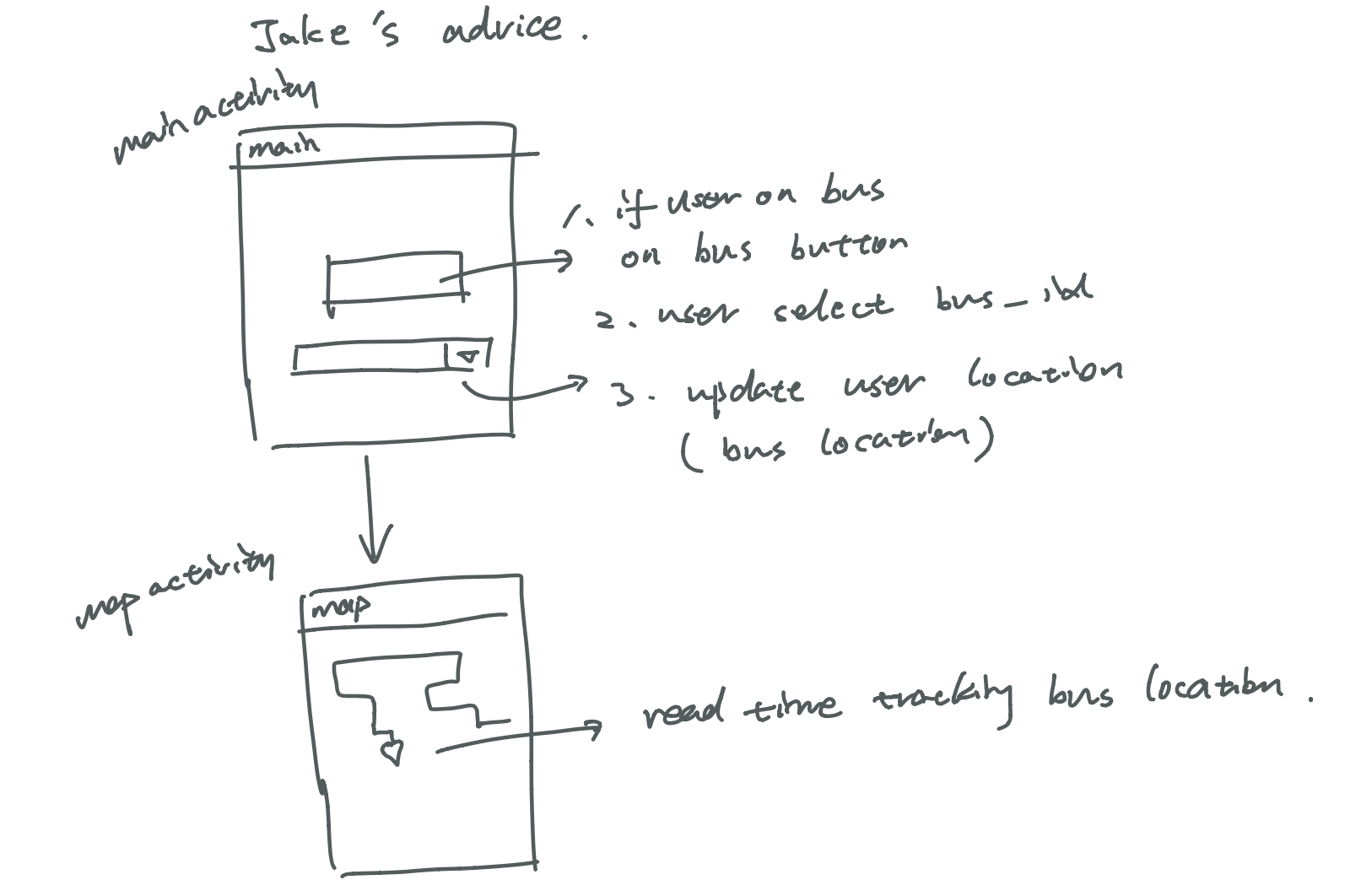
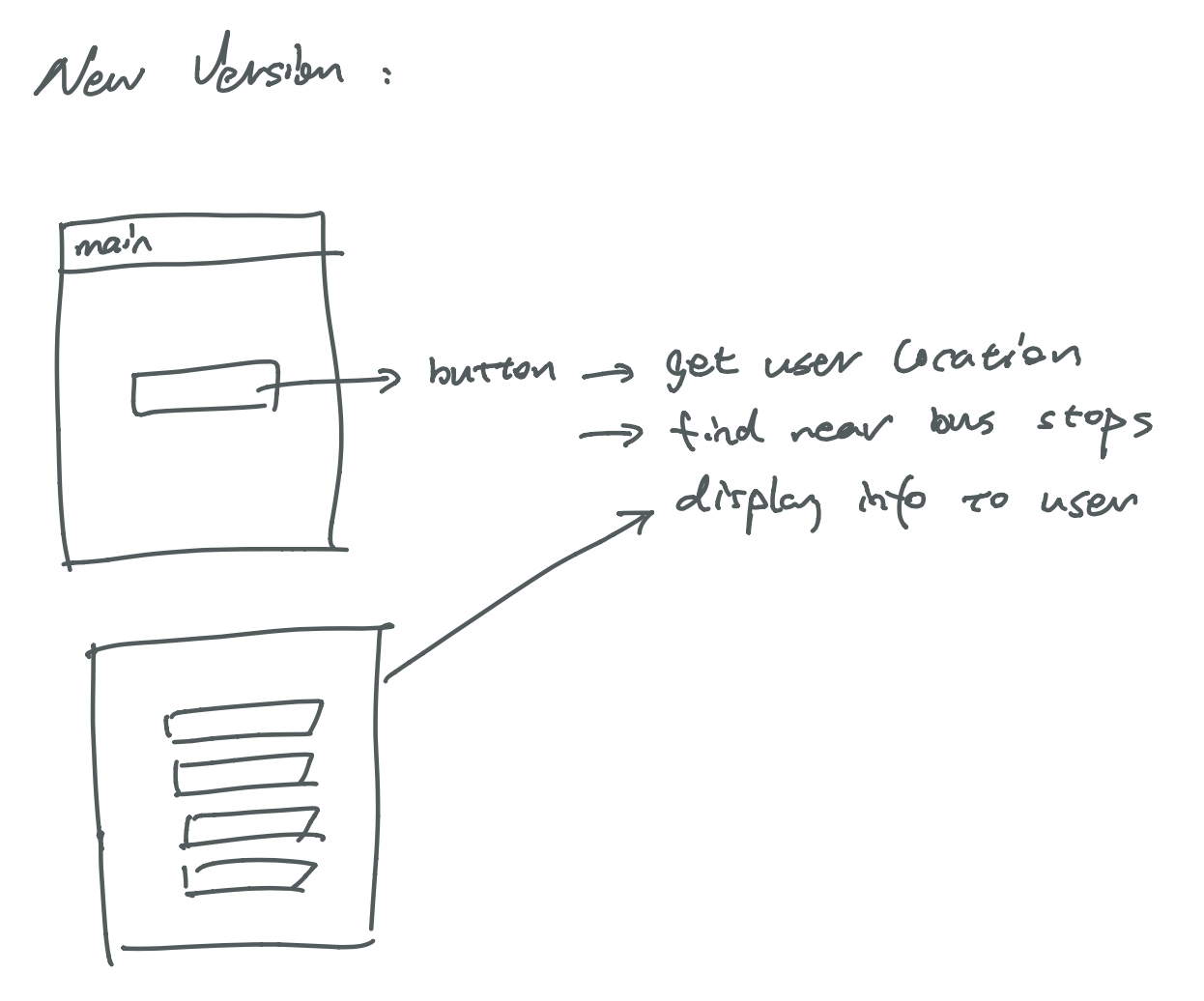
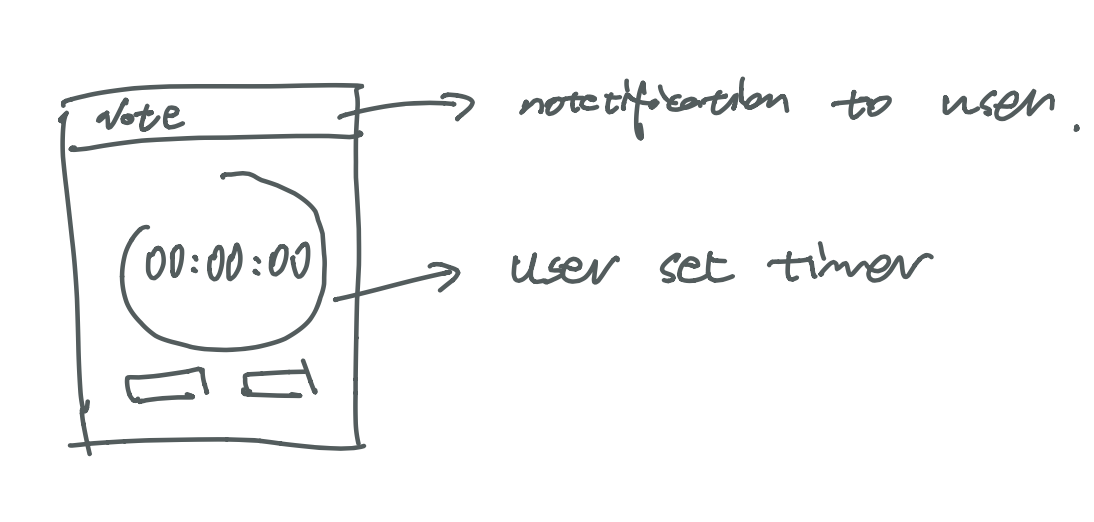
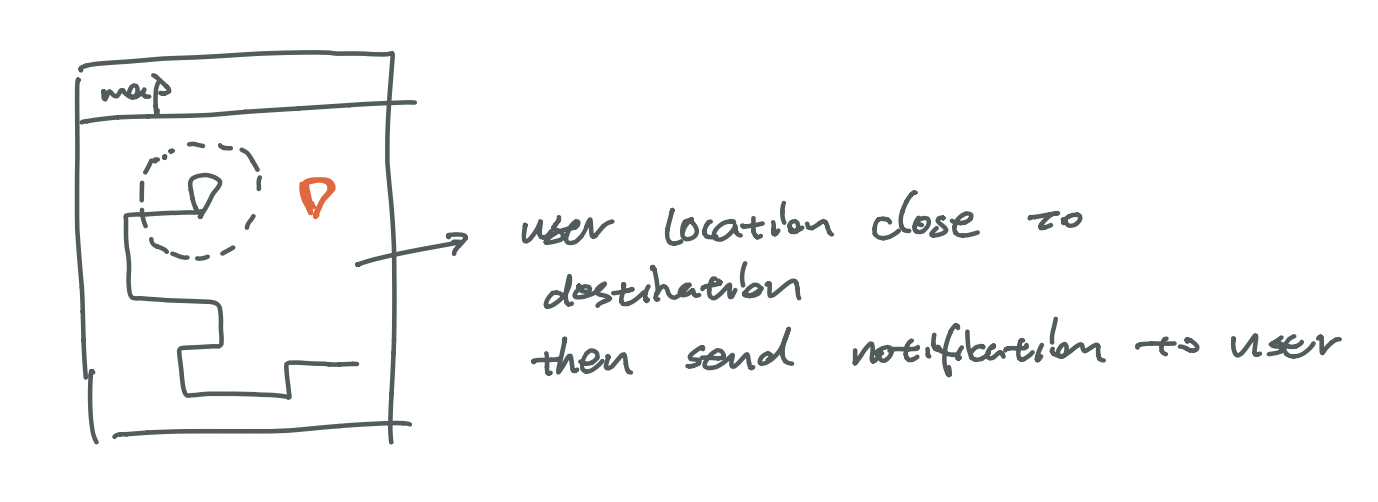
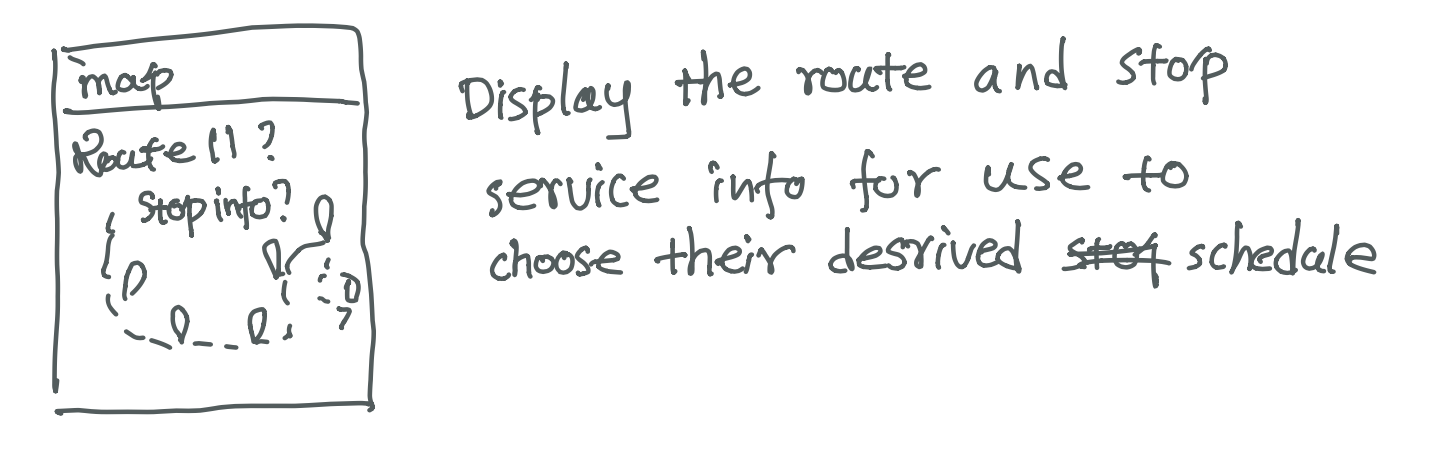


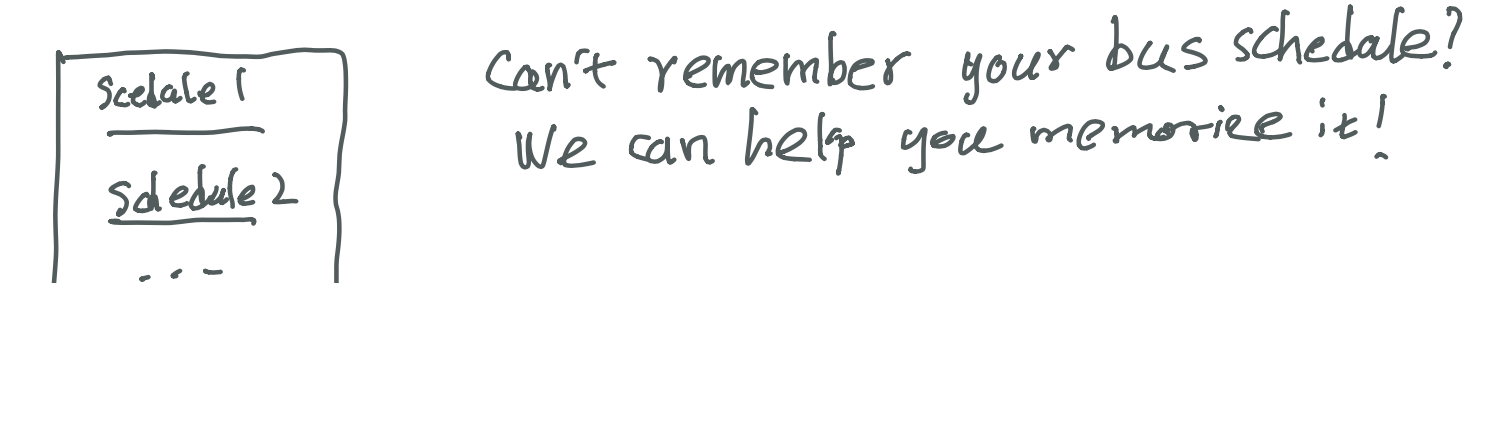
**Design Process:**

1. The original plan was that providing a real time bustrack to user. So that user can see each bus moving on the map. However, there isn’t an API available for us. Based on Jake’s advice, we planned to collect users’ location, and based on users’ locations we can treat them as bus locations which can be presented to other users. However, it will lead to more problems. So we decided to implement the project in another way..
2. We planned download bus and bus stop info from the official website. We also parsed the data and store them into the firedatabase. And the home page, user can press the button to get the near bus stop. All data will be displayed on the screen.



1. Then, user can set the timer for specific bus time.
2. When user is on a bus, that user can set the destination bus stop on the map. When the bus gets close enough to the destination, our app will send a notification to user which notifies the user to be ready.
3. Find the transits for user to choose.
4. Display specific bus route info and 

bus stop info



1. Help you to remebers the

schedules!

**Firebase:**

<https://console.firebase.google.com/u/1/project/busproject-c78fe/database/busproject-c78fe/data>

The Firebase setup is one of the most challenging and time-consuming parts of our project. Since the SBTMD (The Santa Barbara Bus Company) doesn’t provide an API for database access. We have to build our database that synthesizes all the relevant information by some third party’s data, like transit.land and transifeeds. However, the data are in .json or .txt format and is very mussy. For example, in order to retrieve all the available bus information for a specific bus stop. I have to look for the bus stop basic info (Latitude, Longitude, stop\_id) at the stops file. Then, I have to go to stop\_time files to find all the service (a specific bus) information and figure out which bus will arrive at this stop by examining the service\_id and stop\_id mapping. Nonetheless, even I know the service, the specific route names (like route 11, route 28) and when the services will be available (Weekdays, Saturday, Sunday, or special days) from several separate files. It turns out that, in order to ensure the efficiency (speed, data usage) for our program, we have to reconstruct all the information into a more concise structure.

To solve the firebase issue, we firstly put all the available data into the firebase as stops (basic stop information), shape (the geo information of the tripe’s shape), routes (basic route info), tripe\_time (schedule info for each service), trip (info for connecting service\_id, shape\_id, route\_id, and trip\_id). Then, we create lots of helper functions (700 line of codes) to calculate the mapping.

Eventually, our database is generally guaranteed to retrieve any mapping within two queries. Specifically, most service information is mapped into stop class (now you can see that the stop has attributes like Sunday service, Special Sunday service,…), so that now we only need to query one stop by its stop\_id, then we can get all bus service info for that day. The original stop class becomes basic\_stops, which maintain basic stop info to keep the download size smaller (around 5 mb compared to 30 mb to retrieve the whole stop class) The shape becomes an attribute of route. Operation Route is created to keep all possible transit information.

Finally, there are still few issues existing in current database. First, the database needs to be manually updates every three months. Besides, some mappings are incompletely and might produce some error results. We will try to improve this part by creating more advanced code to auto-update this database.

**Map activity:**

As a bus app, map is a virtual part. To provide a user with a pleasant experience, we have implemented most necessary functions and some supplementary functions on map activity.

1. Initial View:

* When user create map, it will automatically display all stops near users based on users’ geolocation (default within 500 meters). If the user is not in the service area of SBMTD, the default location will be UCSB campus.

2. Float Action Button:

* There is a float action button at the bottom left corner of the map activity. After clicking, it will expand to four float action bottoms.
* By clicking the 2nd one, the one with arrow icon, the map will turn into navigation mode. (The icon of this fab will turn into map icon and users can return to navigation mode by clicking this fab again)
* By clicking the 3rd one, the one with cross icon, the map will remove all its markers and polylines, except the markers near the user. At the landscape mode, this fab’s icon will become recycle and user change repick starting marker by tapping it.
* By clicking the 4th one, a tiny notebook will be displayed. The specific functionality is recorded in navigation mode.

3. The map activity’s default mode is map mode.

* Bus Stop Information:
  + Inside this mode, users can check the bus stop information (by clicking on the read markers) and it will pop out available service information at that bus stop (Sorted by route names, like 11, 24x and the maximum display for each route is 15)
* Specific Route Display:
  + By clicking on each service inside the bus info pop out window, the whole route information will be shown on the map, including the shape of the route and all the bus stops for that route in that direction.
* All Routes for a Stop Display:
  + At the end of each pop out window, there is an option to present all routes information that the bus stop has on the map. Each line will be marketed with different color to distinguish, but the marker will be read to indicate they are stop markers.

4. At the navigation mode:

* Navigation:
  + When users click on one red marker, a pop out window will ask users to confirm their choice of their marker as starting point.
    - Users can click “Yes, go now” to start navigation by default starting time (current time calculated by java calendar object)
    - After Users click “Specify a time”, a timepickerdialog will be shown to ask user to choose a time. (If user cancel this action, the current time will be used)
    - If users click “No”, then they can choose another point and the pop out window will be displayed again.
  + After confirming the start point, users can pick another read marker and the first five possible transit information will be displayed as pop out window. The information includes starting time, start route name, ending time, (transit waiting time, transit stops and transfer route name: if users need to switch bus).
  + If there is no available service, users are allowed to restart.
* Transit Route Display:
  + User can check the transit route by clicking display route at the result pop out window. The starting and ending marker will be connected by polyline and the transit stop will be displayed as green markers.
* Geo-notification:
  + Users can pick a stop after the navigation result returns by a pop out window. The app will record that stop and track the distance between users’ current positions and the stop’s position. If the distance is less than 300 meters, a notification will be given to users.
  + The location of user will be updated consistently by every 50 seconds.
* Tiny Notebook for User to memorize their schedule:
  + After the navigation result returns, users can add a bus plan into the schedule list.
  + The schedule list can be viewed by clicking the 4th fab with a calendar icon.
  + The schedule lists will be stored locally, so that unless users delete them, they will be shown when user reopen the application.
  + The schedule can be removed easily by clicking it.
  + The schedule list can also be viewed in map mode.

5. Display Stops by Tapping:

* In any mode, users can find stops near a position by simply tapping that position. All nearby stops (within 500 meters) will be displayed.
* Users can put a maximum two markers at the same time. If the third marker is added, the previous two and the nearby stop markers will be removed.

**Limitations and Possible Improvements:**

Due to the time limit, project scope and the limited number of team members, the project still have some issues and many possible changes can be made in the future development process.

**Mapactivity:**

1. Issue with scroll view
2. For simplicity, the pop out window utilizes the scroll view and we face a problem that if there are too many service information, some of them will be missing. The current solution is to limit the maximum number of service information displayed.
3. In the future, we might utilize the recycle view or find other possible methods to resolve the number limits.
4. Issue with transit finder
   1. The complexity of querying all necessary information (start stop services, destination services, possible transitions, and so on) has gone far beyond our ability (We have nearly 800 line of codes for this, but still not enough). Currently, the transit finder function only allow users to find route that are specified to the two stops they pick and they direction of the stops should be matched. Otherwise, no service can be found. Besides, some potential errors like index out of bounds are not handled or not handled gracefully.
   2. In the future, we will try to optimize the database and the code logic, so that it can use less queries to collect more necessary information and to present a more solid answer to the user.
5. Issue with route display
   1. Because of the unordered and limited available data, the route depicter can’t perfectly show the route info, like which color corresponds which route and there are some issues with polylines. More importantly, the markers are a little large and may lead to mis-click for users.
   2. We will create theme color for each route and create a corresponding legend. Besides, we will alter the icon of markers to make them easier to be picked correctly.
6. Issue with map activity in general
   1. Due to the time limit, the instruction for map activity, especially for the navigation mode and the usage for fabs. Users have to read the manual carefully or try the application by themselves multiple times to fully understand how to work with this application. For example, user can hardly understand how to add schedule to notebook and how to remove them.
   2. In the future, we will implement an instruction process for first time users. Besides, we will have more pop out windows and information bar to help user understand this application. Moreover, we might create a manual fragment for user to find references.

**Main activity:**

The most frequently used feather should be shown as soon as users open applications. Therefore, we decided to make main activity simple and straightforward.

1. Problems, Key technique and further improvements:
   1. Our implementation requires some communication between fragments. So we used basic setter and getter function in order to access messages from different fragments. However, the setter and getter functions are not the proper way for communication. It leads to lack of updating or some null pointer error. This is also the problem we need to solve or make change in the feather.
   2. One of the feathers of our application is timer. However, the simple timer is not that accurate. If user switch back and forth between different fragments the timer goes off dramatically. So, we used the system time. If the timer fragment onStop stage, data will be stored by comparing with the system time. This solves the accuracy problem. On the other hand, we also used the shared preference function to make the time runs in the background.
   3. Notification can be sent if the timer is onfinish. However, when user kill the timer, the application is killed. The timer is still running in the back based on the system time. But the notification function can not be called because the application is killed. Therefore, we realize that using timer and notification in this manner is not a perfect way. In the feather improvement, we should implement the timer and notification from server side. That way the problem mentioned above won't be happened.

**Some future improvements** (BusNotifier):

1. We will create a user system that allows users to share their location, so that we can at least achieve partial real-time. Another mode is to let users rate each bus or bus stop and generate analyse for the late possibility for each bus service or bus stop..
2. We will improve our UI design.
3. We will improve our efficiency of our application.
4. We will implement more functions based on user feedback.
5. Fixing existing bugs and refactor codes to reduce potential bugs.

**Conclusion:**

Our team has really put a lot of effort on this project. Although this project is not perfect now, it already presents many useful functionalities for users to use and we have tried our best to satisfy users' preferences. If anyone has any good suggestions, please email our team!