

## Team 2: DaimlerChrysler

### Fleet Communication Replacement -- Technical Specification

#### Project Description:

Daimler Chrysler Trucking operates a fleet which supports shipments of parts and components throughout Michigan, Ohio, Indiana and Ontario. Current communication costs are unacceptably high and data transfer slow with present technology. Our solution is to replace the hardware with a device from WebTech Wireless which uses GSM/GPRS technology to wirelessly send/receive data through a wireless high-speed network. The device allows the capability with real-time data interchange through an Internet browser application also being developed. This application will allow drivers and dispatchers to interact with one another through messaging, routing and other functionality. Features include the capability to download plain-text routes, to upload status information of shipments and to allow dynamic re-routing.

#### Hardware Specification:

**Purpose:** Provide a reliable cost-effective alternative to satellite communications being used.

**Overview:** The WebTech Wireless WT4000 Locator device provides a wireless Internet connection through a GSM/GPRS network connection provided by a wireless network carrier (T-Mobile). The WT4000 Locator device communicates with the cellular network to provide Internet and data communication capabilities. A server is setup that collects and processes the vehicle data and provides message queuing. A website running on the server processes that data and processes it in a user friendly web application that a dispatcher can see. Dispatchers can send back data to a fleet truck driver. The truck driver in turn uses the web application to receive information from the dispatcher.

**Device:** (Pictured below)



- Network: GSM/GPRS supported through T-Mobile cellular network.
- Dimensions: 105 x 95 x 42 mm.
- Weight: .79 lbs.
- Power: Input Voltage – 9 – 36 VDC.
- Operating Temperature: -30 degrees Celsius to +60 degrees Celsius.
- Storage Temperature: -40 degrees Celsius to +80 degrees Celsius.
- Humidity: 95% max
- Power: 3 wire (ignition, battery, ground)
- Connection device: Laptop Computer (Windows XP OS)
- Protocol: PPP on serial port (USB connection used)

**Features:**

- GPS-based location and status reporting on GSM/GPRS networks
- Geofences on specific landmarks/locations (messages/alerts sent when vehicle enters or exits specifically defined references)
- Internet access through GSM/GPRS connection
  - Supports end-to-end Internet based applications on laptop.
  - “Always on” high-speed GPRS connection (30-60 kbps).

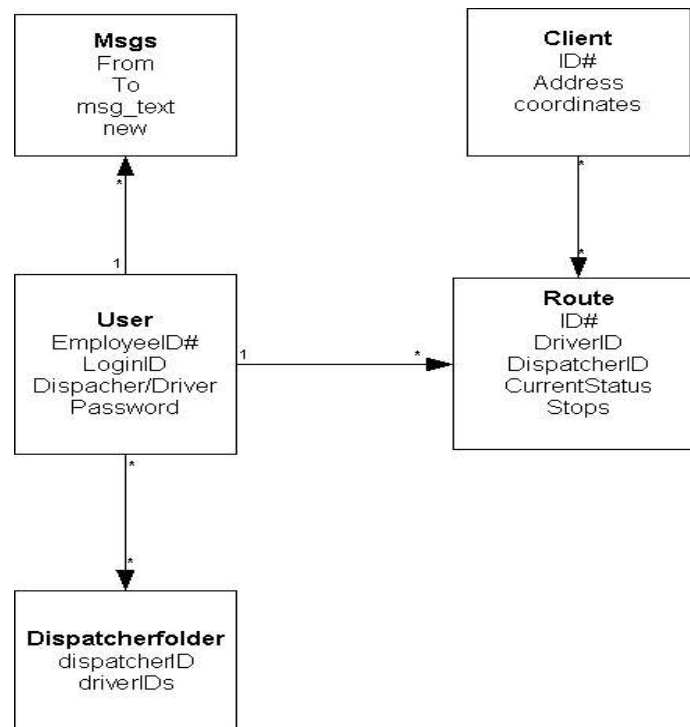
**Optional Features:** (could be implemented in future)

- Telemetry Port (Vehicle monitoring & control capability – unlock doors, panic button, etc.).
- J1708 engine control module interface.
- Hands-free voice access.

**Network Coverage Area:** (Michigan, Ohio, Indiana and Ontario)



## Data Model



**Description:** The database is a MySQL nt server running as a standard Windows NT service on the server. It connects to the web application via ODBC. It uses the MySQL ODBC 3.51 driver. All data access to the database is done through the class `DBInterface`. This has two advantages, it abstracts the use of the ODBC .NET Data provider, allowing for easier programming of multiple web forms without having to duplicate the code for connecting to the database. All the programmer has to do to access the database is instantiate a `DBInterface` object, all the overhead and connection strings are built into `DBInterface`. The other advantage is that it hides the functionality of the database, making the the web application more secure.

## The Database dcx1:

### Tables:

users  
clients  
msgs  
routes  
dispatcherfolders

**users table:** The `users` table is fairly straight forward. Two distinctive users types are distinguished by the field `userType`. The two types are DRIVER and DISPATCHER. They represent the truck drivers and the dispatchers respectively.

Name	Type
ID	int
UserID	varchar(100)
UserType	enum('DRIVER', 'DISPATCHER')
Password	varchar(100)

**clients table:** The `clients` table contains a list of clients that are served. These clients can be either DiahmleChrysler suppliers, or other divisions of DiahmleChrysler itself.

Name	Type
ID	int
Name	varchar(100)
StreetAddress	varchar(250)

**routes table:** The `routes` table describes a route defined by the dispatcher (`DispatcherID`) and assigned to a driver (`driverID`). The list of stops is a comma delimited list of `clientID`'s. `CurrentStatus` indicates what the driver is doing right now

Name	Type
RouteID	int
DispatcherID	int
DriverID	int
CurrentStatus	enum
Stops	varchar(255)

enumeration: `CurrentStatus`

<u>Status</u>	<u>Description</u>
ONDUTY	On the clock and driving
ONDUTYSTOPPED	On the clock, but stopped for some reason, usually waiting at a client
OFFDUTY	Off the clock
ONDUTYBREAK	Taking a legally mandated break from driving
OFFDUTYSLEEPING	Off the clock and sleeping during a route
ONDUTYWEIGHSTATION	On-duty but stopped at a weigh-station

**dispatcherfolders table:** The `dispatcherfolders` table describes which drivers, as defined by a comma delimited list, are assigned to which dispatchers.

Name	Type
dispatcherID	int
driverIDs	varchar(150)

**msgs table:** The `msgs` table contains all messages sent to and from dispatchers and drivers. `From` and `to` fields use user IDs, `msg_text` is a text field containing the text of the message. `New` is a bool indicating a new message. `Timestamp` is the time the message was sent. There is no restriction on who can send messages to whom.

Name	Type
ID	int
From	int
to	int
msg_text	text
new	bool
timestamp	int

## Web Application Specification:

**Purpose:** To provide services to the truck drivers and the purchasers (dispatchers), over a web application written in ASP.Net with C#. Developed on Visual Studio .Net 2003. (See attached figure)

**Namespace:** DCX (under development.)

**Classes:** MainPage  
TruckerPage  
DispatcherPage  
RoutePage  
NewRoutePage  
RouteMapsPage  
MessagePage  
ReadMessagePage  
WriteMessagePage

**Overview:** The web application is designed to facilitate communication between truck drivers and dispatchers. Both drivers and dispatchers have to log in with their own unique ID and password in order to access the application. With the current application design, drivers can get the current routes, new routes, communicate with dispatcher(s) using text message exchanging, and mapping service. Dispatchers can dynamically re-route the route(s) and notify the driver(s), track truck(s), and exchange text messages with driver(s).

**MainPage:** Contains a login interface with `username` and `password` input and if either username and/or password is incorrect it will display a text message to indicate the error, or else it will redirect the user to the next page. Next page will be depending on the user's types (dispatcher or driver). If the user is a driver, the next page will be a trucker page, if the user is a dispatcher, the next page will be a dispatcher page. User's types will be checked with the database.

Login

Checks for authentication of `username` and `password`.

**TruckerPage:** If a user has successfully logged in and the `usertype` is driver, this is the page where the user will get redirected to. This page contains all the functions and information that the driver will need. This page will let the driver to check for current routes that he/she will need to complete or already completed. This page will also display driver's current status and location. Communication functionality is also in this page for the driver's convenience.

CurrentRoute

Returns a list of current completed and uncompleted client stops.

StatusAndLocation

Returns driver's current status and location.

SendMessage

Sends a new message to the dispatcher.

CheckMessages

Checks for new message.

NewMessage

If there is a new incoming message, this will notify the driver.

**DispatcherPage:** If a user has successfully logged in and the user type is dispatcher, this is the page where the user will get redirected to. This page contains all the functions and information that the dispatcher will need. Dispatcher will be able to view the incomplete routes from every driver he/she is responsible for. The dispatcher can also create a new route. Text message communication functionality is also in this page for the dispatcher's convenience.

OutstandingRoutes

Returns a list of routes waiting to be complete.

CreateRoute

Dispatchers can create a new route.

SendMessage

Sends a new message to the driver(s).

CheckMessage

Checks for new message.

**NewRoutePage:** When the dispatcher wishes to create a route, he/she will get redirect to this page. This page simply requires the dispatcher to enter the information of the clients and trucker which will be assigned to accomplish this new route.

Clients

Adding stops at client(s) into the list of routes.

TruckerID

Assigning the new route to the driver(s).

**RoutesPage:** After the new route is created, the dispatcher will be redirected to this page. This page will show the status and location of the assigned driver and confirm the assignment of the new routes.

Clients

Display client's information. Information will include names, address, direction, and shipments information.

TruckerID

Indicates which driver should take which truck and perform which route(s).

StatusAndLocation

Returns the status of the driver and location.

**RouteMapsPage:** This is an external map service which will provide accurate maps for the driver's convenience. This may or may not be implemented because of data transferring speed through the network.

Still under development. Will possibly use Microsoft MapPoint.

**MessagesPage:** When the user, driver or dispatcher, click on check messages, this page will be executed. This page will allow the user to view their inbox which will be a list of read or unread messages. Also this page will allow the user to write a new message and read a message.

NewMessages

If there is a new incoming message, this will notify the user.

ReadMessages

Lists all the read or unread messages. Messages are sorted by received date and time.

WriteMessages

Users can write a new message.

**ReadMessagesPage:** This page will display a message and will also allow the user to reply the message back to the sender and forward the message to other people.

ReadMessages

Read a message.

Reply

While the user is reading a message, this will let the user reply to the message.

Forward

While the user is reading a message, this will let the user forward the message to other people.

**WriteMessagePage:** This page will allow the user to write a new message.



WriteMessage

Users can write a new message.

