
The Adventurer's Inn

Colin Roche - G00349215, Antaine O'Conghaile - G00347577

B.Sc. (Hons) in Software Development

Final Year Project

Advised by: Mr. Daniel Cregg

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)



Contents

1	Abstract	4
2	Introduction	5
2.1	Summary	5
2.1.1	Methodology	5
2.1.2	Tech Review	5
2.1.3	System Design	5
2.1.4	System Evaluation	5
2.1.5	Conclusion	6
2.2	Web Applications	6
2.3	Goals and Objectives	6
2.3.1	Technical Objectives	6
2.4	Work Allocation and Planning	7
2.4.1	Work Plan	7
2.5	Project Objectives	9
3	Methodology	10
3.1	Initial Plan	10
3.1.1	Brainstorming	10
3.1.2	Research	10
3.2	Meetings	11
3.2.1	Project Management Methodologies	11
3.2.2	Discussion Platforms	12
3.3	Development Tools	13
3.3.1	IDE	13
3.3.2	Console	13
3.3.3	Source Control	13
3.3.4	Testing	13
3.3.5	Documentation	13
4	Technology Review	15
4.1	Project Management	15
4.1.1	Agile	15
4.1.2	Scrum	15

4.2	Tools for Project Management	15
4.2.1	Github	15
4.2.2	Microsoft Teams	16
4.2.3	Discord	16
4.3	Application Development Tools	16
4.3.1	Visual Studio Code	16
4.4	Front End	16
4.4.1	Angular JS	16
4.5	Back End	17
4.5.1	Database	17
4.5.2	Cloud	18
4.5.3	SDK's and Libraries	20
4.5.4	Containers	20
4.5.5	Docker	20
4.5.6	Typescript	21
4.5.7	Cmder	21
4.5.8	Testing Tools	21
4.5.9	Code Examples	22
5	System Design	23
5.1	Web Application	23
5.2	Database	23
5.2.1	Cloud Firestore	24
5.2.2	Database Model: JotunenHoard	24
5.2.3	Authentication	27
5.2.4	Sign Up Page	28
5.2.5	Sign In Page	28
5.2.6	Messages Page	29
5.2.7	Forums Page	29
5.2.8	HomePage	31
5.2.9	App.Module	31
5.2.10	Groups Page	32
5.2.11	My Profile Page	32
5.2.12	Create Character	32
5.3	Container	33
5.3.1	Docker	33
5.4	Cloud Storage	33
5.4.1	Azure Virtual Machine	34
6	System Evaluation	35
6.1	Initial Plan	35
6.1.1	Goal of the Application	35
6.2	Evaluation of Objectives	36
6.2.1	Evaluate and establish the requirements of the user with regard to messaging.	36

6.2.2	Evaluate and establish the requirements of the user with regard to sharing of campaign overview information. . . .	36
6.2.3	Evaluate the current tools available for character creation and establish the relevant information that is needed. . .	37
6.2.4	Limitation of the application.	38
6.2.5	Authentication and Security	38
6.2.6	User Experience	39
6.2.7	Evaluation of the Technologies Used	39
6.3	Plan vs Reality	41
6.3.1	Location	41
6.3.2	Rating System	41
6.4	Opportunities for Improvement	42
6.4.1	Categorise Posts	42
6.4.2	Filtering and Searching of Data	42
6.5	Testing	42
7	Conclusion	43
A		46

Chapter 1

Abstract

Tabletop Roleplaying games(TTRPGS) have become increasingly popular in recent years, with shows based around them such as 'Critical Role' raising over 11 million dollars in 45 days in order to create a animated series based on their games. Though popular there is a steep learning curve that requires a decent understanding of the rules which span multiple books and hundreds of pages.

For this project the goal was to create a social media site that allow for people to share their thoughts and ideas on a forum focused on the topic. TTRPG's such as Dungeons and Dragons have systems built in to allow the players to create their own content with an almost infinite amount of freedom.

Because of the contradictory nature of this we wanted to make a website that would cater to the free discussion of ideas, while aiding the user in the more complicated process such as making a character which may seem daunting at first with the size of the reading materials.

As sessions of these games can often last months or years we wanted to create a database to store information about the sessions on one platform where the history of a group can easily be tracked and not lost, as is often the case with sheets of paper.

Our final goal with this project was to create a platform that would allow a person to find a group to play with as despite being popular the topic of TTRPS is not often breached in a standard conversation. Thus this website would allow people to find other players in their area or virtually while aiding in the management and creative process.

Chapter 2

Introduction

2.1 Summary

This section is comprised of brief descriptions of each chapter present in this dissertation.

2.1.1 Methodology

This section is an overview of the various methodologies we used in the development of this project. This includes our Research, Development and Testing methodologies as well as how they were implemented throughout the development life-cycle. It also gives a brief overview of certain technologies that were used to match these methodologies.

2.1.2 Tech Review

This chapter contains a description of the various technologies and tools we used in the development of this project. These include tools for Project Management, Application Development and Front end, Cloud and Server side tools, Containerize Software, Testing and the various Software Development Kit's and libraries used.

2.1.3 System Design

This chapter discusses the components and their technologies in greater detail. As well as how they are implemented into the System as a whole and the design decisions made during the development.

2.1.4 System Evaluation

This chapter is an evaluation of the initial project objectives and compares them to the final implementation of the project. We also reflect over the project as a

whole, including the limitations and what can be improved.

2.1.5 Conclusion

A brief overview of the implemented project and reflection of the development process as well as our thoughts and what we learned.

2.2 Web Applications

A Web Application is an application that is invoked in a web browser while accessing the Internet. Web Applications come in a wide variety of types from small-scale application that are short lived to long-term and large-scale applications that are distributed across multiple servers. Applications that utilize a HTML front-end have the benefit of universal cross-platform access. This eliminates compatibility issues as all users have access to the same version. [15] HTTP protocols are used to fetch these HTML document resources. Other languages such as Javascript and CSS can be used to provide further implementation and styling options that are lacking in a HTML.

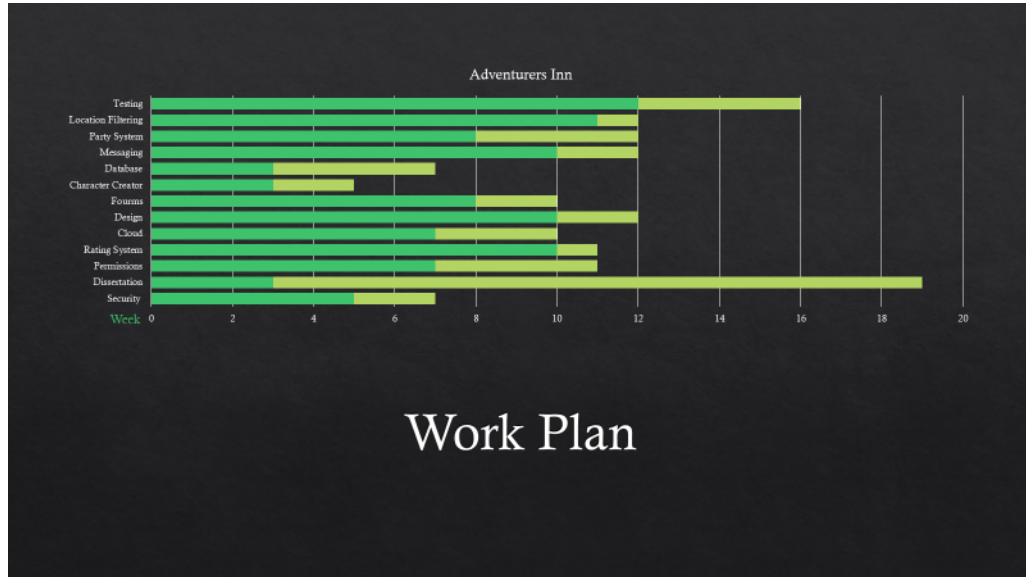
2.3 Goals and Objectives

Our main goal was to develop a web application hosted on a remote virtual machine that would allow the users to share ideas, information and help organize and manage their games.

2.3.1 Technical Objectives

- Store Character and User information on a secure server
- Allow users to Post and Reply to a Forum
- A Messaging system and Friends list between users
- Create private groups between users
- Help users find a group or individual
- Create Characters

2.4 Work Allocation and Planning



We have recorded our work incrementally in the journal page on Github the work we individually did, including what was not found in commits such as changes to the Firebase Cloud Firestore, where rules can be altered for the server. As well we have documented problems we faced and the rationale behind the decisions we made as well as the rationale behind them. Included in this is our original plan from the first semester.

2.4.1 Work Plan

Front End

- Initial research on Angular: Antaine and Colin
- Basic Create Character: Antaine
- Advanced Create Character: Colin
- Bootstrap and Styling of Application: Colin
- Forums: Antaine
- Navigation: Colin
- Messaging: Antaine
- Add Username: Colin
- Profile: Antaine and Colin

Back End

- Initial Research on Databases(NOSQL vs SQL): Antaine
- Firebase Database Creation: Antaine
- Create, Read, Update and Delete functionality: Antaine
- Database Implementation: Antaine
- Angular Routing: Antaine
- Login and Registration: Colin
- Security and Authentication: Antaine
- Docker: Colin
- Database Design: Antaine and Colin

Cloud and Hosting

- Initial research on Cloud Services: Colin
- Azure Cloud Implementation: Colin
- Connected Virtual Machine to Database: Colin

Dissertation

- Abstract: Antaine
- Introduction: Antaine and Colin
- Methodology: Antaine
- Technology Review: Colin
- System Design: Antaine and Colin
- System Evaluation: Antaine and Colin
- Conclusion: Colin

2.5 Project Objectives

The primary goal of this project was to create a web application to be used as tool to enhance a users experience with planning, managing and playing Table-top Role Playing Games (TTRPGS), with a focus on Dungeons and Dragons 5th edition. The user should be able to share their thoughts and ideas, questions and much more on a forum. It will allow users to share information to a group of 'Players' and Dungeon Master/Game Master, as well as allowing them to quickly reference their character information which the web app would help the user make by reducing the amount of information required to make a character and the amount of paper resources required to manage. The app should also aid the user in finding a group by posting to the forums or for a group to search for a player and include the kind of sessions they intend to run.

Chapter 3

Methodology

This section will cover all the methodologies that were used in the development of this project, as well as the rational behind them. This section also covers how we handled meetings, project management and how we worked around not being able to meet in person during the first semester and working remotely. We did this by utilizing Methodologies, such as Agile, Scrum and Test Driven Development. It also covers the tools we used to manage our code, meetings and schedules.

3.1 Initial Plan

3.1.1 Brainstorming

The first step in our plan was a brain storming session were we discussed different ideas for the project and factored in the different technologies that we were interested in learning. We decided on a Web Application that would aid with Table Top Roleplaying Game's with a focus on Dungeons and Dragons, as organizing a session during travel restrictions proved to be difficult task to manage. We quickly realized there was an opportunity to familiarize ourselves with these new technologies including a NoSQL database in firebase and Microsoft Azure. Afterwards we drew up a basic storyboard of what we would ideally like the website to look like and what features we would wanted to implement into it's design., as well as how difficult a task these would be.

3.1.2 Research

We used Qualitative analysis to research our idea to see if there were any samples of this already on the market and found there was nothing that precisely matched our concept. This helped us to narrow down the scope of our project and focus our efforts on specific areas such as the social media aspect of the project, as there were websites that already had detailed 'Character Creators'

such as 'Orc Pub', but none where you could share ideas and information between users , while guiding you through some of the more technical aspects of the game.

Front End

After researching Angular, REACT and Ionic we decided on using Angular as we found a single page application would suit the project's needs as we wanted to try and compact all relevant information onto a small screen to allow the Web App be easily navigate and comprehended, with information relevant to the user. As well as this, the many libraries that Angular includes that makes it flexible when connecting with the other components of the project such as AngularFireStore which contains plugins to aid with Angular to Firebase development and Visual Studio Code having angular specific extensions.

Cloud Services

After deciding on what we wanted to do, we focused our research on what platforms we wanted to use for the application. We compared Cloud Platforms such as Microsoft's Azure and Amazon Web Services, to see what the individual strengths and weaknesses of these platforms were. We found to be equally suitable for the project. We chose Azure as we had used AWS before and were excited to try a different platform.

Databases

Next we researched databases such as MongoDB and Google Firebase and decided to use Firebase as we had not previously worked on a NoSQL database and the amount we would learn for designing and implementing it would be vast and invaluable. Another advantage of Firebase was Angular had libraries such as AngularFireStore that helped connect our Front end of the project with the database.

3.2 Meetings

3.2.1 Project Management Methodologies

Agile

Agile is a Iterative approach to Software Development that aims to shorten the development life cycle, by increasing productivity and to regularly reassess the project as meeting in person was not a viable option. We immediately saw that this approach was far more suited to the project then the traditional Waterfall Methodology, as the flexibility Agile provides to adapt to a problem was perfectly suited to our development with several new technologies. Agile sets short term goals that are set be completed in a finite period of time called

sprints. This included regular iterative testing throughout the project during these sprints.

Scrum

We decided to use SCRUM as our Agile Methodology of choice. We believed this suited our project as it emphasizes short term goal orientated development as well as a respectful amount of independent work. This suited our remote working schedule, while allowing us to regularly discuss our progress and reevaluate our progress and planning in our weekly Scrum meetings. It is also well suited for receiving feedback and quickly implementing it into our plan as we would only have to readjust our short term goals. This was ideal when working with new technologies as we weren't sure what aspects might be easier or tougher than our original expectations during our initial research and planning.

Test Driven Development

We decided to use Test Driven Development as we knew there would be a learning curve with aspects of the project such as building a fully functioning database with a new language. It was vital to test the code regularly as we went along. Especially due to that fact that if we made any mistakes it was vital to figure it out early and fix the error before other features of the project were to become reliant on something that would not be suited to what we needed it to do.

3.2.2 Discussion Platforms

Microsoft Teams

We had weekly meetings with our supervisor Daniel Cregg remotely on Microsoft Teams, where we would discuss our progress and send a digital report of what we had achieved and our goals for the next week. This was our main Scrum meeting where we would set our objectives for the upcoming sprint and discuss what we difficulties we found and were able to adjust our plan. Here we would received crucial feedback that would greatly aid our development and speed up our decision making with the insight we were given, especially with regards on how to document the project regularly throughout the development.

Discord

After our meetings with our supervisor we used Discord, a voice chat and messaging app to meet and discuss our project further in detail and plan our workflow and schedules for the week, as well as sharing some ideas and documentation we found.

3.3 Development Tools

3.3.1 IDE

Our Integrated Development Environment of choice was Visual Studio code as it is a relatively lightweight compared to other common IDEs such as Eclipse. Visual Studio was created by Microsoft and as such is easily integrated with Azure which is one of the reason we choose it. One of the main reasons we chose Visual Studio Code was because of its vast amount of extensions for various languages and quality of life features that aid in the development. The fact that the community can add small extensions for certain preferences is a big advantage.

3.3.2 Console

Cmder

Cmder is a software package that allows the user to use Unix commands on windows as well as customization including history of previous commands used which proved to be a handy tool when regularly committing to Github.

3.3.3 Source Control

Github and Git

Github is a hosting site used by coders which was first launched in 2008, that utilizes Git for it's version controls. It allows for easier collaboration on a single project as well as creating a backup of the project online. It allows the user to revert to a previous version or to create branches to work on certain features of the project before merging it back into the main branch of the project.

3.3.4 Testing

Manual Testing

We did regular manual testing to make sure all methods worked throughout the development, especially before committing to our repository. This was key early on to make sure the foundations of the project and the connections between the front end, back end and database worked as intended, as they would be needed to call on later on in development.

3.3.5 Documentation

Overleaf

We used overleaf to write the dissertation as it allowed to professionally present and customize our layout of a Latex document. Much like Github, the big advantage of overleaf is that it is stored online and compiles, which allows us to work simultaneously on the documentation in real time.

Journal

As we progressed through our project we recorded who worked on what feature in the Journal.md file on the github repository, as well as some insights into our rational for design decisions and bugs encountered along the way.

Chapter 4

Technology Review

4.1 Project Management

4.1.1 Agile

Agile development is a methodology comprised of several principles and methods for project management. Where other project management methodologies such as the waterfall methodology focuses on long term rigid development, Agile takes a short term iterative approach. This provides a level of flexibility suited to a Software Development cycle, due to the unpredictable nature of inevitable setbacks such errors during the software development cycle. This helps to increase production, reduce the scope and costs of a project [2] .

4.1.2 Scrum

A scrum is a flexible method the divides periods of work into sprints. Each sprint is a relatively short period often between 1 and 2 weeks, after which the team and stakeholders will discuss the project as a whole, reevaluate their progress from the previous sprint and plan the following one accordingly. This makes the Scrum Methodology ideally suited to work with Agile Development [19]

4.2 Tools for Project Management

4.2.1 Github



GitHub is a web-based collaborative platforms providing tools to ease distributed development [7]. Officially launched in 2008, it uses Git a decentralized version control system that hosts code and using a master-less peer-to-peer replication to allow relatively seamless collaboration between developers on the same project.

4.2.2 Microsoft Teams



Microsoft Teams is a chat-based collaboration platform, that allows for document sharing, online meeting other useful tools for communication. Teams allows for collaboration between members whether it be through the messaging platform or voice and video calls [1].

4.2.3 Discord



Discord is a free voice communications app used for voice chat, video and messaging between friends. Discord can be used for simple communication but has the facilities to allow for a more professional type of communication. These facilities include group and one-to-one private messaging in which members can text, voice and video chat. Discord provides facilities sharing of documentation, useful links and one's screen which allows for real time sharing of information proving to be invaluable towards the development of a project [9].

4.3 Application Development Tools

4.3.1 Visual Studio Code



Visual Studio Code is a lightweight source code editor created by Microsoft. It has many plugins that can easily be added called extensions that allow the user to customize and add shortcuts to their project based on what language they would like to use as well as allowing the creation of community made extensions. It has built in support for Azure, Github and IntelliSense [6].

4.4 Front End

4.4.1 Angular JS



Angular is a widely used web development framework that creates a single page client application utilizing JavaScript and the Node Package Manager. It supports the editing and managing of various languages such as HTML, CSS and TypeScript while removing some of the complexities by adding specific angular attributes that would not normally be in these languages as well as easily managing a projects imports through the NPM.

With regular updates still being made to date, it is a platform that is currently in use in a wide range of applications. An advantage of using a single-page client application was that they are commonly used and offer a wide range of possibilities and flexibility with regards to web and agile development. With these

characteristic, Angular is a platform that will mostly likely still be relevant and used for the foreseeable future and thus a valuable asset worth understanding.

4.5 Back End

4.5.1 Database



A database, in its most basic concept is a collection of information that has been organized in a way, so that it can be accessed, read, managed, updated and stored. The database used in this project stores a users information by creating a document with the users information entered at registration and adding that document to the 'Users' collection at the root of the database. The database is configured in a way that only gives certain permissions to users who are logged in and only allowing the user to manipulate their own data.

Relational Database Management Systems (RDMS) is the most common way of storing structure data for web applications. The data-stores in this type of database requiring SQL, a language that provides consistent set of facilities for querying, manipulating and controlling data [20]. For use in this project we decided on using a NoSQL database instead as it was something unfamiliar to us.

NoSQL

In more recent years the idea of a "one size fits all" with relation to data-stores has become more popular. This has lead to the creation of alternative types of databases. These are commonly referred to as NoSQL databases which allow for a more flexible way of storing and retrieving data then the reliable and rigid relational SQL, allowing for a wider variety of ways for structuring data which can be more easily reworked during development. The ability to remove the complexity of a relational database and store/process data as it appears is one of NoSQL's greatest strengths in comparison to a SQL database.

The use of NoSQL can avoid unnecessary complexity and strict data consistency for applications where this might not be necessary. NoSQL databases also allows for high throughput and horizontal scalability which is necessary when dealing with large amounts of data [20]. Though because of this freedom a disorganized database can be created if the developer is not careful.

Firestore

The Database we used for this project is Firebase's Cloud Firestore database. This was used for storing user information, such as messages, forum posts,

friends lists and user created characters. It was selected during our research because of its NoSQL schema and its relatively low maintenance costs for a service of its quality being made by Google. Firebase is the collection of Google's many services on the cloud and can be used to enhance web applications and utilize other features such as Firebase storing and Firebase Hosting.

Firestore provides methods that can be implemented into your project such as Authentication, file storage, hosting and real-time databases. A big advantage of Firebase was libraries such as AngularFireStore which provides features and methods to an angular application making them very complimentary to each other. Firebase met the projects needs as we would not have too much traffic and being limited to a certain amount did not impact the project negatively. [13]

Cloud Firestore

Cloud Firestore is an NoSQL type database that is hosted on Google Cloud.

Backend as a Service (BaaS)

BaaS is a cloud computing service model that allows developers a way of connecting web application to the cloud service using application programming languages (API) and software development kits (SDK's).

4.5.2 Cloud



Many company's today provide a cooperative server-side on demand service for running online applications. The rational behind this is that large companies have numerous large storage facilities,with often more servers then what they intend to use. By renting out some of this servers to smaller scale users, whether it be for personal of business use, allows them to reduce some of their expenses. This allows smaller users to host applications to a world wide userbase around the world without having to deal with the expense and time costs of setting up personal servers.

The provider for the cloud computing platform used in this project is Microsoft Azure. The use of this platform in this project is to host a virtual machine in which the project is deployed. There are several reasons behind this, first is that the users of the application would not be as badly affected by latency due to the location of the local device. Secondly if the application was stored locally, if

the device that hosts the application fails then the application would be offline. By having this on professionally maintained servers the likelihood and impact of this is greatly reduced. Finally if the application was to be expanded then it would only need to expand the server usage on Azure.

The Cloud OS(Operating System)

For a lot of cloud-computing applications the entirety of the user interface resides in a single window on the browser or by using the cloud-computing paradigm, it has the ability to host the full facilities of an operating system on a browser. An alternative solution to this is to bypass the web browser on the local machine altogether by substituting it with a software system that runs an application on a client side computer that communicates directly with the servers on the cloud - Virtual Machine [17].

Azure

Azure Cloud Deployment is Microsoft's cloud platform which is a foundation for running applications and storing data on the cloud. Azure provides over 200 products to help build run and manage applications across the cloud with the developer being able to choose the tools and framework needed [18]. On Azure developers can use different types of roles, in our case we are using the Virtual Machine role running a Ubuntu Server image running a single desktop [5].

Linux

Linux is Operating System and is one of the most popular platforms on the planet, with it being a popular OS for computers and is the bases for Android. An Operating System is a software that manages the use of all the hardware resources associated with any device. The Linux OS is comprised of several different parts: bootloader, kernel, init system, daemons, graphics server, desktop environment and applications [14].


Ubuntu




ubuntu The OS we are using on Azure is Ubuntu, it is a complete Linux operating system that is freely available. Ubuntu is the world's most widely accessible and used Linux platform, with a Ubuntu Server being the reference OS for the OpenStack project, popular on cloud platforms such as Azure, AWS and Google Cloud [4].

4.5.3 SDK's and Libraries

AngularFire

 The AngularFire library is part of Firebase Open Source, a platform where user libraries are used and created by the firebase community. It contains predefined methods configured to make communicating with the server an easier process [3].

Node Package Manage

 The Node package Manager was used to install/import packages and imports into the Angular project. It uses Node.js and store the packages in package.json.

4.5.4 Containers



A container is a software used to package up all the code and dependencies of an application so that it can be reliably and easily moved from one computer environment to another. This can be done whether the new environment is similar to previous or completely different. Isolating the software from the environment, is done by the use of container image which become containers at runtime [11]. In this case the use of containers was used when changing from the test environment on our local computer to the production environment on the virtual machine.

4.5.5 Docker


The container software used in this project is Docker. Docker, like any containerizing software takes away repetitive configuration throughout the development of an application and allows for easy, portable application development whether it be on desktop or cloud platform [10].

A Docker container image is a lightweight, standalone executable that include everything that application requires to run: setting, system libraries, code, system tools and runtime. Images become containers when run by the docker engine, which is available on both Linux and Windows, by using the Windows Subsystem for Linux (WSL). Docker's containerized software will always run the same, regardless of the systems infrastructure.

An easy way to work with Docker containers is through the Docker Command Line Interface (CLI), which by use of commands simplifies how to manage the


container instances. There are three types of Docker containers that run on the Docker Engine: standard, lightweight and secure. Standard was implemented for the use of this project as it is the industry standard [11].

4.5.6 Typescript

 TypeScript is an open source language developed and maintained by Microsoft, that continues to build on JavaScript and adds optional static typing to the language. As TypeScript is compiled it is converted to the specified version of JavaScript, this results in existing JavaScript programs being valid TypeScript programs. It makes JavaScript more readable by assigning types to and definitions to variables and methods. This makes debugging a much easier process as often a problem with JavaScript is the lack of information when it cannot compile.

A big advantage of TypeScript is that native JavaScript code will also work as intended as the typescript interprets the code and converts it into JavaScript during the compile process. As JavaScript is widely used in web development as it is found in some form or another on a vast amount of websites. This means that TypeScript gives the flexibility of JavaScript while making it easier on the developer.

Node.js

 Node is a package manager that manages and installs packages onto your machine. It is widely known and is needed to install Angular using the Node Package Manager. Much like Git is required for Github, Node is required to use NPM. NPM stores its packages in JSON files such as package.json which Angular uses to manage a projects dependencies.


4.5.7 Cmdr



Cmdr is a package created as Command Line Interface that allows the user to use commands from both the Unix systems found in Linux and the console commands found in Windows. Having the same commands available of both Operating System is a big advantage as it saves time looking up the syntax.

4.5.8 Testing Tools

Selenium

 Selenium was used to test the feature of the web application.

4.5.9 Code Examples

Chapter 5

System Design

This system is comprised of three primary components, a CRUD web application for managing, manipulating and uploading models to the database. A database for storing the user model and their relevant messages and data. Lastly a virtual machine stored on the cloud for hosting the web application.

5.1 Web Application

This web application functions as a tool for managing a users the and create user model in the project through the registration page of the application.

The user is able to store and view their character information, they are also about to view a post to the forums page in the application enabling the to share and learn information about individual campaigns.

The web client is a single page web application present to the user which is built using the angular framework. The root page of the application presents the user with a navigation bar at the top to allow access to the other pages present in the application. This page also navigates the user to the sign-up page if there are not logged into the application or to the profile page if they are.

5.2 Database

In this project Firebase was used as database the store, access, manipulate and maintain the integrity of a users data. The reason why we elected to use Firebase for this project was for it's Cloud Firestore. As it has a NoSQL database, firestore is vital to the project as users would need instant access to their information and to use the application.

5.2.1 Cloud Firestore

Database Model

Cloud Firestore is the NoSQL database used to store the user models, messages, player made characters and posts. The web app uses 'AngularFireModule', 'AngularFireDatabaseModule', 'AngularFirestoreModule' from 'main.ts'. These Imports to allow the web app and server to communicate with the server. The project knows how to reach the project by the 'environment.ts' file which contains an object with the servers information. This is generated when the Cloud Firestore database is created, allowing for it to be added to the angular project with ease. The environment.ts is then imported into the main.ts where it can be accessed by the rest of the project.

Database Imports and Environment Snippet

```
import { AngularFireModule } from '@angular/fire';
import { AngularFireDatabaseModule } from '@angular/fire/database';
import { environment } from '../environments/environment';
import { AngularFirestoreModule } from '@angular/fire/firestore';

export const environment = {
  production: false,
  firebase:{
    apiKey: "AIzaSyCQhZRo-LBmzsD6zJ3z1JTHtqEGkGfKz0Y",
    authDomain: "jotunenhoard.firebaseio.com",
    databaseURL: "https://jotunenhoard.firebaseio.com",
    projectId: "jotunenhoard",
    storageBucket: "jotunenhoard.appspot.com",
    messagingSenderId: "274897005373",
    appId: "1:274897005373:web:b13f2b5e29d56cd4407354",
    measurementId: "G-6KQ90FG337"
  }
};
```

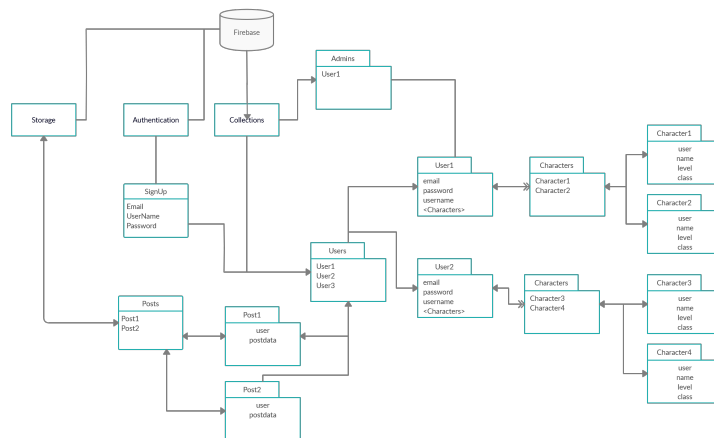
5.2.2 Database Model: JotunenHoard

JotunenHoard is the name of the Cloud Firestore database used in this project. It has three main Collections at the base of the project. These are 'Forums', 'Messaging' and 'Users'.

Database Model

<div> <div> <div>🏠</div> <div>users > kjiLk5t5QL8MLa1...</div> </div> <div> <div>jotunenhoard</div> <div>users</div> <div>kjiLk5t5QL8MLa1Y2aUwVA0EP6VC3</div> </div> </div>	<div> <div>+</div> <div>Start collection</div> </div> <div> <div>Forums</div> <div>Messaging</div> <div>Users</div> <div>users ></div> </div>	<div> <div>+</div> <div>Add document</div> </div> <div> <div>3IzOKf4Ze1SC07NE9I6D7F0K7XN2</div> <div>3uA0Mb5iR1bZmzGI0Jney0Xs0kZ2</div> <div>9hivgnm8TbeAPEG1gdx8W17M6z1</div> <div>Lx1D0vN1EXSH7rLGPfZeHT4qtgg2</div> <div>MMSodjvdNyf0e0w4qB6EQSRx3v12</div> <div>TQoA5n8Caw0VFw8fQbqb0ZP5gWz2</div> <div>hYZcvn0HeIbpSeUtdTaTeDa5Rv2</div> <div>kjiLk5t5QL8MLa1Y2aUwVA0EP6VC3 ></div> <div>m189Ja2P38PaTAKYeAN7K0iciyH3</div> <div>nU9Kk1wn3P98MNdq8BAww0qpAN2</div> <div>vvCXEssJmvS7uERg6f5b2KfTZz2</div> </div>	<div> <div>+</div> <div>Start collection</div> </div> <div> <div>Characters</div> <div>Friends</div> </div> <div> <div>+</div> <div>Add field</div> </div> <div> <div>displayName: "tim"</div> <div>email: "jess@g.com"</div> <div>emailVerified: false</div> <div>photoURL: null</div> <div>uid: "kjiLk5t5QL8MLa1Y2aUwVA0EP6VC3"</div> </div>
--	--	--	--

Database UML:



Collection: Users

The Users collection is a collection of User Documents. Every Document contains field variables that store relevant information about the user and are added when it is created by calling the 'register()' method in 'fire-authentication.service'. It uses a User model predefined by firebase to store a user's displayName, email and uid. Inside each document can be a Friends Collection and Characters Collection that are created once called from the project. These Collections Store documents of Characters the user has created and other users as well as the most recent message sent in a conversation.

User Model:

The screenshot displays the Firebase console's 'Users' collection. On the left, the 'firebase.User' model is detailed, showing properties like `displayName`, `email`, `emailVerified`, `isAnonymous`, `metadata`, `multiFactor`, `phoneNumber`, `photoURL`, `providerData`, `providerId`, `refreshToken`, `tenantId`, and `uid`. It also lists various methods such as `delete`, `getAccessToken`, `getAccessTokenResult`, `linkAndRetrieveDataWithCredential`, `linkWithCredential`, `linkWithPhoneNumber`, `linkWithPopup`, `linkWithRedirect`, `reauthenticateAndRetrieveDataWithCredential`, `reauthenticateWithCredential`, `reauthenticateWithPhoneNumber`, `reauthenticateWithPopup`, `reauthenticateWithRedirect`, `reload`, `sendEmailVerification`, `toJSON`, `unlink`, `updateEmail`, `updatePassword`, `updatePhoneNumber`, `updateProfile`, and `verifyBeforeUpdateEmail`.

On the right, a sample document is shown with the following fields:

```
displayName: "tim"
email: "jess@g.com"
emailVerified: false
photoURL: null
uid: "kijLkt5QL8MLa1Y2aUwVA0EP6VC3"
```

Below the model, the 'users' collection is visible in the sidebar. The main area shows a list of documents, with one document selected and its details displayed on the right. The selected document has the following fields:

```
background: "Criminal"
character: "Test"
class: "Fighter"
level: "Level 3"
name: "jess@g.com"
race: "Dragonborn"
```

Collection: Messages

The Messages collections contains a set of documents who's titles is a Unique ID generated by combining the unique ID of two users. In this document is the information of the latest message information saved and a collection of documents containing every individual Message sent in a conversation between two users. A message document contains the contents of the messages, the time it was sent, the IDs of the users involved and the chatId.

Messages Collection Model:

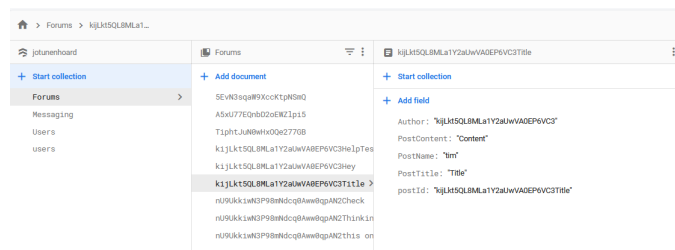
The screenshot displays the Firebase console's 'Messages' collection. The left sidebar shows the 'Messages' collection selected. The main area shows a list of documents, with one document selected and its details displayed on the right. The selected document has the following fields:

```
MessageContent: "Hi"
MessageId: "kijLkt5QL8MLa1Y2aUwVA0EP6VC3"
SenderId: "kijLkt5QL8MLa1Y2aUwVA0EP6VC3"
chatId: "kijLkt5QL8MLa1Y2aUwVA0EP6VC3"
createdAt: "2020-07-15T14:00:00.000Z"
sender: "kijLkt5QL8MLa1Y2aUwVA0EP6VC3"
```

Collection: Forums

The forums Collection contains a set of documents who's titles are uniquely generated by combining a users unique ID and the Title of the post. In these documents are the information and contents of the post, as well as a collection of Comments inside each which store all replies posted on a forum.

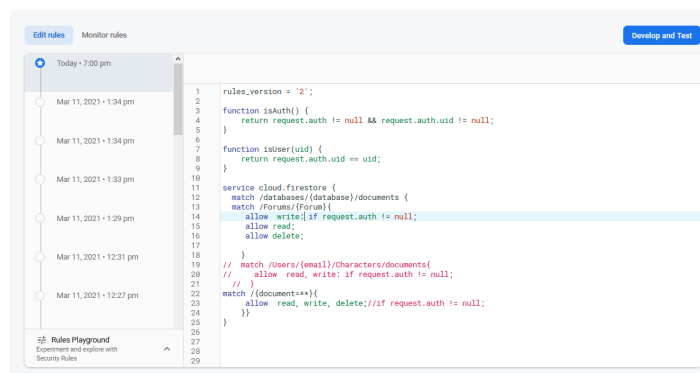
Forums Collection Model:



Firestore Rules:

Firestore has a rules sections where you can manipulate the rules of the server. The rules on this server prevents all users who are not authenticated, will only allow permissions to read documents regardless of any other rules. After the account is authenticated the user has permissions where available.

Firestore Rules Snippet:



5.2.3 Authentication

which use the Angular Authentication method `this.afAuth.createUserWithEmailAndPassword(email, password)`

5.2.4 Sign Up Page

This page prompts the user to register an account for the application. The user is prompted to fill in the Register User form, which includes the email address, password and username. Once all values are filled, the password is valid and the Sign Up button is clicked the Register() method is called from the fire-authentication.service which uses the firebase 'AngularFireAuth's createWithEmailAndPassword(email, password) method, which checks if the user email is unique and creates a User document in the User/ collection. After it is authenticated the homepage is then loaded.

Register User

Email Address:

Email

User Password:

Password

Username:

Username

Already have an account? Log In

Sign Up

```
Register(email:string, password: string, fullname:string){
  return new Promise((resolve, reject) => {
    this.afAuth.createUserWithEmailAndPassword(email, password)
      .then(userData => {
        userData.user.updateProfile({
          displayName: fullname,
          photoURL: ''
        }).then(() => {
          this.SetUserData(userData.user);
          resolve(userData);
        });
      });
  }, err => reject(err))
})
```

5.2.5 Sign In Page

The root-page redirects to the Sign In page, rather than the homepage if the user has not logged in yet. The user is prompted to fill in a Sign In form. This forum is comprised of an email address and user password. Once the fields have been filled the 'Sign In' button is available and upon being clicked the 'SignIn()' method is called. It returns a value using 'AngularFireAuth's signInWithEmailAndPassword(email, password) and then maps the result and saves the user information to a constant variable called userState, if the password and email are valid. The homepage is then loaded.

The "Already have an account? Log in" option at the bottom of the form redirects the user to the Sign In page allowing them to log in with an account they have previously made.

Sign In

Email Address:

Email

User Password:

Password

Forgot Password?

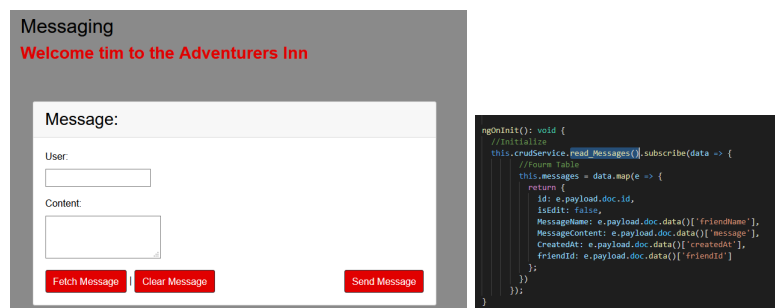
Not registered yet? Register

Sign in

```
Register(email:string, password: string, fullname:string){
  return new Promise((resolve, reject) => {
    this.afAuth.createUserWithEmailAndPassword(email, password)
      .then(userData => {
        userData.user.updateProfile({
          displayName: fullname,
          photoURL: ''
        }).then(() => {
          this.SetUserData(userData.user);
          resolve(userData);
        });
      });
  }, err => reject(err))
})
```

5.2.6 Messages Page

The Messaging Page's main functionality is to allow users to message other users and display their messages. At the top of the page the user is welcomed to the website and their username is displayed. There is a Message form present on the page that allows the user to input a users id and input the message into a text area that they wish to send. Once these are filled the user can then click the 'Send Message' button to send the message to the user by calling the CreateMessage() method. This adds the message to a record and calls the postMessage() method in the 'fire-authentication.service'. It calculates the current time and adds extra information to the post, such as the sender name. It calls the friendUpdateSender() and friendUpdateTarget to update the last messaged sent variable in the User/sampleuser/firends/samplefriend/lastMessage.



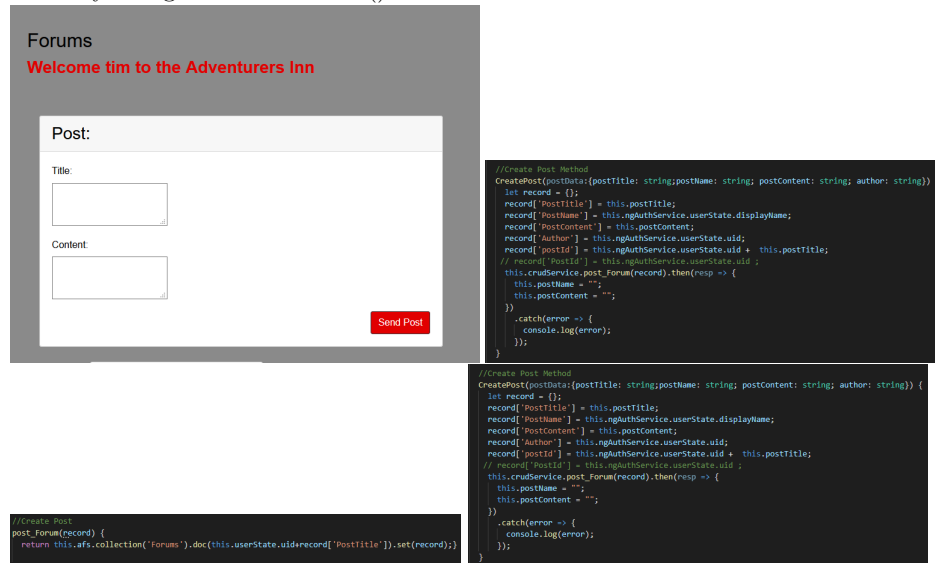
If the user has a message, the most recent message from each individual user is displayed. It does this by readMessages() method in 'fire-authentication.service'. It does this by mapping the result returned values from the snapshotChanges() method. Once clicked it loads a subpage where all messages can be displayed.



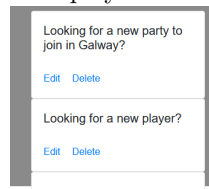
5.2.7 Forums Page

The Forums Page is responsible for allowing the user to post to a forums about their individual campaigns using the CreatePost() method. The CreatePost() packages the input into a record object and then calls the postForum() method. This posts it to a forum where the forum Id is a combination of the user unique id and the post title. This might be the name of there party, campaign other

whatever other topic they would like Other users of the application are then able to see these posts on the homepage and the forums page. The application then reads the posts from the collection Forums and displays them after mapping them by using the readForums() method.



At the top of the page the user is welcomed to the website and their username is displayed.



They also have the option to write about the contents of the post if they wish, once they user has filled in their desired information they can press the "Send Post" button which will add the post the forums collection in the database.

The user may then click on the title of a post to view it in entirely. Upon clicking the tile the postComponent page is loaded which saves the post details as a localStorage JSON object. The post then displays this information and then reads the forum by using the postId to access the data using the readPost().



By mapping the documents in the forums/sampleForum/ collection, the application displays all comments and replies posted on the forums by other verified users. The users will be able to reply to these comments and eventually

Title: HelpTestMe

Content: The only thing your testing is my patience

by: tim

red

tim

10:11 PM

[Edit](#) [Delete](#)

fwsf

tim

10:10 PM

[Edit](#) [Delete](#)

d

tim

10:08 PM

[Edit](#) [Delete](#)

Reply

Send Post

The homepage is where the user will be automatically redirected to, after signing up or logging in to the application. It displays the forums post and will eventually filter by date, popularity and type of post. This feature is not fully implemented yet.

This is where the imports of the project are managed and added to the project.

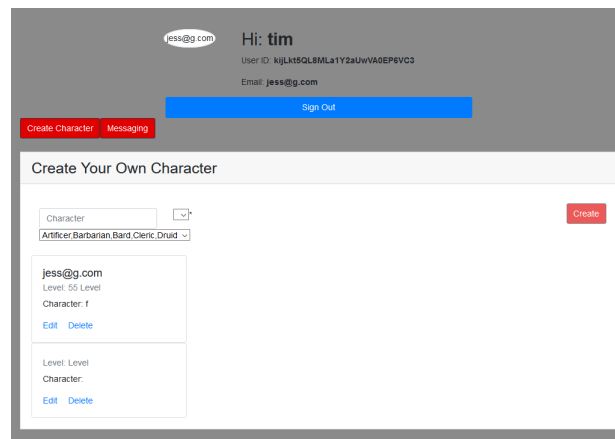
31

5.2.10 Groups Page

This page is a page to manage the different groups the user is a part of. This has yet to be fully implemented.

5.2.11 My Profile Page

This page displays the users information at the top. This allows the user to give his Unique ID to another player if they want to add them to their friends list. It has a Sign Out button which allows the user to log out of their profile. There are two buttons to navigate to the Create character page and the Messaging page respectively. Below this is a card that displays the user's character which upon clicking loads a character details page. The user is able to quickly edit their own characters here.



5.2.12 Create Character

This page allows the user to create a new character based on the rules for Dungeons and Dragons 5th Edition. The user must then input their character details including their character names, level, , Race, Class and Background. It used the CreateRecord() method which maps the information into a record and then calls the createNewCharacter() method from the 'fire-authentication.service'. This pushed the record to the database and creates a character document in the collection Users/sampleUser/Characters/.

Create Your Own Character

Character Name

Level 1

Character Race

Character Class

Character Background

Create

Back

```

//Create Character Method
CreateRecord() {
  let record = {};
  record['name'] = this.userName;
  record['level'] = this.userLevel;
  record['character'] = this.userCharacter;
  record['race'] = this.userRace;
  record['class'] = this.userClass;
  record['background'] = this.userBackground;

  this.crudService.create_NewCharacter(record).then(resp => {
    this.userName = this.userName;
    this.userLevel = undefined;
    this.userCharacter = "";
    this.userRace = "";
    this.userClass = "";
    this.userBackground = "";
    console.log(resp);
    this.router.navigate(['/profile'])
  })
  .catch(error => {
    console.log(error);
  });
}

//Create Character
create_NewCharacter(record) {return this.afs.collection('users').doc(this.userState.aid).collection("Characters").add(record);}

```

5.3 Container

Containerized software was used to make a container image to package the code and dependency of the application, so that it could be easily and reliably be form the environment on our local machines (Windows 10) to the environment on the virtual machine (Ubuntu). This meant that if the application need to be run on a environment there would not be any unnecessary complication.

5.3.1 Docker

For the purpose of this project, Docker was the containerized software used. In this project Docker creates a Container image of the application at runtime, allow the application to run on both of the environments being used in the project. Docker was executed using the Standard Docker CLI interface. The Dockerfile script was used for creating a containerize image of the application at runtime.

```

FROM node:latest as node
WORKDIR /app
COPY .
RUN npm i
RUN npm run build --prod

FROM nginx:alpine
COPY --from=node /app/dist/adventurers1m usr/share/nginx/html

```

This could then be used to run the application.

5.4 Cloud Storage

The cloud platform used for the purpose of this project was Microsoft Azure. The cloud platform was used to extract the running of the application away

for our personal machine, the reasoning behind this was that it would not be dependent on personal computers or WiFi connections.

5.4.1 Azure Virtual Machine

The Azure Virtual Machine Azure was a Ubuntu based server-side virtual machine used for deploying and hosting the application.

Chapter 6

System Evaluation

6.1 Initial Plan

The initial plan for the project was to make a social media platform with a target audience of players Table-Top Role-playing Game (TTRPG's) players community in which they could store and share information about their ideas, characters campaigns that they were part of. The web application should allow the users the communicate privately with a simple messaging system, The project should allow users to create forums for users to share ideas, discuss relevant topics as well as aiding the users in finding other users for both their in person and remote sessions

After researching different TTRPG's we decided that we would focus on Dungeons & Dragons (D&D) in particular as it has one of the biggest consumer bases and is often the entry point for players. We found while there were many small tools that would give the user information about the rules and aided a user in character creation such as Orc-Pub. Despite this we found there was no application or platform for discussing and sharing ideas while allowing users to easily manage their sessions and characters.

6.1.1 Goal of the Application

The general Goal of the application was establish an environment that the players of this popular franchise could use this as a communication tools between parties or individual users and a way to store, access and edit relevant information needed to share with other members of their group. More precisely the objectives of the application can be described as follows:

- Evaluate and establish the requirements of the user with regard to messaging.
- Evaluate and establish the requirements of the user with regard to sharing of campaign overview information.

- Evaluate the current tools available for character creation and establish the relevant information that is needed.
- Limitation of the Application.
- Authentication and Security.
- User Experience.
- Evaluation of the Technologies Used.

6.2 Evaluation of Objectives

The following section outlines how the objects described in the "General Goal of the Application" section were implemented and completed in the system.

6.2.1 Evaluate and establish the requirements of the user with regard to messaging.

Private Messaging

With any communication system, the user must have the user must have the ability to privately message other individual users of the application. To accomplish this users must be able to search for other users in the application, message them and then be able to view previous messages between the two.

Group Messaging

During the research of Table-Top RPGs, players are part of an individual parties, each party is comprised of a certain amount of players that participate as a character in a campaign. In each campaign there is a Dungeon Master(DM) which, for the use of the application acts as administrator for the participant of the campaign. The DMs must have a facility for message all members of the campaign with relevant information and also have the required information of player's characters. With this in mind, a group messaging system which allows these features was vitally important for the application.

6.2.2 Evaluate and establish the requirements of the user with regard to sharing of campaign overview information.

After establishing what is required for each individual campaign, it became clear that users would benefit from being able to find out about ongoing campaigns that they might have an interest in.

Forums

The way this was established in the application was have a forums page in which user could make general post conveying information of ongoing campaigns they are part of, for viewing for the public. In these forms, users can give the title of their campaign and an overview of what their campaign entails. A comment section was then added to this so that users could ask question or just general remarks of what they thought of the campaign. This connects users with other players in the community, giving them ideas or allow them to give information that might be needed, allowing for sharing of information from more experienced players to new players that might be unsure on rulings of the game.

6.2.3 Evaluate the current tools available for character creation and establish the relevant information that is needed.

During the research on the character creation tools currently available, it gives the user to select their, level, class, race, background, stats, armour class, speed, weapons, spells and much more. After evaluating this we came to the conclusion that the scope of this was too large for the use of this application and only the core attributes that the DM is required to know would be import to our character creator. The attributes we felt would be most need was the character's level, class, race, background and stats. [8] [16]

Character Creator

The Character Creators allows the users to fill out the relevant information of their character for us of DMs in campaigns. The user must also have the ability to make multiple character as it is possible for their character to die in the campaign meaning they must start from being with a new one or the user might be taking part in multiple campaigns.

The user is must also be giving to edit or delete there character at any stage, the reasoning behind this is as campaigns progress so does the player character meaning they will have changes to stats and level they will also receive experience points(xp) which will fluctuate constantly.

In the case of character death the user must also have the ability to remove character as they will be unable to use them again in the campaign, without this option to delete the user is storing unused character and also posses as a reminder of when that character died which could take from the user experience of the application.

6.2.4 Limitation of the application.

Character Creator

D&D has a vast amount of choices to pick from when it comes to making a player character, with new options being constantly added to the game, this can be seen by there being several updates to the game from the date we started this project to the current date. With this in mind keeping the character creator up to date would be a difficult task as the options the player can choose from would have to be expanded and changed repeatedly, for scope of this project this idea seemed unfeasible. This being said even if the option where to stay constant the sheer amount of them poses a task in itself, with players being able to choose from a vast of amount of races, classes, sub-classes, backgrounds, skills, the list goes on. Establishing this, we came to the conclusion that the base version of the options would be suited for the scope of this application.

Virtual Machine

When setting up the our virtual machine, we originally went with a Windows OS, the reasoning behind this was to simplify things by keeping the environment consistent throughout the creation of the project. A problem arose when trying to set up Docker on this virtual machine, this caused quite an issue as we had bout been using Docker on our own personal machines which were also running a Windows OS. After researching the issue we established that issue was the version of the Windows Server we were using did not allow for virtualization something that is needed when running Docker on Windows. The for this is that Windows needs WSL (Windows Subsystem for Linux) to run Docker as the platform runs natively on Linux. Once this was established we tried changing the version of Windows but found that the monthly cost to run it would be too expensive. Our solution to this was to then run a Ubuntu Server taking advantage of Linux instead of using a Windows OS.

6.2.5 Authentication and Security

After Evaluating the Firebase Database we found that authentication section there was a multitude of Sign-in methods. After researching these methods further we decided that using the Email/Password provider would be best suited for use in this application. The reasoning behind this was that the application would be able to stand alone not requiring any particular type of web browser to use it or have to be registered with any particular provider [12]. We added this functionality into the application by importing the AngularFireAuth from "@angular/fire/auth" then created a service called FireAuthenticationService to use and manipulate this.

The authentication was necessary in this project so that the user would be able to register an account and login. This would needed so that the applications experience would be unique to the individual user and that the information that

was personal to them could be stored. When the user access the application they are given the option to sign in or sign up to the application, this option is present on the right side of the navigation bar which is at the top of the application. Once the user have filled in the relevant information for either of these options user authentication is enabled and the button in the navigation bar changes to a log out option.

6.2.6 User Experience

The user experience was an important part of the development of the application. It was vital that it was easy to navigate this was achieved through the implementation of a navigation bar present at the top of the application which allows the user to get to any of the desired pages. The look of the application was also something we kept in mind, deciding on using bootstrap for the styling. This allowed the application to be easily viewed and read by the user, the inputting of information was able to be straight-forward and seamless. The colour scheme:

- Red was picked so that the user would be drawn to this part of the application, everything depicted in this colour is used for navigating of confirmation of inputted information.
- White was used for any input information where the user is able to input text, whether it be for the registry, character-creator, messaging or post. This was down so that the text could clearly seen by the user.
- Grey was used and the background of the app to help highlight the rest of the view which is important to the user.

Text was made relatively large and in simple styling to insure that the user would not have any trouble while trying to read it.

Accessing messages and forums was made simple for the user by adding them to the right side of the application this meant that the user to not have to navigate to different pages to check for the most recent messages and posts. The user shown all conversation they are part of between the messaging and group messaging pages, and can check the character they have created in their personal page. A comment section on the posts in the forums sections allows for user to interact with the community and to ask various question which other user may be able to answer, improving their experience.

6.2.7 Evaluation of the Technologies Used

It was our intent for this project to use technologies that we had very limited to no experience using. Firebase, Docker, Azure and Ubuntu fit it this category, while we have had a bit of experience with Angular it was nothing to the extent of the application idea. All of these technologies where interconnected

to produce the final application for the project. When the user attempts to logs in, Firebase authenticates the user, through both Angular and Firebase the user state is set. All information the user puts into the application is stored on the Firebase database which can then be retrieved and view by the user in the application. Docker is used to build a containerized image of the application once it was ready for production and Azure, using Ubuntu as it's OS, was used to host the application on a cloud platform abstracting it from the local device.

Implementation of Angular

The Angular Framework was used to created the web pages for the application, this was then deployed locally before it was containerized. The local deployment of the application proved extremely useful as we could view edits and addition to the code in real time on the app. This meant that a fully functional implementation of the relevant could then be committing to GitHub which would then be pulled to our virtual machine. This was extremely useful throughout the development of the project as multiple changes to code and styling needed to be done before the current part of the component being worked on was ready to be integrated into the application. Meaning that we would not have to unnecessarily run the server which was a plus on our side as this would increase the cost.

There was various types of UI component used on when implementing these pages with Angular and then styled with Bootstrap. These pages were able to be seamlessly navigated with the use of the navigation bar present at the top of the application, we achieved this by using the methods built into the Angular Routing import.

Through the imports that Angular has for Firebase it meant that we could use the methods present to work with the database for authentication and storage. Managing the states of the application when the user was updating information constantly meant that Angular needed to be able to consistently read the changes to Firebase and re-render the page to keep the information up to date.

Implementation of Firebase

Firebase is an important feature of the application as it was used for authenticating the user and for storing all the user information. There was a learning curve when it came to using Firebase to store the information, as it was a NoSQL database and we had previously only worked with SQL databases. This meant we had to but quite some thought and time on exactly how we were going to store all the information in a way that was easy to understand and retrieve.

As previously described in Section 6.2.5, Firebase was responsible for authenticating the user of the application. It was able to successfully, sign the user in or register a new user and, if the user wished, log them out of the application. This was possible using Firebase's predefined methods that we were able to access once the relevant import had been brought into Angular.

Implementation of Docker

Docker was an important part of this project as it gave the ability for the application be abstracted away for our local environment and reliably run on a cloud platform. This was done by making a containerized image of the application at runtime. This abstraction meant that if the application was running in a work environment and needed to be expanded across multiple server with different environments there would be no issue and the application could run reliably.

Implementation of Azure

Azure was used for hosting the application on a cloud environment, extracted away for our local machinery. We thought that this would be a valuable learning experience as the running of server side applications is commonplace in the workplace and continuing to be the norm for the foreseeable future. Setting up the virtual Azure posed some problems, though we had worked with virtual machine on cloud platforms before we had not worked with an application of the scope of the project. We decided initially to use a Windows OS but after various problems we switched to a Ubuntu server instead, this is described in more detail in Section 6.2.4.

6.3 Plan vs Reality

6.3.1 Location

The initial plan for the project was to allow the user, through the use of Google Maps to find users the application in their vicinity. This unfortunately did not get implemented into the application however this might not have been necessary. Making the user show there location for something that is usually played with peers did not seem to be that useful a feature. It is true that people can play Table Top Role-playing games online or with people they might not know, however with the forums table, DMs(Dungeon Master) or admin for the case of the application, have the ability to share information about their ongoing campaigns and all user are able privately message each other. This means if a user shows interest in the campaign they have the ability to communicate with each other and decide meeting new people to play the game with.

6.3.2 Rating System

Initially we had an idea for having a rating system on the forums page for campaigns that have been posted. We decided against this as the campaign experience is personal to the player who took part in them and it is hard to get an overall impression of what the experience was like from a brief description given. Users are still able to comment their opinions of the campaign where they might can also ask question if they are interested, if this is the case the

admin can give the user an inclination of what the aspects of the their campaign are.

6.4 Opportunities for Improvement

6.4.1 Categorise Posts

The ability to categorise the campaigns under genre and level is something they would have liked to implement into the application. This would allow the user to find campaign that suit them more personally, where they can find low level campaigns if the are new to the game or for the more experienced player, high level campaigns where parties are might are implementing ideas that thy might not of experienced before. The ability to find post by genre would allow users to view campaign relative to what they are interested in, this could allow the user to come up with ideas for their own campaigns or they might just have a general interest and want to learn more about it.

6.4.2 Filtering and Searching of Data

The implementation of filtering and searching different types of data for the user would make the experience of the application more enjoyable. With regard to messaging, this would allow the user to search for their friends and start a conversation with ease and if they have already messaged before they could find the conversation without having to look through all their other messages. Similarly implementing this type of system to the users profile page with regard to their character would allow the user to find their character instantly without having to look through all of there characters. They could find characters by name or if they where starting a new campaign they could check all their character of the same level, this would be useful if they would like to use a character they have already made instead of making a new one.

6.5 Testing

For the majority of the project, iterative manual testing was used. The reason for this was that writing test's for a database that was currently being build was a difficult task as the test would need to verify that the data had made it to firebase. Selenium was used to create simple test cases to test for multiple users.

Chapter 7

Conclusion

This project was chosen for the complexity of it's database as well as the new technologies that we got to learn. As we continued our research it became clear to us that not many people had created the specific type of program we aimed to create. As this was the case documentation on Angular and Firebase together was very scarce and this made us read the documentation and invent our own ways to provide creative solutions to the problems we faced.

]

The experience we gathered has shown us the importance of planning, and how a small error can greatly slow down development. As well the challenge of having to complete the entire project remotely and not having access to the same resource gave us a great opportunity for independent learning.

Though we did not finish all our goals due to time constraints and other deadlines, we did learn a great deal about NoSQL databases, firebase and Cloud Hosting. It was an exciting project as we had the opportunity to build something in an area we had always had a clear interest in. In conclusion this was a great learning experience and though the project was not perfect, we are very proud of it.

Bibliography

- [1] Compete 366. What is microsoft teams and who should be using it?
- [2] Sunnia Amjad, Naveed Ahmad, Tanzila Saba, Adeel Anjum, Umar Manzoor, Muhammad A. Balubaid, and Saif Ur Rehman Malik. Calculating completeness of agile scope in scaled agile development. *IEEE Access*, 6:5822–5847, 2018.
- [3] AngularFire. Angularfire (for angularjs).
- [4] Ubuntu Canonical. The story of ubuntu.
- [5] David Chappell et al. Introducing windows azure. *Microsoft, Inc, Tech. Rep*, 2009.
- [6] Visual Studio Code. Visual studio code.
- [7] Valerio Cosentino, Javier L. Cánovas Izquierdo, and Jordi Cabot. A systematic mapping study of software development with github. *IEEE Access*, 5:7173–7192, 2017.
- [8] ddBeyond. Ddbeyond character information.
- [9] Discord. What is discord?
- [10] Docker. Docker.
- [11] Docker. What is a container? a standardized unit of software.
- [12] Firebase. Authentication.
- [13] Firebases. Firebase.
- [14] The Linux Foundation. What is linux?
- [15] Marc J Hadley. Web application description language (wadl), 2006.
- [16] Players Handbook. Player’s handbook.
- [17] Brian Hayes. Cloud computing, 2008.
- [18] Azure Microsoft. What is azure?

- [19] Elizabeth Mkoba and Carl Marnewick. Conceptual framework for auditing agile projects. *IEEE Access*, 8:126460–126476, 2020.
- [20] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. Nosql databases. *Lecture Notes, Stuttgart Media University*, 20:24, 2011.

Appendix A

GitHub Repository: <https://github.com/Antaine/FinalYearProject>