|  |
| --- |
| CO3201 |
| Title |
| Computer Science Project |

|  |
| --- |
| Dominic Cousins  2021 |

Table of Contents

[Abstract 3](#_Toc78230151)

[1. Introduction 3](#_Toc78230152)

[1.1 Motivation 3](#_Toc78230153)

[1.2 Aims and Objectives 3](#_Toc78230154)

[1.3 Challenges 4](#_Toc78230155)

[1.4 Results 4](#_Toc78230156)

[2. Literature Review 4](#_Toc78230157)

[2.1 Similar programs 4](#_Toc78230158)

[2.2 Technical Tools 5](#_Toc78230159)

[2.3 Character Creation Procedure 5](#_Toc78230160)

[3. Requirements 6](#_Toc78230161)

[3.1 Functional Requirements 6](#_Toc78230162)

[3.2 Non-functional Requirements 7](#_Toc78230163)

[4. Design and Implementation 7](#_Toc78230164)

[4.1 Architecture 7](#_Toc78230165)

[4.2 Interface Design 7](#_Toc78230166)

[4.3 Implementation 7](#_Toc78230167)

[5. Testing and Evaluation 7](#_Toc78230168)

[5.1 Testing 7](#_Toc78230169)

[5.2 Evaluation 7](#_Toc78230170)

[6. Results and Discussion 8](#_Toc78230171)

[6.1 Source Code Analysis 8](#_Toc78230172)

[6.2 Results 8](#_Toc78230173)

[7. Critical Appraisal 8](#_Toc78230174)

[7.1 Successes, Failures and Choices 8](#_Toc78230175)

[7.2 Future of the Project 8](#_Toc78230176)

[7.3 Personal Development 9](#_Toc78230177)

[8. Conclusion 9](#_Toc78230178)

[References 9](#_Toc78230179)

[Appendices 10](#_Toc78230180)

[Glossary 10](#_Toc78230181)

[Code Extracts 11](#_Toc78230182)

# Abstract

Summarise report (state problem, background info and results). maybe write this last

# 1. Introduction

## 1.1 Motivation

Stars Without Number (SWN) is a table-top role-playing game in which players improvise according to some rules and guided by a GM (game master), who acts like a narrator and controls the game world. The aim is to: explore a world created by the GM with the help of the rulebook; progress through a story crafted by the GM and influenced by the players; and improve one’s character with items, money, and experience earned by playing. 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 and is a pre-requisite to playing the game. Meaning, players with no experience of the game at all may be required to generate a character, even though it is their first exposure to SWN. As a result, making a character can be tedious or difficult, and could turn these newer players away from the game. Also, from personal experience, there is nothing worse than losing your physical character sheet and having to start the whole process again with a new character, which can be jarring in a game about storytelling, immersion, and continuity.

Gaming has become increasingly popular, with digital games now being seen as mainstream. The ongoing Covid-19 pandemic has likely pushed their popularity higher. Lockdowns, isolation rules, and other restrictions, mean people are less able to enjoy hobbies outside their homes, with large groups, or at packed venues. This has caused many to turn to gaming as their form of entertainment. The largest gains have been seen in video gaming, with Statista [13] reporting a “global increase in time spent video gaming during the covid-19 pandemic [of] 39%”. This expansion has also occurred in tabletop games, with the “global games and puzzle market [reaching] a value of $11bn in 2020”, according to a recent Dicebreaker report [12]. This is an increase of almost $1bn (or 10%) compared to the 2019 market value. A similar increase is expected to occur in 2021. People can also play traditional tabletop games in video game form by using such tools/games as Tabletop Simulator [14]. When playing tabletop roleplaying games like SWN, the conventional pen-and-paper character sheets feel nice to hold, but when the game moves online, they aren’t visible to the other players and, more importantly, the GM. Additionally, when generating a character, the GM is typically present to help players, providing information about their world, or clarification on game rules and procedures. With the move to online game sessions, this is much more difficult to organise. The GMs role in this could be replaced or supplemented by a tool that enforces the rules, guides players, and allows customisation. And, having the character sheet able to be stored digitally makes it much easier to share with others for their viewing, or even editing.

As revealed in the review of literature, there is no such tool currently available for the SWN tabletop game. There do exist spreadsheets for storing the character sheets digitally, however they require extensive knowledge of the game to properly fill in and use. Also, their functionality can only be customised by literally changing the spreadsheet which requires knowledge of spreadsheets and becomes increasingly difficult the more the spreadsheet tries to automate any process. This provides an excellent opportunity for me to build on knowledge of web development I recently acquired from working in industry by providing a web-based tool to aid character creation and storage. One task I had while working in industry was to develop webapps that extend the functionality of another webapp, Jira Cloud, which sees wide use in industry as a project manager and task tracker. They would be embedded inside Jira and connect to a server to perform functionality. I was keen to put my knowledge of current industry tools, such as javascript, react.js and MySQL, to use in a more standalone situation, thus expanding on what I had learnt.

## 1.2 Aims and Objectives

The project purpose was to create and host a webapp that automates the character creation process. The primary aim of the system was to increase the speed and ease of character creation for SWN. A secondary aim was to facilitate the storage of player characters, custom rules, and custom objects such as weapons and skills. Additionally, these custom objects were to be functionally supported by the webapp (i.e. they could be used to create characters). As SWN is a game of improvisation, a lot of players, and especially GMs, like to customize the game and rules. In fact, this was a primary concern when speaking with players about the proposed tool. Since they customise game elements, it would be cumbersome or even impossible to use the tool without sacrificing these customisations. Therefore, this secondary aim ensures they can still use their own rules while benefiting from the enhanced character creation experience offered by the tool.

My objectives included 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 their needs, I aimed to develop a way for users to enter custom game objects into the tool, as well as customize the rules. Additionally, an algorithm was planned which could propose some of the most suitable choices at each step to aid newer players or really speed up the process for experienced users.

The objectives were as follows:

* 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
* Make associated TypeScript models to represent the objects once fetched from the database
* Store all game objects described in the given game rules in the database for use in the app – players should have access to any entity described in the base rulebook
* Host a Node.js web server that connects to the database and services API requests for data from it. The server will also distribute the JavaScript client
* Write React.js-based webapp to allow viewing of stored objects and the interactive creation of characters following the game rules
* Develop JSON formats for storing ruleset customizations and support them in the tool
* Research user interface design and develop an intuitive interface that is pleasing to look at
* Create an algorithm to suggest game object picks and help guide players. This could be utilized in a semi-random generation mode to create random, but balanced characters

May need additional objectives or explanation, simply from interim report

## 1.3 Challenges

### 1.3.1 REST API

A hurdle encountered quite early in the project was how to source all the data the app would need to run. The client needed access to a complete list of every piece of equipment, every consumable item, every skill etc. so players can create their characters with those objects. As well as just a list of available objects, it also needed text descriptions of them, along with all their in-game statistics. This potentially huge amount of data can’t be sent with the client when the webapp is visited – loading times would increase massively, and continue to rise as more objects are added, or their statistics expanded. From my experience in industry, I learnt that a good way to supply this information to the client is via a RESTful API, or REST API. These are used extensively in real applications and provide a language for clients and servers to communicate with over the internet. In an article about REST API [16], RedHat defines what makes an API RESTful. It includes stateless communication “meaning no client information is stored between get requests and each request is separate and unconnected”. Additionally, data should be cacheable (http requests are cached by browsers) to speed up requests, the server setup should be invisible to the client (they simply request what they want and get it), and follows a standard form of data transfer, as defined by the API. From previous work I know publicly available REST API exist for game data, such as the PokeAPI [15], which serves information about all Pokemon creatures and items in all Pokemon games. Optimistically I searched for a SWN REST API that could provide my client with game object information, however SWN is much less mature than Pokemon and lacks such an API hosted by either the creators or the wider community. Therefore, I would need to design such an API, as well as source the information myself from somewhere else.

### 1.3.2 Data Collection

Without access to a pre-existing REST API, I also needed to gather all the data about game objects and store it somewhere accessible to the server program, such as a database. The primary source of information about the game, the core rulebook [11], is available as a pdf and includes all the game objects and their statistics. For example, on page 67 is the complete list of ranged weapons in the game. However, taking the data from the rulebook and putting it in the database would be laborious, manual work which could have taken days. Luckily in my review of literature I discovered several excel spreadsheets aiming to imitate the paper character sheets used in SWN. Some hoped to include some automation, such as drop-down lists of available equipment, though this functionality doesn’t seem to work. As a result, they included extra sheets containing details of all such objects. These were easily exported to CSV, resulting in data sources more readable by a program. Processed versions of those data sources can be found in “/resources/data sources” within the repository. Below is an extract from the Weapons.csv data source:

subtype,skill\_id,name,source,page,tech\_level,encumbrance,cost,attribute,damage,range\_low,range\_high,magazine,shock

Primitive,14,Advanced Bow,Core,35,3,1,50,DEX,1d6,100,150,-,-

Gunnery,12,Automatic Matter Forge,W2,2,5,3,,DEX,3d6 #,500,2000,20,

Proj./Ener.,12,Biocatalytic Induction Gun,W2,2,5,2,,DEX,2d8+2,50,100,6,-

Primitive,14,Blackout Stick,Espionage,93,5,1,5000,STR/DEX,Special,,,1,-

Primitive,14,Chain,MA01,6,1,1,8,DEX,1d6,3,,-,2/AC 13

Primitive,14,Chronal Burner,MA04,7,6,1,,DEX,Special,,,2,

Primitive,14,Club,Core,35,0,1,,STR,1d4,,,-,1/AC 15

The top line describes the columns present in later lines, and was edited manually to correspond with columns for this object class in the database.

What was difficult?

## 1.4 Results

What is the resulting system like at a high level?

Screenshots

# 2. Literature Review

## 2.1 Similar programs

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 data since the tables of game objects they use can be exported to CSV and handled programmatically. 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.

## 2.2 Technical Tools

can introduce tools you are relying on such as React and why you are using it for this purpose, Choice of Database and why? And even why you have chosen the different programming languages you mentioned for this project (Typescript, Java script, SCSS)

## 2.3 Character Creation Procedure

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. It also includes lists of all available objects in each category and their descriptions and/or statistics. However, all content after page 93 will not be needed for this tool specifically - it pertains to areas of the game beyond character creation. On page 4 of the rulebook [11] we have the “summary of character creation”. According to the rulebook there are 18 steps in character creation, however some are overly verbose.

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]
* 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 and apply the gained bonuses
* 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
* Calculate attribute modifiers according to the following conversion: a score   
  of 3 is a -2, 4-7 is a -1, 8-13 is 0, 14-17 is +1, and 18 is +2.
* 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, there are a lot of steps here, even when I have removed or combined some steps as appropriate. 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. An optimised version of this order is available by the tool; however, players can also skip forwards and backwards as desired and revisit steps to make changes. The game rules enforced by the SWN character creator differ slightly from those given in the rules, though not in any significant way. These changes were made to improve the user experience, make the interface easier to navigate, and better enable the use of custom rules. The character creation flow in the tool is as follows: Include flowchart and/or bullet points of actual rules in the resulting system

Highlight complexity of the rules and use of “house rules” and the consequences for the implementation.

# 3. Requirements

## 3.1 Functional Requirements

Separate requirements from interim report into (non) functional and add more

## 3.2 Non-functional Requirements

Separate requirements from interim report into (non) functional and add more

# 4. Design and Implementation

## 4.1 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.

Explain how I store the data – server and users computer (client). Client requests site from server and then data when needed, will explain architecture around this.

## 4.2 Interface Design

User interface design process - include UI sketches I made, outline background research into UI (where users expect buttons to be etc)

## 4.3 Implementation

How I generate the output – form fillable PDF edited with adobe acrobat, library for Java script which lets me edit and fill in the fields on the sheet

Explain how I control the components of the app – high order component (HOC) to contain logic and control stateless components.

# 5. Testing and Evaluation

## 5.1 Testing

how did you test your software? I suppose manually but you have to show case key test cases and they were successful. I do not think you have automated testing but that is fine.

## 5.2 Evaluation

In terms of evaluation, you can compare the functionalities of your software and existing ones if you do not have feedback from potential users of your game.

Test outputs – PDF output, show some choices I made and the PDF it created, screenshots of the system after I have interacted with it

# 6. Results and Discussion

## 6.1 Source Code Analysis

explain how it works and why it was successful, consider elegance, efficiency and complexity of the code I developed (include excerpts from source code)

## 6.2 Results

Analyse success verse aims and objectives

Discussion of developed system, did I achieve what I set out to achieve? Yes but cut some optional features which could be developed in the future (outline what I cut for this project due to time constraints)

• exporting to Excel as well as printing

• work on algorithm to suggest choices user can make

User reviews – outline what user thinks of the app during development (explaining how I acted on feedback) and final product

# 7. Critical Appraisal

## 7.1 Successes, Failures and Choices

Summary and critical analysis of work completed

time scales in your work plan. What went right and what can be improved.

Reflect about the choices you have made in terms of software tools. If you were to redo it, will you use the same tools again?

## 7.2 Future of the Project

Further areas to develop:

* expand custom rules,
* users to be able to register for the site so characters can be hosted on a server,
* dark mode and accessibility considerations

Discussion of commercial and economic context – didn’t develop it to make money but could do so if I wanted via ads or donations

Sustainability of the project - Open source the project for others to contribute to or host a free server (optimise project to run on lower specs)

## 7.3 Personal Development

My personal development during the project – increased confidence in using the software, improve coding skills, practiced using Java script, improved UI design which was previously a challenge for me (link to usefulness in future career)

# 8. Conclusion

Did I achieve what I wanted to achieve? Summarise project and link back to aims and objectives in introduction

# 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/1Ni8nc2ymt7DwF5fU8dCurSvma56n96gI4GOyIdH6DI0/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)

[12] Matt Jarvis, Free time during lockdown helped board game sales jump in 2020 – report - <https://www.dicebreaker.com/categories/board-game/news/games-and-puzzles-market-2020-2021> (last accessed 27/07/2021)

[13] J. Clement, COVID-19 impact on the gaming industry worldwide - statistics & facts - <https://www.statista.com/topics/8016/covid-19-impact-on-the-gaming-industry-worldwide> (last accessed 27/07/2021)

[14] Tabletop Simulator - <https://www.tabletopsimulator.com/> (last accessed 27/07/2021)

[15] PokeAPI - <https://pokeapi.co/> (last accessed 27/07/2021)

[16] RedHat, What is a REST API? - <https://www.redhat.com/en/topics/api/what-is-a-rest-api> (last accessed 27/07/2021)

# Appendices

## Glossary

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.

GM – Game Master. A special class of player. One is required per game session. Instead of controlling their own character, the GM controls the world, including all non-player characters. Usually, they have designed the world, though pre-made campaigns (worlds) exist. They narrate and referee the whole experience for the other players.

Attribute – One of six quantified innate characteristics in SWN. They “reflect the basic potential of [the character]” [11]. Separated into two categories, the physical attributes are: strength, dexterity, constitution; the mental attributes are: intelligence, wisdom, charisma. They give modifiers to dice rolls for related actions, for example shooting a gun (dexterity).

Skill – Represents proficiency with some class of action in SWN. Skills apply their level as a modifier to their class of action. The absence of a skill results in a -1 modifier. Some examples include pilot, shoot, trade.

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 abilities.

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

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

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.

## Code Extracts

Any large code snapshots can go here