**Project Report File**

**Project Overview**

For this project, I began with designing an Entity-Relationship (ER) Model on my iPad. This ER Model meticulously outlined each table, including the specifics of each column such as size, length, and data type. Additionally, it identified primary and foreign keys to establish the relationships between tables. To ensure the accuracy and functionality of the database, I implemented each table sequentially, testing them individually. In this project I also implemented a couple of extra tables to show what other tables could be used if I also had a video game implemented along with the website.

**Database Design and Normalization**

In analyzing functional dependencies, I took a game developer's perspective to determine which values are crucial for tracking and reporting. I aimed to capture essential player statistics and ensure that the database design is both efficient and scalable. The schema was designed to adhere to Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF), which guarantees minimal redundancy and eliminates potential anomalies in data manipulation.

**Functionality**

**Basic Functions**

* **Randomized String Generation**: A fundamental aspect of the project is a function that generates a randomized 10-character string for player IDs. This function uses loops and a predefined character set consisting of uppercase letters, lowercase letters, and digits to ensure the uniqueness and variability of player identifiers.

**Advanced Functions**

* **Triggers and Exception Handling**: Advanced operations include creating triggers that perform critical functions such as checking for duplicates in the loginTable and enforcing uniqueness constraints. These triggers are equipped with exception handling to raise custom errors if duplicates are detected, thus maintaining data integrity and providing meaningful feedback to users.
* **Complex Calculations**: The project also involves advanced techniques such as nested tables and VARRAYs to manage and manipulate collections of data. For instance, calculating the total playtime for each game session requires intricate calculations and updates to ensure accurate tracking of user activity.

**Backend and Frontend Development**

**Backend**

* **Python and Flask**: For backend development, I utilized Python along with Flask, a lightweight web framework that facilitates the creation of web applications. The backend handles database interactions, processes user inputs, and manages sessions. The development process was executed in a reverse order: beginning with SQL code to define and implement the database structure, followed by integrating Python to manage database operations, and concluding with frontend design.

**Frontend**

* **HTML, CSS, and JavaScript**: The frontend was designed using HTML and CSS, with a touch of JavaScript to enhance user interaction. The user inputs are collected through HTML forms, which are then processed by Python to update the Oracle database. The user is redirected to appropriate pages based on their actions, such as logging in or logging out.

**Data Flow and User Interaction**

The data flow begins when users interact with the HTML forms to input their information. Python scripts process these inputs, update the Oracle database with player data, and handle session management. Upon logging out, Python again updates the gameSessions table to reflect the total playtime for the user. Users are then redirected back to the original page. The Python terminal provides feedback in case of errors, such as attempting to sign up with an existing account or entering incorrect login details.