atc visioncollege NEW ZEALAND

Diploma in Software Engineering and Design
Assignment Cover Sheet

Course name: Diploma in Software Engineering and Design	**Student's name:**
Module Name /or number: Computer Programming 1 (*15 credits*)	
Assignment title and/or number: C#.Net Assessment	
Assessment weighting	*Need to pass the assessment to complete the course*
Passing Criteria:	Need to score 50 or more marks to pass the assessment. **Total Marks : 100**
Due date: 1st Sept 2017	**Date submitted**: (late submissions incur 10% penalty, after 7 days late, the assessment will not be marked)
Assessment conditions:	This is a resource-based assessment. This means that you may have access to any relevant resources to assist you. This could include, for example, your learning materials, information on the Internet, and so on. However, all work must be your own with no assistance from any other person.
Submission requirements:	You're required to upload the following on Cloud Campus: • This document, completed where appropriate • Visual Studio project files • Upload your project on Github and paste the link below

	GitHub link below:
Learning Outcomes:	• Develop computer programs in a GUI environment • Demonstrate knowledge of data types and structures • Demonstrate knowledge of user interface controls and user interface animation • Design, develop and test a computer program • Incorporate comments into code • Compile, debug and test computer programs

Assignment Checklist:

Requirement	**Completed**
Database	✓✗
User interface	✓✗
Functionality	✓✗
Coding	✓✗
Testing	✓✗

Disclaimer of Plagiarism and Collusion

I declare that, to the best of my knowledge, this assessment is my own work, and has not been copied from any other student's work or from any other source.

Enter your name here to indicate you agree to the above statement.

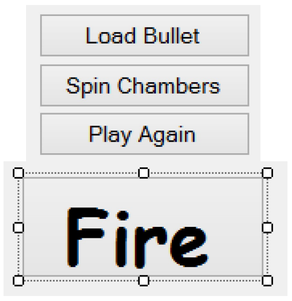
Jayden Stewart

Assessment Overview

This project involves simulating a Russian roulette game using C#.net.

Functionality Requirements

The program runs like this (with a button for each step)

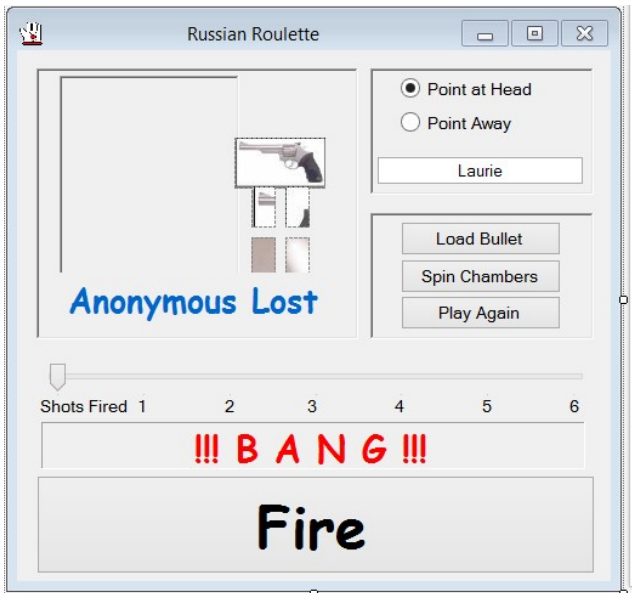


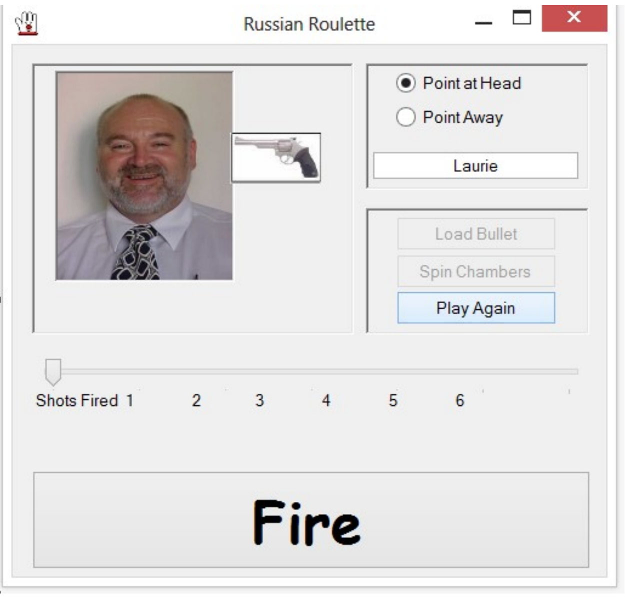
1. You load the bullet into the chamber of a revolver
2. You spin the chamber to create a random place where the bullet is with a 1 in 6 chance.
3. You pull the trigger repeatedly until the gun fires through to the number where the bullet is stored.
4. The player has 2 chances to shoot away during the game, this means that if the bullet is fired during that time they survive. If they shoot away twice and still the bullet has not been fired then the next shot they die.

Project Requirements

- ✓ Should simulate the game
- ✓ Use of appropriate data types and variables
- ✓ Appropriate User Interface(UI) design
- ✓ Creative and Intuitive UI (Reducing the possibility of human errors)
- ✓ At least one sound and image
- ✓ Sound and image should be loaded from the resource folder.
- ✓ All significant code to be commented
- ✓ User profile and a scoring system
- ✓ Include Win, Lose and Total scores
- ✓ Proper Testing performed and a testing sheet is included
- ✓ Project to be hosted on **Github**

Sample UI Layout





Testing Requirements

Testing assesses the quality of the product. Fill out the following testing sheet with the results of your test and submit it. The table below shows a sample-testing matrix with a sample test scenario. Identify as many test scenarios as you can and test your software against it. Feel free to include additional columns or note points as you fix the bugs.

Functionality	Test Scenario	Expected Outcome	Actual Outcome	Result (Pass /Fail)
Fire Button	• Pressing the Fire button without loading a bullet	A message box showing an appropriate message.	Fire button disables on form load until bullet is loaded	
Spin Button	• Pressing spin without loading a bullet	Button Disables	Button disabled	Pass
Radio button Trump	• Checking radio button after 2 attempts.	Disables button	Button Disables	Pass

Code quality checklist

Code quality is important for Maintainability and Testability of a software project. Refer to the code checklist below and check if your code meets the standards specified. (The code checklist given below are very general and some points may not apply to your project)

Code Formatting

1. ☐ Standard language indentation, spacing & bracketing?
2. ☐ No tabs, only spaces?
3. ☐ No lines longer than 80 characters?
4. ☐ Good spacing between function bodies?
5. ☐ Avoids functions longer than a page?
6. ☐ Uses vertical alignment to enhance readability?
7. ☐ Well spaced, clean & clear? Neatness counts!

General Code Quality

1. ☐ Good class names?
2. ☐ Good function/method names?
3. ☐ Good variable names?
4. ☐ Easily understandable?
5. ☐ As simple as possible?
6. ☐ Follows principle of least astonishment?
7. ☐ Extracts commonality & minimizes duplication?
8. ☐ No unnecessary cuteness or cleverness?
9. ☐ Proper names for Controls, Forms (txtName , lbluser etc.)

Comments

1. ☐ Explains anything that is non-obvious?
2. ☐ Could come back & understand in 10 years' time?
3. ☐ Unfinished parts marked with TODO?
4. ☐ Broken/incorrect parts marked with FIXME?

Error Handling

1. ☐ Proper message boxes added
2. ☐ Considers corner cases, worst case scenarios?
3. ☐ Checks for all possible failures?

4. ☐ Catches all possible exceptions?
5. ☐ All paths through the code tested?
6. ☐ All error paths tested if possible?

File Handling

1. ☐ Good filenames?
2. ☐ Following project's conventions?
3. ☐ Correct locations?
4. ☐ Correct permissions?

Marking Schedule

The table below shows a summary of the breakdown of marks. A detailed breakdown is given in the next section.

Points Assessed	Marks
User Interface	**20**
Functionality	**30**
Coding	**40**
Testing	**10**
Total	**100**

Detailed Marking Schedule

% of Grade	Excellent 80 - 100%	Adequate 60 - 80%	Poor 50 - 60%	Not Met 0 - 50%
User Interface 20%	Simple easy to use intuitive UI, no errors spelling mistake and good colour schemes used	Minor errors with the UI, minor layout issues	Major UI errors making it hard to understand. High possibility of human errors.	Significant UI errors with no logical sense and frequent UI issues
Functionality 30%	No errors, program always works correctly and meets the specification	Minor details of the program specification are violated, program functions incorrectly for some inputs.	Significant details of the specification are violated, program often exhibits incorrect behaviour.	Program only functions correctly in very limited cases or not at all
Coding 40%	No errors, code uses the best approach in every case and follows the coding standards	Minor errors or repetition of code, coding and naming standards not followed in some occasions	Code uses poorly-chosen approaches in some places. Naming standards not followed in some places.	Many things in the code could have been accomplished in an easier, faster, or otherwise better fashion. Poor naming and coding standards
Testing 10%	Program is well tested to identify and fix bugs and errors. No major bugs or defects in the program. Testing results matches the actual program.	Program is well tested to identify most of the bugs, but some bugs still exist. Some test cases marked pass which are false	Program has a lot of major bugs and is not tested. Testing sheet incorrect or incompletely filled out.	Program is not at all tested and testing sheet is not filled.