­­­­­­­­

OCR GCE A

COMPUTER SCIENCE PROJECT

H446-03

Name: Abdullah Noman

Candidate Number: S294151

Institution Name: Salford City College Pendleton

Title of Project: Rocket Agency

H446-03 – Project CONTENTS

Table of Contents

[A. Analysis 3](#_Toc124339464)

[Project description 3](#_Toc124339465)

[Stakeholders 3](#_Toc124339466)

[Justification 3](#_Toc124339467)

[Research 4](#_Toc124339468)

[Features 8](#_Toc124339469)

[Hardware and software requirements 10](#_Toc124339470)

[Success criteria 10](#_Toc124339471)

[B. Design 13](#_Toc124339472)

[Systems diagram 13](#_Toc124339473)

[Summary of process 14](#_Toc124339474)

[Key variables and data structures 15](#_Toc124339475)

[development test datA 15](#_Toc124339476)

[Algorithms 15](#_Toc124339477)

[post development test data 16](#_Toc124339478)

[Sign off proposal 16](#_Toc124339479)

[C. Developing the coded solution (“The development story”) 17](#_Toc124339480)

[D. Evaluation 19](#_Toc124339481)

[Evaluation-final solution 20](#_Toc124339482)

[Project Appendixes 21](#_Toc124339483)

# Analysis

## Project description

**Overview**  
The project is going to be a space simulation game, where you can build rockets and shoot them into space. The game will be set in a physical sandbox. You can explore different planets and complete missions while conserving fuel, examples of missions are taking pictures of different planets, landing on a moon, putting scientific instruments around solar systems.

**Main Features**

The game will include progression, as you complete missions, you will unlock more items, and more powerful parts. The fun is in the interactivity and risk of every mission, for example you must be careful while re-entering the atmosphere of a planet, keeping your heatshield in the right place otherwise you will burn up and you lose that rocket.

The game will be a space game with its focus to be exploration and progression. The target audience is ages 10+ as it can get quite complex the more you progress while still being fun in the beginning. This makes the game suitable for a wide range of ages.

## Stakeholders

**Bartlomiej Wierzba**

One of the stakeholders will be a friend of mine, who is interested in my project and the other one will be a classmate as they are doing a similar project. They would both look at the aspects of the game and tell me what they would like to modify or if they want to add any additional features. they are proficient in development and frequently plays games, which makes them a perfect fit for being a stakeholder.

**Osas Osaghe**

This will be my second stakeholder; they have a good understanding of what the game will be like, and they are ready to give constructive feedback to the mechanics of my project & is a peer so they will be available frequently so I can ask them questions. They also have played games like this in the past and are interested in my project.

My stakeholders already have a good visualization of what they would like to see in the game. The stakeholders will have my project description once it is complete and will provide constructive criticism to improve my project.

## Justification

**Abstraction and Visualisation**

The game is relying on abstraction. As in using OOP and frameworks that have been pre-written to display graphics on the screen. OOP helps me as I can use abstract classes that let me create the base objects that will be the building blocks for the game, after that I can inherit from those abstract objects and make them more unique to give a sense of variation in the game.

This helps me in programming, as I can develop while abstracting unnecessary information.

The air resistance will have a linear falloff which is proportional to the altitude above the planet.

This is not the case in real life as there are pockets of lower and higher pressure, which would also cause there to be Brownian motion which creates wind (air moving from high to low pressure areas to balance out the pressure). All of this will not be considered to keep the game from becoming too complex.

In real life, the planet’s gravitational pull is different depending on where on the surface of the planet you are on, this will be constant everywhere and be inversely proportional to the distance from the centre of the planet.

In the game, the distance between planets will be lowered in my game to make the planets and stars closer, if it were to scale, it would take an exceptionally long time to fly between celestial bodies. I will also make the lighting simplistic and removes other objects in space for example asteroids or debris. I also use 2D graphics instead of 3D to make the game more simplistic as it will be seen on two axes only.

The rockets will not deform when they crash, they will just fall apart while playing an explosion sound and an explosion effect on it e.g., a sprite.

The gravity will be simple and calculated by the radius of the planet instead of the mass.

**Thinking ahead**

The position, and size of planets before you start playing is prerequisite, so you can map them out on a 2D plane, instead of having them randomly generated as you explore space.

This allows the user to have a map of the galaxy/solar system.

The game will be built upon progression which is a procedural thing itself. You begin basic thrusters and fuel storage which allow you to fly up into the atmosphere while not leaving it or creating a stable orbit around the first planet. Once you finish that, you will unlock a science module which you can use to record data at various places which you can later exchange for science which is used to progress the game and unlock more areas of the solar system / galaxy.

**Thinking Logically**

This problem will have the main game loop to calculate the velocity and movement of the objects in space. And for branching there will be checks if a user has a specific part unlocked. Or if they are in the correct place to take a picture.

## Research

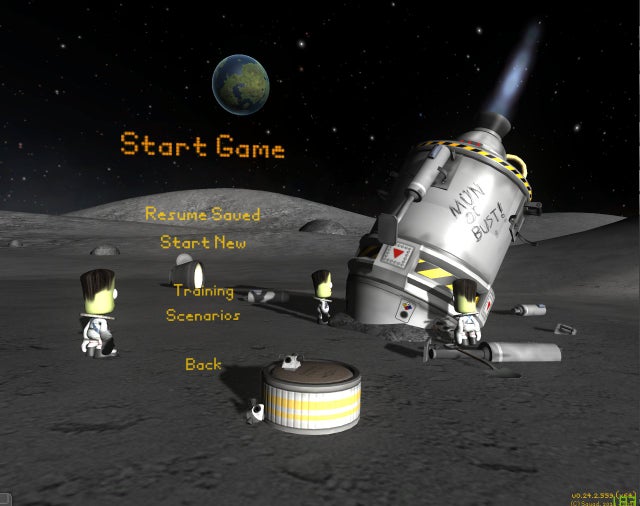
I have already got an idea of two games that are like mine, so I will investigate them and see any inspiration I can get from them. This will include games such as physics based 2D browser games for example space agency / Kerbal space program.­­­­

**First game, Kerbal Space Program**

**Overview**

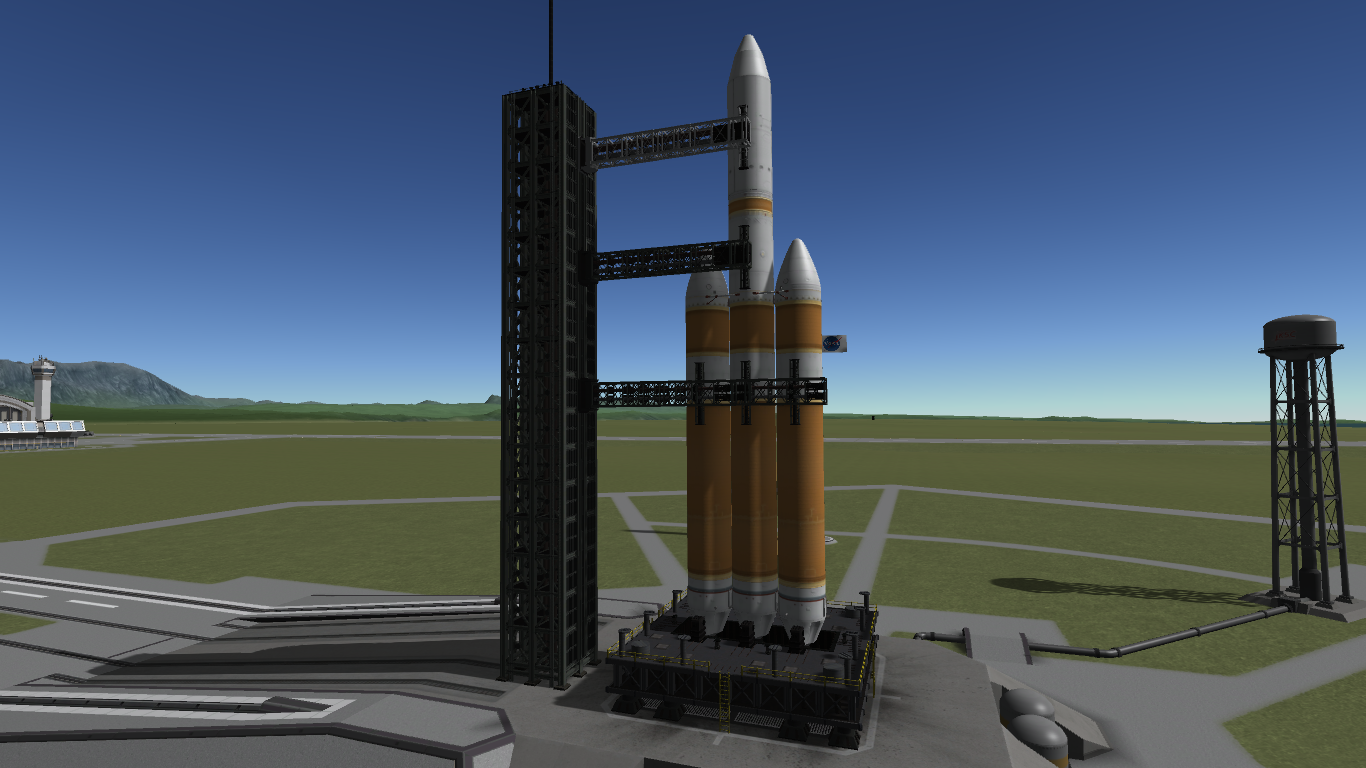
**­­**This game is a 3D space simulation game, which is very technical and intricate. The game is for PC only because of the amount of computational power you need for it. you can create your own rocket in 3D by using given parts and explore the different planets with the rocket. There is an in-game map which you can see all the planets and your ships from. This map allows people to see where they are going.

**Menus**



KSP features a bunch of different buildings which you can go into to do various tasks, for example for hiring crew members, building rockets, building planes, contract management, the tracking station and strategy planning. This makes everything well-laid out and you can easily differentiate between the buildings and focus on only one task rather than being overwhelmed by everything at once. This is also good for performance as the PC does not have to be able to do everything at once.

**Gameplay**



You can plan out your journeys by using the in-game maneuverer nodes on the tracking map which tell you at what angle you need to burn and for how long to achieve a certain orbit, the game also will tell you if you have enough fuel to carry out that journey after you plan it. While you are flying around, you can take measurements like the temperature, atmospheric pressure, crew reports, lab experiments and

This game has five planets and two dwarf planets, all of them are unique and you can explore them with rockets that you create. This is a similar concept to what I am planning to do so this will be good i

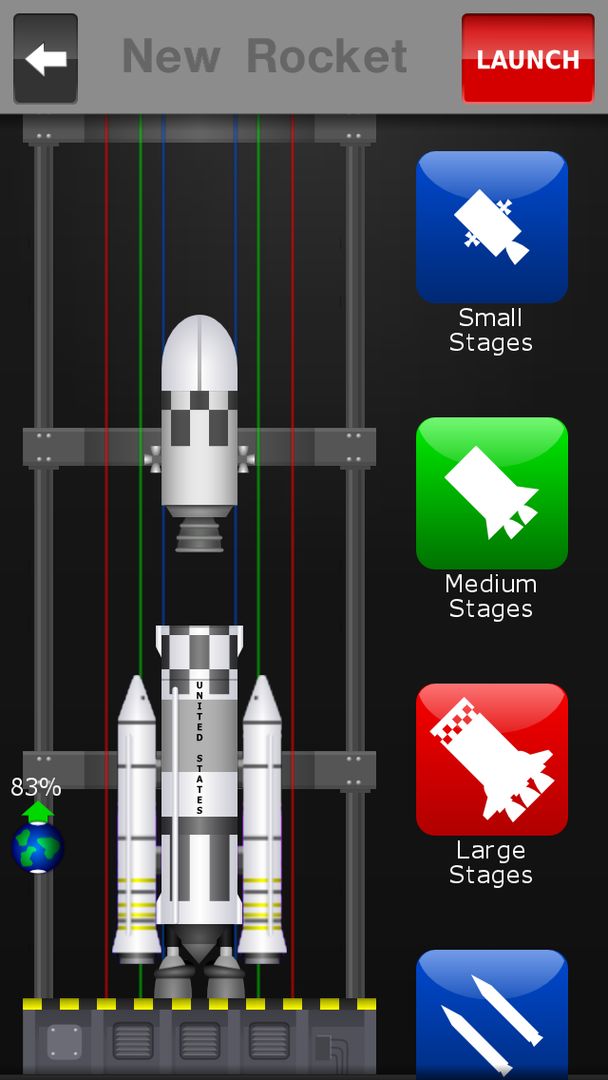
**Second game, Space Agency**

**Overview**

This game is a 2D Space flight game as well, this one is simpler and is a mobile game so that makes it better for the web as you will have less resources. This also makes the game simpler to play as it does not have as many features as KSP, this is the main inspiration.

**Menus**

**Rocket builder Building**



This is an example of how the rocket builder will look like, with having parts with subcategories of smaller and smaller components which show the parts that you can use to make the rocket. The multicoloured lines also give you a reference of the different sizes of the rocket parts.

**Rocket launch system**

A screenshot of a video game

Description automatically generated

This is the menu you will be seeing when you’re launching the rocket and while it is in space, this shows the fuel, the monopropellant, the time that the rocket has been launched for and the launch structure which is just a graphic that looks like it holds the rocket in place.

**EVALUATION**

The games I have researched are relevant and I will be taking a lot of their features as inspiration to my game, as they have features that fit well with the game & they have a lot of possibilities. The second game’s graphics are 2D and they are more simplistic so that game will be inspiration for the visuals, although I will be making it for a wider aspect ratio so I will have to change the format of information to be more friendly to the end user.

## Features

**ESSENTIAL FEATURES**

**Menu screens**

A simple interactive menu screen with a background image and clickable buttons that allow you to navigate the game and change settings. The pause menu will allow you to switch between buildings / missions and a menu for your tech tree so you can pick your next advancement. There will also be a menu for saving systems.

**Justification**

This is so people can navigate the game and change their settings and go into different screens / progress in the game, save / load old saves.

**Rocket builder Building**

An intuitive building system that can build rockets using parts that are given in a menu and calculate how much Delta V (how much energy) it has and it’s TWR (Thrust to weight ratio),

You can drag and drop different parts that you can attack to your rocket, starting from the main stage that will work best ASL (at sea level) and has a good TWR (Thrust to weight ratio). This lets you put a large mass above the first stage as it can lift it. This mass can be the second stage and / or payloads that you can also attach onto the rocket, there will be many different stages, some with more power and some with more efficiency so you can go on further and / or build larger rockets. These parts will be unlocked after progressing a bunch in the game.

**Justification**

This is so users can pick the parts they want and allow them to access the progression system and build their rockets.

**Rocket launch system**

Here, there will be a launch system which has a HUD (Heads Up Display) at the top / around the rocket where you can see how much fuel you have in your current stage, how much Delta V is left in the rocket (how much you can change the velocity of the rocket). The HUD will also show other forms of fuel such as monopropellant which is used to making small corrections to the trajectory of the rocket where even a small change in velocity makes a big difference. The HUD will not show the in-game map which will be in a different menu screen as there is limited space, so you can’t show the rocket and the map at the same time. The HUD will show arrows on the side of the screen to show where all the planets are relative to you

**Justification**

This is so players can read their rocket stats and find out where they are relative to other planets. This also allows the user to control the rocket.

**The mission control building**

This building will have your rocket and the planets on this screen. This screen is so you can track and set planets as your target, so it shows up more information about the planet, e.g., how far it is, the ETA etc.

**Justification**

This is to allow users to see a complete map of their planet system and understand the relation between the planets and their sizes, atmosphere information, and details about planets. This is mainly for quality of life for the user.

**DESIRABLE FEATURES**

**Loading screen**

The loading screen is where the game preloads all the assets and compresses them into one texture atlas which helps increase the game performance as the GPU queues fewer draw calls as it is all assets are one image. Games like Minecraft do this to optimise their rendering efficiency.

**Justification**

Having a loading screen makes the game look more professional and lets the player know that the game is loading instead of just having a frozen frame.

## Hardware and software requirements

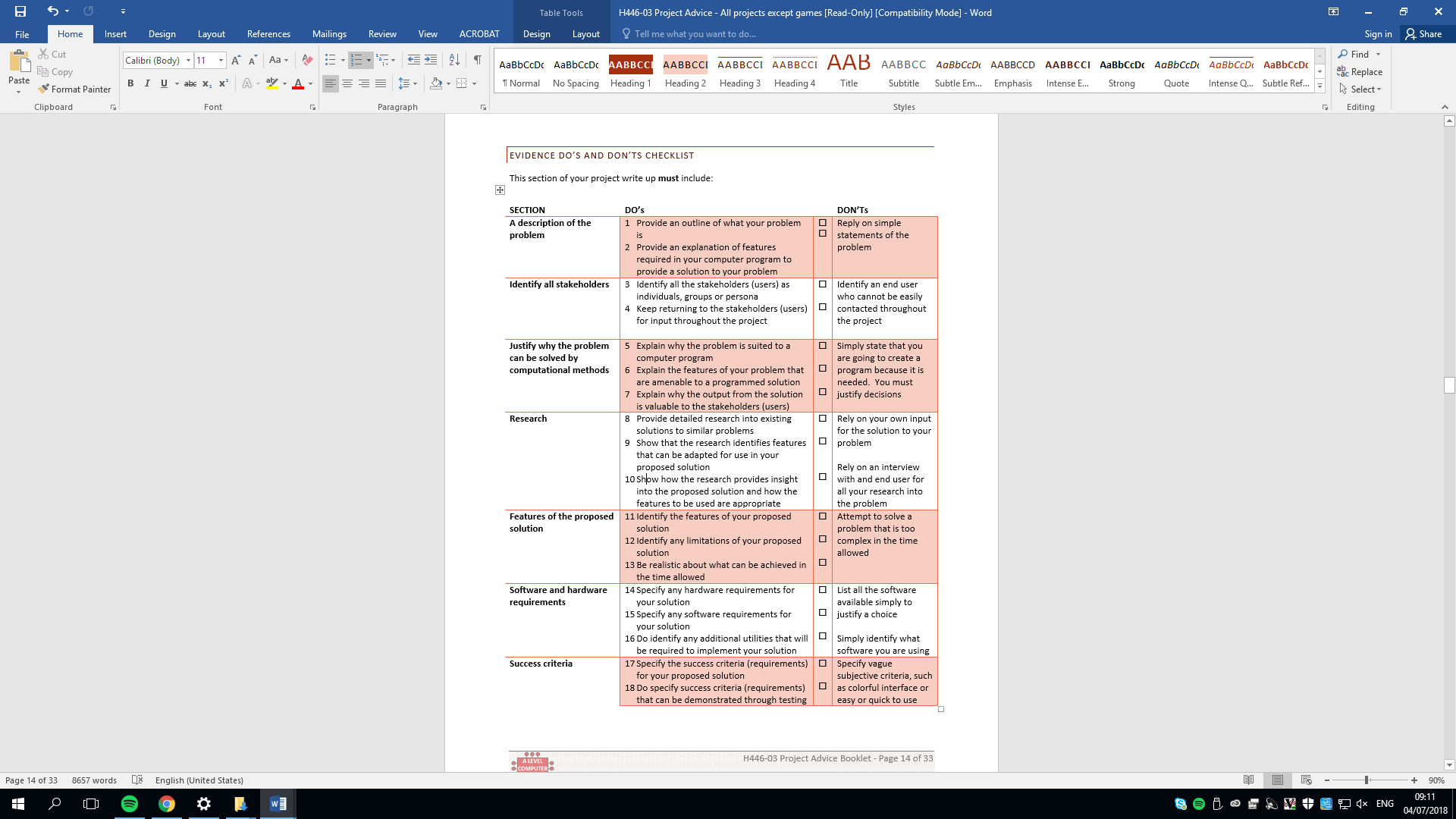
A working browser, with chromium 100 or above, and a modern graphics processor.

OR (if done in python)

Python 3.6+ installed, the game will automatically install pygame if not installed.

## Success criteria

|  |  |
| --- | --- |
| Buttons | The menu system has clickable buttons, sliders, and textboxes to input data and fire events. Every element has a function attached to it which is ran every time it the button is pressed with parameters which show the state of the element |
| Menu layering system | The game needs to be able to swap between menus efficiently, which means not drawing anything unnecessary to help reduce lag and make the game require lower specs to be able to run at a playable framerate |
| Rocket building system | The game has a building with a selection of parts that you can use to build a rocket which is realistic, and the parts fit with each other. |
| Rocket launch system | The rocket can sit on the launch pad and controlled by the player to launch the rocket into space. The rocket needs to be able to escape the atmosphere. |
| Rocket control system | The rocket needs to be able to rotate by using monopropellant buttons and have a controllable thrust factor so the player can finely tune their trajectory. |
| Collision detection | If the rocket collides with an asteroid / planet at a high enough velocity (or does not have the appropriate equipment to land) it will explode. |
| Save / load system | Allow the player to save their progress so they can carry on where they left off and have multiple saves. |
| Pause menu | The pause menu allows you to revert a flight back to launch in-case the rocket blows up. & Save and exit the game or save and go to main menu. |
| Building switch system | Ability to go into different menus and disable others seamlessly. |
| Timer | There should be a timer to display how long you have been in a rocket launch. |
| Graphics to show when the rocket is running | The fire below the rocket (plume) |
| Camera shake when the rocket is going fast in atmosphere | This is to give the user a more immersive experience launching rockets and gives the game a more premium feel. |
| Staging system | The stages should detach as the user wants them to and display a small animation of them moving away from the rocket once detached, this is for when a stage of the rocket runs out of fuel, you can get rid of excess fuel and switch to a more efficient engine that works better at a specific atmospheric pressure. |
|  |  |
|  |  |
|  |  |



# Design

## Systems diagram

Diagram

Description automatically generated

## Summary of process

* Save Game:
  + Open a file and save the game data into it, preferably in a JSON file as it has good formatting and is readable for debugging and is like python dictionaries. This format makes it, so it is easier to serialize data.
* Load Game:
  + Open a previously saved game data file and read the JSON, then desterilize the data into the program.
* Main menu:
  + The main menu has the buttons for settings, help, credits, a quick save button and a quit button.
* Building selection:
  + Display buttons for the different buildings so it is easy to swap between them.
  + This includes, tracking station, contract menu, hiring menu, vehicle assembly bay, and the tech tree menu.
* Tech tree menu:
  + Display the currently loaded tech tree in a graphical user interface
  + On press of button if the user has enough credits unlock that node
  + If node before that is not unlocked, do nothing.
* Tracking station menu:
  + Get the array of all the spacecraft that are up right now, display the list & their positions on a map
  + On click of a spacecraft, let them either self-destruct or control it.
* Contracts menu:
  + This displays a list of quests that you can agree to do, this will out a bounty on a quest
  + Once a quest is completed money is added to the player’s balance
* Vehicle assembly bay:
  + Display a list of rocket parts, the ones that are not unlocked by the nodes are not displayed as grayed out.
  + Load a rocket from a file.
* Build rocket:
  + When an image is clicked, put it on the mouse and when it is dropped in the middle, place it down as a stage.
  + Finalize a rocket after the finish button is pressed
  + Save rocket to file
* Display HUD:
  + Display the fuel, monopropellant, time, etc.
  + Display the quit button & positions on the edge of the screen pointing to planets.
  + Display velocity
  + Display lines on showing where the rocket is moving
* Display Planets:
  + Load images from planets folder
  + Draw planets at position if the position is less than the radius away from the edge of the screen
  + Draw the gravity ring around the planets (their sphere of influence)
  + Rotate the planet image on the screen to make it look like they are rotating
* Display Vessel:
  + Show on-screen controls of the vessel
  + Check if vessel is colliding with something else, if it is, destroy the vessel.
  + Recover vessel if it is on the home planet for funds
* Control vessel:
  + Take keyboard inputs to control the vessel and

## Key variables and data structures

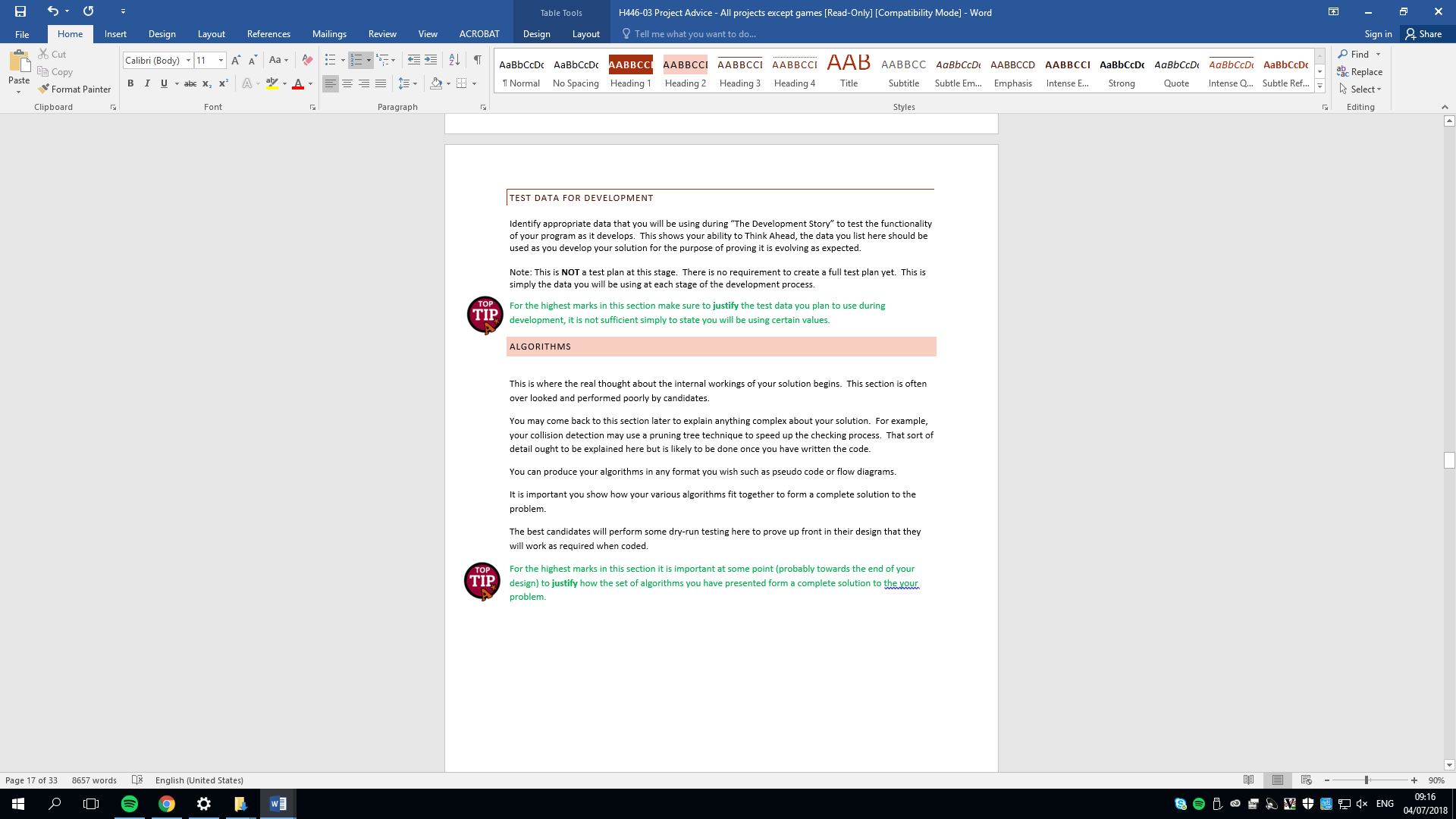
## development test datA

* Check if the stars display in the back:
  + The stars should display in the background of the screen and should be layered with parallax.

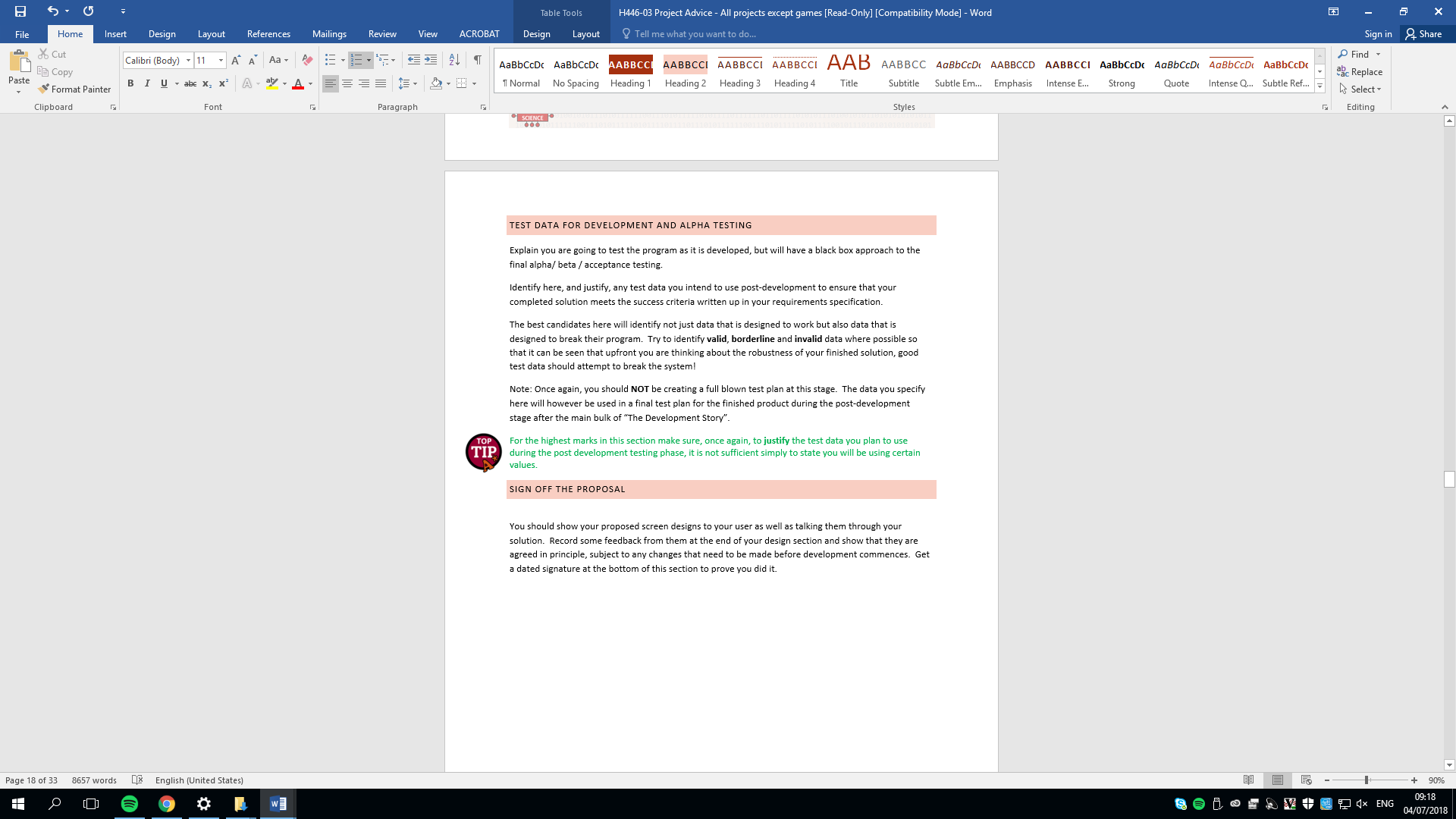
Graphical user interface, application, Word

Description automatically generated

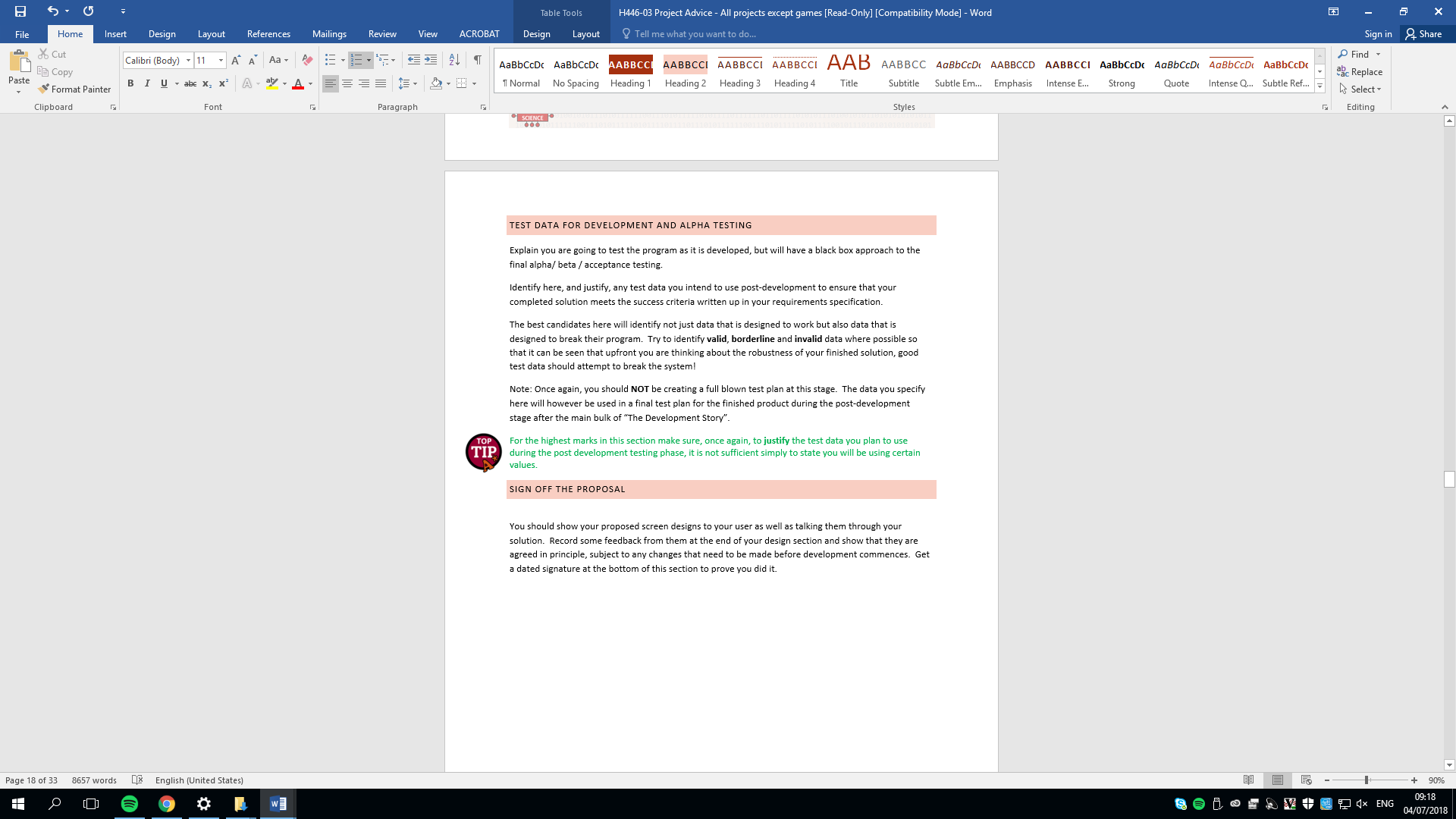
## Algorithms



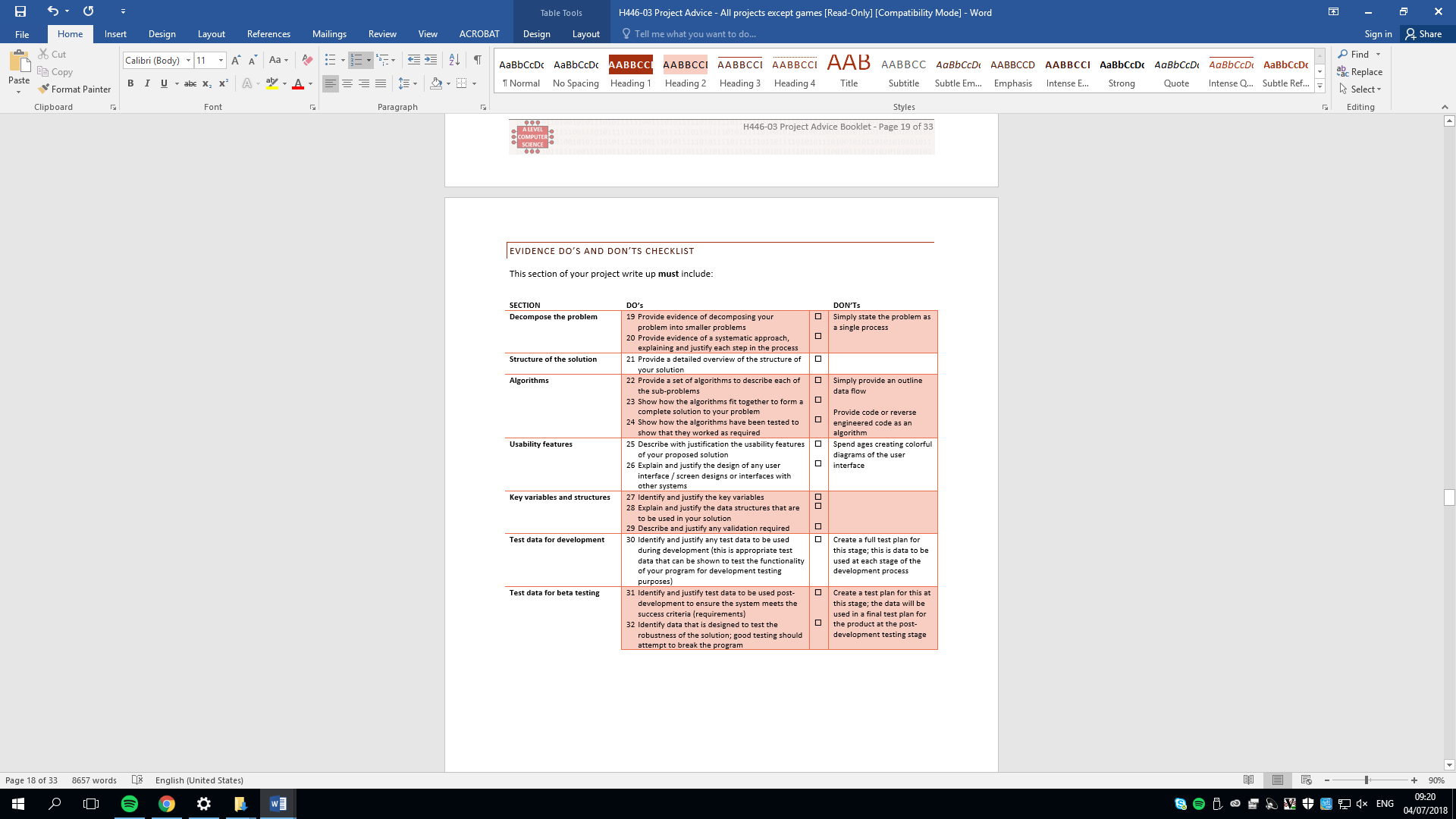
## post development test data



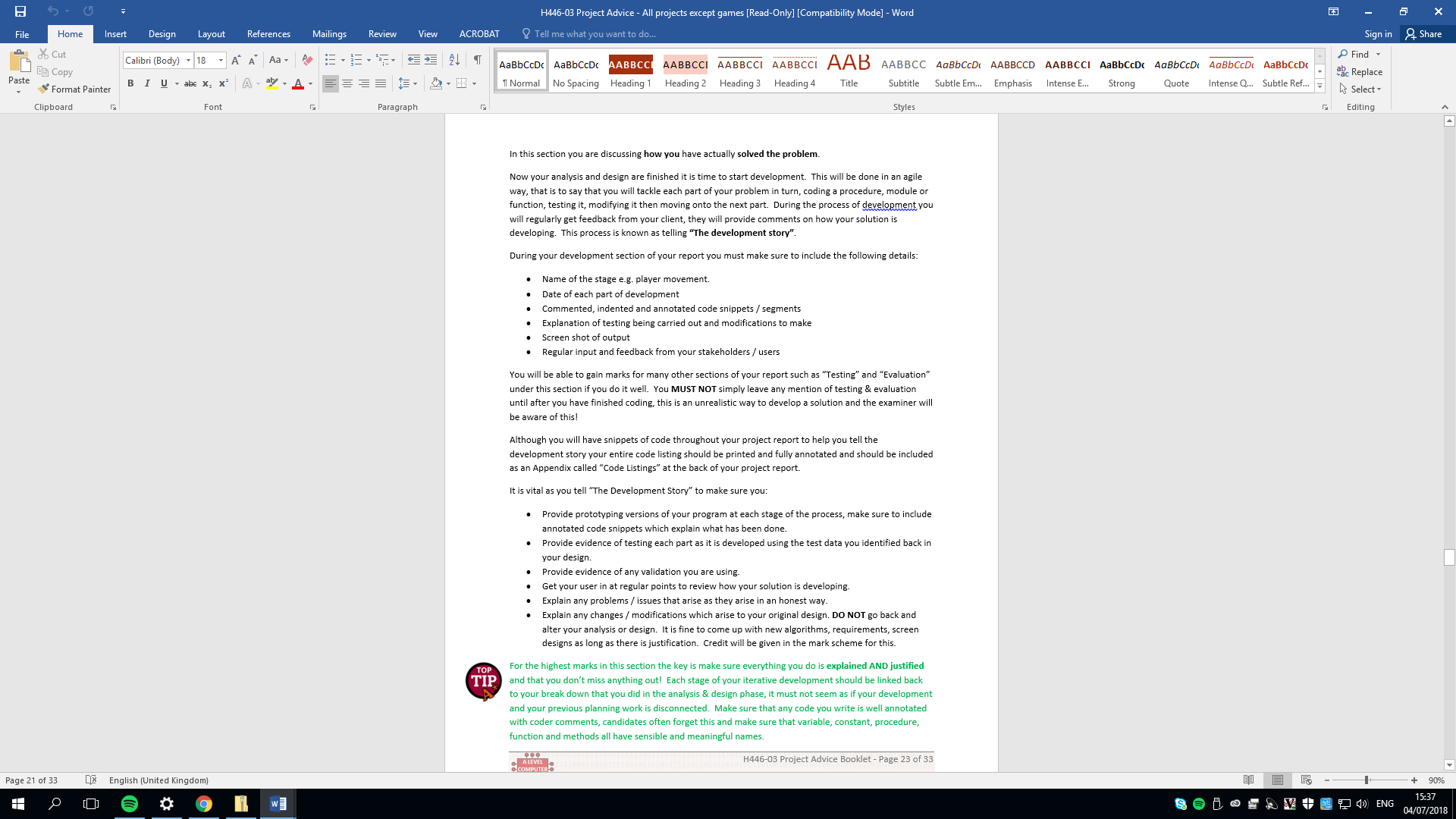
## Sign off proposal



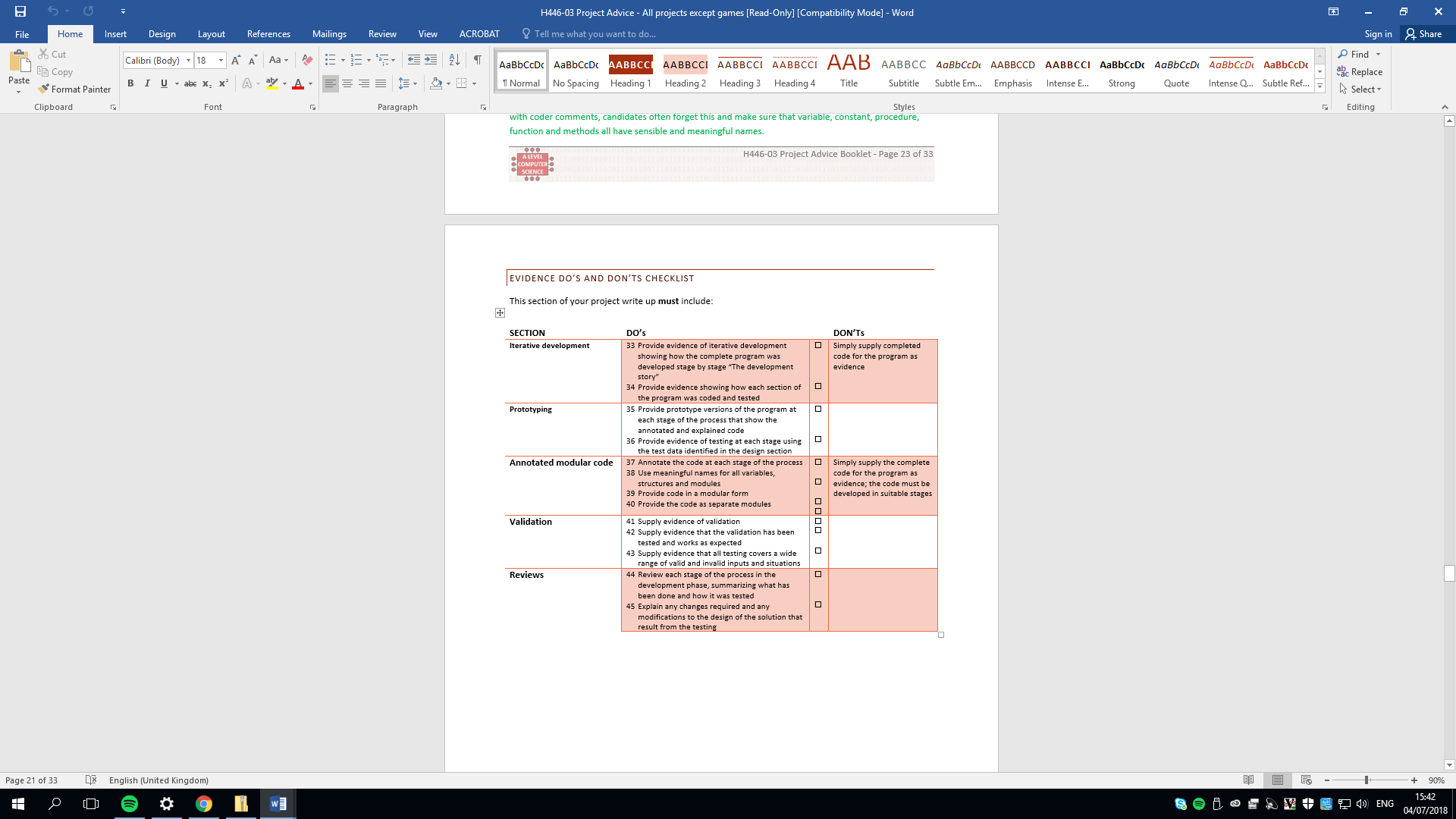
cHECKLIST – DELETE WHEN CHECKED



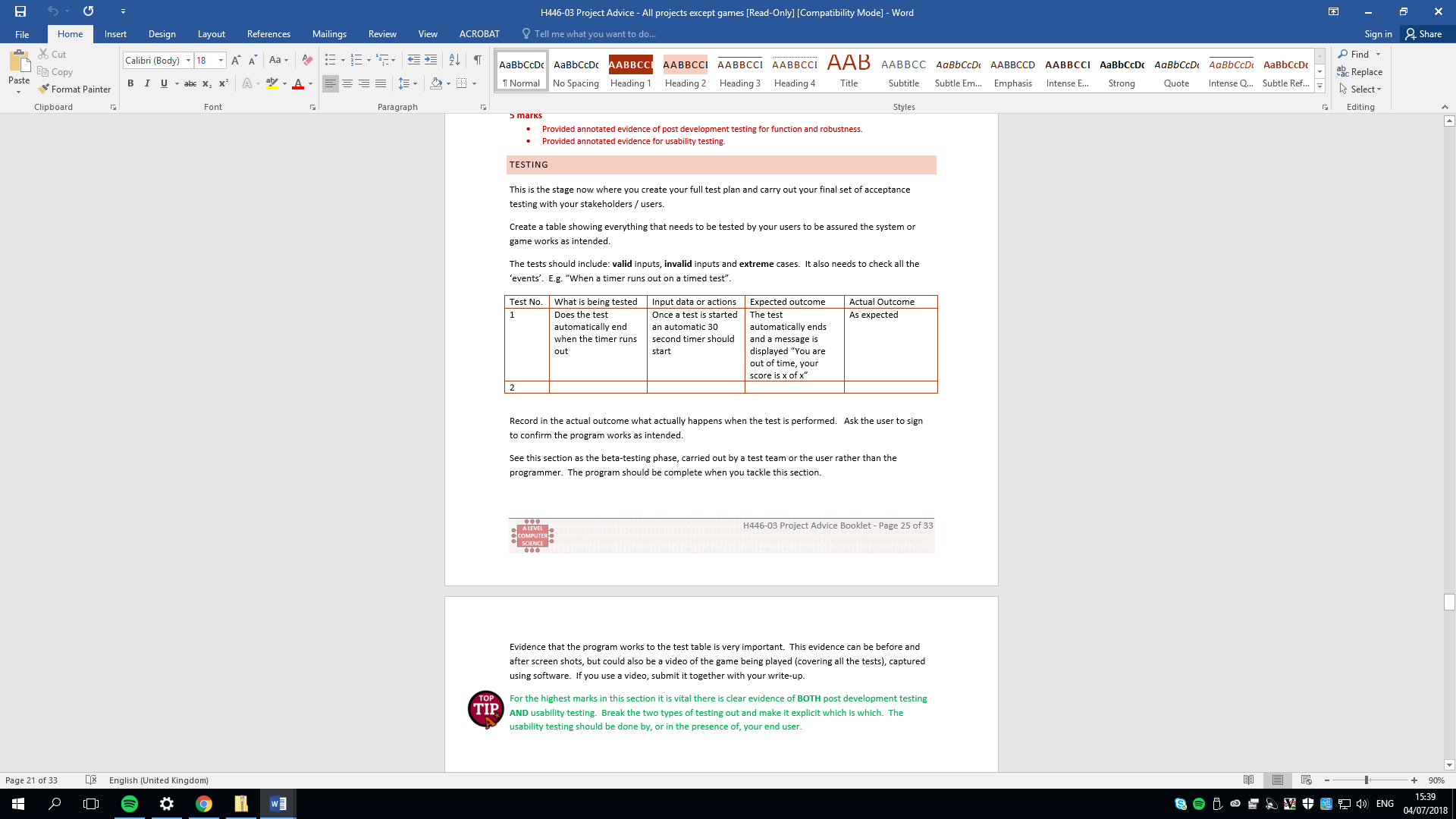
# Developing the coded solution (“The development story”)

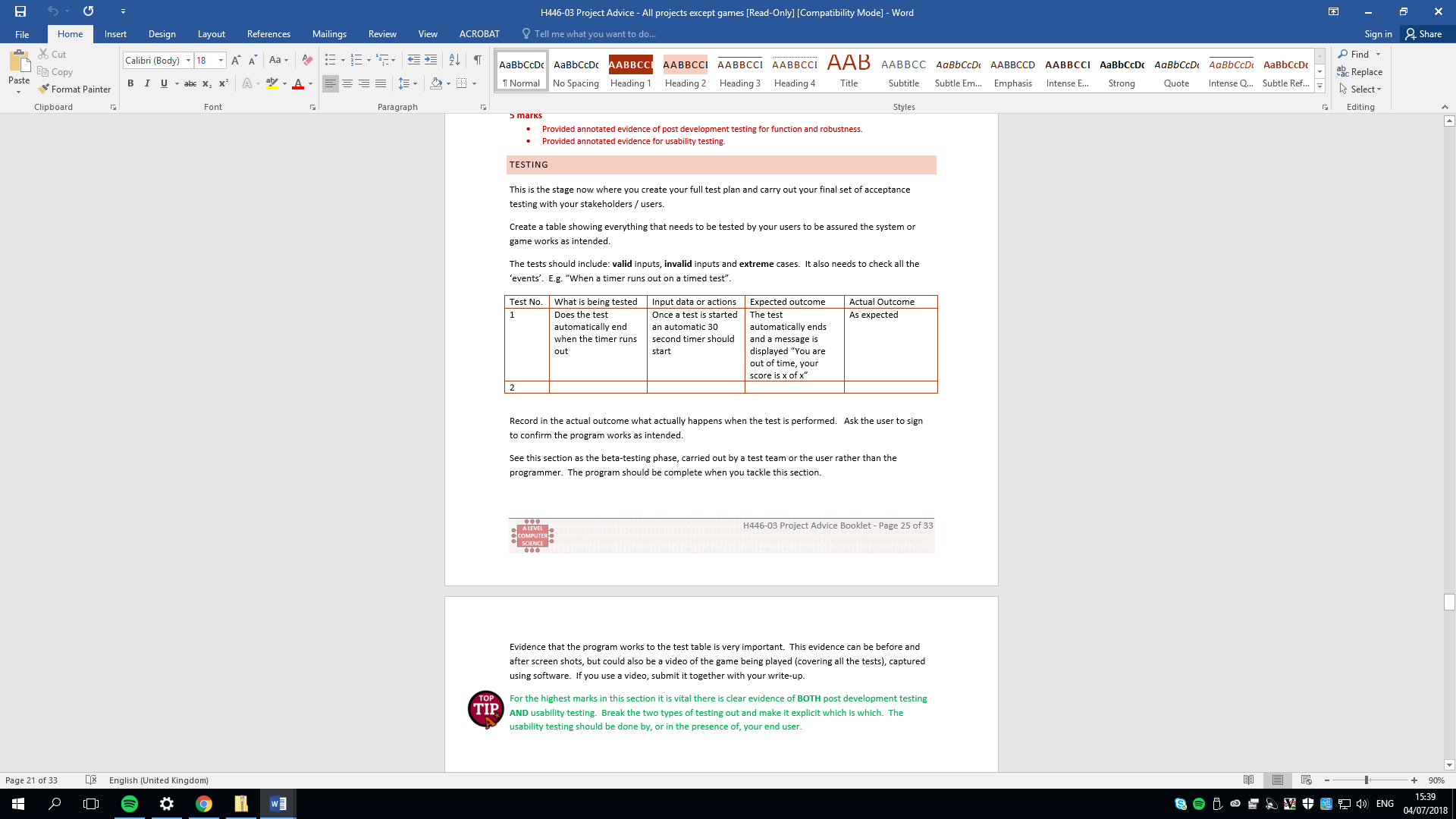


Final checklists delete

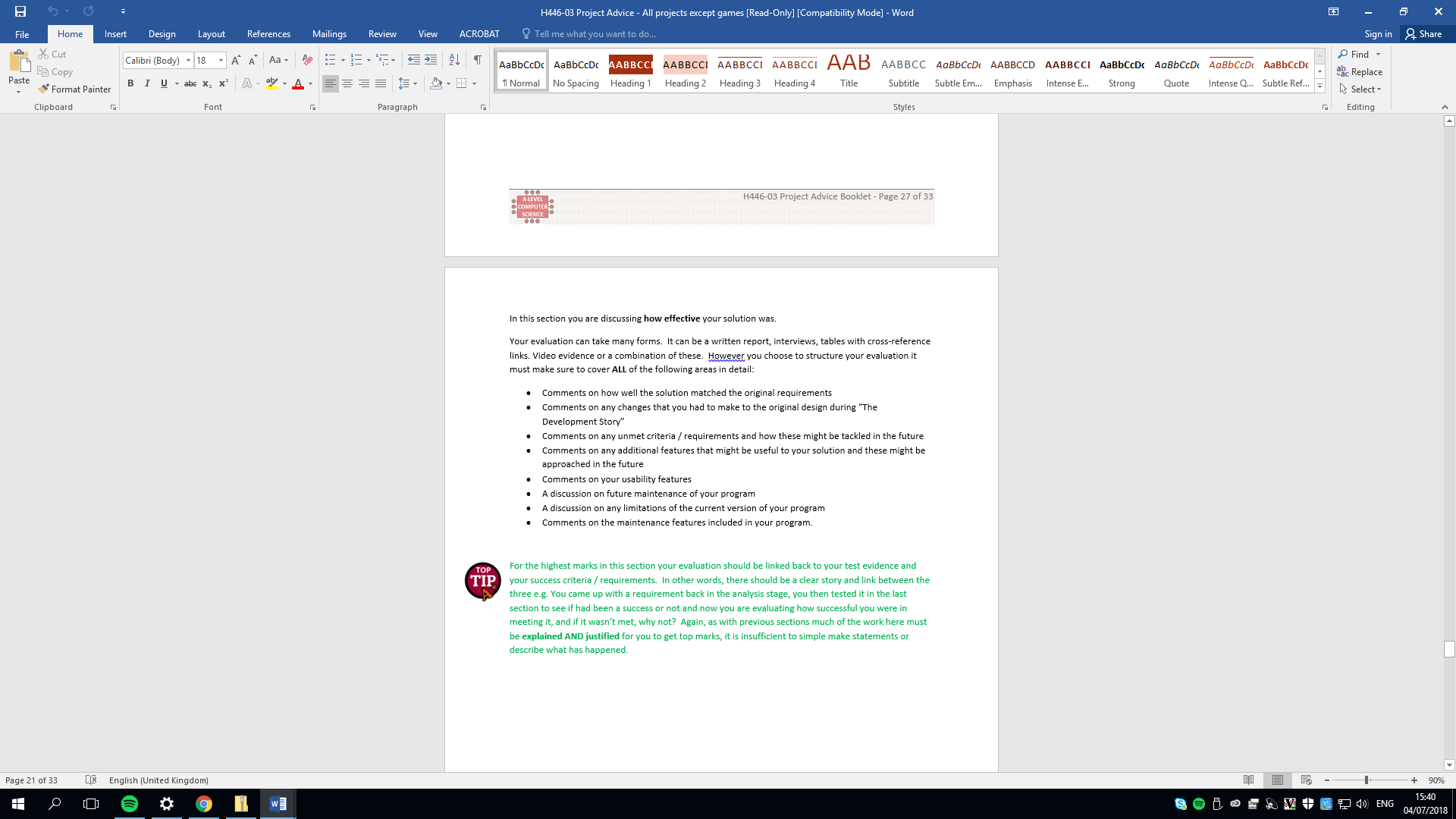


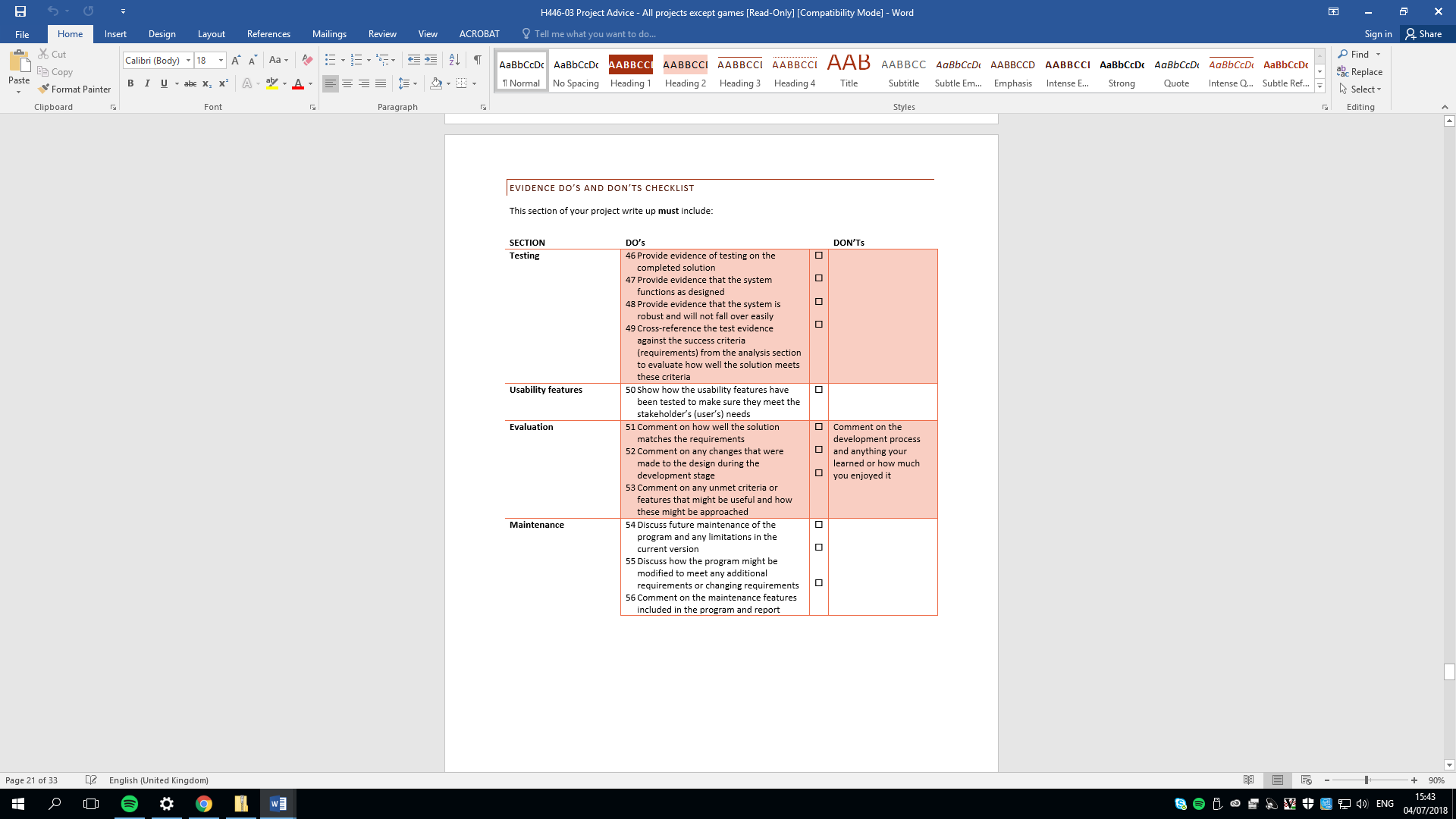
# Evaluation





## Evaluation-final solution





# Project Appendixes

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

* Complete Code Listing (ESSENTIAL)
* Interview Transcripts
* Meeting notes
* Observation notes or questionnaires