**SOFTWARE REQUIREMENT SPECIFICATIONS**

for

“IoT Enabled Integrated Intelligence System for Automatic Pothole Detection, Pollution Monitoring, Post Accident Medical Response and Breakdown Assistance”

1. **Product Overview**

At the completion of the project, we aim to deliver an automated, integrated system on vehicles, using IoT, that has the following features:

*Pothole Detection system:*

The vehicle reports potholes in the background as the vehicle normally drives around. The authorities then have a prioritized list of potholes with locations, which makes it easy for them to fix it. Since this data is available to citizens as well, they can take the authorized to action if they don’t act.

*Pollution monitoring & control system:*

Pollution has become a menace that will certainly cause big problems in the future if we don’t act now, like in the case of Delhi, Beijing. This system continuously reports the pollution levels to the respective authorities in real time and if violations are found, automatic penalties can be imposed.

*Emergency medical help in accident:*

This system automatically notifies nearby hospitals and ambulances if the vehicle has been in a serious accident.

*Accident reporting to police authorities:*

When such serious accidents occur, that may involve death, police find it hard to prove the accused guilty. Hence, if the vehicles involved, their location and a timestamp are recorded and sent to the police in real-time, it will prove to be evidence.

*Vehicle breakdown and servicing:*

This system make servicing very efficient and automatic. It also automates breakdown assistance.

To achieve this, we use a fully functional Raspberry Pi Module equipped with various sensors and a 3G WiFi Dongle. An AWS cloud infrastructure is deployed to gather data from the Pi, perform computations and take respective actions, as described in sections ahead. A robust and efficient web interface is provided for the various organizations to keep track of the information and take action in real time. An automatic SMS and email generation system which notifies the respective actors is also incorporated.

1. **External Interface Requirements**

**User Interfaces**

The user interfaces in our application are all web interfaces. For the Pothole Detection application, we have a web interface for the respective civil development authority, from which they can login and find various options in a dashboard like check status of potholes, existing potholes, contact respective corporates, etc. They can send emails and SMSs directly from the web interface to their employees. For example, to alert the BBMP employee of Whitefield to fix a pothole on a road. Citizens can also login to this web interface through their Google accounts, and check reported potholes and their status.

Next for the Pollution Monitoring system, the traffic or pollution control authorities can see the vehicles that have been reporting higher pollution and levels, and can choose to impose penalties on them. Vehicle owners can also login to check if their vehicles have been penalized. Once a penalty is imposed, an email and an SMS is automatically sent to the owner.

In the Emergency Medical Response system, we will be providing a web interface for hospitals to dispatch their ambulances. Nearby ambulances, will be alerted through SMS to the driver.

For the police to get information about vehicles that have been involved in a serious accident, we will be providing a web interface through which they can get accurate information about which vehicles were involved, at what exact location and the exact cyber timestamp, so that they have hard evidence in accidents that involve death or otherwise.

In the automated servicing and breakdown assistance of vehicles, there are web interfaces for the vehicle owner, and for the service center staff. Automated SMS/Emails are sent to the owners and staff for various necessities.

It will be difficult for the above organizations to create each of these functionalities, so we will be providing REST APIs for them to easily integrate their existing systems with our applications, with few modifications.

**Hardware Interfaces**

* Raspberry Pi 2 is the Soc.
* Various sensors and LEDs for I/O.
* Internet Modem for real time communication.

**Software Interfaces**

* Raspbian Operating System for the SOC.
* Ubuntu 14.04 OS as the Server Operating System.
* Apache httpd 2.2.31 for server program.
* MySQL Database for storing data.
* Adafruit libraries for making http requests and responses.
* Twitter bootstrap 3 for efficient web page creation tools.

**Communication Interfaces**

i. *Web* Interface

The application will be accessed over the Internet. All features will accessible through the web site.

ii. *Emails*

Automated emails will be sent by the server to the customers as well as the service centre representatives.

iii. SMS

As a part of the next phase of our project we also plan to implement an automated Short Message Service to customers about warnings, notifications of their vehicles along with recommended action.

1. **Functional Requirements**

**3.1 Automated breakdown assistance**

This function is used by the server to automatically identify vehicles that have broken down. It sends notifications to the nearest service center, about the break down location and the parameters at the time of the breakdown. The service center can then dispatch help immediately. All through this, the driver is kept informed about the activities through automated SMS.

*Input:* *The vehicle when it reports its parameters sends a status update whether it is running or not.*

*Function:*

*3.1.1 Breakdown identification:*

This function would identify the vehicles which are broken down.

If the car life status is of failure, the server identifies it automatically, obtains its location and the parameters recorded when it failed and sends it to the nearest service center.

*3.1.2 Report updates:*

This function from the server is responsible to keep the driver informed about the actions taken by the service center.

*Output:*

It is an automated message service, as the service representative follows through on the notification.

**3.2 Pothole detection**

This function keeps analyzing the values of the accelerometer fitted on the vehicle and monitors for any marked deviations from the regular readings, which could correspond to a pothole.

*Input:*

The changes in acceleration in z axis, from the accelerometer readings.

*Function:*

This module will analyze the data from the accelerometer and compare it to the patterns of potholes and speed breakers to see if the reading match. The patterns or thresholds that are being compared to, are set after extensive experimentation.

*3.2.1 Elimination of speed breakers*

As the various values of the accelerometer are collected and analyzed, in the event of a value that crosses the set threshold, there is a test made to see if the deviation in regular values is actually a speed breaker. This test can be made based on the difference in pattern of reading of pothole and speed breakers.

*Output:*

In the case of a pothole, the function will send the data (time, location, accelerometer reading) to the authorities’ servers, for them to prioritize.

**3.3 Pollution Monitoring Mechanism at Vehicle Level:**

The SoC monitors exhaust gas emissions through a series of gas sensors placed on the automobile. Gases like Carbon Mono-oxide (CO), Oxides of Nitrogen (NOx) etc. are emitted by automobiles and are harmful to humans. The values from the sensors are relayed to the authorities’ servers periodically.

*Input:*

Values of gaseous pollutant levels from various sensors, i.e CO2 sensors, N0x sensors etc.

*Function:*

These values are compared with the safe levels of pollutants that have been defined by the authorities and if there is an unusual amount of harmful emissions detected, the user is alerted to perform maintenance.

*Output:*

Along with the values of gases, the past history of the vehicle regarding pollutant levels is also kept in the database, so that if the required inspection and maintenance is not done by the owner, the automobile details such as registration number are reported to the pollution control authorities via email generated by the server. This functionality helps the user and local authorities to curb pollution created by automobiles.

**3.4****Automated Dispatch of Ambulances in Major Impact Collisions:**

In this integrated approach, we use an accelerometer to detect collisions that could have caused grievous injuries.

*Input:*

The acceleration values in the x axis, by the accelerometer.

*Function:*

Once the sensor collects values of a major spike in acceleration caused by the abrupt stop due to the collision, we record the location coordinates from the GPS module on-board. The location coordinates of this collision is sent to the nearest hospital through its registered mobile number and email address, which can dispatch its ambulance. Then the SMS/Email functionality is called.

*3.4.1 SMS/Email*

An SMS and Email is sent to the victim (in case he is conscious, or the first responders can be informed)/victim’s family. In the SMS/Email it will contain the current location and time it takes from the location of the hospital to the accident location in present traffic conditions. We use APIs to convert the coordinates to an address and also find the distance and time to reach the spot.

**3.5****Automated Collection of Accident Related Information for Evidence:**

With this function, the moment there is an impact recorded from the impact sensor, it reports the location coordinates, the cyber timestamp and other details of the vehicle like the registration number to the Police authority, which substantially proves and also automates the fact that an accident occurred at a location, and the exact vehicle that was involved in the accident.

*Input:*

Changes in acceleration measured by the accelerometer.

*Function:*

The spike in the acceleration value will be significantly higher than normal changes in acceleration during a collision. On detection of a collision, the location, timestamp, registration number etc. are sent to the police station. Based on the similar location and timestamps the various vehicles that may have been involved in the accident can be identified.

*Output:*

The location, timestamp, registration number and similar details is sent to the authorities’ servers.

**3.6 Vehicle condition reporting:**

This function is to report to the driver, the condition of the vehicle. The vehicle is given this information from the server based on the data it has about other vehicles. If in case there is imminent failure, it is reported to the driver as well as the service station.

*Input:*

Regular data from the vehicle about the temperature of car’s engine.

*Function:*

*3.6.1 Send data to server:*

The SoC then calls upon another function to make a HTTP post request to the server with the attributes as post parameters.

If the request fails, it stores the data and tries again after 5 seconds.

*3.6.2 Receive data and display to user*

Once the response is obtained from the server, the SoC runs a function to check the returned values and what it means.

If it’s a warning of failure, it displays a red LED to the driver.

*Output:*

In case of potential failure, the user is notified through car’s hardware or through mail/SMS.

**3.7 Service center notification:**

This function runs on the server continuously, finding vehicles which have reported warnings. The service center representative can then see the vehicles which are reporting warnings and can prepare the parts required to repair the vehicle, if the driver confirms the repair.

*Input:*

The vehicles keep reporting their data.

*Function:*

The server checks with available data sets to find if its parameters indicate potential failure.

If it is the case, it sends a warning to the driver and also stores the pertinent data on the database at the back end. It sends a notification to nearest few service centers automatically.

*3.7.1 Obtain confirmation:*

This function is used to automatically obtain a confirmation from the driver if he plans to repair the vehicle at the suggested service center, by replying to an SMS that is sent to him. This SMS service is automated.

*Output:*

A warning is sent to the driver and also data is stored in the database for future analysis. Notification of impending failure of vehicle is sent to the service centers.

1. **Software System Attributes**

**a. Reliability**

The system must carry out complete data transactions. Mismatched entries should be corrected and the database must always be in a consistent state.

**b. Availability**

The system must be available for use when it required. Server downtime is not acceptable.

**c. Security**

The system authenticate users before giving access to secure information on the server.

**d. Portability**

The software is made to run on the Raspbian OS which would cover all versions of the Raspberry Pi.

**e. Maintainability**

The code needs to be documented well and designed such that changes in certain areas of the scripts do not cause any dependency issues.

**f. Performance**

The server would have to respond with a status to the vehicle at the maximum within 2 seconds of obtaining the request.

1. **Performance Requirements**

**Internet connection**:

Since the vehicles will be in transit most of the time, the system should be able to send data using a mobile internet connection at speeds as slow as 56Kbps. The response time from the server should be less than 500ms for the sensors to function properly.

**Notification service**:

The notifications via SMS and email would have to be almost instantaneous with only a small delay of 1 minute or two. Automatic notifications would have to reach the service center immediately and no downtime is acceptable.

1. **Database Requirement**

***Vehicle\_owner:***

|  |  |  |
| --- | --- | --- |
| Attribute Name | Attribute Type | Attribute Size |
| FirstName | String | 30 |
| LastName | String | 30 |
| Address | String | 50 |
| City | String | 30 |
| State | String | 2 |
| RegistrationNo | int | 10 |

***Breakdown\_status:***

|  |  |  |
| --- | --- | --- |
| Attribute Name | Attribute Type | Attribute Size |
| RegistrationNo | Int | 10 |
| Servicecentre | String | 30 |
| Location | String | 50 |

***All\_car\_status:***

|  |  |  |
| --- | --- | --- |
| Attribute Name | Attribute Type | Attribute Size |
| Model | String | 30 |
| RegistrationNo | Int | 10 |
| Status | String | 50 |
| Location | String | 30 |
| Temprature | Double | 10 |
| Humidity | Int | 2 |
| IPAddress | String | 10 |
| Updatetime | String | 2 |
| Temp | int | 10 |

***Pothole\_status:***

|  |  |  |
| --- | --- | --- |
| Attribute Name | Attribute Type | Attribute Size |
| Pothole ID | Int | 2 |
| Location | String | 10 |
| Status | String | 10 |
| Updatetime | String | 2 |

***Bbmp\_account:***

|  |  |  |
| --- | --- | --- |
| Attribute Name | Attribute Type | Attribute Size |
| UserID | String | 10 |
| Password | String | 10 |
| Name | String | 10 |
| Email | String | 10 |
| Phone | String | 10 |

***Service\_centers:***

|  |  |  |
| --- | --- | --- |
| Attribute Name | Attribute Type | Attribute Size |
| Center Code | String | 10 |
| Center Name | String | 10 |
| Latitude | String | 10 |
| Longitude | String | 10 |

1. **Design Constraints**

The software on the SoC would have limited hardware capabilities. So it should be designed to work on low processing capabilities as well.

On the backend, the server would have to process a large amount of requests, and hence scripts on the back end would have to be light weight and should be able to execute in a short amount of time.

1. **Other Requirements**

* Secure access of confidential data (user’s details).
* 24 X 7 availability.
* Better component design to get better performance at peak time.