**Lab II Section 3**

* + 1. **Database** (**Odean Maye)**

The database must contain tables that store values that correspond to each particular component of the system.

* + - 1. Time must be stored in the database in the format HHMMSS.
      2. Date must be stored in the database in the format as YYYYMMDD.
      3. The database must contain a table named trains that stores the following information in fields :
         1. The unique identifying number stored as a SMALLINT.
         2. The active status stored as a TINYINT.
         3. A reference to the schedule id.
      4. The database must contain a table named users that stores the following information in fields:
         1. The unique identifying number stored as a MEDIUMINT.
         2. The given name as a TINYTEXT.
         3. The username stored as a TINYTEXT.
         4. The password stored as a TINYTEXT.
         5. A reference to an admin level defined in 3.1.1.10.i .
      5. The database must contain a table named stops with required fields as defined by the Google Transit Feed Specifications at [https://developers.google.com/transit/gtfs/reference#stops\_fields](#stops_fields)
         1. The unique identifying number stored as a SMALLINT.
         2. The name stored as a TINYTEXT.
         3. The latitude stored as a FLOAT.
         4. The longitude stored as a FLOAT.
      6. The database must contain a table named adverts that stores the following information in fields:
         1. The unique identifying number stored as a SMALLINT.
         2. A reference to a stop id defined in 3.1.1.5.i .
         3. The advert name stored as a TINYTEXT.
         4. A reference to a user id defined in 3.1.1.4.i .
         5. The category stored as a TINYTEXT.
         6. The description stored as a MEDIUMTEXT.
         7. The start date stored as a DATE.
         8. The end date stored as a DATE.
         9. The start time stored as a TIME.
         10. The end time stored as a TIME.
         11. The image url stored as a TEXT.
         12. The TIMESTAMP.
      7. The database must contain a table named occupancy that stores the following information in fields.
         1. The unique identifying number stored as a BIGINT.
         2. A reference to a stop id defined in 3.1.1.5.i .
         3. The number of embarks stored as a SMALLINT.
         4. The number of disembarks stored as a SMALLINT.
         5. A reference to a train id defined in 3.1.1.3.i .
         6. The TIMESTAMP.
      8. The database must contain a table named alerts that stores the following information in fields.
         1. The unique identifying number stored as a SMALLINT
         2. The title stored as a TINYTEXT.
         3. The description stored as a MEDIUMTEXT.
         4. The course of action stored as a MEDIUMTEXT.
         5. The severity stored as an INTEGER.
         6. The latitude stored as a FLOAT.
         7. The longitude stored as a FLOAT.
         8. The impact radius stored as a FLOAT.
         9. The start time stored as a TIMESTAMP.
         10. The in progress status stored as an INTEGER.
         11. The stop time stored as a TIMESTAMP.
      9. The database must contain a table named schedule that stores the following information in fields:
         1. A unique identifying number stored as a SMALLINT.
         2. The day of the week stored as a TINYTEXT.
         3. The schedule start time stored as a TIME.
         4. The schedule end time stored as a TIME.
         5. The time between trains stored as a FLOAT.
      10. The database must contain a table named levels that stores the following information in fields:
          1. The access level stored as an INTEGER.
          2. The access title stored as a MEDIUMTEXT.
      11. The database must contain a table named gps that stores the following information in fields:
          1. The unique identifying number stored as a BIGINT.
          2. A reference to a train id defined in 3.1.1.3.i .
          3. The train latitude stored as a FLOAT.
          4. The train longitude stored as a FLOAT.
          5. The TIMESTAMP.

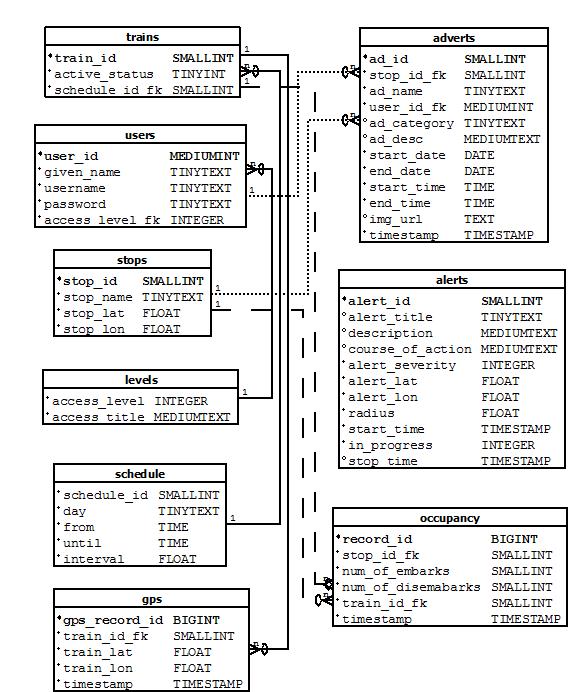


Figure . Database Schema

* + 1. **Decision Engine** (**Nathan Lutz**)

Three algorithms that interface with the Web Application Engine to provide trending and reports on ridership, delay estimates, and ontime performance.

1. **Ridership Trend Analysis**:

Provide an interface for the Web Application Engine to request a ridership report on the past.

Provide the ability to identify a date range to include:

Date(YYYY-MM-DD)

Time range (HH:MM:SS)

Provide the ability to identify a stop ID as an integer

Provide the ability to determine whether the date specified is a future or past date.

Provide the ability to connect to the Current ITS mysql database.

Provide the ability to follow logic beginning in requirement 3.1.2.v.a if the date is in the past.

Values specified in 3.1.2.1.i must be used to query the “Occupancy” table for number of departures and arrivals. (3.1.1.7)

Provide output to Ridership Trend Report function in the form of non-negative integers. (3.1.4.6)

Provide the ability to follow logic beginning in 3.1.2.1.vi.a if the date is in the future.

Specified must be used to query the “Occupancy” table for number of departures and arrivals from past dates during the specified time range. (3.1.1.7)

Provide the ability to average the number of departures and arrivals that occurred during the time range specified on each date in the past up to 15 days.

Provide the ability to query the “adverts” table for events that occur during the date & time range specified.

Provide the ability to follow logic beginning in requirement 3.1.2.1.vi.f if an event is found.

Provide the ability to store the following values for comparison:

Name

Stop ID

Category

Start date

End Date

Start Time

End Time

Provide the ability to query the “adverts” table for events that have occurred using the values stored in 3.1.2.1.vi.f.i-vii

Provide the ability to follow logic beginning in requirement 3.1.2.1.vi.i if an event is found.

Provide the ability to determine the variance between the set of averages found in 3.1.2.1.vi.c and the embark/disembark values during the time of the event found in 3.1.2.1.vi.g

Provide the ability to apply the variance found in 3.1.2.1.vi.i to the set of averages in 3.1.2.1.vi.c.

Provide output to Ridership Trend Report function in the form of non-negative integers. (3.1.4.6)

* + - 1. **Delay Impact Calculator**

Provide an interface for the Web Application Engine to request a Delay Impact report. Provide the ability to query the Current ITS database for the most recent simulated GPS location value of active trains.

Provide the ability to identify a GPS coordinate to include:

* Precede South latitudes and West longitudes with a minus sign.
* Latitudes range from -90 to 90.
* Longitudes range from -180 to 180.

Provide the ability to identify a date as specified in 3.1.2.1.i.a

Provide the ability to query the “trains” table for active trains during the date specified in 3.1.2.2.ii.

Provide the ability to query the Current ITS database “GPS” table for past simulated arrival times at the station during the date. (3.1.1.11)

Provide the ability to query the “schedule” table. (3.1.1.9)

Provide the ability to compare the values specified in 3.1.2.2.iv and 3.1.2.2.v and store the average variance.

Provide the ability to query the “alerts” table for any active alerts and their severity level. (3.1.1.8)

Specified must be used in its calculation of delay using the following categories and associated Delay intervals.

* 1 – Local Warning – No Delay
* 2 – Local Problem – 10 Min Delay
* 3 – System Warning – 20 Min Delay
* 4 – System Problem – 30 Min Delay
* 5 – System Failure – 50 Min Delay
* 6 – System Shutdown - Interpreted as full stop of trains.

Provide the ability to apply the delay interval to the average variance stored in 3.1.2.2.vi

Provide the ability to compare the expected value of time-to-arrival between the calculated variance and current GPS position to the HRT schedule.

Provide output to Train Data Report module in the form of a time value. (3.1.4.7)

* + - 1. **Ontime Performance Reporting**

Provide an interface for the Web Application Engine to request a Delay Impact report.

Provide the ability to identify a date range as specified in 3.1.2.1.i

Provide the ability to identify a stop ID as specified in 3.1.2.1.ii

Specified must be used to query the Current ITS database “GPS” and “STOPS” table for past simulated arrival times at the station. (3.1.1.11) (3.1.1.5)

Provide the ability to query the “schedule” table. (3.1.1.9)

Provide the ability to compare the values specified in 3.1.2.3.iii and 3.1.2.3.iv and return the variance.

Provide output to Train Data Report module the average variance, in the form of a time value. (3.1.4.7)

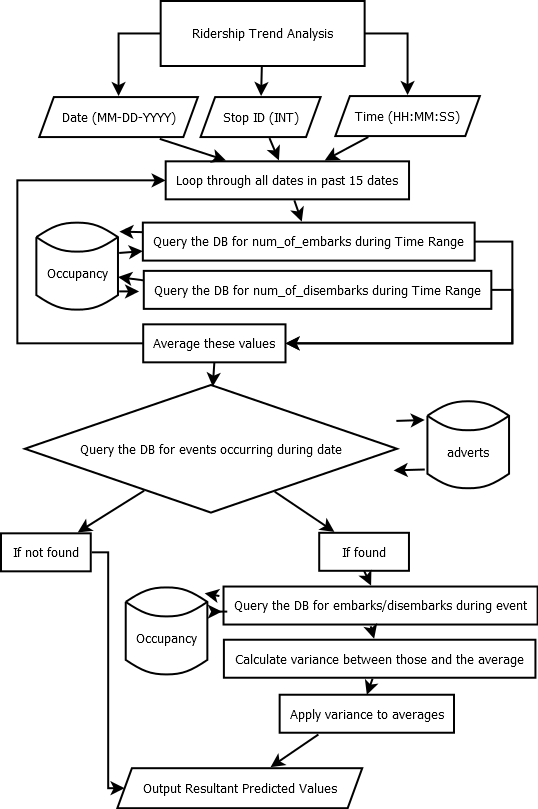


Figure 2. Ridership Trend Analysis Algorithm

**3.1.3 Test Harness (Akeem Edwards)**

The Test Harness will be used to demonstrate the current ITS prototype. This will be a standalone application that will maintain communication with the web application engine. The following functional requirements must be met:

1. **GPS Data Control Module**

Current ITS prototype will use a software module that will take a range of GPS coordinates then creates a virtual route. The GPS Control module then retrieves virtual stops from the database and determines which virtual stops correlate each route created. The GPS control Module will also assign GPS coordinates to each active train along the virtual route. The following functional requirements must be met:

i. Provide the ability to access the virtual stops in the database to determine what GPS coordinates to which stops are assigned (3.1.1.4).

ii. Provide support GPS coordinate parameters:

iii. Input parameters for single GPS coordinate are floating point values for latitude and longitude (3.1.2.2.1.1.1).

iv. Input parameter for GPS coordinate set in an array structure to represent a virtual route.

v. Provide the ability to assign a GPS coordinate to each virtual train active:

a. Each coordinate must translate to the correct virtual route.

b. Each coordinate must be updated in half a minute intervals.

1. **Ridership Data Control Module**

In the Current ITS prototype the Ridership Data Control Module will generate virtual riders to for the prototype demonstration. The Ridership Data Control will assign virtual ridership to each active train. This Module will generate virtual rider numbers at each stop representing departures and arrivals, and compare these numbers to assign the current amount of riders on each train. The following function requirements must be met:

1. Provide the ability to generate virtual riders at each stop.
2. Must utilize mathematical probability distributions based on past ridership data to define:

The amount of arrivals on each train.

The amount of departures on each train

Must utilize probability distributions to estimate amount of virtual riders generated on days with events.

1. **Train control Module**

The Train Control Module will be responsible for simulating virtual trains moving along virtual routes. This Module will assign each active virtual train a GPS coordinate related to the virtual route. The Train Control Module will be interfacing with the Ridership Data Control Module (3.1.1.2). The following functional requirements must be met:

1. Each virtual train must provide the location coordinate abilities defined:
2. Return current GPS coordinate.
3. Return a GPS coordinate not associated with the virtual route to simulate sensor failure.
4. Provide the option to not return the current GPS coordinate assigned to simulate train outage.
5. Each virtual train must provide the ability to return amount of riders on board (3.1.1.1.4)
6. **Business ad control**

The Business ad control will have access with to the following fields:

* + 1. An advertisements End Date (3.1.1.3.1c)
    2. An advertisement Start Date(3.1.1.3.1b)
    3. Advertisements assigned to each stop.
    4. Advertisement start time
    5. Advertisement end time

1. **GUI**

Current ITS prototype will be utilizing a set of graphical user interfaces for controlling different software modules apart of the test harness. The GUI will be interfacing with all the modules in the test harness. The following functional requirements must be met:

Provide a GUI with the ability to view different virtual train properties as defined:

1. The current amount of riders aboard (3.1.3.3.2)
2. The current location assigned(3.1.3.3.1)
   * 1. Provide a GUI with the ability to change each virtual train properties as defined:
        + 1. The current amount of riders onboard
          2. The location assigned (3.1.3.3.1)
          3. Provide the option to simulate train failure (3.1.3.3.1.3)
          4. Provide the option to simulate sensor failure (3.1.3.3.1.2)
     2. Provide a GUI with the ability to change ridership data at each stop
     3. Provide a GUI with the ability to edit each advertisement properties defined
3. Advertisements to each stop (3.1.3.4.2)
4. The End Date of an advertisement (3.1.3.4.1)
5. Advertisement Start Time
6. Advertisement End Time

**3.1.4 Web Application Engine**

1. **Alert Module (Chris Coykendall)**

The Alert Module will provide the ability view and/or manage alerts on the Current ITS website user interfaces.

1. Provide the ability to SQL query the Alerts database table to obtain a list of ongoing outages and exceptions to normal vehicle operations and display them (during the initial page load, select only ongoing alert records which have the ongoing flag set in the table.) (Requirement 3.1.4.11)
2. Provide the ability to SQL query the Alerts database table to create or modify alerts for outages and exceptions to normal vehicle operations. (Requirement 3.1.1.4, 3.1.4.11)
3. Provide the ability for a user viewing HRT GUI to complete the following fields:
   1. Title
   2. Begin
   3. End (Estimated if future)
   4. Description
   5. Course of Action
   6. Submit (Triggers Requirement 3.1.4.1.4)
4. Provide ability to generate an Alert via SQL query to the Alerts database table with an initialized end time stamp of null and ongoing the ongoing flag set to true. (Requirement 3.1.4.11)
5. Provide an additional Close button next to events that are ongoing in the HRT GUI view, which will set the end timestamp for the particular Alert to the current date/time via SQL query in the Alerts database table. (Requirement 3.1.4.11)
6. Provide the ability to view any fetched Alerts in any web interface (HRT, Business or Rider.)
7. **Feedback Module (Chris Coykendall)**

The Feedback Module will provide a mechanism to accept user feedback regarding Current ITS from the website user interface, and email the obtained feedback to the development team.

1. Provide ability for a user to complete the following fields:
   1. Name (REQUIRED)
   2. E-mail
   3. Subject (REQUIRED)
   4. Message
   5. Submit Button (Triggers Requirement 3.1.4.2.2)
   6. Reset Button (Triggers Requirement 3.1.4.2.4)
2. Provide ability to autonomously deliver an email to 411red@cs.odu.edu with the field contents of the input form, so long as required fields have characters. (Requirement 3.1.4.2.1)
3. Provide the ability to display a confirmation message upon completion of Requirement 3.1.4.2.2.
4. Provide the ability to reset the fields of the form by clearing any previously entered input. (Requirement 3.1.4.2.1)
5. **System Overview Module (Chris Coykendall)**

The System Overview Module provides functionality for viewing the overall status of HRT operations using the Current ITS web interface.

1. Provide the ability to SQL query the DE Interface to obtain the stop positions, vehicle positions in latitude/longitude, and coarse (vacant/full) ridership for vehicles in operation. (Requirement 3.1.4.11)
2. Provide overlay graphical markers for stop positions and vehicle positions onto a Google Maps satellite view of The Tide rail system, using their GPS coordinates to position the markers. (Requirement 3.1.4.3.1)
3. **Google Maps Web Form (Chris Coykendall)**

The Google Maps Web Form is an embedded website user interface module which uses the Google API to generate a route of travel given a location and destination.

1. Provide the ability to display a form from the Google Maps API which can accept a current location and destination as text input from the user.
2. Provide the ability to direct a user to the Google Maps routes website upon submission of Requirement 3.1.4.4.1 in a new browser window.
3. **Calendar Event Module (Chris Coykendall)**

The Calendar Event Module is part of the Current ITS website user interface which will enable users to view or manage local events near The Tide.

1. Provide the ability to SQL query the Adverts database table to obtain a list of ongoing Events and display them (ongoing is defined as events with end timestamps of null and non-HRT user ID.) (Requirement 3.1.4.11)
2. Provide an input form which accepts the following fields in the Business GUI:
   1. Title
   2. Start Date
   3. End Date
   4. Message
   5. Submit Button (Triggers Requirement 3.1.4.5.3)
   6. Reset Button (Triggers Requirement 3.1.4.5.5)
3. Provide the ability to confirm all fields are completed. (Triggers Requirement 3.1.4.5.4)
4. Provide the ability to SQL query the Adverts database table to create the new Event. (Requirement 3.1.4.11)
5. Provide the ability to reset the fields of the form by clearing any previously entered input. (Requirement 3.1.4.5.2)
6. Provide an additional Close button next to Events that are ongoing in the HRT and Business GUI views, which will set the end time stamp for the particular Event to the current date/time via SQL query in the Adverts database table. (Requirement 3.1.4.5.1, 3.1.4.11)
7. **Ridership Trend Report (Brian Dunn)**

The Ridership Trend Report will provide ridership information for display on all three GUI Frameworks (Requirement 3.1.1.4).

i. Provide customization of data for the different authentication levels through the GUI Framework. (Requirement 3.1.1.4)

ii. Provide ability to select a date (YYYY -MM-DD) for which to view hourly report.

iii. Provide the ability to select a start date (YYYY-MM-DD) and end date (MM-DD-YYYY) display summarized report averaging data over the timespan.

iv. Provide a table output with ridership information: time, number of departures, and number of arrivals.

v. Must interface with DE to retrieve real-time data for reports. (Requirement 3.1.4.11.2)

1. **Train Data Report (Brian Dunn)**

The Train Data Report provide on-timer performance data for display on the HRT GUI (Requirement 3.1.1.4.2). The following functions shall be provided:

i. Provide ability to select a date (YYYY-MM-DD) for which to view hourly report.

ii. Provide the ability to select a start date (YYYY-MM-DD) and end date (YYYY-MM-DD) display summarized report averaging data over the timespan.

iii. Provide a table output with train performance information: time, on-time percentage, and delay time.

iv. Must interface with DE to retrieve real-time data for reports. (Requirement 3.1.4.11.2)

1. **Business Ad Campaign Module (Brian Dunn)**

The Business Ad Campaign Module will allow the Businesses GUI and HRT GUI to create and modify advertisement campaigns.

1. Must query the Adverts table to obtain a listing of the current user’s advertisement campaigns.
2. Provide an input form with the following fields:
   1. Ad Description (MEDIUMTEXT, Required)
   2. Start Date (MM-DD-YYYY, Required)
   3. End Date (MM-DD-YYYY, Required)
   4. Start Time (HH:MM:SS, Required)
   5. End Time (HH:MM:SS, Required)
   6. Stop Number (Int, Required)
   7. Image Upload (File, Optional)
   8. Submit Button (Triggers Requirement 3.1.1.3.6)

iii. Ability to submit/edit advertisement campaigns through the Business GUI. (Requirement 3.1.1.4.3)

iv. Ability to edit advertisement campaigns through the HRT GUI. (Requirement 3.1.1.4.2)

v. All required fields confirmed to contain proper data format and ad\_name generated as the business name combined with the ad\_id.

vi. Must interface with DB to submit, modify, and retrieve real-time data for advertisements. (Requirement 3.1.4.1.11.1)

1. **User Management (CJ Deaver)**

The User Management utility will give an application administrator the capability to manage system user accounts. It shall be designed as a web based management module with the following capabilities:

1. Include a registration page to allow individual users to input the following:
2. Given name
3. Desired username
4. Email address
5. Password
6. The username request should be validated and suggestions returned for alternates when selected name already exists.
7. Incorporate a method for a user to retrieve their username.
8. The capability to allow user to reset the password.
9. An additional user information update screen allowing a user access to change personal information is also required
10. Contain a web page to allow administrators to perform administrative tasks to the user accounts to include the following
11. Edit user information
12. Change applicable access group
13. Reset a user’s password
14. Administrative capability to create user groups and manage members.
15. **Authentication (CJ Deaver)**

Provide a methodology for security control utilized throughout the application. The following are the minimum requirements:

* + 1. A one-factor authentication mechanism for securely authorizing application access.
    2. Token generation or other means of identifying users.
    3. Access control mechanism controlling account access within the application.
    4. Security time-out feature to log users out after a set amount of time
    5. Include an account locking mechanism in the event of multiple failed login attempts.
    6. Logging capability recording the following:

1. Login time
2. Login location
3. Authentication success or failure
4. Page requested
5. **Data Integration Utility (CJ Deaver)**

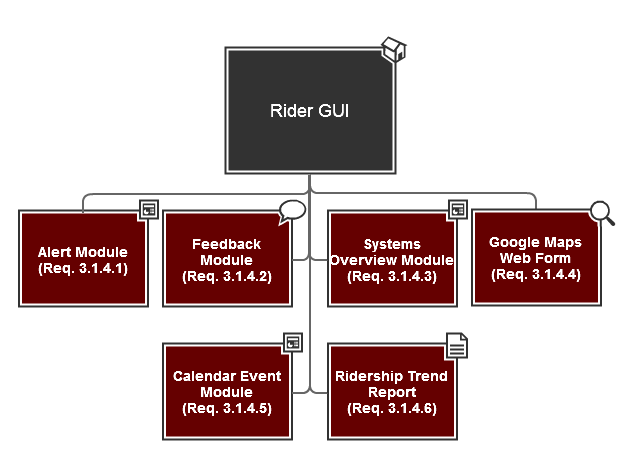
Provide an interface for connecting the web application to the different data sources to pass data throughout the system with the following requirements:

* + 1. DB Interface
       1. Requires the capability to open and close data stream connections
       2. Must complete transfer queries and results between the database and the Web Application Engine for the following modules:

1. Alert Module (3.1.4.1)
2. Feedback Module (3.1.4.2)
3. System Overview Module (3.1.4.3)
4. Calendar Event Module (3.1.4.5)
5. Ridership Trend Report (3.1.4.6)
6. Train Data Report (3.1.4.7)
7. Business Ad Campaign Module(3.1.4.8)
8. User Management Module (3.1.4.9)
   * + 1. Contain the ability to prevent SQL injection attacks.
     1. DE Interface
        1. Send and receive data between the decision engine and the Web Application Engine for the following modules:
9. Alert Module (3.1.4.1)
10. System Overview Module (3.1.4.3)
11. Ridership Trend Module (3.1.4.5)
    * 1. Test Harness Interface
         1. Receive data from the test harness and insert into the database or decision engine as applicable to replicate real-world data.
12. **Graphical User Interface Framework (Brian Dunn)**

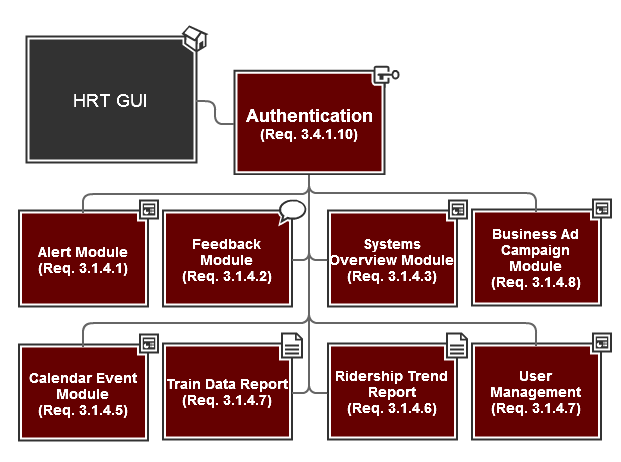
Rider Graphical User Interface

The Rider GUI will provide the following modules to riders on the website:



HRT Graphical User Interface

The HRT GUI will provide the following modules to users logged in with HRT permissions:



Business Graphical User Interface  
  
The Business GUI will provide the following modules to users logged in with Business Owner permissions:

