Development Process and Challenges During the Project

At the start of this project, the goal was to create a Space Gothic Character Generator that would allow users to generate and manage their characters dynamically through a Streamlit web application. The core features included generating character attributes, handling inventory and combat mechanics, and providing a personal dossier for character details.

One of the main ideas was to allow users to save their character sheets by implementing user authentication and data storage using MongoDB. This would enable users to register, log in, and return to their saved progress. It would have made the application more useful that way, instead of being a single-session generator as of now.

The first step in development was rebuilding the core character system from the TBI project, where attributes would be generated randomly with rerolling options. I used Streamlit's session state to ensure values persisted between interactions, which worked well and made the experience smooth. Once this part was functional, I moved on to the inventory and combat system.

The main challenge in combat mechanics was designing the shooting system. I used the stats from only one gun in the game, for simplicity. Eventually, I want to make a system where every gun in the game can be selected.
The system had to track ammunition in a magazine and visually represent it, while also handling different firing modes like single fire, semi-burst, and full burst. Hit chances needed to be calculated based on skill level and range, with successful shots dealing randomized damage. Displaying the results in a fixed space without affecting the layout took some trial and error, but after tweaking Streamlit's layout settings, I found a way to keep it stable.

With the core mechanics in place, I turned to user authentication and MongoDB integration. Setting up a MongoDB cluster and creating collections for user data and character sheets went smoothly at first. The plan was to connect the app to MongoDB Atlas using PyMongo so that user data would persist beyond a single session.

However, once I deployed the app to Streamlit Community Cloud, I ran into timeout issues. The main problem seemed to be a TLS/SSL handshake error, according to ChatGPT troubleshooting. I checked the connection string multiple times, adjusted MongoDB's network settings to allow all IP addresses, and even experimented with different authentication parameters. Despite trying various fixes, including disabling certificate validation for testing purposes, nothing worked within the available time.

Eventually, rather than spending more time troubleshooting the database connection, I opted to remove MongoDB integration and rely on local session storage instead. While this meant that users couldn't save their character sheets permanently, the app remained fully functional for generating characters, managing inventory, and simulating combat.

Looking back, the project was a great learning experience. I was able to work with Streamlit's UI components, session state management, and game-like mechanics while also diving into backend integration and cloud database troubleshooting. Even though MongoDB integration didn't make it into the final version, I gained valuable insight into handling database connections and authentication systems. If I had more time, I would revisit the database implementation or explore other storage solutions, like local JSON files or a different database provider.

From my friends I got great feedback for the project, since they all play the game, the character sheet is designed for. They see the potential the program has, but of course also miss customization and different character classes. These in combination with being able to save the character would make the project actually viable to use in our private setting.

Tom Koehler 26.02.2025