CuseBus Developer and System Integration Guide

Below are the specifications for querying and modifying the CuseBus database. System integrators should only perform inserts, updates or deletes through stored procedures. Queries should only be performed through views listed in the 'Views' section. CuseBus stores schedules from buses and trolleys provided by Syracuse University and Syracuse Centro Bus and users information. Ensure that all integrators follow an equally restrictive privacy policy, and that secure coding techniques are practiced. We should consider all of our user information as confidential.

- ***0_DATABASE_SCHEMA.sql*** Creates the CuseBus database, the tables and constraints. Note that running this script will drop all of the above mentioned items, effectively removing all data from the system in the process. The 'Down' portion of this script is intended for sandbox/test environments only.
- ***1_DATABASE_INSERT.sql*** Create sample data into all tables and will output a small amount of data from all tables. This script is intended for sandbox/test environments only.
- ***2_DATABASE_STORED_PROCEDURES..sql*** Create stored_procedures and test its execution. This script is intended for sandbox/test environments only.
- ***3_DATABASE_VIEWS.sql*** Create views and test its execution. This script is intended for sandbox/test environments only.

1. Views

The following represents how a developer can query the system for data or reports. These will be a combination of views (denoted as v*).

User Information

- ***vGetUserInfo***
  Returns the users information, it does not include the password column

- ***vGetUserNotifications***
  Returns the notifications the user saved in the app

- ***vGetUserFeedbacks***
  Returns the feedback the users gave in the app about the bus crowdedness and arrivals accuracy

- ***vGetUserFavoriteBuses***
  Returns the buses the user favorited in the app

- ***vGetUserFavoriteBusStops***
  Returns the bus stops the user favorited in the app

Bus Information

vGetBuses
Returns all the buses information

2. Stored Procedures

The following represents how a developer can query the system for data or reports. These will be a combination of stored procedures (denoted as usp*).

- ***uspGetBusSchedules***

```
@route_toschool    BIT = NULL,
@bus_id            INT = NULL,
@stop_id           INT = NULL,
@route_weekday     BIT = NULL,
@time              TIME = NULL
```

  Returns the bus schedules information, if none of the parameters are set it will return information from all buses

- ***uspGetBusStops***

```
@route_toschool BIT = NULL,
@bus_id         INT = NULL
```

  Returns the bus stops information, if none of the parameters are set it will return information from all bus stops

3. External Program Logic

The following represents how a developer can query the system for data or reports. These will be a combination of views (denoted as v*).

- ***uspAddUser***

```
@user_email        VARCHAR(30),
@user_password     VARCHAR(10),
@user_firstname    VARCHAR(25),
@user_lastname     VARCHAR(25),
@user_phone        VARCHAR(15),
@user_address      VARCHAR(50),
@user_home_lon     DECIMAL(10,7),
@user_home_lat     DECIMAL(10,7),
@user_id           INT OUTPUT -- Output parameter to return the new user_id
```

  Creates a new user. It will return the *user_id* of the created user.

- ***uspAddUserFavorite***

```
@user_id                INT,
@user_fav_bus_stop_id   INT = NULL,
@user_fav_bus_id        INT = NULL
```

  Adds a bus or bus stop in the user favorite list

- ***uspAddUserNotification***

```
@user_id           INT,
@not_route_id      INT,
@not_min_before    INT
```

Adds a new notification setting for a specific route that the user wants to

- ***uspAddUserFeedback***

  ```
   @user_id              INT,
  @feed_route_id         INT,
  @feed_bus_ontime       BIT,
  @feed_bus_crowded      BIT
  ```

  Creates a new feedback by the user about the crowdedness and accuracy of bus arrivals