# Brief

The objective of this project is to create an application made up of 4 services using the training I have received over the past few weeks. These services will communicate with each other and return an object to the user. The project must have full documentation of the supporting tools used to develop and manage the project as well as justification for the reasoning behind said tools and all design decisions.

# Scope

The project requirements:

* Kanban Board with expansion on tasks and records of any issues and risks
* The application must use CI i.e. must be a part of a Version Control System (VCS), CI server which is connected to the VCS via webhooks and be deployed on a cloud-based machine
* Must have a service-oriented architecture

* Must use containerisation for deployment.
* The project must use an Ansible Playbook

# Solution

The idea for the web app is a Dungeon and Dragon wild surge effect app. A user will be prompted, to roll two dice which will return to the user the number they rolled and a description of a magic effect.

### Service 1

* This service provides the frontend for the user and is the only visible page for the user. It sends a request for the effect description to service 4.

### Service 2

* This service emulates a 10-sided die roll by randomly generating a number in the tens i.e. 0, 10, 20, 30, … it then returns the value generated

### Service 3

* This service also generates a number much like service 2 it generates a number from 0 to 9

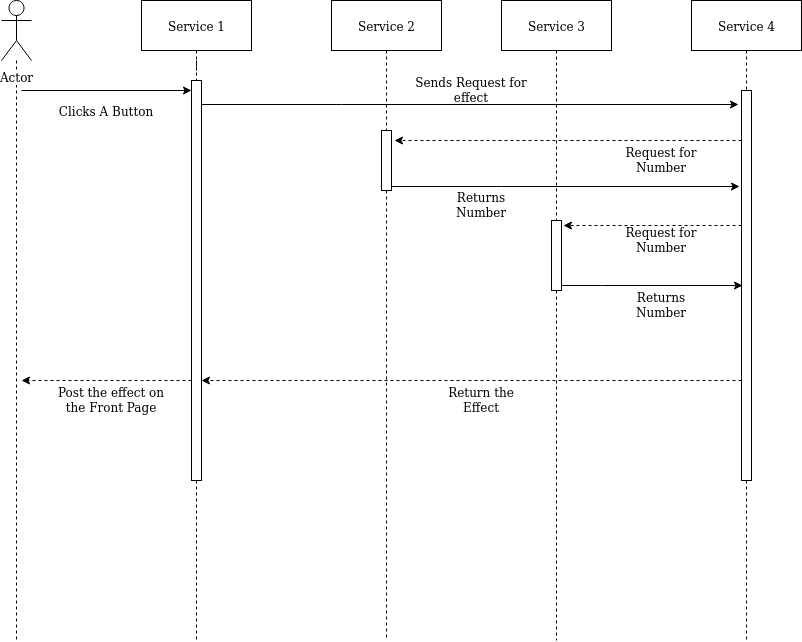
### Service 4

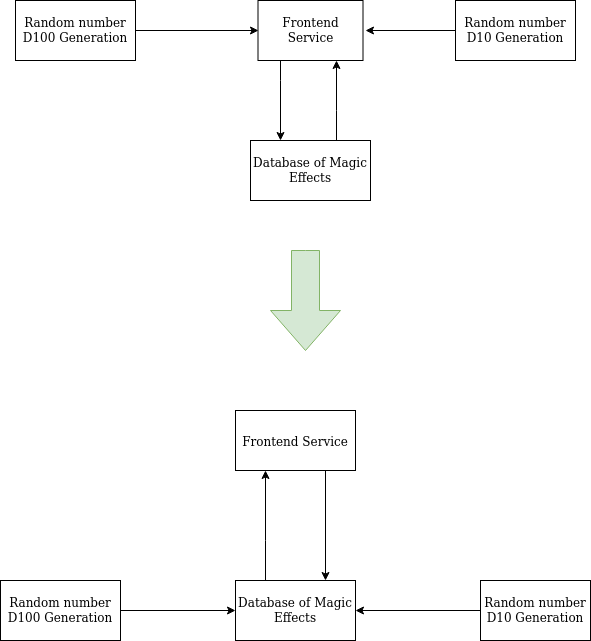
* This service sends a request to services 2 and 3 for the number they generated and then queries the database for the sum of the numbers from 2 and 3 for the matching id and effect. The information from the database is returned for service 1 to request and show to the user.

# MuSCoW

|  |  |
| --- | --- |
| **Must** | Allow Users to roll the dice as many times as they want |
| **Must** | Must be easy to use and the page should update very quickly with an effect |
| **Must** | The database must have at least a hundred unique effects in it fully accessible by the app |
| **Should** | The app should be updateable with zero downtime |
| **Could** | Have multiple table of effects with different number generation depending on the No. of effects in that table. |
| **Would** | Add user accounts and allow users to alter the effects that are produced on certain dice rolls |

# Architecture



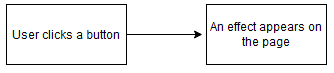


# Risk Management

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Risk | Impact | Action | Owner | | Measure |
| IPs of Instances Changing | The web app or Jenkins would be unavailable until the new IPs are whitelisted. (Medium Impact) (less likely) | Tolerate/Treat | | Ife | The app Instance has been given a static IP so there is no worry of losing connection to the database and service will be uninterrupted however being unable to connect to Jenkins could cause short maintenance update delays. |
| Nginx server down | The web app would be inaccessible to the user (High Impact) (less likely) | Tolerate | Nginx | | The servers are not likely to go down so this can be tolerated in the event this does happen one solution is to have another proxy as a backup which the app can quickly change too. |
| Database is attacked via hacking | The web app would be unavailable and there is a good chance the data in the app will be deleted or/and held hostage. (High Impact) (less likely) | Terminate | Ife/Google | | I have made sure to hash passwords and to not add important passwords or information on GitHub I will depend on Google to protect the VMs from other kinds of attacks |

# Testing

# User Story



# Future Improvements