Team Root Beer Floating Point

Maitham Alghamgham, Jacob Morris, and Adam Fischer

Software Engineering Assignment 3

CS 400-01 Software Design and Development

Fall 2021

Architectural Design Document

Introduction

At its core, the Venture Forge product is an application which allows a user to create, use, and share character creations tools for TTRPG systems as well as discover and use similar tools created by other users. As such, this application requires some specific architectural components in order to function as specified: a comprehensive and easy-to-use user interface, access to the cloud so that users can access each other's created content, and a database to keep track of accounts and profiles as well as the actual content created by the users. In addition, the implementation of a microservice architecture can help to break features down into concrete bits of functionality that can be reused as well as prevent duplicate functionality within other components. However, with the introduction of all these components of the product, the system will need to be able to communicate between components, which points to the need for a service that allows for message queueing. Once all these components function in parallel, every feature and portion of functionality should combine to assemble the full Venture Forge product.

User Interface (XAML)

For a User Interface, Venture Forge will be taking advantage of XAML to create/design a UI. With XAML, the advantages of Microsoft's Windows Presentation Foundation (WPF) and have easy to use visual representation for development. XAML would also allow for easy integration with C# code to give functionality to the UI. Together with such advantages, it would be easier to connect visual styles and elements to better expand upon and make the friendliest user experience possible. Also, by using XAML elements and C#, it's possible when moving towards the development of mobile apps to reuse many of the elements for Xamarin as both Xamarin and XAML are used together in mobile app development.

Cloud Servers (Microsoft Azure)

Venture Forge would utilize cloud services such as Azure to utilize the advantages of Cloud Servers. With Azure, Venture Forge would be able to allow for the easy expansion of server sizes for databases over time as Venture Forge would have a great dependency on databases to function. While it is possible to use a local server instead, it's unknown how much

data would be necessary to keep available and how quickly the user base will evolve. A flexible system that would make it possible to increase the size of servers that can be up 24/7 is the most optical solution for Venture Forge. Also while there are multiple cloud services available, going with Azure seemed to be the best solution in that we could potentially utilize Microsoft's resources (such as on demand software support) to best configure Venture Forge and Venture Forge's mostly Windows based tools with Azure. There's of course the other advantage of being able to work on the server without being at any specific location (such as from home).

Database (SQLite)

At Venture Forge, a lot of data will be created by the user, such as accounts, profiles, and characters. To store these data, we need to keep track of it in a database. We plan on using SQLite an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine as our database. We can use SQLite to store and retrieve data in a lightweight database on the user device and more. Some benefits of SQLite it integrates with XAML well. SQLite is lightweight and self-contained. It's a code library without any other dependencies. There's nothing to configure. Microsoft recommends using Entity Framework Core or the open-source SQLite library built by Microsoft. Entity Framework (EF) is an object-relational mapper that you can use to work with relational data by using domain-specific objects. The Microsoft.Data.SQLite library implements the interfaces in the System.Data.Common namespace. Microsoft actively maintains these implementations, and they provide an intuitive wrapper around the low-level native SQLite API.

Message Queue (AMQP)

There will be a lot of data that need to be moved around within Venture Forge components. When doing microservices, a crucial design point is: how should my services communicate with each other? To establish good communication. we are planning to use AMQP (Advanced Message Queuing Protocol) an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security. With AMQP we will be able to connect the system, feeds components, and microservices with the information they need and reliably transmit onward the instruction that achieves their goals. Security is key, one of the advanced broker servers is RabbitMQ and it supports multiple message protocols such as MQTT, STOMP, and AMQP. RabbitMQ has a flexible authorization mechanism (access control, virtual hosts, and even third-party/custom auth services via many different kinds of transport such as HTTP or AMQP) and will basically remove the burden of orchestrating requests authorization from your service. Plug a custom microservice into Rabbit's auth backend and code your policies in there. The rest will be taken care of by the broker.

Microservices (Java Spring Boot)

Within a product such as Venture Forge which takes in large quantities of user input in its user creation functions, there will be certain tasks, such as cleaning data and user input, that will require frequent use. In order to isolate these sorts of tasks and prevent repeated creation of the same functions, Venture Forge will use microservices for these tasks. These will help to spread out functionality and increase the number of components that can be reused in order to simplify the rest of the components that comprise the Venture Forge product. In order to use this sort of microservice architecture, we plan to use Java Spring Boot, an open-source tool that aids in the development of microservices in the Spring Framework. Additionally, this integrates with AMQP, which helps to handle the movement of data between components of the architecture. Some of the other benefits of the inclusion of microservices within Venture Forge are the ability to develop these fragments of total system functionality in parallel with other components and the flexibility that comes with the ability to work with other programming languages during the implementation of the various features of Venture Forge. However, a microservice architecture also comes with some risks and difficulties that will be kept in mind throughout the development of the Venture Forge product. For example, communication between the front-end component and the microservices poses the threat of slower speeds for the application.