Interim Report

HOSTING A REACT.JS WEB-APP TO AUTOMATE STARS WITHOUT NUMBER CHARACTER CREATION

Dominic Cousins | c03201 - Computer Science Project | 11/2020

Contents

Aims and Objectives

Stars Without Number (SWN) is a table-top role-playing game in which players improvise with a narrator according to some rules. The game systems are quite complicated, and a large amount of information must be calculated, recorded, and stored by players in character sheets or other documents. Additionally, players must execute game systems manually by following the rulebook. This includes character creation, which can be a daunting experience.

The project is to create and host a webapp that automates the character creation process. The primary aim of the proposed system is to increase the speed and ease of character creation for SWN. A secondary aim is to facilitate the storage of custom objects, rules, and characters by users. Additionally, these custom entities should be functionally supported by the webapp (i.e. they can be used to create characters). As SWN is a game of improvisation, a lot of players like to customize the game and rules. Therefore, this secondary aim ensures they can still use their own rules while using the tool to enhance their character creation experience.

My objectives include the necessary steps required to develop a react.js webapp, together with a node.js server and MySQL database. To expand the functionality for users and really cater to the needs I heard, I will develop a way for users to enter custom game objects into the tool, as well as customize the rules. Also, an algorithm will be made to recommend choices for players, mostly with newer players in mind.

Objectives:

- Research the character creation process in detail and investigate popular house rules (customizations to rulesets that player groups use)
- Create a database for storing game objects (entities in the game such as skills, classes, or equipment) and character data, and associated TypeScript models to represent the objects
- Host a Node.js web server that connects to the database and serves API requests for data from it. It will also distribute the JavaScript client
- Write React.js-based webapp to allow viewing of stored objects and the interactive creation of characters following customizable rules
- Develop JSON formats for storing entities and ruleset customizations and support them in the tool
- Create an algorithm to suggest game object picks and help guide players. This can be utilized in a semi-random generation mode to create random, but balanced characters

Literature Review

To prepare for this project I have looked at documentations, blog posts and tutorials to learn about all the features available and best practices for development. And indeed, these can be continuous sources of information as I encounter new problems to solve once the project is underway. Additionally, the rules for SWN must be understood and available to refer to.

GAME RULES

The source of information when it comes to the game rules is the rulebook [11]. The whole rulebook is one big reference on how the game functions, however all content after page 93 will not be needed for this tool specifically, it pertains to other areas of the game. On page 4 we have the "summary of character creation". According to the rulebook there are 18 steps in character creation. Some are overly verbose, and I have condensed them down to the following steps:

- Either: roll 3d6 (3 6-sided dice) for each of the six attributes, then set one to 14; or assign the following array to the six attributes: [14,12,11,10,9,7]. Then calculate the modifiers for these attributes
- Choose any background from the list of backgrounds. Gain the associated free skill.
- Either: pick any two skills from your backgrounds learning table (or the two quick skills); or roll three times as you choose divided between the learning and growth tables for you background
- Select player class (or, optionally, two classes for multi-classing)
- Choose your foci (players get one, expert and warrior classes gets one extra non-combat and combat focus respectively)
- If they are enabled, origin foci (alien species), are available to be picked instead of classical foci. They describe your characters origin in terms of species. For example, being an alien, or an elf.
- Pick one extra skill to represent character hobby
- If you chose psychic class: select one (for partial psychics), or two psychic disciplines. Calculate effort = "1 plus their highest psychic skill plus the better of their Wisdom or Constitution modifiers"
- Calculate hit points: roll 1d6, add your constitution modifier, then add two if you are a warrior
- Note down attack bonus, which is zero by default, warriors get +1
- Choose an equipment package to select characters starting gear. Or you can instead roll 2d6 multiplied by 100 to get starting credits, and use these to purchase gear
- Mark down hit bonus and damage done with each weapon you have

- Record your armor class and saving throws. These are based on your armor and attribute modifiers
- Give the character a name and at least one goal. Completing goals is one way to earn experience in SWN. Additionally, fill out any other flavor fields as wanted (for example the back-story of your character)

Traditionally these steps are executed manually by players. However, as is clear, there are a lot of steps. Additionally, many of the steps (e.g. selecting background) requires the player to refer to other pages in the rulebook to see lists of available objects and more in-depth explanations. The steps are given in the order that the base rules have them executed. This exact order will be available as a mode in the tool; however, the order can also be customized.

TOOLS

A massive source of information for this purpose is the official React.js documentation [1]. This includes a couple of quick starter guides which, after reviewing, are not in depth enough to be very helpful. However, there is also several advanced guides. These go over the technicalities of achieving the goals, but also advise on best practices to implement. Already useful is the guide on error boundaries [1.1] which details how to create a special type of class component that acts as a try catch block for react components. As described in the guide “In the past, JavaScript errors inside components used to corrupt React’s internal state and cause it to emit cryptic errors on next renders”. These errors were unrecoverable since there was no mechanism for catching an error outside of the component itself that emitted it.

Also included are descriptions and guides to using hooks [1.2] which allow functions to access class-like features such as persisting data across renders by using state. This enables functions to be used as full-fledged components instead of classes. Sometimes we do not need all the encumbrance of initializing a full class and instead a function with a few hooks is good enough. A big continuous reference is the API reference – the conventional documentation [1.3]. This puts forth all the functions made available by the library and how to use them.

While of course the documentation for any libraries and tools will be at least browsed, another is especially notable: the Sequelize documentation [2]. The documentation outlines how to create models that represent the database entries so we can pull them straight into objects. Understanding how to do this properly is important to enforce relations in the database. Also, very useful is the description of how to query the database for objects. Sequelize allows us to query by attributes, instead of directly producing SQL. Therefore, knowing how to get as much power out of this feature as possible is essential to not waste memory or CPU resources further filtering results later.

To kick-start the project and learn the practical impacts of certain decisions, I referred to some blog posts and tutorials. These include general and starter guides, as well as some more in-depth pieces for future reference.

I surveyed a lot of starter tutorials and general guides from people using the same technologies as me to find good formatting and layout for the project. The most useful and appropriate tutorial [3] details how to setup a to-do tracking app using the technologies I intend, except with a different database provider. The directory structure used within conceptualizes the API as controllers, models, and routes, with 1 of each per entity in the database. Additionally, the client must define TypeScript interfaces for each model to represent the entities as JavaScript objects.

Another useful tutorial is about distributing the react front-end from a node.js server running inside a docker container [4]. This will be useful later since a working build is needed to put into such a container.

I have also found 2 blogs that will help make future decisions for the project. In one [5], the author tested performance of objects and maps in JavaScript. Since these are the main ways of storing entities with many properties in JavaScript (and many such objects will need to store and manipulated), choosing the right ones could have notable performance impacts. Understanding which is faster and when, together with their other advantages, will allow me to make that optimization at a later date. Another [6] ranks 7 UI libraries for react. A UI library may be needed to deliver a quality UX without re-inventing the wheel.

SIMILAR SYSTEMS

Several similar systems were surveyed, however there are few that do what I propose. My project will enable character creation for SWN within a webapp. The only comparable tools for the same game, Stars Without Number, are excel spreadsheets [9], [10]. They can be found in the resource's directory of the repository. These spreadsheets are a good source of information since the tables of game objects they use can be exported to CSV. These were then processed by a script I wrote to be entered directly into the database. However, in terms of functionality, they are very lackluster. Some calculated values are automated, but no game rules are enforced so they are more like a place to store information than a tool.

When looking at another game, Dungeons & Dragons (D&D), there are a few online character creators available. The most polished of these is Reroll [7], a pixel art visualizer for characters that also functions as a mechanism for generating characters. Its focus is clearly the aesthetics, as seen in figure 2 and 3. It includes a free fill mode where users can enter any values into any fields and a guided method which is very rudimentary. Both result in a nicely laid out character sheet that can be referred to later. Another similar tool for D&D is the character creator by NineTail [8]. This is much more in-depth than reroll, featuring the full rules and always enforcing them. The UI, however, is extremely basic, (as

seen in figure 1) comprising essentially lists of radio buttons and checklists with little explanation of how it all works.

The basic excel sheets that exist for SWN and the fuller tools for D&D shows the desire for such products, but none yet exist that do what I propose for SWN. As such I would release the resulting product as an open source contribution to the community. Additionally, it may be possible to host a server so clients can freely use the original version of the app.

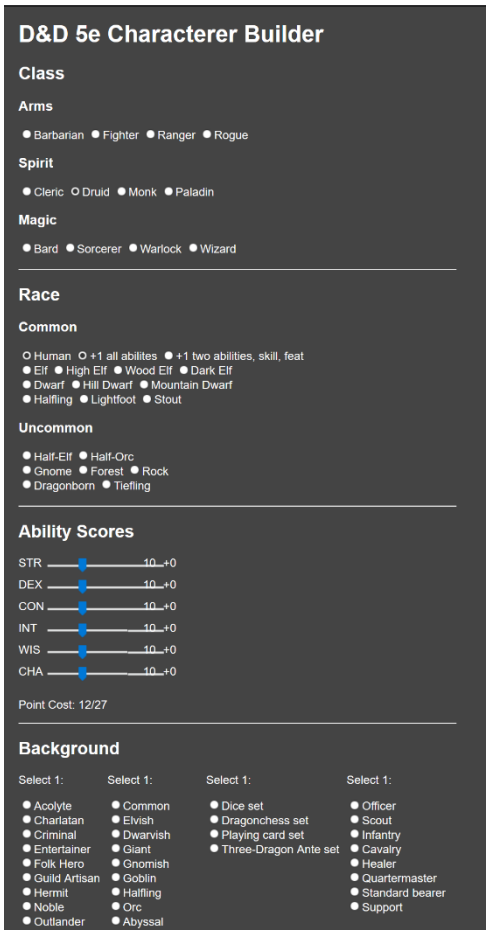

Figure 1 (above): NineTails [8] Character Creator. Below is simply a larger array of radio buttons for further options

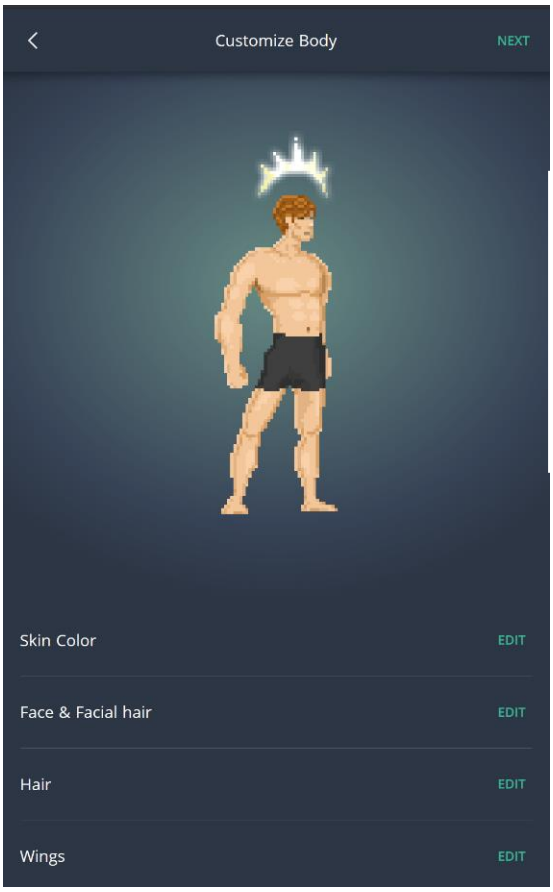

Figure 2: Appearance customization page of Reroll Character Creator [7].

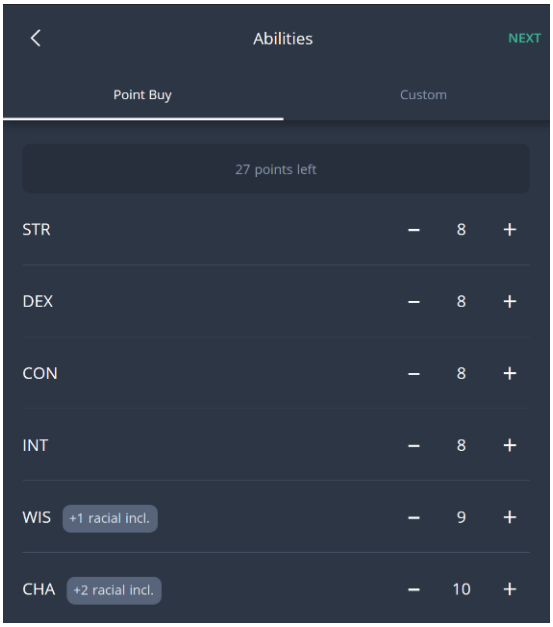

Figure 3 (right): Attrbiutes page for Reroll Character Creator [7]

Requirements, Specification and Design

REQUIREMENTS

Webapp

- Can output character sheets as a spreadsheet, for printing, or as a JSON file
- The tool enforces all character creation rules outlined in the SWN core rulebook. However, users are also able to add arbitrary amounts of all resources, from 'custom' sources. This enables house rules and other customizations
- There are multiple modes which change the rules slightly and may have different UI:
    - Traditional - Guides the user, strictly follows the character creation rules one by one, including the order of execution
    - Choice Optimized - Guides the user but allows jumping forward/backward
    - Advanced - Unguided, the user can fill in the sheet in whatever order
- An additional Custom operation mode will allow users to change the rules of character creation and levelling
- The settings for this mode can be imported and exported from/to JSON
- A random character feature generates a randomized character within 5 seconds
- A simple wiki section lets users search game objects, including custom ones
- It can fulfil such searches in a few seconds for the first page, and up to 1 second per page where more are needed
- The wiki can be accessed while generating a character, maybe as some sort of modal
- Performs other small but useful tasks such as rolling n m-sided die
- (optional) Users can add custom game objects to a database. They can make public groups of these objects to share them with others
- (optional) Publicly shared custom game objects can be copied to a user's own collection
- Custom objects can be imported and exported via JSON
- (optional) A special mobile version of the webapp is served to mobile devices which is optimized for viewing and using on a small screen:
    - Capable of running on a phone with 2GB of RAM
    - No visual lag or loading delay of more than a few seconds
- Content sent over the internet should be the smallest possible size to minimize mobile data usage
- An algorithm can offer suggestions by ranking available choices with a score.
- The suggestion algorithm is used to randomly generate balanced characters

REST API

- The server implements a REST API, allowing clients to query the database and perform operations on the signed in user's custom objects
- The web server responds to unauthorized requests for data about non-custom game objects including:

    - Getting all game objects of each category, or individually by ID
    - Filtering categories of game objects by property
- Can request the expansion of fields in game objects, thus returning related objects in the same request. The full data for all fields is not necessarily supplied by default, else a large chain of related objects would need to be fetched
- (optional) Supports a query language for searching all game objects in the database in one request

ARCHITECTURE

The result of the project will be an interactive webapp, that uses a nice UI to facilitate guided or automated creation of characters for SWN. The proposed system is comprised of 3 main parts: the client, the server, and the database. These can be seen in figure 4, a diagram of the architecture.

The client can be accessed from a browser and will contain an object lookup page (wiki) and the main character creation tool. It will be written in TypeScript, which compiles to JavaScript, and use the React.js UI library for front-end development. I chose React as it has become increasingly popular recently due to its component-based approach to web development. It allows us to essentially generate html elements, together with logic, using JavaScript. Webpack bundles the verbose TypeScript files all the way down to a single minified JavaScript file to be distributed by the server. I chose to use TypeScript, even though it is added complexity, because the additional debugging and control of types is too good to pass up. When creating so many interfaces, including converting between custom types, it is so useful to be warned of type incompatibilities.

Node.js will be used to host the server and is a server runtime for JavaScript. JS is well-suited to web applications. We can easily convert between JSON content and JS objects. This allows us to use the same language in both client and server, to keep everything simpler. The server will serve data from the database to clients, even third-party applications. Also, it will distribute the web content, but all other logic is handled by the client itself. As an additional stretch goal, the server should be able to connect to arbitrary third-party databases to access more custom game objects. The server setup is comparable to a classic model view controller (MVC) layout: the API routes are defined as models, routes, and controllers. Routes let the client request the entities for it to render, thus acting as the view.

A MySql database will be used to store game objects and user data. A separate schema is to be used for each of these. MySQL is chosen because it is very standard, while also allowing the storage of JSON content in table cells. This is most useful for storing lists and JSON rulesets directly in database cells. Sequelize will be used to interface with the database. From the docs: “Sequelize is a promise-based Node.js ORM for [various database providers]”. Promises are a mechanism of asynchronous execution in JavaScript. An ORM is an object-relational mapping which allows us to represent entities/rows in a database,

as objects in code. So Sequelize lets us perform asynchronous requests to the database, and store results as JavaScript objects based on defined models. This makes handling, manipulating, and rendering this data much more straightforward.

The architecture can be seen laid out in figure 4. The Node.js server acts as a middleman between the clients and databases via HTTP requests and the Sequelize connector.

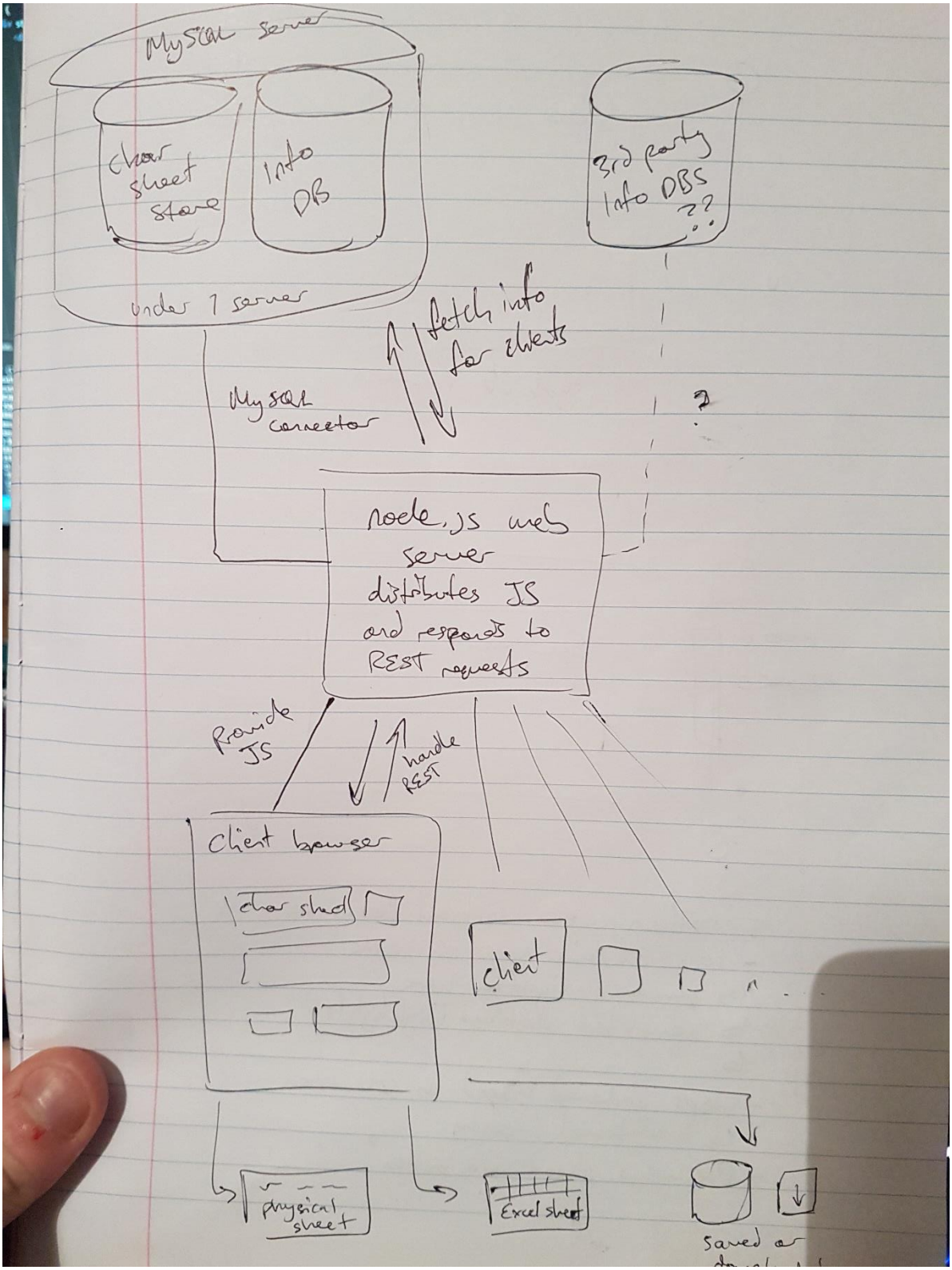

Figure 4: Diagram of project architecture

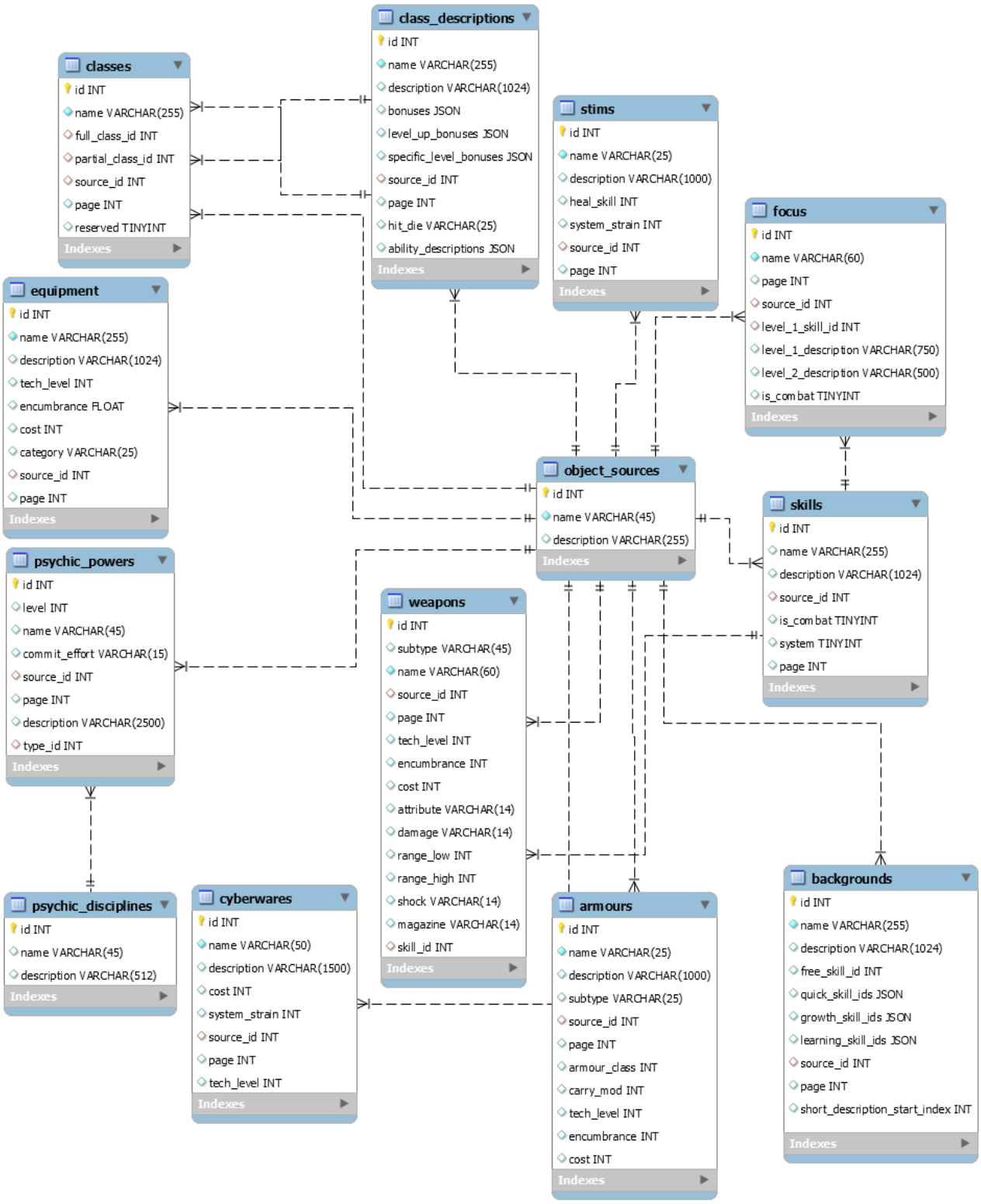

Figure 5: Objects Database ERD

Planning

WORKING METHODOLOGY

I will develop the project while following agile development practices. This involves working in sprints and performing further planning at the end of each sprint. The sprint length will be two weeks, as this should allow me to complete a few major features each sprint alongside my other studies. At the end of these two weeks, a build will be pushed to the release branch and features will only be added to each release if they are complete and tested. Another aspect of agile is getting constant feedback from the customer. This allows the product to adapt rapidly to feedback. I will personally analyze the product on the release branch each sprint to evaluate it. In addition, I have a group of players for the game SWN, including a game master (GM), who is responsible for hosting games. They shall review the product at key milestones, potentially as often as each sprint, to get their feedback as my customers.

Tasks are tracked in Trello, a project management tool that holds lists of tasks. The board can be found here (https://trello.com/b/AFs1wFcp/final-year-project) and includes major milestones, together with a backlog of more granular tasks to be added to sprints. The backlog tasks can be refined upon being added to a sprint when more is known about what they entail. Each large task in the board will have its own branch, to be merged into master then released if it is completed within a sprint.

PROJECT PLAN

I have produced a Gantt chart to help visualize the project schedule (see figure 6). Two periods in the chart is equivalent to one week. Therefore, four periods are a sprint.

Completed Activities:

- **Project Initialization (16/10/2020 – 30/10/2020)**
  - Design Wiki UI (1 period)
  - Implement REST API (4 periods)
  - Design SWN Character Generator (SCG) UI (2 periods)
  - Build placeholder wiki pages/tables (2 periods)
- **Build SCG UI (30/10/2020 – 18/12/2020)**
  - Attributes Panel (2 periods)
  - Backgrounds Panel (2 periods)
  - Skills Panel (2 periods)
  - Class Panel (1 period)
  - Foci Panel (2 periods)

I fell behind schedule temporarily as I was unable to work for about a week. However, as shown in the Gantt chart, some tasks have since been completed quicker than planned so the project is still on track. The completed tasks can be seen in dark purple, where a full

orange represents time the task was worked on, outside of the plan. Hatched purple represents planned time for a task.

Planned Activities:

- **Build SCG UI, continued (30/10/2020 – 18//12/2020)**
    - Psychic Powers Panel (3 periods)
    - Navigation (1 period)
- **Implement SCG Functionality (18/12/2020 – 05/02/2020)**
    - Attributes Panel (1 period)
    - Background Panel (2 periods)
    - Skills Panel (2 periods)
    - Class Panel (1 period)
    - Foci Panel (1 period)
    - Psychic Powers Panel (2 periods)
    - Suggestion Algorithm (2 periods)
    - Saving and Exporting (3 periods)
- **Final UI Pass (05/02/2020 – 05/03/2020)**
    - Complete wiki by using object avatars and table libraries to give a nice representation of objects (2 periods)
    - Add filtering and sorting to wiki tables (2 periods)
    - Make physical skeet imitation/final navigation UI (4 periods)
- **Add User Accounts (optional) (05/03/2020 – 02/04/2020)**
    - Users can create accounts and login (2 periods)
    - Users can save characters to the database and load them (4 periods)
    - Users can save custom game objects in the database and use them in character creation (2 periods)

The focus is to have a functioning UI first, then add full functionality, before going back and finishing the UI. User accounts is a stretch goal. Its allocated time may be needed for remediation or simply writing the dissertation.

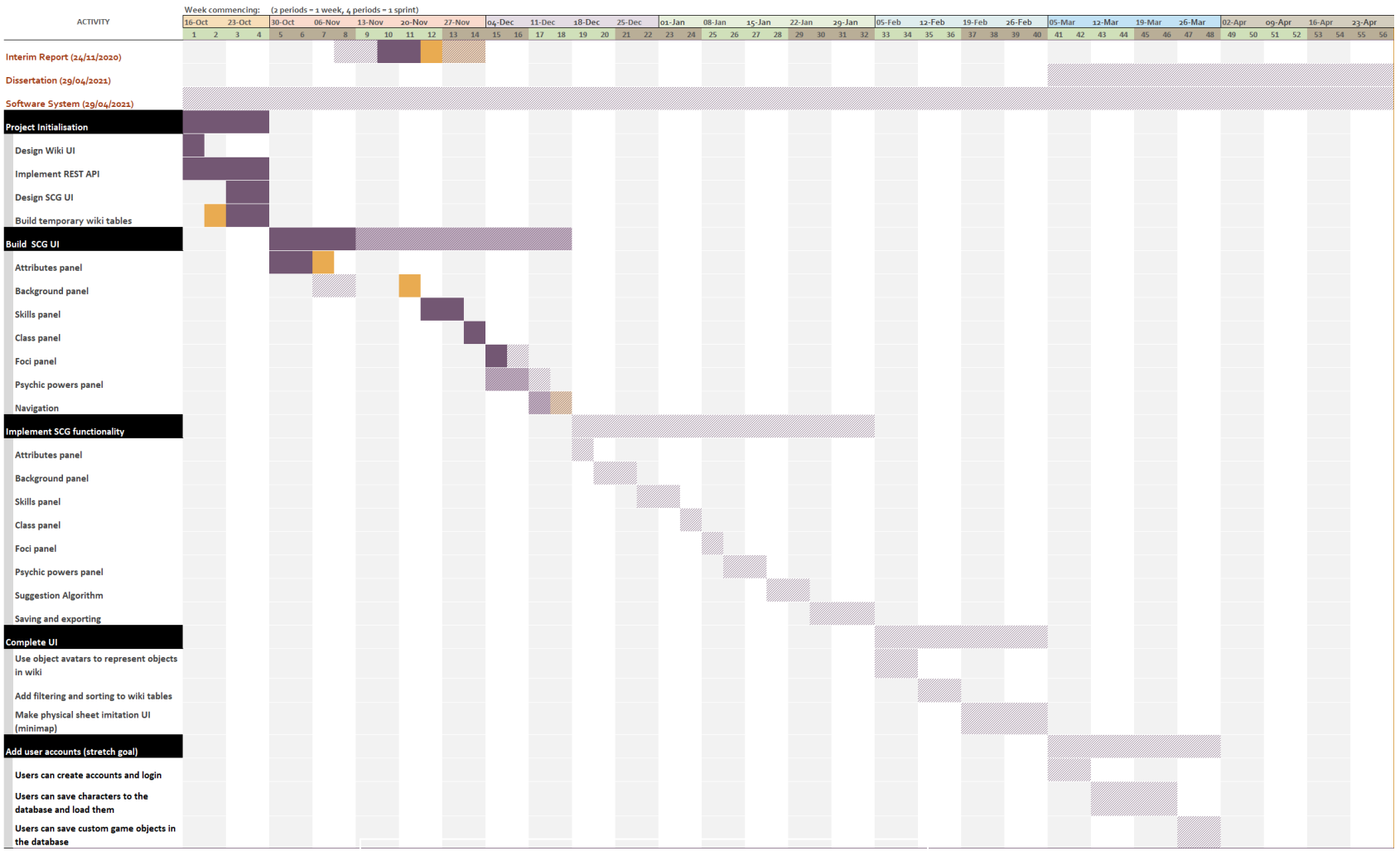

Figure 6: Project plan Gantt chart (available at /documentation/Interim Report)

Glossary

Attribute – One of six quantified physical characteristics in SWN. They are strength, dexterity, constitution, intelligence, wisdom, charisma. They give modifiers to dice rolls for related actions.

Docker Container – A lightweight virtual machine running a pre-built image.

Focus – From the rulebook [11] "A focus is an additional knack, perk, or aptitude that a hero has, one that grants them certain benefits in play". Generally foci include the learning of 1 skill plus two more complicated ability

React.js – An open source Javascript library developed by Facebook for UI and front-end development.

Skill – Represents proficiency with some class of action in SWN. Some examples include pilot, shoot, trade

SWN – Stars Without Number. A science-fiction table-top roleplaying game in which players pilot characters of a certain class and background, with skills and other attributes to be used in combat and non-combat situations. The aim is simply to create a story by following loose game rules and improvising with the help of a game master, who controls the world.

TypeScript – A typed language that compiles to JavaScript. Allows the compiler to enforce that variables be of certain types allowing much richer debugging and more safety than compared to JavaScript.

References

[1] React.js Docuemntation (by Facebook) - https://reactjs.org/docs/getting-started.html (last accessed 02/12/2020)

1. Advanced Guide: Error Boundaries - https://reactjs.org/docs/error-boundaries.html
2. Hooks - https://reactjs.org/docs/hooks-intro.html
3. React Top-Level API - https://reactjs.org/docs/react-api.html

[2] Sequelize Documentation - https://sequelize.org/v5/index.html (last accessed 02/12/2020)

[3] I. Ndaw, How to Build a Todo App with React, Typescript, NodeJS, and MongoDB (tutorial) - https://www.freecodecamp.org/news/how-to-build-a-todo-app-with-react-typescript-nodejs-and-mongodb/ (last accessed 02/12/2020)

[4] B. Bachina, Dockerizing React App with NodeJS Backend (tutorial) - https://medium.com/bb-tutorials-and-thoughts/dockerizing-react-app-with-nodejs-backend-26352561b0b7 (last accessed 02/12/2020)

[5] B. Cameron, How JavaScript Maps Can Make Your Code Faster (blog) - https://medium.com/@bretcameron/how-javascript-maps-can-make-your-code-faster-90f56bf61d9d (last accessed 02/12/2020)

[6] C. Nnamdi, Top 7 UI libraries and kits for React (blog) - https://blog.logrocket.com/top-7-ui-libraries-and-kits-for-react/ (last accessed 02/12/2020)

[7] Reroll, D&D 5E Character Generation App - https://app.reroll.co/ (last accessed 02/12/2020)

[8] Ninetail D&D 5E Character Builder - https://ninetail.org/charBuilder.html (last accessed 02/12/2020)

[9] u/Mromson, Automated Character Sheet for Stars Without Number - https://docs.google.com/spreadsheets/d/1Ni8nc2ymt7DwF5fU8dCurSvma56n96gI4GOyIdH6DIo/edit#gid=2015211282 (last accessed 02/12/2020. Also available at "/resources/Stars Without Number - Character Sheet v2.0984_LIVE")

[10] u/WiredIn1, Revised Edition Character Sheet - https://docs.google.com/spreadsheets/d/1D_zjdbhnfqB74MgV3uwU140GZiwBsSd7P40bNxlGJNI/edit?fbclid=IwAR2pMEsVXuP9APQflgxCiMJTW7cfuSP1kpD_QC5Gs3gFPD7WW0JscLWBW8w#gid=584628805 (last accessed 02/12/2020. Also available at "/resources/Character Sheet V2.7")

[11] K. Crawford, Stars Without Number Revised Edition Rules v2 – (paid content, can be found in /resources/Stars_Without_Number_2E.pdf)