DnDTracker

Team Silicon Valley
Derek Williamson
Tyler Carey
Nathaniel Duncan
Andrew Schlein

SOFTWARE DESIGN DOCUMENT
Version 1.0

Table of Contents

1 Introduction

This document describes the software design of the Dungeons and Dragons (Data) Tracker, or DnDTracker. The purpose of this document is to provide board clarity of the project's back-end implementation and user-end experience.

DnDTracker is a free web application designed to replace the use of pen and paper in Dungeons and Dragons™ campaign participation and management. By providing a simple, modern interface for our users, DnDTracker removes the strain of starting and tracking a campaign for old and new players alike. This utility allows users to customize and organize their characters and campaigns without restricting their creativity.

Software design is essential to any software project because it provides the foundation for user expectations, technical requirements, and product visualization. Attempting to build a software application without design is equivalent to attempting to build a car without schematics.

In the rest of this document, you will find the following sections of information:

2. Overview of Architecture
   This section shows a diagram demonstrating the components of the software and how they interact with each other and third-party APIs.
3. Use Cases
   A. Use Case Diagram
      This section contains a diagram demonstrating the intended primary use cases for the software given each user.
   B. Use Case Specifications
      This section contains use case specifications for the use cases described in section 3.A.
4. Classes and Their Interaction
   A. Class Diagram
      This is a diagram that shows the multiplicity, attributes, and methods of each class as well as the relationships between them.
   B. Class Details
      This is a series of tables that define the attributes and methods of the classes shown in section 4.A.
   C. Sequence Diagrams
      This is a diagram that describes the relationship between five of the primary use cases.
5. State Transition Diagrams
   This section describes the software's transitions between states and the details behind each state.
6. User Interface Diagrams
   This contains a screenshot of the user interface and a breakdown of the different components and functionalities that a user would use.
7. Conclusion
   This section details what we have learned in the design, how it is helpful, what the difficulties that we faced are, and how we plan to mitigate them during development.
8. References
   This is a list of all reference documents using IEEE standards.

2 Overview of Architecture

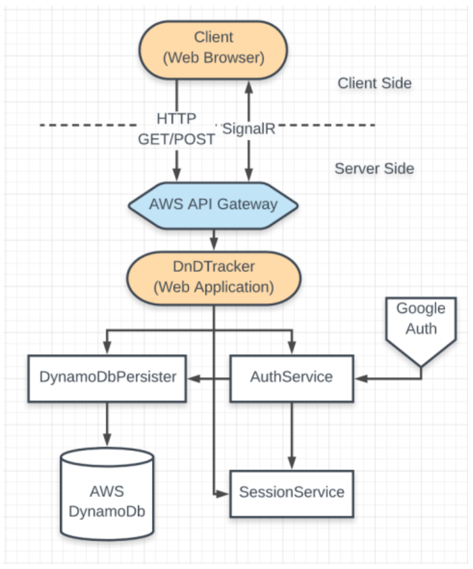


Figure 1 - Architecture diagram of various components

The architecture of DnDTracker is a hybrid of monolithic architecture and client-server architecture, achieving a simple, well-tested, rapidly deployable web application that boasts the client-server communication benefits that come with the usage of thin clients. When clients send HTTP requests to the DnDTracker web application (through a masking AWS API Gateway), the server processes, retrieves, and stores information using the DynamoDbPersister, a service that implements access to the environment's AWS DynamoDb instance. The server also authenticates users using the authentication service, AuthService, which implements Google's Auth (Sign-In) API. All secure authentication information is persisted in the SessionService, allowing access by both the server and client. This architecture best fits our needs because it allows the software itself, void of third-party integrations, to be organized into 3 components: client views, server controllers, and server services.

3 Use Cases

3.A Use Case Diagram

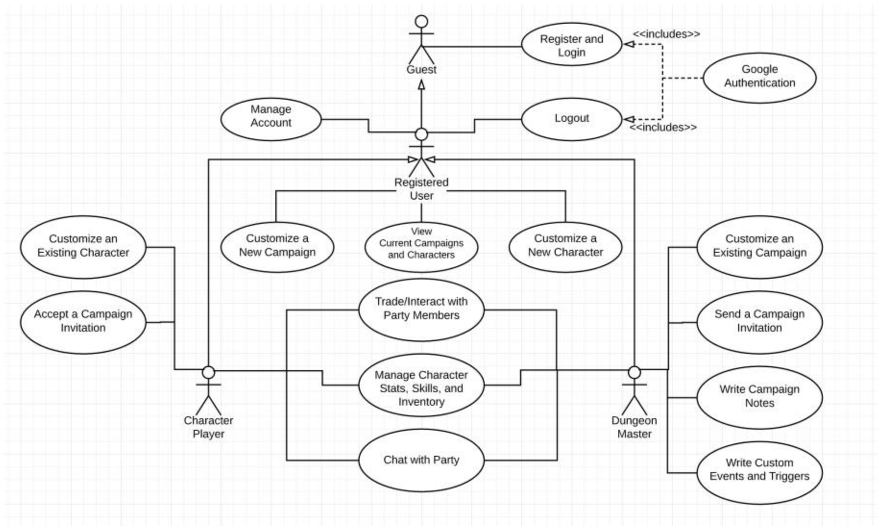

Guest
Register and Login
<<includes>>
Google Authentication
Logout
<<includes>>
Manage Account
Registered User
Customize an Existing Character
Customize a New Campaign
View Current Campaigns and Characters
Customize a New Character
Customize an Existing Campaign
Accept a Campaign Invitation
Trade/Interact with Party Members
Send a Campaign Invitation
Manage Character Stats, Skills, and Inventory
Write Campaign Notes
Character Player
Dungeon Master
Chat with Party
Write Custom Events and Triggers

Figure 2 - User case diagram for DnDTracker's users

3.B Use Case Specifications

3.B.i UC-1 (Registered User customizes a new character)

Use Case Number	UC-1
Use Case Name	Create a Character
Overview	Registered User customizes a new character
Type	Primary
Actors	Registered User
Pre-condition	Guest has logged in as a Registered User
Main Flow	1. Registered User visits create a character page from the taskbar 2. Registered User fills in all of the required fields (at least) in all of the sections 3. Registered User creates their character at the end of their customization 4. System creates the new character and stores it as a part of the Registered User's list of characters 5. System informs the Registered User through a notification that their action was successful
Alternate Flow	4. System fails to create the new character because the Registered User missed the required information OR the database failed to store 5. System informs the Registered User through a notification that their action was unsuccessful and to try again
Post-condition	True or False

3.B.ii UC-2 (Registered User customizes a new campaign)

Use Case Number	UC-2
Use Case Name	Create a Campaign
Overview	Registered User customizes a new campaign
Type	Primary
Actors	Registered User, Dungeon Master
Pre-condition	Guest has logged in as a Registered User
Main Flow	1. Registered User visits create a campaign page from the taskbar 2. Registered User fills in all of the required fields (at least) 3. Registered User creates their campaign at the end of their customization 4. System creates the new campaign and stores it as a part of the Registered User's list of campaigns, making this Registered User a Dungeon Master 5. System informs the Dungeon Master through a notification that their action was successful
Alternate Flow	4. System fails to create the new character because the Registered User missed the required information OR the database failed to store 5. System informs the Registered User through a notification that their action was unsuccessful and to try again
Post-condition	True or False

3.B.iii UC-3 (Character Player accepts a campaign invitation)

Use Case Number	UC-1
Use Case Name	Join a Campaign by Invitation
Overview	Character Player accepts an invitation sent by the Dungeon Master
Type	Primary
Actors	Character Player, Dungeon Master
Pre-condition	Guest has logged in as a Registered User and has received an invitation from the Dungeon Master of the campaign
Main Flow	1. System displays a notification that the Character Player has been invited to the campaign 2. Character Player accepts the invitation 3. System adds the Character Player to the campaign, allowing them to see the details of the campaign 4. Character Player requests to import an existing character 5. System sends request to the Dungeon Master 6. System notifies the Character Player of the success or failure to send the request
Alternate Flow	2. Character Player inputs the campaign's invite code into the Join a Campaign page

	3. System adds the Character Player to the campaign IF they have already been invited, allowing them to see the details of the campaign
Post-condition	True or False
Cross Reference	UC-4

3.B.iv UC-4 (Dungeon Master sends a campaign invitation)

Use Case Number	UC-4
Use Case Name	Send a Campaign Invitation
Overview	Dungeon Master sends a campaign invitation
Type	Primary
Actors	Dungeon Master
Pre-condition	Guest has logged in as a Registered User and is viewing the Dungeon Master page of their campaign
Main Flow	1. Dungeon Master clicks the campaign code OR the invitation button to view the invite prompt 2. Dungeon Master inputs the emails of the recipients into the prompt 3. System sends the invitation via email to the recipients AND via notification if the recipients are signed in as Registered Users 4. System stores the recipients in the campaign as a list of those who are allowed to join
Alternate Flow	3. System fails to send the invitation via email or notification because the email did not pass validation 4. System informs the Dungeon Master that the invitation failed to send
Post-condition	True or False

3.B.v UC-5 (Dungeon Master writes campaign notes)

Use Case Number	UC-5
Use Case Name	Create a Campaign Note
Overview	Dungeon Master writes campaign notes
Type	Primary
Actors	Dungeon Master, Character Player
Pre-condition	Guest has logged in as a Registered User and is viewing the Dungeon Master page of their campaign
Main Flow	1. Dungeon Master inputs the rich contents of the note and the Character Players of the party who will receive it 2. Dungeon Master completes the creation of the note 3. System stores the note in the campaign with specific permissions of who can see it

	4. System notifies the Character Players that have permission that they received a note from the Dungeon Master
Post-condition	True or False

4 Classes and Their Interaction

4.A Class Diagram

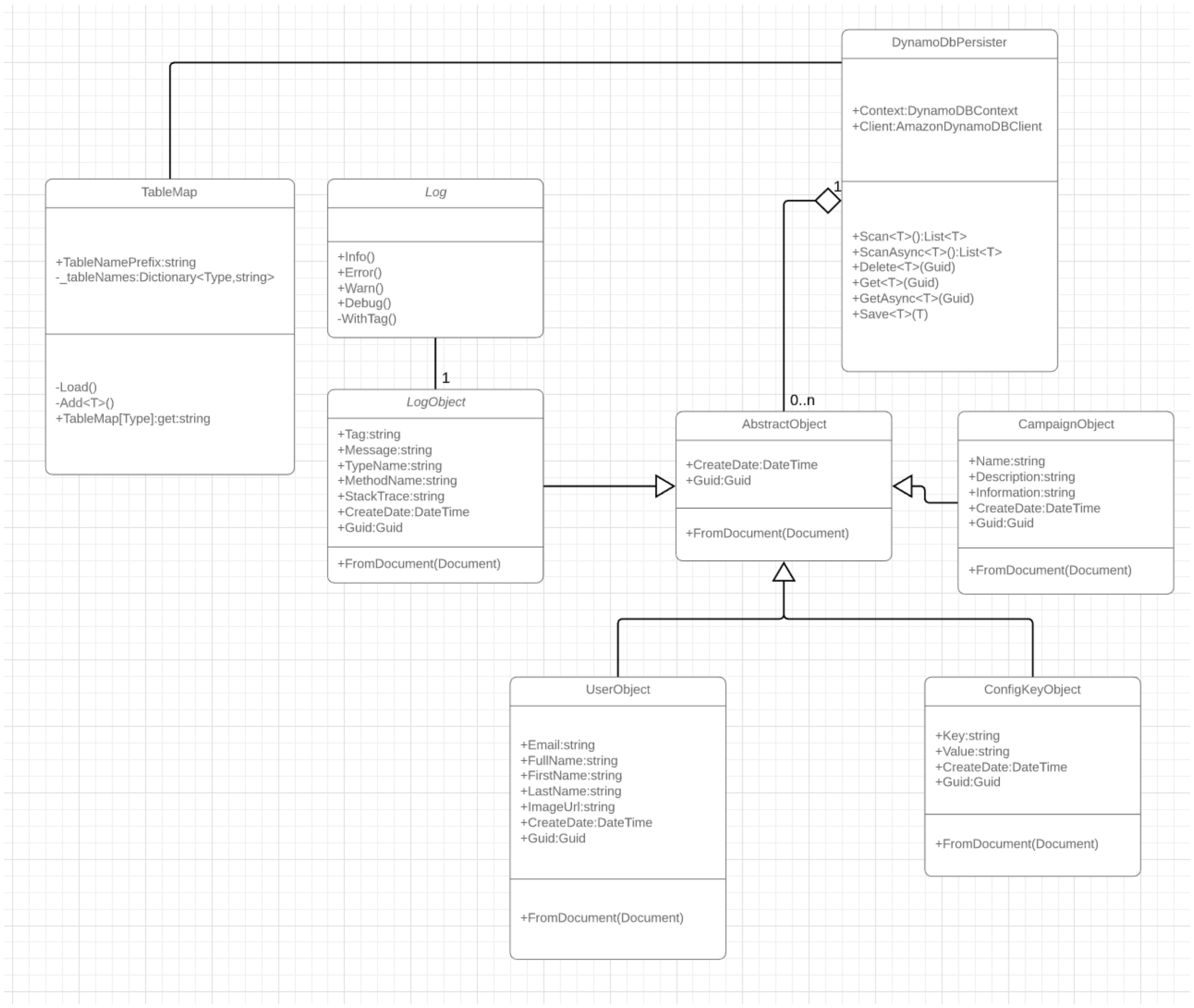


Figure 3 - The diagram of classes that handles fetching and storing

4.B Details of Classes

4.B.i C1 – AbstractObject

Attributes:

Attribute Name	Type	Description

CreateDate	DateTime	The date that the object was created.
Guid	Guid	The unique identifier of the object for fetching and deleting purposes.

Operations:

Method Name	Arguments Passed	Expected Return Type	Description
FromDocument	Document	void	Parses data from the AWS DynamoDb Document into the object when loading from DynamoDb.

4.B.ii C2 – LogObject

Attributes:

Attribute Name	Type	Description
Tag	string	INFO, ERROR, WARN, or DEBUG
Message	string	The log message.
TypeName	string	The class type that the log was stored from.
MethodName	string	The method within the class type that the log was stored from.
StackTrace	string	The stack trace of the associated exception that the log was stored with.

4.B.iii C3 – Log

Operations:

Method Name	Arguments Passed	Expected Return Type	Description
Info	string	void	Logs a message with INFO as the tag.
Error	string	void	Logs a message with ERROR as the tag.
Warn	string	void	Logs a message with WARN as the tag.

Debug	string	void	Logs a message with DEBUG as the tag.
WithTag	string, string	void	Logs a message with a custom tag.

4.B.iv C4 – UserObject

Attributes:

Attribute Name	**Type**	**Description**
Email	string	The user’s Google email.
FullName	string	The user’s full name.
FirstName	string	The user’s first name.
LastName	string	The user’s last name.
ImageUrl	string	The user’s Google profile image url.

4.B.v C5 – CampaignObject

Attributes:

Attribute Name	**Type**	**Description**
Name	string	The campaign’s name.
Description	string	The campaign’s description, including lore, background, and setting.
Information	string	General information the dungeon master wants the character players to know for the campaign.

4.B.vi C6 – ConfigKeyObject

Attributes:

Attribute Name	**Type**	**Description**
Key	string	The key name of the configuration key.
Value	string	The value of the configuration key.

4.B.vii C7 – DynamoDbPersister

Attributes:

Attribute Name	**Type**	**Description**
Context	DynamoDBContext	The current connection to the DynamoDb.

Client	AmazonDynamoDBClient	The client used to establish a Context to the DynamoDb.

Operations:

Method Name	**Arguments Passed**	**Expected Return Type**	**Description**
Scan	<generic T>	List<T>	Scans the DynamoDb for all types of T.
ScanAsync	<generic T>	List<T>	Asynchronously scans the DynamoDb for all types of T.
Delete	<generic T>, Guid	Void	Deletes the specified type T with the specified Guid from the DynamoDb.
Get	<generic T>, Guid	T	Retrieves the specific type T with the specified Guid from the DynamoDb.
GetAsync	<generic T>, Guid	T	Asynchronously retrieves the specific type T with the specified Guid from the DynamoDb.
Save	<generic T>, object T	Void	Saves the specified Type T to the DynamoDb.

4.B.viii C8 – TableMap

Attributes:

Attribute Name	**Type**	**Description**
TableNamePrefix	string	The prefix to prepend to every table name added.
_tableNames	Dictionary<Type, string>	The hashmap dictionary list of all tables mapped.

Operations:

Method Name	Arguments Passed	Expected Return Type	Description
Load	none	void	Maps all the children of AbstractObject to their respective tables.
Add	<generic T>, string	Void	Maps a specified type T to a table suffix.
TableMap[Type]:get	None	String	Retrieves the full table name using the type Type.

5 State Transition Diagrams

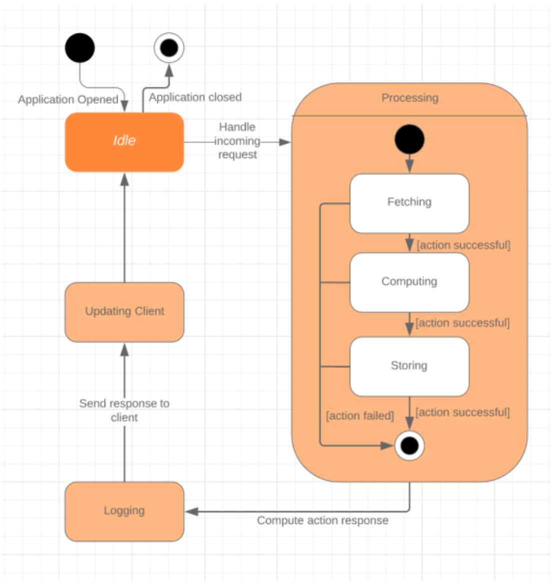
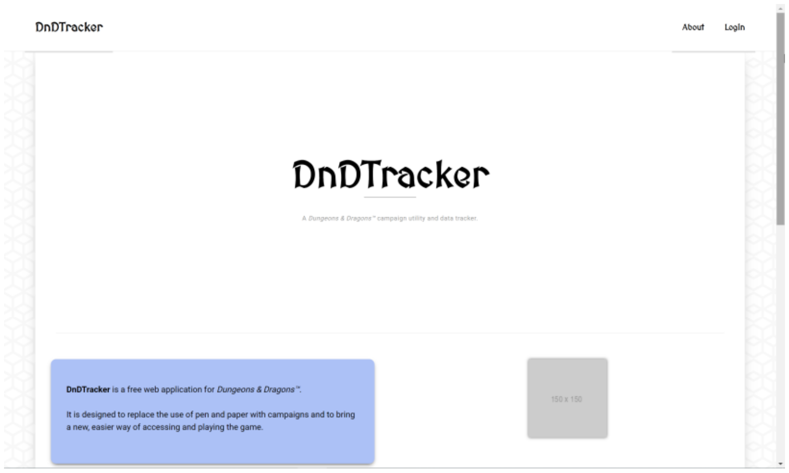


Figure 4 - State transition diagram for the server application

The server application begins in the Idle state as soon as it is opened, signifying the start of the state transition. Once the sever application receives an incoming request from a new or previous client connection, it transitions to the Processing state to handle the request. This state transitions through three sub-states: Fetching, Computing, and Storing. Through each of these states, the application is communicating with a variety of services, third-party integrations, and databases to fetch, compute, and store. Whether at any point one of the sub-states does not complete successfully or if the sub-states all complete successfully, the Processing state is exited with the response in hand. The application then transitions to the Logging state to store the response of the previously handled request. The server application then sends the response to the client to transition into its Updating Client state before returning to its Idle state, regardless of success.

6 User Interface Design

6.A Screen 1 – Home



A *Dungeons & Dragons*™ campaign utility and data tracker.

DnDTracker is a free web application for *Dungeons & Dragons*™.

It is designed to replace the use of pen and paper with campaigns and to bring a new, easier way of accessing and playing the game.

Figure 5 - The home page of DnDTracker

The Home screen is the first screen a user sees when connecting through a browser. Here, DnDTracker is described and showcased through a series of banner texts and images. The only two functionalities of this page are the About and Login buttons in the top-right. The About button takes you to a page describing the origin of the project and the roles of the developers behind the project. The Login button directs you to *Screen 2 – Login* where the user can sign-in with their Google account.

6.B Screen 2 - Login

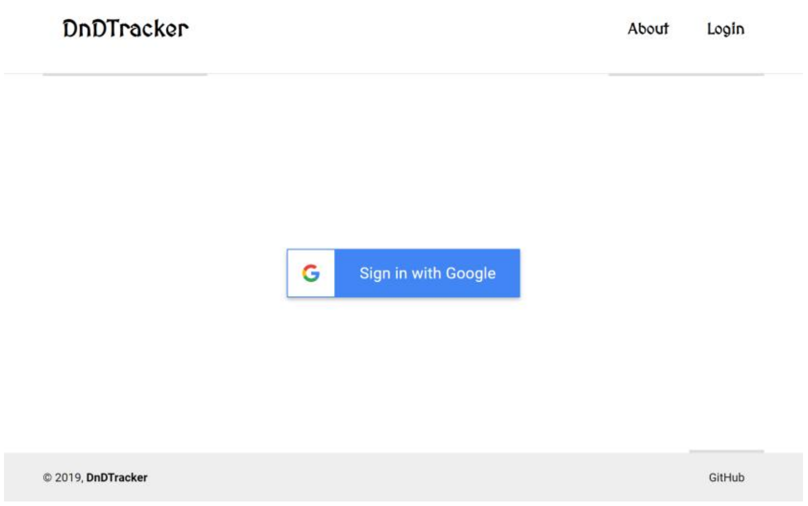
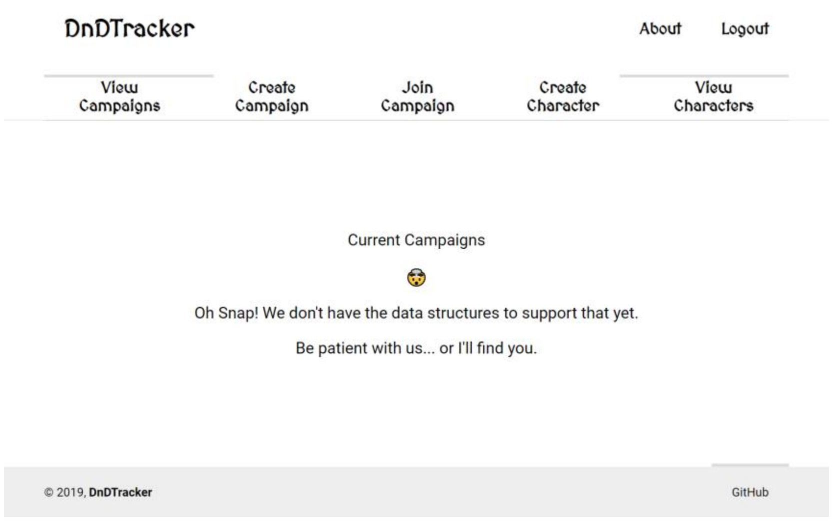


Figure 6 - The login page with a Google Sign-In widget

The Login screen is the screen a user sees when they click the Login (or Logout, when signed in) button in the top-right of every page. When the user is not signed in, only the Google Sign-In widget is visible. Clicking this button will either (a) open a separate Google Authentication window for you to sign-in with or (b) automatically sign you in through your previous Google Authentication attempts. Once you are signed in, this button will be replaced with your Google profile image and a sign-out button.

6.C Screen 3 – View Campaigns



Current Campaigns

Oh Snap! We don't have the data structures to support that yet.

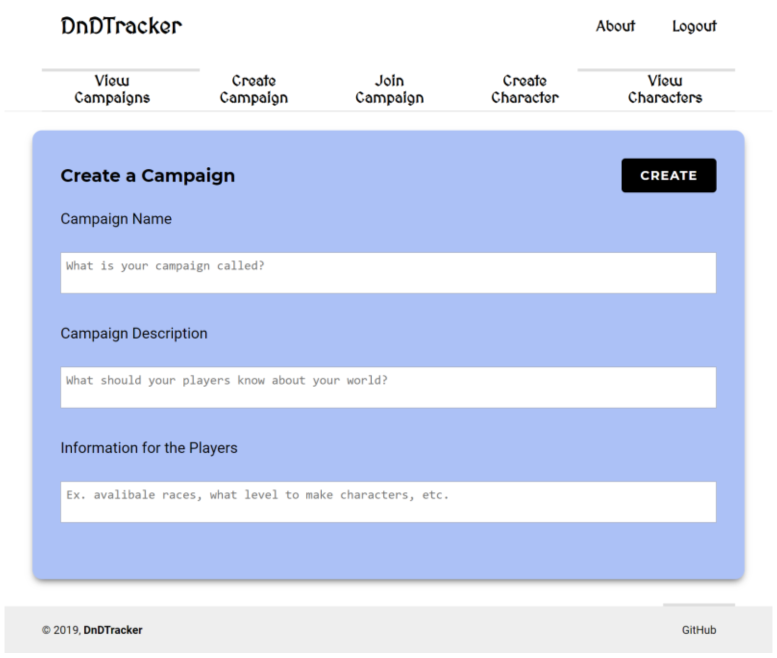
Be patient with us... or I'll find you.

© 2019, DnDTracker

Figure 7 - The view campaigns page where users can see their list of campaigns

The View Campaigns screen is the screen a signed-in user sees once they are redirected by the Login page automatically or once they click the View Campaigns button in the taskbar at the top of every page. On this page, each campaign that the user is a part of is listed with short descriptions and details pertaining to the campaign and its party members. Clicking the campaign name will direct you to the campaign's dashboard.

6.D Screen 4 – Create a Campaign



Create a Campaign

Campaign Name

What is your campaign called?

Campaign Description

What should your players know about your world?

Information for the Players

Ex. avalibale races, what level to make characters, etc.

© 2019, DnDTracker

Figure 8 - The create a campaign page

The Create a Campaign screen is the screen a signed-in user sees when they click the Create Campaign button in the taskbar at the top of every page. On this page, users are expected to input the name, description, and general information of the campaign in their respective fields. The Create button requests that the server create and store a campaign with the current signed-in user as the Dungeon Master. If for whatever reason the server is unable to complete the request, the user is notified and is able to try again. Once the campaign is created successfully, the fields are cleared out and the user is notified.

6.E Screen 5 – Create a Character

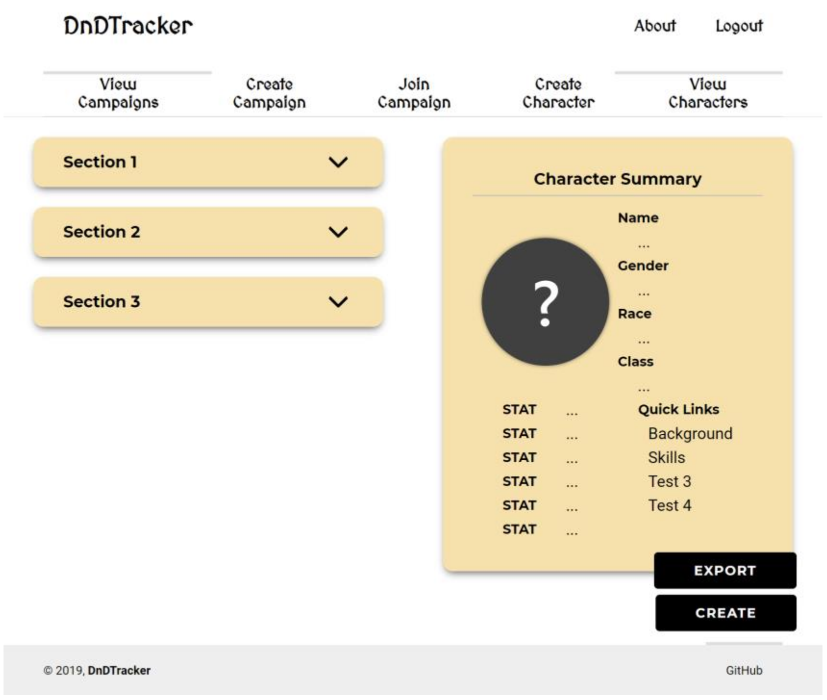


Figure 9 - The create a character page with separate sections and a summary

The Create a Character screen is the screen a signed-in user sees when they click the Create Character button in the taskbar at the top of every page. On this page, users are expected to at least fill the required fields in every section in order for their character to be creatable. Each section banner is a dropdown that expands to show the fields for each section. In the Character Summary, the Quick Links allow the user to quickly navigate the very long list of character creation fields. In the bottom right corner of the page are 2 buttons: Export and Create. The Export button will create a PDF document of the original Dungeons and Dragons Character Sheet with all of the lines filled in with the user's input on the page. The Create button will request the server to create and store the character. If successful, the user is notified and redirected to the View Characters page. If unsuccessful, the user is notified and is allowed to try again.

7 Conclusion

In designing DnDTracker and writing this Software Design document, we have learned the importance of architecture types, third-party integration, and user interface design. As we've come to learn, these three elements are equally integral to the integrity of a software application. Understanding the various types of architecture allowed us to implement a back-end solution that prioritized both an ease-of-workflow with development and stability from load testing. Users can trust in both the architecture of the application and the user interface to be both simple and consistent. Integrating third-party software challenged us with plenty of trial-and-error, where, despite having documentation, complicated our implementations. We faced these complications through local and sandbox environments of the services so as to avoid resource costs and promote diligent tinkering. These challenges allowed us to become more familiar with a line of tools and products we are comfortable implementing on future projects.