The University of Winchester

Faculty of Business, Law & Digital Technologies

Digital Media Development Dissertation

DM3107: Major Research Project

Is the Jamstack the future replacement for building a backend?

Word count: 8,791

Adam Horne 1800270

19-5-2021

Abstract:

Websites have most commonly been built using monolithic architectures. Recently, the Jamstack has introduced a new way of producing websites with dynamic content, one without that of a server. Secondary research showed that the Jamstack's technical capability matches the traditional monolithic architectures and potentially surpasses it, providing cost effective, performant, secure and scalable implementation. Whether this new architecture would be adopted by industry, was unknown. A multi-method research project was undertaken with primary research conducted through the use of surveys and interviews. An interview was used to gauge the development industries landscape to find any correlations between that of the industry and the architecture that the developers used. Interviews were then used to develop a deeper understanding of why developers in a particular industry, according to the correlation, tended to choose certain architectures. The research gave insight into the benefits, drawbacks, liabilities and gaps that these architectures had. It allowed for a discussion into where the Jamstack currently stands in the development industry and how it may also potentially fit into other industries. The research concluded that the development industry a developer finds themselves in does have an influence on the technology they use. It was identified that the majority of agency developers already utilise the Jamstack due to the advantages it provides them, the project and also the agency. In-house and product developers tended to stick with monolithic architectures due to a fear of reliability on third party services. This however did not mean these industries were not utilising these third-party services, instead, they tended to be utilised in areas outside of the critical path where down time wasn't an issue. The project established that the Jamstack does have a place in industry. As the architecture as a whole matures, adoption and migration in other development industries, like seen in that of agencies, will become more apparent.

Acknowledgements

I would like to thank Debs and Rhys for their support over the duration of this Dissertation, but also a massive thank you to the entire DMD department for their help and positivity during my time at university.

I also have to give my appreciation to Emily who, despite being on placement and working incredibly hard, always found the time to let me bore her to death and read through everything.

Finally, I would like to thank everyone I interviewed and all those who answered my survey, this project wouldn't have been possible without your answers.

Contents

List of Figures:

List of Tables:

Glossary:

	Definition
JAM	JavaScript, APIs, Markup
JSON	JavaScript Object Notation
API	Application Programming Interface
LAMP	an acronym for one of the most common stacks for many of the web's most popular applications. Linux, Apache, MySQL, PHP
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
CSS	Cascading Style Sheets
CMS	Content Management System
Time to First Byte (TTFB)	Representing the time a browser waits to receive first bytes of the response from the server
First Contentful Paint (FCP)	Measures the point at which text or graphics are first rendered to the screen
First Input Delay (FID)	Shows how fast web pages respond to user input
Technology agnostic	Unbiased towards the use of different technology tools to solve different problems

Is the Jamstack the future replacement for building a backend?

Introduction:

Since I started web development, it has always been paramount to be confident in the technology I am using, and making sure it is the best possible solution for the project. My intention is to gain an understanding of what the future of web development may look like to be prepared when I enter the industry.

The Jamstack is a new architecture for building websites; traditionally and most commonly, websites have been built using monolithic architectures. Monolith means all in one piece, and a monolithic architecture describes a single-tiered website composed of components all bundled together into a single program (Haq, 2018). Server-side frameworks like Django, Laravel or Express are examples of monolithic technology. The Jamstack is an alternative to these stacks (Myers, 2018b). To consider a website a Jamstack site, it amounts to the markup and how it is served. The HTML must be served without it being compiled on a server at request time, which is what monolithic architectures do (Fayock, 2020). In this dissertation the Jamstack will be explored, with the aim of understanding how it compares to monolithic architectures widely used on the web. Comparisons will be made to grasp a technical understanding to assess whether the Jamstack will be able to technically replace what already exists. Further research will also investigate how the architecture may be able to fit into the industry.

Thesis Statement, Aims & Objectives:

This dissertation will explore how the Jamstack compares to current traditional approaches to web-development and whether the Jamstack will one day be able to replace these legacy structures, eliminating the need to write a backend.

Aim:

To understand whether the Jamstack is a realistic option in industry and viable against currently used technology and architectures.

Objectives:

- To explore how the Jamstack compares technically to traditional monolith stacks.
- To investigate and understand opinions of developers from different development industries on their chosen technology stack and why.
- Analyse the issues developers face with their technology stack and how the Jamstack may be more suitable.

Structure of Dissertation:

Literature Review	Critically analysing the current technical capability of the Jamstack. A range of material will be explored to determine the arguments for and against its use. A conclusion will be made as to how it compares technically to a monolithic architecture.
Methodology	Planning the research to be undertaken in order to understand opinions of developers from different development industries on their chosen technology stack and why. I will be justifying research methods, explaining questions asked, identifying the research sample and explaining how the data will be analysed.
Results and analysis	Analysing the results from the primary research to show that there is divide in the architectures chosen depending on the development industry. Through thematic analysis of responses, three main themes were identified in relation to the justification of the choice of architecture, these were technical capability, developer experience and cost.
Discussion	In this section of the dissertation the aspects identified previously will be discussed, analysed and compared to identify if the Jamstack has a place in industry.
Conclusion	Summarising what has been learnt from research undertaken and how aims and objectives have been met. Explaining if the Jamstack will be the future replacement for building a backend and future work.

Literature Review:

Introduction:

The JAM in Jamstack is an acronym for JavaScript (to handle data requests/interactivity), APIs (data retrieval) and Mark-up (compiled to static HTML files). These are the three concepts that are used to create websites using the Jamstack (Biilmann and Hawksworth, 2019). Not everything about the stack is a new idea, but only recently has technology allowed this approach to making websites possible. New tooling, workflows and ecosystems have been developed to accommodate the renewed interest of returning to the simplicity of HTML (Freestone, 2020). Coyier (2019), owner of CSS-tricks, agrees with this and describes the Jamstack as “a rejection of websites getting so complicated.” (Coyier, 2019 [online]).

Why the Jamstack was created and is used compared to previous monolithic solutions

As mentioned above, Freestone (2020) and Coyier (2019) describe the popularity of the Jamstack being due to an interest in returning to the simplicity of HTML. The Jamstack aims to provide a simpler solution to producing complex websites. Biilmann and Hawksworth (2019) support the idea of simplicity and believe the complexities are due to monolithic architectures that tightly couple the frontend and backend of the website to a server. The tight coupling often leads to complications, this is evident in Drasner’s (2020) case-study on Smashing Magazine. They recently migrated from a monolithic solution to the Jamstack due to issues surrounding performance, scaling and security. An example detailed was when they achieved virility and high traffic, the website often experienced performance issues and outages (Drasner, 2020). The Jamstack approach looks to solve these issues and takes inspiration from mobile apps. The approach states that even web apps should have the frontend completely isolated from the backend, communicating only over HTTPS (Segala, 2019). This inspiration from mobile apps demonstrates where monolithic architectures are lacking, highlighting the need for the Jamstack. Biilmann and Hawksworth (2019) argue that for IOS it was not a consideration to build a model in which the full user interface (UI) of an app would be reloaded from a server every time the user took an action. Instead, IOS introduced a model in which each app could be distributed as an application package, including assets to communicate with APIs (Biilmann and Hawksworth, 2019). The Jamstack follows this approach in order to deliver better performance, higher security, lower costs, ease of scaling and better developer experiences (Ekwuno, 2019). However, Freestone (2020) identifies disadvantages, stating that nearly all the advantages are due to enforced minimalism.

Advantages and Disadvantages of using the Jamstack

The introduction of the Jamstack has brought with it a new system architecture. A comparison between a monolithic stack and the Jamstack can be seen in Figure 1. It is evident that the Jamstack's architecture is considerably shorter, with only a client and a CDN. This will result in a more reliable website with faster deployments and simplicity in pushing new features compared to a monolithic architecture. Myers (2018a [online]) agrees with this argument stating that "the old, heavy" monolithic stack has longer deployment cycles which would often be set to deploy once a month to prevent issues. He compares it to the Jamstack which is quick to add new features, to increase deployment frequency and allow teams to iterate faster. With the Jamstack diagram, a user requests a page, the CDN matches the request to an already compiled response and is directly served to the browser from the CDN (Ekwuno, 2019). The Jamstack removes entire tiers from the stack and essentially shortens the chain compared to the legacy architecture. Tiers in which teams used to spend large amounts of time and money on for speed and reliability do not exist (Biilmann and Hawksworth, 2019). In place of the server, HTML is prerendered into static files which are served from the CDN, tasks that were once managed server side are now performed by APIs. At a Jamstack conference, Chris Coyier spoke about this, giving an example that, creating a website event that triggered a text message previously involved backend engineers. Now with serverless functions, frontend engineers can build this capability (Jamstack TV, 2018). Using APIs means much less happens in order to satisfy each request, with fewer points of failure, less logical distance to travel, and fewer systems interacting for each request (Biilmann and Hawksworth, 2019). Freestone (2020) looks at the stack from a different perspective, stating that advantages such as performance, are only good as there is no overhead to serving pages that come with dynamically generated content from a server. Plus, the site will be considerably more secure, economical and scalable only by virtue of there being no requirements for a server-side language or database for intrusion (Freestone, 2020).

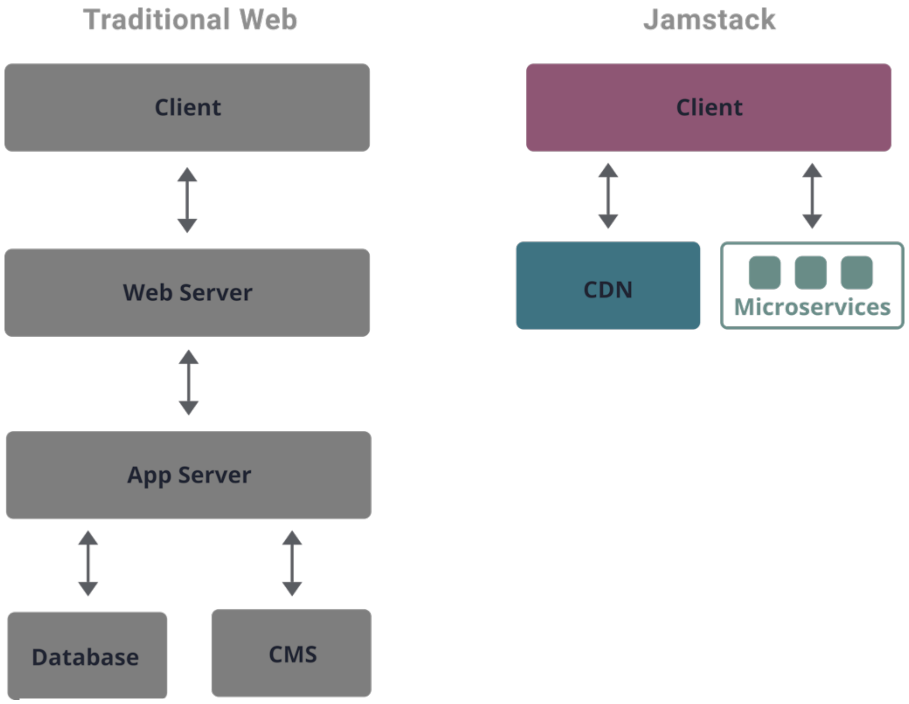


Figure 1 - A comparison between a monolithic, legacy web stack and the JAM stack - (Netlify, 2020)

Smashing Magazine are a perfect example of an organisation who have a high traffic website which they migrated from a monolithic stack to the Jamstack. The Jamstack is performant as pages are not compiled on-demand when users visit them compared to the old monolithic way. Instead, it serves pre-built content directly from a CDN where the site is served instantly (Drasner, 2020). The site is built ahead of time as a series of HTML, JavaScript, and CSS files (Segala, 2019). The performance benefits are evident in an article published by Denysov (2019) who compares how websites built using legacy stacks compare to websites built using the Jamstack. It covers metrics used to measure performance such as Time to First Byte (TTFB), First Contentful Paint (FCP) and First Input Delay (FID). In Figure 2 the results can be seen, they show that Jamstack websites outperformed in all areas (Denysov, 2019). Serving their website statically via a CDN allowed the Smashing team to avoid receiving database connection errors and prevented downtime even when an article gained a huge amount of traffic. When serving without a server, the prebuilt content does not have to be generated and served as it already exists, ready to be viewed. There is nothing being requested on the spot as it had to be with traditional stacks except for the entire static page (Drasner, 2020). However, since there is no longer a server, functionality traditionally provided by application servers will require JavaScript, where before they did not. This logic must now be client-side, creating potential performance issues by moving an abundance of logic to the client (Freestone, 2020).

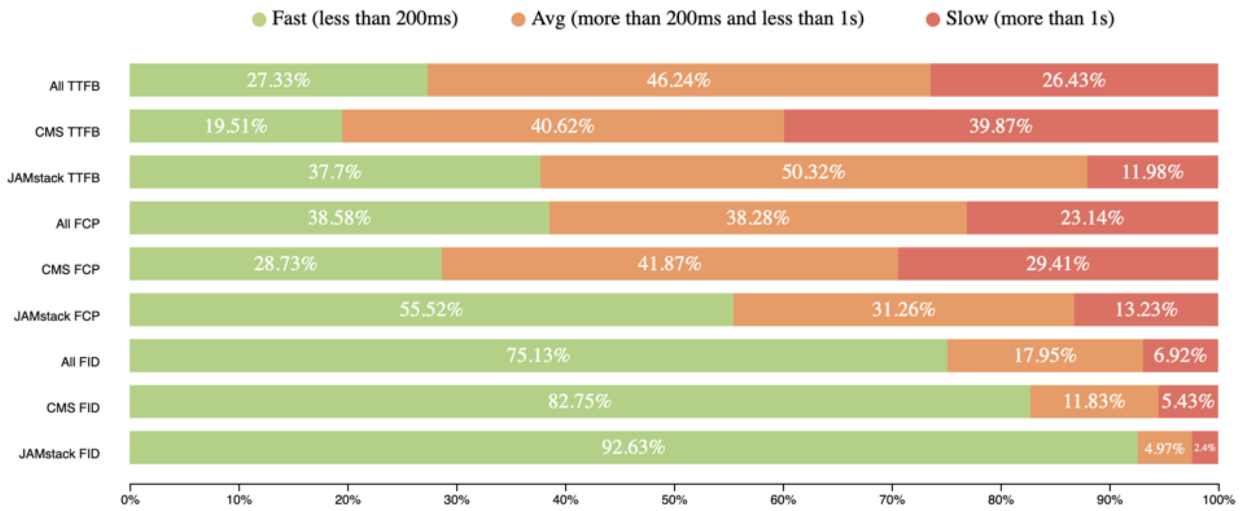


Figure 2 - Mobile speed distribution comparison between all web, CMS and JAM stack sites (CrUX, July 2019)

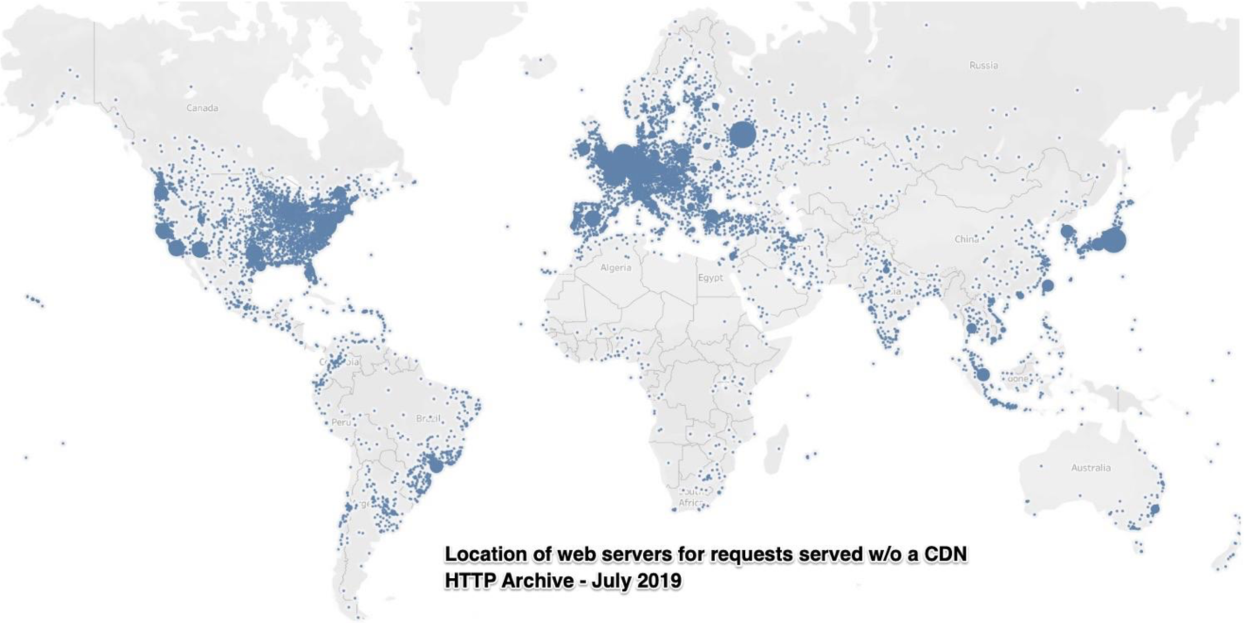
A huge factor to increased website performance using the Jamstack is generating the static files and serving this prerendered content from CDNs, eliminating large amounts of server and network latency (Biilmann and Hawksworth, 2019). Kyle Mathews, the creator of Gatsby agrees with this, and states that prerendering and CDNs provides both performance benefits and up-time benefits. He states that they remove the worry of operating the website since they are files sitting on a CDN meaning it can not go down as it is not running code (The Changelog, 2020). To prove the importance of serving the static HTML instead of using a framework Leatherman (2019) conducted an investigation. He compared the FCP of a raw 8.5MB HTML file with all of his 27,506 tweets on it, to a client rendered React site with one tweet on it. The results of his investigation showed “8.5MB of HTML won by about 200ms” (Leatherman, 2019, [online]). This demonstrates how much faster serving a website statically, using the Jamstack can be compared to even client-side rendering data. In an interview, Kyle Mathews supported these speeds and stated it is due to instant cache invalidation, “in the end of the day, it is about caching” (Chen, 2019, [online]), he explains that “if you have the cache there, it doesn’t redo a lot of the work” (The Changelog, 2020). Having instant cache invalidation allows for the data that has been cached to be instantly available, compared to a monolithic framework that has to fetch the data on every load of the page. By serving this content via a CDN, the content can also be distributed across the globe, closer to potential visitors. There is no central point of origin, which is vital for a website to be fast (Drasner, 2020). Gibb (2019, [online]) defines latency as “the delay between a user’s action and a web application’s response to that action, often referred to in networking terms as the total round trip time it takes for a data packet to travel”. By statically generating websites and using CDNs the content is as close to end users as physically possible (Biilmann and Hawksworth, 2019). Meyer (2020, [online]), recently put out a plea for sites to ‘Get Static’ so they can better serve the traffic demands of COVID-19. Meyer (2020, [online]) explains “too many sites are crashing because their CMSs can’t keep up with traffic surges and too many sites are using dynamic frameworks that drain mobile batteries and shut out people with older browsers”. Meyer (2020, [online]) also highlights that performance is not just something to aspire to, “right now, in some situations, performance could literally be lifesaving to a user, or their family“. He goes on to say “I can’t tell you how best to get static-only you can figure that out. Maybe for you, getting static means using very aggressive server caching.” (Meyer, 2020, [online]). Freestone (2020, [online]) agrees with this, describing the use of caching in his defence of monolithic architectures stating, “a well-configured full-page cache (using something like Redis or FastCGI) will effectively give you the performance of a static-site”. However, for sites to do this on traditional stacks they need to manage their dynamic content into various caching layers. In order to achieve Meyer’s (2020) and Freestone’s (2020) vision of getting static, traditional sites have to add an extra layer of complexity (Biilmann and Hawksworth, 2019; Chen, 2019). By moving to the Jamstack it allowed Smashing Magazine to see instant cache invalidation with no extra overhead.

The migration to the Jamstack can provide an improved workflow and developer experience. Drasner (2020) states that for Smashing Magazine, one reason for the migration was to unify multiple systems,, instead of keeping them separate with different stacks. He states that having to context shift between the different technology stacks became a maintenance problem that tasked their developers.
This migration helped keep development costs low and developer experience high (Drasner, 2020). Additionally, a migration results in cutting out entire tiers from the stack which reduces the size and complexity of the architecture, which is directly proportional to the quantity of people and range of skills required to operate it (Biilmann and Hawksworth, 2019). On the other hand, cutting out entire tiers from the stack along with a server, leaves little choice other than to distribute responsibility of functionality to third parties (Freestone, 2020). In terms of Smashing Magazine, they required search capabilities, e-commerce, multiple multi-user CMSs and comments on articles (Drasner, 2020). These

are integral features to the success and functionality of their business and have been excluded by their decision to migrate to the Jamstack (Freestone, 2020). Freestone (2020) identifies that although the Jamstack aims to take the web back to basics, the removal of a server can be seen to add complexity. The second a dynamic feature is needed they have to rely on a third-party service, awkwardly spreading out the functionality (Freestone, 2020). Fayock (2020) supports this and warns that websites built with the Jamstack may depend on these services. Although, he goes on to state a static-site-generator can be utilised to manage API requests at build time. Biilmann and Hawksworth (2019) expand on this, stating should a build fail, potentially due to a third-party service, the result would be a failed deployment which would not be conveyed to the user and the last successful build would stay deployed.

Advantages and Disadvantages of Monolithic Solutions

Whilst tooling for the Jamstack has come a long way, most static-site builders do not provide the flexibility of monolith frameworks (Freestone, 2020). They are often full featured, and functionality is largely done-for-you (Zey, 2019). Where disadvantages start to be seen, are in terms of performance, speed and ability to scale. These architectures need to generate and deliver HTML each time there is a new visitor, significantly slowing down performance (Biilmann and Hawksworth, 2019). This is similar to what was experienced at Smashing Magazine, when virility was reached they would have issues with outages for this reason (Drasner, 2020). To combat this, organisations would try to scale their infrastructure in preparation of site traffic. Biilmann and Hawksworth (2019) stated that not only is this expensive, but it is difficult to get right, and teams often over-provision their infrastructure to prevent downtime. This is due to there being no clear separation between the infrastructure required to generate the site and that required to serve it. Denysov (2019, [online]) shared this opinion and explained “a fast back-end that takes a long time to reach users will be slow, and likewise, a slow back-end that is quick to reach users will also be slow”. Good web performance requires effort at all levels of the stack, and a site will only be as performant as its least performant level (Biilmann and Hawksworth, 2019). Latency is another aspect that can contribute to site performance, Calvano (2019) found that out of 468 million requests from 5.3 million pages tracked in the HTTP Archive, 48% were not served from a CDN. He then visualised where these were served from which can be seen in Figure 3 – the client being in California. If an organisation wants to be global, they need to serve their content from as close to their client as possible unlike what can be seen in Figure 3. This is especially crucial when clients are being served in areas where internet connections might not be as strong. Due to the way that HTTP works, the round-trip-times (latency) dominate performance more than bandwidth does, making it crucial to reduce latency times (Belshe, 2010). This is demonstrated within the Jamstack when serving from CDNs.



Location of web servers for requests served w/o a CDN
HTTP Archive - July 2019

Figure 3 – Location of web servers for requests served without a CDN (Calvano, 2019).

Summary

Overall, it can be seen that the Jamstack is an excellent decision for the development benefits it brings over Monolithic solutions (Manandhar, 2020). Smashing Magazine highlighted this in their case study. It highlighted how using the Jamstack allows all websites, no matter the size of the organisation, to aid from its benefits for little to no cost. Coyier (2019, [online]) supports this and says “very modern websites can be built on Jamstack. Just about any kind of site can be”. Previously, these advantages were only possible using a monolithic architecture by large, enterprise companies due to their high costs, complexity and need for a large, specialised team. However, there are still situations when there is a need to keep with monolithic architectures despite the risks. This is due to the risk of change often outweighing the risk of staying the same. Biilmann and Hawksworth (2019) identify for large companies, this would mean investment in architecture, talent and a fear of no longer having the flexibility, features, and tooling of a mature ecosystem. They go on to state for small businesses, the change may mean rebuilding web infrastructure, pulling resources away from critical growth initiatives. However, long term a migration will free up these resources with the utilization of APIs offered by external services. It is someone else's responsibility to ensure that they are available and scale as needed. Companies do not need to maintain the third-party APIs as they are maintained by a team that guarantees their reliability. In addition, the burden of General Data Protection Regulation (GDPR) compliance is on the service providers (Segala, 2019). Making it unnecessary to keep skilled teams to maintain these applications. These advantages may make the migration independent of the size of the company worthwhile.

The findings from this literature review were incomplete in answering the research question. My primary research aims to understand the thoughts, feelings and opinions of developers on their stacks to see if for them, there is a need for it. This will be based on if they think the Jamstack's use case fulfils their needs. From this research, inferences will be made in order to find out whether it could replace the need to ever build a backend.

Methodology:

Mixed Methods Research and Research Design

The research conducted for this report has used various primary and secondary methods to collect qualitative and quantitative data. Bell and Waters (2014) assert that primary research is the data which comes into existence during the period of research, whereas secondary sources are the interpretations and collations of pre-existing research. Throughout the literature review, secondary data was sourced. It was important to collate all information relating to the Jamstack and understand the different sources available (Koshy, 2009). During research, all sources found related to the technical feasibility of the Jamstack and allowed the objective of comparing the technical capability of the Jamstack to other solutions to be reached. This process ensured a secure understanding and identification of existing research and exemplified the gaps surrounding the subject (Koshy, 2009). Identified was the technical feasibility, but exemplified was a gap in whether the architecture will have longevity and be adopted in industry over monolithic solutions. The literature review refined the research topic and discovered what primary research needed to be done (Koshy, 2009).

The primary research undertaken will collect quantitative and qualitative data. Leavy (2017) defines quantitative research as a focus on statistics and generalizations to achieve objectivity, control, and precise measurement. She goes on to explain this type of research is most commonly used to identify relationships, associations, and correlations. On the other hand, qualitative data has a focus on depth of meaning and involves people's subjective views to produce rich, descriptive data (Leavy, 2017). Collecting both qualitative and quantitative data allowed the research question to be explored and answered through the use of descriptive data and statistics (Leavy, 2017). Quantitative data allowed for exemplification of the relationship between the development industry and the technologies they use when undertaking a project (Leavy, 2017). Barnham (2015) supports the use of qualitative data and highlights how it allows for expansion upon why certain technologies have been utilised by professionals within the development industry. Leavy (2017) states that there are four types of mixed method designs, they are; convergent or concurrent, explanatory sequential, exploratory sequential, and nested. Within this research the method design used will be explanatory sequential. Creswell (2015) states that sequential designs are based on the order in which the research begins. He identifies that when using explanatory sequential designs, they begin with quantitative methods, which are followed by qualitative methods. This is useful for the research as it allows for the quantitative data to be explained by the qualitative findings (Creswell, 2015).

Validity and Triangulation

To collect the quantitative and qualitative data, there will be two methods of data collection, they are a survey and interviews, which will be discussed in further depth below. Throughout this dissertation, it is important that the research conducted is valid and reliable. Bell and Waters (2014) support this point by stating that data should always be examined to critically assess for these two factors. Zohrabi (2013) describes the use of triangulation as a valid way of collecting information from a wide range of sources to strengthen the reliability and validity of data. Denscombe (2010) defines it as a way of collecting a variety of research with a range of methods to better understand the topic being investigated. Laws *et al.* (2003) further supports Denscombe's (2010) definition and highlights the importance of seeing one topic from many different perspectives to confirm whether the findings in one study correlate with another study which used different methods. Bell and Waters (2014) explore reliability and define it as the extent to which reproducing the study produces similar results. Zohrabi (2013) discusses the benefits of using quantitative research when triangulating research as it is straightforward to assess numerical data. However, Laws *et al.* (2003) Zohrabi (2013) and Bell and Waters (2014) describe the limitations and difficulties with analysing qualitative data as it is subjective to opinions and in narrative context. From this, it was concluded, and supported by Jupp and Sapsford (2006) that validity is a better concept to verify qualitative data. This is how the data in this dissertation will be verified. Jupp and Sapsford (2006) define validity as how the design of research will verify and provide credible conclusions. They explain that research should measure or characterise what the author intended in order to create valid research but recognise the difficulties when using qualitative data due to its subjectivity. However, using validity the purpose is not for the results to be exactly replicable rather to agree that based on the data and the research methods used the findings and results are consistent and dependable (Zohrabi, 2013).

Methods of Primary Data Collection

The first method which will be used is a survey. Koshy (2009) defines a survey as a detailed and quantified description of a population, involving the systematic collection of data. In order to achieve this, a survey was sent to a large sample of developers from different development industries and job roles. This decision was made in order to understand which web technology stacks developers would choose when given a particular scenario. This would allow qualitative data to be collected and generalizations on developer's attitudes towards architectures to be analysed (Grey, 2004). This met the objective of identifying which technology developers use and why. Koshy (2009) describes the use of surveys and exemplifies the two types as analytical and descriptive. The survey used within this dissertation involves a descriptive survey as they are designed to measure the characteristics of a population whilst gaining an insight into that population's attitudes, values and opinions (Jupp, 2006). It was important for the survey to explain thoroughly and descriptively what was expected of the participants in order for them to give their opinions on the use of different technologies. Grey (2004) supports the idea of describing accurately and thoroughly and highlights the importance of describing the opinions contained within the survey so it can be explained. Due to this it is necessary to provide participants of the survey context prior to answering questions. The objective was to identify the technology participants use, however developers are meant to be technology agnostic so it was necessary to provide context of a project for participants when answering the survey.

Survey Sample

In order to generate data from a wide sample of developers from different development industries, the survey was posted on a web development focused community on Reddit: r/webdev. The forum is a very broad community, being based on web development as a whole and therefore not biased towards any technology or architectures. The subreddit also has eight hundred and twelve thousand members so was a good place to obtain a range of participants from various backgrounds. With this in mind a minimum of fifty respondents of the survey will be required, less than this and the sample size will be too small and therefore inadequate to spot patterns, correlations and generalize populations of people. The survey was open for responses for a duration of seven days.

Survey Questions and Expected Responses

Context: For the purpose of these questions, I will provide a brief for the development of a project. With this project in mind, you will be able to answer the questions below.

You or the company/agency you work for are about to undertake the development of a new website, this app will be a simple e-commerce platform and blog, that the owner will need to add content and products to. With this in mind:

Q no.	Questions	Reason for question	Expected Responses
1	Which area of web development do you specialize/are strongest in? • Front-End • Back-End • Full Stack	To see if where abouts on a website you work affects your opinion and see any patterns	Front-End / Back-end / Full Stack
2	What sort of development work do you/the team/company you work at undertake? • Freelance • Agency • In house, working on a product • Learning/student • Hobbyist	To see if the kind of work you do affects your need for a backend	Freelance / Agency / In-house etc
3	What is your current job title? If a student or hobbyist, please just state this again	Understand the job roles of those answering the survey	CTO

4	What architectural approach would you take when producing the website? • Server side rendering e.g. The use of backend frameworks such as Laravel, Django or Express to render and serve pages from the server per user request. Wordpress etc. • Serverless (JAM stack) e.g. pre-rendering a static build of the website and serving that via a CDN to each request. "As-a-service" products often used to provide dynamic capabilities. • Client side rendering e.g. requesting data from the clients browser. Often via a modern frontend framework (such as React, Vue, Next.is etc). Backends aren't used for routing or generating HTML per requests but instead to just provide the data.	To see how people think about approaching projects and the rendering methods available	Server side rendering / Serverless / Client side rendering
5	Which technology stack / technology would you choose to carry out the project? • (Specify) • (Specify) • (Specify)	To see what stacks people would use for developing the example project	E.g. React, Express, MongoDB, node Or MERN

	- …		
6	Please provide a reason for your choice of technology and stack	To understand why they are using the said stack	It is fast, cost effective and what the rest of the team are skilled in
7	Would you isolate your frontend from your backend? - Yes - no	To see if there is potential to use the Jamstack	Yes / No
8	Have you ever had issues surrounding website performance, security or down time with the stack specified?	To see if they have ever experienced any problems with this stack	Checkboxes / yes, issues with performance
9	Have you used any of these "as-a-service" products in projects undertaken: Headless CMSs, Algolia, Auth0, Snipcart, Stripe, FaunaDB, Firebase, Serverless functions, or other SaaS products	To find out if they are not using the Jamstack/serverless but they are already using some of the services used for the Jamstack	Yes / No
10	Which of the following services have you used before? Headless CMSs - Algolia - Auth0 - Snipcart	To Identify the type of services used.	Checkboxes

	- Stripe - FaunaDB - Firebase - Serverless functions - Other (Specify)		
11	Through the use of these services did you find there were any advantages or disadvantages compared to building out the functionality in your own backend? If so what were they? (Specify)	To find out the participant's thoughts and opinions on these services compared to building a backend	I thought there were advantages, it allowed for a faster, more secure implementation of crucial functionality. Reduces the margin for error on our behalf.
12	Do you feel using these 'backend-as-a-service' products together could eliminate the need to develop your own backend? - Yes - No	Understand whether they think these services could completely replace building backends for them	Yes / No
13	Can you give a reason for your answer?	Then understand why they think this is the case	No, there will always be a need for custom functionality
14	Is the JAM stack something you are familiar with and have experience using in projects? - I've heard of it but don't know much about it	Whether the participants have experience using the Jamstack or know what it is	Multiple Choice

	• I'm familiar with it and how it works but choose not to use it over other stacks • I'm familiar with it and how it works but need to find time to try it out • I actively use it in projects • I've never heard of it		
15	Why haven't you/don't you use the "as-a-service" products?	To understand why the participant does not use the APIs that make up the Jamstack and could replace the backends	They can get very expensive quickly and don't offer the customisation we need
16	Why do you choose not to use the Jamstack over other stacks/solutions?	To understand the reasons why the participant chooses not to use the Jamstack over other solutions	We don't want to have our core functionality to our product be reliant on a third party. We want control over it.
17	Why do you use the Jamstack over other stacks/solutions?	To understand the reasons why the participant chooses to use the Jamstack over other solutions	The speed, performance and ability for the project to scale with incredible ease.

Table 1. Survey Questions and Expected Responses

The second method which will be used for the research is an interview. An interview is defined by Grey (2004) as a conversation between people in which one person has the role of researcher who often has a set of questions to pose to the other person involved. For this research, three individuals from various areas of development: The Chief Technology Officer (CTO) of a product, the Head of Technology at a Digital Agency and also a Junior Full Stack developer at an Agency were interviewed. According to Koshy (2009) interviews are an imperative research method as they can be used in conjunction with a survey to follow up issues which have arisen. The survey aimed to find quantitative, statistical data in order to find relationships between the development industry and the architectures developers choose to carry out a project. An interview would then be used to follow up the survey. This will allow for each field of development to be explored in more depth, giving individuals an opportunity to explain their feelings, attitudes, values and motivation. Through the use of a qualitative approach, individuals are able to justify and explain the identified patterns previously recognised in the quantitative data (Koshy, 2009). Grey (2004) supports the use of interviews to extract qualitative data.

Due to the nature of data needing to be collected, participants were required to answer questions about their job and justify the decisions they made. According to Grey (2004) surveys may hinder the quality of the answer. He states this may be due to developers who are in a prestigious role being concerned with where the information will end up. For example, a CTO or a manager may feel worried that the information they share is confidential, and not want it used in research. On the other hand, Grey (2004) explains where a survey is lacking, is where an interview is suitable. Koshy (2009) supports this describing that people are often willing to give more honest answers in a one-to-one situation, enjoying talking about their role, skipping the mundane task of filling in a questionnaire. They also state that the main purpose of conducting interviews is to gather responses which are richer and more informative than questionnaire data. For the interviews, a semi structured format with premade scenarios will be used. Koshy (2009) defines a semi structured interview as a set of questions which will be explored, but also sub questions which can be used if the previous ones do not extract valuable data, see tables 2 and 3. Koshy (2009) implies this is the best way to interview as it allows for all questions to be tailored to the data which needs to be collected. By using this interview style, the researcher is able to probe for more detailed responses, in this case, with relation to the Jamstack, instead of off topic technology which in this scenario may be easy to talk about (Grey, 2004). With permission, the interviews will be recorded to give full attention to the context of the interview and pose sub-questions where necessary. It will also allow me to transcribe the interview in order to give a comprehensive analysis (Koshy, 2009).

Interview Sample

The sample for the qualitative interviews were produced by using opportunity sampling. Jupp (2006) defines opportunity sampling as a technique that uses the knowledge of the researcher to identify a sample. For example, a researcher could contact people in the field they are investigating who they have past experiences with (Jupp, 2006). To identify what area of the development industry the individual was, there was a question within the quantitative survey. This allowed for key industries to be identified: agency or freelance, in-house or product developer and a student or hobbyist. Using opportunity sampling allowed me to obtain more valid and reliable information as I was able to target specific individuals in relation to what technology different developers use and their industry (Zohrabi, 2013). This is opposed to random sampling which may only give a limited insight.

How the Data was Collected

Three Industry professionals were interviewed, each with the intention of identifying the technology they utilise and why. The same set of basic questions were used for all participants. Minor adjustments were made for the participant in the product development industry (Table 3). Developers are taught to be technology agnostic. This is good practice within the industry, however, is not strictly followed. As most of the development industries I was investigating involved a high turnover of projects, prior to the interview questions a context of a hypothetical project would be presented to the participant to ensure later questions on what technology they would use can be answered. However, in-house and product developers do not have a high turn-over of projects and instead tend to work on a singular system. In this case, context of a hypothetical project is not necessary and discussion on the system they work on can take place.

Prior to the research, each participant was approached by email regarding the interview. A few example questions were presented as an example of what the interview would consist of, permission to record the interview was also asked at this point. Due to COVID-19 all interviews took place virtually via Microsoft Teams.

Questions and Expected Responses – Agency / Freelance

Context: For the purpose of these questions, I will provide a brief for the development of a project.

With this project in mind, you will be able to answer the questions below.

You or the company/agency you work for are about to undertake the development of a new website, this app will be a simple e-commerce platform and blog, that the owner will need to add content and products to. With this in mind:

Q. No	Questions	Reason for question	Expected Response
1	What is your current Job Title?	To understand the job role of the interviewee	E.g., Head of Tech
2	What development Industry do you work in?	To identify the development industry they are working in	Agency / Freelance
3	If you were approached to make a simple blog website for a client, how would you approach it in terms of the technology/stacks used?	To understand what their go-to technology stack is when approaching a project	MERN / LAMP / Jamstack
4	What made you decide to use this solution and how does it benefit you, the client and the project?	To understand why they made their decision and identify the advantages it brings them	We use it due to the flexibility, performance and development speed it provides us
5	Have you ever had problems using this approach? Drawbacks. Have you found Any limitations to this approach in projects?	To identify any issues or gaps they have experienced using this technology stack	When the project saw a large spike in traffic the application server would crash
6	Are you aware of any alternative technologies you could have used instead?	To see if the interviewee is aware of what else they could have utilised	Yes, I am aware there are serverless solutions such as the Jamstack
7	Why have you chosen to use this solution over alternative solutions?	To understand why they chose their	This provides a series of full featured solutions out of the box compared

		choice over other solutions out there	
8	Why do you think the alternative solution would not be suitable	To understand why the alternatives were dismissed and their disadvantages	It doesn’t allow the customisation we need
9	If legacy, Have you used any of these "as-a-service" products in projects undertaken? • Headless CMSs • Algolia • Auth0 • Snipcart • Stripe • FaunaDB • Firebase • Serverless functions • Other (Specify)	To understand, if the interviewee is using a legacy stack, if they still use APIs that make up the Jamstack	Yes, we use Algolia, Auth0 and Stripe
10	If yes, through the use of these services did you find there were advantages compared to building out your own backend?	To understand if through using these services there were any benefits compared to building out the same functionality yourself	Yes, advantages were the massive time saving and a reliable, secure implementation
11	If no, Have you considered how a ‘as-a-service’ solution may aid the product?	To understand if they have considered these APIs	It may be a more reliable solution
12	If Jamstack, what did they use previously? Before switching	To identify what the technology was they utilised before switching to the Jamstack and why they made the migration	We used to use the LAMP stack prior, but the performance and ability to scale with the Jamstack is why we switched
13	Have you heard of the Jamstack and what are your thoughts on it?	If they do not use the Jamstack, understand if they have heard of it	I have heard of the Jamstack but

		and considered it in their projects	unfortunately haven't been able to explore it

Table 2. Questions and Expected Responses – Agency / Freelance

Questions and Expected Responses – In house / Product development

Q. No	Questions	Reason for question	Expected Response
1	What is your current Job Title?	To understand the job role of the interviewee	E.g., CTO
2	What development Industry do you work in?	To identify the development industry they are working in	Product Development / In-House
3	What technology / stacks do you use to make your product?	To understand what their go-to technology stack is when approaching a project	MERN / LAMP / Jamstack
4	What made you decide to use this solution and how does it benefit you, the product and the company?	To understand why they made their decision and identify the advantages it brings them	We use it due to the flexibility, performance and development speed it provides us
5	Have you ever had problems using this approach? Drawbacks (costs, scalability, performance, security etc? How do you deal with these things?)	To identify any issues or gaps they have experienced using this technology stack	When the project saw a large spike in traffic the application server would crash
6	Are you aware of any alternative technologies you could have used instead?	To see if the interviewee is aware of what else they could have utilised	Yes, I am aware there are serverless solutions such as the Jamstack
7	Why have you chosen to use this solution over alternative solutions?	To understand why they chose their choice over other solutions out there	This provides a series of full featured solutions out of the box compared
8	If legacy, Have you used any of these "as-a-service" products in projects undertaken? • Headless CMSs	To understand, if the interviewee is using a legacy stack, if they	Yes, we use Algolia, Auth0 and Stripe

	• Algolia • Auth0 • Snipcart • Stripe • FaunaDB • Firebase • Serverless functions • Other (Specify)	still use APIs that make up the Jamstack	
9	If yes, through the use of these services did you find there were advantages compared to building out your own backend?	To understand if through using these services there were any benefits compared to building out the same functionality yourself	Yes, advantages were the massive time saving and allowing us to focus on other parts of the product
10	If no, have you considered how a 'as-a-service' solution may aid the product?	To understand if the interviewee has considered them and what their hesitancies are	It may be a more reliable solution
11	Have you heard of the Jamstack and what are your thoughts on it?	If they do not use the Jamstack, understand if they have heard of it and considered it in their projects	I have heard of the Jamstack but unfortunately haven't been able to explore it

Table 3. Questions and Expected Responses – In house / Product development

How the Data was Analysed

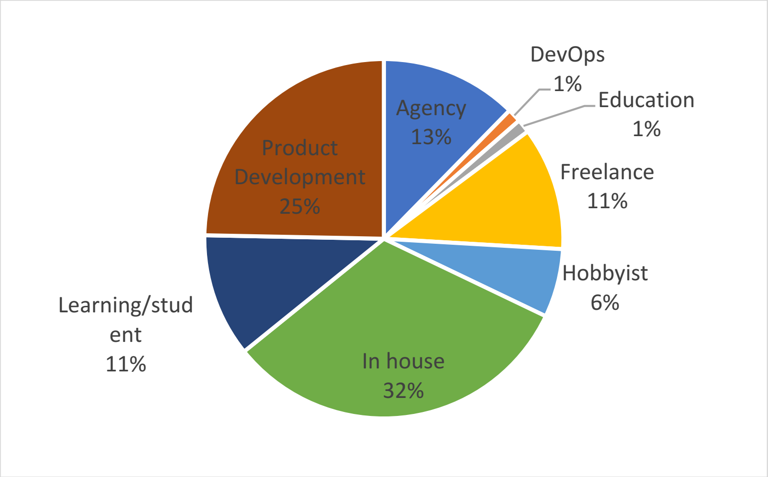
The research methods included a mixture of quantitative and qualitative data collection. For the quantitative closed questions, a descriptive statistics analysis will take place using Microsoft Excel. For the qualitative open questions, thematic analysis will be utilised. Thematic Analysis is a method for identifying, analysing, and reporting patterns within data, by organising and describing the data set in detail (Braun and Clarke, 2006). This process offered insight into patterns within the data and also highlighted some of the less obvious themes that were in participant responses (Braun, Gray and Clarke, 2017). Braun and Clarke (2006) describe a six-step process for thematic analysis; stating that it starts by familiarising yourself with the data, which would involve the transcription of verbal data (Appendix 4, 5 and 6). The next step is where coding then takes place (Appendix 3), this involves the production of initial codes from the data. These codes identify a feature of the data that appears interesting, this coded data differs from the themes, which are often broader. The next stage then involves searching for themes, reviewing the themes and finally defining them before producing the report. Theming is where the interpretative analysis of the data occurs in relation to what arguments about the phenomenon being examined are made.

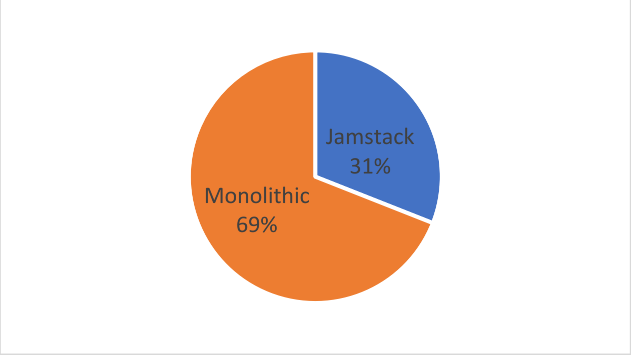
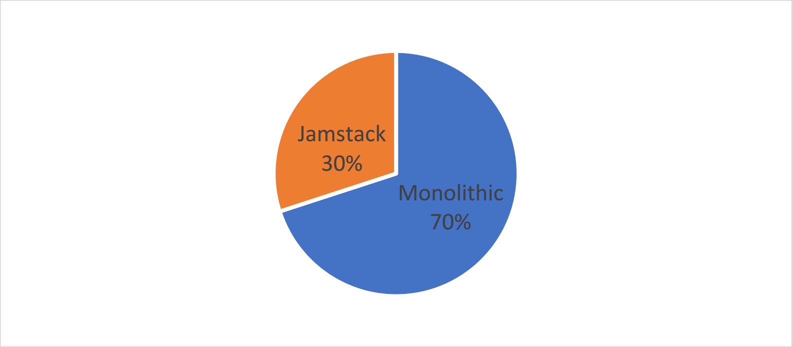
Results & Analysis:

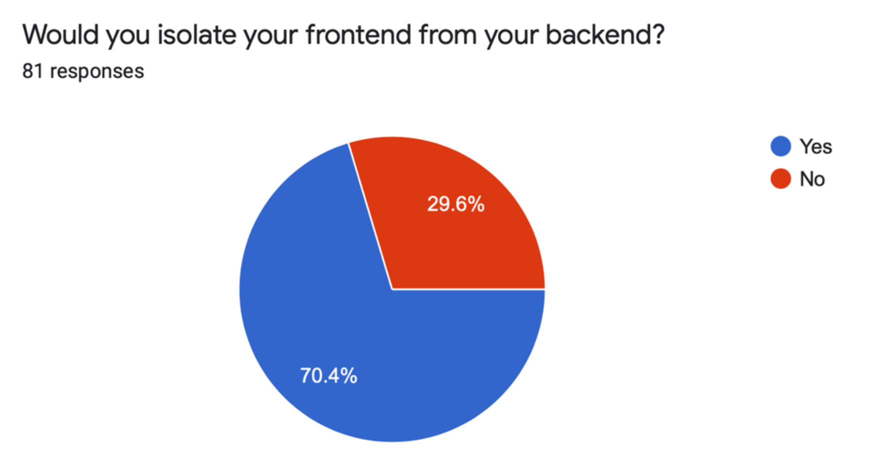
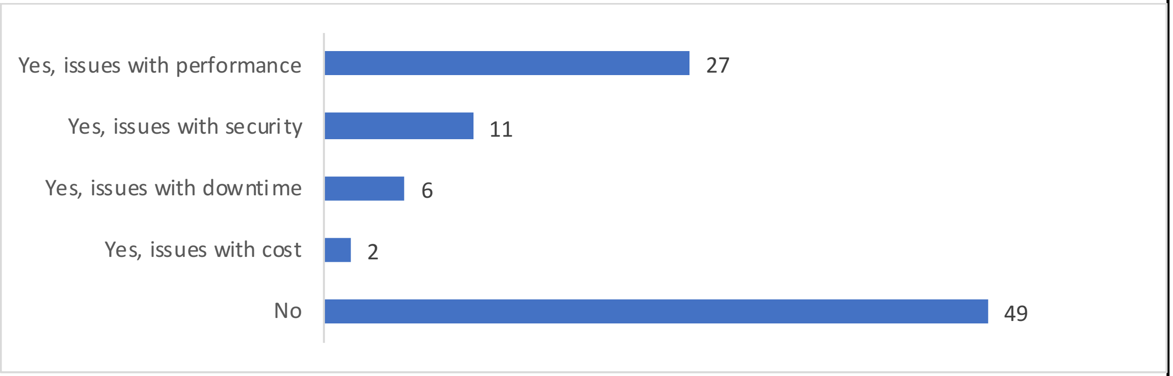
Introduction:

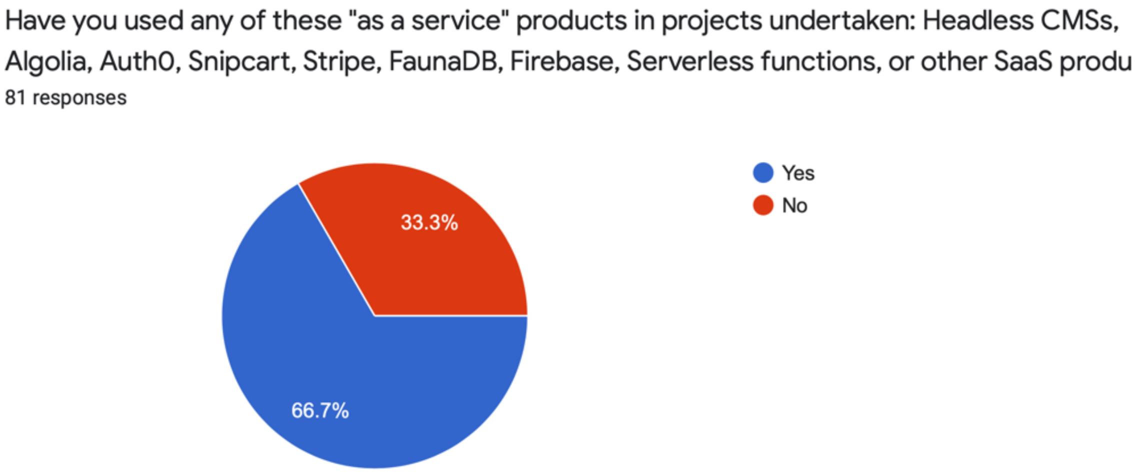
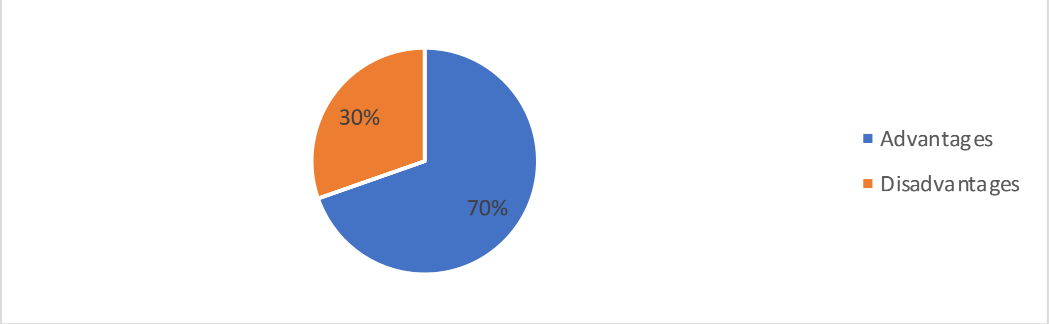
This chapter will present the findings from the primary research. The survey was published and sent out on 23/02/2021, the collection of data ended on 02/03/2021 with a total of 81 responses. A full list of the results from the survey can be seen in Appendix 3. Interviews took place in the last week of February, the first with Participant A being 22/02/2021, the second with Participant B on 24/02/2021 and the last with Participant C on 25/02/2021. Transcribed interviews can be found in Appendix 4, 5 and 6.

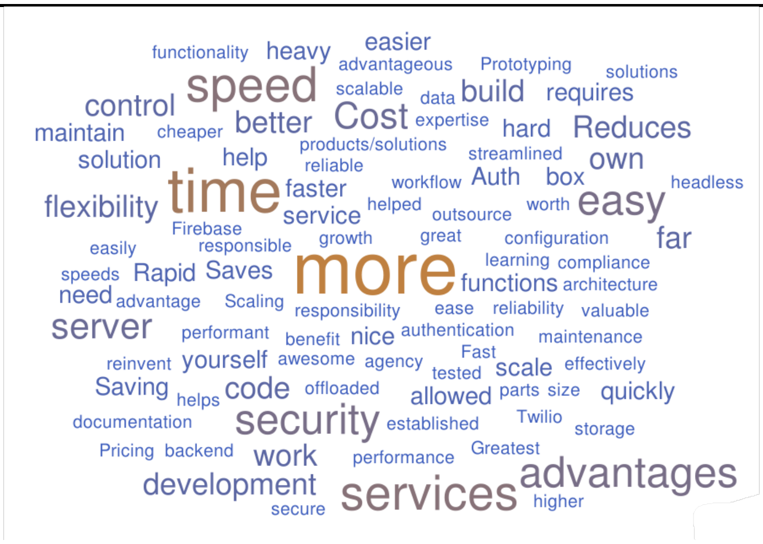
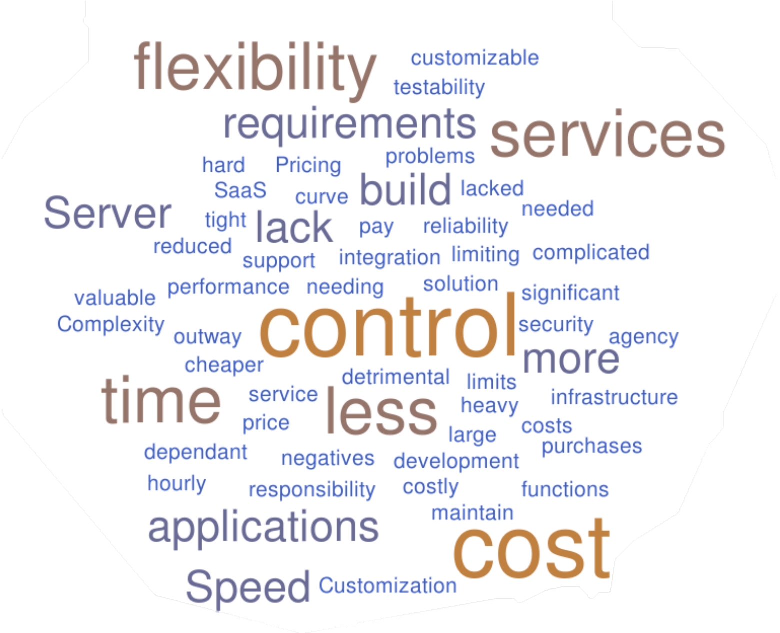
Survey Results

Q no.	Question
2	What sort of development work do you/the team/company you work at undertake?
	 Figure 4 - Question 2 Answers Figure 7 shows that 32% of the respondents worked in-house, 25% worked towards building out a product, 13% worked in agency while 11% were freelance and students/learning respectively.
5	What technology stack/technology would you choose to carry out the project?

	 Figure 5 - Question 5 Answers Figure 10 shows that the majority of respondents chose to use monolithic architectures for the project over that of the Jamstack. This may be due to 57% of respondents working in-house or in product development. Figure 11 shows that 70% of those that worked in these two industries chose to use a monolithic architecture. This may have been due to the age of an existing legacy system and the cost to change.  Figure 6 - Question 5 Answers Interpretation - What percentage of those who worked In-house or for a product chose to use a monolithic architecture or the Jamstack?
7	Would you isolate your frontend from your backend?

	 Figure 7 - Question 7 Answers Figure 12 shows that a large majority of respondents would isolate their frontend from their backend. This demonstrates that 70% of respondents would be able to easily migrate to the Jamstack.
8	Have you ever had issues surrounding your choice of stack? Such as website performance, security or down time?
	 Figure 8 - Question 8 Answers Figure 13 shows that half of respondents had not experienced any issues with their stack, although the other half of respondents had. In fact, 70.96% of those who stated they would choose to use a monolithic architecture also admitted to experiencing issues surrounding it, whereas only 16% of those who stated they would use the Jamstack admitted to experiencing issues. This may be due to extra levels in a monolithic architecture compared to that of the Jamstack with more areas with an opportunity for issues.
9	Have you used any of these "as-a-service" products in projects undertaken: Headless CMSs, Algolia, Auth0, Snipcart, Stripe, FaunaDB, Firebase, Serverless functions, or other SaaS products

	 Have you used any of these "as a service" products in projects undertaken: Headless CMSs, Algolia, Auth0, Snipcart, Stripe, FaunaDB, Firebase, Serverless functions, or other SaaS produ 81 responses Figure 9 - Question 9 Answers Figure 14 shows that 66% of respondents had used “as-a-service” products, the APIs that are commonly used with the Jamstack. This may be the case because despite respondents stating they may use a monolithic architecture, “as-a-service” products may also be used alongside them.
11	Through the use of these services did you find there were any advantages or disadvantages compared to building out the functionality in your own backend? If so, what were they?
	 Figure 10 - Question 11 Answers In figure 16 we can see the majority of respondents expressed advantages. Figure 17 displays a word cloud of some of the most common words to appear in these advantages. Figure 18 displays the most common words in the disadvantages. Advantages

	 Figure 11 - Question 11 - Advantages in Word Cloud Disadvantages  Figure 12 - Question 11 - Disadvantages in Word Cloud
12	Do you feel using these ‘backend-as-a-service’ products together could eliminate the need to develop your own backend?

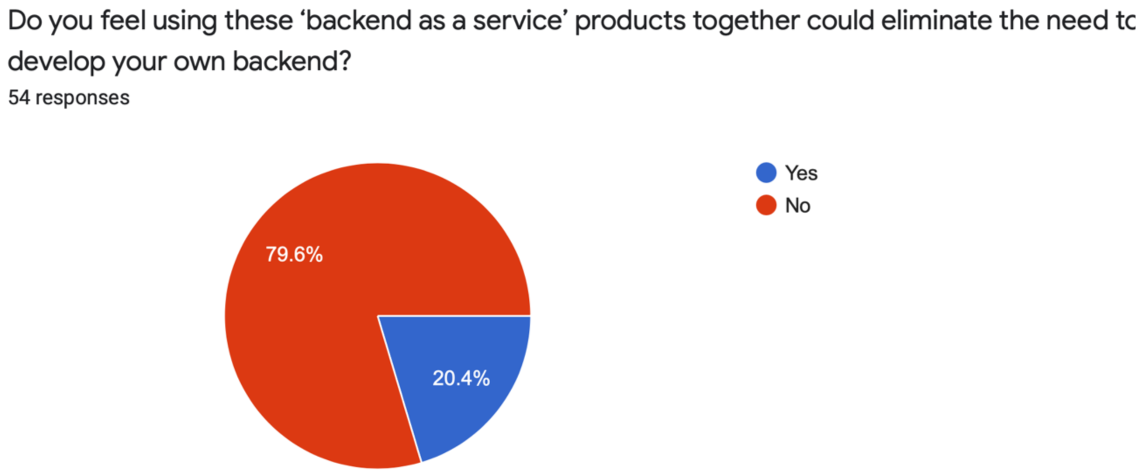


Figure 13 - Question 12 Answers

Figure 19 shows that nearly 80% of respondents feel that 'as-a-service' products will not be able to replace developing a backend. Table 6, for the question below, shows thematic analysis with the main themes that appeared in the justification.

14 Is the JAM stack something you are familiar with and have experience using in projects?

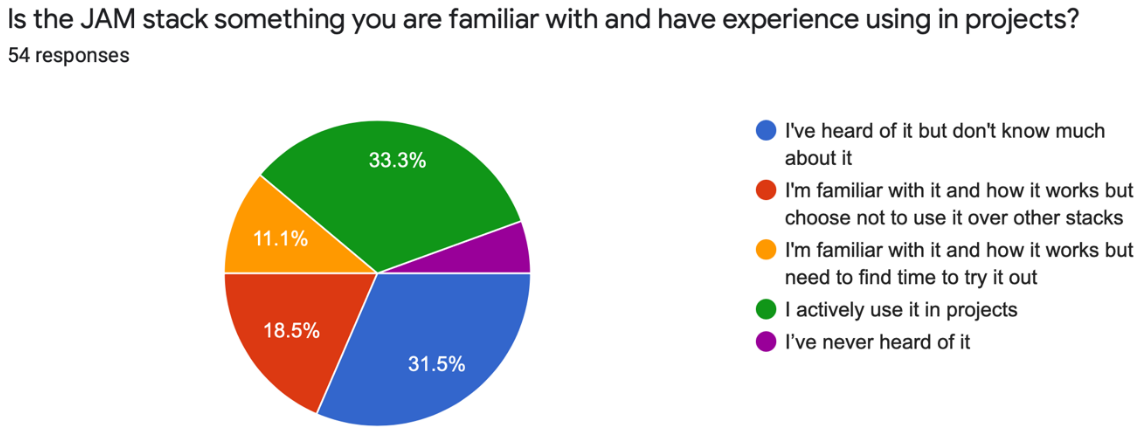
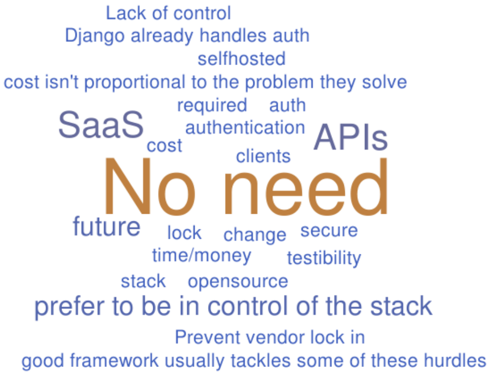
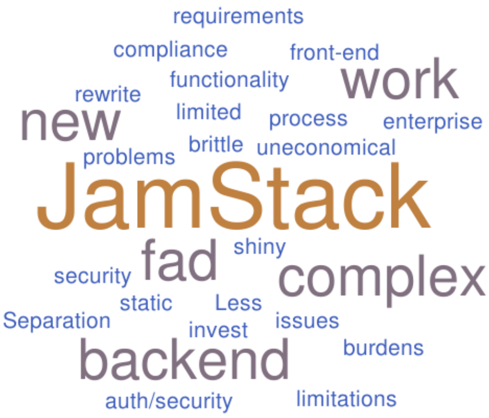


Figure 14 - Question 14 Answers

In figure 21 you can see that 33% of respondents stated they actively use the Jamstack. 71% of these were agency developers. This may be because for agencies to stay relevant and competitive they need to be at the forefront of technology and utilising the latest trends, which the Jamstack currently is.

15	Why haven't you/don't you use the "as-a-service" products?
	 Lack of control Django already handles auth selfhosted cost isn't proportional to the problem they solve required auth SaaS authentication APIs cost clients No need future lock change secure time/money testibility stack opensource prefer to be in control of the stack Prevent vendor lock in good framework usually tackles some of these hurdles Figure 15 - Question 15 Word Cloud In figure 22 you can see a word cloud of all the most common words appearing when respondents justified why they do not use 'as-a-service' products.
17	Why do you choose not to use the Jamstack over other stacks/solutions?
	 requirements compliance front-end functionality work rewrite new limited process enterprise brittle problems uneconomical JamStack shiny security fad complex static Less Separation invest issues backend burdens auth/security limitations Figure 16 - Question 17 Word Cloud In figure 24 you can see a word cloud of all the most common words appearing when respondents justified why they do not use the Jamstack.
18	Why do you use the Jamstack over other stacks/solutions?

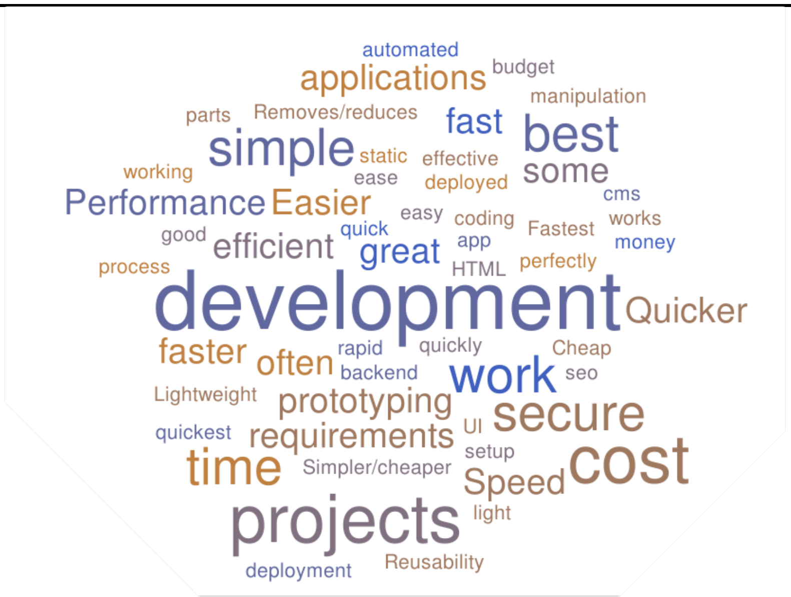


Figure 17 - Question 18 Word Cloud

In figure 25 you can see a word cloud of all the most common words appearing when respondents justified why they do use the Jamstack.

Thematic analysis took place on open questions.

Table 4 shows thematic analysis with the main themes, accompanied with example quotes, on why each development industry chose their architecture. The main themes emerging being ‘performance’, ‘developer experience’, ‘cost’ and ‘support’.

		Themes	
Product Development	Monolithic	Performance	“Stability” “No need for client rendering and better SEO. Try using a heavy react app on an old android.” “Speed.” “Django provides really nice serialization as well as pagination... Node is great if you have a ton of IO and users visiting the site.” “Go is fast”
		Developer Experience	“familiarity” “It's really simple to setup, develop, and deploy. It would also let me quickly develop out the MVP for the client.” “Simplicity and custom functionality is easier to architect without fighting an opinionated system, in the latter case, flexibility and speed is key”
		Support	“Large community support” “well documented”
		Cost	“economy (cheaper devs)”
	JamStack	Performance	“Time to deliver” “Fast and strong” “storefront is backend agnostic.” “It’s all bundled and easy, and great for simple cases like this”
		Developer Experience	“Simple to build in, pre built templates to work from and great hosting/tooling.” “easy”
		Support	“large communities of support” “Modern frameworks with huge communities and support.” “great docs and community”

house	Monolithic	Performance	“It's great to get a fast ramp up of productivity, and allows scaling as the projects grow” “Great Performance, mostly a no-brainer with ORMs” “.NET scales best for this type of application. Node can be used for middleware or small scripts like login, where it really excels” “Django allows for rapid development and I don't want a complicated frontend so I use jquery.”
		Developer Experience	“fast and simple development with those frameworks” “There's an easier handoff of responsibilities between api developer, and UI developers Android/ iOS/ web” “Wide range of libraries available. Easy maintenance. Clean code.”
	Jamstack	Performance	“We know react quite well, it integrates well with many content sources and allows you to balance static pages with dynamic content.” “easy to support. WooCommerce is a popular and well tested and used... There's no point reinventing the wheel.” “allows serverside rendering, static site generation, client side fetching with static pages”
		Cost	“Lower hosting costs” “There are existing solutions for basic e-commerce and they work really well. No need to reinvent the wheel. It would be extremely uneconomical to waste developer/sysadmin hours building and maintaining a new platform when we can just throw $29 a month at the problem.”
		Developer Experience	“A headless CMS allows for others to add blog posts… quick to develop.” “The ability to decouple the frontend from the database. Allow non-technical people to update content.”
Freelance	Monolithic	Performance	“Plain html - seo friendly” “Fast, safe” “Laravel for quick database connection”
		Developer Experience	“familiarity, stability”
		Support	“Big ecosystem/community and great documentation. Battle tested”

	Jamstac	Performance	“These softwares offer a very fast way to build a site from scratch, and is also maintained via git hub for the most part. Mongo also has a console dashboard where data can be added extremely quickly.” “NextJS provide extremely fast website and Strapi is a really configurable cms”
Agency	Monolith	Performance	“Reliable, lots of documentation and can be used in many scenarios. Can incorporate all features required in the brief.” “Effiency” “Docker makes it easier to transfer between servers and manage the deployment/development.” “It's quick to set up and develop for”
	Jam	Performance	“To isolate frontend from backend, and to generate pages to improve their SEO.”

Table 4. Question 6 - Provide a reason for your choice of technology. Thematic Analysis

Table 5 shows thematic analysis with the main themes that appeared in both, the advantages and disadvantages of using services compared to building out your own backend. The main themes emerging being ‘performance’, ‘developer experience’, and ‘cost’.

	Themes	
39 Advantages	Performance	“Fast, reliable, easy“ “they scale way better” “Yes, less to build, maintain” “Rapid Prototyping” “More scalable for rapid growth in usage.” “Makes for a faster development speed and provides better security and means your as a company are not directly responsible for data protection.” “reduces the barrier of entry for junior developers coming into the project. Plus some things (notably payment processing) carry a heavy compliance burden and it's nice to be able to effectively outsource those things.” “Time to market” “Saves time. Reduces complexity in your own code. Reduces the amount of possible failure points in your application” “Advantages: more complete solution, cheaper than dev time, less responsibility, expertise, better support.”
	Developer experience	“Easy setup” “Yes! Much faster to implement things like push notifications, authentication, etc.” “The server less function were more restrictive and harder to run/ deploy with code than standard server solutions.” “There is still a bit of fine tuning using these services, but it definitely speeds projects up when you don’t have to build everything from scratch.” “Easy to scale” “As a front end developer, services like these help a lot by giving easy to understand functions and helpers, as well as most of them coming with great boilerplates and documentation” “Works easily and quickly” “They can significantly speed up development” “Sometimes it requires a little bit more thinking but in the end the result is far more worth it.” “Its a solution out of the box” “Many advantages as it allowed more time to develop code that made other parts of my project more unique.”

	Cost	“Cost saving” “Price and speed” “Saving Time vs Cost” “It saves time”
17 Disadvantages	Performance	“Depends on the Use case. It's easy to build something with Firebase but if there are specific requirements it gets hard. Stripe & Auth0 are great, not limiting requirements and easy to use.” “Complexity of infrastructure, cost, tight coupling to third party” “Server less functions can have a large 'turn on' time, which is especially detrimental for time-sensitive applications, like APIs” “Massive disadvantages. You are simply spreading your problems across multiple interfaces.” Disadvantages: can be costly, may be more complicated than you need, dependant on that service.” “Greatest disadvantage is probably control and flexibility. As we were developing custom products/solutions we found that backend services helped with a lot of the heavy lifting at the cost of reduced control and flexibility in case we needed to do some kind of special integration with one of our other applications.”
	Developer experience	“non customizable snipcart” “Fire base is awesome, however i felt like I lacked control overall - I’m also not so keen on pay as you go services” “Disadvantage is lack of flexibility.” “Customization.” “Less control and lack of testability”
	Cost	“no, just the price / limits” “Pricing can be higher with SaaS but you get the ease of never needing to maintain a server” “Speed and reliability is the main key advantage that typically outweighs doing it ourselves. Disadvantage is the hourly costs that grow.”

Table 5. Question 11 - Advantages or Disadvantages of using services compared to building out own backend. Thematic Analysis

Table 6 is thematic analysis on whether participants felt using these ‘backend-as-a-service’ products together could eliminate the need to develop your own backend?

First order indicator (example quotes from interview)	Second order indicator (sub-themes)	Main Themes
“They can substitute certain parts of a backend but you still need code to bring everything together” “There will also be functionality not covered by these services and cases where they can not be afforded.” “Often we need bespoke elements, using off-the-shelf solutions typically end up requiring work arounds to make them work as required.” “Because it will take a long time to reach the level of customization in writing the backend ourselves..” “I like to have a flexible app” “If the stakeholders require security or massive amounts of data transfer, I’ll use a backend” “loss of control” “if there's enough money, backend services could be implemented for faster mvp” “These days serverless functions can do most things.”	Performance Flexibility Control	Technical capability
“90% of backend is CRUD, and these services tend to make CRUD really expensive to run, and make building the remaining 10% more complicated.” “Cheaper and faster to use 3rd party than to develop a new system” “For a small product, maybe, but not for a larger product as you lose so much fine control“ “no need to reinvent the wheel. If its a cookie cutter project, there is a template for the back end” “Most my backends are similar and I copy and paste between them. Only need to develop your own backend in niche situations.”	Speed Simplicity/complication	Developer Experience
“cases where they can not be afforded”	Expensive	Costs

“90% of backend is CRUD, and these services tend to make CRUD really expensive to run” “Cheaper and faster to use 3rd party than to develop a new system” “Costs add up”	Cheaper	
“Business domain logic is hard to outsource” “There are always going to be edge cases that requires custom code.” “There are always unique issues to your problem domain that are not worth solving for the people who write the "as-a-service" products.” “These services give a good starting off point, but often need retooling for particular use cases. “	Company Value Business Logic	Reliability

Table 6. Question 13 - Do you feel using these ‘backend-as-a-service’ products together could eliminate the need to develop your own backend? Thematic Analysis

Interview Results

Participant A – A student but working part time as a Full Stack Developer at an Agency

Participant B – CTO of a SaaS product company

Participant C – The head of technology at an agency

Thematic Analysis

First order indicator (example quotes from interview	Second order indicator (sub-themes)	Main Themes
"Laravel. We can make that as custom as we need it to be so whatever we need it - you know, for the project" "WordPress tends to be a bit annoying when trying to expand anything." "I'm quite a big proponent of using like services, Like Auth0 you know, these things that can take something where you want a secure implementation, you know, obviously around Auth, flows." "Interestingly we didn't go that way when we were recently building out an API, and the main reason was just, I'm a bit of a Control freak, I guess. I'd rather have the thing that might break under our control than you know someone else drops the ball and suddenly our product doesn't work"	Customisation Scale Secure Control Downtime	Technical Capability

“Anything that’s kind of out of the critical path from a user's perspective and we can tolerate outage. We would outsource if we can.” “Anything that’s kind of out of the critical path from a user's perspective and we can tolerate outage. We would outsource if we can.” “a mobile app and we actually used Pass… They were a database as-a-service kind of rest API thing and we used them just purely for scale 'cause we knew in that case there was no way we would be able to affordably scale for the kind of map downloads this thing was going to get.“ “I think the biggest benefits we're seeing is performance. So compared to sort of your older technologies the performance gains that you get from using something like React its… its massive. “ “So with your older technologies, it's very difficult to to optimize your websites for these web core vitals.”		
“WordPress is obviously quite a big thing in the industry, a lot of clients will know how to use it straight off the bat” “made it easier to hire people with because we were using a popular framework” “very small with a quite small product team; and that means that we need people that are quite multi, multi- experience with multiple languages, or at least the ability to pick up and drop.” “diversity of the stack is a bit of a problem with hiring” “I do wonder if the complexity that we have can be simplified and somehow just to make it easier to onboard new developers.”	Hiring Implementation Context Shifting Training	Developer Experience

"You know things like payments on Stripe would be an obvious thing we'd look to outsource, because we don't really want to be dealing with PCIDs etc" "we always look to see if there is a way to avoid writing ourselves" "we don't really go out and look for, look for what technology to use on an individual basis. We have sort of our tech stacks that we specialize in, and we sort of bring them out depending on the project." "I can't imagine anyone in one years time still doing. You know your static PHP website." "really speeding up our development times, just because of sort of the component based way that React, React lets you work in. You can reuse a lot of the stuff that you've built in other projects really easily. Sort of in a new project that you might be starting." "the biggest problems we've got is sort Of getting everyone up to speed with it, because a lot of time with this new… technologies especially for people that historically haven't really worked with something like that, it's quite a big step to sort of get your head around how everything works... I think that's sort of being our biggest drawback from from using this. It's just getting everyone up to speed and to be able to use this effectively"		
"cost, because it's expensive [Algolia]… There's a site that we worked on. It has about what 40,000 different products?... Erm, and yeah we ran into a bit of a cost issue with that" "Firebase I've used for one personal project but it's just too expensive to be viable on anything big." "When we started the product a lot of that didn't exist... You're almost stuck with your choices up to a point right 'cause the cost	Prices Time Cost to change	Costs

of making a significant change is certainly significant, so there would need to be, you know, a very compelling reason to make that transition, and sometimes it definitely is better that way. “the time investment, right? You know, especially as a small team, you want to be focusing on the problems that matter and where you can add value in in your own space.” “Firebase I've used for one personal project but it's just too expensive to be viable on anything big.” “Because it is just very expensive and in a way also very risky to move to these New technologies because you're sort of going to bump into issues that you've never come across before, and those at sort of add a new layer of frustrations within the team as well”		

Table 7. Interviews Thematic Analysis

Discussion:

Developer Experience

Within the survey 70% of participants who selected they work in-house or towards a product also stated they would use a monolithic architecture. However, 70.96% of those admitted to experiencing issues with it, ranging from downtime, performance or security. Biilmann and Hawksworth (2019) identify this is often due to the size and complexity of monolithic architectures, which require many people with a range of skills to operate it. This statement was supported by Participant B, who uses a monolithic architecture, they identified complexity as an issue. Stating, "the diversity of the stack is a bit of a problem with hiring… We need people that [have] experience with multiple languages… doing a context shift from doing PHP, Symfony… Golang and… node JS… sometimes I do wonder if the complexity that we have can be simplified". This is further supported by Drasner (2020) in Smashing Magazine's migration from a monolithic architecture to the Jamstack, stating that one of the main motivations was to unify multiple systems, instead of maintaining them separately with different stacks. He states that having to context shift between the different technology became a difficult maintenance problem (Drasner, 2020). Participant C, an agency technology-lead who uses the Jamstack, was able to support this improved developer experience stating, "it is really speeding up our development times… you're really saving yourself so much time compared to where you used to almost have to have to re-develop every PHP website". This supported a theme that appeared through the survey: improved development times using 'as-a-service' APIs, with one respondent stating, "it allowed more time to develop code that made other parts of my project more unique". However, Freestone (2020) looks at it from a different perspective, the improved developer experience is simply due to enforced minimalism by cutting out entire tiers from the stack which leaves little choice other than to distribute responsibility of functionality to third party APIs.

Technical Capability and Reliability

When asked if participants of the survey felt 'APIs-as-a-service' could replace the need to develop a backend 79.6% stated they felt they could not. However, when asked what the advantages or disadvantages were compared to building out the functionality in the backend themselves, 70% of the responses were advantages compared to 30% which were disadvantages. I believe the mixed responses are due to the different benefits and drawbacks each service can provide. For example, Participant A stated that, "it depends on the [API] really, because like if it's something like Firebase or Algolia, it just gets a lot more pricey but with stuff like Stripe It's just a part of the website and then it's not really a [problem]”. Participant B shares this opinion, stating “We've made use of Lambda in very small specific instances… we always look to see if there is a way to avoid writing ourselves”, but for them it depends on the problem they are trying to solve and “how much I care about our data”. For example, stating “I'm a bit of a Control freak, I guess. I'd rather have the thing that might break under our control than, you know, someone else drops the ball and suddenly our product doesn't work”. The reliability, control, cost and uptime of the service is what held them back from utilising certain APIs. This was also recognised by Freestone (2020) stating that these are integral features to the success and functionality of their business and would be excluded by their decision to fully migrate to the Jamstack. Instead, Participant B states they “outsource metrics and analytics… if those things go out it's inconvenient but it doesn't stop our users and our customers getting their value from the product”. However, deciding not to use APIs to avoid downtime and choosing to use a monolithic architecture does not mean there is no chance of experiencing it. Hosting providers for servers are also capable of experiencing issues that can cause down time, essentially making the risk of downtime for a Jamstack product and a monolithic architecture product, due to reliance on third parties, the same. However, Participant B justifies hosting on AWS with “when they break, most of the Internet breaks so you always have this. It's not good excuse, but you know it's this ‘You know who else could we have gone with, everyone's using it’ “. On the other hand, Drasner (2020) stated the motivation behind Smashing Magazine's migration to the Jamstack was due to less downtime. He explained that previously when an article went viral and experienced lots of traffic the website would often experience outages due to the server-side rendering. Participant B was able to support this. They explained, despite using a monolithic architecture for their product, when they worked in an agency, they opted to use the Jamstack and a backend-as-a-service API. They stated: “we built a project years ago for Now That's What I Call Music, and it was a mobile app and we actually used Pass… They were a database as a service kind of rest API thing and we used them just purely for scale because we knew in that case there was no way we would be able to affordably scale for the kind of map downloads this thing was going to get”. They were able to quickly produce a highly scalable application using the Jamstack. This supports Billmann and Hawksworth (2019) who state this same scalability was only possible using a monolithic architecture by large, enterprise companies due to their high costs, complexity and the need for a large, specialised team. Uptime and reliability are crucial factors for in-house and developers building a product. However, for agency developers the technical and performance advantages it provides seems to outweigh the potential uncertain reliability of a third-party API.

71% of agency developers stated they actively use the Jamstack in projects, and 66.7% of all respondents of the survey admitted to using 'APIs-as-a-service', whether used in the Jamstack or not. Participant C, who does use the Jamstack, stated he uses it over monolithic alternatives because "compared to… older technologies the performance gains that you get from using something like React is massive". They state they historically used monolithic architectures for many projects but are now "trying to move away" to the Jamstack due to the performance, which "would [not] be viable if you do sort of PHP or any other sort of old school language". They were able to achieve high scores in Google's Core Web Vitals that are metrics relating to speed, responsiveness and visual stability. They explain "with your older technologies, it's very difficult to optimize your websites for these web core vitals". This is supported by Manandhar (2020) who agrees the Jamstack is an excellent decision for the development benefits it brings. Participant A supported this with their use of 'APIs-as-a-service', in order to improve performance and allow them to scale. They were able to utilise Algolia to allow search in an e-commerce project, "the API was able to scale vastly to 40,000 products". Participant B also supported this with their Now That's What I Call Music project discussed previously. However, a reoccurring theme that appeared in the survey when participants were asked whether 'backend-as-a-service' APIs could replace the backend completely, was its technical capability. Specifically, the need to produce custom functionality. One respondent stated, "Often we need bespoke elements, using off-the-shelf solutions typically end up requiring work arounds to make them work as required". Participant A supported this theme and stated, "it's a bit of a nightmare" adding custom code on top of APIs. Freestone (2020) recognised this and stated without a server, logic has to be client side and may create performance issues.

Migration

12% of survey participants admitted to not knowing what the Jamstack was. 31% said they would choose to use the Jamstack to carry out the project while 69% opted for a monolithic solution. Participant B, who uses a monolithic architecture, explains that "When we started the product a lot of that [technology] didn't exist. So, the problem… [of] running a company or product over a long period of time is you're almost stuck with your choices up to a point, right, because the cost of making a significant change is certainly significant". From this we can infer it would be a similar situation for in-house developers. In both industries they tend to work on singular systems for an extended period of time. Participant C recognises this, stating "that is what tends to be the difference between agency developers and sort of in-house developers… a lot of the time it's a historical sort of project that probably has been built 10 years ago”. Agencies on the other hand tend to have a higher turnover of projects, allowing them to utilise the newest technologies with every project.

71% of agency developers stated they actively use the Jamstack in projects. Participant C stated that at their agency they set out a plan to fully adopt and migrate projects to the Jamstack, and monolithic architectures were no longer “a part of that plan”. Going on to say that they are “trying to in the future replace anything that we used to do in Laravel with Strapi”. They justified this by saying “[we are] future proofing ourselves” and that “if you want to be a forward-thinking agency” or “be among the best” then agencies “need to be at the forefront of technologies and you can’t be struggling behind… or you’ll end up in a position where I don’t think you’re going to be able to catchup”. Meyer (2020, [online]) agrees with this, who urged developers to ‘Get Static’ and move away from monolithic architectures so they can better serve traffic demands and be more inclusive. Participant B also agrees and recommends the Jamstack, but only to those “starting out” and building out “a new product”. He states that if this adoption is “going to give you that leg up, that lets you get to market and show your ideas and prove your ideas first, then absolutely”. However, they do not agree with Participant C’s comment about an adoption of the Jamstack being necessary in order to future proof themselves, they recommend instead migrating back to a monolithic architecture. They state, “as you start to see that there's a fit there, I’d say that’ll be when you'd be looking to ideally pull that internally as quickly as possible” due to the reliability of third-party APIs and “how much I care about our data”.

29% of agency developers do not use the Jamstack. Participant C identified why this was the case, “it is just very expensive and… very risky to move to these new technologies because you're sort of going to bump into issues that you've never come across before” going on to explain a migration will cause “stress” and “frustrations within the team”. Participant B, despite recommending a migration from the Jamstack back to a monolithic solution, agrees with this stating a migration would just mean a “rewrite [of] all the bugs that you fixed”. They also describe the reluctancy for product development teams, or in-house teams to migrate due to time investment. They state, “especially as a small team, you want to be focusing on the problems that matter and where you can add value”. This applies to any business and may relate to why 29% of agency developers were not utilising the Jamstack, as their clients cannot commit to a migration. Participant A agrees, stating “[we] work on a lot of company-based software”, so they do not always have the opportunity to work on new projects with the latest technology. Instead, working with existing legacy systems businesses depend on. Due to this, participant C states that “Laravel still has its place”. Biilmann and Hawksworth (2019) recognise the issue businesses and development teams have, stating rebuilding web infrastructure will pull resources away from critical growth. Participant B agrees with this, describing “if someone has to learn technology that they haven't used before, a lot goes into learning... plus, they're not very effective when they start working with it”. Participant B mentioned they are a small team and with that, time investment on what they work on was a very important factor. With this in mind a migration to the Jamstack may be suitable and improve this problem, with features being quick to add, increasing deployment frequency and allowing teams to iterate faster (Myers, 2018a). Chris Coyier speaking on Jamstack TV (2018) supports this explaining that monolithic architectures require double the team, and where certain functionality would require backend engineers, with the Jamstack frontend engineers could build this capability instead.

Time Management and Planning:

Prior to undertaking this research, project planning took place to track progress and set deadlines. The project was split into two halves and two Gantt charts were produced. The first Gantt chart (Appendix 7) focused on the secondary research in the literature review and the second (Appendix 8) focused on the primary research and discussion.

Focusing on Appendix 7, the project began with the 'Research Question, Aims and Objectives' phase. Tasks in this phase were crucial in order for any research to begin and therefore of high importance. Due to this, two-weeks were allocated to produce a research question and another two-weeks for aims and objectives. However, coming up with an adequate question took longer than anticipated due to the screening process of questions. Beforehand, the time it may take was identified as a high risk, consequently, two weeks with a third week buffer was planned and proved effective, allowing enough time. This was similar within the formulation of the aims and objectives tasks, an extra week was factored in as a buffer due to their importance and high risk if not completed. As these are the foundation of the dissertation, it was imperative for time management to be thoroughly thought through, giving clear steps for the continuation of the dissertation.

The research project began during the COVID-19 pandemic so it was important to have contingency planning and buffers factored into the dissertation. No external risks, such as the pandemic, were identified during the first semester and the tasks could all be done without human interaction such as the secondary research, which meant the research was unaffected. However, in the second half of the project (Appendix 8), primary research and face-to-face interviews were originally planned to take place in week seven. Unfortunately, due to national social-distancing restrictions this was unable to take place. However, due to previous lock down restrictions, there was a contingency plan to use video conferencing tools. This proved effective and despite the original plan not being followed, the video conferencing resulted in a faster process with all interviews taking less than a week as there were no travel times. These interviews were flagged to be high risk, due to interviewees perhaps needing to cancel or reschedule, to cater for this, a week buffer was allocated making the interview period two weeks. Luckily this was not the case.

Conducting this dissertation alongside my final major project has impacted both pieces of work positively. My final major project was an entire application built using the Jamstack. It allowed me to apply and put into practise what I had researched. It also allowed me to visualise the technical benefits and drawbacks identified throughout this dissertation and understand why people may or may not choose to utilise the Jamstack.

Conclusion:

Throughout this dissertation, the main aim was to understand whether the Jamstack was a realistic option in industry and viable over currently used technology. This then led on to establish if the Jamstack would ever be able to replace the need to develop a backend. This aim was achieved and every weekly plan and objective were fulfilled. For example, the objective of exploring the technical capability of the Jamstack and monolithic architectures. As this was achieved, I gained understanding on the extent to which these development stacks could be stretched, with particular focus on comparing them. This then allowed for quantitative research to be undertaken with a sample of developers from different development industries. Collecting quantitative data through surveys allowed my objective of identifying any correlations between the development industry and the choice of technology the developers chose to fulfil a project to be met. To further support my research, qualitative data was collected to meet the final objective of explaining correlations found within the survey. Finally, it allowed the of correlations and patterns to overall analyse whether the Jamstack is capable of being a viable migration.

From the research, it was found that the development industry a developer finds themselves in has an influence on the technology they use. The majority of agency developers utilise the Jamstack while in-house and product developers tend to stick with monolithic architectures. This was found to be due to their reluctancy to rely on an API to provide business value, and the concern to produce custom functionality without a backend. However, this did not completely exclude the use of the Jamstack from these development industries. To further my knowledge on the web stacks used in these industries, semi-structured interviews were invaluable. Despite developers stating they use monolithic architectures for their core product, APIs-as-a-service were still utilised alongside servers and monolithic architectures for less crucial functions such as search, error handling, processing and payment. This highlights the key benefits the Jamstack provides and its prominence across the industry.

From the surveys and interviews conducted, it was found agencies who used the Jamstack were able to produce products in a short space of time, something which would have otherwise taken a large team and high costs. Instead, agencies are able to produce more performant, cheaper, more secure and scalable projects with a handful of people, allowing them to focus on what makes the project unique.

With all research in mind, I feel the Jamstack benefits every aspect of the wider development community by reducing the barriers to entry for producing enterprise level applications.

I believe the Jamstack will slowly start to take over the entire development industry, like it has the agency industry. The main concern with full adoption was the reliance on the third-party APIs, and any issues they have would mean down time for the application. However, the immense advantages the Jamstack provides will make it a hard option to overlook for new start-ups. Concerns regarding down time due to reliance on APIs is not just a problem with the Jamstack, in fact there is more opportunity for downtime with a monolithic architecture. This is due to high traffic that application servers cannot handle and security breaches, these factors are not a problem with the Jamstack. Furthermore, downtime as a result of the hosting provider for the server is not uncommon, so the risk for both the Jamstack and monolithic architecture's reliance on third parties is the same. Although, I believe the concern for the Jamstack services and not of that of monolithic hosting services such as Amazon Web Services (AWS), comes from the maturity of these platforms in comparison to how new the 'backend-as-a-service' APIs are. As discussed, I feel the advantages the Jamstack provide will be hard to overlook for new start-ups and in-house teams. This adoption will only see the likes of the 'backend-as-a-service' products establish, growing into the likes of what monolithic hosting services are now such as AWS. With this, I do believe the Jamstack will replace the need to develop a backend for many future projects.

One of the barriers to utilising the Jamstack identified in all development industries was working on legacy system projects. From this research project, future work is able to take place to explore how this barrier can be broken down by utilising the Jamstack to transform legacy systems into modern solutions.

Reference List:

Barnham, C. (2015) Quantitative and qualitative research: Perceptual foundations. *International Journal of Market Research*, 57(6), pp. 837–854.

Bell, J. and Waters, S. (2014) *Doing Your Research Project: A Guide For First-Time Researchers*. 6th edn. London: McGraw-Hill Education.

Belshe, M. (2010) More Bandwidth Doesn't Matter (Much). *Mike Belshe.* 24 May. Available at: *https://www.belshe.com/2010/05/24/more-bandwidth-doesnt-matter-much/* [Accessed 21st December 2020].

Biilmann, M. and Hawksworth, P. (2019) *Modern Web Development on the JAMstack*. O'Reilly Media, Inc. Available at: *https://www.netlify.com/pdf/oreilly-modern-web-development-on-the-jamstack.pdf* [Accessed 23rd November 2020].

Braun, V. and Clarke, V. (2006) Qualitative Research in Psychology. *Using thematic analysis in psychology.* 3, p. 101.

Braun, V., Gray, D. and Clarke, V. (2017) *Collecting qualitative data: a practical guide to textual, media and virtual techniques*. Cambridge: Cambridge University Press.

@Paulcalvano (2019) 29 August. Available at: *https://twitter.com/paulcalvano/status/1167143641124675586* [Accessed 20th December 2020].

The Changelog (2020) Gatsby's long road to incremental builds with Kyle Matthews [Podcast] 22 April. Available at: Available at: *https://changelog.com/podcast/393* [Accessed 3rd April 2021].

Chen, D. (2019) What exactly is JAMstack?. Available at: *https://medium.com/paypal-engineering/what-exactly-is-jamstack-a9c05f513823* [Accessed 25th February 2021].

Coyier, C. (2019) I've had some VC's asking me about JAMstack. *The personal website of Chris Coyier*. 26 November. Available at: *https://chriscoyier.net/2019/11/26/ive-had-some-vcs-asking-me-about-jamstack/* [Accessed 25th February 2021].

Creswell, J. (2015) *A Concise Introduction to Mixed Methods Research*. California: SAGE Publications.

Denscombe, M. (2010) *Good Research Guide: For Small-scale Social Research Projects*. 4th edn. London: McGraw-Hill Education.

Denysov, A. (2019) A Look at JAMstack's Speed, By the Numbers. Available at: *https://css-tricks.com/a-look-at-jamstacks-speed-by-the-numbers/* [Accessed 23rd November 2020].

Drasner, S. (2020) How Smashing Magazine Manages Content: Migration From WordPress To JAMstack. Available at: *https://www.smashingmagazine.com/2020/01/migration-from-wordpress-to-jamstack/* [Accessed 23rd November 2020].

Ekwuno, O. (2019) Why you should be using JAMstack. *LogRocket Blog*. 20 June. Available at: *https://blog.logrocket.com/why-you-should-be-using-jamstack/* [Accessed 23rd November 2020].

Fayock, C. (2020) What is the JAMstack and how do I get started? *Colby Fayock.* 20 February. Available at: *https://www.colbyfayock.com/2020/02/what-is-the-jamstack-and-how-do-i-get-started* [Accessed 2nd May 2021]

Freestone, J. (2020) The issues with JAMStack: You might need a backend. Available at: *https://medium.com/front-end-weekly/the-issues-with-jamstack-you-might-need-a-backend-d101791de36a* [Accessed 23rd November 2020].

Gibb, R. (2019) What is Latency?. *StackPath.* 22 July. Available at: *https://blog.stackpath.com/latency/* [Accessed 20th December 2020).

Grey, D. (2004) *Doing Research in the Real World*. California: SAGE Publications.

Haq, S. (2018) Introduction to Monolithic Architecture and MicroServices Architecture. Available at: *https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63* [Accessed 25th February 2021].

Jamstack TV (2018) *Chris Coyier - The All Powerful Front End Developer*. Available at: *https://www.youtube.com/watch?v=grSxHfGoaeg* [Accessed 1st March 2021].

Jupp, V. (2006) *The SAGE Dictionary of Social Research Methods*. California: SAGE Publications

Jupp, V. and Sapsford, R. (2006) *Data Collection and Analysis*. 2nd edn. California: SAGE Publications.

Koshy, V. (2009) *Action Research for Improving Educational Practice : A Step-by-Step Guide*. 2nd edn. California: SAGE Publications.

Laws, S., Harper, C., Jones, N. and Marcus, R. (2013) *Research for Development: A Practical Guide*. 2nd edn. California: SAGE Publications.

@Zachleat (2019) 6 September. Available at: *https://twitter.com/zachleat/status/1169998370041208832* [Accessed 18th December 2020].

Leavy, P. (2017) *Research Design: Quantitative, Qualitative, Mixed Methods, Arts-Based, and Community-Based Participatory Research Approaches*. Guilford: Guilford Publications

Manandhar, I. (2020) The Rise in JAMStack Application. Available at: *https://medium.com/weekly-webtips/the-rise-in-jamstack-application-2b598e128ccc* [Accessed 21st December 2020].

Meyer, E. (2020) Get Static*. Eric's Archived Thoughts*. 22 March. Available at: *https://meyerweb.com/eric/thoughts/2020/03/22/get-static/* [Accessed 23rd November 2020].

Myers, A. (2018a) JAMstackConf: A Sugar Rush*.* Available at: *https://medium.com/memory-leak/jamstackconf-a-sugar-rush-397a79cb1a39* [Accessed 25th February 2021].

Myers, A. (2018b) The JAMstack: It's Pretty Sweet*.* Available at: *https://medium.com/memory-leak/the-jamstack-its-pretty-sweet-e0834e4e6bb7* [Accessed 25th February 2021].

Netlify (2020) The modern way to build websites & apps that delivers better performance. Available at: *https://jamstack.org/* [Accessed 24th December 2020].

Segala, A. (2019) Your Next App May Not Have a Back End. Available at: *https://medium.com/better-programming/your-next-app-may-not-have-a-backend-aacc728bd032* [Accessed 23rd November 2020].

Zey, T. (2019) JamStack from a monolith. *Tyler Zey.* 24 January. Available at: *https://tylerzey.com/jam-stack/* [Accessed 21st December 2020].

Zohrabi, M. (2013) Mixed Method Research: Instruments, Validity, Reliability and Reporting Findings. *Theory and Practice in Language Studies*, 3(2), pp. 254–262. Published by Academy Publisher. Available at: *http://www.academypublication.com/issues/past/tpls/vol03/02/06.pdf* [Accessed 14th April 2021]

Bibliography:

Copes, F. (2018) *The Express Handbook*. Flavio Copes. Available at: *https://flaviocopes.nyc3.digitaloceanspaces.com/express-handbook/express-handbook.pdf* [Accessed: 5th January 2021]

Copes, F. (2018) *The NODE.JS Handbook*. Flavio Copes. Available at: *https://flaviocopes.nyc3.digitaloceanspaces.com/node-handbook/node-handbook.pdf* [Accessed 5 January 2021].

Dahal, G. (2020) My Long Journey to a Decoupled WordPress Gatsby Site. Available at: *https://css-tricks.com/my-long-journey-to-a-decoupled-wordpress-gatsby-site/* [Accessed 23rd November 2020].

Netlify (2021) Welcome to the Jamstack. Available at: *https://www.netlify.com/jamstack/* [Accessed 25 February 2021].

Traversy, B. (2017) *Build A Node.js & Redis App From Scratch*. Available at: *https://www.youtube.com/watch?v=9S-mphgE5fA&t=1027s* [Accessed 5th January 2021].

Traversy, B. (2019) *Redis Caching in Node.js*. Available at: *https://www.youtube.com/watch?v=oaJq1mQ3dFI&t=809s* [Accessed: 5th January 2021].

Traversy, B. (2019) *Express JS Crash Course*. Available at: *https://www.youtube.com/watch?v=L72fhGm1tfE&t=3147s* [Accessed: 5th January 2021].

Traversy, B. (2017) *Redis Crash Course Tutorial*. Available at: *https://www.youtube.com/watch?v=Hbt56gFj998&t=2068s* [Accessed: 5th January 2021].

Traversy, B. (2019) *Node.js Crash Course*. Available at: *https://www.youtube.com/watch?v=fBNz5xF-Kx4* [Accessed: 5th January 2021].

Wieruch, R. (2020) Gatsby.js - A powerful Static Site Generator. *RWieruch*. 8 July. Available at: *https://www.robinwieruch.de/react-gatsby-js* [Accessed: 5th January 2021].

Wieruch, R. (2020) Best time to become a JavaScript Developer. *RWieruch*. 11 July. Available at: *https://www.robinwieruch.de/become-javascript-developer* [Accessed: 5th January 2021].

Appendices:

Appendix 1 – Survey Questions

<table>
<tr><td colspan="4">Context: For the purpose of these questions I will provide a brief for the development of a project. With this project in mind you will be able to answer the questions below.

You or the company/agency you work for are about to undertake the development of a new website, this app will be a simple e-commerce platform and blog, that the owner will need to add content and products to. With this in mind:</td></tr>
<tr><td>Q no.</td><td>Questions</td><td>Reason for question</td><td>Expected Responses</td></tr>
<tr><td>1</td><td>Which area of web development do you specialize/are strongest in?
• Front-End
• Back-End
• Full Stack</td><td>To see if where abouts on a website you work affects your opinion and see any patterns</td><td>Front-End / Back-end / Full Stack</td></tr>
<tr><td>2</td><td>What sort of development work do you/the team/company you work at undertake?
• Freelance
• Agency
• In house, working on a product
• Learning/student
• Hobbyist</td><td>To see if the kind of work you do affects your need for a backend</td><td>Freelance / Agency / In-house etc</td></tr>
</table>

3	What is your current job title? If a student or hobbyist, please just state this again	Understand the job roles of those answering the survey	CTO
4	What architectural approach would you take when producing the website? • Server side rendering e.g. The use of backend frameworks such as Laravel, Django or Express to render and serve pages from the server per user request. Wordpress etc. • Serverless (JAM stack) e.g. pre-rendering a static build of the website and serving that via a CDN to each request. "As a service" products often used to provide dynamic capabilities. • Client side rendering e.g. requesting data from the clients browser. Often via a modern frontend framework (such as React, Vue, Next.is etc). Backends aren't used for routing or generating HTML per requests but instead to just provide the data.	To see how people think about approaching projects and the rendering methods available	Server side rendering / Serverless / Client side rendering

5	Which technology stack / technology would you choose to carry out the project? • (Specify) • (Specify) • (Specify) • …	To see what stacks people would use for developing the example project	E.g. React, Express, MongoDB, node Or MERN
6	Please provide a reason for your choice of technology and stack	To understand why they are using the said stack	It is fast, cost effective and what the rest of the team are skilled in
7	Would you isolate your frontend from your backend? • Yes • no	To see if there is potential to use the Jamstack	Yes / No
8	Have you ever had issues surrounding website performance, security or down time with the stack specified?	To see if they have ever experienced any problems with this stack	Checkboxes / yes, issues with performance
9	Have you used any of these "as a service" products in projects undertaken: Headless CMSs, Algolia, Auth0, Snipcart, Stripe, FaunaDB, Firebase, Serverless functions, or other SaaS products	To find out if they are not using the Jamstack/serverless but they are already using some of the services used for the Jamstack	Yes / No

10	Which of the following services have you used before? Headless CMSs • Algolia • Auth0 • Snipcart • Stripe • FaunaDB • Firebase • Serverless functions • Other (Specify)	To Identify the type of services used.	Checkboxes
11	Through the use of these services did you find there were any advantages or disadvantages compared to building out the functionality in your own backend? If so what were they? (Specify)	To find out the participant's thoughts and opinions on these services compared to building a backend	I thought there were advantages, it allowed for a faster, more secure implementation of crucial functionality. Reduces the margin for error on our behalf.
12	Do you feel using these 'backend-as-a-service' products together could eliminate the need to develop your own backend? • Yes • No	Understand whether they think these services could completely replace building backends for them	Yes / No
13	Can you give a reason for your answer?	Then understand why they think this is the case	No, there will always be a need for custom functionality

14	Is the JAM stack something you are familiar with and have experience using in projects? • I've heard of it but don't know much about it • I'm familiar with it and how it works but choose not to use it over other stacks • I'm familiar with it and how it works but need to find time to try it out • I actively use it in projects • I've never heard of it	Whether the participants have experience using the Jamstack or know what it is	Multiple Choice
15	Why haven't you/don't you use the "as a service" products?	To understand why the participant does not use the APIs that make up the Jamstack and could replace the backends	They can get very expensive quickly and don't offer the customisation we need
16	Why do you choose not to use the Jamstack over other stacks/solutions?	To understand the reasons why the participant chooses not to use the Jamstack over other solutions	We don't want to have our core functionality to our product be reliant on a third party. We want control over it.

17	Why do you use the Jamstack over other stacks/solutions?	To understand the reasons why the participant chooses to use the Jamstack over other solutions	The speed, performance and ability for the project to scale with incredible ease.

Appendix 2 - Survey Results

Question 1

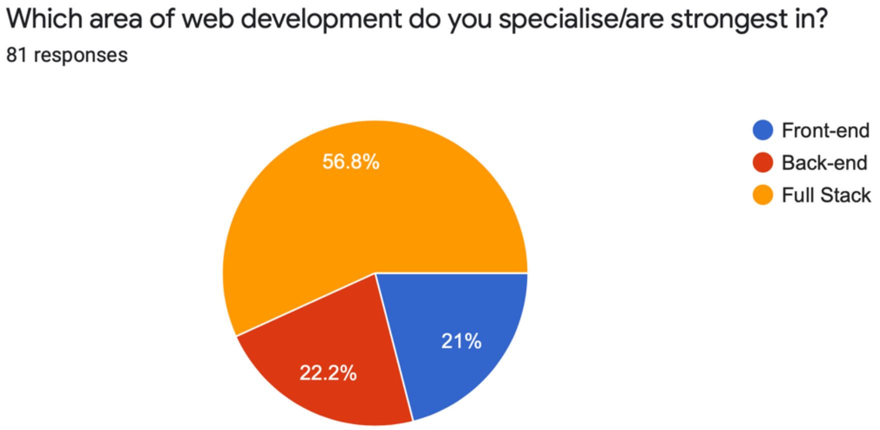


Figure 6 shows that out of the 81 responses 56.8% considered themselves full stack developers, 22.2% back-end developers and 21% front-end developers.

Question 2

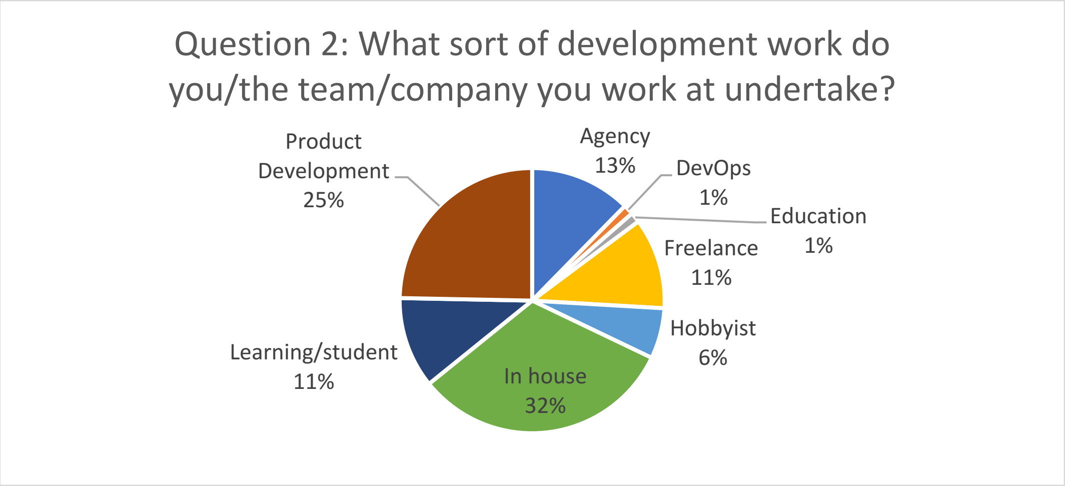
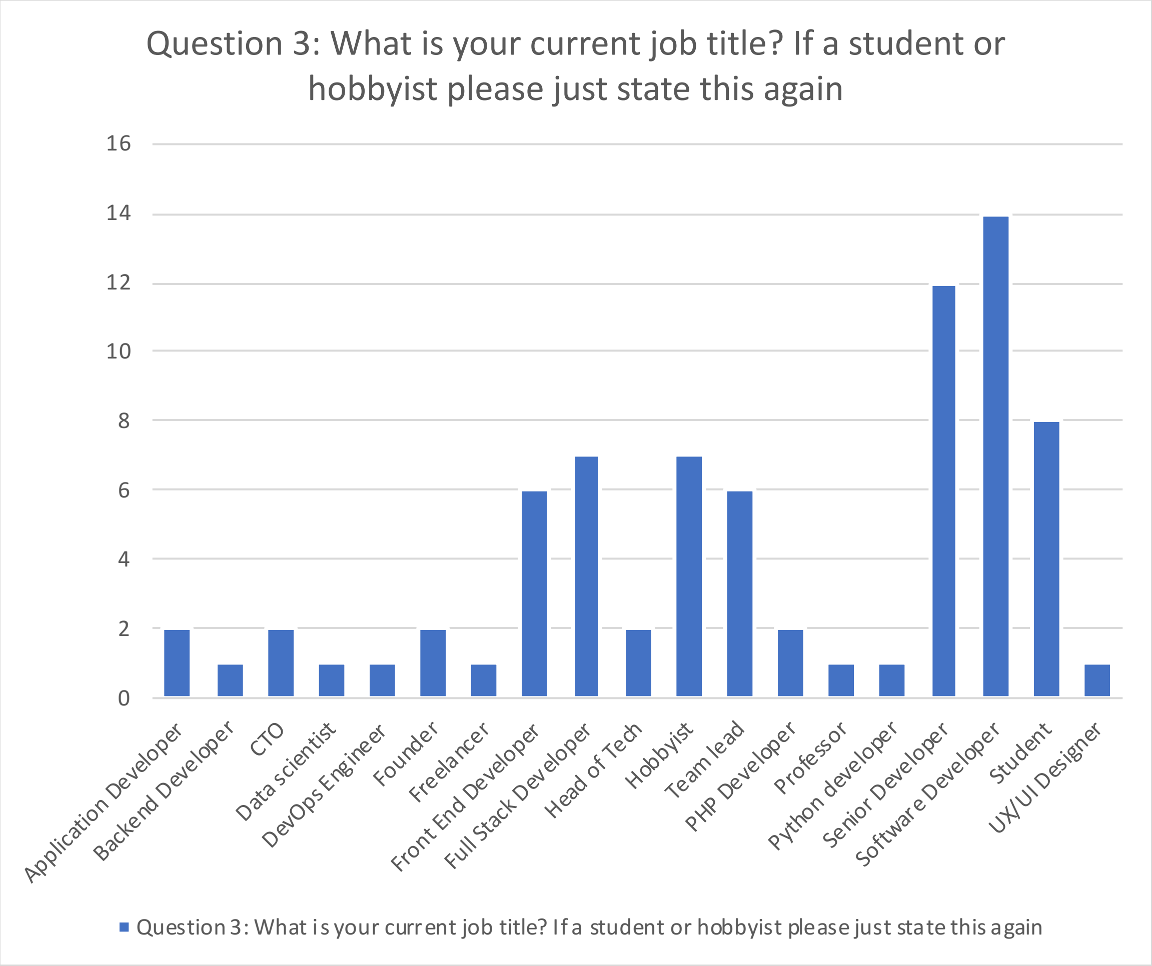
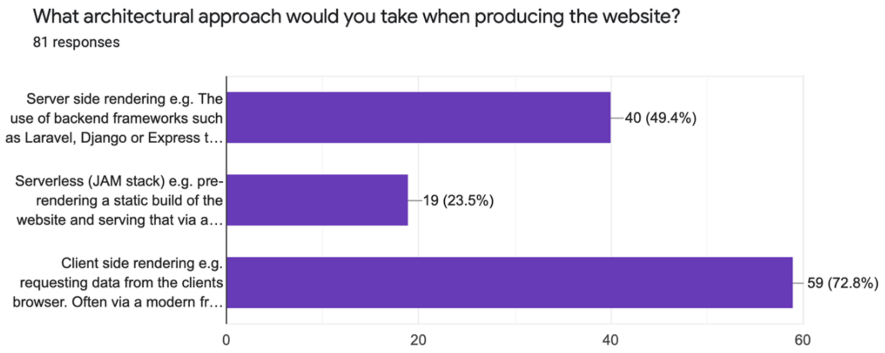


Figure 7 shows that 32% of the respondents worked in-house, 25% worked towards building out a product, 13% worked in agency while 11% were freelance and students/learning respectively.

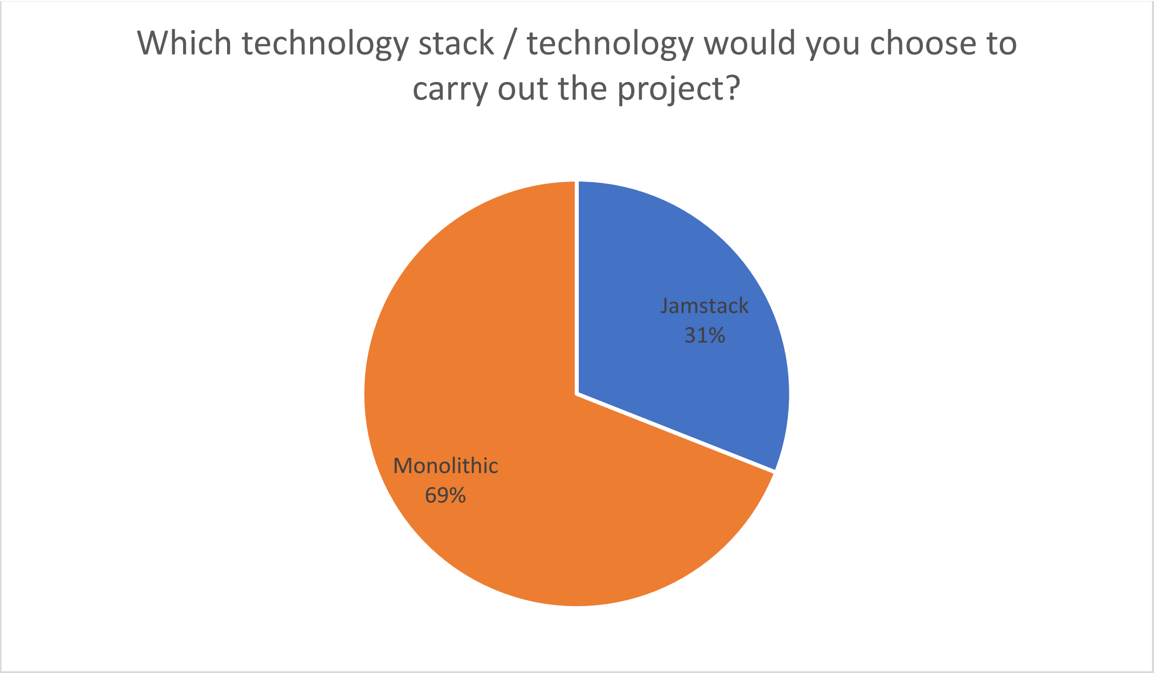
Question 3

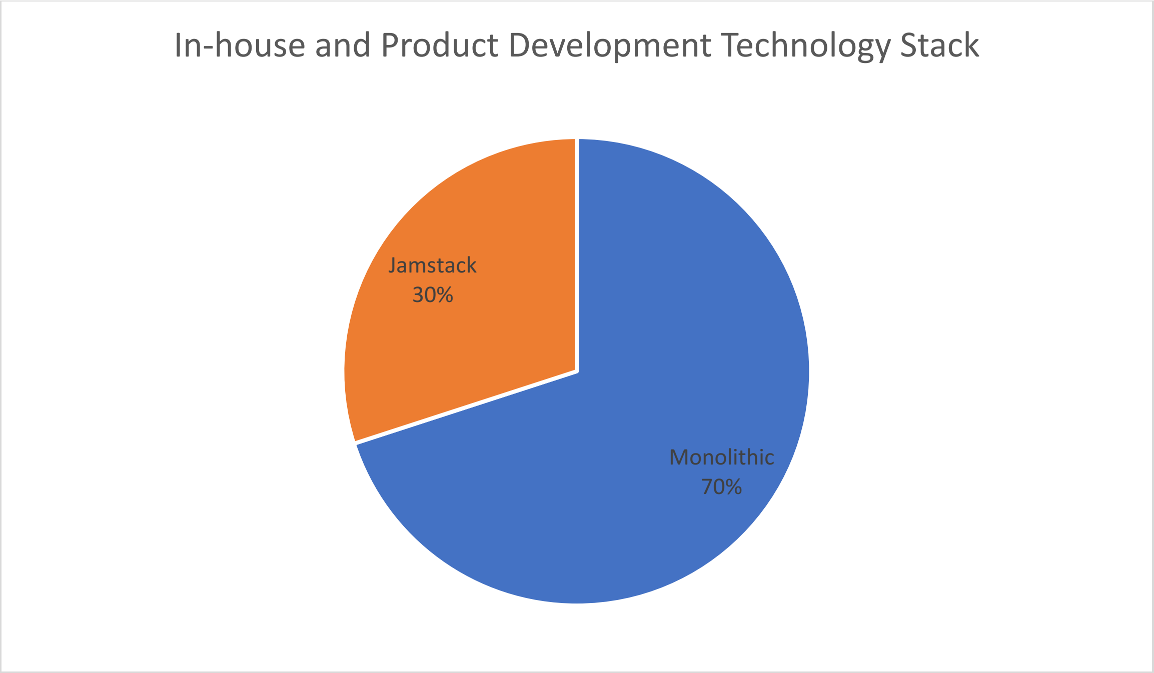


Question 4


What architectural approach would you take when producing the website?
81 responses
Server side rendering e.g. The use of backend frameworks such as Laravel, Django or Express t…
Serverless (JAM stack) e.g. pre-rendering a static build of the website and serving that via a…
Client side rendering e.g. requesting data from the clients browser. Often via a modern fr…
40 (49.4%)
19 (23.5%)
59 (72.8%)
0
20
40
60

Question 5


Which technology stack / technology would you choose to carry out the project?
Jamstack
31%
Monolithic
69%



Question 6

Please provide a reason for your choice of technology and stack and the reason you picked it over other solutions?

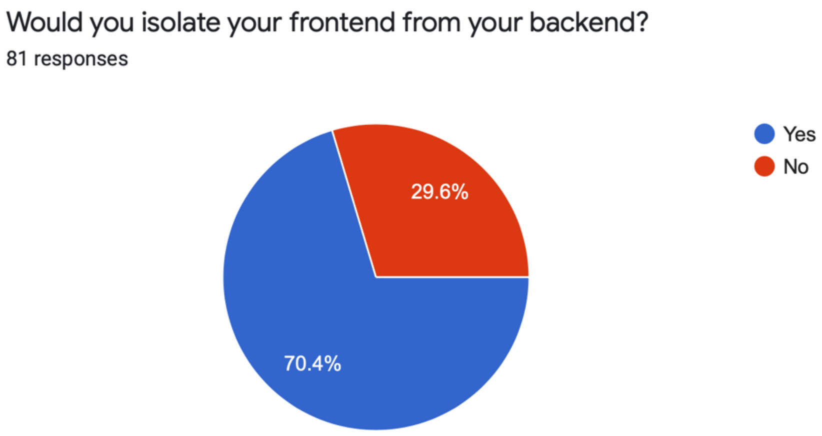
		Themes	
		Performance	“Stability” “No need for client rendering and better SEO. Try using a heavy react app on an old android.” “Speed.” “Django provides really nice serialization as well as pagination... Node is great if you have a ton of IO and users visiting the site.” “Go is fast”

		Developer Experience	“familiarity” “It's really simple to setup, develop, and deploy. It would also let me quickly develop out the MVP for the client.” “Simplicity and custom functionality is easier to architect without fighting an opinionated system, in the latter case, flexibility and speed is key”
		Support	“Large community support” “well documented”
		Cost	“economy (cheaper devs)”
		Performance	“Time to deliver” “Fast and strong” “storefront is backend agnostic.” “It’s all bundled and easy, and great for simple cases like this”
		Developer Experience	“Simple to build in, pre built templates to work from and great hosting/tooling.” “easy”
		Support	“large communities of support” “Modern frameworks with huge communities and support.” “great docs and community”
		Performance	“It's great to get a fast ramp up of productivity, and allows scaling as the projects grow” “Great Performance, mostly a no-brainer with ORMs” “.NET scales best for this type of application. Node can be used for middleware or small scripts like login, where it really excels” “Django allows for rapid development and I don't want a complicated frontend so I use jquery.”
	no	Developer Experience	“fast and simple development with those frameworks” “There's an easier handoff of responsibilities between api developer, and UI developers Android/ iOS/ web” “Wide range of libraries available. Easy maintenance. Clean code.”

		Performance	“We know react quite well, it integrates well with many content sources and allows you to balance static pages with dynamic content.” “easy to support. WooCommerce is a popular and well tested and used... There's no point reinventing the wheel.” “allows serverside rendering, static site generation, client side fetching with static pages”
		Cost	“Lower hosting costs” “There are existing solutions for basic e-commerce and they work really well. No need to reinvent the wheel. It would be extremely uneconomical to waste developer/sysadmin hours building and maintaining a new platform when we can just throw $29 a month at the problem.”
		Developer Experience	“A headless CMS allows for others to add blog posts… quick to develop.” “The ability to decouple the frontend from the database. Allow non-technical people to update content.”
		Performance	“Plain html - seo friendly” “Fast, safe” “Laravel for quick database connection”
		Developer Experience	“familiarity, stability”
		Support	“Big ecosystem/community and great documentation. Battle tested”
		Performance	“These softwares offer a very fast way to build a site from scratch, and is also maintained via git hub for the most part. Mongo also has a console dashboard where data can be added extremely quickly.” “NextJS provide extremely fast website and Strapi is a really configurable cms”

		Performance	“Reliable, lots of documentation and can be used in many scenarios. Can incorporate all features required in the brief.” “Effiency” “Docker makes it easier to transfer between servers and manage the deployment/development.” “It's quick to set up and develop for”
		Performance	“To isolate frontend from backend, and to generate pages to improve their SEO.”

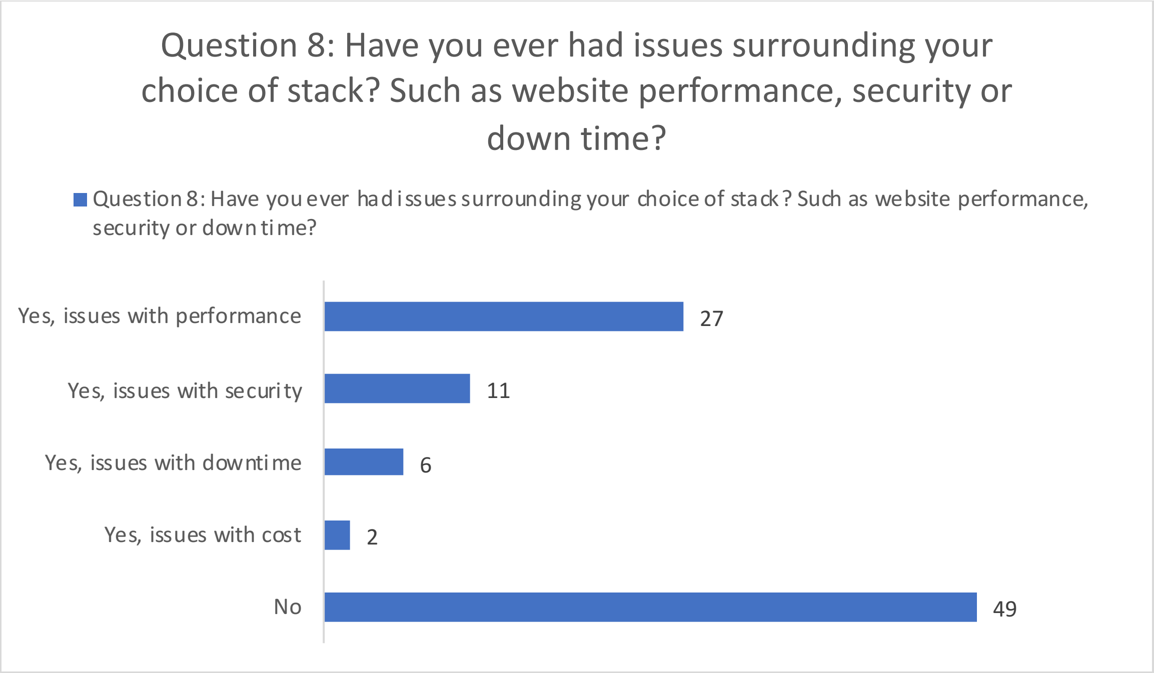
Question 7



Would you isolate your frontend from your backend?

81 responses

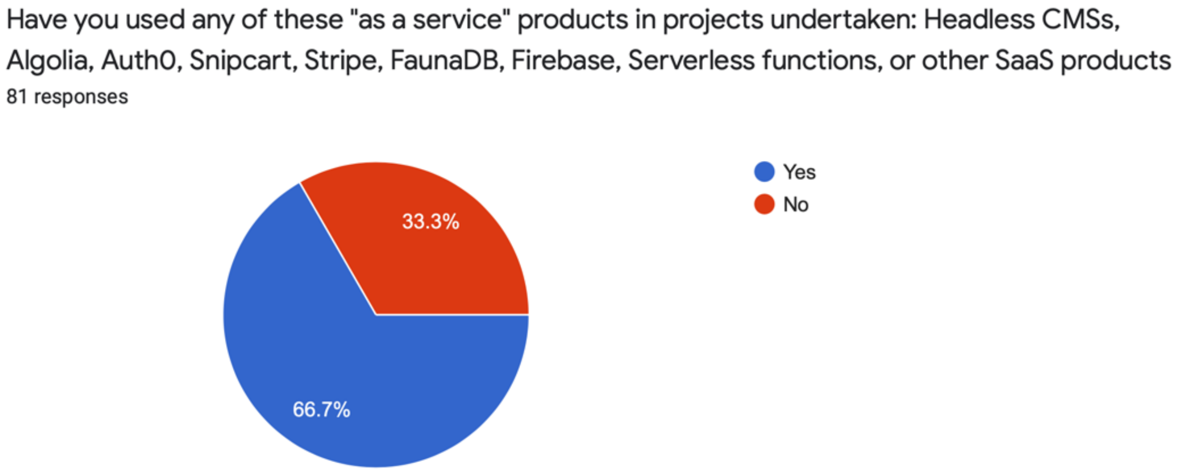
Question 8: Have you ever had issues surrounding your choice of stack? Such as website performance, security or down time?



70.96% of those who stated they would choose to use a monolithic architecture also admitted to experiencing issues surrounding it.

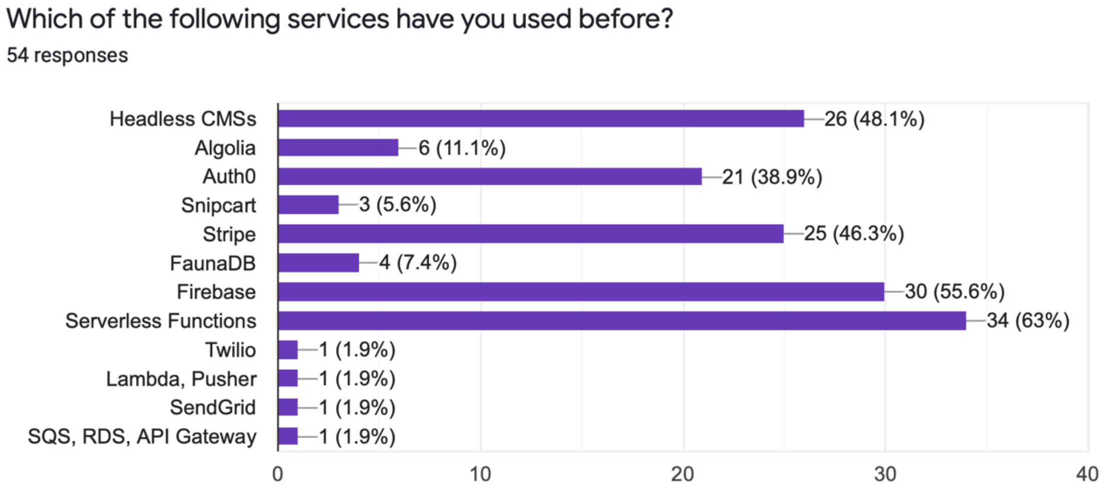
16% of those who stated they would choose to use the Jamstack also admitted to experiencing issues.

Question 9

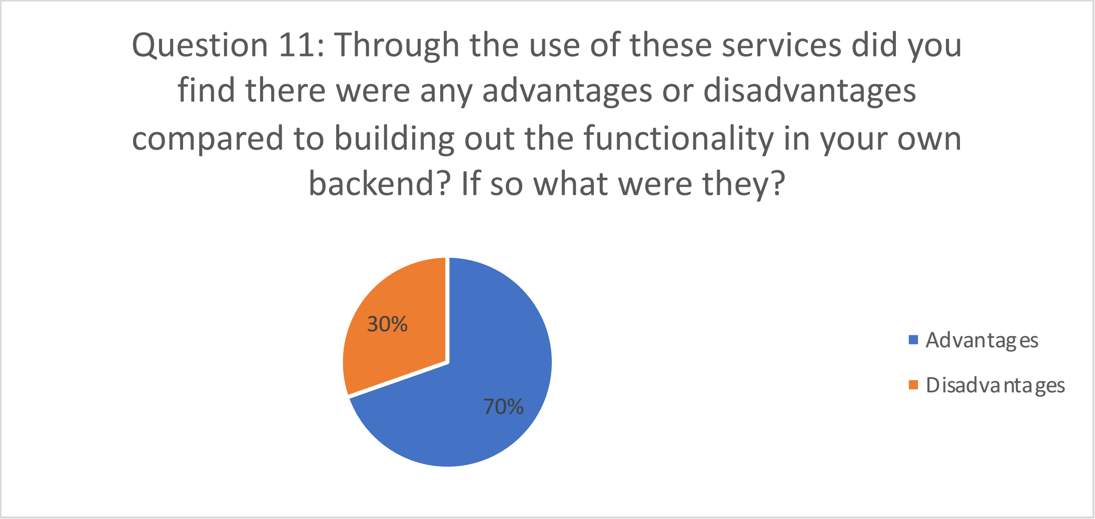

Have you used any of these "as a service" products in projects undertaken: Headless CMSs, Algolia, Auth0, Snipcart, Stripe, FaunaDB, Firebase, Serverless functions, or other SaaS products

81 responses

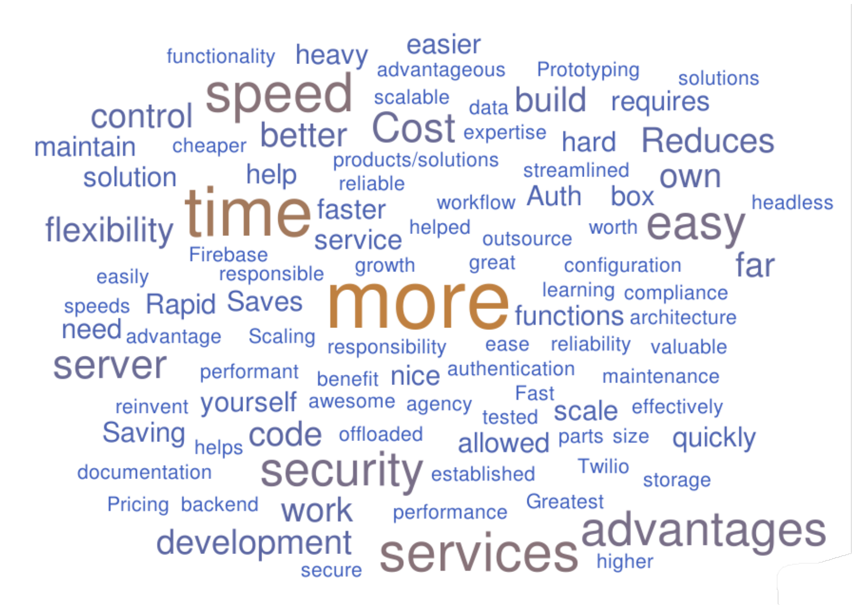
Question 10



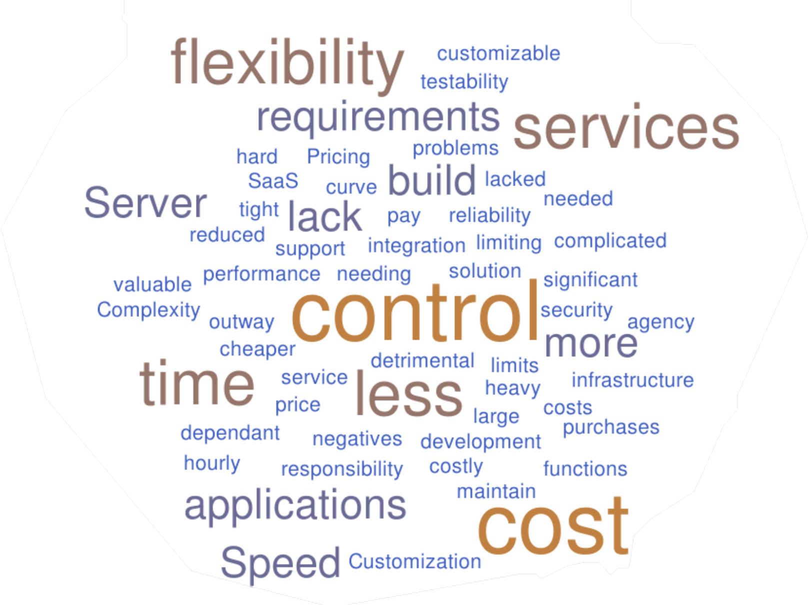
Question 11 - Through the use of these services did you find there were any advantages or disadvantages compared to building out the functionality in your own backend? If so what were they?



Advantages



Disadvantages

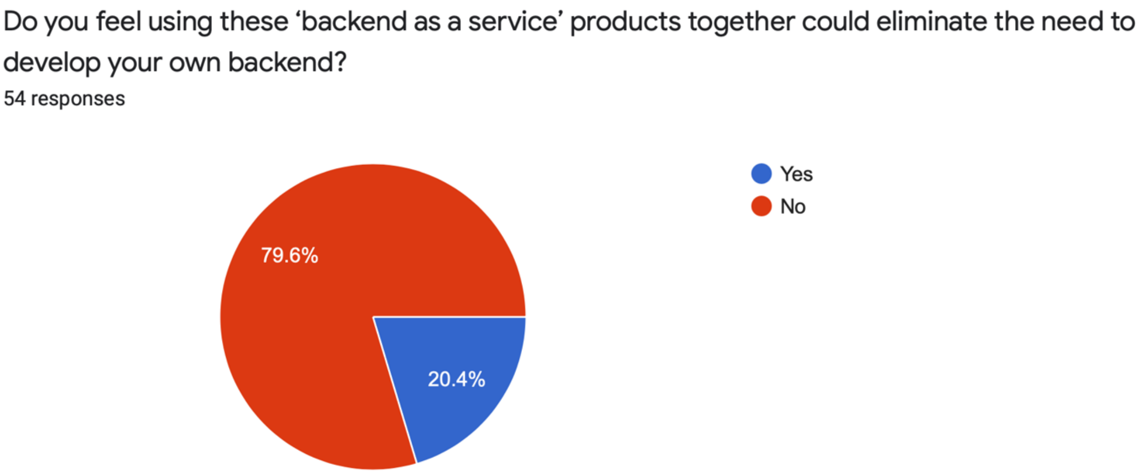


Thematic analysis - Advantages or Disadvantages of using services compared to building out own backend

	Themes	
	Performance	“Fast, reliable, easy“ “they scale way better” “Yes, less to build, maintain” “Rapid Prototyping” “More scalable for rapid growth in usage.” “Makes for a faster development speed and provides better security and means your as a company are not directly responsible for data protection.” “reduces the barrier of entry for junior developers coming into the project. Plus some things (notably payment processing) carry a heavy compliance burden and it's nice to be able to effectively outsource those things.” “Time to market” “Saves time. Reduces complexity in your own code. Reduces the amount of possible failure points in your application” “Advantages: more complete solution, cheaper than dev time, less responsibility, expertise, better support.”
	Developer experience	“Easy setup” “Yes! Much faster to implement things like push notifications, authentication, etc.” “The server less function were more restrictive and harder to run/ deploy with code than standard server solutions.” “There is still a bit of fine tuning using these services, but it definitely speeds projects up when you don’t have to build everything from scratch.” “Easy to scale” “As a front end developer, services like these help a lot by giving easy to understand functions and helpers, as well as most of them coming with great boilerplates and documentation” “Works easily and quickly” “They can significantly speed up development” “Sometimes it requires a little bit more thinking but in the end the result is far more worth it.” “Its a solution out of the box” “Many advantages as it allowed more time to develop code that made other parts of my project more unique.”
	Cost	“Cost saving”

		“Price and speed” “Saving Time vs Cost” “It saves time”
	Performance	“Depends on the Use case. It's easy to build something with Firebase but if there are specific requirements it gets hard. Stripe & Auth0 are great, not limiting requirements and easy to use.” “Complexity of infrastructure, cost, tight coupling to third party” “Server less functions can have a large 'turn on' time, which is especially detrimental for time-sensitive applications, like APIs” “Massive disadvantages. You are simply spreading your problems across multiple interfaces.” Disadvantages: can be costly, may be more complicated than you need, dependant on that service.” “Greatest disadvantage is probably control and flexibility. As we were developing custom products/solutions we found that backend services helped with a lot of the heavy lifting at the cost of reduced control and flexibility in case we needed to do some kind of special integration with one of our other applications.”
	Developer experience	“non customizable snipcart” “Fire base is awesome, however i felt like I lacked control overall - I’m also not so keen on pay as you go services” “Disadvantage is lack of flexibility.” “Customization.” “Less control and lack of testability”
	Cost	“no, just the price / limits” “Pricing can be higher with SaaS but you get the ease of never needing to maintain a server” “Speed and reliability is the main key advantage that typically outweighs doing it ourselves. Disadvantage is the hourly costs that grow.”

Question 12

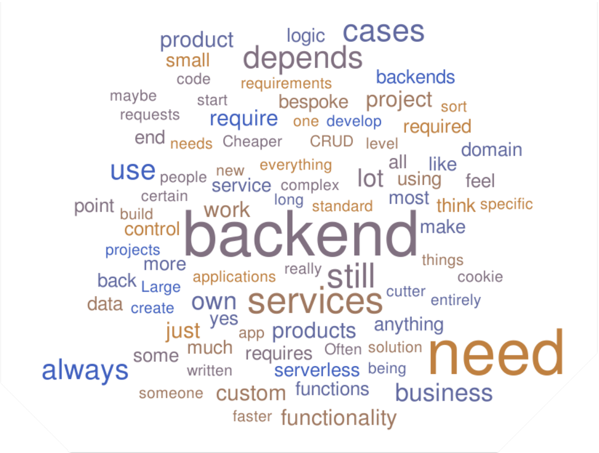


Do you feel using these ‘backend as a service’ products together could eliminate the need to develop your own backend?

54 responses

Question 13

Can you give a reason for your answer?

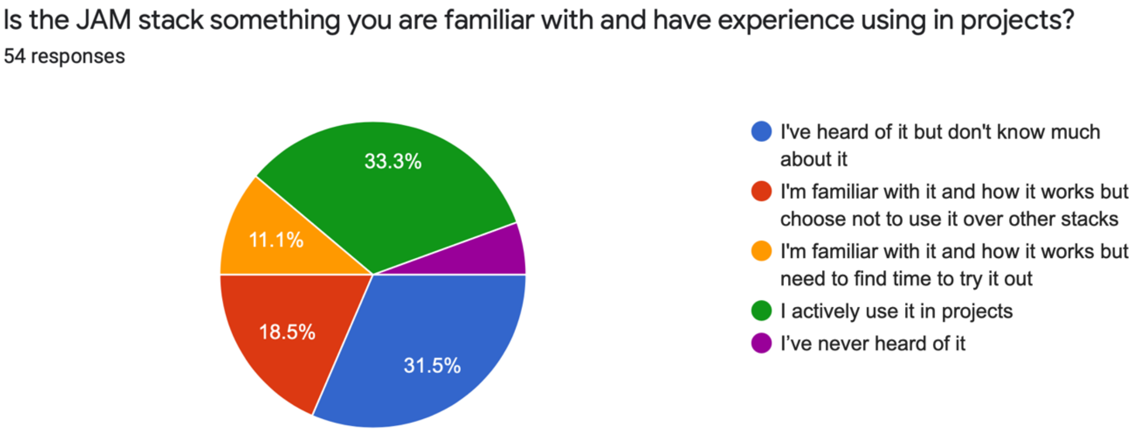


Thematic Analysis

First order indicator (example quotes from interview	Second order indicator (sub-themes)	Main Themes
"They can substitute certain parts of a backend but you still need code to bring everything together" "There will also be functionality not covered by these services and cases where they can not be afforded." "Often we need bespoke elements, using off-the-shelf solutions typically end up requiring work arounds to make them work as required." "Because it will take a long time to reach the level of customization in writing the backend ourselves.." "I like to have a flexible app" "If the stakeholders require security or massive amounts of data transfer, I'll use a backend" "loss of control" "if there's enough money, backend services could be implemented for faster mvp" "These days serverless functions can do most things."	Performance Flexibility Control	Technical capability
"90% of backend is CRUD, and these services tend to make CRUD really expensive to run, and make building the remaining 10% more complicated." "Cheaper and faster to use 3rd party than to develop a new system" "For a small product, maybe, but not for a larger product as you lose so much fine control" "no need to reinvent the wheel. If its a cookie cutter project, theres a template for the back end" "Most my backends are similar and I copy and paste between them. Only need to develop your own backend in niche situations."	Speed Simplicity/complication	Developer Experience
"cases where they can not be afforded" "90% of backend is CRUD, and these services tend to make CRUD really expensive to run"	Expensive Cheaper	Costs

“Cheaper and faster to use 3rd party than to develop a new system” “Costs add up”		
“Business domain logic is hard to outsource” “There are always going to be edge cases that requires custom code.” “There are always unique issues to your problem domain that are not worth solving for the people who write the "as a service" products.” “These services give a good starting off point, but often need retooling for particular use cases. “	Company Value Business Logic	Reliability

Question 14

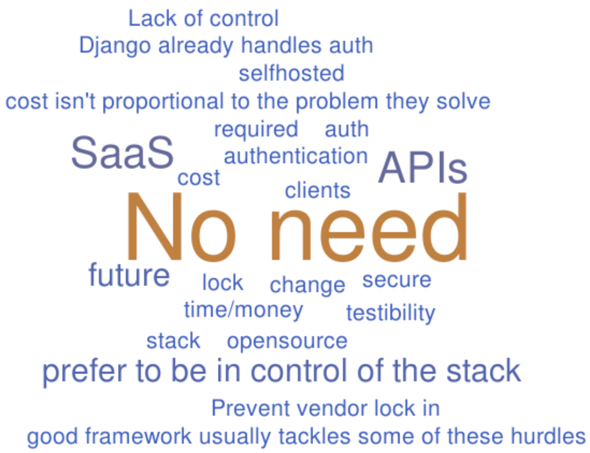


Is the JAM stack something you are familiar with and have experience using in projects?

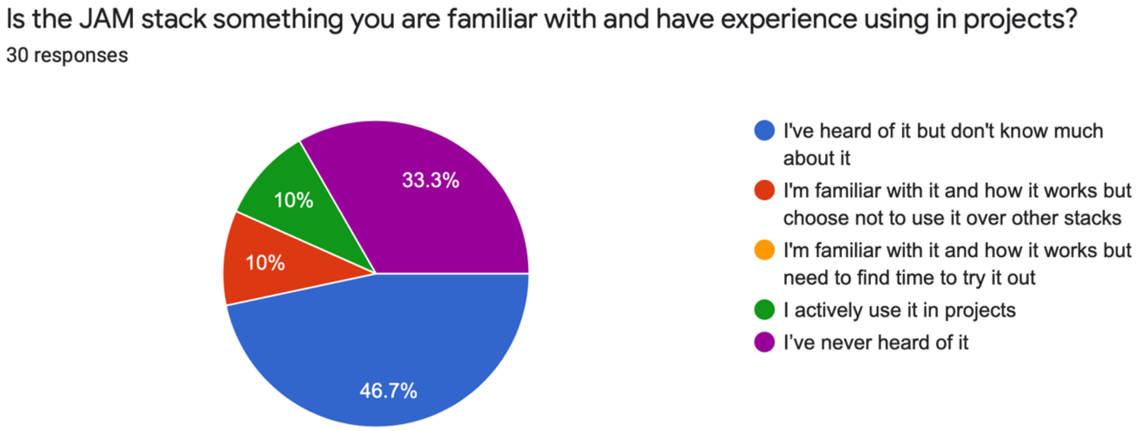
54 responses

Question 15

Why haven't you/don't you use the "as a service" products?



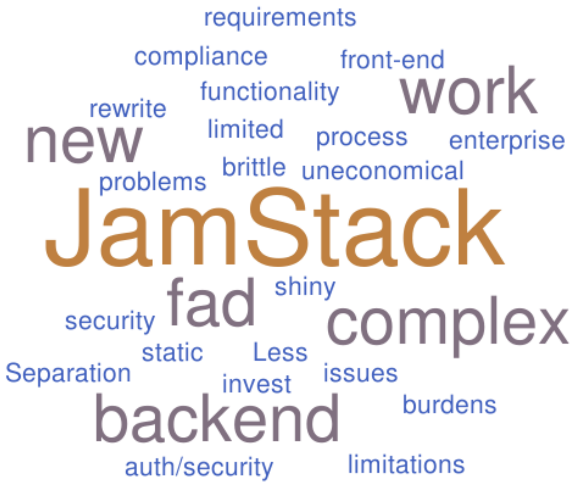
Question 16


Is the JAM stack something you are familiar with and have experience using in projects?

30 responses

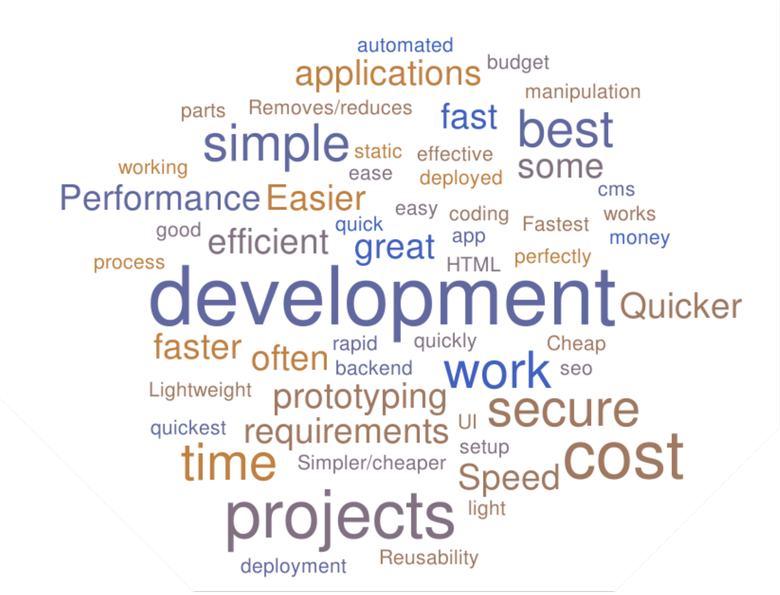
Question 17

Why do you choose not to use the Jamstack over other stacks/solutions?



Question 18

Why do you use the Jamstack over other stacks/solutions?



Appendix 3 - Table of Interview Codes

Colour:	Codes:
Dark Blue	Technology
Purple	Reference to Monolithic
Purple	Issues with Monolithic Architectures
Yellow	Benefits of Monolithic Architectures
Green	Reference to Jamstack
Green	Issues with the Jamstack
Blue	Benefits of the Jamstack
Grey	Service as a solution API
Black	Benefits of using APIs
Red	Issues with them
Dark yellow	Industry position

Appendix 4 – Participant A Interview Transcript

1. **Me**: There we go. Well, hello Participant A.
2. Thank you for, uh, doing this interview with me.
3. The topic is, uh, looking at trends in web development stacks.
4. Yeah, I'll jump right into it.
5. The first question is, what development industry do you currently work in?
6. Like an in-house developer, agency developer, freelance, student etc.
7. **Participant A:** Agency, freelance and student.
8. But officially agency.
9. **Me**: officially you're an agency developer?

Participant A: Yeah

Me: Awesome. What is your current job title there?

Participant A: Full stack developer I think yeah.

Me: Nice, so for this next question I'll give you some context. So if you were approached to make a simple blog website while at the agency you work at for a client, how would you approach it in terms of the technologies and stacks used?

Participant A: Depending on the complexity of it, anything from. We could do WordPress, which is obviously the go-to for most companies or erm, will go for a custom Laravel with backpack. And then front-end we usually use either Nuxt, or some other sort of JavaScript front end.

Me: Oh nice. So like a LAMP stack?

Participant A: Yeah so it'll be LAMP stack and then whatever front end suits the project Really, depending on how mad the front end needs to be.

Me: So what made you decide to use those solutions and how would it benefit the client and the project?

Participant A: So if it's a WordPress one, WordPress is obviously quite a big thing in the industry, a lot of clients will know how to use it straight off the bat. And then with stuff like Laravel. We can make that as custom As we need it to be so whatever we need it - you know, for the project and then the front ends It just depends on what the website is going to be. If it's like a normal consumer website, we can probably deal with it just being a standard Yeah, Flat website. But if it needs something fancy like bit of e-commerce we will generally use like big nuxt frontend that's taking APIs from like Magento, Laravel, WordPress for all the different parts of it. So like Magento for the E Commerce Laravel for the content editor, yeah also sorts of stuff into a big front end.

Me: Nice, have you ever had any problems using this approach? Any drawbacks? Any limitations?

Participant A: WordPress tends to be a bit annoying when trying to expand anything. That's why we tend only to use it for very simple projects, but with.

42. Now with with Laravel and JavaScript its, I've never run into anything mad limitations.
43. **Me:** Nothing to do with performance or security or like scaling the projects?
44. **Lewis Ragget:** Yeah, so the performance thing I've ever run into is using Algolia.
45. **Me:** Oh the er, search-
46. **Participant A:** actually that's not even performance to be honest.
47. That's more cost, because it's expensive.
48. Yeah, see It's 'cause we've we've written like.
49. There's a site that we worked on. It has about what 40,000 different products?
50. Yeah, you need like a fat off search for that.
51. Erm, and yeah we ran into a bit of a cost issue with that,
52. so we've not used it since. We're more geared towards Elasticsearch,
53. which is open source.
54. Yeah, so that was a bit of a nightmare
55. but performance wise. Without stuff like - the trouble is it's costly because without
56. it search could be quite slow depending on how much it's optimized.
57. Depending on what the websites like, especially if it's like an
58. older projects and like, yeah.
59. **Me:** Yeah they're like market leaders aren't they.
60. **Participant A:** Yeah, definitely, It's fast as hell.
61. **Me:** In terms of building this project, are you aware of any other alternatives
62. or stacks that you could have used instead of the word press or lamp stack?
63. **Participant A:** Yes, I see a lot of companies will
64. end up using express with like Mongo DB or like the SQL-less, no
65. SQL databases and stuff like that.
66. Erm, which I've tried on personal projects but I just
67. don't see how it could be made - maybe it's just 'cause I haven't worked on
68. anything commercial but I couldn't see it being used in production.
69. Just 'cause of how finicky it is. I just find SQL databases a lot easier
70. to navigate around and structured data
71. **Me:** over the non-relational databases and stuff.
72. **Participant A:** Yeah, exactly.
73. That answers my next question as well as to why you didn't

choose to use that so, thanks. I just don’t think it’s as scalable, and the cost is there as well, so it's. Yeah.

Me: So also I guess you’ve also kind of answered this as well. Have you used any of these as a service products in projects undertaken so as a service product like a headlessCMS, Algolia, Auth0, Snipcart, Stripe, FaunaDB, Firebase or service functions? Have you used anything of that kind?

Participant A: I’ve used Stripe, Firebase, Algolia erm not Snipcart but Firebase I've used for one personal project but it's just too expensive to be viable on anything big. Stripe, amazing 10/10. Use that, use that- Anything that needs payment we use Stripe. Stripe is the best and then for um. What is the other one? Yeah, Algolia already said, we just use that for search stuff If it's if it's not too big a scale.

Me: And I guess haven’t you used a headlessCMS Like the Wordpress API as well?

Participant A: Yeah, yeah we’ve used. Yeah definitely yeah yeah Word press on them.

Me: So through using these services, did you find there were advantages compared to building out the functionality from scratch and in the back end?

Participant A: A lot quicker

Me: yeah?

Participant A: But like I say, it's not as if- I would only use them on projects that you know the exact scope of and aren't going to evolve past like a basic site and do what you're exactly set out to do. If you've got clients who are going to want version 2's or extra content and stuff, it's a bit of a nightmare to code stuff into that sort of stuff.

Me: Yeah

Participant A: and you end up just adding on a custom thing anyway in the end.

Me: I get that.

Participant A: And it depends on the thing really, 'cause like if it's something like Firebase or Algolia, you can scale them, it just gets a lot more pricey but with stuff like Stripe it's, it's, It's just a part of the website and then it's not really a, yeah.

Me: yeah. So maybe to look at like Firebase or FaunaDB. Would you ever sub out a SQL Database for one of them in a project? Because they might be, for scaling, They might be even more cost effective or more secure because it's a ‘as a service’ product. It's not something you have to look after.

Lewis Ragget: Yeah, so Firebase is really good for Auth and like social Auth and stuff like that. And I've used that in the past. I can see that being used on certain kinds of websites but just 'cause the the nature of the company we work on a lot of company based software, more than just front end websites, so it's not really ideal when you've got so many like relating tables and stuff like that. Yeah, but I could definitely see if it was used for like a like a front end tool or like some sort of website or something I can definitely see posts and stuff being stored in the database like that.

Me: Yeah, awesome. Well, I think that's pretty much it. Cheers, thank you for having an interview with me.

Appendix 5 – Participant B Interview Transcript

Interviewer: So I am with Participant B and I would like to say thank you for doing this interview with me looking at the trends in web development stacks and how they are used by web developers.

Participant B: No worries.

Interviewer: I'll start with some easy questions, I’ll start with what's. your current Job title?

Participant B: I am CTO at [SaaS product company].

Interviewer: Awesome, and what sort of development industry is that in?

Is that an in-house company? An agency? Freelance? Product?

Participant B: Yeah, we're a SaaS, a SaaS product company, so we have a product. that people can use to build bots and other automations to help them be more productive at work.

Interviewer: Ah that's awesome. Do you mind me asking what technology or stacks you use to make your product?

Participant B: No not at all, so we have- fundamentally we base all of our stuff at the moment on AWS. Um, When you're looking at cloud services we just picked the one that was the biggest at the time. We've been going for about four years, five years. Um, so that at the time AWS was kind of the main player, and so far so good and we think we get good stability from them. On top of that, we've run a lot of EC2 instances, we have- This is in production. We have, we don't leverage much in the way of hosted Amazon services. We do use some, so for databases like ElastiCache We also use RDS for SQL and then everything else is kind of running on EC2 instances just to give us the most flexibility we can get. We've got PHP and Symfony on the front-end along with some React JS, On sort of client side, uh that's our kind of you know our bot builder facing stack, which is a relatively low traffic side of our of our product and then we kind of have this other side which is the bit that powers the bots themselves and interestingly that can be, you know, anywhere from 5000 to 10,000 people kind of concurrent depending on what they're doing with the bots at the time. So large customers can send big broadcasts out to significant numbers of users and then all those people engage with the bot around the same time as you see those messages. So we need to have a different stack on that side of things to keep things running quickly. So at the moment that's a mixture of some node JS and increasingly quite a lot of Golang.

Interviewer: Oh, awesome. What made you decide to use those languages like Node you just said and Go and Laravel, Symfony?

Participant B: It kind of evolved overtime and initially we had, well we knew PHP Wouldn't be a good fit for the bot side just in terms of concurrency and the kind of model they use for spinning up you know these single fire once on runtime on the PHP side versus something that's been more invented. So initially we started out with the CMS which was PHP/Symfony we went with them mostly because of the other end layer that came with Symphony the framework itself gives a nice way to fall into an existing paradigm. It made it easier to hire people with because we were using a popular framework and the owner is big on relational sort of foreign keys and such in the database, the back end database which helps us with data integrity. So that was kind of main reasons on the CMS side and then on the back end we just went with something that we were familiar with that we knew had good concurrency model. So we kind of went with node JS initially and then as we kind of developed as a product and we added more features like the broadcasting engine. We kind of pushed more towards like I said Go with the type safety. it's quite easy to read go code. Yeah, that's that tends to be our backend language of choice now. For smaller microservice type build.

Interviewer: Ah, that's awesome. Thank you. Have you ever had any problems with any of these approaches? On any of the back-end stuff you've been using. It could do with costs, scalability, performance or security or anything like that? And how do you deal with these things?

Participant B: I mean what technology stack doesn't give you problems right? I think that the biggest problem that we have for the moment is that we are very small with a quite small product team; and that means that we need people that are quite multi, multi- experience with multiple languages, or at least the ability to pick up and drop. Yeah, you know, so doing a context shift from doing PHP, Symfony front ends and then doing some Golang and doing some node JS. So actually, the diversity of the stack is a bit of a problem with hiring, it's not so much about PHP / Symfony side with the node because people know JavaScript anyway, but the Go stuff tends to be more of a you would hire someone properly that would do Go and that's more difficult to do when you’ve got a small team and the majority of the work that we actually undertake happens on the front sort of customer facing Bot Builder side. In terms of scale, not really. We thankfully have quite a lot of experience around building scalable products, so we tended to approach with the use the best technology for the problem at hand and design everything to be horizontally scalable. So on that basis, we've never really been in a position that scale has been an issue for us, we can just fire up more, more backends. Cost isn't too bad either with us running on EC2, primarily we're able to manage costs quite well. We can, we can deploy smaller instance sizes based on traffic profile and CPU usage etc, which allows us to get quite good utilization. And it's quite cheap to spin up additional nodes. Yeah, performance is generally pretty good and security We pride ourselves on that side of things. So the technology itself isn't a major issue. We do leverage things like HashiCorp Vault, so it's securer storage of secrets we you know we employ, VS Encryption on the volumes, etc. So those sorts of things aren't too, too bad. I'd say the main thing is you know when you're building more and more backends or microservices is quite easy to go into this position where you've got a lot of things talking to each other and it actually becomes quite difficult to reckon about the entire product. So sometimes I do wonder if the complexity that we have can be simplified and somehow just to make it easier to onboard new developers. So I think in terms of scale, that's where we have the biggest problems. It's just it's not, you know, all PHP, Symfony, so you just hire PHP Simply put. And fundamentally, just worth mentioning another kind of large problem we have is the domain knowledge that our engineers have. We kind of sit in the middle between Microsoft Teams and the workplace by Facebook, PPM points to power these bots actually need to understand a lot about how that platforms work. And that's not something that's very common.

Interviewer: No yeah, definitely quite specific that isn't it.

Participant B: Yeah, for sure.

Interviewer: Are you aware of any alternative routes you could have taken in

The technology in the backends you use?

Participant B: Yeah, absolutely. I mean, there's obviously things like Lambda That have got better and better overtime. Yeah, and you know, that obviously comes with its own issues. We've kind of. We've made use of Lambda in very small specific instances. Um, we've got something like PDF generation or something We just really don't care about, you know, throw something at it and build a service around it. And obviously, like these newer approaches these days, which are probably, I would say more useful to start-up's where you know you can kind of build around, I think you mentioned it later on, like headless CMSs and such. When we started the product a lot of that didn't exist. So the, the problem if you like of, you know, running a company or product over a long period of time is you're almost stuck with your choices up to a point right 'cause the cost of making a significant change is certainly significant, so there would need to be, you know, a very compelling reason to make that transition, and sometimes it definitely is better that way. You know?

Interviewer: Yeah, that definitely makes sense. There would be quite a lot of work to like essentially go back on yourself and take away value to then put in something new.

Participant B: Rewrite all the bugs that you fixed, right?

Interviewer: Yeah, yeah, exactly. So similar to that I want to ask have you used any 'as a service' products in any of your products, like headless CMSs, Algolia, Auth0, Snipcart, Stripe like FaunaDB or even like Serverless functions like it does that have any place in your product?

Participant B: yeah serverless, yes we use for the little things I mentioned. We've got PDF generation in there. We've got a couple of other things I think video processing we leverage AWS for. So when Videos are uploaded to S3 we fire off events which transcodes them and produce them. You know animated thumbnails and that kind of thing. I'm quite a big proponent of using like services, Like Auth0 you know, these things that can take something where you want a secure implementation, you know, obviously around Auth, flows. Interestingly we didn't go that way when we were recently building out an API, and the main reason was just, I'm a bit of a Control freak, I guess. I'd rather have the thing that might break under our control than you know someone else drops the ball and suddenly our product doesn't work, so we're quite careful. I mean we do use century for error monitoring. Anything that's kind of out of the critical path from a user's perspective and we can tolerate outage. We would outsource if we can. You know things like payments on Stripe would be an obvious thing we'd look to outsource, because we don't really want to be dealing with PCIDs etc. And yeah, I mean so I think you know we always look to see if there is a way to avoid writing ourselves and with the authentication thing we went with this thing called Hydra, which is open source quite, quite well reviewed, Oauth implementation so it kind of gave us the ability to leverage that work without having to write it ourselves. But being in control of that sort of uptime.

Interviewer: Yeah, well, that's really interesting. So you kind of answered my next question, looking at the kind of advantages of utilizing some of these 'as a service' products, like Auth0 taking away that- You don't. You don't want be handling those sort of details like Stripe as well. You don't have those things because quite a lot at risk there isn't there.

Participant B: And the other consideration, of course, is the time investment, right? You know, especially as a small team, you want to be focusing on the problems that matter and where you can add value in in your own space. Us building our own authentication flow doesn't really add value to our customers. But having that auth flow in place absolutely does.

Interviewer: Yeah, so there's been a quite a few pop up recently or these other services. So FaunaDB is an example where they're like a serverless database. And using it allows, It could take away the work needed in order to allow products to scale as you said. It could take all that work away from you and put the pressure on the service provider.

Have you thought about using this sort of setup like, Fauna DB, just to completely get away or from the extra work that you have to put in order to allow your own database to scale?

Participant B: Yeah, I guess it depends, right? I mean, there's different levels you can go in. You've got. You can self-host on EC2 or you know your own metal. You can outsource databases as I say to AWS where they take care of the running of the database or you know you've got the likes of your as a service things. I think for me it depends on the problem we're trying to solve and how tolerant I feel we can be of outage. The nice thing about hosting on AWS and using the likes of cloud farers that when they break most of the Internet breaks so you always have this. It's not good excuse, but you know it's this ‘You know who else could we have gone with, everyone's using it’ but I think it just depends on you know how much I care about our data. The other things to consider, especially in data, is compliance, right? You know if we've got data for an EU customer and it's important to them that they’ve complied- adhered to the GDPR actually we need to know where that data ends up. We need to be in control of when it gets deleted. Um, and you know again what happens if something like a database goes down your entire app’s screwed, right, you are out as long as they’re out. Interestingly we built a project years ago for Now That's What I Call Music, and it was a mobile app and we actually used Pass. If you remember them? They were a database as a service kind of rest API thing and we used them just purely for scale 'cause we knew in that case there was no way we would be able to affordably scale for the kind of map downloads this thing was going to get.

Interviewer: Yeah

Participant B: And for the most part it worked great, and then Facebook bought them and then Facebook killed them. Then we were sat with an app that just did not work, and sure you can build around that but I think something as fundamental, as you know your database layer or something you probably want to keep in house. That being said, when you're starting out and you're building out new product, if that's going to give you that leg up, that lets you get to market and show your ideas and prove your ideas out first, then absolutely, right? it's just as soon as you start to see that there's a fit there, I’d say that’ll be why you'd be looking to ideally pull that internally as quickly as possible. But it does depend on on what you're outsourcing. Like I say we outsource error reporting to century. We outsource kind of metrics and analytics to datadog because those are the kind of things that you know if those things go out. It's inconvenient, but it doesn't stop our users and our customers getting their, their value from the product.

Interviewer: Yeah, that's really interesting to hear actually about the past Experience and like it some cases it could be just too risky to put all that Crucial part of an app into someone else’s hands, that's really interesting to hear. To finish up, I just wondered what your general thoughts of using the Jamstack are? So do you think there's potential for it to be used to create digital products, like the sort of things that you do, or does it even have a future?

Participant B: I think there, I mean front ends becoming more and more of the thing right? I'm not super o fae with the jam stack, but my my gut is it's for front-ender’s that don't want to have to learn back end. And I think that's a, a legit concern right? And if you know you can say right let's focus on user experience, let's focus on what's in front of the user and data actually doesn't matter that much to me I just need somewhere to put it. Then I definitely think there's a case for that. And you know again, like anything you need to pick the right technology for the problem that you're solving. And you need to prioritize based on what's important, so you know, if control of data is super important. You know, where would you put it? But I guess with the Jamstack stuff I assume you can self-host it? So I guess it then comes down to your product mix inside the team, right?

If you don't have the back end expertise and you can't afford to get it, or you just really do need to prototype and get to market, that's where I would see that being quite valuable, but I guess it would depend right? If you if you could see yourself ingesting a lot of data you kind of need to understand how you need to access their data and if the way they've designed their database or backend side of things is it's kind of compatible with that, you know? Mongo was very popular for a long time and, I'm absolutely sure it was her own naivety with the product, but we had something built on Mongo, and just the simplest of queries was just ridiculously slow. I think in this case we were ingesting large JSON From the Twitter fire host, I think, and we had what we thought was good indexing around it and stuff, but even just to get you know 10 or 11 camps across a large data set was actually quite slow. I'm sure if you knew what you were doing it wouldn't have been, but you know for when we actually look back into it. If we had just thrown everything into MySQL it would've run fine so I think it's understanding what your data is, how your data is going to scale, how you need to actually access that data, and make sure you don't end up in a situation where things you need to do with your data aren't really what the things were designed for. Because every day the database is different right?

Interviewer: Yeah, exactly. Wow yeah well thank you. I appreciate your answers. They were really helpful and yeah, thank you for doing this interview with me.

Participant B: No worries, thanks a lot Adam!

Interviewer: Cheers.

Appendix 6 – Participant C Interview Transcript

Interviewer: thank you for doing this interview with me. I guess the first question Will be a bit about you, what's your current job title and where do you work?

Participant C: Erm, so I'm the head of tech at The Design Lab / BC marketing, so we're sort of a- sort of one company but two companies.

Interviewer: Yeah, thank you and what sort of development industry do these companies work in, like is it an in-house company, an agency, freelance, product?

Participant C: As in our clients or? So the clients we work with are quite broad. we work with a lot of e-commerce companies. The Design Lab sort of historically works with a lot of lifestyle brands, so sort of bigger companies like Wella, H&M, and things like that

Interviewer: Ah nice, so you’re like an agency and you work with lots of-

Participant C: Yeah, so we are an agency ourselves and we basically work with brands and companies.

Interviewer: Erm, so yeah, I'll give you some context. So if you were approached to make a simple blog website, for a client, with maybe a blog and e-commerce so they could add products to their website. How would you approach this in terms of the technology and stacks you'd use?

Participant C: So we sort of work in the way that we've got certain sets of technologies that we use for projects, so we don't really go out and look for, look for what technology to use on an individual basis. We have sort of our tech stacks that we specialize in, and we sort of bring them out depending on the project. So if it would be a simple blog, it probably be a sort of WordPress backend with a Gatsby React frontend. If e-commerce were involved, it really depends on the size of the business so erm, If you're dealing with like a business that sells quite a lot of products and quite a big business, it will probably be a Magento backend. So Magento actually developed their own sort of framework to do PWAs, so it's called PWA Studio, which is what we tend to use to build up the frontend nowadays and erm, which also kind of depends again on what sort of budget we’re talking because, if the budget isn't, you know,

Interviewer: yeah,

Participant C: then we tend to just stick with a basic Magento store, which so I don't know if you know Magento?

Interviewer: I've heard about it but I don't know a lot, is it like using a WordPress headless API? And you put it into the front end? Or is it a front end and back end as well?

Participant C: No so Magento is sort of, sort of a PHP based CMS system. it's sort of the big, basically the market leader when it comes to e-commerce. So they’re owned by Adobe and it is sort of for your enterprise level e-commerce source.

Interviewer: Oh, I see. So does it- do you write all the templates in there and in that same application as well? Is it a well-rounded thing or do you have to add to it?

Participant C: It is well rounded in a way that yeah you build, you build your own sort of FEMA? If your, in Magento. Any sort of functionality that you would write, so for instance, let's say you want sort of a bespoke, erm, payment gateway that isn't out there, you can write in module so It's sort of a module based erm system.

Interviewer: That's really cool.

Participant C: So the great thing is, we've sort of started working with it alongside React. So like I said, they started doing PWA Studio, which is sort of really early days at the moment. So were one of the early adopters of it. So PWA studio is sort of a tool set and so it just provides things like hooks that allow you to really easily retrieve data from magenta through GraphQL.

Interviewer: Oh, that's really cool, so it kind of does turn it into that so you can put it into a front end.

Participant C: Yeah, yeah exactly so It sort of allows you to build a PWA react front end just with hooks that magenta provide through PWA studio.

Interviewer: Yeah, is that something you'd be more interested in using,with like a separate front end than using the old sort of Magento that made the front end for you?

Participant C: Yeah, so that is sort of what we're trying to aim towards in the future and sort of specialize in building, building these magenta PWA stores in instead of building the old school PHP, just you know it's where, it's where the future is going like I can't, I can't imagine anyone in one years time still doing. You know your static PHP website.

Interviewer: Yeah, it seems to be moving away from that. What are some of the benefits compared to using like this old sort of architecture where it's all like server side instead of pulling into and having an isolated front end and back end?

Participant C: erm I think I think the biggest benefits we're seeing is performance. So compared to sort of your older technologies the performance gains that you get from using something like React its... its massive. Also, what we're seeing now is like the more we're getting used to working in that Way it is really speeding up our development times, just because of sort of the component based way that React, React lets you work in. You can reuse a lot of the stuff that you've built in other projects really easily. Sort of in a new project that you might be starting.

Interviewer: Yeah, I found that as well in some of the products I've done the usability is so much better and I like the components are isolated so it makes working with them so much better.

Participant C: Yeah, yeah so for the Gatsby sort of WordPress combo that we've got going we've started developing our own starter theme which basically has almost all of the things that you will find on every WordPress website you build just baked in there and it's just a case of sort of setting off. Like making a couple of changes and you sort of good to go which is such a time saver.

Interviewer: That's really cool that is, yeah, especially when you're using such cool Technology like React or Gatsby then it makes it really nice to use. Like you said you used like React and WordPress for like a blog and stuff which is I think it's really cool. What were some of the technologies that you used to use before you used this combo and why did you change to the React and WordPress?

Participant C: Yes, so we did already use WordPress, sort of historically, which is kind of the reason why we stick with WordPress. We have sort of been looking at potentially moving away from WordPress just because it has a bit of a stigma and most people when you talk, especially clients, sort of people that aren't in the industry. If you talk to them about WordPress, they do sometimes tend to think like, oh, WordPress is just a bit of a shitty system.

You know it's not really something that we want our website built on.

Interviewer: They like turn their nose up at it

Participant C: Yeah, yeah exactly so we have been looking at potentially getting sort of a more enterprise level mechanism in there.

Interviewer: Oh yeah, that's really cool. So at the moment you use like the WordPress API to drag it into Gatsby-

Participant C: Yes, so WordPress has got a GraphQL plug in and then Gatsby, I think earlier this year brought out, I think it was earlier this year, they brought out their own sort of plug in to make it really easy to use Gatsby alongside Word Press, so it still uses GraphQL, but it sort of bakes in a lot of things, and that makes it a lot easier to use it alongside Gatsby.

Interviewer: Yeah, that's awesome. Have you had any drawbacks using this approach this? So yeah, I guess it is The Jamstack you're using here isn't it? Have you had any problems using this front end, isolated front end and then pulling in data from an API like the WordPress API?

Participant C: Erm, I think, I think the biggest problems we've got is sort Of getting everyone up to speed with it, because a lot of time with this new, well, it's not even that new anymore, But with the technologies especially for people that historically haven't really worked with something like that, it's quite a big step to sort of get your head around how everything works. So we've got, you know some of our front end developers have historically Used things like jQuery and HTML and CSS and then for them to move onto something like React, where It's sort of - It's almost, it's almost more like a back end developer than than it is a front end developer working. Because, you know, it's a lot more complicated than just writing plain old HTML, sass and and doing a bit of jQuery. I think that's sort of being our biggest drawback from from using this. It's just getting everyone up to speed and to be able to use this effectively

Interviewer: Yeah, really interesting, so I guess they're used to using like older methods of building websites, but maybe like a full serverside application like Laravel or Node and Express for example. Have you found anyones been like that?

Participant C: Yes, it's we historically did quite a lot of Laravel projects as well, and which we’re also sort of trying to move away from. Yeah, we're trying to, I don't know if you ever heard of something called Strapi?

Interviewer: Strapi oh is that the headless CMS?

Participant C: yes, it's yeah it's a it's a node based headless CMS and we're sort of trying to. Again, this is very early days, but we're trying to in the future replace anything that we used to do in Laravel with Strapi. Just again, because you know. Laravel works very well. and great as, you know, what it is. But it's not great to work with Laravel alongside with React. I know most people tend to go with Vue, but the problem that we also have is. You know you can't do everything, so we're trying to specialize in React and sort of not touch Vue as much as possible. Because in the in the past we have noticed where people started working with Vue on projects and then we've got other people working with React on projects and it's just not viable because you end up with sort of too many technologies for people to sort of stay up to date with.

Interviewer: Yeah, I guess it kind of creates that divide as well, like some people prefer using Vue and some people prefer using React and it's like it creates that weird divide.

Participant C: Yeah, yeah exactly and you know, the way you write something in Vue isn't the way you write something React. So you end up with people trying to sort of write things in a Vue way in React and the other way around as well.

Interviewer: So I know someone who really likes Laravel and they started using node, but they've turned node and express to be used exactly how they would Laravel. It's just really weird.

Participant C: How's that work?

Interviewer: I have no idea. He's created like MVCs and I don’t know, hes just turned it into Laravel. It's so Strange.

Participant C: Yeah, but I've seen people do that as well because we used to have a developer work for us and he sort of builds WordPress like he sort of converted WordPress into an MVC. You know it works. But the problem is that you end up with like a website that isn't documented so you can't go to someone and say like oh are you a WordPress developer? Can you work on this? But it made it hard for other people to manage then.

Interviewer: You mentioned you used to use Laravel for projects. What were some of the issues that you had with that that made you want to move away from it? Like did you have any like security issues, issues with performance or scaling or anything like that?

Participant C: It's a bit of a funny one because I think our biggest issue with Laravel is that we haven't got a Laravel developer at the moment. So a lot of these Laravel websites that we still manage we tend to get freelancers into work on just because the problem is, for someone that hasn't got Laravel experience or you know to hire someone with Laravel experience is- So the the position we're in now, it just makes it a bit of a waste. So we've, I think a really good thing this year is, sort of set out a plan for the next couple of years as in where we want to go with the business. And since that is sort of, like I mentioned before, heavily react based and with sort of the standard CMSs that I mentioned and Laravel isn't really part of that. You know, I think Laravel is a great system and I don't have any issues with it. It’s just another sort of technology that someone needs to maintain and learn. And since we've, I think we've got two Laravel projects at the moment, it's just not viable to hire a full-time person to manage these platforms.

Interviewer: Yeah, so you decided to move away from it?

Participant C: Yeah

Interviewer: Is there any technical reason for that? Like so I know you said that you don't have any- don't want to hire developers for it or like keep using it for projects but like is there a reason like any technical reason for why you want to move away from it and go with the React and API version?

Participant C: No not really to be honest like I think Laravel still has its Place like I think if you're building. You know, sort of a quite big a web application I think, I think Laravel does sort of shine quite well in that. Yeah, it’s really just a money factor. For businesses it is quite expensive to teach people new stuff because you know, if someone has to learn technology that they haven't used before, a lot goes into learning.

And plus they're not very effective when they start working with it. Yeah, plus, like I mentioned, it's just not viable for us to hire someone full time because there's just not enough work on these platforms on their sort of monthly basis to pay for someone's wage. And if we have to hire him, it for us, future wise, It's just more valuable to hire someone that sort of has the knowledge That we sort of want to use going forward.

Interviewer: Yeah, which is the React and API sort of thing?

Participant C: Yeah, yeah so with what we're sort of looking at. The moment is React based on what we use, you know, like I mentioned, we use Gatsby, and Next JS, which I'm sure you've heard of?

Interviewer: Definitely, I love using them.

Participant C: Yeah, yeah, same big fan, yeah, and then backends sort of WordPress, Magento and then Strapi. And I've got a call tomorrow with someone from, I don't know if you’ve ever heard of Kentico Kontents?

Interviewer: Oh it rings a bell

Participant C: So that's potentially the solution that we're that we're going to be using for a more enterprise level, sort of brochure websites to replace WordPress-

Interviewer: Oh like the bigger projects?

Participant C: Yeah.

Interviewer: Oh, that's really cool. Have you ever used- well I guess you have. You must have, have you ever used like any of the like ‘as a service’ product In the projects you do? You definitely have with headless CMSs or like Algolia, Auth0, Snipcart, Stripe, Fauna DB, fire base or like serverless Functions? Any of these things?

Participant C: As in Stripe the payment gateway?

Interviewer: yeah,

Participant C: Yeah so we use Stripe on a lot of Magento builds. We use Stripe, Brain Tree. Erm, what else do we use, Sage Pay? We're actually developing a payment gateway for BoxNets. Sort of a smaller, UK based payment provider. So we're doing a Magenta module for them at the moment. And besides that when it comes to sort of our server stack we tend to use Digital Ocean for all of our websites. Just simple droplets. And then we've got something called RunClouds. Which is sort of a control panel for your server, so it ties in with Digital Ocean through digital Ocean their API, and it's lets you handle sort of your lamp stacks set up really easily. And then when it comes to sort of our React projects, what we tend to do is just set up nodes on droplets and then run PM 2 to manage the processes.

Interviewer: Have you looked at like Netlify and Vercel? Because I think they're free? I think.

Participant C: Yeah, so I'm not sure to what extend they're free. I think they are free if you're-

Interviewer: oh sort of a personal developer

Participant C: yeah depending on the sort of data transfer per month. Because we have looked at Netlify and it's probably something that we might introduce in the future. The problem that we have at the moment is it adds in a new layer of sort of uncertainties, and which would mean that people need to be trained on how to use it. So due to mainly how busy we are at the moment, it is is something that is not viable for us.

Interviewer: Yeah It's cool though, for how busy you are to be able to have moved to this new sort of technology compared to the old stacks because I had a feeling that most sort of like agencies would be stuck using like the Laravels and the serverside languages and stuff. So that's really cool that you've managed to move into this new sort of technologies for while you are still so busy.

Participant C: Yeah you do see a lot of agencies that are stuck in their Old ways. But yeah, you get why? Why that is the case? Because it is just very expensive and in a way also very risky to move to these New technologies because you're sort of going to bump into issues that you've never come across before, and those at sort of add a new layer of frustrations within the team as well. Because you know, if you are very busy and you’re working with new technologies and you run into these problems, it definitely doesn't help when it comes to stress levels and things like that.

Interviewer: Yeah, with all these risks. What made you decide to do it then?

Participant C: Future proving ourselves and I think like I said before, I think realistically, if you want to, you know. I'm going to say be among the best but, you know, I'm not fooling myself in thinking that, but you know, if you want to be sort of a forward thinking agency and sort of be out there, then you need to be at the forefront of technologies and you can't be sort of struggling behind, especially considering nowadays that you know people can quite easily build simple websites themselves. You've you've got things like Weebly, Shopify etc. And those platforms are actually really good. I feel like for a simple website. People you know can use that and you can build something quite decent in that. If you're not moving forward then you know I think you'll end up standing still and like you're gonna end up in a position when I don't think you're going to be able to catch up.

Interviewer: That's really interesting. Yeah, really cool as well. Because you see a lot of like cool projects like, I think Starbucks brought out the like ordering system for their shops. And that's completely done using like React and it's really cool to see.

Participant C: They made it a PWA didn’t they?

Interviewer: Yeah, yeah so you can add it to your home screen. Really cool. Through the services you use, like the WordPress API and Headless CMSs do you see a lot of advantages of using these compared to building out the functionality in a backend? Instead of using like? A Laravel database to store all the content of a blog and using a backend instead?

Participant C: Erm, the perks I think of using WordPress is that, at least the perk that we see is that it allows you to use a plugin called Advanced Custom Fields, which makes it really easy to sort of build out your own data layer so it allows you to really easily in the back end create what the data is that the website owner will have to put in. The great thing that we notice about working in that way, like I said, because we've got a Gatsby starter sets up a lot of the sort of ACF fields, so just call them ACF components to make it easier, we can easily match up with our React components. Which then again, sort of provides us with another layer of being able to reuse stuff between projects. So um, for instance. If you've got um, really easy components where it's, for instance, a cart where you just have a few standard images and a button and a bit of content in the title.

Interviewer: Yeah

Participant C: If we've got an ACF field set up for that, we can easily copy that from one project to another project, then copy the component we've got setup for that. So we’re basically done. All we have to do is setup a bit of styling for it, you know, get things up to sort of the brand guidelines, but besides that you you're really saving yourself so much time compared to where you used to almost have to have to redevelop every PHP website you do.

Interviewer: Yeah, yeah, that's really cool. It kind of like adds a nice layer of comfort or security to the work that you do as well on a daily basis. Like you know, like you got like you're set up for projects that come your way.

Participant C: Yeah, yeah, exactly. And and what we started doing as well is, so our Gatsby starter is set up as a repo on GitHub, and then any sort of website that we are working on in the future that will use that Gatsby starter. I don't know if you know the templating function in GitHub? So GitHub has a templating function so we then set it up as a template repo of our original repo package and then it’s only the node package that we use and that needs updating that might affect anything in our application. We can then upstream to any other of our other sites, which again. Saves us a ton of time having, where, we would sort of sit and do a lot of updates. You just upstream the updates from sort of your main repo.

Interviewer: That is really cool.

Definitely something I need to look into, really cool.

Participant C: Yes, it's a real time saver, it’s something we only started doing an a couple of months ago, because it's not really something I've thought about before. But you know, working in the way that we're working now and- To be honest, I don't think it would be viable if you do sort of PHP or any other sort of old school language. But definitely with like the way we're working with Gatsby. It's really cool and A real time saver.

Interviewer: How do you find working with like static sites that Gatsby Produce and working with the CDNs do you see a lot of benefits through that?

Participant C: How do you mean static sites and CDN?

Interviewer: When you're using- So, like Gatsby, it compiles all the web pages into like just plain HTML.

Participant C: Yeah.

Interviewer: You see a lot of benefits through keeping it as. Plain HTML when I like I, I guess you see advantages through like caching and stuff compared to all the content on the website being server side rendered rendered.

Participant C: Yeah, yeah I think I think the biggest, the biggest benefit we're seeing is just the performance because it is just so fast and especially with sort of a lot of the tools that Gatsby has baked in, like the image optimization, so the smart preloading of assets is just such a massive boost in performance compared to what you used to see. So one of the things that we've recently started looking into as well is Google has something called Web core Vitals, which is actually something That they use for their page ranking update. So with your older technologies, it's very difficult to to optimize your websites for these web core vitals. For instance, you know optimizing images to be web P images, If you doing that in PHP is not as straightforward as it is in, for instance, Gatsby where you just have a component out of the box that optimizes the images for you.

Interviewer: Is it like the Google Lighthouse scores.

Participant C: Yeah, so I think web core vitals is something new,
but I do think they have included it in Lighthouse now.
Sort of that that data group.

Interviewer: Yeah, that's really cool and I guess I'll finish off on the question:
All these technologies that you're moving towards are under like
the Jamstack sort of name.

Participant C: mhm.

Interviewer: Do you think it's got a future? And how do you think it ranks up
compared to these big monolithic architectures like Laravel and stuff?

Participant C: I definitely think I- I think it's the only future like
I feel like that unless Laravel. Unless Laravel is going to change in certain
ways and. I think it's going to, um. Get overtaken by these newer
technologies because you know, I know that Vue. You can sort of use Vue
heavily along side Laravel, Laravel is probably there to stay to be honest, but I
think a lot of so if you're older technology stacks are really going to struggle
keeping up.

Interviewer: Yeah, I think I it's got largely to do with the fact that.
While, still like using like Node, Express and Laravel you have to scale and
optimize and make sure that your own backends are performant.
Whereas if you use like the WordPress API, It's all done for you.
You can keep scaling like no matter the size the project gets.

Participant C: Yeah, that's yeah, that's a good point

Interviewer: and because they're isolated the front end's got nothing
to do with the back end and it's just I really like that.
I think that's the biggest thing, you know, really cool yeah.
So thank you. It's really interesting here. Yeah.
Yeah, thank you. That brings us to the end of the interview.
I really appreciate you answering these questions for me and it's been very
helpful.

Participant C: That's alright mate, happy to help.

Interviewer: Yeah, so I'll end the recording.

…

I’ve noticed developers for in-house companies often get stuck in a bit of a Comfort zone 'cause I've noticed that with a lot of developers, you know it's- The industry we're in is very much Constantly changing so it is very difficult and especially the older you get 'cause I think this is something that a lot of older developers actually struggle with, you constantly need to be learning because if you're not constantly learning in this industry. Then before you know it, you're not. You're not relevant.

Interviewer: You're so far behind and there's so much to catch up on.

Participant C: And I think that is what tends to be the the difference between agency developers and sort of in-house developers, the work for a specific business, 'cause if it's if it's a business doing their own project, a lot of the time it's a it's a historical sort of project that probably has been built 10 years ago and they just sort of keep stacking on top of that. just sort of sticking together with a bit of duct tape.

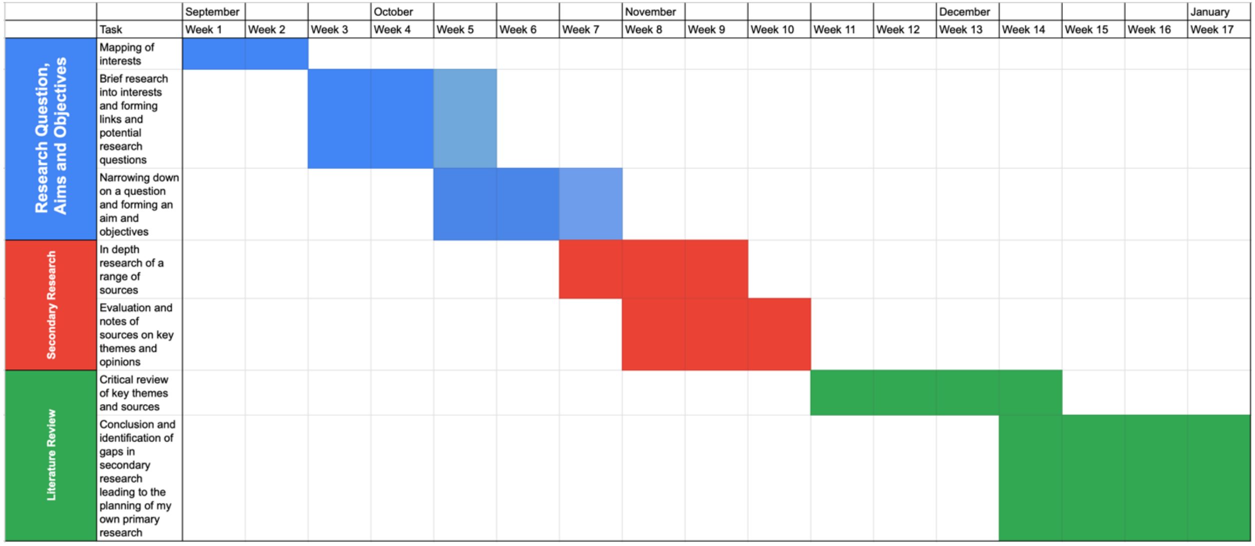
Interviewer: and like you mentioned, In order to change that, it means they need to hire new developers and train developers on new technology. They push those resources to that when they could be using the resources just to build upon their old stack.

Participant C: Yeah, yeah. 'cause it is like I say it's very costly to do those trainings.

Interviewer: and like you're stuck in a continuous loop. And I guess, I guess from an agency POV as well, 'cause you're competing with other agencies, you need to keep at the forefront of it. So you need you need to invest that time into new technology and stuff which is really cool because you get to keep learning.

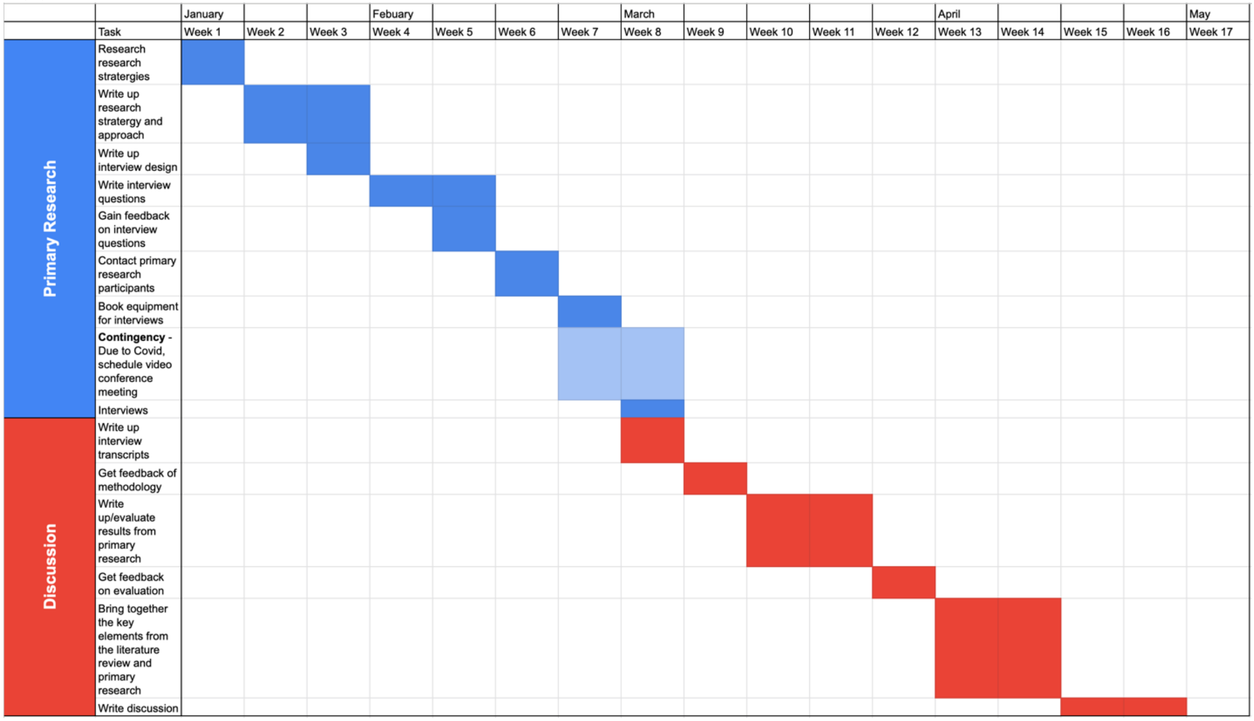
Participant C: Exactly

Appendix 7 - Semester 1 Gantt chart



		September			October				November					December				January
	Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17
Research Question, Aims and Objectives	Mapping of interests	🟦	🟦															
	Brief research into interests and forming links and potential research questions			🟦	🟦	🟦												
	Narrowing down on a question and forming an aim and objectives					🟦	🟦	🟦										
Secondary Research	In depth research of a range of sources							🟥	🟥	🟥								
	Evaluation and notes of sources on key themes and opinions								🟥	🟥	🟥							
Literature Review	Critical review of key themes and sources											🟩	🟩	🟩	🟩			
	Conclusion and identification of gaps in secondary research leading to the planning of my own primary research														🟩	🟩	🟩	🟩

Appendix 8 - Semester 2 Gantt chart



		January			Febuary				March					April				May
	Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17
Primary Research	Research research stratergies	🟦																
	Write up research strategy and approach		🟦	🟦														
	Write up interview design			🟦														
	Write interview questions				🟦	🟦												
	Gain feedback on interview questions					🟦												
	Contact primary research participants						🟦											
	Book equipment for interviews							🟦										
	Contingency - Due to Covid, schedule video conference meeting							🟦	🟦									
	Interviews								🟦									
Discussion	Write up interview transcripts								🟥									
	Get feedback of methodology									🟥								
	Write up/evaluate results from primary research										🟥	🟥						
	Get feedback on evaluation												🟥					
	Bring together the key elements from the literature review and primary research													🟥	🟥			
	Write discussion															🟥	🟥	