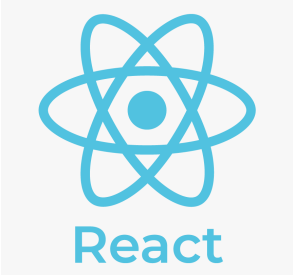
Data Foundation Systems

Assignment 1
Submitted by: Yagyesh Purohit
Roll number: 2020201013



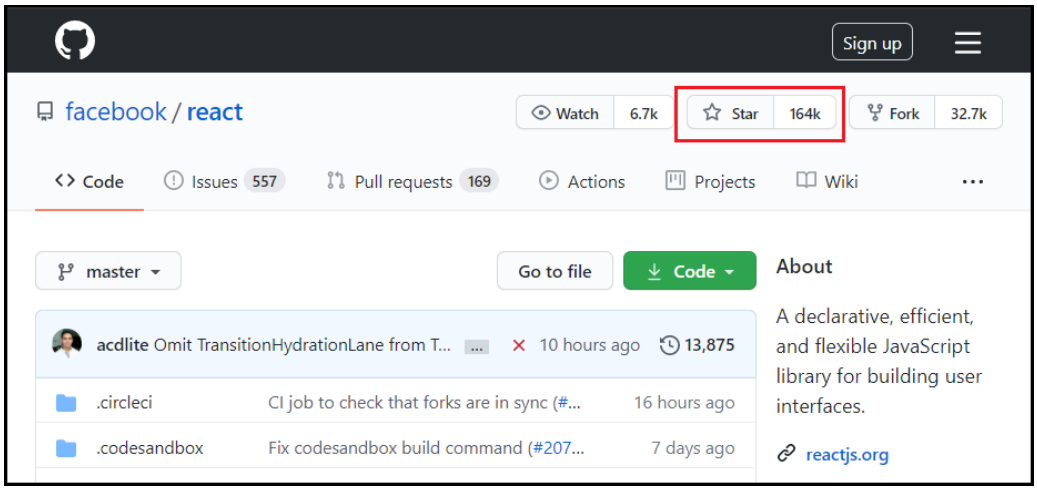
React is a JavaScript library for building fast and interactive user interfaces, developed at Facebook in 2011. It is one of the most popular JavaScript libraries for front end development. When building applications with React, we build pieces of UI called **components**, which are independent, isolated and reusable so that we can compose them to build complex user interfaces. Every react app has at least one component, which is referred to as the **root** component.

Why ReactJS?

React is Flexible: React is remarkably flexible. Once you have learned it, you can use it on a vast variety of platforms to build quality user interfaces. React is a library, NOT a framework. Its library approach has allowed React to evolve into such a remarkable tool.
React was created with a single focus: to create components for web applications. A React component can be anything in your web application like a Button, Text, Label, or Grid.
"Learn React Once and Write Everywhere" - Reactjs.org
You can use React in your existing apps too. React was designed keeping this in mind. You can change a small part of your existing application by using React, and if that **change** works, then you can start converting your whole application into React.js. Facebook used the same approach.
Greater performance: The React team realized that JavaScript is fast, but updating the DOM makes it slow. React minimizes DOM changes. And it has figured out the most efficient and intelligent way to update DOM.

React monitors the values of each component's state with the Virtual DOM. When a component's state changes, React compares the existing DOM state with what the new DOM should look like. After that, it finds the least expensive way to update the DOM.

Broader Community Support: Since 2015, React's popularity has grown steadily. It has a massive active community on Github and StackOverflow. Reactiflux is a community specially made for React Developers. Over 110k community members are involved in helping solve and share React-related problems.

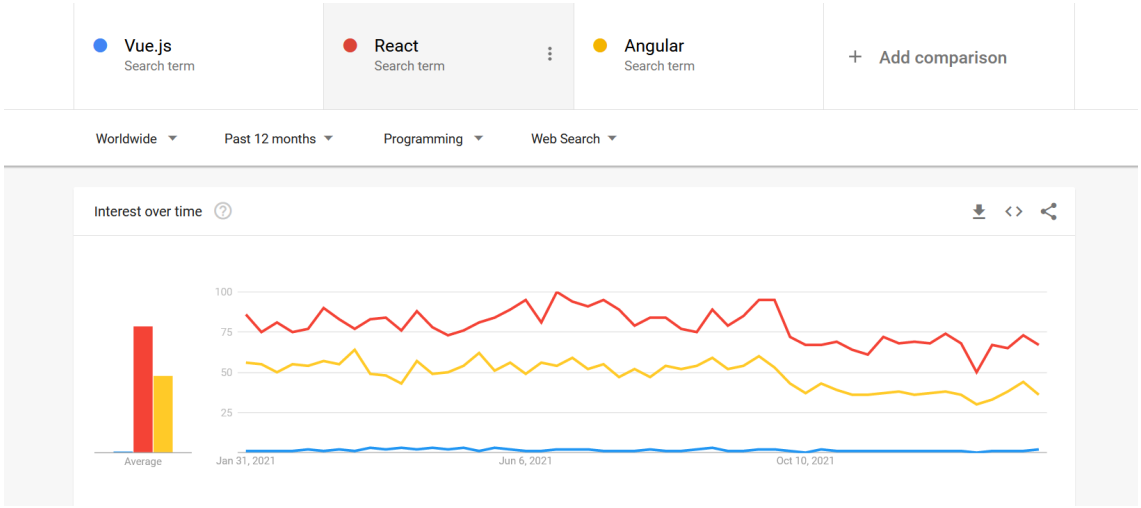


Comparison

While there are many JavaScript libraries and frameworks present in the market, React is mainly compared with two popular JavaScript frameworks, namely Angular and Vue.js.

Popularity:

Following is the graph of preferred frontend technologies, showing the popularity of React in terms of usage over Angular and Vue.js.



Performance:

React is a UI library and, unlike other frameworks and libraries, it doesn't 'enforce' an architecture pattern. A React component is essentially a minimalistic central structure unit – as simple as a label & a button or a bit complex as a user profile or a registration form. It is just a view that caters to your web app's user interface.
Angular, being the oldest one of the lot, has the Model-View-Controller (MVC) architecture. MVC architecture is the classic old way of separating even the most complex UIs into three views – Business Logic, UI Logic, and Input Logic. The execution is simple.
Angular relies heavily on real DOM for its performance while Vue.js and React function using virtual DOM. Virtual DOM saves the trouble of re-rendering/repainting your web app's UI after every state change. This makes it quicker, especially when multiple component states are involved.
When it comes to memory requirements and booting time, React & Vue.js show superior results to Angular.

References:

- https://reactjs.org/docs/getting-started.html
- **React JS - React Tutorial for Beginners**

Flask

Flask is a web application framework written in Python.

Flask is a light-weight framework popularly categorized as a micro framework (Micro-frameworks are the opposite of full-stack frameworks, which also offer additional modules for features such as authentication, database ORM, input validation and sanitization, etc.). Flask comes with some standard functionalities and allows developers to add any number of libraries or plugins for an extension. It only provides the necessary components for web development, such as routing, request handling, sessions, and so on.

Some features which make Flask an ideal framework for web application development are:

1. Flask provides a development server and a debugger.
2. It uses Jinja2 templates - jinja2 is a popular template engine for Python.A web template system combines a template with a specific data source to render a dynamic web page.
3. It is compliant with WSGI 1.0. - The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.
4. It provides integrated support for unit testing.
5. Many extensions are available for Flask, which can be used to enhance its functionalities.

Comparative Analysis of Flask:

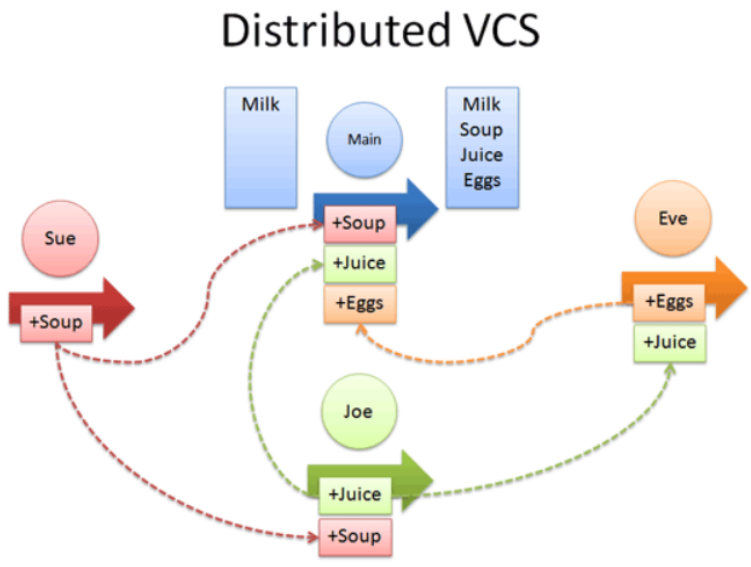
Many Python-based web frameworks enable developers to build scalable applications quickly. Out of the many popular choices, Django and Flask are the most talked about. Django, on the one hand, is a full-stack web framework, whereas Flask is a light-weight, extensible framework. Flask helps you understand how each component from the back-end works to get a simple web application up and running. Django follows lots of design patterns, and hence you learn a lot of exciting concepts. Some other differences between Flask and Django are mentioned in the following table:

Django	Flask
Full-stack web framework that follows the batteries-included approach.	Light-weight framework with minimalistic features.
Developers already have access to the most common features that makes development faster.	Developers can explore and keep control of the core of the application.
Django comes with a ready-to-use admin framework that can be customized.	Flask doesn't have any such feature to handle administration tasks.
It comes with a built-in template engine that saves a lot of development time.	Flask's template engine Jinja2 is based on Django's template engine.
It allows users to divide a single project into multiple small applications which makes them easy to develop and maintain.	Each project can be a single application, however, multiple models and views can be added to the single application.
The Django-admin tool is a built-in bootstrapping tool with which developers can build web applications without any external input.	Admin features are not as prominent as in Django.

References

Flask documentation
https://flask.palletsprojects.com/en/2.0.x/
https://pythonbasics.org/what-is-flask-python/
https://hackr.io/blog/flask-vs-django

Before getting into what Git is, it is important to know about the **version control system**.



A version control system maintains a record of changes to code and other content. It also allows us to revert changes to a previous point in time.

Version control facilitates two important aspects of many scientific workflows:

1. The ability to save and review or revert to previous versions.
2. The ability to collaborate on a single project.

This means that you don't have to worry about a collaborator (or your future self) overwriting something important. It also allows two people working on the same document to efficiently combine ideas and changes.

Git is the most commonly used version control system. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

Git is software that runs locally. Your files and their history are stored on your computer. You can also use online hosts (such as GitHub or Bitbucket) to store a copy of the files and their revision history. Having a centrally located place where you can upload your changes and download changes from others, enables you to collaborate more easily with other developers. Git can automatically merge the changes, so two people can even work on

different parts of the same file and later merge those changes without losing each other's work.

Git Repositories

A Git repository (or repo for short) contains all of the project files and the entire revision history.

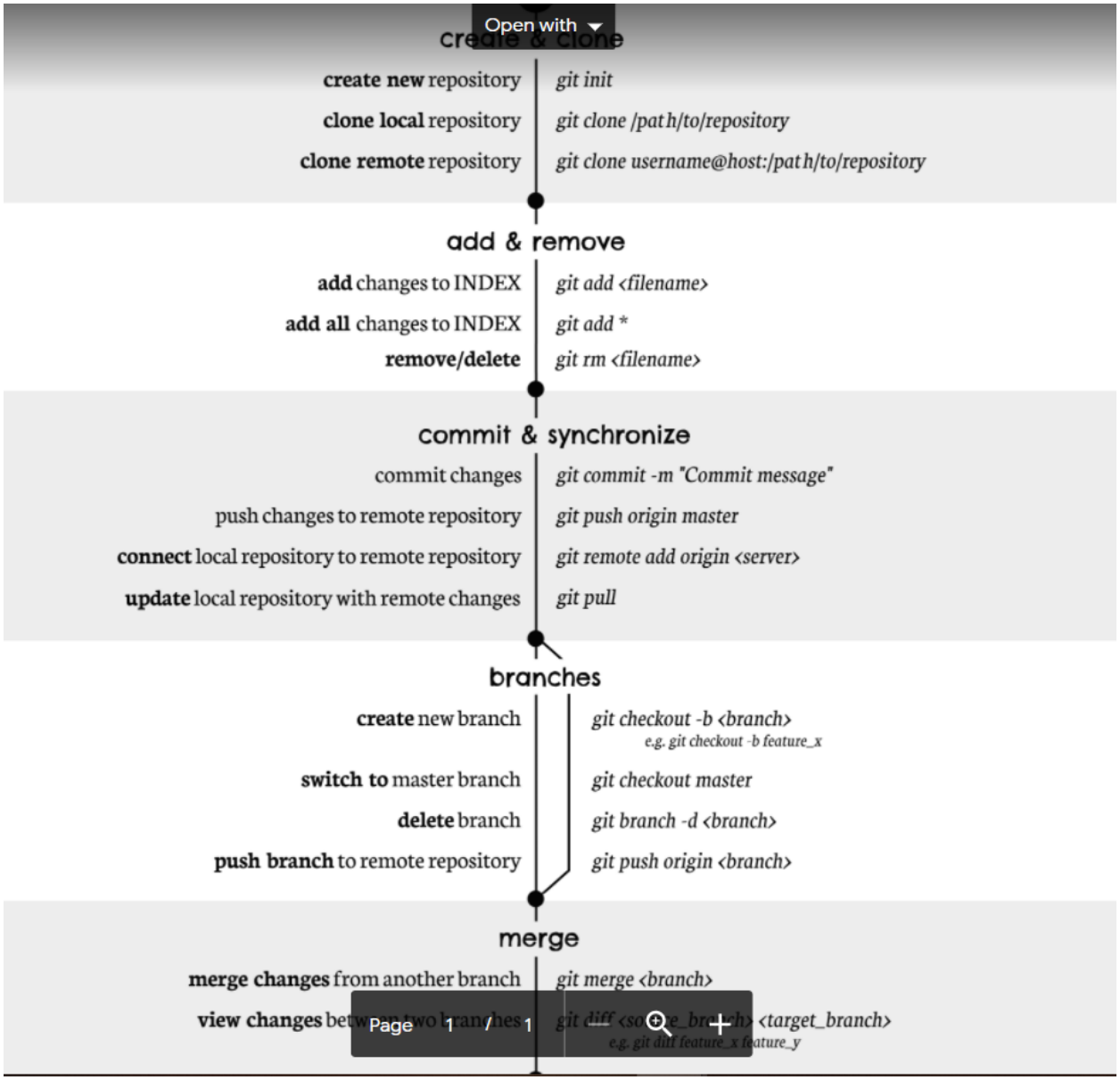
Stage & Commit Files

Before we make a commit, we must tell Git what files we want to commit. This is called staging and uses the add command. We add files to a staging area, and then we commit the files that have been staged.

Branches & Merging

Git lets you branch out from the original code base. This lets you more easily work with other developers, and gives you a lot of flexibility in your workflow. Let's say you need to work on a new feature for a website. You create a new branch and start working. You haven't finished your new feature, but you get a request to make a rush change that needs to go live on the site today. You switch back to the master branch, make the change, and push it live. Then you can switch back to your new feature branch and finish your work. When you're done, you merge the new feature branch into the master branch and both the new feature and rush change are kept.
When you merge two branches (or merge a local and remote branch) you can sometimes get a conflict. For example, you and another developer unknowingly both work on the same part of a file. The other developer pushes their changes to the remote repo. When you then pull them to your local repo you'll get a merge conflict. Git has a way to handle conflicts, so you can see both sets of changes and decide which you want to keep.

Git commands



References:

For installation on windows and setting up SSH

https://dev.to/bdbch/setting-up-ssh-and-git-on-windows-10-2khk

Github tutorial and reading VCS

https://www.neonscience.org/resources/learning-hub/tutorials/version-control-github

Pytorch 

PyTorch is defined as an open source machine learning library for Python. It is primarily used for deep learning applications using GPUs and CPUs. It was initially developed by Facebook artificial-intelligence research group, and Uber's Pyro software for probabilistic programming which is built on it.
The most basic building block of any Deep Learning library is the tensor. Tensors are matrix-like data structures very similar in function and properties to Numpy arrays.

In Pytorch, tensors can be declared using simpleTensor object as follows:

```
import torch
x = torch.Tensor(3, 3)
```

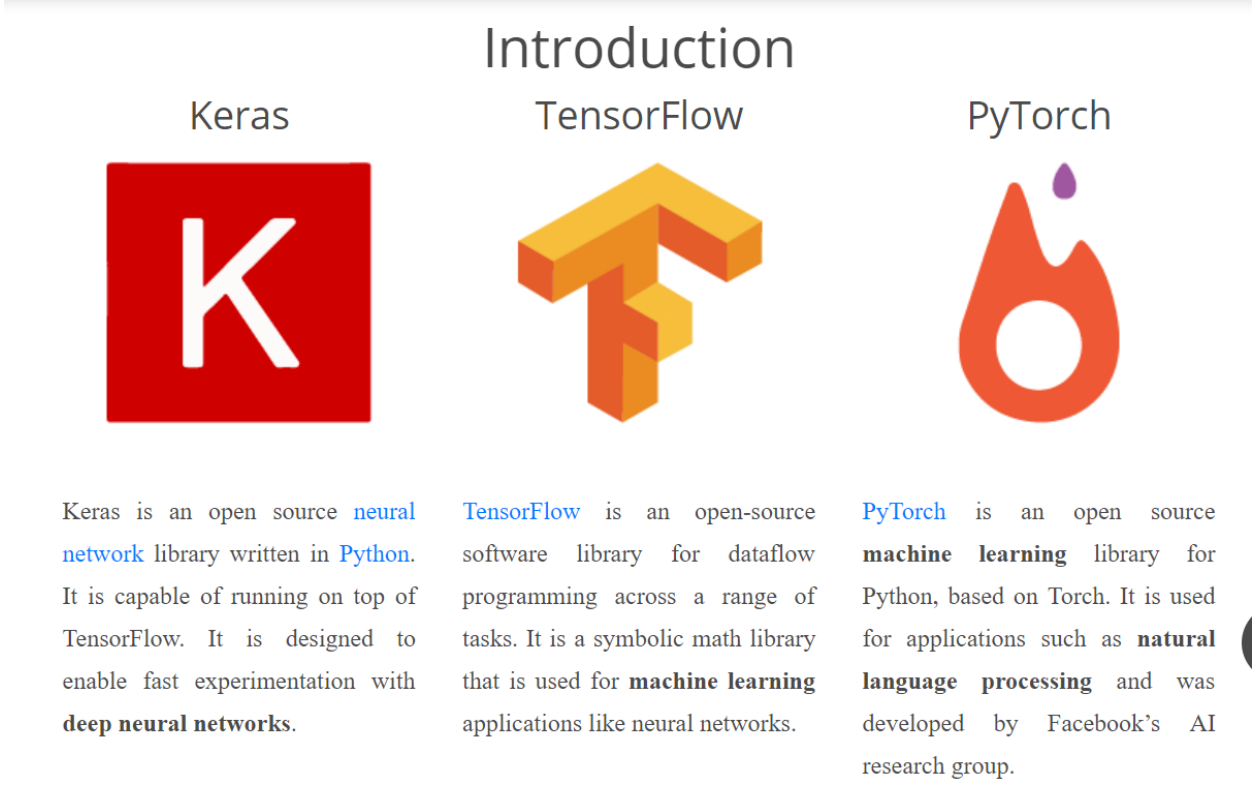
The above code creates a tensor of size (3,3), i.e. 3 rows and 3 columns, filled with floating point zeros.
Basic mathematical operations like addition and multiplication is also easy in PyTorch

```
x = torch.ones(3,3)
y = torch.ones(3,3) * 4
z = x + y
print(z)

"""
Prints out:

tensor([[5., 5., 5.],
        [5., 5., 5.],
        [5., 5., 5.]])
"""
```

With Pytorch, neural networks are defined as Python classes. The class which defines the network extends the *torch.nn.Module* from the Torch library.

Other machine/deep learning libraries:
Following are some libraries developed in python primarily for machine/deep learning applications.

Comparative Analysis of PyTorch:



PyTorch vs TensorFlow

Visualization

In terms of visualization, TensorFlow wins the battle, as it offers better visualization tools, which allows developers to debug better and track the training process. PyTorch, however, provides only limited visualization. TensorFlow also beats PyTorch in deploying trained

models to production, because of the TensorFlow Serving framework. PyTorch offers no such framework, so developers need to use Django or Flask as a back-end server.

Data Parallelism

In the area of data parallelism, PyTorch gains optimal performance by relying on native support for asynchronous execution through Python. However, with TensorFlow, you must manually code and optimize every operation run on a specific device to allow distributed training.

PyTorch vs Keras

Keras is better suited for developers who want a plug-and-play framework that lets them build, train, and evaluate their models quickly. Keras also offers more deployment options and easier model export.

However, remember that PyTorch is faster than Keras and has better debugging capabilities.

Installation:

pip3 install torch torchvision

References:

https://pytorch.org/docs/stable/index.html
https://machinelearningmastery.com/pytorch-tutorial-develop-deep-learning-models/

Express.js is a free and open-source web application framework for Node.js. It is used for designing and building web applications quickly and easily. Since Express.js only requires javascript, it becomes for programmers and developers to build web applications and API without any effort. The JavaScript library of Express.js helps the programmers to build efficient and fast web apps. Express.js enhances the functionality of the node.js.

Why Express.js?

Express.js is based on the Node.js middleware module called connect which in turn uses http module. So, any middleware which is based on connect will also work with Express.js. It is easy to configure and customize and allows you to define routes of your application based on HTTP methods and URLs. Moreover, Express.js is easy to integrate with different template engines like Jade, Vash, EJS etc., and can be easily connected with databases such as MongoDB, Redis, MySQL.

Comparative Analysis of Express.js

Performance

Most developers adore Node.js for its raw speed, and when it comes down to framework selection. Express provides a thin layer on top of Node.js with web application features such as basic routing, middleware, template engine and static files serving, so the drastic I/O performance of Node.js doesn't get compromised.

Express is a minimal, un-opinionated framework. it doesn't apply any of the prevalent design patterns such as MVC, MVP, MVVM or whatever is trending out of the box.

Middleware

Middleware are basically just functions that have full access to both request and response objects. As the name implies, middleware applies some filtering instruction before handing the control over to actual business logic or the next level of middleware. Some common tasks include checking for user login status, validating user authority, or preventing cross-site attacks that are best extracted as middleware.

Database Integration

As a minimal framework, Express does not consider database integration as a required aspect within its package, thus it leans toward no specific database usage whatsoever. While adopting a particular data storage technology, be it MySQL, MongoDB,

PostgreSQL, Redis, ElasticSearch or something else, it's just a matter of installing the particular npm package as a database driver.

References:
Documentation and installation
https://expressjs.com/en/starter/installing.html
https://www.toptal.com/nodejs/nodejs-frameworks-comparison