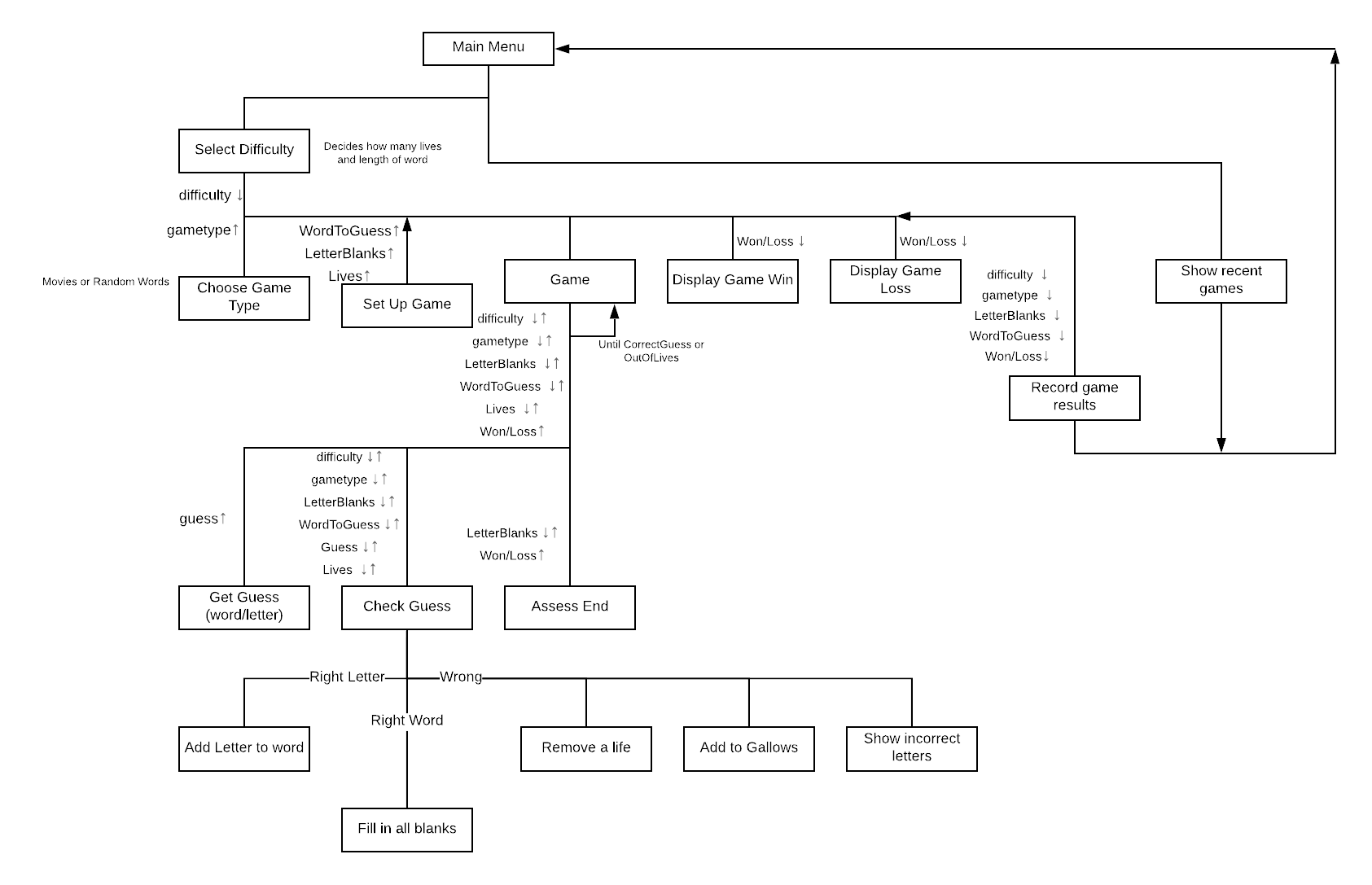
Hangman Game

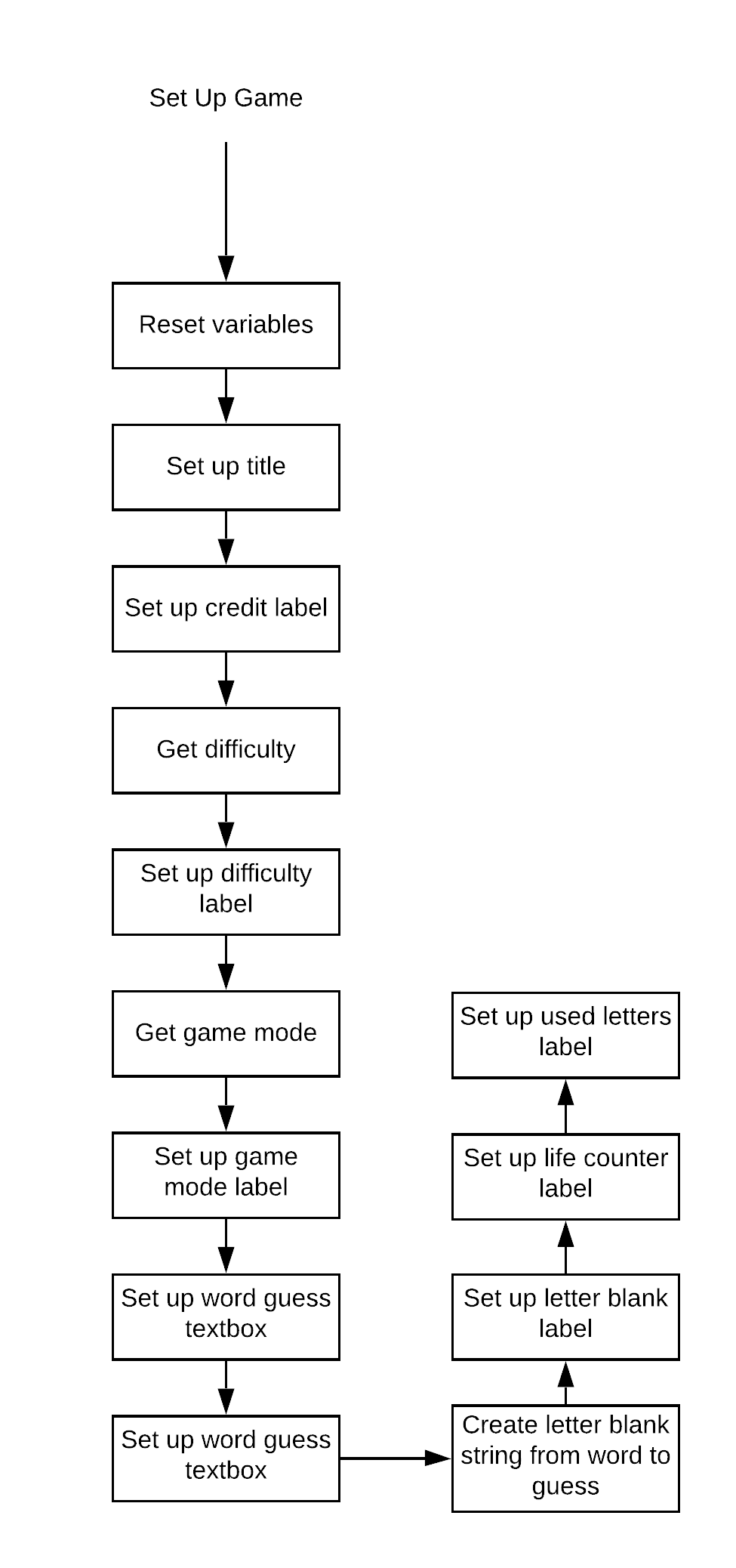
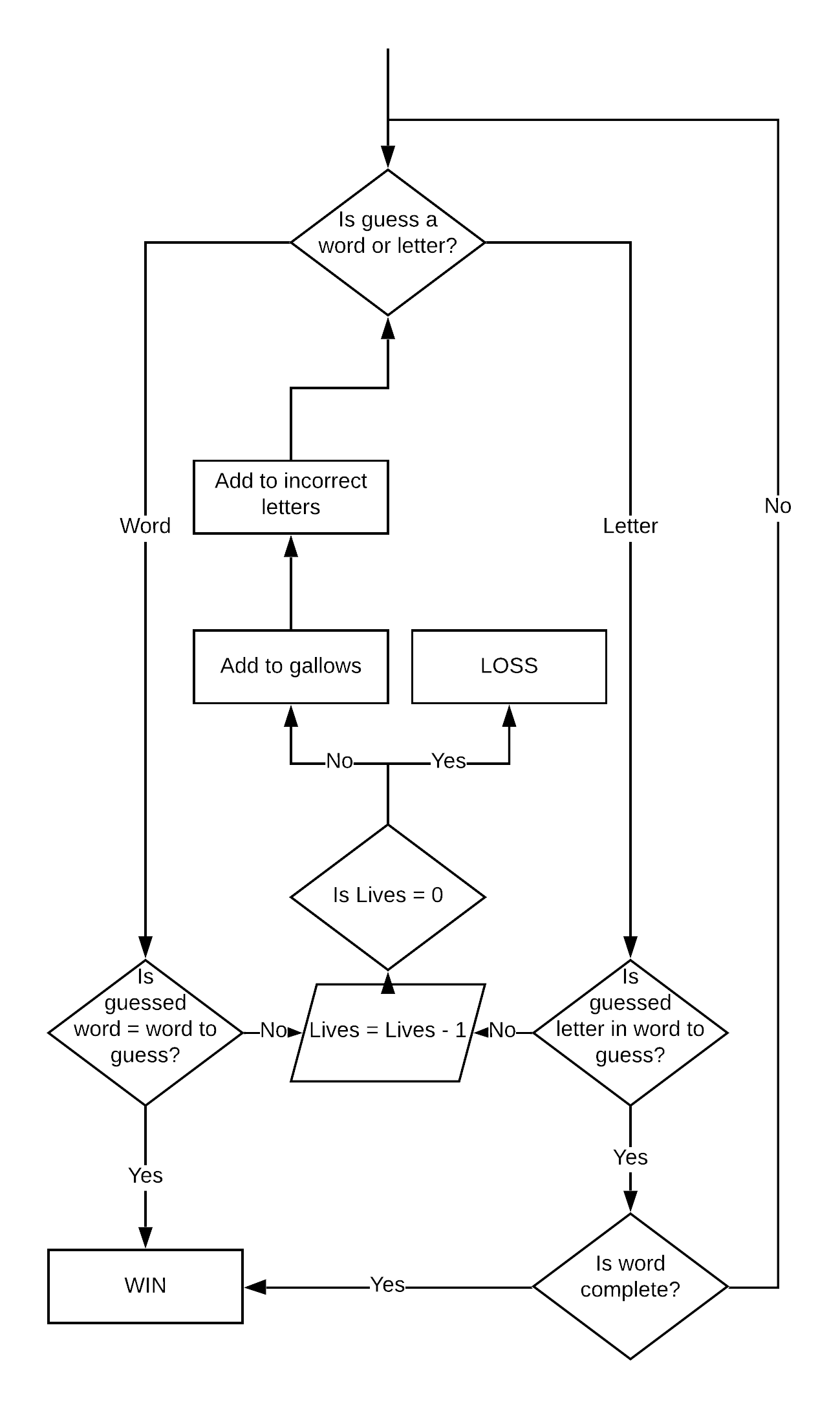
# Design

For the design you will need to provide the following:

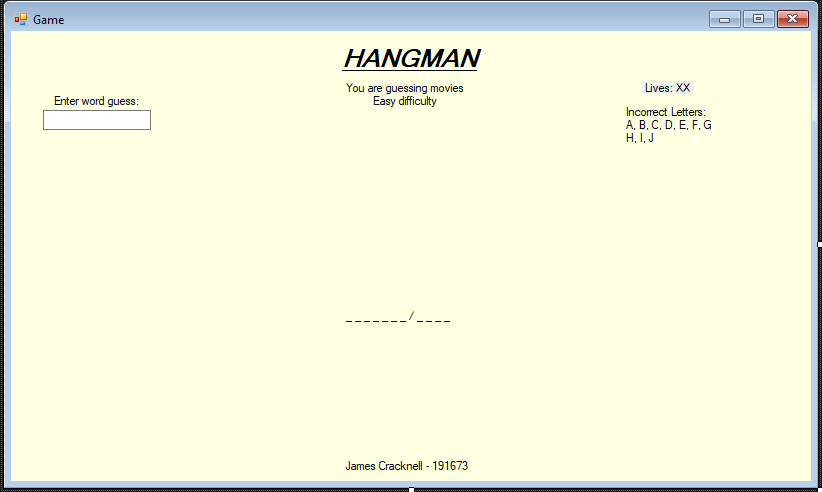
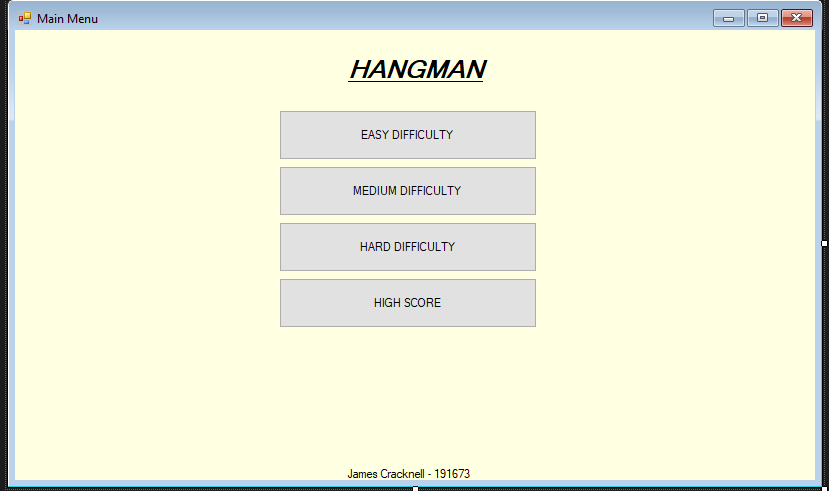
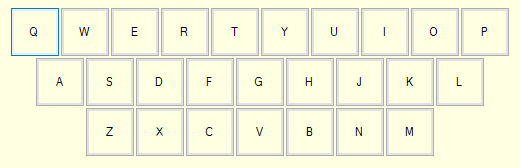
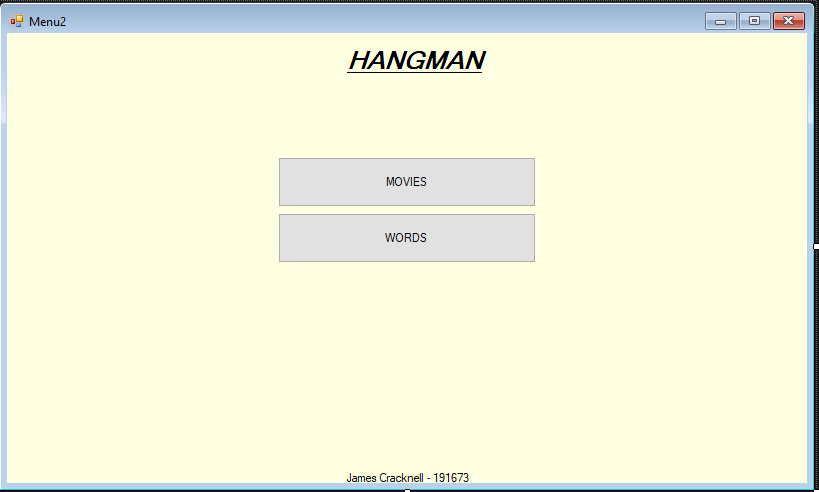
1. Top-level Chart (UML class diagram or Structure Chart)
2. Detailed flowchart / pseudo-code for specific (important) bits of the program
3. Annotated User Interface design (feel free to use the forms designer)
4. Detailed description of how you intend to display the:
   1. Gallows
   2. Letters to Guess
   3. Guessed Letters
   4. Guessed words
5. List of files and locations
6. Details of any libraries you are planning to use
7. List of tests you intend to carry out to show you code works (test plan)
8. **Top-level Chart (UML class diagram or Structure Chart):**

****

2) **Detailed flowchart / pseudo-code for specific (important) bits of the program**



Guess Evaluator

3) **Annotated User Interface design (feel free to use the forms designer)**   
  
  
  
  


Game Page

Buttons should grey out after use

Shows incorrect letters

GALLOWS  
HERE

KEYBOARD HERE

Shows lives left

Allows guess to be entered

Specific for each game: difficulty and game type are accurate

One \_ per letter  
Spaces are /  
Fills in as the game runs

Keyboard aligned in QWERTY order

PAGE 1

PAGE 1

PAGE 2

Credit given, discreetly, at bottom

Prominent title that is clear to read

Prominent buttons that are obvious to click on

Cream (“Info”) background, more interesting than white whilst not be harsh or taking away from the code.

4) **Detailed description of how you intend to display the:**

**Gallows:**I intend to display the gallows using Nakov’s turtle to draw the gallows. This turtle draw will be split into the different sections, either called individually or split with an if statement. The turtle will not be visible, use a pensize of 2 and a colour of black. I will use: Turtle.RotateTo() and Turtle.Forward() to move the turtle around the board.

**Letters to Guess:**

There will be a keyboard generated procedurally at the bottom of the screen. Players click on the buttons in order to submit a letter guess (using AddHandler). Buttons will be stored in an array. Only the 26 English letters are available to guess: numbers and accents will be removed from the list.

**Guessed Letters:**

There are two categories of guessed letters: incorrect and correct. Letters that are guessed will be greyed out on the keyboard and no longer clickable by disabling their function. Incorrect letters will be displayed in the top right under the incorrect letters subtitle. They will be stored in a Label. The correct letters will fill up the blank letters at the bottom, replacing the underscores in relevant places.

**Guessed Words:**

Words can be guessed through a textbox in the top left corner. If they are correct the game is won, if not they are displayed in a label below the word.

5) **List of files and locations:**All stored in bin 🡪 debug:  
*filmfile.txt* stores the film titles in one file *wordfile.txt* stores words in a file *PreviousGames.txt* stores data of previous games in a file

6) **Details of any libraries you are planning to use:**

I intend to use Nakov.Turtle in order to draw the hangman gallows graphics. This is added from the NuGet Package Manager, and provides a simple way to draw the elements onto the form.

I intend to use systems.io in order to access filereader and filewriter to access the files for the high score games.

7) **List of tests you intend to carry out to show you code works (test plan):**

There are six game versions: easy film, medium film, hard film, easy word, medium word, hard word. There is also the previous games button that displays previous games.   
All of these need scenarios need testing:  
- Correct letter guess  
- Incorrect Letter guess  
- Win from guessing correct letters  
- Lose from guessing incorrect letters  
- Win from guessing word  
- Lose from guessing word

Within these tests, these checks must occur:  
- Functionality for all buttons  
- Formatting of page is correct (due to being generated at run time)  
- Letters cannot be re-guessed  
- Program correctly loops   
- Incorrect letters are shown  
- Blanks are correctly shown  
- Blanks correctly fill  
- Game is correctly won  
- Game is correctly lost  
- Gallows correctly drawn

Other tests:   
  
- high score page shows high scores of games (*note: this has been changed to recent games after original writing*)