Summary of analysis

Our assignment is to build an ecommerce web app with basic functionalities such as adding items to cart and proceeding to checkout where a calculation of the total cost is displayed. To sufficiently cover the infrastructure to develop such an app we considered at least three options for the frontend, backend, CI/CD and database. For each option we analyzed the five considerations on the assignment handout as well as a sixth - the personal utility of working with this solution. Stemming from this analysis, we decided to incorporate React, Django Python, GitHub Actions and SQLite3. I will now provide a short summary of the key reasons why we chose to work with each of these technologies as opposed to their competition.

For the frontend we considered Vue.js, Angular and React. Vue.js is the most lightweight and easiest to learn but we have had previous exposure to React giving us a sufficient head start. All three frameworks cover the problem domain sufficiently, integrate with our other technologies and offer libraries that could speed up development. React may be the most popular framework but all three have a gigantic community and rich ecosystem. Next, all three technologies perform similarly on the same benchmarks and easily provide the performance/speed required. Vue.js is the most lightweight as it offers an incremental approach to development. However, all three technologies scale well on the problem domain. Effectively, the most significant reason why we chose React ended up being the personal utility. Our partner on the group project has an existing software that uses React making it necessary to familiarize ourselves with this technology. There is also a large, long-run utility as, given it is the most popular framework, it may ease our way into a software engineer career path. Hence, we chose React.

For the backend we looked at Express.js, Flask and Django. Django may have the steepest learning curve when compared to Express.js and Flask. However, it has a simple yet powerful ORM and comes with a functioning Admin panel. This is extremely relevant to our problem as we would need a way to use SQLite and manage data based on our models. Also, Django is the most mature of the frameworks and we have been interested in working with it for a long time - giving it the greatest personal utility. In terms of the other considerations there are no key differences. For example, all three technologies are popular, performant and scale. Essentially, we decided to use Django due to the ease of development and utility.

For CI/CD we considered GitHub Actions, CircleCI and Jenkins. We have basic knowledge of GitHub Actions and no knowledge of CircleCI/Jenkins. Although GitHub Actions is by far the least mature technology it allows us to create SDLC workflow automation of any difficulty and can compete with older CI/CD technologies. GitHub Actions have a worse time performance, compared with the other CI/CD technologies but it is sufficient for this assignment. Also, it may not be the most popular technology, but we believe that it has great potential on the future market. Since, GitHub Actions is free for csc301 students we believe that now would be the perfect opportunity to familiarize ourselves with it which is why we decided to use it on this assignment. Please see Image 1 for a deployment diagram.

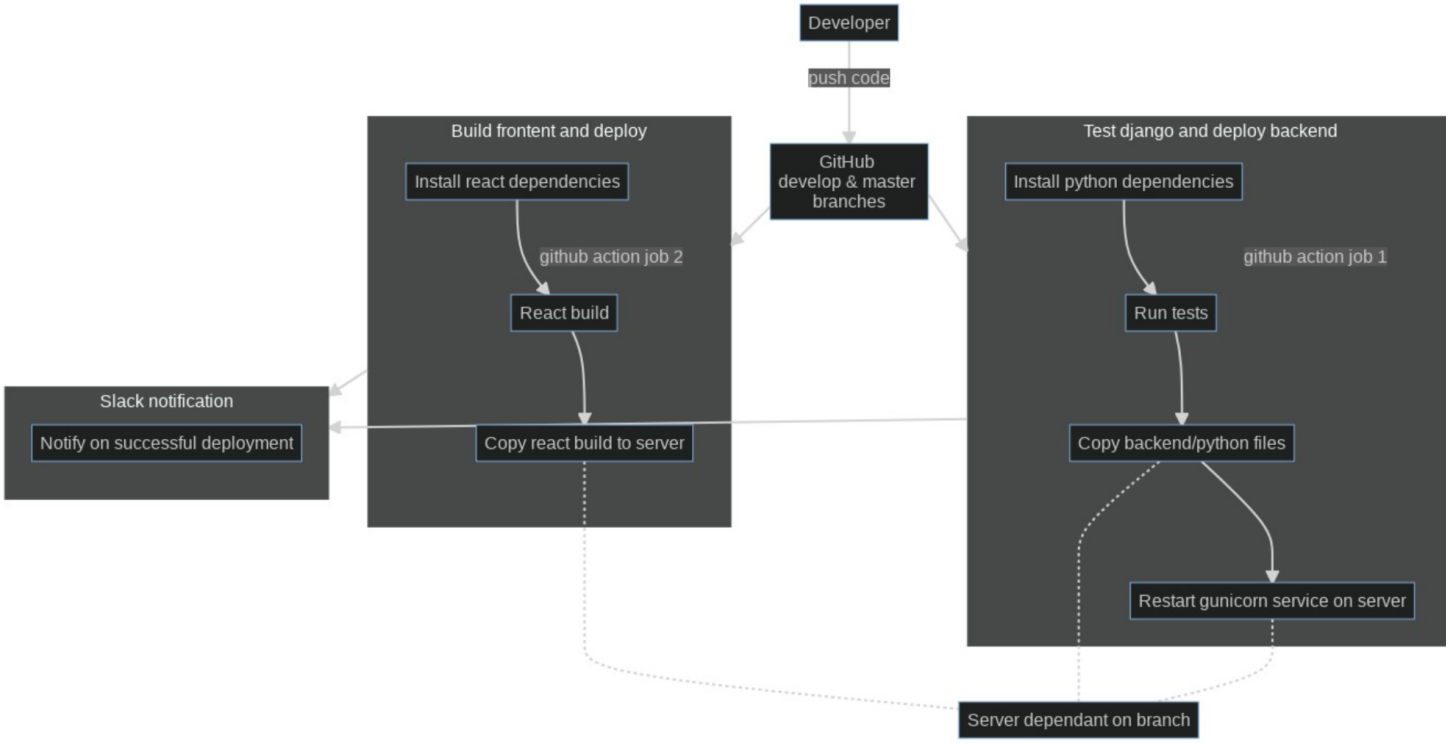

Image 1: Deployment Diagram

Choosing hosting, we decided to use AWS. Although it is expensive, we decided that starting learning AWS now is a good decision for our future career. Alternatively, we could have used a free hosting like Heroku, which is great for student projects because it is easier to configure. We created two virtual machines on AWS: one for the DEV environment and one for the PROD environment. Because our backend runs on Django we used the Python WSGI HTTP Server Gunicorn to wrap the backend. To redirect requests on the port 80 to the calculator backend we used the web server Nginx. Please see Image 2 for a server diagram.

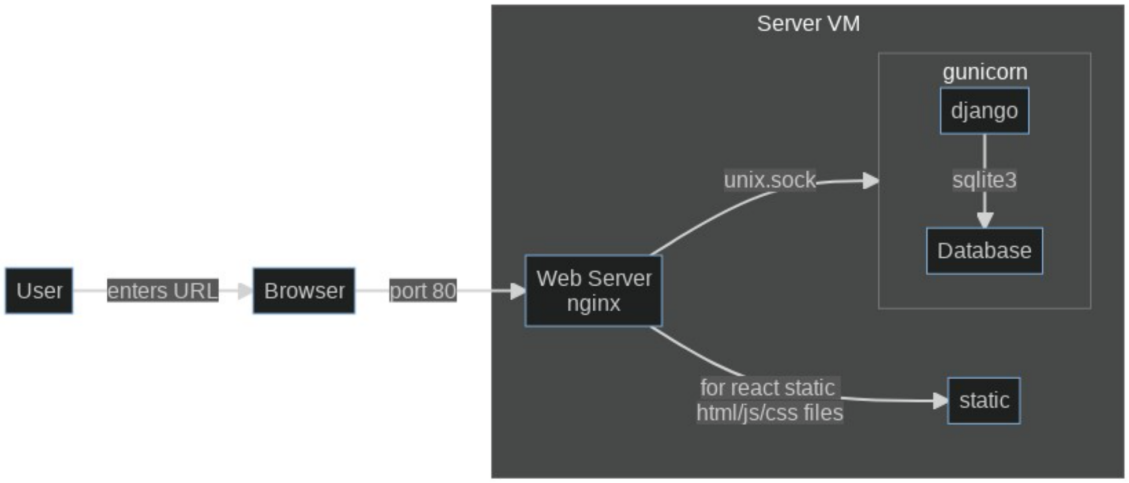

Image 2: Servers Diagram

We decided to use a database for this assignment that we can easily integrate with Django in order to manage the products. We considered PostgreSQL, MongoDB and SQLite3. Among these, SQLite3 is by far the easiest to learn. In fact, we already have experience with both PostgreSQL and SQLite3 and decided that SQLite3 felt more natural for us. MongoDB, in contrast, is something we have little to no experience with and is much harder to learn than SQLite3. SQLite3 is not the most popular database because of its limited functionality. In SQLite3 the database is a file, which means that only one process can access it at a time. This implies a significant performance trade-off that makes SQLite3 unsuitable for large scale projects. However, SQLite3 is performant enough for this assignment so this consideration is not heavily weighted in our analysis. In terms of utility, MongoDB would be great. This is because our project partner has an existing project that uses it. However, MongoDB is not out of the box integrated with Django which could add additional complexity. In short, all three choices seem to have significant trade-offs. Therefore, given that we have made all our previous decisions with a strong focus on utility, we decided to focus on ease of development and select SQLite3 for this assignment.

Frontend : React

Vue.js

(1) Vue.js is a Model-view-viewmodel front-end framework for constructing UI and single-page applications. Vue is based on HTML, CSS and JavaScript which are incredibly popular languages (https://insights.stackoverflow.com/survey/2021). This, plus the fact that Vue.js itself has a major following (with more github stars than React.js) implies that it is easy to develop with as there are plenty of resources such as documentation and stackoverflow to consult. Vues.js also follows an incremental, beginner friendly, approach to building web applications further underlining its low entry barrier.

(2) Vue.js is a mature technology that is frequented by several major corporations such as Adobe. This can partially be attributed to the tooling around it. For instance, Vue CLI is a full system for rapid development providing several features such as interactive project scaffolding and a full graphical user interface to create and manage Vue.js projects which could be useful when maintaining a project. Next, it also offers official libraries and packages such as vue-router as well as a wide range of third party assets such as Nuxt.js and Gridsome which provide similar functionality to Next.js and Gatsby in React. Also, there exist many libraries for easily customizable web elements such as BootstrapVue which would allow us to quickly prototype a simple store/checkout interface.

(3) The popularity of Vue is also outstanding as it is among the top five most frequented frontend solutions.

(4) Vue is mainly a front-end solution but can integrate with back-end solutions like Django. In order to support more domains, such as ML more integrations may be required which could introduce complexity to projects. However, Vue.js+Django (Django is the backend we will use) covers this specific problem domain sufficiently.

(5) Overall performance is what makes Vue a great tool. This is because Vue is exceptionally fast and only 18KB large after gzipping. Additionally, you may incrementally adopt parts of Vue which further underlines this.

(6) One more criterion is the utility of learning the framework. For instance, Vue is incredibly useful for building web applications which we will also work with in our term project. Finally, knowing the basics of Vue is a great skill that can potentially slot you a job with corporations that work with it such as Gitlab. However, it may not be as popular on a production scale as alternatives like React.

Angular

(1) In 2016 Google rewrote AngularJS in order to resolve performance issues and released Angular. Unlike in Vue, it is recommended to write apps in TypeScript when using Angular. TypeScript is JavaScript with syntax for types which gives better tooling at any scale. In the long run, this could make developing web apps with Angular more straight-forward. However, in the short run this could complicate development as we would need to familiarize ourselves with both JavaScript and the syntactical superset TypeScript. In short, Angular has a higher entry barrier than Vue albeit also featuring good documentation and other online resources.

(2) Similar to Vue, Angular is mature. Major companies that use it include Google and Microsoft. Just like Vue, Angular has their own CLI, called Angular CLI. Effectively, it is possible to initialize, scaffold, develop and maintain web apps. Again, just like Vue, Angular has a rich ecosystem of third party libraries and plugins that can speed up development. For example, Angular Material or ng-bootstrap would be useful when constructing UI components.
(3) Angular offers a complete solution to building single page web apps which would make it self sufficient for a client side “ecommerce store”. That said, it would still be highly beneficial to integrate a backend like Django which, in Angular, is entirely possible.
(4) In terms of popularity Angular seems to be taking the lead - albeit not by much. Again, this would not strongly weigh in on our comparison between the two.
(5) Angular appears to be of a larger scale than Vue. This may be due to the fact that it does not support incrementally adopting parts to the same extent that Vue does. However, Angular and Vue perform similarly on benchmarks meaning there are no performance concerns when it comes to our web app. That said, the speed of the solution built in Angular would likely be slower. This is the main reason to pick Angular over Vue would be to utilize its many features.
(6) In terms of utility Angular bests Vue. This is because after developing in Angular we would be familiar with both TypeScript and JavaScript. Also, more of the production scale releases that require teams are built using Angular meaning it may make pursuing a job in software engineering even more viable.

React

(1) React has a gigantic following and is easily the most popular frontend framework. This means that there are a huge number of online resources to consult. The documentation is also good. The key reason why React surpasses Angular and Vue in terms of ease of development is that we already have some exposure to it which will greatly speed up production time. This is despite its moderate learning curve.
(2) React was launched by Facebook in 2015 and continuously developed since. This means React is not strongly bug ridden. Additionally, due to its outstanding popularity there are a lot of libraries to be used. For example, something like semantic-ui would be helpful when prototyping a user interface for our online shop. However, the three frameworks do not differ by much in this respect so this consideration was less central in our analysis.
(3) React covers the frontend domain. However, since React uses JavaScript it would be relatively straightforward to implement a backend without further imports. That said, we have decided to use Django for the backend and React can integrate it. Again, all three frameworks don't differ by much in this respect.
(4) React is by far the most popular framework with a following almost twice as large as that of Vue.
(5) React is less heavy than Angular and heavier than Vue. The performance of all three is comparable. One could argue that the speed of solutions built in React depends on the developers knowledge of the framework. Effectively, the speed of building this specific solution would be fastest in React as we already have had some exposure to it.

(6) The most significant reason for choosing React is the utility. We have already confirmed that our project partner has an existing project that uses React for the frontend. This means that working with React now would also benefit us in the near future. Also, learning React would be extremely helpful in the long run as it is used by many production scale companies such as Facebook who may be looking for React developers in the future.

Backend : Django Python

Node.js Express

(1) Express is a minimalist framework for Node.js. This means that there is only a thin layer of fundamental web application features making it manageable and straightforward to learn. It is also easy to set up and extend with plugins if needed. Also, we are familiar with the technology and could use the same tooling for both the front and backend which further trims the complexity of development with Express.
(2) Express was released in 2010. This means that it was developed for a long time which in turn implies that it is easily mature enough for our purposes. There are several plugins which can be integrated to Express.js and it is even possible to write your own. However, for the purposes of this assignment Express.js is self contained.
(3) This technology would sufficiently but minimalistically cover the backend part of our app. This means that covering domains such as ML would be harder than with other frameworks but this does not apply to our simple ecommerce shop.
(4) Express uses JavaScript which is the most popular programming language. In turn, Express has a significant following.
(5) Due to the minimalistic and opinionated nature of the framework one could argue that we could quickly build our solution with it while maintaining a small scale. In terms of performance, Express.js is definitely performant enough for a small scale web app.
(6) There is some utility to be had when choosing to work with Express. Some companies that use it include Uber and IBM so it is applicable when searching for a job or trying to implement similar functionalities to, for example, Uber. However, Express.js is not the most popular framework and is being outrun by less dated technologies that could be more worthwhile.

Flask Python

(1) Flask is a micro-framework for web development using Python. Flask itself is relatively easy to learn due to the fact that it's lightweight nature makes its functionalities less occluded. The fact that it uses Python is also extremely helpful as this is the programming language we are most comfortable with.
(2) Flask was released in 2010 meaning that it has been around for roughly the same time as Express. This means that in terms of maturity the frameworks come out about equal. Flask however, does not come with libraries that support development with databases such as SQLite meaning that additional effort would have to be put into pinpointing which libraries must be imported for our purposes.

(3) Just like Express, with a few imports Flask would sufficiently cover our problem domain. The fact that it uses Python is, in this respect, helpful as it supports domains like machine learning which could be interesting for us on future projects.
(4) In terms of popularity Flask outranks Express.
(5) Again, this is very similar to express as due to the minimalistic nature of the framework one could argue that we could quickly build our solution while maintaining a small scale. Likewise, Flask is performant enough for this assignment. In fact, Flask is being used by websites such as LinkedIn to power their backend.
(6) The fact that this framework is more popular than Flask could imply that it is more useful to learn. However, Flask is similarly dated which could stunt its usefulness in the future.

Django Python

(1) Django is a full-stack framework with more functionalities than Flask. This means there is a steeper learning curve as we would need to familiarize ourselves with all of its features instead of gradually adopting them into our project. However, we have determined that Django is still easier to develop with than Flask and Express.This is because Django includes a simple yet powerful ORM that supports many relational databases such as SQLite (which we have chosen to use) out of the box. Additionally, Django comes with a functioning Admin panel which will make it very easy to manage data based on our models.
(2) Django is the most mature of the frameworks listed. It was released in 2005 and has been developed since. It supports a huge number of libraries which could further aid us on this project. However, in terms of the scope of this assignment, Django is a self contained solution.
(3) Django is a full-stack framework so it could cover the front and backend domain.
(4) Django is on par with Flask when it comes to popularity.
(5) Flask performs slightly better since it is smaller and has fewer layers. However, as projects get more complex and more features need to be added to Flask this difference vanishes. Again, Django is easily performant enough to support this project.
(6) We have been interested in working with Django as a backend solution for some time. We also believe that despite its age it is still of comparable relevance to Flask and Express. Hence, we see the greatest personal utility in using Django for our backend.

CI/CD : Github Actions

GitHub Actions

1) GitHub Actions is a built-in CI/CD feature of GitHub. Because both of our pair members have been using GitHub for several years already, we had a basic knowledge of GitHub Actions. Also, GitHub provides an extensive documentation for GitHub Actions (the

documentation is an open source! https://github.com/github/docs), which makes it easier to learn this technology.
2) Although GitHub Actions were introduced just a couple of years ago (in 2018), it allows to create SDLC workflows automation of any difficulty and can compete with CI/CD technologies which were on the market before GitHub Actions.
3) Any SDLC workflow can be fully implemented with GitHub Actions, starting with compiling code and running tests and finishing with deployment. Also GitHub Actions supports the majority of popular languages: Python, Java, JavaScript, etc.
4) Because GitHub Actions is a new technology, it is not yet the most popular one, but thanks to its integration with GitHub this CI/CD technology has great potential.
5) GitHub Actions have a worse time performance, compared with other CI/CD technologies. Though for projects using GitHub repositories, GitHub Actions can be a good solution because it does not rely on an additional 3rd party and thus does not create time overhead for communication.
6) Both of us were very interested in learning GitHub Actions because we think that this technology will soon be widely used. Also, GitHub Actions are provided for free to CSC301 students, which seems like a great chance to learn this technology.
7) For a beginner it is not easy to configure or debug GitHub Actions: the configuration has to be of a very precise format and any additional comma or space could ruin the whole CI/CD pipeline. If such an error occurs, it is difficult to find what exactly went wrong. Though, because the GitHub Actions format is standardized, once one learns this format it becomes increasingly easier to configure GitHub Actions.

CircleCI

1) Our pair had no previous knowledge of CircleCI. We would not be able to use this technology within the CSC301 course because to configure CircleCI for a GitHub repository a developer needs to have admin rights.
2) CircleCI is a mature technology which appeared on the market 10 years ago; since 2011 CircleCI grew and developed together with the DevOps community.
3) Like GitHub Actions, CircleCI provides support for the full CI/CD pipeline and the most of commonly used languages.
4) CircleCI is one of the most popular CI/CD technologies alongside Jenkins.
5) CircleCI is known for its high performance. This CI/CD technology supports parallelism for testing and building a software, which makes CI/CD pipeline scalable and improves its performance.
6) Because of the popularity of CircleCI it would have been great to learn this technology, but not possible within the CSC301 course due to the GitHub admin access limitation. Because CircleCI is free and does not require a server we plan to learn it on our own as a part of some side project later.
7) CircleCI offers a short learning curve and an easy setup.
8) CircleCI is a cloud-based technology and does not require running a dedicated server, which can be great for student projects.

Jenkins

1) We have no no previous knowledge of Jenkins. The same as CircleCI, this technology would require us to have admin rights in the CSC301 course repository to configure it, which means we could not use it for this project.

2) Jenkins appeared on the market at the same time as CircleCI - 10 years ago. It is open-source technology with a huge community.
3) Similar to GitHub Actions and CircleCI, Jenkins supports a bigger part of common languages and provides technology for creating a complete CI/CD pipeline.
4) Currently, Jenkins is the most popular CI/CD technology. Many software development companies use this technology because, while being the top market player, Jenkins is free and open source.
5) This is a high-performance technology with a support for parallelism: processes for tests and builds can be executed in parallel on multiple machines.
6) The same as learning CircleCI, learning Jenkins would have been a great idea if it was possible to do as part of CSC301. Both of us would like to know this technology which is highly in demand now. Though because Jenkins is a server-based technology, using it might be problematic for student projects as it requires managing an additional server and environment.
7) Jenkins is famous for its extensibility as well as for the number of plugins - Jenkins has thousands of plugins.

Database : SQLite3

SQLite3

1) Both of us had previous experience with the SQLite3 database, so even before starting this project we knew that it would be easy to use this database for our calculator.
2) SQLite was initially released in 2000. It is a mature technology with a limited application.
3) SQLite3 is a relational database suitable for projects where it is known in advance how the data will be structured.
4) SQLite3 is not the most popular database because of its limited functionality: SQLite3 database is a file, which means that only one process can access it at a time. This makes SQLite3 a good choice for small projects, like student projects or MVP, but not suitable for production of a project with multiple users.
5) Because only one process can access SQLite3 at a time, when a process tries to write to or read from the SQLite3 file, it might be blocked and then it will need to wait for its turn. This creates time overhead, decreasing SQLite3 time performance. Also SQLite3 is not scalable: it is not possible to store and update the same file with the same content over multiple processors.
6) Despite SQLite3 lower performance compared with more advanced relational databases (e.g. PostgreSQL), this database is perfect for a small student project like our calculator. SQLite3 is super easy to configure and it gave us a chance to practice a database deployment, which we had a minimum experience with before.

PostgreSQL

1) Both of us worked with the PostgreSQL database as a part of the CSC343 course. From our previous experience configuring PostgreSQL is more difficult than configuring SQLite3.
2) The first release of PostgreSQL was in 1996. 25 years later it is the most advanced open-source database technologies in the world.
3) As well as SQLite3, PostgreSQL is a relational database - it can be used only for projects with predetermined data structure.

4) PostgreSQL is not the most popular database, but it is definitely in the top 5 popular databases. Because PostgreSQL is free and open-source it is widely used by startups and academia.
5) PostgreSQL runs on a server which allows parallel access for multiple processes. Parallelism increases time performance. Also, running on a server allows PostgreSQL to be scalable: the database can be distributed between multiple servers.
6) Both of our pair members would benefit from having more experience of working with PostgreSQL: it would be helpful if we decide to do a Masters in Computer Science or if after graduation we get a job in a startup. However, for the purposes of this project PostgreSQL seemed like an unnecessary complex solution.

MongoDB

1) One of us had a previous experience with MongoDB as a part of the CSC309 course, but have not worked with it ever since, so this technology would require some learning / remembering curve.
2) MongoDB's first version was released in 2009. Now it is the most advanced non-relational database.
3) Because MongoDB is a non-relational database it can be used for any project where data does not have a uniform structure.
4) MongoDB is less popular than the most popular relational databases (because developers like structure!), but it is definitely the most popular non-relational database.
5) As well as PostgreSQL, MongoDB runs on a server which allows parallel access and makes it scalable. Because the data is written in a JSON-like document, it takes more time to retrieve data from a non-relational database compared to a relational database: to find some particular data, a process cannot skip rows like in a relational database and needs to read the whole JSON-like document.
6) We will have a chance to work with MongoDB during the CSC301 project: our partner's database is MongoDB. We think this is a great opportunity to practice using a non-relational database, which is needed to improve our understanding of databases.