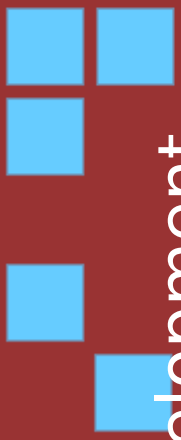
B.Sc. (Hons) in Software Development



Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Arcanium

By
Owen Casey

April 26, 2024

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the ever-evolving area of role-playing games, Dungeons and Dragons (D&D) stands out as not only a pioneer of the genre but also as a canvas for social interaction, creativity, and storytelling. Arcanium aims to offer players, both new and experienced, a unified, accessible platform where useful resources are consolidated in one location. It is designed to be a comprehensive hub that streamlines the complexity of the D&D experience, offering tools for exploration, guidance, character creation, and community building.

1.1 Purpose

The primary aim of Arcanium is to provide a streamlined and centralized digital platform that consolidates essential resources and tools that are needed by players of D&D. D&D has long reigned as the most popular tabletop role-playing game, and in recent times has been evolving as we continue further into the digital age. This evolution has not come without challenges, and a common challenge that players and dungeon masters find themselves in is navigating the complex and fragmented landscape of resources and tools. Traditional D&D resources are typically physical books with hundreds of pages, and finding the needed information within these resources can be challenging, especially for new players, where the intricacy of information that is provided can be overwhelming. It is with this context in mind that the purpose of Arcanium is clear, to make this process more accessible and straightforward.

Arcanium aspires to be more than just a digitization of these resources, it seeks to be a comprehensive application that augments every aspect of the D&D experience. By consolidating resources and tools into one location, Arcanium addresses the aforementioned complexities and challenges. It envisions an application where users can go from creating a character, to searching resources, to getting guidance

all without having to navigate numerous resources or tools in different locations or contexts. Beyond the consolidation of resources and tools, Arcanium recognizes that the social side of D&D is intrinsically the aspect that makes it compelling and popular. The platform offers a location to keep these aspects controlled in one space, through detailed social components. These attributes enhance the accessibility of D&D for all, addressing a common concern surrounding its initial complexity.

In essence, Arcanium aims to offer a digital environment of consolidated resources and tools, with the aim to simplify, enrich and elevate the Dungeons and Dragons experience. Recognizing the evolution of the game into the digital age and its challenges, Arcanium serves as a solution to the complexities and common concerns of new and experienced players alike.

1.2 Background and Rationale

Dungeons and Dragons, since its inception in the 1970s, has transformed from a niche hobby into a cultural phenomenon. It influenced countless pieces of media, including games, television, shows, books and movies. It has evolved from a simple, traditional tabletop game in the 1970s, facing countless challenges and tribulations, being more popular than ever today.

1.2.1 Historical Context

Dungeons and Dragons laid the foundation for the role-playing game industry. Created in the early 1970s by Gary Gygax and Dave Arneson. With a pioneering vision, the pair introduced a game that combined the traditional elements of tabletop games with immersive storytelling and character development, which was a concept previously unexplored in the genre [1]. The impact of D&D extends beyond the confines of the tabletop genre, with its influence being felt extensively in media. D&D is heavily influenced by J.R.R Tolkien's Lord of the Rings, which is considered by many to have set the standard for modern fantasy. Just as Tolkien's work established a benchmark for fantasy, the same can be said for D&D's influence on modern role-playing games as a whole. This comparison showcases the significant role D&D has played in setting the standards for modern RPGs, both tabletop and video games alike. Titles such as "Baldur's Gate" and "Neverwinter Nights" are directly adapted from D&D lore, and are some of the most impactful RPGs of their time, selling millions of copies.

1.2.2 Modern Context

Today, Dungeons and Dragons enjoys a resurgence in popularity, fueled in part by its transformation to digital platforms and its extensive modern presence in media. "Stranger Things", one of the most popular television shows of all time, reached a staggering 1.352 billion hours of viewership within the first 28 days of its season 4 release. The show is heavily influenced by D&D, with its main antagonists across each season being an iconic villain from the D&D universe. The rise of streaming platforms have introduced a new form of D&D content, with live-streamed campaigns such as "Critical Role" attracting millions of followers, even inspiring a direct animated adaption of the groups campaign through Amazon primes "The Legend of Vox Machina". The most recent, and most substantial form of D&Ds success has come in the form of "Baldur's Gate 3", which is a continuation of the hugely successful game series from the early 2000s. The game, set in the mainstream D&D universe, directly adapts the 5th edition mechanics to a video game context. Scoring a 96 on Metacritic, the game won countless awards, including game of the year, making it one of the highest rated games ever, with many considering it to be the best RPG of all time [2].

It's clear that these recent successes have significantly increased interest in the traditional tabletop version of the game, with more new players than ever in the current 5th edition of the game. The modern visibility and representation of D&D in mainstream media have contributed to the appeal of the game, with it now reaching demographics who typically may not have considered tabletop gaming as a pastime. With these considerations in mind, it's evident that Dungeons and Dragons is more relevant than ever, and the concept of Arcanium as an application has a clearly defined audience and offers a service that newcomers and veterans alike would find useful.

1.3 Project Objectives

The overarching objective for Arcanium is to enrich the Dungeons and Dragons experience by providing a digital transformation of essential resources and tools, while enhancing accessibility and encouraging player community. The objectives can be categorized into three main areas: User Experience and Accessibility, Innovation and Integration, and Deployment and Performance.

1.3.1 User Experience and Accessibility

- Centralized hub: Centralize necessary tools, resources, and information for D&D players into a single, accessible platform.

- Support for New and Experienced Players: Implement a user-friendly design, taking into account the context and needs of both new and experienced players.
- Multi-Platform support: Allow users to access the platform on both desktop and mobile devices.

1.3.2 Innovation and Integration

- API Integration: Integrate a comprehensive API for resource searching, reusing it when needed across the application as a whole.
- Dynamic Character Creation: Provide a detailed character creation tool with an interactive, step-by-step process.
- Social Connectivity: Integrate social features to allow players to add friends, join campaigns, and message each other in real time.
- Artificial Intelligence Integration: Employ generative AI models to provide users with a way to generate creative stories or scenarios for use in their campaigns, or get help from a dynamic chatbot.
- Virtual Tabletop: Implement a virtual tabletop to provide users with a virtual space to roll dice, view maps, and interact with friends.

1.3.3 Development, Performance and Security

- Responsive and Accessible Design: Develop Arcanium as a multi-platform application, usable across different platforms.
- Follow Best Practices: Adhere to industry practices in software design, security, and user experience design to provide a reliable and user-friendly platform.
- Comprehensive Security/Authentication: Implement robust authentication to ensure security is maintained across all routes, and sensitive user data is maintained.
- Scalable Database Architecture: Ensure user data is stored to a database, including created characters, stories, and friends or messages.
- Deployment: Ensure the application is successfully deployed, and accessible from anywhere.

- Agile Development Methodology: The project will utilize Agile development practices, specifically Kanban, to ensure the project development cycle is productive, flexible, and continuously delivering.

1.4 Chapter Overview and GitHub

1.4.1 GitHub

Arcanium's source code and related files are hosted on GitHub, providing accessibility and version control for all contributors and users of the application. Below is the URL and a brief overview of what you can find at this URL:

GitHub URL: FYP-Arcanium Repository

- **Codebase:** The repository contains all the source code for both the frontend and backend of the Arcanium application.
- **Video Demonstration:** A video demonstration of an overview of the application.
- **A0 Poster:** The A0 poster is located here
- **README:** The main documentation file, which provides an overview of the project, setup instructions, a list of features, and other essential information about Arcanium.

1.4.2 Introduction

The introduction sets the stage for Arcanium, explaining its scope, the relevance of the topic, and why it's important. It defines the objectives of the project through detailed bullet points categorized into three categories.

1.4.3 Methodology

The methodology explains the approach taken to execute the project, highlighting the types of research used for the project. It discusses the use of Agile methodologies, planning processes and the structure of the development process. This section also covers the testing and validation of the project, as well as the use of any other development tools.

1.4.4 Technology Review

The technology review serves as the literature review section of the dissertation. This chapter is tightly coupled with the projects initial context and objectives. This chapter thoroughly covers the technologies used at a conceptual level, and provides detailed insight into how the project itself was built.

1.4.5 System Design

This chapter provides an in-depth explanation of the system architecture, detailing how components are coupled and the standard of the project. The system design chapter heavily uses diagrams to explain the architecture of the system.

1.4.6 System Evaluation

This chapter evaluates the project against its initial expectations and objectives. The evaluation chapter details how the project was tested, and proves the robustness of the project. It describes the efficiency and performance of the application, and highlights any limitations that the project contains.

1.4.7 Conclusion

The conclusion summarizes the project as a whole, taking into account any insights from each prior chapter. The conclusion identifies any potential for future development and provides a clear reflection on the project itself, including any insights gained

Chapter 2

Methodology

2.1 Software Development Methodology

The development of Arcanium required a flexible and efficient approach to software development, with the ability to adapt to evolving requirements and facilitate continuous development. Considering the project took place over two semesters, it was important to accommodate an iterative approach, considering the length of the project, every aspect was bound to evolve over time. In choosing a software development methodology, it was essential to acknowledge the project's dynamic nature, as well as the fact that the project was a solo development endeavor.

2.1.1 SDLC Model: Kanban

For the development of Arcanium, Kanban was chosen as the SDLC model due to its flexibility, simplicity and effectiveness in managing solo projects. Kanban is especially suited for projects which require incremental improvements and continuous delivery. This methodology suits the projects development approach perfectly, offering Arcanium organization and flexibility in its development process. Kanban focuses on visualizing work, limiting the amount of work in progress and managing flow. Utilizing a Kanban board through Atlassian's Jira, it was simple to glance at the board to have an understanding of what tasks were in progress, and which tasks were upcoming or completed [3].

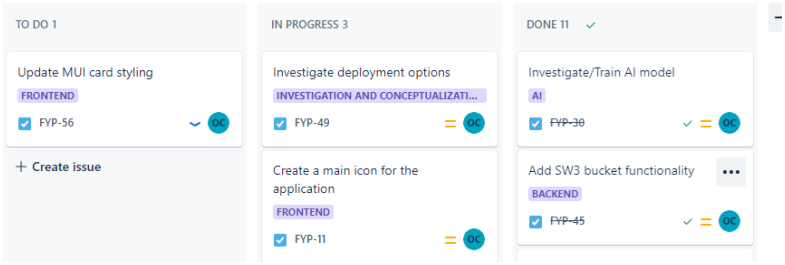


Figure 2.1: Kanban board snapshot showing task organization and categories.

Adaptability and Flexibility

The adaptability and flexibility of Kanban is the most significant factor for Arcanium's development. It was essential to keep in mind that the project spanned two semesters, close to 9 months of development time. This long period of time was always bound to introduce obstacles and this needed to be taken into consideration when choosing a development model. Kanban allows for real-time adjustments and re-prioritization of tasks, without the need for rigid planning cycles or weekly re-evaluations. Tasks were typically split up into broader topics through the use of epics, which categorized the task's purpose.

Solo Project Management

In solo project management, Kanban excels by providing a clear, concise framework that aligns with a solo developer's needs. The ability to visualize all the components of a project on a Kanban board enables the developer to maintain focus on current tasks, and plan upcoming tasks efficiently. Kanban focuses on limiting the amount of current work in progress, which helps prevent task overload and maintain high quality work. Kanban's continuous flow ensures that progress is always ongoing and incremental. This type of progression is especially beneficial in solo project's where workflow can be influenced by personal capacity and changing project needs. Considering that the project cycle takes place during the academic year, where other modules have assignments and exams, this management style becomes paramount. It allows the project to be dynamically adjusted around other academic commitments, ensuring project work does not become overwhelming.

Comparison

When comparing Kanban to other software development life-cycle models, such as Scrum or Waterfall, it shows several key differences that make it more suitable for Arcanium's development needs.

- **Flexibility Over Rigidity**: Unlike Scrum, Kanban doesn't utilize sprints, which are structured, set intervals of development time, opting for a more flexible approach. This flexibility is beneficial in an academic setting, where project work must be balanced with other modules assignments and exams.
- **Solo Management Over Team Management**: Scrum focuses heavily on team-based management, utilizing stand-up meetings and weekly reviews, as well as defined team roles such as a scrum master or product owner. Kanban focuses on task-based management, which makes it more suited to a solo project.
- **Continuous Delivery and Improvement**: Unlike other models, which move through rigid phases or sprints, Kanban encourages iterative, ongoing progress and refinement. With a model such as Waterfall, where tasks must be completed before moving on to the next, Kanban offers a more adaptable approach, which fits with the needs of an ever-evolving project. This ensures the project can adapt to unexpected challenges or discoveries without waiting for the next phase or sprint, maintaining development momentum.

In conclusion, the Kanban SDLC model was instrumental in navigating the unique challenges of a long term solo development project, while juggling other responsibilities in the midst of the academic year. After a comparison with other popular models, Kanban suits the project's needs perfectly. The model's flexibility, adaptability, and efficiency in solo management provided the foundation to evolving project needs and encouraged iterative, ongoing improvements throughout the project development cycle.

2.2 Research Methodology

This section outlines the approach taken to research, conceptualize and validate the requirements and features for Arcanium. The research methodologies applied utilize both qualitative and quantitative methods, to fully understand user needs and the technological standards.

2.2.1 Research Method: Prospective Literature Review

The literature review served as a foundational research aspect of the project. This involved an extensive examination of both academic and industry literature and articles which covered several key areas of the project: the resurgence of Dungeons and Dragons, the integration of artificial intelligence, and the technological considerations of app development. Additionally, the review examined other key

areas of the application, which are less niche and already well-established, but nonetheless are still critical to the applications context. This includes integration of APIs, authentication, and app development frameworks. The detailed exploration of AI was motivated by an intent to ensure the applications technological stack was innovative and reliable, and by examining the current trends of D&D's popularity, the review provided clarity on the current market, as understanding where the context of current trends emerge from provide insight into what sort of players modern D&D attracts and retains. The review also acknowledged the importance of the technical aspect of the applications research. With D&D being such a vast game with a plethora of resources, it was important to research how this data could be attained and translated into a digital context, while ensuring the application maintains fair use of content.

CALYPSO: LLMs as Dungeon Masters' Assistants (2023) provided critical insights into the innovation of AI integration in the context of D&D, as an interface utilizing Large Language Models (LLMs), such as OpenAIs language models, to assist Dungeon Masters (DMs), who are the organizers and driving force behind a D&D campaign. Notably, this source concluded that LLMs are proficient brainstorming partners for DMs, capable of generating ideas that they could grow from through their own creative expression [4]. Similarly, Santiago et al. (2023) demonstrates the transformative potential of generative AI in D&D scenarios. The study presents how AI can significantly enhance the D&D experience through the use of generated images and stories, which can increase the engagement of players in the storytelling process and make the game more visually complex and dynamic [5].

In recent years, D&D has experienced a remarkable resurgence, with a significant driver of this popularity being its embrace by popular media and culture. Once a niche, often shunned hobby, it's now being covered in mainstream media with popular shows such as "Stranger Things" and receiving enthusiastic endorsement by celebrity fans. It has even made appearances on late-night news shows, showcasing the game to an unexpected demographic [6]. The popularity of online platforms and streaming services has played a significant role, pivotal among these, *Critical Role*, a popular live-stream of voice actors playing their D&D campaign, stands out as a monumental influence [7].

The technical challenge of translating an intricate world and system like Dungeons and Dragons into a digital context required extensive research. Such a process involves understanding the format the data will be in, and how it will be utilized. The most efficient way to do this was through utilizing a robust API, which we can use to get information about resources, game systems, and anything else we need. Something to keep in mind is the complexities of fair use and copyright. Seeing as D&D is a licensed product, particular attention was given to the Open Game License (OGL). The OGL permits fair use of D&D related content.

By adhering to these guidelines, we can ensure Arcanium respects the property rights and positively contributes to the D&D community [8].

The literature review provided invaluable insights into the project's development, the current Dungeons and Dragons landscape, the innovative potential of AI, and the technical considerations for the project.

2.2.2 Research Method: Survey

To ensure the development of Arcanium was closely aligned with user needs and preferences, a local targeted survey was conducted among new and experienced Dungeons and Dragons players. This survey's purpose was to understand which features from our project objectives outlined in the introduction were deemed most desirable among new and experienced players.

Participants were selected among peers, submitting their results anonymously to ensure fair feature scoring. Each participant was given a form to fill out, and told to rank each defined feature a priority level, with 1 being the most desirable, and 6 being the least desirable. This method of research resulted in both qualitative and quantitative data. The features that were chosen for ranking were resource searching, help chatbot, generative story generation, character creation, social connectivity and virtual tabletop. Each feature was conceptualized and explained briefly to each participant, ensuring they understood exactly what each feature accomplished.

Respondent	**Res Searching**	**Chatbot**	**Story Gen**	**Character**	**Social**	**Tabletop**
Experienced 1	1	2	4	3	5	6
Experienced 2	1	3	2	5	4	6
Experienced 3	1	2	4	6	3	5
Experienced 4	1	3	2	5	4	6
Experienced 5	1	5	3	2	4	6
New 1	3	2	1	4	6	5
New 2	1	3	6	2	5	4
New 3	1	4	3	2	5	6
New 4	4	3	2	5	1	6
New 5	1	2	4	5	3	6
Average	**1.9**	**2.4**	**2.8**	**3.6**	**4.4**	**5.9**

Table 2.1: Survey Results on Feature Importance

The results shown in the table above indicate some interesting insights. Resource searching was consistently ranked as the most important feature for both new and experienced players, with every experienced participant ranking it as their most desirable feature. The chatbot and story generation were also notably

highly desired. Character creation and social connectivity were mid-level priority across the board, with the virtual tabletop feature consistently ranking as the least desirable feature among each respondent. The survey findings are invaluable for the development of Arcanium, as they offer insight into which features can be deemed the highest priority, which can be used to guide the application's initial development process and evolution.

2.2.3 Research Method: Technology Exploration

After establishing Arcanium's features and understanding the context of the project, it was important to choose a suitable base foundation and framework, technology wise, to develop the project with. Understanding the initial goals and objectives of the project, it was essential to select a technology stack that not only facilitated the objectives of Arcanium, but also aligned closely with our methodologies and principles of multi-platform support.

- **Dynamic and Responsive UI**: To integrate features such as interactive character creation, a chatbot, and more, it's important to have an easy to use, user friendly UI.
- **Scalability**: To accommodate an iterative development process through Kanban, it's important to choose a technological base that supports this development approach, without compromising performance.
- **Comprehensive Community**: The technological foundation should have a thriving and popular community, with significant amounts of frameworks and libraries.
- **Cross-Platform Compatibility**: Users should be able to access the application on both desktop and mobile devices.

Given these requirements, the exploration of the technologies that fit these guidelines point towards web frameworks. Leveraging a popular, well supported and highly customizable language such as JavaScript provides the application with not only a great technological foundation, but also a scalable, iterative system which can be extended with many comprehensive frameworks from the JavaScript ecosystem.

2.2.4 Validation and Testing Methodology

To ensure the application remains as bug-free as possible, it's essential to choose a testing framework that aligns with the application's needs. To do this in Arcanium,

we will utilize testing through Jest, which is a powerful and easy to use testing framework for JavaScript. Utilizing Jest, we can perform methods such as snapshot testing, mocking, and asynchronous testing. Jest provides support for JavaScript, making it perfect for the applications technological stack [9]. While we utilize Jest, it's also essential to manually intervene and test the application personally. Manually testing components ensures that if the automated tests miss something, it can be picked up in the manual test. There are different forms of manual testing, such as exploratory testing, which is testing the application interactively, or usability testing, to ensure the application functions normally in numerous use cases, with different, unexpected actions being tested in tandem to ensure consistency. Utilizing Jest, we can automate the testing and validation of the functionality and logic, and with manual testing, ensure that the application meets user expectations in real world scenarios and use cases.

2.3 Methodology Review

- **Software Development Methodology**: **Kanban** was chosen as the SDLC model for it's adaptability, flexibility and efficiency in solo project management.
- **Research Methodology**: A **literature review** provided a foundational understanding of the resurgence of Dungeons and Dragons, AI integration in the context of D&D, and technological consideration and trends. A **survey** was conducted to gain insight into the importance of specified features to the prospected user demographic. This survey offered valuable insights into what features users prioritize based on their experience level.
- **Technology Exploration**: After research into technological prospects for the application's development, it was decided that **web frameworks**, specifically the **JavaScript** ecosystem, would meet the application's needs and provide customization and scalability.
- **Validation and Testing Methodology**: Utilizing a combined testing approach, **Jest** was chosen as the automated testing and validation framework due to its compatibility with the technology stack of the project. **Manual testing** was also considered essential, to compliment Jest and ensure real scenario usability.

Chapter 3

Technology Review

In the development of Arcanium, a multi-functional platform designed to enhance the Dungeons and Dragons experience, a meticulous and detailed selection of programming languages, frameworks, databases and other relevant technologies was essential. This chapter serves as the literature review of the dissertation, comprehensively exploring the technologies used across the project at a conceptual level. It is coupled directly with the projects objectives and methodologies, showcasing the impact the research had on the development process.

The application's ambition to provide a seamless, multi-platform, enriching experience of D&D, with features such as interactive character creation to social connectivity requires a robust and scalable foundation, with intricate use of other frameworks and technologies to complement the application's needs. This section will heavily utilize authoritative references and reviews of de-facto and de-jure standards, to ensure Arcanium's development aligns with the best industry practices.

3.1 Programming Languages and Frameworks

The foundation of Arcanium's development took a careful selection of programming languages and frameworks. These choices were influenced by our technology research methodology, where we explored the project requirements and decided that a web-based approach suits the application.

3.1.1 JavaScript

JavaScript, one of the most popular web-based languages, has evolved throughout the years to overcome its weaknesses through the use of comprehensive frameworks that complement the needs of any application, with over 80% of JavaScript

applications using some form of libraries [10].

Conceptual Foundations

The architectural cornerstone of JavaScript is its event-driven, non-blocking model. This model enables developers to build applications capable of handling a multitude of processes concurrently without locking the main execution thread—essential for maintaining high performance and scalability.

- The use of real-time interactions without reloading the web page, fundamental for features such as real-time messaging or asynchronous data processing.
- A comprehensive overview of asynchronous patterns like promises and async/await provided by "Semantics of Asynchronous JavaScript", which illustrates methods to maximize efficiency in web applications [11].
- Universal utilization across web browsers and operating systems, ensuring accessibility for all users [12].
- An extensive library and framework ecosystem that enhances development, enabling the use of frameworks like React, Angular, or Vue for building visually compelling, cross-platform user interfaces.
- The ability to use Node.js for server-side logic, remaining within the JavaScript ecosystem and streamlining development by avoiding external backend languages like Python or Ruby.
- Continuous improvements and expansions driven by a thriving community, affirming JavaScript as a solid choice for both server and client-side development.
- An event-driven, non-blocking architecture that enables the creation of high-performance, scalable web applications—a critical feature for the needs of Arcanium.

In conclusion, JavaScript's role in Arcanium extends beyond only logical functionality, instead being integral to providing a seamless, efficient and accessible platform for Dungeons and Dragons players. The languages event-driven, non-blocking, asynchronous nature, which are covered in detail in JavaScript's vast documentation, ensure that Arcanium will operate efficiently [13], and the vast ecosystem will provide flexibility and customization where needed.

3.1.2 React

React, a JavaScript library for building user interfaces has redefined the way a lot of developers approach front-end development. React stands out for its usage of component-based architecture, its usage of virtual DOM implementation, and its flexibility. React was pioneered by Meta developers, and in 2017, was re-licensed to the MIT license, encouraging the growth of the open-source ecosystem [14].

Conceptual Foundations

Reacts design philosophy intends to make UI development more intuitive, dynamic and simplistic. It does this through the following concepts:

- **Declarative Nature**: As outlined in "React Design Patterns and Best Practices", a thoroughly detailed book on React, React enforces a declarative programming paradigm, which is one of the many reasons it's a powerful framework [15]. Since React allows developers to describe their user interfaces in a declarative way, this enables the library to handle the rendering and state management.

- **Component-Based Architecture**: Applications in React are built using components that encapsulate their own state and logic, promoting developers to design components for reuse where applicable, increasing the performance and efficiency of the application.

- **Virtual DOM**: React optimizes performance by minimizing direct manipulations to the Document Object Model, instead creating a virtual DOM, which is a lightweight representation of the DOM [16].

- **Vast Ecosystem**: Reacts ecosystem includes a vast array of tools and extensions, such as React Router for navigation or Material UI for styling.

- **Community Support**: The community behind React offers a vast amount of resources and support, with it being one of the most popular modern JavaScript frameworks.

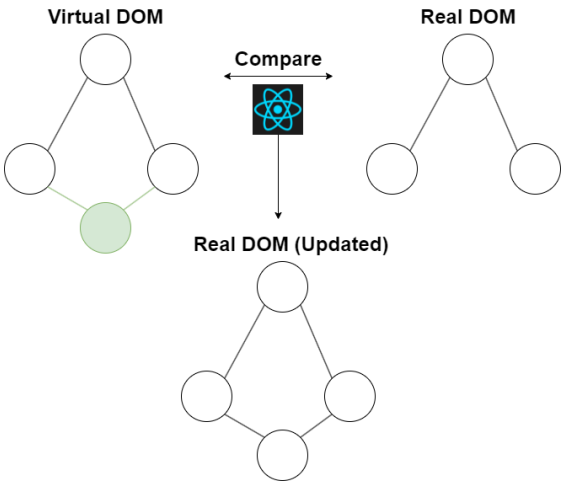


Figure 3.1: React Virtual DOM Diagram

In essence, Reacts features and large community offer a robust framework to develop a visually pleasing, proficient application. It not only encourages these conditions, but also ensures the application remains scalable as it grows in the development process.

3.1.3 Node

Node.js allows for the creation of web servers using JavaScript for server-side scripting, which allows developers to create both client-side and server-side applications in the same language, which can unify the code-base and make it more consistent. It's based on event-driven architecture and handles concurrency, making it suitable for Arcanium's needs.

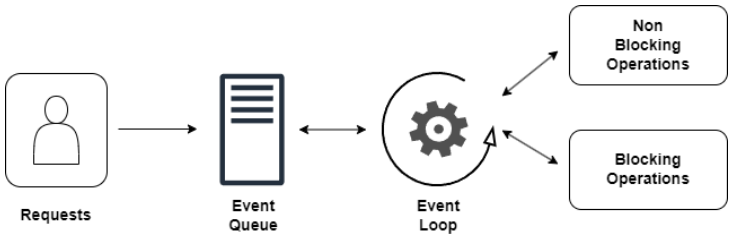


Figure 3.2: Node.js Architecture

Conceptual Foundations

The conceptual strengths of Node.js that are particularly relevant to the development of Arcanium include:

- **Event-driven Architecture**: Node operates on a non-blocking, event-driven architecture, ideal for applications that require concurrency without the overhead of managing multiple threads [**?**].
- **npm Ecosystem**: Included with Node, npm is a robust package manager which offers the capability to easily manage and download packages, modules, and libraries. The package manager sees millions of downloads per day across millions of packages [**?**, **?**], making it the biggest single language code repository in existence.
- **Cross-Platform Development**: Node's cross-platform support facilitates a consistent development and deployment experience across different operating systems and browsers which is essential for Arcanium.

In the context of Arcanium, Node is one of the most significantly used tools. Node is used to start the application, download packages, manage application frameworks and tools, and more. Utilizing Node provides the application with a robust, optimized backend solution.

3.1.4 Other Frameworks

In the development of Arcanium, countless frameworks and packages are utilized, While JavaScript, React and Node form the foundation of the application's technological framework, Arcanium also leverages other frameworks and libraries to address specific, niche needs. Here are some of the most prominent examples of these technologies:

Express

Express.js is essential in the server-side architecture of Arcanium, providing a minimalist, easy to use framework for web server creation. Express is tightly coupled with Node.js, facilitating the efficient development of web servers by offering a layer built on top of Nodes capabilities without obscuring its features. Express offers streamlined route handling and server setup, encouraging reusable and modular code [17, 18].

Material-UI (MUI)

Material-UI enhances the user interface design, through utilizing Google's Material Design principles with the component based approach of React, the combination provides heavily customizable, visually pleasing component styling. The library offers user friendly, relatively easy to implement styling, which is perfect for the application's needs [19].

Axios

Axios is a promise based HTTP framework which works in tandem with Node.js and Express.js. The library simplifies the execution of HTTP requests, streamlining code and providing an efficient, manageable service. Axios relies on the promise ensures the HTTP requests are not intrusive to a users experience, allowing Arcanium to fetch data seamlessly. It also provides invaluable features such as JSON data transformation, and considering that Arcanium heavily uses JSON-based APIs, this technology is crucial to the project [20].

3.1.5 Technology Comparison

Despite outlining the technologies chosen for the project technological stack above, it's important to compare these technologies to their competitors, and rationalize why exactly these technologies were chosen. The following technologies were identified and analyzed as potential options for development:

JavaScript vs. TypeScript

- **JavaScript** is a technology that had some familiarity from previous development endeavors. JavaScript is a relatively straightforward language, but has vast amounts of documentation and support, as well as a significant ecosystem.

- **TypeScript** is a superset of JavaScript, and introduces static typing which offers advantages in code debugging and robustness. It does, however, introduce a layer of added complexity to the development process.

While TypeScript would have been a respectable choice for Arcanium, JavaScript was chosen as the main language due to its simplicity and familiarity.

React vs. Angular

- **React** offers a minimalist, view based approach, offering freedom in choosing additional libraries for other functionalities if needed. React offers a component based approach, which promotes reuse of code and modularity, and also offers vast community support and libraries.

- **Angular** is more of a fully fledged framework, based on the MVC pattern. It is a more opinionated framework, trading flexibility for a more comprehensive approach. Angular is powerful, but more complex overall.

React was chosen as the JavaScript framework to go with, as it offered more flexibility, ease of integration and overall had more relevant libraries to Arcanium.

Node.js vs. Other Backend Technologies

- **Node** excels in building efficient, scalable backends for JavaScript applications. The existence of npm offers a huge bonus to Node, as the package manager is robust and useful. Utilizing Node also introduces the advantage of a singular, unified codebase in one language, and enables the use of popular technology stacks such as the MERN stack.
- **Other backend technologies** were considered, such as Django (Python) or Ruby on Rails (Ruby), as they offer detailed feature-sets, and a more structured approach to web development.

Overall, Node.js offers too many advantages for anything else to be considered. A unified codebase, built in a singular language, with the addition of npm is a massive bonus for Arcanium's development.

3.1.6 Overview

The main technology stack of Arcanium consists of JavaScript as the main programming language, React, Node.js and Express as the primary frameworks, with other libraries such as Express, Axios, and Material-UI being utilized to complement the primary technology stack. This strategic decision of technologies was motivated primarily due to the efficiency, flexibility, and alignment with project objectives, but also to ensure that the project can utilize the robust MERN stack, which provides the application with full-stack coverage [21].

Category	**Technologies**
Main Language	JavaScript
Main Frameworks	React, Node.js
Other Frameworks	Material-UI, Axios, Express

Table 3.1: Main Technologies Used in Arcanium

3.2 Database Technologies

The choice of the database for Arcanium was of utmost importance. The application was envisioned to utilize database technology heavily, as the application will

store numerous different collections of objects. Each user will have a specific ID associated with them, and each user will be able to save data to their ID, with this data being able to be fetched and displayed in various parts of the application. The following data is the envisioned data associated with each user:

- **Characters**: Each user will be able to create numerous characters. These characters will be complex, with a lot of different information being stored about each character such as class, race, images or statistics.
- **Stories**: Users will be able to create dynamic stories or scenarios through the use of generative artificial intelligence. These stories will be saved per user, able to be gone back to and viewed at any point.
- **Friends**: Users will be able to friend other users of the application, therefore, these relationships will be detailed in the database.
- **Messages**: Each user will be able to have messages with friends, these messages will be stored in the database for each user.
- **User Profiles**: When a user logs in, it's conceptualized that a user will be able to set their information, which will be then be saved to the database for friends to view.

3.2.1 Initial Considerations

The initial consideration for the project's database architecture was MongoDB. Envisioning using the MERN stack from early in development, this seemed like the logical choice. First off, MongoDB's document-oriented architecture suits the sort of data that Arcanium will save. MongoDB stores data in JSON-like objects, and considering our prospective usage of JSON-based APIs, this would be ideal, as it maintains consistency of data types across the application. The scalability offered by MongoDB, as well as the ease of integration with other, previously discussed technologies were key considerations [22].

3.2.2 MongoDB Atlas

Upon further inspection, it became clear that MongoDB Atlas, MongoDB's cloud managed database offered all the advantages that local MongoDB does, while also introducing several other specific key functionalities. Firstly, MongoDB Atlas significantly reduces the overhead cost of operating the database. Management, configuration and scaling are handled by MongoDB themselves, which makes the initial setup, and subsequent scaling much more efficient and less time consuming. This aspect is particularly useful for a solo developer, where resource and

time optimization is especially important. Atlas also offers built in support for automated backups, ensuring data availability. Finally, the ability to monitor performance through the Atlas management console provides valuable insights into the database's resources [23].

Feature	**MongoDB (Local)**	**MongoDB Atlas**
Management	Requires manual setup, configuration, and maintenance.	Fully managed with automated setup, configuration, and maintenance.
Scalability	Manual scaling, which can be complex and time-consuming.	Easy and automatic scaling, managed by MongoDB themselves.
Security	Basic security features, personal configuration needed for advanced security.	Advanced security features included, such as encryption and network isolation.
Backup and Recovery	Manual local backup, requiring personal setup.	Automated backups and recovery features.
Performance Monitoring	Possible with third-party tools and manual setup.	Built-in performance monitoring.
Overhead	High; requires significant developer time for database setup and management.	Low; most operational tasks are automated or simplified.
Cost	Variable; dependent on local setup and personal hardware.	Predictable pricing models with a pay-as-you-go option, including a free tier (512mb storage).

Table 3.2: Comparison between MongoDB and MongoDB Atlas

The table above summarizes key considerations between local MongoDB and MongoDB Atlas. The differences in key features justify the selection of MongoDB Atlas for the application's needs. The paramount consideration in the development of Arcanium is to avoid costs, and considering the type of data the application stores, and the generous free tier, MongoDB Atlas is clearly a suitable choice.

Integrating MongoDB Atlas into Arcanium began with a relatively straightforward process of creating an account a configuring a cluster with the setup instructions provided by MongoDB. MongoDB Atlas provides detailed instructions on how to implement the service with Node.js, which is one of our key technological frameworks, these guidelines made setting up the MongoDB Atlas instance straightforward.

3.2.3 The Completed MERN Stack

Now that we have covered our main technological stack, including JavaScript, React, Node, Express, and MongoDB, we have detailed the key components of the MERN stack, which will be the technological stack that drives Arcanium's functionality.

The synergy between these technologies not only accelerates development but also enhances the application's scalability and maintainability. MongoDB Atlas's cloud platform simplifies database management and scales alongside Arcanium's evolving requirements. React and Node.js, supported by a vast ecosystem of tools and libraries, enable implementation of advanced features and optimizations with minimal overhead, while express plays a pivotal role in structuring the application's backend [24]. JavaScript is utilized as the main programming framework which ties it all together.

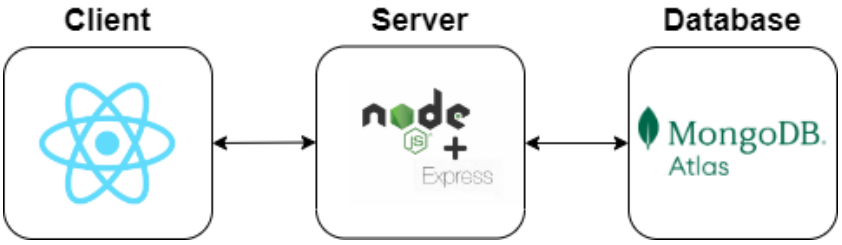


Figure 3.3: Arcanium's MERN Stack Architecture

3.3 Authentication

A critical component of any modern application is robust authentication. Users should be able to sign in through an intuitive and simple experience. Authentication systems enable this, as well as comprehensive security throughout the entire application. Initial considerations were to build a custom authentication system, and save this data in the applications MongoDB Atlas cluster, however, after research into authentication systems, solutions such as Auth0 provide more inclusive solutions to authentication. Auth0 is created by Okta, and is an efficient solution to provide powerful authentication. Auth0 is the chosen authentication for many large companies, such as OpenAI, Pfizer, and AMD [25].

3.3.1 Auth0

Auth0 follows several key principles that make it an ideal solution for application security and authentication:

Conceptual Foundations

- **Universal Authentication**: Auth0 provides access to standard authentication protocols such as OAuth, SAML, and more. This enables developers to choose the protocol that they require for their application, and also allows the integration of social media providers, facilitating users to sign in how they wish.
- **Customization**: Auth0's platform can be customized thoroughly through the management dashboard, allowing the authentication for Arcanium to be custom tailored to the application. Auth0 allows developers to define custom rules or actions, authentication flows, or UI themes.
- **Experience and Integration**: Auth0 emphasizes simplifying the approach to authentication where possible. It provides SDKs and APIs which can be seamlessly integrated into existing applications, with support for frameworks such as the MERN stack, which Arcanium will utilize [26]. Auth0 is also a cloud-based service, which removes the struggle of managing authentication infrastructure, encouraging scalability.
- **Security**: Auth0 provides extensive security features, such as multi-factor authentication, anomaly detection, and attack protection. It also complies with important compliance protocols, such as ISO security standards [27].

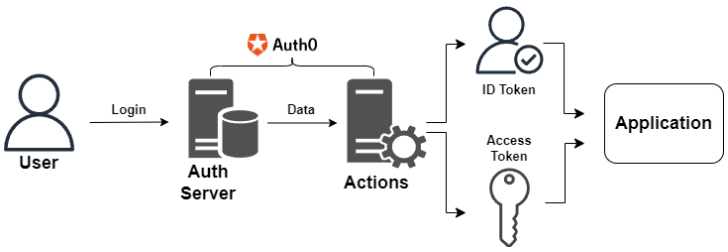


Figure 3.4: Auth0's Authentication Flow

Overall, integrating Auth0 into Arcanium for robust, maintainable and scalable authentication provides the application with the security it needs. Auth0 offers all the features the application requires, while also providing comprehensive documentation and tools [28].

3.4 API Integration

Integrating Application Programming Interfaces into Arcanium is essential to make the application dynamic, informative, and to ensure it follows the best develop-

ment practices by not storing large amounts of data locally. APIs serve as pathways between software platforms, allowing developers to use information or tools from other locations in an efficient and meaningful way. Utilizing APIs in Arcanium is especially essential, due to the nature of Dungeons and Dragons and the information associated with the game.

3.4.1 Open5e API

In conceptualizing Arcanium, one consideration that was immediately identified was the management of the vast amounts of information that the game has associated with it. There are hundreds of races, classes, locations, spells and items, and rules, all with thousands of words of information associated with them. The 5th edition of the player's handbook, an essential D&D resource, contains over 300 pages in an encyclopedia format, resulting in a resource book containing tens of thousands of words alone [29]. Considering there are many other resource based books for the 5th edition of D&D, containing this information within a digital context would be difficult without an API based approach.

Open5e API resolves this concern by providing open source, programmatic access to all the D&D based resources that Arcanium requires [30]. The API is based on the Django REST framework, which is a Python based web framework for building robust APIs. This API will serve as the foundation for Arcanium's development, as the application will utilize this resource to get information for various parts of the application such as resource searching and character creation.

API Features

- **Content Filtering**: The API allows Arcanium to filter resources dynamically, through various properties. For example, filtering monsters by challenge rating or filtering spells by spell level.

- **Resource-specific Searches**: The API offers case insensitive and partial word-search matching. This can be utilized through auto-completing results or providing infinite scroll mechanisms.

- **Comprehensive General Search**: The API allows searching across the entire content base, providing a mechanism to search for multiple resource types.

- **Result Presentation**: The API provides pagination and ordering features, allowing the application to effectively manage how information is presented to the user.

These features are ideal for Arcanium's prospective use of content. Utilizing filtering, resource specific searches and presenting the results in a digestible manner will increase the applications user experience significantly.

3.5 Character Creation

When researching the character creation process, it was essential to analyze the aspects of the traditional character creation process in Dungeons and Dragons. This part of the application should be an interactive, step-by-step process. Utilizing React and Material-UI, we can define these steps and make them a visually pleasing process. Each step should be engaging and informative, providing a detailed view of the potential selections the user will make.

3.5.1 Defining the Steps

According to the player's handbook, the steps of the character creation process can be defined as follows [29]:

Category	**Description**
Race	Heritage of the character, influencing skills, abilities, and interactions.
Class	Profession dictate the character's primary abilities and skills.
Background	Offers skills and a narrative foundation for the character's history.
Ability Scores	Six core attributes (Strength, Dexterity, Constitution, Wisdom, Intelligence, Charisma) impacting world interactions.
Skill Selection	Proficiencies affected by ability scores, defining expertise areas.
Spells	For magic users, detailing available magical spells.
Equipment	Starting gear based on race, class, and background choices.
Character Details	Personalization options like name, appearance, and alignment.

Table 3.3: Characteristics in D&D Character Creation

Above, we can see the definitions of each of the traditional steps of the character creation process. Each of these steps have many considerations around them, so

it's essential to theorize and research the best way to implement each respective step in the character creation process:

Race, Class and Background

The initial steps involve choosing a race, class and background. The most logical way to implement a selection process for these steps is to utilize the Open5e API. The API has parameters for each of these categories, with detailed explanations and other attributes associated with them. Providing players with a detailed view of these selections, before allowing them to select would be intuitive and user-friendly. The race, class, and background choice also has additional impacts. These selections provide the character with increased statistics, more choices in skill proficiency's, or the ability to cast spells. These choices should therefore come at the start of the character creation process, to ensure that these bonuses are properly applied.

Ability Scores and Skill Selection

Determining a character's ability scores and skills are essential parts of the character creation process. Ability scores are traditionally defined in a few different ways, including predetermined point arrays, point buy, rolling dice, or hybrid methods. After consideration, here are the most user-friendly approaches:

- **Point Buy**: A character starts with a base value in each statistic, usually 8 or 10, and is allowed to increase specific characteristics by spending these points. Traditionally in the 5th edition, each character is given 27 points, with different ability scores have different point costs depending on how high they are, with higher scores costing more points.
- **Rolling Dice**: Players roll dice to generate random ability scores. This method typically involves rolling a set of 4d6 dice, and adding the three highest rolls to determine each abilities score, with the lowest number being dropped.
- **Open Buy**: Since D&D is a flexible game, players should be able to choose whatever points they want, if they want. This process is defined by the dungeon master, but the option should be included if needed.

Including these options should allow players to choose their ability scores in any way they wish. Ability scores impact the skill selection too. Each skill is categorized to derive from each ability score, for example, dexterity influences stealth, sleight of hand and acrobatics, while charisma influences deception, intimidation,

performance and persuasion. These skills are used to interact with the game world, so a character who has high charisma would tend to be proficient at deceiving people, or intimidating them. In the 5th edition of D&D, there are 18 total skills. A user should be able to select their skill proficiencies later in the character creation process, so the ability scores are properly applied.

Spell and Equipment Selection

Based on the chosen class, a character will be able to select specific spells or equipment. For example, a barbarian is not a magical class, so it doesn't get access to spells, but a wizard is, so this class does have access to spells. Therefore, nonmagical characters should not get the option to select spells. Magical users, at level 1, typically get access to a set amount of cantrips (basic spells) and level 1 spells, which are more advanced and cost resources to use. Each class will have different selections here, so these classes should be able to see only their own spells they are allowed to choose. For example, *eldritch blast* is a spell unique to warlocks, so only this class should be allowed to choose this spell in the creation process. Each class has a unique set of equipment at level 1, so based on the class selection, each character should see unique options in the equipment selection phase.

Character Details

This step should allow players to personalize their characters by choosing a name, and defining their physical attributes such as height, weight, and other physical characteristics. In an attempt to not make this step consist of numerous textfields, the applications should employ different mechanisms for choosing these attributes. For example, sliders for height or colour pickers for eye colour. Users should also be able to upload an image of their character here.

Image Storage

An additional consideration to keep in mind is that MongoDB is not an ideal place to store images. To get around this, Arcanium must utilize another approach to store images. An alternative approach is to utilize an S3 bucket from AWS. This approach would mean that when an image is submitted, its sent to the S3 bucket, and the URL of the image is stored in MongoDB instead. This approach is more complex, but ensures an efficient way to manage images. User images, through social connections, are managed through Auth0, so we don't need to worry about storing these images in MongoDB.

After defining the character creation process steps and other considerations like image storage, we can implement the character creation process smoothly through the use of the Open5e API, React, the MERN stack and the S3 bucket.

3.6 AI Integration

Artificial Intelligence in Arcanium is one of the key areas of the project. Providing users with dynamic, verbose and interesting interactions would significantly increase the user experience of the application. Before considering models and comparing them, it's essential to define our application's needs in this area:

- **Model Type - Text Based**: The type of AI model that Arcanium needs is text based. The application requires a model that is coherent, intelligent, and capable of reading and interpreting users needs, even if the users questions are not correct structurally or grammar-wise. This is especially important in the context of Arcanium and Dungeons and Dragons, due to the sheer amount of information, it's inevitable that users will misspell something or refer to something incorrectly. The model must be capable of adapting to this. In the spirit of Reacts component based architecture and encouragement of reuseability, it would be beneficial to choose a model that could perform both implementations functions without the need for complex changes.
- **Chatbot Implementation**: This implementation is defined as a chatbot that assists users by answering any questions they have. These questions can be anything, for example D&D rules, tips, or explanations about spells to name a few. The chatbot should be configured to respond thoroughly and dynamically, and always be available to users at every point in the applications flow.
- **Generative Story Implementation**: The other AI implementation is a generative story function, capable of generating unique and interesting stories, based on user input and prompts. These responses should be detailed and engaging. The user should be allowed to do, say, or input specific story prompts to guide the conversation in a specific direction. Users should also be able to generate these stories with the context of their characters being present, essentially operating as both a dungeon master tool and an engaging text based game at the same time, depending on the users needs.

3.6.1 Model Requirements

When selecting a model for the AI integration of Arcanium, it's important to define key factors that can influence the model choice:

- **Model Complexity and Size**: The chosen model must have a balance between complexity and size. It must ideally be capable of generating immersive and dynamic content in real time with little to no hitches or waiting, while ensuring performance.

- **Training**: The ability to train the model on specific data is essential to the application. The model must be able to be trained on D&D based content and have a good understanding of the topic. It has to be considered an expert to be able to answer questions efficiently.
- **Integration**: The model must seamlessly integrate with Arcanium's technology stack. This means that the model must be able to be deployed in the application without the need for compromises in other places. If the model is too large in size, this could affect deployment options or performance throughout the entire application. Ideally, the model is capable of being trained for both the chatbot integration and the generative story integration.
- **Interactivity**: The model should be capable of interacting fluently and naturally with users. The model should be considerate of spelling mistakes, structurally incorrect sentences, and not answer questions that are outside the context of Arcanium of Dungeons and Dragons as a whole.

3.6.2 Model Comparison

Considering the criteria for an ideal model defined previously, the following models are considered potential candidates for the role. Comparing these models will allow Arcanium to choose the best possible solution for this important area of the project.

OpenAI's GPT Variants

OpenAI's GPT (Generative Pre-Trained Transformer) variants, specifically the later versions such as GPT-3, GPT-3.5 and GPT-4 offer a balance between computational complexity and efficiency. OpenAI's architecture allows for training and fine tuning of models, making it possible to train the models on niche or specific D&D related data. The models also see extensive use, so they they tend to have support for most platforms. GPT models also excel on understanding and generating natural language, so they are very capable of handling detailed user inputs, questions, spelling mistakes and other errors. An advantage this model type has over others is access to the OpenAI API. This technology offers easier integration, especially into web-based platforms..

Custom TensorFlow Based Models

Custom TensorFlow based models offer unparalleled complexity, but to reach this stage, a significant amount of development time and expertise would needed. These

models can also be trained on D&D related datasets, and the extent of interactivity would depend on the quality of the training data and the time spent training. However, seeing as these models typically use Python, the integration would prove challenging without the use of external libraries, considering the technology stack of Arcanium is web-based and using the MERN stack. This approach offers the most potential for control, at the cost of complexity, development time and relative file size.

BERT

BERT (Bidirectional Encoder Representations from Transformers) are highly skilled in understanding context and natural human language in text, and can be fine tuned on D&D content. Typically, BERT models are very powerful, but require substantial resources to integrate into an application like Arcanium. BERT models excel at generating answers to queries, so this model would excel at the chatbot section of the AI integration, but would struggle in generating dynamic and creative content for the other portion of the implementation [31].

In conclusion, custom TensorFlow based models offer the highest complexity and control, but require significant development time, integration help and fine tuning. BERT models offer a flexibile, powerful approach with the best question answering capability, which would suit some of the AI integration of the application, but would struggle to generate dynamic content. OpenAI's GPT variants offer the most flexible, robust options for Arcanium's AI needs. The models operate better than the others, even without fine tuning [32], and have other advantages such as ease of integration, better generative abilities and API access, making the models more suitable for Arcanium's AI requirements [5, 4].

Model	**Complexity**	**Training**	**Integration**	**Interactivity**
GPT Variants	Efficient, balanced	Adaptable, straightforward	Simplified via API	Natural, versatile
Custom TensorFlow	Detailed, complex	Extensive, customizable	Requires extra tools	Quality and time dependent
BERT	Contextual, understanding	Complex, sensitive	Relatively straightforward	Excels on queries

Table 3.4: AI Model Comparison

Chatbot

In Arcanium, the chatbot functionality will be utilized to help players throughout the application. It's envisioned to be always accessible, in the case the user has questions at any point during their time while accessing the application. Utilizing a model from OpenAI's models will ensure that the application has a powerful, always accessible chatbot to answer any relevant question, ensuring consistent responses with the natural language responses expected from OpenAI's models.

Generative Story

Generative storytelling will provide Arcanium users with a dynamic resource to develop stories for use in their campaigns, generative scenarios, or stories using the context of their created characters. Utilizing OpenAI's models, this process will assuredly have creative, consistent and entertaining responses while maintaining coherency.

3.6.3 Challenges and Concerns

When utilizing artificial intelligence technology, it's important to talk about the ethical concerns surrounding such a technology. When developing an application such as Arcanium, it's especially essential to ensure that the AI in the application itself should not encourage any ethical issues, such as biases, harmful content or misuse of data. These considerations should be kept in mind during the development of Arcanium. Ethical considerations of artificial intelligence is a widely covered topic [33], however, OpenAI's models have pre-defined measures in place to ensure that any ethical violations aren't exposed during usage.

3.7 Social Features

The social features in Arcanium were conceptualized to complement the fact that Dungeons and Dragons is inherently a collaborative, heavily social game. The defined features from the papers introduction are conceptualized to facilitate communication and promote a social community within the application. Allowing Arcanium users to connect and organize their campaigns together through a friends system and real-time chat functionality will provide players with a meaningful experience.

3.7.1 Real-time Chat

The real-time chat functionality is the basis of Arcanium's social system. Allowing players to keep their D&D related messages contained in the application promotes the platform to be a unified solution.

Initial considerations for the real-time chat functionality were between the WebSocket API, and Socket.IO, two technologies that integrate well with Arcanium's current technology stack. WebSocket provides full-duplex communication through communication channels which is suitable for a real-time chat application. This approach offers full control over the messaging and data paradigms and protocols, but requires significant development time and resources to accomplish the same thing that Socket.IO provides initially, without specific complex configuration. After researching Socket.IO, it became clear that this technology was perfectly suitable for the needs of the application. Socket provides a high level API, built in features such as auto-re-connection and rooms which can be used to broadcast messages. Socket also provides more scalability and management features which would be useful for the application's user experience.

Socket.IO

Socket.IO provides Arcanium with a comprehensive real-time foundation for the chat functionality of the application. Offering a high level API increases the integration process significantly. Socket.IO has a few key conceptual features that justified it's choice:

- **Event Driven Architecture**: To facilitate bi-directional, real-time conversations, Socket.IO utilizes event driven architecture, focusing on highlighting events such as *message sent* or *message received*. Event driven architecture is especially useful in the development environment [34].
- **Auto-Reconnection**: This feature ensures that if a user is disconnected from a room due to unforeseen circumstances, such as network interruptions, the chatroom can be maintained without manual intervention.
- **Room Management**: Socket.IO offers room management, ensuring that an application can utilize private chat rooms, group chat rooms, and more [34].
- **Scalability**: With Socket.IO, scaling chat room availability and other resources is a relatively straightforward process. Combined with the fact that Arcanium plans to store user messages in its database, the consideration for scalability is important.

3.7.2 Friends and Campaign System

Both the friends and campaign system are fundamental to the social aspect of Arcanium. The idea is to create a linked experience, where users can friend players, invite them to campaigns, view their profiles, and chat with them in real-time, which results in a cohesive social experience.

Linking the Systems Together

To store the information, MongoDB Atlas will be utilized. Data models will be defined for friends, messages and campaigns, with this information being stored in the database for retrieval later. For each user to have specific data associated with them, they will require a unique identifier. This identifier will be generated by Auth0 when the user logs into the application for the first time, subsequently, when a user attempts to store information, this identifier will be attached to the data to ensure each piece of data in the database is associated with the relevant user. When a user submits a character, or joins a campaign, this information will be visible on their profile for other friends to see. This system provides a comprehensive, linked experience.

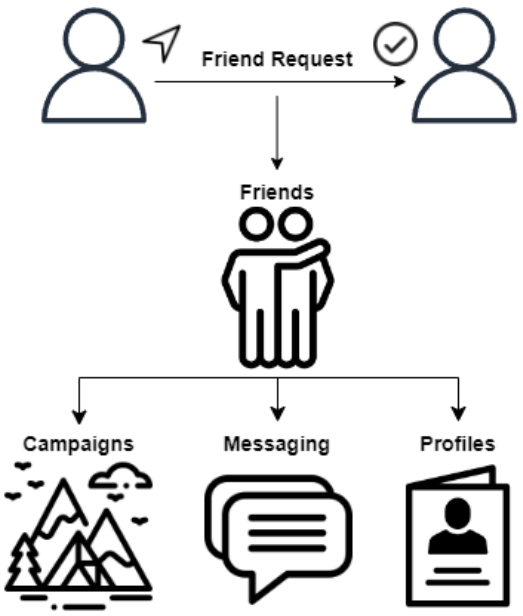


Figure 3.5: Arcanium's Social Interactions

3.8 Revelations and Summary

After a thorough technological review, this section provides an extensive review of the technologies considered, challenges faced during the research period, and a reflection on the process itself before summarizing the technologies chosen through the technological research.

3.8.1 Technological Challenges

The main purpose of the technology review was to define the technological foundation of Arcanium, as well as to provide insight into why each technology was chosen, while providing a conceptual explanation of each relevant technology. However, another main purpose of the review was to identify key areas that were challenging, as it's important to acknowledge these limitations to ensure a smooth development cycle. Considering the scope of Arcanium, it was inevitable, and expected, that some challenges would be faced. The technology review revealed some of the following as key challenges:

Virtual Tabletop

The virtual tabletop was one of the initial key features that was defined for Arcanium. This feature was conceptualized to be an innovative location where users could roll dice, utilize maps for campaign play and more. These features were initially considered to be useful in enhancing the user experience, and encouraging engagement with the application in a more comprehensive way. Despite this feature being one of the foundational features, after investigating the technology behind it, several challenges become apparent:

- **Performance**: Upon initial investigation, both Canon and Three.js were found as suitable libraries to potentially build the functionality. After conceptual tests, this revealed that these technologies were relatively resource intensive, and caused issues with the foundation of the application.
- **Complexity and Compatibility**: These technologies are complex and challenging to implement in the context that Arcanium needs. Both of the technologies required 3D rendering knowledge to create an interactive tabletop. They also introduced another concern, as testing initial concepts with mobile devices introduced some unforeseen issues.
- **User Feedback**: After conducting a survey, seen in the research methodology, it resulted in the virtual tabletop being consistently ranked lowest in terms of desirability among new and experienced players.

- **Scope and Coherence**: Upon attempting to create a visual conceptualization of the virtual tabletop, it became apparent that the scope and coherence of this feature wasn't adequate for the application. Other features have a clearly defined reasoning and function, but this feature felt lacking in terms of vision. Arcanium is intended to be a supplement for physical, in person Dungeons and Dragons sessions, and this feature sought to replace one of the key aspects of D&D: the physical social experience.

These considerations led to a re-evaluation of the project's key technologies. Due to these constraints, and a need to prioritize other features, it was deemed acceptable to cease development on the virtual tabletop feature, with the intention to revisit the feature later in development.

Image Saving

After identifying that MongoDB's file structure was not suitable to save images in, an investigation into alternative technologies was conducted to assess the most suitable option. It became evident that the most appropriate way to store images was to save them in Content Delivery Networks (CDNs), which are designed for media storage. Firebase storage, Cloudinary, and AWS S3 were all considered, but this method had some additional complexity associated with it. Cross-Origin Resource Sharing (CORS) were expected to be an issue in development, as these images would be stored in a different domain. This also introduced security concerns, so choosing the right CDN was important to maintain a secure application. The main issue with this approach is cost. CDNs offer a great service, but beyond the free tier, these services can get somewhat expensive. Some of the services also hold a deposit for some time when initially signing up, which is something that ideally was to be avoided.

Cost Management

Arcanium's development strategy heavily used cloud-based services and APIs. This approach was chosen to leverage maximum scalability and flexibility, however, CDNs, authentication services, cloud-based database architecture and third party APIs all have potential for incurring costs. Given the reliance on cloud-based services, it was important to keep these considerations in mind during the development process. Too many requests to a service may incur charges, so testing mindfully and using services diligently was important.

3.8.2 Summary

To summarize, the technology review of this dissertation provided a comprehensive overview of the programming languages, frameworks, and other technologies selected for development. The review aligned with the context of the introduction and methodologies, by utilizing these sections to inform the decision making process behind both technological research and technological selections.

Programming Languages and Frameworks

JavaScript was chosen as the primary programming language, due to its flexibility, extensive community support, and suitability for both client-side and server side communication. React was selected as the main frontend framework, due to its component based architecture and efficient rendering. Node.js and Express.js were chosen due to forming a comprehensive backbone for the server side logic of the application.

Database Technologies

MongoDB Atlas was chosen for it's flexible, document-oriented, JSON based model, as well as its scalability due to being a cloud-based platform. Database architecture was envisioned to be utilized extensively across the application.

Authentication

Auth0 was chosen as the authentication architecture for the application, due to its security features, ease of integration through API access, and extensive documentation. This choice ensures seamless, secure access across the application.

API Integration

Open5e API was covered thoroughly, as it was envisioned to be used significantly across the application's development. The API provides JSON based information, compiling thousands of pages of resource books into a digital context, all while following the Open Game License (OGL), ensuring fair use of data.

AI Integration

OpenAIs GPT models were chosen as the basis of AI integration in Arcanium, due to their adaptability and suitability for both chatbot based AI and generative story based AI. This technology will allow the application to leverage AI efficiently, ensuring compliance of ethical standards.

Social Features

Socket.IO was chosen as the framework to facilitate real-time chat functionality, with friends and campaign structure interactions being outlined to provide clarity on the linked nature of the social system in the application.

Miscellaneous

The technology review covered the context of the character creation process, providing insight into how data will be stored and fetched. Axios and Material-UI were also explained, with these technologies being used to communicate with backend structure and style the application respectively. It also explained the challenges that the review uncovered, such as the decision to move away from the virtual tabletop feature, and the challenges associated with image saving and cost management.

Technology	Purpose
JavaScript	Primary programming language for both client-side and server-side development.
React	Frontend development framework for building a dynamic and responsive user interface.
Node.js	Server-side JavaScript runtime environment to facilitate backend development.
Express.js	Works in tandem Node.js, creates robust backend structure.
MongoDB Atlas	Cloud-based database for storing flexible data, such as user profiles and characters.
Auth0	Authentication and authorization service.
Open5e API	Third-party API integration for accessing D&D content in JSON.
AWS S3	Cloud-based storage service used for image storage.
OpenAI Models	AI models for chatbot functionality and generative story creation.
Socket.IO	Enables real-time, bi-directional communication, facilitating real-time chat functions.
Material-UI	A UI framework for React that implements Google's Material Design.
Axios	Promise-based HTTP client for the browser and Node.js, used for making HTTP requests.

Table 3.5: Summary of Technologies Used in Arcanium

Chapter 4

System Design

4.1 Introduction

This chapter delves into the design and architecture of Arcanium. The system design and architecture of the application was defined during the technological review, where conclusions were made about the most suitable technologies for the needs of the application, after a thorough review and comparison. This chapter heavily uses System architecture diagrams, UML diagrams and flowcharts. By elaborating on the architectural framework and foundation of the project, the chapter aims to explain the *how* of the project.

4.1.1 Design Philosophy

The design philosophy of the application was heavily influenced by the project requirements set in the introduction, the methodology approaches and the insights from the technological review. The architecture has been developed to make a coherent, consistent and user-friendly experience. Central to the design philosophy are the following key areas:

User Central Design

A key aspect of the application was to ensure that the design was sculpted to be intuitive for both new and experienced players of Dungeons and Dragons. Emphasizing user interface design and relevant tools such as the help chatbot supported this approach.

Scalability

Integrating cloud-based services where applicable ensures that the application follows good software development standards, and is open for extension and scaling if required. Utilizing AWS S3, cloud-based database architecture and APIs supported this area of design.

Seamless Integration

The application was envisioned to be a supplement to Dungeons and Dragons players. Ensuring that the application provides a consistent and meaningful experience was important for the scope and context of the project to be fully realized.

Adaptive and Iterative Development

This principle was followed throughout the project, with reviews of feature scope and context being conducted regularly. It was important to maintain a consistent vision for the project, and change features if required. Listening to feedback, such as in the survey was also important, as it provided insight into feature prioritization which is an important aspect, especially for a solo project where time and resource management is important.

4.2 System Architecture

The architecture of the system covers the foundation of the system, as well as where key technologies exist within the foundation, and how they interact with each other, dictating the scalability, adaptability and coupling of the architecture. This section demonstrates a high level overview of the architecture, as well as a component breakdown to cover the specific internal structure and interactions of the system.

4.2.1 High-Level Architecture Overview

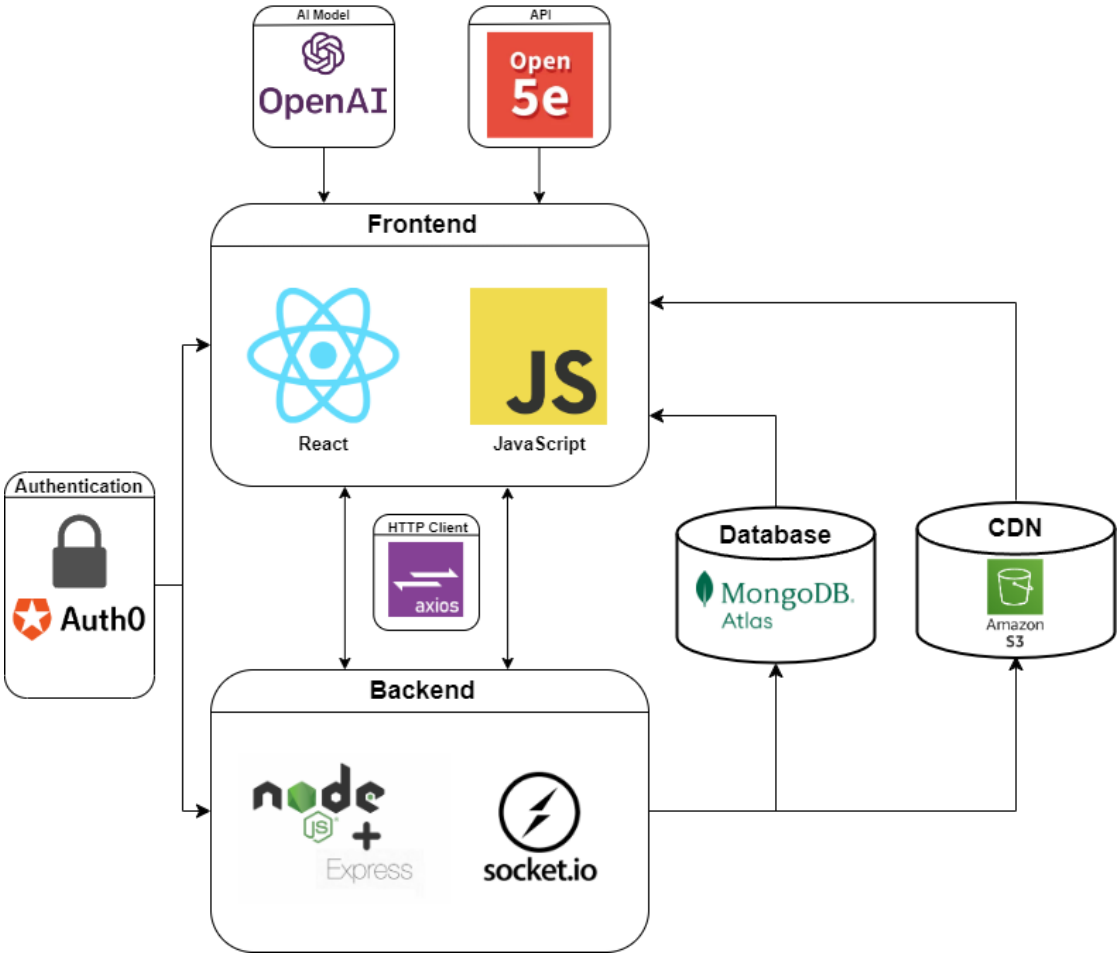


Figure 4.1: Arcanium's Overall System Architecture

As seen in the structural diagram above, the high level architecture overview serves as a map that details the general interactions between each of the technologies in Arcanium, showcasing how user requests travel throughout the system, and how data is managed and retrieved. The diagram presents the technologies labelled with their functionality, for example, frontend and backend. The frontend layer, powered by React and JavaScript, functions as the application's view and main functionality logic. The backend layer represents the server side of the application, which is maintained by Node and Express, as well as Socket.IO for real-time chat communication. Axios facilitates HTTP requests between the frontend and the backend. The authentication of both the frontend and backend are managed by

Auth0, which maintains robust security across the application. MongoDB Atlas and Amazon S3 serve as the storage section of the application, with MongoDB acting as the primary database for the application, with the S3 bucket being utilized for proficient image storage. OpenAI's models are utilized through their API, to encourage an efficient and scalable approach to artificial intelligence integration for both the chatbot and generative story features. Finally, Open5e's API functions as the main information section of the application, being utilized by the frontend to search for data and create characters. In the following sections, we will delve deeper into each specific component and detail their architecture.

4.3 Frontend Design

The frontend of the application is what the user sees, so it's important to ensure this section of the application is intuitive and engaging. Each step in the process of the application should be relatively straightforward.

4.3.1 Application Flow

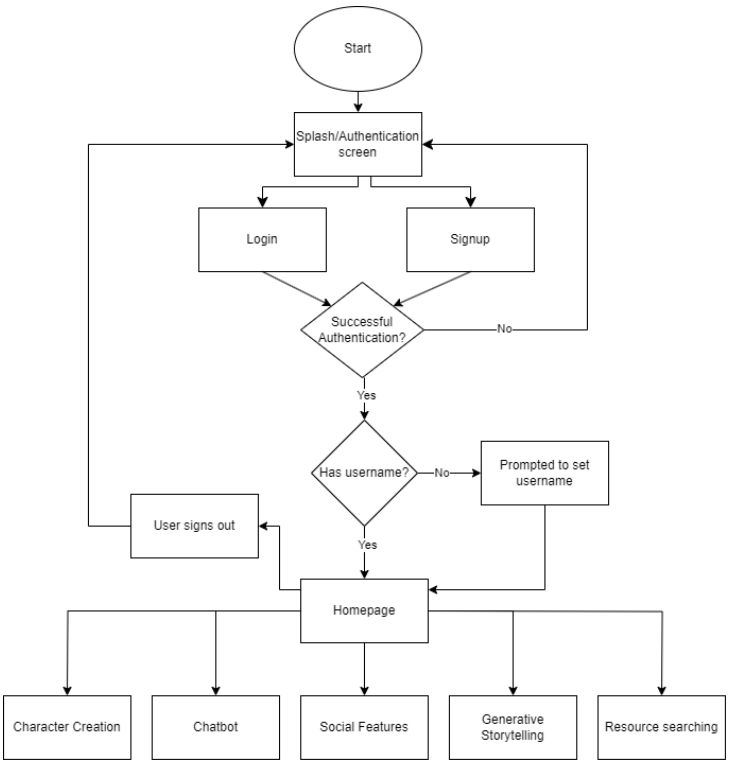


Figure 4.2: Arcanium's Initial Flowchart Diagram

Firstly, it's essential to understand the context of the application flow, which acts as the blueprint of the navigation through Arcanium for the users. The flowchart above defines which routes a user will be prompted with and potentially take during their usage of the application. To access Arcanium, users must login to the application and be authenticated. This is due to the high amount of user-based features which require unique identifiers for each user. Arcanium uses Auth0's defined ID for features such as character creation, real-time chat, and other social features, so having this authentication flow is essential.

4.3.2 General User Interface (UI)

This section will give a general overview of the user interface design. More detailed architectural explanations will be provided per each specific section later in the system design. Note that some of the styling may be subject to adjustments in comparison to the final version of the application.

Theme and Style

The main colour scheme of Arcanium is based in a dark mix of purple, black and grey, with accompanying white text and other visual queues for highlights and tertiary colours. The application's palette was defined as this as it portrays a magical, fantasy-esque feel through this and its fonts and design.

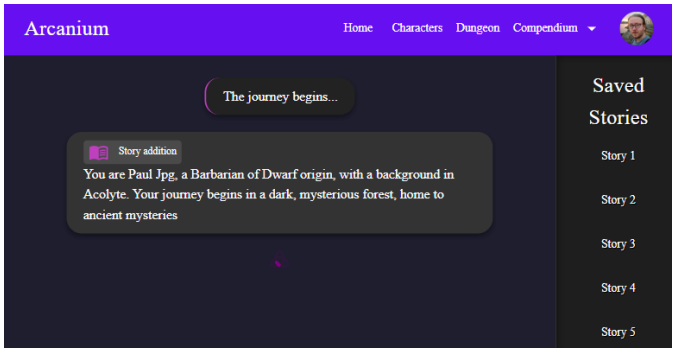


Figure 4.3: Arcanium's Base Theme

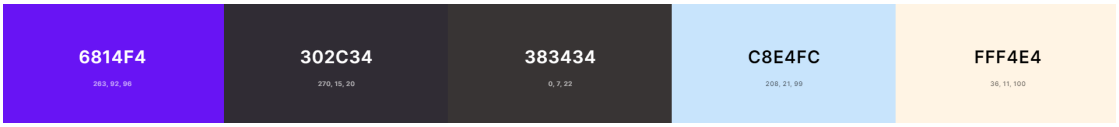


Figure 4.4: Arcanium's Main Colour Palette

Responsive Design

The development of Arcanium ensured that the application was responsive and accessible on mobile. This was done through the use of Reacts component based architecture, which made it straightforward to design responsively. Material-UI also comes with comprehensive responsive based architecture support, so the combination of these libraries enables responsiveness across the application.

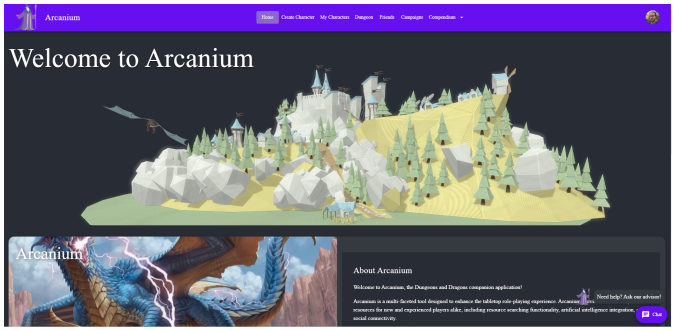


Figure 4.5: Arcanium's UI in Standard View

Figure 4.6: Arcanium's UI in Responsive View

4.4 Backend Design

This section focuses on the backbone of Arcanium, which is the server and database, comprised of Node, Express, and MongoDB Atlas.

4.4.1 Server Architecture

The main technology powering the server-side of the application is Node.js. It's event-driven architecture streamlined developing the server-side architecture, and it heavily benefits an application such as Arcanium which utilizes real-time functionality. Express is used in tandem with Node, simplifying routing and providing useful middleware functionalities.

Node Architecture

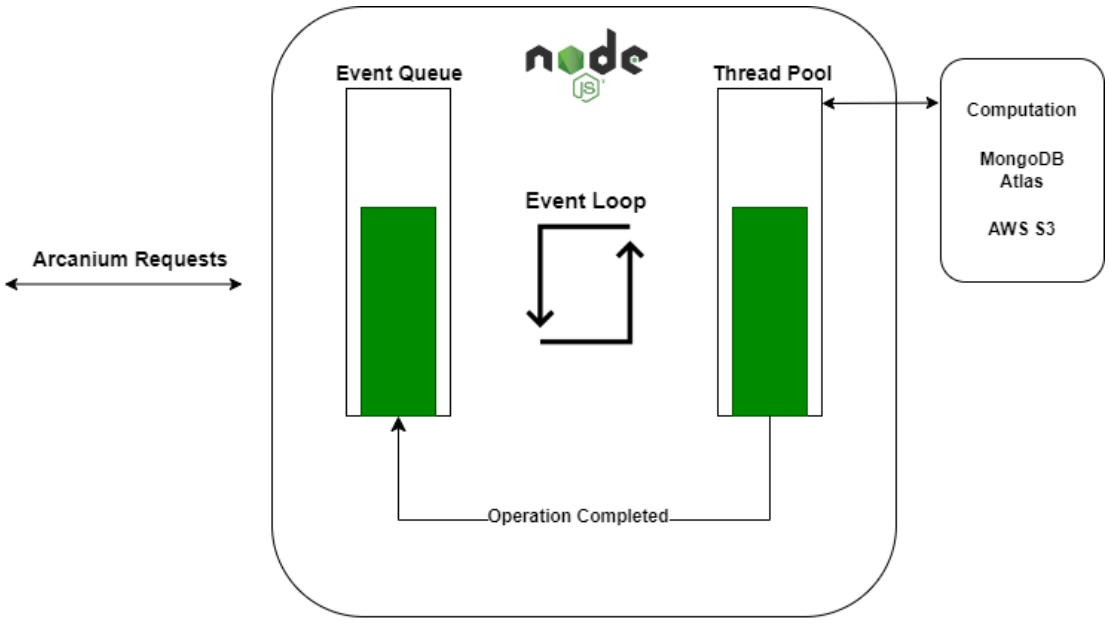


Figure 4.7: Arcanium's Node.js Usage

As shown in the figure above, incoming Arcanium requests are placed in the event queue, the event loop re-polls this queue and forwards the requests to the thread pool for computation, or tasks such as file storage. This also demonstrates Nodes non-blocking operations, showing its efficiency.

Express Architecture

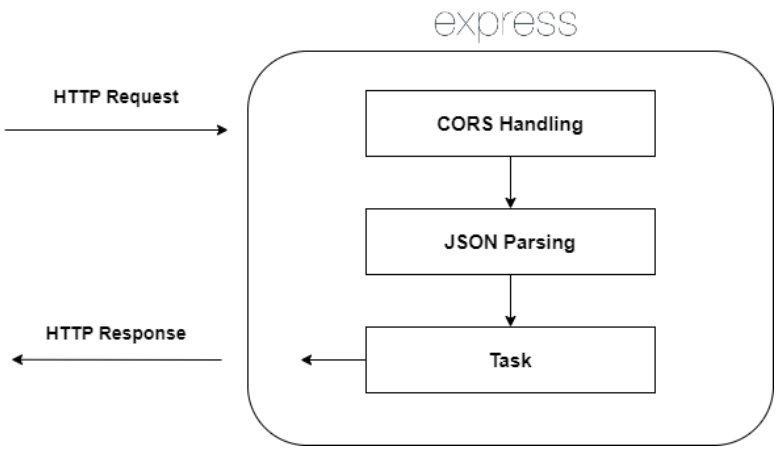


Figure 4.8: Arcanium's Express.js Usage

In Arcanium, the HTTP requests are first sent through CORS handling, JSON parsing (due to the nature of Open5e API data, and then main task is executed, which is a controller or route, before the HTTP response is sent back.

4.4.2 Database Architecture

The database architecture in Arcanium is handled by MongoDB Atlas, which is a cloud-based service for MongoDB. Atlas is referred to as both a Database as a service (DBaaS) and a Platform as a Service (PaaS). MongoDB Atlas, in the context of Arcanium, is used to store user based data such as characters, friends, stories, messages, and profiles. All of these have their own collection in the database. The data is sent from the frontend to the backend to be managed by the Node and Express based server architecture, before it's sent to the database. The entity relationship diagram below shows the collections and how they relate to each other:

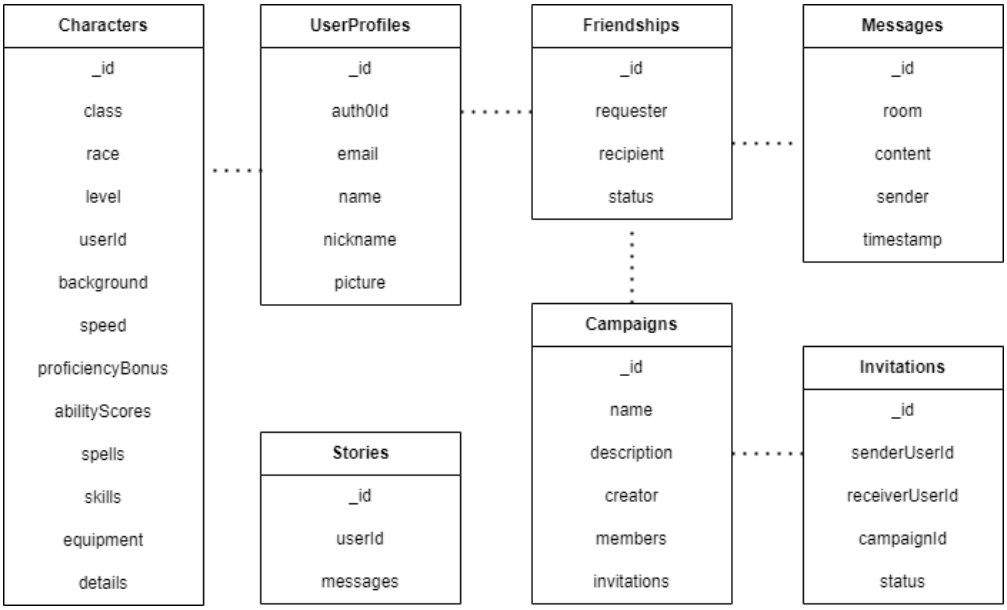


Figure 4.9: Arcanium's Database Entity Relationship

Below is an example of how data would be sent from the frontend to the backend and on to the database, including relevant interactions. A user creates a character on the frontend, which is sent through the backend while being facilitated by Axios. After being handled by Node and Express, the data is sent to the database, and the fields are populated within the relevant collection.

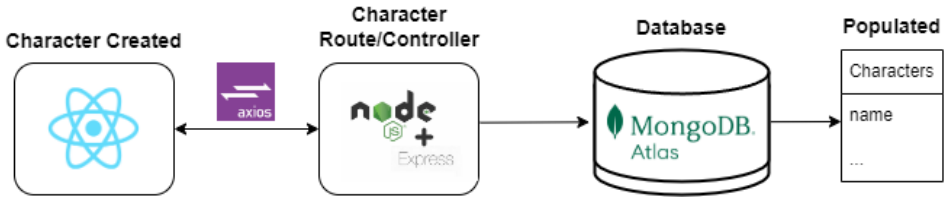


Figure 4.10: Character Created Example

4.4.3 Backend Overview

The backend of Arcanium, as depicted in the diagram below, is mainly designed with efficiency and scalability in mind:

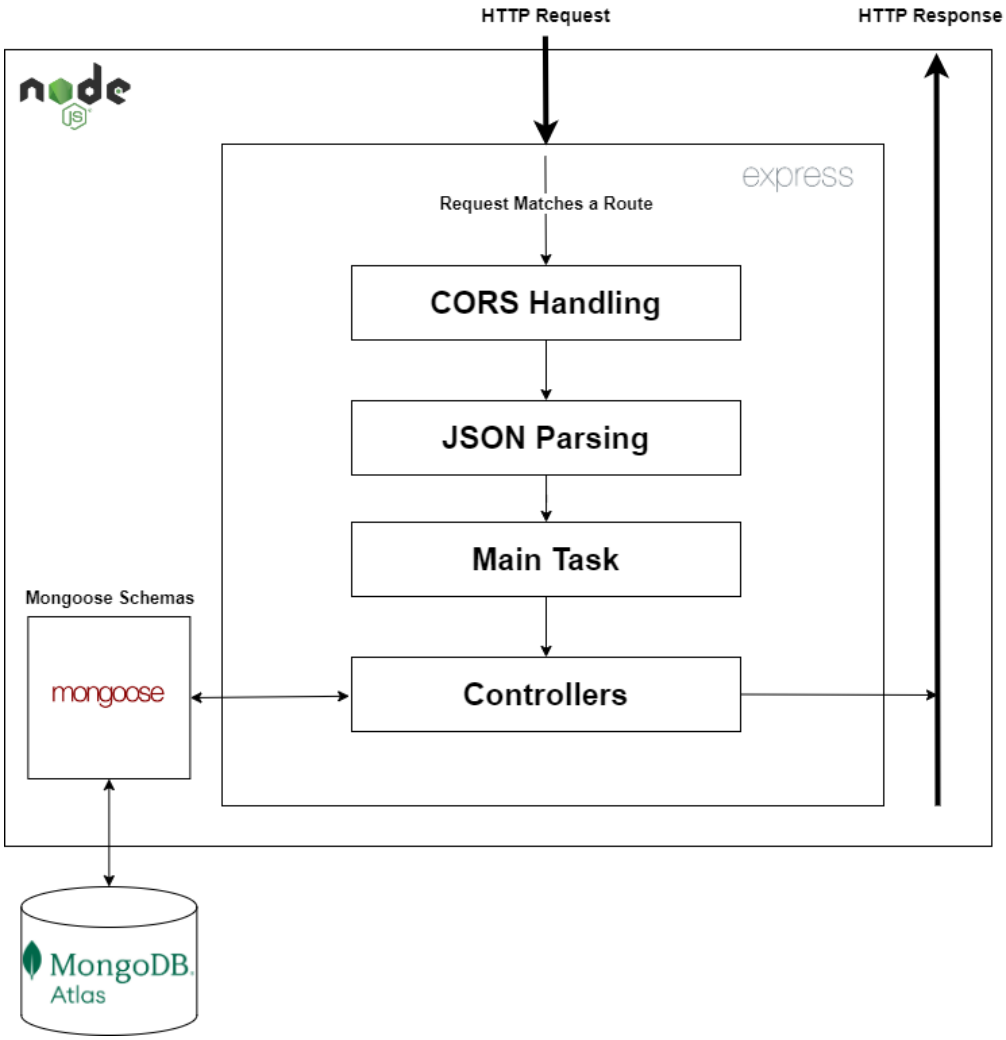


Figure 4.11: Backend Interactions Overview

This diagram details the process of a request through the backend architecture. The request is first received by the server, then Express is used to match a request to a route. For example, if a user was to create a character, this would be matched to the character route, and handled by the logic in the character controller, with the data being defined by the character schema derived from mongoose. The request then passes through the Express handling as previously defined in figure 4.8. Mongoose acts as a bridge between Node and MongoDB Atlas, as it is used to validate data and define object schemas. After the request has been passed through express, validated and managed by the controllers, it is returned as a response, ready to be sent to the database or used in the frontend. This approach represents a modular design, with numerous backend technologies working in tandem to create

a robust and scalable solution for backend interactions in Arcanium.

4.5 Authentication

Arcanium utilizes the Auth0 API for Authentication, with the entire application (with the exception of the landing page) being wrapped in the Auth0 security layer. Not only is this for the purpose of security, but also to assign users a unique identifier. This is essential in Arcanium, as users must be able to store data in the database on a personal level. Each piece of data that is stored in the database is associated with a user, so a decision was made at the start of development to make it a requirement to be authenticated before Arcanium is usable.

4.5.1 Arcanium's Authentication Flow

Earlier in the paper, in figure 3.4, we detailed the general flow that Auth0 typically has. In this section, we will detail how the authentication flow works specifically for Arcanium. Auth0 provides an abstraction of the authentication process, meaning that when users are prompted to login or sign up, they are redirected to the Auth0 domain to complete this process. When this process is complete, they are redirected back to Arcanium as an authenticated user, and they are assigned a unique identifier from Auth0. Without going through this process, users are unable to access beyond the homepage of Arcanium. This abstraction allows for the Auth0 API to manage any typical authentication complexities. It also provides a managerial dashboard which Arcanium utilizes to set rules, define actions and manage users.

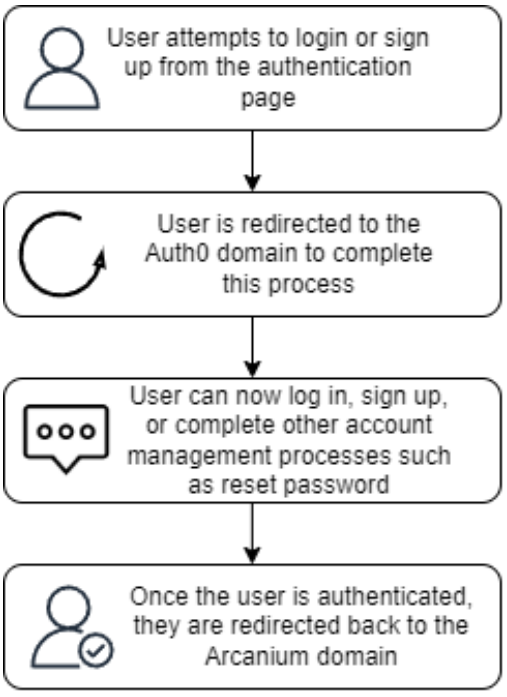


Figure 4.12: Authentication Flow in Arcanium

4.5.2 Authentication Overview

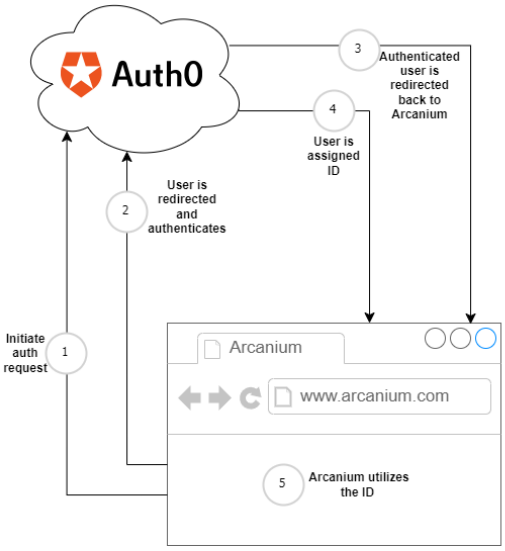


Figure 4.13: Arcanium Login Flow using Auth0

As shown in this figure, the login process starts when a user initiates an authentication request from the Arcanium interface (1). The user is then redirected to Auth0 (2), where they enter their credentials and authenticate. Upon successful authentication, Auth0 assigns a unique identifier to the user (3 and 4), which is used to manage sessions and user related data. After being redirected back to Arcanium (3), the application then utilizes this identifier (5) to personalize the user experience and allow users to submit items to the database.

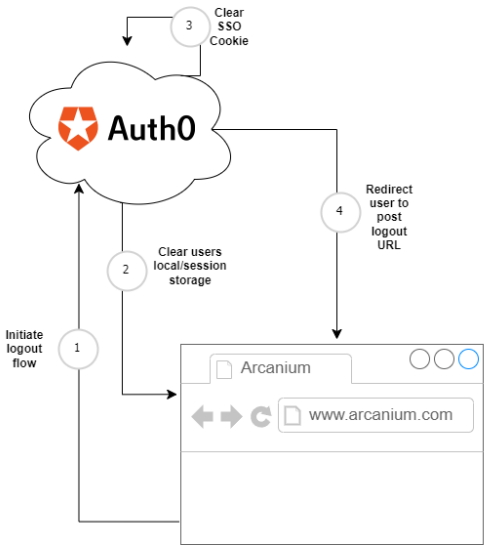


Figure 4.14: Arcanium Logout Flow using Auth0

This figure demonstrates the logout flow. When a user decides to log out of Arcanium (1), the application instructs Auth0 to clear any Single Sign-On (SSO) or session cookies (2 and 3), ensuring that the user is fully logged out from all active sessions. This process culminates in the user being redirected to a post-logout URL (4), which redirects the user to the landing page of Arcanium.

4.6 Open5e API

Arcanium heavily relies on the Open5e API, which is a Django based open source API. Arcanium uses this API for its resource searching capabilities, such as searching for spells, items, backgrounds or classes. It also utilizes the API heavily during the character creation process, which will be covered in detail during that section of the system design.

4.6.1 API Consumption

Open5e offers JSON based content from its API. Typically this can include up to thousands of lines of content depending on the resource. Arcanium polls the API, and fetches the data it needs through constructing query parameters. For example, if a user wants to search for spells in the application, Arcanium queries the following URL: *https://api.open5e.com/spells/*. If a user wants to search for specific spells, such as by level, they can then use the filters in the application, which will append the parameters the user specifies to the URL to get more specific results. Arcanium takes this JSON data and utilizes it in the frontend, and if applicable, saves the data to be sent to the backend. For example, the data is retrieved from the API during character creation, and when a character is submitted, this data is sent to the database.

Figure 4.15: Initial API JSON Content

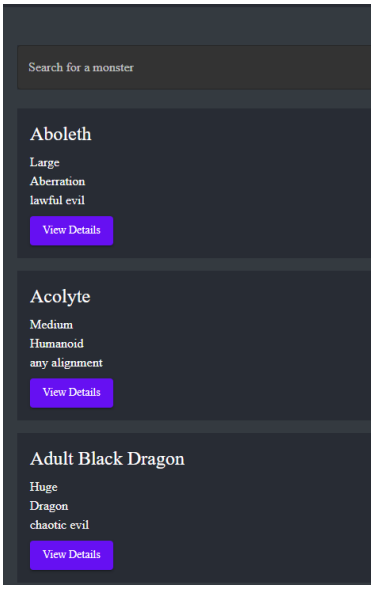


Figure 4.16: E.g. How API Content is Presented

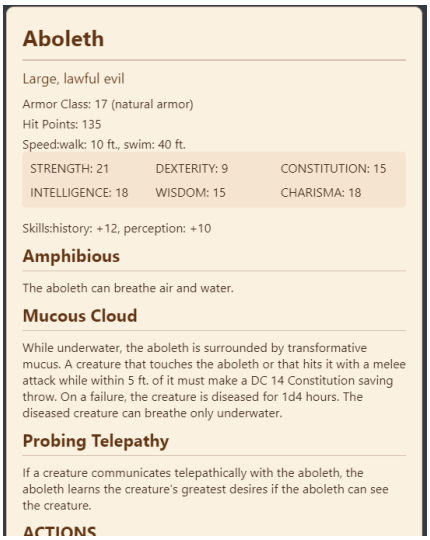


Figure 4.17: E.g. How Detailed API Content is Presented

Above showcases how API content is presented in the application itself. Earlier in this section, we showed how the API content looks initially, before it's handled by the application. The two examples above show the API content after it's been retrieved, and showcases how the content looks in a detailed format. In the first figure above, note the *View Details* button. When this is clicked, the user is shown a modal containing detailed information about the entry. In the example above, the user has clicked the first entry, Aboleth, and can then see all of the information from the API on this entry, rather than the brief information seen in the first figure.

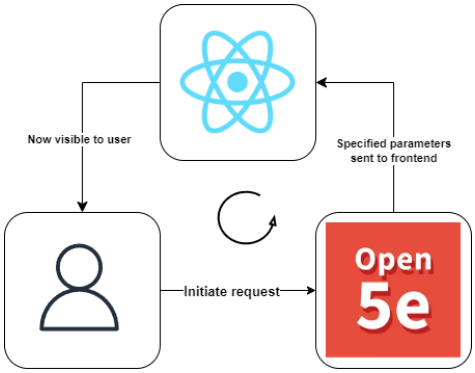


Figure 4.18: API Interaction Overview

4.7 Character Creation

In this section, we detail the user experience of navigating through the character creation process. This feature doesn't necessarily introduce a new technology, but rather utilizes the Open5e API and the database architecture of the application.

4.7.1 Character Creation Process

Figure 4.19: Character Creation Steps

In the figure above, we can see each of the character creation steps. As the user progresses through each of these steps, the data that is selected is constantly updating the character's state. When the users pass through each step, they come to the final step, and are given an overview of the character, which contains all of the choices they have made so far, containerized into a character sheet. This data is then ready to be submitted to the database. The purpose of each of the steps can be defined as follows:

Step	**Purpose**
Race	Select race, affecting abilities and traits
Class	Choose a class, defining skills and roles
Background	Determine character backstory and proficiencies
Ability Scores	Allocate points to define strengths and weaknesses
Spell Selection	Choose spells for magical classes
Equipment	Choose items and gear
Skill Selection	Select skills your character is proficient in
Character Details	Add details such as name, image and physical traits
Review and Submit	Final review and creation of the character

Table 4.1: Character Creation Steps and Their Contents

How Data is Saved

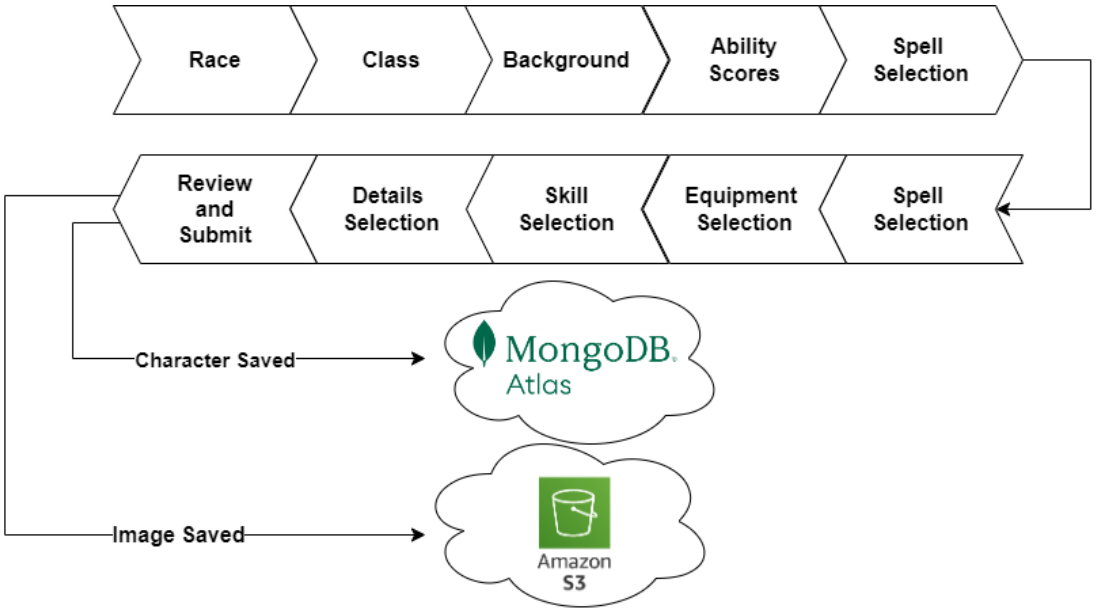


Figure 4.20: Character Creation Saving Mechanisms

Upon submitting the character, the character is saved to MongoDB Atlas, and the image is saved to the Amazon S3 bucket. Throughout the process, the character model is updated with the relevant data, which is passed through the backend, Mongoose based schema, before being submitted to the database. Below is how the image is saved:

Image Saving Architecture

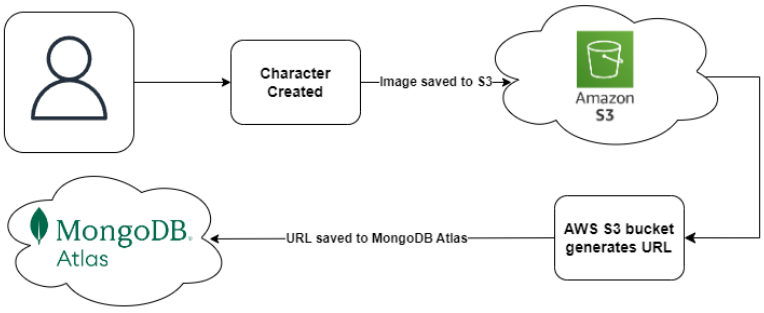


Figure 4.21: Image Saving in Arcanium

When the user submits the character, the image is immediately sent to the Amazon S3 bucket. Then, the image is given a URL by the bucket, and this URL is saved as the image for MongoDB. This allows Arcanium to access the image in an efficient and scalable way, ensuring that the image itself isn't stored in MongoDB. When the character is retrieved throughout the application, the image URL retrieves the image from the bucket and displays it normally. This approach encourages scalability, and follows the most efficient software development practices.

4.8 Artificial Intelligence Integration

4.8.1 AI Usage

The artificial intelligence integration of Arcanium, as detailed in the introduction, methodology, and technology review, is used to enhance the overall experience of the application by providing a help chatbot, which can be used by users to answer any questions that they may have while using the application. The generative storytelling section is used to provide players with a creative outlet to generate scenarios and stories for their campaigns. This process is done through the OpenAI API, which provides access to every OpenAI model. Arcanium opts to go with the 3.5 turbo model, which provides a balance between computational complexity, creativity and efficiency. Here's an overview of how the process works:

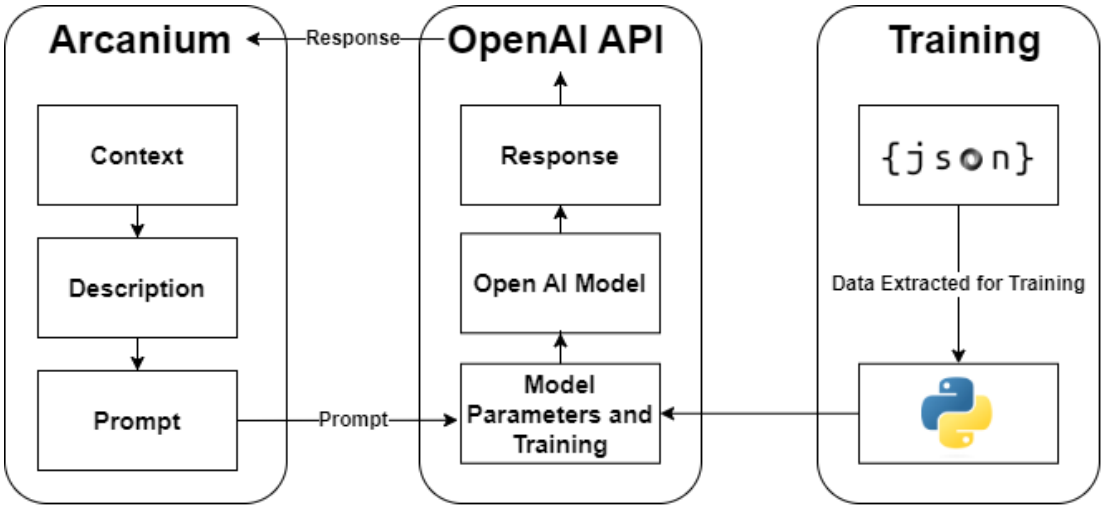


Figure 4.22: OpenAI API Interaction

Training Process

Training and fine tuning the model was an iterative process. Typically, training of models is based on prompts and responses that the model can learn from. This proved a challenge, due to the nature of the Dungeons and Dragons based data that Arcanium had access to. The data the application had access to did not have prompt/response based information, so to get around this issue, Python was utilized to extract data from the JSON files. The Python script was designed to identify each starting key of the relevant JSON files, and from this information, generate a key-value pair based on the prompt and response based format.

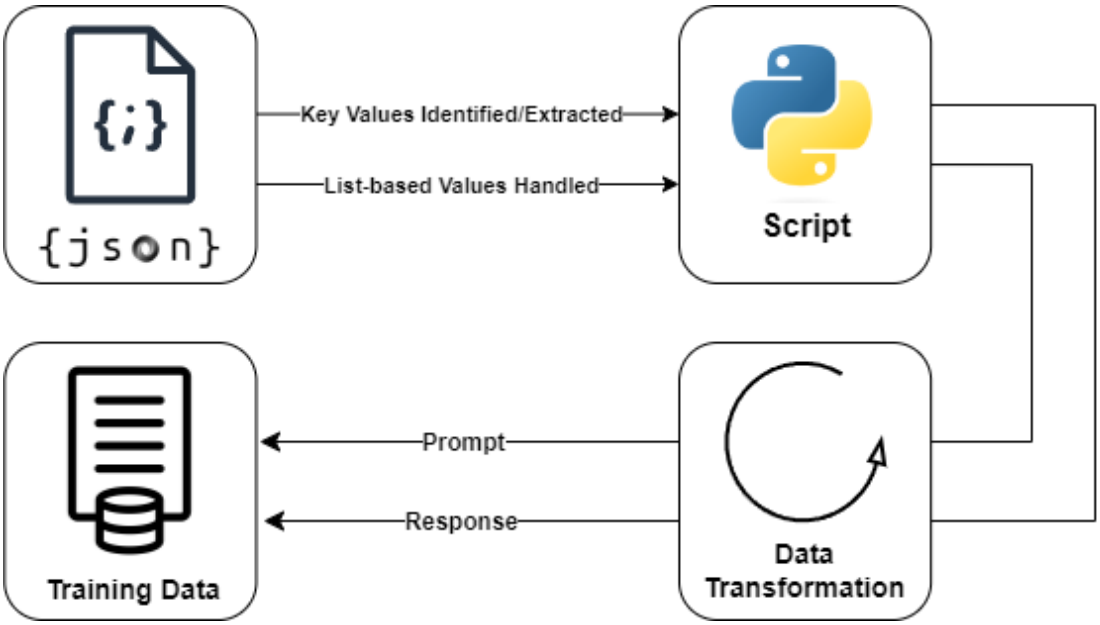


Figure 4.23: Training Process Diagram

Rationale

The initial consideration was to train a local model and then bundle it with the application. However, after conceptualizing and testing this process, a few observations were noted:

- Utilizing an earlier model, which was trained on thousands of lines of JSON, tests were conducted to see the performance. The model struggled to respond in a capable manner to most questions, even general questions, typically responding in an unintelligible manner.

- The model's size was over 3GB, even before training the model on the dataset. This is roughly 4x the size of the entire application as a whole, which would introduce concerns with hosting due to how large the size was.

- Integration with the MERN stack proved difficult. The model was based on the GPT-2 architecture from OpenAI, and locally was stored in Python-based code, proving difficult to integrate with a technology such as JavaScript without considerable effort.

After taking these considerations into account, the choice to go with an API based approach is a much better option. The models are more customizable and trainable, they offer better performance, the file size is incomparably lower and this approach is infinitely more scalable.

4.8.2 Chatbot

The chatbot is used to give users help at any point in the application, meaning that at all times, at every location in the application, the chatbot is available for use. This feature could, for example, be used during the character creation process to determine what to do next, or during resource searching to compare spells or items. The model has been trained and fine tuned to be able to answer specific questions about Dungeons and Dragons, even lesser known, niche questions. The chatbot is essential to both new and experienced players alike. The chatbot also encourages even casual discussions, it can be used as a companion at all times, for example even during sessions for quickly accessible information.

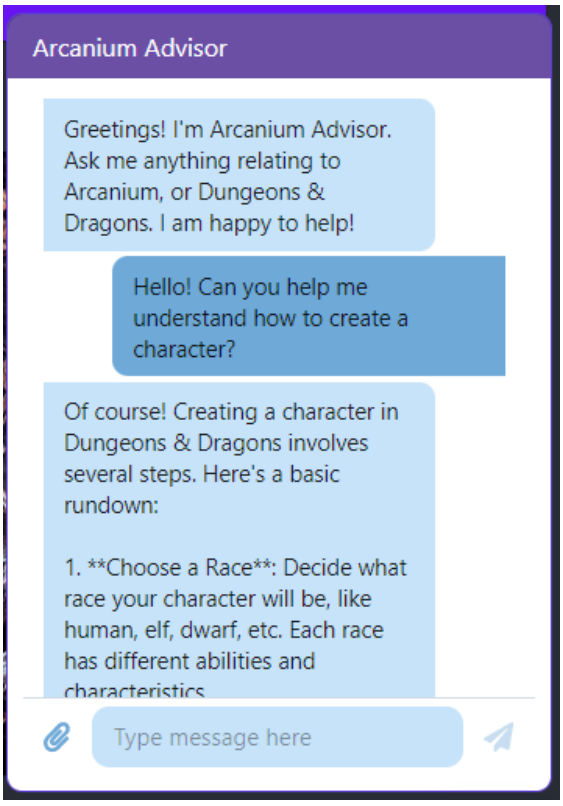


Figure 4.24: Arcanium's Chatbot

4.8.3 Generative Storytelling

The generative storytelling feature, referred to as *Dungeon* in the application, has a detailed and connected feature set. When users open the feature, they are prompted to create their story through a modal which allows players to choose context for the story, including a location and a character from their created character list. When users select these options, the story is generated with these things in mind, allowing users to personalize their experience or play through a text-based adventure with their defined characters. Here's how the generative storytelling feature functions:

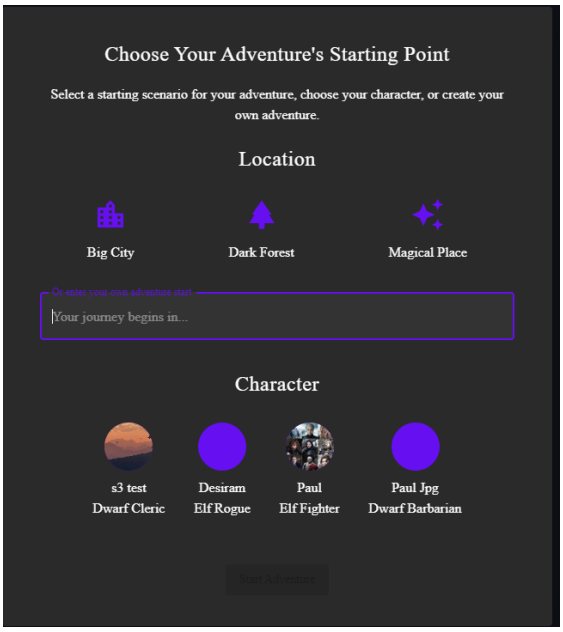


Figure 4.25: Initial Generative Story Dialog

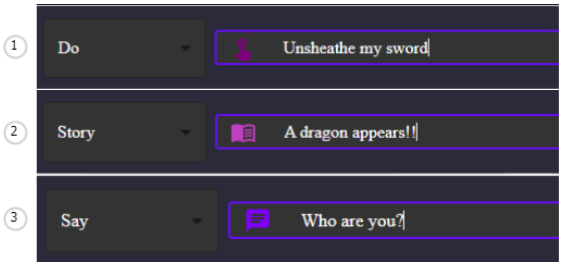


Figure 4.26: Generative Story Options

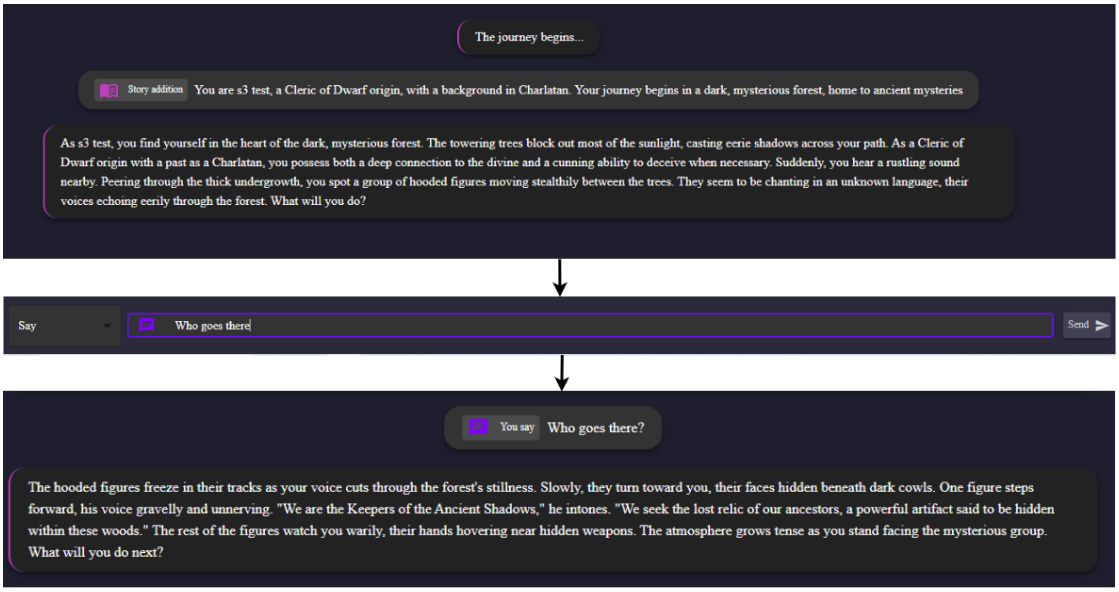


Figure 4.27: Generative Story Functioning

Throughout the figures above, we can see the generative story functioning. In the examples above, a location and character is chosen, and in figure 4.27, these attributes are shown throughout the storytelling process. The journey begins in the selected location, taking into context the character that has been chosen. This feature is done through the process outlined in 4.22, where the prompt is generated by the user through the frontend, sent to the API where the training and parameters are applied, and along with the context and description within Arcanium, is used to generate a response to the players prompt. These stories are also able to be saved, and they are sent to the MongoDB Atlas cluster, stored inside the *stories* collection. These stories can then be returned to later to view, through the *Saved Stories* window within the story creation page.

4.9 Social Connectivity

4.9.1 Friends System

To interact with users on Arcanium throughout the other social features, users must first be friends. This is to ensure security, but also because both users need an appropriate identifier to create something such as a chatroom in the chat system. Users can search for other Arcanium users through the add friends section, via searching by name or by searching all users. When a request is sent, a unique

entry is added to the database. This entry contains the status of the friendship. If it is pending, the recipient will see the request in their friend request section, if it is accepted by the recipient, the requester and recipient will both show up on each others friends list. If it is rejected, the entry will be flagged as rejected and will be removed from the users request list. If the request is ignored or missed, the status will remain as pending until the recipient has made a decision.

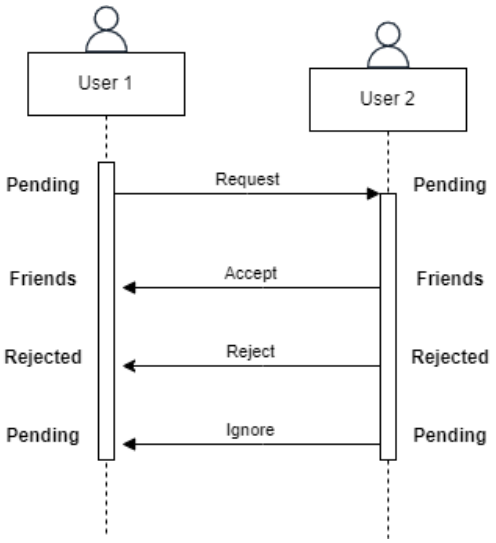


Figure 4.28: Friend Request Sequence Diagram

Once a user has accepted the request, they will be able to see the requester in their friends list. This will then give them options to view their profile, which allows users to see details such as created characters, and more detailed information. Users will then also be able to click a button to be directed to a real-time chat with the specified user.

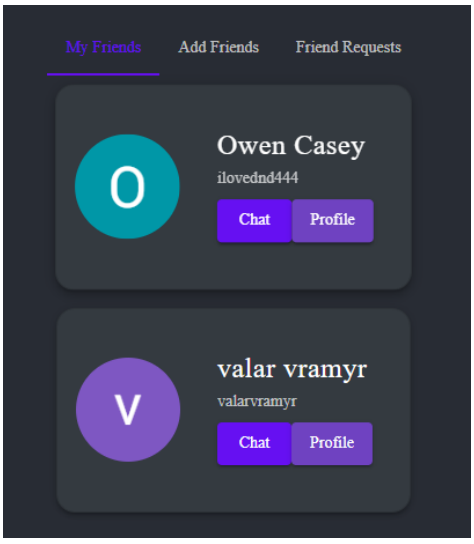


Figure 4.29: Friends List Options

4.9.2 Real-time Chat

The real-time chat functionality is managed through Socket.IO, which provides real-time asynchronous capabilities. First, users must be friends to access this feature. When users are friends, the first time a user navigates to a chatroom with another user, a room is created with both of their IDs. Then, when a user goes to access this chat room, messages persist as they are saved to MongoDB in real-time when the messages are initially sent.

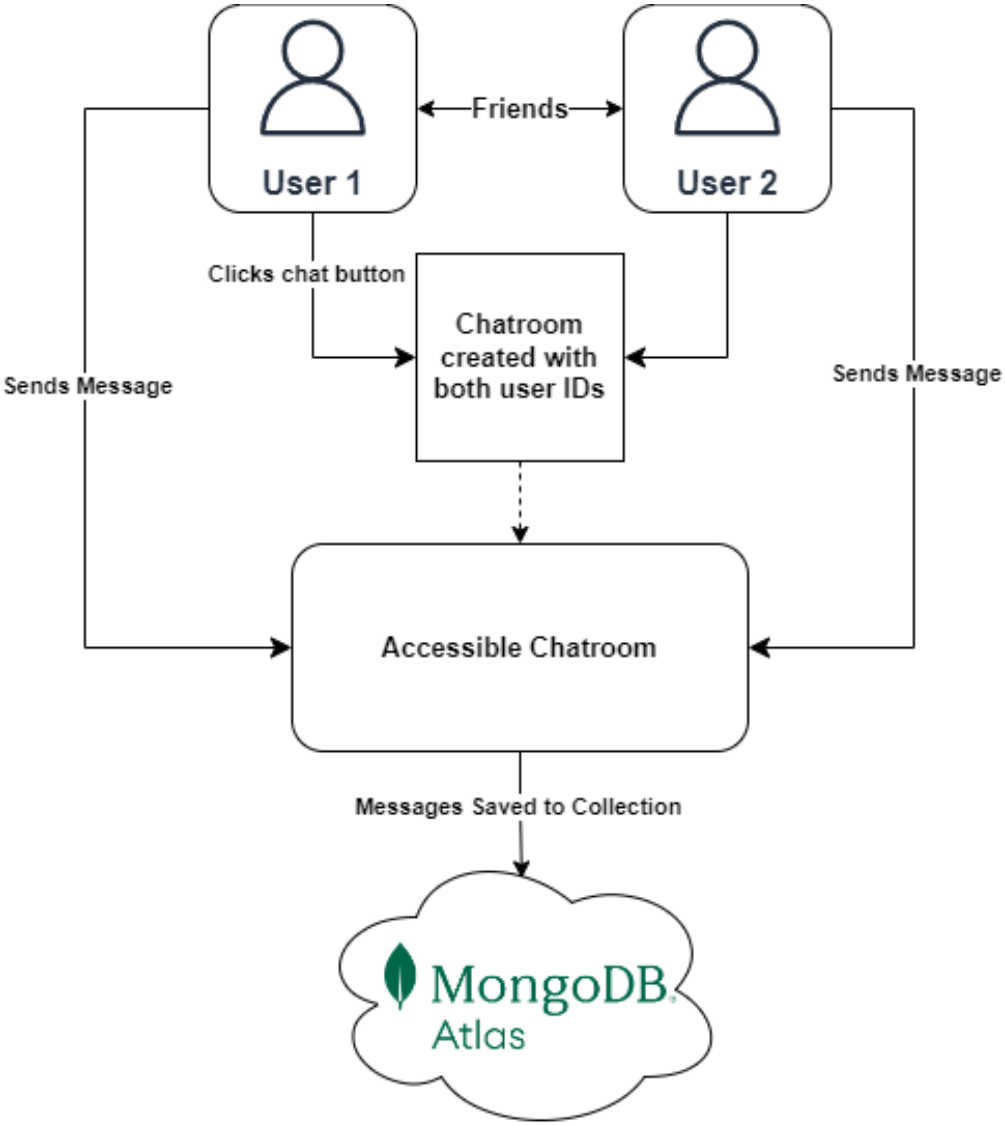


Figure 4.30: Chat System Overview

Socket.IO

Utilizing Socket.IO, when a user joins a chatroom, a connection is established to the server. When another user joins, a new connection is also established with them. The two users are now linked to the same server, joined in the same chatroom. When a message is sent, the server updates the room for both users, allowing real-time chat.

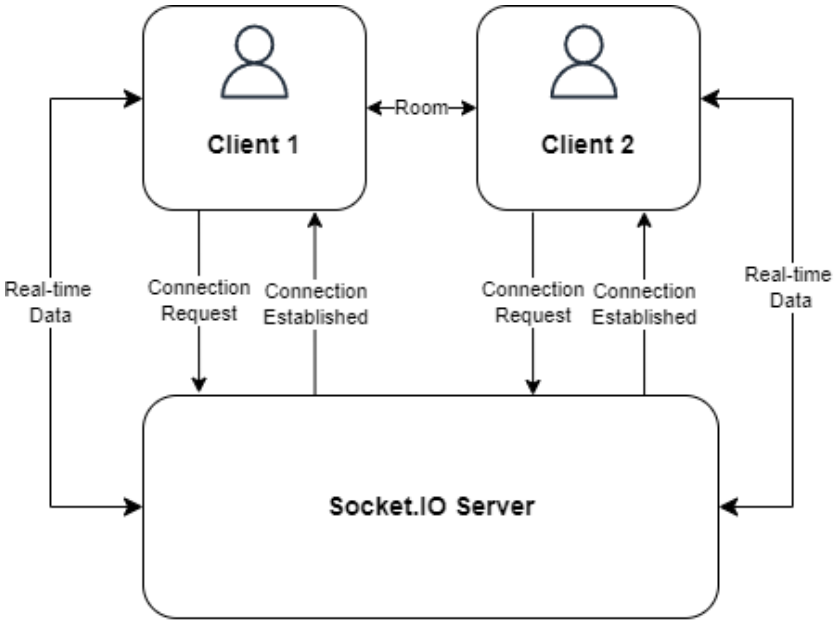


Figure 4.31: Socket.IO Overview

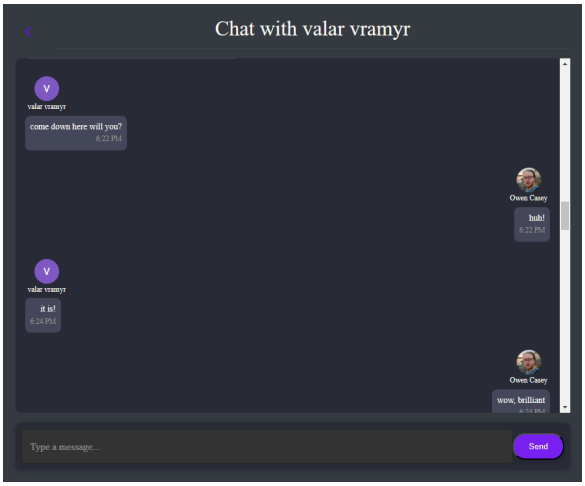


Figure 4.32: Chat User Interface

4.9.3 Campaign System

The campaign system functions in a similar way to the friends system. When creating a campaign, users can opt to invite numerous people, which sends a request to their ID. They can accept the requests in the same way as the friends system, and it functions in the same way, in terms of pending, accepted, and rejected. When users join the campaign, they are then able to view campaign members, as well as their characters, and a description of the campaign itself.

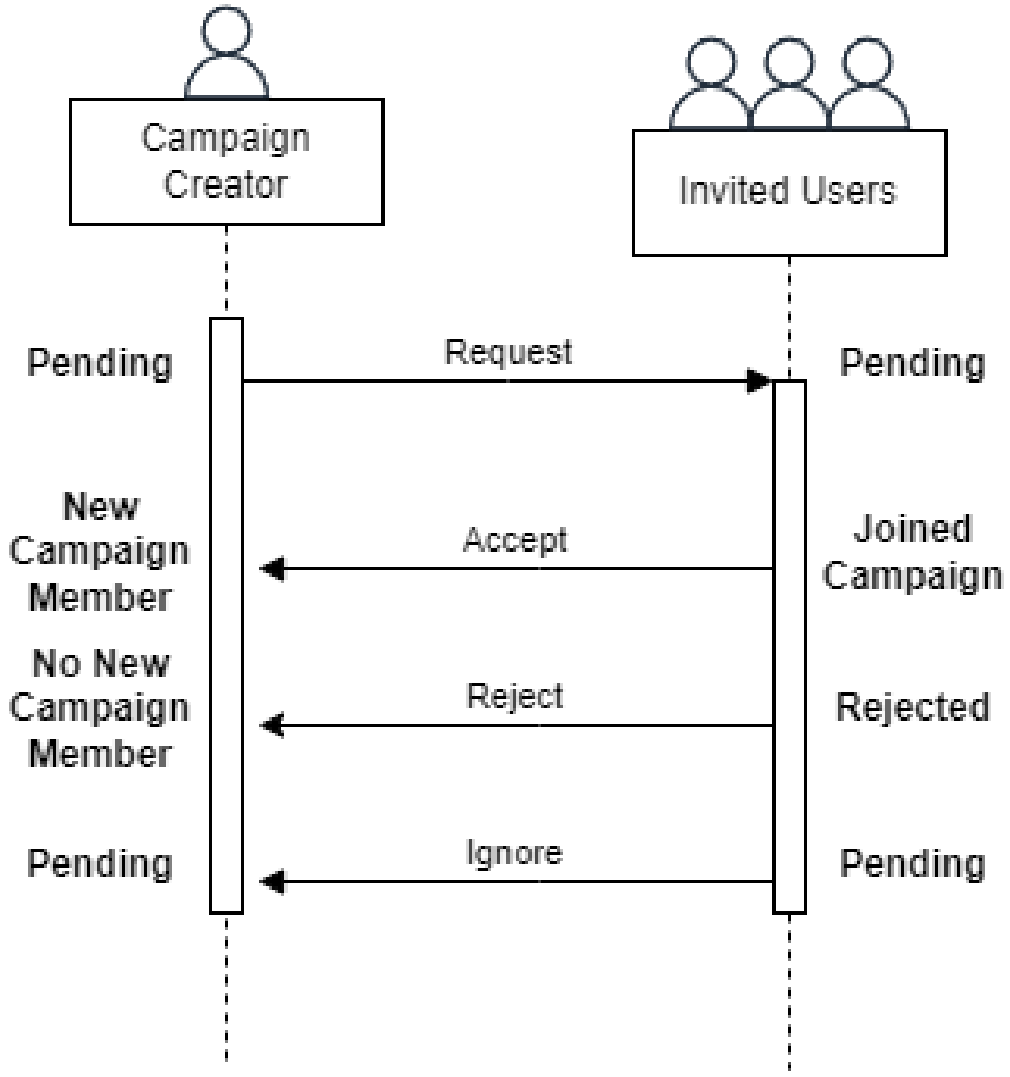


Figure 4.33: Campaign Sequence Diagram

Social Features Overall Architecture

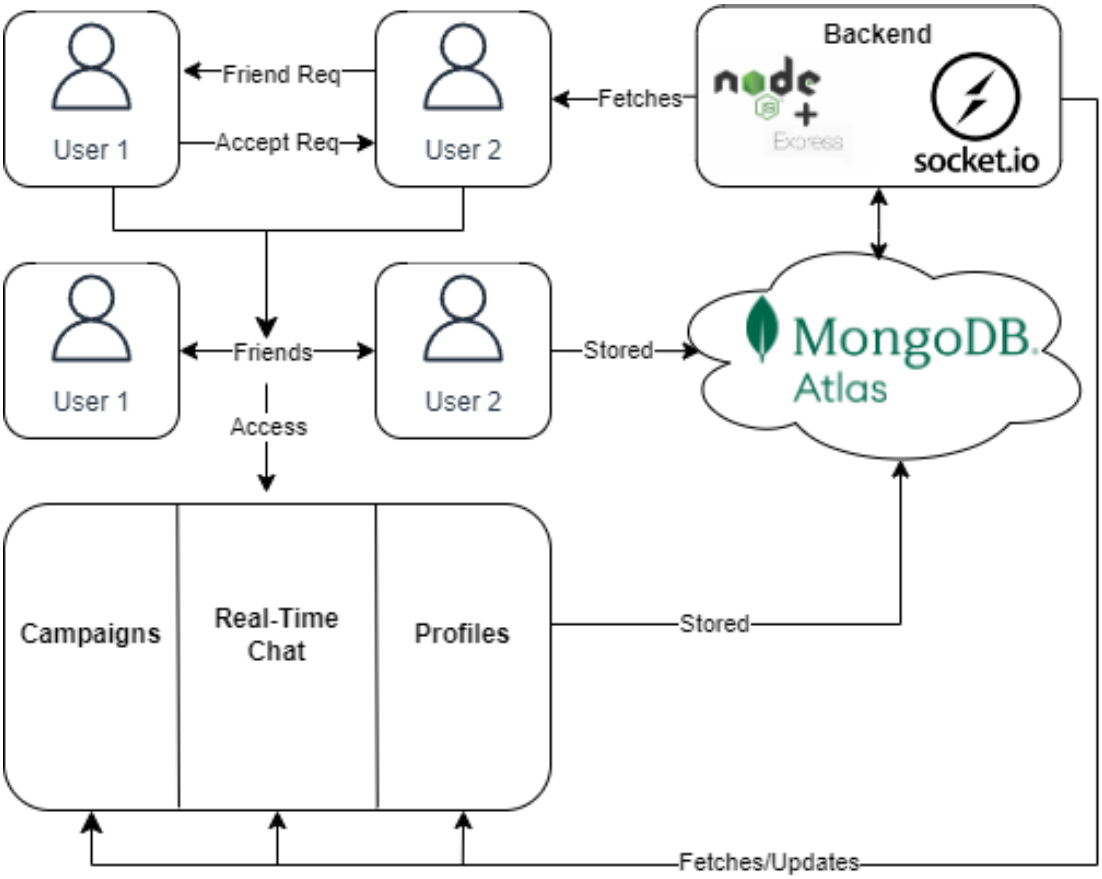


Figure 4.34: Overall Social Architecture

4.10 Deployment

Arcanium was successfully deployed using Render. Render is a cloud service that provides an efficient, cost effective solution for hosting both the frontend and the backend of the application. Render automates many of the aspects of deployment, operations and maintenance, which makes it an ideal choice for Arcanium. Render offers continuous deployment from GitHub repositories, auto scaling and SSL certificate management, while providing a generous free tier [35].

Frontend Deployment

The frontend of Arcanium is based on the single-page application (SPA) framework of React, so the application was deployed as a static site on Render. This method is perfect for the MERN stack, and provides low latency to users, and automatically manages SSL certificates, which ensures that data that is transferred from the frontend to the backend is encrypted.

Backend Deployment

Arcanium's backend uses Node and Express and requires real-time communication and updates. Utilizing Render, the backend was deployed as a web service, which ensures that the application can continuously interact with the cloud-based architecture of Arcanium such as MongoDB Atlas, Auth0 and the S3 bucket. The web service architecture on Render can dynamically allocate resource based on the application's demands, which ensures consistent performance.

Rationale and Advantages

- **GitHub Integration:** Render integrates with GitHub, allowing continuous deployment and automatic updates for the application with each commit. This integration streamlines the development process and allows the application to be up to date at all times.

- **Cost-Effectiveness:** Render provides a generous free tier in comparison to other hosting frameworks. For the needs of Arcanium, Render provides the essentials and enables scaling at a reasonable price if needed in the future.

- **Unified Platform:** Both the frontend and backend were deployed on Render, so the entire application is hosted in one location, rather than the frontend and backend communicating through two different services. This eliminates the complexities of two services communicating, such as CORS issues, and also removes the need to manage the application from two different locations.

- **Scalability:** Render's auto-scaling feature dynamically allocates resources based on application demand, ensuring consistent performance. Render also provides scalability options for expanding the application depending on the application's needs and the number of users.

Comparison

The most popular hosting framework is Heroku. Here's how Render performs in comparison to Heroku, as per the Render documentation [35].

Feature	**Render**	**Heroku**
HTTP Requests	Allows responses to take up to 100 minutes	Hard response timeout of 30 seconds
Data Persistence	Render Disks offer convenient block storage	Cannot store data across deploys
Private Networking	Built-in private networking	Only in Private Spaces, costing more
DDoS Protection	Protected by advanced DDoS protection powered by Cloudflare	Basic DDoS protection
HTTP Protocol	Serves all requests over HTTP/2 and HTTP/3	Serves all content over HTTP/1.1
Application Restarts	Does not have scheduled restarts, comes with zero downtime deploys	Forcefully restarted every 24 hours

Table 4.2: Comparison of Render and Heroku

Chapter 5

System Evaluation

This chapter evaluates Arcanium as a project, assessing it's initial objectives against the outcome of the project. This evaluation seeks to evaluate not only the defined objectives, but also the overall application as a whole, providing evaluations based on features and testing outcomes, as well as outlining the application's strengths and weaknesses, to provide a comprehensive overview of Arcanium.

5.1 Objectives and Evolution

The journey from the initial stages of Arcanium to what it has evolved into has been a long and iterative process. With the time frame of the project and the nature of academia, the objectives and overall scope of the project were bound, and expected, to change. This section covers the initial objectives and the achieved objectives, detailing how they have evolved and adapted.

5.1.1 Initial Objectives

As outlined in the introduction, the primary aim of Arcanium was to provide a streamlined and centralized digital platform that consolidates essential resources and tools that are needed by players of D&D, encouraging creativity and social connectivity. The main initial objectives of the project can be defined as follows:

Objective	Description
Centralized Hub	Centralize tools, resources, and information into a single platform.
User Friendly Design	Cater to both new and experienced players.
Multi Platform Support	Ensure accessibility on both desktop and mobile devices.
API Integration	Integrate a comprehensive API for resource searching.
Dynamic Character Creation	Offer an interactive process for character creation.
Social Connectivity	Enable friend adds, campaign joining, and real-time messaging.
AI Integration	Utilize AI for story generation and chatbot assistance.
Virtual Tabletop	Implement a virtual space for interactive play, dice rolling and more
Responsive Design	Develop the application to support mobile responsively.
Best Practices Adherence	Follow industry standards in design, accessibility and efficiency.
Security/Authentication	Implement robust authentication and security.
Scalable Database	Ensure a scalable architecture for user data storage, utilizing cloud infrastructure where possible.
Deployment	Deploy the application online, allowing it to be accessible at a specified domain.
Agile Methodology	Adopt Agile practices for flexible, iterative and continuous development.

Table 5.1: Summary of Initial Objectives for Arcanium Project

These objectives were designed and conceptualized in the planning stages of the project, and aligned to as best as possible throughout development. The objectives were envisioned to allow Arcanium to be an application with a clearly defined vision, ensuring scalability, interactivity and accessibility.

5.1.2 Objectives Overview

As outlined in table 5.1, the initial objectives of Arcanium were vast and complex. This section specifically seeks to compare the outcome of the objectives to the initial objectives.

User Experience and Accessibility

Objective	Status	Explanation
Centralized Hub	Majority Met	Vast majority of the initial features centralized in one location.
User-Friendly Design	Met	Integrated user-friendly features such as the help chatbot and user-friendly UI.
Multi-Platform Support	Met	Arcanium functions on both mobile and desktop devices through testing, making it support multiple platforms.

Table 5.2: Evaluation of User Experience and Accessibility Objectives

Innovation and Integration

Objective	Status	Explanation
API Integration	Met	API functionality was heavily integrated into Arcanium, being one of the essential tools for the application's features.
Character Creation	Met	Interactive character creation process was integrated successfully, with these characters having detailed information as chosen by the user.
Social Connectivity	Met	Initial social features such as real-time chat, campaign management and the friends system were all successfully implemented, ensuring a socially connected application.
AI Integration	Met	AI integration was successfully implemented on two fronts, with both a chatbot and generative story functionality being utilized through custom trained OpenAI models.
Virtual Tabletop	Unmet	Feature was re-evaluated and it was deemed necessary to cancel development on the feature, due to reasons outlined in the technology review.

Table 5.3: Evaluation of Innovation and Integration Objectives

Development, Performance and Security

Objective	Status	Explanation
Responsive Design	Majority Met	Arcanium was initially envisioned as a PWA, but after considerations around offline features being unrealistic, a decision was made to make the application responsive instead. Arcanium still works on mobile devices, but can't be downloaded or used offline due to the nature of its feature set.
Best Practices Adherence	Met	The development process adhered to industry standards for software design, user experience, and scalability, ensuring a reliable and high-quality application.
Security/Authentication	Met	A robust security framework and authentication system were implemented, through Auth0, with this being the basis for many of Arcanium's user-centric features.
Scalable Database	Met	Arcanium heavily uses cloud infrastructure in almost every area possible, utilizing two cloud-based storage mechanisms through MongoDB Atlas and AWS S3 bucket.
Deployment	Met	Arcanium was successfully deployed using Render, which provides the application with an efficient and scalable framework for future expansion.
Agile Methodology	Met	Agile practices, specifically Kanban, were successfully employed to manage the project development, resulting in a flexible and iterative process that supplemented solo development and evolving project requirements well.

Table 5.4: Evaluation of Development, Performance, and Security Objectives

Overall, the objectives outlined in the tables above clearly showcase that Arcanium achieves the majority of the objectives that it set out in the introduction.

5.1.3 Unmet Objectives and Adaptions

Notably, within the tables above, there are only a few objectives that aren't fully met. These objectives are ones that were less prioritized, or went through revaluation throughout development.

- **Centralized Hub**: The objective to create a centralized hub within Arcanium was largely met, with the successful integration of the majority of planned features and tools. However, this objective was not fully achieved as the virtual tabletop, one of the initially envisioned core features, was re-evaluated and removed from the project. The absence of the virtual tabletop has a valid reason, as detailed in the technology review and below, but this does mean that the initially envisioned centralized hub components were not completely met.

- **Virtual Tabletop**: The development of the virtual tabletop feature was ultimately not realized. Several challenges contributed to this outcome:

  - *Performance*: Initial investigations into suitable technologies for building the virtual tabletop, such as Canon and Three.js, indicated that they were resource-intensive and introduced performance issues to the application.
  - *Complexity and Compatibility*: The complexity of implementing these technologies, especially for a meaningful, interactive 3D experience, proved to be difficult. Also, compatibility issues arose during the conceptual testing, particularly with mobile devices.
  - *User Feedback*: The user survey, as shown in 2.1, indicated that the virtual tabletop was the least desirable feature among both new and experienced players, questioning its value to the overall user experience.
  - *Scope and Coherence*: Conceptual visualizations and planning revealed that the virtual tabletop did not align well with the application's goal to supplement physical D&D sessions, as it attempted to replace the physical social aspect of the game

  Due to these factors, as well as feature prioritization, it was decided necessary to remove this feature for now, with the intention to revisit it in future development of Arcanium.

- **Responsive Design**: The goal to implement responsive design was mostly met. Initially, Arcanium was proposed as a Progressive Web App (PWA), which was featured during the Christmas review slides. However, further

development recognized that one of a PWAs main goals, which is offline capabilities, did not align with Arcanium's always-online features. Since users are required to log in for personalized experiences and real-time interactions, the offline based approach of a PWA was not suitable for Arcanium. Therefore, while the application is responsive and functional across devices, the shift from a PWA to an online based, responsive design means the initial objective was not met. However, the functionality remains the same, so this issue likely should have been recognized during the initial planning stages rather than later into development.

In conclusion, Arcanium performed exceedingly well when compared with its initial objectives. The application had a clear and well-defined goal, which allowed each feature to be fully realized, while ensuring efficiency, scalability, and a user-friendly experience.

5.2 System Testing

Testing was integral to the process of validating Arcanium, to ensure it is a user-friendly experience. Given the dynamic nature of Arcanium, manual testing was utilized heavily and was applicable to Arcanium's needs.

5.2.1 Manual Testing

- **Functional Testing:** Each feature was tested to ensure it functioned as expected and as per the project requirements. This process included the main components such as character creation, social features and chatbot assistance.
- **Usability and Compatibility Testing:** This type of testing was used to ensure that the application functions well across any device, and ensured that the application's interface was intuitive and the features and navigation worked as expected.
- **Integration Testing:** Integration testing was utilized to ensure that each component of the application integrates well together. This included testing the functionalities of each feature but also using them at the same time when possible, and making numerous requests to the application at the same time from different devices.
- **Regression Testing:** Before and after each main commit for the application, the project was tested thoroughly using the methods outlined above to

ensure that new features work as intended and don't adversely affect existing features.

- **Consistency and Repeated Testing:** Each manual testing type was repeated numerous times, to ensure that a wide range of scenarios were covered. This involved repeating the tests to ensure the same expected output was achieved.

5.2.2 Jest

To complement the manual testing efforts, Jest was utilized where possible to ensure consistency across the application.

- **Snapshot Testing:** Utilizing snapshot testing, the UI of Arcanium was tested to ensure that there aren't unintended changes in the UI of the application. Snapshot testing saves a serialized output of a component or page and references it in future tests. The comparison of these snapshots help detect unintended changes to the UI.
- **Mocking and Simulation:** Jest's enabled testing to simulate responses from external APIs and databases. This process was mainly used to test the APISearch component, which is heavily utilized throughout the application.

These testing strategies helped to ensure that Arcanium was a seamless and verified application. Testing revealed bugs and weaknesses in the application which were able to be identified and fixed. Overall, the testing process thoroughly improved Arcanium as an application.

5.3 Weaknesses and Limitations

While Arcanium has met the vast majority of its objectives, and has met them well, like any long term, complex project, there are weaknesses or limitations that must be acknowledged. Identifying these limitations provide us with insight into how the application can be improved, and what it can benefit from further into development.

5.3.1 Campaign System

- **Group Communication**: Arcanium has a detailed campaign system, but for a more integrated communication system, including real-time group chat functionalities using Socket.IO, would have been useful. This would have

allowed more detailed communication in the campaign setting. Due to time constraints, this feature was not fully realized to its best potential, however, it still works as intended.

- **Detail in Creation Process**: While the campaign creation process covers basic functionalities, it lacks intricate depth as it doesn't contain visual representations or other visual queues. A more comprehensive creation process, allowing for intricate campaign customization and management would have enhanced the campaign experience.

5.3.2 Test Coverage

- **Comprehensive Testing**: Due to time limitations, a decision was made to prioritize testing on core functionalities. This approach, while practical, means that certain parts of the application would benefit from more comprehensive testing.

- **Future Testing Plans**: With this in mind, there is a clear road map for expanding test coverage. This includes integrating more extensive unit tests or functional tests through Jest, or even more comprehensive testing frameworks.

5.3.3 Other Limitations

- **Optimization and Responsiveness**: Arcanium was developed with both mobile and desktop versions in mind, but some features function better on desktop due to the increased space. These features work adequately on mobile, but they could potentially be refined more to better the mobile user experience.

- **Mobile/Desktop Designed Features**: The development process prioritized a broad compatibility strategy over mobile or desktop specific features. Some features include different layouts for different device sizes, but with further development time, features could be specifically refined for mobile or desktop to make the user experience better.

5.4 Strengths of the Application

Arcanium embodies many strengths, but the following are what stand out most after reflecting on the development process.

5.4.1 Scalability and Cloud Usage

- **Cloud Database/Storage**: Utilizing MongoDB Atlas, the database section of Arcanium is scalable and efficient, especially with the extent that the application utilizes the database. Images are also stored in a scalable solution, the S3 bucket.
- **API-based Approach**: The application's integration with various APIs, including OpenAI's API models, Open5e for D&D resources, and Auth0 for authentication, showcases its scalable design.
- **Deployment and Future Updates**: The deployment architecture of Arcanium supports real-time updates through GitHub and is accessible at any time. Through the use of continuous integration provided by Render, Arcanium can seamlessly deploy future updates and changes to the live version of the application.

5.4.2 User Experience

- **Accessible Design**: The interface is crafted to be welcoming and intuitive for all users, regardless of their experience with D&D, through the use of the help chatbot and UI design.
- **User-Centered Features**: From dynamic character creation to generative story features, each feature is designed to enhance engagement and immersion, making the experience of Arcanium memorable and useful.
- **Consistent User Experience**: Arcanium delivers a consistent user experience across all devices. This consistency is demonstrated by the visual design, accessibility and user interface design, with mobile having unique navigation as compared to desktop.

5.4.3 Technical Innovation

- **AI Integration**: Arcanium utilizes both help chatbot based AI and generative story AI, showcasing it's innovation and integration of complex features.
- **Scope and Integration**: Each feature in Arcanium is not only comprehensive but also integrates with other features in the application. For example, a user can create a character and utilize this character in the generative storytelling and creating a campaign processes seamlessly. Users can interact with friends and view their characters, invite them to campaigns and view their profiles.

Chapter 6

Conclusion

Arcanium began as an idea of personal passion and evolved into an impressive dynamic experience that is valuable to Dungeons and Dragons players, both new and experienced. It provides a comprehensive, unified platform with centralized tools and resources that make the experience of playing D&D simpler and more inclusive.

Arcanium aimed not to replace the physical D&D experience, but to enhance it. Through the use of its diverse feature set, users are able to create characters, generate stories, search for resources, and socially connect with friends and other D&D players alike.

Arcanium set out its foundations in the introduction and further researched the potential of the application during the methodology chapter. The technological review served as the research basis of the features and provided valuable insights into how the application should be developed efficiently, while the system design chapter detailed the application's complex architecture. The system evaluation concluded the findings and highlighted that Arcanium achieved the vast majority of its initial objectives and adapted to challenges and changes well. The chapter highlighted some of the limitations of the application, before defining its strengths. These revelations provided insights into how the application can be improved with further development while relying on its solid foundation and reliable, scalable architecture.

In conclusion, Arcanium definitively achieved the goals it set out to accomplish. It provides a valuable, dynamic, and scalable platform for Dungeons and Dragons players to be creative and connect. The journey from a personal idea to a fully realized application reflects the importance of passion in driving innovation and transforming an idea into reality.

Appendix

Here are some important links related to the project:

- GitHub Repository: Arcanium on GitHub
- Deployed Application: Arcanium Deployed App
- Screencast: Google Drive or YouTube

Bibliography

[1] John Begy. Playing at the world: A history of simulating wars, people, and fantastic adventures: From chess to role-playing games. *American Journal of Play*, 5(3):391–393, 2013. Available: https://www.proquest.com/scholarly-journals/playing-at-world-history-simulating-wars-people/docview/1459141989/se-2.

[2] The Conversation. 50 years on, Dungeons and Dragons is still a gaming staple: What's behind its monumental success. https://theconversation.com/50-years-on-dungeons-and-dragons-is-still-a-gaming-staple-whats-behind-its-monum 2022.

[3] Kanban - agile methodology | atlassian. https://www.atlassian.com/agile/kanban, 2023.

[4] Andrew Zhu, Lara J. Martin, Andrew Head, and Chris Callison-Burch. Calypso: Llms as dungeon masters' assistants. *ArXiv*, abs/2308.07540, 2023.

[5] Jose Ma. Santiago, Richard Lance Parayno, Jordan Aiko Deja, and Briane Paul V. Samson. Rolling the dice: Imagining generative ai as a dungeons & dragons storytelling companion. *ArXiv*, abs/2304.01860, 2023.

[6] Inside the dungeons & dragons renaissance. https://www.nytimes.com/2022/05/21/style/dungeons-and-dragons.html, 2022.

[7] How 'critical role' helped spark a dungeons & dragons renaissance. https://www.cnbc.com/2020/03/14/critical-role-helped-spark-a-dungeons-dragons-renaissance.html, 2020.

[8] Dungeons & dragons systems reference document. https://dnd.wizards.com/resources/systems-reference-document, 2023.

[9] Jest Contributors. Jest documentation. https://jestjs.io/docs/getting-started.

[10] Usage statistics and market shares of javascript libraries. https://w3techs.com/technologies/overview/javascript_library.

[11] Matthew C. Loring, Mark Marron, and Daan Leijen. Semantics of asynchronous javascript. In *Proceedings of the 13th ACM SIGPLAN International Symposium on on Dynamic Languages*, DLS 2017, pages 51–62, New York, NY, USA, 2017. Association for Computing Machinery.

[12] D. Flanagan. *JavaScript: The Definitive Guide : Master the World's Most-used Programming Language*. O'Reilly Media, Incorporated, 2020.

[13] Javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[14] Adam Wolff. Relicensing react, jest, flow, and immutable.js. https://engineering.fb.com/2017/09/22/web/relicensing-react-jest-flow-and-immutable-js/, 2017.

[15] M. Bertoli. *React Design Patterns and Best Practices*. Packt Publishing, Limited, 2017.

[16] Codecademy. Understanding the react virtual dom. https://www.codecademy.com/article/react-virtual-dom.

[17] E. Hahn. *Express in Action: Writing, building, and testing Node.js applications*. Manning, 2016.

[18] A. Mardan. *Express.js Guide: The Comprehensive Book on Express.js*. Azat Mardan, 2014.

[19] Material-UI. Getting started. https://mui.com/material-ui/getting-started/.

[20] Axios. Introduction. https://axios-http.com/docs/intro.

[21] MERN Stack. https://www.mongodb.com/mern-stack.

[22] K. Chodorow. *MongoDB: The Definitive Guide*. O'Reilly, 2013.

[23] MongoDB. MongoDB Atlas Documentation. https://www.mongodb.com/docs/atlas/.

[24] Vasan Subramanian. *Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node*. 01 2019.

[25] Auth0 Customers. `https://auth0.com/customers`. Showcasing Auth0's biggest customers.

[26] Auth0 API Documentation. `https://auth0.com/docs/api`.

[27] Auth0 Security. `https://auth0.com/security`.

[28] Auth0 Documentation. `https://auth0.com/docs`.

[29] Dungeons & Dragons. *Dungeons & Dragons Player's Handbook (Core Rulebook, D&D Roleplaying Game)*. Dungeons & Dragons. Wizards of the Coast Publishing, 2014.

[30] Open5e api documentation. `https://open5e.com/api-docs`, 2023.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[32] M. Topal, Anil Bas, and Imke van Heerden. Exploring transformers in natural language generation: Gpt, bert, and xlnet. 02 2021.

[33] Ni Li. Ethical considerations in artificial intelligence: A comprehensive disccusion from the perspective of computer vision. *SHS Web of Conferences*, 179, 12 2023.

[34] T. Cadenhead. *Socket.IO Cookbook*. Packt Publishing, 2015.

[35] Render. Render Documentation. `https://docs.render.com/`. Accessed: April 21, 2024.