MetroBike - ZFR Instructions

a. GitHub General Setup (We do not use Phabricator)

1. Create an account at GitHub: https://github.com

2. To use GitHub, you need to have Git on your system first. Follow this guide to install and set up git on your machine: https://help.github.com/articles/set-up-git

3. Access MetroBike repository on GitHub through: https://github.com/alaurenz/metrobike

4. If you want to have full access to the code on this repository, please email Adrian **alaurenz{at}uw{dot}edu** with your GitHub username so you will be added as a collaborator. (Jump to the next step if you have been already added.)

5. Once you have been added as a collarborator, you can start interacting with this MetroBike repository. For security reasons, GitHub requires you to set up the interaction by using SSH public key or HTTP authentication. We **recommend** using **SSH public key**. Please follow this guide to set up an SSH key: https://help.github.com/articles/generating-ssh-keys

6. Clone the MetroBike repository to your local machine. Use Terminal to switch to a directory where you want to put the local copy of the repository, and then type the following to command to create the copy. You will make your changes in that repository copy:

```
git clone git@github.com:alaurenz/metrobike.git
```

7. Done! You can try “pull” and “push” files through the following commands if you would like to:

Pull: `git pull origin master`
Push: `git push origin master`

Remember to always “pull” before you want to “push” anything to the remote repository.

b. Bug-Tracking setup on GitHub (Not Phabricator)

Access Issue-Tracking Page

Access GitHub Issue-Tracking page through: https://github.com/alaurenz/metrobike/issues

Create a New Issue

1. Press the green “New Issue” button on the top-right of the issues page.

2. Enter a title for the new issue in “**Title**” box, such as a bug name or a brief task name.

3. Below the Title box, there are **assignment** and **milestone** buttons, use them to assign people to solve this issue and set milestone (You can leave milestone empty if you do not set it).

4. In “**Write**” tab below, you can write the initial comment to this issue. Press “**Preview**” tab to see what the new issue will appear. You also can **drag** Images to this tab.

5. On the right side of the page, you can **label** the issue.

6. Press “**Submit new issue**” button to publish this issue.

Comment on an Existing Issue

1. On Issues page, you can view all the issues. Click one on which you want to make a comment.

2. You can then view comments already made for this issue by other people (or maybe you) before, and write your new comment in the “**Write**” tab at the bottom. You also can **drag** Images to this tab.

3. You can “**Preview**” your comment and then press “**Comment**” button to post your comment.

Close and Reopen an Existing Issue

1. On Issues page, you can view all the issues. Click one that you want to close/reopen.

2. Below the write tab, you see “**Close**” button for a open page or a “**Reopen**” for a closed page. Press it to change the status of this issue.

c. Source Control and Build Process Instructions

Getting and Building Source Code:

1. (This is the same step as Step 6 in Part a, jump to the next step if you have already done so.) Clone the MetroBike repository to your local machine. Use Terminal to switch to a directory where you want to put the local copy of the repository, and then type the following to command to create the copy. You will make your changes in that repository copy:

```
git clone git@github.com:alaurenz/metrobike.git
```

2. MetroBike is an Android app, to develop this app, you need to install Android SDK. (If you have the SDK already, jump to Step 2.) Go to http://developer.android.com/sdk/index.html and click the button “Download the SDK -- ADT Bundle” to download and unzip the SDK including a modified version of Eclipse IDE called “Android Developer Tools (ADT)”.

3. Open ADT. To run MetroBike code, you need an android device running the latest version (currently 4.2.2) of Android OS. You can follow http://developer.android.com/training/basics/firstapp/running-app.html to set up a real device for running/debugging.

NOTE Google Play services is ***not supported*** on emulators. Since we use this api to access Android maps, our application (even the ZFR) will not run on an Android emulator.

4. To import MetroBike project to ADT, you right-click the blank of “**Package Explorer**”, and select “**Import...**”. Next, select “**Android**” and then “**Existing Android Code...**”. Select the MetroBike folder you have cloned from GitHub as “**Root Directory**”. **DO NOT** select “Copy projects into workspace”. Press “**Finish**” to import the project.

5. Before running the app, you **must** add the Google Play library to the project. To do this, follow the steps listed here:
https://github.com/alaurenz/metrobike/blob/master/Adding%20Google%20Play%20Library%20support.txt (These instructions are also part of our repository)
This step is only required the first time you download and build the application.

6. Good job! Now you are able to edit and run MetroBike code in ADT. To run MetroBike on Android devices, select a .java file of your code and run it. If it then prompts you to select a device, select one and then run. Then app will automatically get installed and display in the device. (Since Android SDK does not support different OSs and devices well, if you tried the guide in Step 3 and still cannot run the application, simply copy the .apk file from <project folder>/bin/ and install it on a real android device.)

Source Control using Git:
Pull from GitHub:

```
git pull origin master
```

View commit history logs. -number indicates the most recent number of logs you want Git to display:

```
git log [-<number>]
```

View tracked files:

`git ls-files` (Lists tracked files)
`git ls-files | wc -l` (Shows number of tracked files)

View number of lines in tracked files:

`git ls-files | xargs wc -l` (Lists number of lines for each file)
`git ls-files | xargs cat | wc -l` (Shows total number of lines)

View number of lines modified with a starting time, <time> could be like “1 month ago” or “2 days ago”:

```
git diff --shortstat "@{<time>}"
```

Commit local changes. -a for staging every tracked file in the MetroBike project folder (or you have to `git add` the newly-created and edited file before commit), -m for adding inline comment:

```
git commit -am "<your comment>"
```

Push changes to remote repository on GitHub:

```
git push origin master
```

d. Data Access Instructions

The main server that we need to access is google map server which is a web server and can be accessed by typing an url in a browser.

Here are some url addresses and what they mean:

1. Basic directions example
http://maps.googleapis.com/maps/api/directions/xml?origin=Seattle%2CWA&destination=New%20York%2CNY&sensor=false
By analyzing this url, we can get the meaning of the parameters:

http:// **<http protocol with port 80>**

maps.googleapis.com **<the domain name which will convert into 140.142.15.27 by DNS>**

/maps/api/directions/ **<the attribute to go to specific direction>**

/xml?origin=Seattle%2CWA&destination=New%20York%2CNY&sensor=false
<xml file start with the origin address (Seattle,WA) to (&) destination address (New York,NY) with (&) no sensor> (Note that %20 is space and %2C is comma)

2. Bicycle directions example
http://maps.googleapis.com/maps/api/directions/json?origin=6504%20Latona%20Ave%20NE%2CSeattle%2CWA&destination=3801%20Brooklyn%20Ave%20NE%2CSeattle%2CWA&sensor=false&mode=bicycling

http://maps.googleapis.com/maps/api/directions/ **<same base url as before>**

json?origin=6504%20Latona%20Ave%20NE%2CSeattle%2CWA&destination=3801%20Brooklyn%20Ave%20NE%2CSeattle%2CWA
<origin address: 6504 Latona Ave NE,Seattle,WA to(&) destination 3801 Brooklyn Ave Ne,Seattle,WA>

sensor=false&mode=bicycling **<turn off the sensor and modeling with bicycling>**

3. Transit directions example
http://maps.googleapis.com/maps/api/directions/json?origin=6504%20Latona%20Ave%20NE%2CSeattle%2CWA&destination=3801%20Brooklyn%20Ave%20NE%2CSeattle%2CWA&sensor=false&arrival_time=1368644400&mode=transit
<same as the previous one but with the extra arrival time and modeling with transit>
(Note that the time format is seconds since Jan 1, 1907 in the UTC time zone)

By knowing all the parameters, you can change the source and distention in order to access the data that you need.

Our code to access this data is located in our MainActivity.java file. To make each web request, we:

1. Start a worker thread
2. Connect to google map server using UrlConnection
3. Read the content to the temporary buffer
4. Copy the temporary buffer into a String and repeat step 3 until there is no more content to read
5. Display the String on the screen

We also access data using Android maps, but this access is handled completely by Google Play services. In our ZFR, we simply add the maps fragment to our UI.