* + 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>
         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.

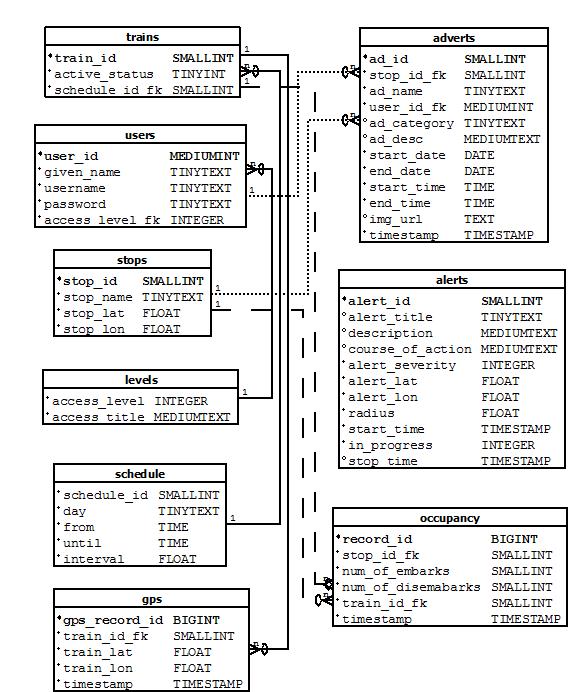


Figure 1. Database Schema

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

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 follow logic beginning in requirement 3.1.2.iv.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.v.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 model the values of departures and arrivals for the future date range.

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.v.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.v.f.i-vii

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

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

Provide the ability to apply the variance found in 3.1.2.1.v.i to the set of averages in 3.1.2.1.v.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 query the “trains” table for active trains.

Provide the ability to query the Current ITS database “GPS” table for past simulated arrival times at the station during the date and time range. (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.ii and 3.1.2.2.iii and return 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.

Provide the ability to compare the expected value of time-to-arrival from 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)

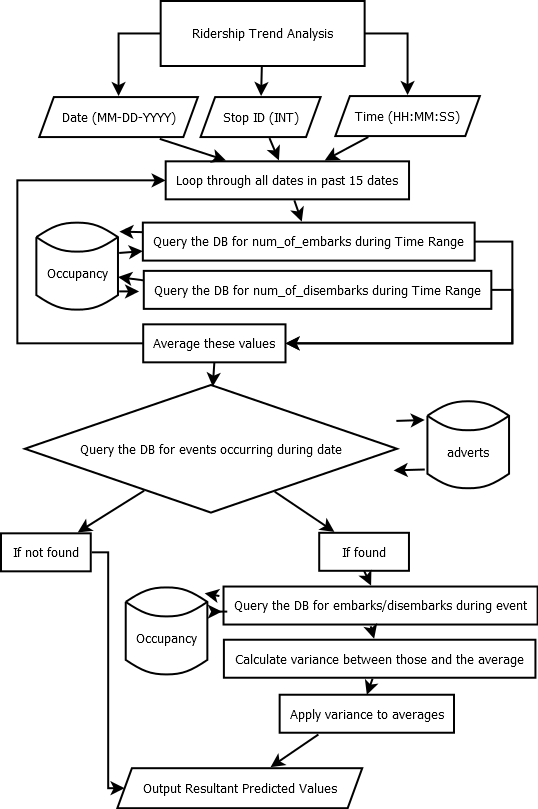


Figure 2. Ridership Trend Analysis Algorithm

**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 (Akeem)**

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)
   * + 1. **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)

2 Advertisements assigned to each stop.

3 Advertisement start time

4 Advertisement end time

3.1.3.5 **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)

* + - * 1. Provide a GUI with the ability to change ridership data at each stop
        2. Provide a GUI with the ability to edit each advertisement properties defined

1. Advertisements to each stop (3.1.3.4.2)
2. The End Date of an advertisement (3.1.3.4.1)
3. Advertisement Start Time
4. Advertisement End Time