Onspot - An AI Powered Secured Utility Complaint System using Cloud Computing

Project report submitted

in partial fulfillment of the requirement for the degree of

Bachelor of Technology

By

Kuluru Vineeth Kumar Reddy **REG NO:18BCS043**

Karthick P S **REG NO:18BCS038**

LaxmiNarayana K **REG NO:18BCS037**

Gorla Jaganmohan Reddy **REG NO:18BCS029**

Under the guidance of

Dr. Malay Kumar

DEPARTMENT OF COMPUTER SCIENCE

IIIT, Dharwad

Feb-2022

CERTIFICATE

It is certified that the work contained in the project report titled **An AI Powered Secured Utility Complaint System using Cloud Computing** by Kuluru Vineeth Kumar Reddy (18BCS043), Karthick P S (18BCS038), Laxminarayana K (18BCS037) and Gorla Jaganmohan Reddy (18BCS029) has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Supervisor(s)

Dr. Malay Kumar,

Dept. of Computer Science and Engineering

April, 2022

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project report entitled "**An AI Powered Secured Utility Complaint System using Cloud Computing**" in partial fulfillment of the requirement for the award of the Degree of Bachelor of Technology and submitted to the Department of Computer Science of Indian Institute of Information Technology Dharwad, is an authentic record of our own work carried out during a period from January to April 2022 under the supervision of Dr. Malay Kumar, Department of Computer Science and Technology, Indian Institute of Information Technology, Dharwad. The matter presented in this report has not been submitted by us for the award of any other degree of this or any other Institute/University.

Kuluru Vineeth Kumar Reddy (18BCS043)

Karthick P S (18BCS038)

Laxminarayana K (18BCS037)

Gorla Jaganmohan Reddy (18BCS029)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 20/04/2022 Dr. Malay Kumar

APPROVAL SHEET

This project report entitled **An AI Powered Secured Utility Complaint System using Cloud Computing** by Kuluru Vineeeth Kumar Reddy, Karthick P S, Laxminarayana K and Gorla Jaganmohan Reddy is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Supervisor (s)

Dr. Malay Kumar

Head of Department

Dr. Uma Seshadri

Examiners

Dr. Malay Kumar

Date : 20/04/2022

Place: Dharwad

ACKNOWLEDGEMENTS

It is indeed a great pleasure to express our sincere thanks to our supervisor Dr. Malay Kumar, Department of Computer Science and Engineering, IIIT Dharwad for her continuous support in this project. He was always there to listen, advise and share her expertise with us at every stage of the project development . He showed us different ways to approach a real world problem and the need to be persistent to accomplish any goal. He had confidence in us when we had doubted ourselves, and brought out the good ideas in us. He was always there to meet and talk about our ideas, share her expertise and views on developing a solid prototype, and to ask us good questions to help us think through our problems. Without his encouragement and constant guidance, we could not have completed this project on time.

We are also indebted to our own team members for their rigorous efforts in questioning the most difficult edge cases and extracting the best out of it and also for their persistent coordination till the end. Without their support and cooperation, this project could not have been finished.

Kuluru Vineeth Kumar Reddy

Karthick P S

Laxminarayana K

Gorla Jaganmohan Reddy

Date: 20/04/2022

ABSTRACT

The main objective of this project is to solve the existing problem of lack of communication between Citizens and Authorities towards road maintenance. The problem is that both the parties are in a perception that the problem is with the other either due to lack of funding from the govt or due to the citizens not actively participating in reporting the potholes to the authorities.

We have come up with a web based solution that aims at generating complaints or reports, where a specific user (citizen) can capture the pic of an area, which will be fed to a Deep learning model that has the ability to geocode, validate and track the potholes in that specific region captured. The feature has been achieved by training a YOLO model for object tracking on multiple images as well as videos. The model uses a convolutional neural network. Users would have the amenity to view the damage on the roads using this application. A dynamic complaint to the authorities in the form of a report is generated which is shared with the closest authority(within 5000m radius) who can view and update this report. This has been achieved using a MySQL function called st_distance_sphere() which returns the radius or distance between 2 points feeded as parameters to it. They can also view the reports generated by users within their municipal locality. An additional feature of filtering based on filters like search bar,etc are also enabled.

From the implementation of this project, we have been able to address the problem of lack of communication, collaboration and coordination both from the Citizens as well as Authority perspectives to uphold and maintain the infrastructure of our Roads. Also proper transparency on the actual road maintenance and pothole complaint and repair features have been achieved.

CONTENTS

Chapter-1

Introduction

A major problem being faced by municipalities around the world is maintaining the condition of roads be it summer, the monsoons (when it is at its worst) or any weather condition as a matter of fact. And although it's the responsibility of the authorities to make sure the roads are free of damage, at times they overlook the problem, and most times don't even know that the problem exists. According to indiatoday, the number of deaths over the past **3 years stands at 9300 and with over 25000 injuries.**

Maintaining Indian roads in a good condition is a challenge that too with continuous weather changes, wear and tear, Not considerable amount of money allocated for the municipalities. Also, it is most important to keep people informed. This is an application developed after thorough research and analysis for solving the above. **The web application aims at generating complaints or reports, where a specific user (citizen) can capture the pic of an area, which will be fed to a Deep learning model that has the ability to geocode, validate and track the potholes in that specific region captured. The feature has been achieved by training a YOLO v2 model for object tracking on multiple images as well as videos. The model uses a convolutional neural network. Users would have the amenity to view the damage on the roads using this application. A dynamic complaint to the authorities in the form of a report is generated which is shared with the closest authority(within 5000m radius) who can view and update this report. They can also view the reports generated by users within their municipal locality. An additional feature of filtering based on filters like search bar,etc are also enabled.**

Keeping the roads in good condition along with tracking damages is a challenge with constant changes in weather, low budgets for the municipalities. Not to forget keeping the people informed is a task. This project was aimed at solving the challenges mentioned above and was aimed at solving the challenges mentioned above by creating an interactive environment between the municipality and any person who identifies a pothole in a dynamic environment.

In articles covered by Guardian News & Media potholes took a deadly toll in 2017, claiming almost 10 lives daily. IndiaTimes stated that "Bereaved Father Mr. Dadarao" filled 600 Potholes in Mumbai in memory of the son he lost in a road accident! In shorts reported, potholes killed more people than terrorists reporting 14,926 deaths in road accidents.

Chapter-2

Literature Survey

2.1 The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring

- This paper was used to investigate an application of mobile sensing: detecting and reporting the surface conditions of roads. It describes a system and associated algorithms to monitor this important civil infrastructure using a collection of sensor-equipped vehicles.

- This system,which is called the Pothole Patrol (P2), uses the inherent mobility of the participating vehicles, opportunistically gathering data from vibration and GPS sensors, and processing the data to assess road surface conditions.

- It was deployed on 7 taxis running in the Boston area. Using a simple machine-learning approach, It is shown that we are able to identify potholes and other severe road surface anomalies from accelerometer data. Via careful selection of training data and signal features, a detector was built that misidentifies good road segments as having potholes less than 0.2% of the time.

- It was then evaluated on system data from thousands of kilometers of taxi drives, and showed that it can successfully detect a number of real potholes in and around the Boston area. After Clustering to further reduce spurious detections, manual inspection of reported potholes shows that over 90% contain road anomalies in need of repair.

2.2 Lightweight Road Manager: Smartphone-based Automatic Determination of Road Damage Status by Deep Neural Network

- Citizens in various locations can report local infrastructure issues to the government by posting reports on certain websites, such as Chiba Report and FixMyStreet. Recently, these systems have begun operating worldwide. In these systems, a large volume of information is collected on infrastructure problems that are identified by citizens (e.g., broken paving slabs, fly tipping, graffiti, potholes). This information is expected to be utilized for infrastructure maintenance.

- However, local problems (especially road defection) identified by citizens are sometimes not deemed an urgent matter for road managers. This is because it is difficult for an average person to determine road damage status. Furthermore, non-critical reports may be a burden for local governments because each report requires visual confirmation. Therefore a smartphone application is proposed based on a deep neural network that can determine road damage status using only photographs of the road.

- This application is based on a deep neural network model trained by citizen reports and road manager inspection results, which are gathered daily on a government server. The application updates the model parameters each time it launches and thereby becomes increasingly more intelligent and effective.

- The proposed system enables average citizens to easily determine road damage status using only a smartphone application. In addition, because not only expert road managers but also local government officials without expert knowledge can inspect the road, the proposed system can be useful for local governments that lack expert road managers.

2.3 Damage Detector: The Damage Automatic Detection of Compartment Lines Using A Public Vehicle and A Camera

- In Japan, there are 1.2 million kilometers or more of the road, of which 900,000 km of road has been paved. Paved road has been painted with a compartment line to separate many vehicles and driver-pedestrians.

- Compartment line is important for drivers and pedestrians because they can be aware of the width of the roadway and the sidewalk by the compartment line. Therefore, municipalities should inspect the ongoing compartment line but they are inspected by people so the burden for road management is big.

- This research was focussed on automatically detecting the damage of the compartment line. The image obtained from cameras, applying random forest. In this experiment, more than about 50,000 of images were captured by a Walking camera and a vehicle-mounted camera.

- With baseline and precision comparison results, the detection of damage to compartment lines, it was confirmed that the proposed method is effective.

2.4 Potholes and bad road conditions: mining Twitter to extract information on killer roads

- Research shows that Twitter is being used as a platform to not only share and disseminate the information but also collect complaints from citizens. However, due to the presence of high volume and large stream data, real-time manual identification of those complaints is overwhelmingly impractical.

- This paper identifies the complaints and grievances posted on bad road conditions causing life risks, discomfort and poor road experience to the citizens. It formulates the problem of killer road complaints identification as a multiclass text classification problem. It also addresses the challenge of keyword based flagging methods and identifies several linguistic features that are unique for the killer road complaints tweets such as the issue reported in the complaint, pinpoint location of the issue, city or region location information.

- The results reveal that not all complaint reports posted to Public agencies contain sufficient information and hence are not useful. Therefore, it further proposes a mechanism to enrich the nearly-useful tweets and convert them into useful reports. It also presents the results using information visualization and gains actionable insights from them.

- The results also show that the proposed features are discriminatory and able to classify killer roads complaints with an accuracy of 67% and a recall of 65%.

2.5 An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with Steerable Filter-Based Feature Extraction

- This study establishes an artificial intelligence (AI) model for detecting potholes on asphalt pavement surfaces. Image processing methods including Gaussian filter, steerable filter, and integral projection are utilized for extracting features from digital images.

- A data set consisting of 200 image samples had been collected to train and validate the predictive performance of two machine learning algorithms including the least squares support vector machine (LS-SVM) and the artificial neural network (ANN).

- Experimental results obtained from a repeated subsampling process with 20 runs showed that both LS-SVM and ANN were capable methods for pothole detection with classification accuracy rate larger than 85%.

- In addition, the LS-SVM achieved the highest classification accuracy rate (roughly 89%) and the area under the curve (0.96). Accordingly, the proposed AI approach used with LS-SVM could be very potential to assist transportation agencies and road inspectors in the task of pavement pothole detection.

- Hence, this research establishes an automatic approach for asphalt pavement pothole detection. Image processing techniques including GF, SF, and IP are used synergistically to extract features from pavement digital images.

2.6 A Deep Learning-Based Approach for Road Pothole Detection in Timor Leste

- This research proposes a low-cost solution for detecting road pothole images by using convolutional neural networks (CNN). The model is trained entirely on the image which was collected from several different places and has variation such as in wet, dry and shady conditions.

- The experiment using the 500 testing images showed that our model can achieve (99.80 %) of Accuracy, Precision (100%), Recall (99.60%), and F-Measure (99.60%) simultaneously.

- The model was trained on 13,244 images which were collected from several different places with various conditions, size and shape. When tested on new images, the model achieved very high accuracy and experimental results showed Accuracy (99.80 %), Precision (100 %), Recall (99.60 %) and F1-Score (99.60%).

- It is demonstrated in the comparison study that the model significantly outperforms the conventional machine learning algorithm such as SVM.

- The result also proved that use of deep learning techniques can provide better solutions than traditional algorithms with many weaknesses in low level feature detection.

2.7 Asphalt Pavement Pothole Detection using Deep learning method based on YOLO Neural Network

- There is an increasing need for assessment of national road conditions. Currently, some automatic devices have been extensively applied to collect up-to-date data about road

conditions, such as the use of survey vehicles for collecting data which make it faster and more accessible, and semi automatic data processing that is useful for policy decision making.

- Yet, demand for more detailed road data is continuously growing. Thus, data improvement needs to be performed, upgrading the existing solution. To date, stages on identification and classification of road damages are being conducted semi manually based on images collected by survey vehicles.

- It is hindered due to the fact that this method is a cost-consuming process and may result in inconsistency. Therefore, this work used YOLO with three different architecture configurations, i.e Yolo v3, Yolo v3 Tiny, and Yolo v3 SPP, that enabled it to create a more accurate assessment for detecting potholes on the road surface.

- The results showed the average mAP values for Yolo v3, Yolo v3 Tiny, and Yolo v3 SPP at 83.43%, 79.33%, and 88.93%. While the area measurement shows the accuracy of 64.45%, 53.26%, and 72.10% respectively.

- And the model needs 0.04 second to detect each image. Conclusively, it shows a satisfying result in pothole detection.

2.8 Image-Based Pothole Detection System for ITS Service and Road Management System

- Potholes can generate damage such as flat tire and wheel damage, impact and damage of lower vehicles, vehicle collisions, and major accidents. Thus, accurately and quickly detecting

potholes is one of the important tasks for determining proper strategies in ITS (Intelligent Transportation System) service and road management systems.

- Several efforts have been made for developing a technology which can automatically detect and recognize potholes. In this study, a pothole detection method based on two dimensional (2D) images is proposed for improving the existing method and designing a pothole detection system to be applied to ITS service and road management systems.

- For experiments, 2D road images that were collected by a survey vehicle in Korea were used and the performance of the proposed method was compared with that of the existing method for several conditions such as road, recording, and brightness.

- The results are promising, and the information extracted using the proposed method can be used, not only in determining the preliminary maintenance for a road management system and in taking immediate action for their repair and maintenance, but also in providing alert information of potholes to drivers as one of ITS services.

- Regarding the experiment results, the proposed method reaches an overall accuracy of 73.5%, with 80.0% precision and 73.3% recall, which is a much better performance than that of the existing method having an overall accuracy of 45.1%, with 55.0% precision and 36.7% recall.

2.9 Automatic Detection and Notification of Potholes and Humps on Roads to Aid Drivers

- One of the major problems in developing countries is maintenance of roads. Well maintained roads contribute a major portion to the country's economy. Identification of pavement

distress such as potholes and humps not only helps drivers to avoid accidents or vehicle damages but also helps authorities to maintain roads.

- This paper discusses previous pothole detection methods that have been developed and proposes a cost effective solution to identify potholes and humps on roads and provide timely alerts to drivers to avoid accidents or vehicle damages.

- Ultrasonic sensors are used to identify potholes and humps and also to measure their depth and height respectively. The proposed system captures the geographical location coordinates of potholes and humps using a GPS receiver.

- The sensed-data includes pothole depth, height of hump and geographic location, which is stored in the database (cloud). This serves as a valuable source of information to the Government authorities and to vehicle drivers.

- An android application is used to alert drivers so that precautionary measures can be taken to evade accidents. Alerts are given in the form of flash messages with an audio beep. The proposed system considers the presence of potholes and humps. However, it does not consider the fact that potholes or humps get repaired by concerned authorities periodically.

2. 10 Pothole Detection Using Computer Vision and Learning

- Describes the techniques for identifying potholes on road surfaces aim at developing strategies for real-time or offline identification of potholes, to support real-time control of a

vehicle (for driver assistance or autonomous driving) or offline data collection for road maintenance.

- Research around the world has comprehensively explored strategies for the identification of potholes on roads. This paper starts with a brief review of the field and it classifies developed strategies into several categories.

- It then presents its contributions to this field by implementing strategies for automatic identification of potholes. It has developed and studied two techniques based on stereo-vision analysis of road environments ahead of the vehicle.

- It also tells about how it designed two models for deep-learning-based pothole detection. An experimental evaluation of those four designed methods is provided, and conclusions are drawn about particular benefits of these methods.

Chapter-3

Our Solution And Feasibility Study

- With this project one can now expect citizens to engage and take an active part in helping maintain the city infrastructure by using their mobiles to capture a pic of the nearby potholes and report it to the nearest municipality.

- An easy to use and sustainable solution is now available for government officials as well to manage and act on citizen issues with transparency and a streamlined process when it comes to reports related to potholes and maintaining city streets.

- Also for a person travelling in a highway, this is an extremely useful solution to avoid the upcoming potholes on his way by having a look at the google maps on where are the potholes present and what are their respective status.

- The features like Heat Mapping, Route Navigation would better help the citizens and authorities to keep a track of their area. Also with a deep learning model at the backend for validation of the potholes detected also facilitates automatic confirmation of authenticity which again reduces the manual workload.

- This project has the potential to solve the issue of pothole reporting, management and tracking with a lot of ease and with the addition of items mentioned in the future work, this statement will be further justified. Looking forward to making more such contributions one step at a time.

Chapter-4

Methodology / Project Flow Proposed

The overall solution is composed of designing the front and backend features for the authority as well as the user. The main features of the solution would include the below mentioned ones:

4.1 Citizen's Application

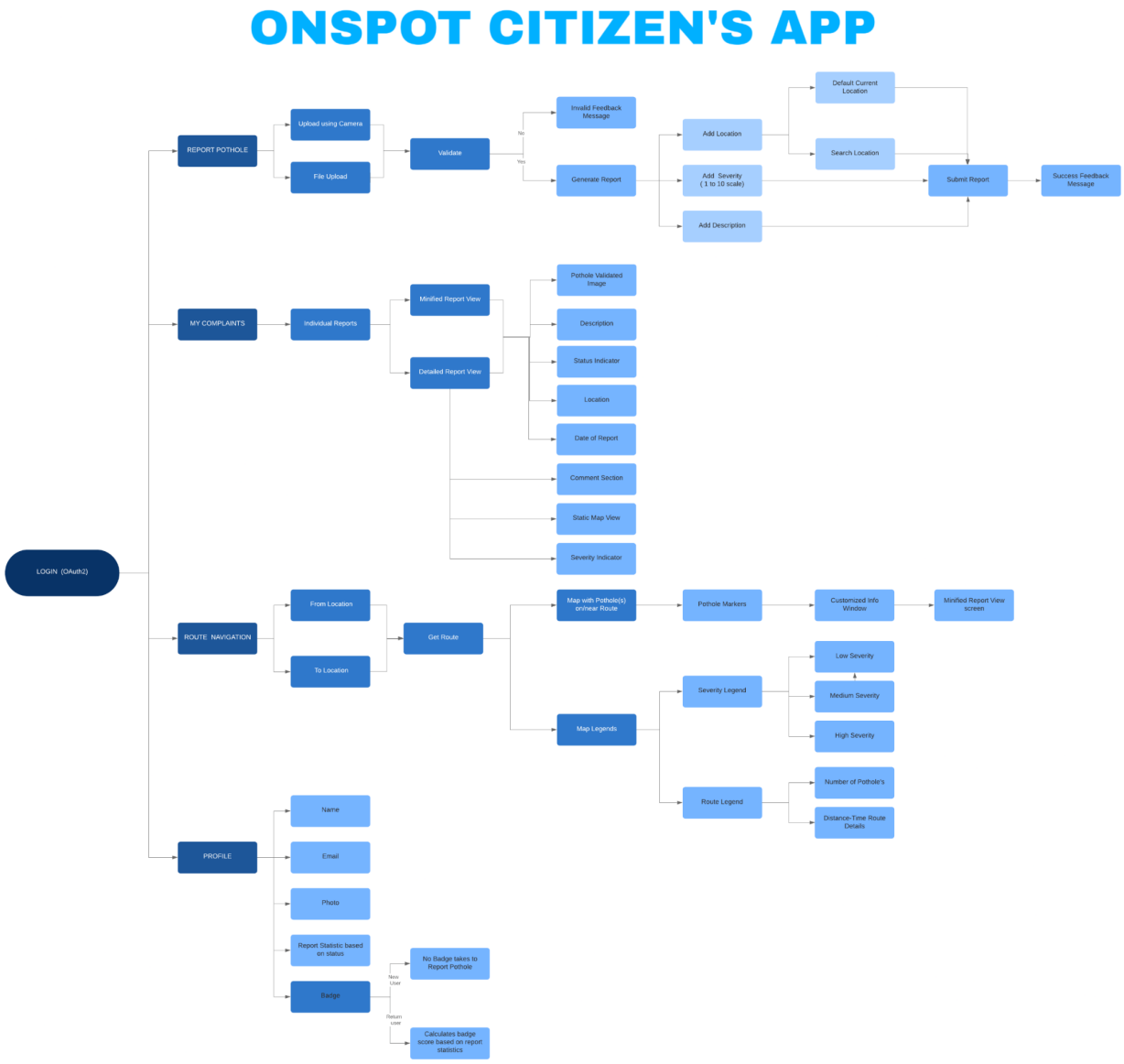


Fig 5.1

4.2 Authority's Application

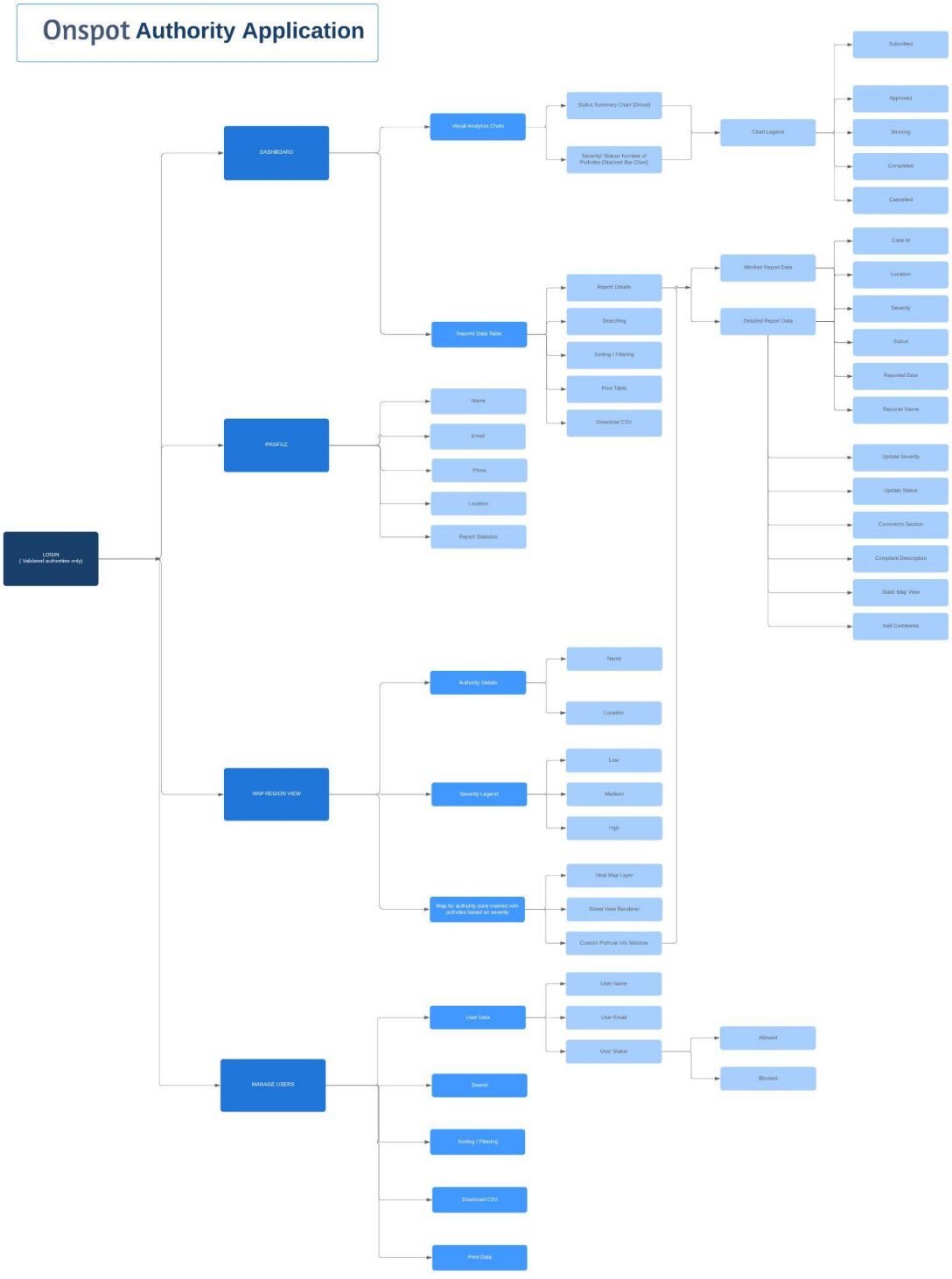

Onspot Authority Application
LOGIN
(Validated authorities only)
DASHBOARD
Visual Analytics Chats
Status Summary Chart (Donut)
Severity/ Status/ Number of Potholes (Stacked Bar Chart)
Chart Legend
Submitted
Approved
Working
Completed
Cancelled
Reports Data Table
Report Details
Searching
Sorting / Filtering
Print Table
Download CSV
Minified Report Data
Detailed Report Data
Case Id
Location
Severity
Status
Reported Date
Reporter Name
PROFILE
Name
Email
Photo
Location
Report Statistics
Update Severity
Update Status
Comments Section
Complaint Description
Static Map View
Add Comments
MAP REGION VIEW
Authority Details
Name
Location
Severity Legend
Low
Medium
High
Map for authority zone marked with potholes based on severity
Heat Map Layer
Street View Renderer
Custom Pothole Info Window
MANAGE USERS
User Data
User Name
User Email
User Status
Allowed
Blocked
Search
Sorting / Filtering
Download CSV
Print Data

Fig. 5.2

4.3 Core Backend Model

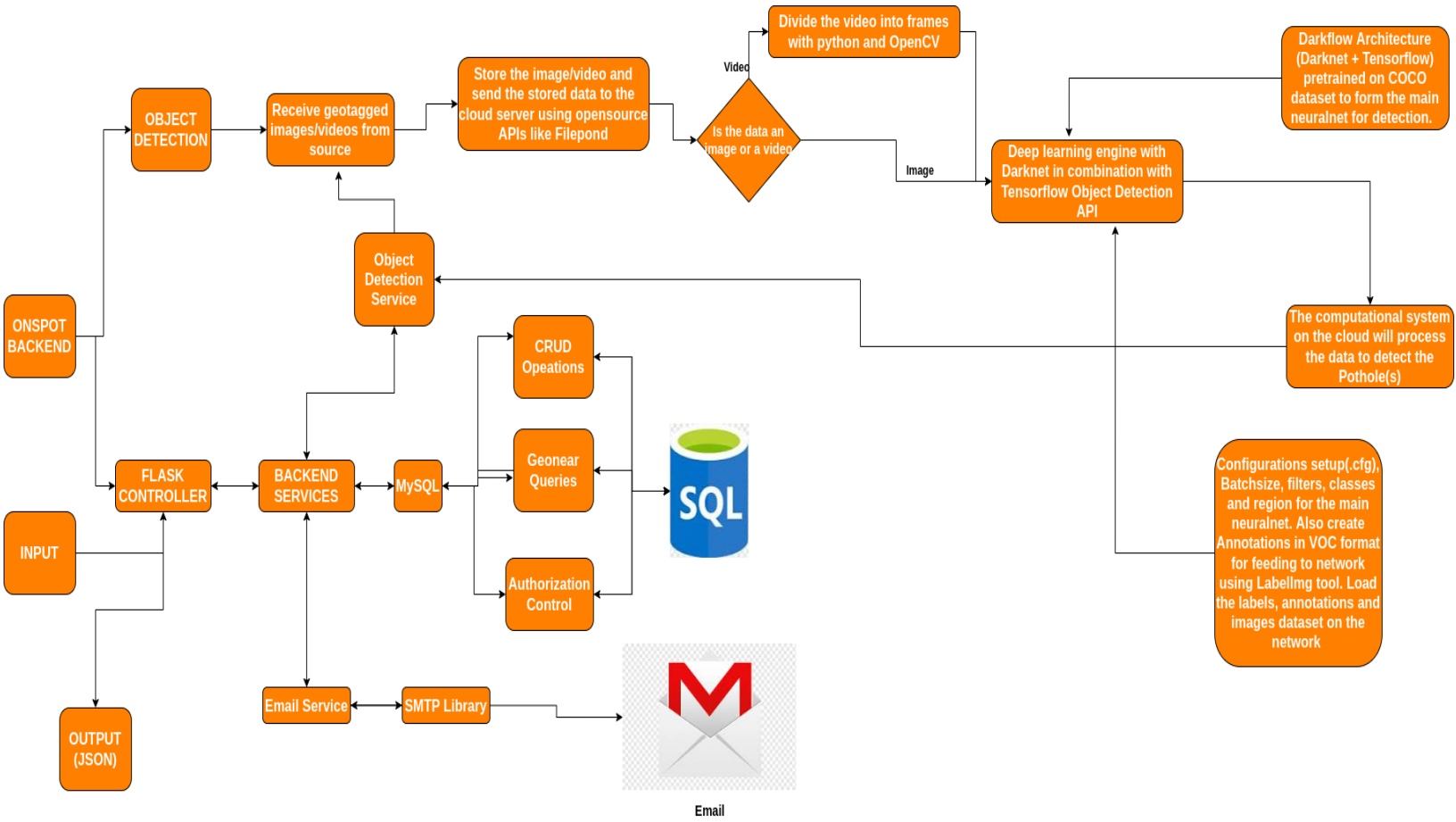


Fig 5.3

Chapter-5

Technology Stack For The Proposed Approach

The solution proposed would be a full stack application with frontend and backend components deployed on the cloud.

5.1 Frontend:

a. **Reactjs**: Used for **building user interfaces specifically for single-page applications**. It's used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components. The frontend part would be coded using reactjs.

b. **Html5**: It used to **create web pages**. It is also used to format text as titles and headings, to arrange graphics on a webpage, to link to different pages within a website, and to link to different websites from our application.

c. **Css3**: Used for styling our web application.

d. **Recharts**: It is a Redefined chart library built with React and D3. The main purpose of this library is to help you to write charts in React applications without any pain.

e. **google maps**: Used to enable both the user as well as the Authority for their communication and pothole detection.

f. **Npm**: Node package manager for installing dependencies for building the website.

g. **Filepond**: Tool to help upload large files such as videos or high quality images with ease.

h. **github pages**: We use this to deploy our frontend for anyone with the url to access the application.

i. **Nodejs**: Provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

j. **google apis(GAPIs)**: Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Here we use the GAPIs for authentication as well as for the updates being done to Google maps based on the status of each pothole in any path.

5.2 Backend:

a. **Python3**: We try to implement the flask api scripts using python3 for enabling communication between the frontend and backend. Also helps to build the Object detection model using darkflow.

b. **Tensorflow**: TensorFlow supports high-level APIs, through which Machine Learning and deep learning models can be built easily using Neural Networks. It also has many pretrained datasets. It also helps in visualisation and graphic representation of trained models.

c. **Flask**: As stated above it helps us to build apis for connecting our frontend applications with the backend.

d. **Apache server:** Helps in establishing a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure). The browser requests for a specific file or resource and initiates the process.

e. **Postman**: It is used for testing the functionality of each endpoint which were created using the flask apis.

f. **Mysql**: This is the database we use to store data about the 4 entities explained above and display the relevant data between the entities for their communication.

g. **AWS for cloud deployment:** We use this to deploy our complete backend consisting of the database, flask apis and the object detection model.

h. **Darknet(for deep learning model):** Darknet is a framework to train neural networks, it is open source and written in C/CUDA and serves as the basis for YOLO. Darknet is used as the framework for training YOLO, meaning it sets the architecture of the network.

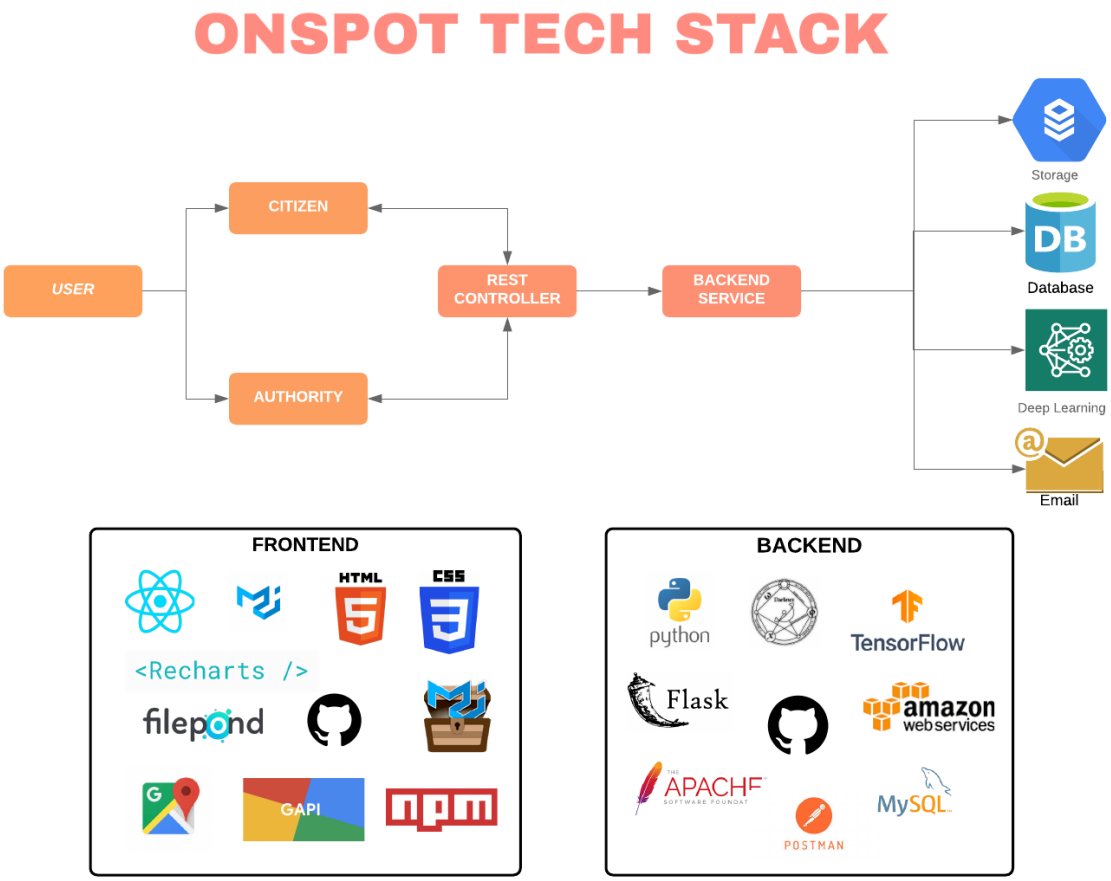

ONSPOT TECH STACK
USER
CITIZEN
AUTHORITY
REST CONTROLLER
BACKEND SERVICE
Storage
Database
Deep Learning
Email
FRONTEND
HTML
CSS
<Recharts />
filepond
GAPI
npm
BACKEND
python
TensorFlow
Flask
amazon web services
APACHE
MySQL
POSTMAN
TOOLS, LIBRARIES AND LANGUAGES USED

Fig. 5.1

Chapter-6

The Object Detection Model

6.1 Introduction

- YOLO has a different way of looking at the task of Object Detection as opposed to the RCNN algorithm which is discussed in this article. It avoids the Region Proposal Phase. YOLO takes the image and divides it into equal-sized grids. For each grid, we first compute the confidence that this grid lies in the center/inside an object or there is an object around it. If there is an object, then the next point is how we adjust this grid(again the same four values w, h, x, y of the true box).

- So, once again we need to predict the displacement from the coordinate that we have here(the red box in the above image represents the true region and the blue box represents the proposed region). So, from the blue box in the above image, we need to find out this displacement that needs to be done in the width and the height so that the entire dog(object/entity) is inside the box both horizontally and vertically and then also how much should we move the origin so what could be new co-ordinates for origin which is x, y.

- So, this is again a combination of classification and regression problems. First, we predict whether this grid/box lies at the centre of an object or inside an object or if there is an object around it or not. If yes, then we predict the four coordinates of the box. Then suppose we are dealing with 20 different classes, then we also do a classification/softmax over these 20 classes. And we know that the true label, which in this case, is ‘dog’ and the model would predict something, so we find the difference between the true label and the predicted label and that would become the loss function.

- The true label can be represented as a probability distribution where all the probability mass is on the true class(dog in this case), and the output is also a predicted distribution.

- So, we have 49 grids(input image is divided into 7 X 7 grids) and for each of the grids we are first computing the confidence value, then the next four values corresponding to the coordinates and then the k values(which are the probability for the k classes) that we have. So, for every grid, we predict (k+5) values. That's how YOLO looks at the task of Object Detection. There are no multiple region proposals, we just take equal-sized grids and then try to stretch all of them to be able to fit the right box.

Now let's explore the steps to train our model in Darkflow using the YOLO algorithm.

- The different YOLO implementations (Darknet, Darkflow, etc) are amazing tools that can be used to start detecting common objects in images or videos "out of the box", to do that detection it is only necessary to download and install the system and already trained weights. For instance, in the official Darknet website we can find the steps to obtain and use the weights trained for COCO dataset or VOC PASCAL.

- There will be cases, however, that the objects we want to detect are simply not part of these popular datasets. In such cases we will need to create our training set and execute our own training.

6.2 Procedure to train our model

- We will follow step by step the procedure to create the dataset and run the training using Darkflow (a Darknet translation to run over TensorFlow).

Step 1: Obtain the images

For this tutorial, we will train Darkflow to detect the presence of potholes.

Fig. 6.1

Step 2: Annotate the objects

- It is the process of labelling our input images before sending them to Darkflow.
- Since this particular problem requires the detection of multiple classes, we will use my fork of LabelImg (https://github.com/tzutalin/labelImg) to annotate the images. It is also easier to install and simpler to use.

- To install LabelImg, we will run:

```
pip3 install labelImg
```

```
cd [path of labelImg repository downloaded]
labelImg
```

- Then click open Dir in the LabelImg tool window and create a RectBox. Now the image has been annotated. Save it and it would save in xml format to the respective location you gave in save Dir option in the LabelImg tool.

- Now, for training the model we use AWS EC2 instances as Darkflow training requires GPUs for processing.

Step 3: AWS EC2 Configuration:

- Go to AWS and create an account if not registered yet. Then create an EC2 instance with the following configurations.
- AWS EC2 C5 Instance (model: c5.xlarge) after choosing Ubuntu 18 which features the Intel Xeon Platinum 8000 series and offers a set of 4vCPUs each with 8 GiB of memory was chosen for training the object detection model. Now follow the other steps of installing Darkflow and training from this EC2 instance only.

Step 4: Installing Darkflow

- To download and install the system, the easiest way is to run the following commands (you may need to install tensorflow and numpy beforehand):

```
git clone https://github.com/thtrieu/darkflow
cd darkflow
python3 setup.py build_ext --inplace
```

Fig.6.2

Step 5: Modifying configuration files (configuring the network)

- There are two possible network configurations that can be used to train, yolo or tiny-yolo. As the name suggests tiny-yolo is a smaller network that obviously will be faster to process but will suffer from lower accuracy.

- Under cfg/ there are configuration files for both of these versions:

```
$ ls -1 cfg/ | grep yolo-voc.cfg
tiny-yolo.cfg
```

```
yolo.cfg
```

- We need to first configure Darkflow by modifying the configuration file and labels.txt file.

- We will use the tiny yolo configuration, for that we need to create a copy of this file tiny-yolo-voc.cfg, that we will need to modify for our problem

```
cp cfg/tiny-yolo-voc.cfg cfg/yolov2-tiny-voc-1c.cfg
# modify cfg/.cfg
vi cfg/yolov2-tiny-voc-1c.cfg
```

- Then we need to make a copy from cfg/tiny-yolo-voc.cfg and create a cfg/tiny-yolo-voc-1c.cfg file with the same content. Change the line 114 to filters=30 [num * (classes + 5)] and set classes=1 as we have only one class 'pothole'.
- In the label.txt file remove all the labels and just keep the pothole label.

Step 6: Starting the training

- We have come a long way, haven't we? The good news is that we are ready to run the training.

- Just as a reminder, in Step 2 we created the training set, consisting of several image files and their corresponding xml file containing the annotations. They will

be stored in these locations (you need to substitute <path_to_labelImg-tool> with the actual path you installed the script.

<path_to_labelImg>/Images/001

<path_to_labelImg-tool>/AnnotationsXML/001

Now, coming back to Darkflow, to start the training we need to run

```
   python3 flow --model cfg/yolov2-tiny-voc-1c.cfg \
   --labels labels.txt  \
   --train --trainer adam \
   --dataset "<path_to_labelImg-tool>/Images/001" \
   --annotation
"<path_to_labelImg-tool>/AnnotationsXML/001"
   --gpu 1.0
```

- Darkflow should then start booting up and loading the images, eventually you should start seeing lines like these, printing the loss of each training step:

```
step 1 - loss 227.32052612304688 - moving ave loss
227.3205261230469
step 2 - loss 226.1829376220703 - moving ave loss
227.2067672729492
step 3 - loss 225.60186767578125 - moving ave loss
227.046277313232
step 4 - loss 227.2750701904297 - moving ave loss
227.0691566009522
```

```
step 5 - loss 227.2261199951172 - moving ave loss
227.0848529403687
```

- As we know by now, deep learning usually takes a lot of time to train. The time will obviously depend entirely on our hardware, the size of our training set, etc. It may take everything from one hour to several days to give useful results.

- By default DarkFlow will save a checkpoint every 250 steps, so we can stop the training at any time to take a break and/or validate the current weights. If you want to restart from the last checkpoint, you just need to add --load -1 to the same command you used to start the training.

- Using the test sets the model can be verified to check the accuracy of our newly trained object tracking model. This can now also be applied to various snippets of video to highlight the potential of object detection on potholes. Refer to the link mentioned in the above step for the testing command. It also contains options to restart training from a previous checkpoint.

Step 6: Validating the results

- Now create a new directory called useful to validate the trained model using custom images named in and out taken from a random resource.

- The testing_1.py, testPotholeImage.py, testPotholeVideo.py, testPotholeVideoCam.py files are used to test images, Pothole videos, Live videos

taken from camera for our need of validating the uploaded image from the frontend of our website.

Run the following in terminal:

```
python3  testPotholeImage.py
```

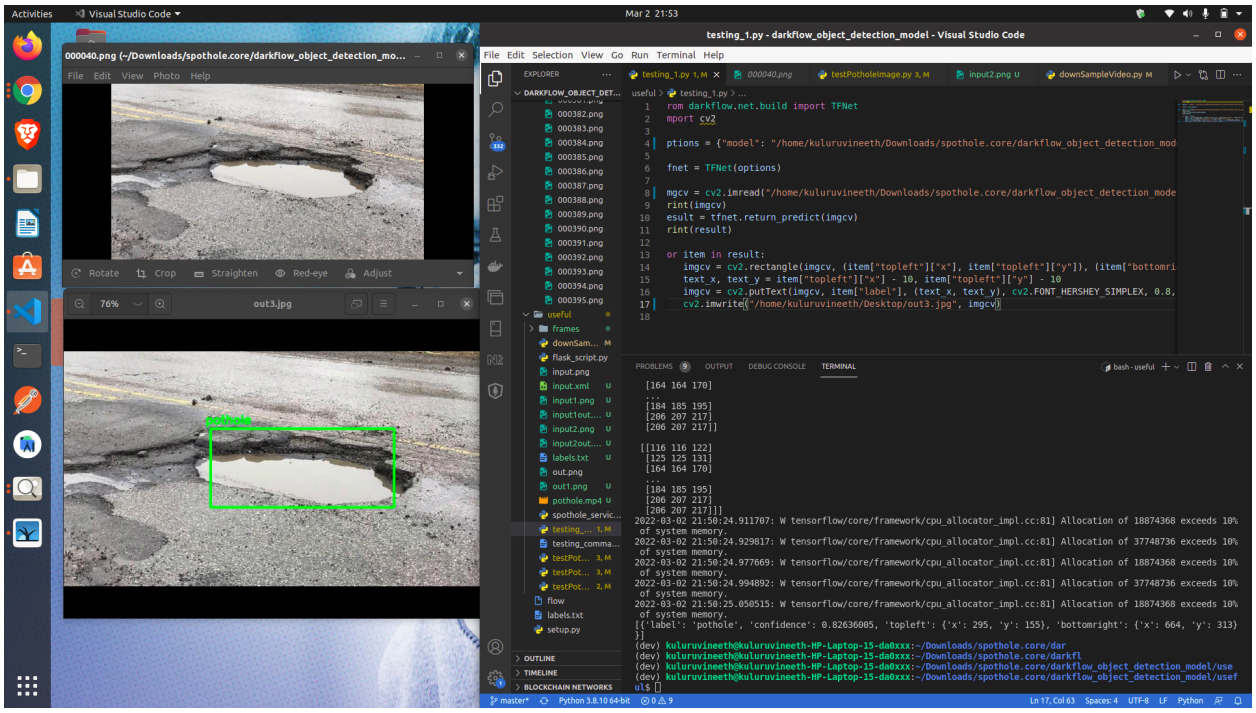


Fig. 6.3

- Darkflow will store the weights in the same directory as the checkpoint information. By default it will use `<your_git_directory>/darkflow/ckpt` . Four files will be created every checkpoint, and a text file called checkpoint will be updated.

- According to this, the .meta file is where the weights are stored. And here it says the .meta, .index and .data are files related to TensorFlow.

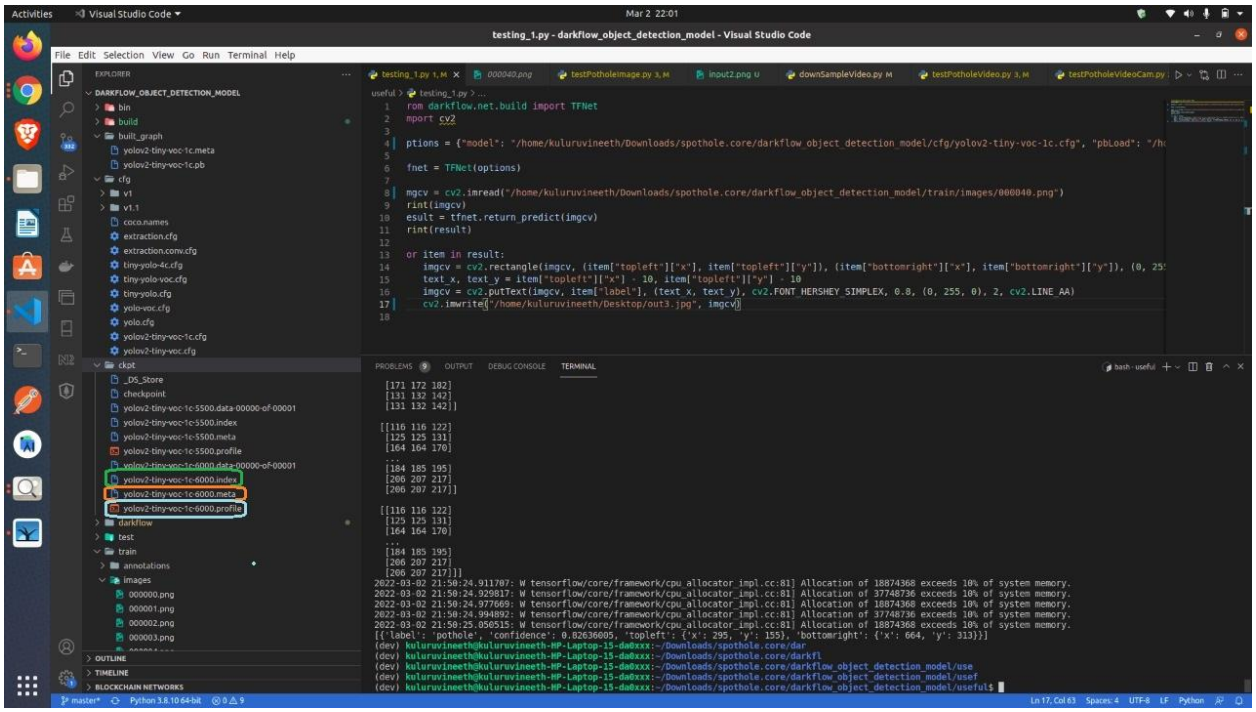


Fig. 6.4

- The above process has been done for image testing alone. Similarly we can test for any videos or live video running as well by running the other test scripts.

Chapter-7

Mysql Database Server Configuration

Step 1 : Before the Database configuration, we build a schema of the actual database to be used for storing data from Authorities and Citizens.

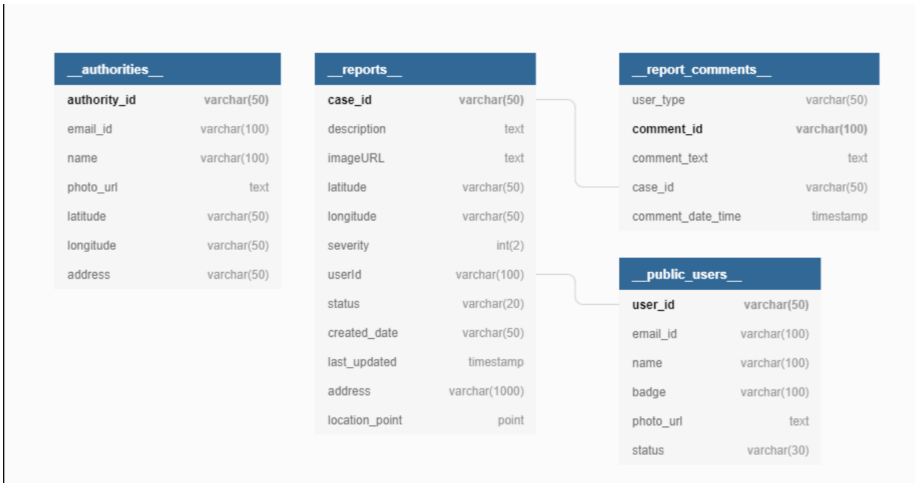


Fig. 7.1

Step 2: Follow the below commands to setup the above database schema in the remote instance configured in aws. Here we have already configured the above schema from local computer and dumped them into the remote instance

```
sudo apt update
```

```
sudo apt install mysql-server
sudo systemctl status mysql
sudo mysql_secure_installation
sudo mysql
mysql -u root -p
create database onspot;
sudo mysql -u root -p onspot < onspot_db.sql
use database onspot;
show tables;
```

```
ubuntu@ip-172-31-89-91:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.37-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use onspot
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables
    ->

^C
mysql> show tables;
+---------------------+
| Tables_in_onspot    |
+---------------------+
| __authorities__     |
| __authority_zones__ |
| __public_users__    |
| __report_comments__ |
| __reports__         |
+---------------------+
5 rows in set (0.00 sec)
```

Fig. 7.2

Step 3: Now to test the above configuration, we add some dummy data using the following commands

```
insert into <table_name> <col1> <col2> values <value1> <value2>;
```

Step 4: After insertion of dummy data lets query the table for testing using the below command

```
select * from <table_name>;
```

```
mysql> select * from __authorities__;
+----------------------+--------------------------+--------------+-----------------------------------------------------------------------
-+-----------+---------+
| authority_id         | email_id                 | name         | photo_url
 | longitude | address |
+----------------------+--------------------------+--------------+-----------------------------------------------------------------------
-+-----------+---------+
| 112945513920895542589 | vineethreddy3268@gmail.com | Vineeth Reddy | https://lh3.googleusercontent.com/a/AATXAJzlzMD6VHRcXc3-EZzugo_r
 | 78.0373   | NULL    |
| 2                    | vineethreddy3268@gmail.com | KuluruVineeth | https://images.theconversation.com/files/4168/original/Jobby.jpg
 | NULL      | NULL    |
+----------------------+--------------------------+--------------+-----------------------------------------------------------------------
-+-----------+---------+
2 rows in set (0.00 sec)
```

Fig. 7.3

```
mysql> select * from __public_users__;
+----------------------+--------------------------+--------------------------------------+------------+-------
-----+
| user_id              | email_id                 | name                                 | badge      | photo_
tus  |
+----------------------+--------------------------+--------------------------------------+------------+-------
-----+
| 107650217403686224156 | 18bcs043@iiitdwd.ac.in      | KULURU VINEETH KUMAR REDDY IIIT Dharwad | LIVE_VALUE | https:
owed |
| 112945513920895542589 | vineethreddy3268@gmail.com  | Vineeth Reddy                        | LIVE_VALUE | https:
owed |
| 117954990956994765891 | kuluruvineeth8623@gmail.com | Kuluru vineeth kumar Reddy           | LIVE_VALUE | https:
owed |
+----------------------+--------------------------+--------------------------------------+------------+-------
-----+
3 rows in set (0.00 sec)
```

Fig. 7.4

```
mysql> select * from __report_comments__;
+-----------+------------------------------------------+----------------------------------------------------------------------------------------------
----------------------------+------------+---------------------+
| user_type | comment_id                               | comment_text
                            | case_id    | comment_date_time   |
+-----------+------------------------------------------+----------------------------------------------------------------------------------------------
----------------------------+------------+---------------------+
| U         | 12958205134026639347055889521846310933 8 | are you there
                            | 3079827395 | 2022-03-18 06:19:14 |
| A         | 139077659785830254122018810201994708365 | Authority's Update - [Status: Approved, Severity: 8] - Good job keep contributing to society.
                            | 3548282361 | 2022-03-19 09:05:41 |
| A         | 145457736180022421447129599445242040110 | Authority's Update - [Status: Working, Severity: 8] - Work has been started.
                            | 3548282361 | 2022-03-19 09:11:18 |
| U         | 281459149958317649236920468783736861866 | Please look into it.
                            | 3545238581 | 2022-03-19 07:47:11 |
| U         | 295391849646097454512186760429697588199 | Thank you for taking action against it.
                            | 3548282361 | 2022-03-19 10:45:59 |
| A         | 45935529808111550823723495601387410344  | Authority's Update - [Status: Completed, Severity: 8] - Repair works have been successfully comp
                            | 895856757  | 2022-03-19 09:28:46 |
| U         | 55355627551665527857625664441838735503  | Thank you for taking immediate action.
                            | 895856757  | 2022-03-19 11:41:24 |
| A         | 56692547502906264872309518190262762958  | Authority's Update - [Status: Cancelled, Severity: 6] - Thank you reporting we already have othe
 taken care at later stage. | 1029691210 | 2022-03-19 09:27:40 |
| U         | 89036228511135132091383045801697771750  | hey
                            | 3079827395 | 2022-03-18 06:19:04 |
+-----------+------------------------------------------+----------------------------------------------------------------------------------------------
----------------------------+------------+---------------------+
9 rows in set (0.00 sec)
```

Fig. 7.5

```
mysql> select * from __reports__;
+------------+------------------------------------------------------------------------------------------+-------------------------------
-------------------+----------+--------------------------+--------------------+------------------------------------------------------
--+--------------------------+
| case_id    | description                                                                              | imageURL
rId                | status   | created_date             | last_updated       | address
  | location_point           |
+------------+------------------------------------------------------------------------------------------+-------------------------------
-------------------+----------+--------------------------+--------------------+------------------------------------------------------
--+--------------------------+
| 1029691210 | On the way to railway beside left entrance i found this....                              | https://i.ibb.co/vZhgxQZ/onspot-im
954990956994765891 | cancelled | 2022-03-19 08:54:38.880395 | 2022-03-19 10:02:45 | Kurnool Railway Station, Narasimha Reddy Nagar, Bangar Pet, Kurnool,
  |        h◆◆◆/@◆◆`◆S@        |
| 2772045637 | I almost crashed.                                                                        | https://i.ibb.co/GMStn4Y/image-url
954990956994765891 | submitted | 2022-03-19 08:52:23.498355 | 2022-03-19 10:24:10 | Kurnool, Andhra Pradesh, India
  |        G◆| ◆/@◆4L◆b◆S@      |
| 3079827395 | vfgffg                                                                                   | https://i.ibb.co/kSt0PhC/image-url
954990956994765891 | submitted | 2022-03-18 05:18:57.086331 | 2022-03-19 10:25:03 | Chennai, Tamil Nadu, India
&U**@;◆DsST@   |
| 3545238581 | Pathethic conditons.                                                                     | https://i.ibb.co/qYYyWHQ/image-url
954990956994765891 | submitted | 2022-03-18 08:03:52.468754 | 2022-03-19 10:25:50 | Bengaluru, Karnataka, India
fS@       |◆◆bu◆)@◆◆◆P
| 3548282361 | Near the bus departure area there is huge pothole and it is very disturbing...please look into this.  | https://i.ibb.co/dGskGXm/image-url
650217403686224156 | working   | 2022-03-19 08:57:04.500836 | 2022-03-19 10:27:03 | Kurnool Bus Stand, Sampath Nagar, Kurnool, Andhra Pradesh, India
  |        ◆|\/@◆}(BS@     |
| 591716269  | sc                                                                                       | https://i.ibb.co/wzwc58S/image-url
954990956994765891 | submitted | 2022-03-18 05:06:59.319609 | 2022-03-19 10:27:39 | 401, Main Rd, near Gokul Chat, Hashmath Gunj, Subhash Nagar, Badi Cho
a |        b◆[>◆b1@X:◆%◆S@     |
| 895856757  | Right infront of medical college entrace the pothole is causing huge traffic...please repair it asap. | https://i.ibb.co/jLBBkHB/image-url
650217403686224156 | completed | 2022-03-19 08:58:19.274204 | 2022-03-19 10:28:16 | Kurnool Medical College, Kisan Ghat Road, Kurnool, Andhra Pradesh, In
  |        18◆뵘/@◆◆>L◆◆S@       |
+------------+------------------------------------------------------------------------------------------+-------------------------------
-------------------+----------+--------------------------+--------------------+------------------------------------------------------
--+--------------------------+
7 rows in set (0.00 sec)
```

Fig. 7.6

Step 5: Now while configuring the MySQL database to comply with any problems on privilege issues, refer the below commands:

```
sudo mysql -u root
```

```
use mysql;
select User, Host, plugin FROM mysql.user;
update user SET plugin='mysql_native_password' where User="root";
FLUSH PRIVILEGES;
exit;

SHOW VARIABLES LIKE 'validate_password%';
SET GLOBAL validate_password_policy=LOW;
set GLOBAL validate_password_length=4;
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'root';
FLUSH PRIVILEGES;
```

Chapter-8

Core Backend Api Endpoints Testing

8.1 Citizen's Endpoints Testing:

8.1.1 /api/submit/report: For submitting a new report

Required parameters: description, location, latitude, longitude, imageURL, severity, userId

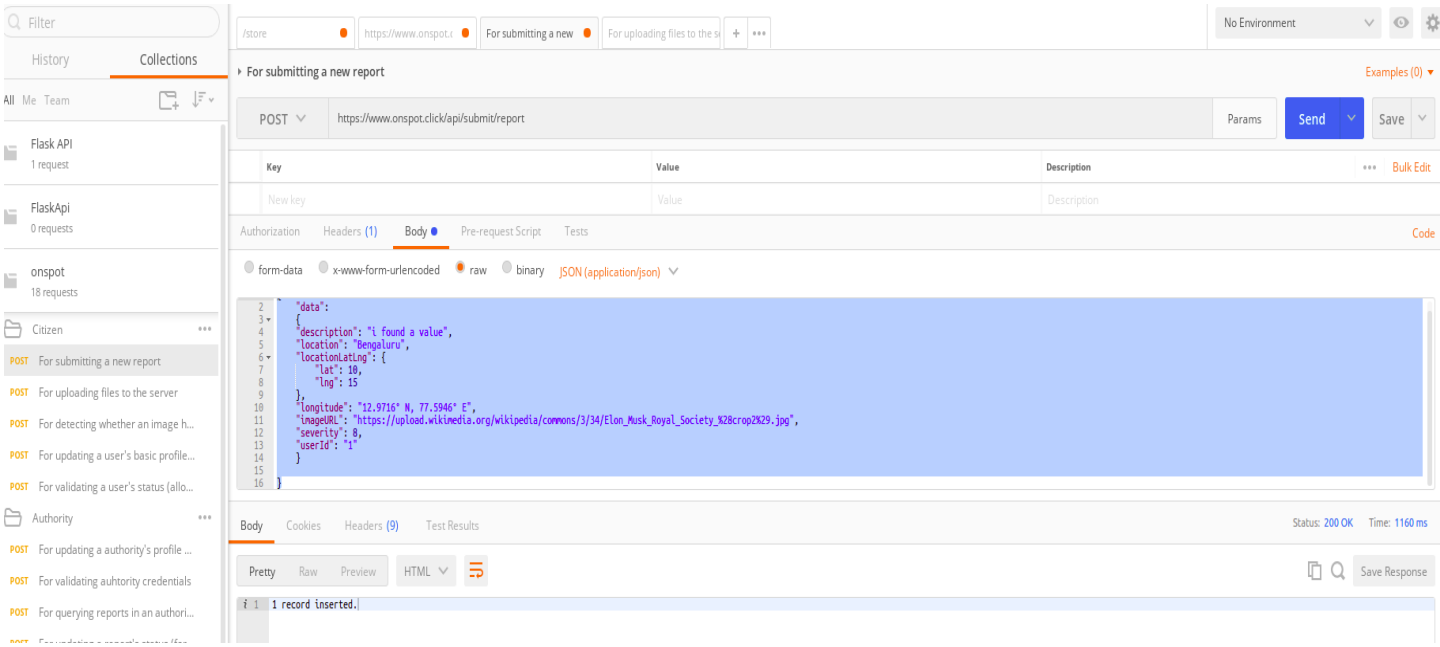


Fig. 8.1

8.1.2 /api/upload: For uploading files to the server

Required parameters: bytes (file data)

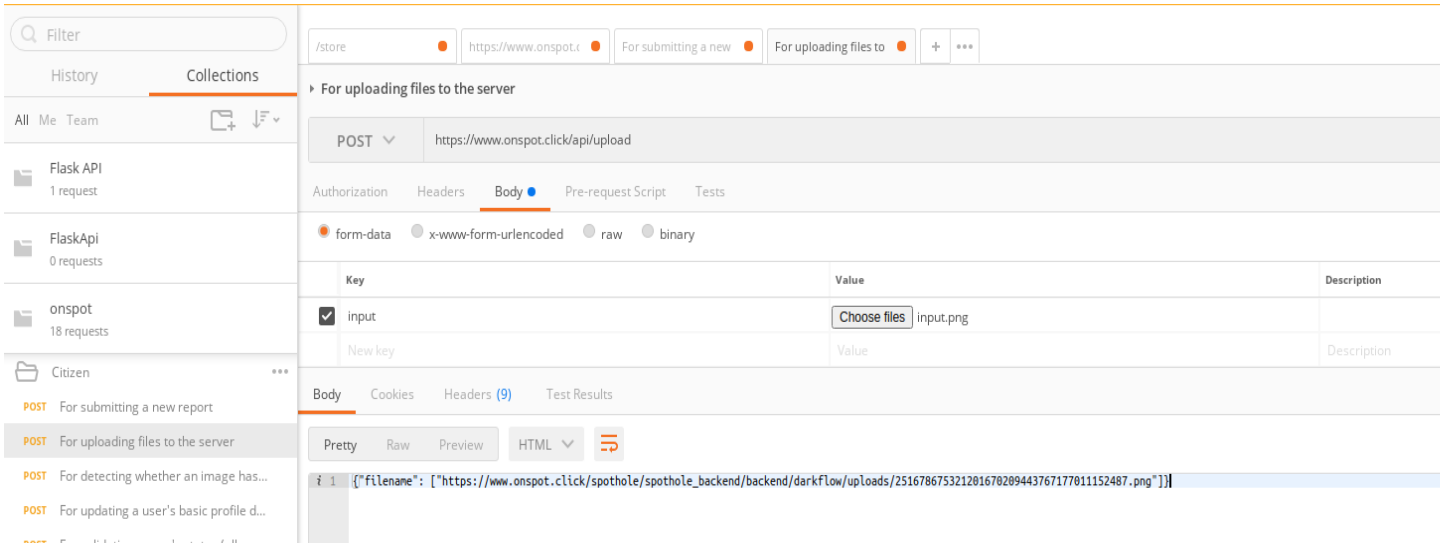


Fig. 8.2

8.1.3 /api/detect/single: For detecting whether an image has a pothole. (Object Detection)

Required parameters: image_url

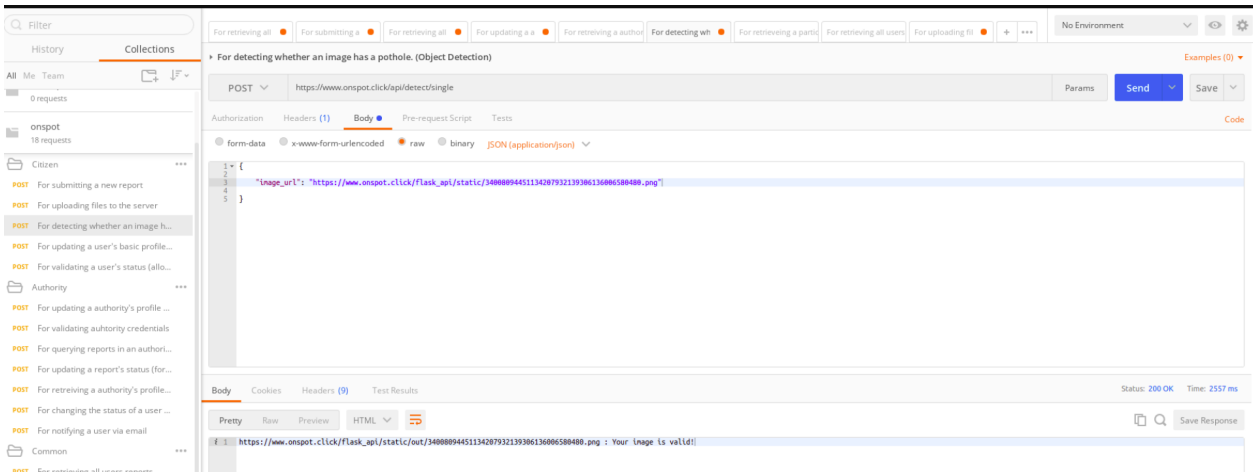


Fig. 8.3

8.1.4 /api/profile/update: For updating a user's basic profile details

Parameters required: userId, emailId, name, photoURL

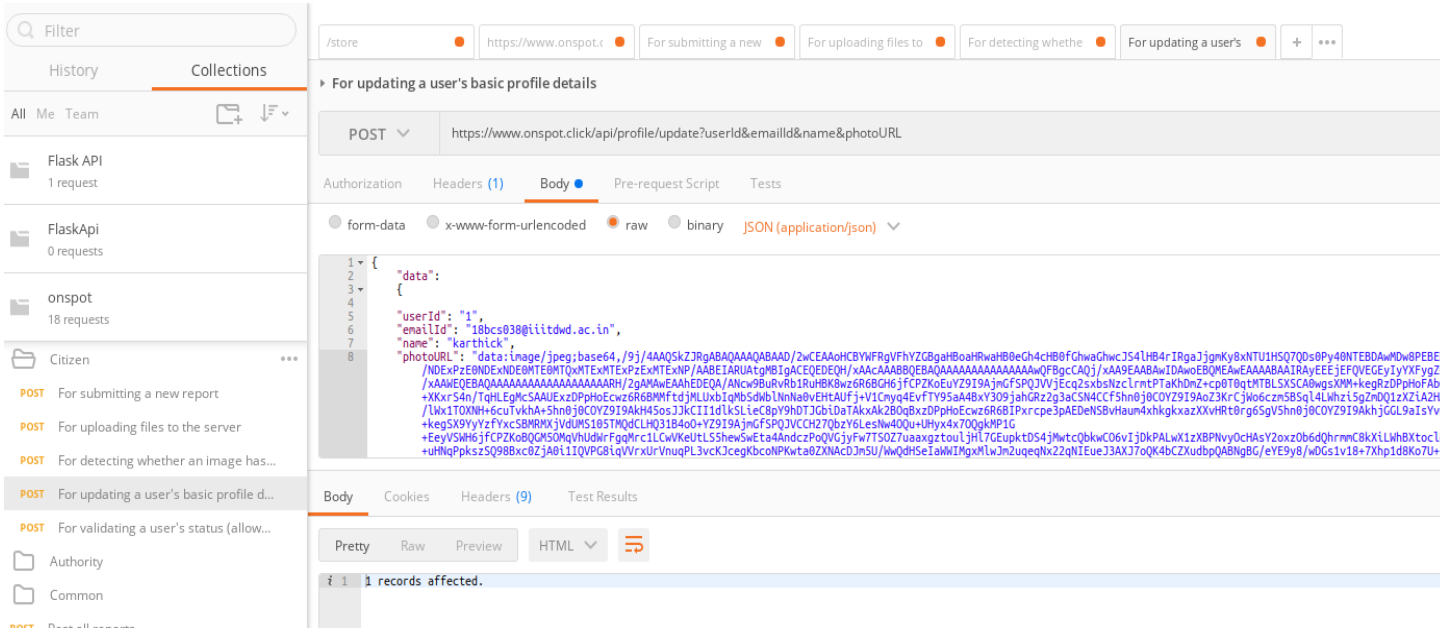


Fig. 8.4

8.1.5 /api/user/validate: For validating a user's status (allowed / blocked)

Required parameters: emailId

Fig. 8.5

8.2 Authority's Endpoints Testing:

8.2.1 /api/profile/authority/update: For updating a authority's profile details

Required parameters: authorityId, emailId, name, photoURL

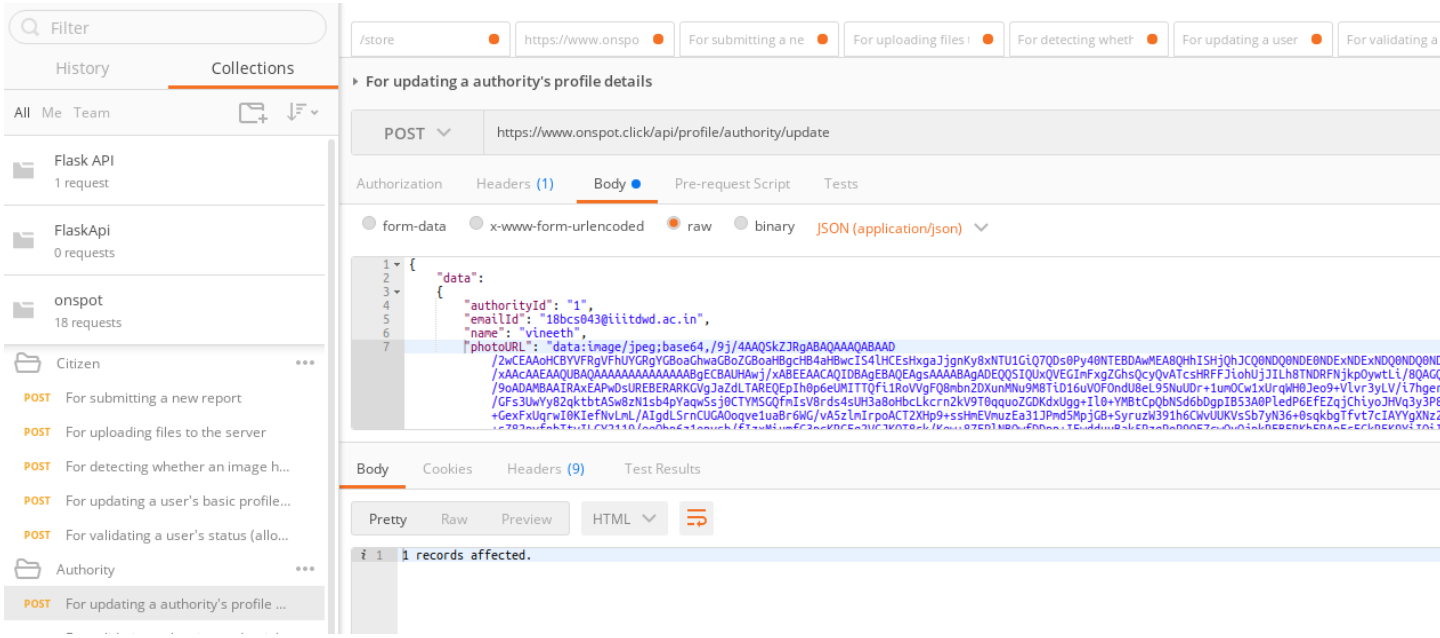


Fig. 8.6

8.2.2 /api/authority/check: For validating authority credentials

Required parameters: emailId

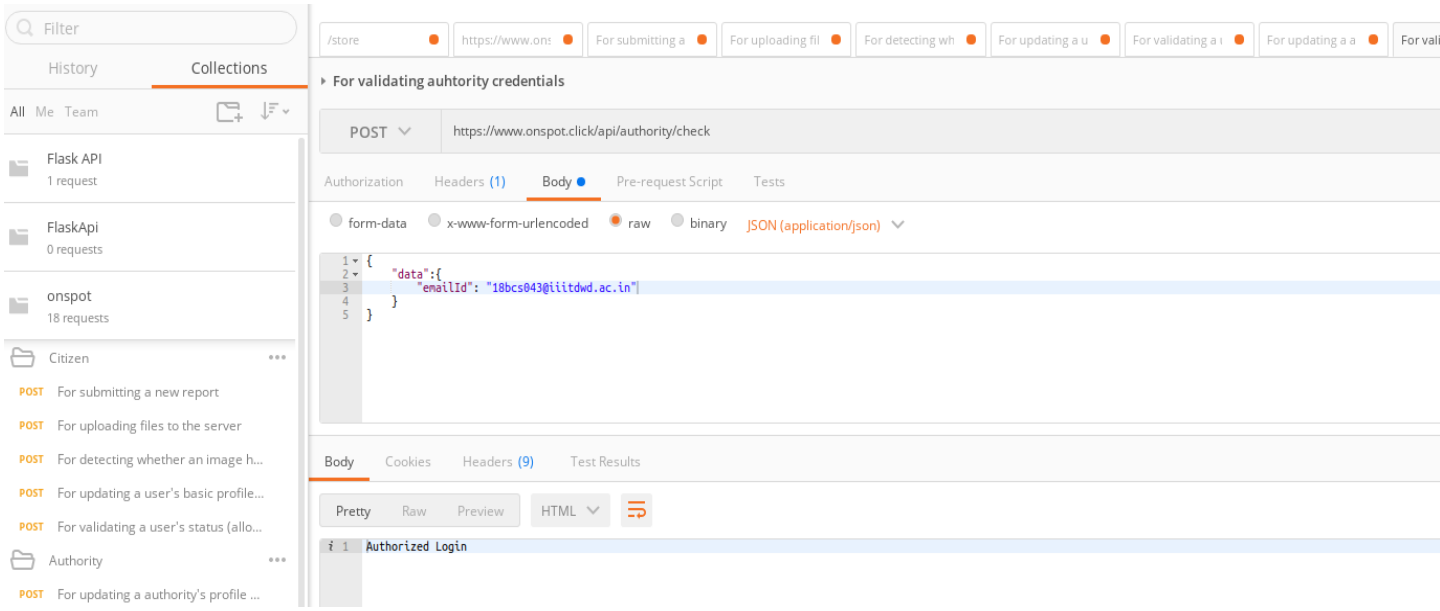


Fig. 8.7

8.2.3 /api/authority/reports/geonear: For querying reports in an authority's zone

Required parameters: authorityId

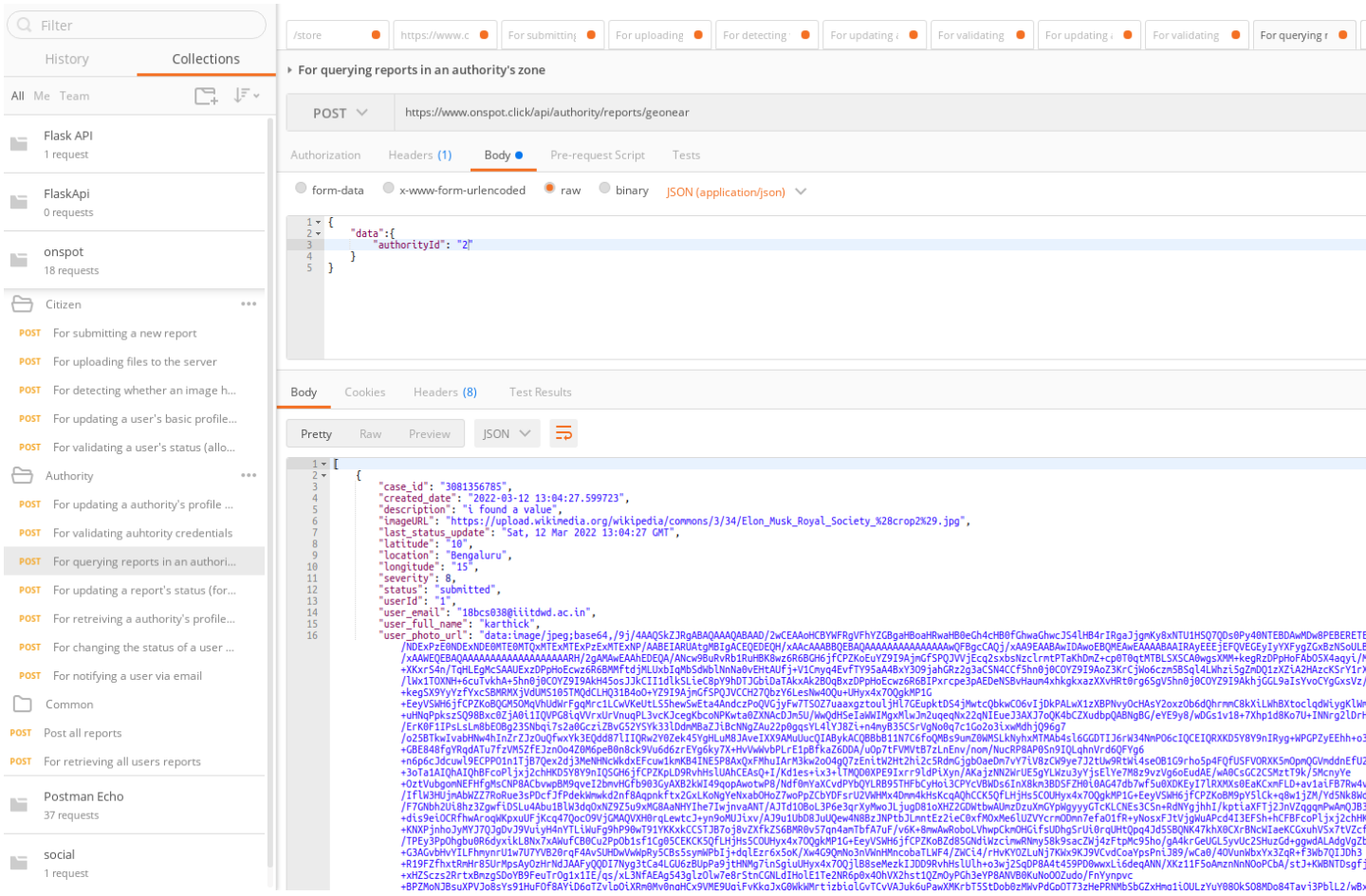


Fig. 8.8

8.2.4 /api/authority/update/report: For updating a report's status (for authority)

Required parameters: severity, status, caseId

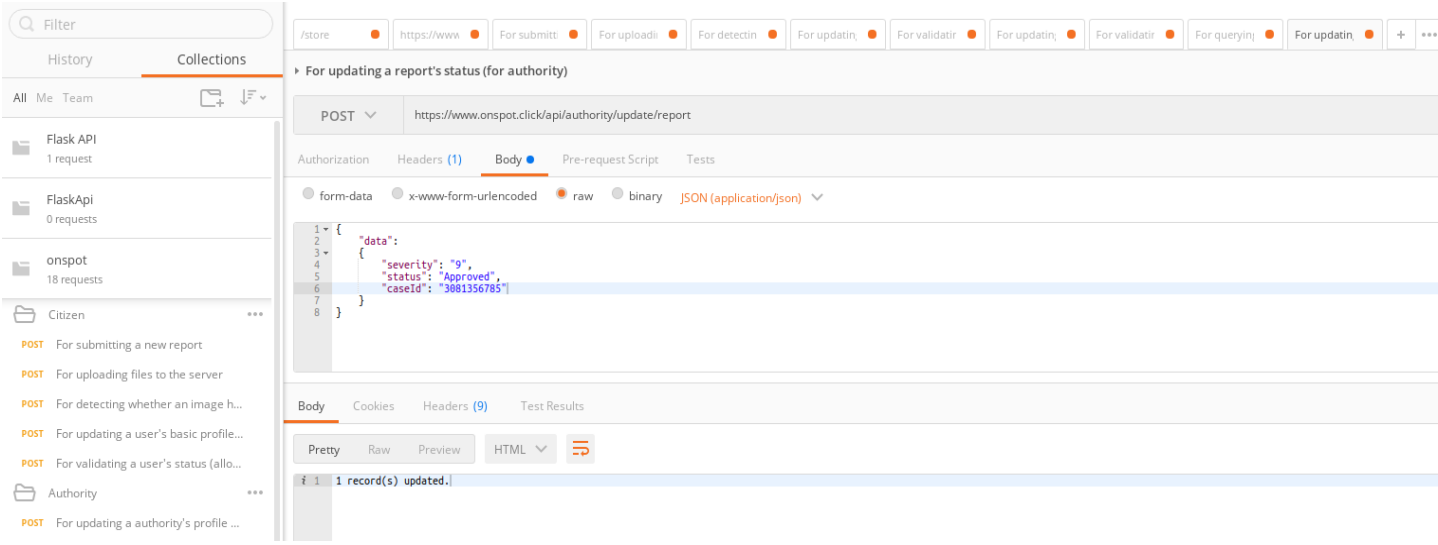


Fig. 8.9

8.2.5 /api/profile/authority/data: For retrieving a authority's profile details, location and address

Required parameters: authorityId

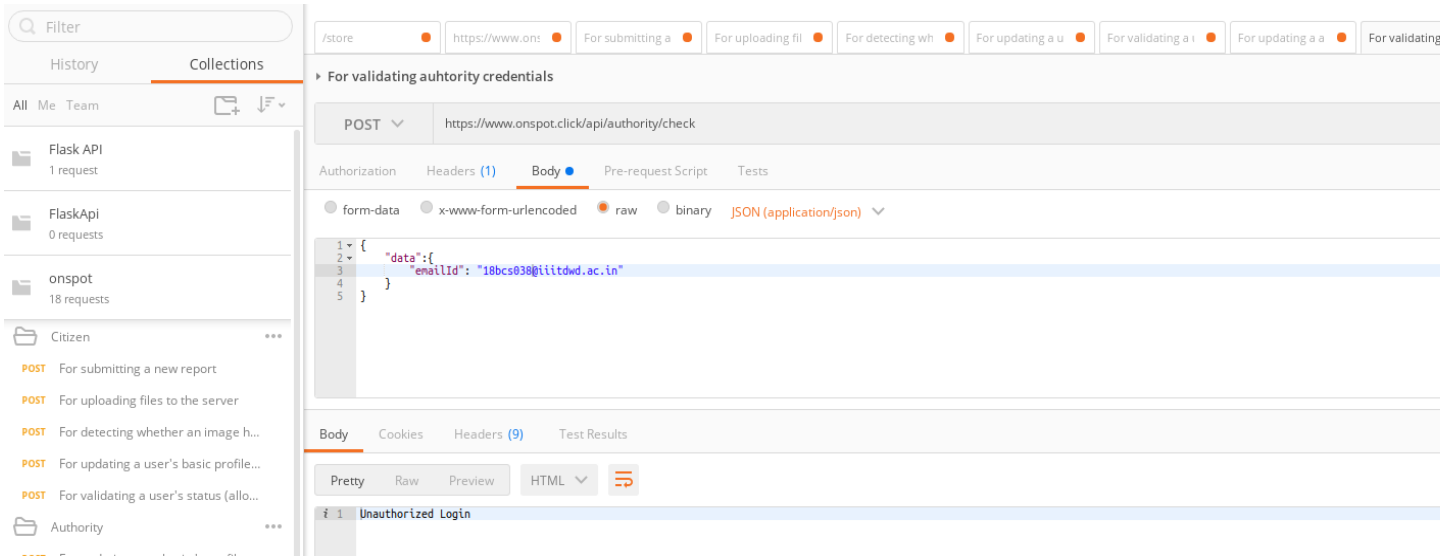


Fig. 8.10

8.2.6 /api/profile/authority/data: For retrieving a authority's profile details, location and address

Required parameters: authorityId

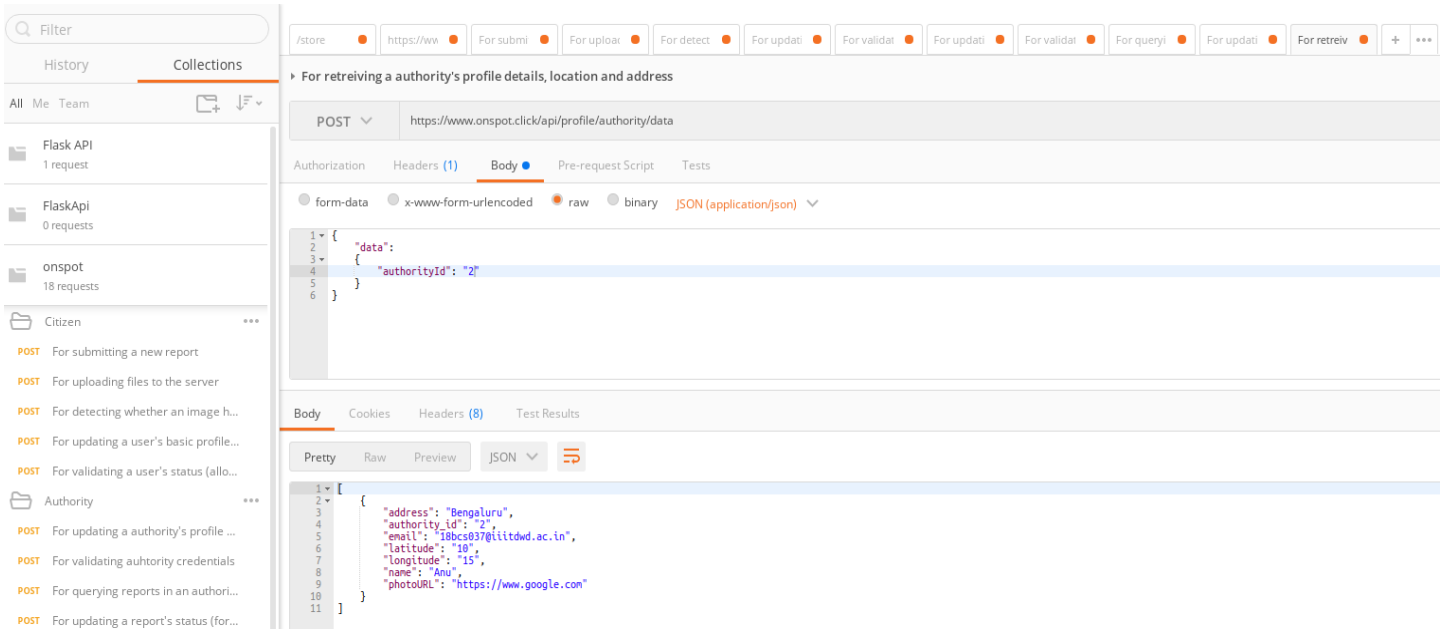


Fig. 8.11

8.2.7 /api/authority/update/user/status: For changing the status of a user (blocked / allowed)

Required parameters: userId, status

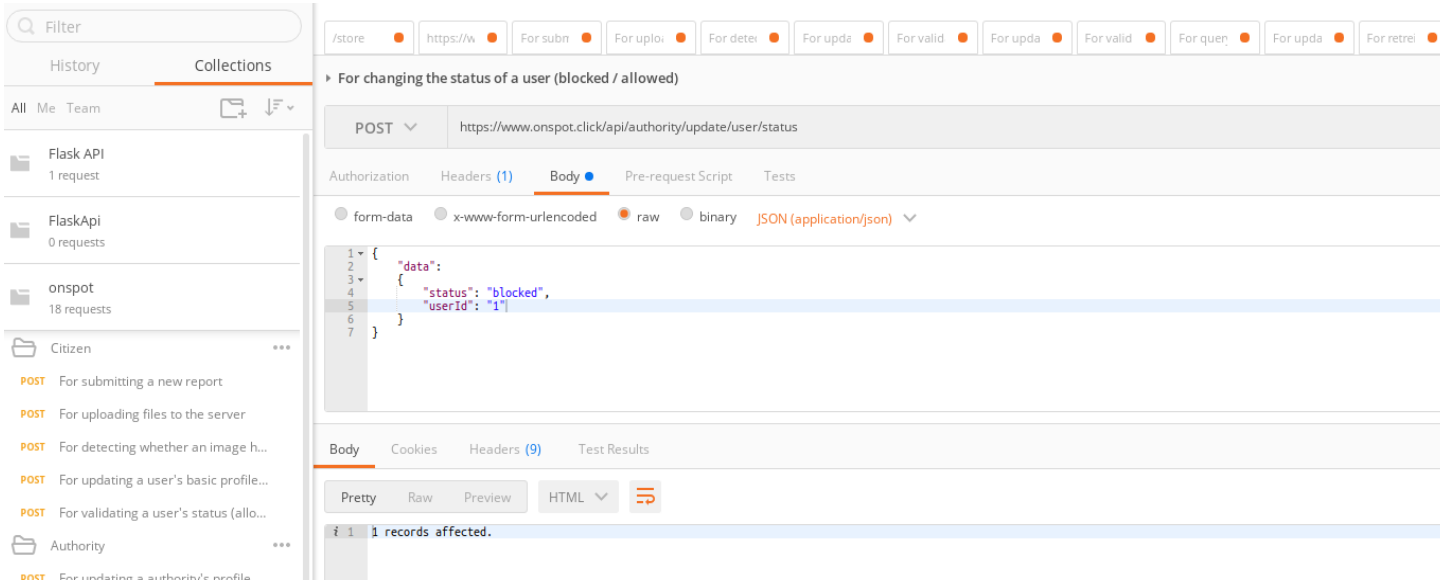


Fig. 8.12

8.2.8 /api/authority/send/email: For notifying a user/citizen via email

Parameters required: emailId, message, subject

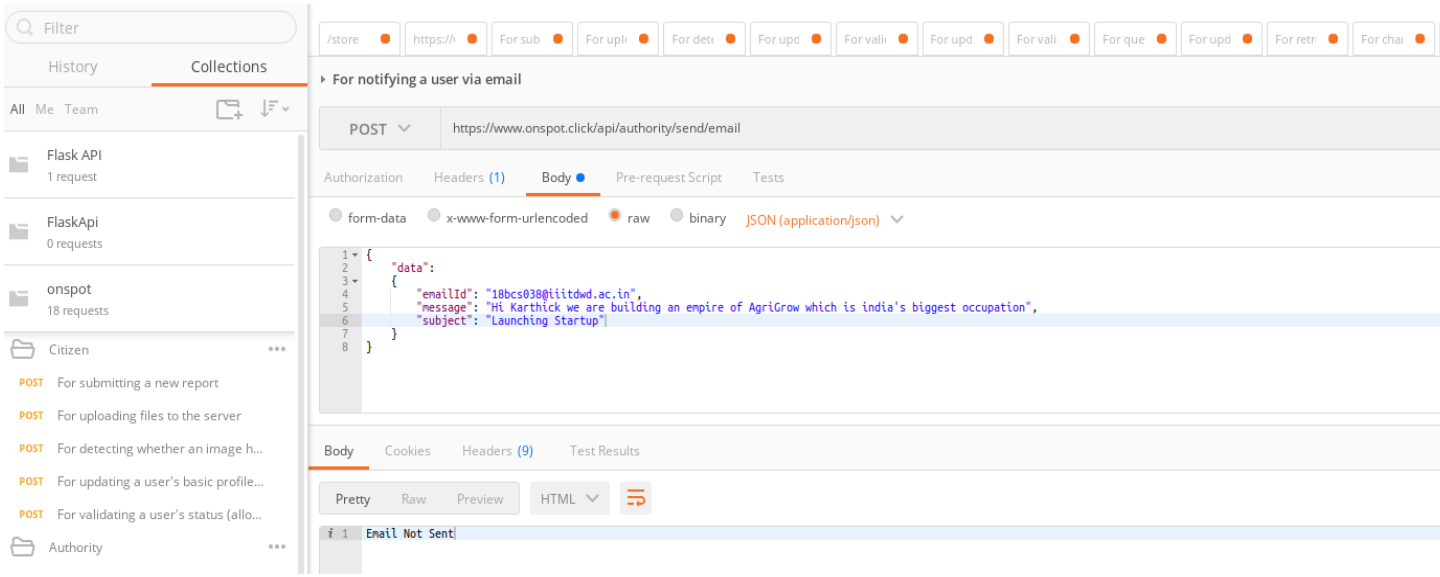


Fig. 8.13

8.3 Common Endpoints Test(Both Authority and Citizen):

8.3.1 /api/reports/all: For retrieving all users reports

Required parameters: (N/A)

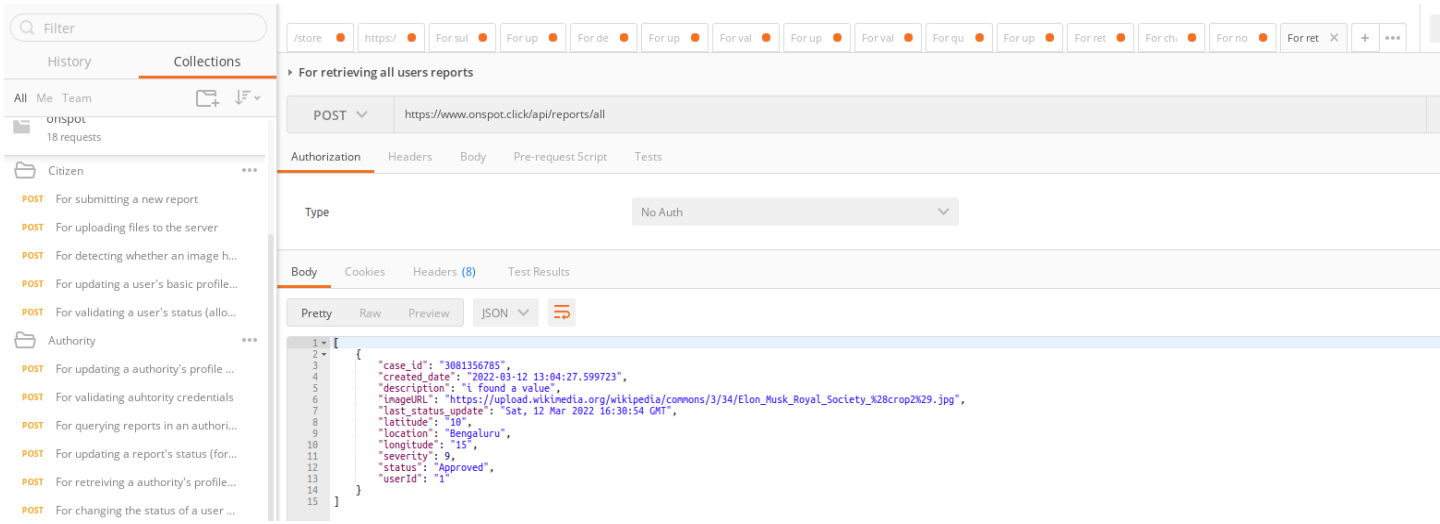


Fig. 8.14

8.3.2 /api/reports: For retrieving a particular user's reports

Required parameters: userId

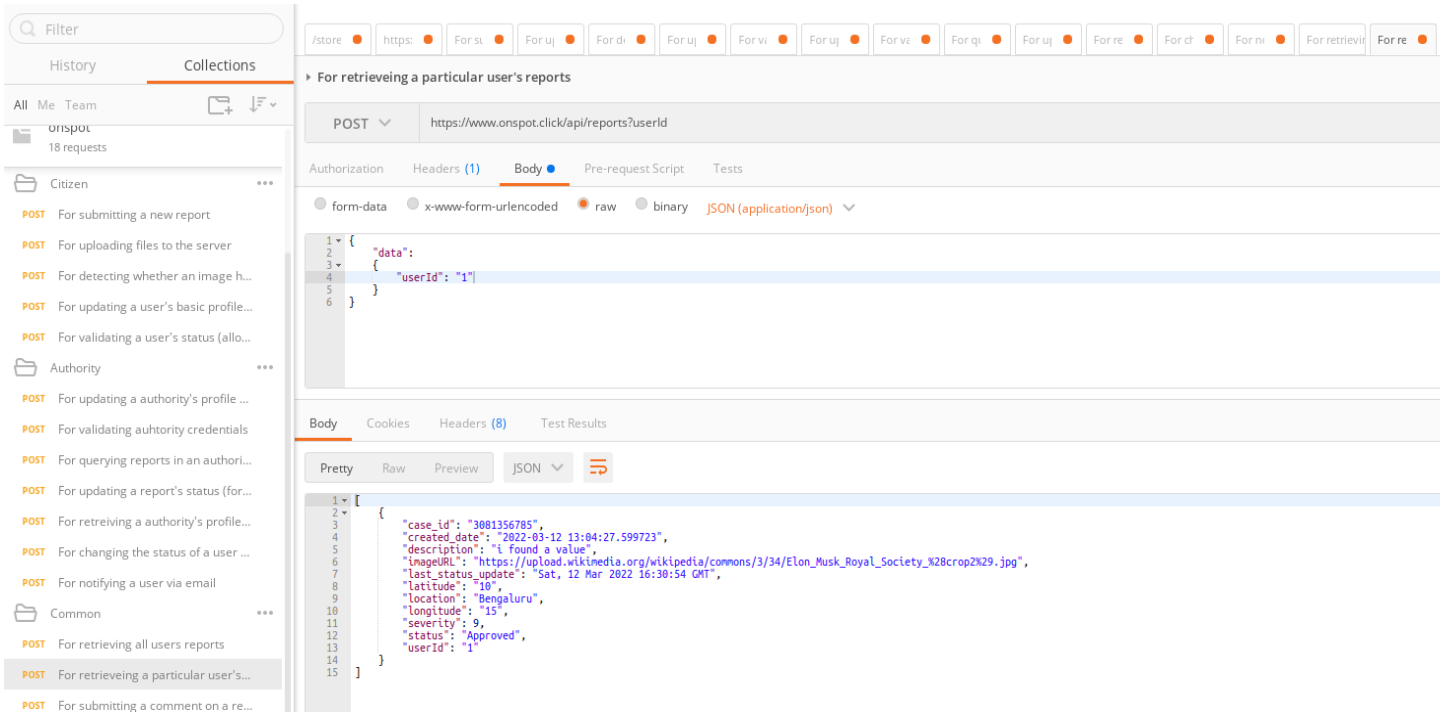


Fig. 8.15

8.3.3 /api/submit/report/comment: For submitting a comment on a report

Required parameters: userType, commentText, caseId

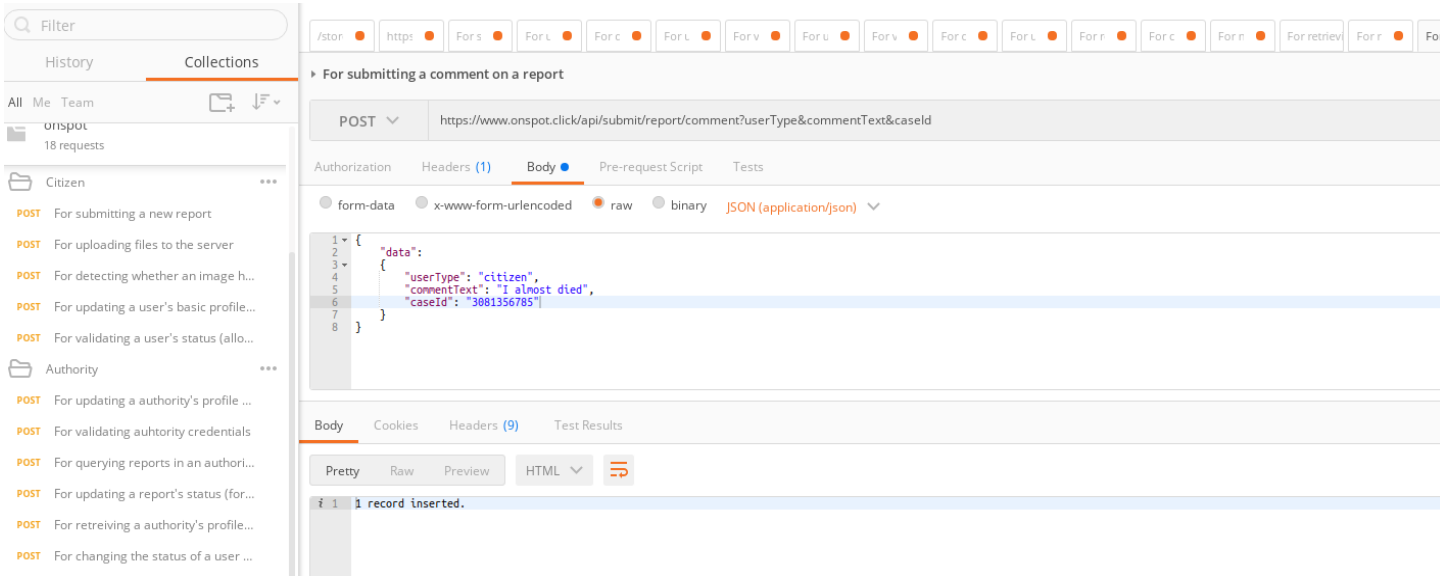


Fig. 8.16

8.3.4 /api/reports/comments: For retrieving all comments on a report

Required parameters: caseId

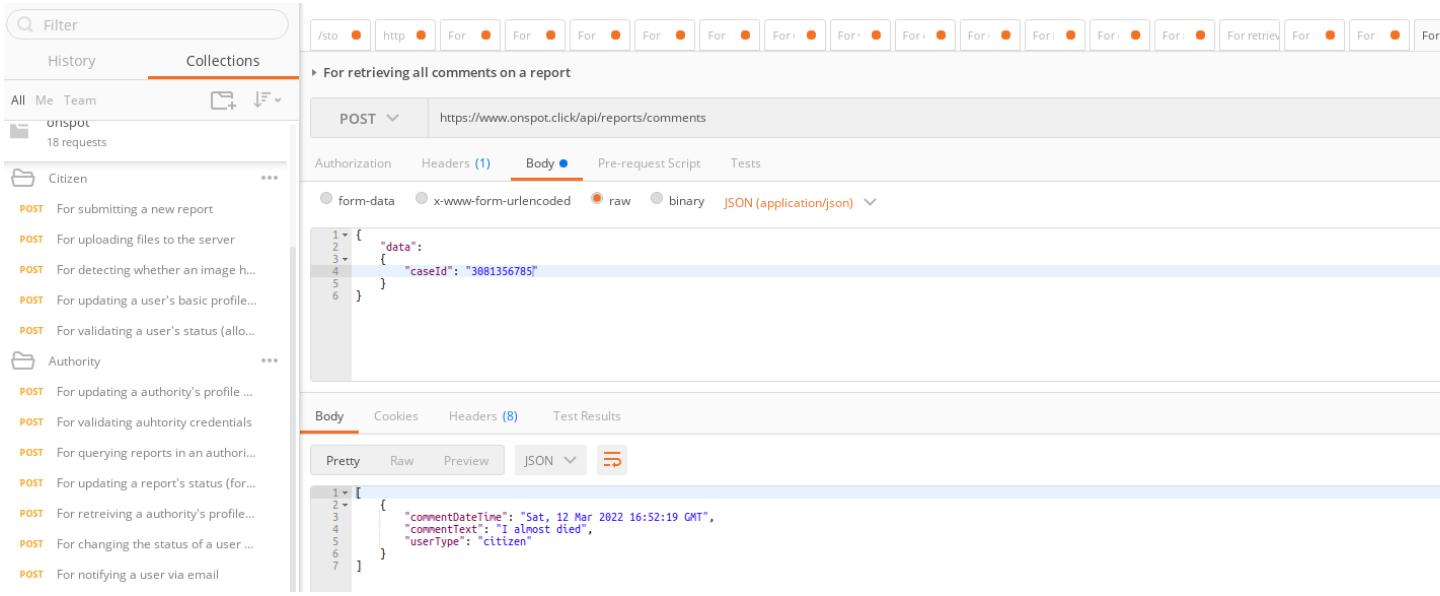


Fig. 8.17

Chapter-9

Apache2 Web Server Configuration

Step 1: Install Apache server using the below command

```
--------apache2-------
sudo apt update
sudo apt install apache2
```

Step 2: Make a directory <flask_api> under /var/www/html

```
cd /var/www/html
mkdir flask_api
cd flask_api
```

Step 3: Now copy the flask_api folder(in local machine) to written locally for api endpoints configuration to the remote instance

```
scp -r -i <private_key(.pem file)>
/https://github.com/kuluruvineeth/onspot.backend/flask_api/*
ubuntu@<remote_instance_ipv4>:/var/www/html/flask_api
```

Step 4: Now copy the darkflow_object_detection_model folder by creating the below repository tree under /var/www/html by following the steps 2 and 3 as shown above

Folder tree to construct : `onspot/onspot_backend/backend/`

Step 5: Now we need to setup the flask_api.wsgi file (Web Server gateway interface) for web server (Apache) - database interface configuration

Here are the contents of the wsgi file

```
import sys
sys.path.insert(0,'/var/www/html/flask_api')
from flask_api import app as application
```

Step 6: Enable the flask_api application using the 000-default.conf file. Refer to the command below

```
sudo vi /etc/apache2/sites-enabled/000-default.conf
```

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port
that
        # the server uses it to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file)
this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        ServerName www.onspot.click
```

```
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        WSGIDaemonProcess flask_api threads=5
        WSGIScriptAlias / /var/www/html/flask_api/flask_api.wsgi
        WSGIApplicationGroup %{GLOBAL}
        <Directory flask_api>
             WSGIProcessGroup flask_api
             WSGIApplicationGroup %{GLOBAL}
             Order deny,allow
             Allow from all
        </Directory>

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Step 7: Now create the utility.py file under the flask_api folder using the below constants

```
# constants
DARKFLOW_PATH = '/var/www/html/onspot/onspot_backend/backend/darkflow/'
MODEL_CFG_PATH =
'/var/www/html/onspot/onspot_backend/backend/darkflow_object_detection_model/cf
g/yolov2-tiny-voc-1c.cfg'
PROTOBUF_PATH =
'/var/www/html/onspot/onspot_backend/backend/darkflow_object_detection_model/bu
ilt_graph/yolov2-tiny-voc-1c.pb'
META_FILE_PATH =
'/var/www/html/onspot/onspot_backend/backend/darkflow_object_detection_model/bu
ilt_graph/yolov2-tiny-voc-1c.meta'
DEFAULT_THRESHOLD = 0.2
LABEL = 'label'
LABEL_X = 'x'
LABEL_Y = 'y'
LABEL_TOP_LEFT = 'topleft'
LABEL_BOTTOM_RIGHT = 'bottomright'
BOX_OFFSET = 10
DOMAIN_NAME = 'https://www.onspot.click' # YOUR HTTPS DOMAIN NAME HERE
WEB_DIR_PATH = '/var/www/html'
IMG_UPLOADS_DIRECTORY = '/var/www/html/flask_api/static/'
IMG_UPLOADS_DISPLAY_URL = DOMAIN_NAME + '/flask_api/static/'
DEFAULT_FILE_TYPE = '.png'

# sample paths
SAMPLE_IMG_PATH =
'/var/www/html/onspot/onspot_backend/backend/darkflow_object_detection_model/sa
mple_img/000008.png'
SAMPLE_IMG_OUTPUT_PATH =
'/var/www/html/onspot/onspot_backend/backend/darkflow_object_detection_model/sa
mple_img/a.jpg'
```

```
SAMPLE_IMG_OUTPUT_DISPLAY_URL = DOMAIN_NAME +
'/onspot/onspot_backend/backend/darkflow_object_detection_model/sample_img/a.jp
g'
```

Step 8: Let us now configure the app_controller.wsgi file under the /onspot/onspot_backend/backend which acts as an interface between the Apache server and the flask api for rendering the endpoints.

```
#! /usr/bin/python
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0,"/var/www/html/spothole/spothole_backend/backend/flask_api/")

# home points to the home.py file
from app_controller import app as application
application.secret_key = "somesecretsessionkey"
```

Step 9: Now, let us create the onspot.conf file for the api endpoints configuration from the flask_api folder

```
sudo vi /etc/apache2/sites-enabled/onspot.conf
```

```
<VirtualHost *:80>
    ServerName www.onspot.click
    ServerAdmin admin@www.onspot.click
    WSGIScriptAlias /
/var/www/html/onspot/onspot_backend/backend/app_controller.wsgi
```

```
    <Directory /var/www/html/onspot/onspot_backend/backend/flask_api/>
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined
RewriteEngine on
RewriteCond %{SERVER_NAME} =www.onspot.click
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Step 10: As we have configured only a development server , we need to use certbot in order to enhance it into a production server by creating an ssl certificate. Refer the below commands for the same.

```
--------certbot-------
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install python3-certbot-apache
sudo certbot --apache -d www.onspot.click

for ssl_contexts in flask application:
pip3 install pyopenssl
openssl req -x509 -newkey rsa:4096 -nodes -out fullchain.pem -keyout
privkey.pem -days 365 // For enabling the private keys for a duration of 365
days
```

Chapter-10

Citizen's Application Features

10.1 Create new report for creating a new report for the users to start reporting

- The users can navigate to the add a new report screen either by clicking on the floating action button at the bottom right or from the drawer by pressing the hamburger icon in the app bar.

- Here, users have the option to start their report by uploading an existing image of a pothole they clicked by browsing through the file system or by clicking a fresh photo using the inbuilt camera.

- Once users have decided on the method for uploading, the image under consideration is validated by the python based deep learning model placed on the backend server to verify if the image uploaded contains one or more potholes.

- If the media file contains one or more potholes, the users are presented with an option to share more details about their report.

- If not, users are presented with a feedback screen for an invalid image and the option to contact support.

- Assuming that the uploaded image is validated successfully, users are given the option to select a location (either current (presented after seeking permission)) or to enter a custom

location.

- After which users are asked to use a progress indicator to specify how severe the reported pothole(s) is(are) according to them.

- Finally they are asked to give more details about their report through a required input text area component.

- Once the user has successfully submitted the report, they can now view the same using the My Complaints Screen.

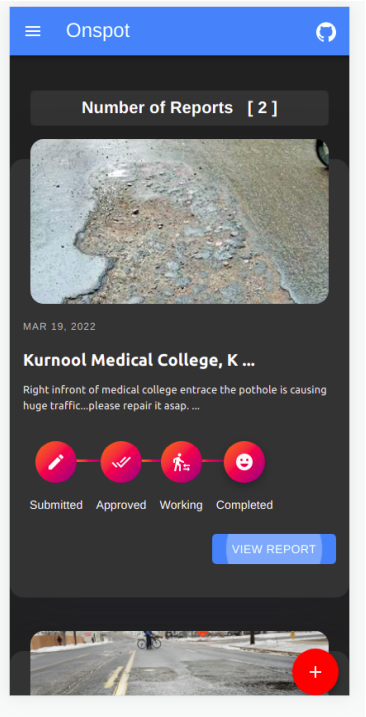


Fig. 10.1

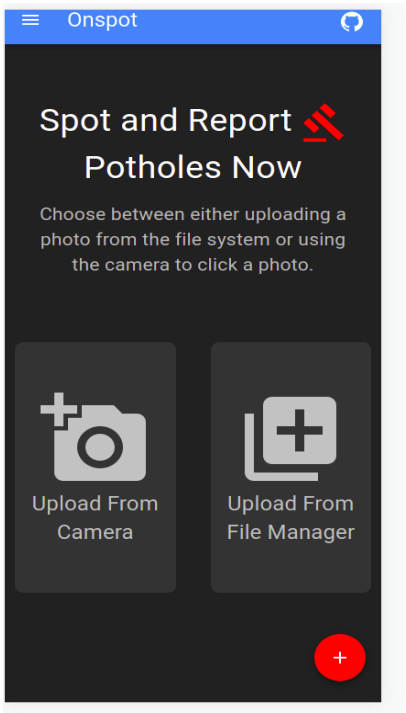


Fig. 10.2

Fig. 10.3

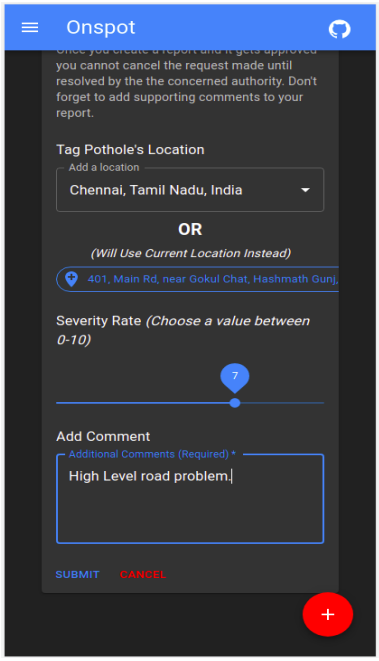


Fig. 10.4

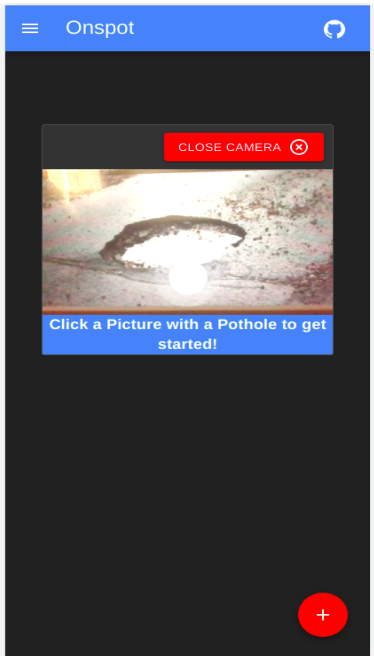


Fig. 10.5

10.2 My complaints dashboard to view the status of his complaint.

- On signing into the app with their Google Accounts, users are presented with the My Complaints Screen.

- Existing users who have at least one report created can manage the status of their report and add additional comments to it or reply to comments from authorities using this section of the App.

- Users can click on any of the reports to view the detailed description for the report and monitor any notifications or to communicate with the authority through the chat section.

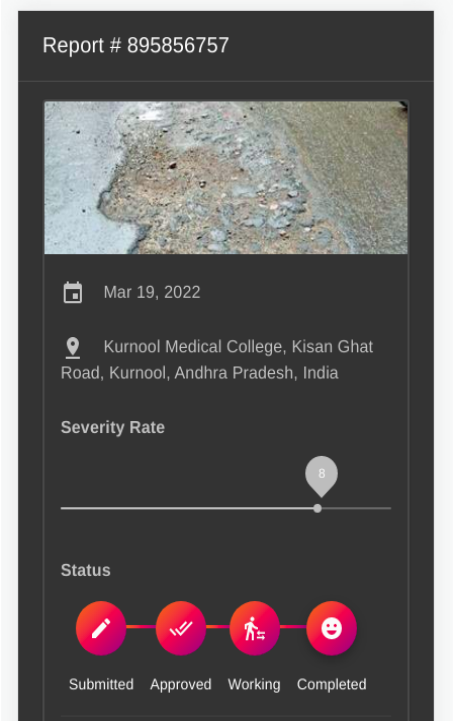


Fig. 10.6

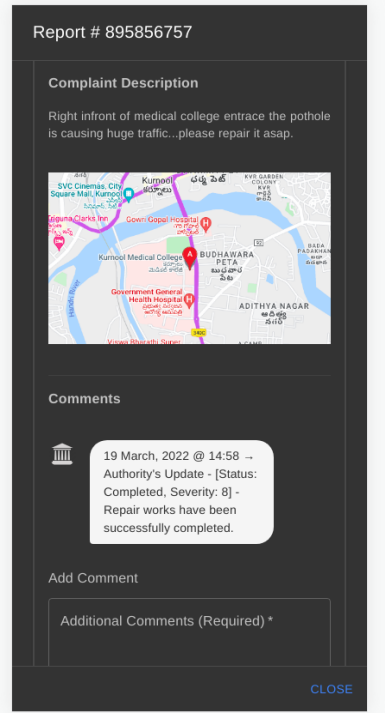


Fig. 10.7

Fig. 10.8

10.3 Route navigation to help users detect potholes in the respective route for which he wishes to travel.

- This screen presents the user with an option to enter the source and destination location.

- After which a map is presented to the user based on the source and destination location entered by the user. A direction route is rendered on the map for the user.

- It displays the route with the marked potholes on the path which were approved by the authorities. It uses Google's Maps API to build the route.

- Custom markers for potholes with status approved ranging on severity are presented to the user on the route if they are close to the route.

- It determines if a pothole is associated with a path using the 'isLocationOnEdge' library function provided by Google Maps.

- A legend explaining the different attributes of the map is presented to the user.

- Also, there is a street view renderer for the users.

- This screen can be used by any logged in user to monitor a route and the status for the different potholes on it and plan travel accordingly.

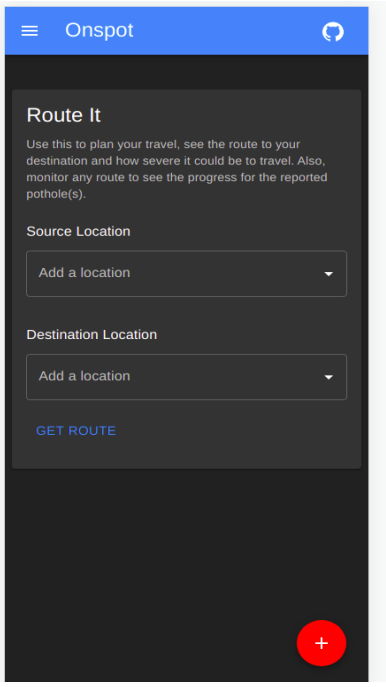


Fig. 10.9

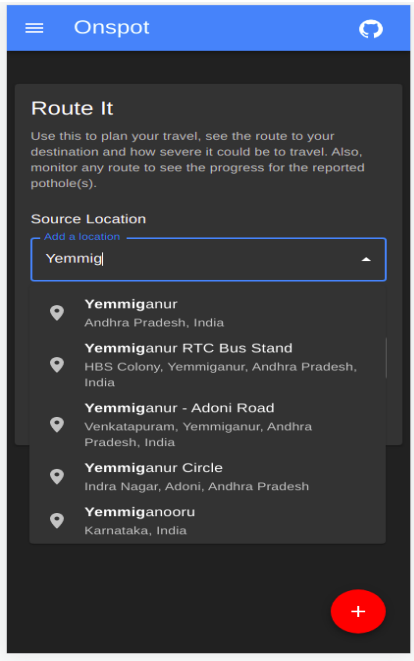


Fig. 10.10

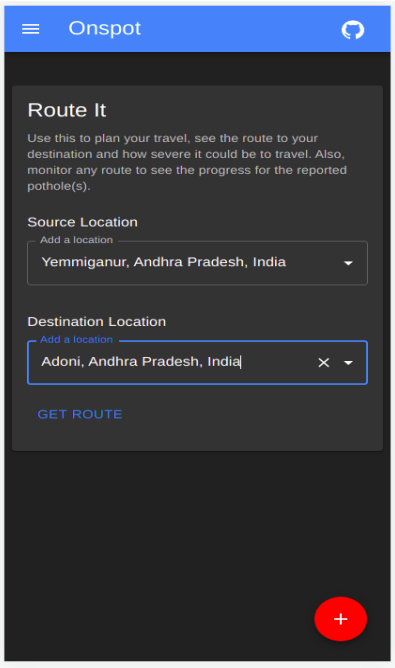


Fig. 10.11

Fig. 10.12

Fig. 10.13

10.4 Profile for viewing basic user details such as name, email,etc.

- This screen contains the basic details (avatar, name, email address of the user)
- It is then followed by a counter for reports with a status of either submitted, approved or completed.
- Based on the above counters the user is assigned with a badge score indicating their contribution to the community.
- This score is a weighted average score based on the counters mentioned above.
- This score can later be used for rewarding the user. (Later)

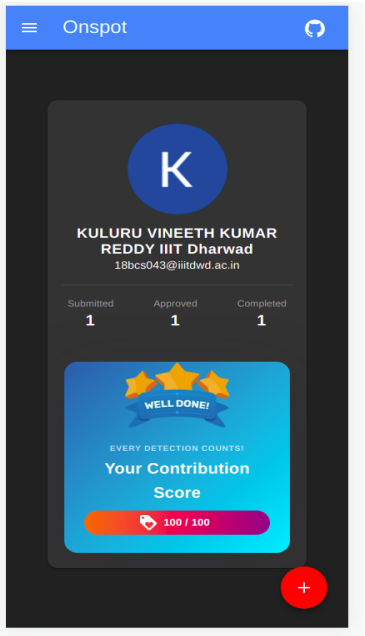


Fig. 10.14

10.5 Sign in screen which uses the oauth google api for necessary authentications for security purposes.

- This screen contains the option to login using Google.
- This uses Google's OAuth 2.0 GAPI for logging in the user.
- This also uses the Unsplash API for generating random backgrounds on the side. (When in desktop mode)
- The app also uses local storage actively to maintain the session state every time in communication with GAPI.
- OAuth2.0 Unsplash Logout Local Storage.

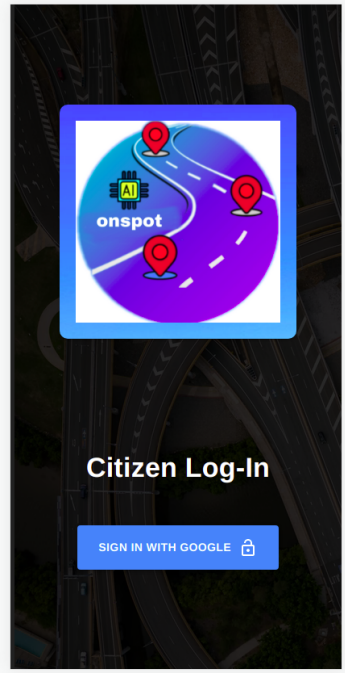


Fig. 10.15

Github Repo link for source code reference: [ONSPOT_Citizen]

Chapter-11

Authority's Application Features

11.1 A Dashboard for displaying an analytics of user reports based on the status of severity. An option for also viewing reports based on filters.

- Once the authority has signed in successfully, they are presented with the dashboard screen which is divided into two main sections.

- The first being the analytics section. This section contains two charts.

- The first is a donut chart which shows a high level view of the user reports in the authority's zone based on the status (approved, submitted, in progress, etc).

- Next to that is the stacked bar chart which provides a more detailed view of the status, but this time based on severity as well.

- The second section on the dashboard is a filterable data table containing details about the various user reports. This datatable has options for sorting, filtering, selecting specific columns, searching, exporting the data as csv and printing the table contents too.

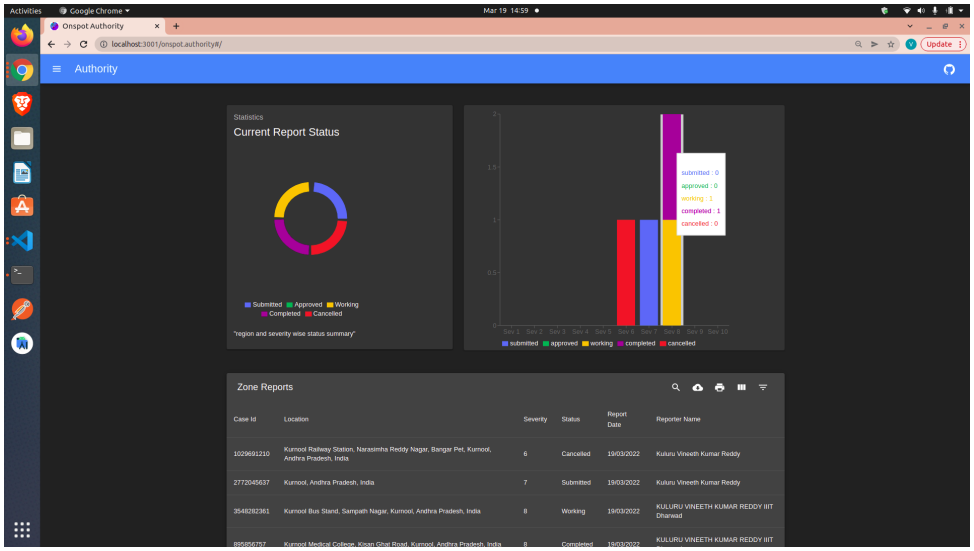
- On clicking any report in the datatable, the authority is presented with the option to update it (status and severity). On pressing update they are asked to describe their reason

for update in a required custom description box.

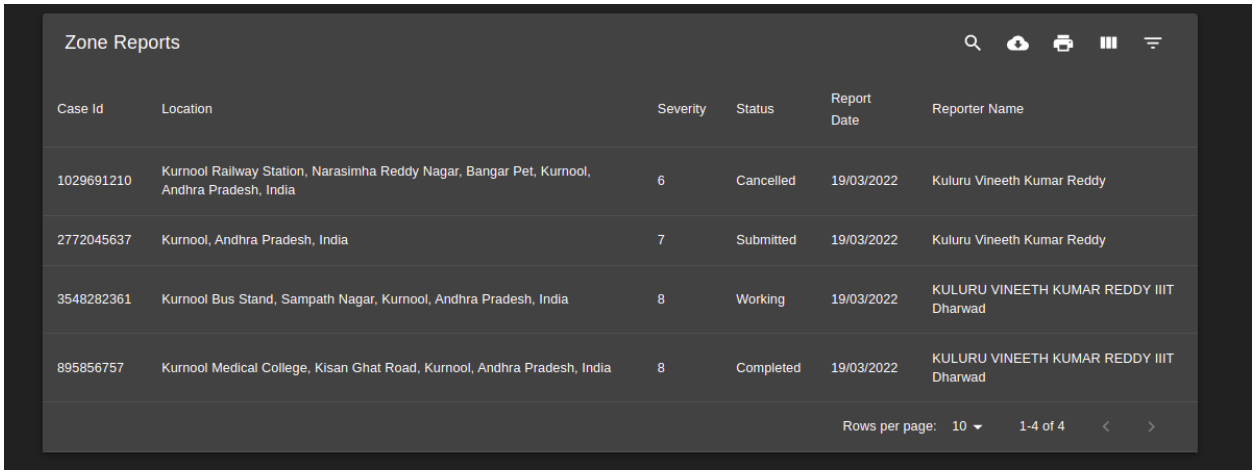
- Authorities can also comment on the report they clicked on and view all the other details of the report as a user would in their My Complaints screen.

- Users are sent notifications through email after any update is made on their report.

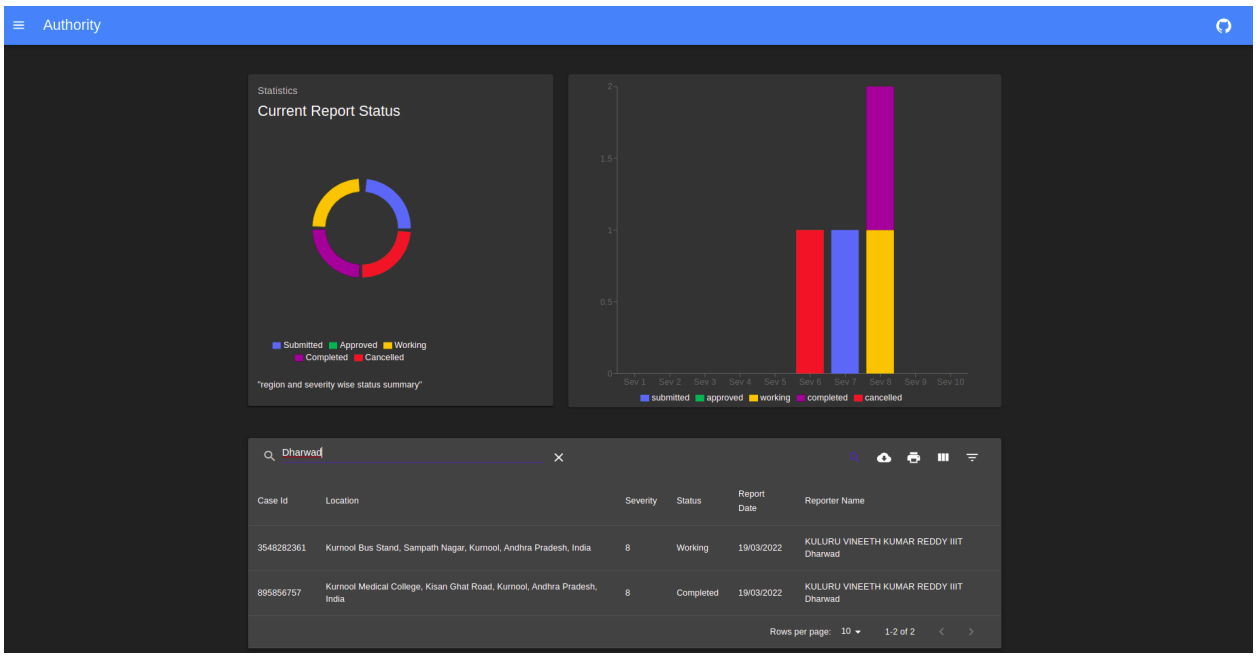
11.1.1 Analytics Section (Authority Dashboard)



11.1.2 Filterable Reports Data table (Authority Dashboard Zone Wise)



11.1.3 Filter Active Users(Authority Dashboard)



11.2 An interactive Map region view to see the potholes in their region.

- Another screen in the authority app is the map region view section. Here they can view potholes reported in their region through an interactive map view.

- Existing users who have at least one report created can manage the status of their report and add additional comments to it or reply to comments from authorities using this section of the App.

- A legend is displayed for helping understand the map better.

- Custom markers for potholes with different status values ranging on severity are presented to the authority on the map.

- Authorities can click on any of the markers to view the information window for it. On pressing the view button in the info window authorities are presented with the view and update detailed report section.

- They can update the status and severity directly from this view after clicking the button in the info window. On pressing update they are asked to describe their reason for update in a required custom description box.

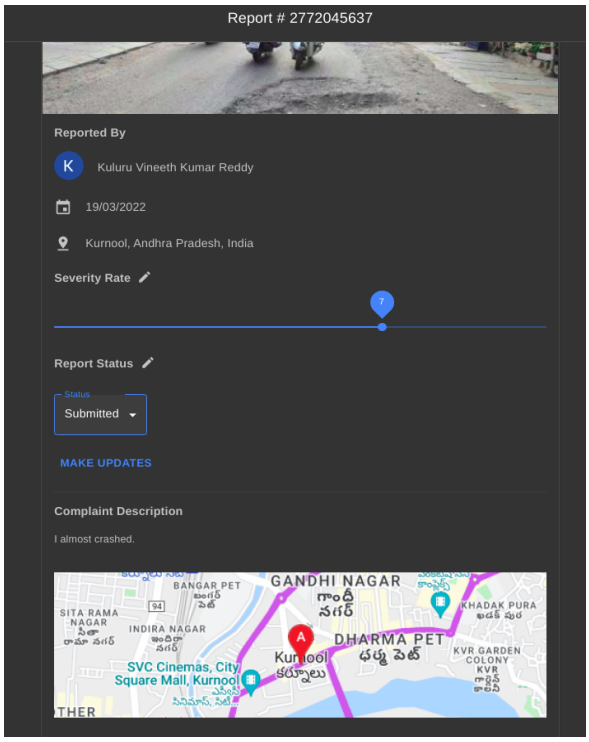
- Authorities can also comment on the report they clicked on and view all the other details of the report as a user would in their My Complaints screen.

- Users are sent notifications through email after any update is made on their report.

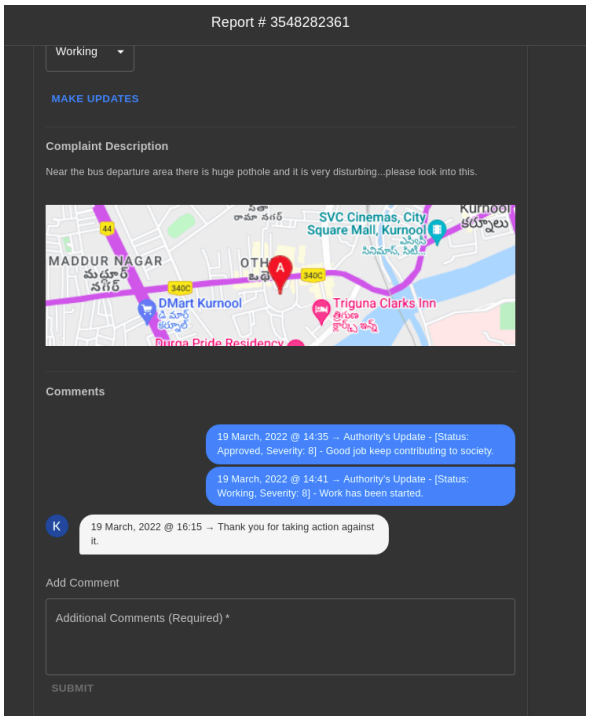
- Also, there is a street view renderer for the users.

- Finally there is an option to toggle on and off a heat map layer which is weighted on the severity of potholes reported in the region.

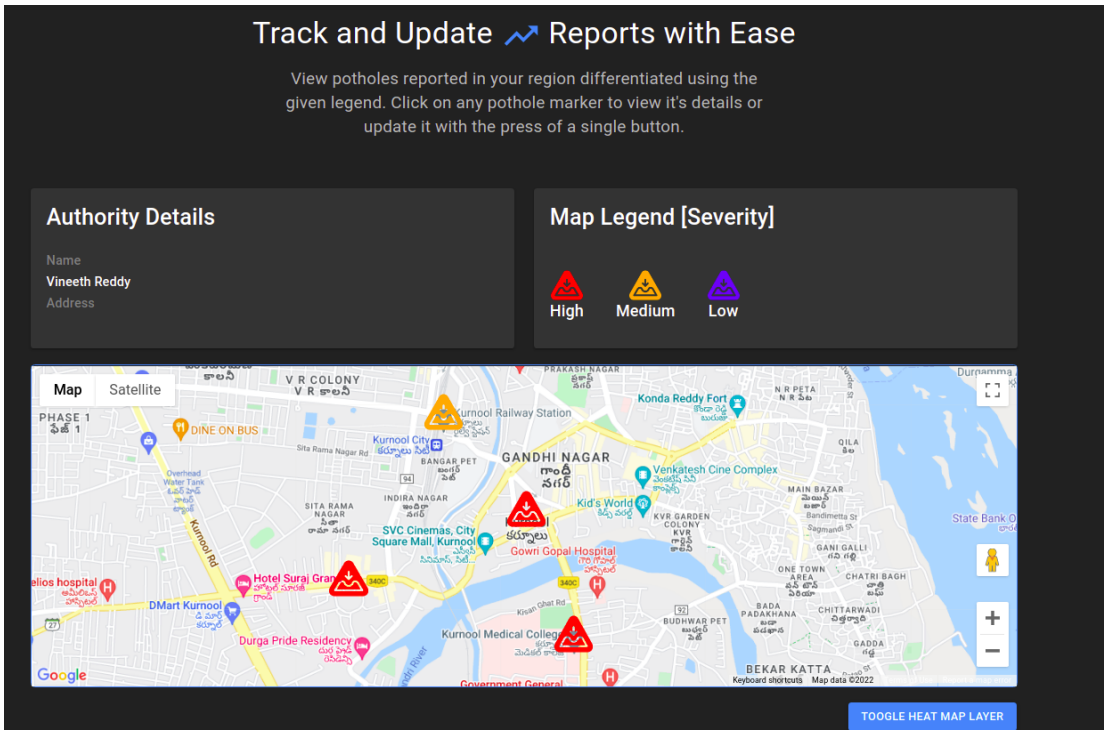
11.2.1 View and Update Report (Authority, Map Region View)



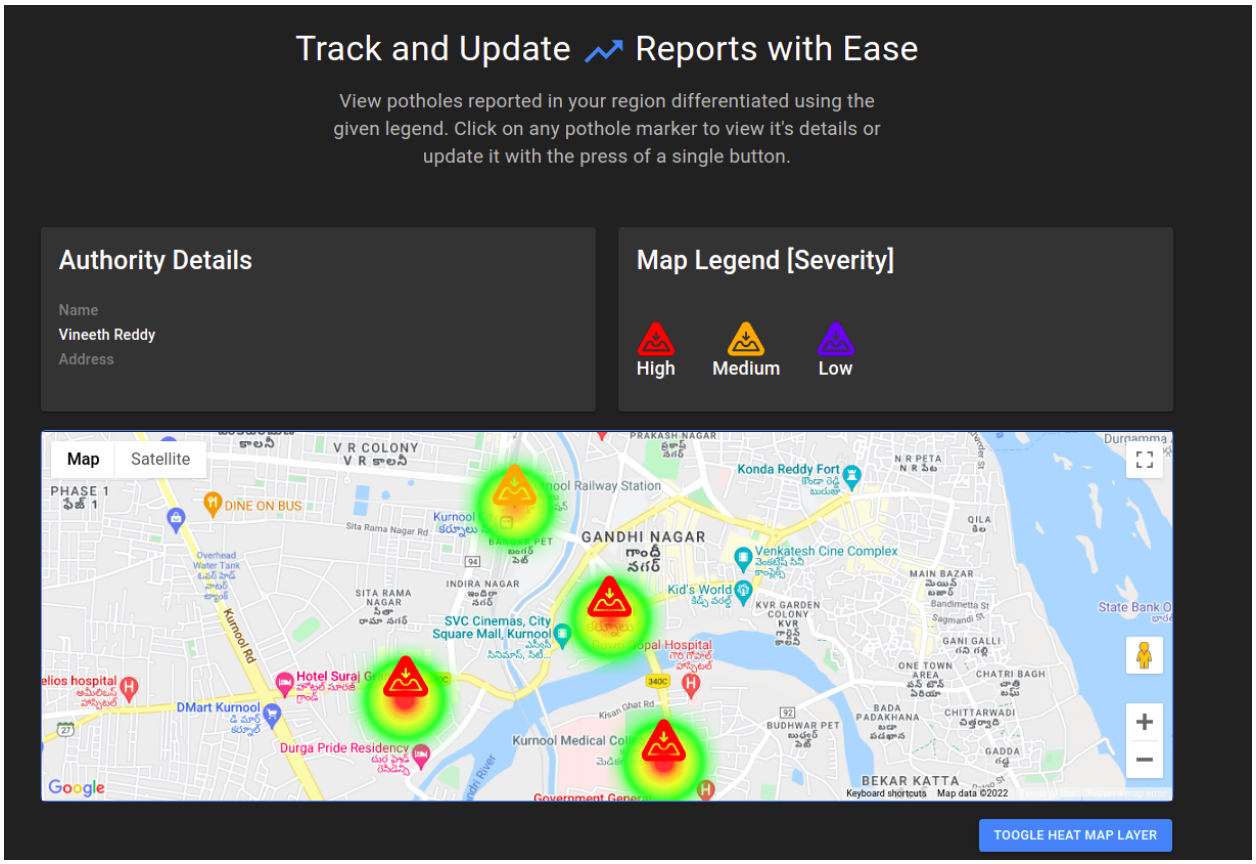
11.2.2 Comments Section and Static Map(Report)



11.2.3 Map Region View (Zone Reports, Severity Based)



11.2.4 Heat Mapping Layer



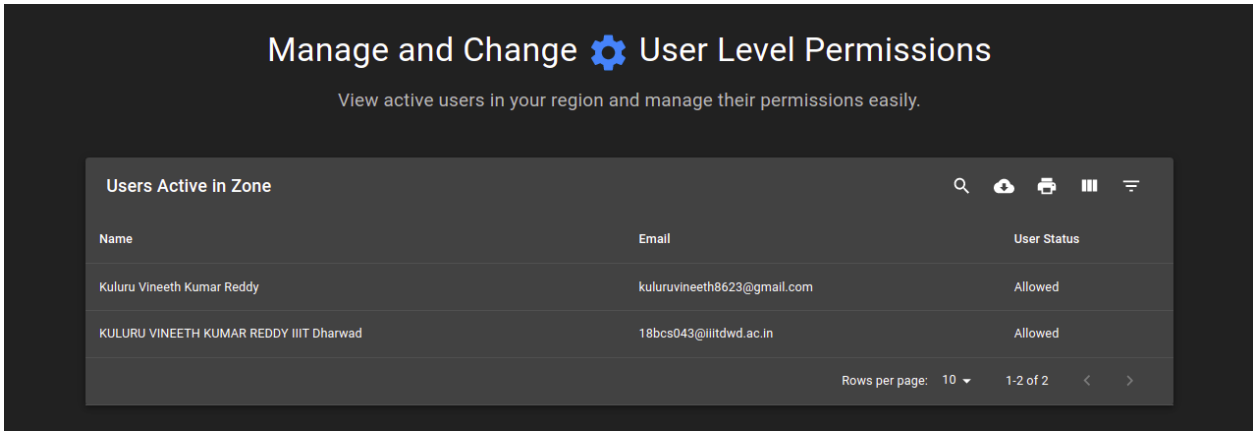
11.3 Manage users to view user details on the reported potholes and provide necessary updates.

- This screen presents the authority to manage all active users in their zone through a filterable data table.
- The data table contains options to filter, search, sort, select columns.
- They can view their basic profile details.

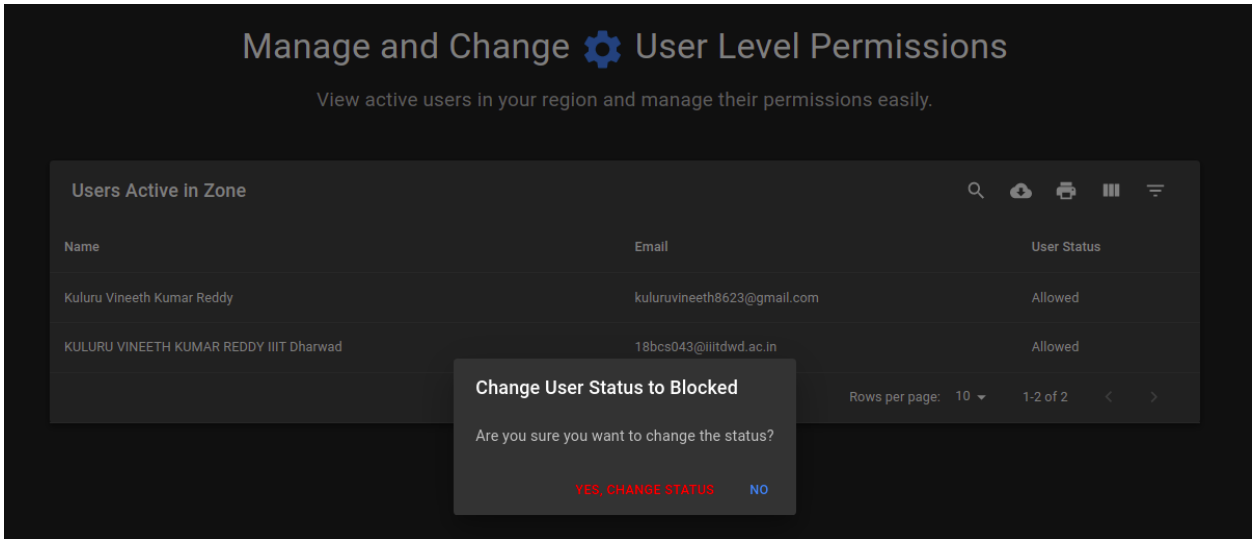
- They can update their status to either blocked or allowed based on their activity too.

- Users are sent out email notifications regarding updates in their status every time an authority makes a change.

11.3.1 Manage Users Section



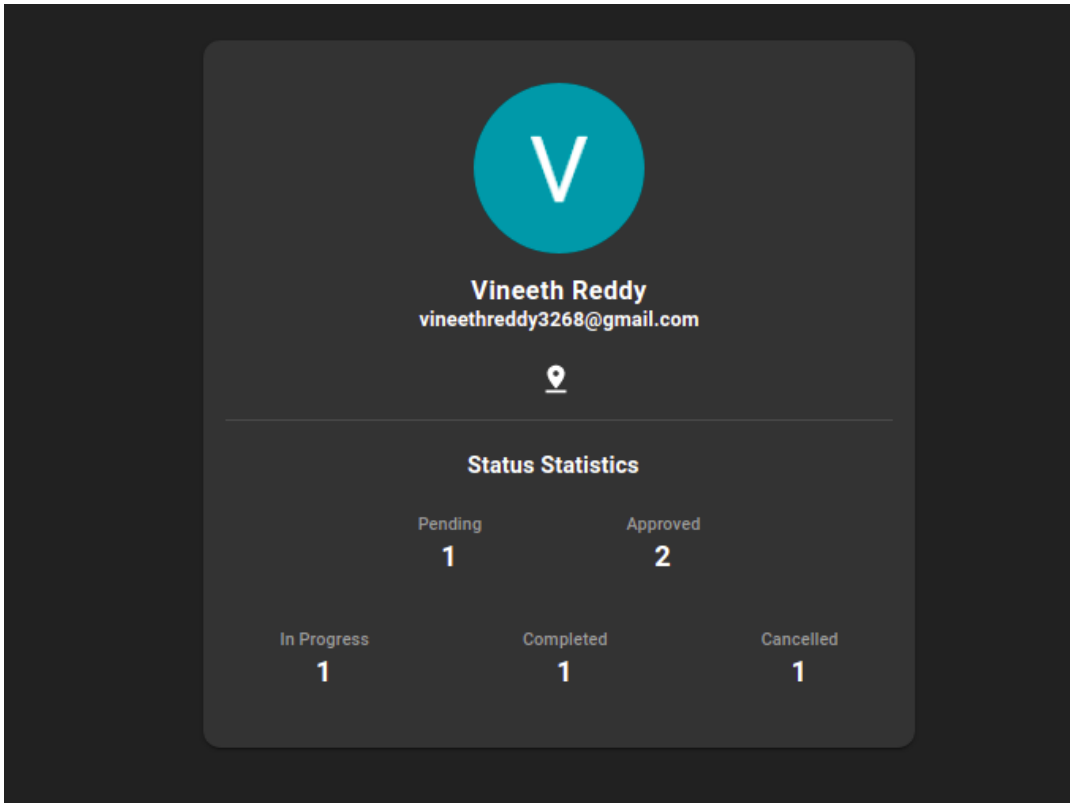
11.3.2 Block / Unblock Status



11.3.3 Profile section to view basic details and number of reports approved, in progress or submitted.

- This screen contains the basic details (avatar, name, email address of the user)
- It also contains the address of the authority.
- It is then followed by a counter for reports in the authority's zone with a status of either pending, in progress, approved, completed or cancelled.

11.3.4 Authority's Profile (User Report Statistics)



11.3.5 Sign in screen which again uses google oauth apis and stand alone database to authorize authority logins for security purposes.

- This screen contains the option to login using Google.
- This uses Google's OAuth 2.0 GAPI for logging in the user.
- This also uses the Unsplash API for generating random backgrounds on the side. (When in desktop mode)
- The app also uses local storage actively to maintain the session state every time in communication with GAPI.
- OAuth2.0 Unsplash Logout Local Storage.
- An important point to note is that in the authority's app the oauth api has been linked to a standalone database to authorise authority log-ins.

Github Repo link for source code reference: ONSPOT_Authority

11.4 The core backend:

The backend is going to compose of 3 main components:

11.4.1 An object detection model for validating the pics taken by the user for the reporting process.

- As the focus of the application was to create a rest api to automate the process of pothole validation with media files, from the beginning itself a cloud server was used for

implementation. An aws ec2 instance would be used for this purpose.

- The model would be trained on top of Darkflow and built on top of pretrained weights which can be obtained from Darknet.

- For crawling images relevant to our label ‘pothole’, images would be crawled using the open source google image search package, along with using the serapis image search tool.

- In addition to this, freely available pothole video feeds to create the dataset would also be used.

- After cloning the DarkFlow repository, to prepare the input files for DarkFlow we need to consider two things. Firstly, we need an RGB image which is encoded as jpeg or png and secondly, we need a list of bounding boxes (xmin, ymin, xmax, ymax) for the image and the class of the object in the bounding box.

- Our class, in this case, is ‘pothole’. We then need to label our images with a tool like LabelImg to identify areas of interest with bounding boxes. LabelImg is a graphical image annotation tool that is written in Python and uses Qt for the graphical interface. The annotations are saved as XML files in the Pascal VOC format We can split the data to train and test sets before running the training command.

- Then we can accordingly train and test our model.

11.4.2 A Mysql database for creating and managing reports. Also for communication between the user and the authority.

- The database would be composed of 4 main entities. **(Authority, Public User, Reports, Comments)**.

- The authority is someone that has the authorization to manage reports in their zone. A zone is specified as a 5000 mts radius.

- A public user is someone that can create a report using the citizen app.

- A report is basically details of a submission that has been validated through the object detection model.

- Comments are basically communication exchanges between an authority and a user on a report.

11.4.3 A Flask api for data exchange between the frontend and the backend.

- The Flask API would be built for data exchange between the front end applications, the object detection model and the database.

- An app controller script would be created to communicate with the service layers (mysql, mail service and parent (object detection).

Github Repo link for source code reference: ONSPOT_Backend

Chapter-12

Conclusion And Future Scope

12.1 Conclusion:

From the implementation of this project, we have been able to address the problem of lack of communication, collaboration and coordination both from the Citizens as well as Authority perspectives to uphold and maintain the infrastructure of our Roads. Also proper transparency on the actual road maintenance and pothole complaint and repair features have been achieved. With this we have also tried to tackle the problem of increasing death tolls due to road accidents in our country. This project has also to some extent been able to provide accurate route navigation to its users/citizens based on the severity of the potholes reported for the specific routes.

While building and developing this complete end to end application, we have had the privilege to get exposed to a wide range of tools and technologies. This project has also helped us think and brainstorm extreme scenarios and come out with feasible solutions. Overall, we have gained a good experience and vast knowledge and the ability to work and collaborate in a team.

12.2 Future Scope:

Although the solution proposed has taken a lot of possibilities even during diverse situations into account some of the features such as region based translation of the application, Ability to make the Citizens communicate among each other using comments and also some of the below mentioned advanced features are proposed as a future scope for people looking forward to working in this project and improvising it.

- Severity based Direction Renderer.
- Locality based language translation.
- Adding Additional Parameters to the Frontend after Implemented in the Backend. (Pothole Dimensions)
- Offline Capabilities for the camera component.
- Realtime Notifications in the application itself.
- Linking via Social Platforms like twitter, facebook, etc.
- News Feeds
- Interaction using a gamified approach to challenge peers using rewards.

Chapter-13

References

1. Eriksson, Jakob & Girod, Lewis & Hull, Bret & Newton, Ryan & Madden, Samuel & Balakrishnan, Hari. (2008). The Pothole Patrol: Using a mobile sensor network for road surface monitoring. MobiSys'08 - Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services. 29-39. 10.1145/1378600.1378605.

2. Maeda, Hiroya & Sekimoto, Yoshihide & Seto, Toshikazu. (2016). Lightweight road manager: smartphone-based automatic determination of road damage status by deep neural network. 37-45. 10.1145/3004725.3004729.

3. Agarwal, Swati & Mittal, Nitish & Sureka, Ashish. (2018). Potholes and bad road conditions: mining Twitter to extract information on killer roads. 67-77. 10.1145/3152494.3152517.

4. Kawasaki, Takafumi & Kawano, Makoto & Iwamoto, Takeshi & Matsumoto, Michito & Yonezawa, Takuro & Nakazawa, Jin & Tokuda, Hideyuki. (2016). Damage Detector: The Damage Automatic Detection of Compartment Lines Using A Public Vehicle and A Camera. 53-58. 10.1145/3004010.3004017.

5. Chaaban, Khaled, Mohamed Shawky, and Paul Crubille. "A distributed framework for real-time in-vehicle applications." In Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005., pp. 925-929. IEEE, 2005.

6. https://www.roadmetrics.ai

7. Hoang, Nhat-Duc. "An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction." Advances in Civil Engineering 2018 (2018).

8. Ukhwah, Ernin Niswatul, Eko Mulyanto Yuniarno, and Yoyon Kusnendar Suprapto. "Asphalt pavement pothole detection using deep learning method based on YOLO neural network." In 2019 International Seminar on Intelligent Technology and Its Applications (ISITIA), pp. 35-40. IEEE, 2019.

9. Ryu, Seung-Ki, Taehyeong Kim, and Young-Ro Kim. "Image-based pothole detection system for ITS service and road management system." Mathematical Problems in Engineering 2015 (2015).

10. Madli, Rajeshwari, Santosh Hebbar, Praveenraj Pattar, and Varaprasad Golla. "Automatic detection and notification of potholes and humps on roads to aid drivers." IEEE sensors journal 15, no. 8 (2015): 4313-4318.

11. ImageNet Large Scale Visual Recognition Challenge, 2015.

12. https://arxiv.org/abs/1409.0575

13. Selective Search for Object Recognition.

14. https://link.springer.com/article/10.1007/s11263-013-0620-5

15. Rich Feature Hierarchies For Accurate Object Detection And Semantic Segmentation, 2013.

16. https://arxiv.org/abs/1311.2524

17. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 2014.

    https://arxiv.org/abs/1406.4729

18. Fast R-CNN, 2015.

19. https://arxiv.org/abs/1504.08083

20. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,

2016

21. https://arxiv.org/abs/1506.01497

22. Mask R-CNN, 2017.

23. https://arxiv.org/abs/1703.06870

24. You Only Look Once: Unified, Real-Time Object Detection, 2015.

25. https://arxiv.org/abs/1506.02640

26. YOLO9000: Better, Faster, Stronger, 2016.

27. https://arxiv.org/abs/1612.08242

28. YOLOv3: An Incremental Improvement, 2018.

29. https://arxiv.org/abs/1804.02767

30. @reactjs - https://reactjs.org/

31. @material-design-react - https://material-ui.com/

32. @react-google-maps - https://www.npmjs.com/package/react-google-maps

33. @google-maps-api - https://developers.google.com/maps/documentation/javascript/

34. @google-oauth-gapi - https://developers.google.com/identity/protocols/oauth2

35. @mui-treasury - https://mui-treasury.com/

36. @axios - https://www.npmjs.com/package/axios

37. @filepond - https://www.npmjs.com/package/filepond

38. @dateformat - https://www.npmjs.com/package/dateformat

39. @loadash - https://lodash.com/

40. @create-react-app - https://reactjs.org/docs/create-a-new-react-app.html

41. @gh-pages - https://www.npmjs.com/package/gh-pages

42. @github - https://github.com

43. @trello - https://trell.com

44. @figma - https://figma.com

45. @google-keep - https://keep.google.com

46. @npmjs - https://www.npmjs.com

47. @snap2html - https://www.rlvision.com/snap2html/about.php