Project Title : Architect

Student Name: Zachary Hightower

Student Email : zphighto@go.olemiss.edu

Sponsor : Jeff Lucas

Sponsor Email : jhlucas1@olemiss.edu

Project Overview

I want to develop a tool to help authors better visualize and develop the narratives they piece together. This stems from a love for maps in fantasy novels and tabletop role-playing games. Architect is a way to create an interactive and informative map from the user's input that can scale from a single town to an entire world. I expect the primary users to be authors and video game developers.

User requirements

The minimum viable product is a web application that allows users to create a map of interconnected locations. Users may organize their locations in a separate list view. Users may also customize the appearance of the locations on the map using available assets or their own uploaded assets. Users may add entries to the locations. Entries will accept user text input, allow for basic formatting, and their header icons may be customized by users. Entries like locations, can be organized into groups.

I expect Architect to have two different types of users, the system administrator and the builder. The system administrator will handle updating the system and ensuring smooth user experience. Builders are the main users of the system. As such, the system will do its best to cater to the needs of the builders. These are our authors, tabletop enthusiasts, and other assorted users. Their use of Architect is divided between viewing their builds and contributing to their builds.

- As an admin I want to ensure the security of the database so that no unauthorized user can access it.
- As an admin I want to gather user feedback so that I can better support the project in the long term.
- As a builder I want to create a map so that I can visually organize and structure my ideas.
- As a builder I want to input entries so that I can keep track of details about my world.
- As a builder I want to format entries so that they are pleasant to read.
- As a builder I want to organize my locations and entries into groups so they are easily identifiable.
- As a builder I want to customize the appearance of locations so that my world looks good.

Design Choices

While HTML, CSS, and Javascript are the only viable options for development of web applications, it is still important to evaluate the benefits and issues brought to the development process.

A big advantage is that I comfortable working with these languages. This comes from my university classes and from personal projects. A variety of different ideas can be developed with these languages. They can be used across a multitude of different platforms. There are also many tools which I can leverage to speed up the development process. Sped up development means I can reach a prototype for user testing faster than I would with something that is not a web application.

The disadvantage is that I don't have a lot of experience attaching a database to web based applications. The project may hit performance bottlenecks with very large maps or service of large amounts of users. The resulting slowdown may significantly harm user experience. With web hosted applications like Architect, it is also difficult to assure security. Offline access is also going to be limited, or non-existent. Adding features may prove difficult.

I do have a significant choice between relational and non-relational databases. Either could function for the project.

Relational databases, like My SQL, have a rigid structure to them, which can help with defining project needs and scope. The primary and foreign key structure of the relational databases makes linking parts of the project more intuitive than with non-relational equivalents. Relational databases are also well known and have extensive documentation that helps with any potential issues one might face. Queries in a relational database can be handled with speed and efficiency. I already have some familiarity with constructing the framework of a relational database.

On the other hand, the rigidity of a relational database makes it less flexible than a non-relational database. Scaling for new tasks or different data becomes a big challenge. As user scale goes up, relational database implementation of Architect may lag behind a non-relational database

implementation.

Non-relational databases, like Firebase, offer flexibility as one of their biggest benefits. They deal with unstructured data well. They also scale better than relational databases, so there should be no significant issue with meeting new tasks or user needs. It allows for storing things in JSON like documents which could make storing the maps or world information for Architect much easier.

Consistency in non-relational databases may pose an issue. Support for complex querying is also lacking. This sort of database tends to be more difficult to learn with less robust documentation than equivalent relational database options. I also have less experience structuring a non-relational database. Data duplication and disorganization may also become an issue, and are not managed by the structure of the non-relational database, unlike a relational database.

Timeline

September 30 – October 13, 2024

Creation of the basic mapping and entry system

October 14 – October 20, 2024

Personal testing, initial user testing, and bug fixes

October 21 – October 27, 2024

Focus on user login experience and creation of explanatory tooltips

October 28 – November 3, 2024

Implementation of custom icons for locations and characters, and integration into the system.

November 4 – November 10, 2024

Creation of light and dark mode, addressing underdeveloped features noted by user and sponsor feedback

November 11 – November 24, 2024

Final testing, refinement of features, and ensuring the application meets or exceeds the minimum viable product