Report

Project Intro:

Our website is called Olibaka, which is a checkout price calculator with some basic functionalities such as adding/removing items, applying discount codes, calculating tax and amount total.

For assignment 1, we use the following techniques: For the frontend, we choose **HTML, CSS, JavaScript, jQuery, Bootstrap**; for the backend, we choose **Python, Flask, Requests, Ajax**; we choose **CircleCI** for building CI; and we also use **Heroku** to achieve our deployment.

In this report, we are going to discuss our considerations when choosing programming techniques for this assignment.

Frontend technologies:

For frontend, we choose **HTML and CSS** as our main techniques which we are most familiar with and have more confidence in, which makes our project easy to be developed and modified(debugged) during the process.

We also select **JavaScript with jQuery library** because it can fix browser problems such as CSS layout issues. JavaScript is a mature technology to use on the web because of its most mature ecosystems among all the programming languages. The language is easy to get started with and it is more flexible and faster in developing applications compared to other programming languages I tried such as Java.

The main reason we choose JavaScript is because it blends so well with HTML and CSS and can also work with Flask, which is another technology we use in the backend. Moreover, JavaScript provides highly responsive interfaces and is not restricted to browsers, which gives our customers great experience.

Note that JavaScript is used across the web development stack, its domain covers both frontend and backend(Node JS, which is one type of JavaScript), we will discuss the reason why we did not use Node JS as our backend technology in the backend technology section.

Bootstrap is a flexible framework that is preferred and recommended by the majority of web designers and programmers. The main reason why we choose it is because it covers basic HTML and CSS templates that includes common UI components, and also JavaScript extensions and jQuery plugins for creating and building web applications. The templates are made to be available, so we can use multiple CSS design templates that can be applied to HTML to achieve different effect goals.

Since Bootstrap includes pre-made code/templates and plugins, we can develop our web application by just simply copy-pasting some lines of code, which results in a faster

development speed compared to other techniques. The framework is guaranteed to work nicely in all major browsers, which saves more time to perform browser testing.

In conclusion, the frontend techniques we choose are easy to develop for us, and all of them are mature and are at the top of the most popular programming languages and frameworks. Most parts of these technologies are pre-written code, which increases development speed and has better performance than the templates coded by us.

Backend technologies:

For backend, according to our programming experience, **Python** would absolutely become the best choice to build the backend environment not only because of its popularity among programmers, but also due to its powerful libraries with pre-written code, which speeds up the development time. Python is an open-source language which was first released in 1991, it has become a mature programming language with 30 years of improvement.

We did consider using **Node JS**, which is also a popular tool for backend development, however, we choose to use Python instead of Node JS for the following considerations:

1. Domain: Python is more suitable for backend applications, numerical computations and also machine learning, whereas Node JS is one type of JavaScript, therefore it is better for web deployment.
2. Syntax/Structure and develop speed: Python has a large library with huge amount of pre-written code, and the syntax is short and clean, we only need to write fewer lines of code, which speeds up the development progress, whereas Node JS shares the same structure/syntax as JavaScript, so there are some functionalities that need us to develop.
3. Maturity: Python was first published in 1991, which has an advantage on library storage than Node JS, which was first published in 2005.
4. Develop Confidence: Since both of us are more familiar with the structure and syntax of Python, we have more confidence on programming using Python instead of Node JS.
5. Project function complexity: Since our functionalities can be implemented with plain JavaScript with decent code clarity, we did not use Node JS in this assignment.

As we mentioned before, Python has countless resources in many forms, including a wide variety of web application frameworks such as **Django and Flask**, which are what we struggled in choosing from. After comparison, we choose **Flask** finally to develop API for the following reasons:

1. Framework style: Flask is a light-weight framework with minimalist style, while Django is a full-stack web framework.

2. Flexibility: Flask is more flexible on expanding applications in a faster way, simple applications can be added more functionality to make it more complex. We are free to use any plugins and libraries. However Django cannot achieve this goal, it actually requires unchanging projects.
3. Overdose: Simple applications can be built with ease in Flask, but Django is more suitable with bigger and more complex projects. Since our project is not that complex, using Django is kind of an overdose, so we finally decided to use Flask instead.

To communicate between the Client and Server, i.e, to connect HTTP requests with JQuery AJAX, we use Python Flask and **Ajax** to get JSON responses from Ajax requests, which can be easily tested with Python **Requests**, which leads to our discussion on the reason why we choose Ajax:

1. Callback: As we mentioned in the frontend technology part, we use JavaScript with jQuery library to build our frontend web applications, the jQuery library is used to make client-side development easier in order to make asynchronous Ajax callbacks. Ajax callback is one of its advantages on web applications, it can make a fast round trip to the server without having to post the entire page back to it.
2. Asynchronous call: Ajax can make asynchronous calls to the web server, which reduces the waiting time.
3. Speed and Performance: The goal of Ajax is to improve the speed of development and performance, which becomes the main reason we choose it.

Python **Requests** is a succinct HTTP library for Python, its main purpose is to send requests to HTTP easily and handle multi-part file uploads. Python Requests also contain a default JSON decoder, which can respond to Ajax. Compared to urllib, Requests do not produce confusion, since there is only one module that does the entire work.

Database:

We decide to use **SQLite** to build our database for this assignment. SQLite, together with MySQL and PostgreSQL is one of the most popular open-source relational database management systems in the global range. In this report, we are going to discuss the advantage of using SQLite with considerations:

1. Ease of use: We do not need to follow a lot of complex installation steps, we can directly use SQLite by simply downloading the SQLite libraries.
2. Lightweight: SQLite has a small footprint so that we can embed SQLite easily in an application. Also there aren't any additional dependencies we need to install on our system for SQLite to work.

3. Performance: SQLite provides faster read and write operations so that edition is also efficient, which makes it a strong performance tool even in a low-memory environment.
4. Flexibility: SQLite supports all modern popular languages such as Python, Java, etc.
5. User-friendly: Since SQLite does not run as a server process, it doesn't need to be stopped or restarted, therefore we do not need to worry about coming out with some configurations.
6. Compatibility: SQLite supports multiple platforms like Windows, Linux, MaxOS, even Android, iOS.

CI/CN:

We used CircleCI for our CI/CD automation to ship the web application quickly and efficiently. For this choice, we have considered multiple other CI/CD tools that can functionally continuously integrate code into production, such as GitLab CI, GitHub Action, Jenkins as well as Azure. However, we still decided to use CircleCI for automation with the following consideration:

1. Jobs Configuration: CircleCI is comparatively easier to build with clear instructions and automatically generate yml config files for selected branches. The yml config file is also easy to read. The way they generate yml config is simple and useful because all users need to do is add a yml file into the git repository and add some testing commands inside.
2. Ease to develop: While other tools such as Jenkins, Azure would require authorization from the organization to start the automation with the selected repo, CircleCI is way more convenient as it can link to a github account with any of the repositories. It also automatically runs CI/CD tests when there are any pushes to the specific branches.
3. Speed: According to the official website, CircleCI is 30% more effective than GitLab CI and GitHub Action, and faster than Azure about 2.5 times for one automation test on average, so choosing CircleCI provided more efficient production.
4. User-friendly : CircleCI is compatible with more languages than other CI/CD tools, including python, node, ruby, java, go, etc. This provides more options for users to use the languages for their software development. It also allows customization and scripting abilities by using tools such as bash, ssh and node. Therefore, committing anything to github is runnable with CircleCI.

Note: we have experienced running out of credit issues on CircleCI after using too many credits on the previous runs as it automatically after each merge, and some of the most recent commits show it fails the CI/CD test, although the same code passed in the previous

commits. We have forked the same code in our own repository and the CI/CD run successfully.

Deployment:

Heroku is a cloud service provider that embraces many modern programming languages such as Python, Node JS. It makes the process of developing as simple as possible, so that we can spend more time developing our project. Since Heroku is widely used by many companies such as Toyota, it is considered as one of the popular deployment tool, therefore we compared Heroku with another famous deploy tool **AWS**, which is also a good platform of computing solutions and choose Heroku as our deployment tool after the following considerations:

1. Ease to develop: The process of deployment in Heroku is easy to follow, since we are using Flask, there is typically a Flask tutorial in Heroku with only simple steps, we would say Heroku is more beginner-friendly than AWS. In addition, the creation of a server using Heroku is simpler than that using AWS.
2. Speed: Heroku provides a pre-written environment so that we can deploy our code by entering code on the instructions, which is faster on deploying our code; whereas the deployment process of AWS is complicated. Since both of us are new to deployment techniques, we would choose Heroku.
3. Roll back feature: Heroku allows us to roll back our database at any stage while AWS only allows us to roll back to the previous version.
4. Computational demands: The computational demands for Herok is lower than that of AWS, for it does not require users to be a DevOps Engineer, whereas AWS asks for a DevOps Engineer.

Based on all the considerations above, we choose to use Heroku as our deployment tool to save our developing time.