**Ground Transport System Requirements Specification**

**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.** **Executive Summary**](#h.1fob9te)

[1.1 Project Overview](#h.1fob9te)

[1.2 Purpose and Scope of this Specification](#h.3znysh7)

[**2.** **Product/Service Description**](#h.2et92p0)

2.1 Product Context

[2.2 User Characteristics](#h.3dy6vkm)

[2.3 Assumptions](#h.1t3h5sf)

[2.4 Constraints](#h.4d34og8)

[2.5 Dependencies](#h.2s8eyo1)

[**3.** **Requirements**](#h.17dp8vu)

3.1 Functional Requirements

[3.2 User Interface Requirements](#h.26in1rg)

[3.3 Usability](#h.26in1rg)

[3.4 Performance](#h.35nkun2)

[*3.4.1* *Capacity*](#h.1ksv4uv)

[*3.4.2* *Availability*](#h.44sinio)

[*3.4.3* *Latency*](#h.2jxsxqh)

[3.5 Manageability/Maintainability](#h.z337ya)

[*3.5.1* *Monitoring*](#h.3j2qqm3)

[*3.5.2* *Operations*](#h.1y810tw)

3.6 System Interface/Integration

[*3.6.1* *Systems Interfaces*](#h.2xcytpi)

[3.7 Security](#h.3whwml4)

[*3.7.1* *Protection*](#h.2bn6wsx)

[3.8 Data Management and Design](#h.qsh70q)

[*3.8.1* *Internal Software Data Structure*](#h.3as4poj)

[*3.8.2* *Global Data Structure*](#h.1pxezwc)

[*3.8.3* *Temporary Data Structure*](#h.49x2ik5)

[*3.8.4* *Database Description*](#h.2p2csry)

[**4.** **User Scenarios/Use Cases**](#h.147n2zr)

[**5.** **Deleted or Deferred Requirements**](#h.3o7alnk)

**6.** **Requirements Confirmation/Stakeholder sign-off**

# Executive Summary

## ***Project Overview***

Some companies have the responsibility of transportation of employees. Such employers need a way to calculate the cost of providing this service to all or some employees. Some employees may live in the same area as other employees and hence they can travel together to save the company some money. This system provides the cost of this service depending on the type of vehicle used, number of vehicles needed, and the distance between the company and the area where a set of employees need to go.

The distributed Ground Transportation System provides the ability for companies to offer transportation of employees using one of three transportation types of vehicles. The types are a motorcycle, cab, and 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.

## ***Purpose and Scope of this Specification***

The purpose of this specification is to outline the features of our distributed system to companies who would like to implement it with their current logistics.

This documentation is meant to describe the context of the problem being solved, and the different features that will be required in order to have an acceptable solution to this problem. This document is aimed at potential customers or reviewers of our product.

The scope of this document includes recommendations for important features and a loose description as to how our program will function to meet these requirements. Items outside the scope of this document are design implementation.

# Product/Service Description

## ***Product Context***

This product is a conglomeration of other products out on the market today. There are no comprehensive human logistic solutions available on 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 modes are Cab Services, Motorcycle Services, and Bus Services. Each of these services will be briefly touched on here, but more thoroughly explored in Section 3.6.1.

The Cab Services interface will work with a cab server that we program to calculate how many cabs will be needed to transport employees from their home zone to the work zone. This will also calculate the total expected cost for the trips.

The Motorcycle Services interface will work with a motorcycle server that we will populate with our own data. This part of the program will find out how many motorcycles would be required to take all the employees from a certain zone and transport them to the work zone. An interesting note about this service is that it cannot operate in the rain. This will be discussed further in Section 3.6.1.

The Bus Services interface will connect to a bus server. The objects retrieved from this information database will help us calculate how many buses are needed to transport employees in one bus service zone to the work zone.

The Train Services interface will connect to a train server. The objects retrieved from this information database will help us calculate how many trains are needed to transport employees in one train service zone to the work zone.

## ***User Characteristics***

The typical users of this program will be the following:

* Human Logistics Engineer
* Transportation Manager
* Financial Reviewer

## ***Assumptions***

We are assuming that the operating systems that run the client systems have a java compiler and are capable of running jar files generated and designated as the client. We are also assuming that the server programs developed are installed onto Linux machines.

## ***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

## ***Dependencies***

The program will depend on the data availability from a weather service, the CTA bus information service, and Google maps APIs.

# Requirements

## ***Functional Requirements***

In the table below, the requirement numbering has a scheme - BR\_0## (BR for Business Requirement). This scheme was designed with having the goal of making the traceability matrix easier to create. SME stands for “Subject Matter Expert”. TMS = “Transportation Management System”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req#** | **Requirement** | **Comments** | **Priority** | **Date Rvwd** | **SME Reviewed / Approved** |
| BR\_01 | Upon client startup connect to weather services via WeatherAPI | Used for motorcycle service | 1 | 9/26/13 | Ibrahim |
| BR\_02 | Upon client startup connect to Motorcycle service | Requires successful Weather connection | 1 | 9/26/13 | Phani |
| BR\_03 | Upon client startup connect to Bus Service |  | 1 | 9/26/13 | Jack |
| BR\_04 | Upon client startup connect to Cab Service |  | 1 | 9/26/13 | Swaraj |
| BR\_05 | Upon client startup connect to Train Service |  | 1 | 9/26/13 | Satabdi |
| BR\_06 | Client has the ability to generate random database of employees | This should be a command or a button | 2 | 9/26/13 |  |
| BR\_07 | The client connects to a TMS server |  | 2 | 9/26/13 |  |
| BR\_08 | The client can choose a type of transportation and the TMS object calculates the number of employees who can take this mode of transportation |  | 2 | 9/26/13 |  |
| BR\_09 | The client can have the TMS object calculate the financial burden of transporting employees |  | 2 | 9/26/13 |  |
| BR\_10 | There can be multiple clients performing operations using the TMS object |  | 2 | 9/26/13 |  |
| BR\_11 | A transportation manager has the ability to take down a server instance |  | 3 | 9/26/13 |  |

## ***User Interface Requirements***

## ***Usability***

There should be a help context menu that is available by entering the help command into a command line interface.

## ***Performance***

The numerical performance requirements for our program include

* Supporting at least 5 simultaneous users per one server instance.
* Completing up to 95% of transactions in less than a second.
* Populating a randomly generated employee database in less than two minutes.

### **Capacity**

The maximum users we are expecting with one server instance is 5. We are expecting this number to scale as the managers spin up new server instances. We will be experimenting with provisioning of resources using a virtual network to better define this capacity.

### **Availability**

The program is expected to be available:

* Twenty-four hours a day
* In the Chicago area
* Downtime shouldn’t affect users because servers will be updated sequentially, balancing the load across other servers as each one is taken down to be updated

### **Latency**

The maximum acceptable latency time is 20ms.

## ***Manageability/Maintainability***

### **Monitoring**

We will rely on the hosting company’s hard drive and system health monitors already in place. S.M.A.R.T. hard disk logs will be gathered periodically.

### **Operations**

Normal operations available to the user include:

* Creating, Updating, and Deleting employees
* Randomly generating and populating employee databases
* Taking a server instance down (Manager only)

## ***System Interface/Integration***

### **Systems Interfaces**

Systems interface requirements:

1. *Client to TMS System Interface*

*The client will request a remote object to be created on the TMS Server application. This object will handle calls to further extrapolated systems and return necessary information to the client.*

1. *TMS System to Weather System Interface*

*The TMS System will now act as a client and request weather information from the Weather server. This Weather server will be part of a publically-available system that can support API calls. This weather data is necessary for the next system to function.*

1. *TMS System to Motorcycle System Interface*

*The TMS System will now act as a client and request a motorcycle management object. Based on the weather data gathered in the previous statement the system will decide if motorcycles are capable of transporting that day and how much it will cost.*

1. *TMS System to Bus System Interface*

*The TMS System will act as a client and request a Bus Management object. This will then calculate the cost of transportation and how many buses are needed for a zone.*

1. *TMS System to Cab System Interface*

*The TMS System will act as a client and request a Cab Management object. This will then calculate the cost of transportation and how many cabs are needed for a zone.*

1. *TMS System to Train System Interface*

*The TMS System will act as a client and request a Train Management object. This will then calculate the cost of transportation and how many trains are needed for a zone.*

## ***Security***

### **Protection**

The factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse are SQL user accounts.

* Only managers can take down server instances or delete data
* All other users can create, edit, and randomly generate employee data

## ***Data Management and Design***

Information that is to be placed into a database, includes

* Randomly generated employee data, including addresses and zones
* Workplace zones that the employees need to get to
* Company data for each company that uses the service
* Cab data
* Train data
* Bus data
* Motorcycle data
* Weather data

### **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’)

)

# User Scenarios/Use Cases

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **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. |

# Deleted or Deferred Requirements

There are no deleted or deferred requirements.

# Requirements 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 |

# 