6-11-2014

NameBuzzer

Diane Keng
Santa Clara University

Haiwen Chen
Santa Clara University

Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 11, 2014

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Diane Keng and Haiwen Chen

ENTITLED

NameBuzzer

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THESIS ADVISOR

DEPARTMENTCHAIR

NameBuzzer

by

Diane Keng and Haiwen Chen

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 11, 2014

Table of Contents

Table of Figures and Graphs

1. Purpose

We have the approval from the advisor, Dr. Joanne Holliday on developing a mobile app called NameExpert.
The advisor is on the faculty of the Santa Clara University Computer Engineering department since September 2000.

2. Abstract: NameExpert

The NameExpert is an iOS mobile application that will help users remember the names of new acquaintances. Many people often experience situations where they would run into someone they have been introduced to before, but could not remember the name to. Situations like this tend to instigate feelings of frustration and awkwardness between both parties. Names are important as part of identity and so the NameExpert would allow individuals to connect names to things like location of the meet, appearances, and other systematic ways. The NameExpert would utilize an algorithm that is derived from a proven psychological research regarding the human memory to help the individual retain the new names.

3. Problem: Forgetting

The big question people face is “How much do people forget?”
And the answer is “it depends.”
There is huge complexity in human learning and how our brains absorb information. Things that could play a role include the type of material that is being learned, the learners’ prior knowledge, the learners’ motivation to learn , the power of the learning methods used, the contextual cues in the learning and remembering situations, the amount of time the learning has to be retained, the difficulty of the retention test, etc.

More meaningful materials (like stories or features) tend to be easier to remember than less meaningful material (like nonsense syllables in a name). More relevant concepts tend to be easier to remember than less relevant concepts. Learners who have more prior knowledge in a topic area are likely to be better able to remember new concepts learned in that area. More motivated learners are more likely to remember than less motivated learners. Learners who receive repetitions, retrieval practice, feedback, variety (and other potent learning methods) are more likely to remember than learners who do not receive such learning supports. Learners who are provided with learning and practice in the situations where they will be asked to remember the information will be better able to remember. Learners who are asked to retrieve information shortly after learning it will retrieve more than learners who are asked to retrieve information a long time after learning it.

4. NameExpert: Differentiation

- Note-Feature: This is a section that is customizable to the user. This area is for the user to input information regarding when, where, and first thing noticed upon introduction. Currently, users can utilize the basic note feature innate with the iOS environment. However, then it does not include any intelligence to help the user remember the individuals they had just wrote about.
- App Notification: The mobile application will send notifications to remind the user of people that were met. The user will have the option to select how often the local notifications will arrive. In addition, the application will allow a user to determine whether or not the newly met individuals are ingrained into the memory yet.
- Profile: Each user will be able to input a picture and a name. Information such as email, phone number, and others would be optional.
- Easy-Share: Users with the mobile application can easily share profiles to simplify information exchange

5. Requirements

After understanding the current problem with remembering names, a list of functional and nonfunctional requirements for the new system. The functional requirements describe what must be implemented in the system, while nonfunctional requirements qualify or define the manner in which the functional requirements are met. Figure 4-a highlights the collected requirements.

Figure 5-a: Table of Requirements

Functional	· Log in - Users are able to log in with email account · Log out - Users are able to log out · Sign up - Users are able to sign up account · Note-taking function · Email notification function · Profile sharing function with Security protected
Nonfunctional	· Symplicity to use. Users should not have the need to consult others on how to use this application · Privacy. Users private information should be protected if they disable the permission to share. · Customizability to each different users. they could edit their personal setting to decide what to share. · Usability, by targeting user community.

5.1 Sign Up/Log In Authentication

The application needs to accept an email address and a password for a new user to sign-up for the application. Our system needs to be able to maintain this duo so that users can re-sign into the application.

It also gives user option to sign up with their facebook account. In that way, users can get rid of the hassles to manually type in information which might prevent them from further using the app. Sign in with facebook account will also give us more useful data for future development of our app. After sign-up with their facebook account, they could just log in without entering any information. In other words, after use click the app, it logs into directly to the main page of the app. We will automatically grasp user's name and profile picture for their information.

5.2 Log Out

When a user clicks on "Log Out", the application needs to logout the current account. Then the app will go back to the sign in and sign up page. Users could log in with another accounts.

5.3 Note-Taking

This feature allows users to take notes about the people they just meet. We prompt the users to enter name in a text box. The next notes we prompt user to enter are location and event/occasion. The app encourages the users to associate the names with where and how they meet people. The location is a drop down menu that connects to google API. The app will also note the time of when the contact is saved.

The app will detect the businesses or tourist sites around where the users are in radius of 500 meters. If the users are not able to find a specific location they are meeting people. They can choose to manually type in location.

The app will provide users an extra text space to enter more input based on their meeting experience. Users could type in keywords to describe people for example, green glasses, chubby, fashionable, etc.

5.4 Local App Notification

Based on the memory curve, the app will remind the users those names after the users enter information into the app. The app will give the users local notification in 1 day, 2 days, 5 days, and 15 days. The users will be reminded by the names associating the notes they enter at the beginning, which help help users to refresh their memory and remember people's names in the long term.

If there are two or more than two names that should be reminded on a same day, we only send user 1 notification with all the notes that they need to refresh memory on. The local app notification will be displayed right up corner of the app icon.

The user can choose to not have the notification on specific contacts. However, the user needs to manually turn on the setting to not notify. The default will utilize our algorithm to notify.

5.5 Profile Sharing

The users will have their own profile with information including names, phone numbers and profile pictures. When people both have the app, they can simply share each other's information.

The app detect other people based on our geolocation features. The app will show the user who is around him/her if people allow the app to share their geo locations. User A will grab basic information profile pictures and first name. If they want to get the person's phone number or last name, a request will be sent for approval. .

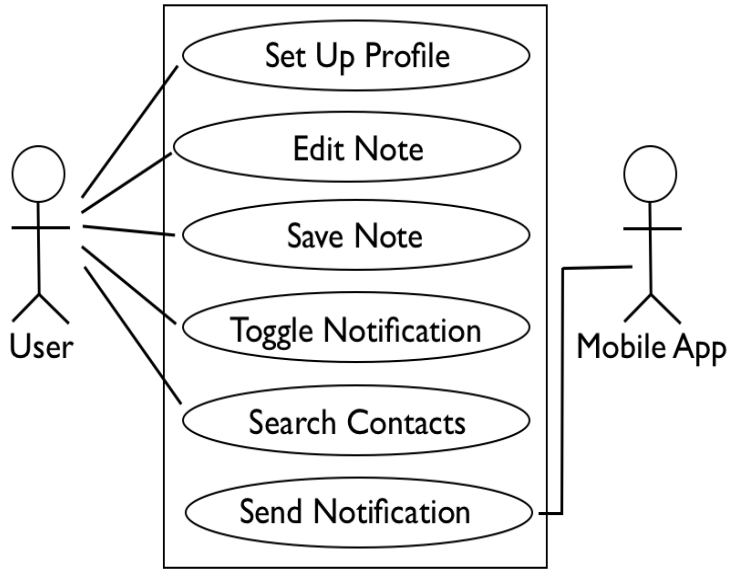
Once a request receives approval, the two users' information will be shared and saved into each other's contacts.

Profile: This feature will need to accept email address, image for a profile picture, and other information. For this milestone, the application will only need a phone number, a job title, and the company the user works at.

6. Use Case

Once the system requirements were collected, use cases were compiled to define the core features necessary for a user to interact with the system. The use cases are meant to provide a detailed understanding of the appropriate procedures and conditions required to complete system tasks.

Figure 6-a: Use Case Diagram



6.1 Set Up Profile

Actor: User
Goal: Set up profile
Preconditions:

- Access to the app on their working iphone
- Access to his/her account
- Access to photos

Postconditions:

- The profile is set up correctly

Scenario:

1. User log in his/her account
2. Click Profile Setting button
3. User click Name text box to type his/her full name
4. User could choose to upload a profile pictures
5. User could choose to enter their phone number
6. User could choose to share all the information or not

Exception:

1. Users upload an image whose extension is not supported

6.2 Edit Note

Actor: User
Goal: Edit Note
Preconditions:

- Access to user's account
- User is on the main page

Postcondition:

- "save" button is enabled
- Note will be populated with keywords and manually added information

Scenario:

1. User Click "Add Contact" button on the main page to get to the Note page
2. User tap name text box to type in the person's name
3. User type in note test box to type in keywords to describe the person
4. User click drop down menu to select where the location is

Exception:

1. Place is not included in the location drop-down menu correctly

6.3 Save Note

Actor: User
Goal: Save Note
Preconditions:

- Note is finished editing

Postconditions

- System back to the main page
- The note is added to the contact

Scenario:

1. User click "save" button to save the person

Exception:

1. System does not catch the "save note" event

6.4 Toggle Notification

Actor: User
Goal: Set Email Frequency
Preconditions:

- Access to the app on their working iphone
- Access to his/her account

Postconditions:

- The notification option is shown on or off

Scenario:

1. User log in his/her account
2. Click Profile Setting button
3. Toggle Notification

Exception:
None

6.5 Search Contacts

Actor: User
Goal: Find the person in the contact
Preconditions:

- The person is in the contact

Postconditions

- The person is found and his information is displayed

Scenario:

1. User type in keywords in the search text box
2. The system finds related people with keywords
3. User select the right person with more keywords displayed
4. The person is displayed with his keywords description, place, time and etc.

Exception:

1. The person is not found because he/she is not in the contact
2. The user typed in wrong search keywords in result of missing the person

6.6 Send Notification

Actor: Mobile App
Goal: Send notification that assists user to remember names
Preconditions:

1. The server is running
2. There are contacts saved in the database that are still need to notify

Postconditions:

1. Notifications are sent to users

Scenario:

1. New entry is saved by user which comes a timer set 1 initially
2. The system notify the user with the information of the newly-met person in one day and increment the timer to 2
3. The system notify the user again in 2 days and increment the timer to 3
4. The system notify the user again in 5 days and increment the timer to 4
5. The system notify the user again in 10 days and increment the time to 5, which means done

Exception:

1. The contact is deleted by user in the process
2. The notification feature is turn off in the process by the user

7. Design

In order to develop this application, we could either create it for the Apple App market or the Android app market. In order to implement the defined features, the Apple route will be selected. Objective C and XCode will be able to successfully provide our users a simple method to remember names and faces. The design will be simple, elegant, and clean. It will promote an user-friendly user interface that allows users to input short keywords about a person they just met.

The application will be downloadable for free through the Apple App market. Users will be responsible for deciding what to share and what descriptions they include for a person. They will be able to add different tags that are associated with that specific user such as “start-up” and “guru” to help remind them of the conversation or other memorable things about that person. The user will have the options of receiving email notifications that will repetitively remind the user of new and past people they have met. Studies have shown that people remember things longer when repetition is in order.

8. Design Rationale

Before deciding on Android or iPhone, an analysis was performed on both.
Research was performed on both approaches to determine how the system will be implemented.
The following discussion analyzes the two approaches against various design considerations.

8.1 Security

Security is quite important. The act of logging in on either the Android or iPhone systems is identical. The user will use either Facebook login or an email and password combo. However, developing with the Objective-C framework seems to be the way to go. It has a framework aimed at doing Facebook connects/sign ins.

8.2 Knowledge of Technologies

Between the Android and iPhone systems, the team is more familiar with the iPhone system. It utilizes Objective-C and Xcode which the team has experience with. In addition, the team has minimal mobile application development skills and the iPhone system provides more documentation than the Android system.

8.3 Accessibility

When it comes to accessibility to download the application from the app market on smartphones, the process on the part of the user would be the same on both systems. They would head to their perspective app markets and download the app. However, from the developers point of view, it is easier to publish an app on the Android system. The Android system requires less authentication to publish. The Android system also takes a shorter amount of time for approval of an app into the market.

8.4 Features and Functionality

Both the Android and iPhone systems provide a variety of features for app validation, app UI, and app back-end programming. Both of the systems will provide customizable functionalities and have built-in functions. However, our team is more familiar with the iPhone environment and so there would be less research regarding built-in functionality so that more features could be enabled on the app.

8.5 User Interface

Both approaches have their own unique view creator. This UI will be sophisticated and clean. The button clicks will need to be minimized in order to increase efficiency.
There is essentially not too many differences between the Android and the iPhone systems.

9. Technologies To Be Used

This system will be implemented with Objective C using the XCode IDE. It will be able to provide the calculations necessary to allow a user to effortlessly save information about new people that they need. It will also allow a user to expand their network and better remember.

The XCode IDE will allow both a coding aspect and the visual designer to create the iPhone app.

In order to ensure the successful creating and development of the iPhone application, Gmail and Git will be used to support collaboration and communication and version control among group members. Each time the code is updated or edited, the group member will recombine it with the Git branch. We will do a check out from the master branch. Git will maintain old copies and allow for merges.

To maintain all the user information, we utilized MYSQL databases.

10. Prototyping Plan

This system helps people to remember names in a systematic way. The platform that we are developing on is IOS mobile environment. XCode IDE is required to be installed on developers laptop. The team has applied for Apple developer accounts. Next we need to reserve server hosting space. Currently, we are utilizing GoDaddy. In the future when we hit 2,000 users, we will convert to Amazon S2.

Before we began prototyping, we are taking the time to create the specification sheet, wireframes, and requirement description analysis. The requirement description is about completed.

To prototype, we will begin by creating the app without the database. We will work on the note-taking and search capability first. Then we will create the profile window within the application. Then, we will connect a MYSQL database so that it can store information from the profile and sign-up/log-in authentication. Lastly, we will implement the push notifications for memory retention. At the moment we are still doing research and testing regarding the “forgetting curve.” Thus, the algorithm has yet to be perfected by us.

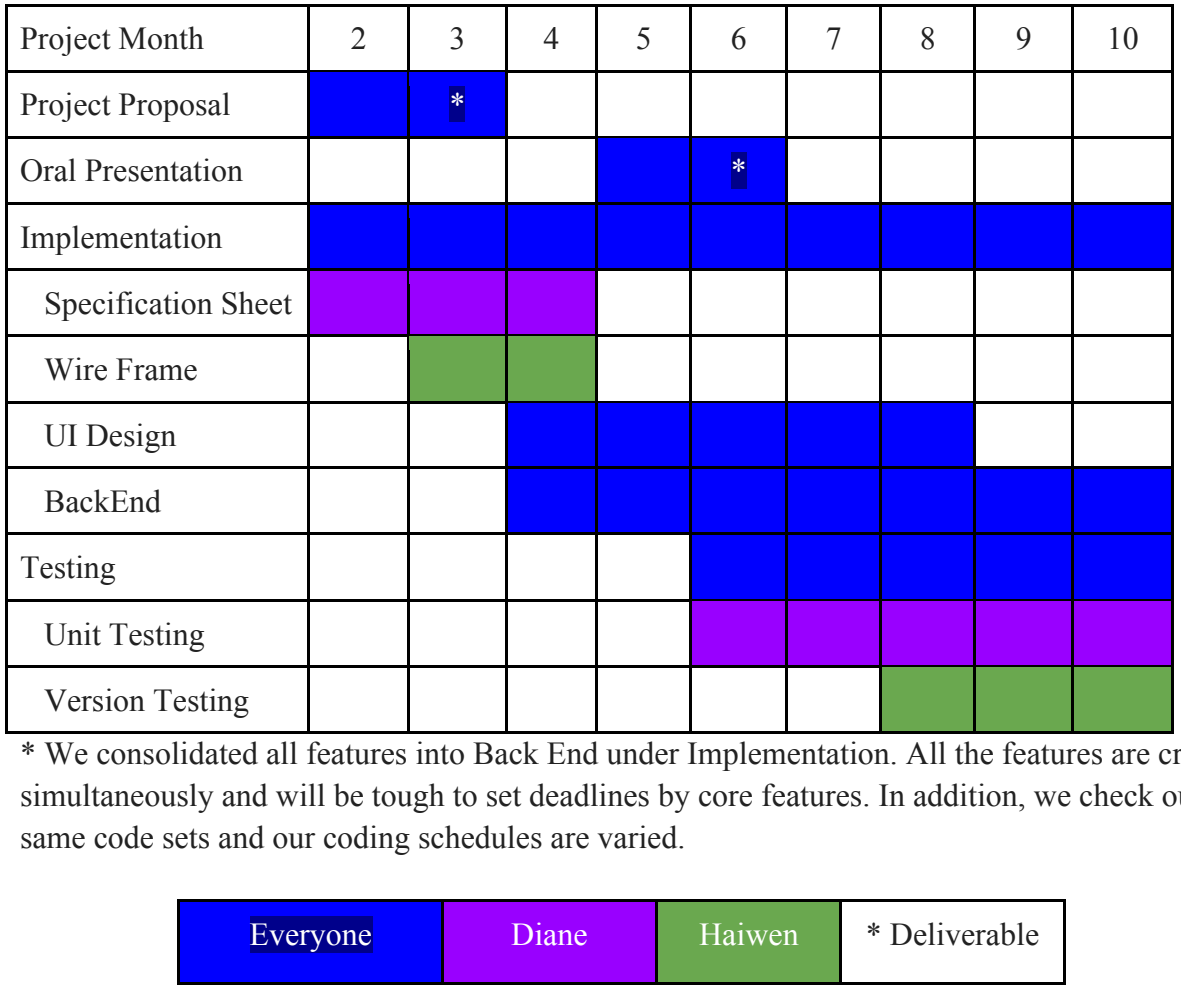
Each functionality (note, search, and notification) could be developed as separate modules. For prototyping purposes, we would develop the note feature first. By developing the note feature first, we can better determine the search methods that will be utilized. The notification feature can also be developed whether or not our algorithm is prepared or not. There is an algorithm that is being developed for proprietary purposes.

Lastly, we will need knowledge on MySQL in order to maintain our user data. It needs an intense set up on MySQL database to make sure that the processing is efficient. The system will remind users by the frequency we deem suitable for memory retention and hold the authentication data.

11. Development Timeline

The development timeline is crucial to the development of our application and to the success of senior design. For the scope of this Gantt chart, we are only addressing by months

Figure 11-a: Gantt Chart



* We consolidated all features into Back End under Implementation. All the features are created simultaneously and will be tough to set deadlines by core features. In addition, we check out the same code sets and our coding schedules are varied.

12. Project Risk

To ensure the successful implementation of the system, the risks had to be analyzed. The analysis is presented below in Figure 11- a,

Figure 11-a: Project Risk Analysis

Risk	Consequence	Probability (P)	Severity (S)	Impact* (I)	Mitigation Strategy
Team member does not complete assigned tasks	1. Other team members must pick up the uncompleted work 2. Time will be lost 3. Project may not be complete by deadline	0.50	6.00	3.00	1. Keep an open line of communication between all team members. 2. Meet frequently to discuss progress 3. Develop and follow Gantt Chart
Time	1. Project is not completed by deadline	0.40	6.00	2.40	1. Prioritize requirements 2. Cut requirements when necessary
The developed system does not pass testing	1. Modify and retest the system 2. System is flawed by the deadline	0.30	5.00	1.50	1. Develop a test plan 2. Test system as implementation is completed
Customer requirements not fully understood	1. Project will not meet customer standards and needs	0.3	9.00	2.7	Continually communicate with customer to ensure full understanding of requirements
Chosen technology does not provide necessary functionality	1. Project will have to be redesigned 2. Valuable time will be lost	0.10	7.00	0.70	Research technological functionalities before design

13. Professional Issues and Constraints

The biggest issue that is raised in our app is the economic aspect. Our app needs a backend database to store all the user's data. In order to store such a big amount data, we have to pay for server space to run our backend applications and data analytics. For most software development, maintenance cost much more than development, which means that we will spend more money on maintenance. Economic issue is very important in our development.

The second issue is the privacy issue. In order to detect or be detected by other devices, users have to turn on the geo-location sharing functionality. This gives people opportunity to know where the users are and what event they are doing. If these information were somehow hacked, users' privacy will be an issue. In addition, we collect our users' information from their Facebook Connect if they choose to use it as well as from the profile page. This information is not used for monetizing but rather for us to understand our user base more.

14. Cost Analysis

1. Purchase of Apple developer account: 99 * 2 = 198
2. Server Cost for 4 Months: 19.99 * 4 = 79.96
3. SurveyMonkey cost for 100 people: 1 * 100 = 100

The total financial cost will be 377.96 including tax.

15. User Guide

Maybe some floating tips. However, the main idea here is to create a simple and user-friendly interface where the user will be able to understand the concept without too much learning. Most of our documentation will not fall under this realm, but rather how and who is developing which portions of the functionality features.

16. Documentation Status/Conclusion

Before we began developing, we have created the specification sheet, requirement description page, and wireframes. As we develop, we keep track of the obstacles to completing each specification and how we are able to overcome them on our development documentation. So far we have completed a basic format of design report.

As we keep moving on, we will keep tweaking the functionalities and use cases depending on our technical aspects. If one of the functionalities is very hard to develop, we might change to an easier one which will require to modify the document.

We have a separate documentation that lists all the deliverables and keeps track of our progress. The checklist will help us get deliverables done on time and keep reminding us the deadlines. As we move forward, we will finally deliver a final version of design report regarding to our final project when it is done.

Overall, we created a working iOS app called NameBuzzer. We learned that time management is tougher than we thought. Bugs that seem easy turned out to be harder to fix and took more than the allotted time.

Update:
Now that June has arrived, our app is completed and satisfies the basic requirements. We have implemented some extra features

16. 1 Sign-Up/Login View

- Added FacebookConnect, which allows users to sign in with their Facebook account.

16. 2 Note View

- Geo-location features have been implemented to determine where the location is. For example: “You are located at Starbucks” instead of “Longitude: Latitude.” This was implemented with the Google Maps API.
- Event feature is in the middle of being implemented. We were hoping to expand this feature so that the user can find people easier if they were registered for the same event and have the app. This is being implemented with the EventBrite API.
- Picture feature has been implemented. When a the user adds a new contact, they can either take or choose a picture to be associated with that user.

16. 3 Personal Profile

- Picture feature has been added. The user can upload a personal photo of themselves.

16. 4 Search

- Continual Search has been implemented. As the user searches through the profile, the contact list will automatically redraw without having to have a delay.

16. 5 Home Page

- Larger Button Design has been implemented. Instead of smaller NSButtons, we have made each button the half of the whole screen.

Appendix

AppDelegate.h

```
//
//  AppDelegate.h
//  NameBuzzer
//
//  Created by Diane Keng on 18/02/14.
//  Copyright (c) 2014 Keng. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "FBConnect.h"
#import "DataSet.h"
#import <CoreLocation/CoreLocation.h>

//@class TypeNoteViewController;
@class loginViewController;
@class homepageViewController;
@interface AppDelegate : UIResponder
<UIApplicationDelegate,FBSessionDelegate,CLLocationManagerDelegate>
{
    Facebook *facebook;
    DataSet *apiData;
    //  NSMutableDictionary *userPermissions1;
    NSArray *userPermissions;

    CLLocationManager *locationManager;

    CLLocation *mUserCurrentLocation;


}
@property (nonatomic, retain) CLLocationManager *locationManager;
@property (nonatomic, retain) CLLocation *mUserCurrentLocation;

@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) loginViewController *viewController;
@property (strong, nonatomic) homepageViewController *homeviewController;
@property (nonatomic, retain) Facebook *facebook;
@property (nonatomic, retain) DataSet *apiData;
//@property (nonatomic, retain)  NSMutableDictionary *userPermissions1;
@property (nonatomic, retain)  NSArray *userPermissions;
@end
```

AppDelegate.m

```
//
//  AppDelegate.m
//  NameExpert
//
//  Created by Diane Keng on 18/02/14.
//  Copyright (c) 2014 Keng. All rights reserved.
//

#import "AppDelegate.h"
#import "loginViewController.h"
#import "homepageViewController.h"

@implementation AppDelegate

static NSString* kAppId = @"681566068552185";
@synthesize facebook,apiData,userPermissions;//userPermissions1;

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    if([[NSUserDefaults standardUserDefaults] objectForKey:@"myArrayKey"] == nil)
    {
        self.viewController =[[loginViewController alloc] initWithNibName:nil bundle:nil];
       // self.viewController =[[loginViewController alloc] initWithNibName:nil bundle:nil];
        self.window.rootViewController = self.viewController;

    }
    else{
        self.homeviewController =[[homepageViewController alloc] initWithNibName:nil bundle:nil];
        //self.viewController =[[loginViewController alloc] initWithNibName:nil bundle:nil];
        self.window.rootViewController = self.homeviewController;

    }

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    //[self GetuserCurrentLocation];

    facebook = [[Facebook alloc] initWithAppId:kAppId andDelegate:self];
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    if ([defaults objectForKey:@"FBAccessTokenKey"] && [defaults objectForKey:@"FBExpirationDateKey"]) {
        facebook.accessToken = [defaults objectForKey:@"FBAccessTokenKey"];
        facebook.expirationDate = [defaults objectForKey:@"FBExpirationDateKey"];
    }

    return YES;
}

-(void)GetuserCurrentLocation
{
    locationManager = [self locationManager];
```

```
        locationManager.desiredAccuracy = kCLLocationAccuracyNearestTenMeters;
        locationManager.distanceFilter = 10;
        [locationManager startUpdatingLocation];
}

- (CLLocationManager *)locationManager
{
    if (locationManager != nil) {
            return locationManager;
        }
        locationManager = [[CLLocationManager alloc] init];
        [locationManager setDesiredAccuracy:kCLLocationAccuracyBest];
        [locationManager setDelegate:self];
        return locationManager;
}

- (void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation *)newLocation
fromLocation:(CLLocation *)oldLocation
{

    mUserCurrentLocation = [[CLLocation alloc] init];
    self.mUserCurrentLocation = newLocation;
}



- (void)applicationWillResignActive:(UIApplication *)application
{
    // Sent when the application is about to move from active to inactive state. This can occur for certain types of
temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application
and it begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games
should use this method to pause the game.
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state
information to restore your application to its current state in case it is terminated later.
    // If your application supports background execution, this method is called instead of applicationWillTerminate:
when the user quits.
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    // Called as part of the transition from the background to the inactive state; here you can undo many of the
changes made on entering the background.
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was
previously in the background, optionally refresh the user interface.
}

- (void)applicationWillTerminate:(UIApplication *)application
```

```
{
    // Called when the application is about to terminate. Save data if appropriate. See also
applicationDidEnterBackground:.
}

@end
```