Software Requirements and Design Document

For

Group <17>

Version 2.0

Authors:
Kristen M
Brady W
Cade G
Dani S

1. Overview (5 points)

Jurassic Journeys is a single-player mobile game for iOS devices, particularly the iPhone. Players will guide a pixelated T-Rex across a scrolling landscape with a wireless external controller using up/down buttons to avoid obstacles and achieve a high score. Based on Google's own "Dino Game." This project is intended for regular users of all ages. The software will be coded using Unity so developers can animate the dinosaur and landscape, as well as customizing the game to feature different backgrounds and colors (and possibly shapes) for the character. It will feature an external controller with 4 buttons for up, down, menu, and speed. This controller will be designed and built from scratch using resources such as KiCad (PCB design and manufacturing), Arduino (prototype testing), Uduino (wireless integration), and GitHub (bug fixes and updates).

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

2. Functional Requirements (10 points)

	priority	details
1	high	Mobile game needs to synch with the hardware. The software will connect to the game via input from the controller, both must be functional.
2	high	Ensuring the controller can send signals wireless and can run fully on its own without bugs or errors.
3	high	Ensuring the software runs seamlessly every time and without bugs. Additionally, working to see if the Unity code can be exported as a mobile IOS game.

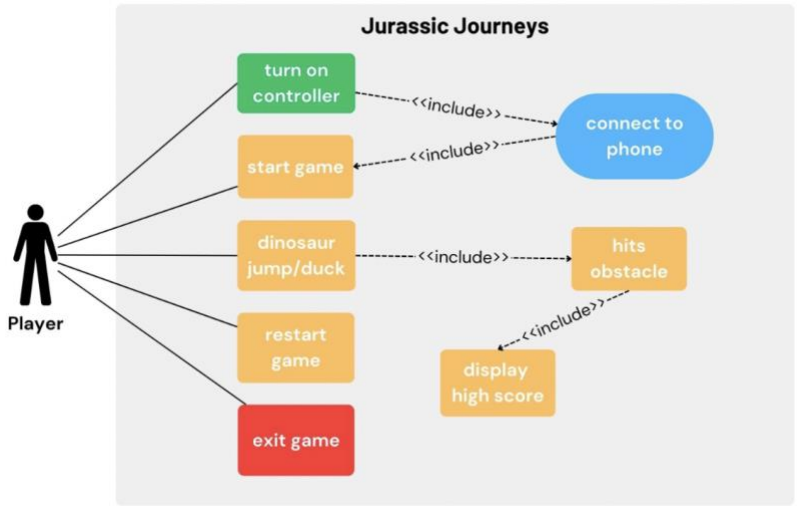
*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

3. Non-functional Requirements (10 points)

	Priority	Details
1	High	Mobile game should not crash. Multiple games should be played smoothly without crashing. As a game, any crash would effect the player's score.
2	High	Controller should not overheat or explode. Proper testing of the parts and electrical currents should happen so users don't get hurt while playing for extended periods of time.
3	Low	Game should be secure. Since we're using wifi and sending input wirelessly, we should make sure the data being sent and received is not susceptible to be hacked.

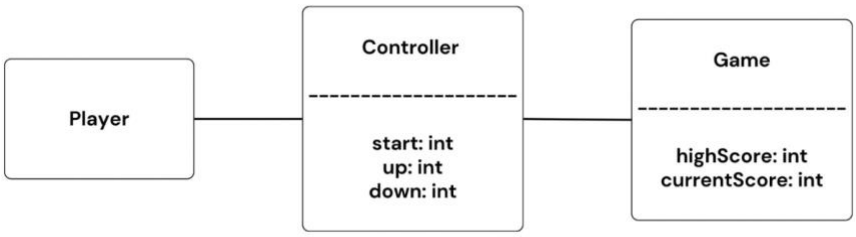
*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

4. Use Case Diagram (10 points)



*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*
***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

5. Class Diagram and/or Sequence Diagrams (15 points)



*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main **paradigm** in your system is **not Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

6. Operating Environment (5 points)

Our project will operate within the Apple iOS platform, the most recent update is version 17.1 (as of 10/30/2023), and run on most iPhones.

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

7. Assumptions and Dependencies (5 points)

This project could be affected by third-party suppliers for all of our hardware: each of the components for our printed circuit board will be made by foreign manufacturers. Even the PCB will be designed, assembled, and delivered by a specialized company. It can also be affected by changes to the software we use to design the game (Unity) and IDE we use to program our hardware (Arduino, Uduino).

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use,

issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.