A. Internal Software Data Structure

B. Global Data Structure

C. Temporary Data Structure

D. Database Description

E. NoSQL Database Description

**Internal Software Data Structure**

The data structure for the Ground Transportation System application is divided into a client side and server side. The client side data, represented by each company, will be stored locally and remotely on the server side database. The client will issue its unique identifier to the server when requesting or updating information on the database. This unique identifier will be used to distinguish each company from one another. The client will send a request command to the server with its unique identifier in order to receive the requested data. When updating the database, the client will issue an update command to the server with its identifier and desired data to be updated. The local data will then be changed to match the data sent to the server for database update.

The data structure on the server side will essentially be the same as the client side. The server will receive a request from the client and issue a command to the database based on the received data. The data returned from the database will be stored on the server side and then forwarded to the client and vice versa for database updates.

**Global Data Structure**

The global data structure for this application is represented by the database. The database will contain all the data needed by both the client and server. The client will not have direct access to the database. Instead, it will issue commands to the server which will then carry out the task of retrieving data and/or updating the database with new data.

**Temporary Data Structure**

Temporary data structures will be present on both the client and server. The temporary objects on the client side contain the data request and/or update information which is sent to the server. A temporary object will also be used to store the received data from the server until it is stored in the permanent data structure of the client. On the server side, a temporary object will be used to store the issued command sent by the client. If the command was a data request, a second temporary object will be created containing the received data from the database. This object will then be sent to the client and deleted once received. Weather information from the public database will also be stored in a temporary object within the client which will be used to calculate possible service types and routes.

**Database Description**

The relational database will contain a table for each company(client) and will be represented by a unique identifier. Each company table will contain a list of employees with the following attributes: Employee id number(primary key), employee name, employee age, employee gender, employee grade, zone start location of employee, zone end location of employee, cost of motorcycle transport, cost of cab transport, cost of bus transport, type of vehicle they travel in, and id of the vehicle.

The NoSQL database will contain a table for each type of service(motorcycle, cab, bus). Each service table will contain a list of all available vehicles with the following attributes: vehicle id(primary key), passenger capacity, and a cost to each zone.

Table structure for ‘company’ table:

CREATE TABLE ‘(company identifier)’ (

‘employee\_id’ int(10) NOT NULL auto\_increment,

‘employee\_name’ varchar(30) NOT NULL,

‘employee\_age’ int(3) NOT NULL,

‘employee\_gender’ varchar(6) NOT NULL,

‘employee\_grade’ varchar(10) NOT NULL,

‘zone\_start\_location’ varchar(10) NOT NULL,

‘zone\_end\_location’ varchar(10) NOT NULL,

‘moto\_cost’ decimal(5,2),

‘cab\_cost’ decimal(5,2),

‘bus\_cost’ decimal(5,2),

‘vehicle\_type’ varchar(10) NOT NULL,

‘vehicle\_id int(10) NOT NULL,

PRIMARY KEY(‘employee\_id’)

)

Table structure for vehicles:

CREATE TABLE ‘(vehicle type)’ (

‘vehicle\_id’ int(10) NOT NULL auto\_increment,

‘capacity’ int(3) NOT NULL,

‘zone\_a\_cost’ decimal(5,2),

‘zone\_b\_cost’ decimal(5,2),

‘zone\_c\_cost’ decimal(5,2),

etc...

PRIMARY KEY(‘vehicle\_id’)

)