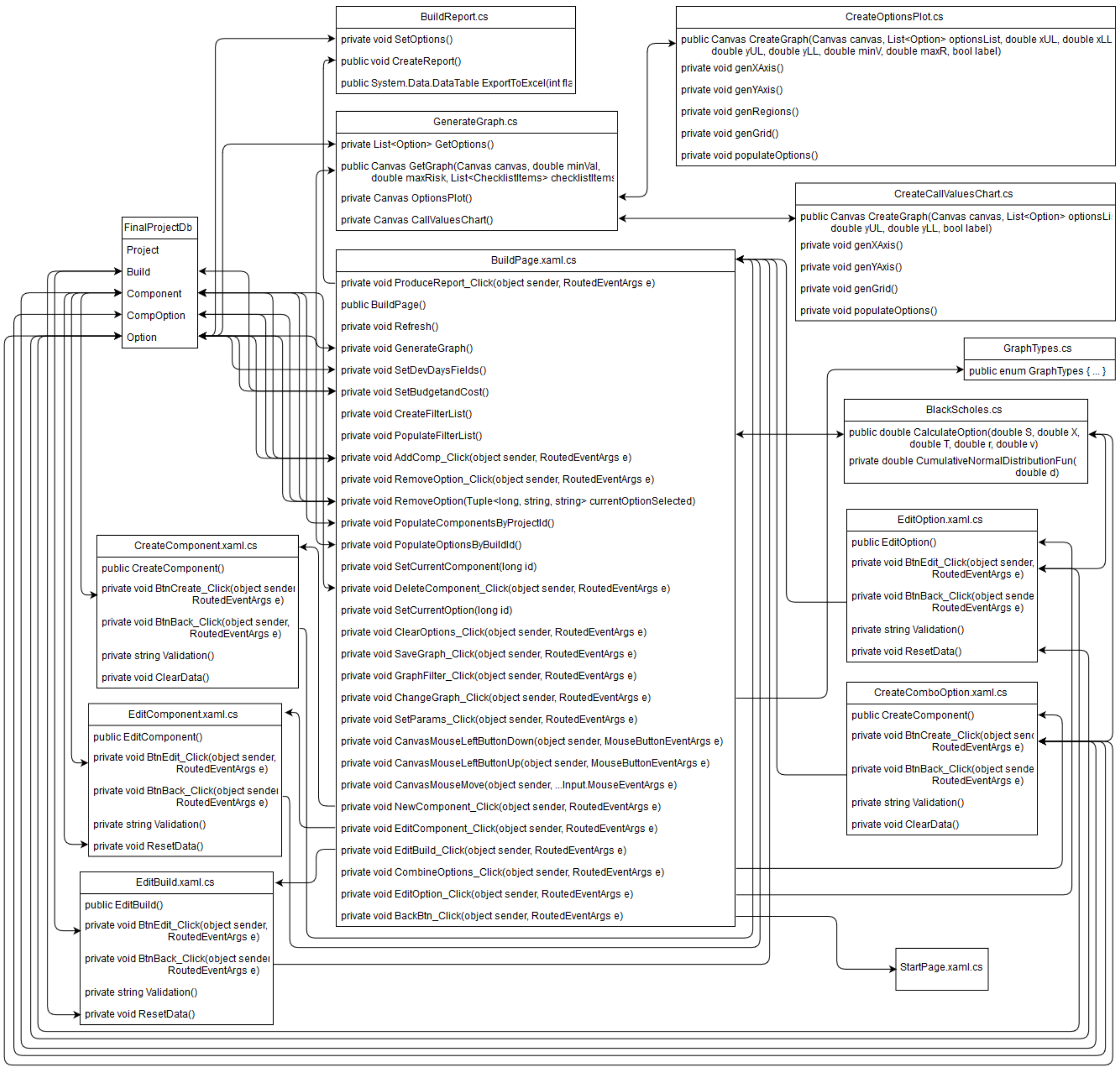


3.3 Software System Design:

Start Page Classes:


FinalProjectDb
Project
Build
Component
CompOption
Option
ProjectReport.cs
public void CreateReport(List<Tuple<string, double>> d
public System.Data.DataTable ExportToExcel()
StartPage.xaml.cs
private void ProduceReport_Click(object sender, RoutedEventArgs e)
public StartPage()
private void Refresh()
private void ClearData()
private void Reset()
private void SetVisible()
private void PopulateFields()
private void PopulateProjects()
private void SetCurrentProjectByName(string name)
private void PopulateBuildsByProjectId(int id)
private void PopulateStats(List<Build> builds)
public class DataObject
private double getAverage(List<double> avList)
private void OpenProject_Click(object sender, RoutedEventArgs e)
private void DeleteProject_Click(object sender, RoutedEventArgs e)
private void SetCurrentBuildById(int id)
private void DeleteBuild_Click(object sender, RoutedEventArgs e)
private void NewProject_Click(object sender, RoutedEventArgs e)
private void EditProject_Click(object sender, RoutedEventArgs e)
private void NewBuild_Click(object sender, RoutedEventArgs e)
private void OpenBuild_Click(object sender, RoutedEventArgs e)
CreateProject.xaml.cs
public CreateProject()
private void BtnCreate_Click(object sender, RoutedEventArgs e)
private void BtnBack_Click(object sender, RoutedEventArgs e)
private string Validation()
private void ClearData()
BuildPage.xaml.cs
EditProject.xaml.cs
public EditProject()
private void BtnEdit_Click(object sender, RoutedEventArgs e)
private void BtnBack_Click(object sender, RoutedEventArgs e)
private string Validation()
private void ResetData()
CreateBuild.xaml.cs
public CreateProject()
private void BtnCreate_Click(object sender, RoutedEventArgs e)
private void BtnBack_Click(object sender, RoutedEventArgs e)
private string Validation()
private void ClearData()

Build Page Classes:


BuildReport.cs
private void SetOptions()
public void CreateReport()
public System.Data.DataTable ExportToExcel(int fla
CreateOptionsPlot.cs
public Canvas CreateGraph(Canvas canvas, List<Option> optionsList, double xUL, double xLL
double yUL, double yLL, double minV, double maxR, bool label)
private void genXAxis()
private void genYAxis()
private void genRegions()
private void genGrid()
private void populateOptions()
GenerateGraph.cs
private List<Option> GetOptions()
public Canvas GetGraph(Canvas canvas, double minVal,
double maxRisk, List<ChecklistItems> checklistItems
private Canvas OptionsPlot()
private Canvas CallValuesChart()
CreateCallValuesChart.cs
public Canvas CreateGraph(Canvas canvas, List<Option> optionsLi
double yUL, double yLL, bool label)
private void genXAxis()
private void genYAxis()
private void genGrid()
private void populateOptions()
FinalProjectDb
Project
Build
Component
CompOption
Option
BuildPage.xaml.cs
private void ProduceReport_Click(object sender, RoutedEventArgs e)
public BuildPage()
private void Refresh()
private void GenerateGraph()
private void SetDevDaysFields()
private void SetBudgetandCost()
private void CreateFilterList()
private void PopulateFilterList()
private void AddComp_Click(object sender, RoutedEventArgs e)
private void RemoveOption_Click(object sender, RoutedEventArgs e)
private void RemoveOption(Tuple<long, string, string> currentOptionSelected)
private void PopulateComponentsByProjectId()
private void PopulateOptionsByBuildId()
private void SetCurrentComponent(long id)
private void DeleteComponent_Click(object sender, RoutedEventArgs e)
private void SetCurrentOption(long id)
private void ClearOptions_Click(object sender, RoutedEventArgs e)
private void SaveGraph_Click(object sender, RoutedEventArgs e)
private void GraphFilter_Click(object sender, RoutedEventArgs e)
private void ChangeGraph_Click(object sender, RoutedEventArgs e)
private void SetParams_Click(object sender, RoutedEventArgs e)
private void CanvasMouseLeftButtonDown(object sender, MouseButtonEventArgs e)
private void CanvasMouseLeftButtonUp(object sender, MouseButtonEventArgs e)
private void CanvasMouseMove(object sender, ...Input.MouseEventArgs e)
private void NewComponent_Click(object sender, RoutedEventArgs e)
private void EditComponent_Click(object sender, RoutedEventArgs e)
private void EditBuild_Click(object sender, RoutedEventArgs e)
private void CombineOptions_Click(object sender, RoutedEventArgs e)
private void EditOption_Click(object sender, RoutedEventArgs e)
private void BackBtn_Click(object sender, RoutedEventArgs e)
GraphTypes.cs
public enum GraphTypes { ... }
BlackScholes.cs
public double CalculateOption(double S, double X,
double T, double r, double v)
private double CumulativeNormalDistributionFun(
double d)
EditOption.xaml.cs
public EditOption()
private void BtnEdit_Click(object sender,
RoutedEventArgs e)
private void BtnBack_Click(object sende
RoutedEventArgs e)
private string Validation()
private void ResetData()
CreateComponent.xaml.cs
public CreateComponent()
private void BtnCreate_Click(object sende
RoutedEventArgs e)
private void BtnBack_Click(object sender,
RoutedEventArgs e)
private string Validation()
private void ClearData()
CreateComboOption.xaml.cs
public CreateComponent()
private void BtnCreate_Click(object senc
RoutedEventArgs e)
private void BtnBack_Click(object sende
RoutedEventArgs e)
private string Validation()
private void ClearData()
EditComponent.xaml.cs
public EditComponent()
private void BtnEdit_Click(object sender,
RoutedEventArgs e)
private void BtnBack_Click(object sende
RoutedEventArgs e)
private string Validation()
private void ResetData()
EditBuild.xaml.cs
public EditBuild()
private void BtnEdit_Click(object sender,
RoutedEventArgs e)
private void BtnBack_Click(object sende
RoutedEventArgs e)
private string Validation()
private void ResetData()
StartPage.xaml.cs

4.0 Implementation:

4.1 Software Used:

I decided to use C# .NET WPF via Visual Studio 2017 to program this system as it's the language which I'm most proficient in using and have most recently used over the course of my studies. Beyond this however, C# .NET is a powerful and well supported language and framework native to the Windows platform, that provides a range of tools I can use to efficiently and effectively develop a system for the purposes of this project. An example of this would be Entity Framework for database integration, which allows me to generate my models from an existing database, saving me much time and effort in implementing my database into the system. Windows Presentation Format (WPF) in particular, affords me a multitude of options for designing my user interface in XAML, such as the ability to design my UI using front end development techniques.

For the database software, I decided to use SQL Express as it is freely available, well supported and documented, and is easy to install and set up. SQL is also fast and is a long-established standard in industry, thus seemed like the natural choice especially considering how LINQ integrates with it using minimal code and in doing so counters the biggest issue with SQL (that being its difficulty in interfacing).

4.2 Key Implementation Decisions:

The primary purpose of the system is to plot components as a portfolio of real options in a manner that aids strategy, thus the key functions in my system are the Black Scholes model algorithm, the graph presentation and filtering, and the ability to save data and produce reports.

Black Scholes:

In order to apply this model to the context of an expanding system developed using build and release, I had to make to make some interpretations of key variables in order to make it fit. Stock Price (being the current value of the asset) was interpreted as being the estimated value in currency of the component to the system. Strike Price (being the agreed set purchase amount) was interpreted as being the estimated cost in currency of the component to develop. I have also set the Risk-Free Interest Rate to 1.0 and the dividend yield to 0 as the value-to-cost ration vs volatility is my main concern as it seemed to possess the most strategic value.

The Call Value calculated is usually the calculated fee to be paid in order to reserve the asset as an option to buy at the Strike Price at a later date, however, in this interpretation, the Call Option value and the Cost to Build fulfil the same function. However, the Call Option value is interpreted as scaling to the strength of the stock investment also factoring in volatility, therefore in my system it will be used as a representation of component investment strength rather than a direct pricing guide. Bar these alterations, the functioning of the model remains the same. In order to program it, I used C# code from YouTube as a source of inspiration [9] and researched other methods of applying cumulative natural distribution as a function for my algorithm. The result of my efforts was the class BlackScoles.cs which is accessed by the classes BuildPage.xaml.cs, CreateComboOption.xaml.cs, and EditOption.xaml.cs.

The OptionsPlot and CallValueChart Graphs:

The data visualisation functionality was the most time consuming and challenging part of the project, as I used no external libraries and built it all from the ground up. I created an enum class called GraphTypes.cs to aid in switching between the graphs and to provide scalability for future graph additions. This is used in the GenerateGraph.cs class which determines the type of plot to be created, retrieves option data from the database, and applies the correct filtering to the data so that the canvas object displays the correct data and appropriately.

This class then calls the CreateCallValuesChart.cs or CreateOptionsPlot.cs classes where appropriate, which mainly just handle the Canvas child object creation and manipulation, returning the completed graph result. Both classes apply x and y axis scaling using the upper and lower limit values in conjunction with the canvas height, width and present margin variable values. Using these parameters, the values are then plotted to their correct corresponding locations and visual formatting is applied. The default graph is the OptionsPlot, which handles most of the strategic heavy lifting, as it utilises regions which correspond to the options' viability for investment (as inspired by the diagram in the DG07 Project description [5].

Saving Graphs and Reporting:

The name of this project being Strategy as a Portfolio of Real Options, it is also essential that the system not only produce strategically relevant information, but also allow for this information to persist as a portfolio for presentation to interested parties. Therefore, it is necessary to provide a facility to save the generated graphs, which I implemented in the BuildPage.xaml.cs class. The system prompts the user to select an output folder and file name, then saves the graph as a PNG image file. After some calibration this proved to be a simple task, however, I felt that simply saving the images was insufficient as a strategy tool.

Thus I created functionality which appends selected data to an Excel object and saves the report to a user specified folder. To achieve this I constructed two separate class to produce two separate reports; ProjectReport.cs and BuildReport.cs. The former pulls data from the project stats datagrid in the StartPage and contains aggregate data regarding the selected project's overall performance. The latter produces a report containing all the Build's current options, filtering them into three worksheets; All Options, Not Completed Options, and Completed Options. The benefit of outputting the data in this manner as an excel document, is that the data can be used for other purposes not provided for within the system and can allow the user to analyse the data themselves.

Management of Project Records:

In order to have any strategy to represent with real options, it's imperative to actually have data and a means to manage it in the first place. As mentioned previously, the context of the strategy is regarding a software asset expanding itself through Agile build and release investment, so to this end I designed a management system which allows the user to add, edit and delete the main record types associated with Agile development, being Projects, Builds and Components. The record management functionality comprises the bulk of the system and is essential in order to appropriately categorise and calculate the Options which are the primary focus. This was achieved by designing the system and user interface around this concept, as is observable throughout.

5.0 Testing:

Func Req No:	Test Action / Data:	Result:	Expected:	Screenshot:
1	Open new project page	CreateProject page opens	YES	8.2 i)
	Create a new valid project	Project created prompt	YES	8.2 ii)
	Leave name blank	Error prompt	YES	8.2 iii)
	Enter 100 character name	Error prompt	YES	8.2 iv)
	Enter 200 character description	Error prompt	YES	8.2 v)
2	Project List contains new project	New project observed	YES	8.2 vi)
3	Click open project without a project selected	Error prompt	YES	8.2 vii)
	Select and open new project	Project menu UI visible	YES	8.2 viii)
4	Save with blank name	Error prompt	YES	8.2 iii)
	Save with 100 character name	Error prompt	YES	8.2 iv)

	Save with 200 character description.	Error prompt	YES	8.2 v)
	Save a valid project	Project modified prompt	YES	8.2 viii)
5	Successfully delete a project	Project deleted prompt	YES	8.2 ix)
6	Open new build page	CreateBuild page opens	YES	8.2 x)
	Create a new valid build	Build created prompt	YES	8.2 xi)
	Leave version blank	Error prompt	YES	8.2 xii)
	Leave budget blank	Error prompt	YES	8.2 xiii)
	Enter string as budget	Error prompt	YES	8.2 xiii)
	Enter minus value budget	Error prompt	YES	8.2 xiv)
	Enter 100 character version	Error prompt	YES	8.2 xv)
	Leave release date blank	Build created prompt	NO	8.2 xvi)
	Enter string as release date	Error prompt	YES	8.2 xvi)
	Enter value as release date	Error prompt	YES	8.2 xvi)
7	Build List contains new build	New build observed	YES	8.2 xvii)
8	Click delete build without a build selected	Error prompt	YES	8.xviii)
	Successfully delete a build	Build deleted prompt	YES	8.2 xix)
9	Generate a project stats report	Report created prompt	YES	8.2 xxi)
10	Click open build without a build selected	Error prompt	YES	8.2 xxii)
	Navigate to selected build page	BuildPage opens	YES	8.2 xxiii)
11	Open new component page	CreateComponent opens	YES	8.2 xxiv)
	Leave name blank	Error prompt	YES	8.2 iii)
	Enter 100 character name	Error prompt	YES	8.2 iv)
	Enter 200 character description	Error prompt	YES	8.2 v)

	Leave estimated value blank	Error prompt	YES	8.2 xxv)
	Enter string as estimated value	Error prompt	YES	8.2 xxv)
	Enter minus value for estimated value	Error prompt	YES	8.2 xxvi)
	Leave estimated cost blank	Error prompt	YES	8.2 xxvii)
	Enter string as estimated cost	Error prompt	YES	8.2 xxvii)
	Enter minus value for estimated cost	Error prompt	YES	8.2 xxviii)
	Leave duration days blank	Error prompt	YES	8.2 xxix)
	Enter string as duration days	Error prompt	YES	8.2 xxix)
	Enter minus value for duration days	Error prompt	YES	8.2 xxx)
	Leave volatility blank	Error prompt	YES	8.2 xxxi)
	Enter string as volatility	Error prompt	YES	8.2 xxxi)
	Enter minus value for volatility	Error prompt	NO	8.2 xxxii)
12	Component List contains new component	New component observed	YES	8.2 xxxiii)
13	Click edit component without a component selected	Error prompt	YES	8.2 xxxiv)
	Save name as blank	Error prompt	YES	8.2 iii)
	Save 100 character name	Error prompt	YES	8.2 iv)
	Save 200 character description	Error prompt	YES	8.2 v)
	Save estimated value as blank	Error prompt	YES	8.2 xxv)
	Save string as estimated value	Error prompt	YES	8.2 xxv)
	Save minus value for estimated value	Error prompt	YES	8.2 xxvi)
	Save estimated cost as blank	Error prompt	YES	8.2 xxvii)
	Save string as estimated cost	Error prompt	YES	8.2 xxvii)

	Save minus value for estimated cost	Error prompt	YES	8.2 xxviii)
	Save duration days as blank	Error prompt	YES	8.2 xxix)
	Save string as duration days	Error prompt	YES	8.2 xxix)
	Save minus value for duration days	Error prompt	YES	8.2 xxx)
	Save volatility as blank	Error prompt	YES	8.2 xxxi)
	Save string as volatility	Error prompt	YES	8.2 xxxi)
	Save minus value for volatility	Error prompt	NO	8.2 xxxii)
14	Click delete component without a component selected	Error prompt	YES	8.2 xxxv)
	Successfully delete a component	Component deleted prompt	YES	8.2 xxxvi)
15	Click add component to option button with no component selected	Error prompt	YES	8.2 xxxiv)
	Add selected component to option list	Component removed from list, new option in option list	YES	8.2 xxxvii)
16	Open the combined option creation page	CreateComboOption page opens	YES	8.2 xxxviii)
	Try to create a blank option	Error prompt	YES	8.2 xxxix)
	Create a valid combo option	ComboOption created prompt	YES	8.2 xL)
17	New option has call and value-to-ratio calculated	Values observed	YES	8.2 xLi)
18	Option List contains new option	New option observed	YES	8.2 xLi)
19	Click edit option without an option selected	Error prompt	YES	8.2 xLii)
	Save name blank	Error prompt	YES	8.2 iii)
	Save 100 character name	Error prompt	YES	8.2 iv)
	Save 200 character description	Error prompt	YES	8.2 v)

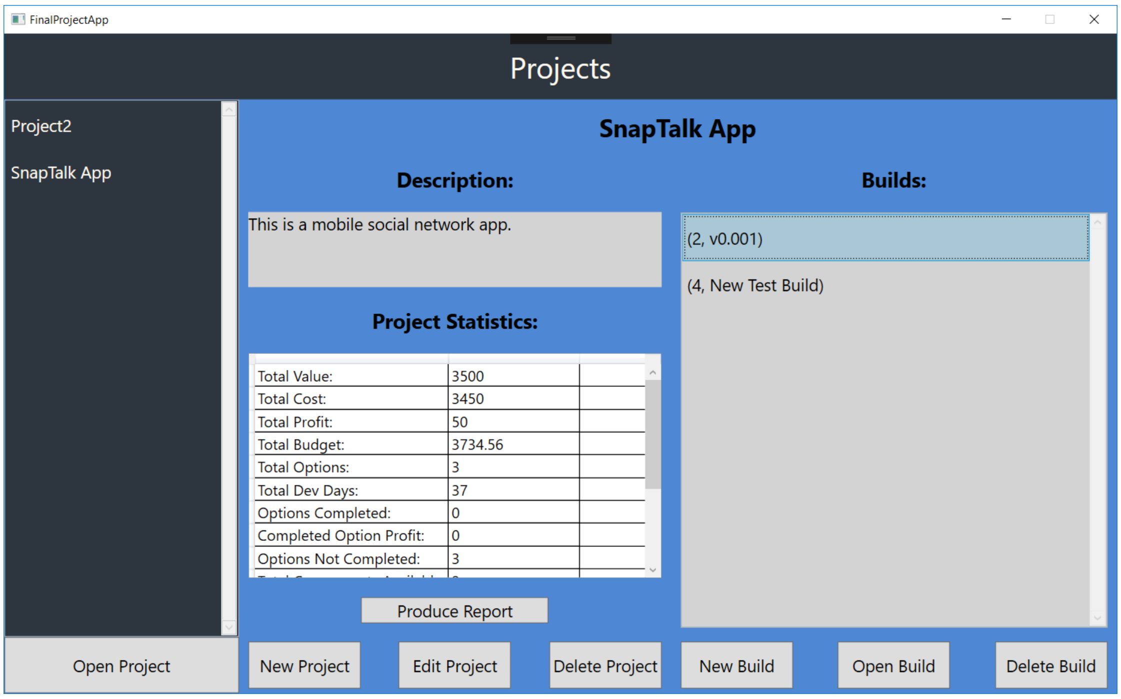
	Save duration days as blank	Error prompt	YES	8.2 xxix)
	Save string as duration days	Error prompt	YES	8.2 xxix)
	Save minus value for duration days	Error prompt	YES	8.2 xxx)
	Set option as completed	Text change observed	YES	8.2 xLiii)
	Save valid option	Option modified prompt	YES	8.2 xLiv)
20	Click delete option without an option selected	Error prompt	YES	8.2 xLii)
	Successfully delete an option	Option deleted prompt	YES	8.2 xLv)
	Component(s) constituting the deleted option are available for selection in components list	Components observed	YES	8.2 xLvi)
21	Clear options delete all options in options list.	Options deleted prompt	YES	8.2 xLvii)
	All constituent components of the deleted options are available for selection in components list	Components observed	YES	8.2 xLvi)
22	Generate OptionsPlot of options	OptionsPlot observed	YES	8.2 xLviii)
	No completed options in graph	None observed	YES	8.2 xLix)
	Graph updates as changes are made to options list	Changes observed	YES	8.2 L)
23	Generate CallValuesChart of options	CallValuesChart observed	YES	8.2 Li)
	No completed options in graph	None observed	YES	8.2 Lii)
	Graph updates as changes are made to options list	Changes observed	YES	8.2 Liii)
24	Set min risk reflected in OptionsPlot graoh	Changes observed	YES	8.2 Liv)
	Min risk set as minus value	Error prompt	YES	8.2 Lv)

	Min risk set as string	Error prompt	YES	8.2 Lvi)
	Min risk set as greater than the max risk value	Error prompt	YES	8.2 Lvii)
	Min risk and max risk are set as equal values	Data is valid	NO	8.2 Lviii)
	Set max risk reflected in OptionsPlot graoh	Changes observed	YES	8.2 Lviii)
	Max risk set as minus value	Error prompt	YES	8.2 Lix)
	Max risk set as string	Error prompt	YES	8.2 Lx)
	Max risk set as less than the min risk value	Error prompt	YES	8.2 Lix)
25	Save graph to selected folder	Report created prompt	YES	8.2 Lxi)
26	Filter list generated for graphs	Filter list observed	YES	8.2 Liv)
	Filter list changes depending on currently selected graph	Changes observed	YES	8.2 Lxii)
27	Checked filter options reflected in the graph	Changes observed	YES	8.2 Lxii)
28	Open edit build page	EditBuild page opens	YES	8.2 x)
	Save a valid build	Build modified prompt	YES	8.2 xi)
	Save version as blank	Error prompt	YES	8.2 xii)
	Save budget as blank	Error prompt	YES	8.2 xiii)
	Save budget as string	Error prompt	YES	8.2 xiii)
	Save minus value for budget	Error prompt	YES	8.2 xiv)
	Save 100 character version	Error prompt	YES	8.2 xv)
	Save release date as blank	Build modified prompt	NO	8.2 xvi)
	Save release date as string	Error prompt	YES	8.2 xvi)
	Save release date as value	Error prompt	YES	8.2 xvi)

29	New release date presented on build page	Changes observed	YES	8.2 Lxiii)
	Null release date	Error Prompt	NO	8.2 Lxiv)
30	Total development days of options displayed	Changes observed	YES	8.2 Lxiii)
	Total dev days updates dynamically in response to option list changes	Changes observed	YES	8.2 Lxv)
	Empty options list	Defaults to 0	YES	8.2 Lxv)
31	Days left from current until release date displayed	Changes observed	YES	8.2 Lxiii)
	Null release date	Null date not possible	NO	8.2 Lxiv)
	Release date current date or in the past	Defaults to DUE	YES	8.2 Lxv)
32	Current budget displayed	Correct budget observed	YES	8.2 Lxvi)
33	Total cost of options displayed	Correct cost observed	YES	8.2 Lxvi)
	Total cost updates dynamically in response to option list changes	Changes observed	YES	8.2 Lxvii)
	Empty options list	Defaults to 0	YES	8.2 Lxvii)
34	Generate a build options report	Report created prompt	YES	8.2 xxi)
35	Navigate to the start page	StartPage opens	YES	8.2 Lxviii)

6.0 System Evaluation and Experimental Results:

Addressing all the key requirements of the project, the system is robust containing no known crashes, it's lightweight, and is designed with an intuitive user interface. The system applies the concept of Real Options to software investment successfully in the context of managing Agile developed software. It also produces outputs in the form of reports and saved graphs which act as portfolios of the calculated options, demonstrably aiding in strategy in the business sense. As a result, this experiment produces an exciting insight into what may be possible, in improving the efficiency and efficacy of decision making for product owners managing live system projects.



Project2

SnapTalk App

SnapTalk App

Description:

This is a mobile social network app.

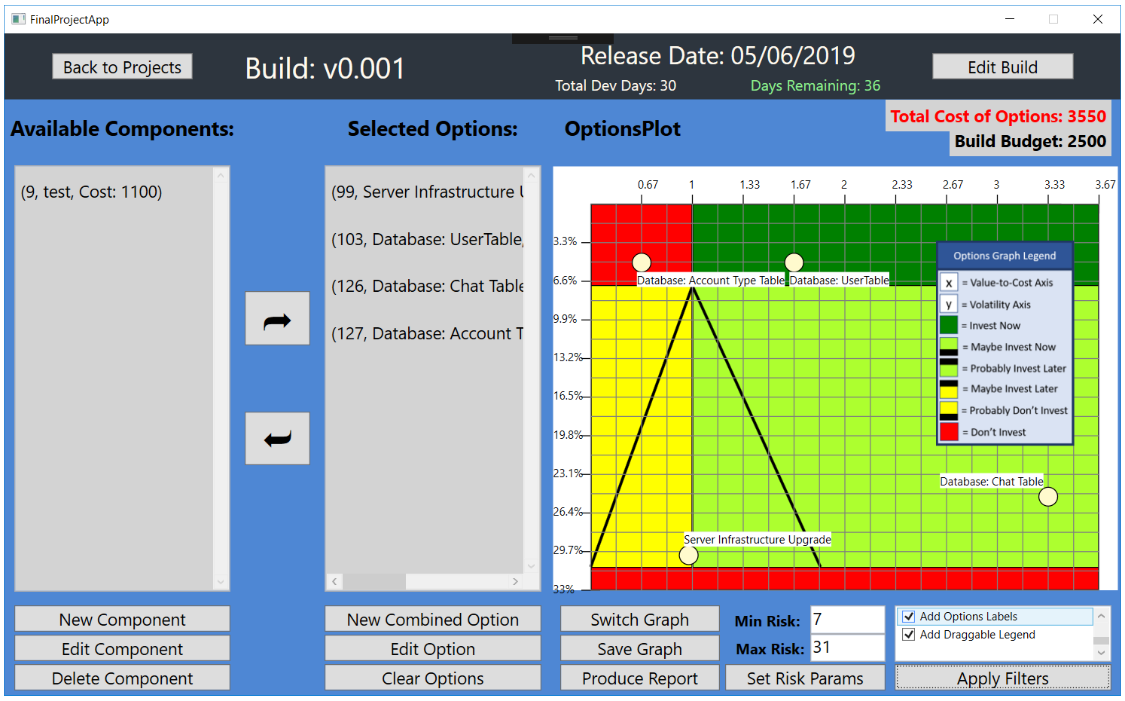
Builds:

(2, v0.001)

(4, New Test Build)

Project Statistics:

Total Value:	3500	
Total Cost:	3450	
Total Profit:	50	
Total Budget:	3734.56	
Total Options:	3	
Total Dev Days:	37	
Options Completed:	0	
Completed Option Profit:	0	
Options Not Completed:	3	



Build: v0.001

Release Date: 05/06/2019

Total Dev Days: 30

Days Remaining: 36

Total Cost of Options: 3550

Build Budget: 2500

Available Components:

(9, test, Cost: 1100)

Selected Options:

(99, Server Infrastructure U

(103, Database: UserTable,

(126, Database: Chat Table

(127, Database: Account T

Min Risk: 7

Max Risk: 31

✓ Add Options Labels

✓ Add Draggable Legend

6.1 Positives:

- User can easily create, edit and delete projects, builds, components and options.
- Projects, builds, components and options are listed clearly, and the buttons used for their interactions are logically laid out and labelled.
- The system generates and saves accurate reports and images to a folder of the user's choosing.
- The provides a range of data on both the build and project level.
- System calculations are quick and experience no noticeable slowdown as well as accurate within the range of expectation, being perceived to produce no anomalous results.
- The system provides professional graphs that accurately represent the data displayed.
- Filters and legends are provided as options to the user to further refine results.
- System provides release date and development duration data to aid in managing projects.
- System provides budget and cost data to aid in project spending.
- System has good usability, with seamless navigation throughout
- The system makes use of an easy to understand user interface with informative tooltips.
- The system is presented with a professional colour scheme, which avoids high contrast and jarring colours.
- Post-setup, is easy to run as it's executed from a single file.
- The code is clear, easy to read, well commented and logically laid out.

6.2 Negatives:

- Some validation error prompts present the Exception message rather than a customised feedback text.
- The titles of the graphs are set to their Enum codes rather than proper English, and thus are in camel case.
- The Build table contains a redundant field in the form of MaxVolatility, which is no longer in use (this isn't visible to the user however).
- If options presented on the graph are too close and the option label filter is checked, the option labels can obscure one another.
- Filter settings are unchecked after saving the graph as an image.
- Due to time constraints, only two graphs could be made and the reports are tables containing no diagrams.
- Currency symbols aren't used in the program, and instead values are simply doubles with the values rounded to 2 decimal places (but not always). Therefore, labelling or context is required to know what is and isn't currency.

6.3 Future Improvements:

- System is scalable and so more graphs can be easily added, such as a Gantt chart.
- Develop further the excel reports to include more data and visualisations.
- Add currency symbols and functionality for currency conversion.
- Design user interface window and elements to be resizable, as it's currently at a locked window size.
- Implementation of machine learning of success and failure of certain investments to refine call option calculation. This would be achieved by making use of the Risk-Free Interest Rate and Dividend Yield variables not used in this system.
- Implement archiving and a means of separating old data from current data.

6.4 Development Process:

- The agile development process approach to creating this system is still, in my view, the ideal development approach.
- However, in the context of final year in a computer science degree at Queen's, the modules are set up in such a way that it is appallingly difficult to manage time effectively, as they demand more time than is available without regard for one another, meaning sacrifices are inevitable.
- It is often necessary to abandon work on a project entirely in order to stem the flow of other obligations, irrespective of time management techniques applied.
- As such, the greatest weakness of this project's development was the difficulty in maintaining any consistency in workflow, causing development to take on a "fits and starts" approach.
- During lull periods between exams and deadlines however, development is efficient and productive, but the nature of having to make up for lost time makes organising supervisor meetings impractical, and instead time is better spent towards development. As such, the Agile approach breaks down in these circumstances.
- For this project a repository was not used for development and was only used for accessibility of the code and relevant software.

7.0 References:

[1]. R. Neill, “FinalYearProject,” [Online]. Available: https://github.com/Hannibal95/FinalYearProject.git. [Accessed 04 2019].

[2]. Wikipedia, “Real Options Valuation,” [Online]. Available: https://en.wikipedia.org/wiki/Real_options_valuation. [Accessed 04 2019].

[3]. Wikipedia, “Call Option,” [Online]. Available: https://en.wikipedia.org/wiki/Call_option. [Accessed 04 2019].

[4]. Wikipedia, “Put Option,” [Online]. Available: https://en.wikipedia.org/wiki/Put_option. [Accessed 04 2019].

[5]. D. Greer, “Project List 2018-19,” [Online]. Available: https://learning.qol.qub.ac.uk/2181/CSC/3002-FYR-QUB/Resources/Project%20List%202018-19.pdf [Accessed 04 2019].

[6]. J. Folger, “Options Pricing: Black-Scholes Model,” [Online]. Available: https://www.investopedia.com/university/options-pricing/black-scholes-model.asp. [Accessed 04 2019].

[7]. Macroption, “Black-Scholes Formula (d1, d2, Call Price, Put Price, Greeks),” [Online]. Available: https://www.macroption.com/black-scholes-formula/. [Accessed 04 2019].

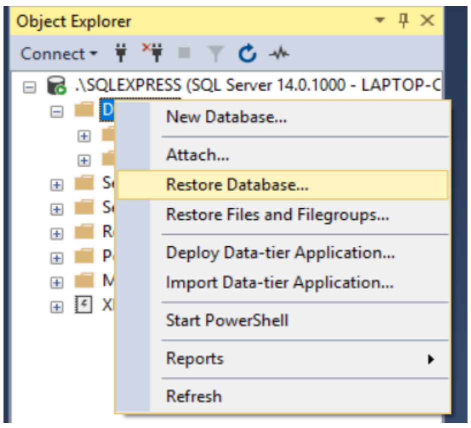
[8]. M. G. Software, “Product Owner”, [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum/roles/product-owner. [Accessed 04 2019].

[9]. B. Byrne, “Black Scholes C# sharp code for Windows Form,” [Online]. Available: https://www.youtube.com/redirect?redir_token=id9YOeu20Iem3Ben5u20dq4zsGN8MTU1NjcyNjAxN0AxNTU2NjM5NjE3&q=https%3A%2F%2F1drv.ms%2Fw%2Fs%21AsWcG8zbg1hc2gCpavJMmi-WmId0&event=video_description&v=g_J9aQ1lDuw. [Accessed 04 2019].

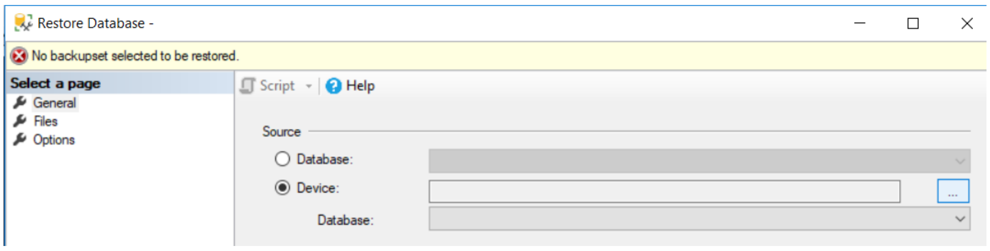
8.0 Appendices:

8.1 Database Installation Guide:

- Download and install the latest version of SQLExpress.
- Download and install the latest version of SQL Server Management Studio (SSMS).
- Open SSMS and connect to the SQLExpress server by clicking Connect.
- Right click on Databases and select Restore database

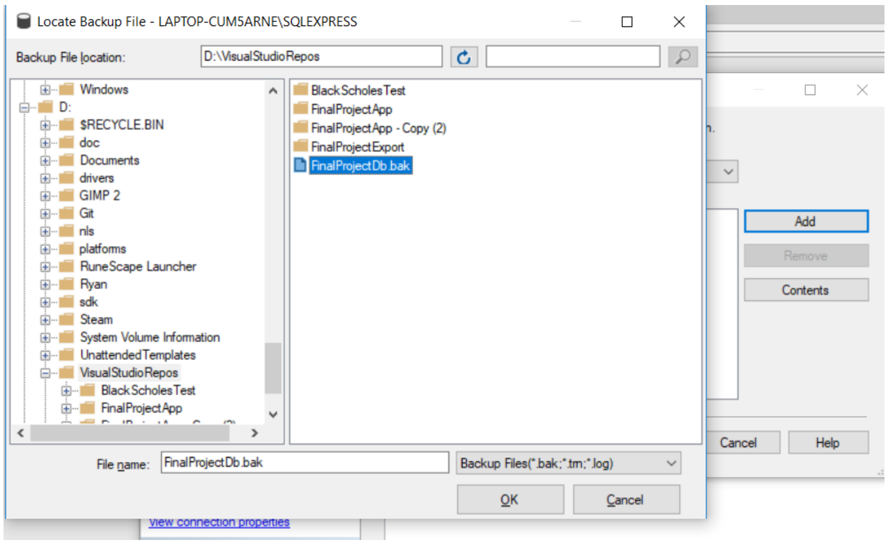
- 
Object Explorer
Connect
.\SQLEXPRESS (SQL Server 14.0.1000 - LAPTOP-C
New Database...
Attach...
Restore Database...
Restore Files and Filegroups...
Deploy Data-tier Application...
Import Data-tier Application...
Start PowerShell
Reports
Refresh

- Select Device and click on the "…" ellipsis

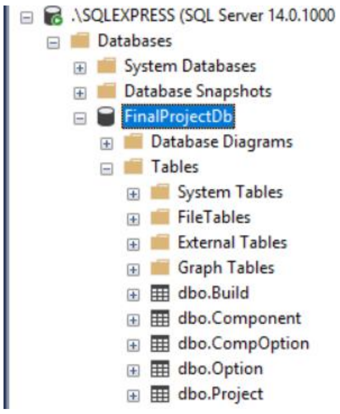
- 

No backupset selected to be restored.

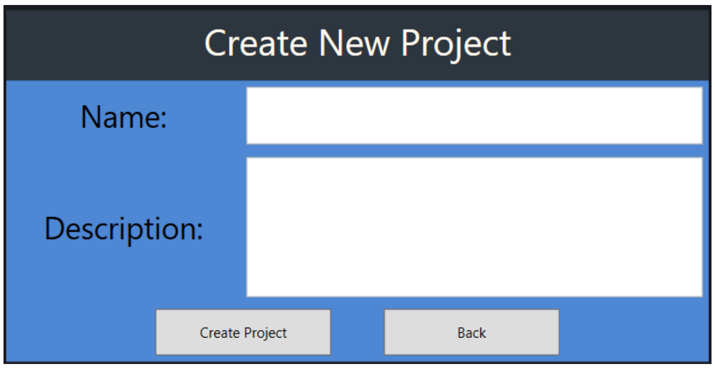
- Click add, navigate to where you are storing the .bak file, select it, and click OK.

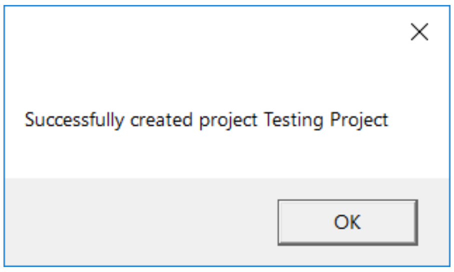
- 

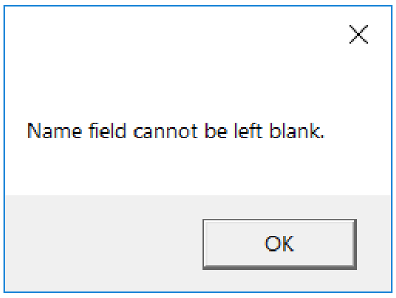
- Click OK for everything from here on out and the database should be added successfully.

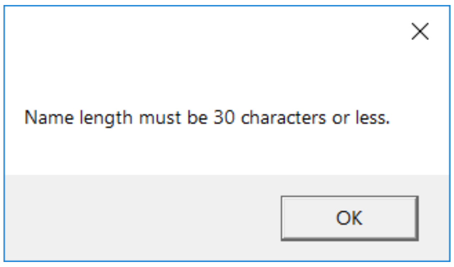
- 

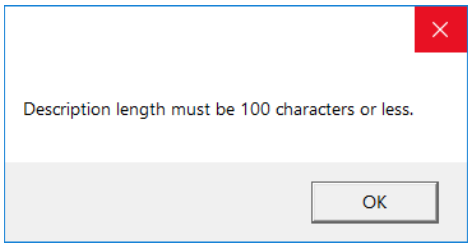
8.2 Testing Screenshots:

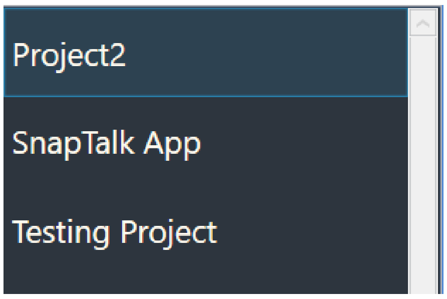
i) 

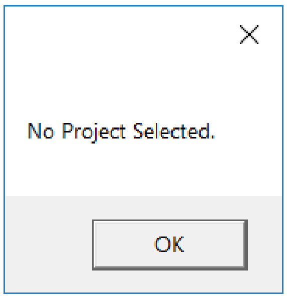
ii) 
Successfully created project Testing Project

iii) 
Name field cannot be left blank.

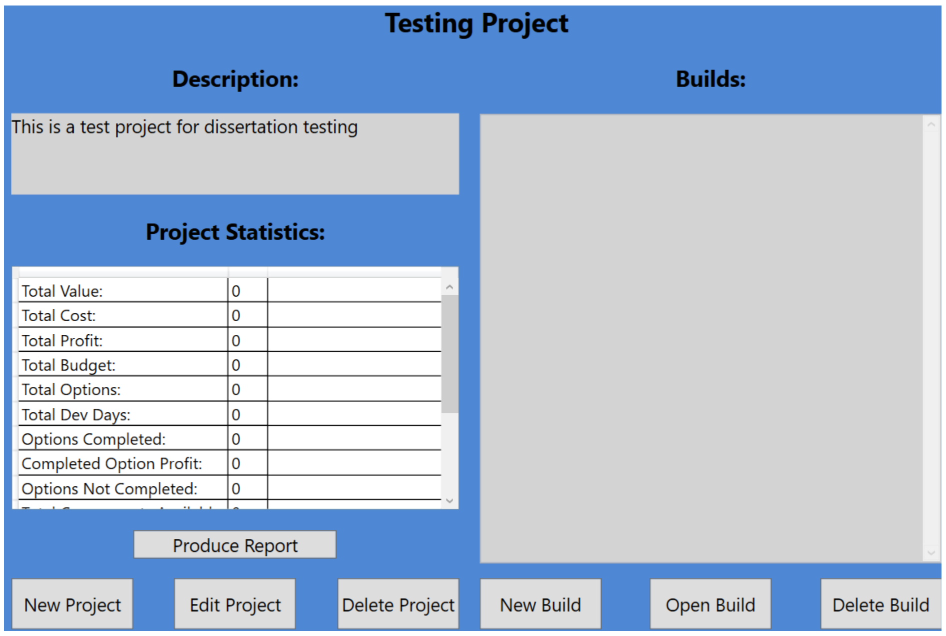
iv) 
Name length must be 30 characters or less.

v) 
Description length must be 100 characters or less.

vi) 

vii) 

No Project Selected.

viii) 

Testing Project

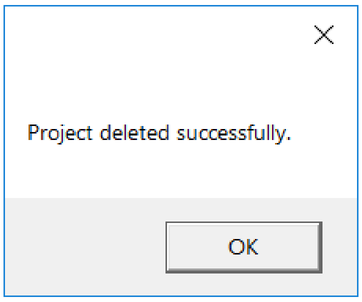
Description:

This is a test project for dissertation testing

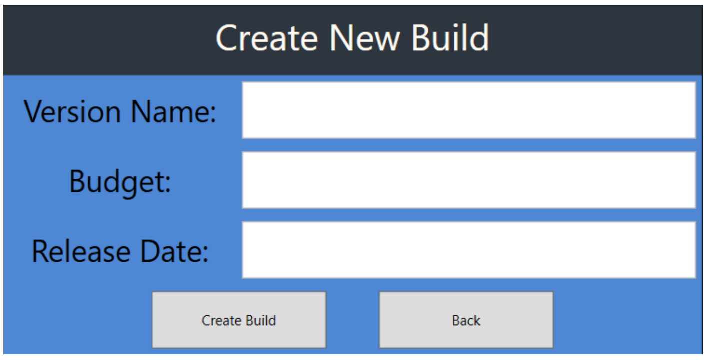
Project Statistics:

Total Value:	0	
Total Cost:	0	
Total Profit:	0	
Total Budget:	0	
Total Options:	0	
Total Dev Days:	0	
Options Completed:	0	
Completed Option Profit:	0	
Options Not Completed:	0	

Builds:

ix) 

Project deleted successfully.

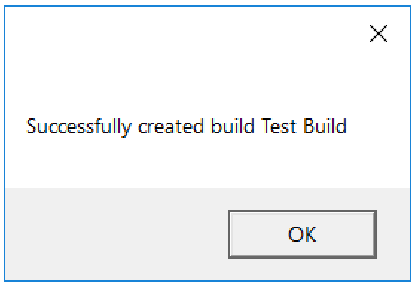
x) 

Create New Build

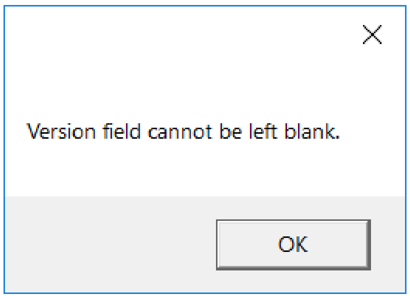
Version Name:

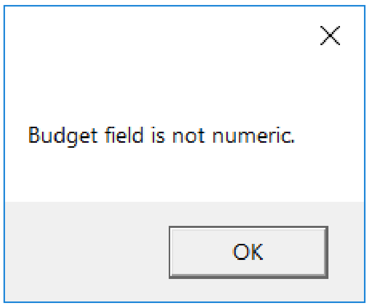
Budget:

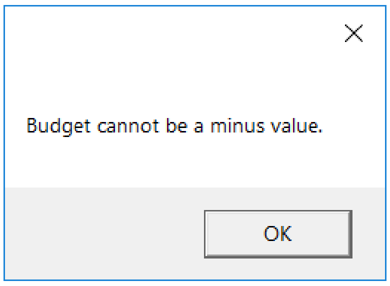
Release Date:

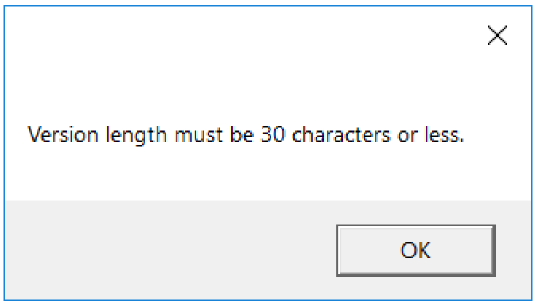
xi) 

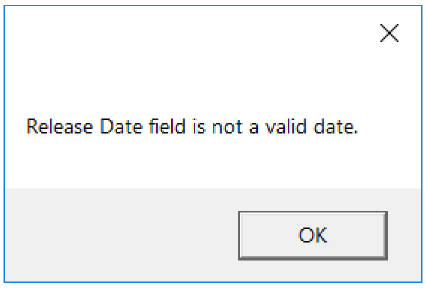
Successfully created build Test Build

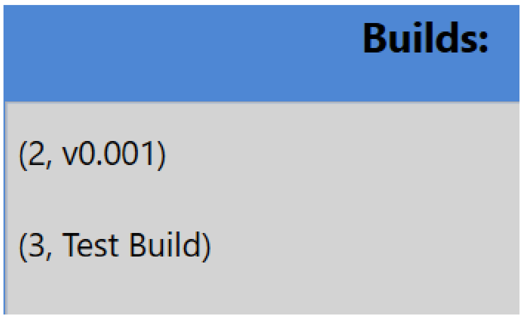
xii) 
Version field cannot be left blank.

xiii) 
Budget field is not numeric.

xiv) 
Budget cannot be a minus value.

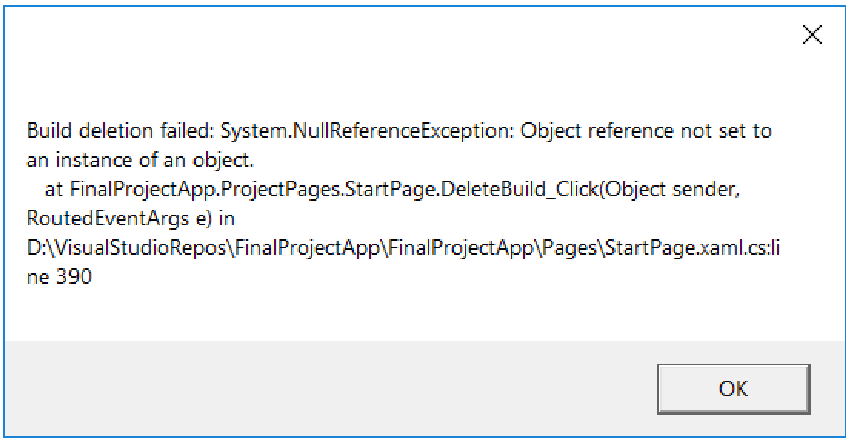
xv) 
Version length must be 30 characters or less.

xvi) 
Release Date field is not a valid date.

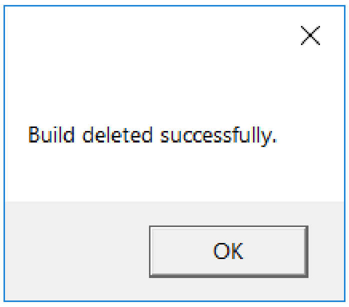
xvii) 
Builds:

(2, v0.001)

(3, Test Build)

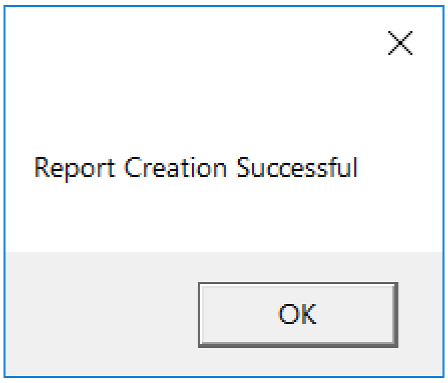

```
Build deletion failed: System.NullReferenceException: Object reference not set to
an instance of an object.
   at FinalProjectApp.ProjectPages.StartPage.DeleteBuild_Click(Object sender,
RoutedEventArgs e) in
D:\VisualStudioRepos\FinalProjectApp\FinalProjectApp\Pages\StartPage.xaml.cs:li
ne 390
```

xviii)

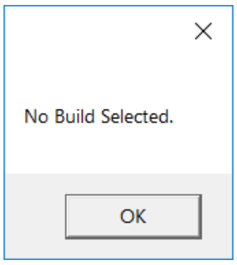

Build deleted successfully.

xix)

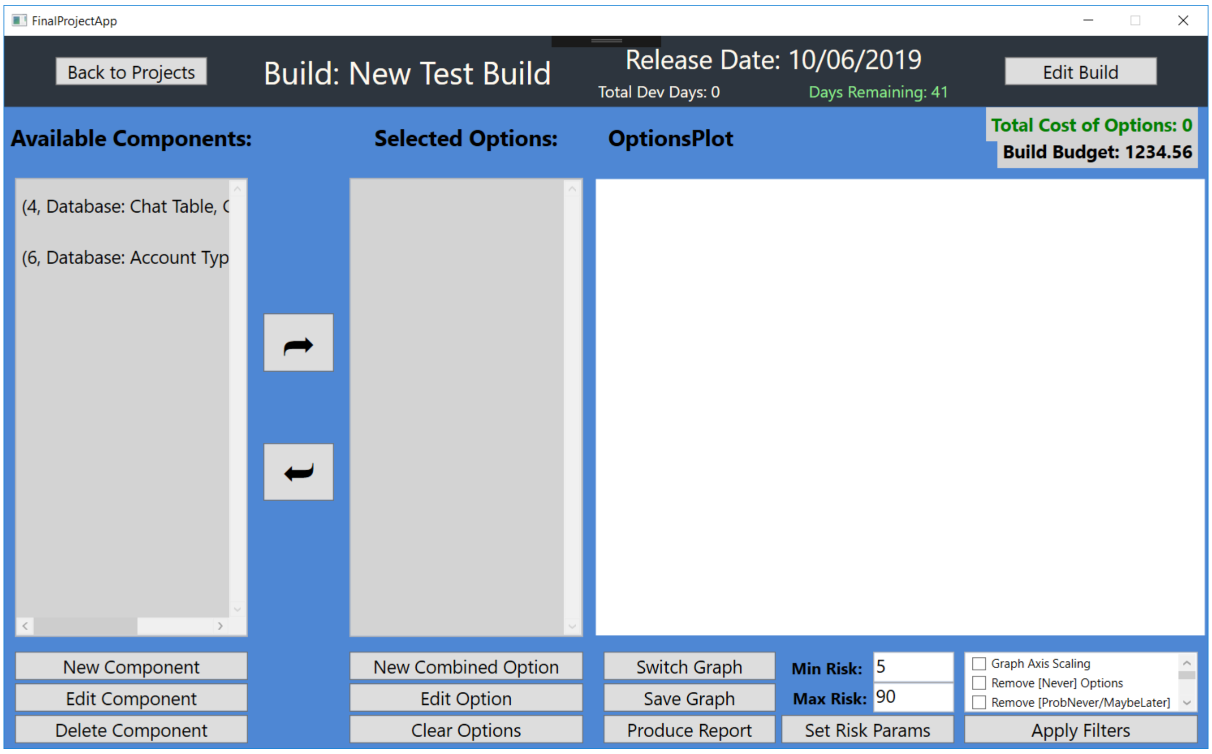
xx) [blank]


Report Creation Successful

xxi)


No Build Selected.

xxii)

xxiii) 

FinalProjectApp

Back to Projects

Build: New Test Build

Release Date: 10/06/2019

Total Dev Days: 0

Days Remaining: 41

Edit Build

Available Components:

Selected Options:

OptionsPlot

Total Cost of Options: 0

Build Budget: 1234.56

(4, Database: Chat Table, (

(6, Database: Account Typ

New Component

Edit Component

Delete Component

New Combined Option

Edit Option

Clear Options

Switch Graph

Save Graph

Produce Report

Min Risk: 5

Max Risk: 90

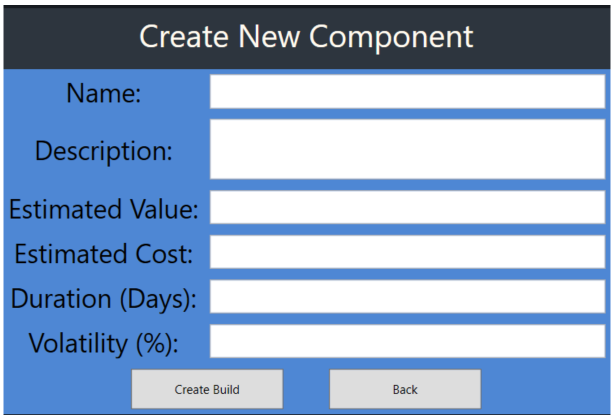
Set Risk Params

Graph Axis Scaling

Remove [Never] Options

Remove [ProbNever/MaybeLater]

Apply Filters

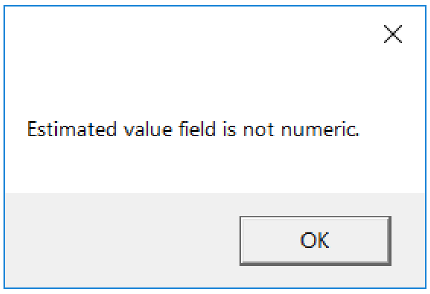
xxiv) 

Create New Component

Name:	
Description:	
Estimated Value:	
Estimated Cost:	
Duration (Days):	
Volatility (%):	

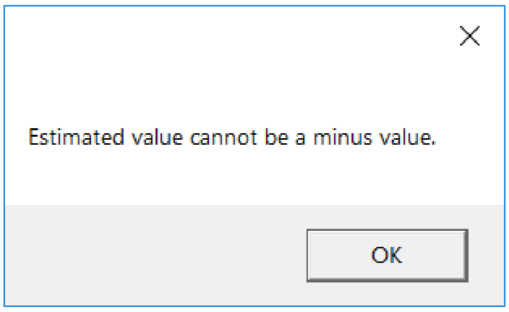
Create Build

Back

xxv) 

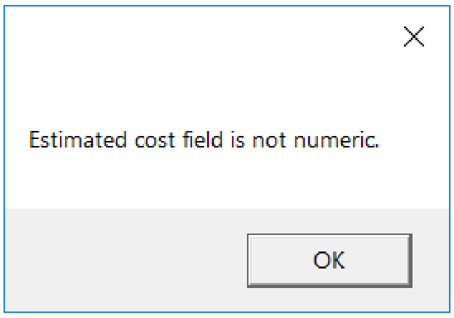
Estimated value field is not numeric.

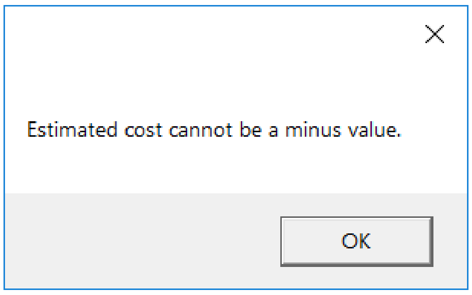
OK

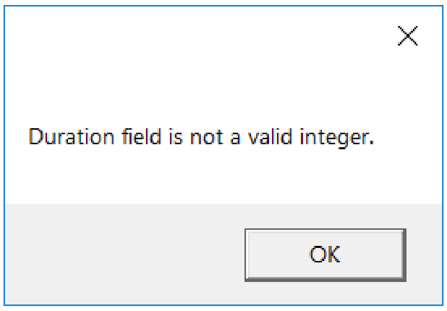
xxvi) 

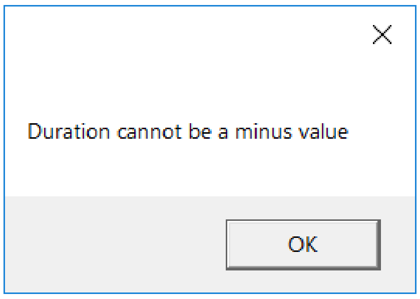
Estimated value cannot be a minus value.

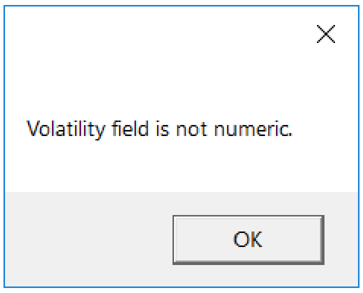
OK

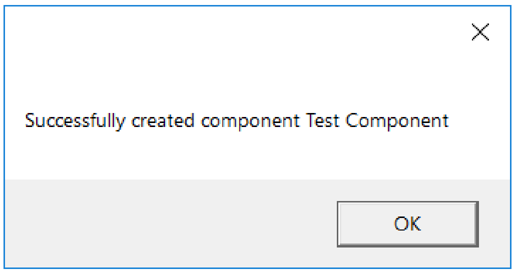
xxvii) 
Estimated cost field is not numeric.

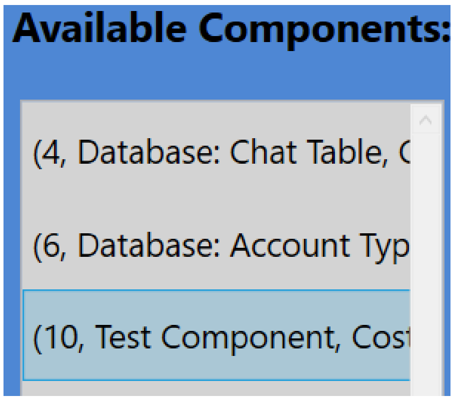
xxviii) 
Estimated cost cannot be a minus value.

xxix) 
Duration field is not a valid integer.

xxx) 
Duration cannot be a minus value

xxxi) 
Volatility field is not numeric.

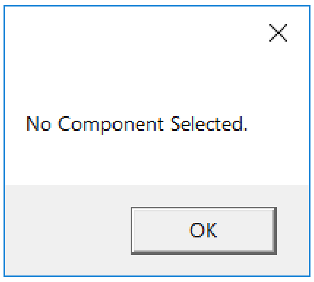
xxxii) 
Successfully created component Test Component

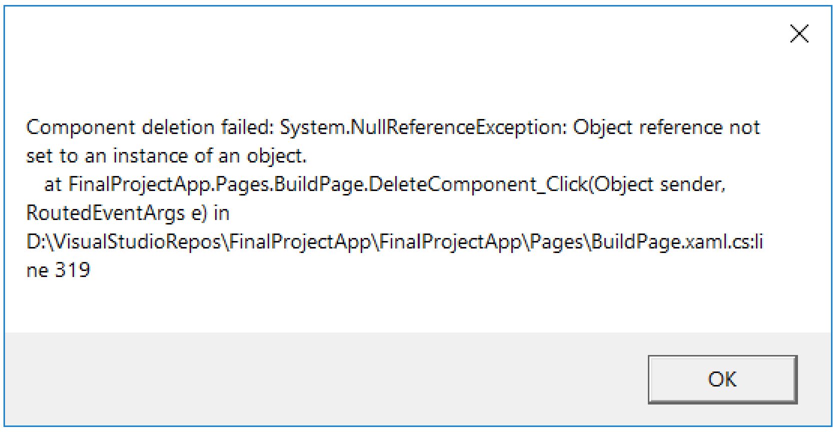
xxxiii) 
Available Components:

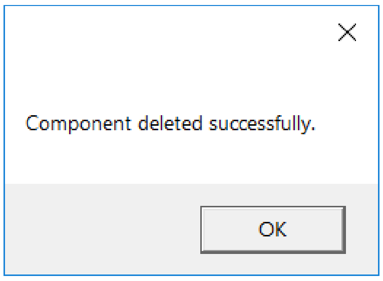
(4, Database: Chat Table, C

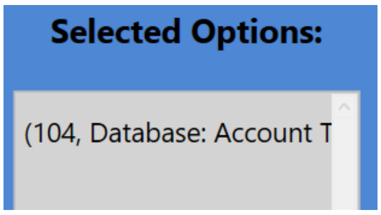
(6, Database: Account Typ

(10, Test Component, Cost

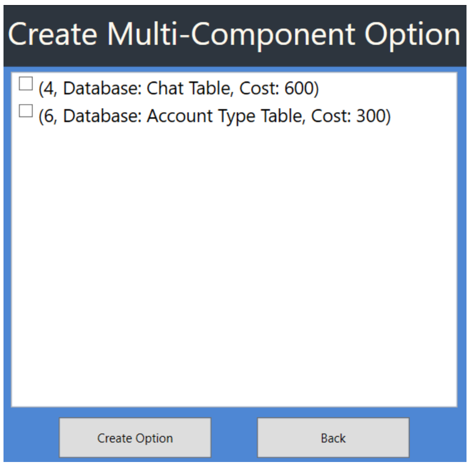
xxxiv) 
No Component Selected.

xxxv) 
```
Component deletion failed: System.NullReferenceException: Object reference not
set to an instance of an object.
   at FinalProjectApp.Pages.BuildPage.DeleteComponent_Click(Object sender,
RoutedEventArgs e) in
D:\VisualStudioRepos\FinalProjectApp\FinalProjectApp\Pages\BuildPage.xaml.cs:li
ne 319
```

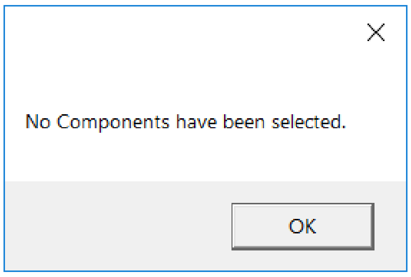
xxxvi) 
Component deleted successfully.

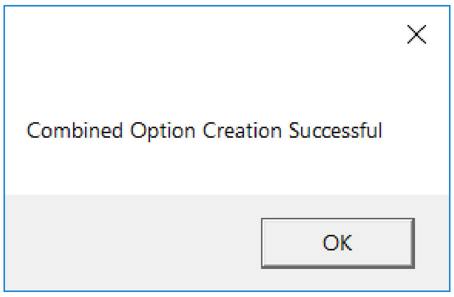
xxxvii) 
Selected Options:

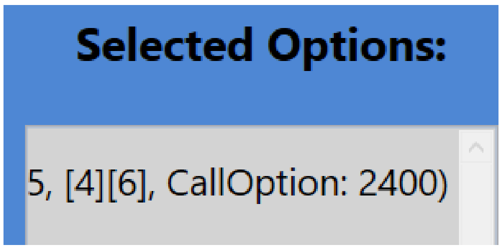
(104, Database: Account T

xxxviii) 
Create Multi-Component Option

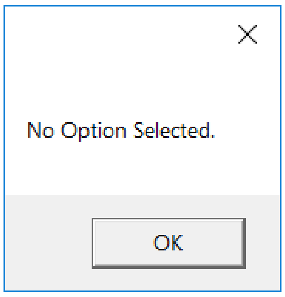
☐ (4, Database: Chat Table, Cost: 600)
☐ (6, Database: Account Type Table, Cost: 300)

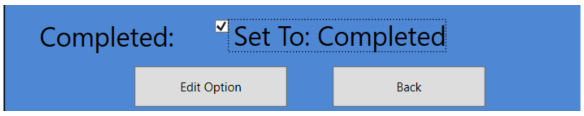
xxxix) 
No Components have been selected.

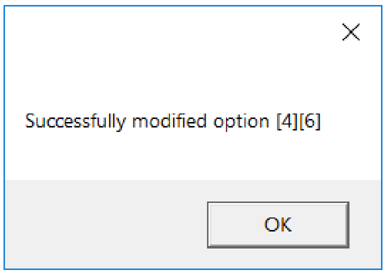
xl) 
Combined Option Creation Successful

xli) 
Selected Options:

5, [4][6], CallOption: 2400)

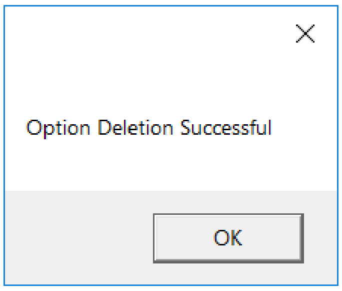
xlii) 
No Option Selected.

xliii) 
Completed: ✓ Set To: Completed

xliv) 

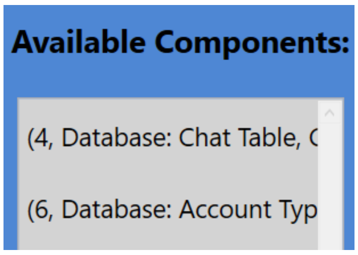
Successfully modified option [4][6]

OK

xlv) 

Option Deletion Successful

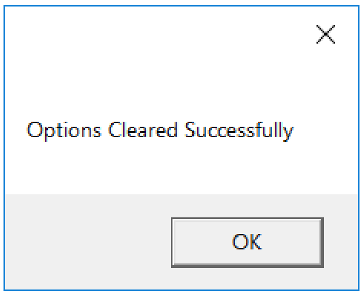
OK

xlvi) 

Available Components:

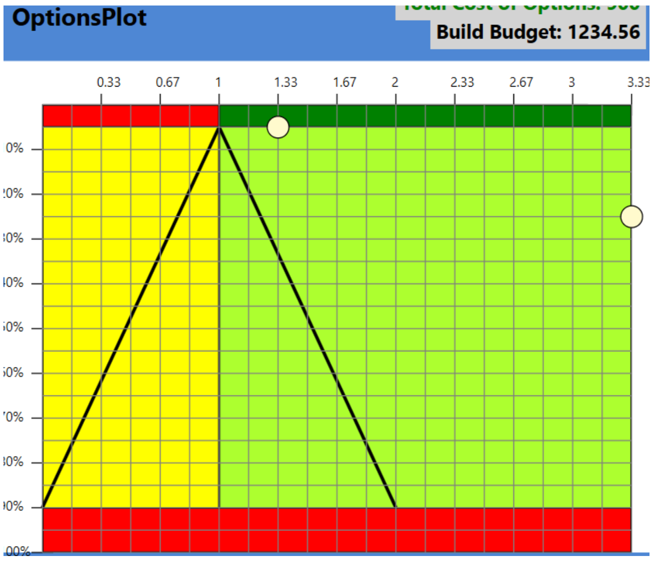
(4, Database: Chat Table, C

(6, Database: Account Typ

xlvii) 

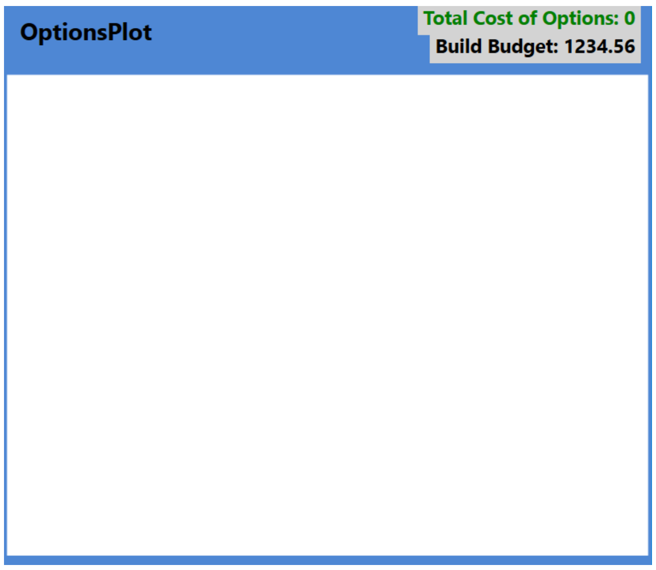
Options Cleared Successfully

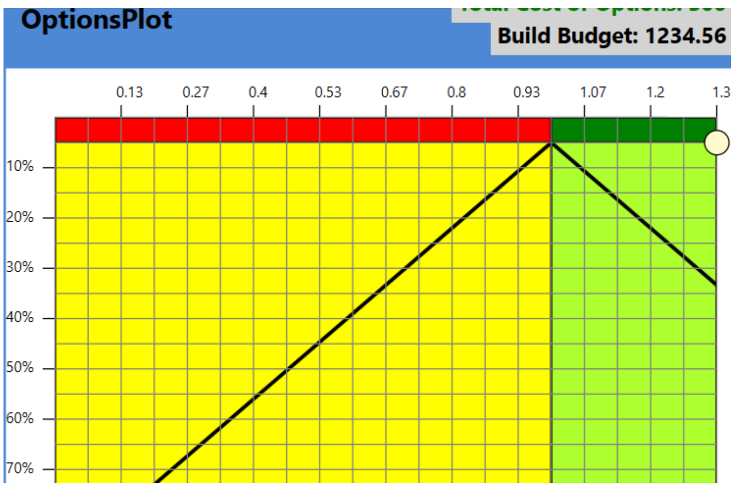
OK

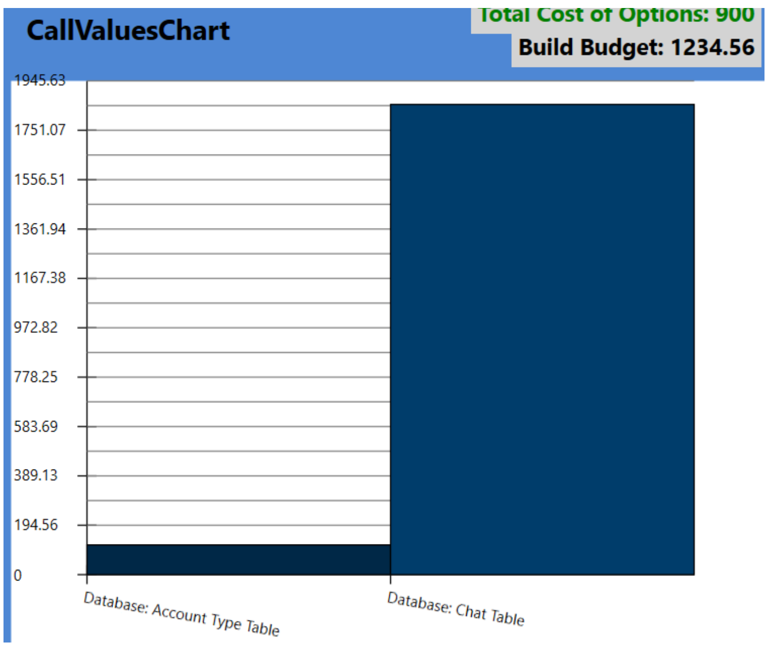
xlviii) 

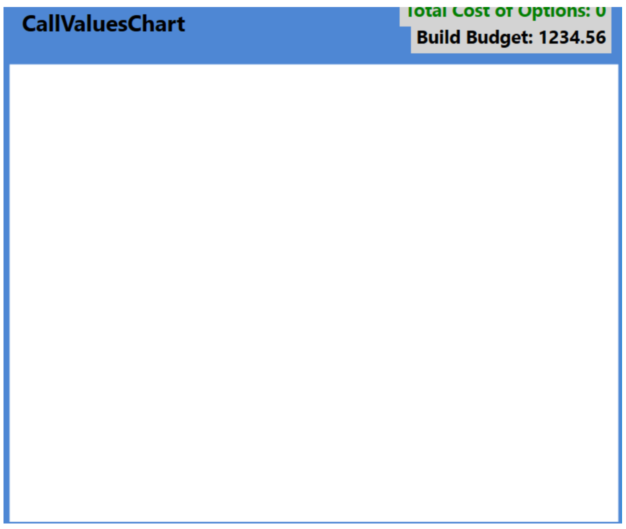
OptionsPlot

Build Budget: 1234.56

xlix) 
OptionsPlot
Total Cost of Options: 0
Build Budget: 1234.56

l) 
OptionsPlot
Build Budget: 1234.56
0.13
0.27
0.4
0.53
0.67
0.8
0.93
1.07
1.2
1.3
10%
20%
30%
40%
50%
60%
70%

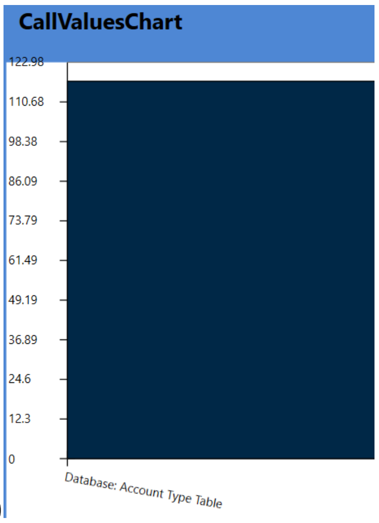
li) 
CallValuesChart
Total Cost of Options: 900
Build Budget: 1234.56
1945.63
1751.07
1556.51
1361.94
1167.38
972.82
778.25
583.69
389.13
194.56
0
Database: Account Type Table
Database: Chat Table

lii) 

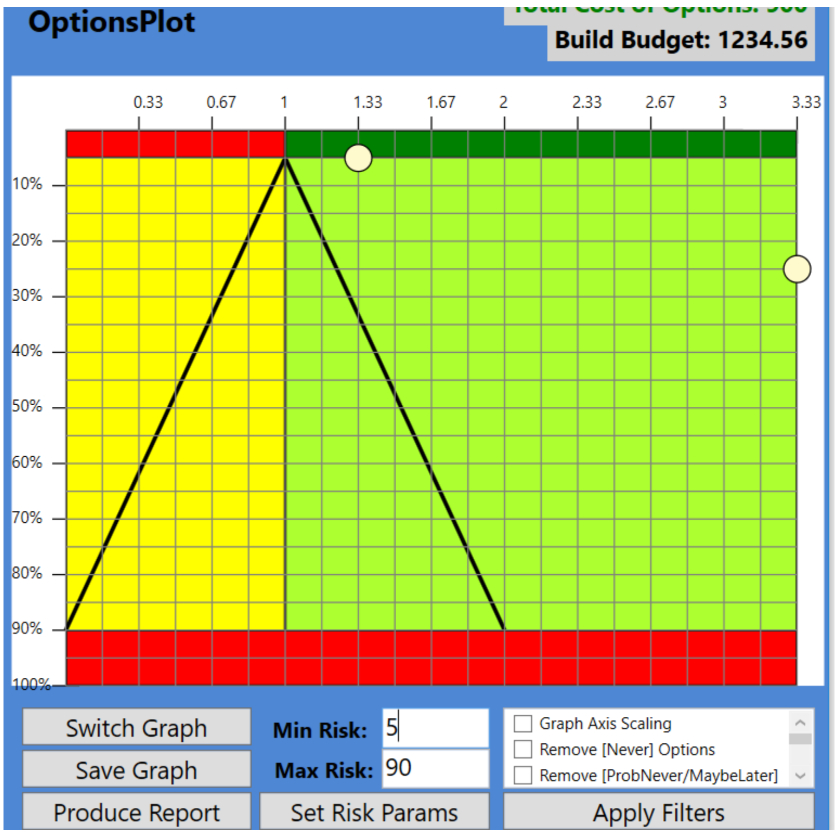
CallValuesChart

Total Cost of Options: 0

Build Budget: 1234.56

liii) 

CallValuesChart

liv) 

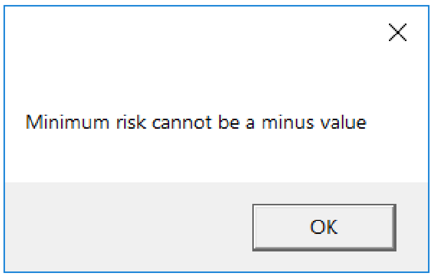
OptionsPlot

Build Budget: 1234.56

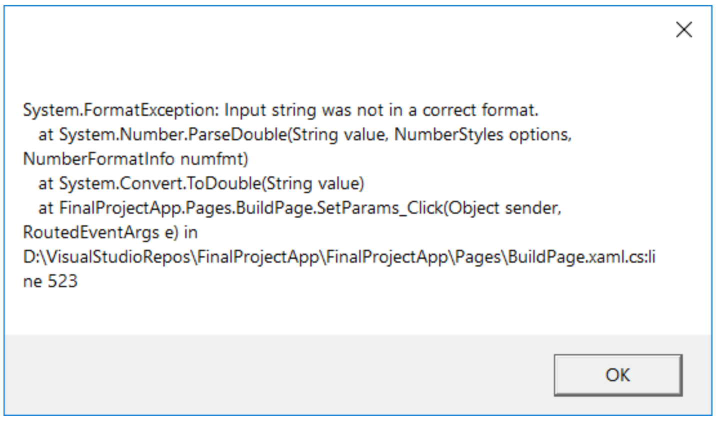
Min Risk: 5

Max Risk: 90

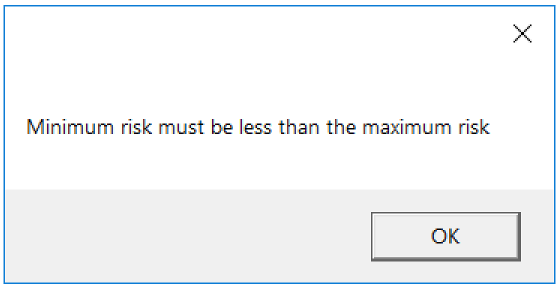
☐ Graph Axis Scaling
☐ Remove [Never] Options
☐ Remove [ProbNever/MaybeLater]

lv) 

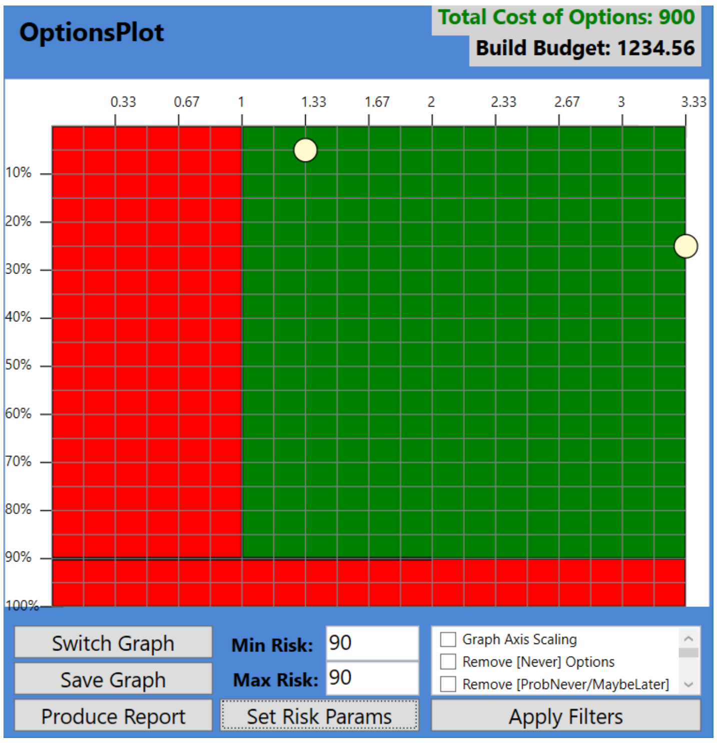
Minimum risk cannot be a minus value

lvi) 

```
System.FormatException: Input string was not in a correct format.
   at System.Number.ParseDouble(String value, NumberStyles options,
NumberFormatInfo numfmt)
   at System.Convert.ToDouble(String value)
   at FinalProjectApp.Pages.BuildPage.SetParams_Click(Object sender,
RoutedEventArgs e) in
D:\VisualStudioRepos\FinalProjectApp\FinalProjectApp\Pages\BuildPage.xaml.cs:li
ne 523
```

lvii) 

Minimum risk must be less than the maximum risk

lviii) 

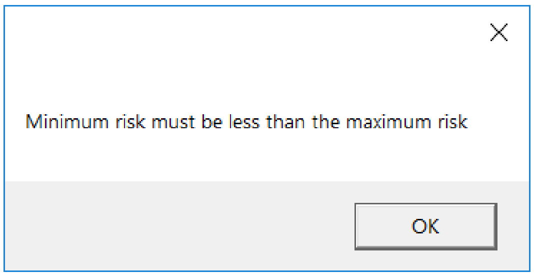
OptionsPlot

Total Cost of Options: 900

Build Budget: 1234.56

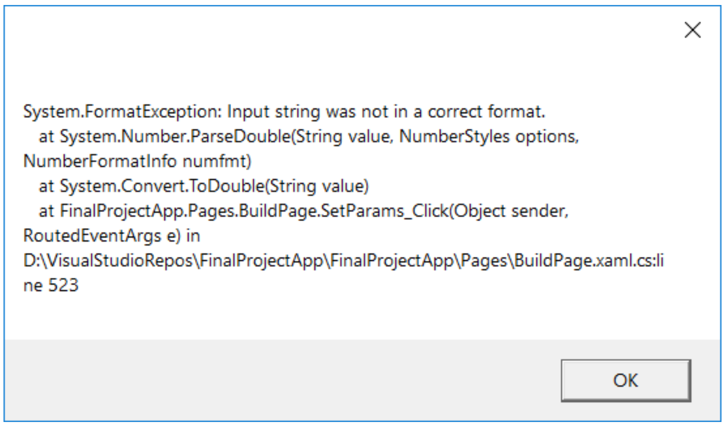
Min Risk: 90

Max Risk: 90



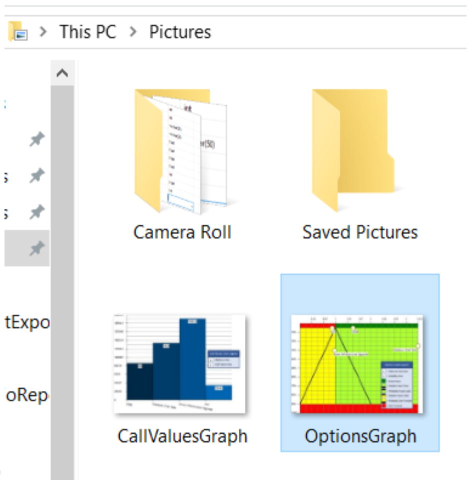
lix)

Minimum risk must be less than the maximum risk

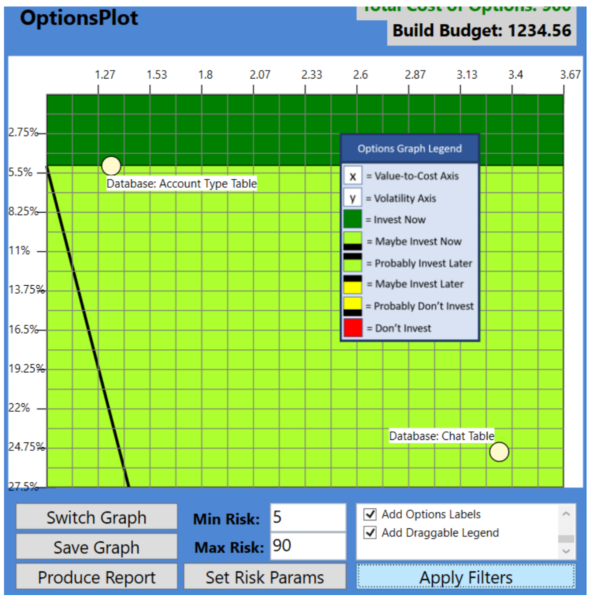


lx)

```
System.FormatException: Input string was not in a correct format.
   at System.Number.ParseDouble(String value, NumberStyles options,
NumberFormatInfo numfmt)
   at System.Convert.ToDouble(String value)
   at FinalProjectApp.Pages.BuildPage.SetParams_Click(Object sender,
RoutedEventArgs e) in
D:\VisualStudioRepos\FinalProjectApp\FinalProjectApp\Pages\BuildPage.xaml.cs:li
ne 523
```



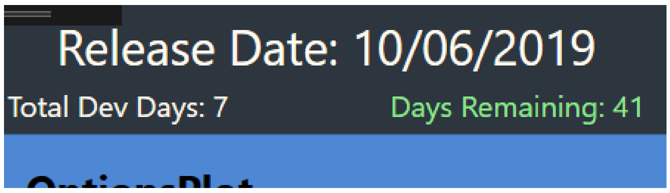
lxi)



lxii)

Min Risk: 5
Max Risk: 90
[✓] Add Options Labels
[✓] Add Draggable Legend

lxiii)

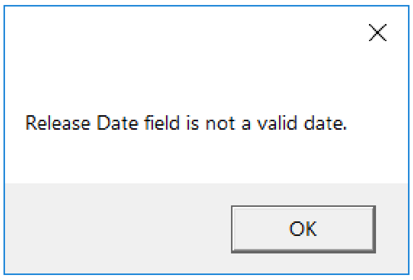


Release Date: 10/06/2019

Total Dev Days: 7 Days Remaining: 41

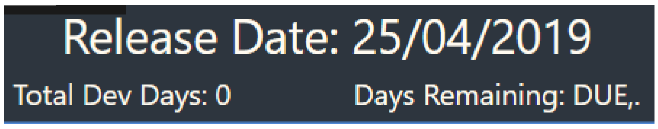
OptionsPlot

lxiv)



Release Date field is not a valid date.

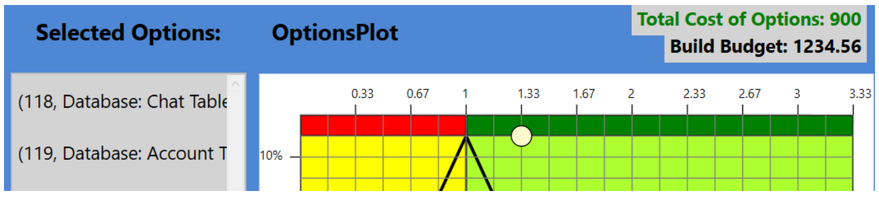
lxv)



Release Date: 25/04/2019

Total Dev Days: 0 Days Remaining: DUE,.

lxvi)



Selected Options:

OptionsPlot

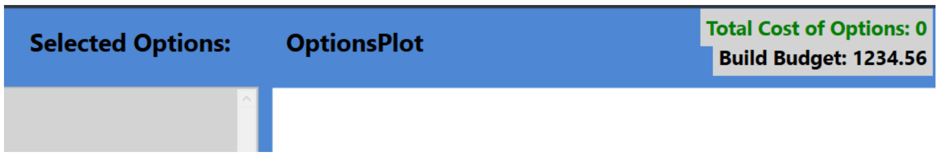
Total Cost of Options: 900

Build Budget: 1234.56

(118, Database: Chat Table

(119, Database: Account T

lxvii)



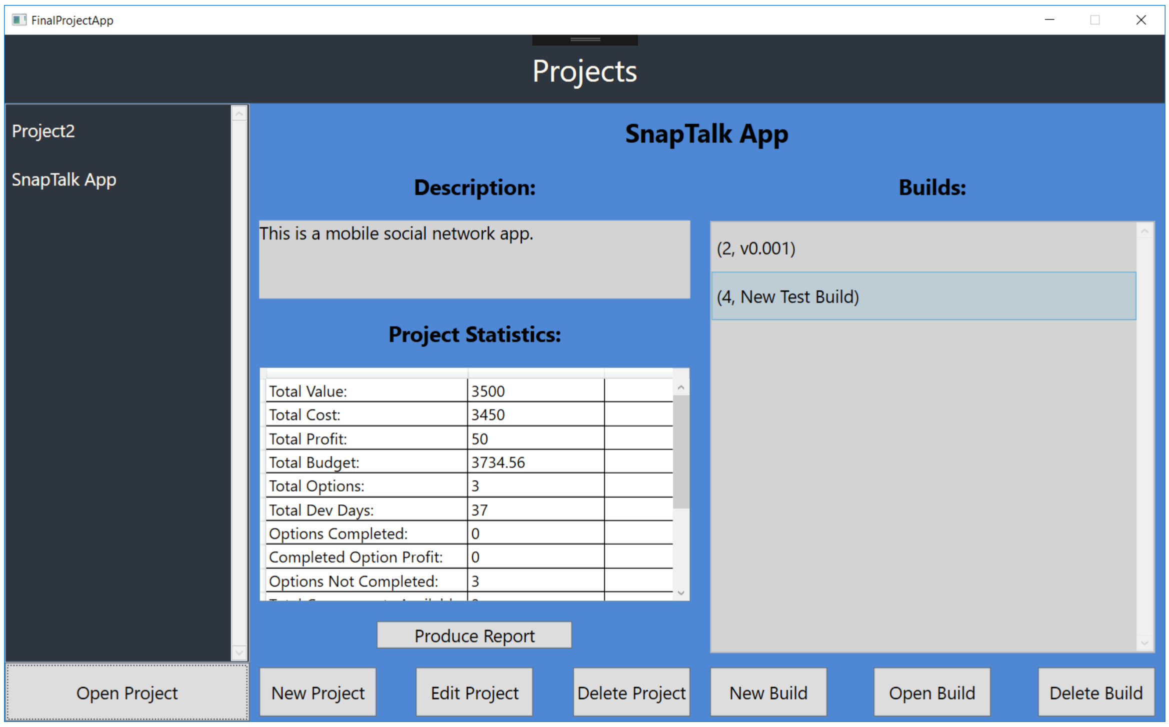
Selected Options:

OptionsPlot

Total Cost of Options: 0

Build Budget: 1234.56

lxviii)



Projects

Project2

SnapTalk App

SnapTalk App

Description:

This is a mobile social network app.

Builds:

(2, v0.001)

(4, New Test Build)

Project Statistics:

Total Value:	3500
Total Cost:	3450
Total Profit:	50
Total Budget:	3734.56
Total Options:	3
Total Dev Days:	37
Options Completed:	0
Completed Option Profit:	0
Options Not Completed:	3