# Flask Assignment

This application was created using Flask framework. Databases are not used in here, because the given task can be achieved completely and easily by using data structures. Since it doesn’t require a database in the task significantly, I used an array of Joke objects to store the jokes in locally. In there, below jokes are stored.

1. Newly created jokes
2. Updated jokes from remote site
3. Deleted jokes from remote site

Whenever a new joke is added, it is stored in this array. When a user updates a joke in remote site, a copy of joke with updated value is stored in here. Only this joke(not the remote joke) will be visible to the user. When a user wants to deletes a joke, then a copy of the remote joke is kept in here and the variable ‘removed’ is changed to true. Therefore, both the joke from local array or remote server is not visible to the user hereafter.

I followed below steps to complete this.

* First, I created the Flask API requested end points
* In here, I created a model class named ‘Joke’
* Add test cases for each end point
* Documentation using Swagger UI
* Containerizing the application using Docker
* Added more extra features.
  + Add an endpoint to get all local jokes (/api/getAllLocalJokes) . Because of this, I can easily check the jokes after done some change to a joke in a local joke list.
  + Add an endpoint (/api/jokes/imageJoke/) to add another type of joke. It is called as ImageJoke. In here, this class is inherited from Joke class and it has additional ‘url’ value. We can create a joke with url in here. The purpose of creating this is to show how we can use this class for future additions.

Open-source libraries used:

* Flask: This provides a simple and lightweight way to create APIs.
* Flask-Swagger-UI: This is an extension for Flask that integrates Swagger UI into Flask applications. This is used because it allows to generate and serve Swagger documentation for the API without any complexity and it generates a clear, user-friendly documentation.
* Flask-Testing: This provides testing utilities and tools specifically designed for Flask applications which simplifies the process of writing tests for Flask-based APIs.
* Docker: Enables to package applications and their dependencies into lightweight, portable containers. It provides portability, scalability, efficiency, isolation, and supports DevOps practices.