Matt Walsh

W0075056@nscc.ca

INFT4000 Special Topics

Portfolio Assignment Planning Document

Contents

[Project Description 2](#_Toc57369212)

[Project Functionality 2](#_Toc57369213)

[Features 2](#_Toc57369214)

[Frameworks 2](#_Toc57369215)

[Data Persistence 2](#_Toc57369216)

[Simple Data Structure 3](#_Toc57369217)

[SQLite3 Quirks 3](#_Toc57369218)

[Data Sharing 3](#_Toc57369219)

[Wireframes 3](#_Toc57369220)

[Main View 4](#_Toc57369221)

[Spell Card 5](#_Toc57369222)

[Add Component 6](#_Toc57369223)

[Repository 6](#_Toc57369224)

# Project Description

Throughout the year, we have learned how to create native apps using web technologies, using the NodeJs package Electron. During this time, I created an app that allows a user to store spell descriptions in a database, retrieve them, and modify them when needed. While the project is functionally complete, the user interface could be considered novel.

To that end, for my Portfolio Assignment, I propose recreating this application with a modern user interface. In addition to redesigning the interface, I will be implementing a search feature, that will allow the user to search for spells via several criteria. I will be using the web framework React to create a Single Page Application (SPA) within Electron, removing the need to launch separate windows for tasks such as adding or modifying spells.

The original app also included a way for a player to track their spell slots, however this data did not persist between sessions. This will be implemented in the new version

# Project Functionality

## Features

The app will provide the following functionality at a minimum:

* Store spells entered by the user. Spells have he following data associated with them:
  + Name
  + School
  + Level
  + Casting Time,
  + Components (Verbal, Somatic, Material)
  + Ritual Casting
  + Concentration
  + Source
* Edit stored spells
* Remove stored spells
* View spell details
* Search for spells by all available criteria
* Track spell slots (used and max)

\*All data will persist beyond the applications life cycle (Stored locally)

## Frameworks

The Following Frameworks will be used:

Electron and NodeJS – Technology to make a native app using web technologies. Core of project. Everything will be contained in this framework.

React and React Router – Technology to make Sigle Page Applications. Includes its own component system for updating content on the screen. Will be using this as primary UI framework, controlling what pages are displayed on the page via routes.

SASS – Technology to add functionality to CSS. Allows the use of variables and basic logic in CSS. Also has a robust module system, allowing such features as mixins. I will be using this to simplify the creation of styles for my application.

## Data Persistence

Due to the simplistic nature of the data needed to be stored, I have opted to save data to disk as a JSON file, rather than store the data in a database. I will attempt to defend my choice below:

### Simple Data Structure

There is only one table in the application, so it is unlikely complicated queries will be required. It is also likely that a level nine wizard will ever learn more than one hundred spells. Because of this, the search speed will not benefit from using indexes.

### SQLite3 Quirks

While I was able to eventually get SQLite3 working for my previous assignment, it was not without its issues. The most relevant being the requirement to install both Visual Studio for its C++ compiler, and Python to run compilation scripts. I wish to make this app easier to build than the last, so I opted not to use SQLite3.

### Data Sharing

In the unlikely event data from this app could be used in another app, having the data stored externally would make the process easier

# Wireframes

These are images exported from the diagrams located in the Diagrams folder. You will need to use [Draw.IO](https://app.diagrams.net/) to view these diagrams.

## Main View

Diagram, shape, polygon

Description automatically generated

## Spell Card

Graphical user interface, diagram, text

Description automatically generated

## Add Component

A picture containing text

Description automatically generated

# Repository

This project’s source code will be hosted on my [Github](https://github.com/matt-walsh/fizznizzards-spellbook) page.