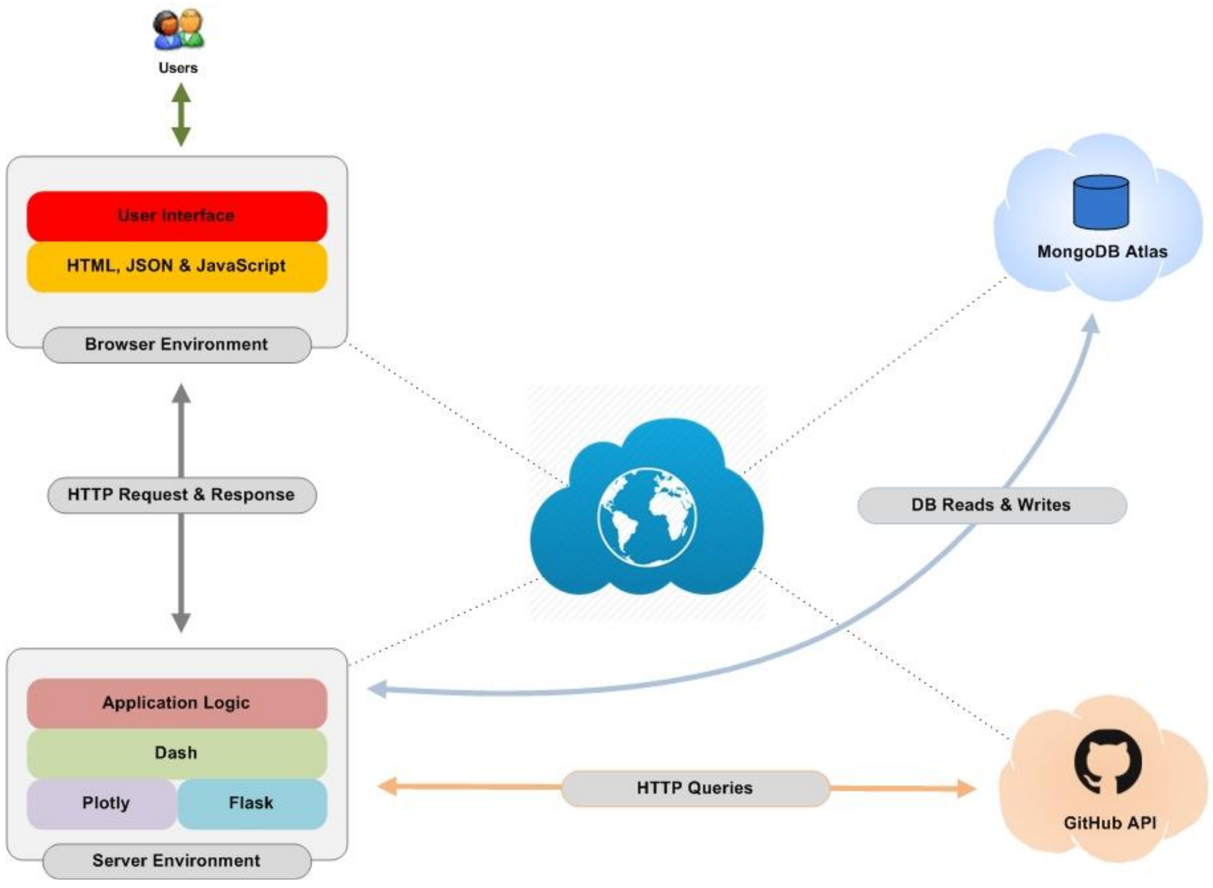
CSU33012 Software Engineering

Data Visualisation Project

Keira Gatt (#19334557)

Table of Contents

1 Architecture



A The Data Visualisation (DV) project is implemented as a web application, using the technology stack shown in the architecture diagram above. The overall design is essentially made up of 2 sets of components - one that runs on the backend infrastructure (server-side) whereas the other executes in the context of a client-side user agent, i.e. the web browser.

B The *Server Environment* is built entirely with Python and consists of the DV application logic and a number of Python framework modules that include *Dash* (the core components), *Plotly* (the visualisation library) and *Flask* (the web application services). Given that this is a web-based application, the server infrastructure incorporates all the runtime code necessary to run the DV application which includes user input processing, URL routing, the core application logic and the production of dynamic content that is delivered to the *Browser Environment* for rendering.

C In addition to handling user requests, the server code is also responsible to interact with 3rd party Internet sites in order to obtain the required data (*GitHub*) and to store the intermediate results, pending further processing (*MongoDB*).

D Each user query received by the server enters a 4-stage processing cycle before a response is sent back to the client. These are as follows –

i) Validate user input and route internally to the appropriate subsystem

ii) Assemble HTTP queries based on the user input, send to GitHub and store the response data on MongoDB Atlas.

iii) Using the data stored in the Mongo database, parse and filter the results received from GitHub and prepare the visualisation frames.

iv) Create the required output layout using the frame objects and assemble the HTTP response as a combination of HTML, JSON containers and JavaScript (JS) code.

E All network communications is carried out using the HTTP Request & Response model. By default, the DV application server accepts connections on all network interfaces (including the *loopback* interface) using port 8050/TCP. Network traffic between the *Browser Environment* and the *Server Environment* is based on *Cleartext* HTTP whereas sessions between the *Server Environment* and *MongoDB Atlas* or *GitHub* use *TLS 1.2 / 1.3* encrypted channels.

F In addition, access to 3rd party sites from the DV application is authenticated by means of a username and password for *MongoDB Atlas* and an access token for *GitHub*, embedded in the HTTP requests. The GitHub API access tokens are valid for a limited time period only and will need to be reset once they expire.

2 Implementation

2.1 Components

app.py	Python code for server declarations
barchart.py	Python code to find the languages that are used the most by the Github user and render the output as a Bar Chart
Country_List.py	Array list of *ISO 3166* country names
Dockerfile	Declarations file required to assemble *Docker* image
dv.png	DV application logo image
favicon.ico	DV application browser address bar image
globals_vars.py	Container for shared global variables
home.py	Python code used to render the site's main menu
index.py	Python code used for initialisation and URL routing
map.py	Python code to find the country's top GitHub repository and render the output as a geographical World Map
piechart.py	Python code to render Pie Charts showing a breakdown of languages for the GitHub user repositories
requirements.txt	File containing a list of Python modules required by the DV application. This file is typically used by the *pip* package installer
reset.css	Simple style sheet used to reset the background

2.2 Development Environment

OS	MS Windows 10 64-bit
IDE	PyCharm 2021.3 (Community Edition)
Language	Python 3.10.0

2.3 Packaging

/DataVis	Dockerfile requirements.txt		
	/src	app.py index.py	
		/apps	barchart.py Country_List.py globals_vars.py home.py map.py piechart.py
		/assets	dv.png favicon.ico

2.4 Runtime Prerequisites

Python	The DV application was tested with Python 3.10.0 on MS Windows and Python 3.8 running on native Linux and in Linux-based *Docker* image
Modules	The runtime environment requires the following Python modules (including any dependencies) : dash dash_bootstrap_components pandas pymongo requests dnspython <u>Note</u> All required modules are already included with the *Docker* image
Network Port	The application is preconfigured to listen for incoming connections on any host interface using TCP port 8050. It is therefore important that the application server can bind with this network port at startup. If TCP port 8050 is being used by another application and cannot be freed, it is possible to specify a different port number by modifying the *port* parameter at line 44 in source file *index.py*. If running under *Docker*, the listening port can be altered by using the *publish* or *expose (-p, --expose*) command line options.

2.5 Host OS Runtime Environment

DV Application	From the command line, change directory to *DataVis/src* Launch the DV application by entering the command : ***python ./index.py***
User Interface	Start web browser and enter the following URL in the address bar : ***http://<IP-Address>:8050/*** where <*IP-Address*> can be any interface address of the application host or simply *localhost (127.0.0.1)* if the web browser and the DV application are running on the same machine.
Termination	From the console window running the application, enter *^C* (MS Windows) or *^D* (Linux)

2.6 Docker Runtime Environment

DV Application	From the command line, ensure that the Docker Engine is running : ***docker info*** Launch the DV image as a runnable container : ***docker run -d -p 8050:8050 kmg2021/datavis:release_version*** Note i) These commands may need to be run from a console window with high privileges (*sudo, root or Administrator*) ii) *-d* : set the container to run in the background *-p* : publish container port to the local host
User Interface	Start web browser and enter the following URL in the address bar : ***http://<IP-Address>:8050/*** where <*IP-Address*> can be any interface address of the *Docker* host or simply *localhost (127.0.0.1)* if the web browser and the *Docker* container are running on the same machine.

Termination	From the command line, list the active *Docker* containers and take note of the *CONTAINER-ID* associated with image ***kmg2021/datavis:release_version*** ***docker container ls*** Stop the DV application container : *docker stop <CONTAINER-ID>*

3 User Guide

3.1 Repositories

Source Code	GitHub	https://github.com/KeiraGatt/Software-Engineering-Metric-Visualisation
	TCD-BB	**Data Visualisation - Keira Gatt - 19334557.zip**
Image	Docker	**kmg2021/datavis:release_version**
Note	The GitHub version does not include the access tokens required by the application to interact with the GitHub API and MongoDB Atlas. Any valid GitHub access token embedded in the code is automatically revoked by GitHub once the source code is uploaded to the repository. For the GitHub version, the access tokens have been replaced by placeholders in file ***/DataVis/src/apps/global_vars.py*** at lines 24 and 25.	

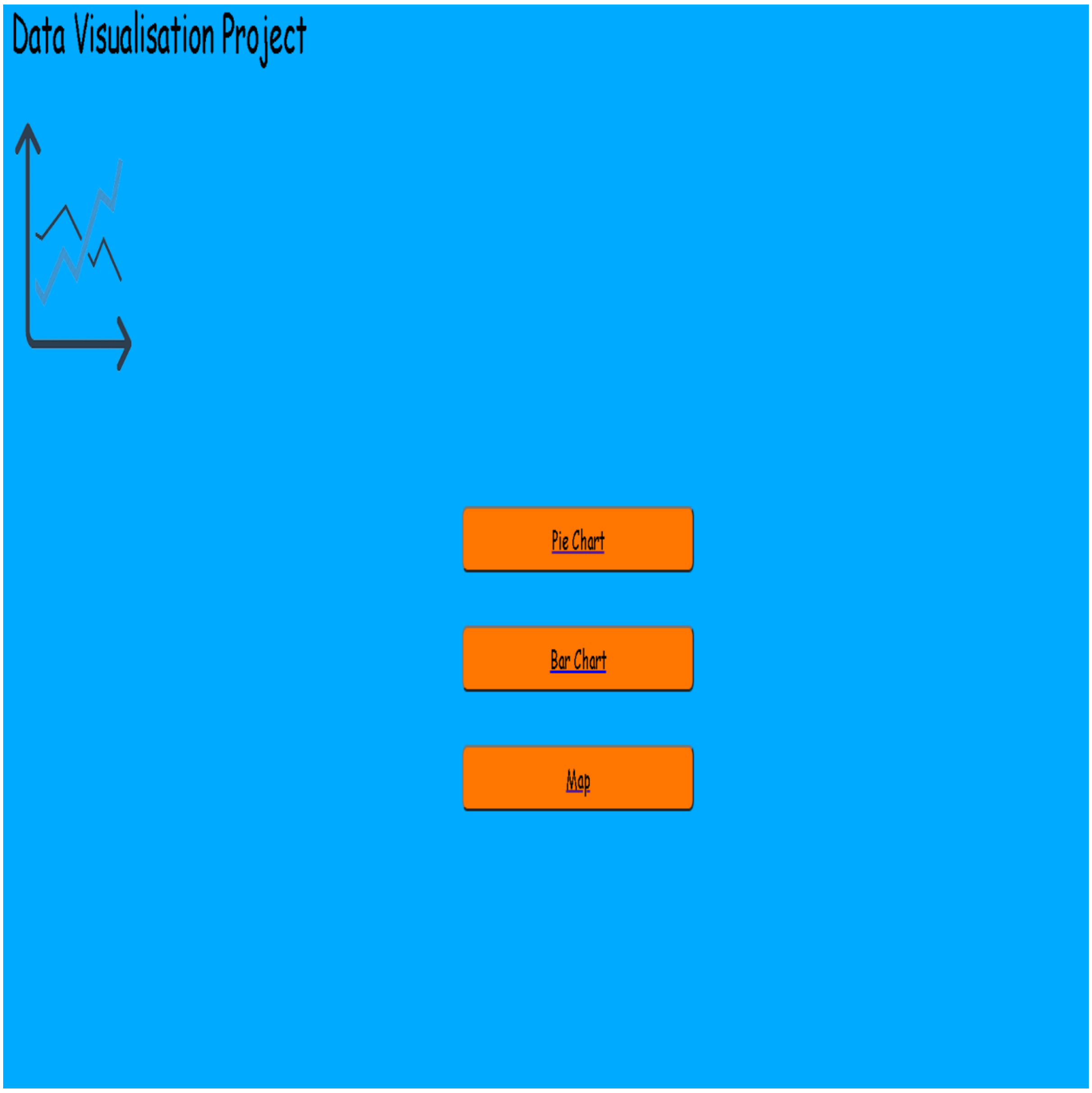
3.2 Convention

Input Format	Tests suggest that case-insensitive strings can be used when querying the GitHub API with usernames and names of repositories. Although this might be true for GitHub, the DV application does not make any assumptions in this regard and query strings are built with the exact same case used in the user input. However, the recommendation is to specify names of users and repositories in the same format as published on GitHub.	
Input Controls	A user input is confirmed by clicking on the **Submit** button, whereas clicking on the **Home** button will cause the DV application to display the *Home Page*.	
Default Settings	The DV application is configured with the following presets :	
	GitHub Username	**KeiraGatt**
	GitHub Respository	**LowestCommonAncestor**
	Top Repository Country	**Ireland**

	These default settings are used on the first call to any of the options on the *Home Page* and whenever an error occurs either because the user input fails the validation checks or because an error occurs while processing the user request. Whenever a request cannot be completed due to an error, a message describing the event is displayed on the main view panel of the output screen.
System Response	On rare occasions, users might experience a slight delay in getting back a response to one of their queries. This is not the result of some internal technical issue but because of slow response times incurred by the system when trying to communicate with 3rd party Internet sites. It is worth noting that the DV application makes use of freely available 3rd party resources and it is normal for site owners to impose limits on connection rates, bandwidth utilisation, etc. to prevent their systems from getting overwhelmed with user traffic.

3.3 Home Page

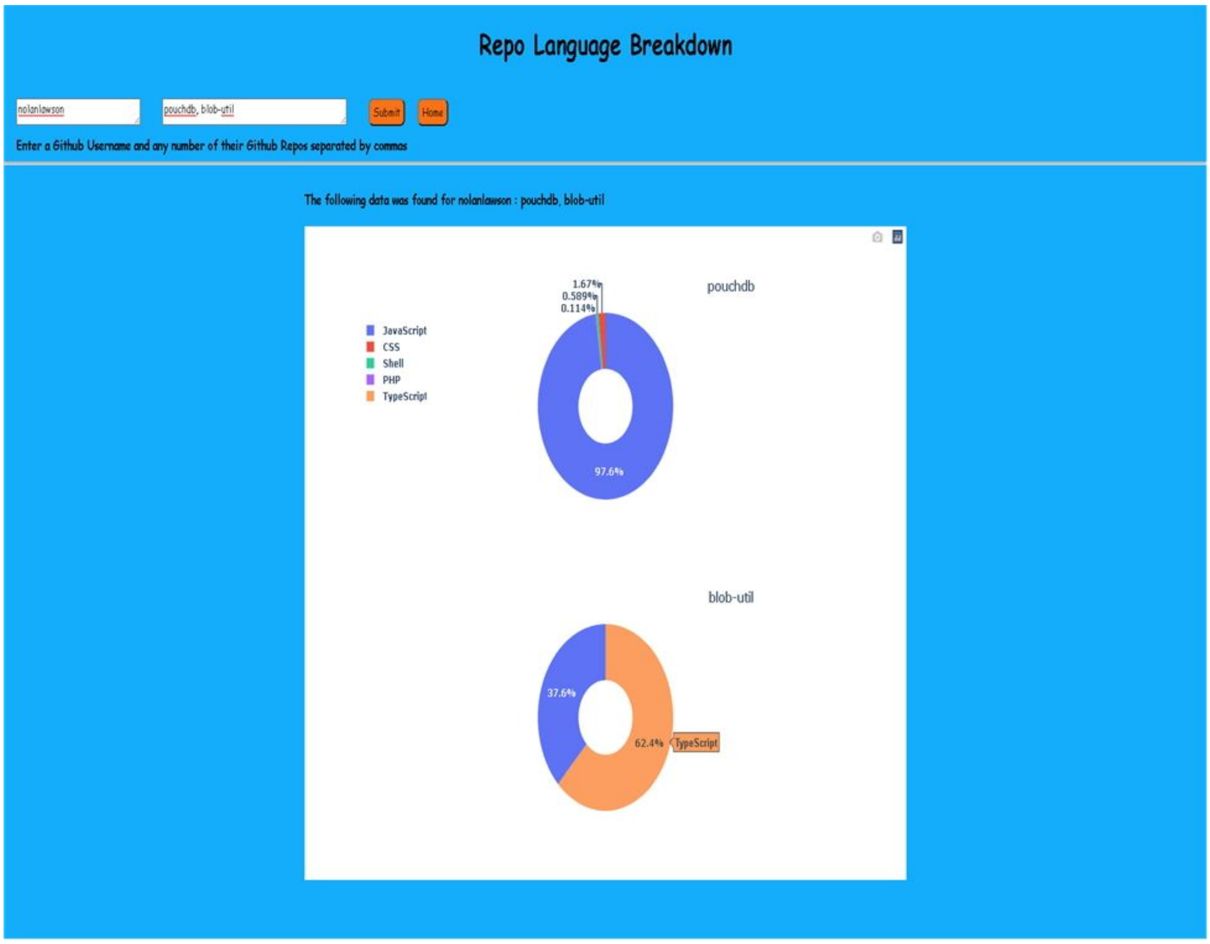
User Input	Select one of the following options :	
	Pie Chart	Breakdown of languages for GitHub user repositories
	Bar Chart	Most used languages by GitHub user
	Map	Top GitHub repository by country



DV Application Home Page

3.4 Pie Chart

User Input	*Field-1*	GitHub Username
	Field-2	A GitHub repository name associated with the GitHub Username or A list of comma-separated GitHub repository names associated with the GitHub Username
System Output	Each repository for the GitHub user is represented as a Pie Chart, showing the languages involved and the extent of their use, evaluated as a percentage of the total. The user can also interact with the Pie Chart by hovering on the segments to highlight the corresponding language and by clicking on the legend colour icons to toggle individual languages on or off.	


Repo Language Breakdown

nolanlawson

pouchdb, blob-util

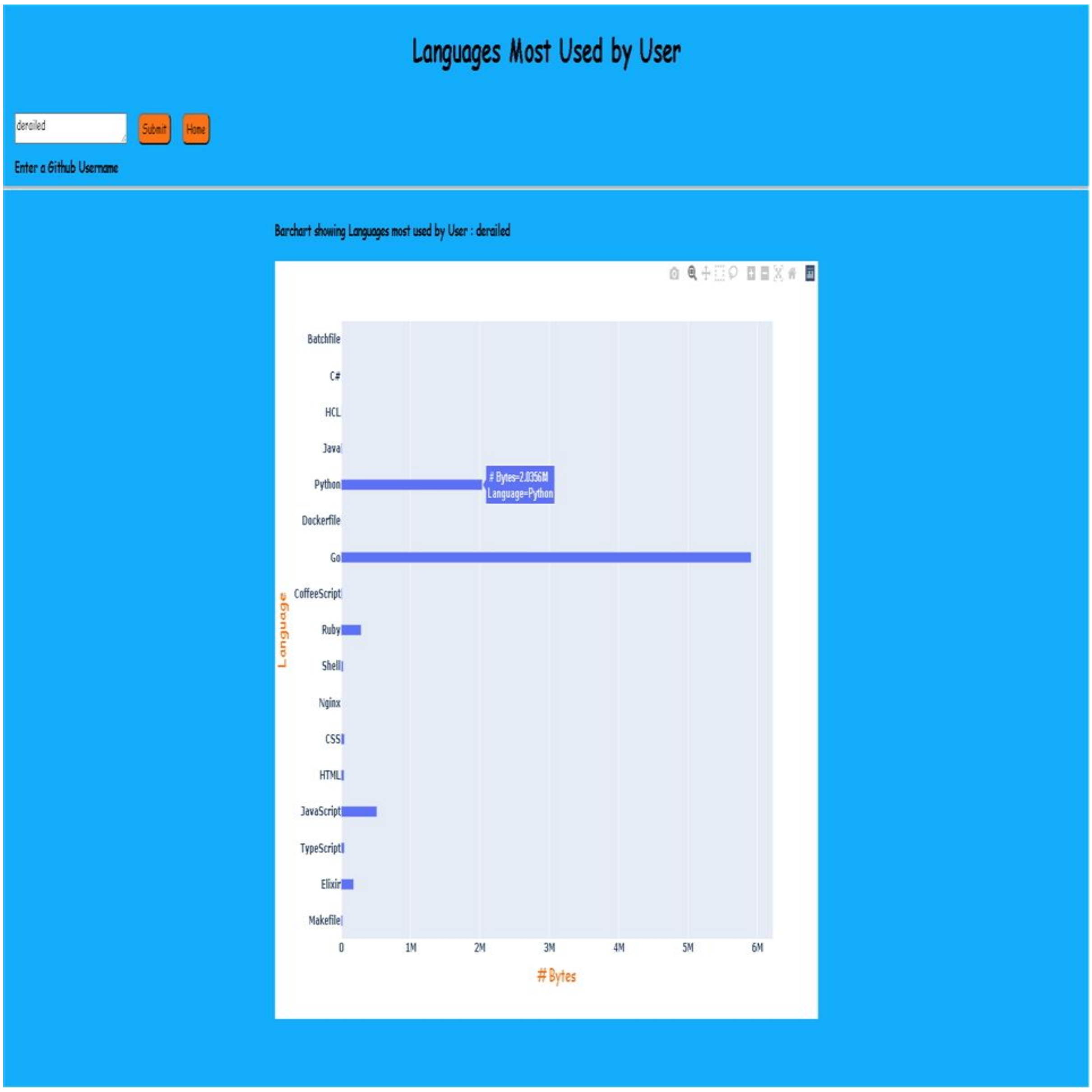
Enter a Github Username and any number of their Github Repos separated by commas

The following data was found for nolanlawson : pouchdb, blob-util

Pie Chart displayed for GitHub Username *nolanlawson* and repos *pouchdb* and *blob-util*

3.5 Bar Chart

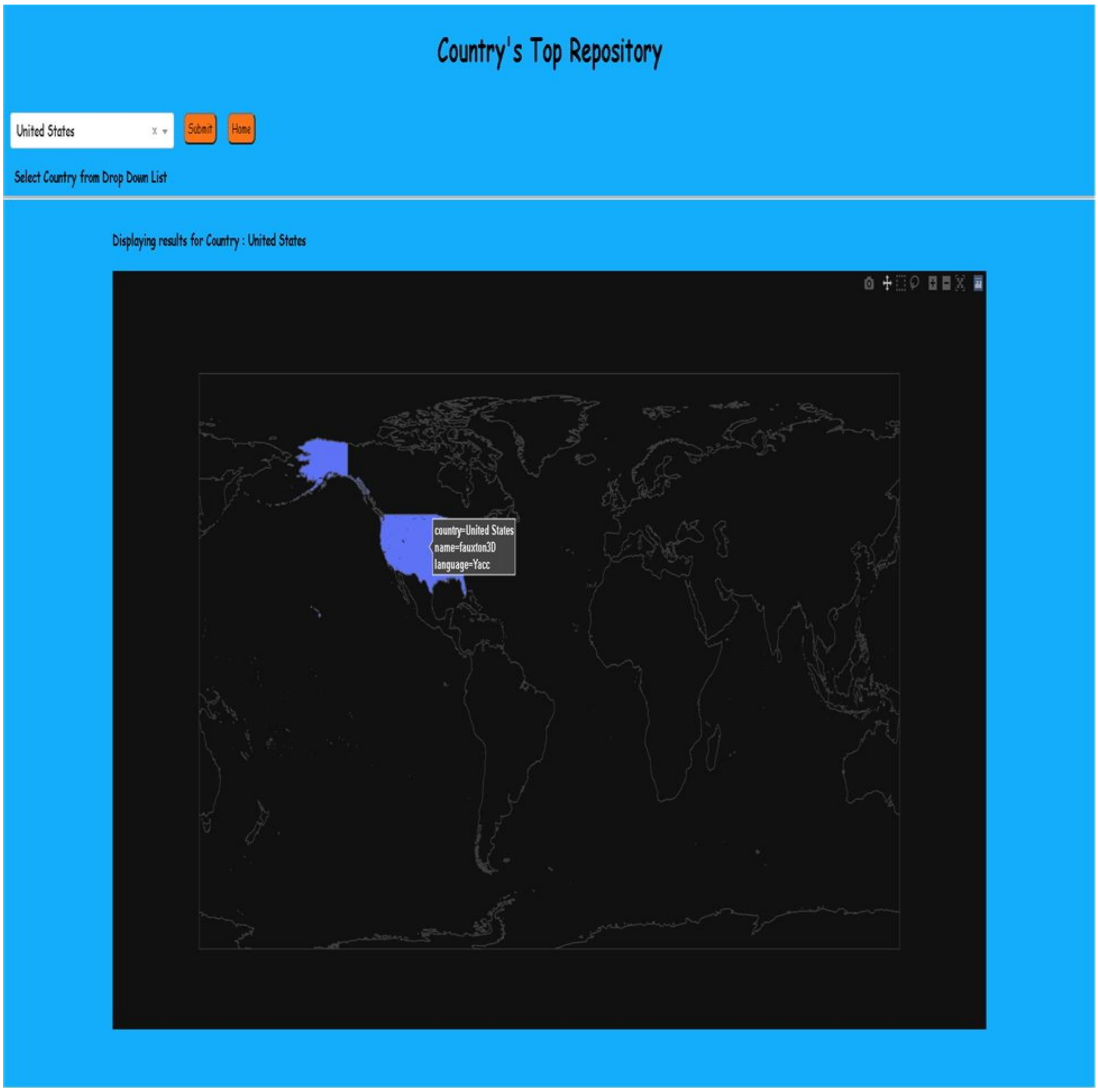
User Input	*Field-1* GitHub Username
System Output	A bar chart portraying all languages associated with the GitHub user, where the aggregate use of each language is expressed in terms of the volume of code submitted and categorised under that language by the user. The interactive chart allows the user to hover on each bar to highlight the corresponding data and provides the user with a tool bar that can be used to zoom, pan, etc.



Bar Chart displayed for GitHub Username *derailed*

3.6 Map

User Input	*Drop-Down List Selection* Country Name
System Output	A geographical world map showing the location of the selected country. Once the user hovers on the map's highlighted region, the top GitHub repository for the selected country and the language used for its implementation are displayed. The map is also interactive where the user is allowed to pan, zoom and save portions of the map based on selected areas. Note that zooming in and out can also be achieved with the use of the mouse scroll wheel.



Map showing the top GitHub repository for the *US*