International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Geospatial Visualisation of devices used for security in Gated Communities

Srinivas Prabhu B[1], Kendaganna Swamy[2]

[1]*Student, Dept. of Electronics and Instrumentation Engineering, R.V. College of Engineering, Karnataka, India*
[2]*Assistant Professor, Dept. of Electronics and Instrumentation Engineering, R.V. College of Engineering, Karnataka, India*

Abstract - *Housing a population as huge as 1.2 billion people isn't an easy task. With rapid urbanisation underway, the most sought after in the Indian housing sector are the Gated Communities. There are several reasons for their rising popularity - safety, security, services etc. However, as a community grows in size, each of these promised reasons pose severe challenges, more so in communities where a gate guard is armed only with a ledger and a telephone. MyGate, a company dealing with security solutions for gated communities provides sustainable solutions for the challenges faced. The heart of all these solutions is a smartphone called the Guard Device, with which the guards are able to monitor the community. Visitor Management, Delivery Management, Child Security, Maintenance Payments are amongst the several solutions the Guard Device helps out with. The importance of these Guard Devices makes their health status an important feature to monitor. This project implements a dashboard which monitors the health status of these guard devices using Geospatial Visualisation. This implementation will be carried out with the help of ReactJS and LeafletJS.*

Key Words: **Geospatial Visualisation, Gated Communities, MyGate, LeafletJS**

1. INTRODUCTION

India has a huge population of 1.2 billion people. With Rapid Urbanisation underway in the country, India's residential sector is expected to grow significantly in the coming decade. The exponential growth in population and urbanisation is gearing India's housing sector towards Gated Communities. What are Gated communities? They are residential communities enclosed by closed boundaries. They are the most sought out in the residential sector due to the plethora of amenities they offer, but mostly due the security they provide.

Security and privacy being the key reasons for residents to choose gated communities, the current state of their implementation is obsolete. How can a security guard be entrusted with the security of several hundred residents, with just an intercom system and a ledger? Larger the size of the community, the larger the problems faced by the security management. The outdated nature of security implementation, and lack of other management features cannot be very promising for residents.

To tackle these problems, a start-up called MyGate has come up with an unique app based system. This solution offers several security features like Delivery Management, Visitor Management, Domestic Help Management, Security Alerts and several others. MyGate platform also provides community management features like communication, management and accounting to help gated communities function smoothly.

MyGate implements its community management features using an app-based system. This app based system is implemented using 2 app interfaces. The first app interface is the MyGate app, which is available for free on the Play Store and App Store. This app implements all the security features MyGate provides. For example, it provides the user a 6-digit security code which can be sent to a certain visitor. Similarly, it provides other features like inter-community communication, house help details and finances on the app.

The second app interface is the Guard App. This application is loaded on the Guard Devices which are used by the security guards. This application acts as the service-side application. The app helps the guards contact any particular resident, register their visitor using their 6-digit code, register a new visitor on the app and perform many such features. This app-based interface makes the guards' work easier and faster. The Guard App along with all the features, also collects information regarding the status and location of the Guard Devices.

The Guard Devices are the heart of all the security solutions MyGate provides. These devices are just normal smartphones with the Guard App installed on them. They help the guard interact with the visitors, the delivery personnel and other guests. This makes their role extremely important. Therefore these devices must be healthy, and if not, they need to be fixed as soon as possible. To implement this feature, this project deals with displaying and monitoring the health status of these guard devices all over the country on a Map using Geospatial Visualization.

2. RELATED WORKS

Related works for this project include research under security as well as technological research.

Security Systems

CCTV cameras are installed to instill fear amongst criminals, and prevent any damages psychologically. However, not all societies can implement CCTV cameras, due to their high cost of initial setup and maintenance[1].

One of the most important security features is visitor management. An advancement to the use of a ledger can be QR Code systems [1], [2]. However, QR codes can be risky since scanning a malicious QR code can lead to disastrous situations.

Several Gated communities use WhatsApp as a medium to send in their complaints to the security management. To prevent this, an app based complaint system was developed [3].

Geospatial Visualisation

Geospatial Visualisation is used to locate and identify objects over a geographical domain, in a particular Region of Interest (ROI). This concept is used to build a dashboard to achieve disaster mitigation and management as seen in [4].

OpenStreetMap

OpenStreetMap provides open-source spatial datasets in map form or in XML/JSON format. Using OpenStreetMap(OSM), Road Perception of Autonomous vehicles was improved as seen in [5]. Similarly, OSM is used to provide information regarding buildings in a particular location as seen in [6].

3. SOFTWARE REQUIREMENTS

Several softwares are required for the project implementation. They are split into several divisions based on the area of requirement.

3.1 Frontend Languages and Frameworks:
The frontend development required the following softwares:

1. HTML and CSS
2. ReactJS
3. React-leaflet and LeafletJS
4. OpenStreetMap

3.2 Backend Languages and Frameworks:
The backend development required the following softwares:

1. Java Spring Boot
2. NestJS
3. RabbitMQ

3.3 Database:
This project uses MongoDB - A NoSQL based Document Database. MongoDB performs extremely well for spatial data [7] and is generally better than SQL databases [8].

3.4 IDE and other tools:
The implementation of the project requires certain tools and IDE's. They are:

1. Visual Studio Code
2. IntelliJ
3. Postman

4. METHODOLOGY

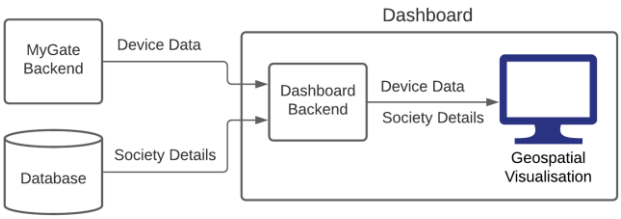

Fig -1: Block Diagram

The methodology of the geospatial visualisation consists of the MyGate Backend, Database, and the Dashboard, as shown in the block diagram. These components are explained below:

4.1 MyGate Backend

This backend system is the primary backend service in MyGate. The real-time data from the Guard Devices after processing is received by this backend. This data is received in the form of metrics like Battery, Hard Disk, RAM Utilisation and Page Count. The status of each of these metrics are received from the device and a calculation is performed based on thresholds to find the health of the particular device. Once the data for each device is received, every device is provided a health status that is either GOOD, MEDIUM, or BAD. The real-time data of all devices in a society is then converted to an average of all device healths in the society based on a calculation shown in the flowchart below. For the ease of understanding, let's call the average of all device healths in a society as "Society Health".

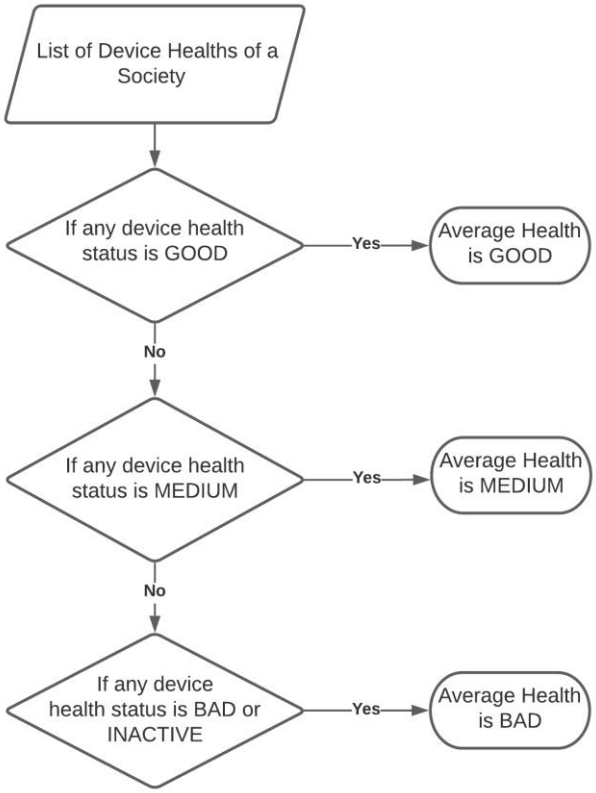

Fig -2: Flowchart for Society Health Calculation

According to the flowchart given above, average health of all devices in a society is performed. The calculation is based on the principle that, if any device in the society is of GOOD health, then the average will be taken as GOOD. This procedure is applied to all societies once the data is received from the Guard Devices. The MyGate backend performs all these operations and then sends this Society Health data over to the Dashboard Backend. For the geospatial visualisation, this real-time data has to be passed down to all instances of the Dashboard.

4.2 Database

The database used is a NoSQL based document-oriented database, which consists of all the details of societies. However, the Geospatial Visualisation requires only certain data like ID, Name, Latitude and Longitude. In accordance with the above mentioned fields, all society data is sent to the dashboard backend. This information is constant and won't undergo any changes. Therefore, this information is also cached on the client side to prevent heavy database read operations.

4.3 Dashboard

The main component is the Dashboard component. It consists of the backend and the frontend. The backend receives the real-time data from the MyGate backend and the static society data from the Database. The frontend implements the geospatial visualisation based on the data it receives from the Dashboard backend.

5. IMPLEMENTATION

This project is a web based implementation. Therefore it consists of two sections, the backend implementation and the frontend implementation.

5.1 Frontend Implementation

The frontend is responsible for the geospatial visualisation, and is implemented using 2 important JavaScript libraries, ReactJS and React-leaflet. ReactJS is an open source JS library used to create single-page web applications, and breaking complex UI into simple reusable UI components. React considers everything as a component, and therefore making it highly reusable and readable. Few optimisation techniques were also implemented as discussed in [9].

React-leaflet is the React wrapper around an important mapping library called LeafletJS. LeafletJS is an open source JS library which displays interactive maps. React-leaflet generates the same map for a ReactJS frontend. The map used for geospatial visualisation is provided by OpenStreetMap. It's an open source alternative to Google Maps, and is extremely crucial for the implementation of the frontend in this project.

The implementation of the frontend involves adding the markers of every society on its location on the map. Each marker color defines the health of the devices in the society.

Features of frontend implementation:

1. Color Coding: Markers of each society are assigned a color based on its health.
    a. GOOD - Green
    b. MEDIUM - Blue
    c. BAD - Red
2. Popup: Every marker once clicked shows a popup which provides information particular to the society - ID, Name, Address and Health
3. Filtering: LeafletJS provides a function to add layers to the map. Using layers, filtering can be performed to display only societies with a particular health status.
4. Clustering: Based on the closeness of markers in a region, clustering can be added to the map. This feature can be added to every layer of the map making it easier to find clusters of GOOD, MEDIUM or BAD health.

5. Geocoding: This feature converts a text based address to a pair of latitude and longitude. The map will then orient itself with these coordinates as its center.

5.2 Backend Implementation

The backend implementation involves 2 parts: the main MyGate Backend, and the Dashboard Backend.

5.2.1 MyGate Backend

This backend is the main backend service on the MyGate platform. This service performs several operations. The operations with respect to the project are given below:

Guard Device Data collection

The data from the Guard Devices are sent by the Guard App installed on those devices. This data is received in the form of metrics like Battery, Hard Disk, RAM Utilisation and Page Count. The status of each of these metrics are received from the device and a calculation is performed based on thresholds to find the health of the particular device. This same calculation is done for every device in the society and a GOOD, MEDIUM, BAD health status is provided to every device. A similar calculation is performed for the Society Health calculations.

The Guard Devices send the data over to the main backend over REST API's.

Publishing to Message Queues

Once the Society Health is calculated, the same data is sent over to every instance of the Device Dashboard using RabbitMQ message queues. A fanout exchange is used to publish the data onto all the existing queues. Similarly the Society Health data is also updated in the database for further usage.

Similarly, the backend implementation for the calculations and serving the data to RabbitMQ is based on the Java Spring Boot.

5.2.2 Dashboard Backend

This backend is the backbone of the Dashboard. It provides the information to be displayed on the dashboard. It performs several tasks as given below:

Subscribing to Message Queues

Every time a change is observed in the health of the society, the data is published onto all queues bound to the fanout exchange. This data is received by all the instances of the Dashboard backend.

Static Data

Dashboard backend serves static data like Society ID, Name, Location to the frontend from the database, and caches the same for a particular period. This implementation is based on REST API's.

Dynamic Data

Once the real time data is received from the Message Queues, the data is passed onto the frontend in a stream using the concept of SSE (Server-Sent Events). This provides an unidirectional stream of data to the frontend as soon as it is received by the backend. This helps the frontend in implementing the real-time functionality. The entirety of the Dashboard Backend is implemented on a NestJS backend.

6. RESULTS AND DISCUSSION

The website performs as expected, with geospatial visualisation used to visualise society healths on a map, with real-time changes. The testing for the platform was performed in two stages, Unit Testing and Integration Testing. In Unit Testing, every unit of the entire platform was tested individually after development. However, all these microservices should perform in sync when integrated. This is where Integration Testing helps in verifying the application. Integration Testing checks the operation of the entire application once all the backend and frontend implementations are integrated. The results of Integration testing are tabulated and shown in the figure given below.

Description	Sample Input	Expected Output	Test Result
Verify whether all society healths are displayed on the map	Load the URL	All the society markers should be displayed with appropriate colors	PASS
Verify whether the markers provide society details when clicked	Click on a marker	Society details like ID, Name should appear on a Popup above the clicked marker.	PASS
Verify whether filtering displays only societies of a particular health status	Click on GOOD, MEDIUM, or BAD filtering checkbox	Only societies with a particular health should display markers.	PASS
Verify if the markers are clustered when zoomed out	Zoom out	Clustering markers into one marker with a number indicating the number of clustered markers	PASS

Fig -3: Test Results

7. CONCLUSION

Gated communities require a security upgrade, and MyGate provides the security solutions required with the help of a Guard Device. Due to the importance of these devices, their health status needs to be constantly

monitored. The proposed application was designed in order to monitor and track the statuses of Guard Devices in a gated community. This project implements the overview of all societies using Geospatial visualisation. The most important objective when it comes to frontend of the website is ease of use, and responsiveness. The entire system is also real-time, i.e. the operations are performed nearly instantly. Sooner the issues in the devices are recognized, sooner they can be rectified.

Different softwares and dependencies can be used for the implementation. However, understanding how softwares interact with each other forms an integral part of the development. Therefore, if chosen softwares don't link well, scalability of the project drops drastically. However, in this project, these softwares form an important technological stack (MERN Stack), i.e. each software has dedicated support to link to another. This not only helps in scalability, it also improves the longevity of this project. Therefore, this application is scalable, real-time, easy to use and responsive.

8. FUTURE SCOPE

The implementation of this project involves data received and processed by several backends to be displayed in the frontend in real-time. However, with this much influx of data over time, it would be very beneficial to perform data science and machine learning on the data. It is very likely that there are trends in the data we don't notice firsthand, but some analysis and oversight can prove to be beneficial for the application. Similarly, machine learning can be used for anomaly detection in the data.

This project is a commercial application which will be used by a team of engineers and salespeople. However, it is now only available in the Desktop version. Future scope in this field would be a responsive native mobile application for easier usage.

Technology is forever growing, and we must adapt to the changes as they come. For this, the application must be scalable, yes. But it also needs to be maintained whenever newer technologies are rolled out, so that the application remains future proof.

REFERENCES

[1] M. N. H. A. Hassan, M. H. Jumali, and D. P. Dahnil, "Enhancement of access features for a gated system in a guarded community," in 2019 IEEE Conference on Wireless Sensors (ICWiSe), IEEE, 2019, pp. 24–28.

[2] A. Jarali, S. Kodilkar, S. Patel, S. Tondare, and G. Kudale, "Digintry securing gated premises using qr-code," in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), IEEE, 2019, pp. 1115–1121.

[3] N. A. Razak, U. Ujang, S. M. Salleh, S. Azri, and T. Choon, "Development of mobile application for gated and guarded community management," ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 4216, pp. 17–22, 2019.

[4] K. K. Lwin, Y. Sekimoto, W. Takeuchi, and K. Zettsu, "City geospatial dashboard: Iot and big data analytics for geospatial solutions provider in disaster management," in 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), IEEE, 2019, pp. 1–4.

[5] Y. Zheng and I. H. Izzat, "Exploring openstreetmap capability for road perception," in 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 1438–1443.

[6] M. R. Rifat, S. Moutushy, S. Ishtiaque Ahmed and H. Shahid Ferdous, "Location based Information System using OpenStreetMap," 2011 IEEE Student Conference on Research and Development, 2011, pp. 397-402, doi: 10.1109/SCOReD.2011.6148772.

[7] D. Laksono, "Testing spatial data deliverance in sql and nosql database using nodejs fullstack web app," in 2018 4th International Conference on Science and Technology (ICST), IEEE, 2018, pp. 1–5.

[8] C. Gyo˝ro¨di, R. Gyo˝r¨odi, G. Pecherle, and A. Olah, "A comparative study: Mongodb vs. mysql," in 2015 13th International Conference on Engineering of Modern Electric Systems (EMES), 2015, pp. 1–6. doi: 10.1109/EMES.2015.7158433.

[9] A. Javeed, "Performance optimization techniques for reactjs," in 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2019, pp. 1–5. doi: 10.1109/ICECCT.2019.8869134.

[10] M. Ismail, A. Tarek, P. M. Plaza, D. M. Gomez, J. M. Armingol, and M. Abde- laziz, "Advanced mapping and localization for autonomous vehicles using osm," in 2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES), IEEE, 2019, pp. 1–6.

[11] S. Just, K. Herzig, J. Czerwonka, and B. Murphy, "Switching to git: The good, the bad, and the ugly," in 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), 2016, pp. 400–411. doi: 10.1109/ISSRE.2016.38.

[12] L. Liang, L. Zhu, W. Shang, D. Feng, and Z. Xiao, "Express supervision system based on nodejs and mongodb," in 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, pp. 607–612. doi: 10.1109/ICIS.2017.7960064.