System analysis and design   
COMP 2311

Assignment 2: Unity Game Project

Submission Date: 7th May 2019

# Project Initiation

# Project Proposal

## Team Roles and Responsibilities

|  |  |  |
| --- | --- | --- |
| **Team Member Name** | **Major Responsibility** | **Minor Responsibility** |
| Kyle Candlin | Game Developer | Assistant System Analysist |
| Toby Parsons | Database/Mobile Developer  Project Manager | Tester |
| Simon Boyles | Systems analysist | Assistant Game Developer |
| Nathan Abbatiello | Game Developer | Tester |

## Aims

### Game

The project aims for the game are:

1. Create a moveable, playable character;
2. Create a 2-Dimensional side-scrolling game;
3. Create multiple, playable levels for the player to traverse;
4. Use in-game puzzles to progress through levels;
5. Develop the game in the unity engine.

### Mobile Application

The project aims for the Mobile Application are:

1. Create a minigame/series of minigames that interact with the main game;
2. Read and write files to a connector database;
3. Display player statistics to the user.

### Database

The project aims for the database are:

1. Allow connections from the mobile application and the game;
2. Store data from the game and mobile application.

### Scope

The system will be a game created in Unity with a mobile companion application as an add-on; these will both be connected to a database so information can be sent and accessed between both the game and the application. The game will be coded in C# because this is the native language of Unity and the mobile application will be created using Cordova and coded using web technologies (HTML, CSS, JavaScript and PHP) as this will be a hybrid application. The database will be coded in SQL and stored on Worcester Universities MySQL server.

The game will be created using DSDM (Dynamic Systems Development Method) which is an agile methodology. We chose to use an agile methodology because it allows us to re-iterate over designs and the implementation to develop improvements to both the game and the mobile application.

The game and mobile application will be stored on GitHub so will always be accessible by all team members. There will also be an Office SharePoint page set up so all group documents can be stored and shared. Both GitHub and SharePoint are stored using the cloud so all data will be kept secure and always accessible.

The system will be aimed at 11-14-year-old students who are learning the basics of logic gates and how they are used. The game will become progressively more and more difficult resulting in more challenging tasks for users to complete as time goes on.

The system will be completed by Friday 24th May 2019 with periodic intervals where development stages will be completed.

### Risk Assessment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Identifier: 01 | Description: Team member is ill | | | Risk category: Health |
| Probability: Unlikely | Impact: Project set back | | | Proximity: N/A |
| Countermeasures: Stay ahead of targets, eat healthy, stay active. | | | | |
| Contingency: Take medication, get rest | | | | |
| Owner: All members | Author: Simon and Kyle | Date identified: 23/01/2019 | Date of last update: 23/01/2019 | Current status: Static |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Identifier: 02 | Description: Computer failure | | | Risk category: Technical |
| Probability: Very Unlikely | Impact: Minor setback | | | Proximity: Distant Future |
| Countermeasures: Have anti-virus systems on, defrag regularly | | | | |
| Contingency: Use University computers, buy new computer/Fix computer | | | | |
| Owner: All Members | Author: Simon and Kyle | Date identified: 23/01/2019 | Date of last update: 23/01/2019 | Current status: Dead |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Identifier: 03 | Description: Lack of programming language knowledge | | | Risk category: Technical |
| Probability: N/A | Impact: Slowed progress in coding game, more mistakes, less efficient code | | | Proximity: Now |
| Countermeasures: Learn code syntax | | | | |
| Contingency: Use tutorials | | | | |
| Owner: Nathan, Toby | Author: Simon, Nathan and Kyle | Date identified: 23/0/2019 | Date of last update: 23/01/2019 | Current status: Reducing |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Identifier: 04 | Description: Car breaks down/ Failure | | | Risk category: Mechanical |
| Probability: Unlikely | Impact: Cannot attend meetings, reduces communication | | | Proximity: Near Future |
| Countermeasures: Route planning, maintain car | | | | |
| Contingency: Call break down service, take public transport, work from home, tell group | | | | |
| Owner: Simon, Kyle, Nathan | Author: Simon and Kyle | Date identified: 23/01/2019 | Date of last update: 23/01/2019 | Current status: Reducing |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Identifier: 05 | Description: Disagreement | | | Risk category: |
| Probability: Likely | Impact: Slow progress, reduce teamwork and friendship, Reduce efficiency | | | Proximity: Near Future |
| Countermeasures: Plan ahead, Good communication | | | | |
| Contingency: Find resolution, | | | | |
| Owner: All members | Author: Simon and Kyle | Date identified: 23/01/2019 | Owner: All members | Author: Simon and Kyle |

# Analysis

## Functional Requirements

## Must

1. The game must be puzzle based; players must solve puzzles to advance through the game and progress through levels.
2. The game must use logic gates as the problem-solving technique. The problems shown to the user in-game must all be solvable using a range of different logic gates; made available to the user in-game.
3. The system must include a range of logic gates will be available for the user to use in-game. The logic gates will be used to solve puzzles and progress through the game. There must be three logic gates for the user to use.
4. The game must include player statistics. The user should be able to see the playable characters health and stamina; the user should have a visual representation of the playable characters statistics so they can make decisions in game.
5. The game must include multiple playable levels. There must be more than one level that the user can traverse; each level should be unique; presenting different challenges each time with different user interactions.
6. The game must include player movement. This is so the user can navigate around the 2 Dimensional in-game canvas to complete levels and progress through the game.
7. The game must include a double jump feature. The player can double jump when double pressing the jump key. This will allow the player to gain extra height and can be used to solve puzzles in-game.
8. The game must include a mobile companion application. This is so the user can interact with the game on multiple platforms. The companion application will include minigames that the user plays to gain items for in-game use.
9. The system must include a database that connects the mobile companion application to the game. This must store log in details, level details, item details and player statistics.
10. There must be combat element in the game. This is so the user has more obstacles to face while playing the game and, therefore, increasing the difficultly and diversity of the game.
11. The game must include enemies that the user can inflict damage upon. This is so the user can destroy the enemies and obtain the items dropped.

## Should

1. The game should have an in-game ‘hotbar’. This is so the user can access the players inventory through a selection of on-screen buttons. This allows the user to quickly access the items they need.
2. The game should include a story. A story for the game will add a unique feel and will make the game less repetitive. The story should be simple but should be applied throughout the game.

## Could

1. The game could have an inventory system. This would allow the player to pick-up and store a range of different items. These items should also be able to be dropped out of the inventory.
2. The game could have an on-screen damage indicator. This will be a visual representation to the user that the playable character has taken damage.
3. The game could use more logic gates in the game; this would be increased from three to five. This could increase level complexity.

## Would

1. A requirement that the game could have would be increase the logic gates used in-game to seven instead of three or five. This would be included if there was more time available for project development.

## Non-Functional Requirements

1. The system design must take into consideration colour blindness. This means that colours that conflict should not be used in conjunction with each other as this could present problems to users.
2. There must be a main menu that the user loads into when they start the game. This increases usability in the game.
3. The game must include a pause menu. This is so the user can pause and resume the game via an in-game menu; this could be created to work in unison with the inventory system.
4. The user should be able to change the in-game controls. This allows the user to personalise the game to tailor to their needs.
5. The user should be able to alter their graphics and sound setting. This would include game resolution and volume of the game.
6. The database must keep all data stored secure. This is so GDPR (General Data Protection Regulations) can be maintained and monitored. This would include secure data back-ups and password hashing.
7. The system loading times should always be less than three seconds. This can be completed by reducing image quantity on the mobile application and by creating efficient code in the game.
8. The code behind all aspects of the system should be well structed and commented correctly. This makes code maintenance and further development easier as each script, and the contents, can be located with ease.

# Analysis Modelling

## Data Flow Diagram

This shows the flow of data for the system.



## Assumptions

* Usernames must be unique,
* Max number of items is controlled in the game.

# Project Management

## Gantt Chart

The Gantt chart is used to manage our time, so we stay on target for the set deadlines. It gives a visual representation of when a task should start and end. As well as which team member(s) are assigned. The tasks are grouped into the development life cycle sections.

## Minutes of meetings

Date: 23/11/2018 Attendance: All

What was completed:

A1 Form completed and handed in

Group Project Discussed

Date: 30/11/2018 Attendance: All

What was completed:

A1 Form was accepted and looked over

A2 Form handed in

Group Project Discussed

Date: 07/12/2018 Attendance: All

What was completed:

A2 Form Accepted

Start on Project

Group planning of ideas around game

Started coded of base game to get basic wireframe and functions working

Date: 14/12/2018 Attendance: All

What was completed:

Talk with stakeholder to determine date for requirements for system, date set for 25/01/2019

Date: 21/12/2018 Attendance: All

What was completed:

Completed base game code ready for stakeholder meeting

Date: 25/01/2019 Attendance: All

What was completed:

Stakeholder requirements meeting completed: Requirements as follows

* Story Progression
  + Writing a narrative
  + Theme decided with narrative
    - Futuristic Theme
  + Stakeholder opens to interpretation
  + Topical story – physical progression through environment
    - Programming concept learning tool
    - “Tron esc’
    - Puzzle system (and or gates, with final puzzle etc.)
    - Boolean logic problems
    - Locked doors/chest puzzle (scalable to progress)
    - Inventory used for in-game purpose

Frame story. Escape with Boolean logic puzzles, rationale for presenting logic problems in context.

Companion App:

* Puzzles on app
* Increase resources with application
* Locational collection

Date: 01/01/2019 Attendance: All

What was completed:

Stakeholder requirements start to be developed. Implementation of more features into the game including: Double Jump, Enemy Script Started Development. Documentation including requirements and project initiation sections completed

Date: 07/01/2019 Attendance: All

What was completed:

Aims and scope of the project added to documentation

Addition of game main menu and currency system

Date: 14/02/2019 Attendance: All

What was completed:

Started the development of database and application side of the project. Databases set up and linked to the application. Testing carried out

Date: 21/02/2019 Attendance: All

What was completed:

More development on player movement and enemy movement within the game. Addition of particle systems as enemies and weaponry researched and started.

Mobile UI development started and edited

Date: 28/02/2019 Attendance: All

What was completed:

Link of the database to the application and database. Creating a complete loop between the two systems. Logic gate puzzle system on game updated and start of UI design.

Project Management Gannt chart completed for current progress

UI designs including wireframes for app and game completed. Phase 1

Date: 08/03/2019 Attendance: All

What was completed:

Game:

Puzzle Logic gate updated, and more code added. Started research into graphics packs for the game.

Analysis modelling carried out and diagrams completed

Application:

More UI code completed, addition of login script

Date: 15/03/2019 Attendance: All

What was completed:

Game:

Puzzle Logic gate updated, and more code added. Started research into graphics packs for the game.

Analysis modelling carried out and diagrams completed

Application:

More UI code completed, addition of login script

Documentation:

Design Section completed

Date: 22/03/2019 Attendance: All

More data modelling completed including menu maps and class diagrams all descriptions added. Sequence diagram completed. ERD completed. Data dictionary Completed.

UI designs including wireframes for app and game completed. Phase 2

Game starting to be finalised with most functions added and UI elements now added including graphics packs. App and Game fully linked with database.

Date: 29/03/2019 Attendance: All

Full Testing of both application and game completed in testing tables. White and box completed.

Date: 06/04/2019 Attendance: All

Documentation mostly complete and game and app exported with little code being added

Date: 26/04/2019 Attendance: All

Final stages of documentation added. Game and App checked and database implementation checked also.

Date: 03/05/2019 Attendance: All

Final checks occurring

Zipping of all files together. Reading through documentation to check for spelling mistakes and grammatical errors.

Creation of PowerPoint for showcase.

## Conflict resolution and communication

To manage conflict within our group, we would have group meetings to discuss any changes within the system. This allows us to keep all team members up to date on the systems progress and reduce any conflict that could arise.

To communicate between group members, we used Emails, Facebook Messenger and OneDrive. This allows us to stay in touch and share resources. To communicate with stakeholders and lecturers, we set up meetings via Email. In the meetings, we updated our stakeholder on the systems progress as well receive feedback on the current version.

# Design

## Story

The story of the game will be a player that is inside a computerised system. His goal is to defeat the virus invading the system and to do so he must advance through the system; solving puzzles and defeating small viruses on the way.

## Design Considerations

### System Type

Trojan is a casual learning tool that can be played by anyone currently learning or with knowledge of Boolean logic gates and their functionality. The game will use player movement, enemy interactions and Boolean logic puzzles. The mobile application will be used in conjunction with the game but is not necessary for game progression, the application allows the user to gain additional logic gates for in game use by playing the minigame.

### System Controls

The player in the game can be controlled using ‘A’, ‘D’ and ‘Spacebar’ and the left mouse button. ‘A’ and ‘D’ move the player left and right across the screen and ‘Spacebar’ allows the player to jump; left mouse click activates the player attack which can be used to damage enemy players.

The mobile application uses screen taps to gain ‘points’ which are traded in for logic gates.

### System Builds

The game will be developed for PC and mobile application will be built for Android devices.

The game will be developed using the Unity Engine and the app will be developed using Apache Cordova.

## Program Flow Diagram



### Assumptions

* The user has already created an account,
* The user is already logged in.

## Menu Map



This menu map shows which pages are connected. This shows which pages can be accessed from other pages and the route that a user might use to get to a desired page.

## Game Modelling Technique – Use Case Diagram



### Written Use Case

The user must first login to the game, if the user does not have an account then they must register. The user is then taken to the menu screen where they have five options; Play Game, Settings, Codex, Switch Account, Exit.

If the user chooses ‘Play Game’, then they will be taken to the first level where they must defeat enemies to gain logic gates which are used to complete puzzles and complete the level. The logic gates amount is sent to the database so they can be displayed on the mobile application.

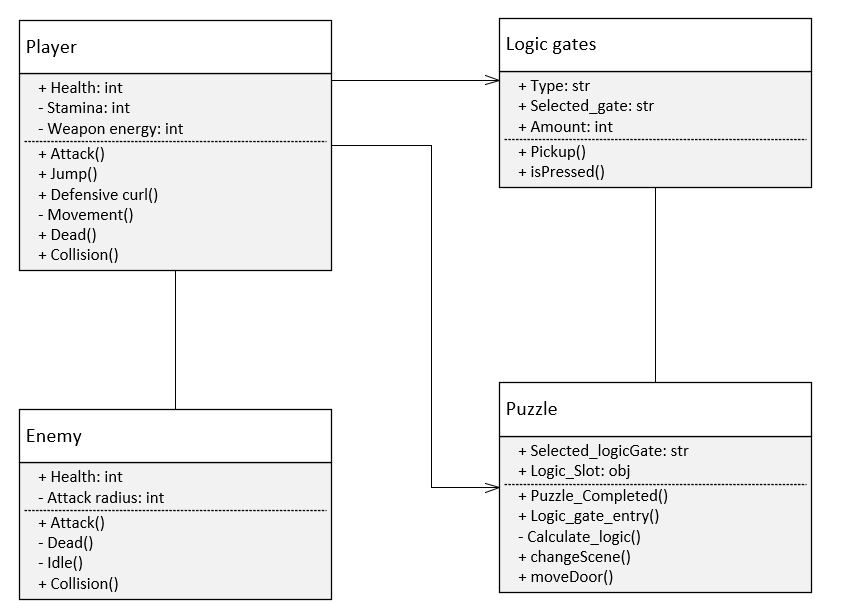
If the user chooses ‘Settings’, they will be taken to a screen which allows them to alter game volume.

If the user chooses ‘Codex’, they will be shown information about the logic gates the game includes and how they can be used.

If the user chooses ‘Switch Account’, they will be logged out and taken back to the login screen.

If the user chooses ‘Exit’, they will be logged out and the game will close.

## Game Modelling Technique – Class Diagram



This class diagram shows the relationship between the player, the enemy, the puzzle and the logic gates. This class diagram also details each classes objects and methods.

## Mobile Application Modelling Technique – Use Case Diagram

### Written Use Case

The user must first login to the mobile application. Once they are logged in, they will have three options; Play Minigame, Buy Log Gates, Buy Cosmetics.

If the user chooses ‘Play Minigame’, they will be taken to a new page where they play a small game that earns the player coins which is used as currency.

If the user chooses ‘Buy Logic Gates’, they will be taken to a page displaying options for them to purchase logic gates for a specified price.

If the user chooses ‘Buy Cosmetics’, they will be taken to a similar page except it will show cosmetic items to purchase which will not affect gameplay.

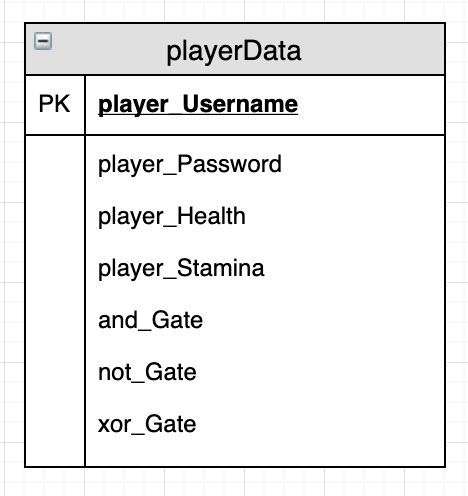
## Mobile Application Modelling Technique –Sequence Diagram



This sequence diagram shows what tasks will be executed by the user, between the mobile application and the database. It also shows what order the tasks will be executed in. The narrow boxes on the lines show when that object is in use. If there is no box, then the object is not in use and is no involved in the current processes.

## Database Modelling Technique

### Entity Relationship Diagram



This Entity Relationship Diagram gives a visual representation of what tables will be created and how they will be related. For our project, only one table is necessary, so this does not have any relations.

### Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute | Description | Type | Length | Constraints | Example |
| player\_Username [PK] | The primary key of the table. | String | 15 | NOT NULL | Test |
| player\_Password | The password for the user; hashed and salted for security. | String | 15 | NOT NULL | test |
| player\_Health | The health value for the player. | String | 6 | 0 < x < 100 | 65 |
| player\_Stamina | The stamina value for the player. | String | 6 | 0 < x < 300 | 165 |
| and\_Gate | The amount of and gates available for use. | Integer | 2 | 0 < x < 99 | 65 |
| xor\_Gate | The amount of xor gates available for use. | Integer | 2 | 0 < x < 99 | 65 |
| not\_Gate | The amount of not gates available for use. | Integer | 2 | 0 < x < 99 | 65 |

This data dictionary shows what each attribute of the table will be and its details. The data dictionary shows; the type of variable it will be, the length, the constraints, an example of the attribute and a small description.

## Assumptions

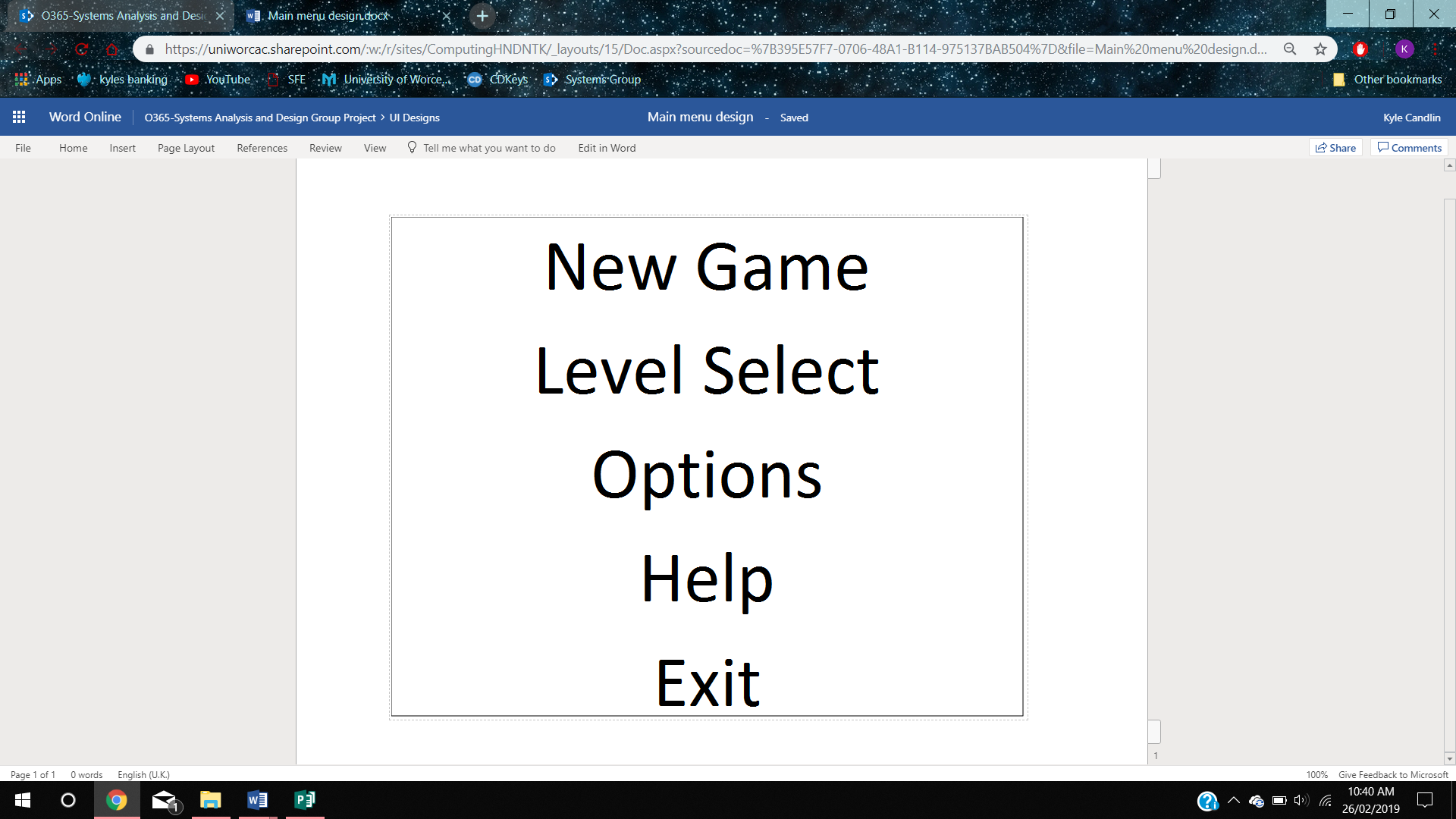
1. A Username is unique to each user
2. Player currency is stored locally in the application

## UI Designs (Wireframes)

### First Designs

#### Game

##### Main Menu



This is the first design we created for the main menu. This design is simplstic and is lacking both colour and theme but we able to fill the requirements by including all the necessary components. We decided to replace the ‘switch account’ button with a ‘level select’ button as we felt a level select would be more useful to the user.

if (newGameButton = pressed)

Level = 1

Load New Game

else if (levelSelectButton = pressed)

if ( chosenLevel = 1)

set chosenLevel = 1

else if ( chosenLevel = 2)

set chosenLevel = 2

else if ( chosenLevel = 3)

set chosenLevel = 3

Load chosenLevel

else if (optionsButton = pressed)

Load Options Menu

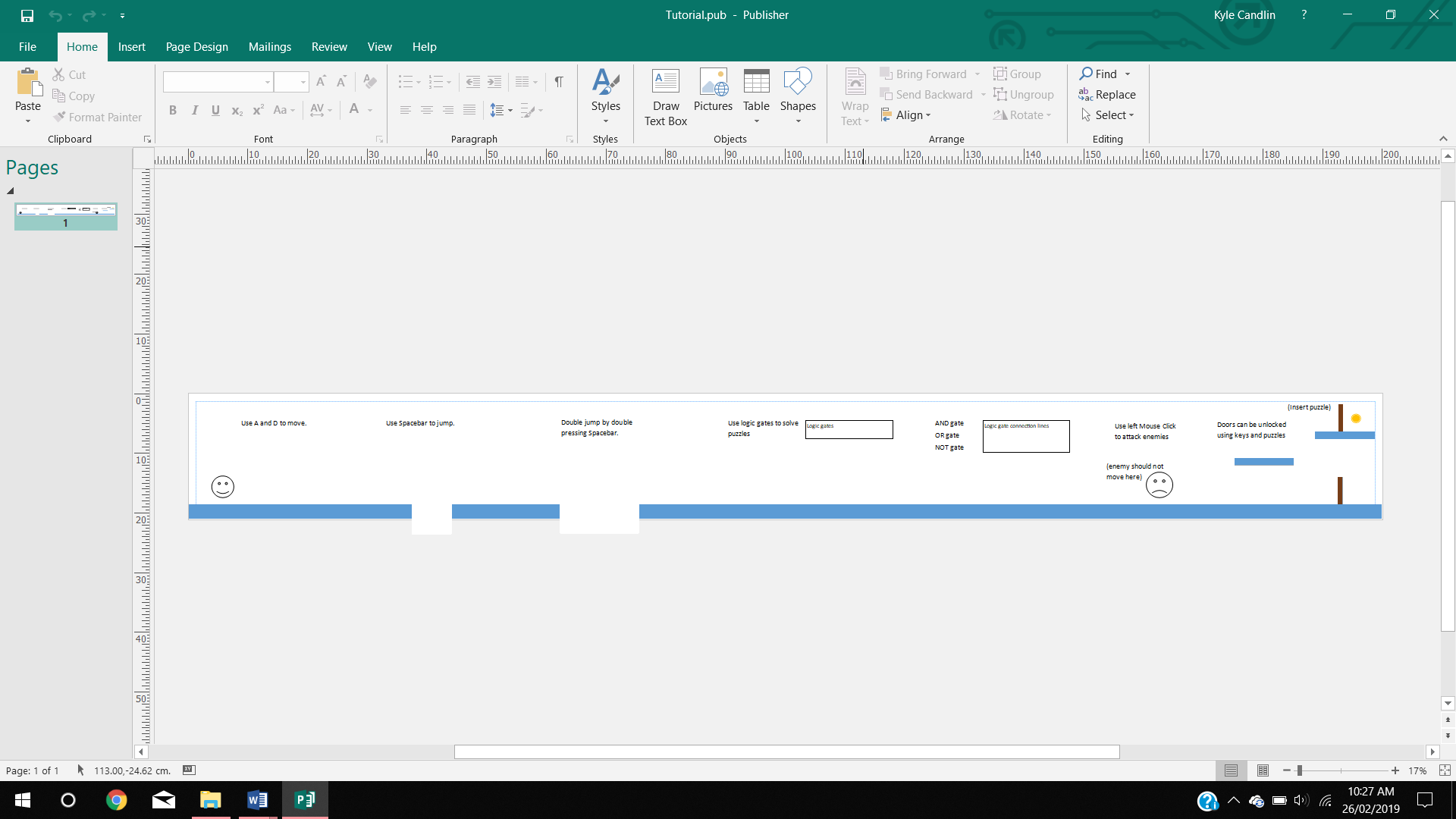
else if (helpButton = pressed)

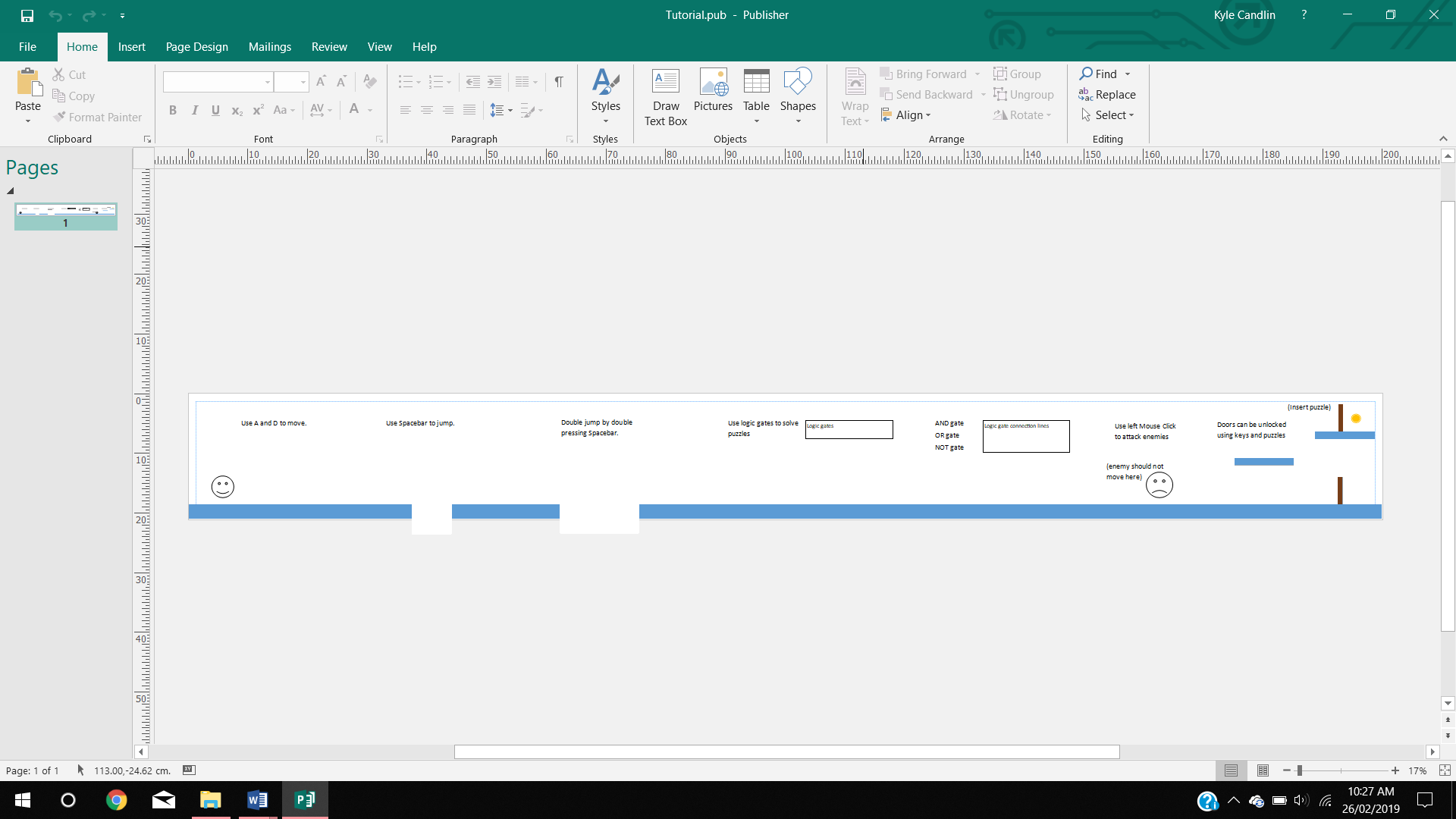
Load help page

else if (exitButton = pressed)

close application

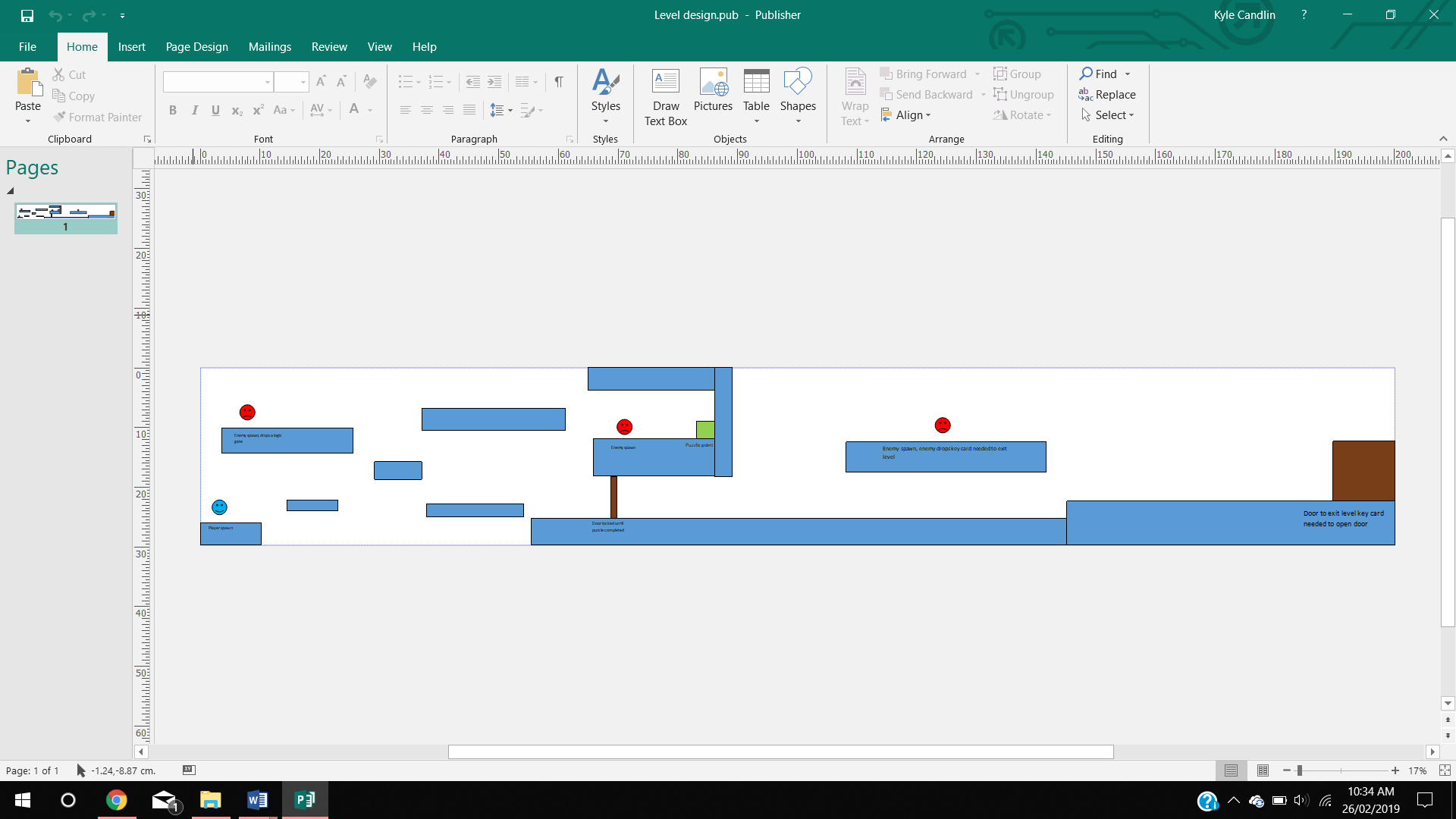
##### Tutorial

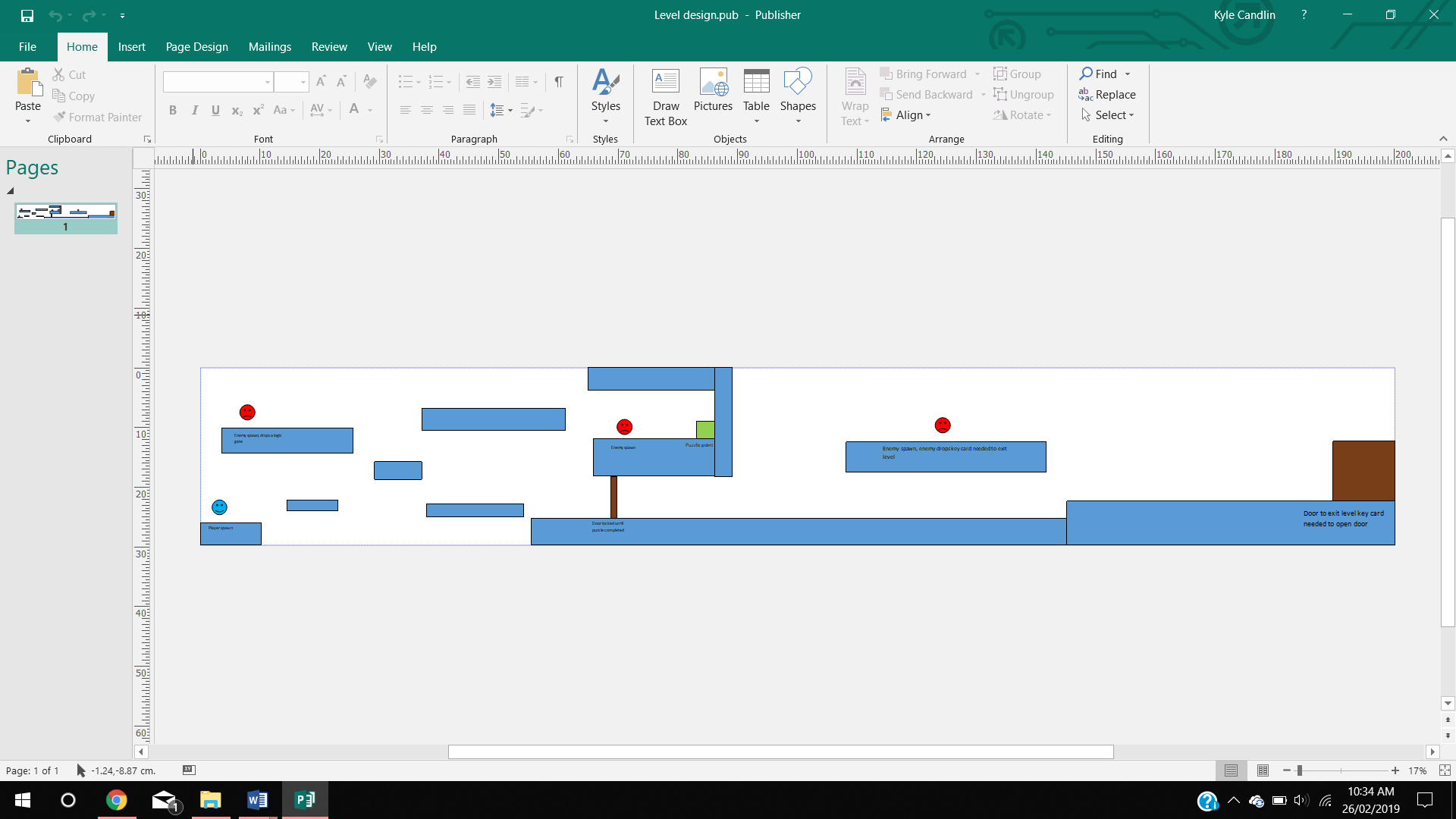




This is the first design for the tutorial level of the game. The tutorial level is simple for the player to follow and includes all the main aspects of the game; teaching them how to play. The design itself, once again, lacks in colour and theme but still fills all the requirements set.

##### Level 1

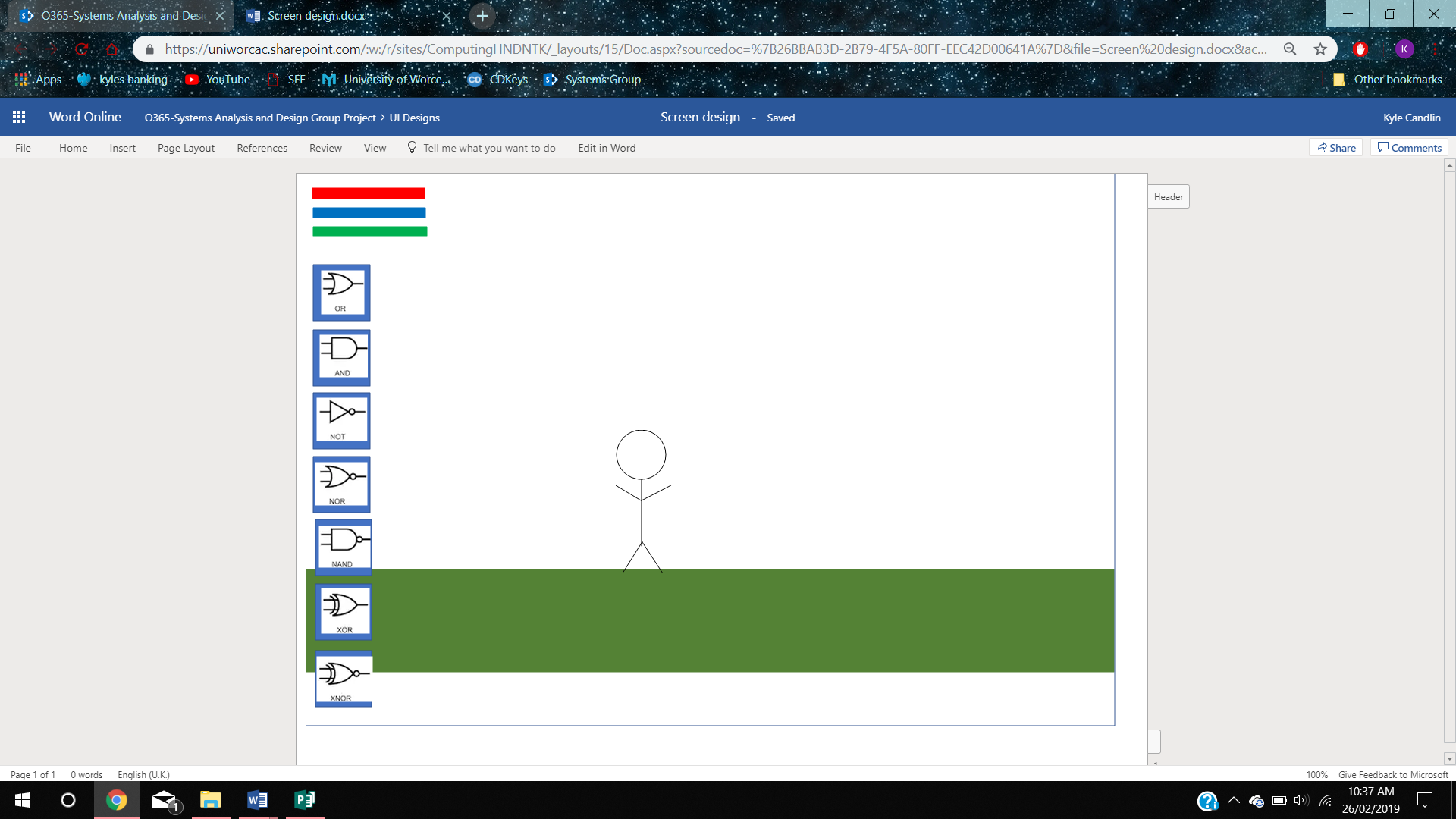




Here is the first design for the first level of the game. It is similar to the tutorial level in the way it played but is less helpful to the player. The enemies in this level can move and inflict damage and the player will not be helped to complete the objectives.

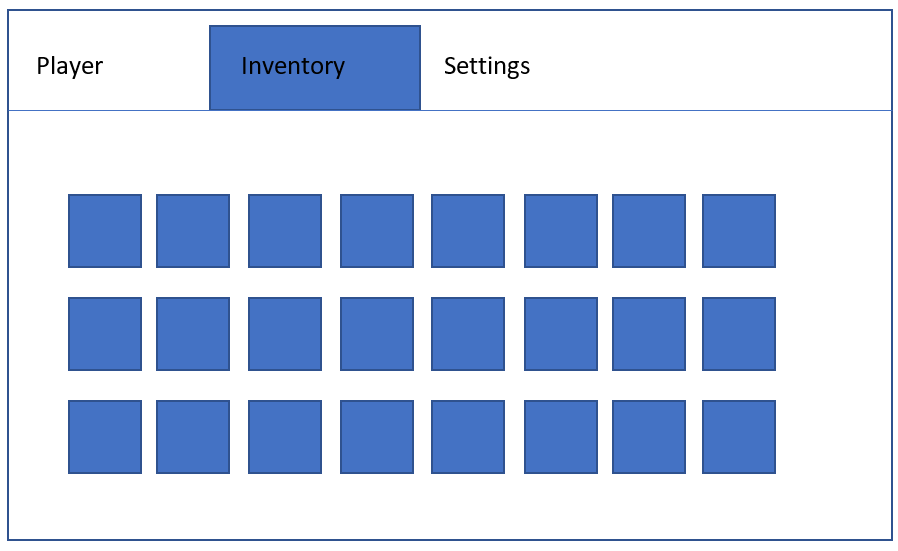
The colour and theme of the level are once again not present, but it does fill the requirements set to us and creates an immersive playing field for the player.

##### Camera View

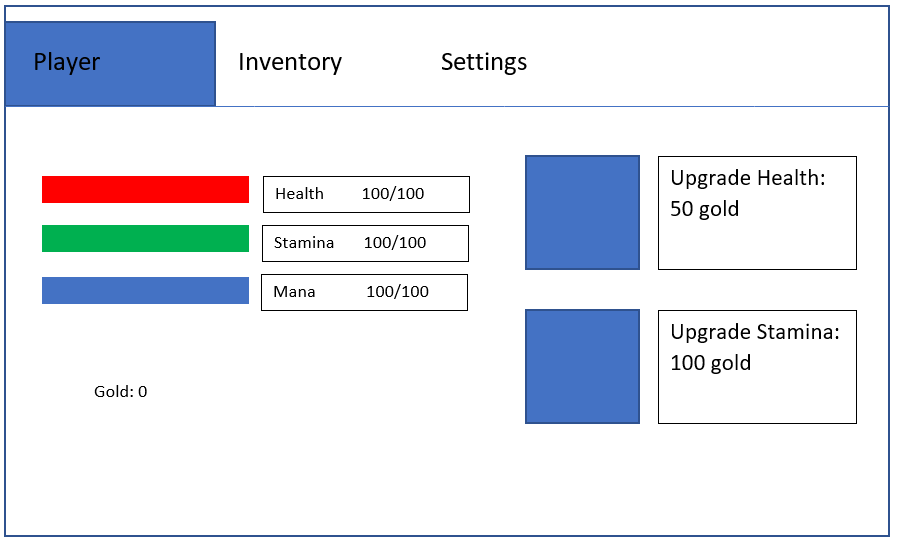


Here is our design for the camera view for the game; this is what the player will see when they are playing. All UI elements will be placed on the left including the loss of logic gates and the amount the player has at that current time and the amount of health and stamina the player has. The player will then be in the centre of the screen.

##### Inventory System



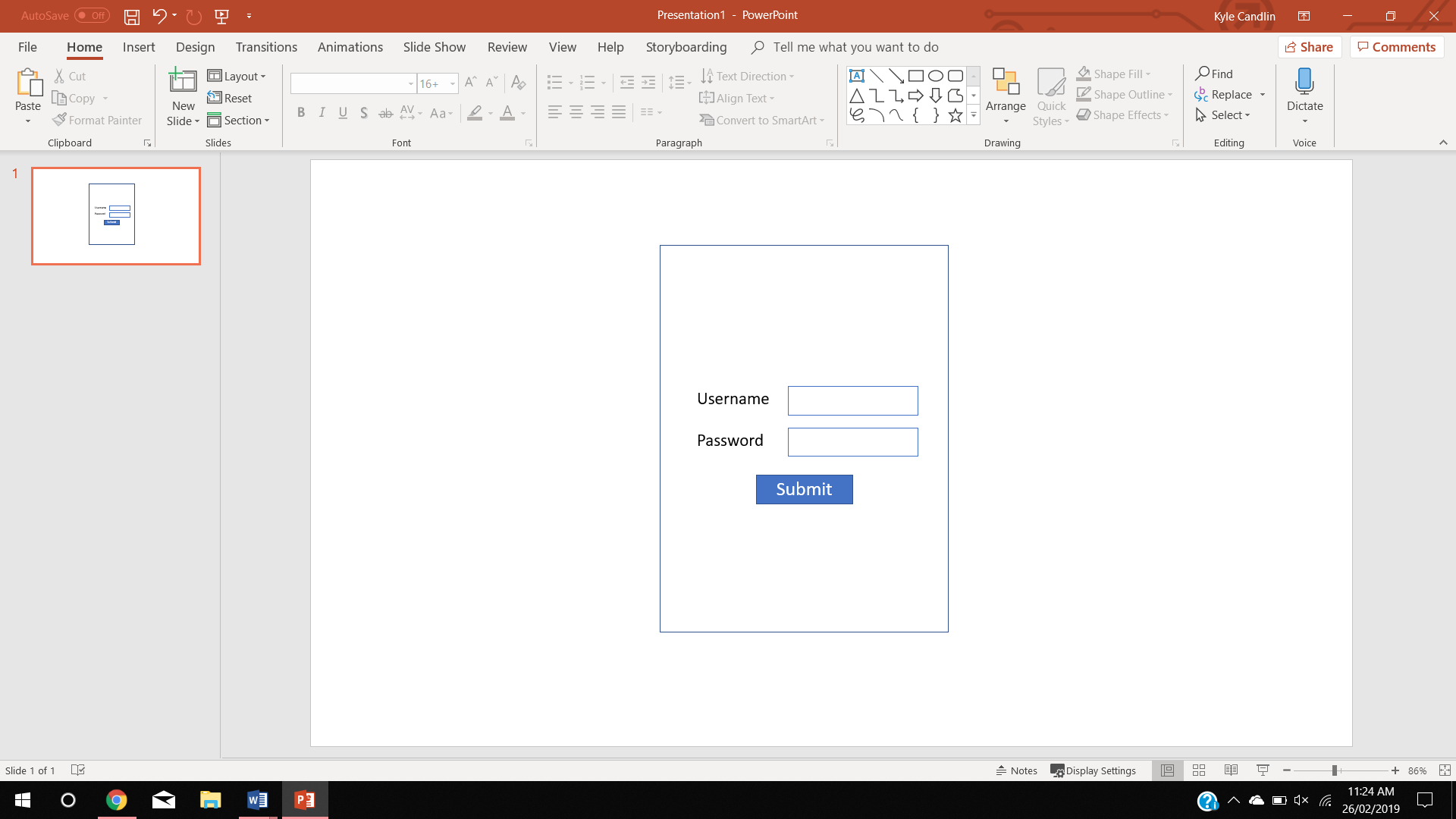
This is the design for the inventory system. This is where items the player picks up will go. From here the player will be able to move items about in the inventory an also use items such as health potions.



This is the design for the player details screen. This screen shows the player statistics such as their current health, stamina, mana and gold. These update in real time. This is also where the player can spend gold to upgrade their max health and stamina.

#### Mobile Application

##### Log in



If (submitButton pressed)

If((usernameTextBox != “” or white space or in database) and passwordTextBox = correct)

Login

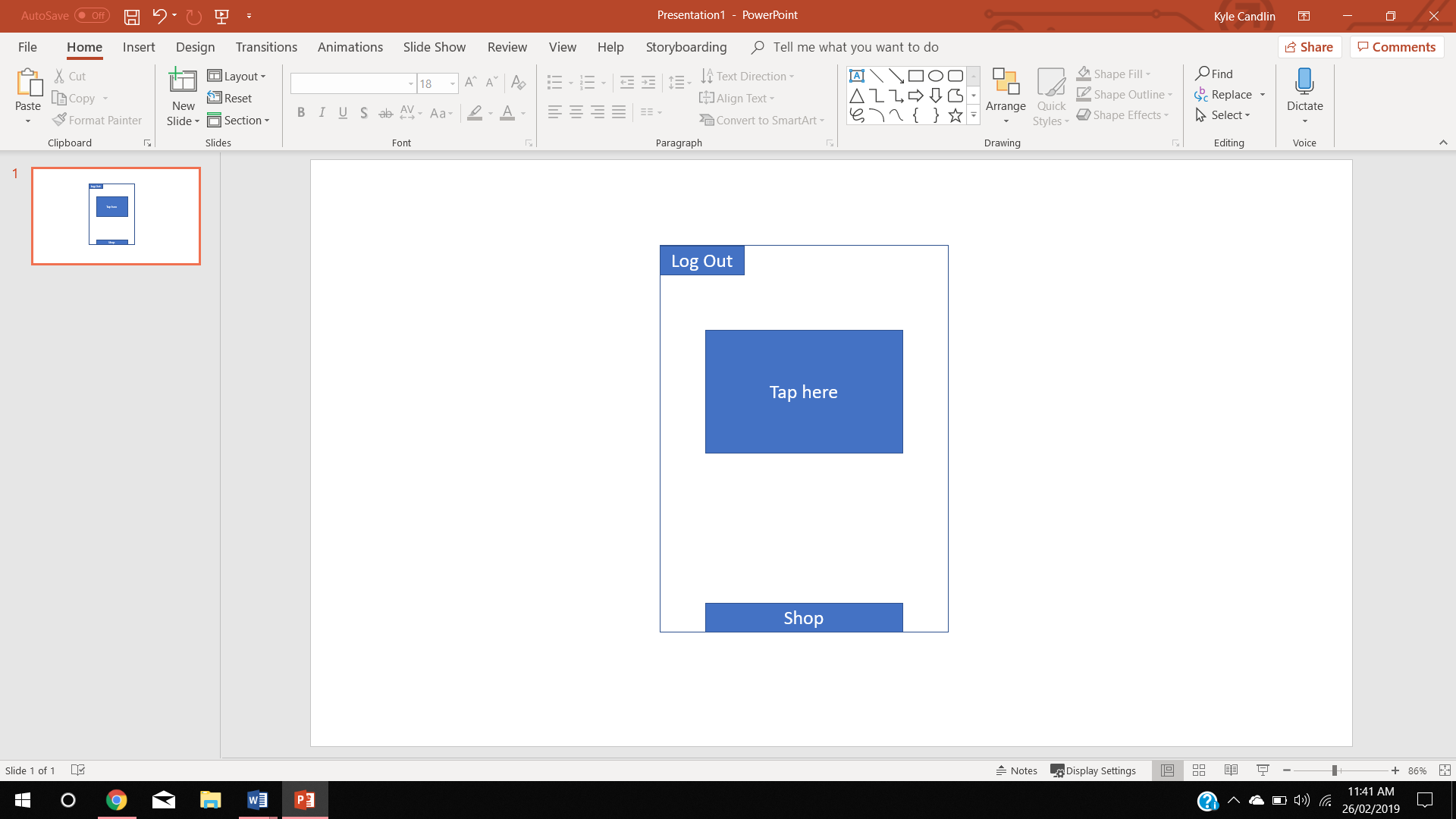
Set gates = database values

Else

Print error

This is the first design we had for the log in page on the mobile application. This page has a simplistic design, much like the game designs, but includes all the relevant boxes for the user to log in to the game.

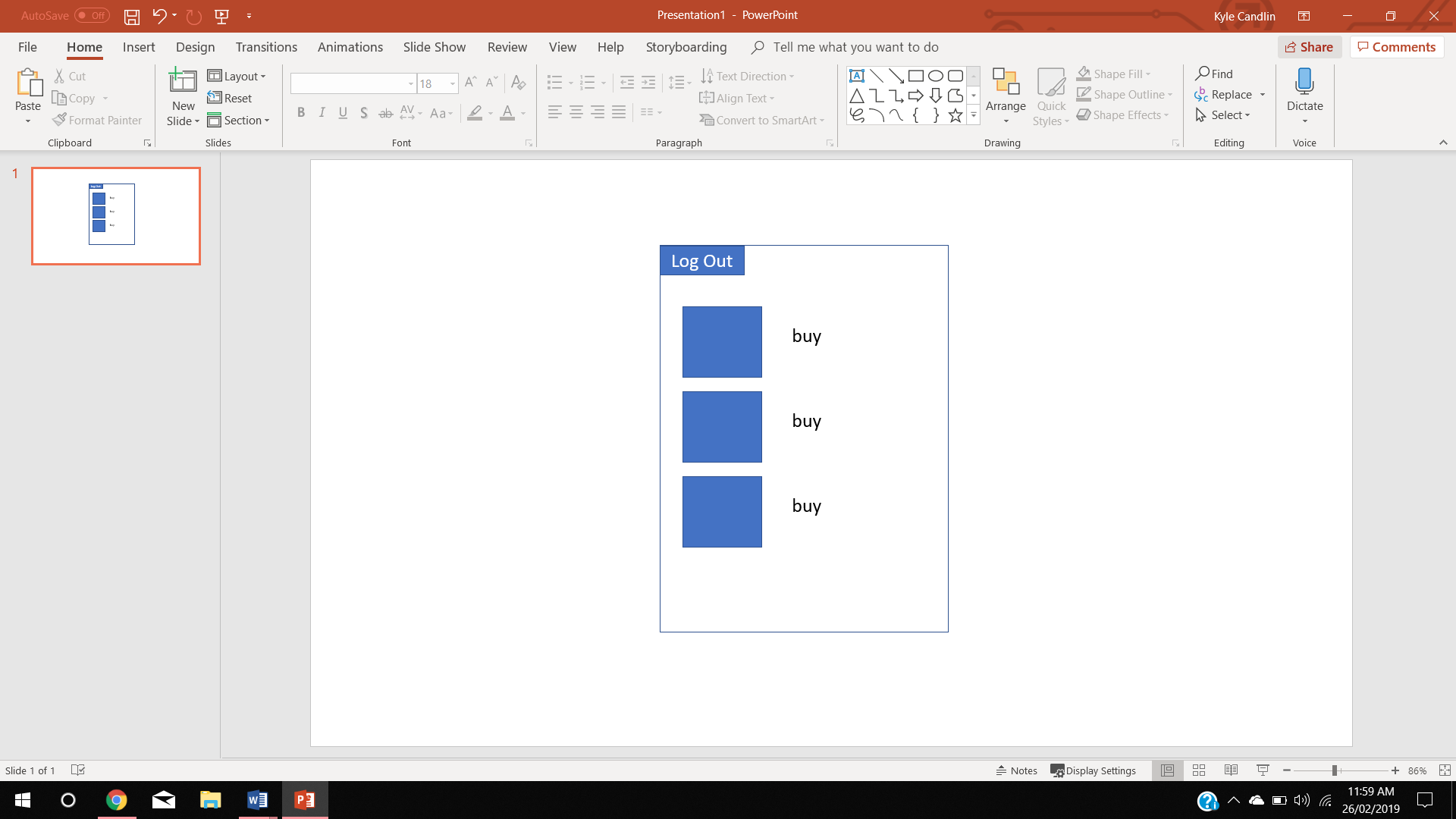
##### Minigame



This is the first design for the minigame page. This is very simple but still provides the user with another activity other than playing the game.

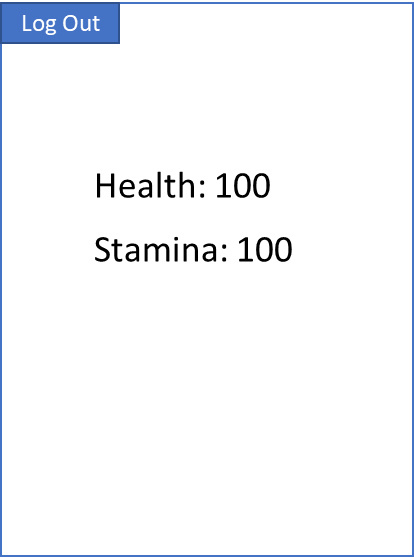
The log out will log the user out of their account and return them to the log in screen. The large button in the centre of the screen will be the button that is clicked which allows the user to gain currency to purchase logic gates for in game use. The shop button at the bottom of the page will be the button that takes the user to the shop and allows them to purchase gates for a set price.

##### Store



This is the first design for the shop page on the mobile application. The user must accumulate points using the minigame and can then purchase logic gates on this page. Each logic gate will have a specific cost and that price will be deducted from the current point score once they purchase the gate.

##### Player Statistics

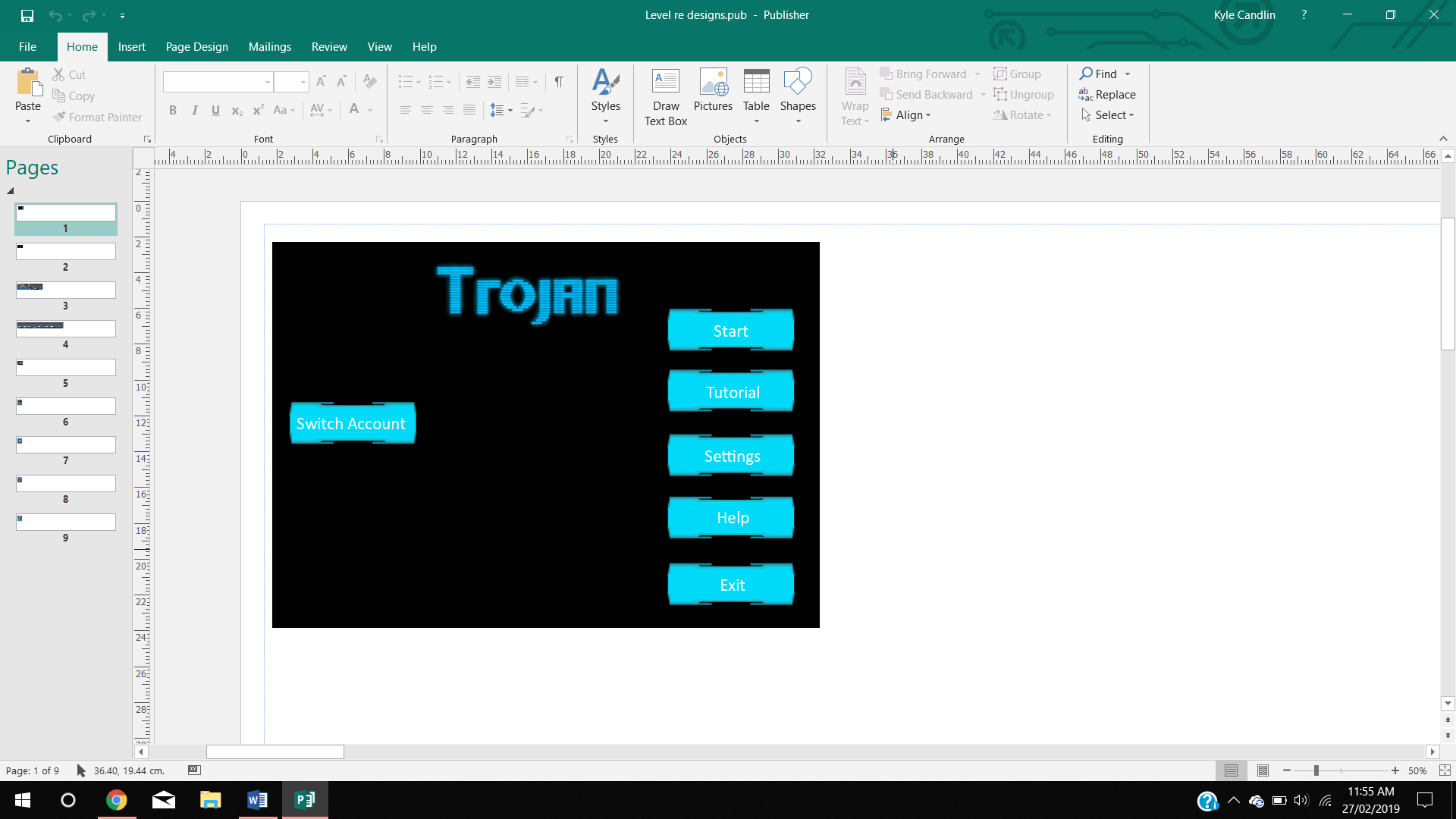


This is the first design of the player statistics page on the mobile application. It displays the health and stamina of the in-game player and will update in real time.

### Second Designs

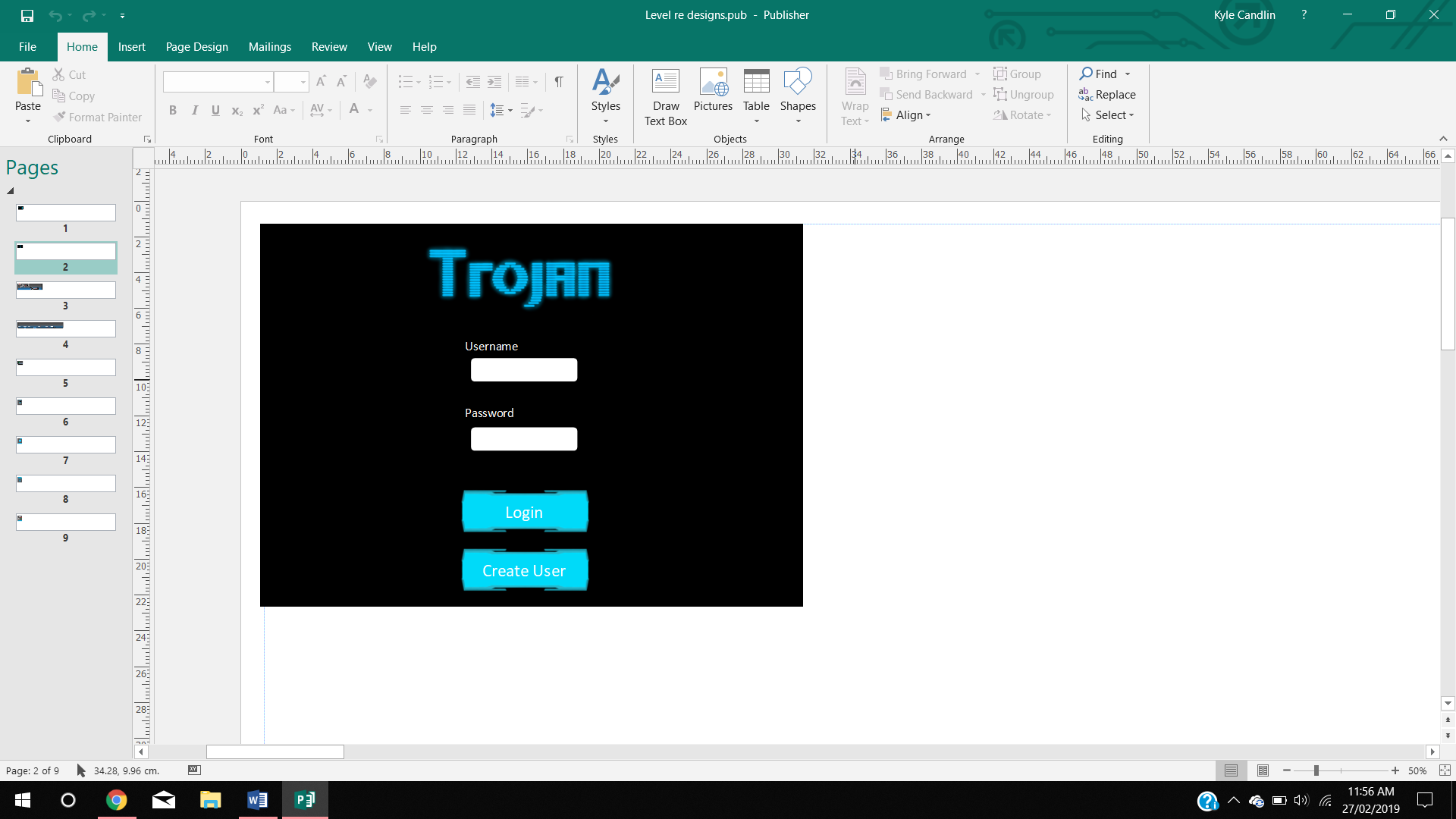
#### Game

##### Main Menu



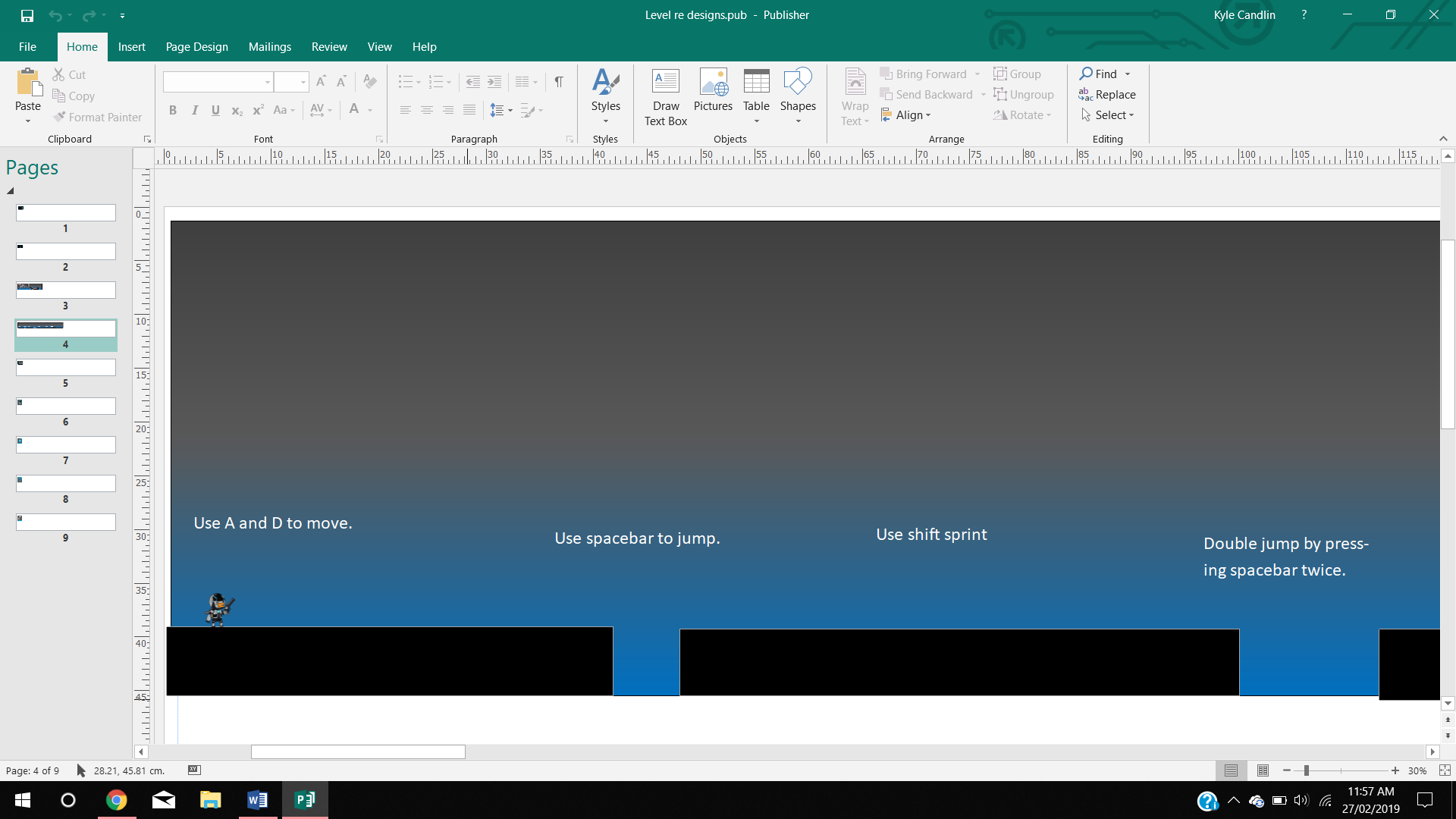
This is the second design for the games Main Menu. The design is now more complex, with a set colour scheme that matched the games theme. Each button has a on mouse hover effect to show the user which menu item they have selected.

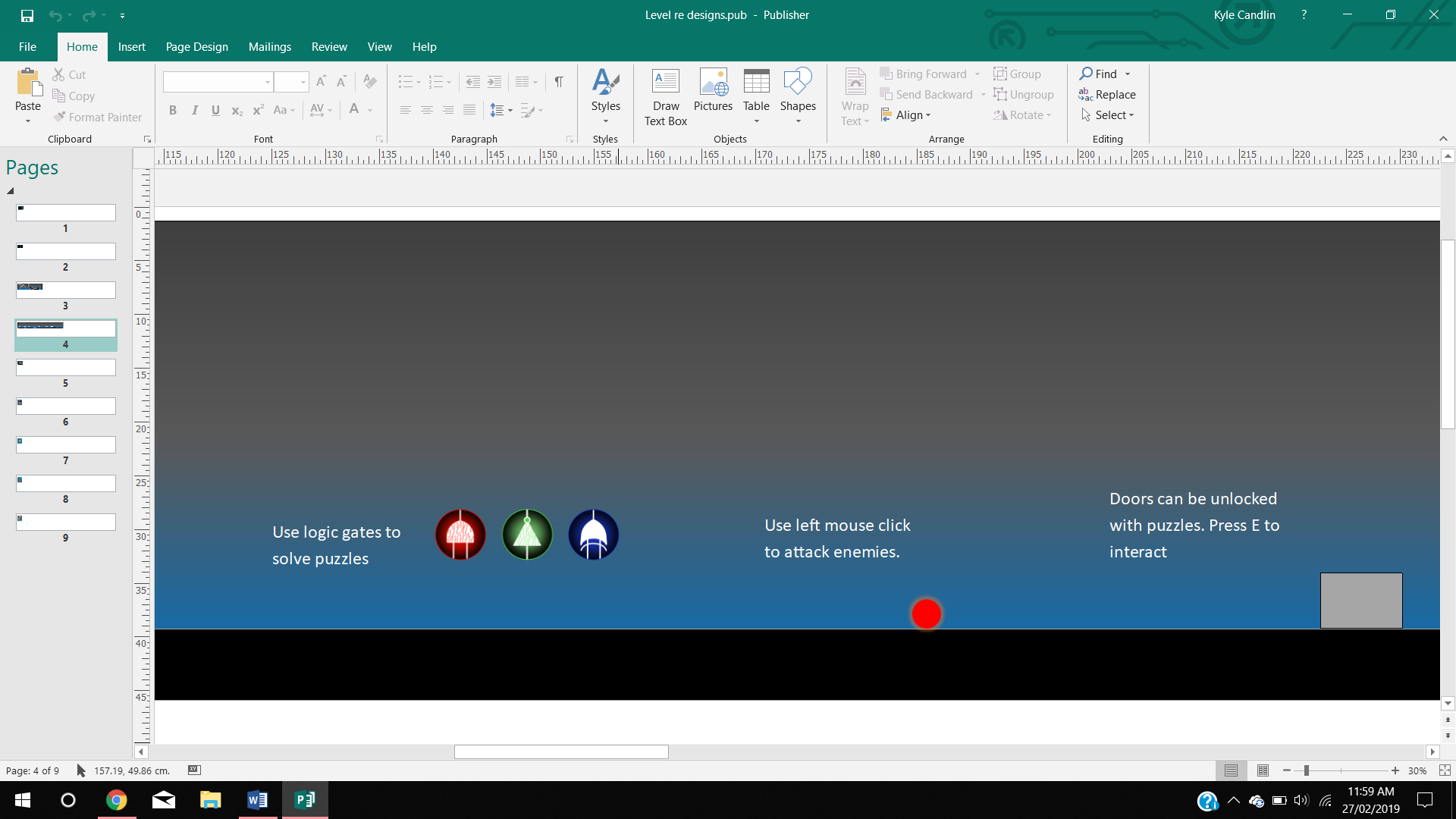
##### Login



The is the login design for the game. It features two text boxes for users to enter a username and password. The colour scheme is continued on from the main menu.

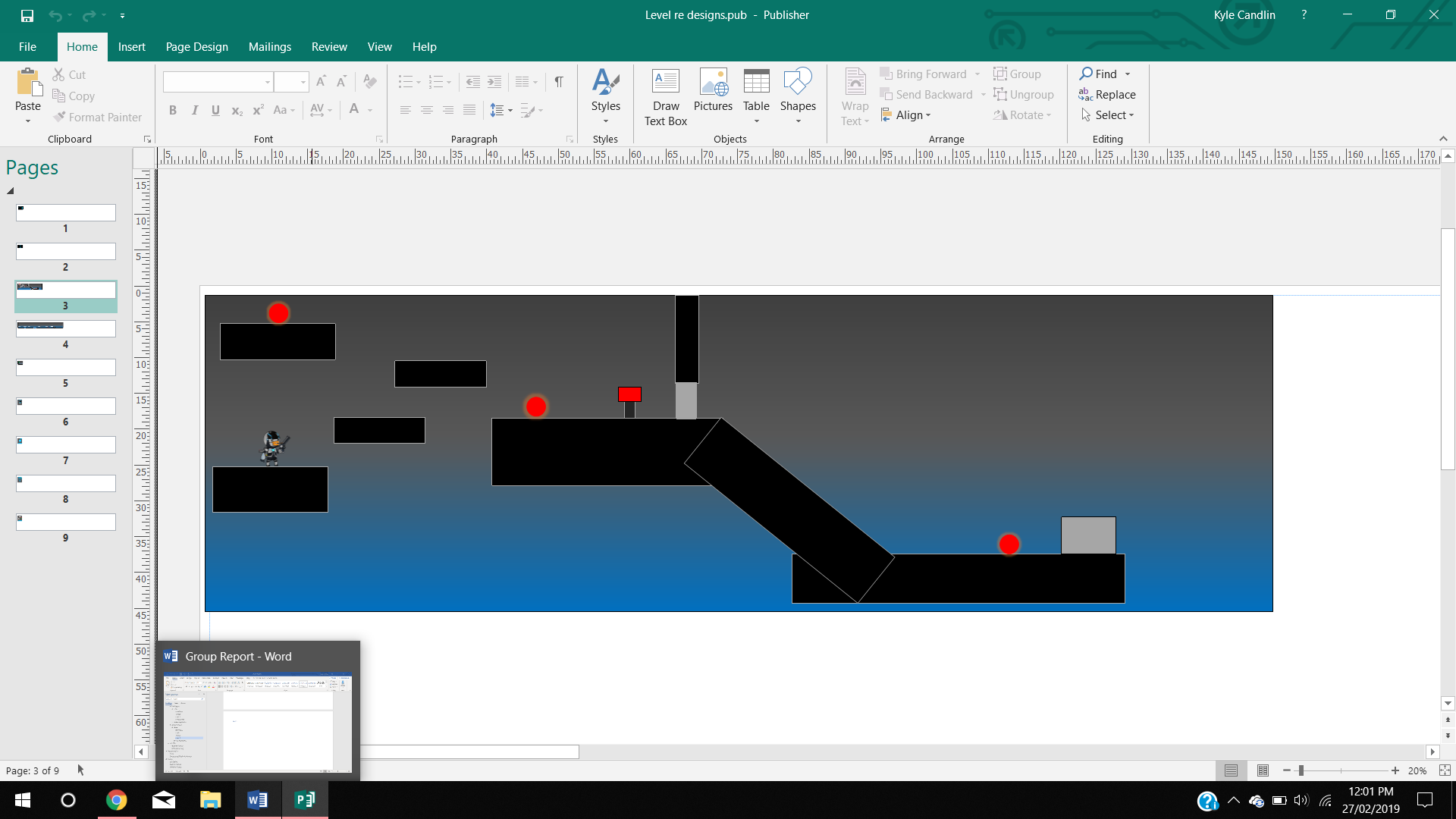
##### Tutorial





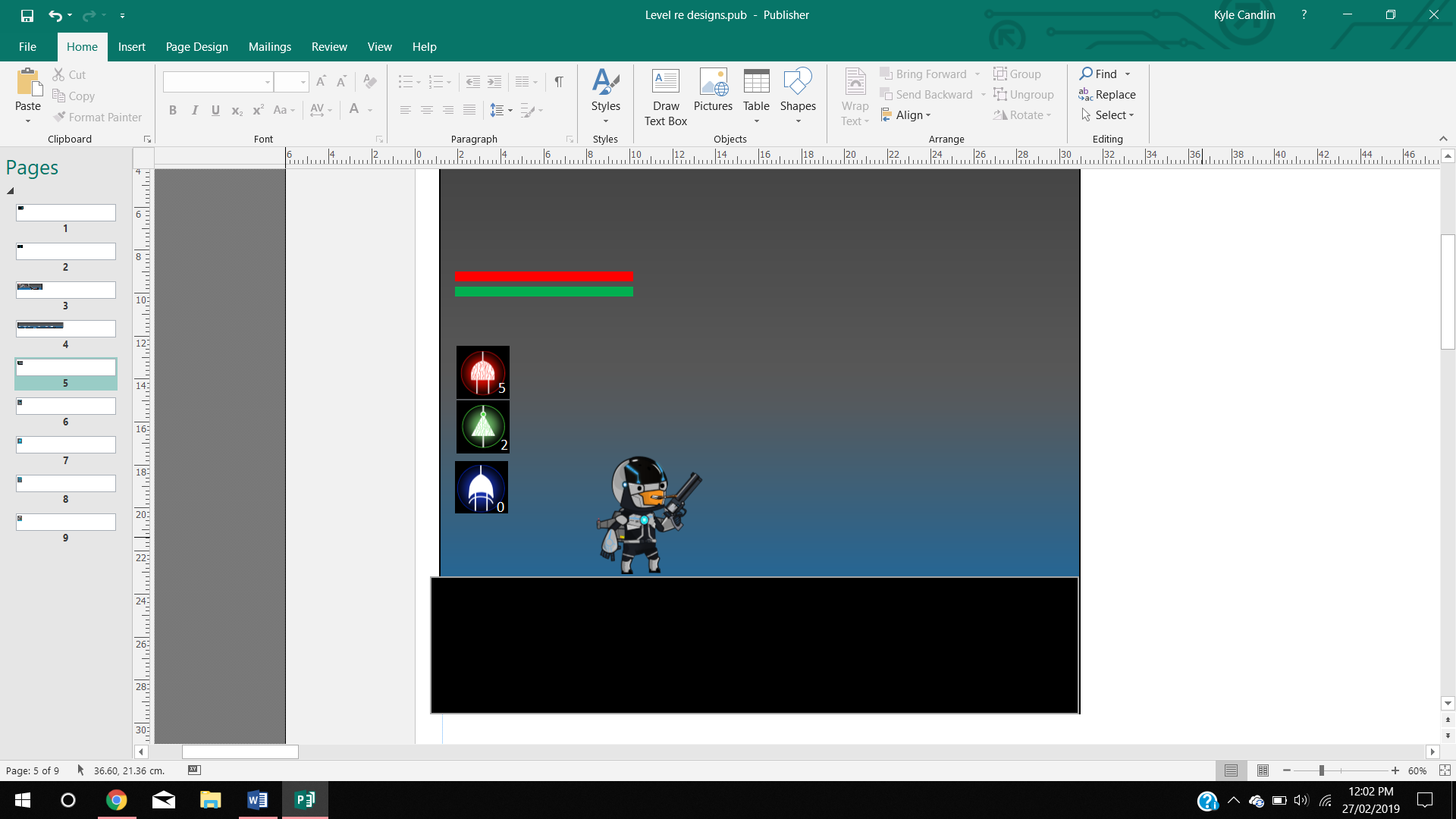
These are the designs for the tutorial level of the game. Like the first design, it includes a simple level system to teach the player about each aspect of the game. However, new textures have been added to make the level more aesthetically pleasing and to match the games theme (MoonStar2d).

##### Level 1



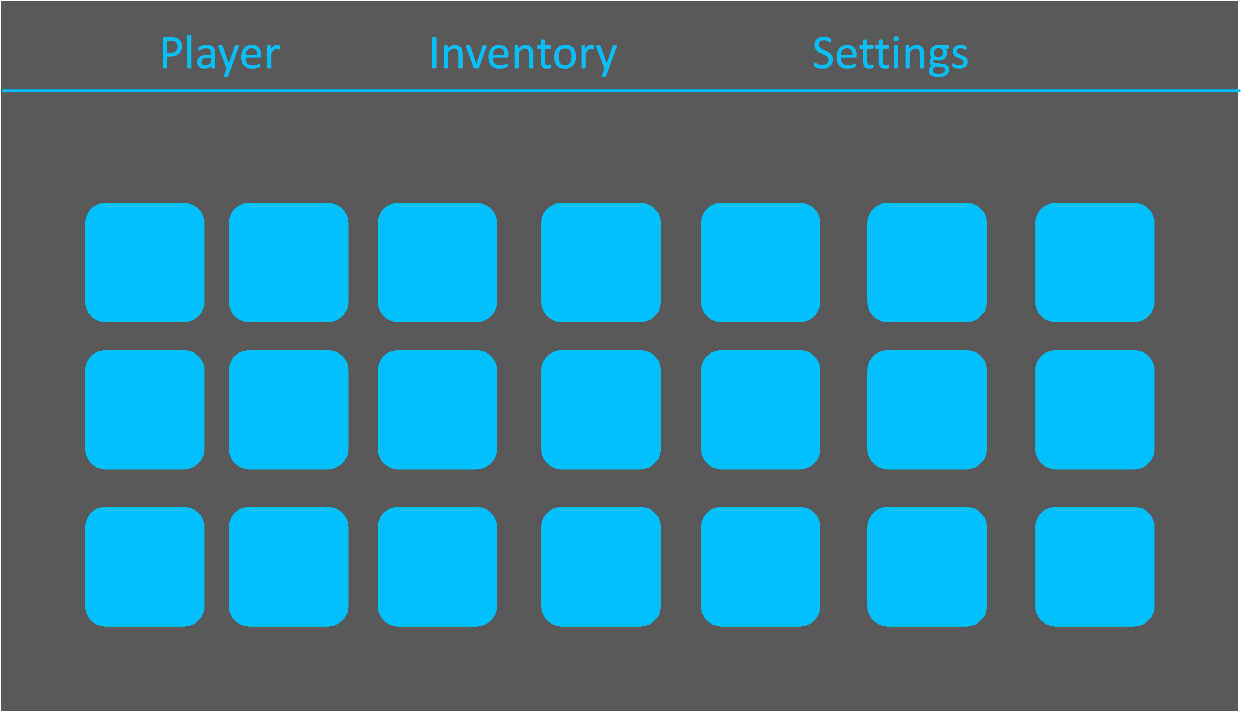
This is the level 1 design of the game. All functional aspects of the level are like the first designs. However, textures have been updated and a puzzle and weapon system have been implanted. Also, the level design has become more complex (MoonStar2d).

##### Camera View

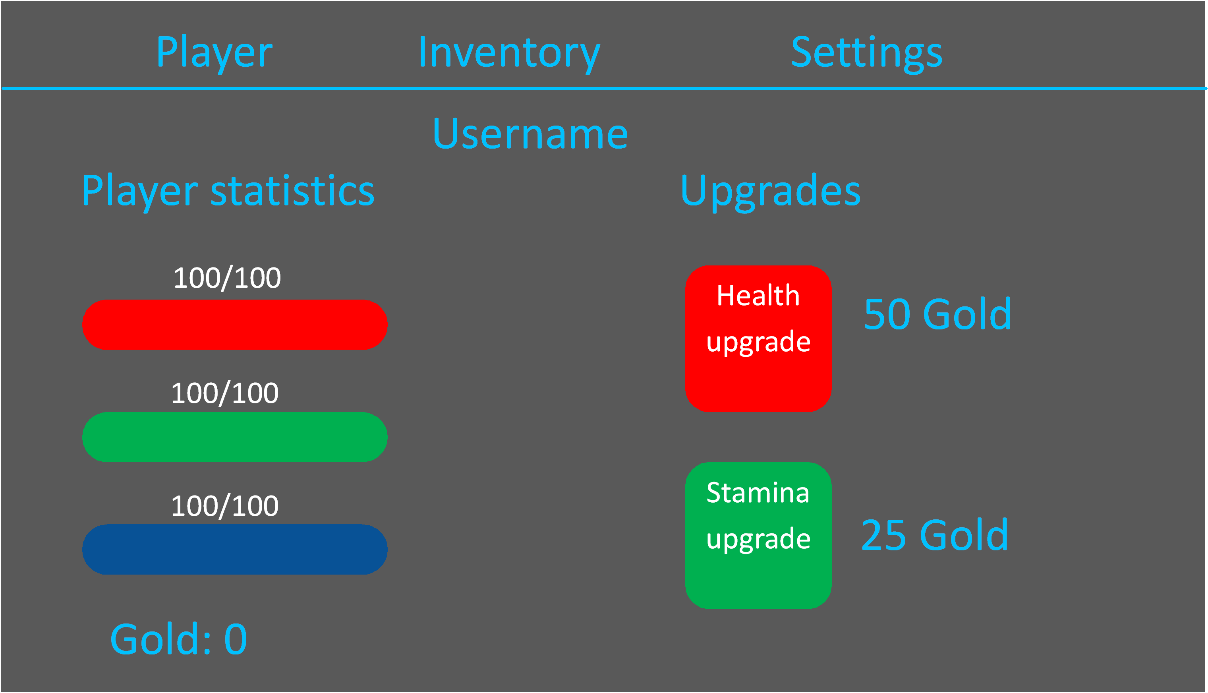


This is the camera view design for the game. The gates avaliable to the user have been limited to three gates (NOT, AND and XOR). Textures have also been updated to match the games theme.

##### Inventory Sytem



This is the inventory system for the game. It is functionally the same as the first design except in this one the textures have been updated to fit within the theme for the game.

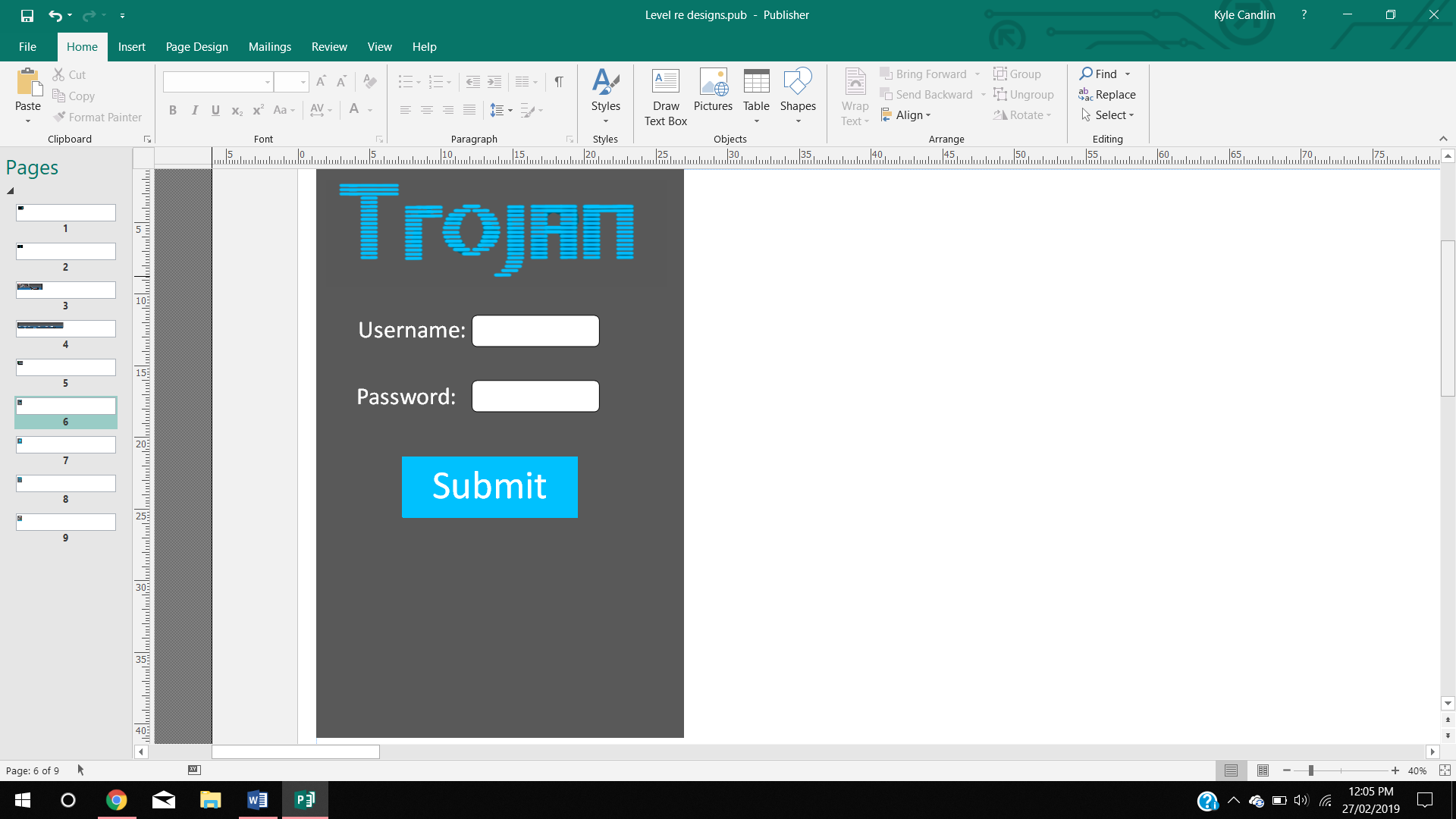


This is the player statistics screen. This is functionally the same as the first design except that the textures have updated to fit the theme of the game. The amount of gold required to purchase the upgrades have also been listed.

#### Mobile Application

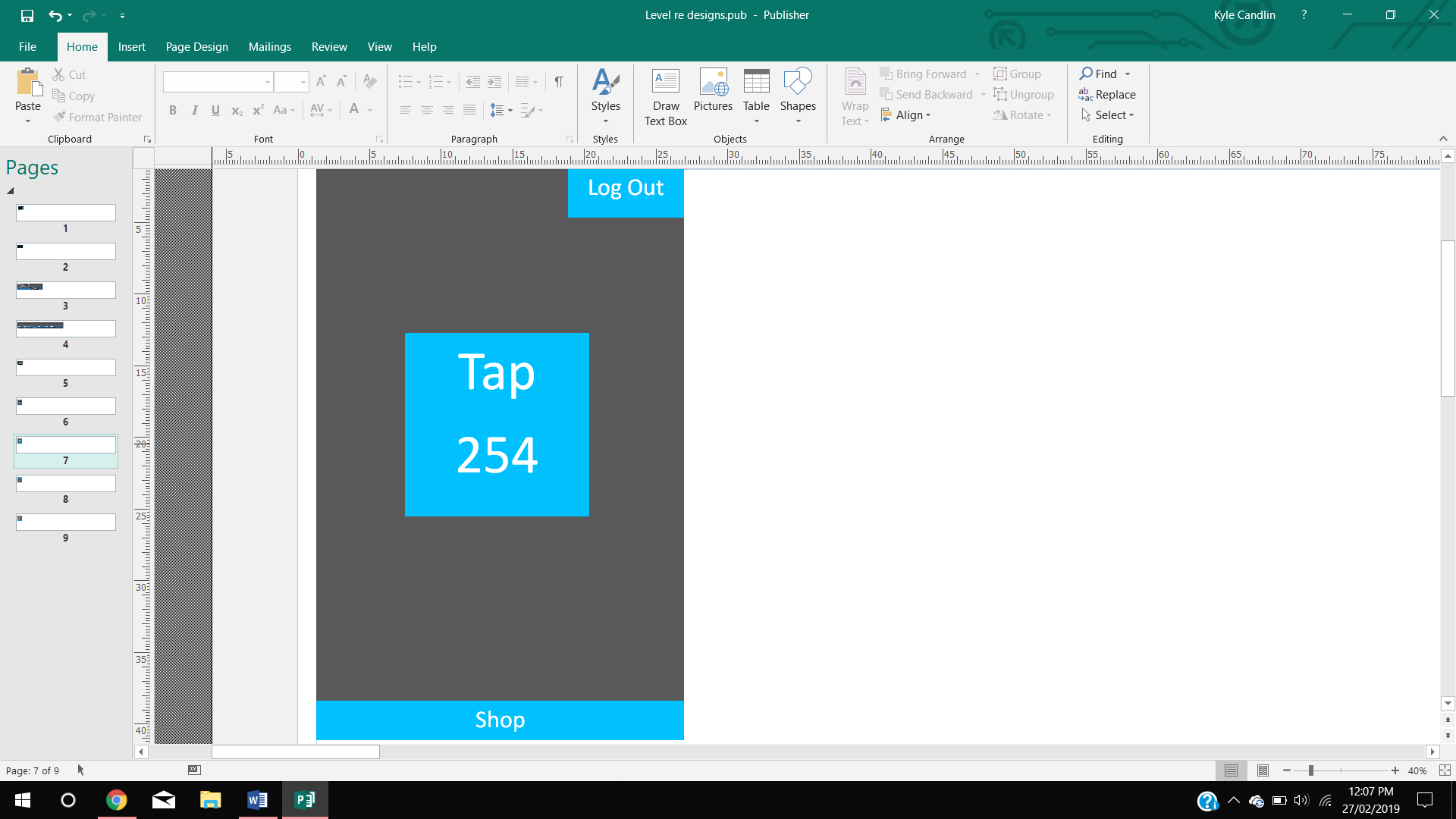
These are the second designs for the mobile application. The aesthetics have been redone to match the theme of the game and make the User Interface easier to navigate.

##### Login



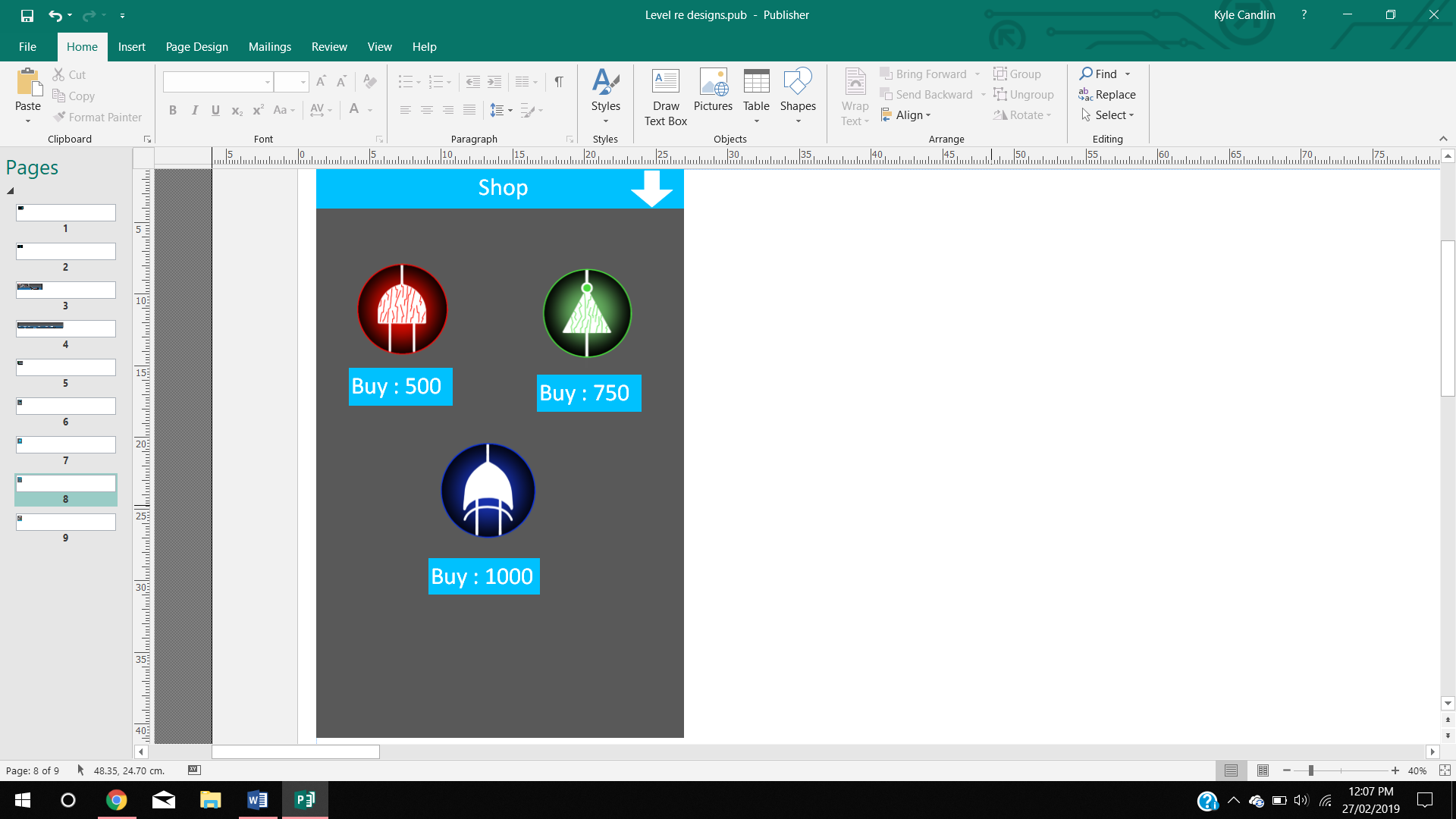
This is the login screen the user will see when they open the app, if they aren’t already logged in.

##### Minigame



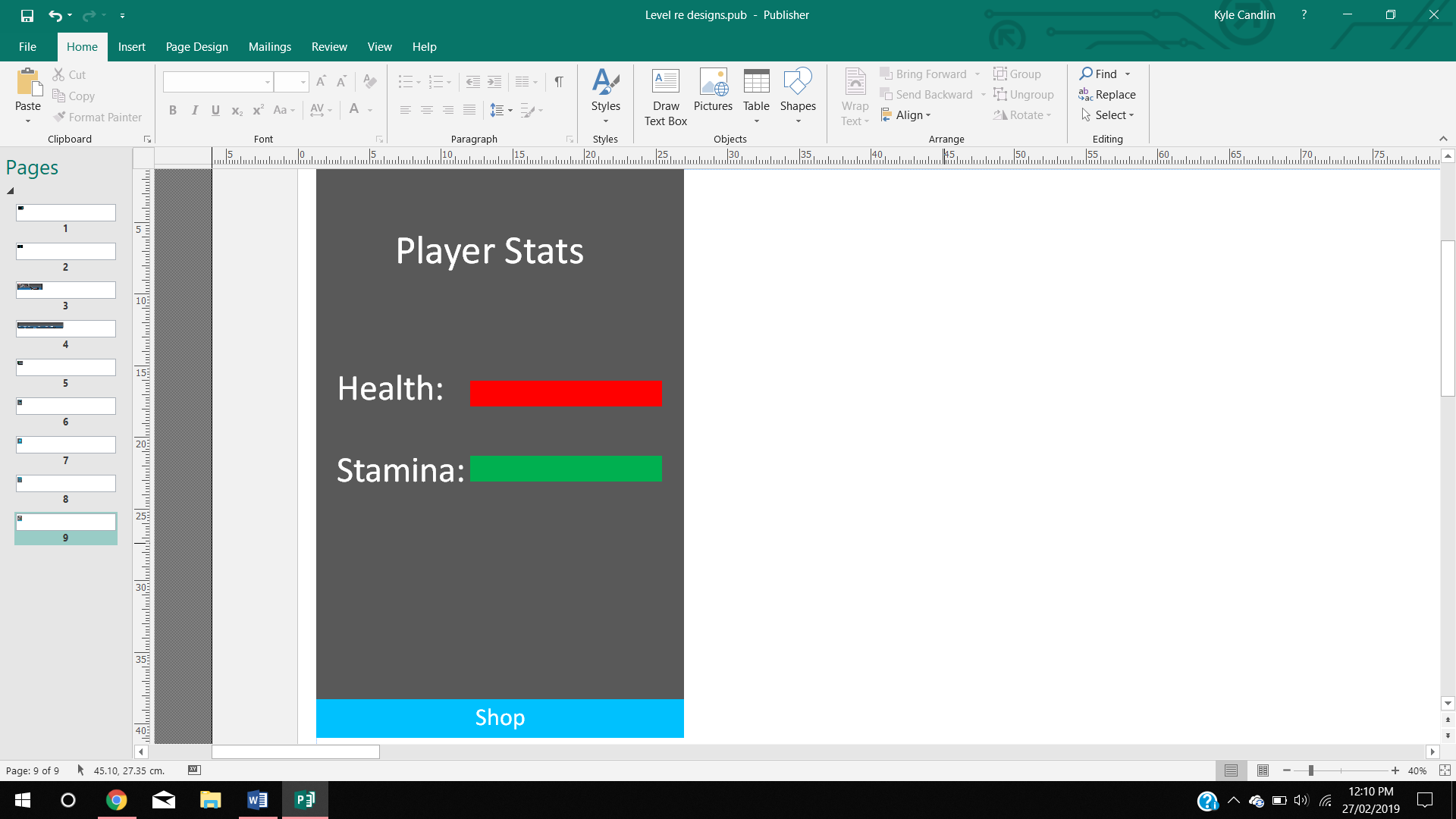
This is the page the user will see when they are playing the minigame. Clicking the square button in the middle of the screen will increment their score.

##### Shop



This is the page that the user will see when they open the in-game shop. Each button will add a logic gate to their in-game inventory when bought.

##### Player Stats



This is the page where users can see what their in-game player statistics are. It shows the health and stamina of the player.

# Test Plan

## Testing Tools

Many different approaches were used over the different sections of the project to check aspects of the code. The most common method was logging the status of a function to the console or screen. This was applied to the game, app and database. The more specific platform tools used were unity editor and Chrome mobile device tester. Unity editor allows variables to be seen on screen and many in game functions to be tested on screen. As an example, debug.drawline will recreate a ray cast’s (straight line from one point to another) path and show the path on screen so the direction and length can be tested. Chrome’s testing tool allows debugging and testing on mobile devices meaning mobile platform specific errors can be identified.

## Black Box Testing

We are using black box testing to help us identify the outputs are the correct result according to the inputs.

### Game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
| 1 | Does ‘Play Game’ button take user to game level? | User goes to game level. | User goes to game level. | None. |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

### Mobile Application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
| 1 | Does currency increment when button is clicked on minigame. | Currency increments. | Currency increments. | None. |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

### Database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
| 1 | Are user login details stored? | Data stored and can be accessed by the game and mobile application. | Game and mobile application can access stored login details. | None. |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

## White Box Testing

White box testing allows us to check each individual function work correctly according to the logic of each function. By going through a function step by step, we are able to see if results are calculated correctly.

### Game

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Location (Class) | Purpose | Expected Result | Actual Result | Actions Taken |
| DisplayStats() | Player\_Stats | Display the players health and stamina on screen. | Player stats displayed on screen. | Player stats displayed on screen. | None. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### Mobile Application

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Function Name | Location (Script) | Purpose | Expected Result | Actual Result | Actions Taken |
| Add(amount, type) | Login.js | Ajax call that pushes data to the user currently logged in. | Pushes the correct data to the correct user. | Pushes the correct data to the correct user. | None. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### Database

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Name | Location (Script) | Purpose | Expected Result | Actual Result | Actions Taken |
| Password Hashing | checkUser.php | Hashes the password for security. | Password is hashed. | Password is hashed. | None. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
| UI | | | | |
| B.1 | Does ‘Play Game’ button take user to game level? | User goes to game level. | User goes to game level. | None. |
| B.2 | Does ‘Tutorial’ button take sure to tutorial level | User goes to tutorial level | No action occurred | OnClick event not set. MainMenu.ChangeScene tutorial added in editor |
| B.3 | Does ‘Options’ button take sure to options menu | User goes to options menu | Loads the options menu over main menu | Main menu canvas needed to be set to active = false |
| B.4 | Does ‘Codex’ button take sure to codex menu | User goes to codex menu | User goes to codex menu | N/A |
| B.5 | Does ‘Quit’ button quit game within deployed version | Game quits | Game quits | N/A |
| General Gameplay: | | | | |
| B.7 | Does play fall through floor | Player runs on floor | Player falls through floor | Rigid body change to rigid body 2d |
| B.8 | Press of A and D keys | Player moves left and right | Player moves left and right – speed varies on frames | Movement multiplied by time to make motion frame independent |
| B.9 | Press Space | Player Jump | Player Jump | N/A |
| B.10 | Press Space Twice | Player should double jump | Player infinitely jumped | Modified angle of recasts to point at floor |
| B.11 | Press X to decrease players stats (testing purposes) | Player should lose stats | Player does not lose stats | Proper foreign script link corrected |
| B.12 | Alter Stats | Stat bars update to new values | Stat bars update to new values | N/A |
| B.13 | Pickup logic gate from floor | Logic Gate picked up and object should disappear | Logic gate picked up and disappeared | N/A |
| B.14 | Does LogicGatePickup update number of gates | Logic Gate increments by one | Logic Gate increments by one | N/A |
| B.15 | Does enemy wander left and right when player out of attack radius | Enemy moves left and right | Enemy moves left and right | N/A |
| B.16 | While enemy is in wonder state. Should not fall of platform edges | Does not fall off platform edges while in wonder state | Enemy does not fall of platform | N/A |
| B.17 | Player walks towards enemy | Enemy follows player once inside attack radius | Enemy follows player once inside attack radius | N/A |
| B.18 | When player comes into contact with enemy | Player loses health while touching enemy | Health does not decrease | Player stat script not defined correctly – define player stat script |
| Puzzle System: | | | | |
| B.19 | Player walk into puzzle interaction zone | Interaction prompt Is displayed | Prompt is displayed | N/A |
| B.20 | Player presses interaction button within interaction zone | Puzzle window appears in screen space | Puzzle window fails to appear | Visibility toggle incorrect – code change to .setActive which sets an object as currently active |
| B.21 | Insert logic gate into puzzle slot | Chosen logic gate Image should appear in slot selected | Chosen logic gate appeared in slot selected | N/A |
| B.22 | Insert incorrect logic gate into puzzle | No puzzle line changes colour and puzzle remain uncompleted | No puzzle line changed and remains uncompleted | N/A |
| B.23 | Insert correct logic gate into puzzle | Line to should change and puzzle should complete | Puzzle complete but lines do not change colour | Logic added for specific logic gates to calculate when output lines need to change colour from red to green |
| B.24 | Complete normal puzzle | Door moves out of players path | Door moves out of players path | N/A |
| B.25 | Final puzzle within level complete | Return to main menu | Does not return to main menu | Added scene management into Unity script |

White Box Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
| W.1 - Enemy Idle Function: | | | | |
| Description: This function controls the enemies whilst not attacking the player | | | | |
| 1.1 | Transform.Translate | Enemy should move | Enemy moves | N/A |
| 1.2 | Raycast towards direction of floor – right and left side | Enemy should detect when floor not present | Enemy detects when floor not present – outputted by Boolean in editor | N/A |
| 1.3 | Raycast horizontal – left and right | Enemy detects walls (floor layer) | Enemy does not detect wall | Raycast length shorter than radius of enemy box collider therefore enemy can’t detect wall. |
| 1.4 | Enemy moves towards edge of floor | Enemy should change direction to avert falling off edge | Enemy changed direction | N/A |
| 1.5 | Enemy moves near wall | Enemy should change direction when interacting with wall | Enemy does change direction | N/A |
| W.2 - Alter Stats (Player) | | | | |
| Description – Alter player health and stamina statistics | | | | |
| 2.1 | Player health should not go below 0 | Player health is set to 0 if current health + change in health results in a value less than 0 | Player health is set to 0 | N/A |
| 2.2 | Player health should not go above maximum health | Player health is set to max health if current health + change in health is > than max health | Player heath is set to max health | N/A |
| 2.3 | Change player stats | Player stats should change if health + change in health > 0 and < max health | Heath changes within range | N/A |
| W:3 - Player Attack | | | | |
| Description: Particle system that interacts with enemies to deal damage | | | | |
| 3.1 | Attack sound | Attack sound should play on attack button pressed | Attack sound plays constantly | Changed sound code from play and stop to pause and unpause |
| 3.2 | Test particle system emission | Particle are emitted on attack press | Particles are emitted | N/A |
| 3.3 | Test particle relative space | Particle system reacts to world space | Particle system reacts to local player space | Changed relative space in editor to world space [particle system will now react to player motion] |
| 3.4 | Particle system interaction with enemy | Particles should rebound off of enemy | Particle system rebounds | N/A |
| 3.5 | Particle system interaction statistics | Particles should damage enemy whilst in contact | Particles collide but no damage taken | Damage script set to specific enemy – Changed to receive script from the enemy the particle system is damaging |
| W.4 - Puzzle | | | | |
| Description: Game object puzzle element for players to solve | | | | |
| 4.1 | Player press logic gate UI element | Each logic gate name is printed on press as a test | Each logic gate name is printed | N/A |
| 4.2 | Click on selected slot with logic gate selected | Selected gate image appears in slot | No image appears in slot | Image slot set to renderer – changed to sprite renderer |
| 4.3 | Click on slot | Image appears only within selected slot | Same image appears in every slot | Logic slots added to an array and referenced individually |
| 4.4 | Input correct logic gate for slot | Red line should change to green | Red line changes to green | N/A |
| 4.5 | Input correct logic gate sequence | Puzzle should complete | Puzzle completed | N/A |
| 4.6 | Audio players on puzzle complete | Audio should play when puzzle is complete | No audio playback | Audio.play changed to Audio.playOneShot |
| W.5 – Scene Logic | | | | |
| Description: Background processes of each level | | | | |
| 5.1 | Death zone moves player | When player enters death zone, coordinated are changed | Coordinated are changed | N/A |
| 5.2 | Respawn coordinated | Player resets to respawn coordinates | Player resets to respawn coordinates | N/A |
| 5.3 | Enemy spawn points | Enemies respawn in selected positions | Enemies respawn in selected positions | N/A |
| 5.4 | Tile map box collided | Box collided functions efficiently | Mild latency due to box collided | Added composite box collider |

## Database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
|  | | | | |
| B.1 | Create User | New user added to the database. | New user is added to the database. | N/A |
| B.2 | Login | User connects to the database using their details. | Receive error access control denied. | Login.php file access control origin tells browser that page is accessible by application. |
| B.3 | Handle Player Statistics | Player statistics updated on database. | Statistics were updated. | N/A |
| B.4 | Handle Logic Gate Statistics | Logic gate numbers update to same numbers as game. | Logic gate numbers are different and don’t update. | Incorrect username post variable – changed to same variable name as database file. |
| B.5 | Send SQL injection from game to database – usernametextfield = drop table player data. | Print to user error message and deny code snippet. | Table dropped. | Use prepared statements and validation techniques to prevent against future attacks. |

## Mobile Application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Expected Result | Actual Result | Actions Taken |
|  | | | | |
| B.1 | Login – User inserts username and password | If details correct main game screen loads, if fail error message is displayed. | When details were correct main game screen loaded, when failed error message was displayed. | N/A |
| B.2 | Tap box - Press button | Currency increases by one. | Currency increases. | N/A |
| B.3 | Logout | User taken to login screen and local storage data removed. | User taken to login screen but local storage still present. | Add localstorage.removeitem on button press. |
| B.4 | Buy shop item | On chosen shop gate pressed, money decrease by gate price and gate number inserted into database. | Gate number inserted into database, but money doesn’t decrease. | Update HTML element. |
| B.5 | Display game statistics | User Statistics should be displayed. | Only health is displayed. | Statistics added to an array, rounded to integers and displayed. |
| B.6 | Page switching | Switch to page on click. | Page switches. | N/A |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| W.1 – Local Storage | | | | |
| Description: Unique storage on mobile device | | | | |
| 1.1 | Initiate local storage | Console log – local storage initialised | No log message | Local storage initialisation moved to document.ready function. |
| 1.2 | Get previous click data from local storage and parse into variable. | Storage value should parse in variable and be displayed | Variable is displayed | N/A |
| 1.3 | Get username from local storage. | If username found in local storage display game page otherwise display login page | Correct page is displayed | N/A |
| 1.4 | Define local Storage values | On events: click of tab box, login of user local, storage should be updated | Local storage not updated | Fixed local storage key errors |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| W.2 – Real Time Database Access | | | | |
| Description: Ajax scripts for accessing the database | | | | |
| 2.1 | Ajax initiation | On login button ajax function called | Function is called | N/A |
| 2.2 | Fetch text from form | Text should be assigned to variables | Text found but whitespace is present | Added .trim onto variables to remove whitespace |
| 2.3 | Ajax call | Ajax sends username and password data to defined php file | Username and password sent | N/A |
| 2.4 | Ajax response | On success login message and game page displayed and stored locally. On fail error message displayed. | Success and fail scenarios executed | N/A |

# Reference List

Alex (2013) *Space Station*. [Online] Available from: https://soundcloud.com/alxdmusic/alxd-space-station [Accessed 20 February 2019].

*Boolean Logic* (2019). [Online] Available from: https://www.bbc.com/bitesize/guides/zqp9kqt/revision/1 [Accessed 20 February 2019].

Cameron, A. (2011) *ping 2*. [Online] Available from: https://freesound.org/people/cameronmusic/sounds/138419/ [Accessed 20 February 2019].

*cuboos\_alien\_flame* (2007). [Online] Available from: https://freesound.org/people/Syna-Max/sounds/43583/ [Accessed 20 February 2019].

Disagree (2019) *Running on ground*. [Online] Available from: https://freesound.org/people/Disagree/sounds/433725/ [Accessed 20 February 2019].

Engel, K. (2018) *Fryeri*. [Online] Available from: http://freemusicarchive.org/music/Kai\_Engel/Brume/Kai\_Engel\_-\_Brume\_-\_05\_Fryeri [Accessed 20 February 2019].

Grunz (2010) *success\_low*. [Online] Available from: https://freesound.org/people/grunz/sounds/109663/ [Accessed 20 February 2019].

Hickler, E. (2007) *archi\_sonar\_01*. [Online] Available from: https://freesound.org/people/Argitoth/sounds/38700/ [Accessed 20 February 2019].

MidFag (2017) *alien\_engine*. [Online] Available from: https://freesound.org/people/MidFag/sounds/381564/ [Accessed 20 February 2019].

MoonStar2d *Cyberpunk Platformer Tileset*. image. [Online] Available from: https://moonstar2d.itch.io/cyberpunk-platformer-tileset [Accessed 20 February 2019].