Purpose

The product aims to offer a streamlined yet enjoyable method for managing tasks and habits. Unlike other task apps that can be overly complex, we prioritize simplicity by integrating due dates and difficulty levels into each task. By completing tasks, users earn gold coins, which can be used to purchase items that enhance success in the game. This gamified approach increases motivation, making it more likely for users to achieve their goals and realize their aspirations.

Reliability, Performance, Functionality, Competitiveness, Compatibility of our technologies

Kotlin- We wanted to start something from scratch and Kotlin makes it super easy to make an android app. It can be used in android studio, which has UI components so that we can easily customize and design our app.

Python/Flask - Python is very easy to learn, use, and set up a server for our limited time on this project. We decided to do the backend server in python to set it up quickly and have quick transactions with the SQL database to update our minigame logic quickly but efficiently.

SQL - Very competitive in the world of database management because it is a powerful and versatile language that allows for efficient querying, updating, and managing of data. It is widely used in various industries and is supported by python really well with lots of documentation.

Lifespan, Deployment, Support and Cost

Overall, one of our main priorities was choosing languages that would be quickest to build a simple project which was a success. It also worked out that on a small scale it would be free to use these technologies, but they also have easy ways of scaling up if we have more time after release.

Biggest Issues

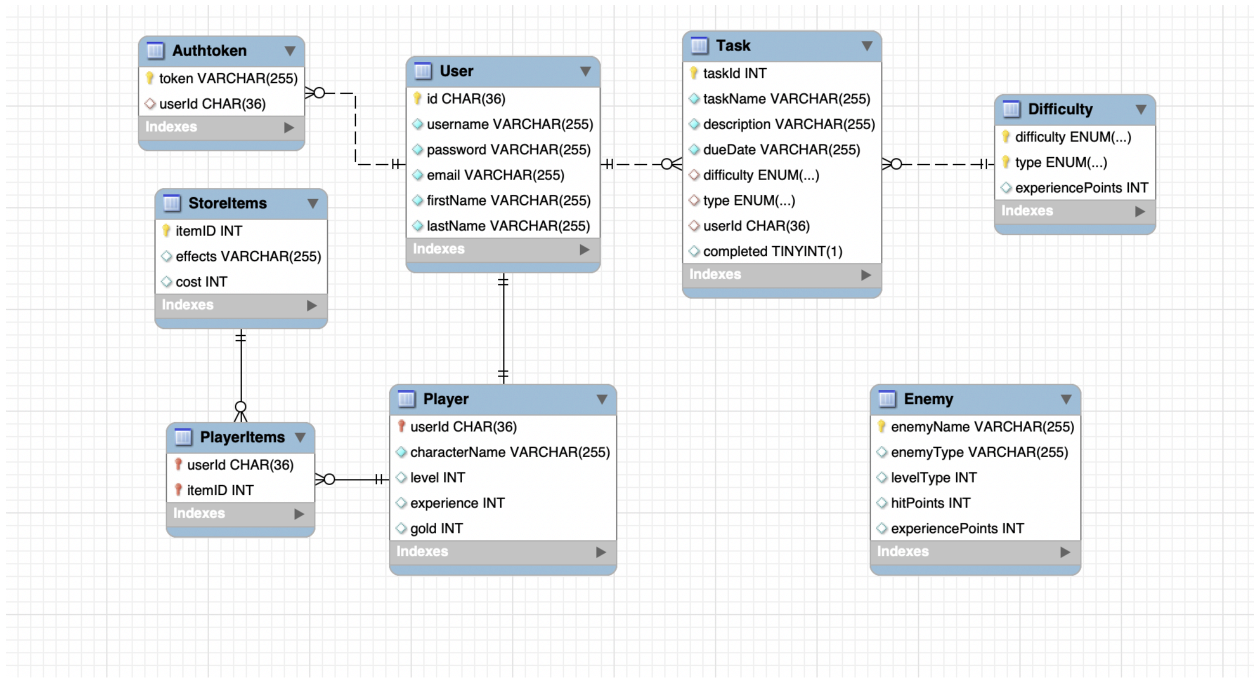
After reviewing the Design and Architecture below there were a few larger issues that we decided to give more time for including:

- **Game logic with all of the methods listed**
- **Connecting the Front-End to the backend with all of these methods**
- **Notification for an app**

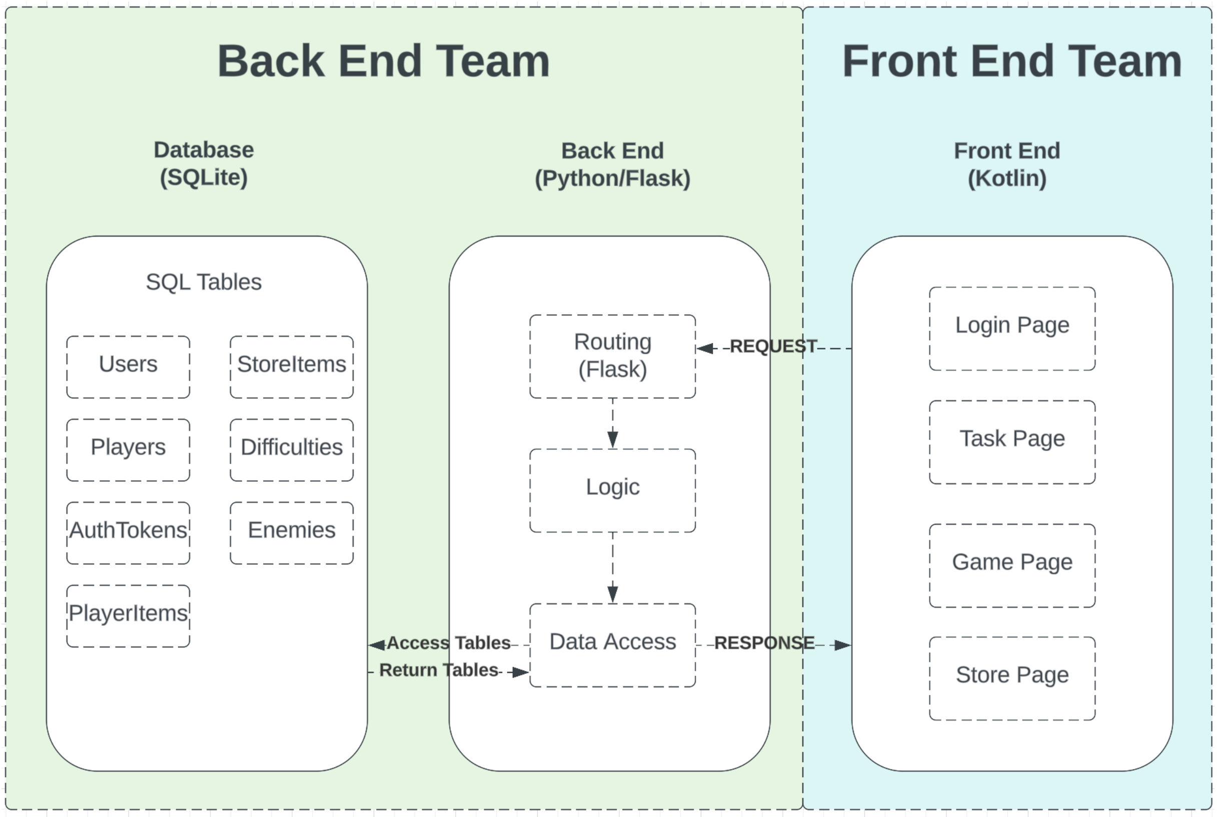
*** To make this issue easier we created a READ.me for all developers to refer to so that they know what data will be passed from the frontend to the backend and what the methods will look like from both ends.**

Tables & Methods

Tables & Methods READ.me: **Attributes and Methods**



How they Connect



Quality Assurance