Zzazan Final Documentation

Team Members

Jingqiao Chen	jchen109
Junqing Deng	jdeng5
Si Chen	sichen5
Xing Gong	xgong6
Yaojie Feng	yfeng10
Yiming Jiang	yjiang16
Zhujia Liu	zliu32

Table of Contents

Zzazan - Project Description

1. Description

We want to develop an iOS app called “zzazan” that helps users find friends for specific activities in real time. People will receive notifications telling them who are up for the same thing right now from their friends' list. It works as a matchmaker that only notify the people who share the same intent and would like to hangout with each other at the same time.

2. Motivation

A lot of my friends keep posting statuses like "Anyone wanna go to the gym now?" or "Who wants to get lunch?" on various social networks everyday and expect some of their friends to see it right away. That may because they do not want bother texting maybe 10 friends to find out who is available now and willing to join them.

It came to us that it will be great to have an App with the function of finding companions in a more convenient way. By using the App users can find company for something they want to do now easily, and they can avoid being bothered if they are busy or not up for the activity. It is like posting an invisible status to the App and let the App pull the list of potential hangout candidates.

3. Comparison with similar software

Meetup

- Meetup helps users find groups of people who share the same interests meeting up for relatively large events in real life .
- Our app only allow users to share their intentions among friends and focus mostly on daily activities like dinning, exercising and studying. It also cares more about what users would like to do right now instead of planning ahead.

Google Calendar

- Google Calendar helps people plan and share their schedules, or invite and notify contacts of appointments.
- Our app works more like a notifier instead of an invitation sender. It does not try to set up the plan but only tell people who are up for the activity.

4. Programming languages, libraries, frameworks, platforms

Programming languages:

- Objective-C (Application)

Frameworks:

- Parse
- Firebase

Platforms:

- iPhone iOS 8

5. Risks/Challenges

Working in parallel using a new IDE Xcode with GitHub for version control would be a challenge. Deciding on how to divide tasks into branches to allow pairs to work in parallel and merging them back together would also be difficult.

The database schema design would be complicated, since each user needs to maintain an adjacency matrix to store their friends current status, which needs to be updated efficiently in real time.

It may be hard to recruit team members with desired skills. Learning a new language, Objective C makes it hard to get the project started and may slow down the progress of initial stages.

Lack of experience in using iOS Developer Libraries may put obstacles in accomplishing specific App features.

6. Process

We are going to proceed with XP. We were familiar with extreme programming from CS427 and decided it would be a good fit for this project as well.

7. Tools

Version control: GitHub

Project management: Wiki (this is the default, but you can propose some other issue tracking system) Development Tool: Xcode

8. Coding Style

All classes created by ourselves should named with prefix "ZZ" Avoid meaningless variable and function names, for example, "t", "p", "a". Variables and functions should have long and meaningful names.

UC 2

Description

This Use case is for the basic functionalities and UI including Welcome, Login, Signup and Log-out screens. When user launches the application, Welcome page would show up, then user can choose to login or sign up for the application. User can sign-up using email address/ Facebook account. This is implemented by using Parse as database, supporting login of Facebook account. As well as registration by unique email address. The registration would create a new entry in the base, along with other essential attributes including password, friend relations, friend circle comments as so on.

Users can login after registration using their account names and password of Zzazan/Facebook by verifying again the Parse database. User can log-out then sign-in with another account.

Architecture and Design

Login and signup functions are supported by the Parse database. By using the API provided by Parse, the verification and data retrieval can be done efficiently. The following code is an example of using login API by Parse.

In the Parse database, we created a table to save essential data of users including avatar, email address, activity. Another table for friend relations for each pair of friends.

```
- (IBAction)loginButtonClicked:(id)sender {
    [SVProgressHUD showWithStatus:@"Connecting"];
    [PFUser logInWithUsernameInBackground:self.name.text password:self.password.text
                                    block:^(PFUser *user, NSError *error) {
        if (user) {
            [SVProgressHUD dismiss];
            [self presentViewController:[ZZUtility createMainViews] animated:YES completion:nil];
        } else {
            NSString *errorString = [error userInfo][@"error"];
            [SVProgressHUD showErrorWithStatus:errorString];
        }
    }];
}
```

UC3 - Add/Accept/View Friends

Description

A user can add friends by searching at the top of the Contacts view. There will be a "+" button at the top. When the user clicks on it, a new page will pop out and the user can type user name in the search text field. If the user name exists, the user can click add friend to send a notification to him/her. If the user name does not exist, the user will get a notification that tells him/her the username does not exist.

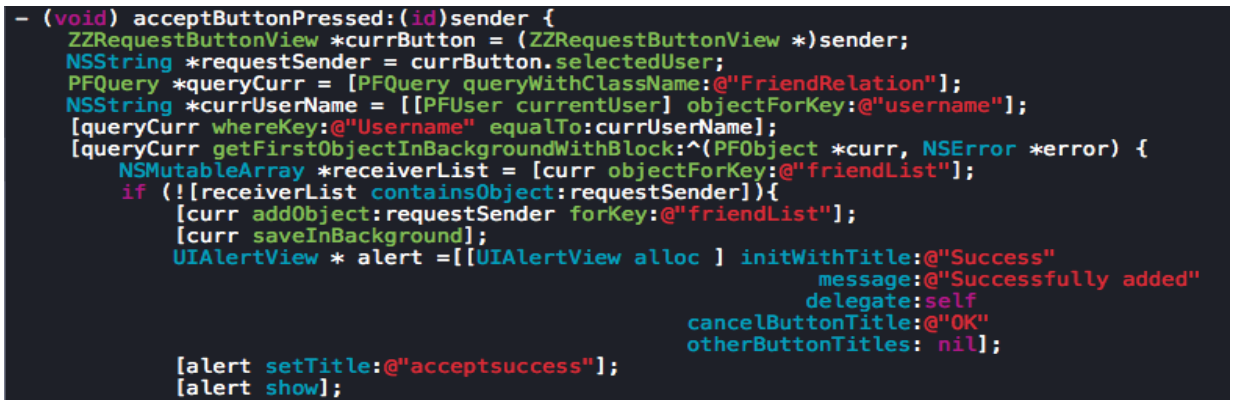
If some other users send friend request to a certain user, after this user log in, he would be able to see friend request notification. By go the friend request mailbox, he can view all the pending friend requests. For each friend quest, he can tap on certain request to view the information of the sender and annotated message. A user can tap on the green accept button to accept the request or swipe the request to left to decline this request.

After two contacts became friends, they can see each other in the contact list view. If the user does not have any friends yet, the list will be blank.

Architecture and Design

We created databases in Parse to store friend requests and relationships. When requesting to add a new friend, the app will check if the two users are friends already. Only if two users are not friends yet a new friend request will be sent.

```
NSArray *checkResult = [object objectForKey:@"friendList"];
if ([checkResult containsObject:requestSender]){
    UIAlertView * friends =[[UIAlertView alloc ] initWithTitle:@"Denied"
                                                       message:@"You are already friends"
                                                      delegate:self
                                             cancelButtonTitle:@"OK"
                                             otherButtonTitles: nil];
    [friends show];
}else{
    PFObject *requestForm = [PFObject objectWithClassName:@"FriendRequest"];
    [requestForm setObject:receiveUer forKey:@"RequestReceiver"];
    [requestForm setObject:requestSender forKey:@"RequestSender"];
    [requestForm saveInBackgroundWithBlock:^(BOOL succeed, NSError *error){
        if (!error){
            UIAlertView * alert =[[UIAlertView alloc ] initWithTitle:@"Success"
                                                             message:@"Request has been successfully sent"
                                                            delegate:self
                                                   cancelButtonTitle:@"OK"
                                                   otherButtonTitles: nil];
            [alert setTitle:@"addsuccess"];
            [alert show];
            PFQuery *userQuery = [PFUser query];
```



```
- (void) acceptButtonPressed:(id)sender {
    ZZRequestButtonView *currButton = (ZZRequestButtonView *)sender;
    NSString *requestSender = currButton.selectedUser;
    PFQuery *queryCurr = [PFQuery queryWithClassName:@"FriendRelation"];
    NSString *currUserName = [[PFUser currentUser] objectForKey:@"username"];
    [queryCurr whereKey:@"Username" equalTo:currUserName];
    [queryCurr getFirstObjectInBackgroundWithBlock:^(PFObject *curr, NSError *error) {
        NSMutableArray *receiverList = [curr objectForKey:@"friendList"];
        if (![receiverList containsObject:requestSender]){
            [curr addObject:requestSender forKey:@"friendList"];
            [curr saveInBackground];
            UIAlertView * alert =[[UIAlertView alloc ] initWithTitle:@"Success"
                                                             message:@"Successfully added"
                                                            delegate:self
                                                   cancelButtonTitle:@"OK"
                                                   otherButtonTitles: nil];
            [alert setTitle:@"acceptsuccess"];
            [alert show];
```

UC5 - Chat

Description

In the contact view, the user can see a list of his/her friends. A user can start a conversation with any of these friends by clicking on the name of the friends. A dialogue will pop out and the user can see the newest chat history. He/she can type text in the text bar and click on “send”. The message will appear in the chat history. The friend will have notifications shown in their device and he/she can see the message just received. He/She can also send messages in the dialogue.

Architecture and Design

Use Firebase the fulfill the real time feature. In Firebase, we have a table for all the chat information. Each dialogue has a “chat room”, the name of the chat room is created by the cancatenation of the two users’ name. Once a new message arrives in the chat room, firebase will send a signal and we will ask Parse to send notification in real time.

```
 */

JSQMessage *message = [[JSQMessage alloc] initWithSenderId:self.senderId
                                        senderDisplayName:senderDisplayName
                                                     date:date
                                                     text:text];


NSDictionary *signal = @{
                         @"reload" : @"YES",
                         @"name": self.senderId,
                         @"text": text
                         };
Firebase *ref = [[Firebase alloc] initWithUrl: @"https://vivid-torch-7789.firebaseio.com/"];
Firebase *alanRef = [ref childByAppendingPath: [NSString stringWithFormat:@"privateChatRoom/%@", room]];
[[alanRef childByAutoId] setValue:signal];
[self.chatData addObject:message];
PFQuery *userQuery = [PFUser query];
[userQuery whereKey:@"username" equalTo:self.senderName];
PFQuery *pushQuery = [PFInstallation query];
[pushQuery whereKey:@"user" matchesQuery:userQuery];
PFPush *push = [[PFPush alloc] init];
[push setQuery:pushQuery]; // Set our Installation query
NSDictionary *data1 = @{
                        @"alert": [NSString stringWithFormat:@"%@: %@", self.senderId, text],
                        @"type": @"message",
                        @"badge": @"increment"
                        };

[push setData:data1];
[push sendPushInBackground];
isFirst = (NSInteger*)1;

[self finishSendingMessageAnimated:YES];
}
```

Figure 5.1 Connect and query the firebase

UC7 - Circle

Description

The scheduled activities are posted on a news feed board called circle. Users can view, like and comment on their friends' past events. The newest feed will be pushed onto the top of the screen. A user can scroll down to view more past events. If a user clicks "like" on accident or if he/she changes mind later, the user can undo it by clicking the "like" button again. Total number of "likes" will be shown near the label.

Architecture and Design

The size of the cell will be chosen dynamically at run time, depending on the length of the content. At the beginning the app will first load 10 feeds, as the user scrolls down onto the 5th cell, 10 new cells will be loaded automatically(if available) to enhance reading experience.

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath {
    int device = [ZZUtility getDeviceModel];
    ZZFeedPost *eachPost = self.dataSourceModel[indexPath.row];

    CGFloat height = 0.0f;
    CGFloat commentViewWidth = 0.0f, commentFontSize = 0.0f, commentMargin = 0.0f;
    CGFloat viewMoreCommentsHeight = 0.0f, likeCommentShareButtonHeight = 0.0f;

    if (device == iPhone5) {
        height = (eachPost.likes != 0) ? kCommentIconToTopWithLike5 : kCommentIconToTopWithoutLike5;
        commentViewWidth = kCommentLabelWidth5;
        commentFontSize = kCommentFontSize5;
        commentMargin = kCommentMargin5;
        viewMoreCommentsHeight = kViewMoreCommentsButtonHeight5;
        likeCommentShareButtonHeight = kLikeButtonHeight5;
    }

    if (eachPost.comments.count > 0 && eachPost.comments.count <= 5) {
        for (int i = 0; i < eachPost.comments.count; i++) {
            ZZComment *eachComment = eachPost.comments[i];
            NSString *eachCommentString = [NSString stringWithFormat:@"%@  %@", eachComment.name, eachComment.content];
            height += [ZZUtility calculateCommentLabelHeight:eachCommentString withViewWidth:commentViewWidth withFont:
                [UIFont systemFontOfSize:commentFontSize]];
            height += commentMargin;
        }
    }
    else if (eachPost.comments.count > 5) {
        height += viewMoreCommentsHeight+commentMargin;

        for (int i = (int)eachPost.comments.count-5; i < eachPost.comments.count; i++) {
            ZZComment *eachComment = eachPost.comments[i];
            NSString *eachCommentString = [NSString stringWithFormat:@"%@  %@", eachComment.name, eachComment.content];
            height += [ZZUtility calculateCommentLabelHeight:eachCommentString withViewWidth:commentViewWidth withFont:
                [UIFont systemFontOfSize:commentFontSize]];
            height += commentMargin;
        }
    }
    height += likeCommentShareButtonHeight;
    height += 20;
    return height;
}
```

Figure 4.1 Calculate the height to draw for the feed

UC40 - Activities

Description

When a user logged into the app, he or she can tap the activity button from the bar on the bottom. The user can scroll down the home screen to see the full list of activities. There are switches right beside the activity name. The user can turn on the status by pulling the toggle and turn it off by pulling the toggle back. After a user press on the switch selector for specific activity on their home screen, the friends of certain user who also open the search for this type of activity will receive certain notification.

There are fixed activities listed in the same order showing on the view.

The trigger happens when the user tap the activity view button at the bottom. User wants to indicate if he or she is interested in the activity now and is looking forward to some friends matches. A user can pull the toggle to the left if the toggle is on the on position. In this way, the status for that specific activity will be turned off. Touching instead of pulling If the user simply touch the button instead of holding and moving it to the right or left, the system will not respond to that and the status will not be changed. Not connected to the internet If the user is not connected to the internet, the system will lock the activities statuses and prevent them from changing on the screen.

Architecture and Design

In each activity toggle, there will be showing the friends list of the user. Each friend name has an activity state corresponding to it. The state uses “YES” and “NO” to indicate the whether the friend is selected.

```
PFQuery *query = [PFQuery queryWithClassName:@"FriendRelation"];
NSString *currentName = [currentUser objectForKey:@"username"];
[query whereKey:@"Username" equalTo:currentName];
[query getFirstObjectInBackgroundWithBlock:^(PFObject *object, NSError *error) {
    NSArray *result = [object objectForKey:@"friendList"];
    for (NSString *friendname in result){
        PFQuery *query1 = [PFUser query];
        [query1 whereKey:@"username" equalTo:friendname];
        [query1 getFirstObjectInBackgroundWithBlock:^(PFObject *object, NSError *error) {
            NSString *status = [object objectForKey:activityState];
            if ([status isEqual:@"YES"]){
                if ([addList containsObject:friendname])
                    state = YES;
                else
                    state = NO;
                ZZActivitiesSelectFriends *entry = [[ZZActivitiesSelectFriends alloc] initWithAvatar:image name:
                    friendname  state:state];
                [self.dataModel addObject:entry];
                [self.tableView reloadData];
            }
        }];
    }
}];
```

Figure 4.1 in appendix is our UML diagram.

UC43 - Matches

Description

The server will check the status of users in real time to find out if two users who are friends are willing to do the same event. A user can receive push notification for a potential match. All match information is displayed in Activity Screen by either time or type.

Server find match and push notification. There must be at least two users and the two users must be friends. And they must turn on the status of the same event in order for a match. Users who are friends and they both turn on the status of the same event will receive notifications. A success scenario will be that one user turn on the status in the event. A friend in his friend list also turns on the status on this event. Both of them will receive a notification. An alternative scenario would be that a user turn on the status in the event while no friends in the friend list turns on the status on this event. The user will not receive notification.

Architecture and Design

The swipe view is provided with to switch between “Time” and “Activity”.

```
#define NUMBER_OF_VIEWS 2

@interface ZZActivityViewController () <SwipeViewDataSource, SwipeViewDelegate, UITableViewDataSource, UITableViewDelegate>

@property (nonatomic, strong) SwipeView *swipeView;
@property (nonatomic, strong) UIButton *timeButton;
@property (nonatomic, strong) UIButton *typeButton;

@end
NSArray *realActivity;
BOOL flag = YES;

@implementation ZZActivityViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    [self.navigationItem setTitle:@"Matches"];
    // Do any additional setup after loading the view from its nib.
    realActivity = [NSArray arrayWithObjects:@"Gym", @"Coffee",@"Eat", @"Study",@"Movie",@"Shopping",@"Anything",nil];


    CGFloat statusHeight = [UIApplication sharedApplication].statusBarFrame.size.height;
    CGFloat navBarHeight = self.navigationController.navigationBar.frame.size.height;
    CGFloat tabBarHeight = self.tabBarController.tabBar.frame.size.height;
    CGFloat buttonHeight = 50;
    CGFloat buttonWidth = [ZZUtility getScreenWidth]/2.0f;

    CGFloat swipeViewHeight = [ZZUtility getScreenHeight]-statusHeight-navBarHeight-tabBarHeight-buttonHeight;

    CGRect swipeViewRect = CGRectMake(0,buttonHeight, [ZZUtility getScreenWidth], swipeViewHeight);
    self.swipeView = [[SwipeView alloc] initWithFrame:swipeViewRect];

    self.swipeView.delegate = self;
    self.swipeView.dataSource = self;

    self.swipeView.pagingEnabled = YES;
```

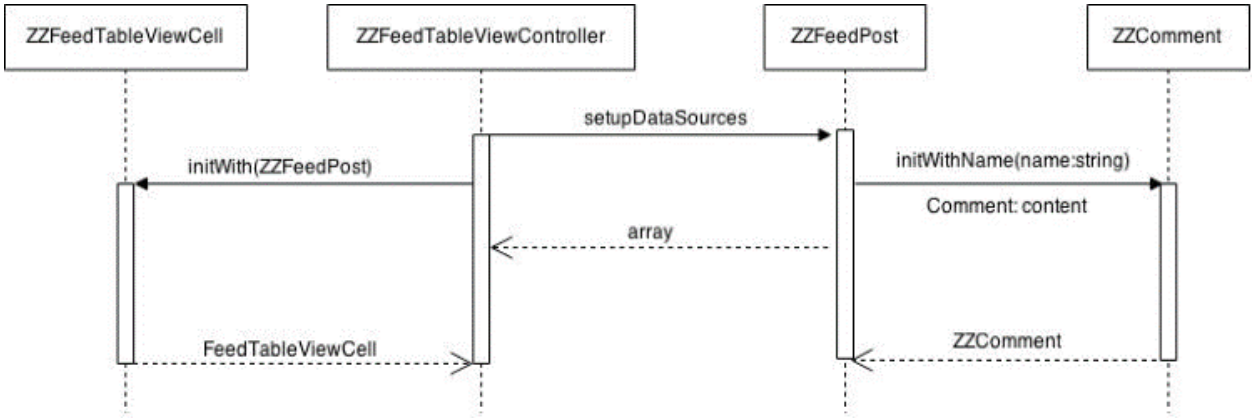


Figure 8.1 - Sequence Diagram for FeedPost

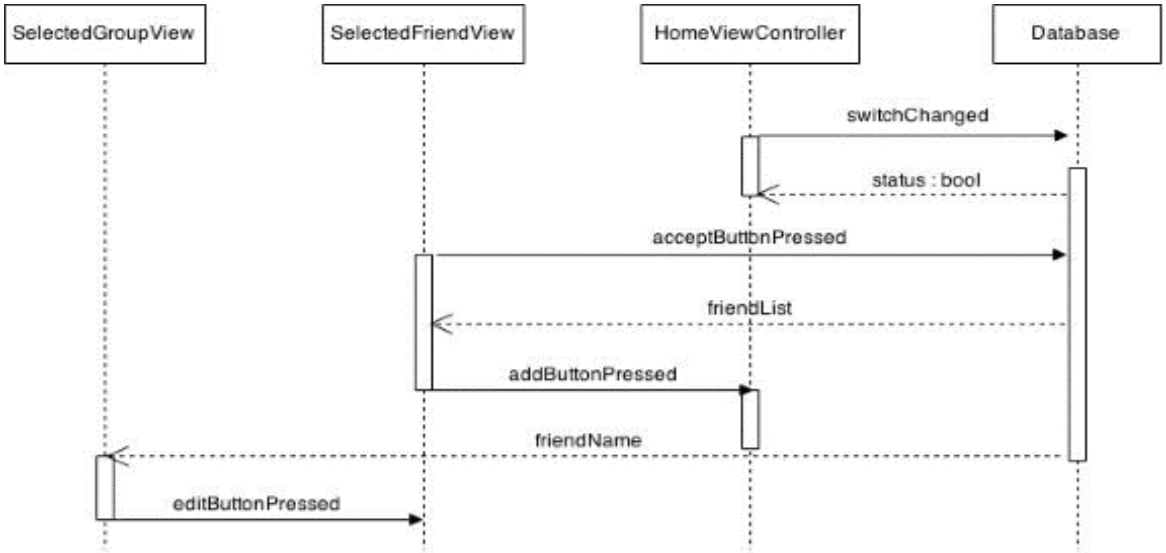


Figure 8.2 - Sequence Diagram for Select Friend View

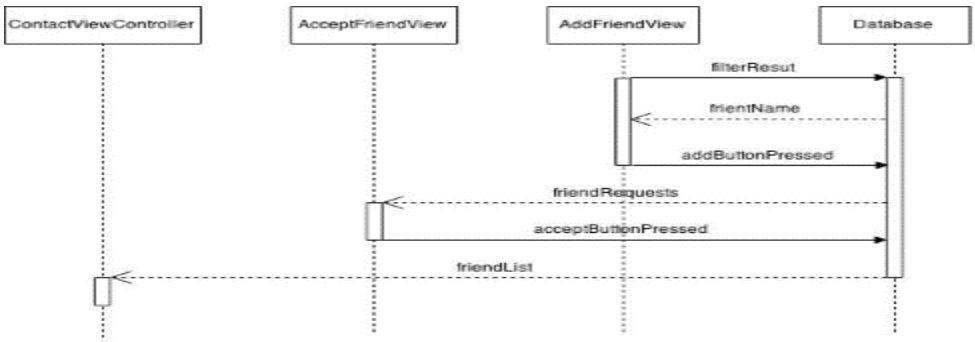


Figure 8.3 - Sequence Diagram for Add Friend View

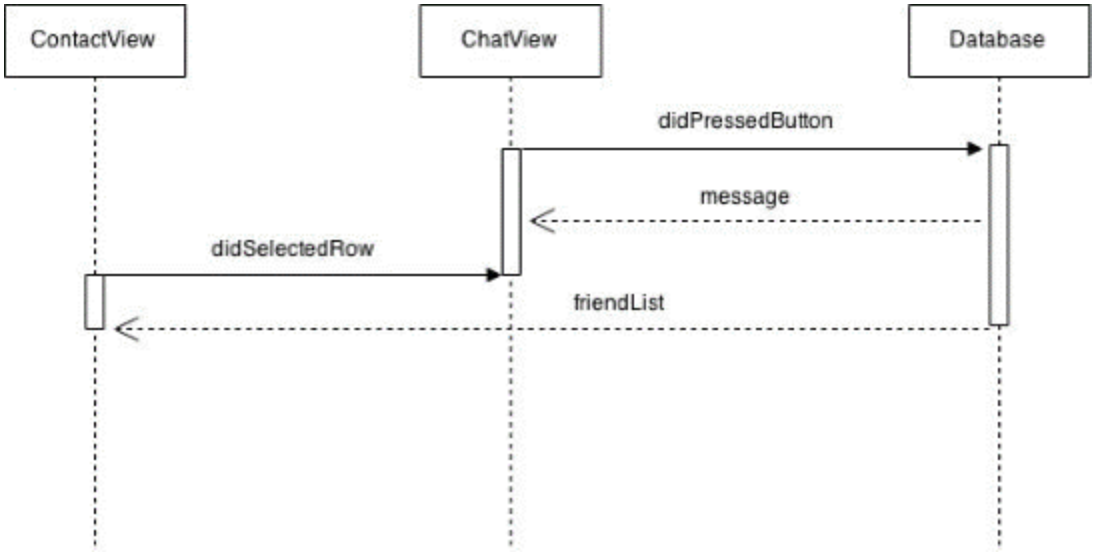


Figure 8.4 - Sequence Diagram for Chat View

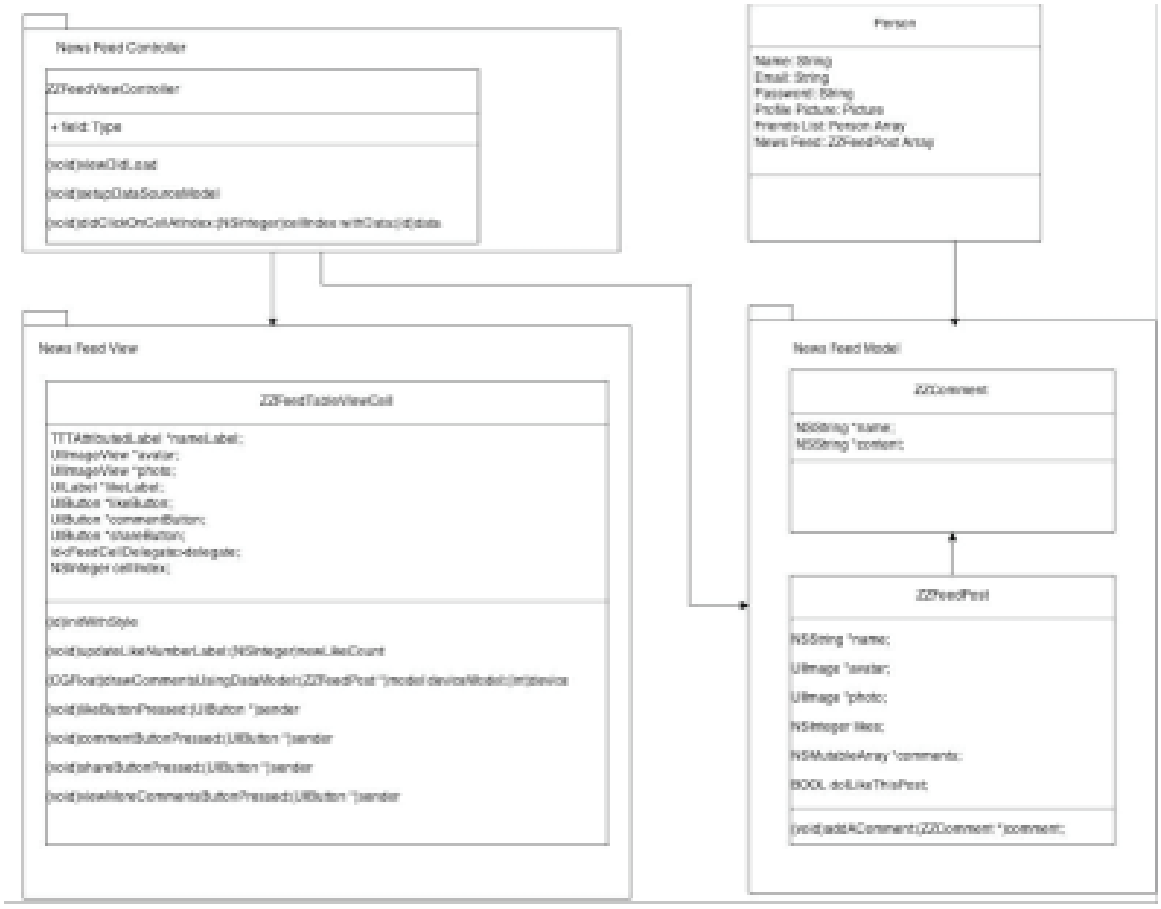


Figure 9.1 - NewsFeed UML

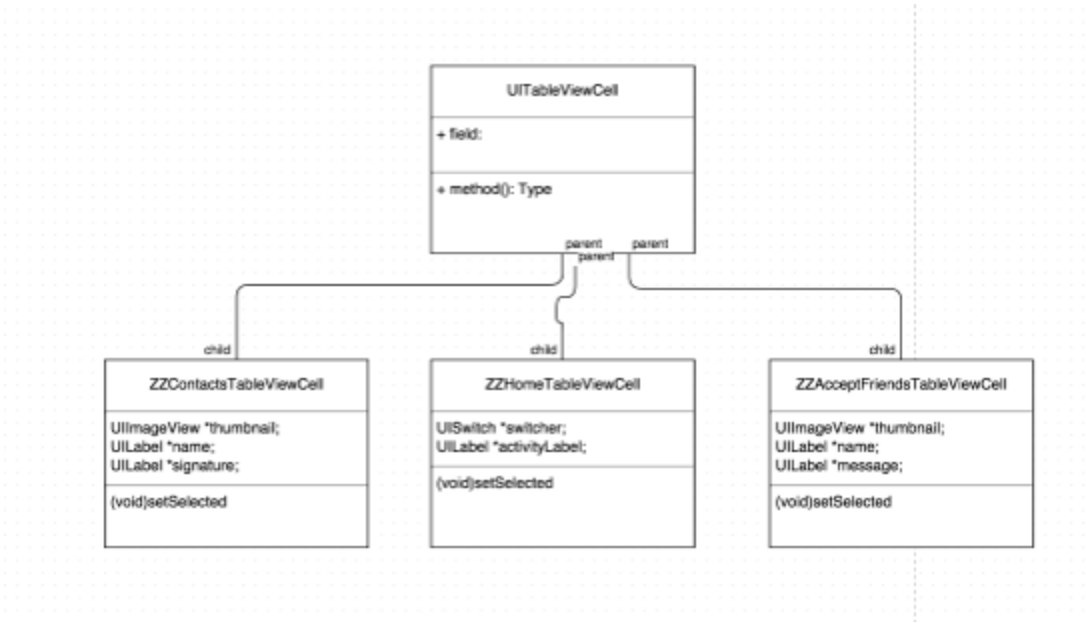


Figure 9.2 - Cell Inheritance

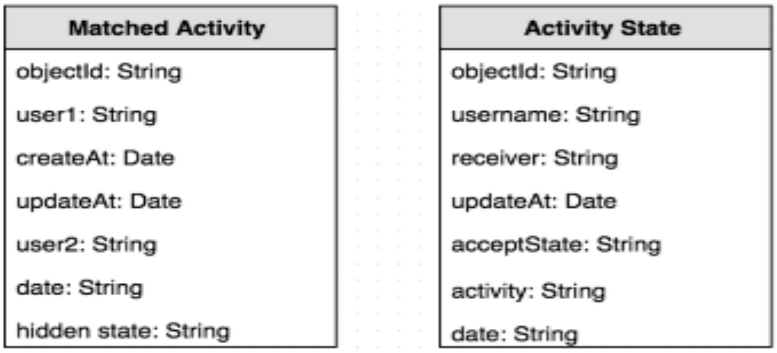


Figure 9.3 - Activity Schema

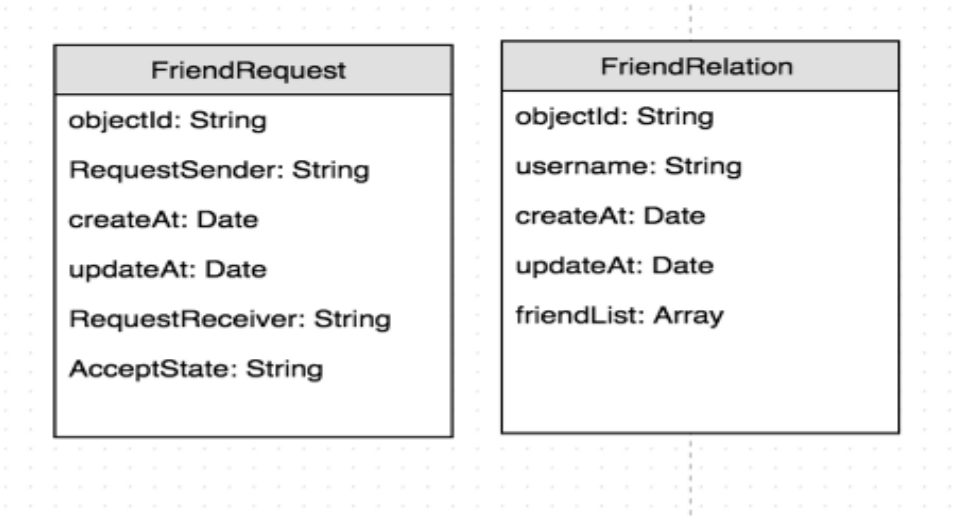


Figure 9.4 - Friend Schema

Self-reflection

Junqing Deng: There have been a lot of practices of coding Objective-C with iOS devices. Learned to use different libraries and research for new technologies in the market. Gained XP coding process experience for future industry development.

Zhujia Liu: I had some experience in IOS programming before CS428. In this project, I learnt the use of Parse and Firebase. I had a better understanding on designing the front end and back end.

Jingqiao: I learned some basic Object-C programming skills and get some taste with Parse API. With the help of the two mentors, I get some idea of the big picture of the whole life-cyle of IOS development.

Yiming: Improved my existing iOS programming skills and integrating the Parse API and Firebase API. I used to do solo iOS development, now I know how to lead a iOS developing team.

Yaojie Feng: I was new to IOS programming before CS428. In this project, I learnt some basic knowledge on IOS design, as well as the use of Parse database.

Xing Gong: This is the first big project we did the brainstorming, development process and UI design all by ourselves. I learned a lot about how to distribute tasks to different teammates based on their expertise. I also learned a lot of techniques of iOS programming and iOS UI design.

Si Chen: In CS428 I learned to design a project from scratch, including brainstorming the ideas, recruiting teammates, dividing tasks into UCs, learning Objective-C as well as XCode, and designing/deploying UI.
