# Technical Design Document (TDD)

## **COVER**

(Insert screen shot of all assets in one place here)

## **YOUR GAME’S TITLE**

Document version number (1)

Written by (Elegushi Ademola)

Point of contact (Elegushi Ademola st20236422.)

Date of publishing tbd

Version number (1.0.0).

**TDD Outline**

**Table of contents**—

**Development Requirements** — 3ds max, blender(animation) and unreal engine 5

* **Development** — IDE, test suites, editors etc.
* **Game Engine**— unreal engine 5.
* or **API Versions** — A list of APIs and version numbers, used in the game.
* **2D/3D Software** — all 3d assets were produce in 3ds max. and 2d assets(textures) were produced using Adobe photoshop 2024. While the animation
* **Project Management** — none.
* **Source Control** — Details of the server and client software used.
* **Sound Software** — Sound recording and editing software.

**Asset Specifications** — Details of supported asset formats and restrictions, for example model file formats, vertex winding, polygon counts or sprite file formats, width and height. Be sure to consider all required assets as appropriate to your game.

**Project Structure** —Details of the directory structure.

**File Naming Convention** —Convention for naming code and asset files.

**Level / World Details** — Layout and asset list including appropriate size limitations.

**Development Plan** —Outline of the production plan from pre-production to delivery.

|  |  |  |  |
| --- | --- | --- | --- |
| Milestones | Date | Deliverable | Approval |
| Pre-Production End |  | TDD  Prototypes |  |
| Milestone 1 |  |  | Ademola |
| Alpha |  |  | Ademola |
| Beta |  |  | Ademola |
| Final |  |  | Ademola |
| Pitch and Play |  |  | Ademola |

|  |
| --- |
| **Terminology:**  The terminology here is based on that used by Douglass in [Doing Hard Time](https://www.amazon.co.uk/Doing-Hard-Time-Developing-Addison-Wesley/dp/0321774930). Architectural design refers to the structural relationships between classes represented using Class and Package diagrams in UML. Mechanistic design refers to the interaction of runtime entities in performing particular behaviours represented using sequence diagrams. Finally detailed design is reserved for complex sequences of actions (Activity Diagrams) or state changes (State Chart Diagrams) performed by individual objects. |