**­­­300/303COM Detailed project proposal**

The objective of the detail project proposal is to help you refine your general research question down to a well-focused and achievable piece of practical research work.

The first section: “Defining your research project” focuses on your research question and the plan for conducting your primary method. The second section: “Abstract and Literature Review” is to help you identify current academic sources of literature that are highly relevant to your project and to help you get a head-start in producing your literature review.

Your detailed project proposal will be graded in the second semester – however, it is highly recommended that you submit it by the end of the first semester (04/01/2016) in order to obtain detailed supervisor feedback on your project.

There is no suggested word length for the detailed proposal – although 2000-2500 words would be in order.

The Detailed Project Proposal is worth 20% of the project mark.

**300/303COM Detailed Project Proposal**

|  |  |
| --- | --- |
| First Name: | Robert |
| Last Name: | Rigler |
| Student Number: | 4939377 |
| Supervisor: | Dr. Faiyaz Doctor |

Section one: Defining your research Project

**1.1 Detailed research question**

**Help:** Your detailed research question is the statement of a problem within the computing domain which you will address in your project. Refining the research question involves narrowing down an initial question until it is answerable using a primary research method(s) that you will conduct during the time of your project. The refined research question must not be so general that it is answerable with a yes or no answer. It must not be so broad that you would be unable to achieve a solution during your project. The key to this is BEING SPECIFIC: Narrow down the method or technology you will use, narrow down the group that the question refers to (localize a general question) If the project is still ‘too big’, can you think of a way to work on a part of the problem? Avoid using words that cannot be measured, by you, without a huge research budget e.g. 'effects on society', 'effects on business'. *Example:* The initial question "Does cloud computing effect business" needs narrowing down *(for a start the answer is yes) W*hat is meant by cloud computing? Or 'effect'? Or 'business', in this question? Refining this first question will involve narrowing it down to something you, personally, can measure. A refined version of this question might be: "Does implementing a cloud based voting system improve the speed of decision making in a small company in Coventry?" This refined question is implementable: You can now identify a small company to work with, document their current decision making processes, implement a cloud based voting system, compare decision making speeds over a limited time period (say 1 month) and evaluate your findings. *A small piece of genuinely new knowledge is produced.*

|  |
| --- |
| Does developing Android mobile applications in the cross-platform development framework React-Native produce any quantitative benefits in terms of performance, development time, cost, and stability compared to developing applications exclusively in the platform native language (Java)? |

**1.2 Keywords**

**Help:** Include up to 6 keywords separated by a semi-colon; what keywords are appropriate to describe your project in an online database like Google Scholar? Keywords should include the general research area and the specific technologies you will be working with. *Example.* A project that proposes a novel way of visualising large amounts of twitter feed data may have the keywords: Data visualisation; twitter; hashtags; database design; graphics libraries. For further help take a look at the ACM keywords list <http://www.computer.org/portal/web/publications/acmtaxonomy>

|  |
| --- |
| react-native; android; performance; development-time; mobile-application; comparison; |

**1.3 Project title**

**Help:** The project title is a statement based on your detailed research question. For example, the research question *'to what extent does a mobile application reduce the number of errors made in class registers at Coventry University in comparison to current paper based registers'* may be stated in the project title*: "A Wi-Fi driven mobile application for large group registers using iBeacons".*

|  |
| --- |
| A comparative analysis of developing Android mobile applications using the native Android language Java, compared to using the cross-platform development framework React-Native. |

**1.4 Client, Audience and Motivation:**

**Help:** Why is this project important? To whom is this project important? A research project must address a research question that generates a small piece of new knowledge. This new knowledge must be important to a named group or to a specific client (such as a company, an academic audience, policy makers, people with disabilities) to make it worthwhile carrying out. This is the ***motivation*** for your project. In this section you should address who will benefit from your findings and how they will benefit. Example: If you intend to demonstrate that a mobile application that automates class registers at Coventry University will be more efficient than paper based registers - the group who would be interested in knowing/applying these findings would be both academic and administrative staff at Coventry University and they would benefit by time saved and a reduction in their administrative workload. If you are making a business case for an organization explain how the organisation will benefit from your findings.

|  |
| --- |
| **Motivation**  React-Native is a relatively new cross-platform mobile application development framework for Android and iOS. Its combination of the popular JavaScript-based web-design framework: React, and Platform Native development has gained a huge amount of interest from hobbyists and professionals alike, many of whom believe this new framework to be a superior method of creating mobile applications. However, despite its popularity, very little formal research has been done and almost no quantitative data has been produced to properly compare the two types of development. There is a large amount of uncertainty concerning the performance of React-Native, and whether the performance trade-off is worthwhile when also considering the reduced development time and cross-platform aspects of React-Native.  The goal of this project is to produce accurate, unbiased analytical data and use it to objectively compare the two types of development. This data should be able to alleviate some of the uncertainty surrounding the subject and act as a reliable resource for individuals, businesses, and development companies when assessing the usefulness of the React-Native framework.  **Audience**  This research and the conclusions therein are intended for use by any individual or company who are deciding whether to using Native application development or using React-Native. The data I produce aims to alleviate the ambiguity between the two development techniques.  Specifically, I am looking into the 4 areas of app development which are most relevant to businesses: Performance, Cost, Stability, and Development-Time.  Depending on the application in question and the resources available (time and budget, etc...), the choice between using traditional app development and using React-Native becomes difficult as there is little data supporting either side of the argument. |

**1.5 Primary Research Plan**

**Help:** This is the plan as to how you will go about answering your detailed research question - It must include a primary research method (an extended literature review is not an acceptable primary method). Think and plan logically. Primary methods may include experiments, applications or software demonstrators, process models, surveys, analysis of generated data …  
  
Example: In the class register example above "to what extent does a mobile application reduce the number of errors made in class registers at Coventry University in comparison to current paper based registers" - the research plan may involve: 1) Collecting and analysing paper based registers in a given class on five occasions. 2) Identifying the error rate average on these occasions 3) Designing and implementing a mobile application that automatically records attendance in class. 4) Deploying the application in the class on five occasions. 5) Identifying the error rate average of the mobile application on these occasions. 6) Comparison of data and summary of findings.

|  |
| --- |
| The primary research plan for this project will be to create two identical mobile applications and then compare them to each other. One of these applications will be developed using the traditional Java method, the other will be using the cross-platform framework React-Native.  Firstly I am going to design the application, then I will reduce this design into a set of *features*, and those features will be broken down into *tasks.* Using this structure I can then proceed to develop the applications, and record my progress and the relevant metrics for each task and feature, thus producing a set of data for each application which can be directly compared to each other.  The following areas will be compared between the two applications:   * Performance * Development-time * Stability * Cost (Derived)   **Performance**  For most features, a performance metric will be directly measurable. E.g. For measuring the performance of animations, the metric would be average FPS (Frames per Second) over the duration of the animation.  For large list manipulation I.e. (Removing an item from the middle of a large list), the metric would be TTD (Time to Draw) which is the time it took to remove the element from the list and redraw the screen with the new data.  **Development-Time**  For the development-time metric, I will record the time it took to complete a feature. A feature is "Completed" when it passes the set of unit tests that have been created for that feature (Both applications will use the same unit tests, however, they will be implemented using the platform respective unit-testing framework).  Being very familiar with both Java and React-Native, I will be able to efficiently implement both applications.  **Stability**  Stability is of great importance when creating any application, I will measure stability by recording each time a feature implementation cause a previously successful unit test to fail. Any bugs not caught by the unit tests will also be recorded in this metric.  I can produce the following metrics:   * Bugs (number of reworks) per feature ( **B/F** ). * Broken lines of code(SLOC) per feature. * Time spent fixing Bugs.   All of the above three metrics will also be produced for the application as a whole, rather than just on a per feature basis. |

This is the end of section one.

Section Two: abstract and Literature review (1500 words suggested)

**2.1 Abstract**

**Help:** An abstract is a short summary of a research project that enables other researchers to know if your report or research paper is relevant to them without reading the whole report. It is usually written retrospectively so that it can include findings and results. It is fully expected that you will rewrite your abstract when you come to write your final paper. For now, you should write an abstract of about 250 words that define the project described in section one. Before writing your abstract you MUST read some abstracts from conference or journal papers on *Google Scholar* or from *portal.acm.org* (to understand their style) and then provide your own abstract that outlines what your question is and what you 'did' to answer it.

|  |
| --- |
| React-Native is a cross-platform mobile application development framework created by Facebook in 2015. It allows a developer to simultaneously create mobile applications for both the Android and iOS platforms in JavaScript instead of the platform native language (Java or Objective-C respectively). React-Native’s popularity stems from its intelligent combination of native and non-native code using “Bridges”. Bridges are a way of exposing native APIs to the JavaScript thread, allowing the utilization of the platforms’ high-performance native modules. This allows React-Native to achieve comparable performance to purely native applications whilst using the arguably simpler application design structure of React and JavaScript as well as having the benefit of creating applications for both platforms at once. React-Native operates very differently to other cross-platform technologies such as Cordova and Xamarin, which use an embedded web browser to render the application in HTML and CSS as if it were a normal web application. Whilst this method is considered cross-platform it sacrifices performance and freedom of design. Those who use React-Native claim to have seen reduced development time and cost, whilst sacrificing very little in terms of performance. These claims, however, are backed by very little evidence and almost no statistical data has been produced. Most of the available data comes from blogs that have a little objective evaluation of the development methods, or do not consider any special business requisites for the application, such as high-performance requirements. Businesses and individuals who are deciding which method to use when creating a new mobile application will find it difficult to form an accurate, unbiased decision using the data available, and may waste a large amount of time and money using a development method that is entirely unsuitable for their needs. The primary objective of this research is to fill the substantial gap in the data used to compare the two development methods. Using this data, we can accurately assess if using React-Native to develop mobile applications produces any substantial improvement in performance, development time, cost, and stability compared to using purely platform native development. These are arguably the most important areas of concern for any business looking to develop a mobile application. The research involves creating two identical Android mobile applications and individually assessing each aspect of the application. The measurements will be taken for each feature of the application, so to produce a fine-grained analysis as well as considering the two methods on a broader scale. |

**2.2 Initial/Mini Literature Review (500 words – 750 words)**

**Help:** A literature review is a select analysis of current existing research which is relevant to your topic, showing how it relates to your investigation. It explains and justifies how your investigation may help answer some of the questions or gaps in this area of research. A literature review is not a straightforward summary of everything you have read on the topic and it is not a chronological description of what was discovered in your field. Use your literature review to:

• compare and contrast different authors' views on an issue  
• criticise aspects of methodology, note areas in which authors are in disagreement  
• highlight exemplary studies  
• highlight gaps in research  
• show how your study relates to previous studies

|  |
| --- |
| My research focuses on the statistical benefits of creating mobile applications in React-Native, as opposed to using purely native development. The aim of this research is to be able to compare both types of development and assess each of them in key areas of mobile development. These areas include: Performance, Development Time, Cost, and Stability. React-Native was created by Facebook in 2015, and the framework is currently still in an ‘Unstable’ state as an official release version is yet to be produced (The current version is 0.42). Despite its relative youth compared to native mobile application development, React-Native has exceeded google search trends for both “android development”, and “iOS development”.  **Performance**  Generally, most authors agree that there is a noticeable performance deficit when using React-Native. However, there is some ambiguity over where this deficit occurs. According to Vytenis Narusis, a Junior mobile developer at Devbridge, the greatest performance impact comes from the JavaScript Language itself: “It’s natural that JavaScript code is not as efficient for calculation-intensive tasks, and there is an overhead when JavaScript is controlling native elements.” (Narusis), He also states that “In performance, React Native stays behind an optimised native application.”. No statistical data has been given to support this argument, one could surmise that a *low-level* compiled language such as Java, will perform better than a JIT (Just in time) compiled language such as JavaScript, however this is purely conjecture as it is impossible to know the optimisations made in either language, and often performance will have a greater dependence on *how* a problem is implemented, rather than in what language it is implemented.  Others argue that the performance deficit stems from the Single Threaded nature of JavaScript. React-Native runs all JavaScript code on a single separate thread, as JavaScript cannot be declaratively parallelised. React-Native utilises Native API’s and components by passing data from the JavaScript thread to the Native thread, this is known as a *bridge.* Many authors indicate that these bridges are the source of React-Natives performance bottle-necks. In the blog article *Performance Limitations Of React Native And How To Overcome Them*, by Tal Kol, a Start-up founder and Senior Developer at Wix.com, he states that: “*The performance bottleneck often occurs when we move from one realm to the other. In order to architect performant React Native apps, we must keep passes over the bridge to a minimum.”* (Kol, 2016). He refers to the Native threads and JavaScript thread as separate *Realms,* he advocates to minimise the use of Bridges. This is a better indicator of where the performance problem lies, the use of *bridges* in React-Native is unavoidable, and thus careful optimisation is imperative to both the framework itself, and the developer using React-Native. The *bridging* performance bottleneck seems to be especially common when attempting to create animations solely using JavaScript. Sriraman Paneer (a mobile application developer a Witworks), states in his article “*What we learned after using React Native for a year*” that “Animation APIs run all the processes on the JS Thread. Consequently, we had faced some performance issues while using these APIs.” (Panneer). A similar claim is made by Robin Chen, a senior developer at Discord, who chose to migrate their VOIP application from Native to React-Native. “The Animated library also cannot deliver the animations as smooth as the native while doing heavy duty works on the JS thread.” (Chen).  After reading many articles, and using the sources that I have given here, I am left with more questions about React-Native performance that when I began. It seems clear that there is a definitive decrease in performance when using React-Native, however, it is not clear about how large or small that sacrifice is. E.g. How much of a difference In FPS is there between Animations in Native and React-Native? How much longer would an intensive calculation take using JavaScript than Java? Does the problem lie in the developers implementation rather than the framework? My research aims to fill these gaps in the knowledge, as the ambiguous information available is simply not enough to make an informed decision about Whether to use React-Native of not.  **Development-Time**  An important factor of creating any application is development-time, especially for prototyping new features, or creating an MVP (Minimum viable product). How quickly an application can be developed may be quintessential to a business model, or may be entirely irrelevant. In cases, development-time has a direct correlation to the cost of developing that application.  Being a Hybrid application framework (i.e. Can develop applications for both iOS and Android simultaneously, using the same code base), one would expect that the development time for a React-Native to be half that of developing a native application for both platforms. However, my research will only be focusing on the Android platform, so whilst the cross-platform development is of great importance, I must primarily focus on single platform development.  Greg Gomez, a Project Manager at 3-Sided-Cube said that: “*The initial benefit that strikes me is the ability to prototype with even more rapidity than previously. React-Native will offer developers unparalleled speed in developing high fidelity prototypes, so ideas can be validated, assumptions debunked and implementations battle-tested in record time.”* (Gomez, 2017).This is certainly high praise for React-Native, however he has not stated why React-Native had a faster development-time. He later advocates the re-usability aspect of React-Native, *“Developing a component once allows us to reuse it easily in other projects, meaning that over time you’ll amass a nice library of drop-in components to build prototypes from. It means that time spent re-inventing the wheel, can be spent solving your client’s problems and delivering value.”*. This is a far more important than it seems, React, and therefore React-Native, has a component based architecture, which advocates creating components of the application independently. This allows for a minimal amount of dependency between different parts of the application, meaning that a developer could develop generic components that can be used in anywhere in any application, or perhaps use open-source components that others have developed. Whilst this is still possible in Native development, it is not possible to the extent that React-Native allows composition of components.  In Native development, every time that new code is implemented, changed or updated, the entire application needs to be re-compiled, if the application is large this could take quite some time, and quickly becomes aggravating, especially when the code changes are minimal. React-Native being able to take advantage of both Native and non-native JavaScript, has an impressive feature called *Hot-Reloading*. *Hot-Reloading,* allows any changes to the JavaScript code to be almost instantaneously reflected in the application; “*React Native’s feedback loop is bewitchingly low. It takes less than one or two seconds between you saving a file and seeing the change in your app. That’s easily ten times less than the typical Build and Run cycle we’re used to in (Native development).”* (Elkin, 2016). This does not apply to Native code, as this will need to be recompiled. This feature alone could make an enormous impact on the speed of the application development.  Naturally, not all developers will be able to immediately become productive with React-Native, traditional Native developers with little experience with JavaScript or React may struggle. “*As native developers we found the learning curve of React Native pretty high, as it may take up to a month until one could start feeling comfortable developing with this framework.”* ("React Native: Is It The End Of Native Development?", 2016). This may entirely negate the benefits of rapid prototyping and Hot-Reloading, having to potentially delay development for a month or more has the potential to incur serious costs. This however, is not true for all situations, Developers that are familiar with JavaScript, and more importantly familiar with the React framework will take to React-Native far quicker that purely Native developers.  Essentially, my research will focus on single platform development, specifically the Android platform. I aim to provide an analysis into the time taken to develop different parts of a mobile-application. Depending on the results I will offer reasons as to why there was or was not a difference in development time. “Assuming equal skill level on the part of the developer, it shouldn’t take longer to build an app on one platform or the other,” (Chernov, 2013), said  Joe Chernov, VP of marketing on behalf of Kinvey’s engineering team. Perhaps React-Native has some particularly obtuse API’s that hinder development time for particular areas of development. Perhaps the only reason React-Native has faster development time is because of Hot-Reloading. These are the types of questions that are missing from the current information available.  **Cost**  The cost of developing mobile applications is very high, many businesses will be looking to minimise the cost of creating an application whilst still sacrificing as little as possible in other areas. Whilst I have separated cost into a separate category, it is strongly tied to the amount of time it takes to develop an application, and so for the sake of clarity, some points from the previous section may be repeated.  Bernard Kohan of Comentum created an estimated ball-park cost of development for creating an MVP application (Minimum Viable Product) for both native and hybrid applications. He estimated that the cost of creating a native MVP mobile application for just one platform to be approximately one-hundred and fifty-two thousand dollars ($152,000), and creating the same native application for both iOS and Android platforms would cost two-hundred and fifty-one thousand dollars ($251,000). He also estimated the development cost for a hybrid application, i.e. An application that can be simultaneously developed on both the iOS platform and Android platform, to be one-hundred and sixty-two thousand dollars ($162,500). Either way, this is a substantial investment, Kohan goes on to state that: “If the developer needs to build the app for all platforms, and the app needs to be built using the native programming language of each platform, the cost of development will be doubled or tripled because of the amount of time that is needed to build the app using different programming languages of each platform.” (Kohan). The premise that hybrid development is almost half the price of native development is backed up by Keerti, a developer at Walmart Labs, who created the Walmart application in React-Native, he states that; “The defining feature of React Native, and arguably its best selling point, is that it’s cross platform — allowing for simultaneous development on iOS and Android by the same team, which can cut labor costs roughly in half.” (Keerti). With the price of Native application development being so high, it is natural that Hybrid development would seem the more attractive choice. Others, such as [NAME HERE] would dispute this claim, stating that: “*As native developers we found the learning curve of React Native pretty high, as it may take up to a month until one could start feeling comfortable developing with this framework.”* ("React Native: Is It The End Of Native Development?", 2016). Whilst this is not a direct opposition to the previous argument, I do feel that this is a relevant piece of information to consider. If it takes up to a month and possibly more for a native developer to familiarize themselves with the framework then this may effectively negate any improvements in development time, and thus an implicit reduction in cost. However this is a fairly subjective claim, even though the React-Native framework is relatively new, there are those who have been using the Web framework React for quite some time, and they would take far quicker to acclimatize themselves to using React-Native.  Ultimately, cost is a category that I will only be able to estimate by comparing the amount of time It will take for me to complete the development of both applications and average developer salaries, so it is more than likely that my research will not yield any results that are different from the sources above. However, I may be able to ascertain which parts of the application development would cost the most, and produce a relationship between cost, development-time and performance, which I believe would be a valuable new piece of information.  **Stability**  The stability of an application has the ability to significantly impact the cost and development-time of an application, so it must be closely monitored. In the article “Why I’m not a React-Native developer” by Ariel Elkin, she states that: *“JavaScript lacks these safeguards against programmer error, making preventable runtime crashes and preventable programmer errors part of your routine.”* (Elkin, 2016). Implying that the use JavaScript is a disadvantage and will adversely affect the stability of an application. Mostly this is a reference to the absence of type checking in JavaScript, and the ability to send data over a bridge from JavaScript to Native code that has not been type-checked, which will in most cases cause a fatal runtime error. These “Preventable programmer errors” that Elkin suggests may however be negated by the decreased development time in React-Native. Earlier in the article Elkin also stated that: “*React Native’s feedback loop is bewitchingly low. It takes less than one or two seconds between you saving a file and seeing the change in your app. That’s easily ten times less than the typical Build and Run cycle we’re used to in Xcode.”* (Elkin, 2016). Do these two points contradict each other? Is an error produced when compiling an application different to when an error is found when testing an application? Whilst the compiled application may catch the error quicker (Only if the time it takes to compile the application is low), it may only be relevant if those bugs are deployed into the production application. it seems that React-Native may be more vulnerable to developer error than native applications, and thus React-Native has a greater dependence on Unit-Testing and User-testing than Native development.  React-Native is a relatively new Framework, made public in 2015, but it still does not have an official release as the current version is 0.42 and it may not get a stable 1.0 release for some time. However, being an open source framework, when bugs are found, they are rapidly fixed and integrated. “*While it is good that they bring more features and push the framework towards maturity, often they bring breaking changes as well*” ("React Native: Is It The End Of Native Development?", 2016). Whilst new features and bugs are rapidly integrated into the framework, there’s also the possibility of producing breaking changes, i.e. Major changes in the API that will cause previous code to break. This may be viewed as a positive light, when assuming that breaking changes are only introduced when they are a vast improvement, however, depending on an applications reliance on these features, an equally large amount of time may have to spent on refactoring and redesigning code to be able to take advantage of these latest features, which in-turn would increase the costs of supporting the application. This quote from the same article validates this assumption: “*These changes sometimes where a burden to overcome, as we had people spending a lot of time fixing things when upgrading the framework.*” ("React Native: Is It The End Of Native Development?", 2016).  You may have noticed that all my sources have come from either, independent blogs, or from a company blog. This is because there are no existing scientific reviews, or articles covering React-Native, there were however a plethora of blogs and web-based articles. Many of these I had to exclude as sources entirely because of their sensationalist or inaccurate nature. I also chose to exclude any sources that were over one year old, this is because of the quick release schedule of react-native, the version of React-Native that existed a year ago, is entirely different from the one available today, and thus most of the information in the sources is similarly out-dated.  In summary, there is a large amount of informal data surrounding the advantages and disadvantages of using React-Native, apart from cost however, there is no statistical data available to create an accurate depiction of the differences between Native and non-Native development. My research aims to produce this statistical data. |

**2.3 Bibliography (key texts for your literature review)**

**Help:** Please provide references, in correct Harvard style, for at least three key texts that have informed your literature review. If you are implementing an application, select texts which demonstrate how other researchers have tackled similar implementations? The references should be recent and sufficiently technical or academic. Your markers will be looking for you to identify technical reports, conference papers, journal papers, and recent text books. Avoid *Wikipedia* entries, newspaper reports that do not cite sources, and general or introductory texts.

|  |
| --- |
| * Elkin, Ariel. (2016). *Why I'm Not A React Native Developer* - *Arielelkin.github.io [online]* . Available at: <https://arielelkin.github.io/articles/why-im-not-a-react-native-developer.html> /[Accessed 1 Mar. 2017]. * Novoda . 2017 *React Native: Is It The End Of Native Development? – Novoda.com*. [online] . Available at: [https://www.novoda.com/blog/is-it-the-end-of-native-development /](https://www.novoda.com/blog/is-it-the-end-of-native-development%20/) [Accessed 2 Mar. 2017]. * Kol, Tal. 2016. *Performance Limitations Of React Native And How To Overcome Them*. *Medium*.com. [online] Available at: <https://medium.com/@talkol/performance-limitations-of-react-native-and-how-to-overcome-them-947630d7f440#.ayr73y94x/> [Accessed 3 Mar. 2017.] * Panneer, Sriraman. 2016 *What We Learned After Using React Native For A Year - Hashnode*. *Hashnode.com*. [online] Available at: <https://hashnode.com/post/what-we-learned-after-using-react-native-for-a-year-civdr8zv6058l3853wqud7hqp/> [Accessed: 3 Mar. 2017] * Narusis, Vytenis. 2016. *Pros And Cons Of React Native - A Course Of Building Native Apps With Javascript*. - *Devbridge.com*. [online] Available at: <https://www.devbridge.com/articles/pros-cons-of-react-native-crash-course/> [Accessed at: 2 Mar. 2017.] * Keerti. 2016 *React Native At Walmartlabs – Walmartlabs*. *Medium*.com [online] Available at: <https://medium.com/walmartlabs/react-native-at-walmartlabs-cdd140589560#.e8esee7za> [Accessed 4 Mar. 2017.] * Kohan, Bernard.2015. *Mobile App, Iphone App Development Cost, Pricing*. *Comentum.com*. [online] Available at: <http://www.comentum.com/mobile-app-development-cost.html> [Accessed 4 Mar. 2017.] * Chen, Robin. 2016 *Using React-Native: One year later*, *Discord.com. [online] Available at:*<https://blog.discordapp.com/using-react-native-one-year-later-91fd5e949933#.sqsegx7nm> *[Accessed 1 Mar. 2017]* * Gomez, G. (2017). *React-Native: The Good, The Bad & The Ugly - 3 SIDED CUBE*. [online] 3 SIDED CUBE. Available at: <https://3sidedcube.com/blog/2016/04/react-native-the-good-the-bad-the-ugly/> [Accessed 8 Mar. 2017]. |

This is the end of section two.