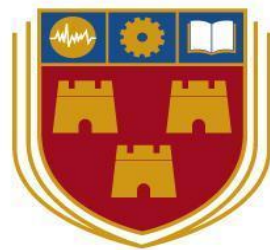


5th April 2017

Final Report

Institiúid Teicneolaíochta Cheatharlach



INSTITUTE *of*
TECHNOLOGY

CARLOW

At the heart of South Leinster

Department of Computing and Networking

Software Development Degree

Project name: MaaP (Message as a Platform)

Student: Chihabeddine Ahmed

Student Number: C00210496

Supervisor: Joseph Kehoe

Table of Contents

1. Introduction.....	1
2. Project Description.....	2
2.1 What is MaaP?	2
2.2 System Description	2
3. Submitted Product Description	3
3.1 Login activity	3
3.2 Register activity.....	4
3.3 Main Interface activity	5
3.4 Messaging activity.....	6
3.5 Friends Activity.....	7
3.6 Search Activity.....	8
3.7 Notifications.....	9
4. Conformance to Specification & Design	10
5. Learning outcomes	11
5.1 Personal achievements	11
5.2 Technical achievements.....	11
6. Review of Technologies choices	13
6.1 Node.js	13
6.2 Other Technologies	13
7. Project Review	14
7.1 What went right?.....	14
7.3 Problems encountered.....	14
7.3 What could have been better?	14
7.4 Advice for others.....	15
8. Conclusion	17
9. Acknowledgment	18

1. Introduction

This document will be present the final result of MaaP (Message as a Platform) project developed as part of the BSc. (Hons) in Software Development at IT Carlow. The project was developed between October of 2016 and April of 2017, and without any hesitation, I can say that this was the biggest and the most challenging project that I have ever encountered up to date.

This final document is divided into nine sections. In the first three sections of this final report, I will explain what is MaaP is and present description for each of the main functions implemented in the final product. The fourth section will deal with explaining how the application design was followed during the implementation, and where an exception was made if any. In the fifth chapter, I will describe both personal and technical learning and achievements that resulted from working on this project. The sixth section of this document is a review of the project and it will include on my reflections regarding the work, complexity, my decisions and their impact on the outcome of this project. In the seventh chapter, I review the technologies was used in the development of this project and result of this suitability of these technologies. In section eight I will conclude the final report and I will give my final say on the project. The last chapter will be dedicated to extending my gratitude to people who have been involved in the creation of the MaaP System.

The details of project implementation can be found in the Functional Specification and Design Documents of this project and details of research conducted as part of this project can be found in the Research Document.

2. Project Description

2.1 What is MaaP?

MaaP (Messaging as a platform) is a self-hosted messaging application built for android.

The front-end of the application build using Java and the backend is built purely on Node.js. The application will allow the user to authenticate, i.e. Login, Register, and Logout of the system. It will allow the user to send and receive messages to a particular user. Also, it will allow a user to search for another user then send him an invitation and upon the invitation acceptance, they can send messages instantly.

The goal of this project to build the application for many types of users such colleges, companies etc.

Where these users can host the application on their own servers and own the data. Each of the functionality is supposed to be built in a micro-services style, where each service is dependent.

2.2 System Description

MaaP is divided into two major components: the user interface build for android devices using Java and the second part is server-side. The server side is composed of three functions and build using Node.js.

Authentication function which allows user to Register, Login, and Logout. And this is done using node and Mongo database. Messaging function is concerned with the idea that users can send a message to an individual user using socket.io. Socket.io is a node.js library that It enables real-time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API. But in the of socket.io I used it to send and receive a message from android to android device. The other part of the messaging function is the Notifications. Notifications are implemented using FCM (Firebase cloud messaging). FCM is a Google owned messages service that enables messaging functionality. This function is built in at the backend server that allows the MaaP application to send and receive a message via notification in case of the user is not using the application. The last part of the application is friendship function. For a user to send a message they first need to add a friend to their “friendslist”. And the way this function work is to allow a user to search for a user then request invitation then the other hand user will accept the friendship and then they can start communication between each other. The front end or the user interface concern with translating the backend function into action, i.e. allow the user to Login and register and send messages using the interface.

For further details on these processes please refer to the functional specification and the design document of this project.

3. Submitted Product Description

3.1 Login activity

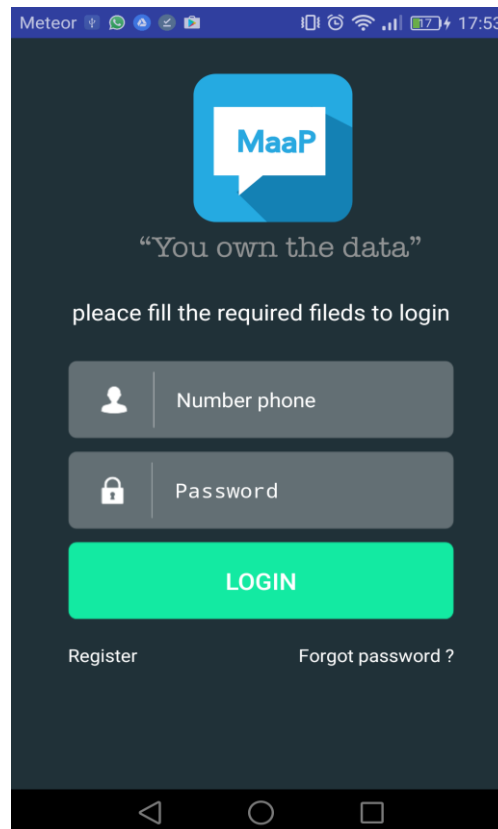
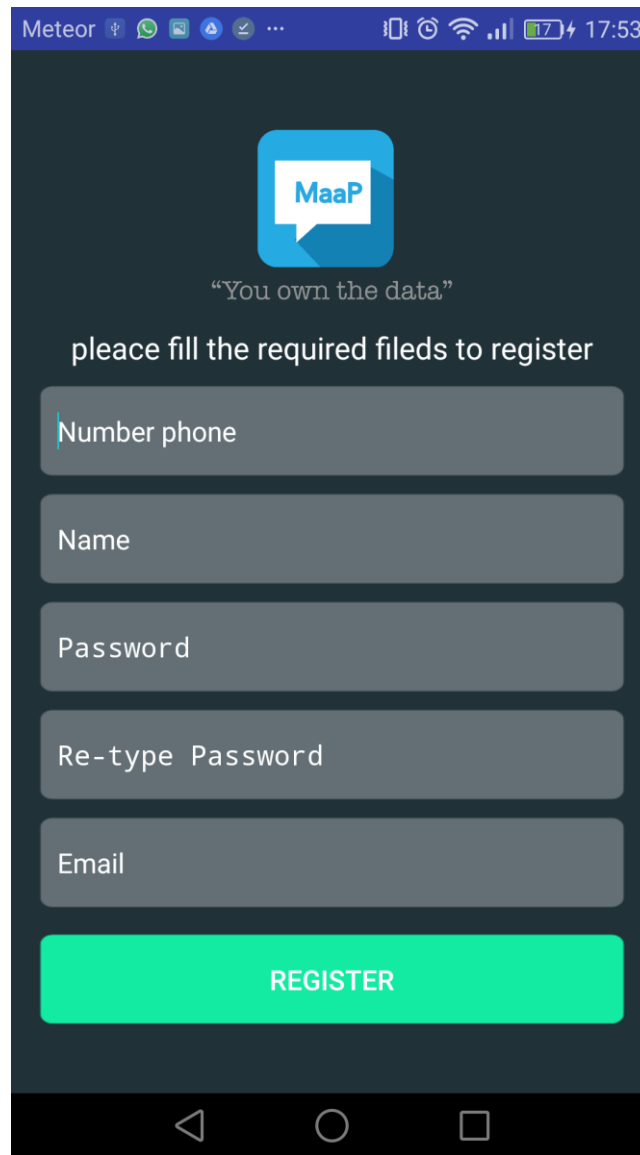


Figure (1)

The "login" interface is a typical interface design is used by many applications, messaging or non-messaging applications. From the interface, we can see that the user will input their phone number and password and once they input these details the user will hit the “Login” button indicated by the green button. In case the user is not registered the user will click the “Register” link then the user will be directed to the “Register” activity at figure (2). If the user login is successful they will be directed to activity “Main interface”. And from there they can perform their operations.

3.2 Register activity



Meteor 17:53

MaaP

"You own the data"

please fill the required fields to register

Number phone

Name

Password

Re-type Password

Email

REGISTER

Figure (2)

Again the Register account is a typical interface is used by almost many types of applications. And as we can see it compose of many input fields. Each has a hint on them. The user is commanded to fill all fields. After they supply the required details they can click the "Register" button and they are successful they will get a message that says "Registration successful" and they will be directed to the Login screen (figure 1).

3.3 Main Interface activity

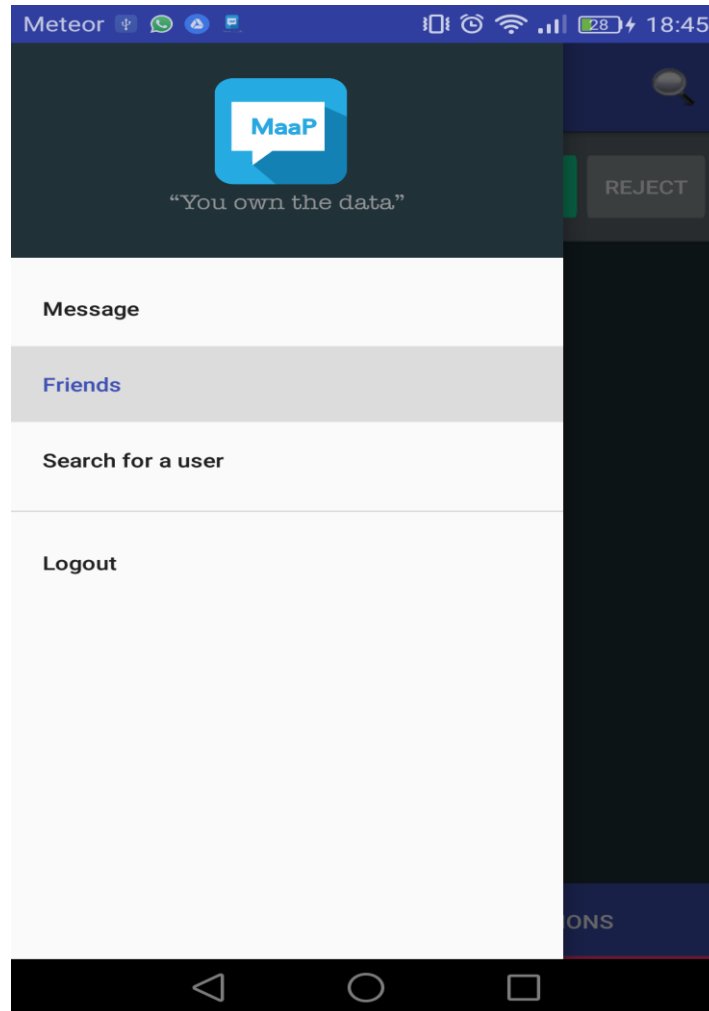


Figure (3)

This main interface provides a navigation for a user to another activity and it's represented by a "Drawer". Each activity is named accordingly such "message", "Friends" etc. and lastly, a user can hit the "Logout" button and will be able to exit the application.

3.4 Messaging activity

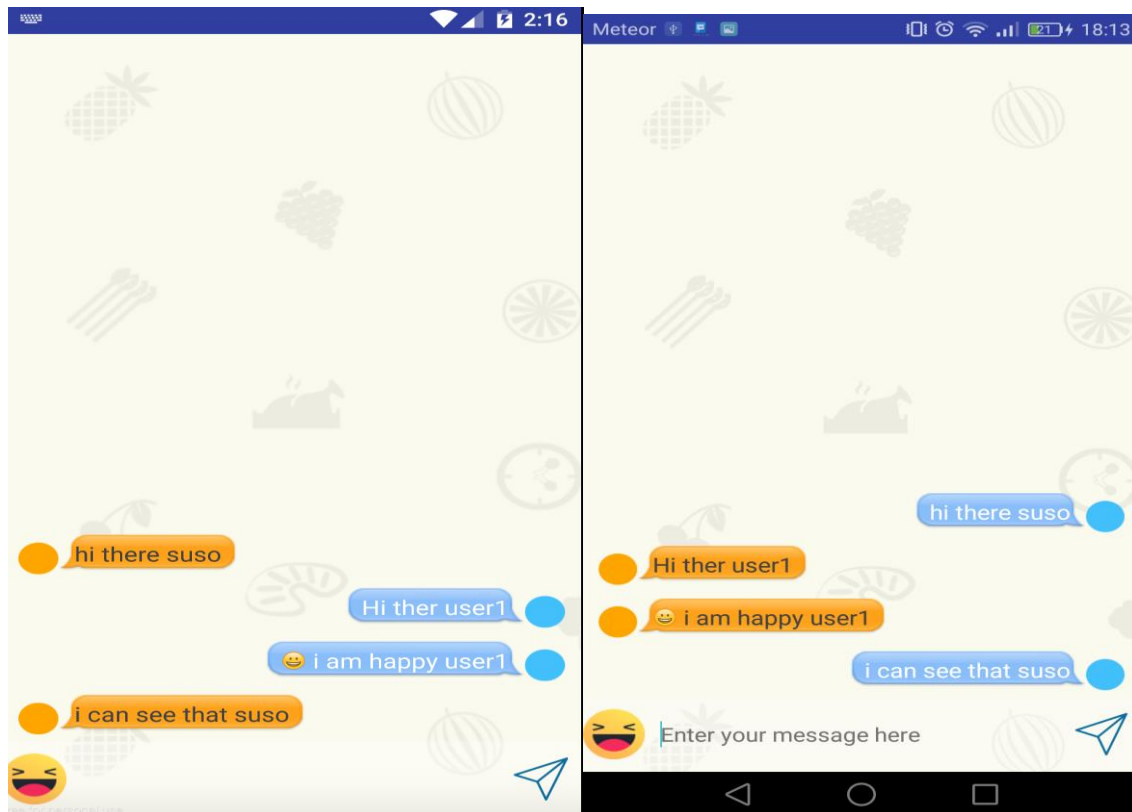


Figure (4)

This illustrates messages interchange between two users. The first user is called “user1” and the second user is called “suso”. From here we can see that there are 4 instant messages have been exchanged using this interface. From here we can user can send a normal text and Emoji. Video and audio are not working at the moment but hopefully I can work on that at a later stage if I have time. The only observation that I have on the interface is that this interface does not display a username at the top of the screen to indicate to the users they messaging with. I think it would have been a good idea to include that.

3.5 Friends Activity

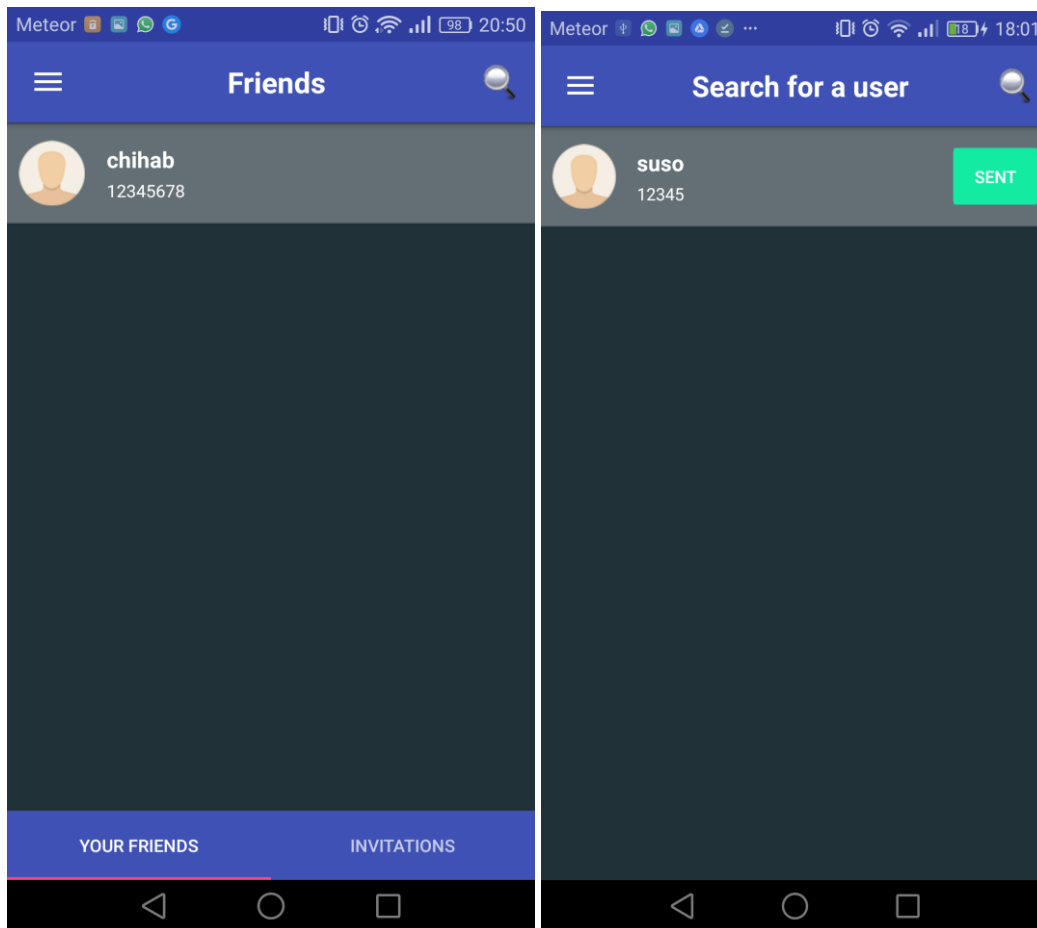


Figure (5) Friend list at right and invitation at left.

The user interface for the “Friends” activity is a simple interface. The function of this is to display a user that have been added by another user, i.e. if a user sends an invitation to another user and that user accepts the sender's invitation they will become a friend and their name will be displayed in this interface. From here a user can tap on the user “chihab” for instance and another activity will be opened. This other activity is called “messages” activity. You can navigate to figure (4) to see this activity. This activity also includes the invitation activity which allows a user to see all invitations sent by other users. In this case, there is only one invitation by “suso” and indicated by “sent” which means the invitation is sent.

3.6 Search Activity

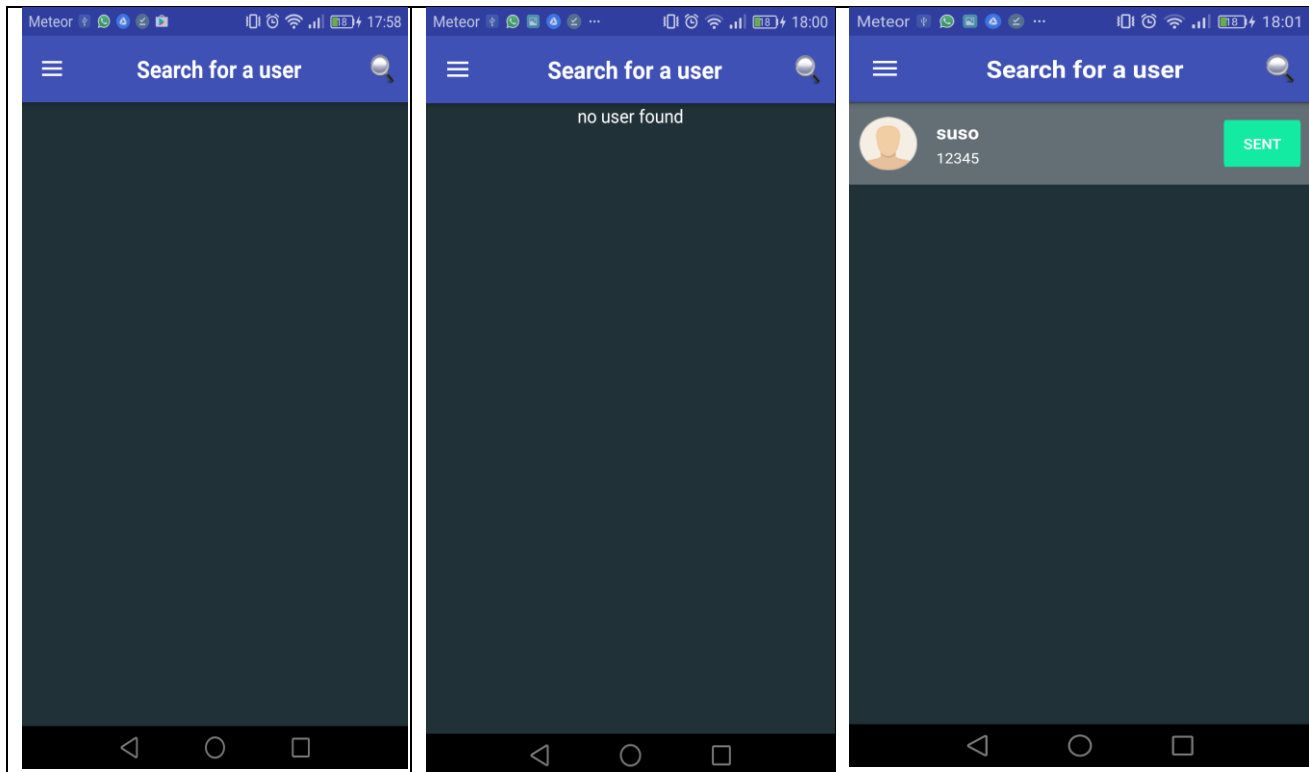


Figure (6)

This Interface mainly composed of three function.

A function to type a user phone number (search box) that has hint called “search for a user”.

A function to perform the search that is indicated by the search icon.

And a lastly the result function. Weather the user is found or not a user will get a return message accordingly. The return message can be “no user found” or a result from the database for the searched user. Upon a successful search the user can request an invitation as shown figure (5).

3.7 Notifications

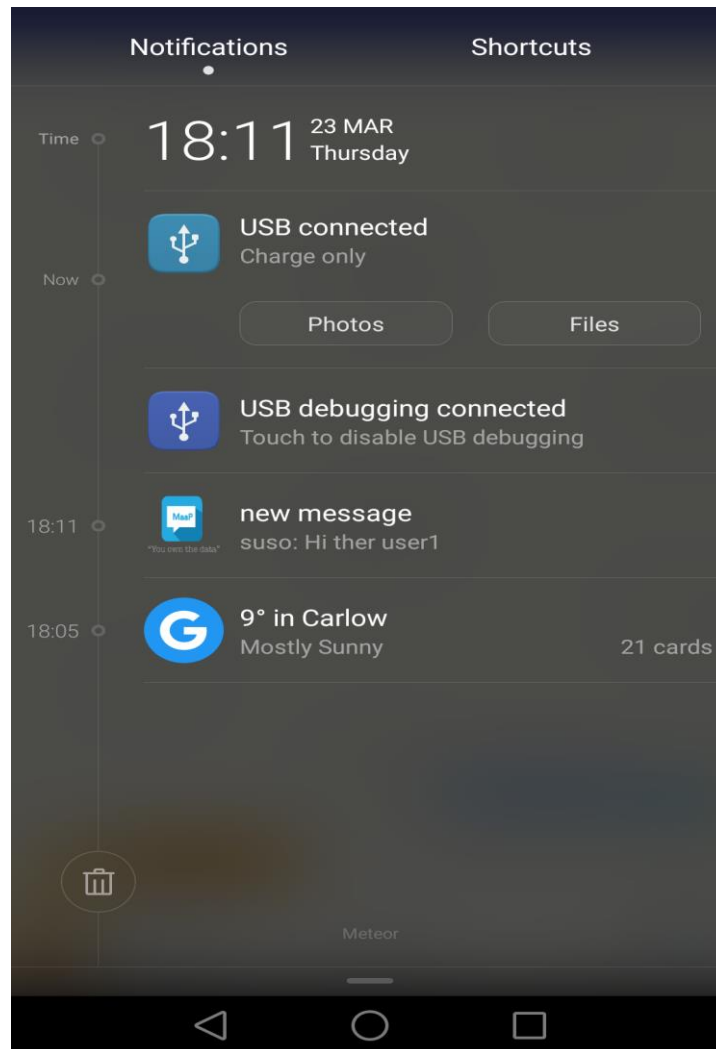


Figure (7)

As a part of android, Notification drawer is included in every android device. In this case when the user sends a message to a user but the user is not using the app we need a way of delivering that message to him/her. The way we do this is using Push notification. In this case, I used FCM (Firebase cloud messaging) to deliver that message. The message is indicated by the MaaP logo and to be precise in this screen the message is included in this drawer it says “suso: Hi there user1”. In this case, the sender is called “suso”, and the message body is the “hi there, user!”

4. Conformance to Specification & Design

The final submitted project conforms to most of the functionalities which were outlined in the functional specification document. The main functionalities which were outlined in the Functional Specification document were as follows: Authenticate user to the system by mean of creating an account, Login, and logout of the system. Messaging, which means sending and receiving messages between users instantly using peer to peer or P2P Design. Friending, by mean of allowing a user to search for a particular user then send an invitation or a request to that user and once the user accepts the request they can now communicate. The last part of the application is the Integration service. The integration service as an additional service that would allow the user, companies or Colleges to integrate any service to the system. For instance, a college that might have service for sending and receiving message between students and teachers and at the same time they want to see timetables and any announcement made by a teacher. The college would create a separate service and integrate it into the system. Unfortunately, due to the time for submission, I was unable to implement that service but I would try to implement it if I have time.

In term of the user interface, I have prototyped the UI before starting implementing the system and I thought it has saved me a significant amount of time. And the aim for the user interface was to keep it as simple as possible and to try and not to complicate things which would make it difficult for the user to navigate through the app, I feel achieved. The design of the app was based on trying to keep the app as lightweight and as responsive as possible. A constant internet connection for the submission and retrieval of item information from the server as nothing was stored on the application itself is needed. Given that all the design and functionalities outlined above were achieved within the project, it conforms well to the design and functional specification which was set out in the corresponding documents.

5. Learning outcomes

5.1 Personal achievements

This project has provided many opportunities for personal learning, by giving me the opportunity to tackle the problem advertised by “Redhat”. This gave the opportunity to apply all the software development skills that I have acquired for the past four years of me of learnings.

The ability to create a formal specification for this project resulted in a large number unanswered questions in areas which were uncovered by the course syllabus. This situation was somehow intimidating at the start. It quickly turned into a rewarding experience where with the help of research, existing knowledge and creativity, each of the initial requirements was broken down into much smaller and manageable problems. With the help of this project, I was to put everything that I could have learned about messaging and give me an idea about why are messaging applications are so important right now. The various computer languages that I have studied including database structure, software design, and many others I was able to apply them to this project and it was very helpful. From this project, I did learn a lot more about micro-services, Node.js, Java and android as a technical skill. Not only that but also it gave me the skills to tackle things like documentations, presentations, asking questions and also communications. Despite this being a solo project there were many times in which I had an opportunity to improve my communications skills. I had clearly improved my communication with my supervisor during the weekly meetings. Presentations were great because it gave me skills of presenting a meaningful content that it would surely help me in my future's presentations

5.2 Technical achievements

(JavaScript) Node.js: I have previously touched on Node.js but I did not get the chance to implement such project with this. Due to the lack of using technology previously at first if found a bit intimidating but I was able to overcome this and by the time goes on I was able to learn a lot about this technology. The things that I learned about this technology was the ability to work with servers as this application concern with servers. Express server is a node.js server that works as an HTTP server and it does a lot of cool stuff and I did learn a bit from it. The idea of creating micro-services was new to me and I am still learning more about it. Hopefully, in the future I get to work with Micro-services a bit more. Because I found it really interesting to working with individual application and putting them together was one of the hardest part.

Java: I used this technology alongside XML (Extensible mark-up language) to build the front-end for my android application. I have studied Java before as a language but I was not that familiar with the concepts of Android development. At first I watched a couple of tutorial to get my head around this. Going to place like YouTube and other I was able to learn the basics. With extra research I learned a vast deal about android development and I fall in love with it.

Documentations: The collection of documentation that I have written including this report is the largest amount I have written for any past project. I found the experience of doing so very educational. Other technical skills are illustrated in the final product.

Github (open sourcing the project)

If familiarized myself with Github as this is the first time that I have used Github. I have learned to use Github. From creating a repo to uploading/pushing project/file to that repo. Also, making changes to a file then apply commit to that file and push it again to the repo. I thought I have learned a lot from this.

Managing the project: I was able to use to project to learn about time management and solo development. Developing a final year project as an individual is always going to be a very hard task but this has opened my eyes to some of the challenges a project of this nature present. In terms of time management, I learned to schedule my work according to each iteration. I.e. I would plan exactly what I had to achieve in each iteration by allocating time for Designing and time for implementation.

FCM (Firebase Cloud Messaging): Push notification is a very important feature in a messaging application and I had no idea what Push notification service should I integrate to the backend system. Having researched, AeroGear (a push server for notification purpose), FCM a cross-platform messaging solution that lets you reliably deliver messages at no cost.), I had to make my mind on which Technology should I use and after more research, I found FCM was a little bit easier as there are many tutorials available. I had to read a lot of documentations regarding this technology and running sample FCM to get my mind around it. And at the end of the project, I thought I have learned a lot from this.

6. Review of Technologies choices

Overall, I feel the technologies chosen for this project were well-suited. Although since the project meant to be in micro-service architecture I would have been better to develop different services in a different languages and link and deploy all services together.

6.1 Node.js

Node was proposed choice for this project such as this but I found it to be highly suited when the right tools were used. Compared to languages such as Java and PHP I found the code to be easier to read, write and understand. At no point in this project did I feel that the lack of lower-level details in the code restrictive and at no point did I find the performance of Node to be problematic. Some technologies that could have done the Job is Python and Go but I decided to use node.js because I have touched on it previously and I did not get a chance to develop a backend with it. So I think the use of Node as a backend language was well suited for the development of this application and it was highly effective in this role also.

6.2 Other Technologies

Node.js, Java, FCM, and Mongo database were all suited this project well. All were chosen on the basis of being simple and robust technologies that supplied the services needed. This approach worked out well, as there were no significant negative issues involved in the use of any of them.

7. Project Review

This section will provide an overall assessment of the successes and failures of this project as well as a review of the technology used and what the next steps would be if this project were to continue.

7.1 What went right?

Overall, I would say this project was successful. Most of the features specified at the start of this project were implemented in what I believe to be a satisfactory manner. In addition, some supplementary features were included due to having free development time near the end of the project.

7.3 Problems encountered.

Creating Microservice:

The idea of the project was to build a Microservices in which all services are separate and dependable. I created the first micro-service (authentication API) but when I get to create the other service I found myself lost because there isn't many tutorial or resource so that I can use and get my head around this. So I ended up using that Micro-services and adding the messaging function to it and so friendship function.

Getting the messaging function to work:

One of the most difficult part of the system is to get the messaging function to work. Normally many messaging application uses messaging API like Twilio REST API. The Twilio REST API allows you to query meta-data about your account, phone numbers, calls, text messages, and recordings. You can also do some fancy things like initiate outbound calls and send text messages. Or any other messaging API. In my case, I had to build my own messaging API. I could have used Parse server as a backend but again, this is an API-less development. I.e. I design my own API for your system.

7.3 What could have been better?

Due to the rather short duration of this project I had to start using many new technologies without much time to learn about them prior to writing the production code. One of the most difficult concepts I had to learn was the API design and microservices. The difficulty of this task lay not only in understanding what is required for each API endpoints, and what response should be provided but also in finding or defining some standard for an API. This is the second issue is one of the main reasons why software developers have to read so many pages of API documentation even for the simplest services. When I learned about

the Collection+JSON standard, it was already too late for me to go and change the entire API of the services.

While I know that the Collection+JSON standard may not be suitable for many applications, I can clearly see how using a certain part of it would make the API much more consistent and easier to use. The concepts of the Micro-services were a very interesting and I would have been better if the whole system is completed in Microservices architecture. But so far the system is working and I would be able to make changes to the system architecture if I have more time. Also, should the development of this project continue I would focus on addressing the security concerns of the application, perform more testing with more users. Also, I would like to finish the integration service. The system was meant to allow the user to authenticate, send messages between users and have an integration service that would allow users to integrate their services within the MaaP system. Due to the time that I have, I did not implement this features and other features like allowing the user to change their profile data, such password, name, phone etc

7.4 Advice for others

Dealing with such project is always going to be a strong challenge. The main advice I would give to someone looking to undertake a similar project would be to thoroughly research all of the technologies that they will use. Also to spend extra time refining the design for the project before commencing with coding. Choosing the right technologies from the start is key to keeping your project on time and reduce the chance you will have of any major setbacks. And most importantly Time management, as iterations really pass way very fast and without planning for each iteration you will find yourself a little bit behind.

Keep documentations up to date:

When using agile practice, the work seems to flow much faster, a week is quickly passing by with constant focus on produces good design, quality implementation and collecting feedback. Quick whiteboard or something similar is often necessary to gain a better understanding of the features to be developed.

It's very easy to put the documentation aside for a couple of weeks, and it's very difficult to remember every diagram that was drawn earlier. Assuming that the design was followed, it is possible to reproduce these diagrams, but this is rather a slow and inefficient way of working on the design document. The advice I would give in this case, every single time you make changes to the project diagrams or adding, modifying features keep the documentation up to date and keep checking with your supervision to see if you are in the right direction.

Know your limits:

As this project is not the only task to be completed during the final of the college, it is not wise to dedicate all the time to it, by trying to add and deliver every use case proposed by customers and supervisors. Keeping track of the tasks and the progress using a Kanban board can help in planning my sprints and making decisions to descope if necessary.

I strongly believe that focusing on the most difficult and most valuable part of the system (messaging function) is a good strategy, as when it becomes necessary to descope the project, the use cases that are left out are often the simple features which would not bring much of a value both business and learning perspectives.

Avoid early UI work:

Consider that the user interface is the only visible part of the system and many of us love a good looking UI, I starting the develop the UI at first and I thought I spend a lot of time that I could have used to deliver the backend first. So I turned my attention to deliver the backend first and appoint a time for the UI decision. At the end of the first iteration, my project offered a little in terms of the user interface, but it had a good enough backend (Authentication API) to present to the user and explain the idea behind my project.

Design before coding:

After dealing with many simple assignments which did not require any serious design, jumping straight into the code without any preparation becomes a bad habit for many software development students. Approaches like this most likely lead to a very poor design with very strong coupling, which in turn will impair the ability to introduce change and accommodate user feedback. So, in a nutshell, I would say follow the principle, “Do design before coding”.

8. Conclusion

The project acquired a lot of time from me, and I think taking something from this project was the most important things. I thought the project went in the right direction in term of design and implementation with some ups and downs but at the end, I thought I have accomplished something that would really help in my future career. The technologies that were selected worked very smoothly. Issues that faced my during the development were addressed to my supervisor and also the team of Redhat and they were very helpful. So to conclude, I would say this project was successful. Most of the features specified at the start of this project were implemented in what I believe to be a satisfactory manner. In addition, some supplementary features were included due to having free development time near the end of the project.

9. Acknowledgment

Firstly, I would like to thank my project supervisor, Joseph Kehoe, for his help and advice during all stages of development of the MaaP system.

Secondly, I would like to thank the Redhat team, Wei Li, Leigh Griffin for their continuous support and encouragement that I have received in order to complete this project.

Friends and family for their support and words of encouragement, and everyone else that I am forgetting to mention, for all the help and advice along the way which has helped me to expand my knowledge and grow as a software developer.