National College of Ireland
BSc in Computing
2015/2016

Anderson Cahet
X11104821
anderson.cahet@gmail.com

Finding Friends

Technical Report

Table of Contents

Executive Summary

With nowadays social networks where people leave most of their aspect for everyone with access to see it, where people use mobile devices in order to find people for a coffee, a night out and others (apps such as Tinder, Facebook).

Finding Friends target people who also like to meet people outside the safe zone, where through the check-in through the application, users will be able to find others that are also nearby alone or with friends.

The application at this point is not aimed to become a chat, as such already exists, but for travellers who want to meet people and make friends outside the social networks, even if they are having a coffee, going for a walk around the city, going to a night club.

Finding Friends will allow user to find users that are in a 1.5 km radius of the users check-in and will enable users to visualize other users nearby and their usernames where when interact they will be able to send a picture from their library.

Finding Friends aim iOS users at first as it has been written using Swift language on Xcode. User will need to have access to the internet, using 3g, 4g or wireless as app need to collect data regarding their location to be shown in a map.

The application has been tested on iPhone 4s and also on iPhone 6s.

1 Introduction

1.1 Background

As the new trend nowadays is regarding developing an app that will give a portfolio to jobs related to iOS/ Android/ Window Phone developer I decided that as I already had some class with Java, had a semester with C# and also Ruby, after a researcher online, I decided that I should try to develop my app using Swift language with a focus for iOS devices.

I had the idea around 1 to 2 years ago where I wanted to build on that time, not an mobile application where 'A' would be able to check in place 'x' and this person's friends, would be able to find 'A' through Finding Friends as sometimes we have the desire to go for a coffee, walk, a night club, however some of the friends are not in the same mood, but through Finding Friends would be possible to find those friends that for some reason user 'A' did not get in touch with before going out.

Finding Friends is an app where people will be able to interact with others that have the same desire that is to meet a company, friend to share a moment with.

1.2 Aims

To offer an iOS app, where users will able to:

- Add the name of the location where they checked in
- Go to nearby options to verify if there is any other user nearby user. (limit of 1.5km)
- Change settings regarding, username, email, password, profile picture.

1.3 Technologies

To develop Finding Friends I decided to use Swift programming language which is the latest language that Apple has developed in order to build applications that will used on Apple products, such as iOS, OS X, watchOS and tvOS. It is new, but is possible to find many tutorials, courses and also information on Apple developer website

(https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/#//apple_ref/doc/uid/TP40014097-CH3-ID0).

Within Swift Language Core Location has been used to access, after request being approved, user's location and establish other users that will be close.

I based the use of swift through a few courses that I signed up for on Udemy website (https://www.udemy.com).

1.4 Structure

2 System

2.1 Requirements

2.1.1 Functional requirements

ID: FR01

Title: Download Finding Friends App

Description: User that holds an Apple store account and an iPhone should be able to download the application for free.

ID: FR02

Title: Registration

Description: Registration and based on Facebook Login Button and only.

ID: FR03

Title: Login

Description: As user registered the correct information he/she can now login and access application.

Dependency: FR01, FR02

ID: FR04

Title: Check in

Description: User goes and access the Check in button, where system request permission to use user's current location and if approved, user will get the

geocoded location on the bottom of the screen and will have an opportunity to name the place where user is at the moment and add a picture from his/ her library.

Dependency: FR03

ID: FR05

Title: Nearby

Description: Given user gave mobile application to access his/her location, a map view will be displayed and populated with pins that will contain other users locations that are close to the current user. These other users will appear on a table view below which has: user profile picture, username and the distance from the current user.

Dependency: FR03

ID: FR06

Title: Post

The post will display more users than the nearby section. There is also a table view however without a map and with the difference that will have the text that has been input by the user together with with profile picture, name and picture used while doing the checkin.

Dependency: FR02, FR03

2.1.2 Data requirements

2.1.3 User requirements

Login/ SignUp: Figure represents the code that is used when Facebook button is pressed and fields that app will be requesting users authorization.

After retrieves the necessary data to be saved in Parse.

```
@IBAction func logoutBtn(sender: AnyObject) {

    if PFUser.currentUser() == PFUser.currentUser() {
        PFUser.logOut()
    }

     self.performSegueWithIdentifier(SEGUE_LOGIN_AFTER_LOGOUT, sender: self)
}

func saveFBintoParse() {
    let graphRequest = FBSDKGraphRequest(graphPath: "me", parameters: ["fields": "id, name, gender, email"])
    graphRequest.startWithCompletionHandler( {

        (connection, result, error) -> Void in

        if error != nil {

            print(error)

        } else if let result = result {
            print(result)
            PFUser.currentUser()?["gender"] = result["gender"]
            PFUser.currentUser()?["name"] = result["name"]
            PFUser.currentUser()?["email"] = result["email"]
            PFUser.currentUser()?.save()

            let facebookID = result["id"] as! String

            let facebookProfilePictureUrl = PROFILE_LINK + facebookID + PICTURE_SIZE

            if let fbpicUrl = NSURL(string: facebookProfilePictureUrl) {

                if let data = NSData(contentsOfURL: fbpicUrl) {

                    self.profilePicture.image = UIImage(data: data)

                    let imageFile:PFFile = PFFile(data: data)

                    PFUser.currentUser()?["profilePicture"] = imageFile

                    PFUser.currentUser()?.save()

                }

            }

        }
```

Requirements:	Login/ Sign Up
Number:	1
Descritpion	User Launch app for first time and sign up for an account
Rationale:	User needs to click on the Facebook button and login on his/her Facebook account

Success	User input correct email and password and app sends user to main page using a Segue
Level of importance	5, High

Requirement	Check In
Number	3
Description	Registered users will be able to check in based on their location. They also will be able to name their location, add a picture receive the his/her geolocation which will saved in Parse.com
Rationale	Users will add the name of the location they are or any text and a pitcure.
Success	Check In button is pressed and after a name for the location where user is is saved into parse.com. in case user leaves name of location empty or do not select an image a pop up message is displayed
Level of importance	5

Requirements	Nearby People
Number	4
Description	After checked In users will be able to view other users nearby by an “x” amount distance
Rationale	User will be able to view other users nearby with their name, profile picture and distance to current user
Success	User is able to see others which checked in nearby as pinpoints and their usernames in a table below map. In case no user checked in a pop up message will be displayed saying there is no user nearby and no username will be displayed on the table
Level of importance	5

Requirement	Log Out
Number	6
Description	User will logout from application
Rationale	User will be returned to login page
Success	User returned to login page

Level o importance	3

2.1.4 Environmental requirements

iPhone with latest firmware.

Latest xCode to test purpose

2.1.5 Usability requirements

2.2 Design and Architecture

2.3 Implementation

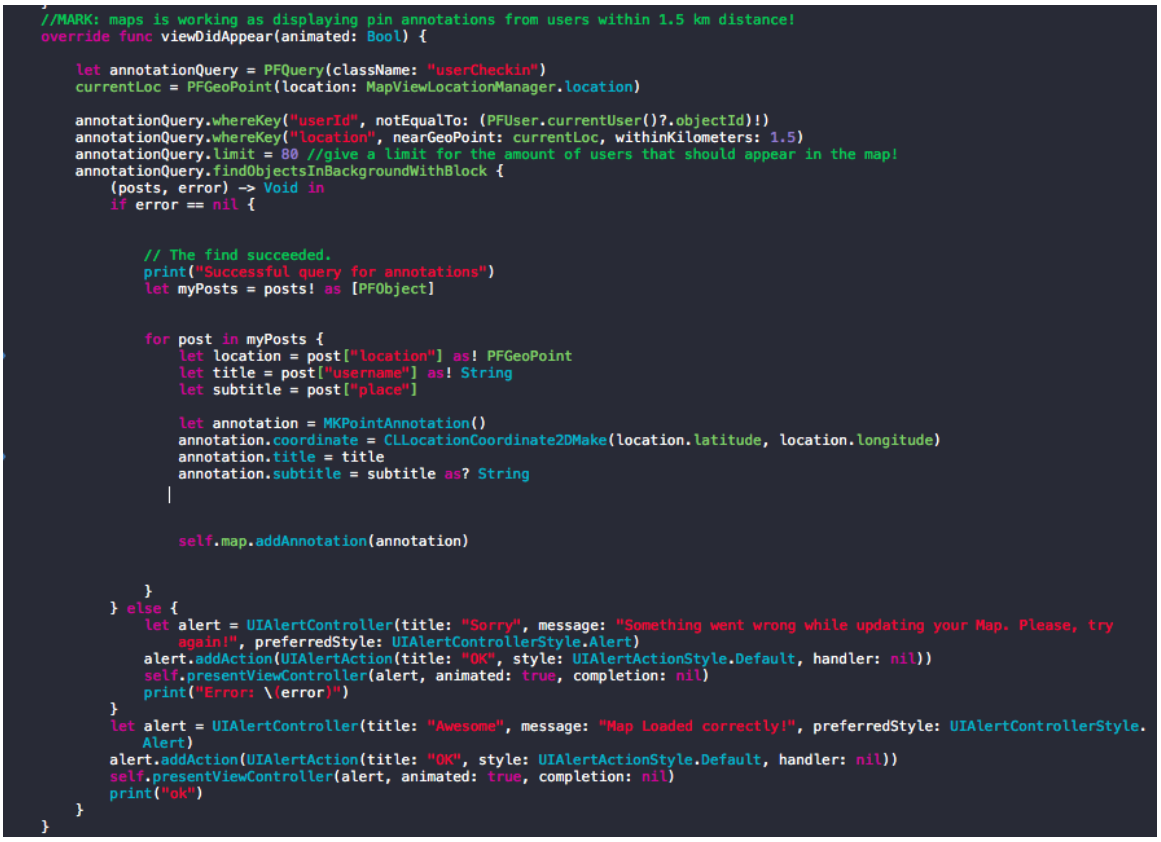
Figure 1 displays that is used to create a Parse query :

- UserId (unique for each user)
- location, within 1.5 km of current user

Adding to data a limit of 20 users that should display on the page.

After that, using "**findObjectsInBackgroundWithBlock**" in case there are no errors code will get the location (to match the 1.5 km search), the title (username) and subtitle (location) which will be later de added to the map using MKPointAnnotation using a pin. In case of errors, alert messages will be displayed to users.

Figure 1:



```
//MARK: maps is working as displaying pin annotations from users within 1.5 km distance!
override func viewDidAppear(animated: Bool) {

    let annotationQuery = PFQuery(className: "userCheckin")
    currentLoc = PFGeoPoint(location: MapViewLocationManager.location)

    annotationQuery.whereKey("userId", notEqualTo: (PFUser.currentUser()?.objectId)!)
    annotationQuery.whereKey("location", nearGeoPoint: currentLoc, withinKilometers: 1.5)
    annotationQuery.limit = 80 //give a limit for the amount of users that should appear in the map!
    annotationQuery.findObjectsInBackgroundWithBlock {
        (posts, error) -> Void in
        if error == nil {


            // The find succeeded.
            print("Successful query for annotations")
            let myPosts = posts! as [PFObject]


            for post in myPosts {
                let location = post["location"] as! PFGeoPoint
                let title = post["username"] as! String
                let subtitle = post["place"]

                let annotation = MKPointAnnotation()
                annotation.coordinate = CLLocationCoordinate2DMake(location.latitude, location.longitude)
                annotation.title = title
                annotation.subtitle = subtitle as? String


                self.map.addAnnotation(annotation)


            }
        } else {
            let alert = UIAlertController(title: "Sorry", message: "Something went wrong while updating your Map. Please, try
                again!", preferredStyle: UIAlertControllerStyle.Alert)
            alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default, handler: nil))
            self.presentViewController(alert, animated: true, completion: nil)
            print("Error: \(error)")
        }
        let alert = UIAlertController(title: "Awesome", message: "Map Loaded correctly!", preferredStyle: UIAlertControllerStyle.
            Alert)
        alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default, handler: nil))
        self.presentViewController(alert, animated: true, completion: nil)
        print("ok")
    }
}
```

Figure 2 shows a function where it creates the userCheckin class with columns. It also contains a line of code that looks if the current user has already checked before and in case user did, this previous checkin will be deleted before saving a new one

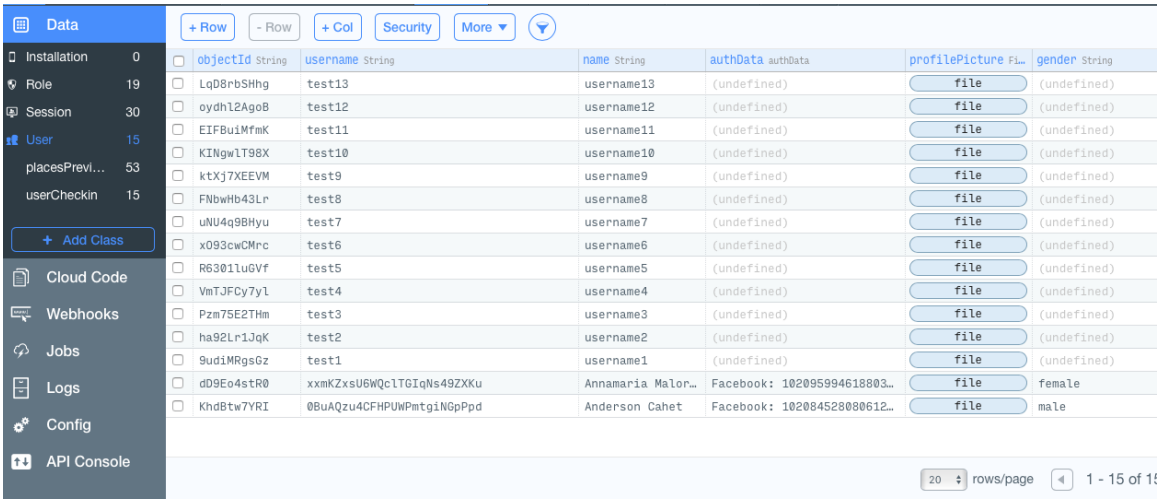
Figure 2:

```
func saveCheckinIntoParse() {
    //create the class that will store the checkin made by users
    let userCheckin = PFObject(className: "userCheckin")
    userCheckin["username"] = PFUser.currentUser()?["name"]
    userCheckin["userId"] = PFUser.currentUser()?.objectId
    userCheckin["geoLocation"] = addressGeoLabel.text! as String
    userCheckin["location"] = PFGeoPoint(latitude: latitude, longitude: longitude)
    userCheckin["place"] = locationNameText.text! as String
    userCheckin["profilePicture"] = PFUser.currentUser()?["profilePicture"]

    if imageSelectorImage.image != nil {

            if let img = imageSelectorImage.image where imageSelected == true{
                let imgData = UIImageJPEGRepresentation(img, 0.2)!
                userCheckin["photo"] =  PFFile(name: "photo.jpg", data: imgData)
            } else {
                self.showErrorAlert("Oooooops", msg: "Something went bad...")
            }
        }
    }

if imageSelected == false {
    self.showErrorAlert("Oooooops", msg: "Please add a picture!")
} else {
    if locationNameText.text == "" {
        self.showErrorAlert("Oooooops", msg: "Please add some text to your location!!")
    } else {

        //working code
        //query to verify if current users has checked already, in case user had, it will be deleted from server
        let query = PFQuery(className: "userCheckin")
        query.whereKey("userId", equalTo: (PFUser.currentUser()?.objectId)!)
        query.findObjectsInBackgroundWithBlock ({( previousCheckin, error) in

            if error == nil {
                if let objects = previousCheckin {
                    if objects.count >= 1 {

                        for object in previousCheckin! {
                            object.deleteInBackgroundWithBlock({ (success, erro) -> Void in
                                if erro == nil {
                                    self.showErrorAlert("Please Wait", msg: "Previous checkin is being removed to save a new one")
                                    //usercheckin is saved in parse
                                    userCheckin.saveInBackgroundWithBlock{ (success, error) -> Void in
                                        if success {
                                            print("deleted checkins before add a new checkin")
                                        } else {
                                            self.showErrorAlert("Sorry", msg: "Something went wrong while processing your checkin. Please try Again!!")
                                        }
                                    }
                                }
                            })
```

Figures 3 shows a function to add fake users so app can be tested within different situations. Figure 4 shows the User class in Parse.com with 2 users that signed with Facebook login and the others with usernames and profile pictures save for testing.

Figures 3, 4:

```
func addFakeUsersForTest() {

    let urlArray = ["https://itfinspiringwomen.files.wordpress.com/2014/03/scooby-doo-tv-09.jpg", "http://static7.comicvine.com/uploads/square_small/0/2617/103863-63963-torongo-leela.JPG", "
        http://static.makers.com/styles/homepage_carousel/s3/field/image/lisa-simpson-cartoon-576km071213.jpg?itok=WiYL2FoE", "https://s-media-cache-ak0.pinimg.com/236x/9c/5e/
        86/9c5e86be6bf91c9dea7bac0ab473baa4.jpg", "https://s-media-cache-ak0.pinimg.com/236x/9a/f1/6a/9af16a2c96ffbe92a2bff0e451745f48.jpg", "http://www.polyvore.com/cgi/img-
        thing?.out=jpg&size=l&tid=44643840", "https://allcartooncharacters.files.wordpress.com/2014/11/dee-dee.jpg", "https://metrouk2.files.wordpress.com/2015/02/ad_158595188.jpg", "https://s-
        media-cache-ak0.pinimg.com/736x/7e/e2/af/7ee2af286de01181330f3fdaf1ec296a.jpg", "http://f.tqn.com/y/animatedtv/1/L/1/m/simp2006_HomerArmsCrossed_1.jpg", "http://cdn.pastemagazine.com/www/
        blogs/lists/2010/05/13/spofy.jpg", "http://farm4.static.flickr.com/3629/3322479894_eddec3e8cb.jpg", "http://cdn2-b.examiner.com/sites/default/files/styles/image_content_width/hash/be/aa/
        beaa5b0e55f1f8a9f87eede992c2c5da.jpg?itok=fY7VE-vh"]
    var counter = 1

    for url in urlArray {
        let test = NSURL(string: url)
        print(test)

        if let data = NSData(contentsOfURL: test!) {
            self.profilePicture.image = UIImage(data: data)
            let imageFile:PFFile = PFFile(data: data)
            var user:PFUser = PFUser()
            var username = "test\(counter)"
            user.username = username
            user.password = "pass"
            user["profilePicture"] = imageFile
            user["name"] = "username\(counter)"

            counter = (counter + 1)

            user.signUp()
        }
    }
}
```



2.4 Testing

2.5 Graphical User Interface (GUI) Layout

Figure 1: App Icon displayed on an iPhone smartphone.

Figure 2: Launch screen

Figure 3: Main Page after login with Facebook

Figure 4: Check in button is pressed and user is shown the map with option to check in.

Figure 5, 6, 7: Nearby button is pressed and map is populated with pins regarding the users and users are displayed in a table view displaying the distance from the current user together with profile picture and usersame.

Figure 8: Post button is pressed and a list with users is bigger radius are displayed.

Note that map is provided by MapKit framework and location by CoreLocation. User location is saved in Parse as GeoPoint.

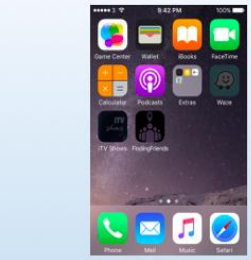


Fig. 1

Fig. 2

Fig. 3

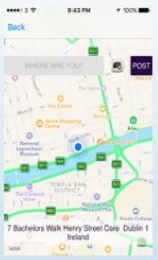


Fig. 4

Fig. 5

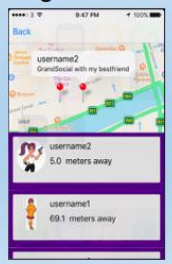


Fig. 6

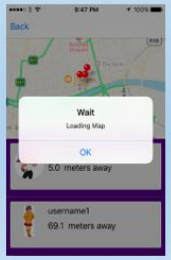


Fig. 7

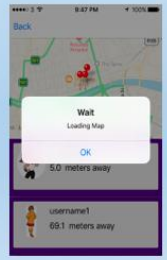


Fig. 8

2.6 Customer testing

Testing was done with 2 extra iPhones where the layout would change between versions and also the behavior would be different in terms of functionality, but that could be related to the provider.

2.7 Evaluation

Feedback received was that even if it has a good idea behind, the application could have a better layout and other functionalities could be add to project such as chat, option to like other post and have pictures instead of pins in the map as it looks too simple.

3 Conclusions

4 Further development or research

Pin annotation can be changed to custom annotation with user Facebook profile picture.

In the post view, extra information could be added and such as like buttons, comment sections which would be erased once user would check in another place the previous location is deleted.

Before add a new location users could verify locations that were previously used in their location, making than just need to choose for a table view and after having to add a picture to finish checkin.

The project could lead to a chat where users nearby could chat among each other in private or in public, where users could have a big chat room available to all that would be displayed in the check in options.

Also with more time and staff other segments of this app could be created based in different groups of society.

5 References

https://www.parse.com/docs/ios/guide#objects-retrieving-objects (Accessed: 10 April 2016).

https://www.parse.com/docs/ios/guide#queries (Accessed: 10 February 2016).

Swift Course (no date) Available at: https://search.itunes.apple.com/WebObjects/MZContentLink.woa/wa/link?path=StanfordSwift (Accessed: 3 February 2016).

The swift programming language (swift 2.1): About swift (2015) Available at: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html (Accessed: 3 February 2016).

The complete iOS 9 developer course - build 18 iOS9 Apps (2016) Available at: https://www.udemy.com/the-complete-ios-9-developer-course/learn/#/ (Accessed: 3 February 2016).

Keur, C., Hillegass, A. and Conway, J. (2014) *Ios programming: The big nerd ranch guide*. 5th edn. United States: Pearson Education (US).

Xcode 6 swift multiple segues with prepare for segue (2016) Available at: http://stackoverflow.com/questions/28175069/xcode-6-swift-multiple-segues-with-prepare-for-segue (Accessed: 19 December 2015).

How do I convert a swift array to a string? (2016) Available at: http://stackoverflow.com/questions/25827033/how-do-i-convert-a-swift-array-to-a-string (Accessed: 17 December 2015).

Swift get directions in maps app (2016) Available at: http://stackoverflow.com/questions/28547492/swift-get-directions-in-maps-app (Accessed: 17 December 2015).

Parse (no date) Available at: https://www.parse.com/questions/limit-push-notifications-by-geolocation (Accessed: 15 December 2015).

6 Appendix

6.1 Project Proposal

6.2 Project Plan

7 Objectives

The aim of this project is to create an application where users are going to be able to locate friends that checked in areas nearby. This will be possible using the checkin button, which use the user's location through the geolocation of their mobile phones, if swiched on.

Before posting the location, user will also be able to name it. Example : Theater, Restaurant, museums, bars, etc.

The user can login using the Facebook login button and after that, the application will display in a map all his friends that have done previously the checkin and, also random, people that have done the checkin using Meet Friends.

There will be one tab where Users can have displayed only their friends that checked in nearby in the map and another tab where it will be possible to visualize others who also have downloaded the application.

8 Technical Approach

At the beginning this application will be developed for iOS, afterwards can be expanded to Android and Windows phone.

In order to achieve the final product, I will need to learn Swift language, which is used for iOS development.

To have Meet Friends working, it will be also necessary that users have, at the same time, both their GPS working on their mobile and a Wi-Fi connection,

or the network provided by their mobile operator active to deploy correctly Meet Friends.

In future there will be an implementation of a public chat, where all Users can chat among each other and/or specific user or group, and in this case a Wi-Fi connection will be necessary. The User who starts the chat will be the host of the conversation.

9 Special resources required

In order to developer this App, the following hardware/ Software is necessary:

Mac or a PC with a VMware or Virtual Box in order to deploy Xcode;

Knowledge of Swift language

IOS device for test purposes;

Functional checklist, that will be developed once application is running;

Parse.com

10 Project Plan

Gantt chart using Microsoft Project with details on implementation steps and timelines

11 Technical Detail

The language that will be used is the Swift language.

I will also use Facebook login button (Facebook API), Geolocation, Maps.

I will use parse.com to store users login and others activities such as checking that will be displayed in the map for other users that want to check who is closer to his destination, that could be friends, or just other users that he/ she wants to join.

12 Evaluation

Going through social networking asking people to download the App and send me a notification via email where they can answer a survey with a feedback regarding design, functionality and improvements that would help the application.

As a Localization QA tester, I will ask my colleagues to fill a small checklist, created in order to get a better understand of the issues that the system could have, so I can fix them on time for the final presentation.

12.1 Monthly Journals

12.2 Other Material Used

Any other reference material used in the project for example evaluation surveys etc.

CD containing code should be glued to the technical report.

(**Only applicable to the Final Report in May 2016**)