Jordan Han - hanseu24 - 1004726175
Jacob Sahlmueller - sahlmuel - 1005050374

CSC301 A1 - Report

Summary:

For our backend framework, we selected Django. The primary reason for this selection was that both team members are in the same partner project group for CSC301, and Django is the preferred backend of said partner. Therefore, learning the basics of creating a Django web application backend would benefit the team members both in terms of gaining general technical knowledge (neither member has worked with Django before) as well as knowledge that is specifically applicable to the semester-long partner project. Additionally, both of the team members have previous experience with Python while only one has experience with Javascript and neither have experience with Go. We found out that Django also offers generally better performance when compared to other backend frameworks, as well as better security when compared to Node. Lastly, Django is reportedly ideal for quick development times, which suits this project perfectly.

For our frontend framework, we selected React. Due to the limited size and scope of this project, any frontend framework would overcomplicate development. Thus, our decision to use React was based solely on the educational value of the experience, rather than any specific utility it would provide for this project. Through our research, we discovered that React has the most active job market of any frontend framework (specifically compared to Vue and Angular) and therefore would be the most beneficial to know, career-wise. Furthermore, React's use of Javascript rather than Typescript meant a simpler learning process, as one of the team members has previous experience with vanilla JS. Finally, we found out that React is ideal for the use case of single-page applications, which fits the specifications of this project.

For our CI/CD tool, we selected GitHub Actions. This decision was almost entirely based on the pre-existing necessity of using GitHub for this project and Actions' ease-of-integration with its eponymous version control software. Other CI/CD solutions we looked into were promising, but they would require separate installations (in the case of Jenkins) and/or working out how to integrate them with GitHub (in the case of GitLab CI). We did find that GHA was a less popular choice than some of its alternatives—perhaps due to its relative infancy—but this factor was relatively inconsequential to our decision.

For our database, we selected SQLite. Our main reason for this decision was that SQLite, unlike the other database options, does not require us to set up a specific database server. Instead, it integrates directly with the web application. Furthermore, SQLite comes bundled with Python, and Django includes a SQLite database as part of its default configuration, so considering the other parts of our tech stack, SQLite makes the most practical sense. SQLite does provide a more limited range of functionality compared to PostgreSQL or MySQL (for instance, not supporting user management), but the small size of the project means that we likely would not have cause to use any of those more advanced features regardless. Therefore, the simplicity and ease-of-use of SQLite shine through.

Backend

Django (selected)

Benefits:

1. This is the backend framework preferred by our project partner for the CSC301 group project. By utilizing it here, we can get accustomed to using the framework before we need to work on it for the project. Django is not a necessity for the project, but many of the existing parts of our partner's codebase use Django and so the group elected to use it for our project's backend. Learning the basics while building this simple application will allow for a swifter onboarding process during the partner project.
2. All team members have some previous experience with the Python language.
3. Django offers better performance than other backend frameworks.[1]
4. Django is designed to "warrant quick development,"[2] which is ideal for this project.

Drawbacks:

1. Utilizing a python backend framework means a greater number of technologies will be needed to complete this project, as all of the frontend frameworks require Javascript or Typescript.
2. Django may introduce unnecessary complexity to this small-scale project as "it is not suitable for small apps and should be avoided."[3]

Node.js

Benefits:

1. One of the team members has previous experience developing web applications that use Node, both for an internship as well as for personal projects.
2. Using Node would provide increased skill and familiarity with Javascript, which would be beneficial both for front- and back-end development.
3. Using Node along with a JS front-end framework such as React would mean needing to learn fewer technologies for this project and would allow for the tech stack of the project to be more unified.

[1] Mistry
[2] Ibid
[3] Ibid

Drawbacks:

1. Javascript's weak typing and the general design of Node make it somewhat more vulnerable to security risks and exploits.[4]

Go

Benefits:

1. Go offers increased security when compared to Node,[5] but Django does this as well.[6]

Drawbacks:

1. No team members have previous experience programming with Go, and in particular no experience writing a web application with a statically-typed language.
2. Go is less widely used than either Django or Node and therefore may have less available libraries and APIs.[7] Though this is less of a concern for this particular small-scale project, it must be considered along with the benefits and drawbacks of learning the language and framework in general.

Frontend

React.js (selected)

Benefits:

1. React is more easily onboardable in a short period of time, with a lot of more advanced functionality coming as a result of third-party libraries. This would allow for a more gradual learning curve.[8]
2. React has the most active job market of any frontend framework and thus would be highly beneficial to know.[9]
3. React is the most widely used web framework on the market today.[10]
4. React is an easier option to choose as the team already has some experience with Javascript.
5. React is ideal for single-page applications, which this will be.[11]

Drawbacks:

1. Using React may be overcomplicating the frontend for this very minimal project, which does not require a great deal of functionality. However, this drawback is present for all

[4] Devathon
[5] Ibid
[6] Mistry
[7] "2021 Developer Survey"
[8] Daityari
[9] Ibid
[10] "2021 Developer Survey"
[11] Souza

frontend frameworks under consideration (and indeed all frontend frameworks beyond vanilla Javascript), so it does not count against React specifically.

Angular

Benefits:

1. As a complete solution, taking the time to learn Angular would lead to somewhat better preparedness to create frontend solutions in the future.
2. Using Angular would necessitate learning Typescript, which is a technology of interest for a team member. However, learning even more new technologies for the frontend framework is also a drawback, as discussed below.
3. Angular is quite popular as the fourth most widely used web framework, though it is still not nearly as prevalent as React.[12]

Drawbacks:

1. Angular would require more time and effort on learning new components (such as TypeScript and MVC).[13]

Vue

Benefits:

1. Vue is entirely open-source and the team values supporting open-source projects.[14]
2. Vue is more customizable than React or Angular, which is less of a consideration for this project, but it also is highly integratable with popular responsive CSS frameworks like Bootstrap.[15]

Drawbacks:

1. From a career perspective, many fewer jobs are seeking experience in Vue compared to React and Angular.[16]
2. Vue is also somewhat less popular than the other two frameworks, with fewer contributors and exclusive reliance on open-source contributions rather than the dedicated development of a company.[17]

CI/CD

GitHub Actions (selected)

Benefits:

[12] "2021 Developer Survey"
[13] Daityari
[14] Ibid
[15] Ibid
[16] Ibid
[17] Ibid

1. Our project repository is already set up via GitHub Teams, so using a GitHub-integrated tool makes the most immediate sense. Additionally, the team is most familiar with GitHub.
2. *Actions* has a "wide varierty [sic] of available pre-made workflows."[18]
3. Supports asynchronous CI/CD.

GitLab CI

Benefits:

1. GitLab CI appears to be more widely used than the other options and thus, presumably, would have a larger amount of technical and troubleshooting information available.

Drawbacks:

1. We would need to look into and figure out GitHub integration for GitLab CI, which would be a lot more work, particularly when compared to GitHub Actions.

Jenkins

Benefits:

1. Jenkins is an open-source project with "a devoted community worldwide."[19]

Drawbacks:

1. Learning Jenkins seems like it would require a much larger amount of onboarding time compared to the other CI options.[20]
2. Jenkins requires separate server installation.

Heroku Pipelines

Benefits:

1. Heroku Pipelines is directly integrated with our hosting server (as we decided to use Heroku to host our web page).
2. Heroku Pipelines is set up to support different environments, such as a development environment and a production environment.

Drawbacks:

1. Heroku pipelines works on git repositories in Heroku, and as we are using Github as our main repository for our project, we would not have been able to benefit from this.

Database

PostgreSQL

[18] "Github Actions vs Gitlab CI"
[19] Almeida
[20] Ibid

Benefits:

1. One of the team members has previous experience using PostgreSQL in connection with a web application.
2. PostgreSQL offers a wide range of functionality and getting some experience with it, even just with its basics, would be useful and potentially helpful for the careers of the team members.[21]
3. PostgreSQL is very widely used among professional developers, second only to MySQL.[22]

Drawbacks:

1. The utility provided by PostgreSQL databases, though extensive, is in many respects overkill for the minimal functionality needed for this application.
2. One of the team's experience with this technology, though helpful, also means that they would be learning less during this project, as we would most likely just be implementing basics.

MySQL

Benefits:

1. MySQL includes user management features — though these are unnecessary for this project.[23]
2. MySQL provides more security features than SQLite.[24]
3. MySQL is the most widely used database tool among professional developers.[25]

Drawbacks:

1. As with PostgreSQL, many of the features supported by MySQL are not necessary and would just introduce unneeded complexity to the project.

SQLite (selected)

Benefits:

1. As the name would imply, SQLite is the most lightweight solution we considered for databases. Unlike the other options, SQLite does not require a specific database server and integrates directly with the web application.[26] This highly simplifies database setup and configuration.
2. SQLite comes bundled with Python, and Django includes a SQLite database as part of its default configuration. This means less work for us.

[21] Drake & ostezer
[22] "2021 Developer Survey"
[23] Ibid
[24] "SQLite vs MySQL - Comparing 2 Popular Databases"
[25] "2021 Developer Survey"
[26] "SQLite vs MySQL - Comparing 2 Popular Databases"

3. One of the team members has previous experience working with SQLite integrated into an application.

Drawbacks:

1. SQLite does not support user management, which would be interesting to include for our application but is ultimately not necessary.[27]
2. The team member with previous experience with this tool would be learning less during this project.
3. Of the three technologies under consideration, SQLite is the least widely used (though it is still the third most used database).[28]

[27] Drake & ostezer

[28] “2021 Developer Survey”

Citations

"2021 Developer Survey," *Stack Overflow.* https://insights.stackoverflow.com/survey/2021

Almeida, Bruno Amaro. March 2021. "The CI/CD War of 2021: A Look at the Most Popular CI/CD Tools," https://blog.thundra.io/the-ci/cd-war-of-2021-a-look-at-the-most-popular-technologies

Daityari, Shaumik. 15 March 2021. "Angular vs React vs Vue: Which Framework to Choose in 2021," https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

Devathon. 28 August 2020. "NodeJs vs Golang in 2020," https://medium.com/@devathon_/nodejs-vs-golang-in-2020-179db4629d51

Drake, Mark & ostezer. 19 March 2019. "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems," https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

"Github Actions vs Gitlab CI," *Knapsack Pro.* https://knapsackpro.com/ci_comparisons/github-actions/vs/gitlab-ci

Mistry, Jigar. "Django vs Node.js: A Comparative Analysis," https://www.monocubed.com/django-vs-node-js/

Souza, Diogo. 2 January 2021. "React and Django: Your guide to creating an app," https://blog.logrocket.com/creating-an-app-with-react-and-django/

"SQLite vs MySQL - Comparing 2 Popular Databases," 4 October 2018. https://www.keycdn.com/support/sqlite-vs-mysql