OBJECT-ORIENTED ANALYSIS AND DESIGN - PROJECT

CRASH DETECTION OF VEHICLES AND ALERTS

Project Report

04.01.2022

TEAM NAME: A-Team

TEAM MEMBERS:

Keerthana A
2019103028

Mullapudi Renuka Devi
2019103037

Ramya S
2019103049

TABLE OF CONTENTS:

ABSTRACT

A large number of people die/are severely injured due to traffic accidents globally because the emergency services/contacts are not informed in time. The main aim and purpose of our project are to reduce the response time of emergency services in traffic accident cases and ensure more safety. This technology detects an accident by utilizing the sensors on a smartphone and sends immediate alerts along with the basic victim details to the nearest emergency service(ambulance service, nearby police stations) and pre-selected emergency contacts by providing the location using real-time tracking which helps in saving the time of emergency services and increase the chance of survival of a victim.

1.3 INTENDED AUDIENCE:

This mini-project is a prototype for detecting crash/accident and alerting system which is being done by us students under the guidance of our college professors. This project is basically useful for all kinds of people because whenever some kind of crash is detected it'll immediately notify the emergency services and our beloved family and friends and thus the series of actions take place on time.

- All the emergency services(police station, hospital, fire etc)

1.4 PROPOSED MINI CASE STUDY STATEMENT:

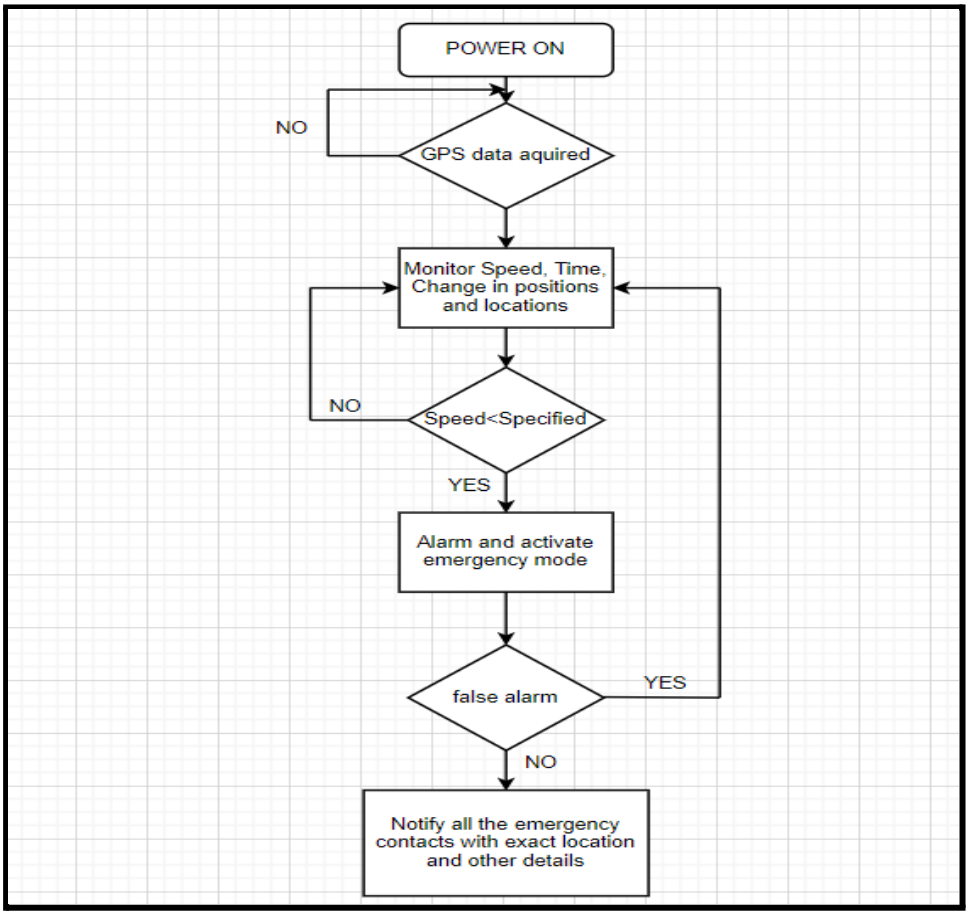
To design a crash detection and alerting system using IoT to increase the chance of survival of the victim during a crash and help emergency services reach in time to the destination in the shortest route possible.

1.5 BENEFITS OF THIS SYSTEM:

- Since the exact location of the victim is tracked, **emergency services** without any delay will be able to reach the destination of the victim in the shortest path possible using G-Maps tracking the victim's location.
- The survival chance of **the victim** increases.

PROCESS FLOW CHART

Detect Accidents and send Alerts to emergency services/contacts



NORMAL SCENARIO

- The user registers all the details and starts using the app
- When the user is driving the inbuilt accelerometer on the phone reads the acceleration values and monitors the position of the vehicle
- If there is a sudden change in the force and twist values it is considered to be an accident
- A timer of 30 seconds is activated to avoid false positives.
- If it is a false positive the user can abort within 30 seconds
- If not false positive the details of the victim and location are sent to nearest hospital, nearest Police Station which are found with help of Gmaps and are sent to the pre-selected emergency contacts

- As soon as the emergency services receive the accident details they reach the destination in the shortest route possible with the help of Gmaps to carry the rest of the procedures
- When an accident is detected a certain amount from the registered bank account is detected and if there is no minimum balance penalty will be applied
- Incase of any vehicle damage, insurance can be claimed from the application by uploading all the documents and verification will be done.

ACTORS:

- User/Victim
- Accelerometer
- User's phone
- G-maps
- Hospital
- Police dept
- Pre-selected emergency contact
- Insurance Company
- Bank

ACTORS AND THEIR GOALS:

ACTORS	GOALS
User	Registers all the details and Uses the application for the safety purpose
User phone	When an accident is detected all the functions are automatically implemented with its help
Accelerometer	Helps in calculating force and twist values and detecting the accident
GPS and Gmaps	Locates and navigates short routes
Hospital	Emergency responder which comes to the rescue of victim after getting notified about the accident(Medical)
Police dept	Emergency responder which comes to

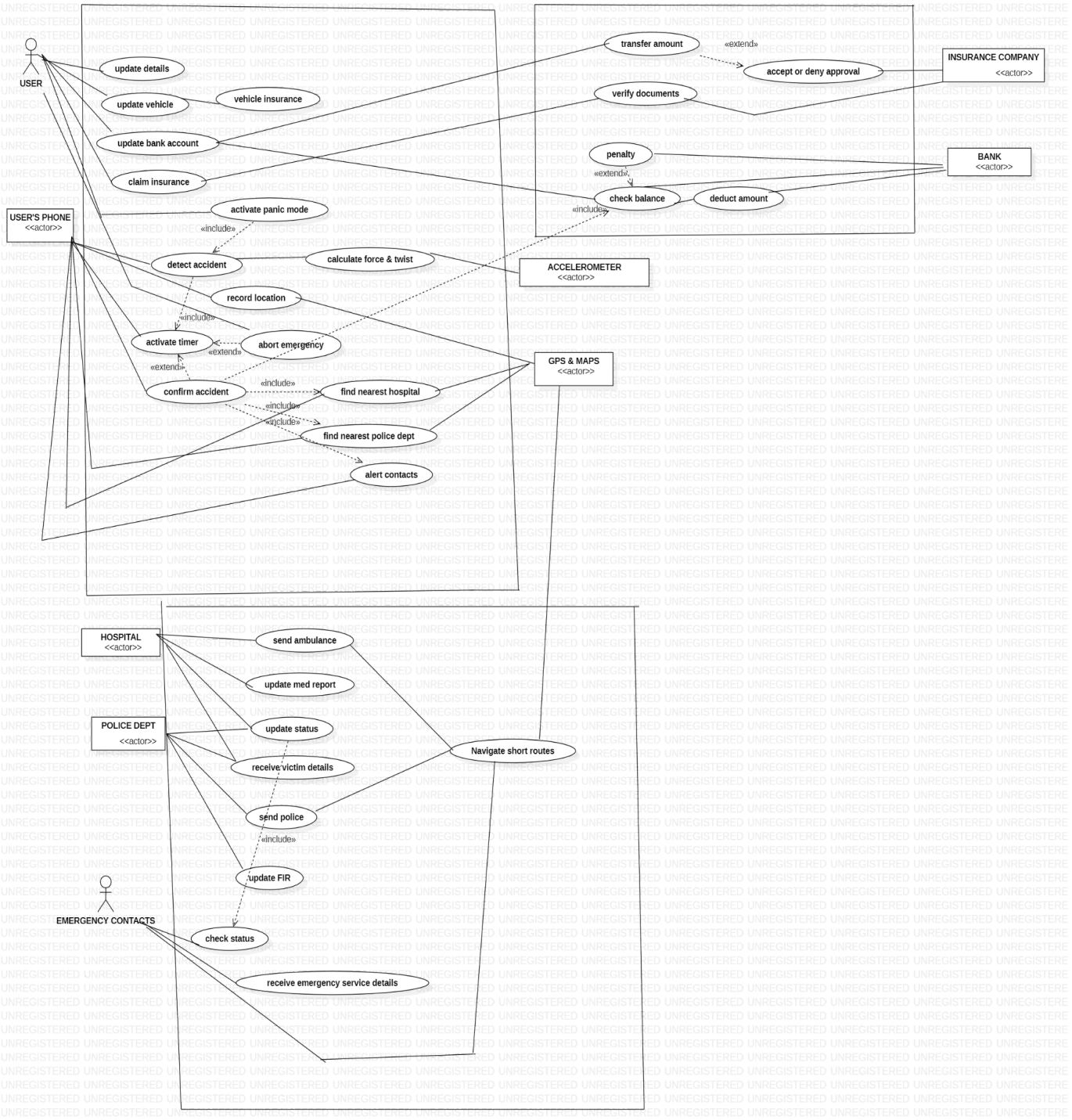
	the rescue of victim after getting notified about the accident.(Legal)
Bank	Transactions from the registered account takes place with the help of bank
Insurance Company	Verifies the submitted document and denies/approves the insurance claim and transfers the amount through the bank

USECASES:

- Update details
- Update Vehicles
- Update Bank account
- Vehicle Insurance
- Activate Panic Mode
- Calculate force and twist
- Detect accident
- Activate timer
- Record location
- Abort emergency
- Confirm accident
- Find nearest hospital
- Find nearest police dept
- Alert contacts
- Send Ambulance
- Update med report
- Update status
- Receive victim's details
- Send police
- Update FIR
- Check status
- Receive Emergency service details
- Navigate Short routes
- Claim insurance
- Transfer amount
- Accept/Deny Approval
- Check balance
- Deduct amount
- Penalty

- Verify Documents

USE CASE DIAGRAM:



USE CASE DESCRIPTIONS

Update Details

SCOPE: To get the basic details and little medical information about the user/victim

ACTOR: User

STAKEHOLDERS AND INTERESTS:

THE USER/VICTIM: Fills/Updates required details

PRE-CONDITION: Phone must be turned on and must have internet service

POST-CONDITION: The details of user are registered

BASIC FLOW/SUCCESS SCENERIO: The user enters all details required about himself.

FREQUENCY OF OCCURANCE: Occurs initially while registering in the app and whenever user want to add/delete/edit details

Update Vehicle

SCOPE: for easy identification of vehicle and insurance purposes

ACTOR: User

STAKEHOLDERS AND INTERESTS:

THE USER/VICTIM: Fills/Updates required vehicle details

PRE-CONDITION: Phone must be turned on and must have internet service

POST-CONDITION: The details of vehicles are registered

BASIC FLOW/SUCCESS SCENERIO: The user enters all details required about the vehicle.

FREQUENCY OF OCCURANCE: Occurs initially while registering in the app and whenever user want to add/delete/edit details

Update Bank Account

SCOPE: For deduction of amount from the account given if the accident detected is true

ACTOR: User

STAKEHOLDERS AND INTERESTS:

THE USER gives the bank account details

PRE-CONDITION: Phone must be turned on and must have internet service

POST-CONDITION: The details of bank account are registered

BASIC FLOW/SUCCESS SCENERIO: The user enters all details required about the bank account.

FREQUENCY OF OCCURANCE: Occurs initially while registering in the app and whenever user want to add/delete/edit details

Calculate Force

SCOPE: To continuously monitor the G-forces to detect when an accident occurs

ACTOR: Accelerometer

PRE-CONDITION: Phone must be turned on and must have internet service

POST-CONDITION: The force will be being on calculated

BASIC FLOW/SUCCESS SCENERIO: Keeps track on the force and twist values

FREQUENCY OF OCCURANCE: Occurs everytime we are travelling by a vehicle

Record Location

SCOPE: To identify and get the location of the place where the accident took place

ACTOR: GPS & G-Maps

STAKEHOLDERS:

THE EMERGENCY SERVICES/CONTACTS will get to know the location *OF VICTIM* and thus ll be able to help

PRE-CONDITION: Phone must be turned on and must have internet service and accident must be detected

POST-CONDITION: This location will be sent to emergency contacts and services

BASIC FLOW/SUCCESS SCENERIO: The location of accident is recorded.

FREQUENCY OF OCCURANCE: Occurs whenever accident is detected.

Alert Contacts

SCOPE: To send alerts to pre-selected emergency contacts about the accident

ACTOR: VICTIM'S PHONE

STAKEHOLDERS:

THE PRE-SELECTED CONTACTS will get to know the accident that happened *TO VICTIM*
PRE-CONDITION: Accident must be detected.

POST-CONDITION: All the necessary details and location will be sent to the contacts

BASIC FLOW/SUCCESS SCENERIO: The contacts are alerted successfully

FREQUENCY OF OCCURANCE: Occurs whenever accident is detected.

Alert Nearest Hospital

SCOPE: To send alerts to the nearest hospital from the location of the accident

ACTOR: GPS & G-Maps

STAKEHOLDERS:

THE HOSPITAL will get to know the accident that happened *TO VICTIM* and thus would take appropriate steps

PRE-CONDITION: Accident must be detected.

POST-CONDITION: All the necessary details and location will be sent to the nearest hospital services so that they could respond quickly

BASIC FLOW/SUCCESS SCENERIO: The hospitals are alerted successfully

FREQUENCY OF OCCURANCE: Occurs whenever accident is detected.

Alert Nearest PoliceDept.

SCOPE: To send alerts to the nearest police dept from the location of the accident

ACTOR: GPS & G-Maps

STAKEHOLDERS:

THE POLICE DEPT will get to know the accident that happened *TO VICTIM* and thus would take appropriate steps

PRE-CONDITION: Accident must be detected.

POST-CONDITION: All the necessary details and location will be sent to the nearest police dept so that they could respond quickly

BASIC FLOW/SUCCESS SCENERIO: The policedept. are alerted successfully

FREQUENCY OF OCCURANCE: Occurs whenever accident is detected.

Activate Timer

SCOPE: To avoid false positives

ACTOR: Victim (phone)

STAKEHOLDERS:

THE VICTIM can be sure of using the app since false positives can be avoided

PRE-CONDITION: Detection of an accident

POST-CONDITION: After the basic time period (60 seconds) is passed the alerts will be sent if not called off

BASIC FLOW/SUCCESS SCENERIO: The emergency services and contacts are alerted successfully

FREQUENCY OF OCCURANCE: Occurs whenever accident is detected.

Abort Emergency

SCOPE: To avoid false positives

ACTOR: Victim

STAKEHOLDERS:

THE VICTIM can avoid false positives and thus the emergency services need not panic unnecessary

PRE-CONDITION: False Detection of an accident

POST-CONDITION: The false detection will be aborted manually.

BASIC FLOW/SUCCESS SCENERIO: False alarms are successfully turned off by user

FREQUENCY OF OCCURANCE: Occurs whenever false accident is detected.

Activate Panic Mode

SCOPE: In case of failure of detection and the person is in the condition to call emergency

ACTOR: Victim

STAKEHOLDERS:

THE VICTIM can activate panic mode in case of failure of detection and the person is in the condition to call emergency

PRE-CONDITION: Failure of Detection of an accident

POST-CONDITION: This activation of panic mode could help reach the emergency services like hospital and police dept.

BASIC FLOW/SUCCESS SCENERIO: This mode alerts emergency services successfully

FREQUENCY OF OCCURANCE: Occurs whenever accident is un-detected.

Send Ambulance

SCOPE: To send an ambulance to the accident location

ACTOR: Hospital

STAKEHOLDERS:

THE HOSPITAL will send *TO VICTIM* in the process of rescue

PRE-CONDITION: Alert must be sent to the hospital.

POST-CONDITION: take the victim to the hospital and keep on updating the status

BASIC FLOW/SUCCESS SCENERIO: Ambulance is sent to accident location

FREQUENCY OF OCCURANCE: Occurs whenever hospital is notified about the accident

Send Policemen

SCOPE: To send policemen for investigation and enquiry to the accident location

ACTOR: Police Dept

STAKEHOLDERS:

THE POLICEDEPT will send policemen *TO VICTIM* in the process of rescue and investigation

PRE-CONDITION: Alert must be sent to police dept.

POST-CONDITION: investigate and keep on updating status.

BASIC FLOW/SUCCESS SCENERIO: Policeman are sent to accident location

FREQUENCY OF OCCURANCE: Occurs whenever Policedept. is notified about the accident

Update Status

SCOPE: To update the status of the victim (like reached location, on the way to the hospital, in-hospital etc)

ACTOR:Police Dept, Hospital

STAKEHOLDERS:

THE POLICEDEPT and HOSPITAL update status so that the *emergency contact responders* can view it and react accordingly

PRE-CONDITION: Accident alert must be received and procedures should be started

POST-CONDITION: will keep on continuously updating the condition of the victim

BASIC FLOW/SUCCESS SCENERIO: Police aresent to accident location

FREQUENCY OF OCCURANCE: Occurs whenever Police Deptl is notified about the accident

Emergency Service Details

SCOPE: To get the details of the emergency services(hospital, police dept) who responded

ACTOR: Emergency Contact

STAKEHOLDER:

THE EMERGENCY CONTACT RESPONDERS can know what emergency services responded

PRE-CONDITION: The emergency services must respond to the accident

POST-CONDITION: The contacts will know the details of emergency services

BASIC FLOW/SUCCESS SCENERIO: The contacts will know the details of emergency services to reach there

FREQUENCY OF OCCURANCE: Occurs when hospital, police dept respond to the accident

Receive Victim's Details

SCOPE: To get details and all basic info about the victim

ACTOR: Hospital, Police Dept, Emergency Contacts

STAKEHOLDER:

THE RESPONDERS get to know the details for further procedures

PRE-CONDITION: Accident alert must be received

POST-CONDITION: This information can be used for medical purposes and contacting anyone

BASIC FLOW/SUCCESS SCENERIO: The information of victim has been sent to responders successfully

FREQUENCY OF OCCURANCE: Occurs whenever an accident occurs

Check Status

SCOPE: To check the status of the victim being updated by police dept. and hospital

ACTOR: Emergency Contact

PRE-CONDITION: Accident alert must be received and the(emergency service) hospital and police dept. must update on a regular basis

POST-CONDITION: Based on the status the emergency contact will get to know how to react according to the situation

BASIC FLOW/SUCCESS SCENERIO: can check status

FREQUENCY OF OCCURANCE: whenever the hospital and police dept add/edit status

File FIR

SCOPE: To file FIR report

ACTOR: Police Dept.

PRE-CONDITION: Accident alert must be received and police dept must finish little investigation about the accident

POST-CONDITION: This can be useful for further investigation and insurance claim

BASIC FLOW/SUCCESS SCENERIO: FIR is filed and uploaded successfully

FREQUENCY OF OCCURANCE: whenever an accident occurs and policemen investigate

Medical Report

SCOPE: To diagnose, treat and write a med report

ACTOR: Hospital.

PRE-CONDITION: Accident alert must be received and police dept must finish little investigation about the accident

POST-CONDITION: This can be useful for further investigation and insurance claim and can be saved to medical history

BASIC FLOW/SUCCESS SCENERIO: Medical report is uploaded successfully

FREQUENCY OF OCCURANCE: whenever an accident occurs and victim is admitted to a hospital

Navigate Shortest Routes

SCOPE: To find the shortest routes possible to reach the victim

ACTOR: GPS & G-Maps

PRE-CONDITION: Accident must be detected and alerts must be set

POST-CONDITION: Can be reached the destination in shortest route as soon as possible

BASIC FLOW/SUCCESS SCENERIO: Shortest and Fastest routes are shown

FREQUENCY OF OCCURANCE: whenever hospital/police dept/contacts need to reach respective destinations

Detect Balance

SCOPE: To detect the balance if the accident detected is positive

ACTOR: Bank

PRE-CONDITION: Accident must be detected and must be true

POST-CONDITION: Amount is deducted from the bank account

BASIC FLOW/SUCCESS SCENERIO: Amount is successfully deducted

FREQUENCY OF OCCURANCE: Whenever accident detected is true

Verify Docs

SCOPE: To verify all the documents related to insurance

ACTOR: Insurance Company

PRE-CONDITION: Vehicle damage due to accident

POST-CONDITION: Can approve/deny insurance claim after verifying document

BASIC FLOW/SUCCESS SCENERIO: Documents verified successfully

FREQUENCY OF OCCURANCE: whenever user validates permission to verify

Accept/Deny Approval

SCOPE: insurance claim

ACTOR: Insurance Company

PRE-CONDITION: all sorts of document verification must be done

POST-CONDITION: either accepted or denied approval

BASIC FLOW/SUCCESS SCENERIO: After verifying all docs, decision is made

FREQUENCY OF OCCURANCE: whenever user validates permission to verify and verification is done

Transfer Amount

SCOPE: transfer insurance amount

ACTOR: Insurance Company

PRE-CONDITION: claim must be approved

POST-CONDITION: amount will be transferred to the bank account

BASIC FLOW/SUCCESS SCENERIO: The insurance amount is successfully transferred

FREQUENCY OF OCCURANCE: whenever insurance claim is approved

Deduct Balance

SCOPE: for deducting the required amount as soon as the accident takes place

ACTOR: Bank

STAKEHOLDERS AND INTERESTS:

THE BANK: deducts the amount

PRE-CONDITION: Accident must be detected

POST-CONDITION: The amount from the registered bank account is deducted

BASIC FLOW/SUCCESS SCENERIO: When an accient is confirmed the amount is deducted

FREQUENCY OF OCCURANCE: Occurs whenever accident is confirmed

VERIFY DOCS

SCOPE: document verification for insurance approval

ACTOR: Insurance Company

STAKEHOLDERS AND INTERESTS:

THE INSURANCE COMPANY: verifies all the documents for the approval

PRE-CONDITION: Accident must be detected and user must claim insurance

POST-CONDITION: The insurance amount is transfered to registered bank account

BASIC FLOW/SUCCESS SCENERIO: When an accient is occurred and user vehicle is damaged he can claim for insurance through the application.

FREQUENCY OF OCCURANCE: Occurs whenever user claims for insurance and submit docs

DOMAIN MODELLING:

DOMAIN CLASSES

- USER
- VICTIM INFORMATION
- VEHICLE
- ACCELEROMETER
- ACCIDENT
- ACCIDENT SPOT
- EMERGENCY SERVICES
- NEAREST HOSPITAL
- NEAREST POLICE DEPT
- LOCATION
- EMERGENCY CONTACT
- AMBULANCE
- POLICEMEN
- MEDICAL REPORT
- FIR
- STATUS
- BANK ACCOUNT
- BANK
- TRANSACTION
- INSURANCE POLICY
- INSURANCE COMPANY
- DOCUMENT VERIFICATION
- APPROVAL
- TIMER
- EMERGENCY SERVICES

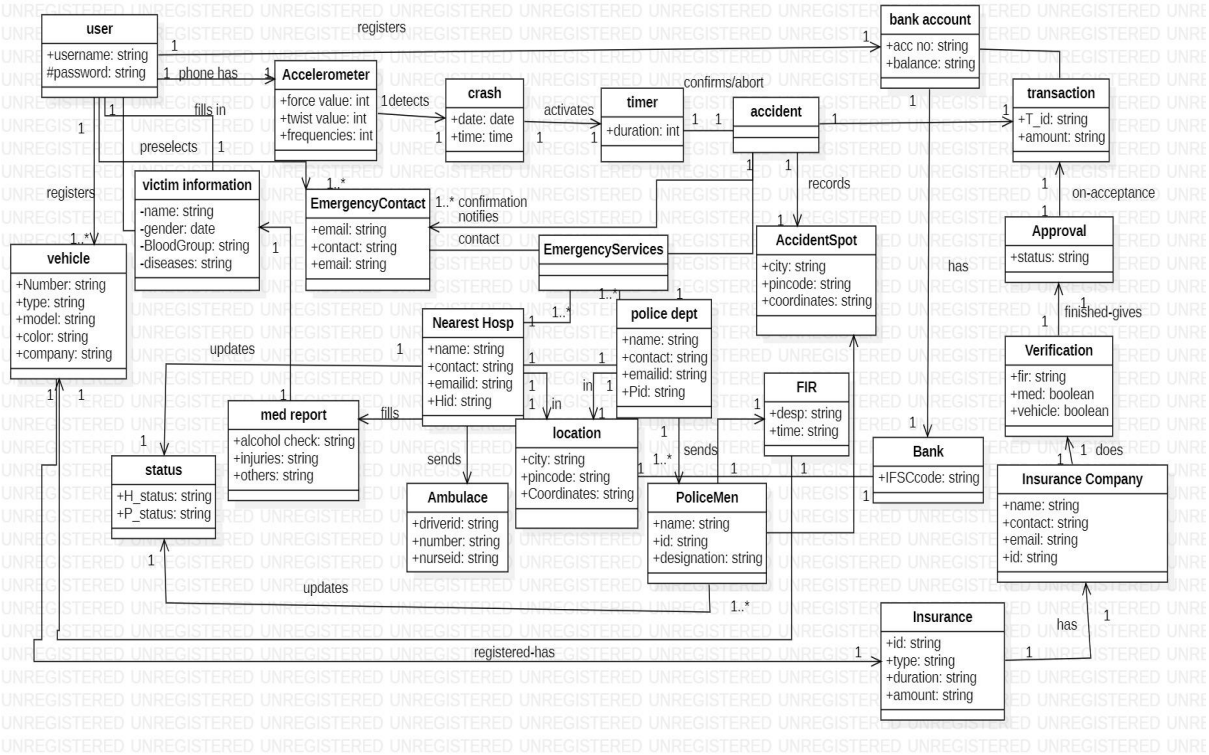
DATA DICTIONARY

CLASS	ATTRIBUTES	DESCRIPTION
USER	• Username • Password	The user is the person who uses this application. They can log in and fill required info and preselect emergency contacts for safety purposes
VICTIM INFORMATION	• Name • DOB • Gender • Blood Group • Diseases	This section consists of the basic required information of the user like name, age, blood group etc that can be helpful in case of an accident
VEHICLE	• Number • Type • Colour • Model • Company	This section consists of the vehicle details of the user.
ACCELEROMETER	• Force value • Twist value	It is inbuilt in the user's phone which calculates and monitors force and twist angles values and helps in accident detection in case of abnormal force/twist values.
ACCIDENT	• Date • Time	It is detected by the accelerometer and the time and date of the crash are stored
ACCIDENT SPOT	• City • Pincode • Coordinates	Location details of the place where the accident took place will be recorded to notify emergency responders
EMERGENCY SERVICES		It is an abstract class that gets notified in case of an emergency which contains subclasses hospital and police dept
NEAREST HOSPITAL	• Name • Id • Contact • email	It is a subclass of emergency services. As soon as the information s received ambulance is sent for the rescue of the victim to the accident spot
NEAREST POLICE DEPT	• Name • Id • Contact	It is a subclass of emergency services. As soon as the information s received appropriate procedures are followed and

	• Email	policemen are sent for the rescue of the victim and to investigate the accident spot
LOCATION	• City • Pincode • Coordinates	This class describes the location details of emergency services, the bank and the insurance company user is involved with
EMERGENCY CONTACTS	• Name • Contact • Email	The user preselects them. Incase of an emergency they get notified about the details and they can continuously monitor the status of the victim in the system and can contact the emergency services(vice versa)
AMBULANCE	• Number • Driver ID • NurseID	An ambulance is sent to the rescue spot in the shortest route possible with the help of Maps.
MED REPORT	• Alcohol check • Injuries • Others	The medical report with required details will be updated by the hospital for the emergency contacts, policemen to view and for future references
POLICEMEN	• Name • Id • Designation	This class consists of the details of policemen who investigated the case.
FIR	• Description • Date • Time	It consists of the details of the accident and investigation that happened for further references and purposes in the system
STATUS	• H_status • P_status	Hospital can update the status of victim. Policemen can view the status.
BANK ACCOUNT	• Account number • balance	The user registers a bank account and this class represents the details
BANK	• IFSC Code	The details of the bank which user has an account in
TRANSACTION	• T_id • Amount	It consists of transaction details Credits/Debits money based on the situation and each time balance is updated
INSURANCE POLICY	• ID • Type	Insurance policy can be claimed by the victim if the vehicle is damaged in the

	• Amount • Valid duration	crash without physically interacting/doing the procedures
INSURANCE COMPANY	• Name • Id • Contact • Email	Insurance company verifies all the documents present in the system and approves/denies the claim
DOCUMENT VERIFICATION	FIR Med Vehicle	Document verification is done by Insurance company to approve/deny the claim of insurance by the victim.
APPROVAL	• Status	Once all the documents of the victim are verified perfectly by the insurance company, approval can be given for the insurance policy claimed by the victim, or else it'll be denied.
TIMER	• Duration	Timer is introduced to avoid false positives. In case of detection of an accident, a timer is activated of duration 30 seconds. If it is false it can be aborted within the duration, after the duration the accident is confirmed

DOMAIN MODEL



NOTATIONS

RELATIONSHIPS AND MULTIPLICITY:

ASSOCIATIONS

A user fills in one set of victim information (one to one)
A user registers many vehicles (one to many)
A user preselects one or many emergency contacts (one to many)
A user phone has one accelerometer (one to one)
A crash activates one timer at a time (one to one)
A timer confirms an accident (one to one)
An accelerometer detects one crash at a particular time(one to one)
One transaction in one bank account at the particular time(one to one)
One approval on acceptance runs one transaction at a particular time(one to one)

One document verification finishes give one approval at a particular time(one to one)
One crash confirmation notifies one or many emergency contacts(one to one)
One Accident confirmation notifies many emergency services(one to one)
One user registers one bank account(one to one)
A policeman files an FIR(one to one)
The nearest hospital updates Med Report(one to one)
The hospital updates status(one to one)
The police dept updates status(one to one)
The status is viewed by emergency contact(one to many)
Bank has a particular location(one to one)
Nearest police dept in location(one to one)
Nearest hospital in location(one to one)
Many policemen reach the accident spot(one or many to one)
Ambulance reaches the accident spot(one to one)
Many emergency services can contact many emergency contacts or vice versa(many to many)
An accident initiates a transaction (one to one)
A transaction from/to bank account (one to one)
A vehicle has an insurance policy (one to one)
An insurance company does doc verification (one to one)
An approval on confirmation initiates the transaction (one to one)
A hospital includes location (one to one)
A police dept includes location (one to one)
A bank includes location (one to one)
An insurance company includes location (one to one)
Accident records accident spot (one to one)
Med report viewed by many policemen(one to one or many)
Nearest hospital contacts nearest police dept or vice versa (one to one)

DEPENDENCIES

Emergency services that respond are dependent on the location of the accident spot.

GENERALIZATIONS

Emergency Services has 2 derived class

- Nearest hospital
- Nearest Police Dept

AGGREGATIONS

FIR has vehicle details
FIR has victim information
FIR has an accident spot

Med Report has victim information
Bank Account has one bank

COMPOSITIONS

Insurance Policy is part of Insurance Company
Policemen are part of the police dept
Nearest hospital sends an ambulance to the accident spot

ABSTRACT CLASS:

We have Emergency services as the abstract class. It has the method rescue. The emergency services class has the method rescue and is connected to the nearest hospital and nearest police dept which has the same method of rescue in addition to their methods.

CLASS DIAGRAM

This class diagram is drawn to provide a conceptual perspective for our domain/system. We pick domain objects from the domain and model them into a class diagram that shows associations between these various classes.

CRC - (Class, Responsibility, Collaboration)

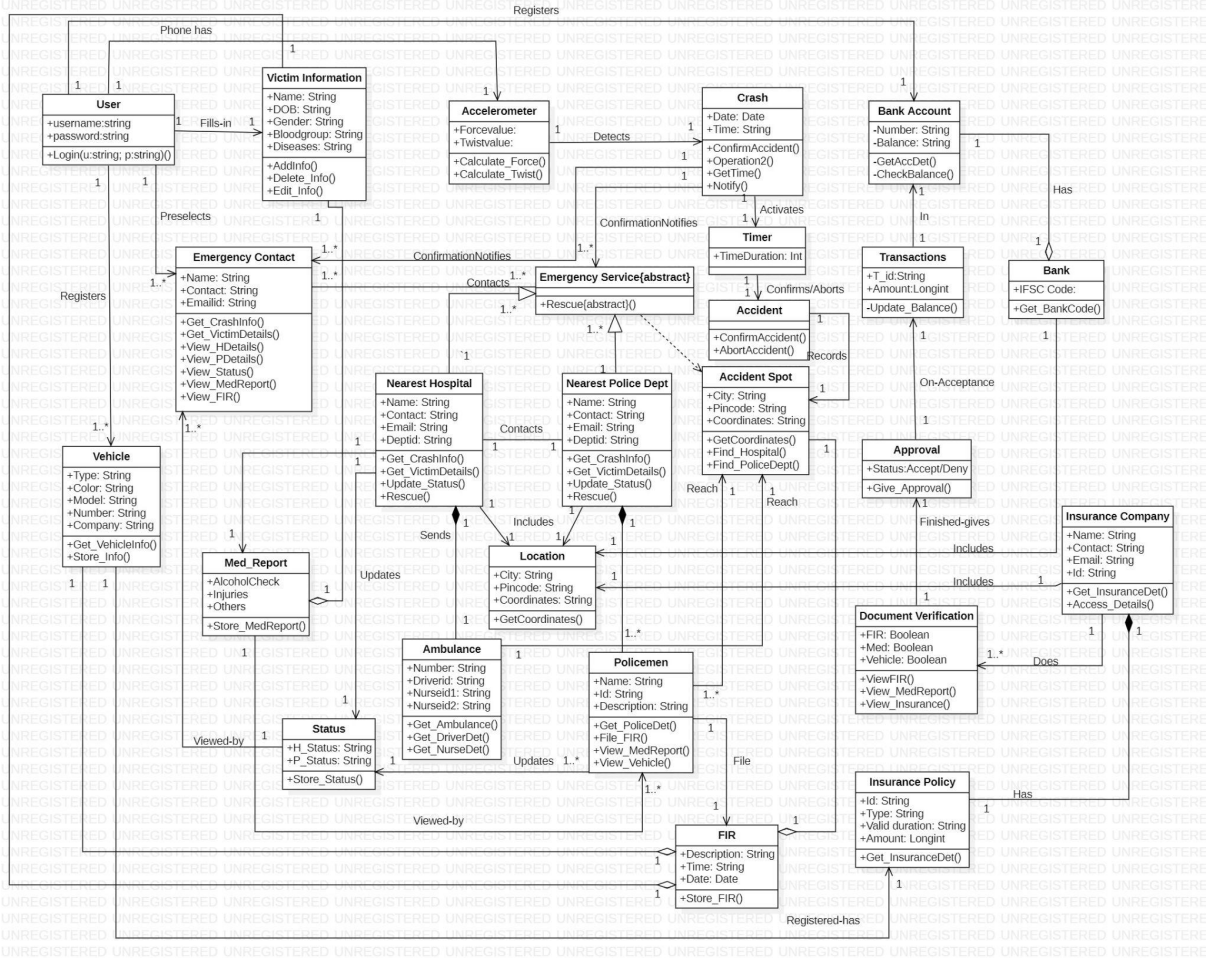
CLASS	RESPONSIBILITY	COLLABORATIONS
USER	• Login() • Logout()	Vehicle Bank account Victim information Emergency contacts Accelerometer
VICTIM'S INFORMATION	• add_Info() • delete_Info() • edit_Info()	User Med Report Status
VEHICLE	• GetVehicleDet() • StoreVehicleInfo()	User Insurance Policy FIR

ACCELEROMETER	• CalculateForce() • CalculateTwist()	Crash User
ACCIDENT	• ConfirmAccident()	Timer Accident Spot
ACCIDENT SPOT	• GetCoordinates() • Find_hospital() • Find_PoliceDept()	Accident Emergency services PoliceMen Ambulance FIR
EMERGENCY SERVICES	Rescue()	Crash Accident spot Nearest hospital Nearest police dept
NEAREST POLICE DEPT	• Get_CrashInfo() • Get_VictimInfo() • Update_P_status() • Rescue()	Location Policemen Emergency services Nearest hospital
NEAREST HOSPITAL	• Get_CrashInfo() • Get_VictimInfo() • Update_H_Status() • Rescue()	Location Med Report Ambulance Status Emergency Services Nearest Police dept
EMERGENCY CONTACTS	• Get_CrashInfo() • ViewHospitalDet() • ViewPoliceDeptDet() • ViewStatus() • ViEw_MedReport() • View_FIR() • GetVictimDetails()	User Crash Emergency Services Status
LOCATION	• GetCoordinates()	Nearest Hospital Nearest Police Dept Bank Insurance Company
AMBULANCE	• Get_AmbulanceDet() • Get_DriverDetails() • Get_NurseDetails()	Nearest Hospital Accident Spot

POLICEMEN	• RetreivePoliceDet() • File_FIR() • View_MedReport() • View_VehicleDet()	FIR Med Report Status Nearest police dept Accident spot
MED REPORT	• Store_MedReport()	Nearest hospital Policemen Victim information
FIR REPORT	• Store_FIR()	Policemen Accident spot Vehicle User
STATUS	• Store_Status()	Emergency contacts Nearest hospital Policemen
BANK ACCOUNT	• Get_AccDet() • CheckBalance()	User Transactions Bank
TRANSACTION	• Update_Balance()	Bank account Approval
BANK	• Get_BankCode()	Bank account Location
INSURANCE POLICY	• Get_InsuranceID()	Insurance company Vehicle
INSURANCE COMPANY	• GetInsuranceDet()	Insurance Policy Document Verification Location
DOC VERIFICATION	• View_InsurancePolicy() • View_FIR()	Insurance company Approval
APPROVAL	• GiveApproval()	Document Verification Transaction
TIMER	-	Crash Accident
EMERGENCY SERVICES	Rescue()	Crash Accident Spot

		Nearest Police dept Nearest Hospital Emergency Contacts

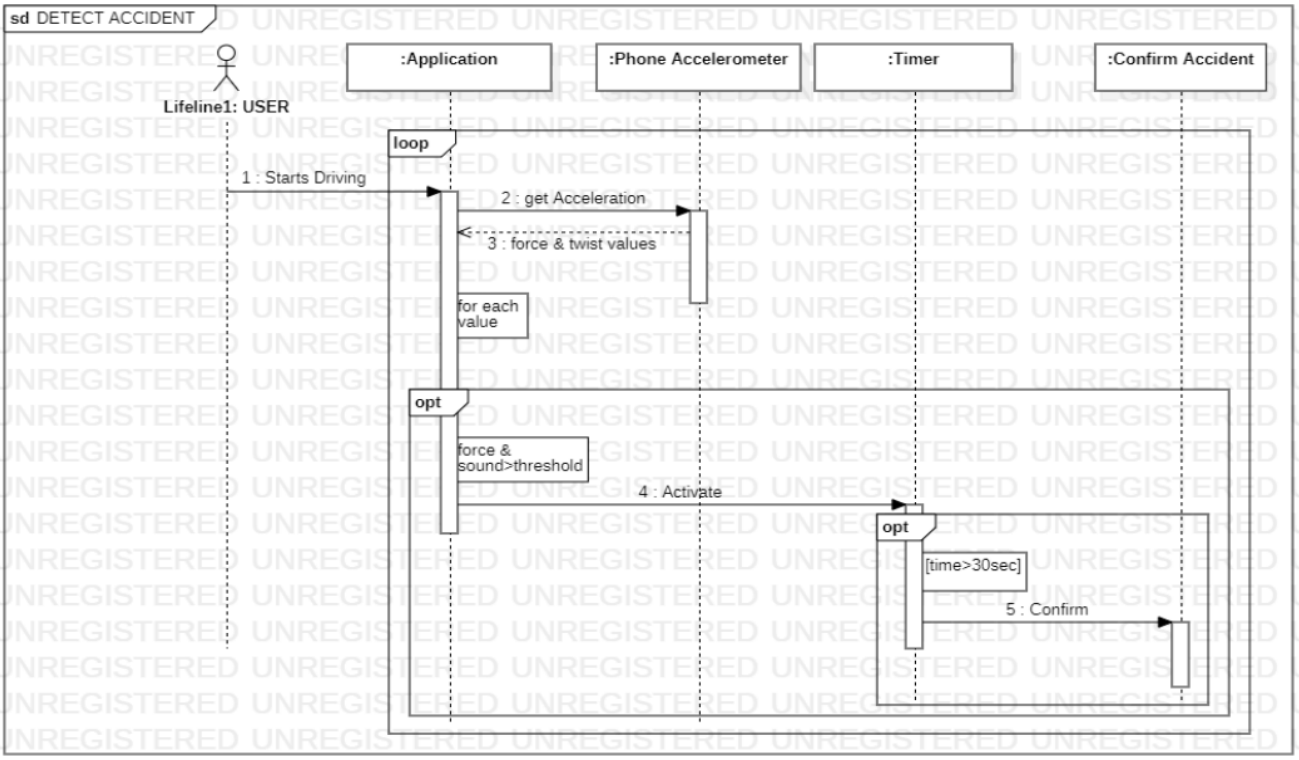
CLASS MODEL:



SEQUENCE DIAGRAMS:

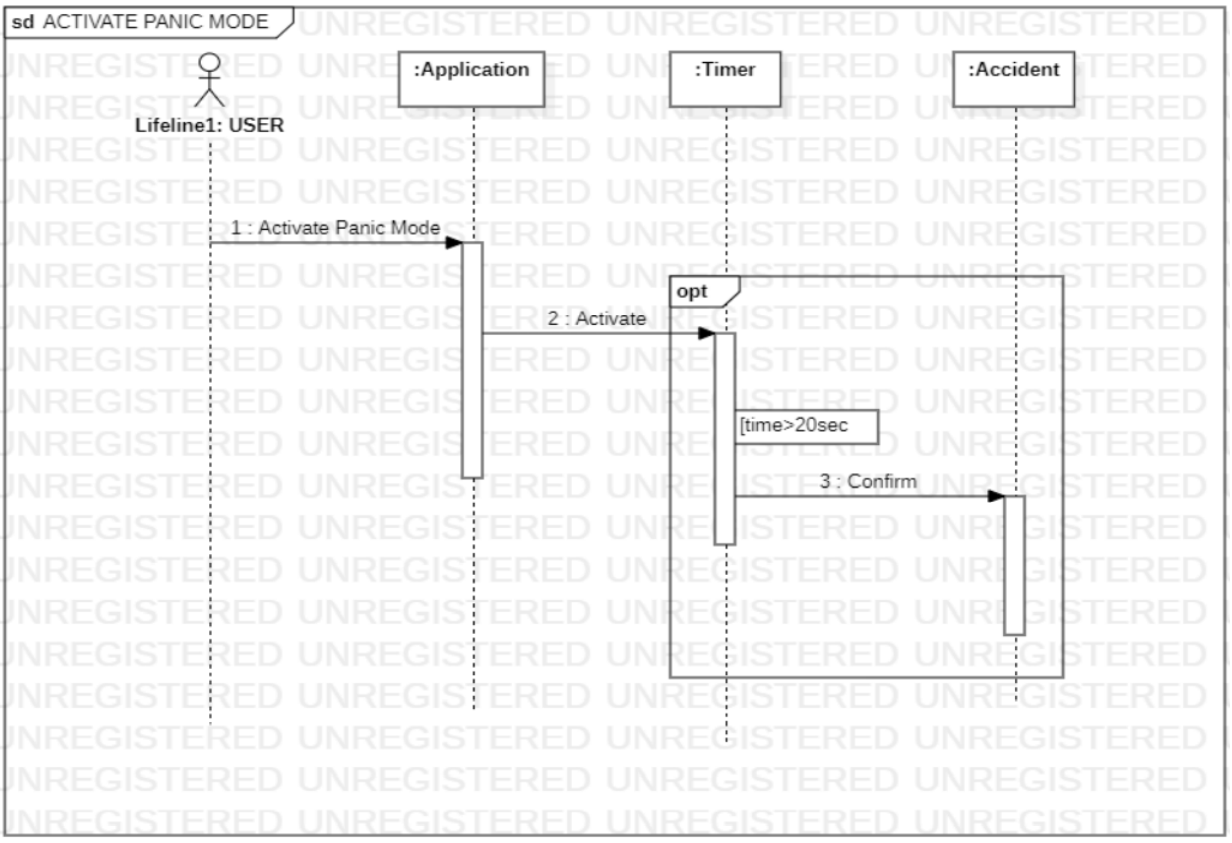
SEQUENCE DIAGRAMS: The sequence diagram shows a scenario in a sequence of events that occurs during one particular execution of the system. It captures the dynamic behaviour of the system.

DETECT ACCIDENT:



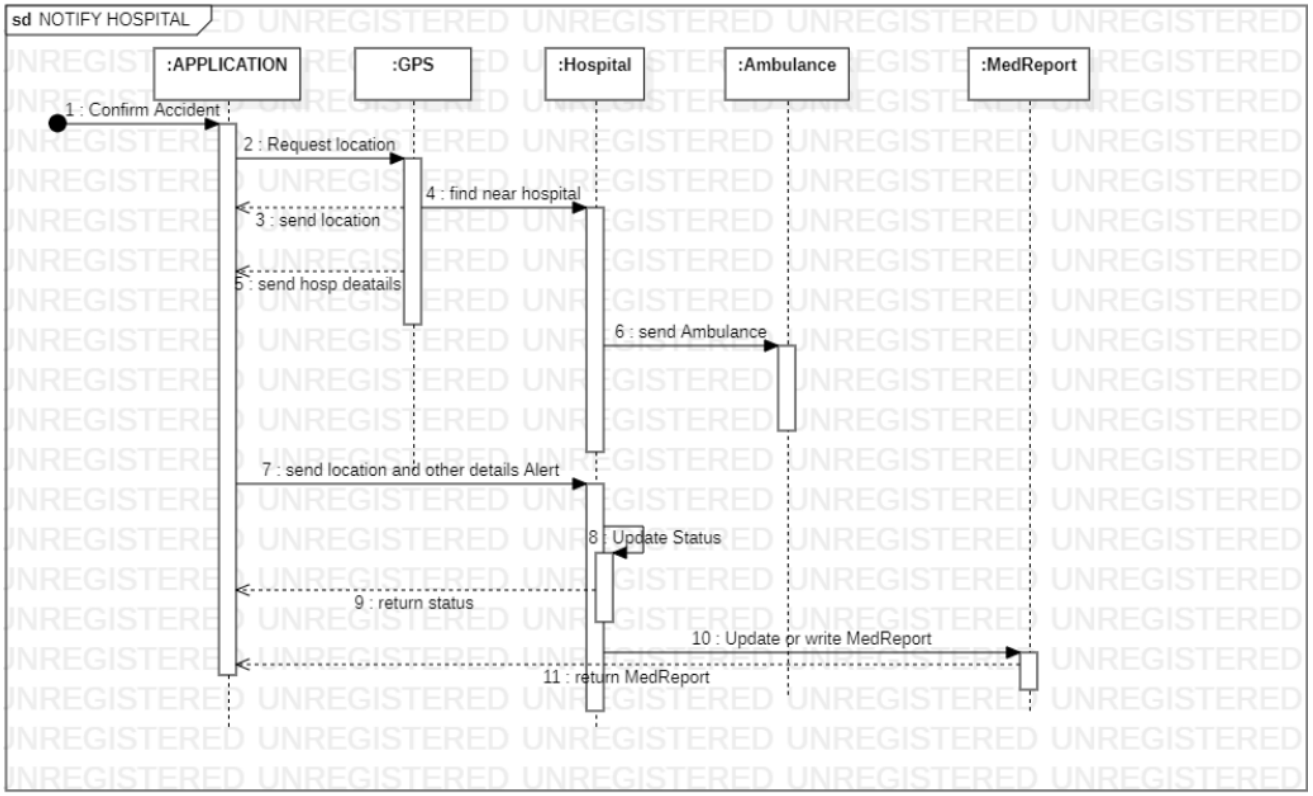
SEQUENCE OF EVENTS: User starts driving and the inbuilt accelerometer on the phone gets the acceleration values and returns the force and twist values. For each value if it is greater than the specified value a timer is activated. If the timer finishes its duration of 30 seconds it is confirmed as an accident

ACTIVATE PANIC MODE:



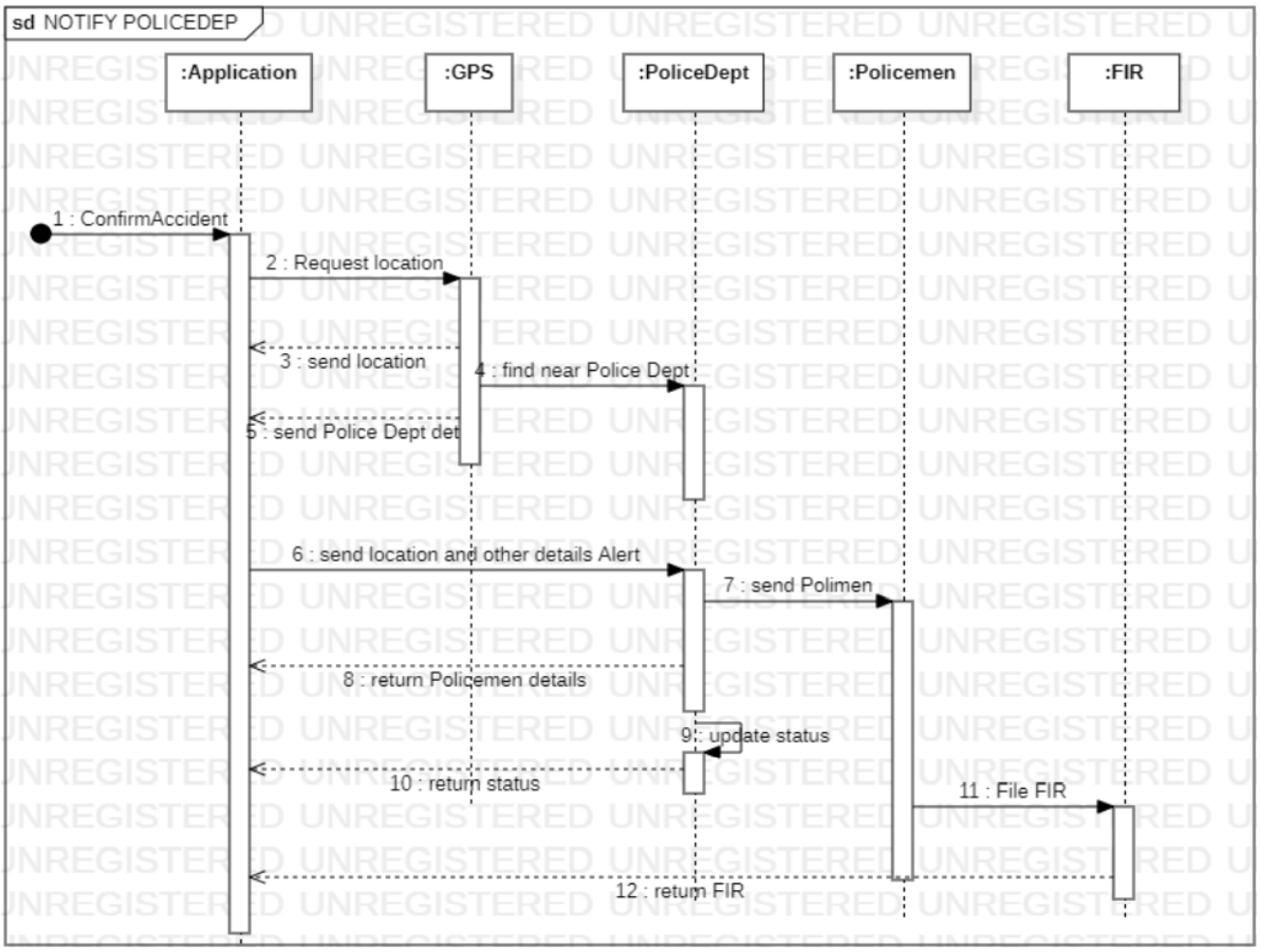
SEQUENCE OF EVENTS: When there is a failure in detection of an accident user can activate the panic mode which triggers an accident in the system. a timer is activated. If the timer finishes its duration of 30 seconds it is confirmed as an accident.

NOTIFY HOSPITAL:



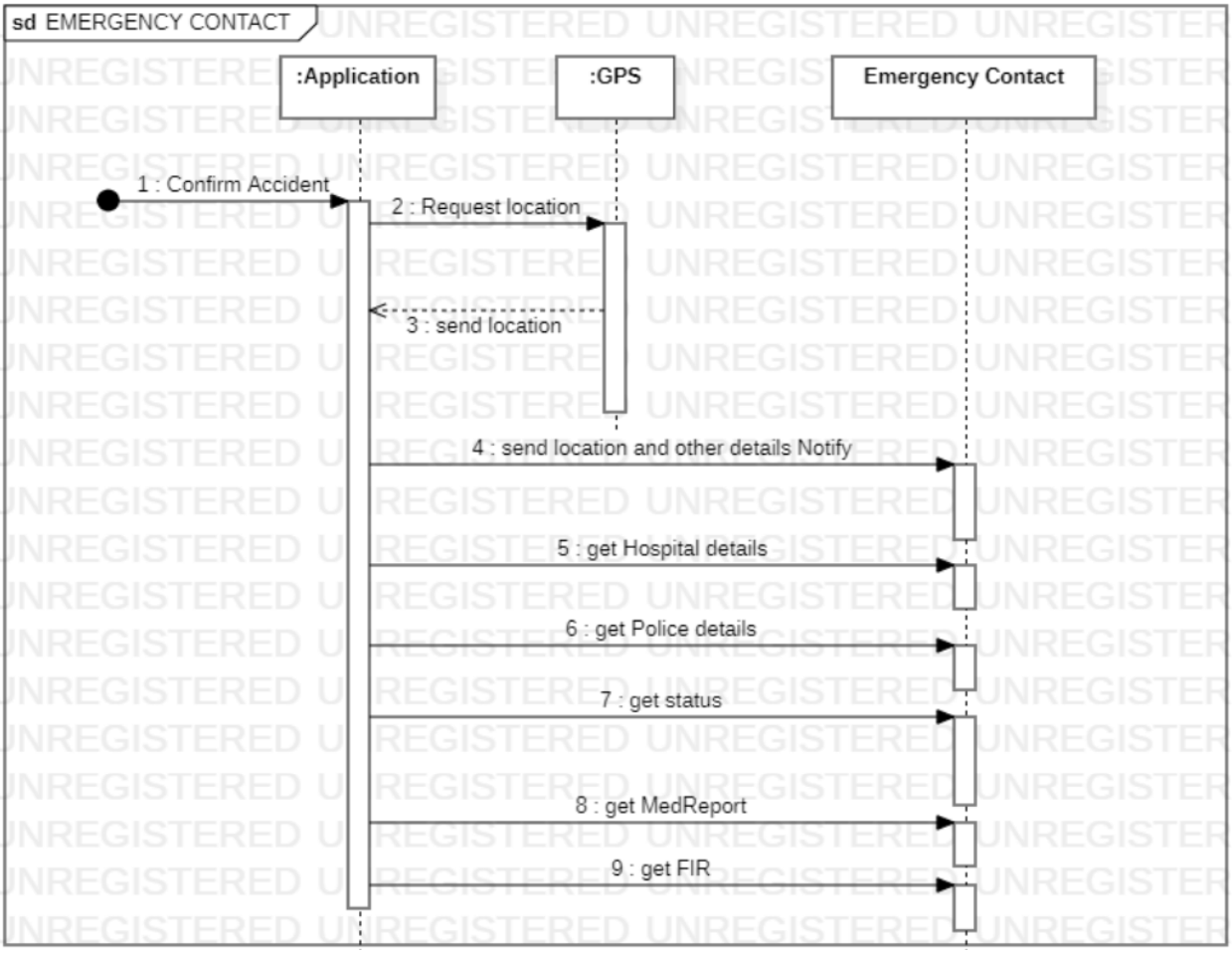
SEQUENCE OF EVENTS: After the accident is confirmed using GPS nearest hospital is found and notified about the accident location. And the hospital sends an ambulance to the location and updates its status continuously and medical reports are also uploaded to the system and all these returned to the application

NOTIFY POLICE DEPT:



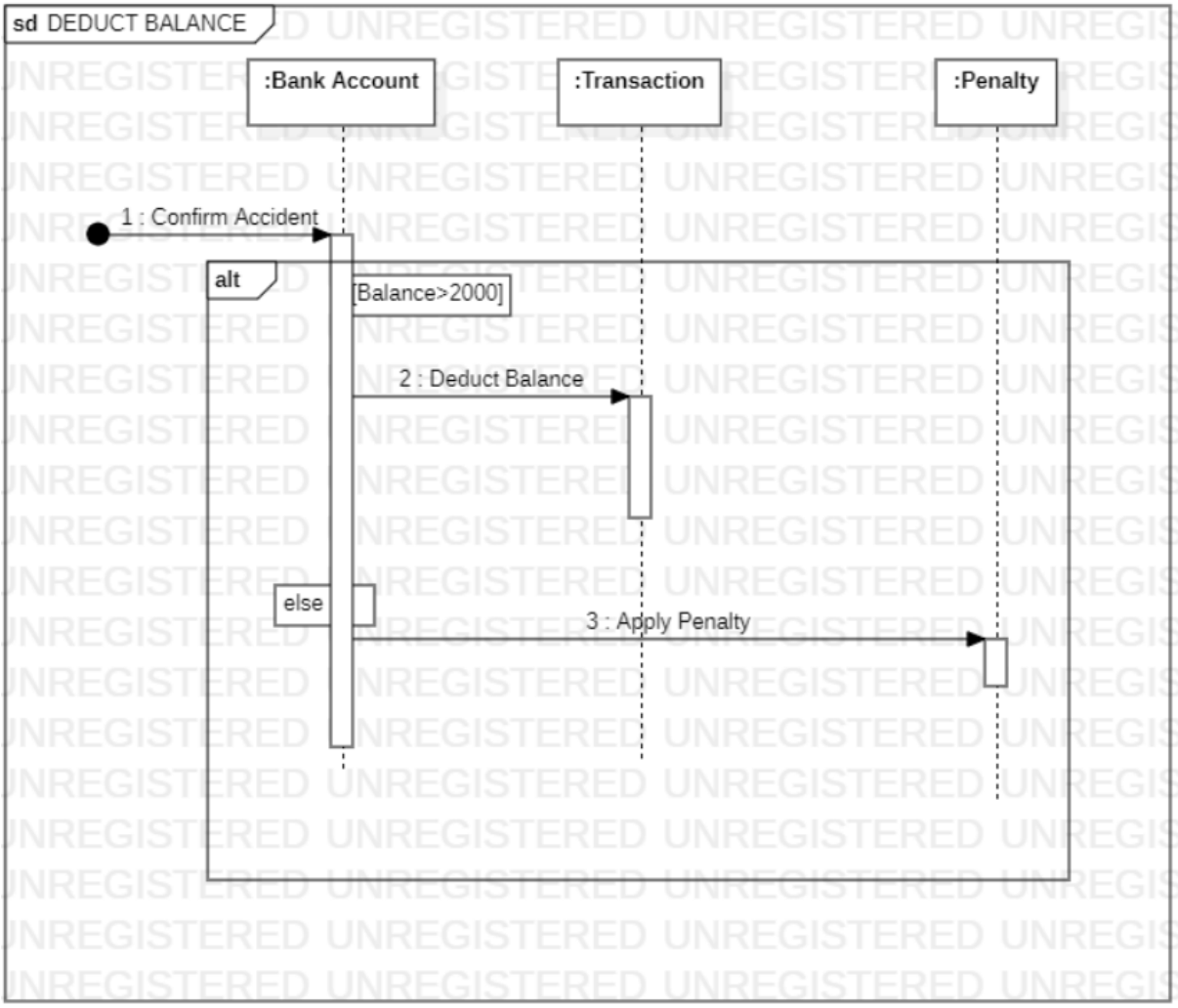
SEQUENCE OF EVENTS: After the accident is confirmed using GPS nearest POLICE DEPT is found and notified about the accident location. And the police dept sends policeman to the location for inquiry and investigation and updates its status continuously and FIR are also uploaded to the system and all these returned to the application

EMERGENCY CONTACTS:



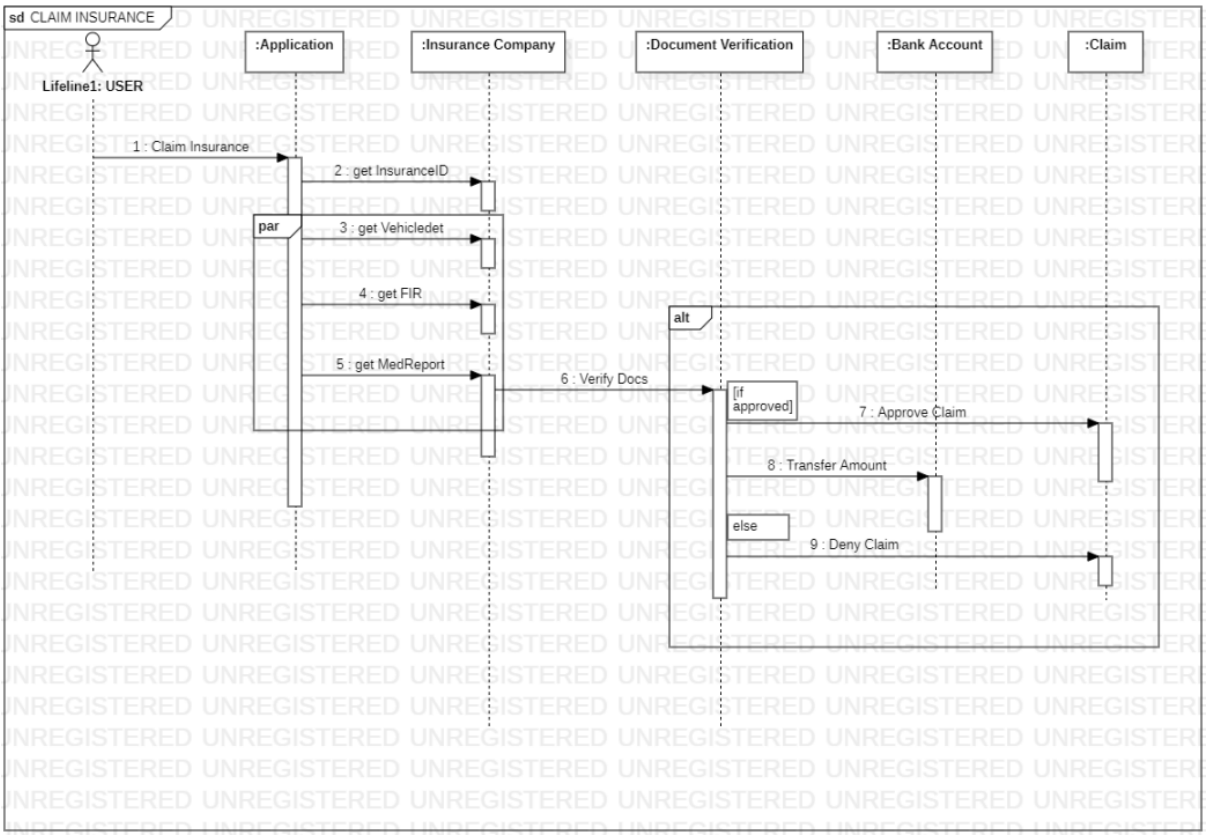
SEQUENCE OF EVENTS: After the accident is confirmed using GPS the location of the victim is recorded and is sent to the emergency contacts. The emergency contacts can also get the information of the emergency services that responded for the accident and their statuses and med report and FIR.

DEDUCT BALANCE:



SEQUENCE OF EVENTS: After the accident is confirmed an automatic transaction of certain amount is initiated and balance is deducted from the bank account registered. If the minimum balance is not available a penalty will be applied

CLAIM INSURANCE:



SEQUENCE OF EVENTS: After the accident is confirmed, Incase the user vehicle is damaged, he claims insurance and the insurance company does all the document verification and either approves/denies it.

STATE DIAGRAMS:

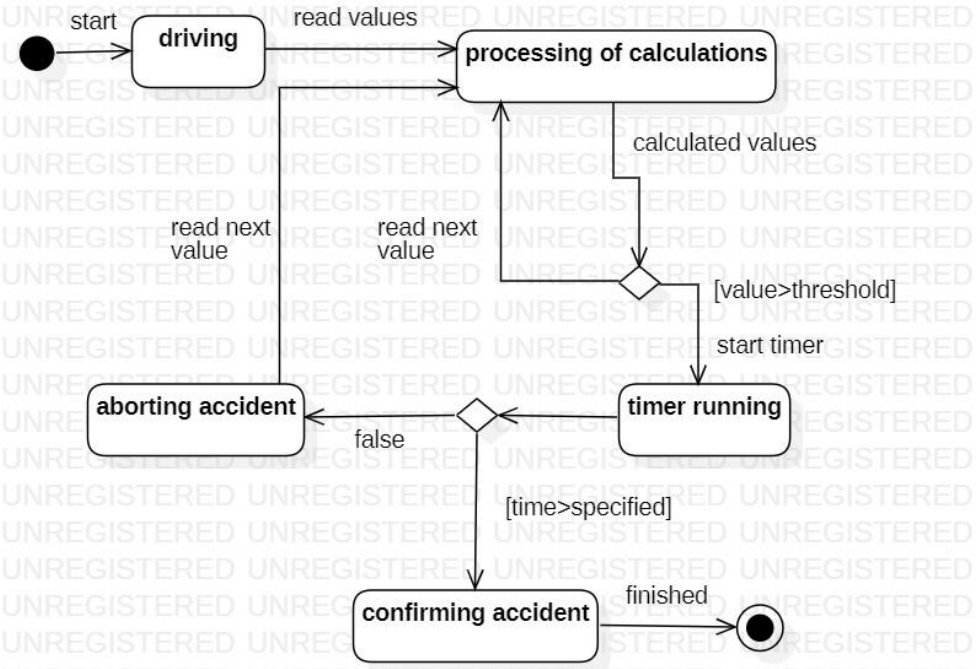
STATE DIAGRAM:

The state diagram describes the sequence of operations that occur in response to external stimuli as opposed to what they operate on/ what they do.

STATE DIAGRAM 1:

The states the objects undergoing to detect an accident

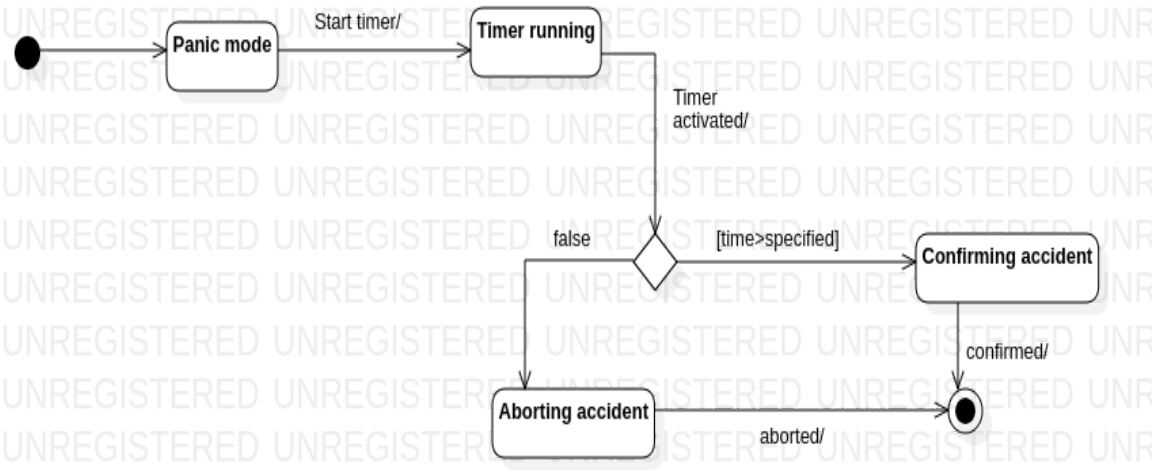
The user starts driving and the accelerometer processes all the calculations and for each value it is checked if it is greater than a specific value. If it is greater than a specific value the timer starts running. Accident can be aborted anytime when the timer is running. When it is timed out it is confirmed as an accident



STATE DIAGRAM 2:

The states objects undergoing after activating panic mode

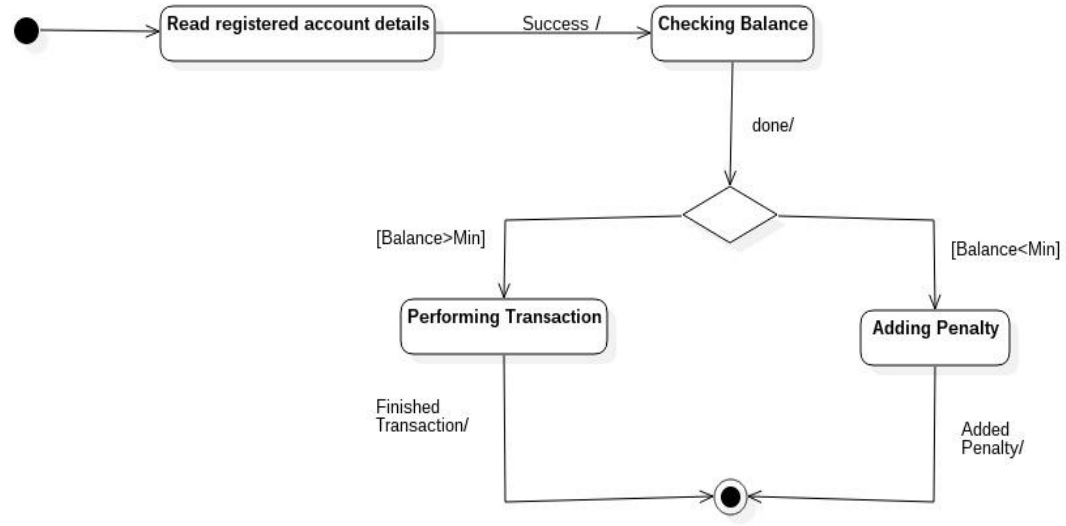
The user activates panic mode mode and the timer starts running. If it is accidental aborting can be done when the timer is running. When the timer duration is finished it will confirm the accident.



STATE DIAGRAM 3:

The states objects undergoing for a bank transaction(deduct amount)

After an accident is confirmed , the bank details are read successfully and the balance s checked. After done if balance > min a transaction will be formed else a penalty will be applied.



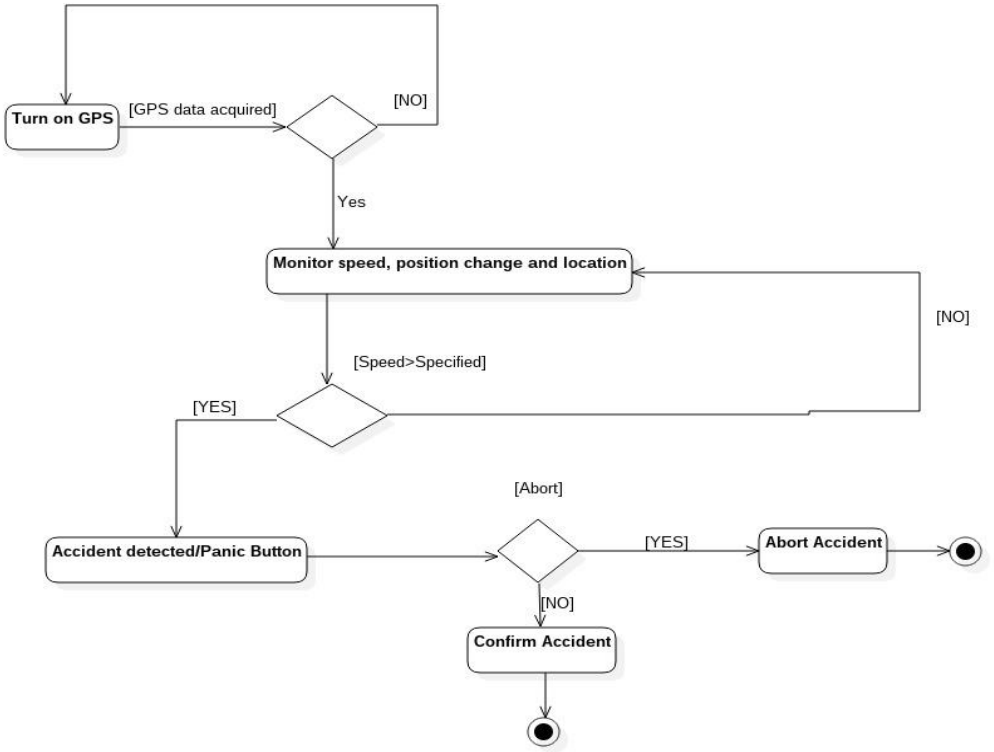
ACTIVITY DIAGRAMS:

ACTIVITY DIAGRAM 1:

(confirm accident)

- The user turns on the GPS. If the GPS is active it moves to the next activity
- The user speed, position change and location are monitored in loop

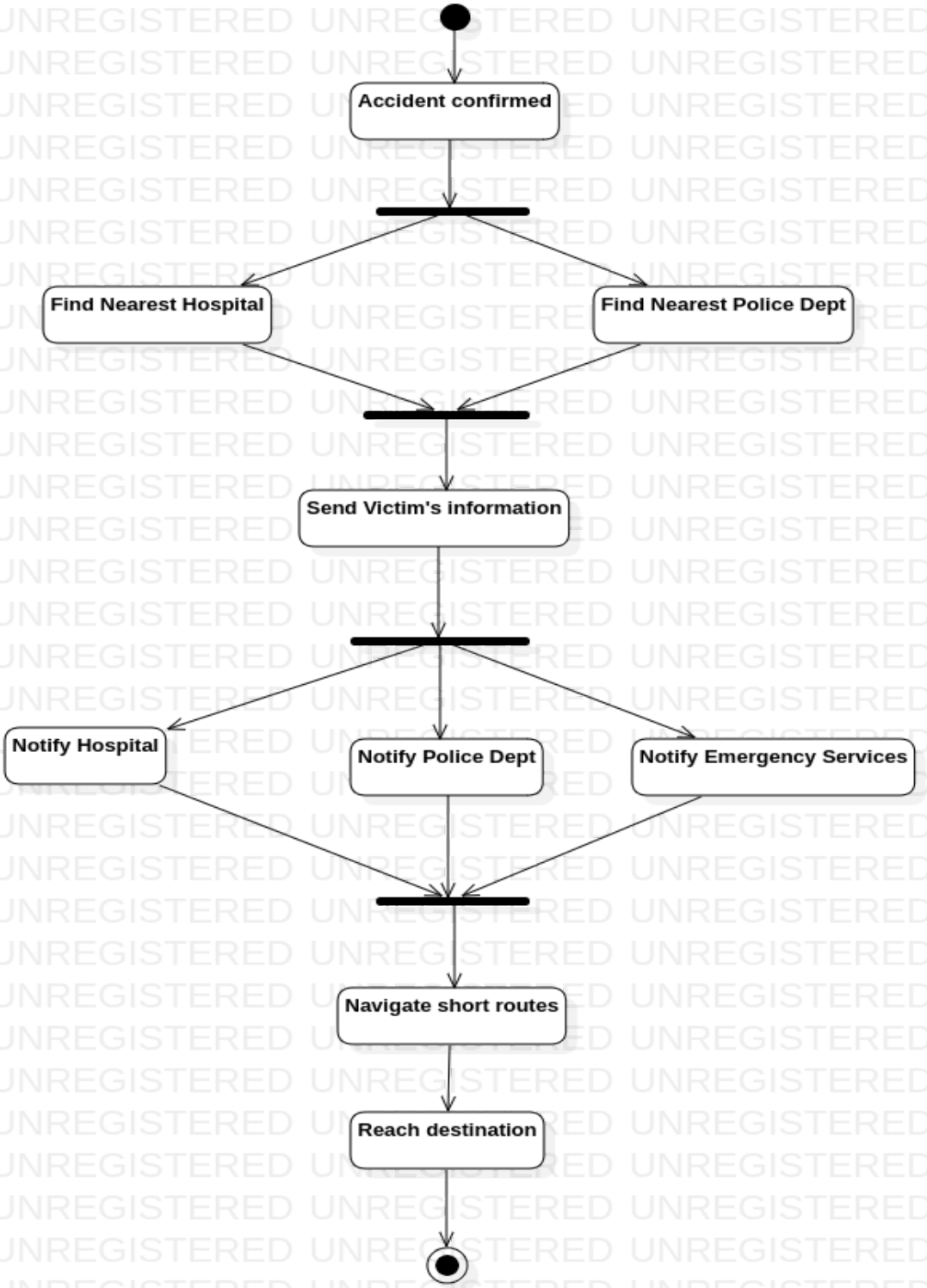
- If the value is greater than specified it goes to the next activity, that is detect accident
- The timer is activated.
- Incase of false positive it can be aborted
- If the duration of timer runs out it moves to the next activity, that is confirm accident



ACTIVITY DIAGRAM 2:

(notify emergency services)

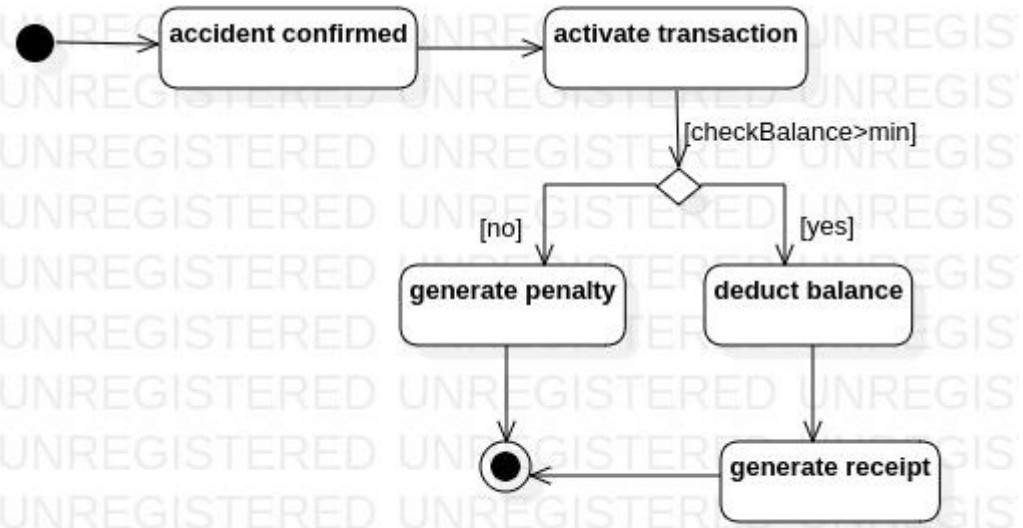
- As accident is confirmed it will move to the next activity
- The next activity is find nearest hospital and find nearest police station , which will occur parallelly
- After finding all the required details are sent to nearest hospital and police station found and to the pre-selected emergency contacts parallely
- They navigate through shortest routes and reach destination


Accident confirmed
Find Nearest Hospital
Find Nearest Police Dept
Send Victim's information
Notify Hospital
Notify Police Dept
Notify Emergency Services
Navigate short routes
Reach destination

ACTIVITY DIAGRAM 3:

(bank transaction)

- When an accident is confirmed , the next activity to be confirmed is transaction
- After checking balance , if balance is greater than the min balance the next activate will be deduct balance else the next activity will be generate penalty.
- The receipt is generated



CONCLUSION

Detecting accidents through the smart phone application is an innovative approach to minimize the deaths caused due to road accidents as it notifies the emergency services and preselected emergency contacts immediately without any delay.

As a part of this project, we diligently worked on the various models associated with the Software Development Life Cycles namely: Use case modelling, Domain Modelling, Class Models, Sequence Diagrams, State Diagrams and Activity Diagrams. The outcome of these models was a better indepth understanding of our problem domain, it's behaviour and functionalities. By modelling these diagrams we have laid a strong foundation to proceed to the coding and implementation phase.

FUTURE OF THE SCOPE

We started the project with the intent of building a startup. The software or the application developed can be not only limited to traffic accident monitoring , it can be extended to various other emergencies like health, fire emergencies etc.

CODE GENERATION

```
import java.util.*;
/**
 *
 */
public class User {
    /**
     * Default constructor
     */
    public User() {
    }
    /**
     *
     */
    public void username:string;
    /**
     *
     */
    public void password:string;
    /**
     *
     */
    public void Login(u:string; p:string)() {
        // TODO implement here
```

```
    }
}
import java.util.*;
/**
 *
 */
public class Vehicle {

    /**
     * Default constructor
     */
    public Vehicle() {
    }

    /**
     *
     */
    public String Type;

    /**
     *
     */
    public String Color;

    /**
     *
     */
    public String Model;

    /**
     *
```

```
     */
    public String Number;

    /**
     *
     */
    public String Company;



    /**
     *
     */
    public void Get_VehicleInfo() {
        // TODO implement here
    }

    /**
     *
     */
    public void Store_Info() {
        // TODO implement here
    }

}
import java.util.*;
/**
 *
 */
public class Victim Information {
    /**
```

```
 * Default constructor
 */
public Victim Information() {
}
/**
 *
 */
public String Name;
/**
 *
 */
public String DOB;
/**
 *
 */
public String Gender;
/**
 *
 */
public String Bloodgroup;
/**
 *
 */
public String Diseases;
/**
 *
 */
public void AddInfo() {
    // TODO implement here
}
/**
```

```
     *
     */
    public void Delete_Info() {
        // TODO implement here
    }
    /**
     *
     */
    public void Edit_Info() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Accelerometer {
    /**
     * Default constructor
     */
    public Accelerometer() {
    }
    /**
     *
     */
    public void Forcevalue:;
    /**
     *
     */
    public void Attribute2;
    /**
```

```
     *
     */
    public void Twistvalue:;
    /**
     *
     */
    public void Calculate_Force() {
        // TODO implement here
    }
    /**
     *
     */
    public void Calculate_Twist() {
        // TODO implement here
    }

}
import java.util.*;
/**
 *
 */
public class Crash {
    /**
     * Default constructor
     */
    public Crash() {
    }
    /**
     *
     */
    public Date Date;
```

```
    /**
     *
     */
    public String Time;
    /**
     *
     */
    public void ConfirmAccident() {
        // TODO implement here
    }
    /**
     *
     */
    public void Operation2() {
        // TODO implement here
    }
    /**
     *
     */
    public void GetTime() {
        // TODO implement here
    }
    /**
     *
     */
    public void Notify() {
        // TODO implement here
    }
}
import java.util.*;
/**
```

```
 *
 */
public class Timer {
    /**
     * Default constructor
     */
    public Timer() {
    }
    /**
     *
     */
    public Int TimeDuration;
}
import java.util.*;
/**
 *
 */
public class Accident {
    /**
     * Default constructor
     */
    public Accident() {
    }
    /**
     *
     */
    public void ConfirmAccident() {
        // TODO implement here
    }
    /**
     *
```

```
     */
    public void AbortAccident() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Accident Spot {
    /**
     * Default constructor
     */
    public Accident Spot() {
    }
    /**
     *
     */
    public String City;
    /**
     *
     */
    public String Pincode;
    /**
     *
     */
    public String Coordintes;
    /**
     *
     */
    public void GetCoordinates() {
```

```
        // TODO implement here
    }
    /**
     *
     */
    public void Find_Hospital() {
        // TODO implement here
    }
    /**
     */
    public void Find_PoliceDept() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Emergency Service{abstract} {
    /**
     * Default constructor
     */
    public Emergency Service{abstract}() {
    }
    /**
     *
     */
    public void Rescue{abstract}() {
        // TODO implement here
    }
}
```

```
import java.util.*;
/**
 *
 */
public class Nearest Hospital extends Emergency Service{abstract} {
    /**
     * Default constructor
     */
    public Nearest Hospital() {
    }
    /**
     *
     */
    public String Name;
    /**
     *
     */
    public String Contact;
    /**
     *
     */
    public String Email;
    /**
     *
     */
    public String Deptid;
    /**
     *
     */
    public void Get_CrashInfo() {
        // TODO implement here
```

```
    }
    /**
     *
     */
    public void Get_VictimDetails() {
        // TODO implement here
    }
    /**
     *
     */
    public void Update_Status() {
        // TODO implement here
    }
    /**
     *
     */
    public void Rescue() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Nearest Police Dept extends Emergency Service{abstract} {
    /**
     * Default constructor
     */
    public Nearest Police Dept() {
    }
    /**
```

```
 */
public String Name;
/**
 *
 */
public String Contact;
/**
 *
 */
public String Email;
/**
 *
 */
public String Deptid;
/**
 *
 */
public void Get_CrashInfo() {
    // TODO implement here
}
/**
 *
 */
public void Get_VictimDetails() {
    // TODO implement here
}
/**
 *
 */
public void Update_Status() {
    // TODO implement here
```

```
    }
    /**
     *
     */
    public void Rescue() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Location {
    /**
     * Default constructor
     */
    public Location() {
    }
    /**
     *
     */
    public String City;
    /**
     *
     */
    public String Pincode;
    /**
     *
     */
    public String Coordinates;
    /**
```

```
     *
     */
    public void GetCoordinates() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Emergency Contact {
    /**
     * Default constructor
     */
    public Emergency Contact() {
    }
    /**
     *
     */
    public void Attribute1;
    /**
     *
     */
    public String Name;
    /**
     *
     */
    public String Contact;
    /**
     *
     */
```

```
public String Emailid;
/**
 *
 */
public void Get_CrashInfo() {
    // TODO implement here
}
/**
 *
 */
public void Get_VictimDetails() {
    // TODO implement here
}
/**
 *
 */
public void View_HDetails() {
    // TODO implement here
}
/**
 *
 */
public void View_PDetails() {
    // TODO implement here
}
/**
 *
 */
public void View_Status() {
    // TODO implement here
}
```

```
    /**
     *
     */
    public void View_MedReport() {
        // TODO implement here
    }
    /**
     *
     */
    public void View_FIR() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Emergency Contact {
    /**
     * Default constructor
     */
    public Emergency Contact() {
    }
    /**
     *
     */
    public void Attribute1;
    /**
     *
     */
    public String Name;
```

```
/**
 *
 */
public String Contact;
/**
 *
 */
public String Emailid;
/**
 *
 */
public void Get_CrashInfo() {
    // TODO implement here
}
/**
 *
 */
public void Get_VictimDetails() {
    // TODO implement here
}
/**
 *
 */
public void View_HDetails() {
    // TODO implement here
}
/**
 *
 */
public void View_PDetails() {
    // TODO implement here
```

```
    }
    /**
     *
     */
    public void View_Status() {
        // TODO implement here
    }
    /**
     *
     */
    public void View_MedReport() {
        // TODO implement here
    }
    /**
     *
     */
    public void View_FIR() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Med_Report {
    /**
     * Default constructor
     */
    public Med_Report() {
    }
    /**
```

```
     *
     */
    public void AlcoholCheck;
    /**
     *
     */
    public void Injuries;
    /**
     *
     */
    public void Others;
    /**
     *
     */
    public void Store_MedReport() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Policemen {
    /**
     * Default constructor
     */
    public Policemen() {
    }
    /**
     *
     */
```

```
public String Name;
/**
 *
 */
public String Id;
/**
 *
 */
public String Description;
/**
 *
 */
public void Get_PoliceDet() {
    // TODO implement here
}
/**
 *
 */
public void File_FIR() {
    // TODO implement here
}
/**
 *
 */
public void View_MedReport() {
    // TODO implement here
}
/**
 *
 */
public void View_Vehicle() {
```

```
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class FIR {
    /**
     * Default constructor
     */
    public FIR() {
    }
    /**
     *
     */
    public String Description;
    /**
     *
     */
    public String Time;
    /**
     *
     */
    public Date Date;
   /**
     *
     */
    public void Store_FIR() {
        // TODO implement here
    }
```

```
}
import java.util.*;
/**
 *
 */
public class Status {
    /**
     * Default constructor
     */
    public Status() {
    }
    /**
     *
     */
    public String H_Status;
    /**
     *
     */
    public String P_Status;
    /**
     *
     */
    public void Store_Status() {
        // TODO implement here
    }
}
```

BANK ACCOUNT:

```
            import java.util.*;
/**
 *
 */
```

```
public class Bank Account {
    /**
     * Default constructor
     */
    public Bank Account() {
    }
    /**
     *
     */
    private String Number;
    /**
     *
     */
    private String Balance;
    /**
     *
     */
    private void GetAccDet() {
        // TODO implement here
    }
    /**
     *
     */
    private void CheckBalance() {
        // TODO implement here
    }
}
BANK:
import java.util.*;
/**
 *
```

```
 */
public class Bank {
    /**
     * Default constructor
     */
    public Bank() {
    }

    /**
     *
     */
    public void IFSC Code:;
    /**
     *
     */
    public void Get_BankCode() {
        // TODO implement here
    }
}
```

TRANSACTION:

```
import java.util.*;
/**
 *
 */
public class Transactions {
    /**
     * Default constructor
     */
    public Transactions() {
    }
    /**
```

```
     *
     */
    public void T_id:String;
    /**
     *
     */
    public void Amount:Longint;
    /**
     *
     */
    private void Update_Balance() {
        // TODO implement here
    }
}
INSURANCE POLICY:
import java.util.*;
/**
 *
 */
public class Insurance Policy {
    /**
     * Default constructor
     */
    public Insurance Policy() {
    }
    /**
     *
     */
    public String Id;
    /**
     *
```

```
     */
    public String Type;
    /**
     *
     */
    public String Valid duration;
    /**
     *
     */
    public Longint Amount;
    /**
     *
     */
    public void Get_InsuranceDet() {
        // TODO implement here
    }
}
```

DOCUMENT VERIFICATION:

```
import java.util.*;
/**
 *
 */
public class Document Verification {
    /**
     * Default constructor
     */
    public Document Verification() {
    }
    /**
     *
     */
```

```
    public Boolean FIR;
    /**
     *
     */
    public Boolean Med;
    /**
     *
     */
    public Boolean Vehicle;
    /**
     *
     */
    public void ViewFIR() {
        // TODO implement here
    }
    /**
     *
     */
    public void View_MedReport() {
        // TODO implement here
    }
    /**
     *
     */
    public void View_Insurance() {
        // TODO implement here
    }
}
```

INSURANCE COMPANY:

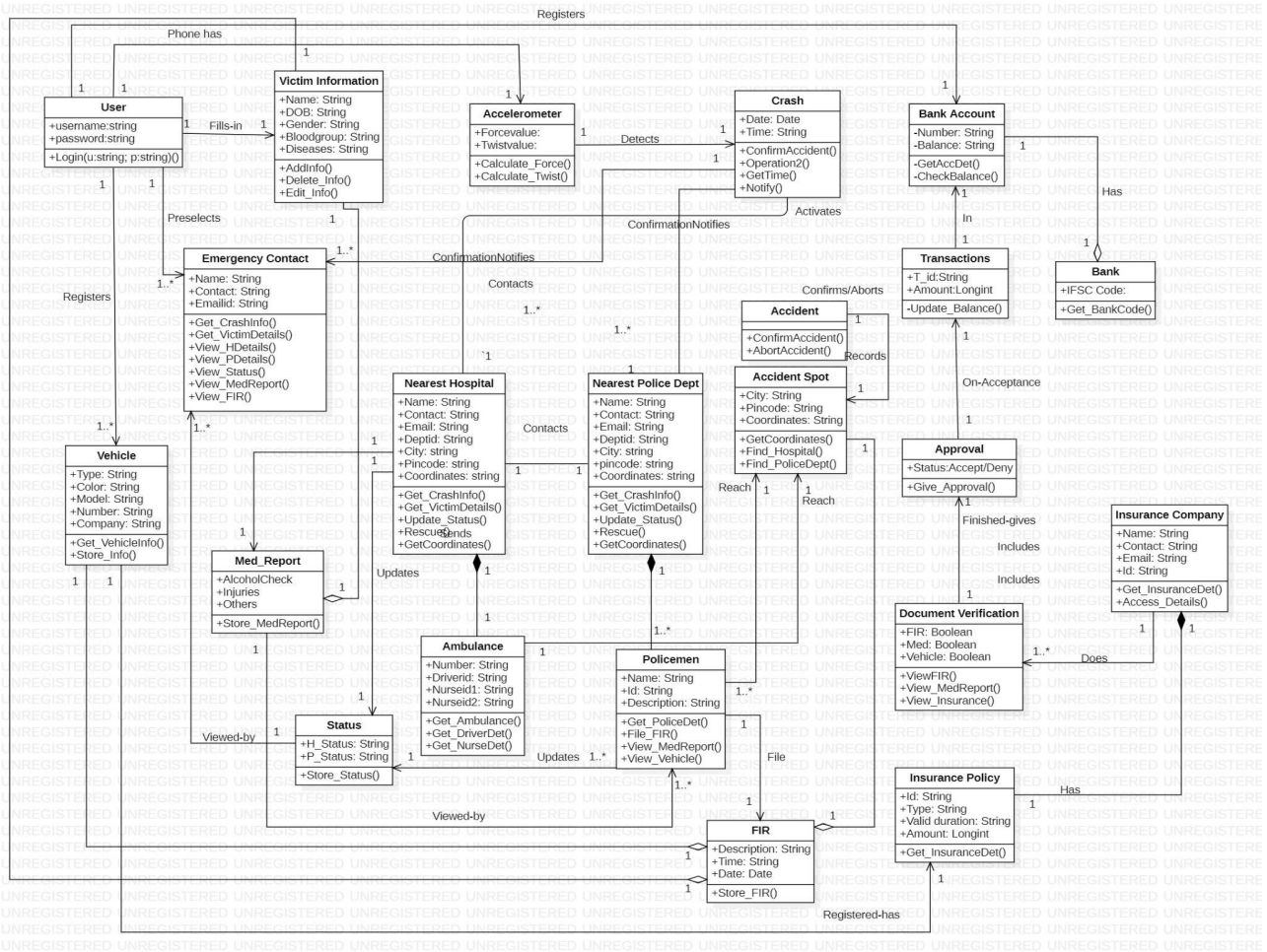
```
import java.util.*;
/**
```

```
 *
 */
public class Insurance Company {
    /**
     * Default constructor
     */
    public Insurance Company() {
    }
    /**
     *
     */
    public String Name;
    /**
     *
     */
    public String Contact;
    /**
     *
     */
    public String Email;
    /**
     *
     */
    public String Id;
    /**
     *
     */
    public void Get_InsuranceDet() {
        // TODO implement here
    }
    /**
```

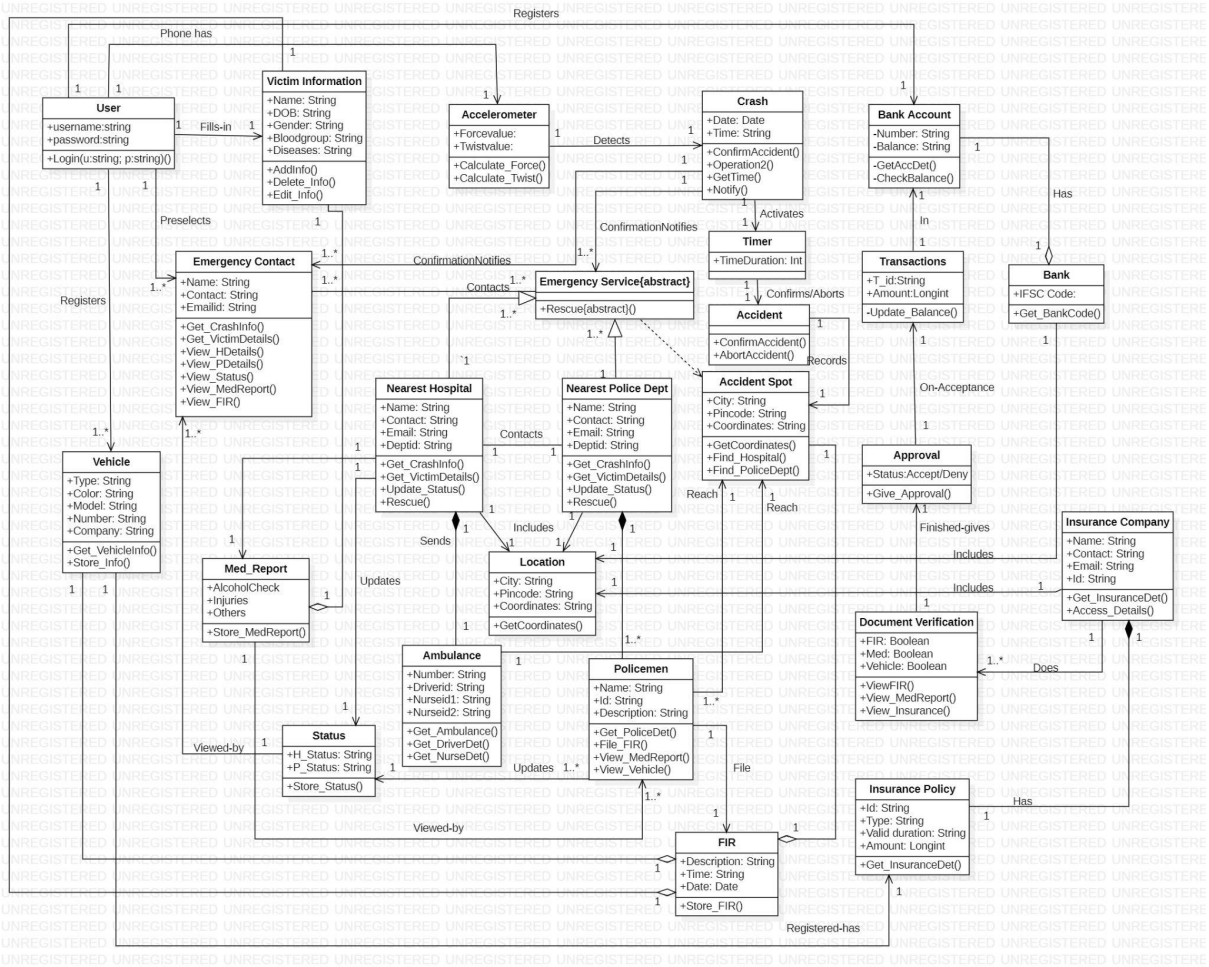
```
     *
     */
    public void Access_Details() {
        // TODO implement here
    }
}
APPROVAL:
import java.util.*;
/**
 *
 */
public class Approval {
    /**
     * Default constructor
     */
    public Approval() {
    }
    /**
     *
     */
    public void Status:Accept/Deny;
    /**
     *
     */
    public void Give_Approval() {
        // TODO implement here
    }
}
```

APPLYING PATTERNS FOR FEW CLASSES

OLD CLASS DIAGRAM



NEW CLASS DIAGRAM



INDIRECTION:

Where to assign a resposibility to avoid direct coupling between two or more things?
How to develop objects so that low coupling is supported and reuse potential remains higher?

Assign the responsibility to an intermediate object between other components or services. This intermediary creates an indirection between the other components.

Here the intermediary in our project is emergency services

OLD CODE:

```
import java.util.*;

/**
 *
 */
public class Nearest Hospital {

    /**
     * Default constructor
     */
    public Nearest Hospital() {
    }

    /**
     *
     */
    public String Name;

    /**
     *
     */
    public String Contact;

    /**
     *
     */
    public String Email;

    /**
     *
     */
```

```
public String Deptid;

/**
 *
 */
public string City;

/**
 *
 */
public string Pincode;

/**
 *
 */
public string Coordinates;
/**
 *
 */
public void Get_CrashInfo() {
    // TODO implement here
}
/**
 *
 */
public void Get_VictimDetails() {
    // TODO implement here
}
/**
 *
 */
public void Update_Status() {
    // TODO implement here
```

```
    }
    /**
     *
     */
    public void Rescue() {
        // TODO implement here
    }
    /**
     *
     */
    public void GetCoordinates() {
        // TODO implement here
    }
    /**
     *
     */
public class Nearest Police Dept {
        /**
         * Default constructor
         */
        public Nearest Police Dept() {
        }
        /**
         *
         */
        public String Name;
        /**
         *
         */
        public String Contact;
        /**
         *
         */
```

```
public String Email;
/**
 *
 */
public String Deptid;
/**
 *
 */
public string City;
/**
 *
 */
public string pincode;
/**
 *
 */
public string Coordinates;
/**
 *
 */
public void Get_CrashInfo() {
    // TODO implement here
}
/**
 *
 */
public void Get_VictimDetails() {
    // TODO implement here
}
/**
 *
 */
public void Update_Status() {
```

```
        // TODO implement here
    }
    /**
     *
     */
    public void Rescue() {
        // TODO implement here
    }
    /**
     *
     */
    public void GetCoordinates() {
        // TODO implement here
    }
  }

}
```

NEW CODE:

```
 import java.util.*;
/**
 *
 */
public class Emergency Service{abstract} {
  /**
   * Default constructor
   */
  public Emergency Service{abstract}() {
  }
  /**
   *
   */
  public void Rescue{abstract}() {
    // TODO implement here
```

```
    }
}
import java.util.*;
/**
 *
 */
public class Nearest Hospital extends Emergency Service{abstract} {
    /**
     * Default constructor
     */
    public Nearest Hospital() {
    }
    /**
     *
     */
    public String Name;
    /**
     *
     */
    public String Contact;
    /**
     *
     */
    public String Email;
    /**
     *
     */
    public String Deptid;
    /**
     *
     */
    public void Get_CrashInfo() {
        // TODO implement here
    }
    /**
     *
```

```
     */
    public void Get_VictimDetails() {
        // TODO implement here
    }
    /**
     *
     */
    public void Update_Status() {
        // TODO implement here
    }

    /**
     *
     */
    public void Rescue() {
        // TODO implement here
    }
}
import java.util.*;
/**
 *
 */
public class Nearest Police Dept extends Emergency Service{abstract} {
    /**
     * Default constructor
     */
    public Nearest Police Dept() {
    }
    /**
     *
     */
    public String Name;
    /**
     *
     */
    public String Contact;
```

```
    /**
     *
     */
    public String Email;
    /**
     *
     */
    public String Deptid;
    /**
     *
     */
    public void Get_CrashInfo() {
        // TODO implement here
    }
    /**
     *
     */
    public void Get_VictimDetails() {
        // TODO implement here
    }
    /**
     *
     */
    public void Update_Status() {
        // TODO implement here
    }
    /**
     *
     */
    public void Rescue() {
        // TODO implement here
    }
}
```