WeRide

Final Project for 08723M

Team 12 - July 27, 2015

Yifan Jia | Chang Liu | Dian Wen | Xiaodong Zhou

Table of Contents

1. Product Overview

1.1 General Description

- Application Name: WeRide
- Target Device: iOS 8+, Android 4+ (ICS +)
- Hardware Requirement: Network accessing, GPS
- Software Requirement: Facebook account, Uber account.

WeRide is a ride-sharing platform for the users to take a ride in a group format. It mainly targets the young people who are friends and wants to share a Uber ride. Usually after the party, work or a meeting, they can take a shared riding, in this way, not only cheaper but also can pays easily. Moreover, the advantages to utilize the Facebook friend relationship in this application is to greatly lower the unpaid case.

The main working flows are as follows. A person, as the group holder, need to first login using his/her Facebook account. And then the holder can choose some friends from the friend-list to form a group together. Once the group is formed, the Uber can be called. During the riding, the holder should perform a corresponding action when anyone gets off. After all the passengers, including the holder gets off, the whole riding is finished and the total bill will shown at the top. The cool thing is, the bills are split and payment requests are sent to each rider through the email separately.

1.2 Key Features

Facebook Login

Figure 1 shows the first page of this application. In this view, the user needs to click the "Login with the Facebook" button and then allow the WeRide application to access to his/her Facebook account.

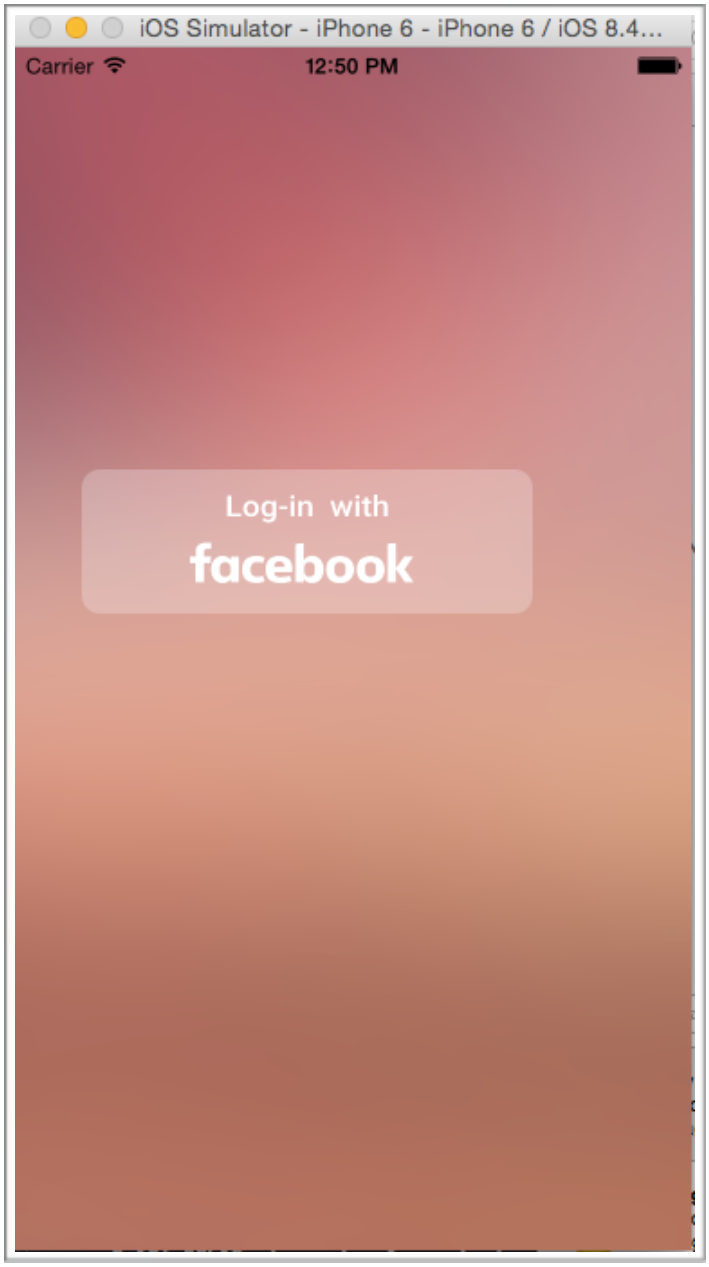

Figure 1. Login Page of WeRide

Friend List Table

This figure shows one of the main views. In table view, all persons, who is not only the holder's Facebook friend and also has allowed the WeRide application, will be listed here. Email of each one just be displayed here, which needs to be utilized later to sent the emails.

From this friend table, the user (holder) can choose whom to share a Uber ride by clicking certain friends. There will be a blue check mark for each group member. Once the group is chosen, the user clicks the "Call Uber" button to continue.

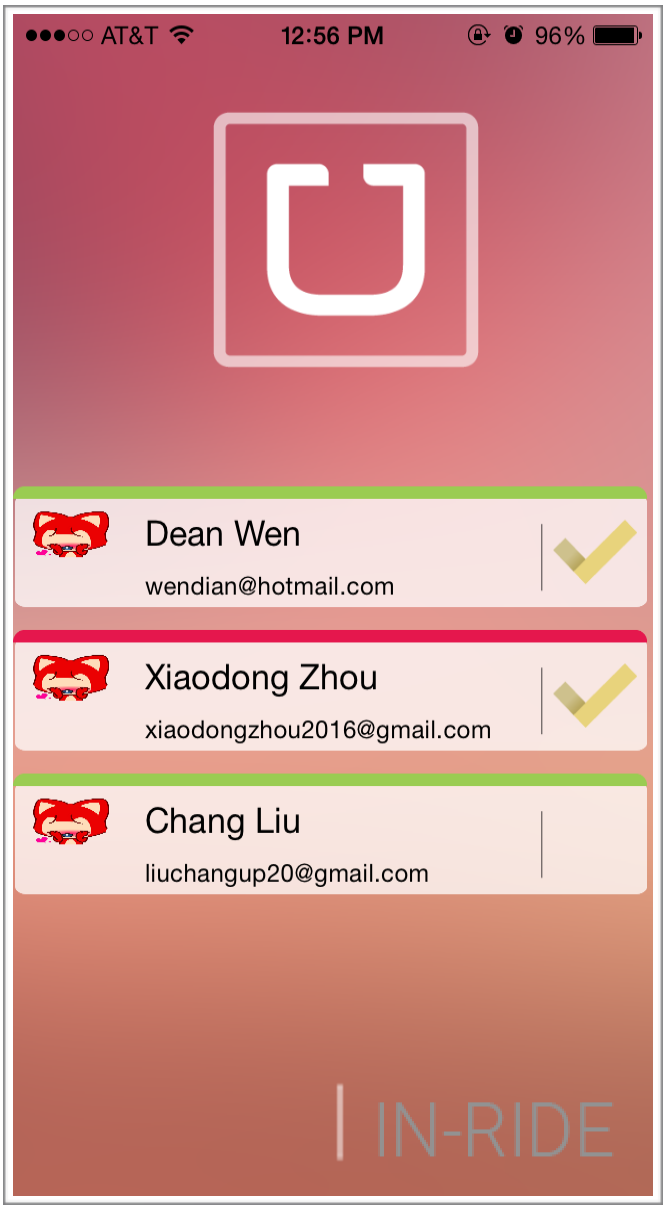

Figure 2. Form a ride group from the Friend List

Call Uber

The user can click the Uber icon (shown in Figure 2) make the uber request, and the interface will jump to Uber app (if the app has been installed) or the Uber webpage (if there is no uber app on the phone).

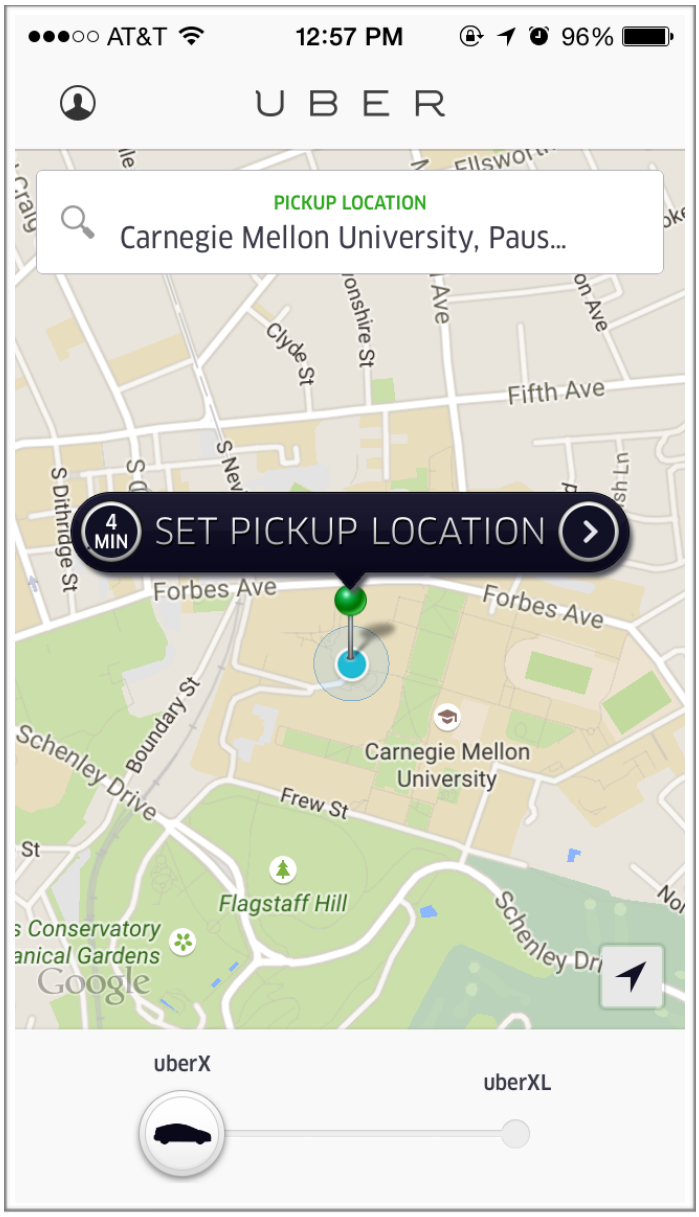

Figure 3. Call Uber

Riding Status

This table view lists the riding status for each group member. During the riding, the corresponding "Get-off" button should be chosen if that ride member gets-off at certain

place. As it is shown in the Figure 4, only Dean gets off in the left diagram, but the right one shows all other riders, except the holder, have got off already.

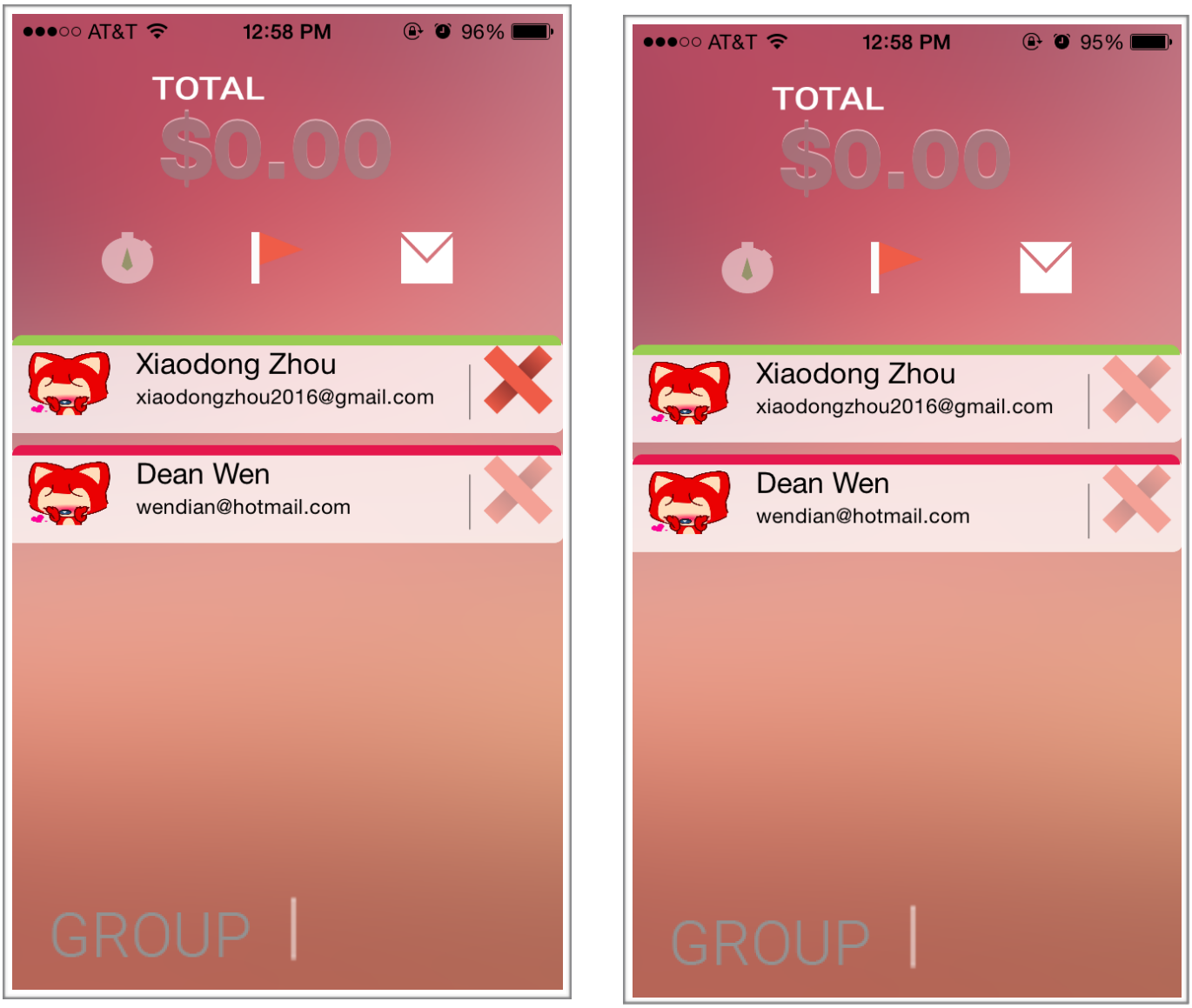

Figure 4. Rider Dean gets off (Left). All other riders have got off (Right)

Split Bill Table

The left diagram shows another important user case in the WeRide application. Once all the riders (including the ride holder) get-off, the whole riding is finished. In this case, the total bill will be shown in the total textview. In this Figure 5, total amount is $5.18.

Moreover, the bill will be automatically separated according to the get-off place and riding time. Once the button is clicked, separate payment request will be sent to each group member through email. The right diagram in Figure 5 shows the payment request sent by the ride holder.

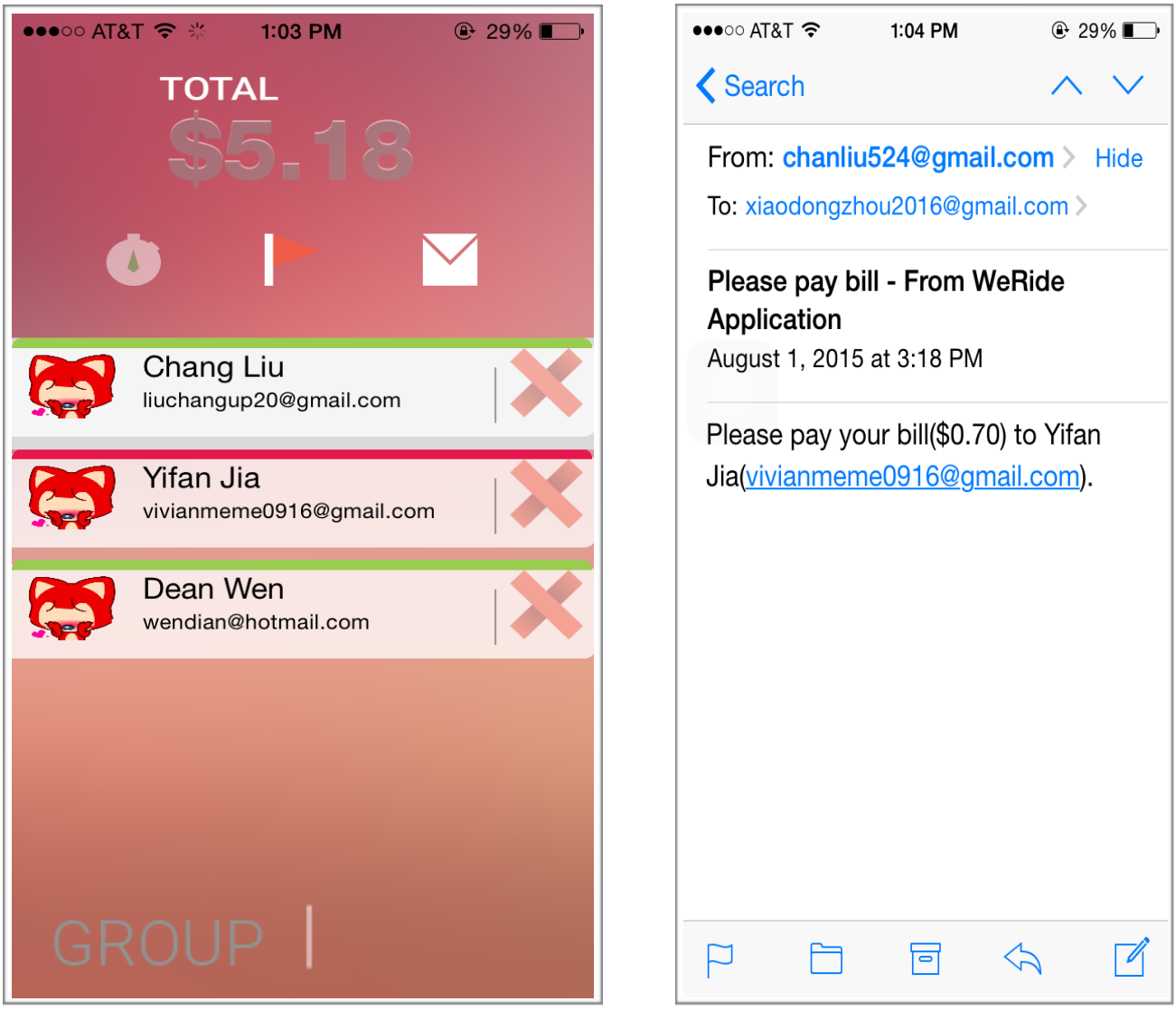

Figure 5. The whole ride finished (left). Receives the bill detail from the holder (right).

1.3 Navigation Flows

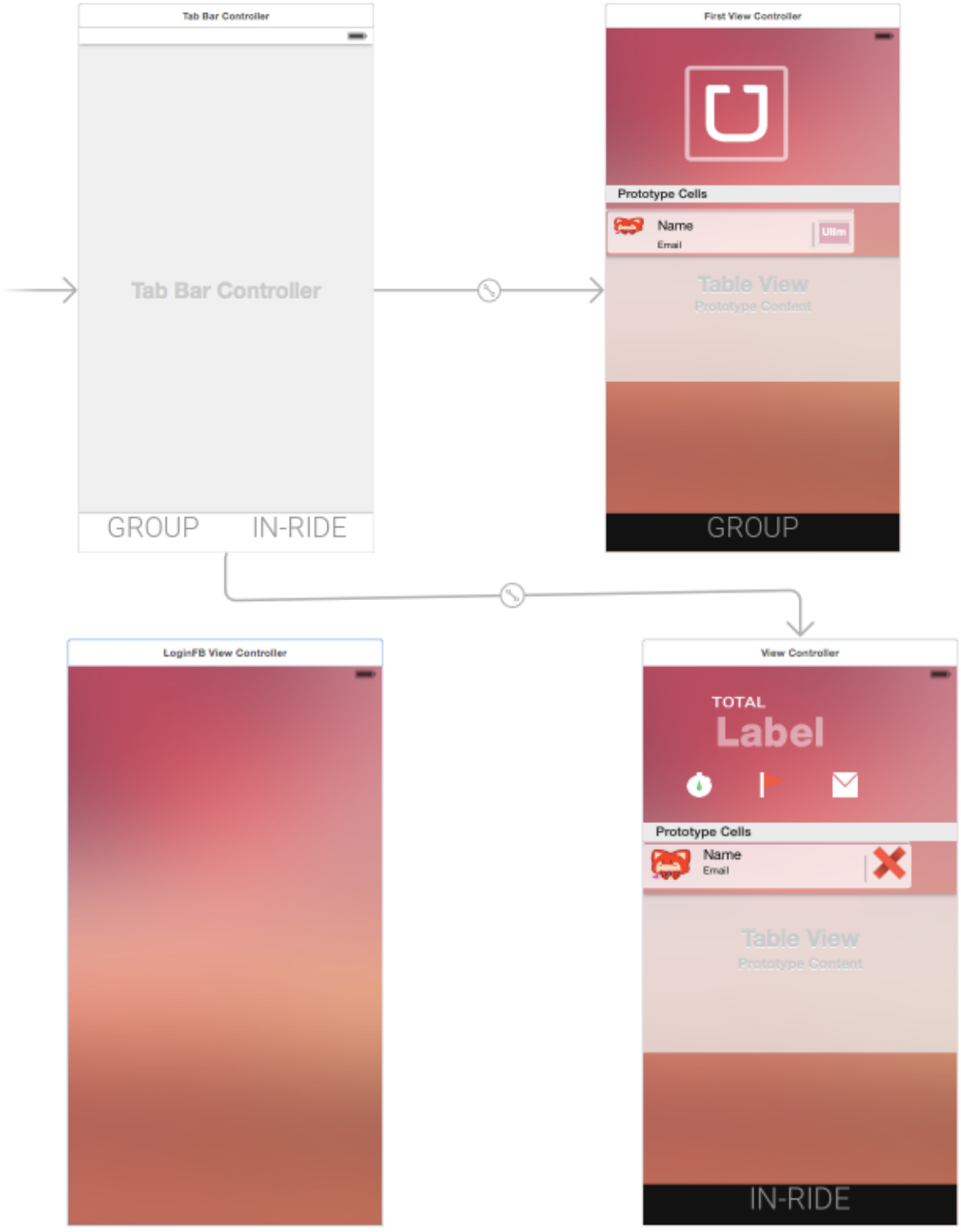

Figure 6. Whole working flow of the WeRide application.

The Figure 6 shows the overall navigation flow of the WeRide application. And the overall structure of this application is based on two-tab.

First, the user needs to login with his/her Facebook account and then shows the first tab-"Group". This first tab is the table view listing all the Facebook friends who has also allowed the WeRide application. By tapping certain table cells, relative friends are dragged into a riding group (shown with a green right mark on the right side). Then the user should click the uber icon at the top to call a Uber ride.

The second tab contains two views. One is the riding status shown during the riding. During the ride, each rider has a fresh right cross mark on the right and once he/she gets off, the ride holder should click corresponding row so that mark will become transparent. While another scene appears once the whole riding is completed. In this view, the total bill will show at the top. Finally, the email icon in the second tab can help the holder separate the bill according to the riding time and distance, and then send all the payments requests to each ride member.

2. Solution Overview

2.1 Design Rationale

- **Social API**

In the WeRide, Facebook API and Uber API are utilized. The former are mainly used to login with this application. Another function is to retrieve the friend list from Facebook. This design way can greatly lower then unpaid risk. For the Uber api, it is the riding media of this application. Also, the implementation way of two operating systems are different. For the iOS version, several FBSDK Kit frameworks are used to provide the function like login, retrieve data, etc. And the UberKit is for utilize the Uber service. Relating to the Android version, we mainly use the facebook_android_sdk_4.0.0 as well as the Auber-java-client for the corresponding use cases. However, the basic working flow is pretty similar, like Oath 2.0 process.

- **Data Handling**

We deploy the different data handling method for the two operating systems. In iOS, the plist is used. Since we only have several pieces of friend record to handle, so plist is a much lighter and more convenient way to implement in iOS. As for Android, the SQLite way is chosen.

2.2 Technological Challenges

- **Complex Facebook configuration**

This is a real trivial thing for configuring both the iOS and Android the Facebook environment. Also when creating the app in the Facebook developer website, there are multiple options for each thing and not easy to choose since the meaning of some items are not so obvious.

- **Facebook Utility Right**

This is a very import and common thing to pay attention to when developing with Social API. For Facebook, only user id and user name can be get using the normal right. To retrieve the user email, further advanced right should be asked. In this case, the application identity not the normal user are used.

- **Asynchronous display**

As for displaying the total amount of bill for the whole ride, we met a problem. Usually, the amount retrieved from Uber side can show correctly in the console; however it may not show in the app (front-end). Later we figuring out this may be an asynchronous problem. When parsing Jason, the activity or controller may extends something like “AsyncTask”, and use the execute.get() to handle the history retrieving.

2.3 Architecture Overview

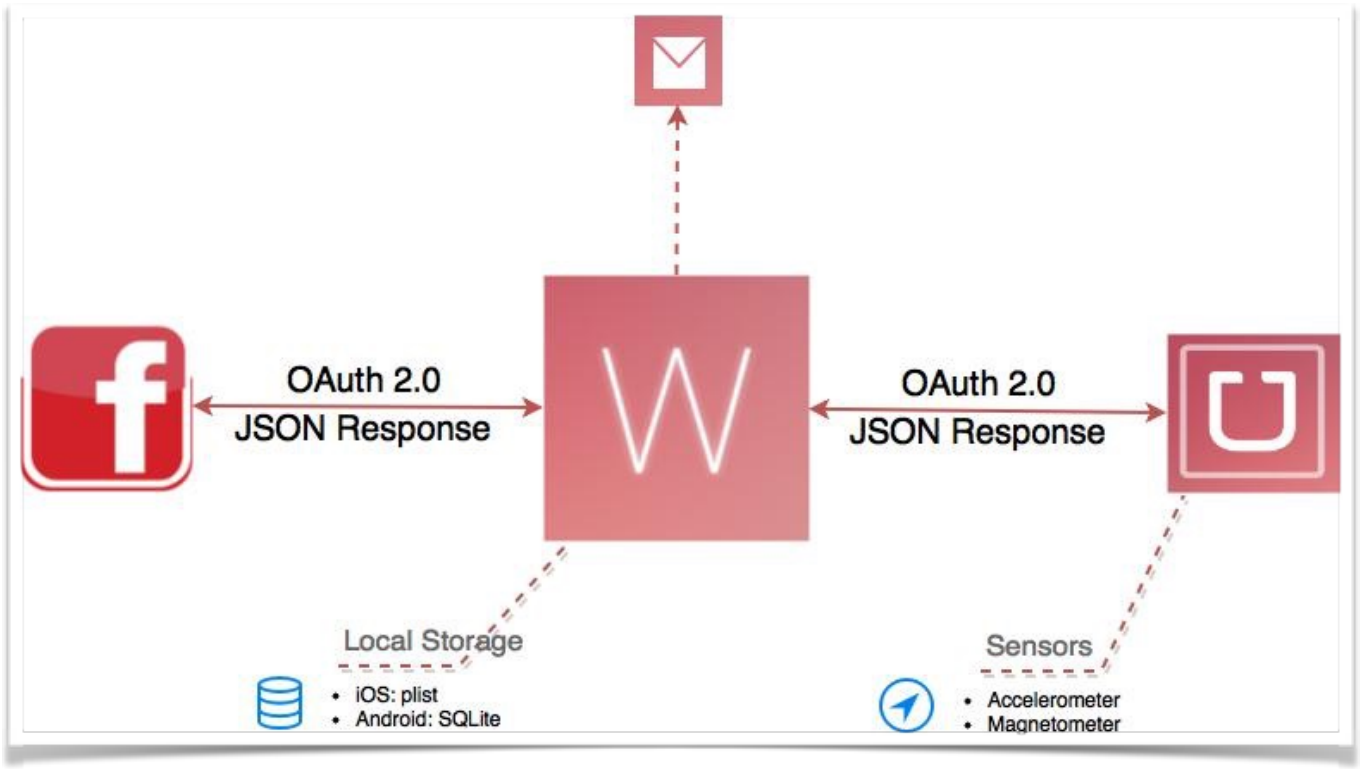

Figure 7. Architecture Design of the WeRide

As it is shown in Figure 7, the WeRide mainly interact with two apis, including Facebook and Uber. They both utilize the OAuth 2.0 authentication and send the JSON response.

Usually the OAuth 2.0 contains two http request, one is to ask for the code thing, the sunken request is used to exchange the access token with the code. JSON is a standard response format nowadays. Once our application receives the JSON response, it should be parsed first and then get the required key-value pair for certain functions.

For data storage, we use plist in iOS and SQLite for Android. Also when considering the sensors, the accelerometer, magnetometer should be used for locating the user, which is necessary for the Uber service.

3. Lessons Learned: iOS vs. Android

3.1 Comparison between iOS & Android

❖ Environment

Xcode is used for the iOS developing; however Android Studio is for Android.

As for emulator part, the emulator runs much slower than the Xcode. Moreover, the model and version of iOS is more standard. For example, the iOS only has several phone model, iPhone 4, 5, 6, etc. However, there are a bunch of phone model and version to choose. So for developing,

❖ Front-end

- Xcode can integrate all the screens into one uniform main.Storyboard however Android doesn't have this thing.
- iOS has more standard component to use directly than Android.
- For the Tab-based application, the default style in iOS and Android are totally different. The former has the tabs at the bottom; however the latter put the tab on the top by default.

❖ Data Handling

The plist is really a convenient way to use in iOS compared with the database. So we utilize the plist for the WeRide iOS version. In Android, the shared preference is also a way to process the small set of data, but the SQLite is a much more standard way and it is really simple to use in Java.

❖ Facebook Configuration

Setting up environment for iOS and Android are totally different.

3.2 Challenges and Workaround Solutions

- **Custom table rowing iOS**

Actually there is a default cell (table row) style in iOS, but it only contains the title, sub-title together with one image view. However, we want to show the special cell, including these name, user email, photo as well as the riding status together, so that we design our own cell style. For Android, there is no default style for the list, so we just design it directly.

4. Future Improvements

1. **Introduce more login method**

   More social network like Twitter, wechat or weibo can be integrated. This way can not only facilitate the login process, but also provide more friend choices. Also, the phone "contacts"is also a friend resources. On the other hand, we may provide the register use case later.

2. **Deep linking the Uber into our application.**

   As for the Uber part, now we only link the Uber icon (at the top of the first tab) to the Uber app or webpage to make the uber request. That means the user should go back our app manually after making a uber request and this way is much easier to implement. Actually, the Uber side provides the way of integrating the uber into the third-party application. So later, we plan to do this deep link part.

3. **Data Handling Way**

   For our iOS first version, we only utilize the plist for convenience since our data record is pretty small; however this way may not enough for bunch of data in the future. So we plan to improve the plist way into SQLite way in the future.

4. **Photo retrieving for the friend**

   For user friendly, the photo for each friend needs to be retrieved from Facebook; however processing image is a little bit more complex than the plain text, so for convenience, we didn't retrieve the photo now. But for later versions, this is one necessary part.