POTHOLE TRACKER AND REPAIR MANAGEMENT SYSTEM

Image Reference: https://www.india.com/

CONTENTS:

ORGANIZATION DESCRIPTION

"More deadly than terrorism: Potholes responsible for killing 10 people a day in India" – guardian.com (Jul-25-2018)

Roads with potholes has become a ubiquitous problem in India. According to a leading daily newspaper, in 2018(till July) alone 9300 people were killed and nearly 25000 were injured in road accidents which were caused due to potholes. This motivated me to create a mobile application known as Pothole Stalker which will help the government in recognizing these potholes and carry out their repair as soon as possible to avoid the inconveniences caused to the public. This mobile application will use mobile in-build gyroscope and Global Positioning System(GPS) to track the movement of the vehicle and if in case the vehicle crosses a pothole, it will trigger an API call with relevant information about the pothole.
The purpose of this non – profit business organization is to help the Indian Government and also alert the user of this application so as to accidents that happen due to these potholes can be avoided.
For this application to work efficiently, database is of utmost importance. Pothole Tracker and Repair Management System will ensure that the purpose of the application is fulfilled.

Anybody can download the application. The database will record information about the User. If the user feels a bump, the data about the pothole such as latitude, longitude, depth, RoadID and timestamp will be recorded in a Pothole-Event table. One user can provide many pothole events. The pothole events will then be summarized as Min latitude, Max latitude, Min longitude, Max longitude to find the pothole region. The PotholeID will connect it to the road where the pothole region is identified. Data such as SpeedLimit permissible, date of construction completed, traffic density will be recorded.
India has the second largest road network in the world, these roads can be divided into three categories:

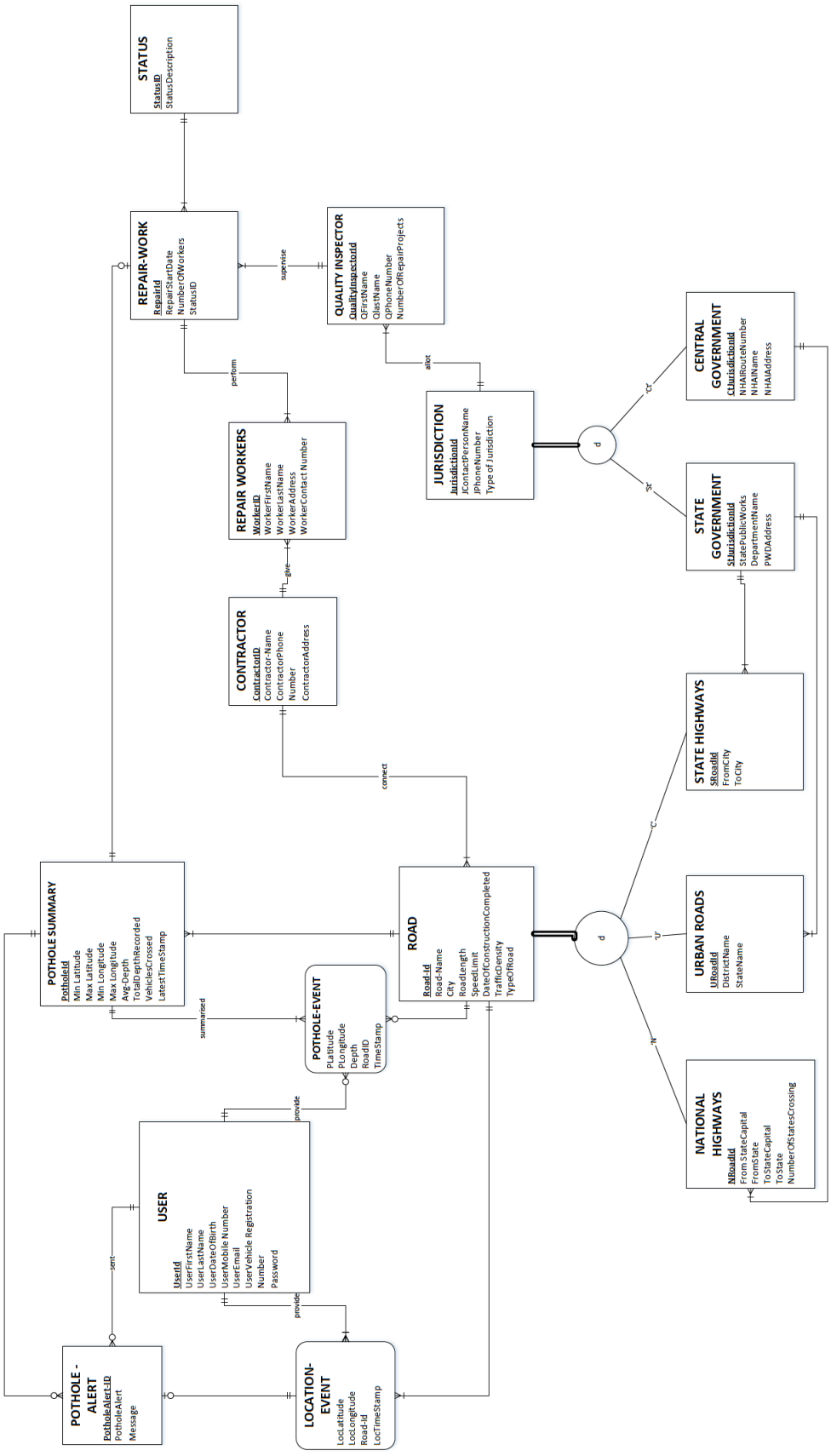
1. National Highways: that connect one state to another
2. State Highways: that connect cities within a state
3. Urban Roads: that are used for travelling within a city.

These roads come under different jurisdiction. For example Central government is responsible for construction and maintenance of National Highways (National Highway Authority of India) and State Highways and Urban Roads construction and maintenance is the responsibility of the State government(State Public Works Department). Also in India, roads are constructed and maintained with the help of the public private partnership. After recognizing the pothole region and the type of road, the respective jurisdiction with the help of the Contractor will start a repair work and the status of the repair will also be recorded in the repair table. The Contractor who built the road will provide repair workers and Quality Inspector allocated by the jurisdiction will oversee the repair work. Each Quality Inspector will supervise many repair projects. The application will also track User's latitude and longitude every minute of the drive which will be recorded in the Location Event table. If the distance between the pothole region and user is 1km, an alert will be recorded in the alert table.

Values Gained by Database Development

1. Main purpose of this Pothole Tracking and Repair Management is to save as many lives as possible as this system will provide assistance in speeding up the repair work of the pothole.
2. Corruption and incompetence of various parties involved in road construction and management can also be kept in check.
3. Citizens can contribute and help the government in keeping track of the road condition and its maintenance

ENTITY – RELATIONSHIP DIAGRAM



CREATE TABLE QUERIES

1. User

```
CREATE TABLE User_T
  (
     UserID                    bigint NOT NULL CHECK(UserID > 0),
     UserFirstName             nvarchar(100) NOT NULL,
     UserLastName              nvarchar(100) NOT NULL,
     UserDateOfBirth           date,
     UserMobileNumber          char(10) NOT NULL,
     UserEmail                 nvarchar(200),
     VehicleRegistrationNumber varchar(20),
     Password                  nvarchar(100) NOT NULL,
     CONSTRAINT User_PK PRIMARY KEY (UserID))
```

2. AddressPincode

   As address was used in many tables, the city and state are transitively dependent on pincode.So I have created this table to full the requirements of 3NF

```
  CREATE TABLE AddressPincode_T (
     PincodeID bigint not null check(PincodeID > 0),
     Pincode   int,
     City      nvarchar(100),
     StateName nvarchar(100),
     Constraint AddressPincode_PK primary key(PincodeID))
```

3. Contractor

   This table stores the contractor details and PincodeID is used as a foreign key to store the contractor's address

```
CREATE TABLE Contractor_T (
     ContractorID          bigint NOT NULL CHECK(ContractorID > 0),
     ContractorName        nvarchar(100) NOT NULL,
     ContractorPhoneNumber char(10),
     ContractorStreet      nvarchar(100),
     PincodeID             bigint NOT NULL CHECK(PincodeID > 0),
     CONSTRAINT Contractor_PK PRIMARY KEY(ContractorID),
     CONSTRAINT AddressPincode_FK FOREIGN KEY(PincodeID) REFERENCES
     AddressPincode_T(PincodeID))
```

4. Road

   This table stores the details about road and ContractorID is used as a foreign key to know which contractor built this road.

```
CREATE TABLE Road_T (
     RoadID                      bigint NOT NULL CHECK(RoadID > 0),
     RoadName                    nvarchar(100) NOT NULL,
     RoadLength                  bigint,
     SpeedLimit                  int,
     DateOfConstructionCompleted date,
     TrafficDensity              int,
     ContractorID                bigint NOT NULL CHECK(ContractorID > 0),
     TypeOfRoad                  varchar(3) CHECK(TypeOfRoad IN ('N', 'S', 'U'))
     NOT NULL,
     CONSTRAINT Road_PK PRIMARY KEY(RoadID),
     CONSTRAINT Contractor_FK FOREIGN KEY(ContractorID) REFERENCES Contractor_T(
     ContractorID))
```

5. Jurisdiction

   This table stores the details of the person to be contacted in jurisdidction for pothole repair and informs about the type of jurisdiction

```
CREATE TABLE Jurisdiction_T (
     JurisdictionID     bigint NOT NULL CHECK(JurisdictionID > 0),
     JContactPersonName nvarchar(100),
     JPhoneNumber       char(10),
     TypeOfJurisdiction varchar(2) CHECK(TypeOfJurisdiction IN('St', 'Ct')) NOT
     NULL,
     CONSTRAINT Jurisdiction_PK PRIMARY KEY(JurisdictionID))
```

6. StateGovernment

   This table stores details about the state government who will be responsible for state highways and urban roads. As it is a subtype of jurisdiction,its primary key will be used as a foreign key

```
CREATE TABLE StateGovernment_T (
     StJurisdictionID bigint NOT NULL CHECK(StJurisdictionID > 0),
     PWDName          nvarchar(100),
     PWDStreetName    nvarchar(100),
     PincodeID        bigint NOT NULL CHECK(PincodeID > 0),
     CONSTRAINT StateGovernment_PK PRIMARY KEY(StJurisdictionID),
     CONSTRAINT StateGovernment_FK1 FOREIGN KEY(StJurisdictionID) REFERENCES
     Jurisdiction_T(JurisdictionID),
     CONSTRAINT AddressPincode_FK2 FOREIGN KEY(PincodeID) REFERENCES
     AddressPincode_T(PincodeID))
```

7. CentralGovernment

   This table stores details about the central government who will be responsible for National highways As it is a subtype of jurisdiction, its primary key will be used as a foreign key

```
CREATE TABLE CentralGovernment_T (
     CtJurisdictionID   bigint NOT NULL CHECK(CtJurisdictionID > 0),
     NHAIRouteNumber    int,
     NHAIRegionalOffice nvarchar(100),
     NHAIStreetName     nvarchar(100),
     PincodeID          bigint NOT NULL CHECK(PincodeID > 0),
     CONSTRAINT CentralGovernment_PK PRIMARY KEY(CtJurisdictionID),
     CONSTRAINT CentralGovernment_FK1 FOREIGN KEY(CtJurisdictionID) REFERENCES
     Jurisdiction_T(JurisdictionID),
     CONSTRAINT AddressPincode_FK3 FOREIGN KEY(PincodeID) REFERENCES
     AddressPincode_T(PincodeID))
```

8. StateHighways

   This table stores details about the cities connected by state highways and it is a subtype of road its primary key is also its foreign key and foreign key StJurisdictionID informs which jurisdiction it comes under

```
CREATE TABLE StateHighways_T  (
     SRoadID          bigint NOT NULL CHECK(SRoadID > 0),
     FromCity         nvarchar(100),
     ToCity           nvarchar(100),
     StJurisdictionID bigint NOT NULL CHECK(StJurisdictionID > 0),
     CONSTRAINT StateHighways_PK PRIMARY KEY(SRoadID),
     CONSTRAINT StateHighways_FK1 FOREIGN KEY(SRoadID) REFERENCES Road_T(RoadID)
     ,
     CONSTRAINT StateGovernment_FK2 FOREIGN KEY(StJurisdictionID) REFERENCES
     StateGovernment_T(StJurisdictionID)
  )
```

9. NationalHighways

This table stores details about the state capital onnected by National highways and it is a subtype of road its primary key is also its foreign key and foreign key CtJurisdictionID informs which jurisdiction it comes under

```
CREATE TABLE NationalHighways_T (
     NRoadID               bigint NOT NULL CHECK(NRoadID > 0),
     FromStateCapital      nvarchar(100),
     FromState             nvarchar(100),
     ToStateCapital        nvarchar(100),
     ToState               nvarchar(100),
     NumberOfStatesCrossing int,
     CtJurisdictionID      bigint NOT NULL CHECK(CtJurisdictionID > 0)
     CONSTRAINT NationalHighways_PK PRIMARY KEY(NRoadID),
     CONSTRAINT NationalHighways_FK1 FOREIGN KEY(NRoadID) REFERENCES Road_T(
     RoadID),
     CONSTRAINT CentralGovernment_FK2 FOREIGN KEY(CtJurisdictionID) REFERENCES
     CentralGovernment_T(CtJurisdictionID))
```

10. UrbanRoads

    This table stores details about the district in the city where the road is and it is a subtype of road its primary key is also its foreign key and foreign key StJurisdictionID informs which jurisdiction it comes under

```
CREATE TABLE UrbanRoads_T (
     URoadID          bigint NOT NULL CHECK(URoadID > 0),
     UDistrictName    nvarchar(100),
     PincodeID        bigint NOT NULL CHECK(PincodeID > 0),
     StJurisdictionID bigint NOT NULL CHECK(StJurisdictionID > 0),
     CONSTRAINT UrbanRoads_PK PRIMARY KEY(URoadID),
     CONSTRAINT UrbanRoads_FK1 FOREIGN KEY(URoadID) REFERENCES Road_T(RoadID),
     CONSTRAINT StateGovernment_FK3 FOREIGN KEY(StJurisdictionID) REFERENCES
     StateGovernment_T(StJurisdictionID),
     CONSTRAINT AddressPincode_FK6 FOREIGN KEY(PincodeID) REFERENCES
     AddressPincode_T(PincodeID))
```

11. PotholeEvent

    This table stores details about the Pothole events sent by the user. Foreign key RoadID informs about which road it is on and foreign key informs about the user that provided it.

```
CREATE TABLE PotholeEvent_T (
     PotholeEventID bigint CHECK(PotholeEventID > 0),
     PLatitude      decimal(10, 8),
     PLongitude     decimal(11, 8),
     PotholeDepth   float,
     Timestamp      datetime,
     RoadID         bigint CHECK(RoadID > 0),
     PotholeID      bigint,
     UserID         bigint CHECK(UserID > 0),
     CONSTRAINT PotholeEvent_PK PRIMARY KEY(PotholeID),
     CONSTRAINT Road_FK1 FOREIGN KEY(RoadID) REFERENCES Road_T(RoadID),
     CONSTRAINT Pothole_FK2 FOREIGN KEY(PotholeID) REFERENCES PotholeSummary_T,
     CONSTRAINT User_FK2 FOREIGN KEY(UserID) REFERENCES User_T(UserID))
```

12. PotholeSummary

    This table summarizes the pothole event to find the pothole region and RoadID (foreign key) informs about the road where the pothole is

```
CREATE TABLE PotholeSummary_T (
     PotholeID          bigint NOT NULL IDENTITY(1, 1),
     MinLatitude        decimal(10, 8),
     MaxLatitude        decimal(10, 8),
```

```
    MinLongitude       decimal(11, 8),
    MaxLongitude       decimal(11, 8),
    AvgPotholeDepth    float,
    VehiclesCrossed    bigint,
    TotalDepthRecorded float,
    LatestTimestamp    datetime,
    RoadID             bigint CHECK(RoadID > 0),
    CONSTRAINT PotholeSummary_PK PRIMARY KEY (PotholeID),
    CONSTRAINT Road_FK FOREIGN KEY (RoadID) REFERENCES Road_T(RoadID))
```

13. StateHighways

    This table stores details about quality inspector who oversee the repair work and JurisdictionID (foreign key) tells jurisdiction allotted him/her

```
CREATE TABLE QualityInspector_T (
     QInspectorID           bigint NOT NULL CHECK(QInspectorID > 0),
     QFirstName             nvarchar(100),
     QLastname              nvarchar(100),
     QPhoneNumber           char(10),
     NumberOfRepairProjects int,
     JurisdictionID         bigint NOT NULL CHECK(JurisdictionID > 0)
     CONSTRAINT QualityInspector_PK PRIMARY KEY (QInspectorID),
     CONSTRAINT Jurisdiction_FK FOREIGN KEY (JurisdictionID) REFERENCES
     Jurisdiction_T(JurisdictionID))
```

14. Status

    This table stores repair work status.

```
CREATE TABLE Status_T (
     StatusID          bigint NOT NULL CHECK(StatusID > 0),
     StatusDescription nvarchar(100),
     CONSTRAINT Status_PK PRIMARY KEY (StatusID))
```

15. RepairWork

    This table stores details about the pothole repair work and foreign key StatusID informs the status, PotholeID tells which on pothole repair work is going on and QualityInpectorID tells which Quality Inspector is overseeing its work

```
CREATE TABLE RepairWork_T (
     RepairID        bigint NOT NULL CHECK(RepairID > 0),
     RepairStartDate date,
     NumberOfWorkers int,
     StatusID        bigint NOT NULL CHECK(StatusID > 0),
     PotholeID       bigint NOT NULL CHECK(PotholeID > 0),
     QInspectorID    bigint NOT NULL CHECK(QInspectorID > 0),
     CONSTRAINT RepairWork_PK PRIMARY KEY(RepairID),
     CONSTRAINT Status_FK1 FOREIGN KEY(StatusID) REFERENCES Status_T(StatusID),
     CONSTRAINT PotholeSummary_FK2 FOREIGN KEY(PotholeID) REFERENCES
     PotholeSummary_T(PotholeID),
     CONSTRAINT QualityInspector_FK3 FOREIGN KEY(QInspectorID) REFERENCES
     QualityInspector_T(QInspectorID))
```

16. RepairWorker

    This table stores details about the pothole repair workers. Foreign key repair id informs about the repair work, contractor id tells repair workers are from which contractor and pincode id is used for storing there address.

```
CREATE TABLE RepairWorkers_T (
     WorkerID          bigint NOT NULL CHECK(WorkerID > 0),
     WorkerFirstName   nvarchar(100),
     WorkerLastName    nvarchar(100),
     WorkerStreetName  nvarchar(100),
     PincodeID         bigint NOT NULL CHECK(PincodeID > 0),
     WorkerPhoneNumber char(10),
     ContractorID      bigint NOT NULL CHECK(ContractorID > 0),
```

```
RepairID          bigint CHECK(RepairId > 0),
CONSTRAINT RepairWorkers_PK PRIMARY KEY(WorkerID),
CONSTRAINT Contractor_FK1 FOREIGN KEY(ContractorID) REFERENCES Contractor_T
(ContractorId),
CONSTRAINT RepairWork_FK2 FOREIGN KEY(RepairID) REFERENCES RepairWork_T(
RepairID),
CONSTRAINT AddressPincode_FK3 FOREIGN KEY(PincodeID) REFERENCES
AddressPincode_T(PincodeID))
```

17. LocationEvent

When user is driving their mobile application is suppose to share location. This table collects location, road information (road id) and user information(user_id) coming as the foreign key.

```
CREATE TABLE LocationEvent_T (
     LocationEventID bigint CHECK(LocationEventID > 0),
     LocLatitude decimal(10, 8),LocLongitude decimal(11, 8),
     LocTimeStamp datetime,RoadID bigint CHECK(RoadID > 0),
     UserID bigint CHECK(UserID > 0),
     CONSTRAINT LocationEvent_PK PRIMARY KEY(LocationEventID),
     CONSTRAINT Road_FK1 FOREIGN KEY(RoadID) REFERENCES Road_T(RoadID),
     CONSTRAINT User_FK2 FOREIGN KEY(UserID) REFERENCES User_T(UserID)
  )
```

18. Pothole Alert

If user is close to the pothole, trigger will automatically run on every entry of location event table, if user is close to the pothole then an alert entry is made to this table. This table contains user information, location information and pothole information as the foreign keys.

```
CREATE TABLE PotholeAlert_T (
     PotholeAlertID bigint NOT NULL CHECK(AlertID > 0),
     PotholeAlertMessage nvarchar(200),UserID bigint NOT NULL CHECK(UserID > 0),
     LocationEventID not NULL CHECK(LocationEventID > 0),
     CONSTRAINT PotholeAlert_PK PRIMARY KEY(AlertID),
     CONSTRAINT User_FK1 FOREIGN KEY(UserID) REFERENCES User_T(UserID)
     CONSTRAINT LocationEvent_FK2 FOREIGN KEY(LocationEventID) REFERENCES
     LocationEvent_T(LocationEventID)
     CONSTRAINT PotholeSummary_FK3 FOREIGN KEY(PotholeID) REFERENCES
     PotholeSummary_T(PotholeID)
  )
```

TRIGGER

TRIGGER – POTHOLE EVENT

On every row insertion the TRIGGER will look whether this pothole event is close to the values already present in the pothole summary table. For that I am using HAVERSINE formula to compare lat/long for the POTHOLE_EVENT with min/max lat/long of all the unique pothole entries in the summary table :

IF this lat/long is closer to the boundaries then the trigger will update the **PotholeSummary** table as it will find that pothole already exist.
ELSE trigger will take this as a new pothole and insert it as a new row in the summary table.

In the end PotholeID from **PotholeSummary** Table gets updated to **PotholeEventTable** for relationship mapping.

```
CREATE TRIGGER [dbo].[trigger1]
ON [dbo].[PotholeEvent_T]
after INSERT
AS
    DECLARE @pLat DECIMAL(10, 8),
@pLong DECIMAL(11, 8),  @pDepth
FLOAT, @pTimestamp DATETIME,@roadId BIGINT,
@userId BIGINT,@pEventId BIGINT,
@minLat DECIMAL(10, 8), @maxLat
DECIMAL(10, 8),@minLong DECIMAL(11, 8),
@maxLong DECIMAL(11, 8),@avgDepth
FLOAT, @totalDepthRecorded FLOAT, @potholeId
BIGINT, @vehicleCrossed BIGINT,
@latestTimestamp DATETIME

    SELECT @pLat = a.PLatitude,@pLong = a.PL
ongitude,@pDepth = a.PotholeDepth,
@roadId = a.RoadID,@userId = a.UserID,@pTime
stamp = a.Timestamp,@pEventId = a.PotholeEve
ntID,@minLat = b.MinLatitude,@maxLat = b.Max
Latitude,@minLong = b.MinLongitude,@maxLong
= b.MaxLongitude,@potholeId = b.PotholeID,@v
ehicleCrossed = b.VehiclesCrossed,@latestTim
estamp = b.LatestTimestamp,@totalDepthRecord
ed = b.TotalDepthRecorded

FROM   inserted a,PotholeSummary_T b
WHERE  a.RoadID = b.RoadID
            -
- this is to check whether new pothole event
s fits in the existing summary
            -
- or its a new pothole which we have to inse
rt
AND (1609.34 * 2 * 3961 * asin(sqrt(power((s
in(radians((b.MinLatitude - a.PLatitude) / 2
))), 2) + cos(radians(a.PLatitude)) * cos(ra
dians(b.MinLatitude)) * power((sin(radians((
b.MinLongitude - a.PLongitude) / 2))),2))) <
 10.0
       OR     1609.34 * 2 * 3961 * asin(sqrt
(power((sin(radians((b.MaxLatitude - a.PLati
tude) / 2))), 2) + cos(radians(a.PLatitude))
 * cos(radians(b.MaxLatitude)) * power((sin(
radians((b.MinLongitude - a.PLongitude) / 2)
)),2))) < 10.0
       OR     1609.34 * 2 * 3961 * asin(sqrt
(power((sin(radians((b.MinLatitude - a.PLati
tude) / 2))), 2) + cos(radians(a.PLatitude))
 * cos(radians(b.MinLatitude)) * power((sin(
radians((b.MaxLongitude - a.PLongitude) / 2)
)),2))) < 10.0
       OR     1609.34 * 2 * 3961 * asin(sqrt
(power((sin(radians((b.MaxLatitude - a.PLati
tude) / 2))), 2) + cos(radians(a.PLatitude))
 * cos(radians(b.MaxLatitude)) * power((sin(
radians((b.MaxLongitude - a.PLongitude) / 2)
)),2))) < 10.0  )

    IF @pLat < @minLat
      SET @minLat = @pLat

    IF @pLat > @maxLat
      SET @maxLat = @pLat

    IF @pLong < @minLong
      SET @minLong = @pLong

    IF @pLong > @maxLong
      SET @maxLong = @pLong

    IF Datediff(second, @pTimestamp, @latest
Timestamp) > 0
      SET @pTimestamp = @latestTimestamp

    IF @pLat IS NOT NULL
      BEGIN
          PRINT 'updated row  :'
                + Cast (@potholeId AS VARCHA
R(100))

          UPDATE PotholeSummary_T
          SET    MinLatitude = @minLat,MaxLa
titude = @maxLat, MinLongitude = @minLong,
MaxLongitude = @maxLong,AvgPotholeDepth = (
@totalDepthRecorded
+ @pDepth) / (@vehicleCrossed + 1),
VehiclesCrossed = @vehicleCrossed + 1,
LatestTimestamp = @pTimestamp,
TotalDepthRecorded = @totalDepthRecorded + @
pDepth

WHERE  PotholeID = @potholeId
      END
    ELSE
      BEGIN
          PRINT 'new row inserted'

          SELECT @pEventId = inserted.Pothol
eEventID
          FROM   inserted

          INSERT INTO PotholeSummary_T
          SELECT inserted.PLatitude,inserted
.PLatitude,inserted.PLongitude,
                 inserted.PLongitude,
                 inserted.PotholeDepth,1,-
- vehicles crossed,
                 inserted.PotholeDepth,inser
ted.Timestamp,inserted.RoadID
          FROM   inserted

          SELECT @potholeId = PotholeSummary
_T.PotholeID
          FROM   PotholeSummary_T,inserted
          WHERE  PotholeSummary_T.MinLatitud
e = inserted.PLatitude
                 AND PotholeSummary_T.MaxLat
itude = inserted.PLatitude
                 AND PotholeSummary_T.MinLon
gitude = inserted.PLongitude
                 AND PotholeSummary_T.MaxLon
gitude = inserted.PLongitude
                 AND PotholeSummary_T.RoadID
 = inserted.RoadID
      END

    UPDATE PotholeEvent_T
    SET    PotholeID = @potholeId
    WHERE  PotholeEventID = @pEventId
```

TRIGGER - LOCATION EVENT

The application will track the user's location such as latitude, longitude and every event gets recorded in **location_event** table. On every location event entry, trigger will execute to check if user is coming closer to pothole. If user comes with 1 km range of a pothole then trigger will fire an alert in the the alert_table.

```
CREATE TRIGGER [dbo].[Trigger2]
ON [dbo].[LocationEvent_T]
AFTER INSERT
AS
    DECLARE @lLat decimal(10, 8), @lLong
decimal(11, 8), @lTimestamp  datetime, @road
Id  bigint,  @userId          bigint,
@StatusId  bigint,  @locationEventId bigint,
@potholeId  bigint,  @minLat
decimal(10, 8),  @maxLat  decimal(10, 8),  @
minLong
decimal(11, 8),  @maxLong  decimal(11, 8),
@latestTimestamp datetime

    SELECT @lLat = a.LocLatitude,  @lLong =
a.LocLongitude,  @lTimestamp = a.LocTimeStam
p,  @userId = a.UserID,  @locationEventId =
a.LocationEventID,  @minLat = b.MinLatitude,
@maxLat = b.MaxLatitude,  @minLong = b.MinLo
ngitude,  @maxLong = b.MaxLongitude,  @potho
leId = b.PotholeID

FROM   inserted a,  PotholeSummary_T b
WHERE  a.RoadID = b.RoadID
AND ( 1609.34 * 2 * 3961 * asin(sqrt(power((
sin(radians((b.MinLatitude - a.PLatitude) /
2))), 2) + cos(radians(a.PLatitude)) * cos(r
adians(b.MinLatitude)) * power((sin(radians(
(b.MinLongitude - a.PLongitude) / 2))),2)))
< 1000.0
      OR     1609.34 * 2 * 3961 * asin(sqrt
(power((sin(radians((b.MaxLatitude - a.PLati
tude) / 2))), 2) + cos(radians(a.PLatitude))
 * cos(radians(b.MaxLatitude)) * power((sin(
radians((b.MaxLongitude - a.PLongitude) / 2)
),2))) < 1000.0
      OR     1609.34 * 2 * 3961 * asin(sqrt
(power((sin(radians((b.MinLatitude - a.PLati
tude) / 2))), 2) + cos(radians(a.PLatitude))
 * cos(radians(b.MinLatitude)) * power((sin(
radians((b.MinLongitude - a.PLongitude) / 2)
),2))) < 1000.0
      OR     1609.34 * 2 * 3961 * asin(sqrt
(power((sin(radians((b.MaxLatitude - a.PLati
tude) / 2))), 2) + cos(radians(a.PLatitude))
 * cos(radians(b.MaxLatitude)) * power((sin(
radians((b.MaxLongitude - a.PLongitude) / 2)
),2))) < 1000.0 )

IF @lLat IS NOT NULL
 BEGIN
  SELECT @StatusId = StatusID
  FROM   RepairWork_T
  WHERE  @potholeId = RepairWork_T.PotholeID

  PRINT 'new row inserted'
  IF @StatusId IS NULL OR @StatusId = 1
   INSERT INTO PotholeAlert_T VALUES('Pothol
e open, please drive carefully',  @userId,
@locationEventId,  @potholeId)


ELSE IF @StatusId = 2 INSERT INTO PotholeAle
rt_T VALUES     ('Pothole repair work is in
progress, please drive carefully' , @userId,
  @locationEventId,  @potholeId)

END
```

TRIGGER – STATUS CHANGE LOG

This trigger keep track of the status id which gets updated in the repair work, whenever the status id is updated, a row gets inserted in the **StatusUpdates_Log** table. This can assist in knowing what is the current status of the repair work and when the pothole was closed.

```
CREATE TRIGGER [dbo].[StatusUpdate]
ON [dbo].[RepairWork_T]
FOR UPDATE, INSERT
AS
    IF UPDATE(StatusID)
      BEGIN
          INSERT INTO StatusUpdates_Log
                      (StatusID,
                       NewStatusID,
                       RepairID,
                       UpdateTimeStamp)
            SELECT deleted.StatusID,
                   inserted.StatusID,
                   inserted.RepairID,
                   Getdate()
            FROM   inserted,
                   deleted
            WHERE  deleted.RepairID = inserted
.RepairID
      END
```

TRIGGER – PASSWORD LENGTH CHECK

This trigger keep track of the status id which gets updated in the repair work, whenever the status id is updated, a row gets inserted in the **StatusUpdates_Log** table. This can assist in knowing what is the current status of the repair work and when the pothole was closed.

```
CREATE TRIGGER [dbo].[PasswordCheck]
ON [dbo].[User_T]
FOR INSERT
AS
DECLARE @password nvarchar(max)
SELECT @password = inserted.password FROM
inserted

IF( Len(@password) <= 8 )
BEGIN
 PRINT 'Password should be more than 8 chara
cters'
 ROLLBACK
END
```

VIEWS

POTHOLE REPAIR PRIORITY VIEW - FOR JURISDICTION(BUSINESS USER)

	RoadName	NumberOfPotholes	AgeOfRoadinYears	TrafficDensity	TotalVehiclesCrossed	ContractorName
1	MP-SH18	1	33	253	7	Simplex Infrastructures Ltd
2	NH-1	1	30	400	6	Lanco Infratech
3	Carmichael Road	2	63	250	5	Reliance Infrastructure limited
4	Tees January Marg	1	40	245	4	Larsen & Toubro
5	JC Road	1	46	425	3	Larsen & Toubro
6	UP-SH 10	1	29	320	3	Vascon Engineers

Jurisdiction can decide on the basis of this view, which pothole should be given more priority for the repair as here they can see age of road, traffic density and total Vehicle crossed and can also see the performance of the contractor like which contractor roads get potholes frequently

```
CREATE TABLE PotholeRepairPriority_View
  (
     RoadName nvarchar(max) NOT NULL,NumberO
fPotholes int,AgeOfRoadinYears int,
     TrafficDensity int,TotalVehiclesCrossed
 int,ContractorName nvarchar(max)
  )

CREATE PROCEDURE RefreshPotholeRepairPriorit
y_View
AS
    DELETE FROM PotholeRepairPriority_View

    INSERT INTO PotholeRepairPriority_View
    SELECT r.RoadName,Count(DISTINCT p.Potho
leID) AS NumberOfPotholes,
           Datediff(year, r.DateOfConstructi
onCompleted, '11/23/2018')AS
           AgeOfRoadinYears,r.TrafficDensity
, Sum(p.VehiclesCrossed) AS TotalVehiclesCro
ssed,c.ContractorName
    FROM   Road_T r,PotholeSummary_T p,Contr
actor_T c
    WHERE  r.RoadID = p.RoadID
           AND r.ContractorID = c.Contractor
ID
    GROUP  BY r.RoadName,r.DateOfConstructio
nCompleted,r.TrafficDensity,
    c.ContractorName
    ORDER  BY AgeOfRoadinYears DESC,TotalVeh
iclesCrossed DESC

EXECUTE RefreshPotholeRepairPriority_View

SELECT * FROM   PotholeRepairPriority_View
```

QUALITY INSPECTOR PERFORMANCE VIEW - FOR JURISDICTION(BUSINESS USER)

From this view, Jurisdiction can see the performance of their quality inspectors as they can see the number of days since the pothole has been registered and have been put for repair and the status of repair has still changed.

RoadName	PotholeID	Open_SinceinDays	QualityInspectorFirstName	QualityInspectorLastName
TN-SH3	16	105	Kavita	Mohan
MG Road	18	40	Roma	Mathew
NH-12	23	39	Karan	Khatri
NeckLace Road	24	38	Akriti	Krishnamurthy
NH-5	22	36	Krishna	Sharma
GJ-SH2	17	30	Arun	Mishra

```
CREATE TABLE QualityInspectorPerformance_Vie
w (RoadName nvarchar(max),PotholeID bigint,O
pen_SinceinDays int,QualityInspectorFirstNam
e nvarchar(max),QualityInspectorLastName nva
rchar(max))

CREATE PROCEDURE RefreshQualityInspectorPerf
ormance_View
AS
    DELETE FROM QualityInspectorPerformance_
View

    INSERT INTO QualityInspectorPerformance_
View
    SELECT r.RoadName,p.PotholeID,Datediff(d
ay, p.LatestTimestamp,t.RepairStartDate) AS
Open_SinceinDays, q.QFirstName AS QualityIns
pectorFirstName, q.QLastname AS QualityInspe
ctorLastName
    FROM   Road_T r,PotholeSummary_T p,Repai
rWork_T t,QualityInspector_T q
    WHERE  r.RoadID = p.PotholeID
           AND t.PotholeID = p.PotholeID
           AND q.QInspectorID = t.QInspector
ID
           AND t.StatusID = 1
    ORDER  BY Open_SinceinDays DESC;

EXECUTE RefreshQualityInspectorPerformance_V
iew
SELECT * FROM
QualityInspectorPerformance_View
```

POTHOLE REPAIR STATUS VIEW - QUALITY INSPECTOR(BUSINESS USER)

This view is for the Quality Inspector where he can see how many pothole are open, how many are repair in progress and how many have been closed under his Supervision and
How many potholes are still awaiting action, that means they are registered but not put up for repair

	RoadID	RoadName	PotholesRepairAwaited	PotholesOpen	PotholesRepairInProgress	PotholesClosed	QualityInspectorFirstName	QualityInspectorLastI
1	1	NH-1	NULL	NULL	NULL	1	Rohan	Sharma
2	2	Tees January Marg	NULL	NULL	1	NULL	Akash	Verma
3	3	MH-SH 27	NULL	1	NULL	NULL	Arun	Mishra
4	4	NH-24	NULL	1	NULL	NULL	Kavita	Mohan
5	5	MP-SH18	NULL	1	NULL	NULL	Kirti	Sanon

```
CREATE TABLE PotholeRepairStatus_View
  (
     RoadID bigint,RoadName nvarchar(max),Po
tholesRepairAwaited int,
     PotholesOpen int,PotholesRepairInProgre
ss int,PotholesClosed int,  QualityInspector
FirstName nvarchar(max),
QualityInspectorLastName nvarchar(max)
  )

CREATE PROCEDURE RefreshPotholeRepairStatus_
View
AS
    DELETE FROM PotholeRepairStatus_View

    INSERT INTO PotholeRepairStatus_View
    SELECT r.RoadID,r.RoadName,d.NumberOfPot
holesRepairAwaited AS PotholesRepairAwaited,
 a.NumberOfPotholesOpen AS PotholesOpen, b.N
umberOfPotholeRepairInProgress AS PotholeRep
airInProgress, c.NumberOfPotholesClosed AS P
otholesClosed, e.QFirstName AS QualityInspec
torFirstName,e.QlastName AS QualityInspector
LastName
    FROM   Road_T r
    LEFT JOIN (SELECT p.RoadID,Count(t.Statu
sID) AS NumberOfPotholesOpen FROM   PotholeS
ummary_T p,Status_T s,RepairWork_T t WHERE
p.PotholeID = t.PotholeIDAND t.StatusID = s.
StatusID AND t.StatusID = 1 GROUP  BY p.Road
ID) a ON r.RoadID = a.RoadID
    LEFT JOIN (SELECT r.RoadID,Count(t.Statu
sID) AS NumberOfPotholeRepairInProgress
FROM   Road_T r,PotholeSummary_T p,RepairWor
```

```
k_T t WHERE  t.PotholeID = p.PotholeID AND r
.RoadID = p.RoadID AND t.StatusID = 2 GROUP
BY r.RoadID) b ON r.RoadID = b.RoadID
    LEFT JOIN(SELECT r.RoadID,Count(t.Status
ID) AS NumberOfPotholesClosed FROM   Road_T
r,PotholeSummary_T p,RepairWork_T t WHERE  t
.PotholeID = p.PotholeID AND r.RoadID = p.Ro
adID AND t.StatusID = 3 GROUP  BY r.RoadID)
AS c ON r.RoadID = c.RoadID
    LEFT JOIN (SELECT r.RoadID,Count(p.Potho
leID) AS NumberOfPotholesRepairAwaited FROM
 PotholeSummary_T p,Road_T r WHERE  r.RoadID
 = p.RoadID AND p.PotholeID NOT IN (SELECT P
otholeID FROM RepairWork_T)
 GROUP  BY r.RoadID) d
   ON r.RoadID = d.RoadID
    LEFT JOIN (SELECT r.RoadID,QFirstName,QL
astname FROM   QualityInspector_T q,Road_T r
,RepairWork_T t,PotholeSummary_T p WHERE  q.
QInspectorID = t.QInspectorID AND r.RoadID =
 p.RoadID AND t.PotholeID = p.PotholeID) e
   ON r.RoadID = e.RoadID

EXECUTE RefreshPotholeRepairStatus_View
SELECT * FROM PotholeRepairStatus_View
```

QUALITY INSPECTOR SUPERVISION VIEW - QUALITY INSPECTOR(BUSINESS USER)

From this view, The Quality inspector can recognize how seriously the contractor is taking the repair works, as he/she can see Number of workers needed for a repair project and how much the contractor is providing. As less workers means more time to repair the pothole.

RoadID	PotholeID	ContractorName	NumberOfWorkersNeeded	NumberOfWorkersByContractor
1	5	Dilip Buidcon	4	4
2	6	Lanco Infratech	4	5
5	7	Larsen & Toubro	4	4
7	11	Reliance Infrastructure limited	3	7
21	22	Simplex Infrastructures Ltd	5	5

```
CREATE TABLE QualityInspectorSupervision_Vie
w (
     RoadID bigint,PotholeID bigint,Contract
orName nvarchar(max),
     NumberOfWorkersNeeded int,NumberOfWorke
rsByContractor int
  )

CREATE PROCEDURE RefreshQualityInspectorSupe
rvision_View
AS
    DELETE FROM QualityInspectorSupervision_
View

INSERT INTO QualityInspectorSupervision_View
    SELECT p.RoadID,p.PotholeID,c.Contractor
Name,r.NumberOfWorkers AS NumberOfWorkersNee
ded ,Count(DISTINCT w.WorkerID) AS NumberOfW
orkersByContractor
    FROM   RepairWork_T r,RepairWorkers_T w,
PotholeSummary_T p,Contractor_T c
    WHERE  p.PotholeID = r.PotholeID
    AND w.ContractorID = c.ContractorID
    AND w.RepairID = r.RepairID
    GROUP  BY p.RoadID,p.PotholeID,c.Contrac
torName,r.NumberOfWorkers

EXECUTE RefreshQualityInspectorSupervision_V
iew
SELECT * FROM QualityInspectorSupervision_Vi
ew
```