Dungeon Forge Service Layer Design

Prepared for: Joseph Gradecki, Professor
Prepared by: Adam Ure, Student
October 29, 2023

OVERVIEW

This document outlines the service layer design for the Dungeon Forge application. Topics covered in this document will be the services identified for implementation of the web application, the endpoints used with those services, and how these endpoints leverage the user's data. A brief overview of the Dungeon Forge project is provided to remind the reader the context of the application.

Dungeon Forge Project

In the game of Dungeons and Dragons, one group member takes the role of dungeon master. The dungeon master is responsible for planning and preparing the adventures for the rest of the team members, playing the enemies and non-playable characters within the game, and adapting the plot to make the group's experience enjoyable. This is a taxing responsibility, forcing the dungeon master to spend days, weeks, or even months preparing for the adventure to come. To save the dungeon master's time and reduce the complexity of starting a new adventure, the dungeon forge project automatically generates adventures for groups of adventurers. These campaigns come with an environment, a plot, enemies, combat, intrigue, and sometimes even lead to a player character's demise. They are designed to be runnable out of the box, with the dungeon master simply needing to read ahead to know what happens for the adventurers prior to the next session. No mapping characters, defining plot twists, or creating scale replicas of the scene. The dungeon forge project generates the materials needed start and run a fun campaign the entire group can enjoy.

Service Layer(s)

The web application portion of the Dungeon Forge is rather simple. As mentioned above, the dungeon masters who represent our user persona look to Dungeon Forge to save them time by generating a Dungeons and Dragons campaign in a readily available format. To enable this feature, the application will consist of the CampaignService. The responsibility of this service will be to interact with the web server to generate and fetch campaigns. The MVP consists only of generating and fetching a campaign. Additional service layers would be implemented to handle other features such as authentication. However, for the MVP, we will only address the CampaignService.

Open API Documentation

To better support the development of the web API, endpoint documentation has been drafted and submitted along with this document. Certain aspects of the document will be copied into this document for reference, but examples of API calls and their responses can be found within the API documentation.

CAMPAIGN SERVICE

The role of the campaign service is rather simple — handle the generation of new campaigns. In our application, this involves two user interactions. The first interaction is when the user submits the campaign generation form; second is when the user view a campaign from the success page.

Campaign Submission

The main page of the MVP is the campaign submission page, or as the title says, the “Create a New Campaign” page. The user’s goal is to generate a campaign with as little effort expended as possible. This is why the form is extremely simple. Figure 1 shows the button in the wireframes where the user submits that form and the service layer functions.

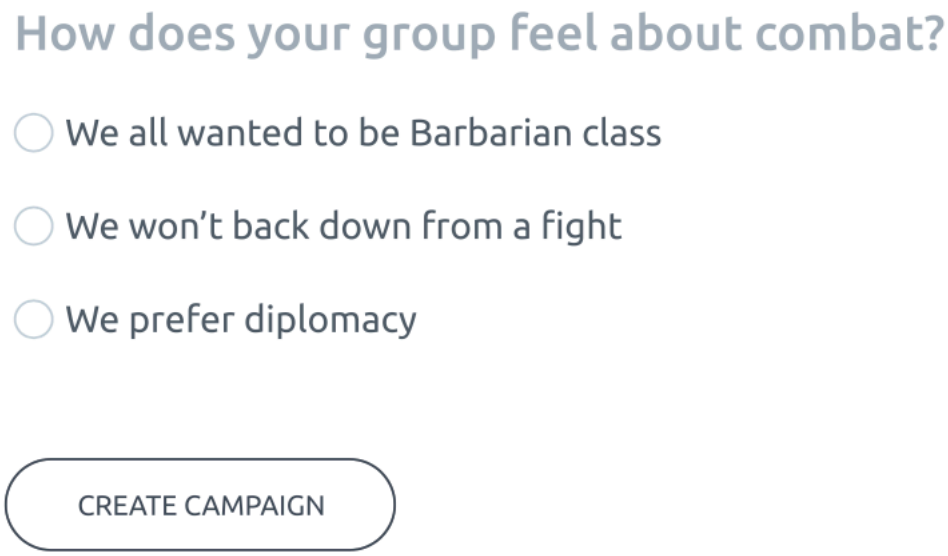

Figure 1. Create a Campaign Button

When the user taps the create campaign button, the application calls the generate campaign command, as seen in Figure 2. This command takes a request body containing the form inputs. Details on the JSON Schema the input requires can be found in the Open API documentation provided with this document.

```
POST /campaign/generate
```

Figure 2. Generate Campaign Command

This request can fail, and is likely the most error prone section of this document, so error handling will be crucial. If this command fails, an error modal is displayed to the user. However, if the command is successful, a success modal is displayed to the user. A successful response contains a campaign identifier, which can be used in the next command to see the generated campaign. Details on the response format can be found in the Open API documentation.

View Campaign

The other command our users need to make is to fetch the final generated campaign. This is done through a separate API command, which is sent when the user taps the action button on the success modal popup. Figure 3 shows the action button as designed in the wireframes.

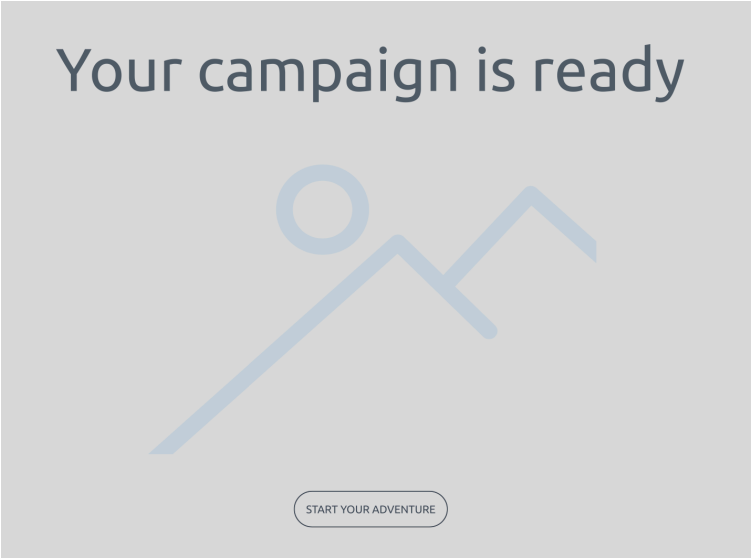

Figure 3. Success Modal Action Button

When the user taps the “Start Your Adventure” button, an additional API request is made to fetch the details of the campaign. In the MVP, this will return a JSON object that will contain all of the details needed to display the adventure to the user on the web page. As a stretch goal, the command will return the URL of the campaign’s pre-generated PDF document, which the user will then view in a PDF viewer and have the option to download.

Figure 4 illustrates the command URL and structure in order to fetch a campaign. This command can fail with either a 404 error, which indicates the requested campaign does not exist, or a 200 which includes the JSON object containing the campaign information. Details and examples of these response bodies can be found in the Open API documentation.

```
GET /campaign/{id}
```

Figure 4. Get Campaign Command

REVIEW

The implementation of the Campaign Service is rather simple, only including two (2) distinct commands. Future iterations of the Dungeon Forge will likely include more commands, but the difficult aspect of the system will be generating the content to make up an entire Dungeons and Dragons campaign. We will leverage multiple apis for this, specifically OpenAI’s language models and the DnD 5e API referenced in the Database Design document. These tools will be combined to generate content, while the server validates the content and performs error checking and handling. The client is a simple client, performing a single POST request and a single GET request, but it leverages a powerful content generator to provide its value to the end user..