X5GON Mobile: Application Development and Information Retrieval from a legal and sustainability perspective

Patrick Wu 17134402

18 March 2020

In this day and age, it is hard to remember performing a daily routine without using a mobile application. With these applications, we are effectively interacting with all sorts of information, including providing our usage habit data in return for more accurate content recommendation. Human interactions with information can be divided into two main categories, namely active retrieval and information feeding(Tsai et al., 2010). An example of active retrieval might be searching while-as browsing social media represents information feeding. X5GON Mobile, as my main project for System Engineering course, provides users with both ways of acquiring information. Therefore, in this essay, we aim to discuss legal and sustainability issues considered when developing and delivering a mobile application that involves deeply in user information and client's server data.

Potential Liability

In regards to potential liabilities, it can be considered from standpoints of both a developer who will eventually be delivering the application to clients and a service provider who hosts services and enters an agreement with application users.

Developer's Liability

From a developer's perspective, we have the responsibility to ensure clients' server resources, data and internal Application Programming Interfaces (APIs) provided to us is handled with caution. The usual lecture in `sudo` software states that "With great power comes with great responsibility" (Miller, Coggeshall, and Spencer, 1980). Access to remote Amazon Web Services (AWS) servers hosting `X5Learn` are provided by our clients such that we can deploy, manage and test applications more efficiently. These privilege yields us with great flexibility on their servers, and at the same time, a legal liability not to use these resources and power for purposes other than application development.

It is also a developer's liability to proceed with caution when it comes to server-logon and API credentials. A study from North Carolina University (Meli, Mcniece, and Reaves, 2019) on leaked information on GitHub, a source-hosting platform, showed there are more than $200,000$ API keys or secrets publicly available and accessible by anyone on the Internet. This phenomena showed a lack of circumspection on client data and possibly ignorance of the legal liability as a developer. Our team utilised `git` features, specifically `gitignore` and `gitsubmodules` extensively in our project to ignore sensitive configuration files, and at the same time incorporating the client's private back-end `X5Learn` repository in our project through remote `sumbodules` support.

Service Provider's Liability

Whilst developing our application, our client left us with high feasibility of designing the whole application from scratch. Therefore, we constantly put ourselves in a service provider's viewpoint to explore the liability to end-users with regards to our project.

At the core of our mobile application is a Software-as-a-Service (SaaS) model, where the proprietary `X5Learn` platform fetches learning materials from public platforms worldwide and present them to mobile users with its own analysis. Thus, we believe our application has the liability to cite and use the source contents according to their original license and specification. Take Massachusetts Institute of Technology's (MIT's) Open Course Ware, for example, all its lecture videos and contents come with Creative Commons Attribution-NonCommercial-ShareAlike 4.0 License (CC BY-NC-SA 4.0), which clearly states attributions to the original content (*MIT Open Courseware Privacy and Term of Use,* 2020) and using the same license upon re-distribution. We specified such material and made these licenses easily accessible for users in the application.

In the meantime, we also have the critical responsibility of clearly differentiating the original third-party content and the derived summary that comes from the Machine Learning (ML) algorithm. On that note, we used multiple design techniques, including but not limited to in-app pop-ups, notifications, to inform users of the difference. Users are provided with clear options to turn ML-derived content off as well as to block certain content from designated source providers when they do not agree to a specific term of the license from that source.

Intellectual Property and Licenses

Back when the project started, we had our evaluation of different approaches towards `iOS` application development. One of the key reasons why we oppose using `ReactNative` was licensing concerns. Since importing a good amount of modules is unavoidable when it comes to a `JavaScript` project, it would be difficult to check the license of all of them. If any module is imported with GPL license, and we use it as part of our proprietary software, we would also have to open source, as the license requires. A report from Synopsys (Armstrong and Kosinski, 2019) states that over 68% of `Node.js` packages have licensing conflicts and might cause legal issues.

Dependencies

For this reason, we took the native approach of using `Swift` to develop our mobile application. This decision provides us with a clearer perspective on licenses involved, including a limited amount of dependencies. The primary language `Swift`, is distributed under the relatively flexible Apache License v2.0. Specifically, with the Runtime Library Exception included in the LICENSE (*Swift - Apple* 2014), it is agreed that we can use this software to compile and embed them into binary files without specifying attributions.

Project License

`CocoaPods` and `SwiftSoup` are the only two dependencies we imported in our project. The first one is for managing dependencies and the latter one for parsing `HTML` files. Both software are provided with the most permissive MIT license (Chatbi, 2016 , Durán et al., 2015). Such that we can freely incorporate both of them into our proprietary software as long as we are giving attributions to them.

As we are developing proprietary software, incorporating permissive or copyleft license in our project would not be an option. Therefore, we have drafted our own End-User-License-Agreement (EULA) with our client. This agreement applies to both our developers and our clients. Whereas our developers agree to have no copyright claims to our clients as long as attributions are given, in return, we do not have any liability for extended maintenance of this software nor do we cover for any potential loss of future distribution of the application.

Data Privacy Considerations

Ever since the General Data Protection Regulation (GDPR) is enforced in 2018 (Voigt and Bussche, 2017), the definition of personal data has broadened from information that identifies a person to any information that relates to an identified or identifiable individual. Moreover, failing to comply with GDPR would result in a hefty fine of up to £17 million.

From the beginning, our team kept user privacy in mind throughout the process. We intend to provide the user with the freedom of opting out of all data collection regime while still have a fully functioning application. All core features of the app, including viewing, note-taking and contributing are generally unrestricted without logging in. Also, in compliance with the "explicit consent" section of the GDPR (Voigt and Bussche, 2017), we require users to read the full pop-up article of EULA before continuing to sign up or log in for the first time in our application.

Concerning private data collected from a specific user, ranging from browsing history, preference of material to learning progress, the application handle these data and communication with encryption and security in mind. `Swift`'s safety features and its elimination of unsafe behaviours guarantees implicit data flow within the application (AppleInc, 2014). Amongst all communications and data exchanges with the `X5Learn` back-end, extra-sensitive information are all hashed and salted to prevent any possible leak in its original state, while standard requests will always go through an encrypted `HTTPS` connection (*Content Security Policy (CSP)* 2020).

Sustainability Statement

We utilised as much optimisation techniques as possible in our application to ensure it is sustainable and efficient. One of the methods we used is dynamic loading of contents on `ViewControllers`. Simply put, a pre-fetch of materials will be conducted as soon as the user sees a content thumbnail on the screen. This strategy ensures a smoother user experience as contents usually start rolling instantly when the user clicks on the content. Also, this avoids networks spikes and potential slowing down of processing units.

Meanwhile, as an application that involves constant loading of significant video elements, the performance was taken into consideration at every step of engineering the application. A research conducted by (Bilberg, 2018), clearly showed that `Swift` applications perform consistently better in both memory and CPU usage, especially so when the task gets more substantial, a 114% in rendering time difference can be observed. For that reason, we believed our choice of using `Swift` and minimising the use of external libraries contributes to the great performance of our application.

Conclusion

In conclusion, we discussed potential liabilities, intellectual property and licenses, privacy concerns, as well as the sustainability issues involved in our project. From our perspective, we have made a substantial amount of effort to address legal concerns and to promote sustainable use of applications and their resources. In the next stages, we aim to seek further improvements, both legally and sustainability wise to deliver better products.

References

AppleInc (Sept. 2014). *Swift.org.* URL: https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html.

Armstrong, Gary and Rich Kosinski (Nov. 2019). *The license and security risks of using Node.js: Synopsys.* URL: https://www.synopsys.com/blogs/software-security/node-js-license-security-risks/.

Bilberg, Dennis (2018). "Comparing performance between react native and natively developed smartphone applications in swift : A comparative analysis and evaluation of the React Native framework". In:

Chatbi, Nabil (2016). *SwiftSoup.* URL: https://github.com/scinfu/SwiftSoup.

Content Security Policy (CSP) (Mar. 2020). URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP.

Durán, Eloy et al. (2015). *CocoaPods.* URL: https://github.com/CocoaPods/CocoaPods.

Meli, Michael, Matthew R. Mcniece, and Bradley Reaves (2019). "How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories". In: *Proceedings 2019 Network and Distributed System Security Symposium.* DOI: 10.14722/ndss.2019.23418.

Miller, Todd C., Bob Coggeshall, and Cliff Spencer (1980). *Sudo Main Page.* URL: https://www.sudo.ws/.

MIT Open Courseware Privacy and Term of Use, (2020). URL: https://ocw.mit.edu/terms/.

Swift - Apple (Sept. 2014). URL: https://github.com/apple/swift.

Tsai, F. S. et al. (Jan. 2010). "Introduction to Mobile Information Retrieval". In: *IEEE Intelligent Systems* 25.01, pp. 11–15. ISSN: 1941-1294. DOI: 10.1109/MIS.2010.22.

Voigt, Paul and Axel Von Dem Bussche (2017). "The EU General Data Protection Regulation (GDPR)". In: DOI: 10.1007/978-3-319-57959-7.