Software Requirements Specification

DnDTracker

A web application utility for Dungeons and Dragons

Derek Williamson
Andrew Schlein
Nathaniel Duncan
Tyler Carey

Team Silicon Valley
Monday, September 23, 2019

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this project is to develop a web application for character players and dungeon masters of the table-top game “Dungeons and Dragons.” This application will assist in campaign management and data tracking to streamline the user experience of the currently manual process of playing “Dungeons and Dragons.” The scope of this project includes the development of the design, architecture, and implementation in order to provide a full web application.

1.2 Scope

The software product of our project is titled “DnDTracker,” a web application for managing Dungeons and Dragons campaigns, tracking data from past and current campaign games, and providing a set of grievance-removing features to users.

These features include digital implementations of presently manual and physical processes, such as character designing, stat rolling, and storyline tracking. The objective of this project is (1) to provide an open source repository for other developers to use as a template or base for their applications per the MIT license, (2) to develop a digital version of the tools and processes the target users already require, and (3) to design other tools not yet implemented in current gameplay that would improve gameplay management.

Dungeon Masters will be able to start, maintain, customize their campaigns. Players will be able to create and customize their characters before importing them into a campaign they’re invited to. All users will be able to view the campaigns they’re currently in, past to present, from a web-portal.

1.3 Definitions

The software application of this project, as previously mentioned, will be henceforth referred to as “DnDTracker,” or Dungeons and Dragons (Data) Tracker. Any mention of AWS refers to Amazon Web Service(s).

1.4 Overview

The following Software Requirements Specification document details the following sections:

- Overall description of the application (Sections 2.1-2.3)
- Specific functional and performance requirements of the application (Sections 4.1-4.2 and 5.1)
- Appendix items, including a glossary of reference terms (Section 6.1-6.2)

2. Overall Description

2.1 Product Perspective

The web application is designed to run within the Amazon Web Services ecosystem and therefore does not require a set of on-site resources. The application is designed for use on desktop, laptop, and tablet devices and requires that the user have both a browser and an internet connection to make use of the software. Support for smartphone devices is out-of-scope for this project but is kept as a stretch goal. The application implements the following AWS and CI/CD resources: DynamoDb, S3, ElasticBeanstalk, and Travis CI. Through the browser, the user will interact with the application by accessing a publicly accessible domain. A flow diagram is demonstrated in Figure 1.

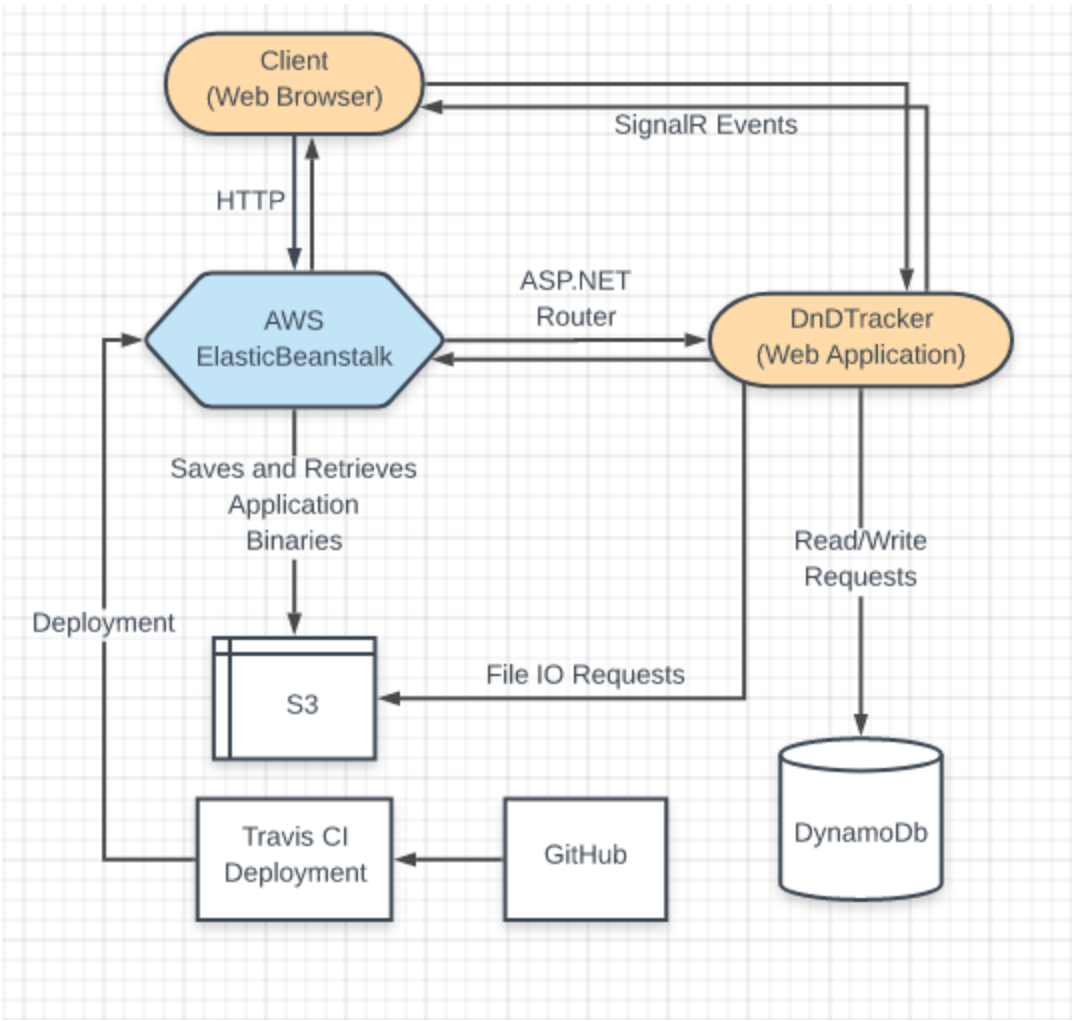

Figure 1 - Interaction between the major components

2.2 Product Functions

DnDTracker implements the following major interfaces and features:

- Enables campaign creation and customization through a flow of pages
- Enables character creation, customization, and importing through a flow of pages
- Allows the Dungeon Master to manage their campaigns through an admin panel
- Provides players with a view of any campaign and the respective character party
- Provides all users with a list of their past and current (on-going) campaigns

2.3 User Characteristics

DnDTracker is primarily intended for use by two separate classes of users: the players and the developers. DnDTracker is setup and documented for further development by other developers with experience in .NET, AWS, and web development or are pursuing said experience. The web application itself is intended for the Dungeons and Dragons player base, the character players and dungeon masters. These users do not require any specialized technical expertise to use the application. They are only required to have access to a desktop/laptop and the internet.

2.4 Operating Environment

The software will operate on an AWS ElasticBeanstalk application instance, connected to an AWS S3 bucket for file store and an AWS DynamoDb instance for data persistence. No other external components or applications are required for the operation of the program.

2.5 Constraints

Due to the design and nature of ASP.NET, the developers are constrained to the C# programming language for core development with occasional script development in JavaScript. Because the database, file storage, and machine environment are housed and managed by Amazon, the developers must be wary of the costs associated with AWS resource usage. Fortunately, AWS auto scale according to the hardware demand, so manual memory and system related configuration is not a concern. The developers are also constrained to the maintainability, stability, and consistency of the AWS resources. Deployment and software delivery are automated, so developers can continue development without the concern of upgrading.

2.6 Assumptions and Dependencies

The project could be affected if the cost of the AWS resources exceeds $50 per month. Because the university does not cover the costs of Amazon’s services, the project lead will be responsible for resource costs. The web application of this project assumes that the users would have a tablet, laptop, or desktop device and an internet connection to access it.

3. External Interfaces

3.1 User Interfaces

User interfaces of the web application must follow a series of style guidelines and standardizations. This constraint is to ensure that the GUI is uniform throughout the user experience. Figure 2 shows the general UI design that should be shared between pages.

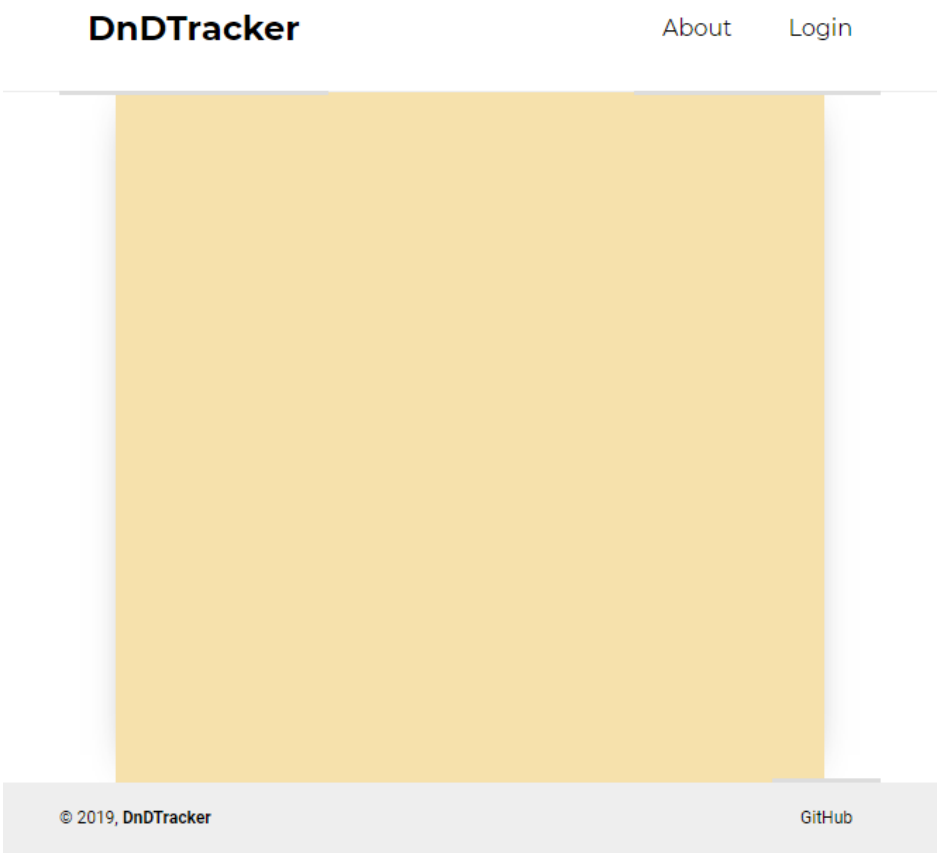

Figure 2 - The primary shared layout of the application's pages

Also as seen in Figure 2, the header and footer must be identical between pages on the web application. Header and footer buttons must display the hover style shown in Figure 3, while normal buttons must display the styles shown in Figure 4.

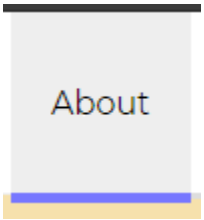

Figure 3 - A menu button that is being hovered over

Figure 4 - A standard button vs a hovered button

The application will include the following pages (designs to-be-determined):

- Home
- Login
- About
- User Dashboard
- Character Dashboard
- Create Character
- Character Details
- Start Campaign
- Join Campaign
- Campaign Dashboard
- Dungeon Master

The web application will also populate these pages through a back-end interaction with the user's profile and DynamoDb.

3.2 Hardware Interfaces

The software product is compatible with any Windows or Linux environment that is capable of running the dotnet toolset for compiling and executing .NET applications. Because of the auto scaling nature of the hosting resources, hardware components are not a direct concern of the developers. All that is required of the AWS hardware by the software is that there must be a stable internet connection, continual scalability, and public accessibility. The hardware mentioned includes instances of DynamoDb, ElasticBeanstalk, and S3.

3.3 Software Interfaces

The software currently integrates the following software components and packages to function:

- SignalR (1.1.4) – for asynchronous RPC server-client communication
- AWSSDK.DynamoDBv2 (3.3.101.60) – for retrieval and persistence of data objects
- Google.Apis.Auth (1.41.1) – for user account authentication
- Microsoft.AspNetCore.* (2.2.0) – for the dynamic web application page framework
- Newtonsoft.Json (2.2.0) – for JSON object conversion and utilities
- Moq (4.13.0) and NUnit (3.10.1) – for unit testing the application during CI/CD

Because of the abstraction involved in AWS's resources, system specifications and operating systems are not specified by the developer for the three services used for this product. The data sent within the system between the server and the client comes in the form of RPC events and invocations. This data is passed through asynchronous streams to receivers that handle the invocations, allowing the client and server to maintain a non-requesting communication/relationship. The application will integrate with the DynamoDb instance and Google API through an in-app authentication service that passes around session data and login information, comparing for validity.

3.4 Communications Interfaces

The DnDTracker requires users to connect to the web application's domain through a web browser on a desktop or laptop device in order to use it. The application will then communicate between the server and the browser through HTTP requests and RPC communication using ASP.NET's native router and content delivery framework, JavaScript's ajax library, and SignalR's event signaling. The server will also use AWS's .NET libraries to communicate with the environment's AWS resources.

4. Functional Requirements

4.1 Players (All Users)

4.1.1 Login Page

- The user shall access the login page from the header.
- The user shall login through a Google Sign-In widget.
- The server shall notify the user whether their sign-in attempt was successful or not through a line of text by the widget. If it is not successful, an error message will be displayed in that line of text.
- The user's GUID will be persisted across pages via the client's session.
- The server shall replace the sign-in widget with a sign-out widget once the user is logged in.
- The user shall sign-out completely once the sign-out widget is clicked.

4.1.2 User Dashboard

- The user shall be redirected to their dashboard immediately after a successful login attempt.
- The server shall list the user's active and inactive campaigns in a list view with to-be-determined details shown within the campaign item's container by retrieving their campaign data from the database.
- The user shall be redirected to a campaign page when a campaign is clicked in the list view.
- The user shall have a header menu that allows them to navigate to any of the pages detailed in 4.1.3, 4.1.4, and 4.2.1.

4.1.3 Character List Page

- The user shall be redirected to this page by the relevant button on their header menu.
- The server shall retrieve the user's entire character list from the database and list summaries of each character in a list.

4.1.4 Character Creation Page

- The user shall be redirected to this page by the relevant button on their header menu.
- The user shall specify all the necessary fields (to-be-determined) relevant to character creation before clicking the create button at the end of the process.
- The application shall provide buttons for generating templatized data for the user, such as stats and names, if they prefer not to decide on their own field values.

4.1.5 Character Details Page

- The user shall be redirected to this page after clicking the character item in their Character List page.
- The server shall display all the user-defined field items for this character.
- The user shall be provided an Edit button that redirects back to the Character Creation page with the existing information prefilled.

4.1.6 Campaign Page

- The user shall be redirected to this page after clicking the campaign item in their User Dashboard.
- The dungeon master shall be redirected to their Dungeon Master page when the Dungeon Master button is clicked.

- The server shall provide the user with a list of players, characters, campaign details, Dungeon Master details, and notes (specified by the Dungeon Master) that are relevant to the selected campaign. Line items will be determined during more in-depth analysis of Dungeons and Dragons handbook.

4.2 Dungeon Masters

4.2.1 Start Campaign Page

- The user shall be redirected to this page by the relevant button on their User Dashboard header menu.
- The server shall prompt the user with a list of fields required for the creation of a new campaign, such as name, emails to invite, and story background.
- The user shall be redirected to the campaign page by the Start button.

4.2.2 Modify Campaign Page

- The server shall list all the pre-populated input fields that existed on the Start Campaign page with the data originally filled by the user.
- The user shall be able to save any changes they've made to the input fields.

4.2.3 Dungeon Master Page

- The user shall be redirected to this page by the Dungeon Master button on the Campaign page.
- The server shall provide the user with a note creation interface that allows the user to create and send notes to the participating party.
- The user shall apply stat modifications to the characters of the campaign through an event interface that includes dropdowns and input fields for each type of event: Modify Stats, Kill, Resurrect, and more to-be-determined.
- The user shall be redirected to the Campaign page by a Back to Player View button.

5. Non-Functional Requirements

5.1 Performance Requirements

The following requirements detail the user's performance needs when accessing and interacting with the application:

- Each of the application's pages must load in under two seconds.
- The application must handle an undefined number of campaigns and users in order to fit the unbound needs of the user. Resource needs are a separate problem and will be handled on a case-by-case basis.
- Server-client interactions and events must occur in under five seconds.
- UI elements of the same type must be consistent across all pages.
- The UI must maintain the same color scheme and stylesheet theme applied across all pages.

5.2 Security Requirements

The following requirements detail the security needs of the user and the client:

- Account information shall not be stored on the application's database or file storage.
- Details of a user's Google account shall not be compromised and shared with any other user or administrator.
- Google Sign-In requests must be authenticated by Google's Auth API.
- The application's AWS credentials must be secured by a set of secret keys that shall not be stored publicly.

5.3 Availability Requirements

The following requirements detail the expected availability of the application in multiple regards:

- The application must be accessible per Amazon's resource availability and stability.
- The application must be accessible from all devices.
- The application must be accessible from any WAN.

6. Appendix

6.1 Glossary

ASP.NET – Microsoft's open-source server-side web application framework for dynamic web pages

AWS – Amazon Web Services

AWS DynamoDb – Amazon's non-relational, no-SQL database service

AWS ElasticBeanstalk – Amazon's cloud deployment and provisioning service that automates the setup and tasking of an application

AWS S3 – Amazon's Simple Storage Service that provides scalable object storage infrastructure

Domain – Short for "domain name;" an identification string for a defined IP address.

Campaign – A Dungeons and Dragons term for a customized storyline for a party of players.

CI/CD – Continuous Integration and Deployment; a set of practices that involves automated testing, packaging, and deploying without manual intervention to increase productivity

DnDTracker – Dungeons and Dragons (Data) Tracker

Dungeon Master – A Dungeons and Dragons player that maintains the storyline of a campaign and interacts with the party of players through other non-player characters, events, and phenomenon

GUI – Graphic User Interface; the interface users interact with

MIT License – A permissive free software license that allows the end user to copy, modify, merge, and distribute the software

Moq – A .NET package that provides service mocking in test cases to prevent the need for internal or external environment dependencies.

NUnit – A .NET framework that provides means for writing and executing test cases.

OAuth – An open standard for access delegation through tokens that prevents the need for providing sensitive data, such as passwords.

RPC– Remote Procedural Call; An invocation of a method over the network from one peer to another.

SignalR – A .NET package that provides client-server communication without the need for client-to-server requests.

Travis CI – A CI/CD platform that automatically builds, tests, and deploys code.

WAN – Wide Area Network; any network that is both accessible and able to access the Internet.

6.2 Index