Magic 8 Ball Design Document

August 01, 2019

"This document is developed for the DevOps for Defense meetup and is for demonstration purposes only. This document and tooling that produces it are published to GitHub under an MIT Open Source license. Hopefully this is useful and may be tailored for use in your work.

Honestly it is amazing how much text the lawyers can provide for these title pages sometimes. Thankfully nobody seems to care when we make it all fit by using very small font. I guess it is technically there, so it will stand up in a court if things ever go terribly wrong."

Prepared by:	JD Black 123 Somewhere Madison, AL
Prepared for:	DevOps for Defense 321 Out There Huntsville, AL
Contract:	ABCD-123456-A1
CDRL:	A0123

Table of Contents

Table of Figures

Acronyms

Acronym	Description
TRL	Technology Readiness Level
TRL	Technology Readiness Level
TRL	Technology Readiness Level
TRL	Technology Readiness Level
TRL	Technology Readiness Level
TRL	Technology Readiness Level
TRL	Technology Readiness Level
TRL	Technology Readiness Level

1.0 Introduction

The Magic 8 Ball application is a digital representation of a well known toy (see Figure 1.0-1). The toy has mystical powers to answer any question you ask. You do this by asking the Magic 8 Ball a question, shaking it, and looking at the printed response on the bottom (note, you must invert the magic 8 ball to see this response).

Figure 1.0-1: The Magic 8 Ball toy to be replicated.

This project will create a digital representation of the Magic 8 Ball toy. The purpose of this is to provide a simple basis for demonstrating DevOps concepts and technology at the DevOps for Defense Meetup. This project was created because of the simplicity of the implementation. The point is to demonstrate DevOps solutions, not a fancy or complex software implementation. This design document is an aspect of what can be done to improve configuration management and document delivery through automation using DevOps principles. This is not meant to be perfect, but rather serve as an example for others to tailor to their needs.

2.0 Requirements

The Magic 8 Ball requirements are purposefully simple as the toy for which the project is based upon is simple.

Id	Requirement	Version	TADI
001	Magic 8 Ball shall receive a question.	V1	T
002	The Magic 8 Ball shall randomize a response when shaken.	V1	T
003	The Magic 8 Ball shall provide a response when viewed.	V1	T

2.1 Interface Requirements

The Magic 8 Ball interface requirements are taken from the Magic 8 Ball standard as described on Wikipedia. This standard defines 20 standard responses that may be provided to any question.

The following responses are allowed.

Type	Text
Affirmative	It is certain.
Affirmative	It is decidedly so.
Affirmative	Without a doubt.
Affirmative	Yes - definitely.
Affirmative	You may rely on it.
Affirmative	As I see it, yes.
Affirmative	Most likely.
Affirmative	Outlook good.
Affirmative	Yes.
Affirmative	Signs point to yes.
Non-Committal	Reply hazy, try again.
Non-Committal	Ask again later.
Non-Committal	Better not tell you now.
Non-Committal	Cannot predict now.
Non-Committal	Concentrate and ask again.
Negative	Don’t count on it.
Negative	My reply is no.
Negative	My sources say no.
Negative	Outlook not so good.
Negative	Very doubtful.

2.2 Derived Requirements

The Magic 8 Ball derived requirements are based on the design strategy. Because the Magic 8 Ball is a simple command line application, the derived requirements in Table 2.2-1 revolve around command line user interaction.

ID	Requirement	Version	TADI
D001	The Magic 8 Ball shall allow the user to provide multiple questions in sequence.	V1	T
D002	The Magic 8 Ball shall terminate if the user provides a response of upper or lower case Q.	V1	T

3.0 Design

The Magic 8 Ball software design is currently in the Minimum Viable Product (MVP) phase. The design and implementation are purposefully simple but perform the basic functions of a Magic 8 Ball in a demonstrable way. The desired behavior of this MVP is captured as a sequence diagram in Figure 3.0-1.

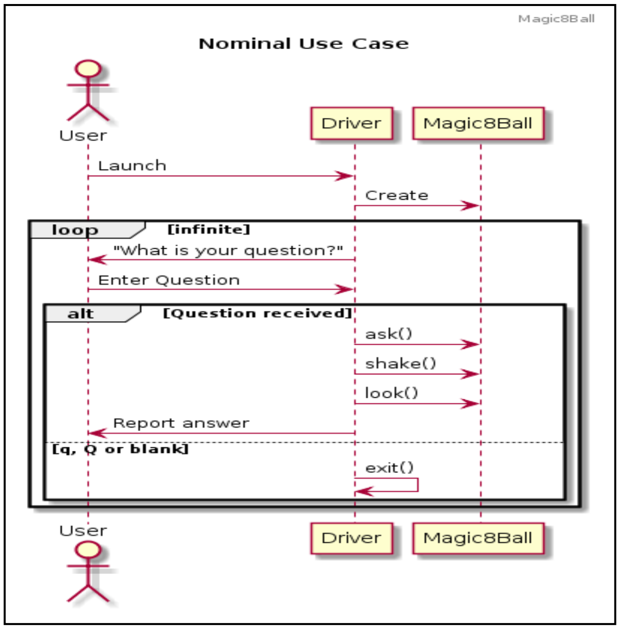

Figure 3.0-1: The Magic 8 Ball behavior sequence diagram.

As you can see, the user interacts with a driver. This driver is simply a main which can be launched from the command line. The driver will create an instance of the Magic 8 Ball and then ask the user to provide a question. Once the user provides the question, the driver provides the question to the Magic 8 Ball, shakes the Magic 8 Ball, and looks at the response provided by the Magic 8 Ball. The driver then provides that response to the user. At this point, the process is repeated by asking the user for another question. If the user chooses to quit, rather than providing a question the user may simply type “q”, “Q”, or hit enter. This will cause the driver’s user interaction loop to terminate which in turn will terminate the application.

The Magic 8 Ball class diagram is also very simple. The core classes represented in Figure 3.0-1 are the Driver class and the Magic 8 Ball Interface. Figure 3.0-2 provides the class diagram for the Magic 8 Ball. As you can see, there are 2 concrete implementations of the Magic 8 Ball interface. First is the Standard Magic 8 Ball which must perform as a standard off the shelf Magic 8 Ball would perform, including providing the standard responses in a randomized manner. The second is the DevOps Magic 8 Ball which extends this functionality by appending a DevOps quote to the Magic 8 Ball response.

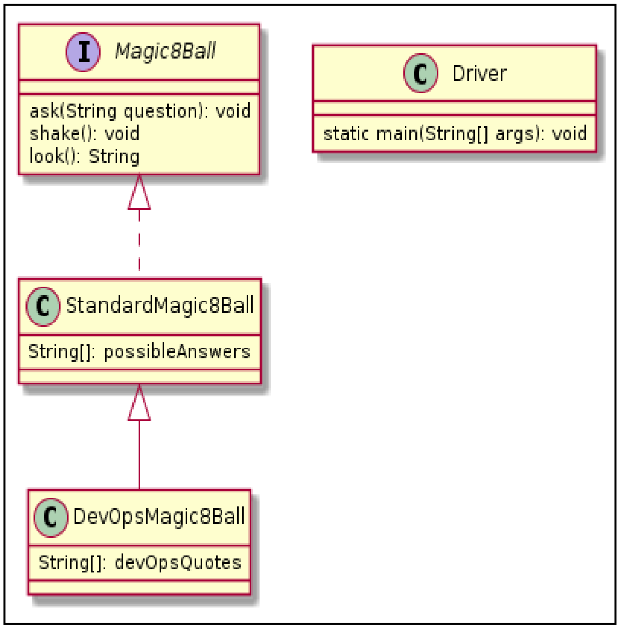

Figure 3.0-2: The Magic 8 Ball class diagram.

4.0 Test

The Magic 8 Ball testing section was going to include graphs and figures generated during the execution of the pipeline, but since other sections already demonstrate this I’m just going to skip this for now. Yes, I’m being lazy…but in reality I already spent a few days putting this together so I feel OK about it.

5.0 Deployment

The Magic 8 Ball currently has no plans of ever being deployed.

If you want to try and demonstrate this for yourself, it takes some setup of your host machine to ensure all dependencies are in place. I have not done a good job of maintaining all the necessary dependencies. I intend to correct this in a future version of the project.