Joseph Lloyd 10177148

Joe.lloyd.22.24@gmail.com

Skripto 5000 final project report

DBS Higher Diploma in Science in Computing (Software Development)

24/07/2015

Supervisor: Clive Gargan

# Abstract

Scripto 5000 is an Apache Cordova app using the Ionic frame work. It is an instant messaging app powered by Node JS and Socket IO on Google Cloud. It uses its own API to make calls to a google cloud MySQL Database.

# Acknoledgments

Contents

[Abstract 1](#_Toc425105122)

[Acknoledgments 1](#_Toc425105123)

[Introduction 4](#_Toc425105124)

[Interim Recap 4](#_Toc425105125)

[Aims 4](#_Toc425105126)

[Aproche 4](#_Toc425105127)

[Background 4](#_Toc425105128)

[Litriture Review 4](#_Toc425105129)

[Documentation 4](#_Toc425105130)

[Visual Studio Apache Cordova 5](#_Toc425105131)

[Ionic 5](#_Toc425105132)

[Angular JS 5](#_Toc425105133)

[Socket IO 5](#_Toc425105134)

[Node JS 5](#_Toc425105135)

[Google Cloud 6](#_Toc425105136)

[Git & GitHub 6](#_Toc425105137)

[Other Litriture 6](#_Toc425105138)

[Stack Overflow 6](#_Toc425105139)

[W3Schools 7](#_Toc425105140)

[Code School Angular Tutorials 7](#_Toc425105141)

[Project discussion 7](#_Toc425105142)

[Tech used 7](#_Toc425105143)

[Apache Cordova 7](#_Toc425105144)

[Ionic 7](#_Toc425105145)

[Google Cloud 8](#_Toc425105146)

[Socket IO 8](#_Toc425105147)

[Git bash & git hub 8](#_Toc425105148)

[Database & Data Transfer 8](#_Toc425105149)

[MySQL 8](#_Toc425105150)

[JSON 8](#_Toc425105151)

[Version Control & backups 9](#_Toc425105152)

[Security 9](#_Toc425105153)

[Firewall 9](#_Toc425105154)

[Login 9](#_Toc425105155)

[Session Variable 9](#_Toc425105156)

[Results & evaluation 10](#_Toc425105157)

[Conclution & Future Work 10](#_Toc425105158)

[References & Bibliography 10](#_Toc425105159)

[Appendices 10](#_Toc425105160)

[1.0 Gant Chart 10](#_Toc425105161)

# Introduction

Scripto 5000 uses a large array of different technologies to perform its instant messaging. It currently falls short of what I had anticipated when I wrote the scope. Instant messaging from a phone app proved much more challenging than I had predicted.

## Interim Recap

At the time of the Interim report I had the basic framework built with Apache Cordova and Ionic working in tandem and deploying to my mobile phone. I had laid out a chart detailing different aspects of the app along a timeline. Unfortunately I overestimated the speed at which I could learn and implement new technologies and new app features. For example I expected the Node JS and web API to take no longer than a day but in reality this was an ongoing process that took closer to 2 weeks.

Because of the extra time spent on each task the final project only has its basic functionality. Please see appendix 1.0 the original Gantt chart now filled in with the details of how long each task took.

## Aims

After realising that I had scoped the project incorrectly I changed the Aim of the app to produce an instant messenger with basic functionality using cloud tech.

The aim is to show the use of new technology Ionic, Angular JS, Apache Cordova and Node JS. I also wanted to show the use of Cloud technologies in the form of the Google Cloud platform. I intend to show a proficiency with databases and data conversion. Finally I aim to show an aptness for version control via Git and GitHub

## Aproche

Almost all of the technology used in the project was not covered on the course. Because of this I have taken extra time to research learn and practice with things like AngularJS and Ionic. I even made several smaller projects to test basic functionality. I spent a large portion of time researching and testing tech like Socket IO and learning how Google Cloud would work with my project.

A key part of the development process was making sure I had a backup. This was achieved by committing my work to git hub every time I made some progress.

# Background

# Litriture Review

When building Scripto 5000 I used many sources of information and documentation since the project is built with such a large array of different technologies. This section is broken down between documentation and other forms of information that I used.

## Documentation

This section will review the official source documentation for the tech used throughout the project

### Visual Studio Apache Cordova

Source: <https://www.visualstudio.com/en-us/features/cordova-vs.aspx>

The Visual Studio documentation provided me we simple instructions on how to get set up and begin using Apache Cordova in Visual Studio. It then provided me with the necessary details for installing and using the built in nu-get packages that unlock extra functionality.

It was informative but heavy. There were some outdated methods and things that didn’t match up to the current version of Visual Studio Community 2013 edition. Whilst it was a fountain of knowledge I did find myself looking for other online tutorials to do things that should have been easily shown in the Visual Studio documentation.

### Ionic

Source: <http://ionicframework.com/docs/>

The Documentation for Ionic was the most clear and concise material I read in relation to the project. Each new HTML tag the ionic framework added is first explained with example code and then emulated on the right hand side to show what it looks like in practice. This was very helpful as it allowed me to build the app using the new Ionic tags confidently.

Because if Ionics explicit documentation I actually did not have to use any other source to get information on how to use the tags.

### Angular JS

Source: <https://docs.angularjs.org/guide>

The Angular JS documentation is extensive giving each area (Controllers, Scopes, Security etc.) its own page that details every aspect of it with links to examples on git hub. Each page normally starts off with the concept and some small bit of theory then it goes on to explain how angular handles the topic. It also supplies us with detailed examples of code on the page starting off relatively simple and getting more complex as we move down.

One thing I really appreciated about the Angular JS documentation was that it tells you when some element is depreciated and gives links to what should be used in the angular framework to replace its functionality.

### Socket IO

Source: <http://socket.io/docs>

I found the Socket IO documentation to be very simplistic which made it hard to achieve the more difficult messaging in the app. Because of its simplicity I was however able to follow part of the documentation and create a very basic chatroom in under an hour. I invited some friends to join and test out since I had it hosted on my Node JS instance. This was fun but also a very important part of the Socket IO learning process. Whilst it was great to build something so fast and see how the basics worked it still left me hunting online for more detailed/complex examples and explanations.

### Node JS

Source: <https://nodejs.org/documentation/>

The Node JS documentation Provides allot of detail on how to set up the server but I didn’t use it for this. Instead I used the Node JS documentation to help me trouble shoot errors I was getting as it gives details on how to make http connections and make requests. The documentation is very large and seems to cover each topic in only a few sentences not giving too much detail. This did lead me to search sites like Stack Overflow for more extensive answers to my queries allot of the time.

### Google Cloud

Source: <https://cloud.google.com/docs/>

The Google Cloud documentation provided me with the necessary steps for basic setup. There were some basic tutorials that I followed through to help me understand the Google Cloud platform. The steps to Implementing the SQL database with google cloud were explained in detail and I had no problems setting that up. The documentation regarding Node JS was limited as they didn’t have instructions on how to connect it to a SQL database and I had to look elsewhere to find that information.

Over all I found the examples to be pretty specific and only useful if you planned on doing what they had set up in their tutorials. On the other hand the built in documentation as you go through setup was very informative and made basic set up pretty strait forward.

### Git & GitHub

Source: <https://git-scm.com/doc> & <https://help.github.com/index.html>

The Git documentation was really just used as a reference point along with the setup of my GitHub repository. This documentation is clear and strait to the point, both sites make it easy to find information on any part of this version control process. They highlight BASH commands and expected outputs which made Git very strait forward to use.

As a side note since I’ve used Git a number of times before I was only referencing this material as opposed to learning a new concept from it. So in regard to Git & GitHub as reference material they were informative and easy to search for any query I had.

## Other Litriture

This section will review literature used that was not source documentation.

### Stack Overflow

Source: <http://stackoverflow.com/>

The site Stack Overflow provided me with many answers to questions I had during the development process. Some were questions I asked and some were already up there from the community. As a source Stack Overflow is only as reliable as the person making the post whether it be a question comment or answer. But because of its large and well respected community I feel that it is one of the most important sources I used in the project. When well respected and highly rated members of the community discus a problem the information can be better than the most detailed documentation. It shows real world application as well as all of the years of experience people have in their given field. Because of the reputation system it is easy to distinguish new people from veterans meaning that you can always pick a reliable answer if one exists.

Over all Stack Overflow helped me tease out some serious problems that the documentation just didn’t cover with professionals that had years of experience. On the other hand there is the issue that anyone with an account can answer a question. Because of the reputation system I don’t see this as a problem since you can disregard low ratted users answers/comments.

### W3Schools

Source: <http://www.w3schools.com/js/default.asp>

The W3Schools site provided me with some great information in regards to JavaScript. It’s very readable with interactive examples. The interactive examples are really what separates this site from others as a source of information. Example code is nearly never a mirror of your exact situation, this is why the interactive code on W3S is so useful as a source. I was able to bend and modify how things were used such as how to iterate through an array of JSON data and return a result.

W3Schools provided me with some very basic understanding of new concepts that I hadn’t come across during my time on the course and provided me a testing ground for code snippets that I wanted to modify.

### Code School Angular Tutorials

Source: <http://campus.codeschool.com/courses/shaping-up-with-angular-js/intro>

I found a series of videos that lead you through all of the basic concepts of Angular JS. They cover the creation of a whole single page web app using Angular JS. These videos come with tests afterwards, they explain in detail the way Angular JS directives work alongside the controllers and factories to give a real responsive site.

I watched these comprehensive videos many times. They gave me an idea of what I can and can’t do with Angular JS. They also manage to explain some more difficult Angular JS concepts with relative ease making them a really important source for learning.

# Project discussion

## Tech used

Below is a list of all the different types of technologies, languages and tools that went into the making of the Skripto 5000 messaging app. Most of which were learned as part of the project.

### Apache Cordova

Apache Cordova allows access to a phones native functions via plugins added to the config.xml file. Things such as local storage and the phones camera. Because of this an app can be designed with Apache Cordova and then deployed to different platforms. Since I used the version of Apache Cordova that comes with Visual Studio I was limited to deploying to Windows, ISO or Android.

### Ionic

Ionic is a front-end SDK for developing mobile apps with web technologies. This means that the majority of the app is written with HTML, CSS and JavaScript (Angular JS). The learning curve here is how the new Ionic HTML tags interact with the Apache Cordova tags to bring mobile functionality and look to the app.

#### angular js

Angular JS was by far the most difficult thing to learn on the project. It is used alongside Ionic and JQuery to extend the HTML language. It provides a client side MVC framework that lets the app take JSON data and feed it to the user depending on the request made. It also uses Controllers that help set up the page and deals with the login. The app is a single page application that loads different tabs based on requests by the user, these requests are all handled by the user.

### Google Cloud

Because of the nature of the application I knew it had to be deployed on the cloud for the purposes of testing. To test the basic functionality it required the use of one or two phones that had the app installed and sent and received data via the cloud. This is why I chose google cloud to host the app

#### GoogLe clouds sql database

I used a SQL database stored on google cloud to manage all of the data storage since I knew this data would have to be supplied to each user on the app. I created the Database on the cloud and managed it with MySQL Workbench where I could locally insert data or build tables and relationships and then save the changes to Google Cloud SQL database.

#### Google clouds compute engine (Node JS)

Node JS was necessary for data conversion originally but as I learned more about its functionality it became one of the most important parts of the app. It was on Google Clouds Node instance I created the web API by using the express.js framework. This gives the app its most complex functionality.

### Socket IO

I used Socket IO my web API and in the Angular JS controllers to send and receive messages instantly. This is done by getting my Node JS server file to look for incoming sockets and then sending the message onto the user that is listening on that channel.

### Git bash & git hub

For version control and code backup I user the GIT born again shell to upload my code to GIT hub where its east see changes made to code and review things that I have changed since the last time I uploaded code. GIT was one of the most essential tools used through the development process.

## Database & Data Transfer

I deal with two types of data in the project, JSON and SQL data. Converting between the two proved a challenge but with the help of my web API I was able to quite effectively transfer data from the MySQL database to the app.

### MySQL

The MySQL database was hosted on Google Cloud and was built with MySQL Workbench. I created a connection to Google Cloud SQL from the MySQL Workbench and manipulated data locally then pushed it up to the cloud after it was saved.

### JSON

Angular JS is very compatible with JSON data, as to be expected since they are both JavaScript. My web API was able to make SQL queries to the database and return some data. It would then take that data and convert it to JSON and send it to the app requesting it. This is the principal the entire app works off from the login request to the contact list. Each page full of data that is pulled from the database and turned to JSON and then served to the user.

## Version Control & backups

For version control I used Git BASH to commit to an online repository on GitHub. This was essential to keep code safe and backed up as well as seeing what I’ve changed over the course of project. This is a link to all the commits I made to the project.

<https://github.com/joeLloyd/Scripto5000/commits/master>

I only made commits when there was something of value added to the project. As can be seen by the commit dates I didn’t commit between 28th of May to the 26th of June, this is because I spent near 3 weeks working on Google Cloud which was outside of the main code (building my web API).

For the web API, I made local copies of the server,js file and managed my database with the local too MySQL Workbench. Both of these were deployed on the cloud but had local backups.

## Security

This section will talk about the security of the app mostly in relation to the weaknesses in the system.

### Firewall

As part of my Google Cloud setup I had to configure special fire wall rules to allow users to make JSON requests to my web API. This was a protection put on my Instance of Node JS. I created the Firewall rule to allow any IP address to make a request to the instance of Node JS so they could access my web API.

The instance of MySQL was handled differently, I allowed only two addresses to access the database. The first was the IP address of my Node JS instance so as it could make calls to the database when an app requested it. The second was my own local IP address so as I could access the data from home via MySQL Workbench.

### Login

The login verification for the app is done in a vulnerable place. After setting up the basics of the app and the Node JS instance I moved onto the login function. This took some time since it was the first real use of Angular JS and the Node JS instance. When I finally got it to work I had built an insecure login since it calls passwords and usernames and iterates through them in the app instead of just returning a promise from the server.

Passwords are stored and read as strings from the database, I did not have the time to encrypt passwords before sending them to the database with Angular JS as the login took me longer than anticipated.

I do now know how to create a server side login and plan on doing this as a first task after the hand up as future work.

### Session Variable

The session variable is held in a $rootScope variable that is unique to each user and comes from the database. It should be stored as a local storage on the phone with the ng-cookie directive from Angular JS. I did originally attempt to store the session with ng-cookie but I didn’t get results quick enough and knew I needed to move onto much more important functionality. I used $rootScope from Angular JS to store the session object because just like a cookie it would be accessible everywhere I needed it.

# Results & evaluation

# Conclution & Future Work

# References & Bibliography

# Appendices

## 1.0 Gant Chart