I Heart Corvallis Technology Review

Prepared by Dylan Tomlinson of Group 14

CS Senior Capstone, Fall 2017

June 4, 2018

Abstract

This document covers three pieces of the I Heart Corvallis project and the technologies needed to create these pieces. The pros and cons of each technology will be discussed for each piece, and one technology will be chosen for each piece.

Contents

1 Purpose

In this document we will be going over in detail three pieces of the I Heart Corvallis project, User Security, Application Scaling, and Application Testing. We will then discuss several technologies we can use to implement them. For each piece, we will list three different technologies and we will analyze the pros and cons of each technology. We will then pick one of the three technologies, indicating why that decision was made.

2 User Security

In the I Heart Corvallis application, users who are not students of OSU will need to make accounts before accessing the full functionality of the application. These accounts will require a unique email address and a password. We are obligated to protect the user's passwords as it has been shown that most users will repeat their passwords for different and possibly more important accounts.

2.1 Overview

Our goal is to require a third party expend a certain amount of resources in order to obtain our user's passwords. In order to do this we will be implementing a hashing algorithm. This will keep the user passwords more secure, as with hashing algorithms the passwords themselves will never be stored in the database. Instead they will be replaced with the resulting, irreversible hash [1].

2.2 Criteria

We will be deciding which technology to use based upon the programming difficulty, it's effectiveness, and the memory required to store the hashes.

2.2.1 Technology 1: SHA-3

Secure Hash Algorithm 3 is a hashing algorithm that utilizes sponge construction. Which is a type of algorithm with finite internal states that take an input bit stream of a given length and give an output bit stream of a that length. It is easy to implement and can be implemented in any of the languages we are using for this application. SHA-3 is a fast hashing algorithm, so it is somewhat susceptible to brute force attacks [3].

2.2.2 Technology 2: PBKDF2

Password-Based Key Derivation Function 2 is a hash function aimed to reduce the encrypted keys vulnerability to brute force attacks. It was originally written in 2000, but is still recommended for password hashing by RFC 8018 in 2017. PBKDF2 has many different implementations, including Javascript, PHP, Java, and C. So Regardless of our design decisions for the rest of the application, we will still be able to use PBKDF2. It is rather easy to implement as well. One of it's weaknesses is that brute force attacks using a GPU can be done somewhat cheaply [3].

2.2.3 Technology 3: Argon2

Argon2 is a key derivation function designed in 2015. It has three different versions, argon2d, argon2i, and argon2id. Each of these has a different strength and are specified by three common parameters. These are execution time, memory required, and degree of parallelism [2].

2.3 Discussion

The Argon2 hash is the most secure of the three hashes. In contrast, it also requires the most work to implement and the most memory to store the hashes. The SHA-3 hash requires little implementation and a small amount of space to store the hashes. Compared to Argon2 and PBKDF2, SHA-3 not nearly as effective at creating hashes as it's hashes are vulnerable to brute force attacks [3].

2.4 Selected Technology

We chose Choice 2, PBKDF2, because it hits the exact middle ground of what we need for this project. Our audience for this application consists of users in the Corvallis area, so we do not expect the application to be used or even known outside of this area. The argon2 hashing algorithm is a little overkill for our needs, and in contrast SHA-3 has been proven to be crackable with too little effort. We currently do not know our resources regarding database storage, but from what we have gathered so far we do not have much to work with.

3 User Interface Scaling

Our application is going to be available on android platforms. This means that we have to deal with several different types of mobile devices with different screen sizes and resolutions. We will need an easy way to ensure that our application retains its quality regardless of the phone being used. There are several technologies that aid in this, and we are going to choose one to implement.

3.1 Overview

Our goal is to have the I Heart Corvallis application be easily view-able on most android phones built in the last few years. We want this goal to be achieved rather fluently without having to create ten different versions of our user interface.

3.2 Criteria

Our criteria for this piece is its ease of implementation, it's effectiveness, and how well it meshes with our other user interface technologies.

3.2.1 Technology 1: Scalable Layout

An Independent android library released on Github that allows for easy scaling of the user interface. It includes measurements for objects that use relational units rather than pixels or dpi [4]. This ensures that objects that are created will take up the same percentage of the space on larger or smaller screens. It is implemented in Java and XML. There are no tutorials on how to use it so learning it will have to be trial and error.

3.2.2 Technology 2: Android SDK

The Android Software Development Kit consists of development tools, an emulator, required libraries to build android applications, and lots of sample projects with source code. It is one of the more popular development kits for android platforms. The kit itself already has libraries with several ways to build an interface with scaling capabilities [5]. It requires a little more knowledge of the android language and will require a lot of tinkering. But there is a vast amount of walk-throughs, guides, and examples of other projects using these libraries.

3.2.3 Technology 3: Semantics UI

Semantic UI is a html5 based framework that is powered by jQuery. It is used for developing Android apps as well as websites. It focuses on using semantics based commands, which makes its syntax very easy to understand [6]. In order to utilize Semantic UI you must install nodeJS and gulp, which increases the overhead of the framework. It is used in several applications such as Snap Chat. Its downsides include a strange build system and large CSS files with few ways to reduce their size [6].

3.3 Discussion

Scalable layout has a few great examples of how easy scaling can be done in similar applications. In contrast it has very little documentation so we would have little outside help with any problems we may run into. Semantic UI has the benefit of being used commercially in many different applications and websites. There is a great deal of documentation as well. A downside to is though is the huge CSS file, it could prove to be difficult to navigate through it to make a single change. The Android SDK has the benefit of being supported by Google with more documentation than we will need. The downside to it is it becomes difficult to separate your application from other applications using the base kit without extensive knowledge of the system.

3.4 Selected Technology

We chose the Semantic UI largely because we like the way the project can look utilizing this technology rather than using the Scalable layout or the Android SDK. We will also be using Semantic UI for the rest of our user interface design, so that will keep compatibility problems to a minimum.

4 Application Testing

In order to determine that a requirement has been completed, we have to be able to test that requirement's functionality. There are many different technologies that will aid us by allowing us to emulate running the application on an android phone.

4.1 Overview

Our goal here is to find a technology that can allow us to easily simulate and test our application on a variety of devices. The Android market is filled with many different types of mobile devices, and we have to ensure that our application works on the majority of these devices.

4.2 Criteria

We will be determining our choice in technology upon three ideals. The technology's effectiveness, the technology's ease of use, and the price of the technology.

4.3 Potential Choices

4.3.1 Technology 1: GenyMotion

GenyMotion is a multi-platform Android emulator. It can emulate most Android devices and supports older versions of Android as well. GenyMotion allows you to simply drag and drop an application (.apk file) to start running it on your emulator [8]. It utilizes the openGL capable GPU in your desktop, so even if the run times are not the same it takes a substantially less amount of time to emulate your app [7]. It also has a very simple install process.

4.3.2 Technology 2: Android SDK Emulator

The Android Software Development kit mentioned earlier, contains an android emulator. This emulator allows you to simulate many different android devices including their screen sizes, hardware characteristics, and android version. Utilizing this technology would allow us to easily test our program on any android device we need. [9]

4.3.3 Technology 3: Visual Studio Android Emulator

The Visual Studio Android Emulator can be used either through the Visual Studio program or as a stand alone emulator. It shares many similarities with the other SDKs. It can simulate many different Android devices including their screen sizes and hardware characteristics [10].

4.4 Discussion

Each of these technologies have their advantages and disadvantages. GenyMotion is very easy to use with it's drag and drop concept, and it runs much faster than the other two choices. In contrast to the other two technologies which are free, Genymotion requires a subscription cost of 119 dollars per year. In contrast, the Visual Studio Android Emulator is only slightly faster than the Android SDK emulator. It also has the benefit of having a usable stand alone version, or it can be implemented inside of Visual Studios itself. It is also compatible with the Xamarin IDE, which we may decide to use.

4.5 Conclusion

In conclusion, we chose the Android SDK emulator due to it's vast documentation and it's ability to emulate most devices with most versions of android. We also chose it because unlike GenyMotion, the Android SDK emulator is free.

5 Bibliography

[1] Coates, Michael, ”Safely Storing User Passwords: Hashing vs. Encrypting”, Dark Reading, 2014, https://www.darkreading.com/safely-storing-user-passwords-hashing-vs-encrypting/a/d-id/1 269374

[2] Dinu, Khovratovich, ”phc-winner-argon2”, GitHub, 2017, https://github.com/P-H-C/phc-winner-argon2

[3] Rietta, Frank, ”Use Bcrypt or Scrypt Instead of SHA* for Your Passwords, Please!”, Rietta, 2016, https://rietta.com/blog/2016/02/05/bcrypt-not-sha-for-passwords/

[4] Kim, Youngmann. ”Scalable Layout for Androi”, Github, 2017, https://github.com/ssomai/ScalableLayout

[5] ”Supporting Multiple Screens”, Developer.Android, 2017,

[6] Gerchev, Ivaylo, ”Introducing: Semantic UI Component Library”, Site Point, 2014

[7] Hindy, Joe, ”15 best Android emulators for PC and Mac of 2017” Android Authority, 2017, https://www.androidauthority.com/best-android-emulators-for-pc-655308/

[8] Favaro, Philippo, ”Genymotion: the best Android emulator for app development” Android Pub, 2015, https://android.jlelse.eu/genymotion-the-best-android-emulator-for-app-development-24f5c8 933768

[9] https://developer.android.com/studio/run/emulator.html

[10] Gorski, Edmund, ”Using the Visual Studio Emulator with Android Studio”, Clearly Agile Inc, 2017, https://www.clearlyagileinc.com/blog/using-the-visual-studio-android-emulator-with-android-studio