Database Justification

Team Name: Education Gamification Team

Team Members:

Justin Mitchell

Matt Adams

Nathan Luiten

Trevor Stewart

Date: 2/14/2025

Eastern Washington University

Introduction

For our project, a NoSQL database made the most sense. Since we are making a game for students to learn and play through the story of Macbeth, the goal for our database is to be able to store their progress so they have the option to return where they left off after school.

This means we would need to store things such as:

- What scene they we last in.
- Their position in that scene.
- What items they held (inventory).
- Who they were (email and username).
- Score for the leaderboard.
- A timestamp of when they last saved.

Being that this is a simple structure to store, this makes the json style of databases a good option for us. Below is the schema that we settled on, using the types that Firebase uses.

Our json schema for Players:

```
{
    "SaveData" : {
        "scene" : { "type" : "string" },
        "score" : { "type" : "number" },
        "inventory" : { "type" : "array" },
        "position" : {
            "x": { "type" : "number" },
            "y": { "type" : "number" }
        },
        "lastSaved" : { "type" : "timestamp" }
    },
    "SchoolEmail": { "type" : "string" },
    "Username" : { "type" : "string" }
}
```

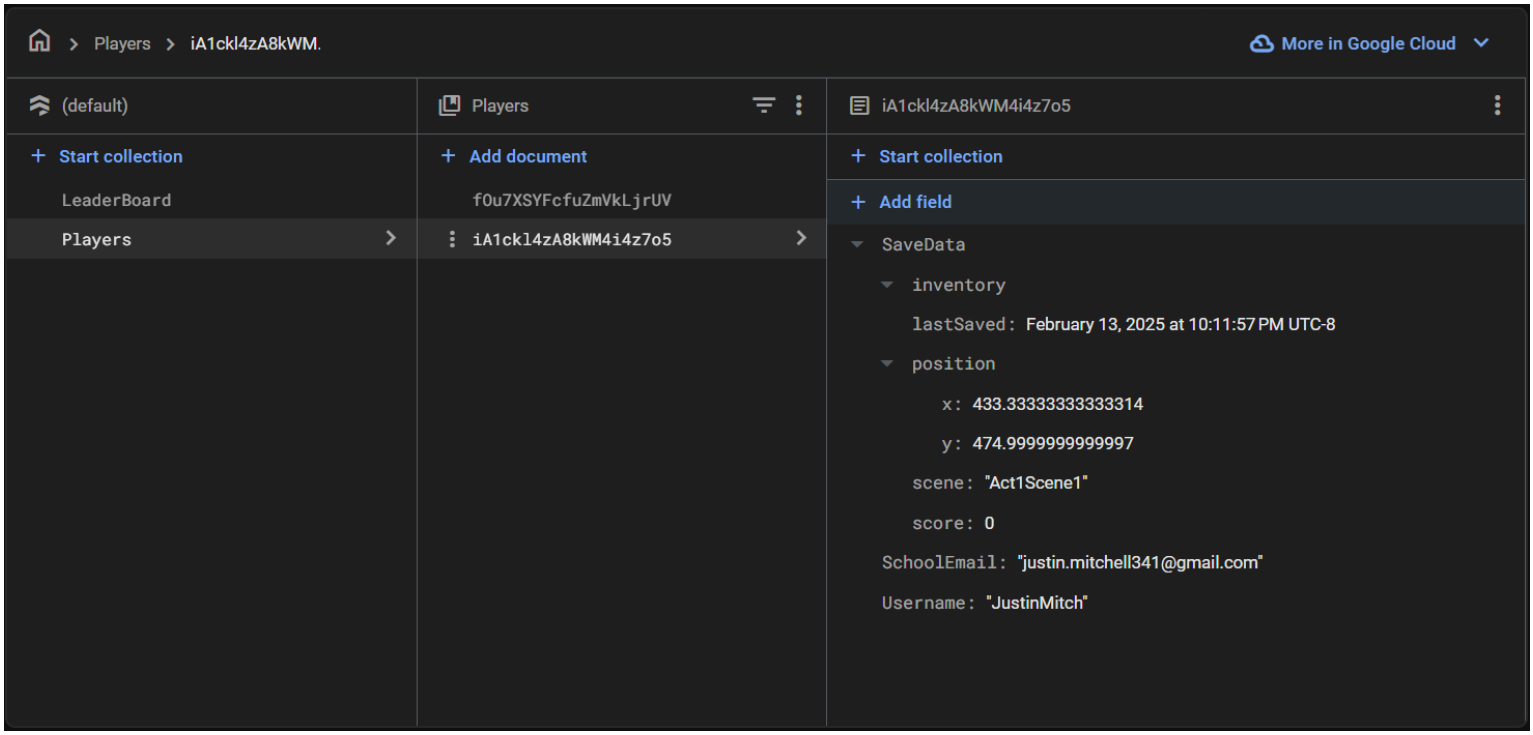
Why Firebase?

One of the big perks of working with Firebase is being able to use their email verification for accounts. This means we don't have to worry about storing hashed passwords, providing a way to reset passwords, as well as the bonus that users won't have to remember yet another password.

The downside of Firebase is with complex queries, searches that use more than one property or field will require index on that combination, which is unfortunately a feature you must pay for in Firebase. Luckily for us, our project will never require queries like these. There are only two queries that will ever need to be done within the game.

1. Reading and writing data to a players account, based on the email.
2. For the leaderboard, pulling the top players based on score.

Neither of these require us to search on more than one field, making Firebase a perfect fit for our project. We have already implemented everything that needs to be done with the database, and have no issues or plans for future additions. Below is a screenshot of our Players collection:



Code Snippet

Here is a code snippet of initializing a new account into the database:

```
//Register User
export async function registerUser(email) {

    // Check if the email already exists in the Players collection
    const playersRef = collection(db, "Players");
    const q = query(playersRef, where("SchoolEmail", "==", email));
    const querySnapshot = await getDocs(q);

    if (!querySnapshot.empty) {
        alert("This email is already registered.");
        console.warn("Email already registered, preventing duplicate registration:", email);
        return false;
    }

    const username = prompt('Enter your desired username:');
    if (!username || username.trim() === '') {
        alert("Username cannot be empty.");
        return false;
    }

    try {      // Normalize email to lowercase
        const normalizedEmail = email.toLowerCase();
        const docRef = await addDoc(playersRef, {
            SchoolEmail: normalizedEmail,
            Username: username,
            SaveData: {
                scene: "Act1Scene1",
                score: 0,
                inventory: [],
                position: { x: 100, y: 100 },
                lastSaved: serverTimestamp()
            }
        });
        alert("Registration successful!");
        return true;
    } catch (error) {
        console.error("Error registering user:", error);
        alert("An error occurred during registration. Please try again.");
        return false;
    }
```

Breakdown of the Snippet

When you go to the website for the very first time, the user will be asked to either login if they already have an account, or be asked to register a new account. The snippet is a chunk of code that processes their attempt to register a new account.

First there is a pull from the database to see if there is already a matching email, if there was then the user is told that the email is already tied to an existing account. If not then they can continue on to make the account.

Next the user is asked to set a username, this is what will be displayed on the main menu as well as the leaderboard. Username do not have to be unique as they are not what are used as the main identifier/primary key. All that is required is that it is not left empty.

The last step is in the try block where we actually create and add the account to our ‘players’ collection in Firebase. This includes some fresh save data (empty inventory, starting at the first scene), as well as the username and email to identify the account.