A Framework for Detection and Reporting Traffic Violation using Distributed Intelligent Agents

Project Team

Sl. No.	Reg. No.	Student Name
1	17ETCS002026	ANIMESH SRIVASTAVA
2	17ETCS002049	CY SHREEYA
3	17ETCS002050	CHAITANYA PRIYA KJ
4	17ETCS002012	ADITI SHARMA

Supervisors: Mr. PRAKASH P

JULY – 2020

B. Tech. in Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

M. S. Ramaiah University of applied sciences

Bengaluru -560 054

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "A Framework for Detection and Reporting Traffic Violation using Distributed Intelligent Agents" is a bonafide work carried out in the Department of Computer Science and Engineering by ***Mr. Animesh Srivastava*** *bearing Reg. No.* ***17ETCS002026*** *in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

July – 2020

Mr. Prakash P
Assistant Professor - Department of CSE

Dr. Raghavendra V. Kulkarni
Professor and Head – Dept. of CSE

Dr. M. Arulanantham
Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "A Framework for Detection and Reporting Traffic Violation using Distributed Intelligent Agents" is a bonafide work carried out in the Department of Computer Science and Engineering by ***Ms. CY Shreeya*** *bearing Reg. No.* ***17ETCS002049*** *in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

July – 2020

Mr. Prakash P
Assistant Professor - Department of CSE

Dr. Raghavendra V. Kulkarni
Professor and Head – Dept. of CSE

Dr. M. Arulanantham
Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

*This is to certify that the Project titled "A Framework for Detection and Reporting Traffic Violation using Distributed Intelligent Agents" is a bonafide work carried out in the Department of Computer Science and Engineering by **Ms. Chaitanya Priya KJ** bearing Reg. No. **17ETCS002050** in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

July – 2020

Mr. Prakash P
Assistant Professor - Department of CSE

Dr. Raghavendra V. Kulkarni
Professor and Head – Dept. of CSE

Dr. M. Arulanantham
Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "A Framework for Detection and Reporting Traffic Violation using Distributed Intelligent Agents" is a bonafide work carried out in the Department of Computer Science and Engineering by ***Ms. Aditi Sharma*** *bearing Reg. No.* ***17ETCS002012*** *in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

July – 2020

Mr. Prakash P

Assistant Professor - Department of CSE

Dr. Raghavendra V. Kulkarni

Professor and Head – Dept. of CSE

Dr. M. Arulanantham

Professor and Dean-FET

Declaration

A Framework for Detection and Reporting Traffic Violation using Distributed Intelligent Agents

The project work is submitted in partial fulfilment of academic requirements for the award of B. Tech. Degree in the Department of Computer Science and Engineering of the Faculty of Engineering and Technology of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name	Signature
01	17ETCS002026	ANIMESH SRIVASTAVA	
02	17ETCS002049	CY SHREEYA	
03	17ETCS002050	CHAITANYA PRIYA KJ	
04	17ETCS002012	ADITI SHARMA	

Date: July 2020

Acknowledgements

The successful completion of this project could not have been possible without the guidance of Mr. Prakash P, our mentors, whose mentorship was indispensable and also for providing their invaluable guidance, comments and suggestions throughout the course of the project.
A debt of gratitude is also owed to Dr. HM Rajashekara Swamy and Dr. PVR Murthy for their constant, unwavering support.
We would also like to extend our sincere gratitude to everyone who have been a part of this project for their guidance that was required to complete this project. Lastly, I would like to thank everyone not mentioned here but without whose direct or indirect support this project could not have been completed.

Summary

In the new evolving world, traffic rule violations have become a central issue for the majority of the developing countries. Due to migration from villages, semi-urban to urban, the number of vehicles on road is increasing rapidly. Due to the increase in the number of vehicles, negligence of traffic rules and accidents caused due to traffic violations are also on the rise because of no proper supervision. Many traffic violations go unnoticed and the violator gets away with no penalty. Therefore, the proposed system uses intelligent agents which are distributed across the city such that it detects and gathers data of the violation by communicating with other agents/vehicles. This ensures that no traffic violation goes unnoticed and eliminates detection of violation manually and also promotes a corruption free environment.

The actual number of traffic violations committed without being recorded is very huge compared to those recorded by the traffic patrols. Even with all the efforts made by the government to curb traffic violations, there are certain challenges due to which the number of traffic violations are still on the rise. The major challenge is the lack of traffic patrols and the increasing number of streets and roads where the proportion of the number of patrols to the roads and streets is very low. Creating an environment with one on one supervision is not feasible. Therefore, the project emphasises on the fact that it is necessary to create an automated detection of traffic violation using distributed agents across all vehicles to create a distributed network where violation is detected and reported as and when it occurs.

The model is a framework to curb traffic violations and also implement a more organised, reliable and unerring method of collecting fines from the wrongdoer. The main aim of this project is to eliminate manual traffic monitoring so that the patrols can be assigned with other important duties and to cope with the limited man power and corruption.

Table of Contents

List of Tables

List of Figures

1. Introduction

Migration from village, semi urban to urban areas has increased the population. There are many challenges due to migration, among those, increased usage of vehicles is a leading concern. India is among the countries with the largest road networks in the world. The roads are overburdened with vehicles almost all the time. With the increasing number of vehicles, violations of traffic rules are increasing rapidly and so are the accidents caused by it. It is difficult to keep an eye on every other vehicle for violation. It is a very tedious task since the manpower available to detect violations to maintain the traffic flow is not enough at present. Even if a violation of rule is detected, you may not be able to stop it and as a result the violator goes without a penalty which is not a good example to set for the law and order of a democratic country like India. The lives of innocent people are also at risk and the violators should not be entertained for the same. To be able to detect as many violations as possible is the need of the hour.

1.1 Introduction to the framework

The framework uses distributed agents present in different vehicles to detect and report traffic violations. The main objective of the project is to create agents which are uniquely identified by their agent ID, make them intelligent enough to understand the different violations it has to detect and report. The agents are designed in such a way, that it detects a rule violated by a different agent if present in the vicinity. Certain rules are given a certain buffer before it has been reported. If the rule is still being violated even after the end of the buffer, a report will be sent to the server to update the database connected to the RTO. This information can only be accessed by the RTO and cannot be meddled with by the user. This makes sure that there is no direct interaction between the detecting and reporting system with the user but only with a violation reporting notification system installed on the vehicle which gives information about the time, type and the location of the violation committed. The same information will be updated in the RTO database as well. This ensures that traffic violations are detected and reported without any human interaction and meddling.

1.2 Problems solved by the proposed framework

Detecting violations through distributed intelligent agents does not let violators go unnoticed and a penalty is charged irrespective of the circumstances. It detects if the driver is wearing the seat belt or not, if the vehicle is moving in the wrong direction, if the insurance is valid or if the car is driving beyond the speed limit. It ensures if the violation has been rectified or not after an interval which eventually leads to a lesser number of accidents. Manpower expected is minimized. Corruption free system is established since human interaction is eliminated. Drivers are forced to obey the traffic rules because they cannot escape the consequences of violating the rules. It builds a dataset which can be used for future studies, surveys, predictions etc. It helps in maintaining the law and order.

1.3 Scope of the project

This framework is solely applicable to traffic violations committed by vehicles. Almost all kinds of vehicles can be detected. In the real world, the distributed agents constantly look for vehicles violating one of the mentioned traffic violations (no seat belt, uninsured vehicle, over speeding and driving in the wrong direction). The distributed agents are notified if a violation is committed and further a warning is given to the driver real quick along with a time interval to rectify it. If not rectified the driver is provided with an e-penalty and punished accordingly.

1.4 Applications of the project

This framework is solely applicable to traffic violations committed by vehicles. Almost all kinds of vehicles can be detected. In the real world, the distributed agents constantly look for vehicles violating one of the mentioned traffic violations (no seat belt, uninsured vehicle, over speeding and driving in the wrong direction). The distributed agents are notified if a violation is committed and further a warning is given to the driver real quick along with a time interval to rectify it. If not rectified the driver is provided with an e-penalty and punished accordingly.

1.5 Assumptions

- All agents are connected to the same network.
- It is assumed that the user has already bought the car and agents are installed.
- A database of 10 people is assumed for simulation purposes.
- A single area with limited co-ordinates is considered for simulation purposes. Speed limits and direction are assumed in the roads. The area considered is Rajajinagar, Bengaluru.
- A buffer time of 10 minutes is considered as 10 seconds and 15 minutes is considered as 15 seconds for simulation purposes. For seatbelt, assuming that the engine has already started in the beginning and later based on a random number it can turn off.
- The car’s speed is assumed to be fetched using a speedometer for now. In future if there is any new technology to detect speed in an efficient way then we can use the same technology.
- It is assumed that the driver has seen the speed limit board sign and maintains his speed accordingly.
- For simulation purposes, insurance is checked from year 2018 to 2021.

1.6 Features of the project

- The ability to detect violations real quick sets this framework apart from all other methods of detecting traffic violations.
- Being a distributed system and negligible dependency on manpower, the number of vehicles going unnoticed and unpunished can be brought down rapidly.
- Since all the data related to vehicles and violations committed are recorded, it can be used for studies, surveys in the future.

2.Background Theory

2.1 Literature Survey

Table 1: Literature review

Sl. No	Author	Name and year of publication	Research focus	Techniques and technologies	Conclusion
01	Prof. GG Chiddarwar Amey Narkhede Vikrant Nikham Abhishek Sathe Akshay Soni	Automatic traffic rule violation detection and number plate recognition, 2017 [1]	To detect traffic signal violations using image processing	Edge detection algorithm Hough transform algorithm K-nearest neighbour algorithm	The system described, detects violations with the help of IR cameras and image processing algorithms. The number plate of the violating vehicle will be identified and then reported to the respective authorities.
02	Robert Ciolli Andrew Mack Peter Whyte	Automated traffic violation monitoring and report system, 2003 [2]	To detect traffic rule detection using image processing along with a notification system	Intersection camera system Data processing system	The proposed system uses image processing algorithms to detect traffic violations made at road intersections. The reporting system of these violations is efficient. The number

	Gurchan Ercan			Police department interface system Charge Couple Device (CCD)	of violations that will be detected are limited.
03	Yenikaya, Gokhan	Automatic Detection of Traffic Rule Violations, 2005 [3]	Automation of traffic monitoring system with the help of digital maps	RAM unit (placed inside the vehicle) GPS module GSM module	The paper proposes a system that does not use image processing to monitor traffic and the concept of installing a RAM unit inside the vehicle that will keep track of the violations being made is emphasized.
04	Samir A. Elsagheer Mohamed	Automatic Traffic Violation Recording and Reporting System to Limit Traffic Accidents, 2019 [4]	To detect traffic violations using vehicular Ad-hoc network	VANET DGPS module	The proposed system gives an idea on how traffic violations can be detected automatically inheriting properties of VANET. Applicability of IoT in such

					a system is suggested for further improvement.
05	D. R. Agrawal Kasliwal Komal S. Gandhi Labhesh R. Deore Punam D. Saitwal Pooja D	Android based Traffic Rules Violation Detection System Vehitrack, 2017 [5]	To detect traffic violation and traffic density using sensors and cameras to detect RFID tag number using a RFID reader.	Sensors RFID reader RFID tag technology	The paper emphasises on assigning a RFID tag number to every vehicle which can be read by a reader when a violation is detected. Whereas, documentation of actual detection and reporting of the violation is not available.

- **Automatic traffic rule violation detection and number plate recognition, 2017**

The system defined in paper [1] is as follows: a connection is established between the 2MP camera with integrated infrared system and Arduino for desktop, which will be placed at a zebra crossing. After this, the number plate of the vehicle is identified with the help of the image processing algorithms mentioned in Table.1. Depending on whether the violation occurred or not, an SMS alert is sent to the user as well as the RTO.

The system works quite well however, there is still room for improvement. The camera used in the system for this project is sensitive to vibration and fast changing targets due to the long shutter time. The system speed can be increased with a high resolution camera. The character recognition method is sensitive to misalignment and to different sizes, so the affine transformation can be used to improve the character recognition from different sizes and angles. The statistical analysis can also be used to define the probability of detection and recognition of the vehicle number plate. At present there are certain limits on parameters like speed of the vehicle, script on the vehicle number plate, skew in the image which can be removed by enhancing the algorithms further.

- **Automated traffic violation monitoring and report system, 2003**

The system described in the given paper [2] consists of a digital camera system deployed at a traffic location. The camera system is remotely coupled to a data processing system. The data processing system comprises an image processor for compiling vehicle and scene images produced by the digital camera system, a verification process for verifying the validity of the vehicle images, an image processing system for identifying driver information from the vehicle images, and a notification process for transmitting potential violation information to one or more law enforcement agencies.

The system is efficient but there can be improvements. The above system works well for traffic violations that can be captured or identified by the camera like skipping traffic signals, going in the wrong direction, etc. But it cannot be used for detection of violations like over speeding, insurance expiry, drink and drive, etc; the violations that cannot be identified using cameras. Problems with the camera can pose as a difficulty to track the violations that might occur.

- **Automatic Detection of Traffic Rule Violations, 2005**

The methodology used in paper [3], relates to a method and system for automatically detecting traffic rule violations and applying the respective penal procedures. There have been several traffic monitoring systems in use with the main aim of detecting the violations correctly without any human interactions. Here the automation is done using digital maps, which contain the data of all the roads of a particular geographical area and the information on each roads of that particular region (GIS database) and available satellite-based communication systems and hardware (GPS and GSM). There will be a RAM unit in each vehicle to be traced, containing all the necessary information like traffic signal positioning, speed information about a particular road. Accordingly, a controller which will receive geographical data from the GPS. Data from the GPS and the data from the RAM unit will be compared continuously and if there is a violation, the GSM transmitter in the vehicle will report the violation to the required authorities and a notification in the form of an SMS or email will be sent to the driver by means of a cellular network.

- **Automatic Traffic Violation Recording and Reporting System to Limit Traffic Accidents, 2019**

The above paper [4], implements an embedded system which is installed in each car can be integrated with the computer of the vehicle. It consists of several modules such as DGPS receiver; and database of the streets and roads (in the form of digital maps); the traffic rules information (e.g. speed limits and if the road is unidirectional or bidirectional); a wireless transceiver; VANET software stack; and the software programs to operate the system. Using the DGPS, the on-board unit can obtain in real time and with high accuracy the current position of the vehicle and hence it can calculate the vehicle's direction (displacement); its speed; and its acceleration or deceleration. As a result, using simple software, many kinds of traffic violations can be detected. The detected violations will be stored in the local memory of the on-board unit and will be used by the other systems.

- **Android based Traffic Rules Violation Detection System "Vehitrack", 2017**

The paper [5] discusses about developing an android application which maintains a database related to capture image through camera as proof and RFID Tag number such as name of owner, address of owner, license number, photo of vehicle user, mobile number, their bank account number and also the list of previous rules broken with image as proof, date and time and fine paid by vehicle owner. All the data about the vehicle will be displayed on a smart phone of traffic police. The rules violated are detected by means of sensor, RFID reader and RFID tag technology and capture image by means of camera. The system will control the traffic density of the specified location. The application automatically receives the fine from the owner's bank account and send the message to the user mobile application or about the number the rules and their fine. If the same vehicle is found to be flouting rules repeatedly then a specific action could be done.

2.2 Keywords

The meaning of the words that repeatedly occur in the report are listed below:

Image processing - processing digital image by means of a digital computer to get enhanced image either to extract some useful information.

Traffic violation - violation of the law committed by the driver of a vehicle.

GPS - Global Positioning System, used to track location.

Sensors - a device that detects or measures a physical property or that responds to it.

2.3 Merits and Demerits

The merits and demerits of the papers on the existing automated traffic violation detection systems, that were studied are listed below:

Merits:

- The violations are detected and reported with acceptable accuracy and efficiency.
- In addition to designing a framework for traffic violation detection, a secure system is designed which allows secure position verification, the authentication, data integrity, confidentiality, and privacy for the wireless ad-hoc networks and the VANET itself.

Demerits:

- In the systems that use image processing to detect the violations, if the images are not captured properly then the algorithms will not identify the images. Image processing is time consuming.
- In the systems that RAM units are installed in the vehicles to keep track of the violations, all the units and modules that are being used in the system are expensive. The systems need to be installed in every new and existing vehicle for this system to work.
- All the papers mentioned do not necessarily detect and report all traffic violations that might occur. A framework or a model is designed to detect and report traffic violations takes into consideration only a couple of them.

2.4 Demonstration of related work and originality

The papers that were studied for the literature review focussed on detecting the traffic rules violation using image processing, installing RAM units, using VANETS or installing computers in the car to track the violations. None of the papers gave a solution for the detection of all kinds of violations. In all the systems, detection was either done by another system installed at certain parts of the road or by the device installed in the vehicle which is violating the rules. This project is unique from all the papers that were studied, here, intelligent agents that are equipped with GPS and a communication model, will be installed in the cars, but violation detection will be done through another nearby agent on the road and that agent will report it to a central server that is accessible to all the agents, which will then report it to the RTO and notify the user about the violation and fine details. So in this way, all the agents will be monitoring all the nearby agents. The aim behind making agents report other agents is to avoid the user tampering with the device that is installed in the user's car to avoid being reported for the violations being made by the user.

`

3. Aim and Objectives

Title

A framework for detection and reporting traffic violation using distributed intelligent agents

Aim

Elimination of detecting violations manually, corruption free agents and builds dataset which can be used for future prediction

Objectives

- To gather information about traffic violations from government rules and regulations documents and to do literature surveys on available solutions to detect traffic violations.
- To model agents to understand traffic violations by adapting rules.
- To implement the agents to understand the traffic violation and integrate all the modules.
- To simulate the detections and reporting by deploying/ distributing the designed agents.
- Documentation of the whole project after each and every objective is achieved.

Methods and Methodology

Table 2: Methods and Methodologies used to fulfil the objectives

Objective No.	Statement of the Objective	Method/ Methodology	Resources Utilised
1	To gather information about traffic violations from government rules and regulations documents and to do literature surveys on available solutions to detect traffic violations.	**1.1:** Studied IEEE and research papers and patents to understand the existing solutions. **1.2:** Studying the gathered traffic violation rules from the government of India. **1.3:** Understood the framework and architecture from the existing solutions.	1. IEEE papers 2. International Journals 3. Google Scholar 4. Patents
2	To model agents to understand traffic violations by adapting rules.	**2.1:** Put forth the functionality of the proposed project. **2.2:** Design the flowchart or block diagram of the model. **2.3:** Design sender model, receiver model and communication model of the agents.	1. Dia, to make the flowcharts and the framework

3	To implement the agents to understand the traffic violation and integrate all the modules.	**3.1:** Use agent class for implementation. **3.2:** Implementing the necessary modules required by the system. **3.3:** Integrating all the modules of the system.	1. Java, Netbeans IDE 2. MySql database 3. Object oriented programming concepts
4	To simulate the detections and reporting by deploying/ distributing the designed agents.	**4.1:** Gathering prerequisite information and applying it on the system to get the appropriate output. **4.2:** To test the agents with different possible cases.	1. Socket programming 2. Single server, multiple clients 3. Object oriented programming concepts
5	Documentation of the whole project after each and every objective is achieved.	**5.1:** Develop a scientific project report as per the template specified. **5.2:** Documentation of framework of the model throughout the development stages **5.3:** Document test cases used during the simulation of the model.	1. Microsoft word 2. Google docs

4. Problem Solving

4.1 Design of the model

Framework design of the model:

The basic idea is to have agents in vehicles (newly registered cars). All these agents together constitute a distributed network. The distributed network has an agent monitoring system which gets a report from each agent of the rule violated (recognised by an ID) by other vehicles and the agent ID of the same. Each agent is uniquely identified by an ID. After the agent monitoring system receives a report, it is sent to the RTO which already has a database from which the violation ID and the agent ID is matched to its violation and vehicle is matched respectively. After this, the corresponding fine is deducted from the person's account with a notification sent to the person on the android app. This android application will only have the initial login credentials and will show notifications to the user when they have violated any rule and the amount deducted from the account for doing so.

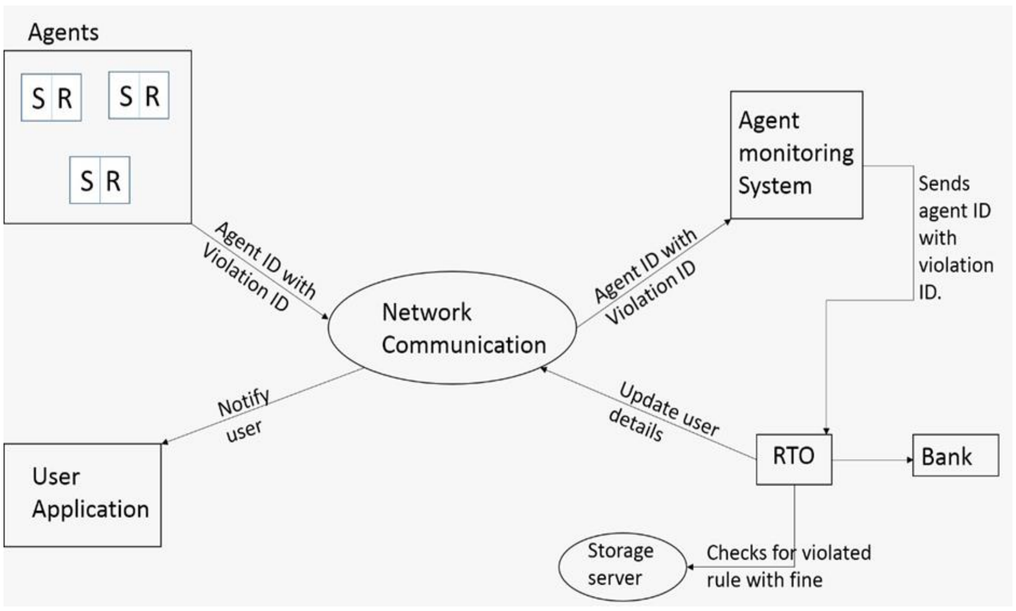


Figure 1: Framework of the model

- Agents: The agents have a receiver and a sender end. The receiver end is for receiving reports from agents who have violated rules. The sender end is to send a report consisting of violation ID and the agent ID who has violated a rule to the agent monitoring system. The agent has to have information about the violation that is under consideration and should be able to detect that in other vehicles.
- A database of streets and roads is primarily fed to the agents with information about the rules it has to detect and report. The information about the rules may be information about speed limits and if a particular road is unidirectional or bidirectional. If any of the rules is violated, it is reported to the agent monitoring system with the violation ID and gent ID.

Design of agents and communication among them:

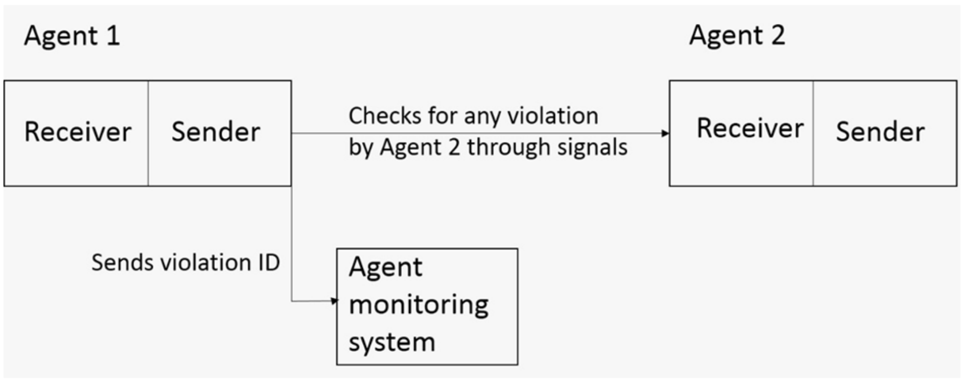


Figure 2: Design of communication between agents

- Agent monitoring system: The agent monitoring system is responsible for receiving the report consisting of the agent ID and the rule the agent has violated in the form of a violation ID. It is also responsible for reporting the committed violation to the RTO. From this report sent the RTO will be able to take necessary actions for the committed violation.
- RTO: The RTO receives the violation ID and the agent ID who has committed the violation. The RTO matches the violation ID with the violation in their storage server with matching the agent ID with the owner of the vehicle. The RTO then updates the details of the fine that will be deducted from the owner's account to the user.
- User application: The user application is responsible for notifying the user with the violation committed with the fine that is deducted from the user's account. The user

application is only an interface that informs the user after the fine is deducted and nothing else. The application is installed initially only with basic credentials.

Class diagram to show static view of the model:

Structure of the model is shown in figure 3. Diagram shows a client server based model where Agent acts as client. Agent is super class for classes- SeatBelt, WrongDirection, OverSpeeding and UninsuredVehicle. gps is a subclass of wrongDirection and overSpeeding class and reportingAgent class is a subclass of SeatBelt, WrongDirection, OverSpeeding and UninsuredVehicle.

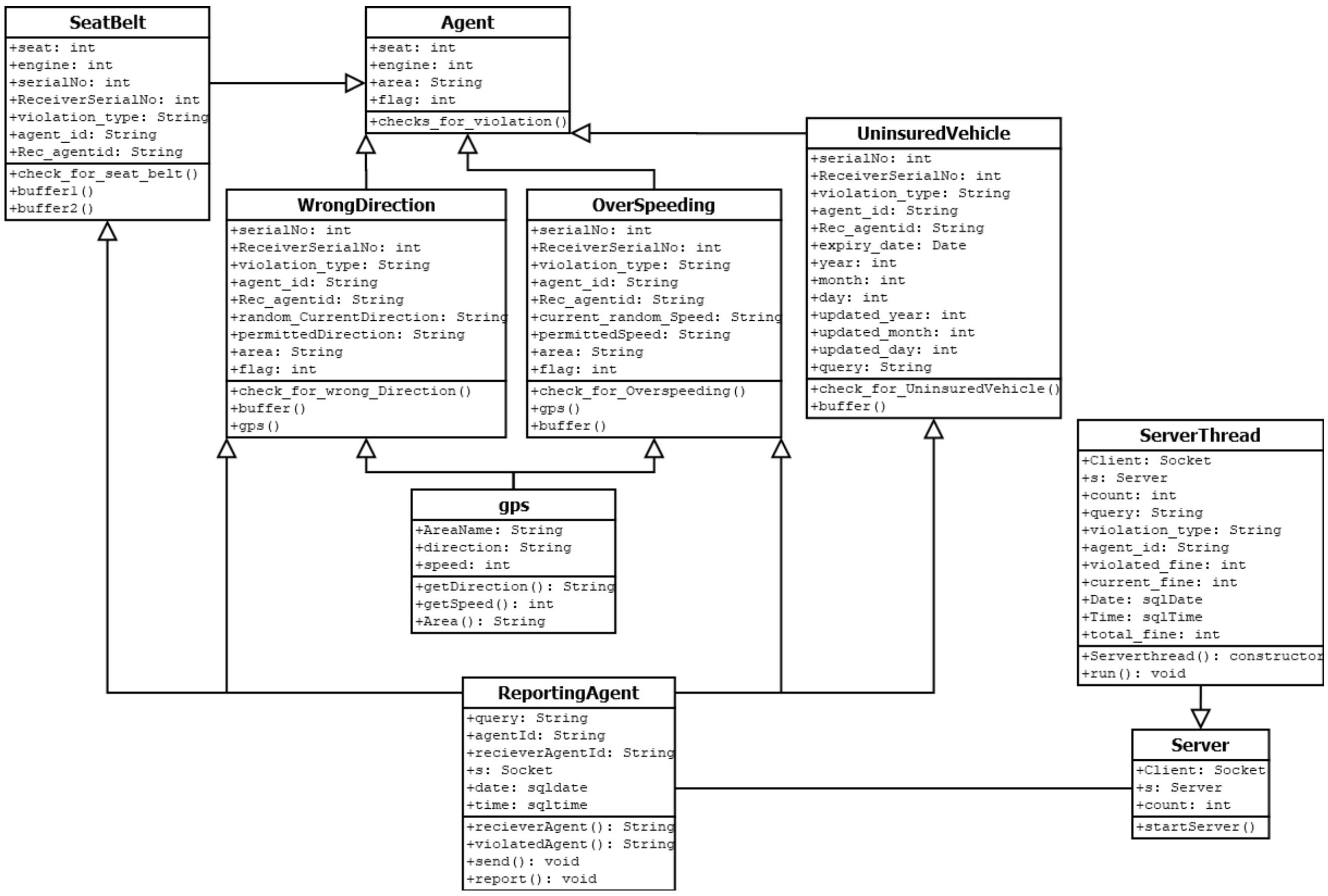


Figure 3: Class diagram of the proposed system

ER Diagram to represent how data is related to each other:

Figure 4 shows an ER diagram showing how data is disintegrated into entities and attributes and relationship between entities is shown. Using this ER diagram a database is created with assumed data.

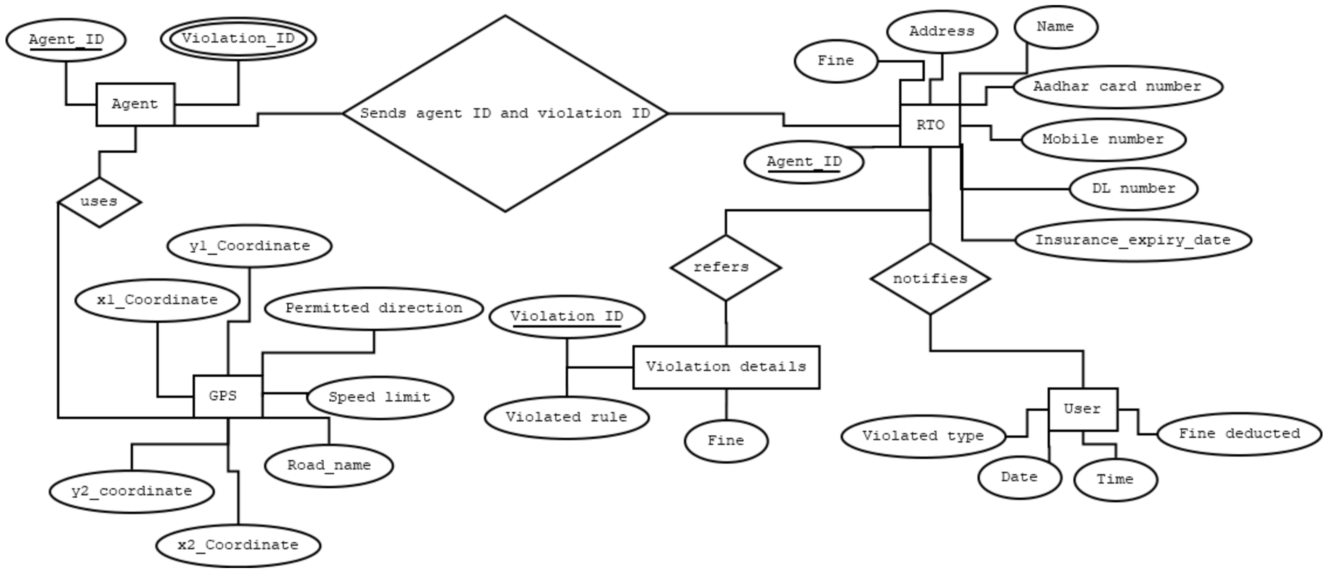


Figure 4: ER diagram of the proposed system

4.2 Simulation and Implementation of the model

Due to the increase in the number of vehicles, there is a lack of enough people to monitor. So automated detection of violation of traffic rules can be proposed as a possible solution to the problem by installing intelligent agents in cars which will be in a distributed system. Therefore, four traffic rules- Seat Belt, wrong direction, overspeeding and uninsured vehicles, from the website of the government of India were picked which were being violated frequently. Based on the design, flowcharts and algorithms for how agents will detect each violation committed by some other agent which is installed in another car is made. Using the algorithm, a java program is written for performing simulation and predicting how many times which violation occurred by running the program for at least 1 hrs. Based on the design there should be a server class and an agent class. Server class will act as rto server which will have all the details of the registered agents. Agents class will be superclass for all the violation classes and reporting agent’s class will be subclass for all the violation classes which will send the violated agents' details to the server and also notify the user about the violation committed. For all the modules which we are using in our design, a separate class will be defined with their use for them from where we can fetch the details if needed to detect violations. In agent class, all the four violations will be called one by one to one and based on the algorithm, it will check whether the violation is committed or not. For simulation purpose we will run the program in a while loop for at least 1hrs.

Algorithms for how each violation detection will work is as follows:

a) **Seat Belt Violation**:

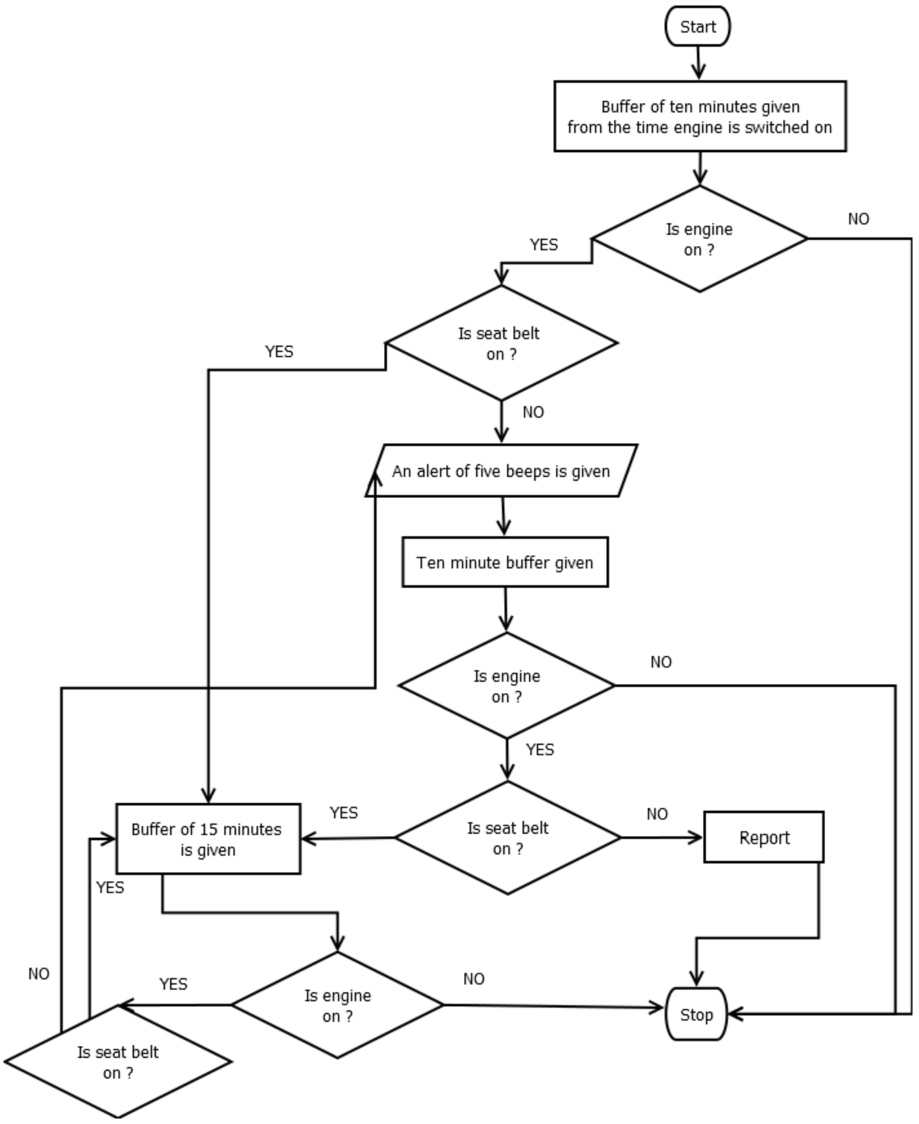


Figure 5: Flowchart for detection of seat belt violation

1. Begin.
2. Define initial value of engine as 1 and randomly generate value of seat as 1 or 0.
3. Then generate two random serial numbers and call reporting agent class by creating its object and send two randomly generated serial numbers to that class.
4. Call violated_agent and reciever_agent function and get random violated and reporting agent id respectively.
5. Give a buffer time of 10 min and then start a while loop which will run till the engine is not equal to 0.
6. Check if the randomly generated value of the seat is 1 or 0. If it's 1 then go to step 6 or else go to step 10.
7. If the value of the seat is 1 then it means that the driver is wearing the seatbelt.
8. Give buffer time of 15min and randomly generate value for seat and engine.
9. If engine is 0 then terminate the loop and go to step 18.
10. If engine is 1 then repeat step 5 to 8.
11. If the value of the seat is 0 then it means that the driver is not wearing the seatbelt.
12. Give an alert of 5 beeps as warning to wear the seat belt and after warning provide a buffer time of 10min.
13. Again generate random value for engine and seat. If the value of the engine is 0 then it means the engine is stopped and thus terminates the loop and goes to step 18.
14. If engine ==1 then check if the seat is 1 or 0. If it is 1 then repeat step 6 to 9 else go to step 14.
15. If the value of the seat is still 0 even after giving a warning, then it means that seatbelt violation is committed by the driver. Send agent details to the server through reporting agent class.
16. Provide a buffer time of 10sec to let the server update the databases.
17. After updating databases send notification to the user.
18. Return seat, agent_id, reciever_agent_id and engine to the agent class.
19. In the main class print the messages according to the returned values.
20. End.

b) Wrong Direction Violation:

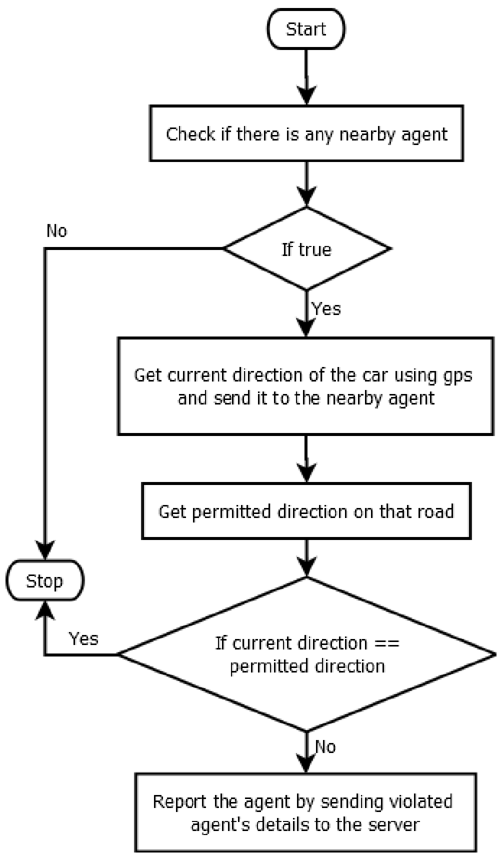

Start
Check if there is any nearby agent
If true
No
Yes
Get current direction of the car using gps and send it to the nearby agent
Get permitted direction on that road
Stop
Yes
If current direction == permitted direction
No
Report the agent by sending violated agent's details to the server

Figure 6: Flowchart for detection of vehicles moving in the wrong direction

1. Begin
2. Generate two random agent ids where one will be receiving/reporting agent and another will be the agent who will be checked for any committed violation.
3. To check if there is any agent present in the area, the coordinates of the violating agent and the reporting agent are checked if they are equal.
4. If true:
    - A random direction for the agent is generated
    - Actual permitted direction for the area and the area in which the agent is present is fetched from the database using the gps class.
    - The generated random direction is compared to the actual permitted direction, if false then agent has to be reported by reporting agent by sending details of the violated agent to the server. A buffer is added

here so as to provide time for the server to fetch details from the database and update it accordingly.

- If the condition is false, that means the agent is moving in the permitted direction.

5. After updating databases send notification to the user.
6. Return agent_id and reciever_agent_id to the agent class.
7. In the main class print the messages according to the returned values.
8. End

c) Overspeeding Violation:

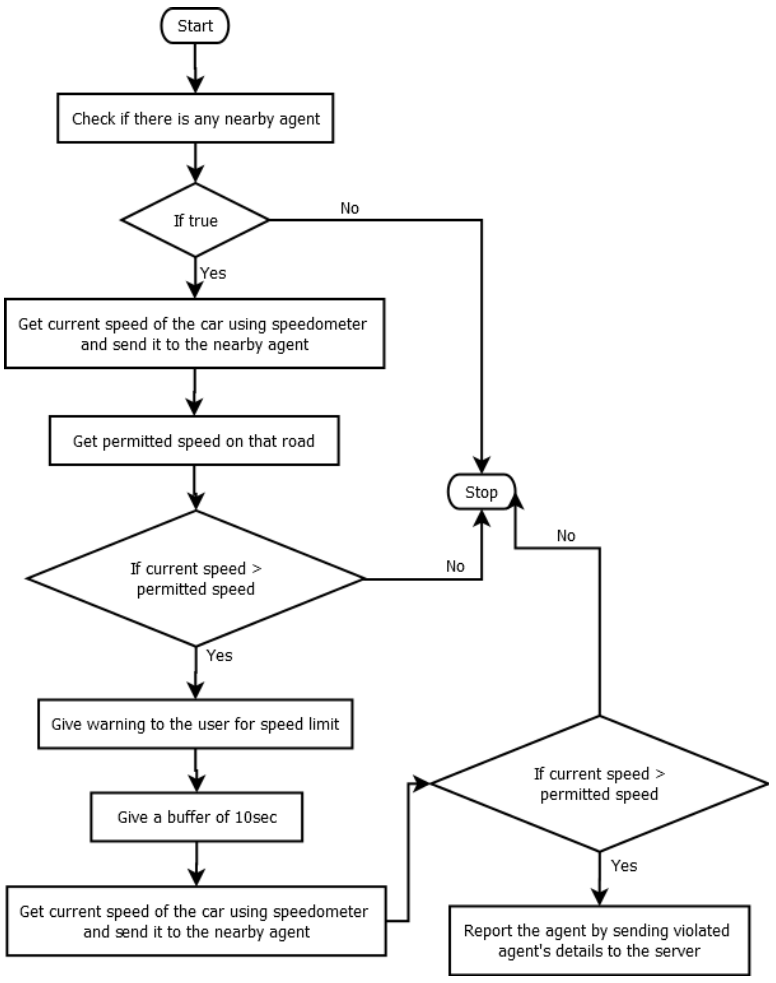

Start
Check if there is any nearby agent
If true
No
Yes
Get current speed of the car using speedometer and send it to the nearby agent
Get permitted speed on that road
Stop
If current speed > permitted speed
No
Yes
No
Give warning to the user for speed limit
Give a buffer of 10sec
If current speed > permitted speed
Yes
Get current speed of the car using speedometer and send it to the nearby agent
Report the agent by sending violated agent's details to the server

Figure 7: Flowchart for detection of overspeeding vehicles

1. Begin
2. Generate two random agent ids where one will be receiving/reporting agent and another will be the agent who will be checked for any committed violation.
3. To check if there is any agent present in the area, the coordinates of the violating agent and the reporting agent are checked
4. If true then, the current speed of the agent is fetched and the area in which the agent is present is also fetched using gps.
5. For that area fetch the permitted speed.
6. Compare the current speed with the permitted speed of the area, if the speed is within the speed limit, there is no violation, else a buffer of 20 seconds is given for the agent to slow down.
7. After 20sec again the current speed is fetched from the speedometer.
8. If the new current speed is within the speed limit then no violation, else the violation has been committed by the user.
9. Violated agent's details will be sent to the server by the reporting agent and then a buffer of 10sec will be given for the server to update the database.
10. After updating the database, send notification to the user.
11. Return agent_id and reciever_agent_id to the agent class.
12. In the main class print the messages according to the returned values.
13. End.

d) Uninsured Vehicle:

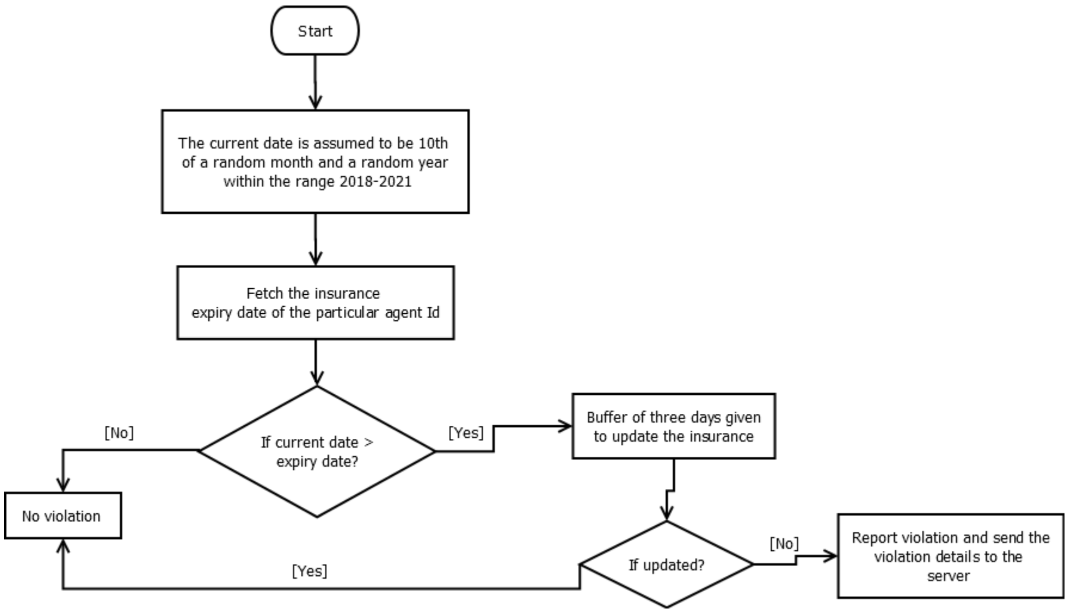

Start
The current date is assumed to be 10th of a random month and a random year within the range 2018-2021
Fetch the insurance expiry date of the particular agent Id
If current date > expiry date?
[No]
[Yes]
No violation
Buffer of three days given to update the insurance
If updated?
[No]
[Yes]
Report violation and send the violation details to the server

Figure 8: Flowchart for detection of uninsured vehicles

1. Begin
2. Generate a random agent id. The agent itself will check on every 10th of the month and report by itself rather than sending it to any other agent. (Sending another agent for this violation will only reduce efficiency.
3. From the database, fetch the vehicle's insurance expiry date of the randomly generated agent.
4. For current date, let date be 10 and month and year be random numbers. Months will be generated between 1 to 12 and years between 2018 to 2021.
5. Compare Expiry date and current date.
6. If the current date is greater than expiry date then it means the insurance of the vehicle has expired already.
7. Give a warning to the user by sending notification to the user and provide a buffer of 3days to let the user update insurance of the vehicle.
8. Again fetch the insurance expiry date from the database and check it with the current date. If current date is still greater than expiry date then violation has been committed which will be reported to the server by sending his details to the server.

9. After sending the details, give a buffer of 10sec to let the server update the database and then send notification to the user.
10. If the current date is less than expiry date then the insurance is valid.
11. On 10th of every month check for the violation i.e. repeat step 3 to 9.
12. Return agent_id to the agent class.
13. In the main class print the messages according to the returned values.
14. End.

Algorithm for how server will update the database when violation is committed is as follows:

1) Begin.

2) Define port number and socket object.

3) Create thread pool to execute the code

4) Start an infinite while loop.

5) Inside the loop whenever an agent tries to connect to the server then socket connection between server and agent will be created and data can be transferred between server and agent till the connection is closed.

6) When an agent is connected to the server then the server thread class object is created and executed where it receives agent id and violation type and updates the database.

7) In server thread class, using sql query fetch violation type and fine to be charged using violation type.

8) Fetch fine of violated agent from rto table using agent id.

9) Add both the fine and update rto fine value and set the total fine. RTO 's fine column consists of fines which the agents have committed till date.

10) After that check if a table with name as agent id exists or not. If it exists then don't create a new table and if it doesn't then a table with name as agent id is created and then further steps are followed.

11) Then update the table with violation type, date and time at which violation was committed and fine charged for that violation.

12) After updating the database, close the connection with the sql database and agent.

13) Connection between agent and server will be created only when violation is committed and has to be reported by the reporting agent and once the database is updated, connection will be closed.

14) So everytime agent tries to connect using the server's port number, repeat steps 5 to 12.

15) End.

4.3 Testing and Analysis

Based on the design and algorithm, a Java program was made and simulated for an approximate time of 1hour. All the four violations were checked one by one and if committed then reported to the server and also kept the count of number of times that violation was committed. Simulation basically means to solve real world problems by making a software for that and predict its features and future values. In this project also we brought this real world problem to a java program and simulated it with random values and predicted some values and results.

Based on the number of times violation committed, a bar chart shown in figure 9 has been made which shows that the total agents reported to the server was 53 in which, 24 agents were reported for seat belt violation, 10 agents for wrong direction violation, 12 agents for overspeeding violation and 7 agents for uninsured vehicle violation. Main purpose of this project was to create distributed agents which can check and report violations committed since it is difficult for traffic police to check everyone for violations and also there might be chances of corruption. In one hour if users were checked by traffic police then the number of vehicles who will be reported by the traffic police will be less than 53 because of the following reasons such as chances of corruption, not present everywhere to check each vehicle, not able to check each vehicle passing by.

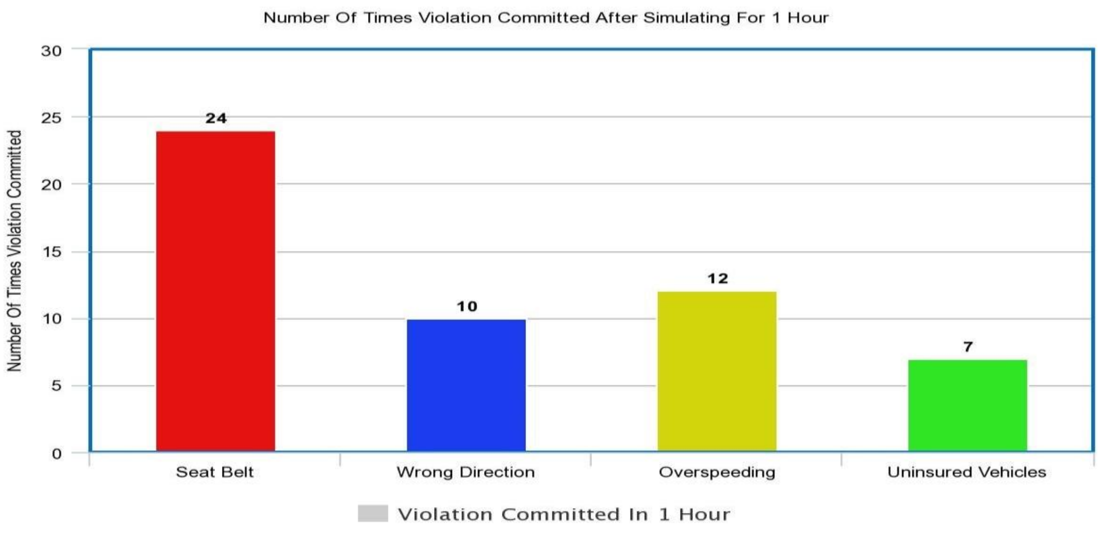
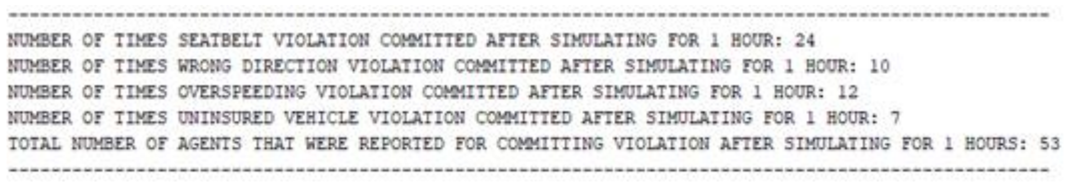


Figure 9: Graph depicting number of types of violations committed in one hour of simulation

5. Results

In this section, results obtained by this project are shown. Each and every outcomes of the project are explained with screenshots along with justification. Based on the design and algorithm, a java program was made for simulation. Random values were generated and based on the algorithm, it was simulated. Since it is a simulation, whatever results we got from this project are just predictions and not actual results. We simulated the program for approximately 1 hour and got the output with different results. 53 agents were reported for violations in 1 hours as shown in figure 10.



```
----------------------------------------------------------------------------------------------------
NUMBER OF TIMES SEATBELT VIOLATION COMMITTED AFTER SIMULATING FOR 1 HOUR: 24
NUMBER OF TIMES WRONG DIRECTION VIOLATION COMMITTED AFTER SIMULATING FOR 1 HOUR: 10
NUMBER OF TIMES OVERSPEEDING VIOLATION COMMITTED AFTER SIMULATING FOR 1 HOUR: 12
NUMBER OF TIMES UNINSURED VEHICLE VIOLATION COMMITTED AFTER SIMULATING FOR 1 HOUR: 7
TOTAL NUMBER OF AGENTS THAT WERE REPORTED FOR COMMITTING VIOLATION AFTER SIMULATING FOR 1 HOURS: 53
----------------------------------------------------------------------------------------------------
```

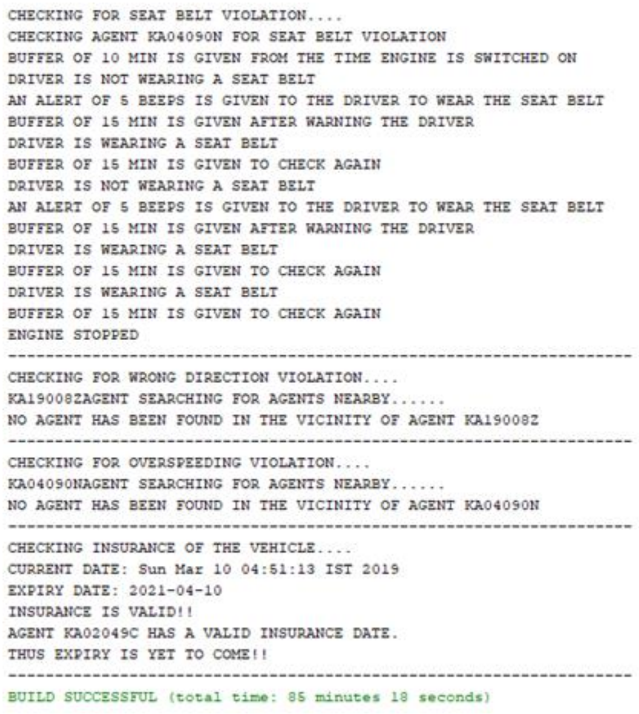
Figure 10: Screenshot of actual simulation to count the number of times each type of violation is committed

Rather than showing the whole output which were obtained after running the program for 1 hour, we have just shown main output screenshots which show different scenarios such as- when violation is committed then what will happen and if not committed then what will happen and also when the user is given warning and after that if the violation is still committed or not. Different outcomes like these are shown as follows:

(All these violations are checked based on the algorithm explained in chapter 4)

1. **<u>When violation is not committed or no agent is found nearby to check</u>:**

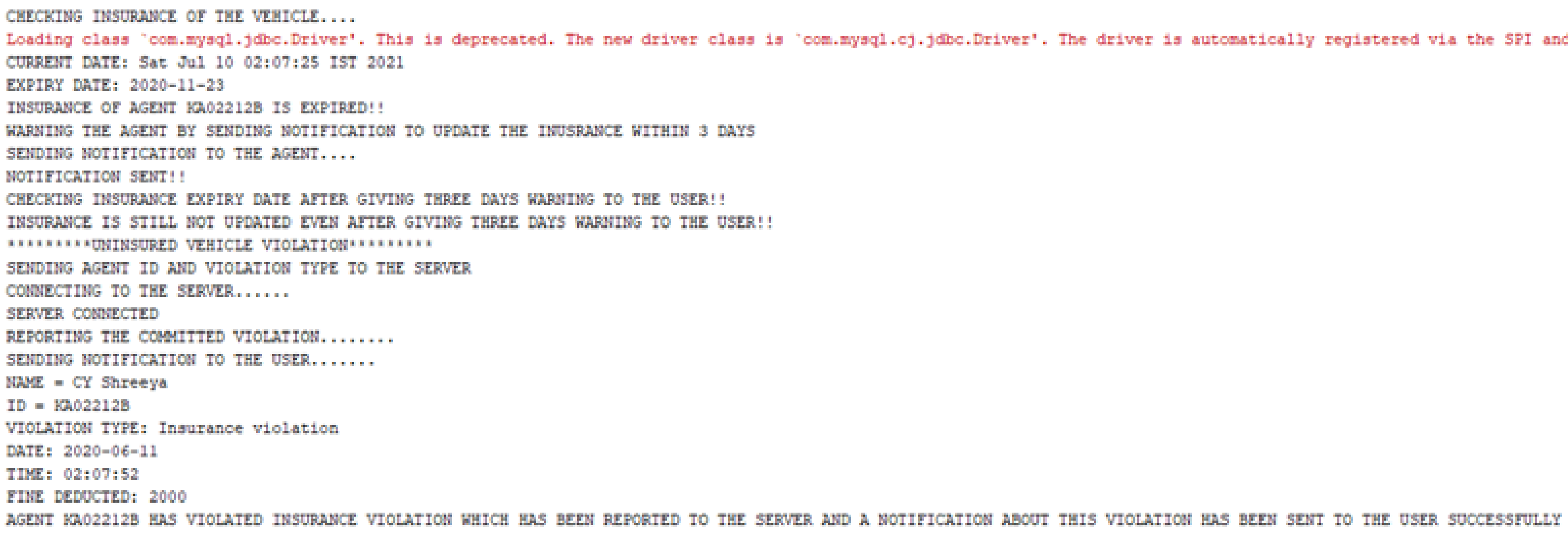
In Figure 10, it can be seen that the first seat belt violation is checked where the driver was warned for not wearing a seatbelt but later the engine stopped so next violation gets checked i.e. wrong direction where no agents were found in the vicinity thus it stopped checking and same with overspeeding violation. Next insurance is checked and found that insurance is valid thus no violation was committed.



```
CHECKING FOR SEAT BELT VIOLATION....
CHECKING AGENT KA04090N FOR SEAT BELT VIOLATION
BUFFER OF 10 MIN IS GIVEN FROM THE TIME ENGINE IS SWITCHED ON
DRIVER IS NOT WEARING A SEAT BELT
AN ALERT OF 5 BEEPS IS GIVEN TO THE DRIVER TO WEAR THE SEAT BELT
BUFFER OF 15 MIN IS GIVEN AFTER WARNING THE DRIVER
DRIVER IS WEARING A SEAT BELT
BUFFER OF 15 MIN IS GIVEN TO CHECK AGAIN
DRIVER IS NOT WEARING A SEAT BELT
AN ALERT OF 5 BEEPS IS GIVEN TO THE DRIVER TO WEAR THE SEAT BELT
BUFFER OF 15 MIN IS GIVEN AFTER WARNING THE DRIVER
DRIVER IS WEARING A SEAT BELT
BUFFER OF 15 MIN IS GIVEN TO CHECK AGAIN
DRIVER IS WEARING A SEAT BELT
BUFFER OF 15 MIN IS GIVEN TO CHECK AGAIN
ENGINE STOPPED
------------------------------------------------------------------
CHECKING FOR WRONG DIRECTION VIOLATION....
KA19008ZAGENT SEARCHING FOR AGENTS NEARBY......
NO AGENT HAS BEEN FOUND IN THE VICINITY OF AGENT KA19008Z
------------------------------------------------------------------
CHECKING FOR OVERSPEEDING VIOLATION....
KA04090NAGENT SEARCHING FOR AGENTS NEARBY......
NO AGENT HAS BEEN FOUND IN THE VICINITY OF AGENT KA04090N
------------------------------------------------------------------
CHECKING INSURANCE OF THE VEHICLE....
CURRENT DATE: Sun Mar 10 04:51:13 IST 2019
EXPIRY DATE: 2021-04-10
INSURANCE IS VALID!!
AGENT KA02049C HAS A VALID INSURANCE DATE.
THUS EXPIRY IS YET TO COME!!
------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 85 minutes 18 seconds)
```

Figure 11: Screenshot of output where no violation is committed.

1. **<u>Insurance violation committed</u>:**

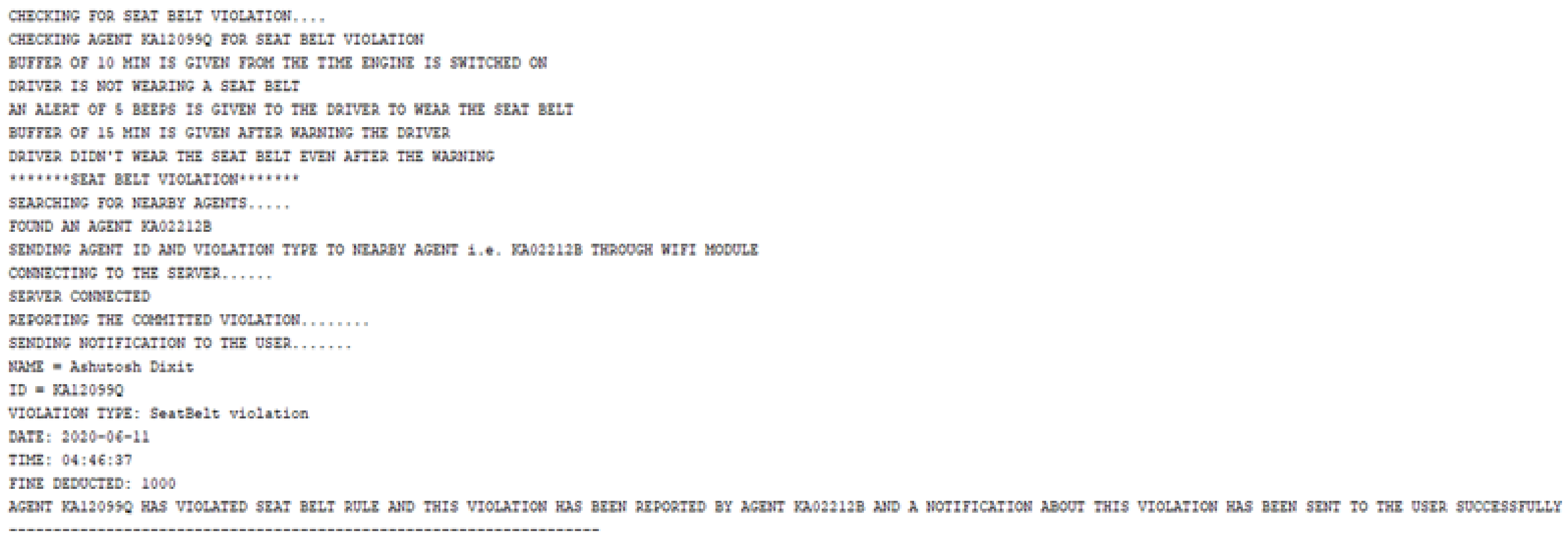


```
CHECKING INSURANCE OF THE VEHICLE....
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and
CURRENT DATE: Sat Jul 10 02:07:25 IST 2021
EXPIRY DATE: 2020-11-23
INSURANCE OF AGENT KA02212B IS EXPIRED!!
WARNING THE AGENT BY SENDING NOTIFICATION TO UPDATE THE INUSRANCE WITHIN 3 DAYS
SENDING NOTIFICATION TO THE AGENT....
NOTIFICATION SENT!!
CHECKING INSURANCE EXPIRY DATE AFTER GIVING THREE DAYS WARNING TO THE USER!!
INSURANCE IS STILL NOT UPDATED EVEN AFTER GIVING THREE DAYS WARNING TO THE USER!!
*********UNINSURED VEHICLE VIOLATION*********
SENDING AGENT ID AND VIOLATION TYPE TO THE SERVER
CONNECTING TO THE SERVER......
SERVER CONNECTED
REPORTING THE COMMITTED VIOLATION........
SENDING NOTIFICATION TO THE USER.......
NAME = CY Shreeya
ID = KA02212B
VIOLATION TYPE: Insurance violation
DATE: 2020-06-11
TIME: 02:07:52
FINE DEDUCTED: 2000
AGENT KA02212B HAS VIOLATED INSURANCE VIOLATION WHICH HAS BEEN REPORTED TO THE SERVER AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY
```

Figure 12: Screenshot of insurance violation detection

Figure 12 shows that an insurance violation is committed by the agent. Current date and expiry date is compared and found out that current date is greater than the expiry date and thus the user is sent a warning notification on mobile and given a three days' buffer to get insurance renewed. After 3 days again the current and expiry date is checked and found that still it is not updated even after the warning and thus violation is committed by the user and reported to the server by sending the details to the server. After reporting, the user receives a notification in her mobile for the violation she has committed, date and time at which this violation is reported and fine which has been deducted accordingly.

2. **<u>Seat belt violation committed</u>:**

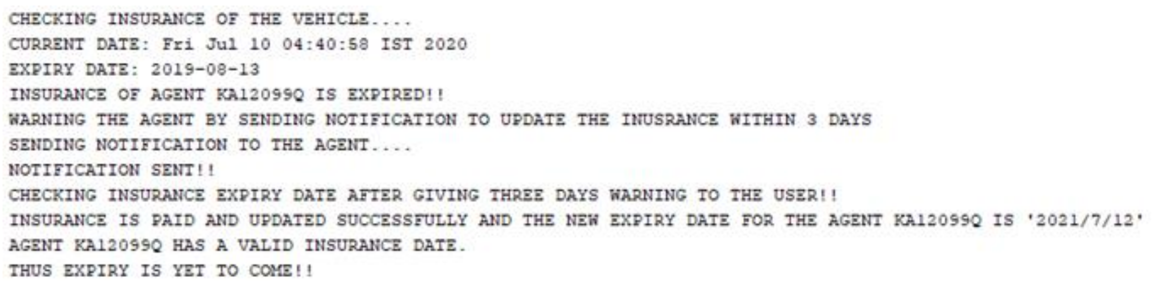


```
CHECKING FOR SEAT BELT VIOLATION....
CHECKING AGENT KA12099Q FOR SEAT BELT VIOLATION
BUFFER OF 10 MIN IS GIVEN FROM THE TIME ENGINE IS SWITCHED ON
DRIVER IS NOT WEARING A SEAT BELT
AN ALERT OF 5 BEEPS IS GIVEN TO THE DRIVER TO WEAR THE SEAT BELT
BUFFER OF 15 MIN IS GIVEN AFTER WARNING THE DRIVER
DRIVER DIDN'T WEAR THE SEAT BELT EVEN AFTER THE WARNING
*******SEAT BELT VIOLATION*******
SEARCHING FOR NEARBY AGENTS.....
FOUND AN AGENT KA02212B
SENDING AGENT ID AND VIOLATION TYPE TO NEARBY AGENT i.e. KA02212B THROUGH WIFI MODULE
CONNECTING TO THE SERVER......
SERVER CONNECTED
REPORTING THE COMMITTED VIOLATION........
SENDING NOTIFICATION TO THE USER.......
NAME = Ashutosh Dixit
ID = KA12099Q
VIOLATION TYPE: SeatBelt violation
DATE: 2020-06-11
TIME: 04:46:37
FINE DEDUCTED: 1000
AGENT KA12099Q HAS VIOLATED SEAT BELT RULE AND THIS VIOLATION HAS BEEN REPORTED BY AGENT KA02212B AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY
--------------------------------------------------------------------
```

Figure 13: Screenshot Of Seat Belt Violation Detection

Figure 13 shows that seatbelt violation is committed by the user. Based on the algorithm, the first 10min is given after the engine has started. Even after the 10 min buffer driver was not wearing the seatbelt and thus the warning was given to the user with an alert of 5 beeps. And after a buffer time of 15min, again he was checked but still he wasn't wearing the seatbelt thus seat belt violation is committed. Then a nearby agent is searched and to that agent, details of the violated agent are sent and through that agent, details have been sent to the server and the database is updated. After that user is sent a notification for the committed violation with all the details.

3. **<u>Insurance was expired but then was updated after giving warning</u>:**

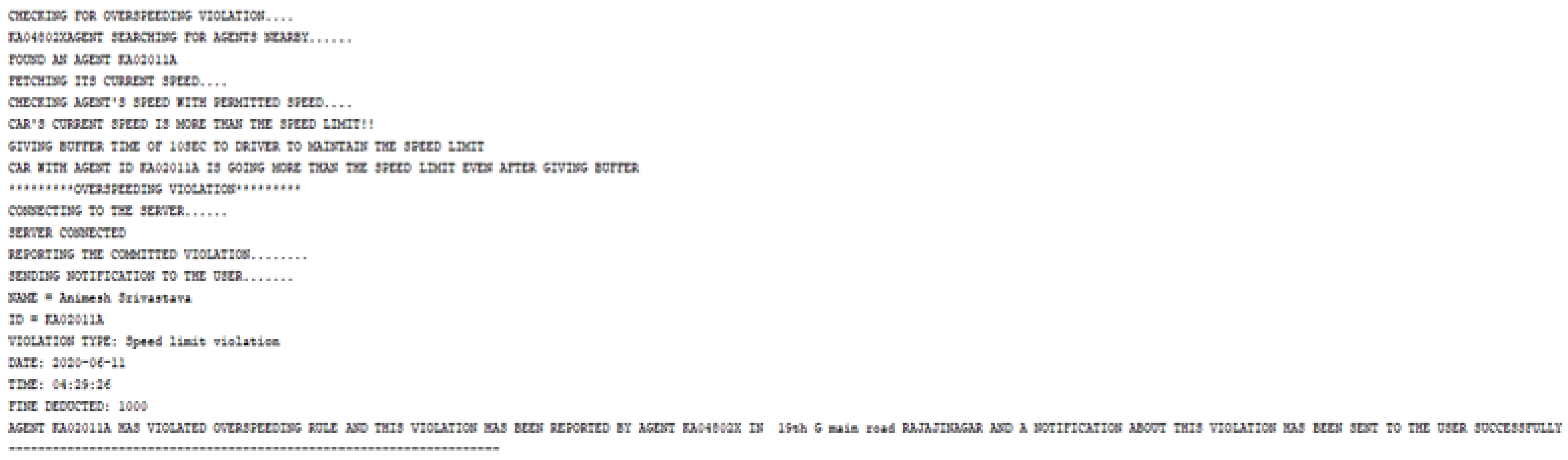


```
CHECKING INSURANCE OF THE VEHICLE....
CURRENT DATE: Fri Jul 10 04:40:58 IST 2020
EXPIRY DATE: 2019-08-13
INSURANCE OF AGENT KA12099Q IS EXPIRED!!
WARNING THE AGENT BY SENDING NOTIFICATION TO UPDATE THE INUSRANCE WITHIN 3 DAYS
SENDING NOTIFICATION TO THE AGENT....
NOTIFICATION SENT!!
CHECKING INSURANCE EXPIRY DATE AFTER GIVING THREE DAYS WARNING TO THE USER!!
INSURANCE IS PAID AND UPDATED SUCCESSFULLY AND THE NEW EXPIRY DATE FOR THE AGENT KA12099Q IS '2021/7/12'
AGENT KA12099Q HAS A VALID INSURANCE DATE.
THUS EXPIRY IS YET TO COME!!
```

Figure 14: Screenshot of output when insurance is renewed after warning

Figure 14 shows that insurance of the vehicle is checked by comparing the current date with the expiry date. It is found that the vehicle’s insurance is expired and thus warning is sent to the user's mobile and given a three days’ buffer to get the insurance updated. After three days again current and expiry date is checked and found that the user got his insurance renewed i.e. expiry date has been updated successfully thus no violation was committed.

4. **<u>Overspeed violation committed</u>:**

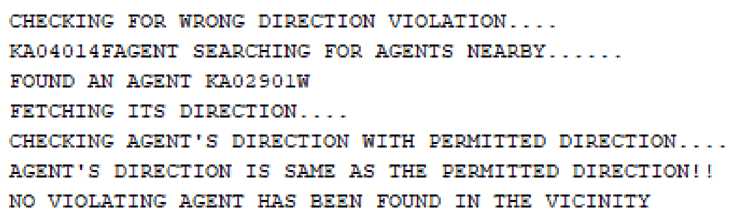


```
CHECKING FOR OVERSPEEDING VIOLATION....
KA04802XAGENT SEARCHING FOR AGENTS NEARBY......
FOUND AN AGENT KA02011A
FETCHING ITS CURRENT SPEED....
CHECKING AGENT'S SPEED WITH PERMITTED SPEED....
CAR'S CURRENT SPEED IS MORE THAN THE SPEED LIMIT!!
GIVING BUFFER TIME OF 10SEC TO DRIVER TO MAINTAIN THE SPEED LIMIT
CAR WITH AGENT ID KA02011A IS GOING MORE THAN THE SPEED LIMIT EVEN AFTER GIVING BUFFER
*********OVERSPEEDING VIOLATION*********
CONNECTING TO THE SERVER......
SERVER CONNECTED
REPORTING THE COMMITTED VIOLATION........
SENDING NOTIFICATION TO THE USER.......
NAME = Animesh Srivastava
ID = KA02011A
VIOLATION TYPE: Speed limit violation
DATE: 2020-06-11
TIME: 04:29:26
FINE DEDUCTED: 1000
AGENT KA02011A HAS VIOLATED OVERSPEEDING RULE AND THIS VIOLATION HAS BEEN REPORTED BY AGENT KA04802X IN  19th G main road RAJAJINAGAR AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY
---------------------------------------------------------------------
```

Figure 15: Screenshot of Overspeeding Violation Detection

Figure 15 shows that overspeeding violation is committed by the user. First an agent searches for nearby agents in the vicinity and when found fetches it’s current speed. From the database, permitted speed is fetched and both permitted and current speed is checked and found that the user is going above speed limit and thus user is given a 10sec buffer time to maintain his speed according to the speed limit. After 10sec again speed is fetched and found that the speed is still more than the speed limit and thus violation has been committed by the user. Receiving agent will then send the details of the violated agent to the server and the database gets updated. After that user is sent a notification about the violation committed with all the details of the violation.

5. <u>Checking for wrong direction and found that the violations not committed</u>:

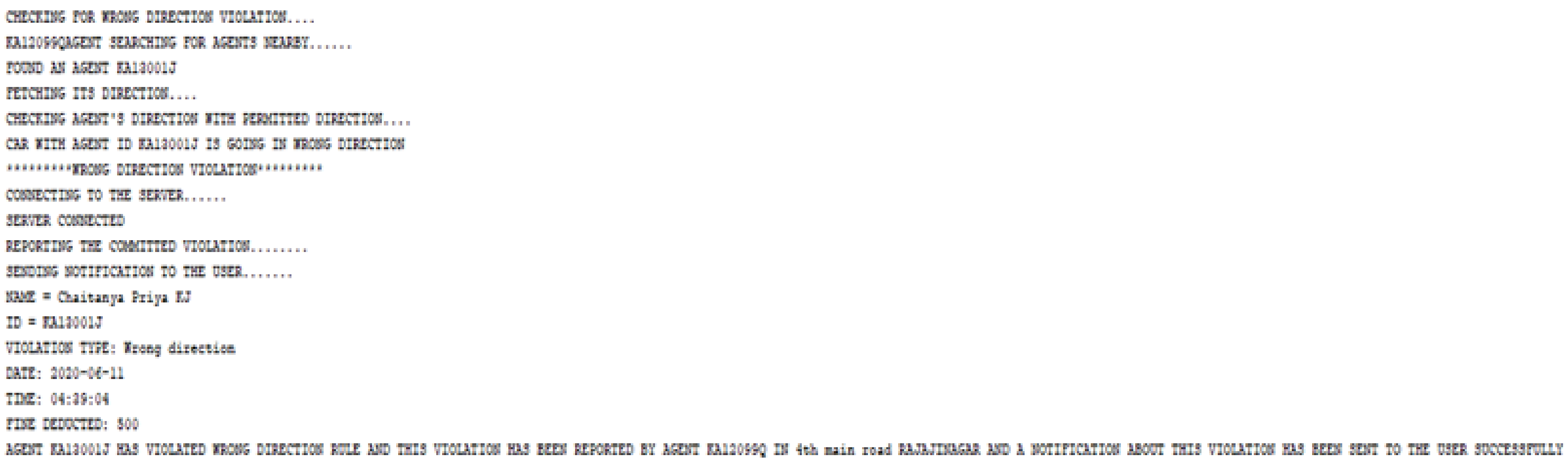


```
CHECKING FOR WRONG DIRECTION VIOLATION....
KA04014FAGENT SEARCHING FOR AGENTS NEARBY......
FOUND AN AGENT KA02901W
FETCHING ITS DIRECTION....
CHECKING AGENT'S DIRECTION WITH PERMITTED DIRECTION....
AGENT'S DIRECTION IS SAME AS THE PERMITTED DIRECTION!!
NO VIOLATING AGENT HAS BEEN FOUND IN THE VICINITY
```

Figure 16: Screenshot of Output where user didn't commit wrong direction violation

Figure 16 shows that an agent first searches for any nearby agent and finds an agent. Then it fetches the current direction in which the car is going. Then from the database, permitted direction is fetched and compared with the current direction and found that the agent's direction is the same as permitted direction thus no violation was committed by the agent.

6. <u>Wrong direction violation committed</u>:

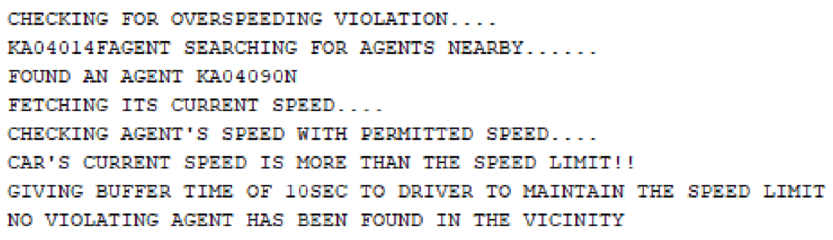


```
CHECKING FOR WRONG DIRECTION VIOLATION....
KA12099QAGENT SEARCHING FOR AGENTS NEARBY......
FOUND AN AGENT KA13001J
FETCHING ITS DIRECTION....
CHECKING AGENT'S DIRECTION WITH PERMITTED DIRECTION....
CAR WITH AGENT ID KA13001J IS GOING IN WRONG DIRECTION
*********WRONG DIRECTION VIOLATION*********
CONNECTING TO THE SERVER......
SERVER CONNECTED
REPORTING THE COMMITTED VIOLATION........
SENDING NOTIFICATION TO THE USER.......
NAME = Chaitanya Priya KJ
ID = KA13001J
VIOLATION TYPE: Wrong direction
DATE: 2020-06-11
TIME: 04:39:04
FINE DEDUCTED: 500
AGENT KA13001J HAS VIOLATED WRONG DIRECTION RULE AND THIS VIOLATION HAS BEEN REPORTED BY AGENT KA12099Q IN 4th main road RAJAJINAGAR AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY
```

Figure 17: Screenshot Of Wrong Direction Violation Detection

Figure 17 shows that an agent searches for any nearby agents and finds an agent. Then it fetches the current direction in which the car is going. Then from the database, permitted direction is fetched and compared with the current direction and found that the agent is going in the wrong direction thus the wrong direction violation is committed by the user. Agent's details are sent to the server by the receiving agent and the database is updated and after that a notification is sent to the user with all the details of the violation committed.

7. **<u>Checking for overspeeding and found that the violations not committed</u>:**

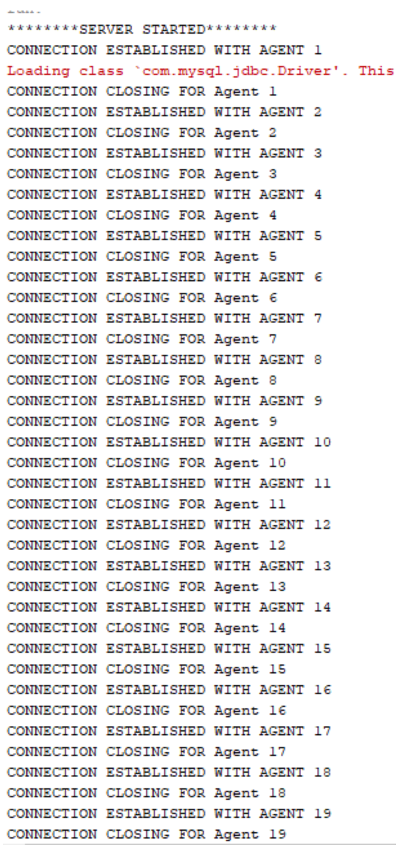


```
CHECKING FOR OVERSPEEDING VIOLATION....
KA04014FAGENT SEARCHING FOR AGENTS NEARBY......
FOUND AN AGENT KA0409ON
FETCHING ITS CURRENT SPEED....
CHECKING AGENT'S SPEED WITH PERMITTED SPEED....
CAR'S CURRENT SPEED IS MORE THAN THE SPEED LIMIT!!
GIVING BUFFER TIME OF 10SEC TO DRIVER TO MAINTAIN THE SPEED LIMIT
NO VIOLATING AGENT HAS BEEN FOUND IN THE VICINITY
```

Figure 18: Screenshot of output where the detected agent has not committed overspeeding violation

Figure 18 shows that an agent first searches for any nearby agent and finds an agent. Then it fetches the current speed at which the car is moving. Then from the database, permitted speed is fetched and compared with the current speed and found that the agent's speed is within the speed limit thus no violation was committed by the agent.

8. Server side output:



```
********SERVER STARTED********
CONNECTION ESTABLISHED WITH AGENT 1
Loading class `com.mysql.jdbc.Driver'. This
CONNECTION CLOSING FOR Agent 1
CONNECTION ESTABLISHED WITH AGENT 2
CONNECTION CLOSING FOR Agent 2
CONNECTION ESTABLISHED WITH AGENT 3
CONNECTION CLOSING FOR Agent 3
CONNECTION ESTABLISHED WITH AGENT 4
CONNECTION CLOSING FOR Agent 4
CONNECTION ESTABLISHED WITH AGENT 5
CONNECTION CLOSING FOR Agent 5
CONNECTION ESTABLISHED WITH AGENT 6
CONNECTION CLOSING FOR Agent 6
CONNECTION ESTABLISHED WITH AGENT 7
CONNECTION CLOSING FOR Agent 7
CONNECTION ESTABLISHED WITH AGENT 8
CONNECTION CLOSING FOR Agent 8
CONNECTION ESTABLISHED WITH AGENT 9
CONNECTION CLOSING FOR Agent 9
CONNECTION ESTABLISHED WITH AGENT 10
CONNECTION CLOSING FOR Agent 10
CONNECTION ESTABLISHED WITH AGENT 11
CONNECTION CLOSING FOR Agent 11
CONNECTION ESTABLISHED WITH AGENT 12
CONNECTION CLOSING FOR Agent 12
CONNECTION ESTABLISHED WITH AGENT 13
CONNECTION CLOSING FOR Agent 13
CONNECTION ESTABLISHED WITH AGENT 14
CONNECTION CLOSING FOR Agent 14
CONNECTION ESTABLISHED WITH AGENT 15
CONNECTION CLOSING FOR Agent 15
CONNECTION ESTABLISHED WITH AGENT 16
CONNECTION CLOSING FOR Agent 16
CONNECTION ESTABLISHED WITH AGENT 17
CONNECTION CLOSING FOR Agent 17
CONNECTION ESTABLISHED WITH AGENT 18
CONNECTION CLOSING FOR Agent 18
CONNECTION ESTABLISHED WITH AGENT 19
CONNECTION CLOSING FOR Agent 19
```

Figure 19: Screenshot of server side output which shows agents connecting to the server and disconnecting after updating the databases

Figure 19 shows the server side output which tells that whenever the agent reports any violations, a connection will be established between agent and server and after the violation is reported successfully by the server, connection between agent and server will be closed. In the above figure, only for 19 agents, output is shown rather than displaying it for all 53 agents.

6. Project Costing

Since the whole project has been simulated there is no expenditure on any hardware components. So the only cost in this project that comes into picture is the payment of the developers, tester, simulation etc. Here, the engineers are paid for the amount of hours they worked on the project.

Generally, a sum of 180 rupees per hour is paid to the developers and a sum of 1000 rupees for testing at least five functionalities.

It is assumed that one software simulation costs around 2000rupees. Thus in the proposed project, server and client connections, different violation classes which are based on the algorithm, database and also distributed system have to be tested. Overall that makes approximately ten functionalities that have to be tested.

Thus for testing purposes, the cost will sum up to approximately 2000 rupees.

For developing of the software, it is assumed that one person has worked for at least 5 hours a day, that makes 4*180 = 720 rupees per day for one person. Considering it took approximately three weeks to complete the coding, costing for a week would be

720*7= 5040 rupees per person for a week

Overall cost for this project is as follows:

1. Developing the program was done by two students who worked for three weeks and two students worked for one week, therefore total developing cost will be:

   (5040 * 3 weeks * 2 students) + (5040 * 1 week * 2 students) = **Rs 40,320**

2. Testing each functionality was done by two people thus it will be

   2000 * 2 students = **Rs 4000**

3. Simulating of the program was done by two people thus it will be

   2000 * 2 students = **Rs 4000**

Thus total cost for this project will be Rs (40,320 + 4000 + 4000) = **Rs 48,320**

7. Conclusions and Suggestions for Future Work

The following section concludes the report and suggests future improvements that can be made on the project to increase the efficiency of the system.

Conclusion

- Information about the existing government rules and regulations for traffic rules violations was collected. Literature survey has been successfully conducted on different papers, journals and patents which were based on automation of traffic rule violations.
- Intelligent agents to understand traffic violations were successfully modelled and simulated.
- The intelligent agents were successfully implemented in java.
- The simulated agents were deployed/distributed to detect traffic violations made and report them respectively.
- Documentation has been successfully completed in every phase of the project.

In this project, the agents can detect four types of traffic violations, namely, driving without seat belt, overspeeding, driving in the wrong direction and driving with expired insurance. All the agents are connected through a network, violations caused by a particular will be reported to a central server which will make the necessary changes in the database, notify the user and report it to the RTO, by another agent. The agents are simulated with the help of client server program, where the clients are acting as the agents and the server is the central server here. Respective classes are called to check if a particular violation is being made or not.

Future work

- More number of violations can be added to the list of the violations that are currently being detected and reported.
- This project is a simulation, an actual working model with all the necessary hardware components can be built and tested on real cars.
- This project focuses on installing these intelligent agents on cars and detecting violations occurred while driving cars, it can be extended to two-wheelers and other vehicles as well.
- This project has information about the roads and traffic rules for a particular area, here Rajajinagar, Bangalore. It can be extended to cover a larger geographical area or a complete city or a country.
- These agents can be installed to the newly manufactured cars as of now, a way to install these agents to existing cars should be found.

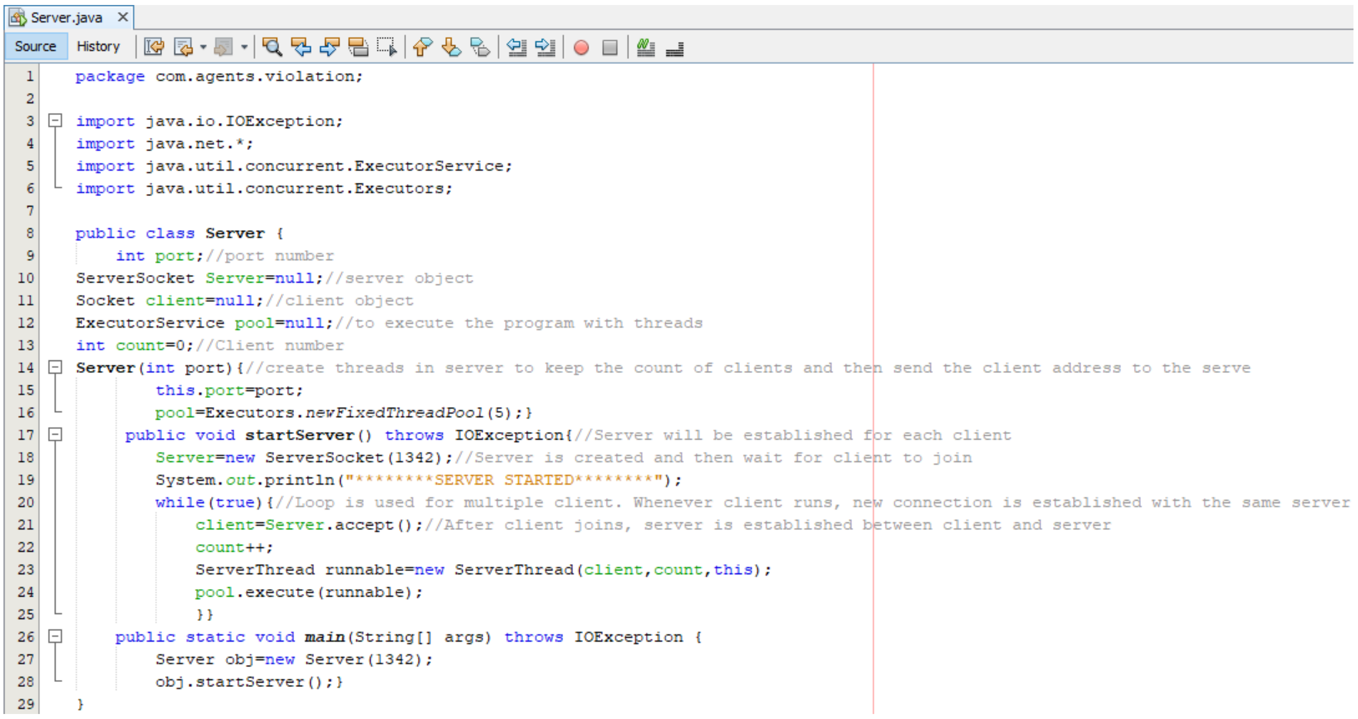
References

- [1] Aliane, Nourdine & Fernández, Javier & Bemposta Rosende, Sergio & Mata. Traffic Violation Alert and Management. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC. 10.1109/ITSC.2011.6082811. (2011)
- [2] Iqbal, Mazher & Kousar, S.Heena. Automatic Vehicle Accident Detection and Reporting With Black Box. International Journal of Applied Engineering Research. Vol. 10. 139-145. (2015)
- [3] GG Chiddarwar, Akshay Soni, Vikrant Nikham, Akshat Soni, Abhishek Sathe, Abhey Narkhede. Automatic Traffic Rule Violation Detection and Number Plate Recognition. IJSTE - International Journal of Science Technology & Engineering, Volume 3, Issue 09. (March 2017)
- [4] Rober Ciolli, Peter Whyte, Gurchan Ercan, Andrew Mack. Automatic Traffic Violation Monitoring And Reporting System. United States Patent, US 6,546,119 B2. (April 2003)
- [5] Yenikaya. Automatic Determination of Traffic Rule Breaks And Application Of Penal Procedures. PCT - Patent Cooperation Treaty, WO 2005/071637 A1. (August 2005)
- [6] Samir A. Elsagheer Mohamed. Automatic Traffic Violation Recording and Reporting System to Limit Traffic Accidents. ITCE - International Conference on Innovative Trends in Computer Engineering , Aswan, Egypt. (2019)
- [7] D.R Agrawal, Kasliwal Komal S, Gandhi Labhesh R, Deore Puram C, Saitwal Pooja D. IJSTE - International Journal of Science Technology & Engineering, Volume 163, Issue 08. (April 2017)

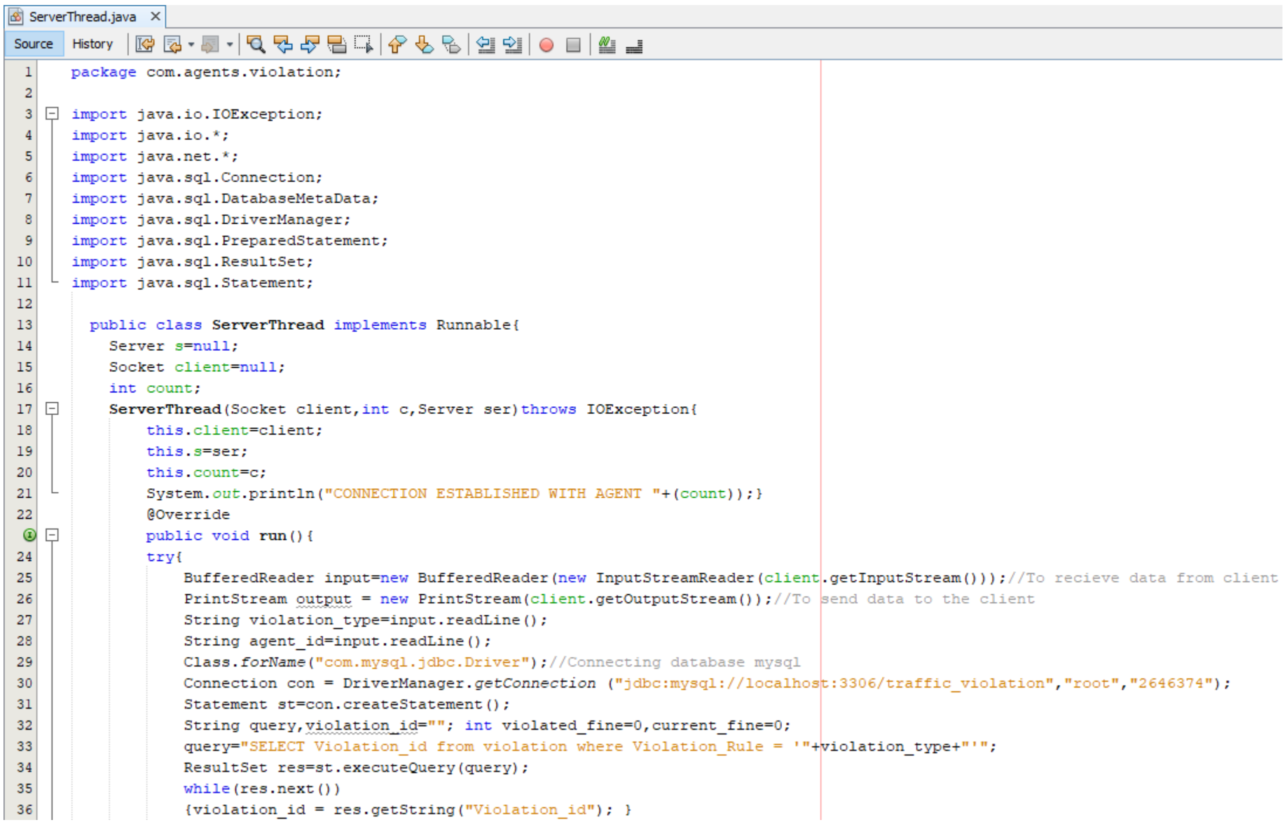
Appendix

Part A: Program Screenshots

Main Server class and connected ServerThread class:



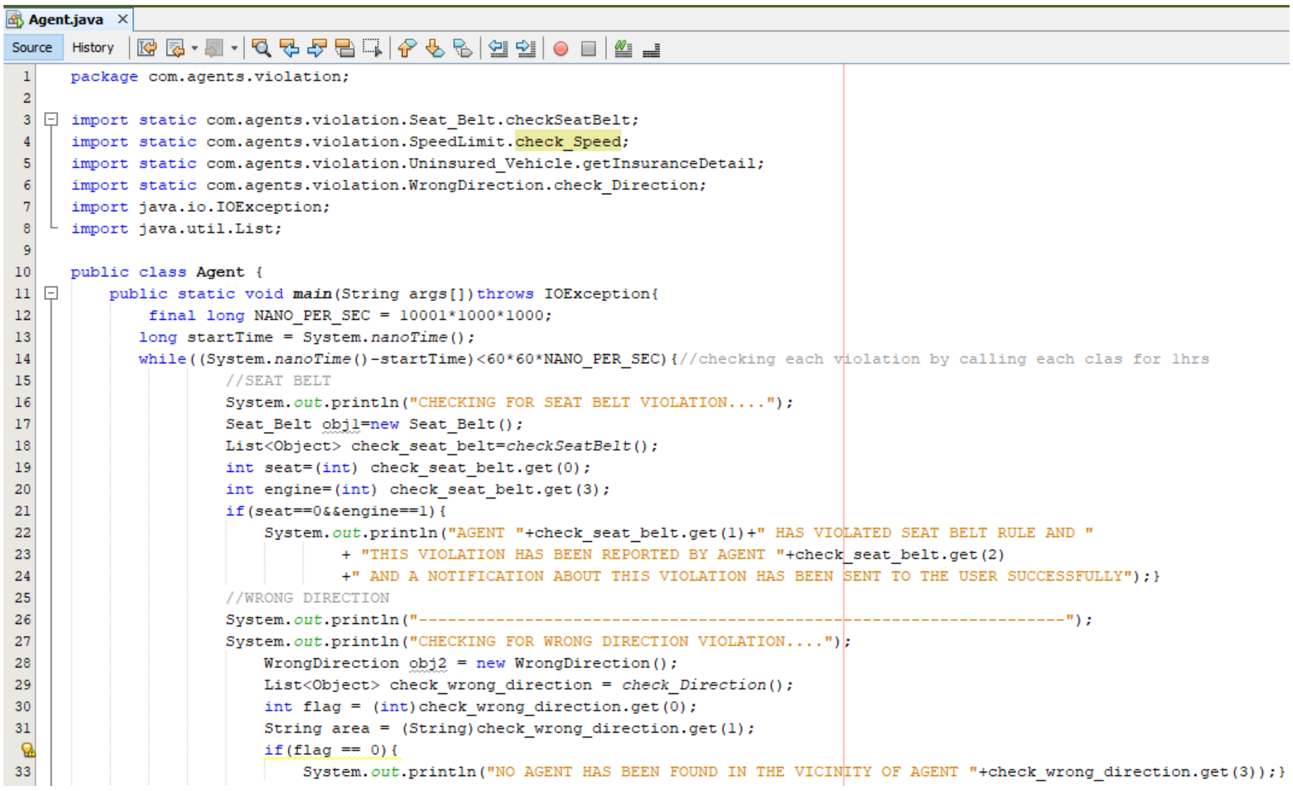
```java
package com.agents.violation;

import java.io.IOException;
import java.net.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Server {
    int port;//port number
ServerSocket Server=null;//server object
Socket client=null;//client object
ExecutorService pool=null;//to execute the program with threads
int count=0;//Client number
Server(int port){//create threads in server to keep the count of clients and then send the client address to the serve
        this.port=port;
        pool=Executors.newFixedThreadPool(5);}
     public void startServer() throws IOException{//Server will be established for each client
        Server=new ServerSocket(1342);//Server is created and then wait for client to join
        System.out.println("********SERVER STARTED********");
        while(true){//Loop is used for multiple client. Whenever client runs, new connection is established with the same server
            client=Server.accept();//After client joins, server is established between client and server
            count++;
            ServerThread runnable=new ServerThread(client,count,this);
            pool.execute(runnable);
            }}
    public static void main(String[] args) throws IOException {
        Server obj=new Server(1342);
        obj.startServer();}
}
```



```java
package com.agents.violation;

import java.io.IOException;
import java.io.*;
import java.net.*;
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

  public class ServerThread implements Runnable{
    Server s=null;
    Socket client=null;
    int count;
    ServerThread(Socket client,int c,Server ser)throws IOException{
        this.client=client;
        this.s=ser;
        this.count=c;
        System.out.println("CONNECTION ESTABLISHED WITH AGENT "+(count));}
        @Override
        public void run(){
        try{
            BufferedReader input=new BufferedReader(new InputStreamReader(client.getInputStream()));//To recieve data from client
            PrintStream output = new PrintStream(client.getOutputStream());//To send data to the client
            String violation_type=input.readLine();
            String agent_id=input.readLine();
            Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
            Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374");
            Statement st=con.createStatement();
            String query,violation_id=""; int violated_fine=0,current_fine=0;
            query="SELECT Violation_id from violation where Violation_Rule = '"+violation_type+"'";
            ResultSet res=st.executeQuery(query);
            while(res.next())
            {violation_id = res.getString("Violation_id"); }
query="SELECT Fine from violation where Violation_Rule = '"+violation_type+"'";
res =st.executeQuery(query);
while(res.next())
{violated_fine=res.getInt("Fine");}
query="SELECT Fine from rto where AgentID = '"+agent_id+"'";
res =st.executeQuery(query);
while(res.next())
{current_fine=res.getInt("Fine");}
int total_fine=current_fine + violated_fine;
query="UPDATE rto SET Fine = "+total_fine+" where AgentID = '"+agent_id+"'";
st.executeUpdate(query);
DatabaseMetaData dbm =con.getMetaData();
ResultSet table = dbm.getTables(null,null,agent_id,null);
if(!(table.next())){
query="CREATE TABLE "+agent_id+" (Violation_Rule VARCHAR(50), Date DATE, Time TIME, Fine_deducted INTEGER)";
st.executeUpdate(query);}
java.util.Date date = new java.util.Date();
java.sql.Date sqlDate = new java.sql.Date(date.getTime());
java.sql.Time sqlTime = new java.sql.Time(date.getTime());
query="INSERT into "+agent_id+"(Violation_Rule,Date,Time,Fine_Deducted) VALUES(?,?,?,?)";
PreparedStatement psmt = con.prepareStatement(query);
psmt.setString(1,violation_type);
psmt.setDate(2, sqlDate);
psmt.setTime(3,sqlTime);
psmt.setInt(4, violated_fine);
psmt.executeUpdate();
psmt.close();
res.close();}
catch(Exception e){
    System.out.println(e);}
        try{
System.out.println("CONNECTION CLOSING FOR Agent "+count);
client.close();}//closing server connection
catch(IOException e){//if there's any problem while closing server
System.out.println("********SOCKET CLOSED ERROR********");}}}
```

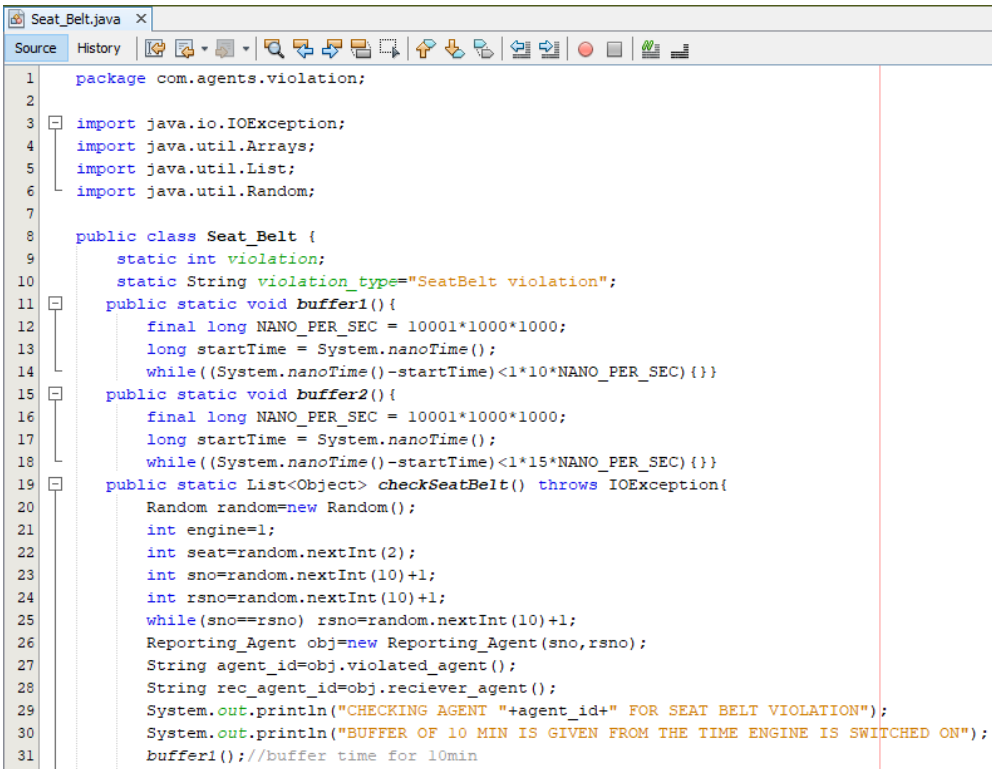
Main Agent class and its connected classes:-

1. Agent class:-

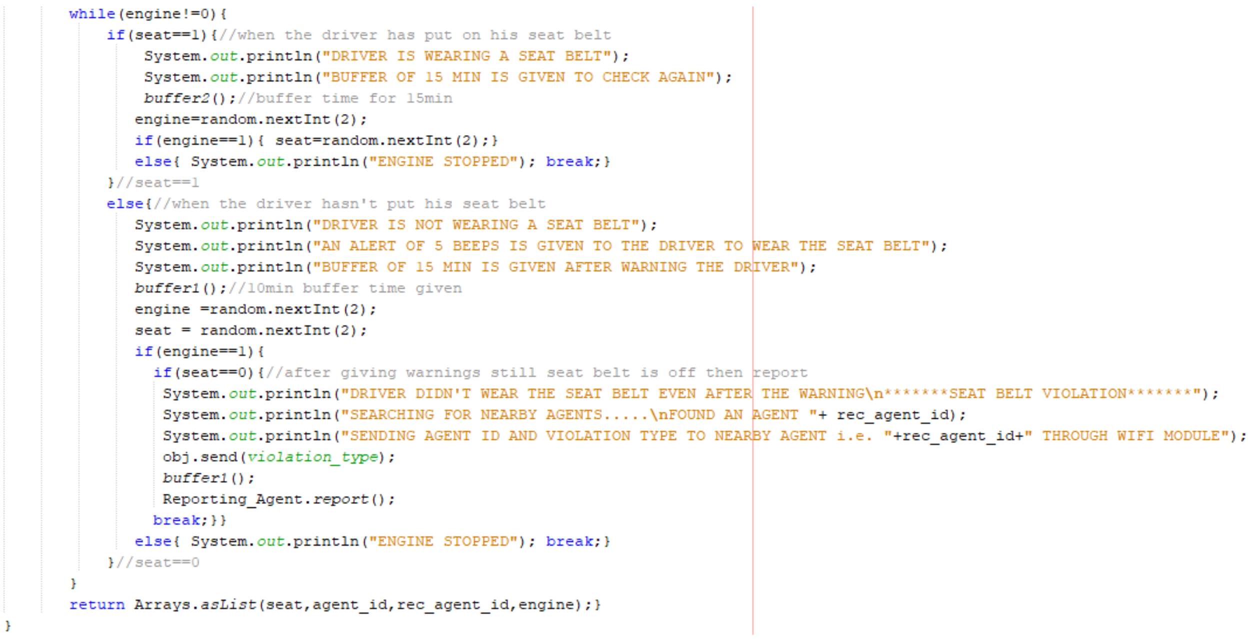


```
package com.agents.violation;

import static com.agents.violation.Seat_Belt.checkSeatBelt;
import static com.agents.violation.SpeedLimit.check_Speed;
import static com.agents.violation.Uninsured_Vehicle.getInsuranceDetail;
import static com.agents.violation.WrongDirection.check_Direction;
import java.io.IOException;
import java.util.List;

public class Agent {
    public static void main(String args[])throws IOException{
         final long NANO_PER_SEC = 1000l*1000*1000;
       long startTime = System.nanoTime();
       while((System.nanoTime()-startTime)<60*60*NANO_PER_SEC){//checking each violation by calling each clas for 1hrs
               //SEAT BELT
               System.out.println("CHECKING FOR SEAT BELT VIOLATION....");
               Seat_Belt obj1=new Seat_Belt();
               List<Object> check_seat_belt=checkSeatBelt();
               int seat=(int) check_seat_belt.get(0);
               int engine=(int) check_seat_belt.get(3);
               if(seat==0&&engine==1){
                   System.out.println("AGENT "+check_seat_belt.get(1)+" HAS VIOLATED SEAT BELT RULE AND "
                           + "THIS VIOLATION HAS BEEN REPORTED BY AGENT "+check_seat_belt.get(2)
                           +" AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY");}
               //WRONG DIRECTION
               System.out.println("------------------------------------------------------------------");
               System.out.println("CHECKING FOR WRONG DIRECTION VIOLATION....");
                   WrongDirection obj2 = new WrongDirection();
                   List<Object> check_wrong_direction = check_Direction();
                   int flag = (int)check_wrong_direction.get(0);
                   String area = (String)check_wrong_direction.get(1);
                   if(flag == 0){
                       System.out.println("NO AGENT HAS BEEN FOUND IN THE VICINITY OF AGENT "+check_wrong_direction.get(3));}
    else if(flag == 1){
        System.out.println("AGENT "+check_wrong_direction.get(2)+" HAS VIOLATED WRONG DIRECTION RULE AND "
            + "THIS VIOLATION HAS BEEN REPORTED BY AGENT "+check_wrong_direction.get(3)+" IN "+area+" RAJAJINAGAR"
         +" AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY"); count_direction++;}
    else
        System.out.println("AGENT'S DIRECTION IS SAME AS THE PERMITTED DIRECTION!!\n"
                + "NO VIOLATING AGENT HAS BEEN FOUND IN THE VICINITY");
//OVERSPEEDING
System.out.println("-------------------------------------------------------------------------");
System.out.println("CHECKING FOR OVERSPEEDING VIOLATION....");
SpeedLimit obj3 = new SpeedLimit();
    List<Object> check_overspeeding = check_Speed();
    int flag1 = (int)check_overspeeding.get(0);
    String area1 = (String)check_overspeeding.get(1);
    if(flag1 == 0){
        System.out.println("NO AGENT HAS BEEN FOUND IN THE VICINITY OF AGENT "+check_overspeeding.get(3));}
    else if(flag1 == 1){
        System.out.println("AGENT "+check_overspeeding.get(2)+" HAS VIOLATED OVERSPEEDING RULE AND "
            + "THIS VIOLATION HAS BEEN REPORTED BY AGENT "+check_overspeeding.get(3)+" IN  "+area1+" RAJAJINAGAR"
        +" AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY"); count_speed++;}
    else System.out.println("NO VIOLATING AGENT HAS BEEN FOUND IN THE VICINITY ");
//UNINSURED VEHICLE
System.out.println("-------------------------------------------------------------------------");
System.out.println("CHECKING INSURANCE OF THE VEHICLE....");
Uninsured_Vehicle obj4=new Uninsured_Vehicle();
List<Object> check_insurance = getInsuranceDetail();
int check= (int) check_insurance.get(1);
if(check==1)
System.out.println("AGENT "+check_insurance.get(0)+" HAS A VALID INSURANCE DATE.\nTHUS EXPIRY IS YET TO COME!!");
else{
  System.out.println("AGENT "+check_insurance.get(0)+" HAS VIOLATED INSURANCE VIOLATION WHICH HAS BEEN "
+ "REPORTED TO THE SERVER AND A NOTIFICATION ABOUT THIS VIOLATION HAS BEEN SENT TO THE USER SUCCESSFULLY");count_insurance++;}
System.out.println("-------------------------------------------------------------------------");}
System.out.println("NUMBER OF TIMES SEATBELT VIOLATION COMMITED AFTER SIMULATING FOR 1 HOUR: "+count_seat);
System.out.println("NUMBER OF TIMES WRONG DIRECTION VIOLATION COMMITED AFTER SIMULATING FOR 1 HOUR: "+count_direction);
System.out.println("NUMBER OF TIMES OVERSPEEDING VIOLATION COMMITTED AFTER SIMULATING FOR 1 HOUR: "+count_speed);
System.out.println("NUMBER OF TIMES UNINSURED VEHICLE VIOLATION COMMITTED AFTER SIMULATING FOR 1 HOUR: "+count_insurance);}}
```

2. Seat_Belt class:-

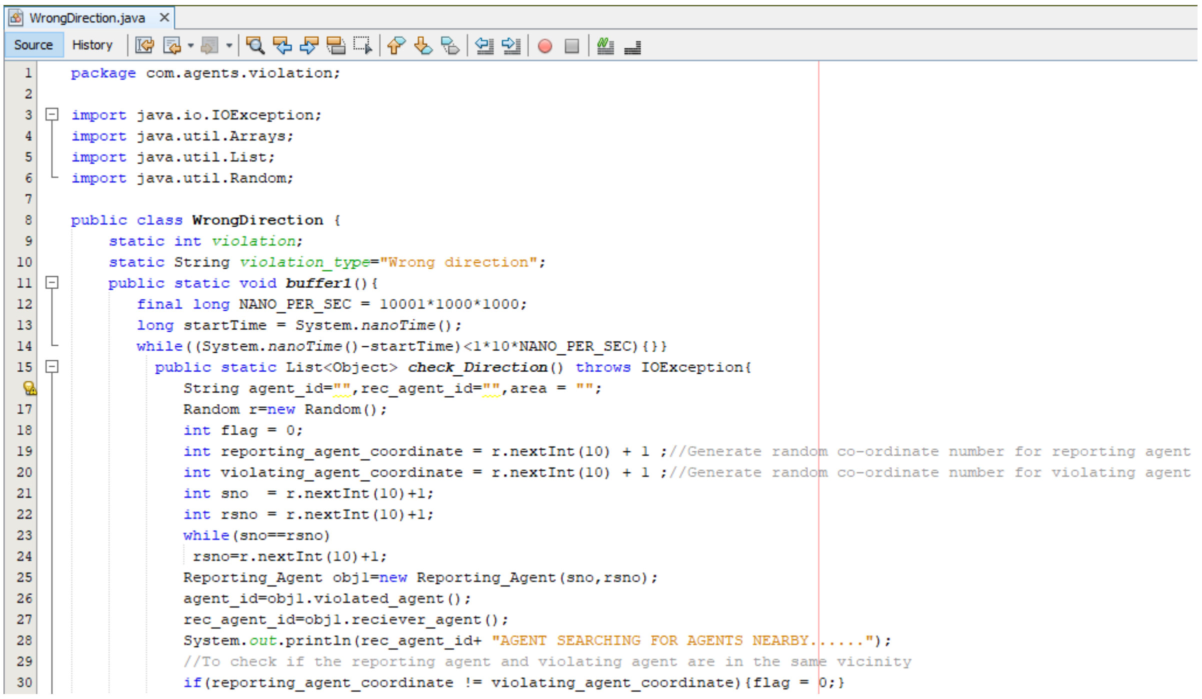


```
package com.agents.violation;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class Seat_Belt {
    static int violation;
    static String violation_type="SeatBelt violation";
   public static void buffer1(){
       final long NANO_PER_SEC = 1000l*1000*1000;
       long startTime = System.nanoTime();
       while((System.nanoTime()-startTime)<1*10*NANO_PER_SEC){}}
   public static void buffer2(){
       final long NANO_PER_SEC = 1000l*1000*1000;
       long startTime = System.nanoTime();
       while((System.nanoTime()-startTime)<1*15*NANO_PER_SEC){}}
   public static List<Object> checkSeatBelt() throws IOException{
       Random random=new Random();
       int engine=1;
       int seat=random.nextInt(2);
       int sno=random.nextInt(10)+1;
       int rsno=random.nextInt(10)+1;
       while(sno==rsno) rsno=random.nextInt(10)+1;
       Reporting_Agent obj=new Reporting_Agent(sno,rsno);
       String agent_id=obj.violated_agent();
       String rec_agent_id=obj.reciever_agent();
       System.out.println("CHECKING AGENT "+agent_id+" FOR SEAT BELT VIOLATION");
       System.out.println("BUFFER OF 10 MIN IS GIVEN FROM THE TIME ENGINE IS SWITCHED ON");
       buffer1();//buffer time for 10min
```

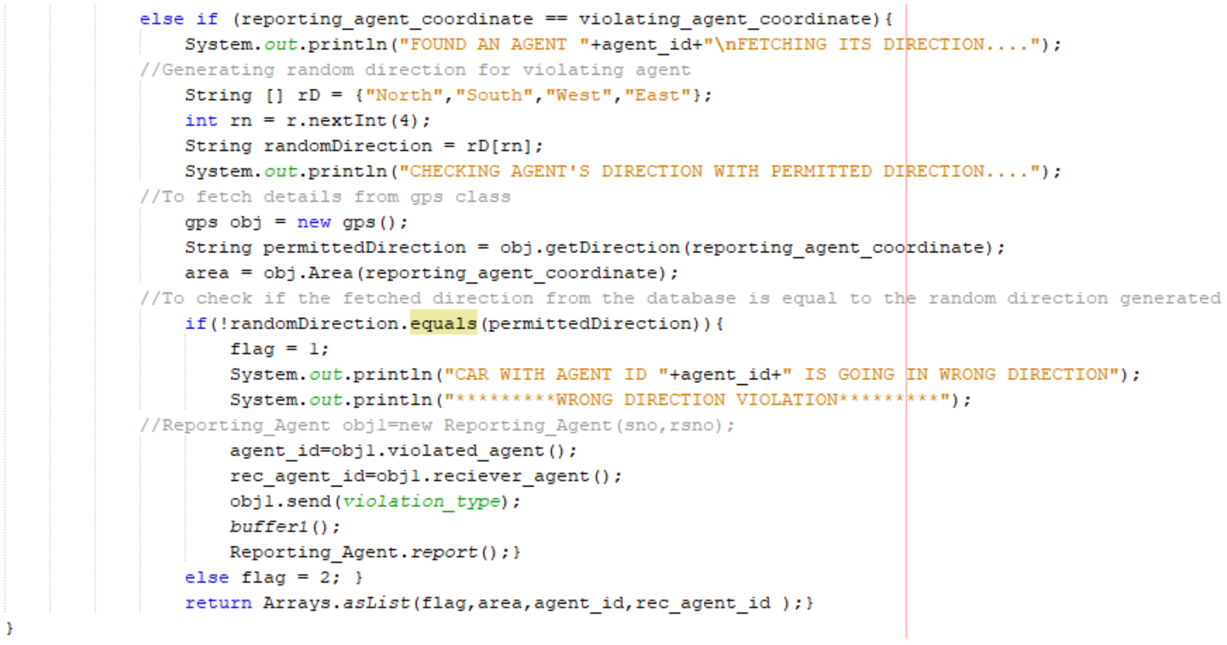


```java
        while(engine!=0){
            if(seat==1){//when the driver has put on his seat belt
                System.out.println("DRIVER IS WEARING A SEAT BELT");
                System.out.println("BUFFER OF 15 MIN IS GIVEN TO CHECK AGAIN");
                buffer2();//buffer time for 15min
               engine=random.nextInt(2);
               if(engine==1){ seat=random.nextInt(2);}
               else{ System.out.println("ENGINE STOPPED"); break;}
            }//seat==1
            else{//when the driver hasn't put his seat belt
               System.out.println("DRIVER IS NOT WEARING A SEAT BELT");
               System.out.println("AN ALERT OF 5 BEEPS IS GIVEN TO THE DRIVER TO WEAR THE SEAT BELT");
               System.out.println("BUFFER OF 15 MIN IS GIVEN AFTER WARNING THE DRIVER");
               buffer1();//10min buffer time given
               engine =random.nextInt(2);
               seat = random.nextInt(2);
               if(engine==1){
                 if(seat==0){//after giving warnings still seat belt is off then report
                  System.out.println("DRIVER DIDN'T WEAR THE SEAT BELT EVEN AFTER THE WARNING\n*******SEAT BELT VIOLATION*******");
                  System.out.println("SEARCHING FOR NEARBY AGENTS.....\nFOUND AN AGENT "+ rec_agent_id);
                  System.out.println("SENDING AGENT ID AND VIOLATION TYPE TO NEARBY AGENT i.e. "+rec_agent_id+" THROUGH WIFI MODULE");
                  obj.send(violation_type);
                  buffer1();
                  Reporting_Agent.report();
                 break;}}
               else{ System.out.println("ENGINE STOPPED"); break;}
            }//seat==0
        }
        return Arrays.asList(seat,agent_id,rec_agent_id,engine);}
}
```

3. WrongDirection class:-

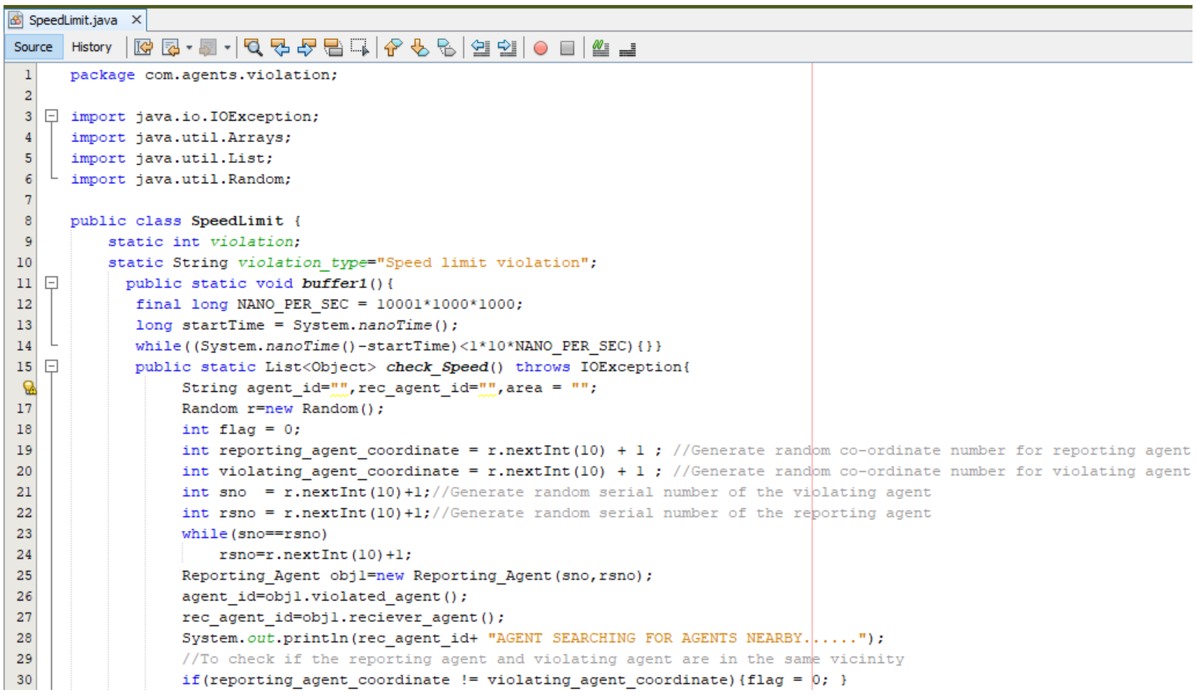


```java
package com.agents.violation;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class WrongDirection {
    static int violation;
    static String violation_type="Wrong direction";
    public static void buffer1(){
       final long NANO_PER_SEC = 10001*1000*1000;
       long startTime = System.nanoTime();
       while((System.nanoTime()-startTime)<1*10*NANO_PER_SEC){}}
         public static List<Object> check_Direction() throws IOException{
            String agent_id="",rec_agent_id="",area = "";
            Random r=new Random();
            int flag = 0;
            int reporting_agent_coordinate = r.nextInt(10) + 1 ;//Generate random co-ordinate number for reporting agent
            int violating_agent_coordinate = r.nextInt(10) + 1 ;//Generate random co-ordinate number for violating agent
            int sno  = r.nextInt(10)+1;
            int rsno = r.nextInt(10)+1;
            while(sno==rsno)
             rsno=r.nextInt(10)+1;
            Reporting_Agent obj1=new Reporting_Agent(sno,rsno);
            agent_id=obj1.violated_agent();
            rec_agent_id=obj1.reciever_agent();
            System.out.println(rec_agent_id+ "AGENT SEARCHING FOR AGENTS NEARBY......");
            //To check if the reporting agent and violating agent are in the same vicinity
            if(reporting_agent_coordinate != violating_agent_coordinate){flag = 0;}
```

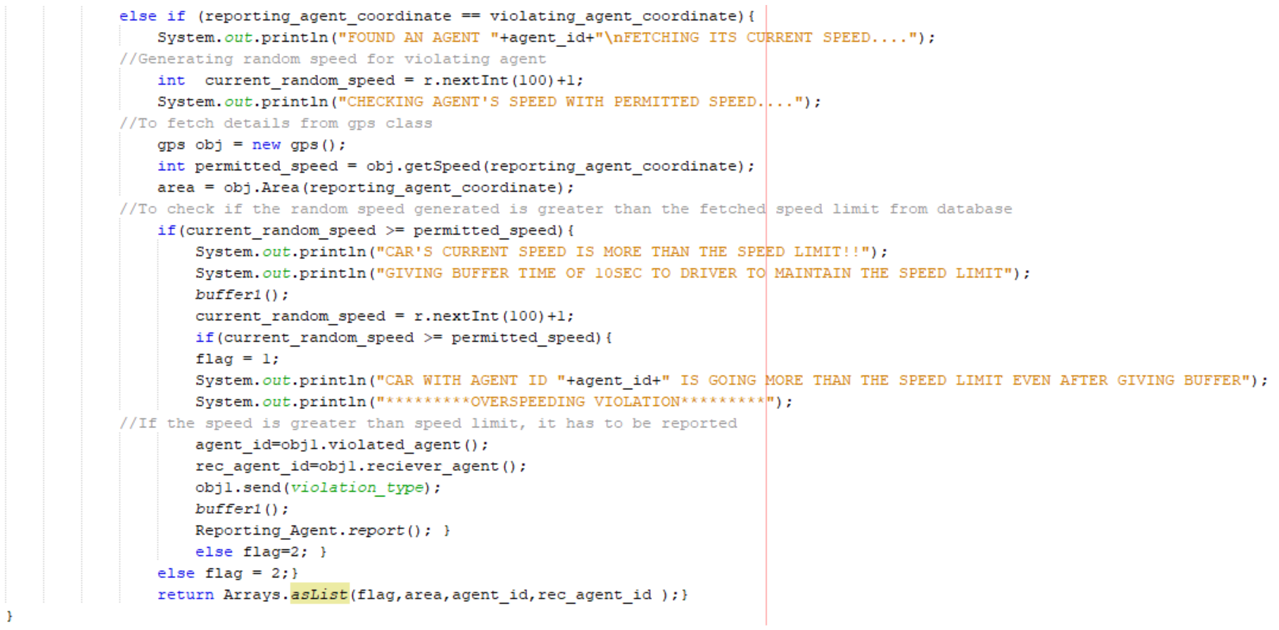


```
            else if (reporting_agent_coordinate == violating_agent_coordinate){
                System.out.println("FOUND AN AGENT "+agent_id+"\nFETCHING ITS DIRECTION....");
            //Generating random direction for violating agent
                String [] rD = {"North","South","West","East"};
                int rn = r.nextInt(4);
                String randomDirection = rD[rn];
                System.out.println("CHECKING AGENT'S DIRECTION WITH PERMITTED DIRECTION....");
            //To fetch details from gps class
                gps obj = new gps();
                String permittedDirection = obj.getDirection(reporting_agent_coordinate);
                area = obj.Area(reporting_agent_coordinate);
            //To check if the fetched direction from the database is equal to the random direction generated
                if(!randomDirection.equals(permittedDirection)){
                    flag = 1;
                    System.out.println("CAR WITH AGENT ID "+agent_id+" IS GOING IN WRONG DIRECTION");
                    System.out.println("*********WRONG DIRECTION VIOLATION**********");
            //Reporting_Agent obj1=new Reporting_Agent(sno,rsno);
                    agent_id=obj1.violated_agent();
                    rec_agent_id=obj1.reciever_agent();
                    obj1.send(violation_type);
                    buffer1();
                    Reporting_Agent.report();}
                else flag = 2; }
                return Arrays.asList(flag,area,agent_id,rec_agent_id );}
}
```

4. SpeedLimit class:-

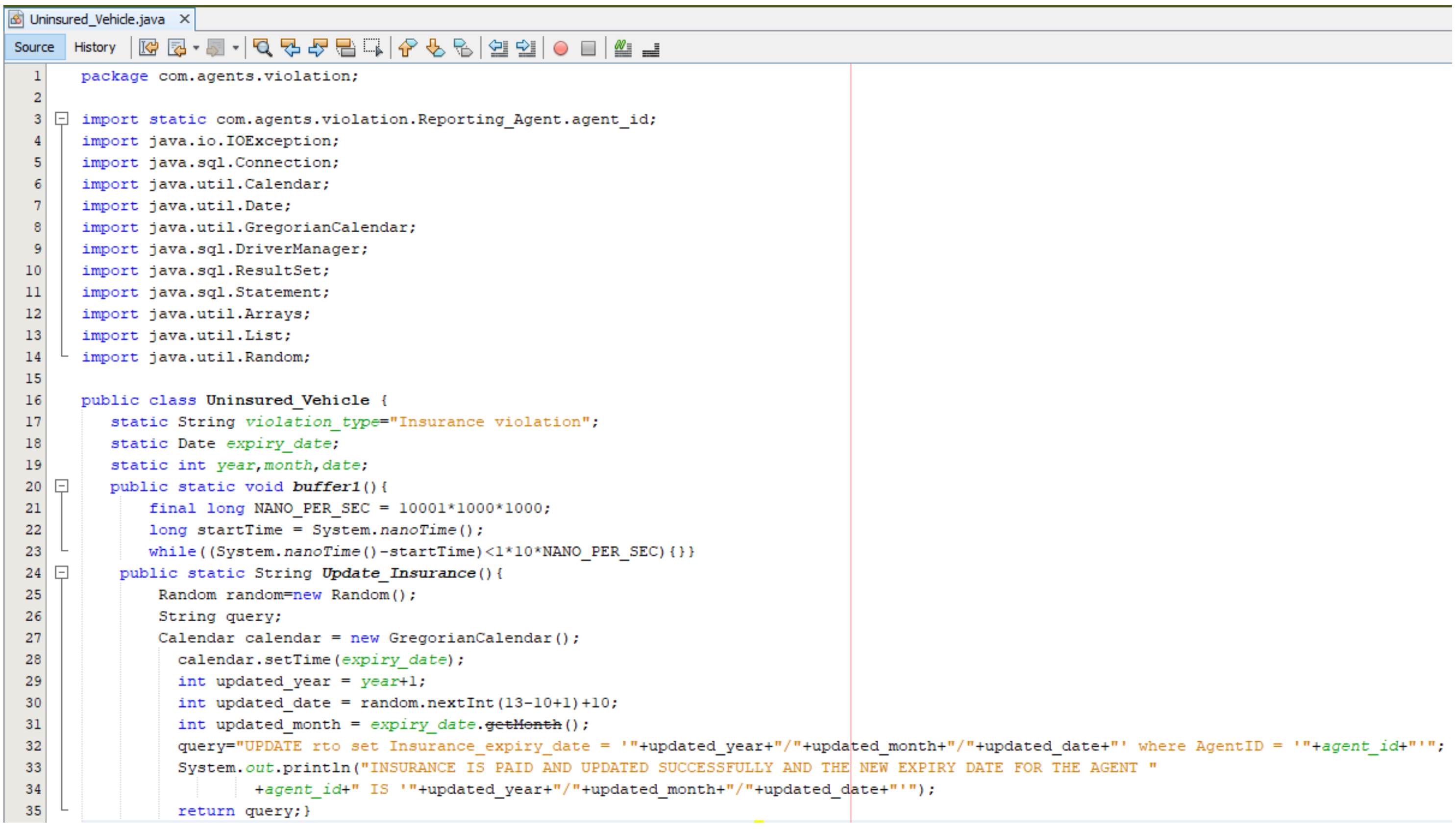


```
package com.agents.violation;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class SpeedLimit {
    static int violation;
    static String violation_type="Speed limit violation";
      public static void buffer1(){
       final long NANO_PER_SEC = 10001*1000*1000;
       long startTime = System.nanoTime();
       while((System.nanoTime()-startTime)<1*10*NANO_PER_SEC){}}
       public static List<Object> check_Speed() throws IOException{
            String agent_id="",rec_agent_id="",area = "";
            Random r=new Random();
            int flag = 0;
            int reporting_agent_coordinate = r.nextInt(10) + 1 ; //Generate random co-ordinate number for reporting agent
            int violating_agent_coordinate = r.nextInt(10) + 1 ; //Generate random co-ordinate number for violating agent
            int sno  = r.nextInt(10)+1;//Generate random serial number of the violating agent
            int rsno = r.nextInt(10)+1;//Generate random serial number of the reporting agent
            while(sno==rsno)
                rsno=r.nextInt(10)+1;
            Reporting_Agent obj1=new Reporting_Agent(sno,rsno);
            agent_id=obj1.violated_agent();
            rec_agent_id=obj1.reciever_agent();
            System.out.println(rec_agent_id+ "AGENT SEARCHING FOR AGENTS NEARBY......");
            //To check if the reporting agent and violating agent are in the same vicinity
            if(reporting_agent_coordinate != violating_agent_coordinate){flag = 0; }
```



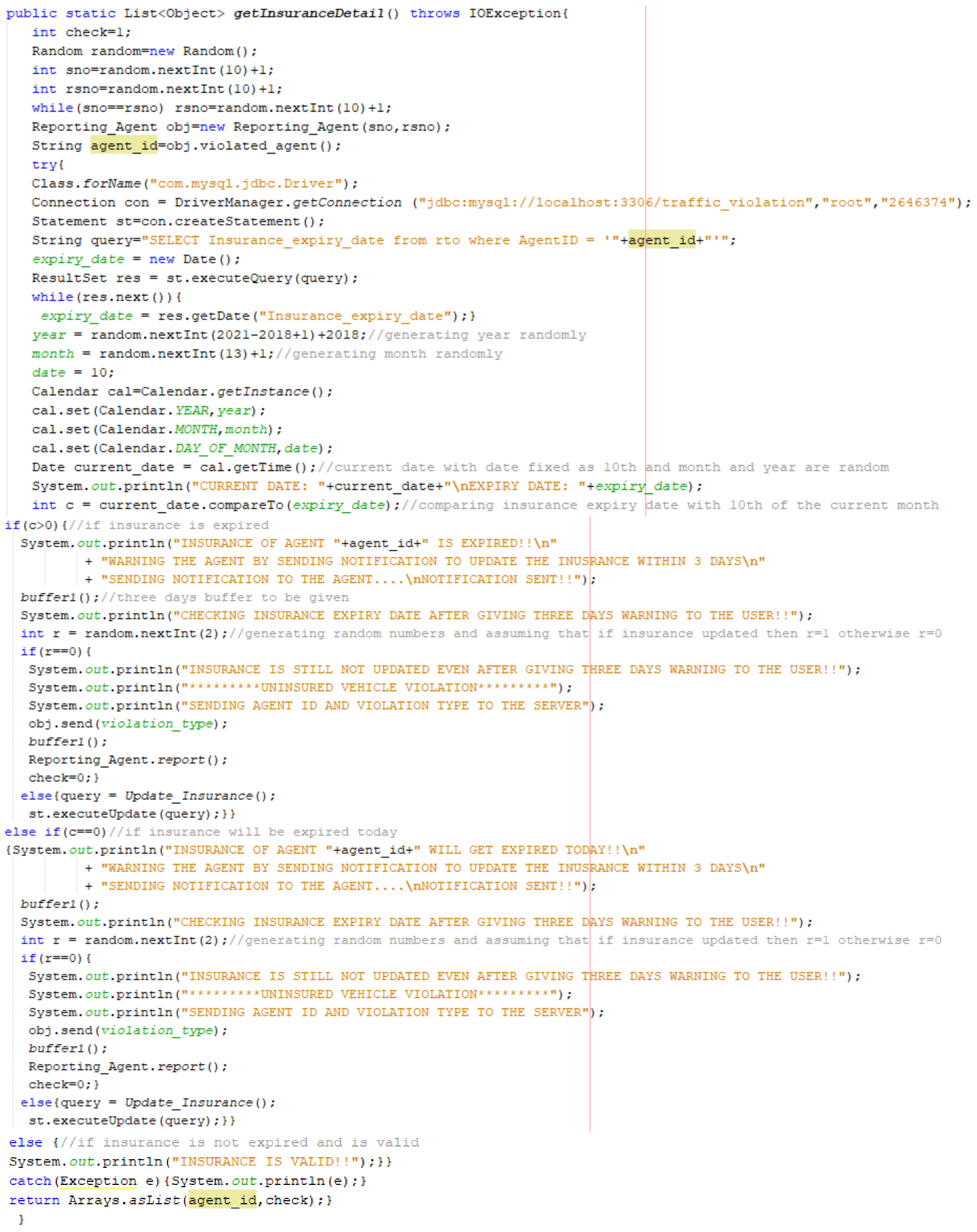
```
            else if (reporting_agent_coordinate == violating_agent_coordinate){
                System.out.println("FOUND AN AGENT "+agent_id+"\nFETCHING ITS CURRENT SPEED....");
            //Generating random speed for violating agent
                int  current_random_speed = r.nextInt(100)+1;
                System.out.println("CHECKING AGENT'S SPEED WITH PERMITTED SPEED.....");
            //To fetch details from gps class
                gps obj = new gps();
                int permitted_speed = obj.getSpeed(reporting_agent_coordinate);
                area = obj.Area(reporting_agent_coordinate);
            //To check if the random speed generated is greater than the fetched speed limit from database
                if(current_random_speed >= permitted_speed){
                    System.out.println("CAR'S CURRENT SPEED IS MORE THAN THE SPEED LIMIT!!");
                    System.out.println("GIVING BUFFER TIME OF 10SEC TO DRIVER TO MAINTAIN THE SPEED LIMIT");
                    buffer1();
                    current_random_speed = r.nextInt(100)+1;
                    if(current_random_speed >= permitted_speed){
                    flag = 1;
                    System.out.println("CAR WITH AGENT ID "+agent_id+" IS GOING MORE THAN THE SPEED LIMIT EVEN AFTER GIVING BUFFER");
                    System.out.println("*********OVERSPEEDING VIOLATION*********");
            //If the speed is greater than speed limit, it has to be reported
                    agent_id=obj1.violated_agent();
                    rec_agent_id=obj1.reciever_agent();
                    obj1.send(violation_type);
                    buffer1();
                    Reporting_Agent.report(); }
                    else flag=2; }
                else flag = 2;}
                return Arrays.asList(flag,area,agent_id,rec_agent_id );}
}
```

5. Unisured_Vehicle class:-



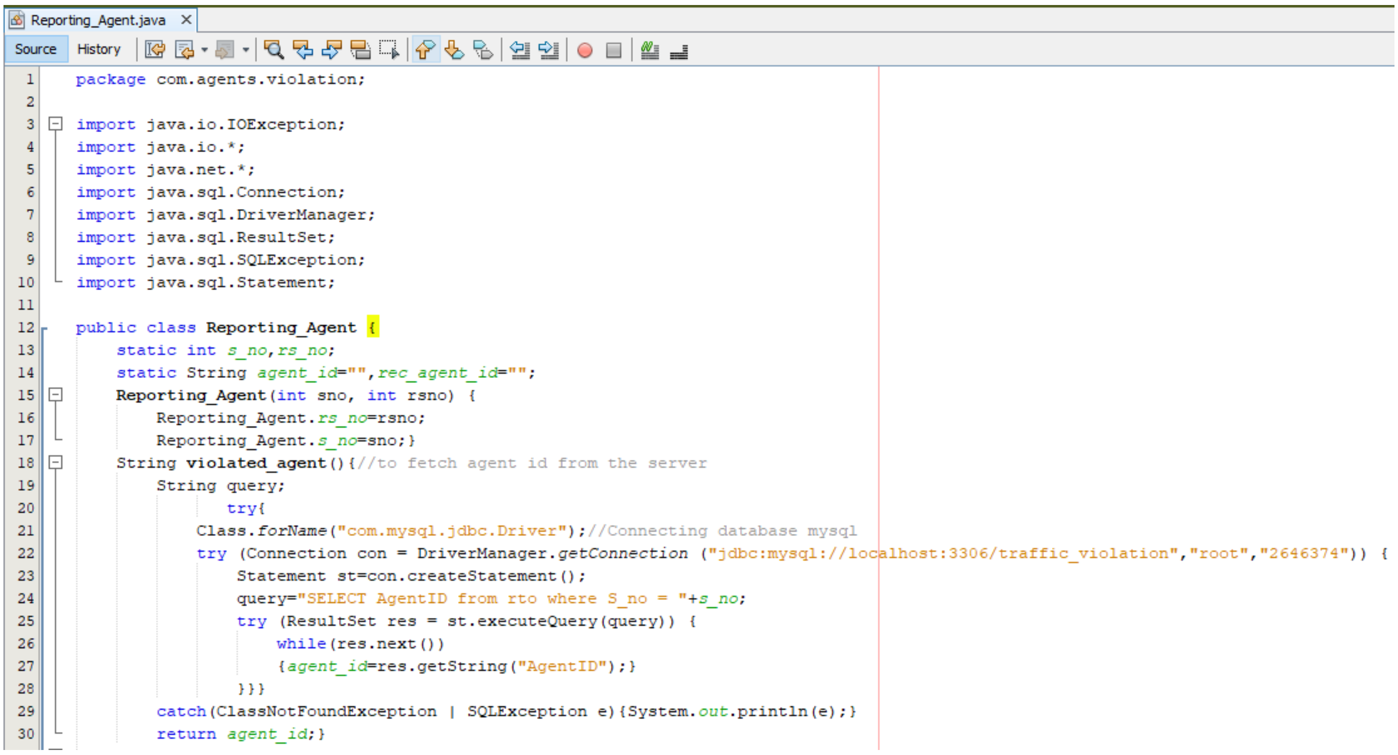
Uninsured_Vehicle.java

```
package com.agents.violation;

import static com.agents.violation.Reporting_Agent.agent_id;
import java.io.IOException;
import java.sql.Connection;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class Uninsured_Vehicle {
    static String violation_type="Insurance violation";
    static Date expiry_date;
    static int year,month,date;
    public static void buffer1(){
        final long NANO_PER_SEC = 1000l*1000*1000;
        long startTime = System.nanoTime();
        while((System.nanoTime()-startTime)<1*10*NANO_PER_SEC){}}
     public static String Update_Insurance(){
         Random random=new Random();
         String query;
         Calendar calendar = new GregorianCalendar();
           calendar.setTime(expiry_date);
           int updated_year = year+1;
           int updated_date = random.nextInt(13-10+1)+10;
           int updated_month = expiry_date.~~getMonth~~();
           query="UPDATE rto set Insurance_expiry_date = '"+updated_year+"/"+updated_month+"/"+updated_date+"' where AgentID = '"+agent_id+"'";
           System.out.println("INSURANCE IS PAID AND UPDATED SUCCESSFULLY AND THE NEW EXPIRY DATE FOR THE AGENT "
                   +agent_id+" IS '"+updated_year+"/"+updated_month+"/"+updated_date+"'");
           return query;}
```

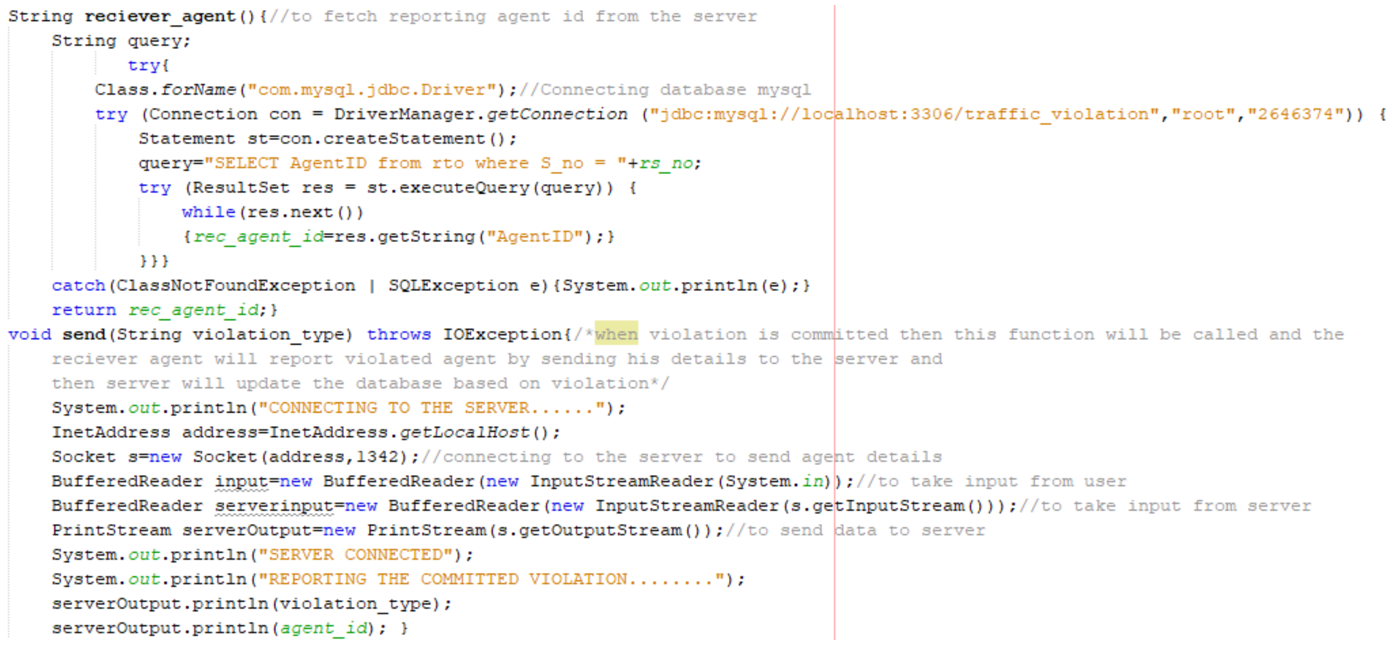


```
public static List<Object> getInsuranceDetail() throws IOException{
   int check=1;
   Random random=new Random();
   int sno=random.nextInt(10)+1;
   int rsno=random.nextInt(10)+1;
   while(sno==rsno) rsno=random.nextInt(10)+1;
   Reporting_Agent obj=new Reporting_Agent(sno,rsno);
   String agent_id=obj.violated_agent();
   try{
   Class.forName("com.mysql.jdbc.Driver");
   Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374");
   Statement st=con.createStatement();
   String query="SELECT Insurance_expiry_date from rto where AgentID = '"+agent_id+"'";
   expiry_date = new Date();
   ResultSet res = st.executeQuery(query);
   while(res.next()){
    expiry_date = res.getDate("Insurance_expiry_date");}
   year = random.nextInt(2021-2018+1)+2018;//generating year randomly
   month = random.nextInt(13)+1;//generating month randomly
   date = 10;
   Calendar cal=Calendar.getInstance();
   cal.set(Calendar.YEAR,year);
   cal.set(Calendar.MONTH,month);
   cal.set(Calendar.DAY_OF_MONTH,date);
   Date current_date = cal.getTime();//current date with date fixed as 10th and month and year are random
   System.out.println("CURRENT DATE: "+current_date+"\nEXPIRY DATE: "+expiry_date);
   int c = current_date.compareTo(expiry_date);//comparing insurance expiry date with 10th of the current month
if(c>0){//if insurance is expired
  System.out.println("INSURANCE OF AGENT "+agent_id+" IS EXPIRED!!\n"
              + "WARNING THE AGENT BY SENDING NOTIFICATION TO UPDATE THE INUSRANCE WITHIN 3 DAYS\n"
              + "SENDING NOTIFICATION TO THE AGENT....\nNOTIFICATION SENT!!");
  buffer1();//three days buffer to be given
  System.out.println("CHECKING INSURANCE EXPIRY DATE AFTER GIVING THREE DAYS WARNING TO THE USER!!");
  int r = random.nextInt(2);//generating random numbers and assuming that if insurance updated then r=1 otherwise r=0
  if(r==0){
   System.out.println("INSURANCE IS STILL NOT UPDATED EVEN AFTER GIVING THREE DAYS WARNING TO THE USER!!");
   System.out.println("*********UNINSURED VEHICLE VIOLATION*********");
   System.out.println("SENDING AGENT ID AND VIOLATION TYPE TO THE SERVER");
   obj.send(violation_type);
   buffer1();
   Reporting_Agent.report();
   check=0;}
  else{query = Update_Insurance();
   st.executeUpdate(query);}}
else if(c==0)//if insurance will be expired today
{System.out.println("INSURANCE OF AGENT "+agent_id+" WILL GET EXPIRED TODAY!!\n"
              + "WARNING THE AGENT BY SENDING NOTIFICATION TO UPDATE THE INUSRANCE WITHIN 3 DAYS\n"
              + "SENDING NOTIFICATION TO THE AGENT....\nNOTIFICATION SENT!!");
  buffer1();
  System.out.println("CHECKING INSURANCE EXPIRY DATE AFTER GIVING THREE DAYS WARNING TO THE USER!!");
  int r = random.nextInt(2);//generating random numbers and assuming that if insurance updated then r=1 otherwise r=0
  if(r==0){
   System.out.println("INSURANCE IS STILL NOT UPDATED EVEN AFTER GIVING THREE DAYS WARNING TO THE USER!!");
   System.out.println("*********UNINSURED VEHICLE VIOLATION*********");
   System.out.println("SENDING AGENT ID AND VIOLATION TYPE TO THE SERVER");
   obj.send(violation_type);
   buffer1();
   Reporting_Agent.report();
   check=0;}
  else{query = Update_Insurance();
   st.executeUpdate(query);}}
 else {//if insurance is not expired and is valid
 System.out.println("INSURANCE IS VALID!!");}}
 catch(Exception e){System.out.println(e);}
 return Arrays.asList(agent_id,check);}
  }
```

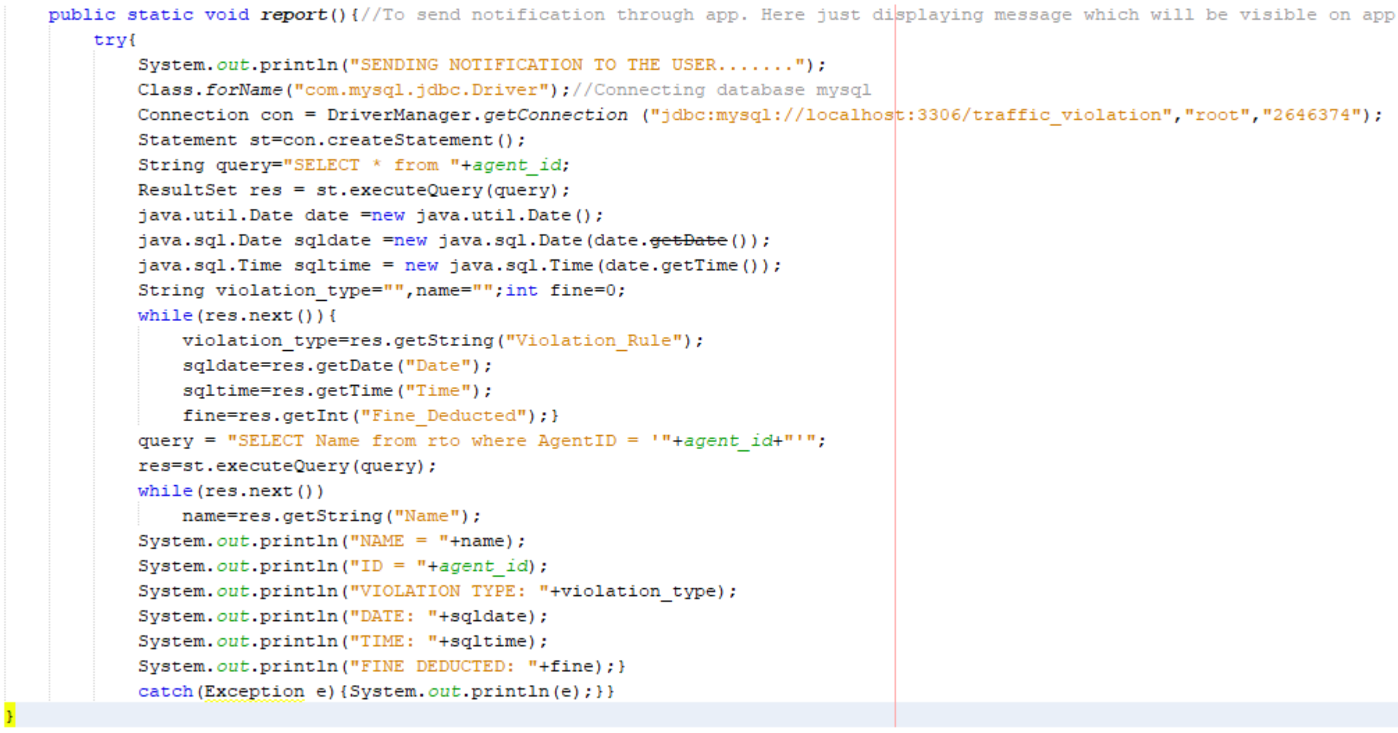
6. Reporting_Agent class:-



```java
package com.agents.violation;

import java.io.IOException;
import java.io.*;
import java.net.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Reporting_Agent {
    static int s_no,rs_no;
    static String agent_id="",rec_agent_id="";
    Reporting_Agent(int sno, int rsno) {
        Reporting_Agent.rs_no=rsno;
        Reporting_Agent.s_no=sno;}
    String violated_agent(){//to fetch agent id from the server
        String query;
               try{
            Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
            try (Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374")) {
                Statement st=con.createStatement();
                query="SELECT AgentID from rto where S_no = "+s_no;
                try (ResultSet res = st.executeQuery(query)) {
                    while(res.next())
                    {agent_id=res.getString("AgentID");}
                }}}
        catch(ClassNotFoundException | SQLException e){System.out.println(e);}
        return agent_id;}
```

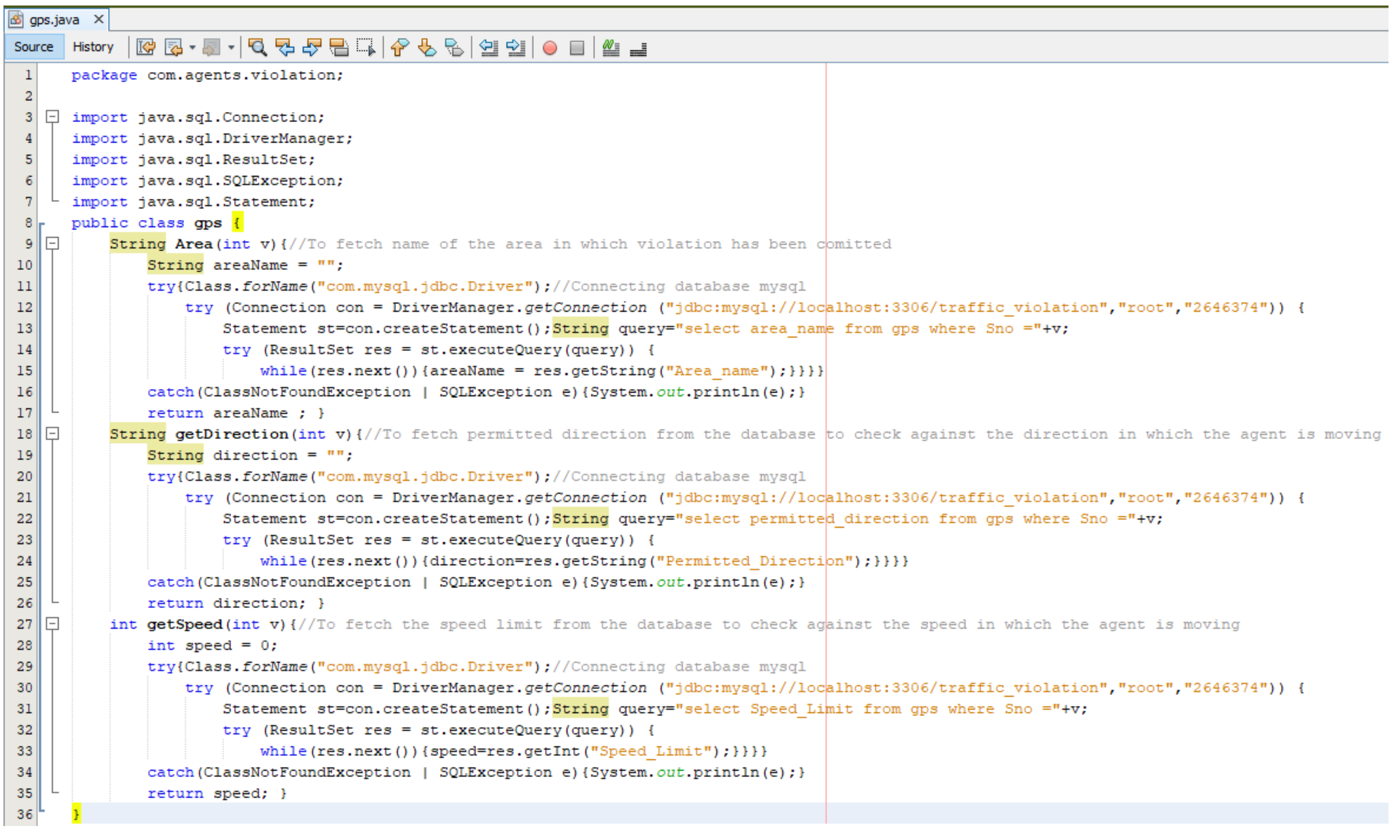


```java
String reciever_agent(){//to fetch reporting agent id from the server
    String query;
           try{
        Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
        try (Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374")) {
            Statement st=con.createStatement();
            query="SELECT AgentID from rto where S_no = "+rs_no;
            try (ResultSet res = st.executeQuery(query)) {
                while(res.next())
                {rec_agent_id=res.getString("AgentID");}
            }}}
    catch(ClassNotFoundException | SQLException e){System.out.println(e);}
    return rec_agent_id;}
void send(String violation_type) throws IOException{/*when violation is committed then this function will be called and the
    reciever agent will report violated agent by sending his details to the server and
    then server will update the database based on violation*/
    System.out.println("CONNECTING TO THE SERVER......");
    InetAddress address=InetAddress.getLocalHost();
    Socket s=new Socket(address,1342);//connecting to the server to send agent details
    BufferedReader input=new BufferedReader(new InputStreamReader(System.in));//to take input from user
    BufferedReader serverinput=new BufferedReader(new InputStreamReader(s.getInputStream()));//to take input from server
    PrintStream serverOutput=new PrintStream(s.getOutputStream());//to send data to server
    System.out.println("SERVER CONNECTED");
    System.out.println("REPORTING THE COMMITTED VIOLATION........");
    serverOutput.println(violation_type);
    serverOutput.println(agent_id); }
```



```
    public static void report(){//To send notification through app. Here just displaying message which will be visible on app
        try{
            System.out.println("SENDING NOTIFICATION TO THE USER.......");
            Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
            Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374");
            Statement st=con.createStatement();
            String query="SELECT * from "+agent_id;
            ResultSet res = st.executeQuery(query);
            java.util.Date date =new java.util.Date();
            java.sql.Date sqldate =new java.sql.Date(date.~~getDate~~());
            java.sql.Time sqltime = new java.sql.Time(date.getTime());
            String violation_type="",name="";int fine=0;
            while(res.next()){
                violation_type=res.getString("Violation_Rule");
                sqldate=res.getDate("Date");
                sqltime=res.getTime("Time");
                fine=res.getInt("Fine_Deducted");}
            query = "SELECT Name from rto where AgentID = '"+agent_id+"'";
            res=st.executeQuery(query);
            while(res.next())
                name=res.getString("Name");
            System.out.println("NAME = "+name);
            System.out.println("ID = "+agent_id);
            System.out.println("VIOLATION TYPE: "+violation_type);
            System.out.println("DATE: "+sqldate);
            System.out.println("TIME: "+sqltime);
            System.out.println("FINE DEDUCTED: "+fine);}
            catch(Exception e){System.out.println(e);}}
}
```

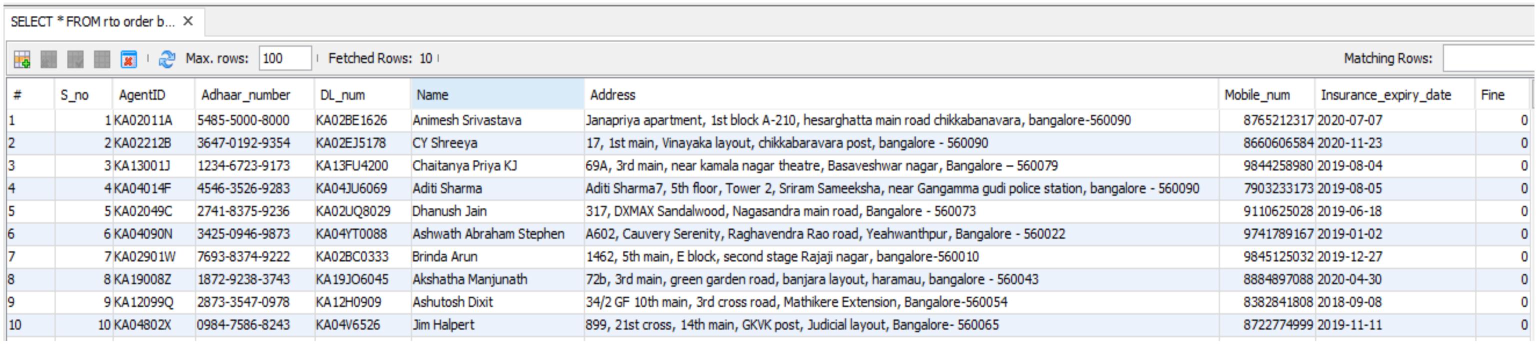
7. Gps class:-



```
package com.agents.violation;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class gps {
    String Area(int v){//To fetch name of the area in which violation has been comitted
        String areaName = "";
        try{Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
            try (Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374")) {
                Statement st=con.createStatement();String query="select area_name from gps where Sno ="+v;
                try (ResultSet res = st.executeQuery(query)) {
                    while(res.next()){areaName = res.getString("Area_name");}}}}
        catch(ClassNotFoundException | SQLException e){System.out.println(e);}
        return areaName ; }
    String getDirection(int v){//To fetch permitted direction from the database to check against the direction in which the agent is moving
        String direction = "";
        try{Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
            try (Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374")) {
                Statement st=con.createStatement();String query="select permitted_direction from gps where Sno ="+v;
                try (ResultSet res = st.executeQuery(query)) {
                    while(res.next()){direction=res.getString("Permitted_Direction");}}}}
        catch(ClassNotFoundException | SQLException e){System.out.println(e);}
        return direction; }
    int getSpeed(int v){//To fetch the speed limit from the database to check against the speed in which the agent is moving
        int speed = 0;
        try{Class.forName("com.mysql.jdbc.Driver");//Connecting database mysql
            try (Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/traffic_violation","root","2646374")) {
                Statement st=con.createStatement();String query="select Speed_Limit from gps where Sno ="+v;
                try (ResultSet res = st.executeQuery(query)) {
                    while(res.next()){speed=res.getInt("Speed_Limit");}}}}
        catch(ClassNotFoundException | SQLException e){System.out.println(e);}
        return speed; }
}
```

Part B: Screenshots for Changes in Databases

1) Initial rto database before running the program:

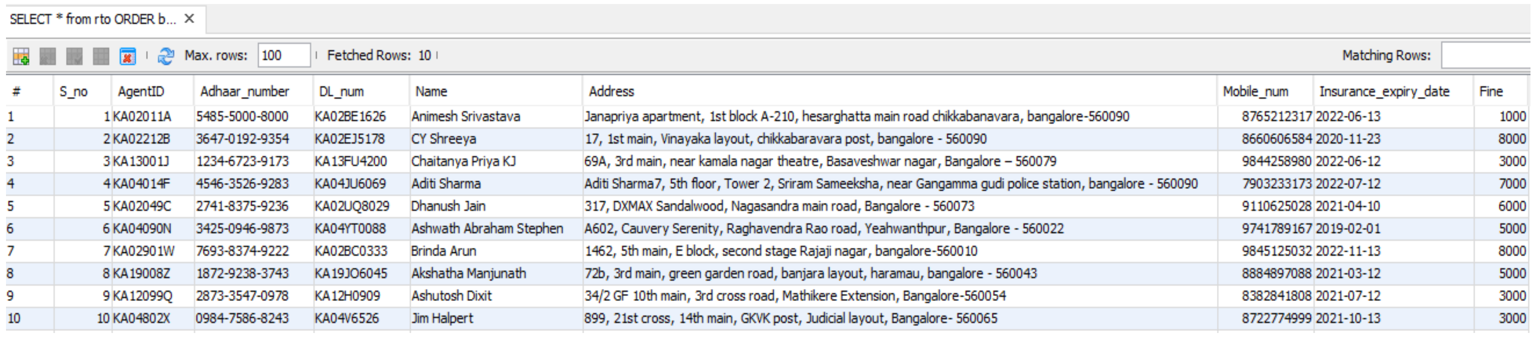


SELECT * FROM rto order b... ✕

Max. rows: 100 | Fetched Rows: 10 | Matching Rows:

#	S_no	AgentID	Adhaar_number	DL_num	Name	Address	Mobile_num	Insurance_expiry_date	Fine
1	1	KA02011A	5485-5000-8000	KA02BE1626	Animesh Srivastava	Janapriya apartment, 1st block A-210, hesarghatta main road chikkabanavara, bangalore-560090	8765212317	2020-07-07	0
2	2	KA02212B	3647-0192-9354	KA02EJ5178	CY Shreeya	17, 1st main, Vinayaka layout, chikkabaravara post, bangalore - 560090	8660606584	2020-11-23	0
3	3	KA13001J	1234-6723-9173	KA13FU4200	Chaitanya Priya KJ	69A, 3rd main, near kamala nagar theatre, Basaveshwar nagar, Bangalore – 560079	9844258980	2019-08-04	0
4	4	KA04014F	4546-3526-9283	KA04JU6069	Aditi Sharma	Aditi Sharma7, 5th floor, Tower 2, Sriram Sameeksha, near Gangamma gudi police station, bangalore - 560090	7903233173	2019-08-05	0
5	5	KA02049C	2741-8375-9236	KA02UQ8029	Dhanush Jain	317, DXMAX Sandalwood, Nagasandra main road, Bangalore - 560073	9110625028	2019-06-18	0
6	6	KA04090N	3425-0946-9873	KA04YT0088	Ashwath Abraham Stephen	A602, Cauvery Serenity, Raghavendra Rao road, Yeahwanthpur, Bangalore - 560022	9741789167	2019-01-02	0
7	7	KA02901W	7693-8374-9222	KA02BC0333	Brinda Arun	1462, 5th main, E block, second stage Rajaji nagar, bangalore-560010	9845125032	2019-12-27	0
8	8	KA19008Z	1872-9238-3743	KA19JO6045	Akshatha Manjunath	72b, 3rd main, green garden road, banjara layout, haramau, bangalore - 560043	8884897088	2020-04-30	0
9	9	KA12099Q	2873-3547-0978	KA12H0909	Ashutosh Dixit	34/2 GF 10th main, 3rd cross road, Mathikere Extension, Bangalore-560054	8382841808	2018-09-08	0
10	10	KA04802X	0984-7586-8243	KA04V6526	Jim Halpert	899, 21st cross, 14th main, GKVK post, Judicial layout, Bangalore- 560065	8722774999	2019-11-11	0

2) After running the program for 1hrs, the fine column in rto table got updated based on the violation:

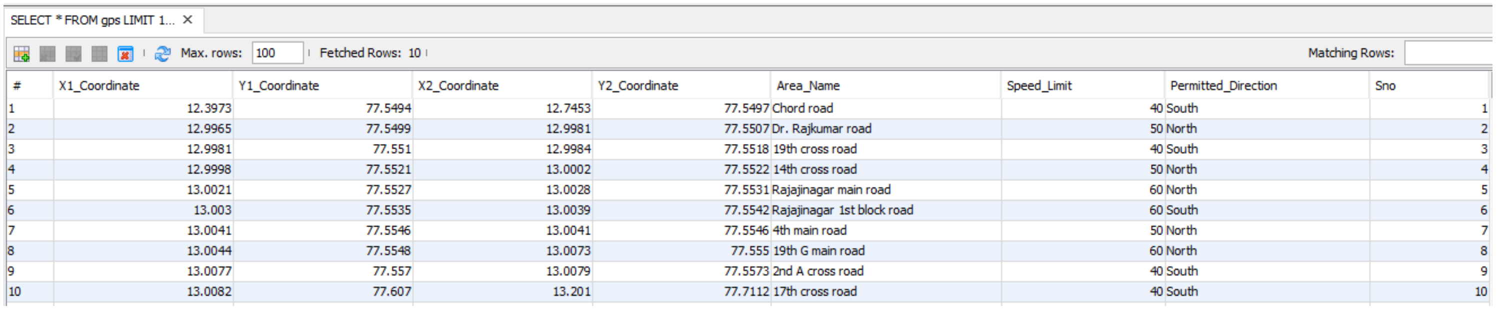


SELECT * from rto ORDER b... ✕

Max. rows: 100 | Fetched Rows: 10 | Matching Rows:

#	S_no	AgentID	Adhaar_number	DL_num	Name	Address	Mobile_num	Insurance_expiry_date	Fine
1	1	KA02011A	5485-5000-8000	KA02BE1626	Animesh Srivastava	Janapriya apartment, 1st block A-210, hesarghatta main road chikkabanavara, bangalore-560090	8765212317	2022-06-13	1000
2	2	KA02212B	3647-0192-9354	KA02EJ5178	CY Shreeya	17, 1st main, Vinayaka layout, chikkabaravara post, bangalore - 560090	8660606584	2020-11-23	8000
3	3	KA13001J	1234-6723-9173	KA13FU4200	Chaitanya Priya KJ	69A, 3rd main, near kamala nagar theatre, Basaveshwar nagar, Bangalore – 560079	9844258980	2022-06-12	3000
4	4	KA04014F	4546-3526-9283	KA04JU6069	Aditi Sharma	Aditi Sharma7, 5th floor, Tower 2, Sriram Sameeksha, near Gangamma gudi police station, bangalore - 560090	7903233173	2022-07-12	7000
5	5	KA02049C	2741-8375-9236	KA02UQ8029	Dhanush Jain	317, DXMAX Sandalwood, Nagasandra main road, Bangalore - 560073	9110625028	2021-04-10	6000
6	6	KA04090N	3425-0946-9873	KA04YT0088	Ashwath Abraham Stephen	A602, Cauvery Serenity, Raghavendra Rao road, Yeahwanthpur, Bangalore - 560022	9741789167	2019-02-01	5000
7	7	KA02901W	7693-8374-9222	KA02BC0333	Brinda Arun	1462, 5th main, E block, second stage Rajaji nagar, bangalore-560010	9845125032	2022-11-13	8000
8	8	KA19008Z	1872-9238-3743	KA19JO6045	Akshatha Manjunath	72b, 3rd main, green garden road, banjara layout, haramau, bangalore - 560043	8884897088	2021-03-12	5000
9	9	KA12099Q	2873-3547-0978	KA12H0909	Ashutosh Dixit	34/2 GF 10th main, 3rd cross road, Mathikere Extension, Bangalore-560054	8382841808	2021-07-12	3000
10	10	KA04802X	0984-7586-8243	KA04V6526	Jim Halpert	899, 21st cross, 14th main, GKVK post, Judicial layout, Bangalore- 560065	8722774999	2021-10-13	3000

3) Gps database which is used to get speed and permitted direction of some particular area:

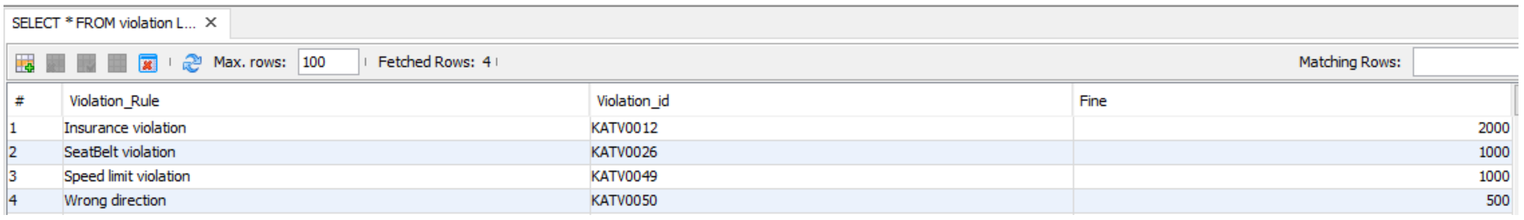


SELECT * FROM gps LIMIT 1... ✕

Max. rows: 100 | Fetched Rows: 10 | Matching Rows:

#	X1_Coordinate	Y1_Coordinate	X2_Coordinate	Y2_Coordinate	Area_Name	Speed_Limit	Permitted_Direction	Sno
1	12.3973	77.5494	12.7453	77.5497	Chord road	40	South	1
2	12.9965	77.5499	12.9981	77.5507	Dr. Rajkumar road	50	North	2
3	12.9981	77.551	12.9984	77.5518	19th cross road	40	South	3
4	12.9998	77.5521	13.0002	77.5522	14th cross road	50	North	4
5	13.0021	77.5527	13.0028	77.5531	Rajajinagar main road	60	North	5
6	13.003	77.5535	13.0039	77.5542	Rajajinagar 1st block road	60	South	6
7	13.0041	77.5546	13.0041	77.5546	4th main road	50	North	7
8	13.0044	77.5548	13.0073	77.555	19th G main road	60	North	8
9	13.0077	77.557	13.0079	77.5573	2nd A cross road	40	South	9
10	13.0082	77.607	13.201	77.7112	17th cross road	40	South	10

4) Violation database which has violation id and fine to be charged for a particular violation:

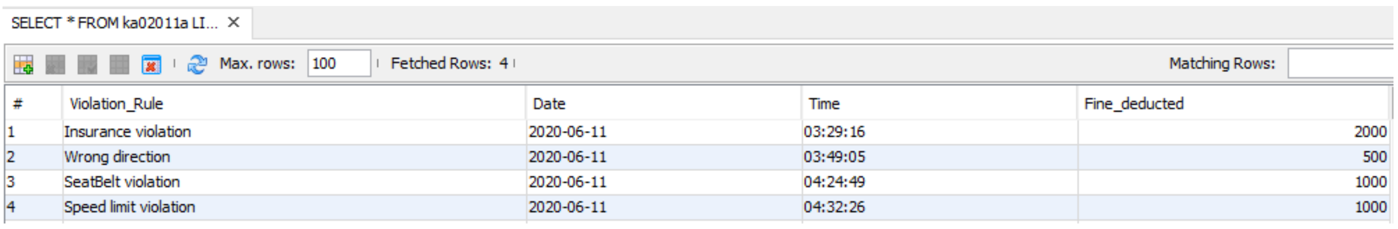


SELECT * FROM violation L... ✕

Max. rows: 100 | Fetched Rows: 4 | Matching Rows:

#	Violation_Rule	Violation_id	Fine
1	Insurance violation	KATV0012	2000
2	SeatBelt violation	KATV0026	1000
3	Speed limit violation	KATV0049	1000
4	Wrong direction	KATV0050	500

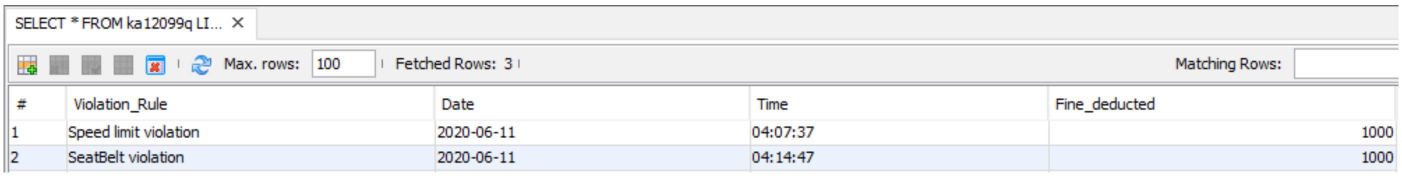
5) User tables named as agent id which were created if not present in the database whenever violation is committed and then updated accordingly based on the type of violation committed on some date and time:



SELECT * FROM ka02011a LI...

Max. rows: 100 Fetched Rows: 4 Matching Rows:

#	Violation_Rule	Date	Time	Fine_deducted
1	Insurance violation	2020-06-11	03:29:16	2000
2	Wrong direction	2020-06-11	03:49:05	500
3	SeatBelt violation	2020-06-11	04:24:49	1000
4	Speed limit violation	2020-06-11	04:32:26	1000

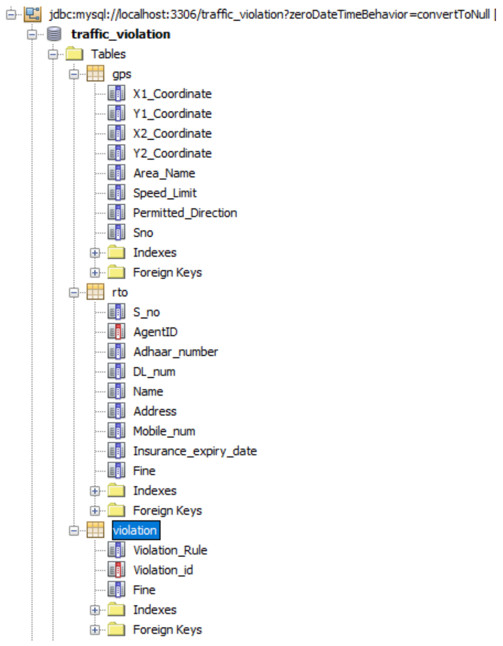


SELECT * FROM ka12099q LI...

Max. rows: 100 Fetched Rows: 3 Matching Rows:

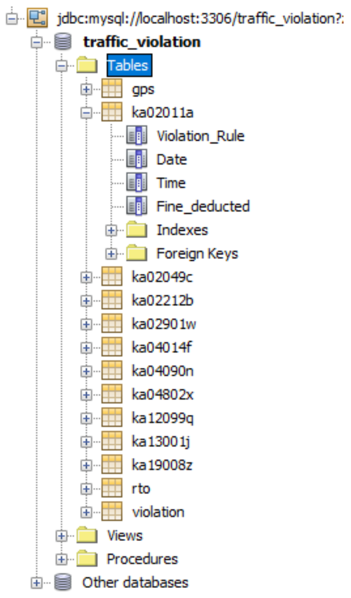
#	Violation_Rule	Date	Time	Fine_deducted
1	Speed limit violation	2020-06-11	04:07:37	1000
2	SeatBelt violation	2020-06-11	04:14:47	1000

6) List of databases before executing the program:



```
jdbc:mysql://localhost:3306/traffic_violation?zeroDateTimeBehavior=convertToNull |
  traffic_violation
    Tables
      gps
        X1_Coordinate
        Y1_Coordinate
        X2_Coordinate
        Y2_Coordinate
        Area_Name
        Speed_Limit
        Permitted_Direction
        Sno
        Indexes
        Foreign Keys
      rto
        S_no
        AgentID
        Adhaar_number
        DL_num
        Name
        Address
        Mobile_num
        Insurance_expiry_date
        Fine
        Indexes
        Foreign Keys
      violation
        Violation_Rule
        Violation_id
        Fine
        Indexes
        Foreign Keys
```

7) List of databases after executing the program for 1hour:



```
jdbc:mysql://localhost:3306/traffic_violation?:
  traffic_violation
    Tables
      gps
      ka02011a
        Violation_Rule
        Date
        Time
        Fine_deducted
        Indexes
        Foreign Keys
      ka02049c
      ka02212b
      ka02901w
      ka04014f
      ka04090n
      ka04802x
      ka12099q
      ka13001j
      ka19008z
      rto
      violation
    Views
    Procedures
  Other databases
```

Demonstration video for this project:

https://drive.google.com/file/d/1rOM75ixuErk2-6sHiLOzojnLpU7aFb6l/view?usp=sharing