## CS3281 Project

## Requirement Documentation



## [T0790410]

## Functional requirements

## Based on potential user investigation :

To shape the functional requirements of our app, we did an investigation on what potential users want from our product. The investigation questions and statistic results are attached in the appendix. From result of question 2 we see there is a large amount of people who usually travel with bus and MRT, providing us the faith that our users can be a large group with crowd data that can be taken good used of in our app. From Q4 and Q5 respectively we can conclude that information such as arriving time for next bus/MRT and crowdedness of the bus/MRT is greatly cared by users, which gives strong supports for our two major functions (namely providing estimated arriving time for next bus/MRT, and providing degree of crowdedness on the bus/MRT).

* The user shall be able to register a new account and log in to the system.
* The user shall be able to search the arrival time of trains for a particular station.
* The system should be able to calculate and display the shortest path, and the estimated travel time, from the starting point to the destination point, taking into account the bus/MRT timing and traffic conditions reported by the users at real time.
* The system shall provide voice alert to the user if the train he is trying to catch has left the station.
* The system should be able to collect data on traffic conditions posted by users, and display them on the map of the app. The system should also allow users to rate the accuracy of this information at this point of time, and remove outdated/inaccurate information automatically.
* The user shall be able to know whether a particular bus/MRT or bus stop/MRT station is crowded. The user will be able to look for this information (in form of text or images) in the app.
* The user shall be able to request other users to provide information for them about the crowd information on the relevant bus/MRT or bus stop/MRT station.
* The user shall be able to rate whether a source of information provided by other users is useful or not.
* The user shall be able to provide information about the crowdedness of a place by himself.
* The user shall be able to provide information about the crowdedness of a place upon request from other users.
* The system shall have a route tracking function. In this function, each bus stop/MRT station is represented by a circle. At the beginning of his travel, all circles in his route are colored red. Once the bus/MRT passes a bus stop/MRT station, the circle representing the bus stop/MRT station will be colored green. Therefore, user can know when to get off the bus/MRT.
* The system shall allow addition of new bus/MRT information, including the routes of the bus/MRT, name and number of the stops/stations through its travel path, the time interval between two consecutive buses/MRTs, the time for earliest/ latest bus/MRT for user who has a unique identity of administrator.
* The system shall allow update of existed data of bus/MRT information, including the routes of the bus/MRT, name and number of the stops/stations through its travel path, the time interval between two consecutive buses/MRTs, the time for earliest/ latest bus/MRT for user who has a unique identity of administrator.
* The system shall allow user who has a unique identity of administrator to browse all the comments/posts in the database from ordinary users, and allow him to delete improper or obsolete comments/posts.
* The system shall allow administrators to check through all accounts of ordinary users, and shall allow administrators to delete malicious accounts or zombie users.

## Non-functional requirements

* Accessibility

The system should be accessible to as many people as possible. People with weak vision should be accessible to the system with aid of voice guides and notifications.

* Availability

The system should be available for any time when the bus/MRT system is working.

* Capacity (current and forecast)

Currently, the system should be capable to handle traffic information from all buses and MRTs in Singapore. From the view of long-term development, the following versions might also include information from other cities.

* Compliance

The use of data and dissemination of traffic information in our system should follow local policies and legislations in Singapore strictly.

* Extensibility

## The system should have a good support for extensions. Improvement of existed functions and addition of new features shall be supported. In the next major version upgrade the system should also include customizations in terms of UI and functions.

* Maintainability

## The maintenance of the system should be simple and straightforward. The database should have elegant structures to increase the maintainability of the system. The system should also have limitations on user’s posts to make maintenance easier.

* Performance

## The system should have a reasonably quick response to user’s operations. The system should have relatively low occupation of device’s CPU and memory.

## Additionally, the system should also have an effective method to retrieve online data to shorten the waiting time for users to see the results of their requests.

* Platform compatibility

This system should be compatible on any of Android Platform 4.x versions.

* Pricing

This system should be a free application for any Android 4.x devices.

* Reliability

The system should regularly keep information updated and reliable.

* Security

## The system should have enough protections for user accounts. The system should defend the database from malicious SQL injections and other types of attacks.

* Stability

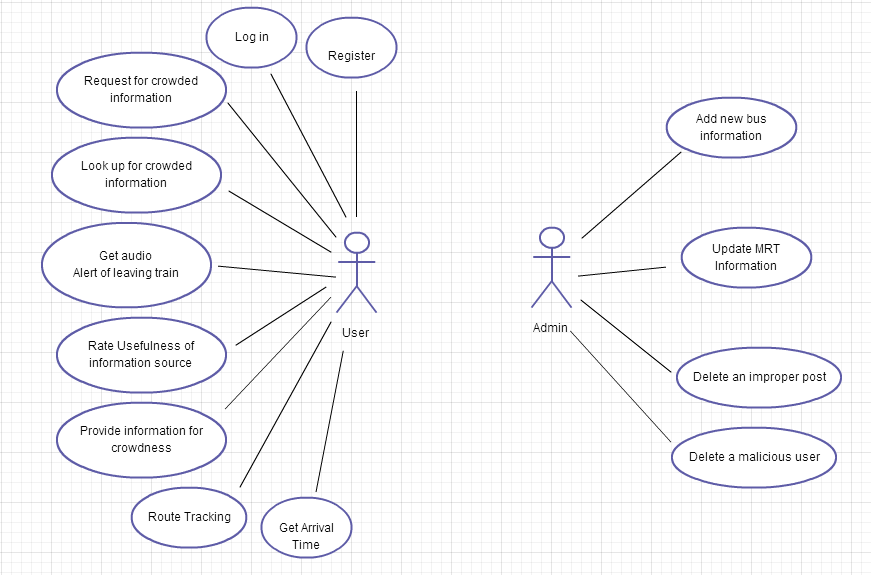
## The system should keep its stability on Android devices. It should be working without frequent crashes or break-down.

* Usability by target user community

## The system should have a user-friendly UI and be easily used by our target users.

## The operations and the presentation of information should be simple and neat.

## Use Case Diagram



## Main Scenarios

**Use case: UC01 - Register**

Priority: medium

Actor: user

1. User types a user name of his choice

2. User types a password

3. Use retypes the password

Submit

4. System validate the existence of the username

5. System checks whether the two passwords are the same

6. System registers the new user

**Use case: UC02 - Log in**

Priority: medium

Actor: user and administrator

1. User types his username

2. User types his password

Submit

3. System validate the user

4. User logs in to the system

Precondition: the user is registered.

**Use case: UC03 - Get Arrival Time**

Priority: medium

Actor: user

1. User chooses “Get Arrival Time” from the menu

2. User selects the train he wants to take and the direction of the train

3. User enters his departure time

Submit

4. System checks whether the departure time is valid

5. System returns the estimated arrival time of the train

Preconditions: User has logged in

**Use case: UC04 - Shortest Route search**

Priority: low

1. User chooses “Map” from the menu

2. User types in the beginning position and the ending position

2a. If the user wants to use his/her current position, the user can do so by typing “here” as the beginning position

3. User enters his departure time

4. System returns the shortest route the user have to take, the different methods of transport the user can choose (driving, walking, public transport), and the estimated time of journey.

Preconditions: User is logged in

**Use case: UC05 - Get Audio Alert of leaving train**

Priority: medium  
Actor: user

1. System sends out voice alert “The train to XXX has just left”, the alert is also popped in the screen

2. User dismisses the message

3. System shows the arrival time of the next train of the same direction

Preconditions: GPS detects that user is heading to an MRT station and a train has left the station

**Use case: UC06 - Look up for crowded information**

Priority: high

Actor: User  
Brief description: This use case describes the procedures user will go through in order to look up for crowded information in the system.

Preconditions: User is logged in.

Basic flow:

1. User chooses “Crowded Info” button from the menu
2. System shows new screen that allows user to select/enter information
3. User selects/enters the type of information (bus stop, MRT station, bus) and the corresponding number of type (bus stop number, MRT station name, bus number)
4. System shows hints on the fly while user enters number
5. User chooses “Submit”
6. System shows user information user is looking for  
   If the information is not available, system notifies user and prompts user whether he/she wants to request for that information
7. 7.1. User views the information he/she wants  
   7.2. User chooses to request for that information

Post conditions: User gets crowded information they want or notified that the information he/she is looking for is not available

**Use case: UC07 - Request for crowded information**

Priority: high

Actor: User  
Brief description: This use case describes the procedures user will go through in order to request for crowded information from other users through the system.

Preconditions: User is logged in.

Basic flow:

1. User chooses “Request” button from the menu
2. System shows new screen that allows user to select/enter information
3. User selects/enters the type of information (bus stop, MRT station, bus) and the corresponding number of type (bus stop number, MRT station name, bus number)
4. System shows hints on the fly while user enters number
5. User chooses “Submit”
6. System notifies user that the request has been sent  
   When the response to that request is available, system notifies user
7. User view the information he/she requested

Post conditions: User’s request for particular crowded information is sent successfully and user can view the response when it’s available.

**Use case: UC08 - Rate Usefulness of Information Source**

Priority: medium  
Actor: User

1.  User clicks on the information provided by other users.

2.   Another window showing the degree of usefulness of this information

3.   User clicks the usefulness degree (range between 1 and 5)

4.   Another window asking user to confirm this rating

5. User clicks confirm button.

**Use case: UC09 - Provide Information for Crowdedness**

Priority: medium  
Actor: User

1.  User clicks on the feedback button

2.   Another window asking user to indicate the place

3. User types in the place information

4. Another windows asking user to indicate the rate for crowd

5.   User clicks the crowd degree (range between 1 and 5)

6. Another windows asking user to confirm this feedback

7. User clicks confirm button

**Use case: UC10 - Route tracking**

Priority: medium

Actor: user

1. Initially, all stops in the route are colored red

2. Every time the bus passes one stop, the stop is colored green

3. User alight at the destination

Precondition: the user is commuting

**Use case: UC11 - Admin adds new bus information**

Priority: low

Actor: Admin

1.   System prompts Administrator for new bus information:

·         Bus number

·         Starting stop

·         Destination stop

·         Name and number of bus stops along the way

·         Time for the departure of the first bus

·         Time for the departure of the last bus

·         Time interval between two consecutive buses

2.  Administrator enters the required information

3.   Administrator submit the entered information by clicking the confirm button

4.   System validate information in terms of formats and correctness

5.   System store new information on server and new bus information successfully added

**Use case: UC12 - Admin updates MRT information**

Priority: low  
Actor: Admin

1.  Administrator selects the MRT line that he wants to update

2.   System provides a list of the previous data for this specific MRT line:

·         MRT line name

·         Starting MRT station

·         Destination MRT station

·         Name and number of stations along the way

·         Time for the departure of the first MRT

·         Time for the departure of the last MRT

·         Time interval between two consecutive MRTs

3.   Administrator selects information he wants to update

4.   System display modification interface for Administrator to update the information

5.   Administrator updates the information

6.   Administrator submit the updated information by clicking the confirm button

7.   System validate information in terms of formats and correctness

8.   System store this new information on server to replace the old one, and MRT information successfully updated.

**Use case: UC13 - Admin deletes an improper post**

Priority: low  
Actor: Admin

1.  Administrator requests to delete an improper post

2.  System deletes the improper post from the database

3.  System notifies Administrator that the post has been deleted

4.  Improper post successfully deleted

**Use case: UC14 - Admin deletes a malicious user**

Priority: low  
Actor: Admin

1.  Administrator requests to delete a malicious user

2.   System deletes the malicious user’s account from database

3.   System notifies Administrator that the malicious user has been deleted

4.   Malicious user successfully deleted

## Appendix

**User requirement survey result**

