## Team: JMHS\_ComputerScienceClub1

Hardware Requirements:

* Memory: at least one GB of RAM
* Disk Space: At least three MB of disk space
* Active internet connection- uses google api; map does not load without it.

Software Requirements

**Software Requirements**

1. The solution shall handle multiple simultaneous GPS tracked packages sending updates.

The solution is able to track more than one package at a time, as well as display info-boxes for more than one package marker. We tested our code first with a single package, then with multiple packages to display needed information.

2. The solution shall be easily accessible from a Windows 7 computer.

Our solution is accessible with Windows 7. We did all our coding and testing on two Windows 7 computers.

3. The solution shall support an admin mode that shows all package location updates on a map.

Our solution has an admin ID: 1 Password: 1

We used our admin account to do most of our testing. Our packages showed locations and updates accurately under the admin account

4. The solution shall support a user mode that shows a subset of package location updates on a map.

We have a user mode that can work with single and multiple UUID’s. We conducted our tests after making the program work with our admin account.

5. The solution shall accept a list of UUIDs in user mode to control the subset of package location updates displayed on the map.

Our solution can accept a list of UUID’s in user mode to control the subset of packages. We have made tests with our program to make single and multiple package updates to display on the map.

6. The solution shall accept name, destination, and GPS unit UUID information as HTTP query parameters on a HTTP GET of the URL path "/tracknewpackage". An example follows: GET http://127.0.0.1:8080/tracknewpackage?name=Some+Name+Here&destinationLat=42.4877185&destinationLon=-71.8249125&uuid=b0f9bb21-160f-4089-ad1c-56ae8b2d5c93

Our solution accepts name, destination, and GPS unit UUID information as HTTP query parameters on a HTTP GET of the URL path “/tracknewpackage” We tested this by displaying them on the info-boxes.

7. The solution shall respond with a JSON encoded body which includes the registered uuid on an HTTP GET of the URL path "/tracknewpackage". An example follows: GET Response Body: { "ackUUID":"[b0f9bb21-160f-4089-ad1c-56ae8b2d5c93]" }

The solution responds with a JSON encoded body which includes the registered UUID on an HTTP GET of the URL path "/tracknewpackage" While testing, we ran status updates on the left of the webpage to print our UUID’s

8. The solution shall accept a JSON encoded body which includes location, elevation, and time on a HTTP POST to the URL path "/packagetrackupdate/". An example follows: POST http://127.0.0.1:8080/packagetrackupdate/b0f9bb21-160f-4089-ad1c-56ae8b2d5c93 POST Body: {"lat":"42.4879714","lon":"-71.8250924","ele":"195.9","time":"2015-12-08T08:42:33.188-05:00"}

Our solution accepts a JSON encoded body which includes location, elevation, and time on a HTTP POST to the URL path "/packagetrackupdate/" We tested this by printing the data in the js console.

9. The solution shall accept a JSON encoded body which includes a delivered flag on a HTTP POST to the URL path "/packagetrackupdate/". An example follows: POST http://127.0.0.1:8080/packagetrackupdate/b0f9bb21-160f-4089-ad1c-56ae8b2d5c93 POST Body: {"delivered":"true"}

Our solution accepts a JSON encoded body which includes a delivered flag on a HTTP POST to the URL path "/packagetrackupdate/" We tested this by printing the status on the js console.

10. The solution shall calculate and display distance to destination.

Our solution accurately calculates and displays the distance to destination in miles. We tested this by printing the distance on the info-box.

11. The solution shall calculate and display estimated arrival time.

Our solution calculates and displays arrival time in info-boxes displayed by clicking on the markers. We were unable to test the accuracy of these calculations.