CS 3240 - DevOps Manager Report **TA: ____Houck___/ Team: ________A-22_________**

DevOps Manager Report

Summary

The first tool we used in our project was the Django framework. At the beginning of our team's project, all team members individually worked on gaining fundamental knowledge about Django framework and its MVC architecture by stepping through well-documented tutorials on an official Django documentation site. After we completed the tutorial and completed building an example Django application, we were able to communicate more efficiently and professionally in setting our project approach and goal, because every team member used the same, official learning resource and therefore utilized the consistent terms when explaining one's technical approach to solve problems.

Our project aims to build a full-stack web application that provides users with an assignment and note organization tool. The Django framework perfectly suits our needs because it comes with MVC architecture in which client side and server side can seamlessly interact with each other, and database management is abstracted on a high level with using "models" instead of manually managing data tables. Despite its benefits, our team encountered minor issues while implementing the first feature: adding a new assignment and deleting an existing assignment from the list. This was mainly due to our lack of understanding in the framework and could be gradually resolved as we gain more experience using the framework and reading more online resources.

The second tool we incorporated in achieving our goal was GitHub. After setting up the basic Django project, our team created a GitHub repository and cloned the repository into our local machines. The tool definitely facilitated multiple people collaborating on the same code base, sharing updated files and tracking a record of past commits. Our team also took advantage of descriptive commit messages to help others in the team easily understand the changes made to the project. I, as a DevOps manager of the team, personally put most effort into building up a good, well informed README documentation that includes our project roadmap, a local set up instructions, and contribution guide. This allowed our team to avoid confusion when reiterating a consistent local development set up. We encountered some issues in collaborating on GitHub because not every member in the team had a good understanding of the tool and its commands. There were times when one team member accidentally overwrote another's work by pushing his changes to the wrong branch. We resolved these issues by sharing additional online resources to learn more about git commands and communicating more frequently on Discord. Our team now has a more structured way of contributing by writing a more detailed contribution guide on the README file.

After we finished implementing the first feature, our next biggest challenge was deploying our locally developed Django project. The main issue arose from setting up the project's database for deployment. In our local development setting, the site interacts with our local PostgreSQL database, but when the site gets deployed onto a Heroku server, a different PostgreSQL database was required to be set up in the Heroku server. No one in the team had past experience in using PostgreSQL on Heroku, so we spent a number of hours trying to figure out the right way to set up the database. We soon found out there are a set of packages needed and changes to our project configuration file required to deploy our site without any issue. Once we finished setting up our initial heroku deployment, the subsequent deployments could be easily configured.

The most recent tool we incorporated into our project is a continuous integration tool, GitHub Actions. Initially, setting up this new tool didn't seem as hard as deploying the site because the resource provided to us was straightforward, and we haven't encountered any unexpected problems following the instructions in the resource. However, we soon realized GitHub action also requires a separate remote PostgreSQL database to properly run our unit tests. So, we spent some time searching online for ways to set up the database exclusively used for testing purposes. After browsing through countless online resources, I returned to the course site and

found a list of good resources under the additional resources section that exactly matched the issues we are facing. Following one of the resources explained, in detail, what each line in a workflow file meant and the proper way to set up a docker image with PostgreSQL database built into it.
Overall, learning a new tool and finding ways to incorporate it into our workflow has always been a challenge with no prior experience. However, it helped us better understand the benefits of using these tools and provided us with a good skill set we can utilize in our future projects.

Individual Platform/Tool Discussions

Django

The official Django documentation worked great for our team. The site's tutorial was helpful in understanding the basics of the framework and it thoroughly explained how each component of a project should be implemented with simple, yet intuitive examples. Everyone on our team could use the official terms from the documentation to explain their thought process and clearly communicate what parts of the project they want to take part in. Also, the framework is based on the most intuitive programming language, Python, which everyone on the team had prior experience with. This allowed us to focus more on logics and structures than a minor syntactical issue.
On the other hand, our team struggled to understand its MVC architecture at the beginning. Few members in the team posted numerous questions on our team Discord channel about the roles of each component and the way they play together behind the scenes. Those who could quickly understand the concept explained how each component in the architecture works with one another. I believe there are still some who do not fully understand the overall structure of the framework, so I am planning to share additional online resources that give a high-level description of the framework and answer some of their questions during our separate weekly meetings.
If I were to offer suggestions to teams next semester, I would tell them to thoroughly read what architecture this framework is based on and understand how different components (Model, View, Template) play roles in constructing the site. Watching a well-known video resource can also help visualize the architecture. This is an essential, high-level overview of this tool before diving into the actual tutorial, and having at least a partial understanding of the knowledge will help them build up a strong, fundamental knowledge as they walk through the tutorial.

GitHub

As GitHub is already well known for its versatility in project management and collaboration, our team benefited the most from this tool when collaborating in one single project. Each team member could simultaneously work on a project, and we could later merge these separate works into one single project without any issue. Sharing the README file on GitHub also allowed us to set up a consistent local development environment and pursue more structured collaboration methods. In addition to its core functionality, GitHub has a record of our past commits/changes in each file, and this feature has saved us considerable time reverting our project to the previous version even after making erroneous changes. Also, others in the team could review one's commits to ensure its code quality before merging with the production-ready main branch.
Despite its benefits, it took us some time to feel comfortable using git commands and learning how to resolve merge conflicts. At one time one team member accidentally pushed an erroneous commit directly to the main branch, and other members who pulled from the main branch without any doubt ended up wasting time resolving unexpected changes to the files. This could be avoided if every team member was well aware of the effects their commits can bring to the entire team and grows a habit of carefully reviewing the commits before

actually merging the changes into their local project files. As we move forward in our project, we plan to encourage creating a separate branch and submitting a pull request with at least two reviewers registered. This will help us tremendously in the long term, saving us time from fixing the code and focusing more on introducing more features.
I want to suggest to teams next semester to set up one in-person meeting solely dedicated to walk through every git command the teams will use and a well defined series of steps to contribute to the project. This will ensure everyone on the team will take a consistent approach when contributing, which will make debugging issues easier.

Heroku

Heroku was a great cloud service tool for our team in deploying our project and quickly reviewing its production version. Although initially setting up our project for deployment was a challenge, the service has numerous autonomous features built in which allowed us to save time in the long term. For example, we could link our GitHub repository to the Heroku service, so when we push commits/changes to the main branch, Heroku automatically detects that the project is using Django framework, install Python and PostgreSQL database, install all required packages from ‘requirements.txt’, run build, and deploy on one of its cloud servers. Also, using the free cloud service saves us money hosting on our own physical server and the service can even automatically adjust the server’s capacity to meet the site’s needs. After we finished setting up the initial deployment, we could continuously deploy one of our branches with just a single click to check any issue in production. The tool will help us largely in maintaining the service, too.
At first, our team ran into issues with setting up a remote database on Heroku server because we were so used to our local development environment and did not expect the remote database will be required when deploying the site. The issue could be quickly resolved because the tool has a great community support where similar issues were addressed a number of times. If we could change the approach using this tool, we would also look into its pipelining feature which can create a pipeline for CI/CD. This can allow our team to thoroughly test our branches before merging into the main branch and deploying a credible production version of the project.
I would suggest the teams next semester take some time individually to try out deploying a simple app using Heroku and its command line interface. This will help them better understand the terms they will encounter in various online resources and take advantage of the tool’s full potential.

Continuous Integration

Implementing a continuous integration using GitHub Action was an amazing experience for our team. Like GitHub, GitHub Action also has an extensive and insightful official documentation and we could get most of our questions answered by reviewing the documentation. In particular, the way we can set up a workflow in a .yml file is so intuitive that it is, even with no prior experience, easy to understand what each line in the file represents and how it will change the workflow. Also, because the service is provided by GitHub, it was a seamless experience incorporating GitHub Actions into our current workflow with GitHub. There was no required change in the way we previously pushed our commits, but all the builds and tests can run in the background, even informing us if any of the tests fails and therefore should not be merged. Our overall experience with the tool was simple, seamless, and powerful.
We ran into one issue similar to the one in Heroku where a separate remote database is required to run the tests. Thankfully, the course site already provided us with a list of resources we can refer to, so we could manage to create additional steps in our workflow to set up a PostgreSQL database in a docker image and change the project configuration file to use the remote database instead of the default database. If we could change the

approach using the tool, we would review the list of resources already posted on the course site before wasting unnecessary amounts of time searching unverified resources online.
I would suggest the teams next semester to fully understand the basics of GitHub Actions workflow and what each line in the .yml file represents in the workflow. This approach will allow them to have a good control of the workflow and customizing it to perfectly satisfy their needs.

Instructor-Specific Comments

Overall, the tools we are using in the course are highly credible and full of benefits. I really enjoyed seeing how different developer tools can benefit my team's workflow in achieving the project goal. However, one thing I would like to suggest to the staff is that Heroku official documentation is somewhat hard to follow because some terms used in the documentation are specific to the service. If the staff team can find an additional resource explaining the tool with more general terms or on a higher level, students would be able to walk through two resources and gain a better understanding of the concept, fully benefiting from its use.