**Ground Transport System Design Document**

**CS441, Group 3**

**Version 1.0**

**Fall 2013**

**By Jack Petraitis, Ibraham Al Thomali, Satabdi Aditya,   
Swaraj Gunda, Phani Vempalli, Kevin Richner, Marc Moylan**

**Table of Contents**

[**1.** **Introduction**](#h.1fob9te)

[1.1 Goals and Objectives](#h.3znysh7)

[1.2 Project Overview and Scope](#h.2et92p0)

[1.3 Project Context](#h.tyjcwt)

[**2.** **Data Design**](#h.3dy6vkm)

[2.1 Internal Software Data Structure](#h.1t3h5sf)

[2.2 Global Data Structure](#h.4d34og8)

[2.3 Temporary Data Structure](#h.2s8eyo1)

[2.4 Database Description](#h.17dp8vu)

[**3.** **Architectural & Component-Level Design**](#h.3rdcrjn)

3.1 System Structure

[3.2 Package Structure](#h.lnxbz9)

[3.3 Component Structure](#h.35nkun2)

[3.4 Class Structure](#h.44sinio)

[*3.4.1* *Processing Narrative*](#h.2jxsxqh)

[*3.4.2* *Interface description*](#h.z337ya)

[*3.4.3* *Processing detail*](#h.3j2qqm3)

[**4.** **User Interface Design**](#h.1y810tw)

[4.1 User Interface](#h.4i7ojhp)

[*4.1.1* *Objects and actions*](#h.2xcytpi)

[**5.** **Restrictions, limitations, and constraints**](#h.1ci93xb)

[**6.** **Technology Specification**](#h.2bn6wsx)

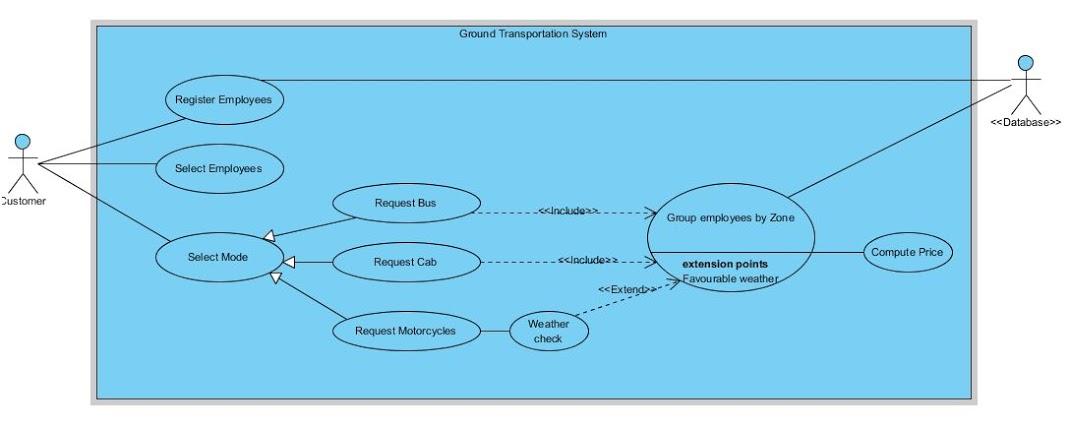
[**7.** **Design Confirmation/Stakeholder sign-off**](#h.qsh70q)

# Introduction

## ***Goals and Objectives***

Most companies have transportation services for employees. The main objective of this software is to minimize the cost of transportation for the employees. For example: The employees living in the same area would be able to share a vehicle to work. The factors that have been used for the minimization of cost using this software are: the type of vehicle used, the number of vehicles needed and the distance between the company and the area where a set of employees needs to go.

The distributed Ground Transportation System provides the ability for companies to offer transportation to employees using one of three transportation types of vehicles. They are motorcycle, cab, or a bus. Each type has a different capacity with a standard rate per distance. The distance is based on zones. A zone is a location identifier that is assigned to the physical addresses (address of company as well as address of employees). The price is based on the distance i.e. zone to zone and type of vehicle. The main purpose of this design document is to demonstrate the methods we have used to design the software and also the major objective of the software. It also provides a brief outline of the features of our distributed system to companies who would like to implement it with their current logistics. It aims at describing how the problem is being solved, and the different features that will be required in order to have an acceptable solution to this problem. The scope of this document also includes the design of the important features and a loose description as to how our program will function to meet these requirements. To recap, I have also included the use-case diagram of the application below:



There are three major functionalities that the software does:

·      **Register Employees**- This is to add new employees within the system. This would include details of the employee like the address and what kind of transportation he would prefer. This information would help us to divide employees based on zones and then assigning the type of vehicle based on the number of people travelling to one area.

·      **Select Employees**- Once the employees have been registered, we divide the employees based on zones. This functionality helps us to group the employees based on their location.

·      **Select Mode**- Here, when a particular employee selects a type of transport, a database search is done as to how many employees living in the same area have asked for transportation. Once the data has been fetched, the total cost of travel has been computed based on they type of vehicle he asks for. In certain cases when the employee looks for motorcycle as his preferred mode of transportation, it also has the option of checking whether the weather is suitable to ride a motorcycle. If it is, he is assigned one.

## ***Project Overview and Scope***

This product is a conglomeration of other products out on the market today. There are no comprehensive human logistic solutions available in our market. Our distributed system takes the major features of publicly available transportation APIs and combines them with privately contracted transportation options.

Our program is supposed to act as an interface to every mode of ground transportation available to get an employee to work. The different distributed objects that would be used are:

·      **Company**: The Company is a client in the system that has employees that need transportation. When the client starts the company generates/registers the number of employees with their address details (automatic random generation including the location of that company). This data is stored in the database. The company has an option to select the type of vehicle and also number of employees that need transportation. The system then contacts the TMS object and a cost is returned. Since this is the client in our distributed system, it can have multiple instances.

·      **Transportation Management System (TMS)**: The TMS has communication with three service providers that provide their own service. For example a cab service provides transportation using cabs only. Now, on receiving the request from the Client/Company, based on client’s choice of vehicle type TMS forwards the request to the service provider. TMS has contact with a public API to access weather information. If it is raining it will not forward the request to the motorcycle service provider since it cannot operate in the rain.

·      **Databases**: This object is responsible for generating/registering employees into the database for each client. It will also be used as the skeleton for interaction with the database.

·      **CabServices**: This object will calculate how many cabs are needed to transport from one zone to another and the total cost. It has access to the database and can calculate cost of transport for all employees in each zone for all zones. A vehicle travels only to one zone.

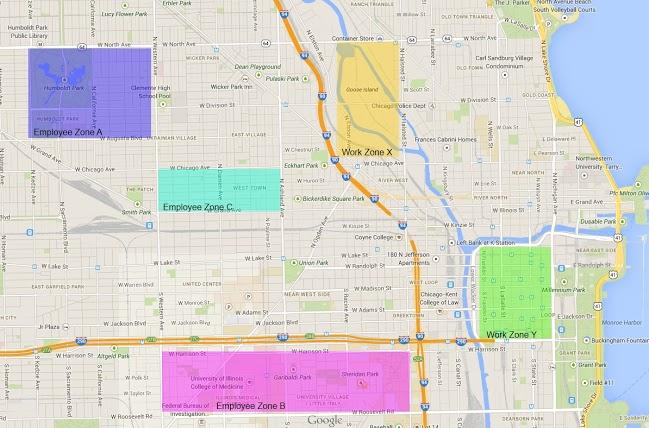
·      **MotorcycleServices**: This object will calculate how many motorcycles are needed to transport from one zone to another and the total cost. This type of vehicle cannot operate in the rain. It has access to the database and can calculate cost of transport for all employees in each zone for all zones. A vehicle travels only to one zone.

·      **BusServices**: This object will calculate how many buses are needed to transport from one zone to another and the total cost. It has access to the database and can calculate cost of transport for all employees in each zone for all zones. A vehicle travels only to one zone.

The Ground Transportation System does utilize a public API. The system will use a public API from a weather service to determine if the vehicle can operate in current weather conditions.

## ***Project Context***

The reason behind the development of this software can be better described with the help of this example.



The picture above is a single instance of how the Ground transportation system can be used. Here there are two work zones: Work zone X and work zone Y. With the employee information and which work zone they are requesting the particular vehicle from, the users are divided into particular Employee zones. Thus the company incurs the minimum cost of travel.

Please note that the picture above is NOT indicative of our program’s user interface. It is only meant to better describe the context of our problem at hand.

# Data Design

## ***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’)

)

# Architectural & Component-Level Design

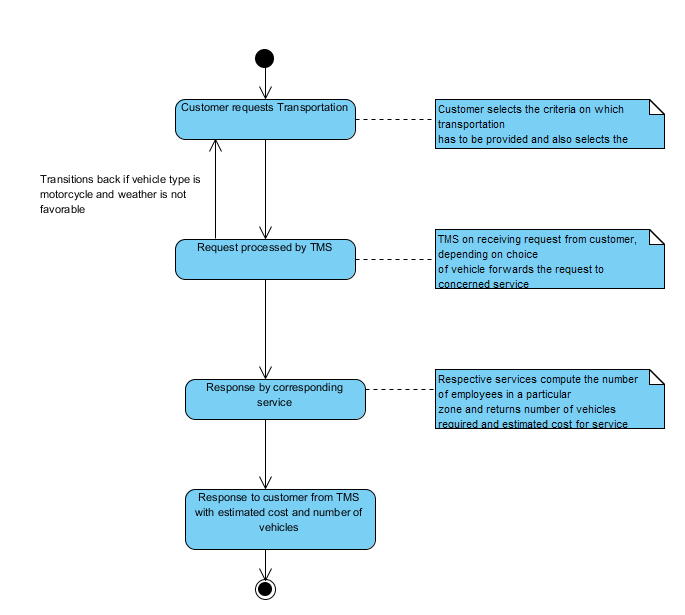
## ***System Structure***

The system structure is based on a distributed environment deployment. A company is the client to the system and may have multiple remote deployments. The Transportation Management System is the main server for all client requests requiring a vehicle service. Nevertheless, the company

can directly register all employees with the database server. The Transportation Management System is also responsible for contacting an internet API for a weather service. The information gained from the

internet source is used to determine if a type of vehicle is able to operate in the current weather. Each type of vehicle service (Cab Services, Motorcycle Services, Bus Services) is a distributed part of the system. Hence, the system structure, for its main components, is a distributed structure that utilizes a database to maintain the data and uses a public API for some of its internal logic.

The following State Diagram illustrates how our system operates:

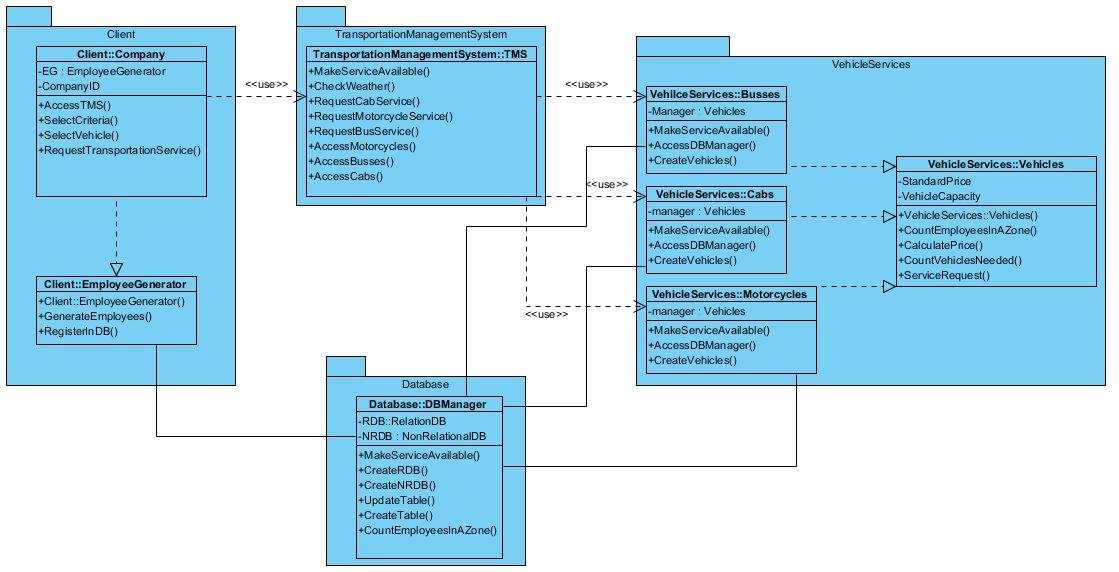
*Explanation of states:*

**Customer requests Transportation:** The customer decides to provide transportation to his employees. The transportation is provided only to employees who agree with certain criteria like employees in certain age group or all female employees etc. After the criteria are selected the customer has the choice to select the type of vehicle from the available list. The vehicle list includes bus, cab and motorcycle. When these details are filled, transition of state happens and now its task of TMS to process the request from client.

**Request Processed by TMS:** In this state the TMS has information about the criteria of employees on which transportation has to be provided and the type of vehicle. Depending on the customer choice of vehicle the TMS transfers the request to corresponding services. In this state if the vehicle type is motorcycle, TMS performs an additional check on weather. If the weather is not favorable, the system transitions back to previous state of selecting vehicle mode.

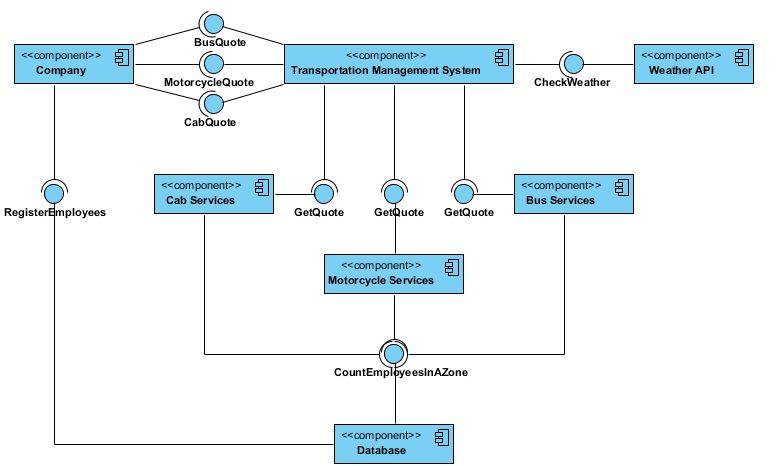
## ***Package Structure***

Using a simplified class diagram, the packages required by the system are represented using this package diagram. This initial design uses four packages to develop the system. The client package contains the necessary classes to create a company and generate employee data. The ability to store and update data in the database is in the database package. The transportation management system package contains the functionality to serve any company quote, the ability to contact APIs available on the internet, and forward quote requests to the vehicle services. Vehicle serves has three servers composed of the vehicle type. They were grouped together because of common functionalities between these services. It is still possible to contain each type of vehicle service in separate package if these services are to be deployed on distributed remote locations. The vehicle services package and through its main vehicle type component know how to contact the database and make the necessary calculations to provide a quote.



## ***Component Structure***

There are seven main components of the Ground Transportation System. Each has its own functionality to ultimately provide the cost of transporting a group of employees to their family destination and identifying the number of vehicles needed. The company component is responsible for generating and registering employees with the database component which will store it in a database. The company component can request a quote from the transportation management system for any of the three vehicle types (Motorcycle, Bus, Cab). The transportation management system is responsible for insuring proper weather condition for operating a vehicle. It gains this information from the weather API component which is provided from an internet source. The quote is forwarded to one of the vehicle components where they will be able to respond with a quote after obtaining the necessary information from the database component.



## ***Class Structure***

* *Explain how the client class is spit up.*
* *Explain what it represents*

### **Processing Narrative**

*What object is created when this class is newed up. What are the data members in that class?*

### **Interface description**

* *Provide sample getters and setters*
* *e.g. :  
  string getName();*

setName(string);

etc

### **Processing detail**

# User Interface Design

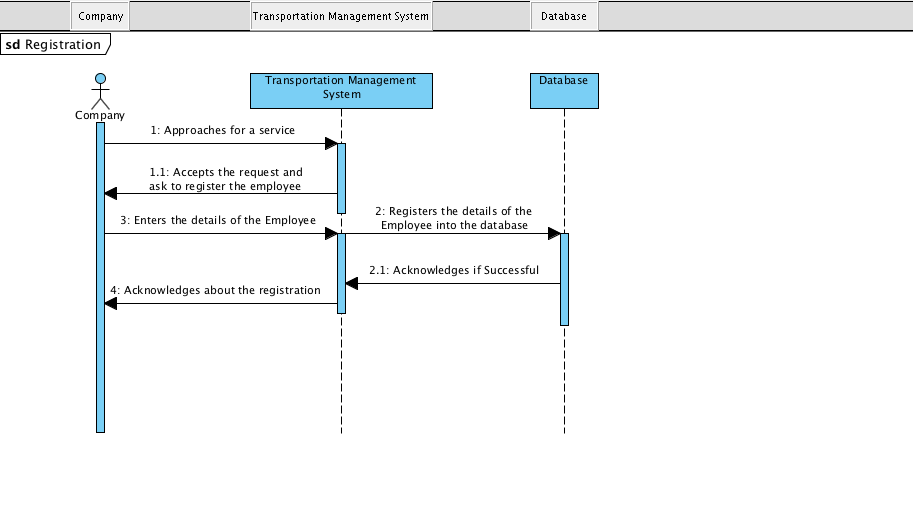
## ***User Interface***

The user will interact with our client program by using a web interface. Buttons and selections on these web pages will trigger functions in the client program that will attempt remote procedure calls to our server program. The actions that they can perform from within this web interface are detailed in the following sub-section.

### **Objects and actions**

|  |  |
| --- | --- |
| **Use Case Name:** | Register Employees |
| **Actors:** | Customer |
| **Preconditions:** | Database should be running. |
| **Description:** | 1. The company wants to provide transportation for all the employees. 2. The company registers details like name, age, grade, gender of all the employees into the database.     1. The employee details are populated automatically. |
| **Postconditions:** | All company employees are in the database. |

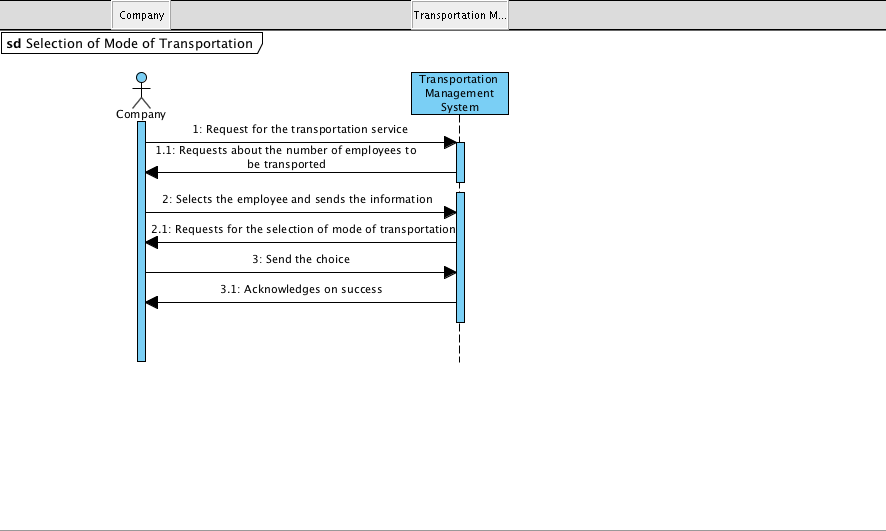
The following sequence diagram illustrates this user scenario:



|  |  |
| --- | --- |
| **Use Case Name:** | Select Employees |
| **Actors:** | Customer |
| **Preconditions:** | The database should be up and running. It should have details of all the employees. |
| **Description:** | 1. The system shows the criteria available for choosing employees who need transportation. 2. The customer chooses one criteria from the available options. The criteria can be from below     1. Age of the employee    2. Gender of the employee    3. Grade of the employee |
| **Postconditions:** | We have the criteria to be used for selecting employees who need transportation. |

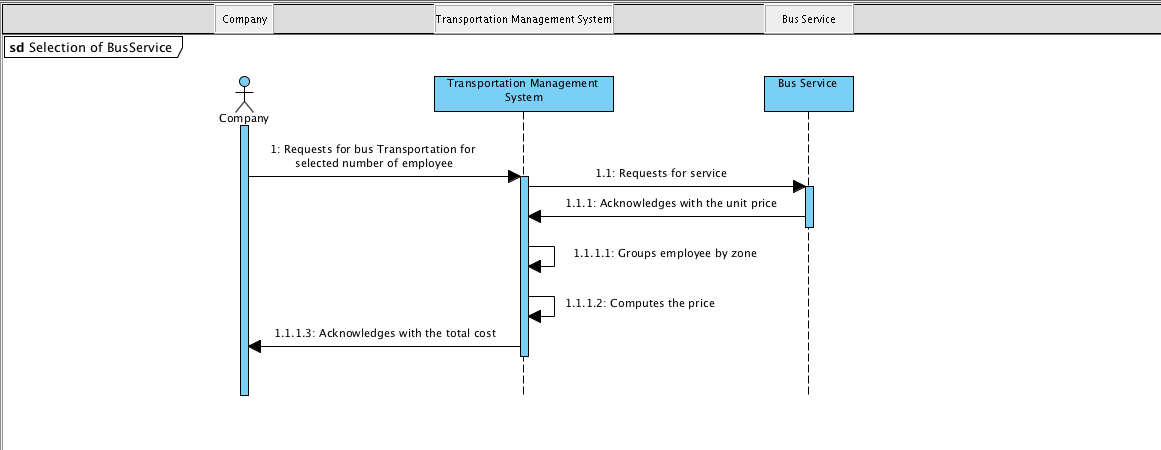
|  |  |
| --- | --- |
| **Use Case Name:** | Select Mode |
| **Actors:** | Customer |
| **Preconditions:** | The database should be up and running. The different modes of transportation should be available in the database. |
| **Description:** | 1. The company wants a specific type of vehicle for transportation of all the employees. 2. The company can choose one vehicle type from the following     1. Bus    2. Cab    3. Motorcycle |
| **Postconditions:** | Vehicle type for transportation is chosen. |

The following sequence diagram illustrates the Select Mode user scenario:



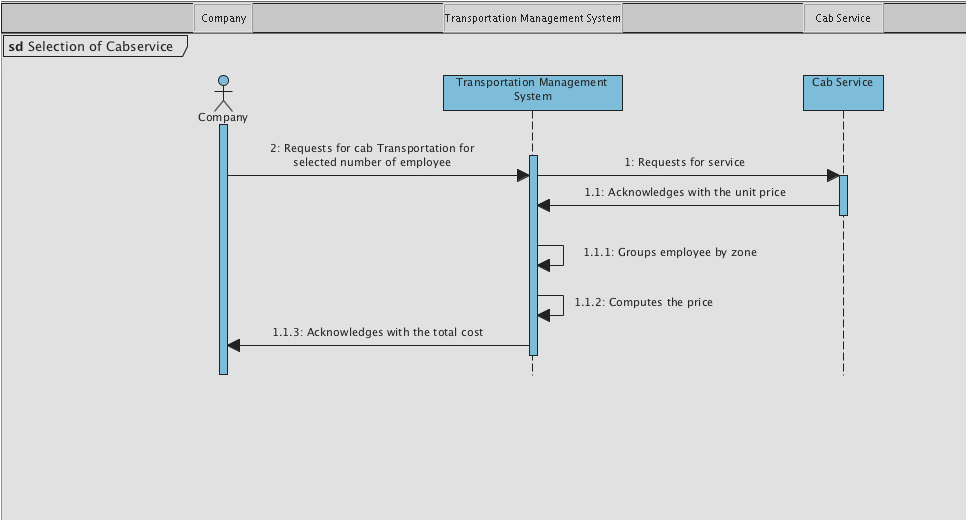
|  |  |
| --- | --- |
| **Use Case Name:** | Request Bus |
| **Actors:** | Customer |
| **Preconditions:** | Selected vehicle type should be ‘Bus’ |
| **Description:** | 1. The company would like to use bus for transportation. 2. The request is sent to Transport Management System with the specified employee criteria. |
| **Postconditions:** | The company gets the total price and number of busses needed for transportation. |

The following sequence diagram illustrates the Request Bus user scenario:



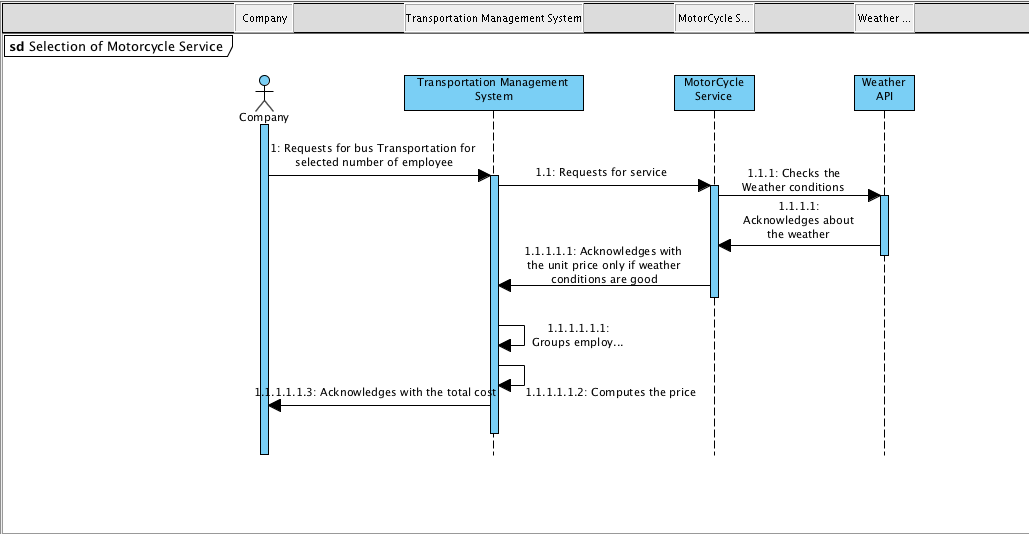
|  |  |
| --- | --- |
| **Use Case Name:** | Request Cab |
| **Actors:** | Customer |
| **Preconditions:** | Selected vehicle type should be ‘Cab’ |
| **Description:** | 1. The company would like to use cab for transportation. 2. The request is sent to Transport Management System with the specified employee criteria. |
| **Postconditions:** | The company gets the total price and number of cabs needed for transportation. |

The following sequence diagram illustrates the Request Cab user scenario:



|  |  |
| --- | --- |
| **Use Case Name:** | Request Motorcycle |
| **Actors:** | Customer |
| **Preconditions:** | Selected vehicle type should be ‘Motorcycle’ |
| **Description:** | 1. The company would like to use motorcycle for transportation. 2. The request is sent to Transport Management System with the specified employee criteria. |
| **Exception**: | Required vehicle type is not available because of bad weather. The customer is informed of this situation and must provide another mode of transportation. |
| **Postconditions:** | The company gets the total price and number of motorcycles needed for transportation. |

The following sequence diagram illustrates the Request Motorcycle user scenario:



|  |  |
| --- | --- |
| **Use Case Name:** | Weather check |
| **Actors:** | Customer |
| **Preconditions:** | Selected vehicle type should be ‘Motorcycle’. Weather public API is available. |
| **Description:** | 1. The information about weather is required for motorcycle to operate. 2. Connect to weather API and get the weather information. 3. The motorcycle transportation is possible when the weather is good. |
| **Exception**: | Required vehicle type is not available because of bad weather. The customer is informed of this situation and must provide another mode of transportation. |
| **Postconditions:** | The information about the weather is obtained. |

|  |  |
| --- | --- |
| **Use Case Name:** | Group employees by zone |
| **Actors:** | Customer |
| **Preconditions:** | The source and destination zones of employee should be available. The weather should be favorable for transportation. |
| **Description:** | 1. The system must determine number of employees in each zone. |
| **Postconditions:** | The number of employees in each destination zone is determined. |

|  |  |
| --- | --- |
| **Use Case Name:** | Compute Price |
| **Actors:** | Customer |
| **Preconditions:** | The number of employees and vehicle type are available. |
| **Description:** | 1. The system computes the number of vehicles needed. 2. The system will compute the price for transportation of selected employees. 3. The system will use defined price for this computation. |
| **Postconditions:** | The price and number of vehicles required for transportation are computed. |

# Restrictions, limitations, and constraints

The constraints on our project include:

* Databases must be between 300MB and 2GB.
* Must have 20 database tables with 5 average attributes
* Each table must have primary keys and some foreign keys
* Data must come from a publically available source using another API

# Technology Specification

We will use the following technology stack to realize our goals:

* MySQL
* MongoDB
* Java RMI
* HTML & CSS
* Ubuntu Server 2012

# Design Confirmation/Stakeholder sign-off

|  |  |  |
| --- | --- | --- |
| **Meeting Date** | **Attendees (name and role)** | **Comments** |
| 9/26/13 | Jack Petraitis,  Ibraham Al Thomali,  Satabdi Aditya,  Swaraj Gunda,  Phani Vempalli,  Kevin Richner,  Marc Moylan | Confirmed |

# 