**Software Requirements Specification**

**for**

**Vehicle Fitness Automation**

**Version 1.0**

**Prepared by Sanjana Reddy**

**Nishchal J**

**Navya Priya N**

**Date created:1-11-2020**

**Table of Contents**

**1.Introduction**

**2.Exisitng system**

**3.User requirements**

**4.System requirements**

**5.Functional requirements**

**6.External interface requirements**

**7.Non functional requirements**

**User manual**

**1. Introduction**

All commercial vehicles are mandated to undergo fitness tests periodically, depending on vehicle age. This is done to check and monitor the pollution which is produced by the release of smoke from your motor vehicles, to maintain and continue the motor vehicle in good and appropriate condition and to avoid road accidents which might occur due to the improper condition of the vehicle. The goal of this project is to automate the various steps involved in vehicle fitness tests.

**2. Existing System**

RTO offices often face an issue of long queues. This causes a lot of inconvenience to people and the entire process becomes very time consuming. The vehicles are to be brought to designated fitness centres for inspection and certification. There are normally long queues and the process is troublesome since there are many applications but few officers to handle them. Also, not much work has been done to automate these tests.

**3. User Requirements**

* To provide a user friendly environment for the user to easily upload the required documents for verification.
* To make the work of RTO officers lesser and so reduce the requirement of manpower .
* The application should allow access to authorized users (RTO officers) for performing the automated tests.
* The tests to be automated may include the Visual inspection of car parts, tyre fitness, Vehicle plate recognition etc..
* To securely maintain the data in the database for future references which is also secure.

**4. System Requirements**

The **hardware requirements** for the project are as below:

* Processor: Pentium® Dual Core CPU or Higher versions
* Hard Disk: Minimum 1 TB
* Memory: Minimum 4GB RAM

The **Software Requirements**: The package is developed using HTML,CSS and Flask for front end. IBM Cloud as the backend to store the database for the web application.

* Operating System: WINDOWS 8 or higher, Android 5.0 or higher
* Language: Python, HTML, CSS
* NoSQL Database: IBM Cloud/ MongoDB
* Version Control System: GitHub
* Softwares used for ML Models : Tensorflow, PyTorch, OpenCV, Keras

**System Specification:**

* To build a system that can receive input and generate automated test results.
* To store properly the documents of each applicant in order to maintain their security and to make verification easier.
* To provide an interactive UI for easy usage for RTO Officers to upload documents, test inputs and easily recover test outputs.
* Finally a Form filled with all the tests performed for the vehicle with their results generated from automation methods.
* List out the manual tests to be performed yet.

**5. Functional requirements**

The functional requirements of the project include:

**1. Authentication**

The admin(RTO officer) can perform special functions .They get their credentials authenticated in the form of a username and password. Each applicant has separate credentials for authentication. He/She is given the privilege of uploading required documents. On invalid credentials, applicants cannot proceed with the test, unless an admin will address the issue.

* Transaction logs must be kept for each service provided.
* The user’s password must never be exposed to compromise.
* User session logs stored for usability and other research must be anonymous.

**2. Uploading the required files for verification**

* A separate user interface for the applicant to upload vehicle and personal documents for verification via OCR technology.
* The VAHAN API is used for the verification process of the uploaded documents.

**3.** **Automated Vehicle Testing**

* Vehicle Parts Detection using Multiclass Image Classification Deep learning models. The detected parts should include headlights, Tyres, Mirrors, etc.. as per rule 62, Vehicle Fitness.
* The indicator blinking test and Horn sound test is to be automated. This can be done using Euclidean distance, Module difference etc..
* Additionally, an optional Break test can be performed live on the road. This is to be automated using vehicle localisation, pixel per meter and triangular similarity calculations for braking distance.
* RTO officers should upload the images of the respective parts taken from the CCTV live footage and the automated results should be generated.

**4. Geolocation**

* Current location of the user is fetched from the geocoder library and compared with the location of the predefined location of the RTO.
* For location, latitude and longitude coordinates are used. The distance between them is calculated and returned in kilometers.
* The distance calculation is done using the haversine library.

**5. Generation of test results**

Finally, a Form is to be generated after consolidating the results of all the tests. The tests can be reviewed once as they will be accessible through the cloud storage. The same form is to be sent to the applicant for future reference and also saved into the cloud for future verification purposes.

Based on the form generated, the result of the Vehicle Fitness Test should be decided and the Fitness certificate should be issued.

**6.External Interface Requirements**

User interface is built using HTML, CSS and Javascript. The framework used is Flask.

* The user can access the portal by logging in. In the Log in page, the user should enter a username and password which will be validate

1. **Login for document upload**

* **Upload documents**: The user interface has two upload buttons where they should upload the Pollution Under Control Certificate and Registration Certificate.
* The necessary details will be extracted from OCR and displayed on the frontend.
* The uploaded documents also get stored in the database.
* These extracted details can be used to validate against the data fetched from the VAHAN API.

1. **Login for automated visual inspection**

* The images of the various car parts should be uploaded to get the respective.
* **License plate detection**- Image of the front view of the car should be uploaded and the model will validate the presence of a license plate and display the same on the frontend.
* **Tyre condition detection**- Image of the tyre should be uploaded and the model returns back the condition of the tyre. The results are one among the 6 predefined classes.
* **Working of the headlights**- The image of the frontview should be uploaded and the model will return whether the headlights are on or off. Hence the user should upload an image with headlights on to make sure the model returns a confirmation of the same.
* **Geolocation**- On button click the current location of the user is fetched from the geocoder library and compared with the location of the predefined location of the RTO. For location, latitude and longitude coordinates are used. The distance between them is calculated and returned in kilometers.
* **Form**- Once all tests are done, a form will be created which will consolidate results of all the tests in one place. The results of all the automated tests will be pre-entered. The results of the manual tests can be entered in the form and submitted.

**7. Non functional requirements**

* **Performance**- The response time should be quite low which will make it user friendly.The throughput should be quite high, so that a lot of results can be retrieved within no time.
* **Usability**- the software is user friendly and hence a lot of prior knowledge is not required in order to use it.
* Every unsuccessful attempt by a user to access an item of data shall be **logged** for further assessment.
* The application should be capable enough to **handle a million users** without affecting its performance.
* The software should be **portable**. So moving from one Android version to another does not create any problem.
* **Security**- The details of the users shouldn’t be available to the public, only the admin will be able to access it . Also, authentication and authorization makes sure that only the concerned person uses their account.
* **Recoverability**- Since, most of the data is stored in the database, the data can be easily recovered using various methods.
* **Manageability**- Since there is an admin , if anything goes wrong, they can manage and control the entire software.
* **Scalability**-Since all the models used are light weight , the requirements needed to set up the environment is minimal

**User Manual**

**Applicant-**

* Creates an account in the website and hence has credentials
* Gets an application number and time for the test
* On the day of the test, uploads the required documents into the application in the RTO
* Car is tested for various parameters which are mostly automated
* Final report is generated which gives the overview of the results
* This form can be used for future reference

**RTO officer-**

* Has an account with certain credentials
* Once the applicant arrives, the live CCTV footage of the vehicle is captured and the automated tests are done.
* Further the tests which cannot be automated are done by the officer
* The final report generated is reviewed and the time stamp and digital signature of the officer is put.