DELIVERABLE 1

Project 7: Mobile sniper game

For this project, you will design and implement a mobile sniper game which can run on any mobile platform. Note that the game should work on mobile devices, so you will need to design the user interface accordingly. The game should have at least 3 levels where the player needs to shoot one or more targets in each level. The targets should be mobile (not just stationary) and headshots should give extra points for the player. The player should not be able to buy a certain augmentation for his rifle which can increase the zoom ratio (buying an advanced scope) or accuracy of the rifle (updating the muzzle of the rifle). The player should be able to select from 3 different rifles in the beginning of the game. Those 3 rifles will differ from each other by certain settings like accuracy and zoom. You may add more augmentation and rifle options. The game should have gunshot sounds for different rifles. You also need to add background music for each level. Before starting the new level, the user should be able to save their game but not in the middle of the level. The player should be able load the game they have saved.

TABLE OF CONTENTS

Concept of Operations

THE CURRENT SYSTEM - Alec York 2/9

The goal/idea of the project is to create a Hunting simulator that works on both IOS and Android devices. Our game will support a casual playstyle, whilst not taking up too much storage on the different, respective mobile devices. This essentially targets to increase accessibility to more players who aren't able to download other similar games, or simply don't like the playstyle of said other games.

PROPOSED SYSTEM: MOTIVATION - Alec York 2/9

The advantage this game will have over other games of similar themes is the download size; other hunting simulators tend to have significant download sizes that require not only a sizable download to play the game, but also additional downloads to play different levels. Therefore, our game will be more accessible, as storage is valuable, and not everyone has a lot of space to spare. In addition, our game will focus on a more casual-style gameplay. Other hunting games tend to require the player to grind, and play more in order to progress further into the game. Despite this, the players are limited to how many times they can play within a certain timespan. This puts players under a strict set of gameplay that can not only become repetitive, but also inconvenient (in terms of having only a select number of tries unless more are purchased or the user decides to wait). Our new game will be free from these restrictions, therefore giving the users more freedom to how they want to play, and when.

USERS & MODES OF OPERATION - Done by Chris Robertson 2/9

Non-Logged in User: They will be able to play through the levels once unlocked, but their progress will not be saved. Any changes made in the settings will be set to default when starting.

Logged in User: The user will be able to play through each of the levels and choose one of the three weapons for said level. They will also be able to change some basic control settings.

Admin: The admin will have full control over each level and what targets can spawn in each level. They will also be able to adjust point values for the targets on the level.

Single-player: There will be a single player mode for each level, which does not require the user to be online to play. The levels will also be repeatable after completion.

Free Version: The player will be able to play through all the starting levels in single-player mode as well as the starting rifles.

OPERATIONAL SCENARIOS - Done by Chris Robertson 2/9

In single player mode, the user/admin will play through each level and upon completing the stage, their progress up to that point will be saved. Single-player can be played without an internet connection, so the user will not be required to be online for the game to save. If the user quits during a stage, any progress made after the last save will not be kept. The user cannot save while in a level, so if they decide to quit they will have to start from the beginning

of that stage again. If the player decides to quit the game, they should be prompted to save in case they haven't to avoid losing progress.

If the game crashes unintentionally, then the user's most recent save should be loaded when reopening the game. The game won't be saved during a level regardless, so this progress won't be retrieved. If the user is not signed in at the time of the crash, then the game will start from the beginning of the first level with default settings.

<u>OPERATIONAL FEATURES</u> - Done by Chris Robertson 2/9

Must Have Features:

- **The game should be playable on mobile devices.**
- **There will be a single-player mode with at least 3 playable levels.**
- **The user will be able to save their progress and load from the last saved point.**
- **The user will have access to three different rifles and change parts on each rifle.**
- **There should be sound effects for each of the rifles available.**
- **There will be background music for all the levels.**

Would Like to Have Features:

- **We would like to implement multiplayer functionality that allows the user to hunt with another player.**
- **We would like to have a leaderboard that displays the player's highest score on a level and compares it to other signed in user's score for that level.**

<u>ANALYSIS</u> - Done by Chris Robertson 2/10

Our game is going to be developed using the Unity game engine. This game engine uses C# for the coding language and can be run with MacOS, Windows, and/or Linux systems. We aren't all too familiar with C#, but it will be easier to learn compared to C++ as it is very similar to the C coding language. The frontend, backend, and database will all be handled through Unity, and the game will be able to function on iOS and Android mobile devices that have the minimum requirements needed to play.

We are new to the Unity game engine, but there are some helpful tutorials available already which shorten the amount of time needed to get familiar with the software. While the basic personal license is free, there are some features that require you to have Unity pro to access them. That said, Unity allows students to temporarily use Unity pro for free. In order to edit/create a project, you need to have a desktop/pc with either macOS (High Sierra 10.13), Windows 7(SPI+), 10, and/or 11 to use Unity 2020. If you have the project file downloaded, then you do not need an internet connection in order to edit and develop the project. The game itself will not require an internet connection in order to play once it has been downloaded.

As mentioned before Unity uses C#, so in terms of performance it will not run as fast as a game engine like Unreal Engine which uses C++. It also gets marginally outperformed in terms of animation quality when compared to Unreal Engine. As new developers though, the learning

curve for both C++ and Unreal Engine is much higher than that of C# and Unity. That means that for starting out, using Unity will shorten the time needed for a project with a limited time frame without the sacrifice of visual and gameplay quality. Unreal Engine may be slightly better when it comes to hyper-realistic graphics, but not so much that it makes Unity a worse game engine overall. Unity also gives the full ownership of the projects made with the software to the creators, while Unreal Engine will have royalties on the game sales if the game makes over a certain amount of revenue. And for smaller games, the performance difference won't be as significant as compared to a large project. So when making a relatively small game in a limited time frame as less experienced game developers, Unity stood out as the best option for our project.

Project Management Plan

APPLICABLE STANDARDS - Chris Cao 2/10

- **Coding standard:** We will be using a similar style to the one used in Dr. Szumlanski's CS classes.
  - **Spacing:**
    - Indentations will be done with tabs.
    - Any new code block will be indented 1 level deeper.
    - Use of blank lines for organizational purposes will be limited to 1 or 2 only.
    - Binary operators will have spaces around them. (i.e. "(a + b + c)")
    - Spaces will be used before the opening parenthesis in loops. (i.e. "if (i = 0)")
    - No space before the opening parenthesis in methods. (i.e. "main(string args[])")
    - Blank lines (1 or 2 at a time) may be used for organization.
    - Curly braces ("{ }") will always be on a new line.
  - **Comments:**
    - We will mostly be using in-line ("//") instead of blocks. ("/**/")
    - Block comments may be used only when temporarily commenting out large sections of code that are not working as intended.
    - We will insert a space, followed by the writer's name and a colon at the start of each comment or block of comments. (i.e. "// Bob: Error here!")
    - Comments will be used to describe the purpose of the lines below them, or to make note of errors where applicable. (i.e. "// Ed: Weapon stats")
  - **Other:**
    - We will be using camelCase for variable/function names. (*https://en.wikipedia.org/wiki/Camel_case*)
    - We will be using PascalCase for class names. (*https://en.wikipedia.org/wiki/Naming_convention_(programming)#Pascal,_Modula-2_and_Oberon*)
    - Variables will have meaningful names that say what they're used for.
    - Lines will be limited to 100 characters long at most.
- **Artifact Size Metric Standard**
  - The project should be around 1 GB at most as this game is for mobile devices, so it shouldn't be too large.
  - We expect to see roughly 20K lines of code.
  - We expect to use 20 packages in order to implement all the functionalities for the game requirements.

DELIVERABLES - Alec York 2/10

Artifact	Due Dates
Individual Weekly Progress Reports	Due by Every Friday
Concept of Operations	Due by 2/10
Software Project Management Plan (SPMP)	Due by 2/10
Software Requirements Specification (SRS)	Due by 2/10
Test Plan	Due by 3/31
High-Level Design	Due by 3/31
Detailed Design	Due by 3/31
Test Plan	Due by 4/22
Test Results	Due by 4/22
Source, Executable, Build Instructions	Due by 4/22

SOFTWARE LIFE CYCLE PROCESS - Chris Cao 2/10

- We will be using a variation of the waterfall method with prototyping, where we collectively work on one part at a time until it's mostly finished, as it is relatively straightforward and easy to follow as a small group. However, we may decide to "go back" to previous stages to make tweaks and improvements if they come up later during development.

TOOLS & COMPUTING ENVIRONMENT - Chris Cao 2/10

- The project will be done on Unity with C# (Unity has a built-in C# compiler) and using Visual Studio (comes by default with Unity) as the IDE. The project will be made to run on iOS and Android operating systems.

CONFIGURATION MANAGEMENT - Alec York 2/10

Version & change control to be handled through GitHub

QUALITY ASSURANCE - Alec York 2/10

- **Different responsibilities will be assumed by various group members depending on what they want to work on. Obviously, communication is vital when working together on a project, and peer review will be a responsibility of all group members. It is therefore expected for each member to offer their own criticism/opinion (if necessary) on how to improve things, and to just guarantee everyone's on the same page and everything is accurate. Changes will be noted,**

and tests will be run to check whether there can be further improvements, or if the changes themselves were improvements.

RISK MANAGEMENT - Alec York 2/10

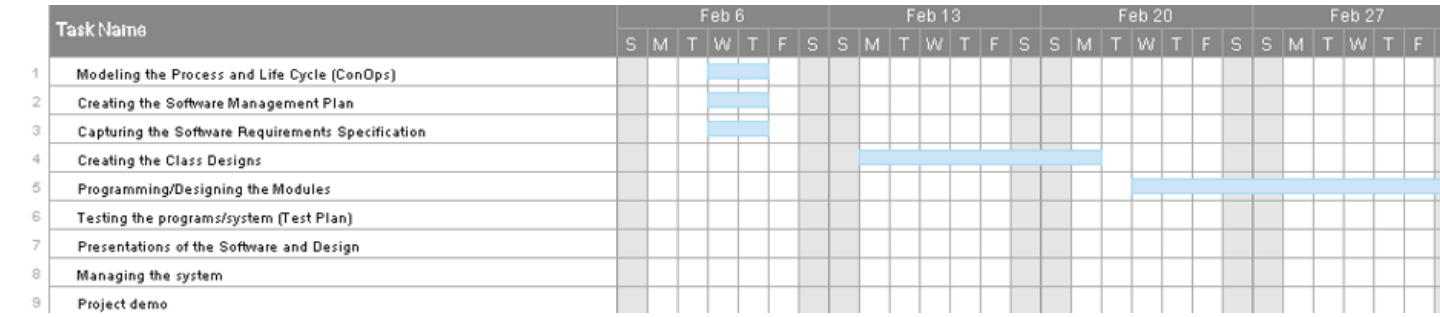
- **Generic risks:**
    1. **Misunderstandings - whether a teammate misunderstands a requirement, or an idea of a feature of the project, etc. it will be covered essentially through peer review and communication.**
    2. **Procrastination - procrastination will be handled through working together and communicating who and when someone might be doing one thing or if someone is busy at certain times and when they will be available, etc. Team members will come together to discuss when they can/will work, and depending on what needs to be done, what they will work on.**
- **Project Specific risks:**
    1. **IDE crashes - risks such as IDE crashes will be solved via debugging - the team member that has most recently worked on the project will be most adept at solving the issue, but if necessary, will have the support of the rest of the team.**
    2. **input file issue - issues like this can be solved through various different methods including getting the opinion of the problem from other team members, to completely dissecting what the issue can be stemming from.**
- **Business risks:**
    1. **Similar product - there are already products similar to ours that are already available, however, ours will be different in features, overall playstyles, and will be more accessible to users.**
    2. **Reviews/ratings are important and tests of the game and opinions of it will be taken not only by ourselves, but also by some of our other peers such as close friends/family to help improve upon our game.**

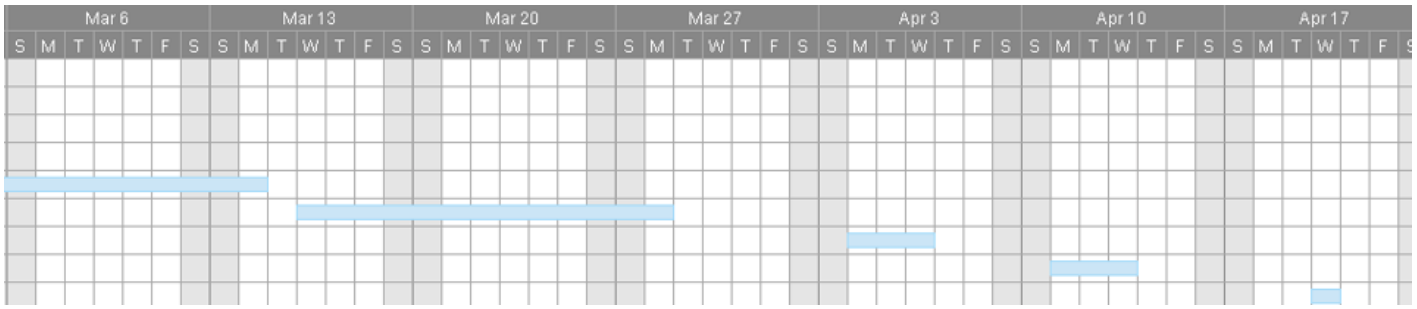
TABLE OF WORK PACKAGES, TIME ESTIMATES & ASSIGNMENTS - Alec York 2/10

Currently, we have roles designated to all members for all packages, as we are all responsible for reviewing the work of other team members, and all also want the experience in creating the project through all of the packages.

	Class Design	**Class Programming**	**Frontend Dev**	**Backend Dev**	**Testing**
Estimated Time for completion	6 days	14 days	36 days	36 days	9 days
Responsibilities	All members	All members	All members	All members	All members

GANNT CHART OR PERT CHART - Alec York 2/10





If it is too small to see, here's a link:
https://drive.google.com/file/d/1kP0YXanTrYoZM_stHxN4VZNtk6s95RJb/view?usp=sharing

TECHNICAL PROGRESS METRICS - Done by Chris Robertson

Throughout the project, we'll track how many of the must-have features have been successfully implemented as these are the main requirements for the project. If we add to the total number of requirements, we will adjust the progress count in order to make sure we stay on track for completion.

During the detailed coding, we will track the number of packages, and classes used to make code analysis, debugging easier as well as track the project size along the way. Keeping track of the number of classes and methods will also allow us to keep the code from becoming too complex.

As we develop the project, we will check the overall file size to track whether the project is becoming too large. For a mobile game, we don't want a single app to take up a majority of the memory on the device.

PLAN FOR TRACKING, CONTROL, & REPORTING OF PROGRESS - Done by Chris Robertson 2/10

We will post our progress report weekly through google drive: the weekly report should include time spent in each activity, completed task(s), tasks in-progress, tasks planned for the following week, and individual issues and problems.

Each week, we'll read and analyze the logs; examine the technical content of the work done to date; examine the technical progress metrics; consider the QA results; reassess the potential project risks; and take corrective action if necessary.

Software Requirements Specifications

INTRODUCTION - **DONE BY ISHMAEL(2/9)**

- This project is a mobile application simulator game. It is intended to simulate a hunting experience across multiple ecosystems with varying animals. You may refer to “Concepts of Operation” for details of game functions and design.
- Our standards for this game is ensuring an enjoyable user-friendly environment to be easily accessible for all ages. Simultaneously we will make security a priority to ensure data security and privacy.
- This product is COTS(commercial off-the-shelf) and is intended to be used as-is.
- Abbreviation/Acronyms/Technical word Definitions:
    - OS - Operating System | IDE - Integrated Development Environment | GB - Gigabyte
    - FR-01 - Functional Requirement No. | IR-02 - Interface Requirement No.
    - PER-03 - Physical Environment Requirement No. | UHFR - 04 - User & Human Factors Requirement No.
    - DOR-05 - Document Requirement No. | DAR-06 - Data Requirement No.
    - RR-07 - Resource Requirement No. | SR-08 - Security Requirement No.
    - QAR-09 - Quality Assurance Requirement No.

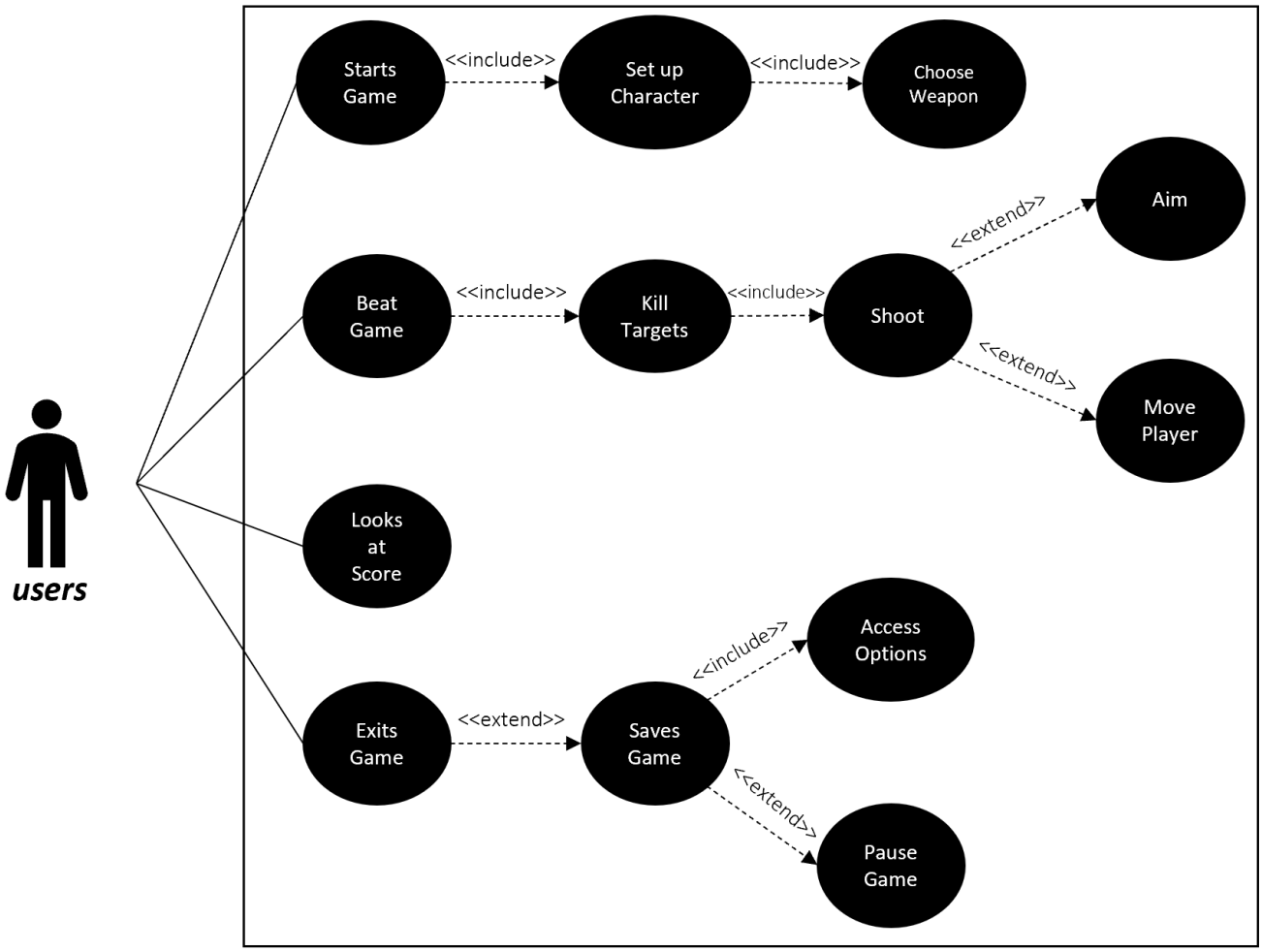
PRODUCT OVERVIEW

- Specification Assumption
    - With this product we expect certain minimal phone specifications. This includes:
        - At least 2GB Memory
        - At least 1GB Storage Space (for game/save data)
        - Adheres to App Store requirements (Android: KitKat 4.4 | IOS: ver 11.0)
        - Uses .NET Standard 2.0
        - Ability to download/install application file (.apk or .ipa)
        - Ability to load and render a 3-D Game with touchpad controls
- Stakeholders
    - Developers: Invested in quality/performance/reliability when creating products.
    - Platform Owners: Invested in safety, reliability and security of the product to be uploaded on their platform device/software
    - Users
        - TA: Invested in development/performance of product
        - Consumer: Invested in utilizing a quality product for enjoyment while maintaining their security

Event table

Event Name	External Stimuli	External Responses	Internal Data & State
Start Game (menu)	Tap “Start New Game” button	Loads game interface	Process request Run/load game files
Load Game (menu)	Tap “Load Prev. Game” button	Loads game interface	Process request Grabs previously saved state game data from memory Run/load game files Sends Error message if could not retrieve any save data
Exit Game (menu)	Tap “Exit Game” button	Exits Program	Process request Exits and closes all files and processes Frees up memory
Options (in game)	Tap “Options” button	Displays option menu with “Save” & “Quit” buttons.	Process request Display options menu to interface
Save Game (in game)	Tap “Save” button	Display “SAVED!” message	Process request Store current game state data to memory (display error if unable)
Choose Set Up (menu)	Tap “Set Up” button	Display character interface	Process request Grab/run character interface files to display
Move Player (in game)	Touch Screen Controls	Move Character	Process request Run inputSystem files
Aim Weapon (in game)	Touch Screen Controls	Display scope	Process request Run inputSystem files & scope files
Shoot Weapon (in game)	Touch Screen Controls	Weapon Effect displayed Kill target (if lands)	Process request Run inputSystem files & external object interaction files for hitting target Update Score files for gaining points/display points
Show High Score Table (in menu)	Tap “High Scores” button	Display high scores table	Process request Grabs saved scores from memory Orders from highest to lowest Display interface

Use Case Diagram



Use Case Diagram Description

- **Starts Game:** Begin game from starting screen once application launches
    - **Sets Up Character:** Before gameplay starts selects character gear
        - **Choose Weapon:** Must choose a weapon to start
- **Beat Game:** Complete Objective to complete gameplay
    - **Kill Targets:** Must kill all targets
        - **Shoot:** Must shoot targets to kill
            - **Aim:** May need to aim at target
            - **Move Player:** May need to move character to reach target
- **Looks at Score:** View HighScore table in starting page
- **Exits Game:** Leave application or game play to start page
    - **Saves Game:** May save game before exiting
        - **Access Options:** Must reach options at end of every level to save
        - **Pause Game:** May pause game before exiting

SPECIFIC REQUIREMENTS:

FUNCTIONAL REQUIREMENTS

No:	FR-01
Statement	The system shall: Run on any mobile platform | Have 3 levels | Have moving targets | Have Level System Have varying point system | Have various weapons | Have responsive sound Have background music | Have load/save states | Have High Scores table
Source	Client
Dependency	IR-02
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	After developmental phase of the application there will be a heavy testing phase. Will be tested on devices with IOS and Android. Will make sure the application itself loads all necessary files without bugs or glitches and functions as intended with multiple playtests and multiple users. Before release there will be beta testing for bugs in case any feature does not properly display within the device interface. Will make sure memory allocation is successful by checking device files so users saved games are safe.
Revision History	Ishmael – 2/10

INTERFACE REQUIREMENTS

No:	IR-02
Statement	The interface shall: Be responsive| Be Fluid | Display touchpad controls | Have high quality visuals Have minimal latency
Source	Client
Dependency	DAR-06
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	User input through the control interface will be the data input and the reaction of the objects within the game will be the output. Example would be tapping shoot button on device to shoot an object within the game. Grabbing location data of user character and target object will generate values to determine if object is hit and points (integer value) to be allotted. Data is sent and received only when user interacts with input controls. Accuracy must be exact and match with what is being displayed. For this reason, quality is important to differentiate objects clearly. For a fluid feel, the timing between user input and response must give the impression of a synchronous system. The time it takes for data to travel must be close to 0 as possible.
Revision History	Ishmael – 2/10

PHYSICAL ENVIRONMENT REQUIREMENTS

No:	PER-03
Statement	The Physical Environment shall: Be on mobile platforms| Run on most modern devices
Source	Client
Dependency	N/A
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	The environment for this product must be a mobile environment that can handle and install .APK or .IPA files. Mobile specification requirements will follow specifications outlined in Google Play Store & Apple App Store. Most modern devices will have enough memory (At least 2GB) and Storage (At least 1GB)
Revision History	Ishmael – 2/10

USER AND HUMAN FACTORS REQUIREMENTS

No:	UHFR-04
Statement	The User and Human Factors shall: Have easy usability of the system functions | Need basic instruction to explain game Not be able to disrupt/break system
Source	Team
Dependency	DOR-05
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	Users will be constricted to only playing the game and using the display interface. No access to integral files/data. This ensures simple use while maintaining integrity of the system. No external information needed to explain the system itself. Minor text displayed to explain game mechanics and rules.
Revision History	Ishmael – 2/10

DOCUMENTATION REQUIREMENTS

No:	DOR-05
Statement	The Documentation requirement shall: Provide text explanation of game
Source	Team
Dependency	N/A
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	Users will be prompted in the display just before game starts. The prompt will explain general rules of game mechanics. Will refrain from technical terms to be as simple and short as possible for the user. Rules will be updated to match game version. Prompts will be reviewed and tested for accurate readings. Will also ensure legibility by making sure the prompt covers the device as much as possible (utilizing entire screen).
Revision History	Ishmael – 2/10

DATA REQUIREMENTS

No:	DAR-06
Statement	The Data requirement shall: Calculate Points to be outputted | Retain previous game data if saved
Source	Client
Dependency	IR-02
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	This requirement will ensure accurate points are calculated when target is hit. More points when level increases. The formula used will be: - 1 Point per target hit and 2x points (2 Points) for head shots (LEVEL 1) - 3 Points per target hit and 1.33x points (4 Points) for head shots (LEVEL 2) - 5 Points per target hit and 1.6x points (8 Points) for head shots (LEVEL 3) Points will always be an integer to make it simple. Only the 3 highest scores will be stored for high score. Will always be stored until a higher score is reached, then lowest score will be removed. Will be evaluated by hand to ensure accurate calculations are made by system.
Revision History	Ishmael – 2/10

RESOURCE REQUIREMENTS

No:	RR-07
Statement	The Resource requirement shall: Describe actors to maintain/build system | Describe resources to work on project
Source	Team
Dependency	FR-01, PER-03
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	The developer team is required to build and maintain the project. Team must be knowledgeable in game development specifically using Unity engine and C# language for scripting. A computer that runs unity is all that is needed to run and build project. Software will run on most modern mobile devices through the installation of app file - Minimum Android KitKat 4.4 or iOS 11.0 - Uses .NET Standard 2.0 Minimum hardware specs: - 2GB Memory - 1GB Storage
Revision History	Ishmael – 2/10

SECURITY REQUIREMENTS

No:	SR-08
Statement	The Security Requirement shall: Ensure user privacy and protection | Securely access and store user data Ensure system data is hidden/inaccessible from user
Source	Team
Dependency	DAR-06, PER-03
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	How local data is stored will be dependent on user's mobile device. Will follow their security procedures. Users will not need to access system data and therefore will have no options to access system information. The application is inherently isolated from the rest of the device's software. The application will interact with an isolated folder that will associate only with the software. The software follows publishers' security requirements to ensure maximum safety for user's device (publisher refers to store that will host the files for download)
Revision History	Ishmael – 2/10

QUALITY ASSURANCE REQUIREMENTS

No:	QAR-09
Statement	The Quality Assurance Requirement shall: Ensure software will never crash | Ensure software is streamlined | Have easy installation Follow security protocols
Source	Team
Dependency	DAR-06, RR-07
Conflicts	N/A
Supporting Materials	N/A
Evaluation Method	Software will be updated regularly to ensure bugs are fixed if program breaks. With project files being organized and following the Project Management Plan we ensure easy maintainability and easy debugging. When error occurs program will close and must be reopened. By following minimum standards in order to interact with other devices and software we ensure security and ease of use. Minimizing bloat in the code and optimized algorithms will guarantee efficient data usage and fast run time.
Revision History	Ishmael – 2/10

SUPPORTING MATERIAL -NOTHING

Weekly Progress Reports

Project 7: Mobile sniper game

COP4331 Spring 2022

Contents of this Document

This document contains the weekly reports for the team (either cumulative or individual reporting) for the weeks working towards the submission of each deliverable. If cumulative report format is chosen, please make sure to include names of the team member(s) associated with each task.

Semester week # (include dates Monday through Sunday): 2/7 - 2/13

Cumulative team progress report

What are the completed accomplishments?

- **Completed Entire Deliverable 1**

What were the issues encountered (solved vs. unsolved)?

- **Figuring out coding standard, project functionality**
- **Scope of project regarding amount of code and file due to lack of previous experience**

What are the plans for next week?

- **Research regarding framework to build project off of**

Weekly Reports for Team

What are the completed accomplishments?

- **Software Requirements Specifications**
- **What were the issues encountered (solved vs. unsolved)?**
- **Figuring out device specifications by looking at google store requirements to run app services**
- **N/A**

What are the plans for next week?

- **Research to prep and build framework of project**
- **What are the plans for next week?**
- **Research to prep and build framework of project**