Software Design Document
Road Eye

Faculty of Computer Science - Misr International University

Mohamed Alaa, Mohamed Eleish, Nour El Din tarek, Omar Maysor

Supervised by
Dr Soha A. Ehssan & Eng. Youssef Mobarak

May 14, 2019

1 Introduction

1.1 Purpose

The purpose of this software design document is to provide a full description of the "Road Eye" system architecture, it will describe the functionality of each sub-system in a detailed manner, it will also show the communication between the sub-systems and the interaction between the components in the same sub-system.

1.2 Scope

"Road Eye" aims to develop a deep-learning approach as a mechanism to automate the localization of road anomalies like bumps, potholes, etc, which is a very time and man power consuming process to achieve manually. This approach will make use of crowd-sourcing to achieve both highest and fastest data collection and propagation rate, such data will be used by users through both a web and mobile application to enhance navigation by determining the best and fastest route based on the road quality which can be very helpful for both ordinary users and drivers working for either public or private sectors like Uber and Careem. The system can also be used by the General Authority for Roads to monitor the condition and usability of the roads.

1.3 Overview

It's undeniable that road anomalies like bumps, potholes, poor conditions and unexpected events like accidents and road blocks became very common, those conditions or road anomalies can effect both the safety of the driver and the condition of the vehicle as it can damage suspension systems, tires and overall vehicle condition. "Road Eye" is a project focused on detecting and collecting data about potholes, speed bumps, manholes and rumble-strips through crowd-sourcing by developing an anomaly detection system mounted to vehicles and also developing an analysis system that make use of the collected data to generate useful information and statistics that will be used through a mobile app and a website to enhance navigation by determining the best and fastest route based on the road quality.

1.4 Definitions and Acronyms

- **GSM** Global System for Mobile communications
- **Python** An interpreted high-level programming language for general-purpose programming.
- **OpenCV** (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision.
- **GPS** The Global Positioning System.
- **Raspberry Pi 3 B+** Small single-board computer

2 System Overview

2.1 Overview

In order to have a data collection and detection system that provides consistent data regardless of the vehicle type, a hardware consisted of Raspberry Pi 3 B+, Camera along with both GPS and GSM modules will be used for the purpose of detecting and collecting data categorized as follows, Potholes, Speed Bumps, Manholes, Rumble Strips. A software will be developed using python programming language, openCV and Deep Neural Networks to work with this set of hardware to develop an embedded system installed to the driver's vehicle to serve as a data collection system for both the mobile app and the website available to the public, they shall serve as a portal for users to be able to determine the best route based on the collected data and will also allow the preview of such data..

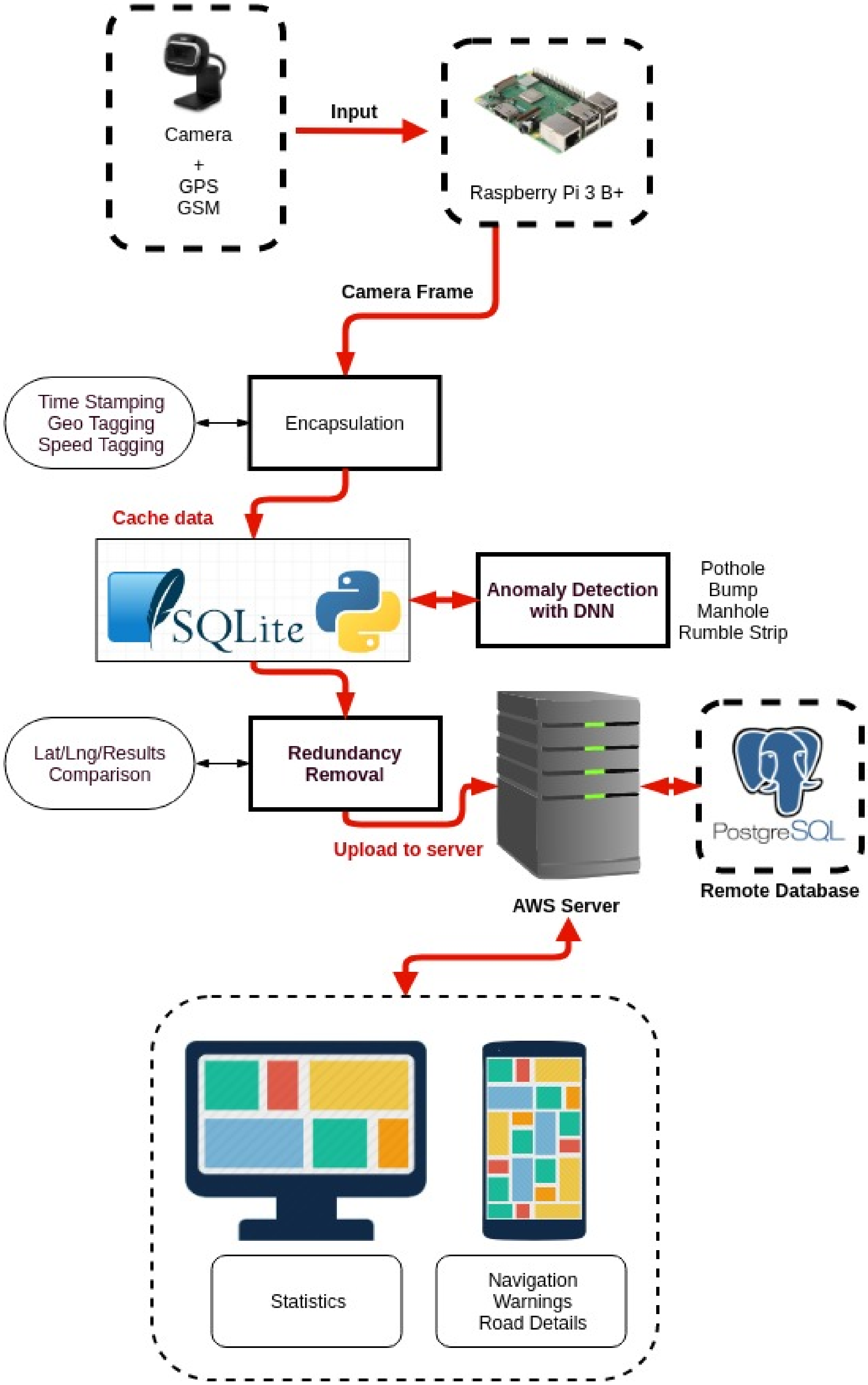


Figure 1: System overview

The system composes of 3 main stages as follows:

2.1.1 Data Acquisition

The hardware consists of camera, GPS, and a raspberry pi 3 B+. This set of hardware is mounted to vehicles to collect data. The hardware is responsible for the acquisition of camera frames and GPS data through the corresponding hardware.

2.1.2 Data Processing

The frames from camera are then encapsulated with time stamp, geo-location, and speed value, then cached to a SqlLite local database on the raspberry pi in order to be processed by another thread later when the thread is not busy. When an anomaly is detected it is also saved to the local database and then uploaded to the server after redundancy removal process has been executed on the batch to be uploaded.

2.1.3 Data Presentation

The website displays useful road statistics and condition to users, as how many anomalies are detected in a road or route along with their details, it also provided the overall condition of a single road or a route (multiple roads). For the mobile application the user can search for a road, enter destination and get all possible routes to the destination, how many anomalies in the road and it's condition. During navigation the user will also receive real-time warnings for anomalies in his way.

3 System Architecture

3.1 Architectural Design

3.1.1 Vehicle System

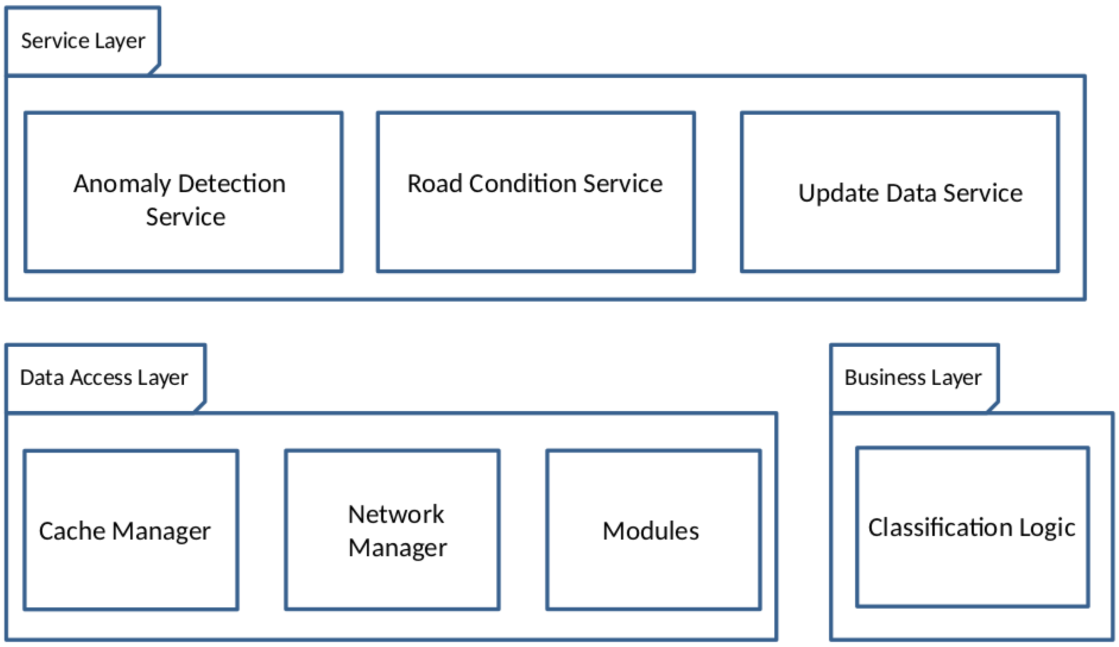


Figure 2: Layered architecture for the vehicle module

Our layered architecture approach consists of three layers. Application logic (service) layer, business logic layer and data logic (data access) layer.

Service layer is responsible for services of anomaly detection, road condition detection and upload data but no actual business login which happens at business logic layer.

Our business logic here consists of classifying data. Moreover, data access layer is responsible for fetching data from modules, caching and uploading data to the server.

Service layer is at the highest level beneath it exists business logic layer and data access layer so only service layer can communicate with the other layers and the other two layers can't communicate with service layer or communicate with each other.

In previously mentioned layers the components of each layer also is not allowed to communication with each other.

Such separation of concerns was intended to ensure scalability with minimum changes to the base code.

3.1.2 Mobile App

Event Driven Architecture was chosen as the main architecture of the mobile app to reduce the coupling between the components of the system and ensure that all the components present at the same time (ex: multiple fragments) are all in the correct state and populated with the correct data. Events are published and propagates through the app components using the EventBus and each components that is listening for such event acts as specified.

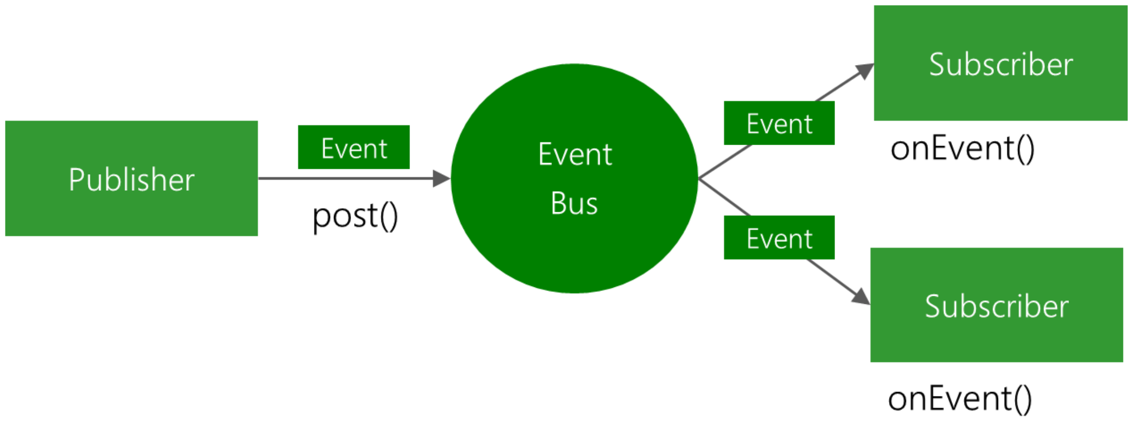


Figure 3: Event Driven architecture for the Mobile App

3.1.3 Web App

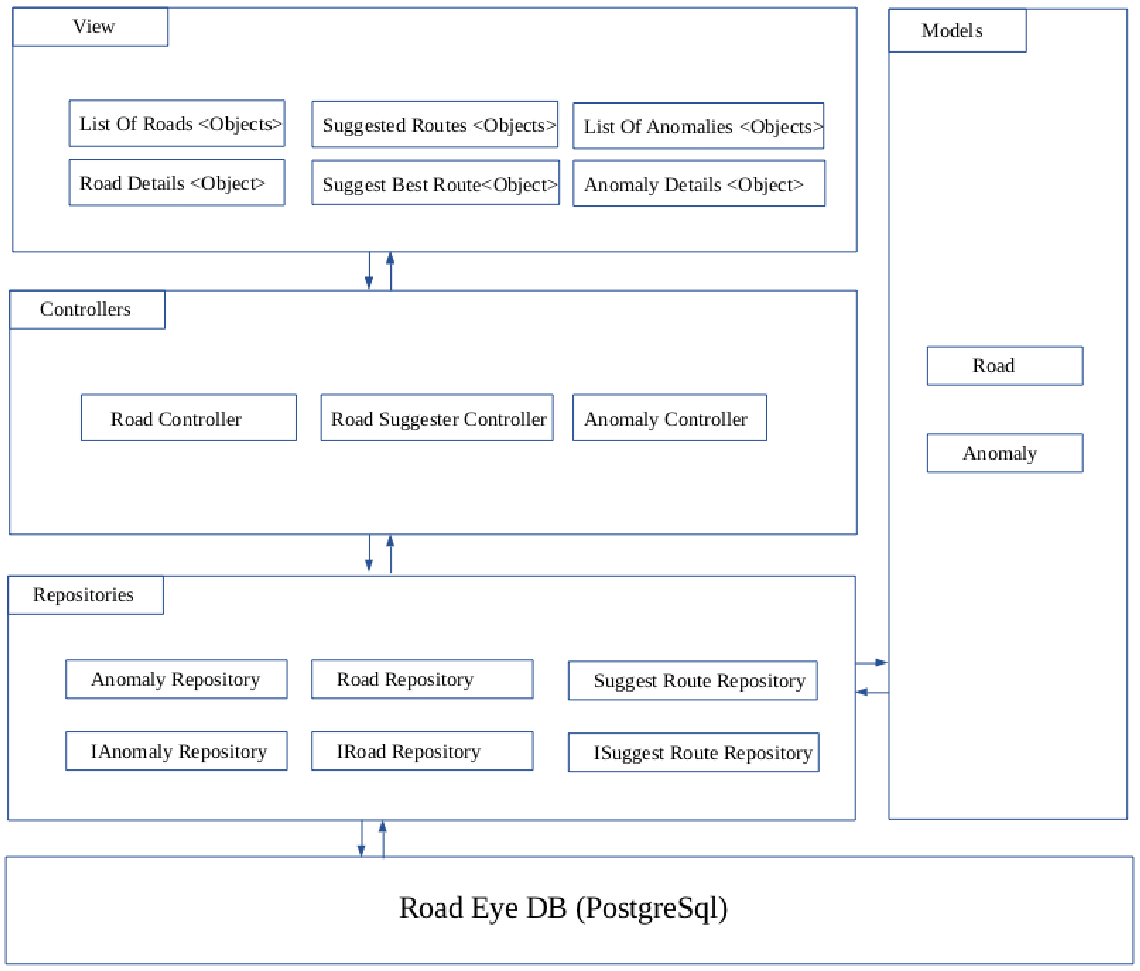


Figure 4: MVC architecture for the web App

3.2 Decomposition Description

3.2.1 Class Diagram

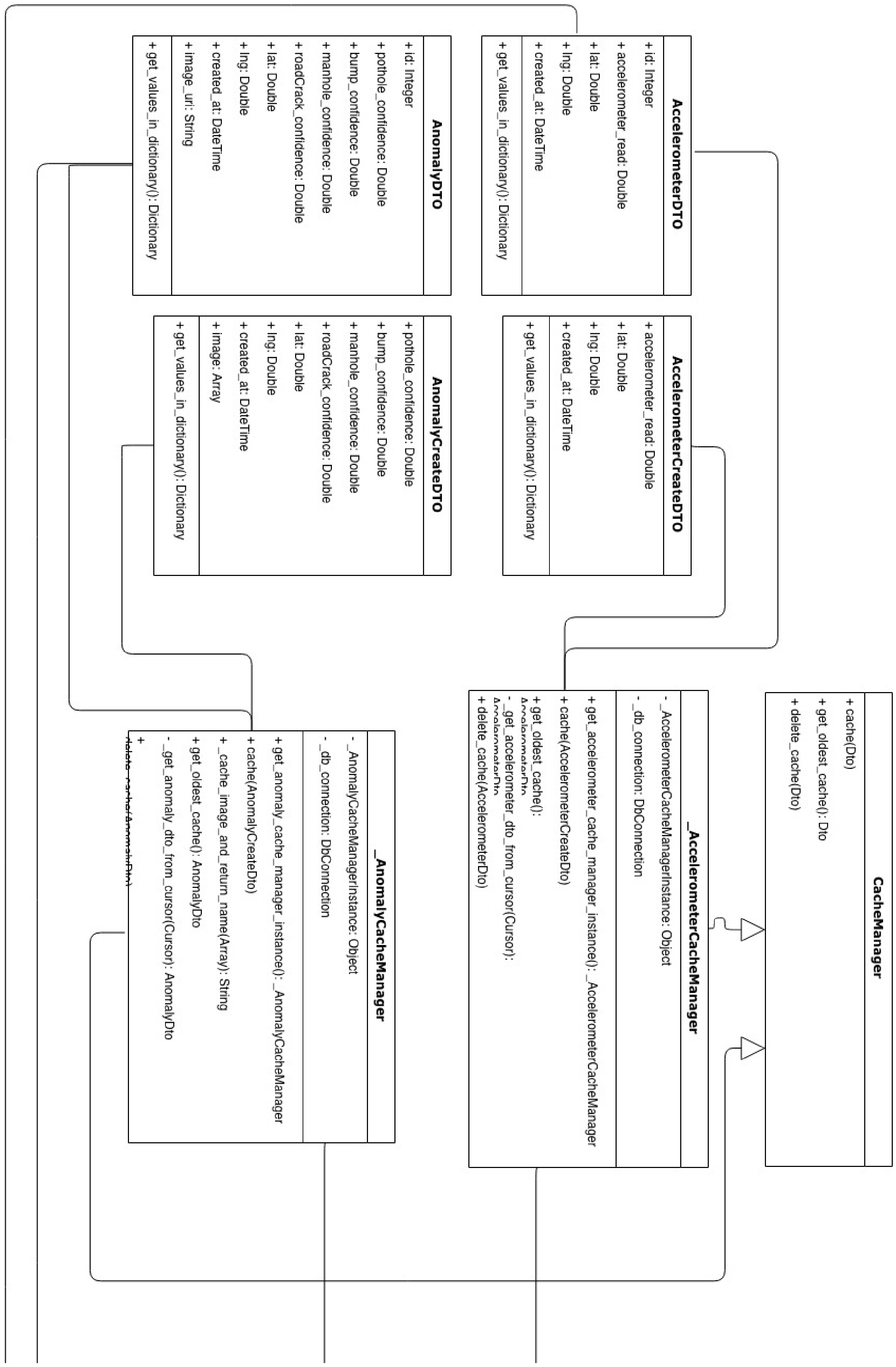

AccelerometerDTO
+ id: Integer
+ accelerometer_read: Double
+ lat: Double
+ lng: Double
+ created_at: DateTime
+ get_values_in_dictionary(): Dictionary
AnomalyDTO
+ id: Integer
+ pothole_confidence: Double
+ bump_confidence: Double
+ manhole_confidence: Double
+ roadCrack_confidence: Double
+ lat: Double
+ lng: Double
+ created_at: DateTime
+ image_uri: String
+ get_values_in_dictionary(): Dictionary
AccelerometerCreateDTO
+ accelerometer_read: Double
+ lat: Double
+ lng: Double
+ created_at: DateTime
+ get_values_in_dictionary(): Dictionary
AnomalyCreateDTO
+ pothole_confidence: Double
+ bump_confidence: Double
+ manhole_confidence: Double
+ roadCrack_confidence: Double
+ lat: Double
+ lng: Double
+ created_at: DateTime
+ image: Array
+ get_values_in_dictionary(): Dictionary
CacheManager
+ cache(Dto)
+ get_oldest_cache(): Dto
+ delete_cache(Dto)
_AccelerometerCacheManager
- _AccelerometerCacheManagerInstance: Object
- _db_connection: DbConnection
+ get_accelerometer_cache_manager_instance(): _AccelerometerCacheManager
+ cache(AccelerometerCreateDto)
+ get_oldest_cache(): AccelerometerDto
- _get_accelerometer_dto_from_cursor(Cursor): AccelerometerDto
+ delete_cache(AccelerometerDto)
_AnomalyCacheManager
- _AnomalyCacheManagerInstance: Object
- _db_connection: DbConnection
+ get_anomaly_cache_manager_instance(): _AnomalyCacheManager
+ cache(AnomalyCreateDto)
+ _cache_image_and_return_name(Array): String
+ get_oldest_cache(): AnomalyDto
- _get_anomaly_dto_from_cursor(Cursor): AnomalyDto
+ delete_cache(AnomalyDto)

Figure 5: Vehicle module class diagram - part 1

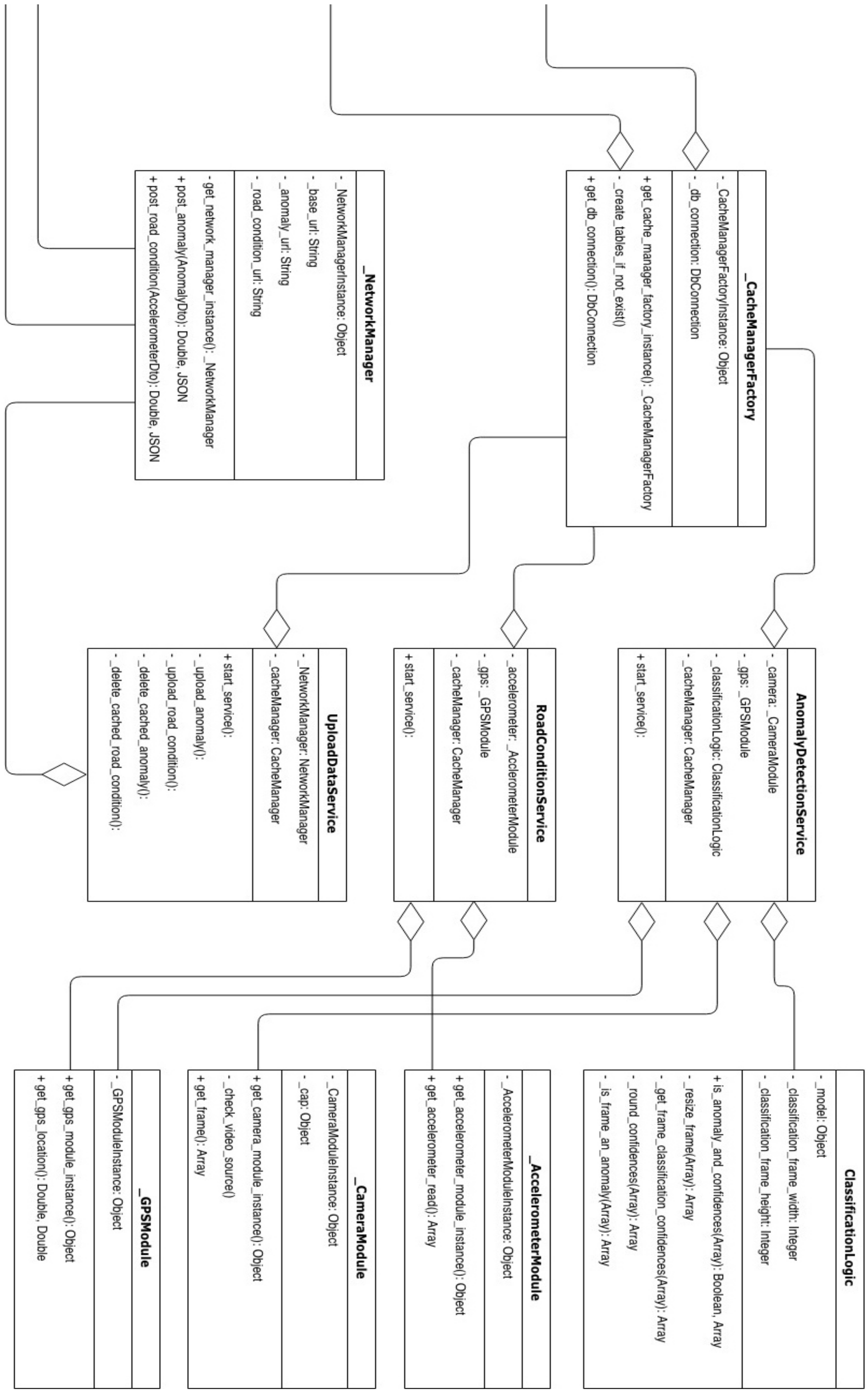

_NetworkManager
- _NetworkManagerInstance: Object
- base_url: String
- _anomaly_url: String
- _road_condition_url: String
- get_network_manager_instance(): _NetworkManager
+ post_anomaly(AnomalyDto): Double, JSON
+ post_road_condition(AccelerometerDto): Double, JSON
_CacheManagerFactory
- _CacheManagerFactoryInstance: Object
- _db_connection: DbConnection
+ get_cache_manager_factory_instance(): _CacheManagerFactory
- _create_tables_if_not_exist()
+ get_db_connection(): DbConnection
UploadDataService
- _NetworkManager: NetworkManager
- _cacheManager: CacheManager
+ start_service():
- _upload_anomaly():
- _upload_road_condition():
- _delete_cached_anomaly():
- _delete_cached_road_condition():
RoadConditionService
- _accelerometer: _AccelerometerModule
- _gps: _GPSModule
- _cacheManager: CacheManager
+ start_service():
AnomalyDetectionService
- _camera: _CameraModule
- _gps: _GPSModule
- _classificationLogic: ClassificationLogic
- _cacheManager: CacheManager
+ start_service():
_GPSModule
- _GPSModuleInstance: Object
+ get_gps_module_instance(): Object
+ get_gps_location(): Double, Double
_CameraModule
- _CameraModuleInstance: Object
- _cap: Object
+ get_camera_module_instance(): Object
- _check_video_source()
+ get_frame(): Array
_AccelerometerModule
- _AccelerometerModuleInstance: Object
+ get_accelerometer_module_instance(): Object
+ get_accelerometer_read(): Array
ClassificationLogic
- _model: Object
- _classification_frame_width: Integer
- _classification_frame_height: Integer
+ is_anomaly_and_confidences(Array): Boolean, Array
- _resize_frame(Array): Array
- _get_frame_classification_confidences(Array): Array
- _round_confidences(Array): Array
- _is_frame_an_anomaly(Array): Array

Figure 6: Vehicle module class diagram - part 2

3.2.2 Sequence Diagram

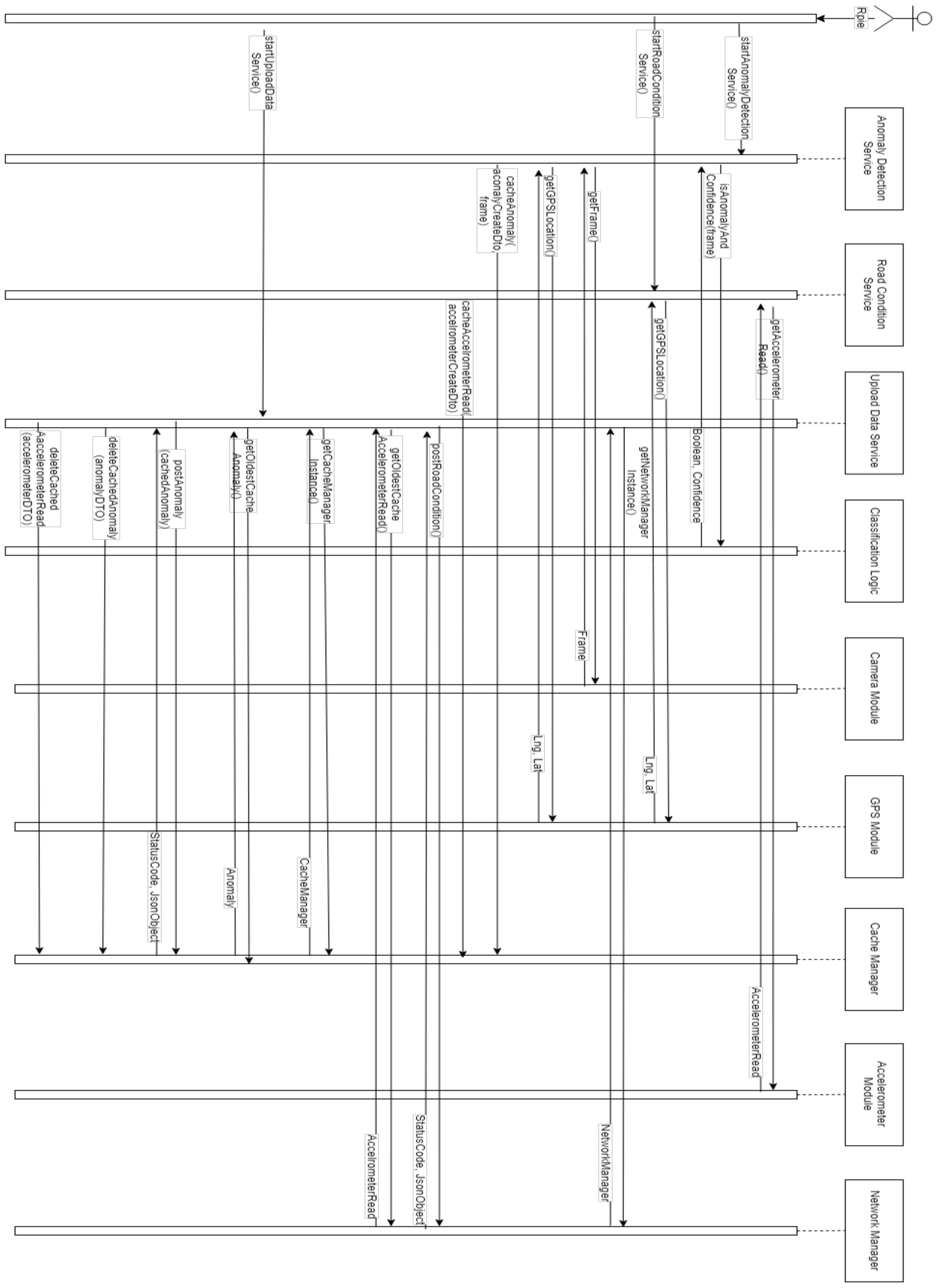


Figure 7: Sequence diagram showing the sequence of the anomalies detection system running on the raspberry pi

3.3 Design Rationale

3.3.1 Vehicle System

Considering the fact that the logic can be separated clearly into three main parts, application logic, business logic and data logic. So in the light of the previous conclusion layered architecture approach has been chosen.

3.3.2 Mobile App

The mobile app sub-system is based on Model-View-ViewModel (MVVM) design pattern along with event-driven approach, ViewModel backs up and simplify the event driven approach as it exposes streams of events to which the Views can bind to. Like this, the ViewModel does not need to hold a reference to the View anymore.

3.3.3 Web App

It has become a standard to use MVC as the architecture for the web development in backend and also lately in the frontend (main frontend frameworks tend to use MVVM instead of MVC) and that's due to the different types of views, JSON in case of API services or web pages in case of server rendered web sites, also the controllers is quite suitable for backend development as each controller represents an end point with repositories as an extension to the controllers to hold the business logic.

4 Data Design

4.1 Data Description

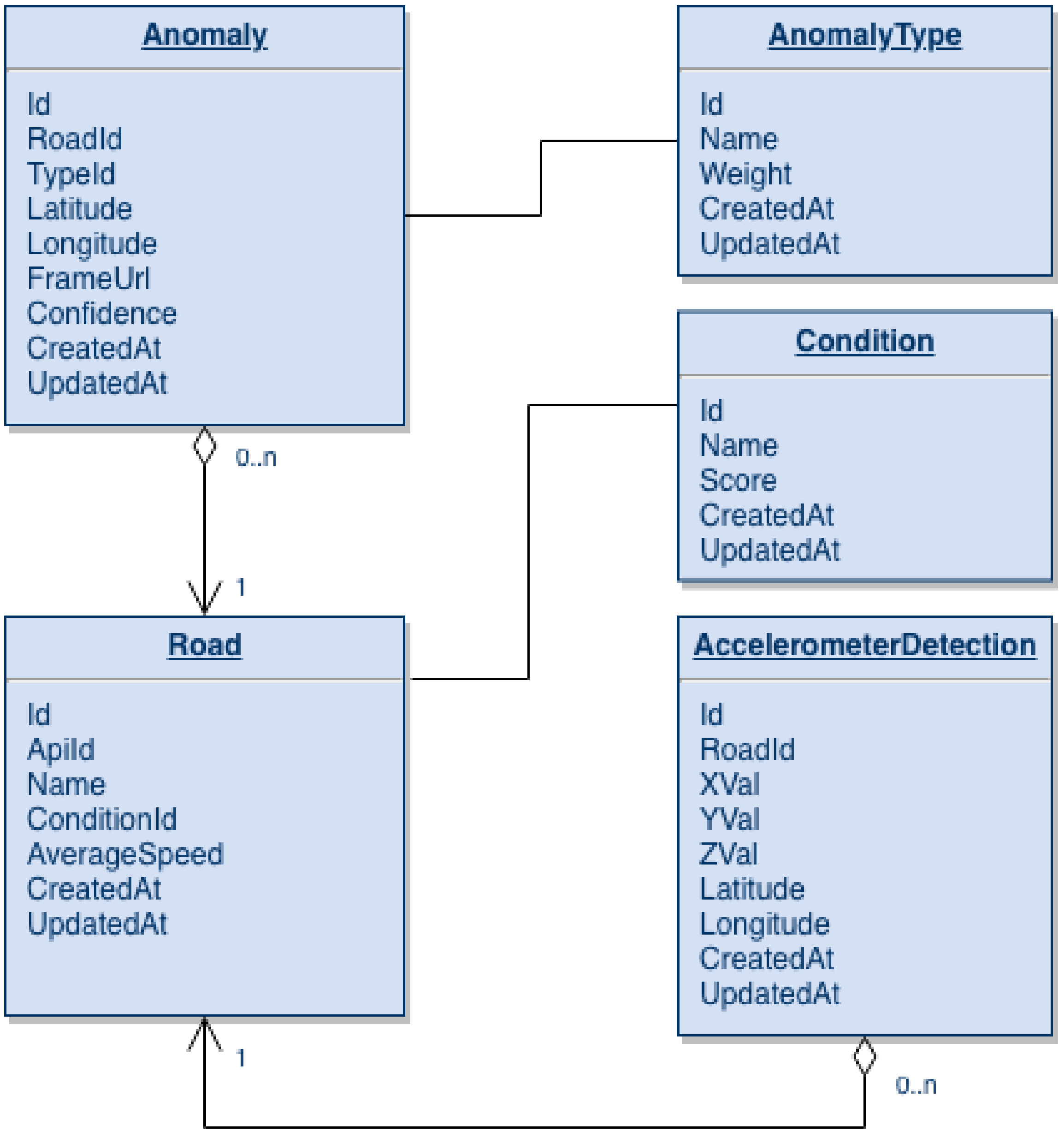


Figure 8: Database schema used

4.2 Data Dictionary

4.2.1 Anomaly

- **RoadID** refers to a specific road in table Road where the anomaly is located.
- **TypeId** refers to the anomaly type in AnomalyType table (ex: id = 1 means it is a Pothole).
- **Latitude** latitude coordinate of the anomaly.
- **Longitude** longitude coordinate of the anomaly.
- **FrameUrl** url of the captured frame containing the anomaly.
- **Confidence** confidence of the classification model for this detection.

4.2.2 AnomalyType

- **Name** describe type of the anomaly (ex: Pothole)
- **Weight** represents the impact of the corresponding anomaly type on the road condition.

4.2.3 Road

- **ApiId** corresponds to the actual place id in the Google Places API.
- **Name** name of the road/
- **ConditionId** refers to the road condition in Condition table (ex: id = 1 means it is in a Very Good condition).

4.2.4 Condition

- **Name** describes the condition (ex: Very Good)
- **Score** represents the target score that must be met by a specific road to be able to refer to this condition or tier.

5 Component Design

5.1 Inception V4 Network

Deep convolutional nueral networks have been central to the largest advances in image recognition performance in recent years. One example is the Inception architecture that has been shown to achieve very good performance at relatively low computational cost.

Recently deep convolutional neural networks have been the largest advances in image recognition and object detection fields. For Instance, the Inception architecture which was introduced in [3] and in its early stages was called GoogleNet and there is 4 implemented versions that has been shown a steadily and remarkable pace at achieving very good performance at relatively low computational cost.

Inception-V1 was the initiative. Later on Inception-V2 introduced the architecture with batch normalization added after each convolutional layer. Finally, in the final 2 version there was improvements such as adding additional factorization and make it more tun-able, meaning that there are a lot of possible changes to the number of filters in the various layers that do not affect the quality of the architecture. In order to optimize the training speed, layer sizes was tuned carefully in order to balance the computation between the various model sub-networks that will be explained later on.

For training very deep architectures it is argued that residual connections are of inherent importance. Since Inception networks tend to be very deep, it is natural to replace the filter concatenation stage of the Inception architecture with residual connections. This would allow Inception to use all the benefits of the residual approach while retaining its computational efficiency.

5.2 SeNet

Residual connection were introduced for giving the advantages of utilizing additive merging of signals both for image recognition, and especially for object detection. However, one of the most important findings in the Inception architecture that residual connections are not by default necessary for training very deep convolutional models. Moreover, there is an additional component that helps improving the quality of representations produced by a network by explicitly modeling the inter dependencies between the channels of its convolutional features, which is the Squeeze-and-Excitation (SE) block,this mechanism allows the network to perform feature re-calibration, through which it can learn to use global information to selectively emphasis informative features and suppress less useful ones.

As for the InceptionV4-SEnet architecture it will be explained in the following figures. Bear in mind that all the convolutions not marked with “V” in the figures are same-padded meaning that their output grid matches the size of their input. Convolutions marked with “V” are valid padded, meaning that input patch of each unit is fully contained in the previous layer and the grid size of the output activation map is reduced accordingly.

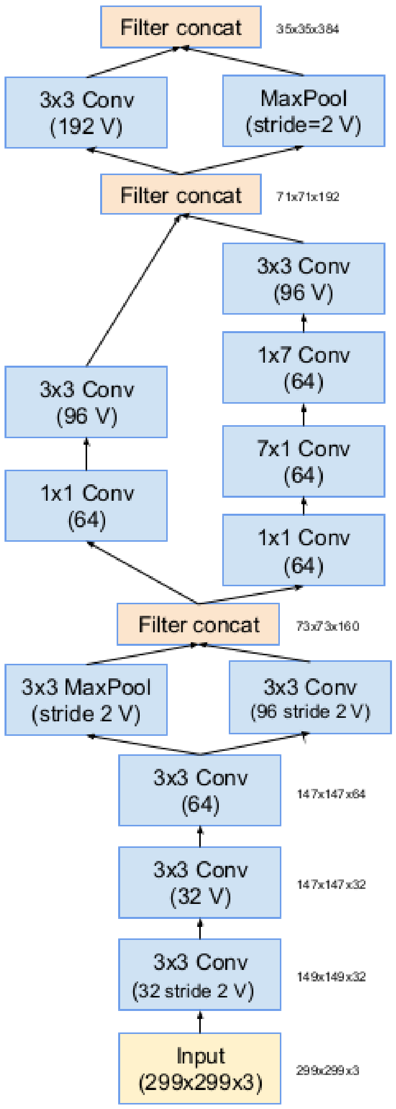


Figure 9: The schema for stem of the pure Inception-v4. This is the input part of those network.

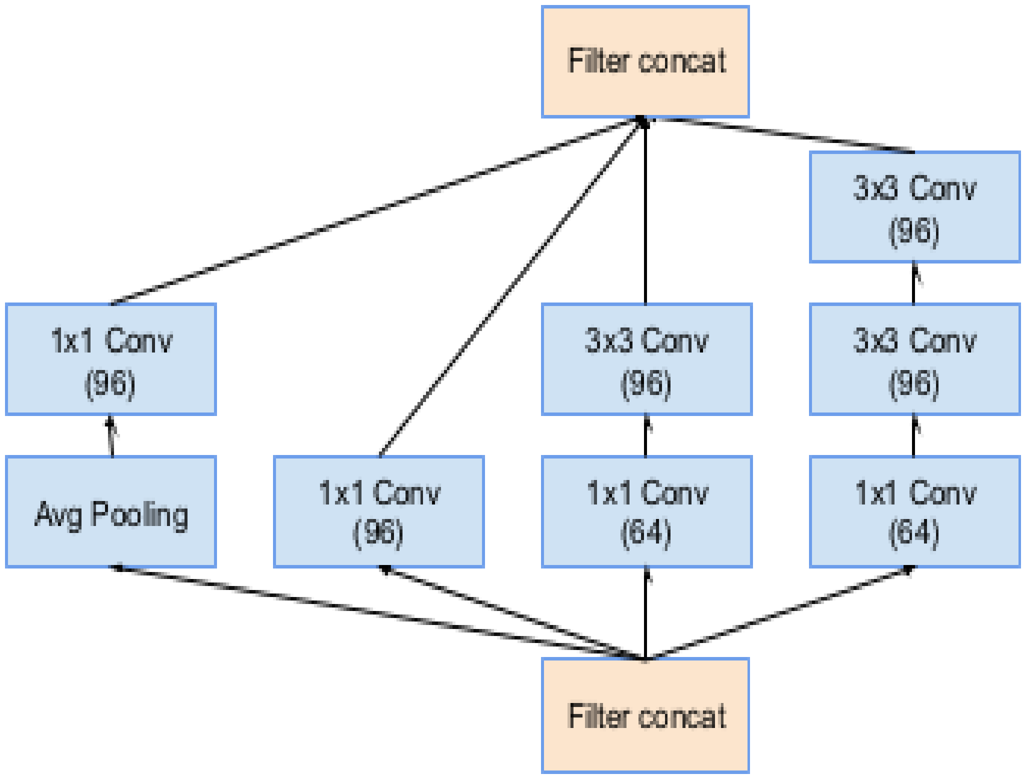


Figure 10: The schema for 35 x 35 grid modules of the pure Inception-v4 network. This is the Inception-A block.

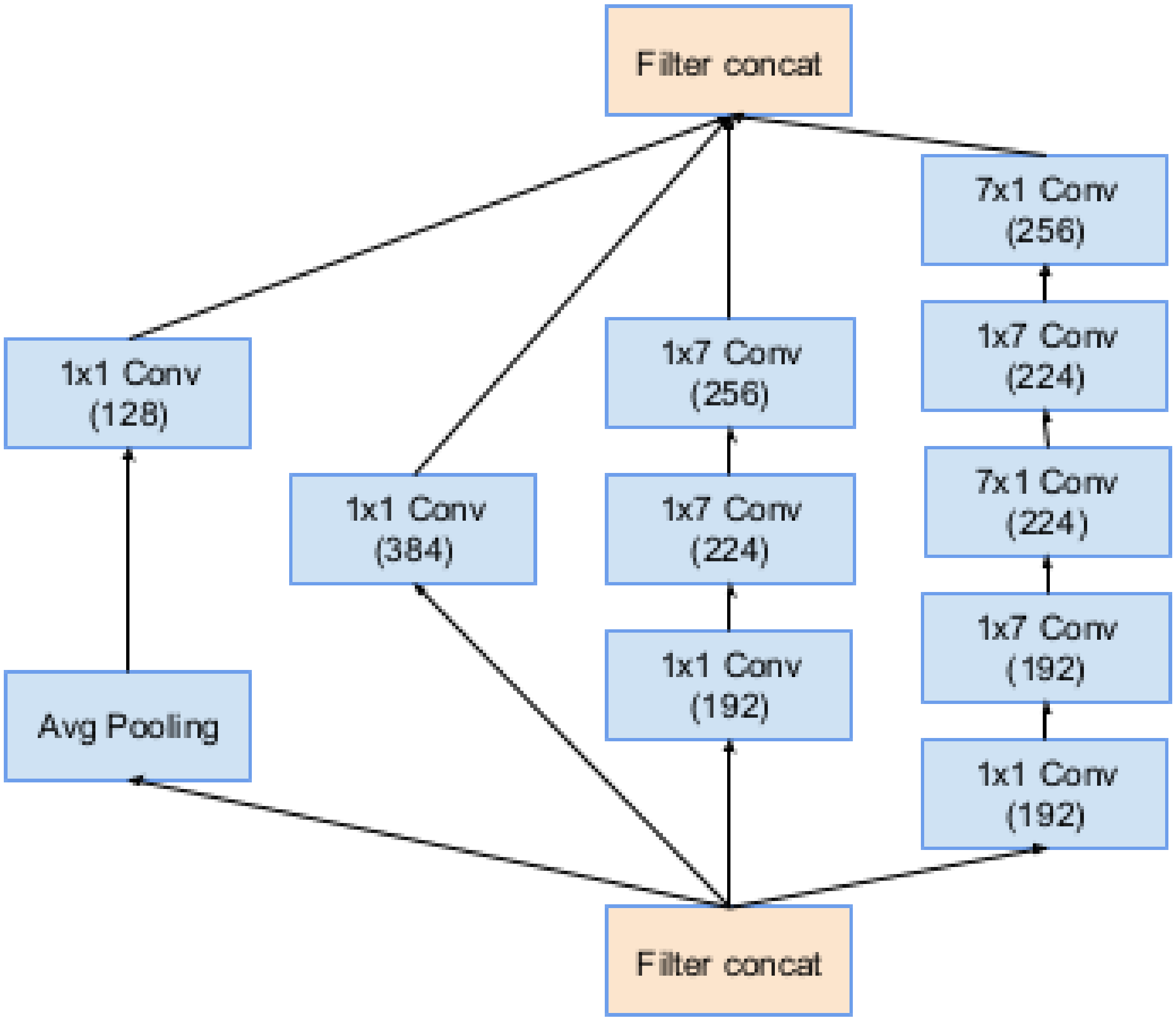


Figure 11: The schema for 17 x 17 grid modules of the pure Inception-v4 network. This is the Inception-B block.

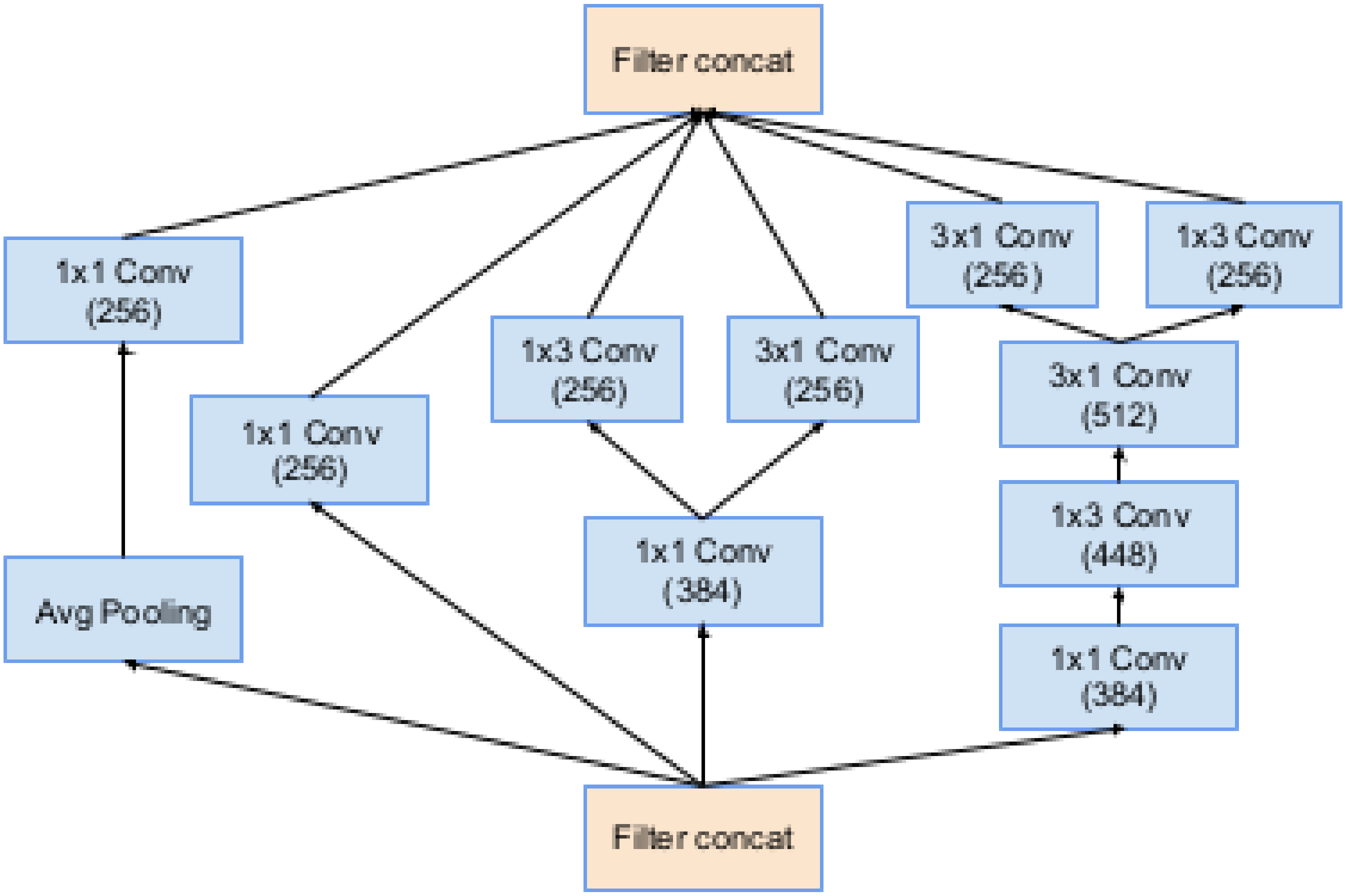


Figure 12: The schema for 8 x 8 grid modules of the pure Inception-v4 network. This is the Inception-C block.

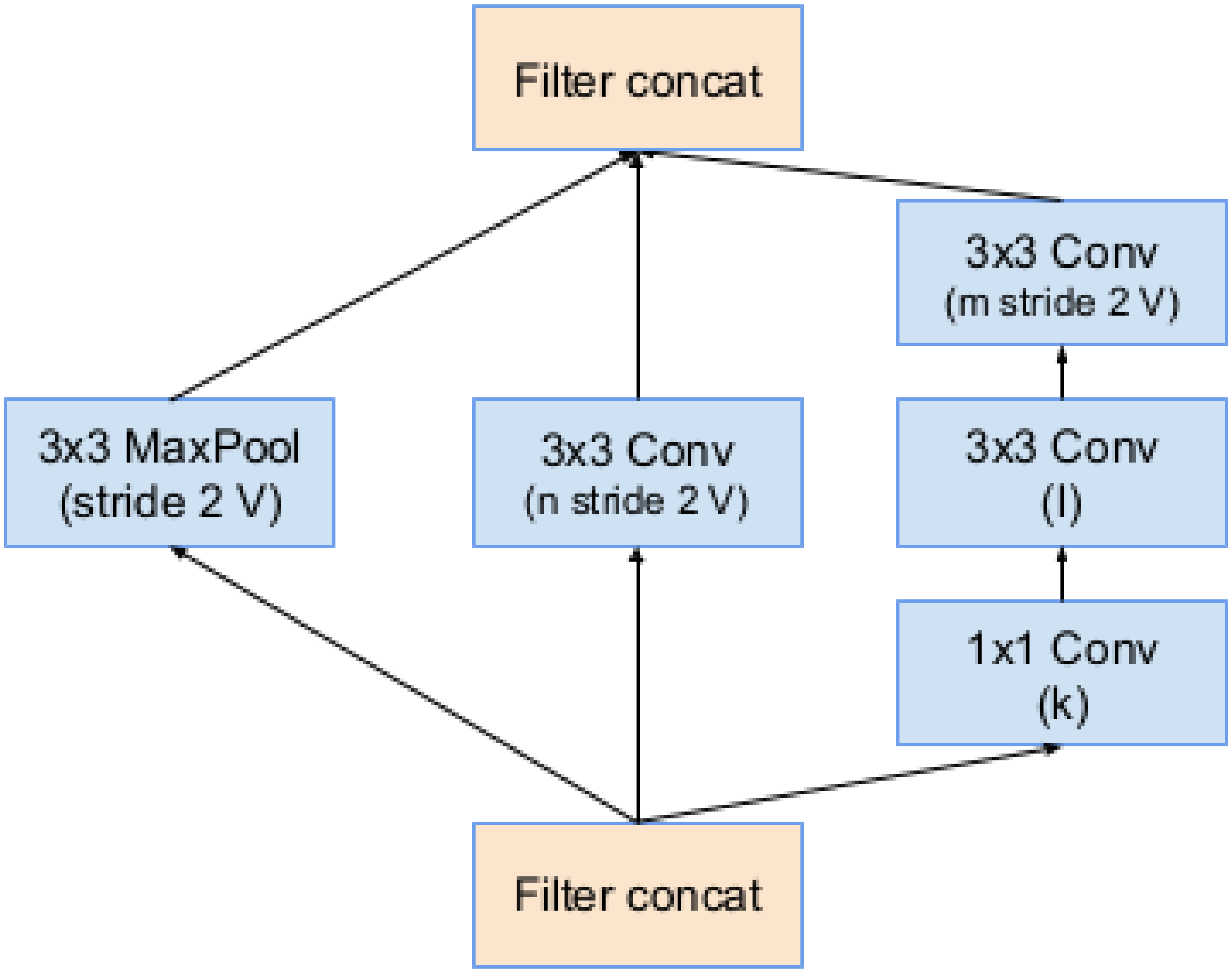


Figure 13: The schema for 35 x 35 to 17 x 17 Grid reduction A module.

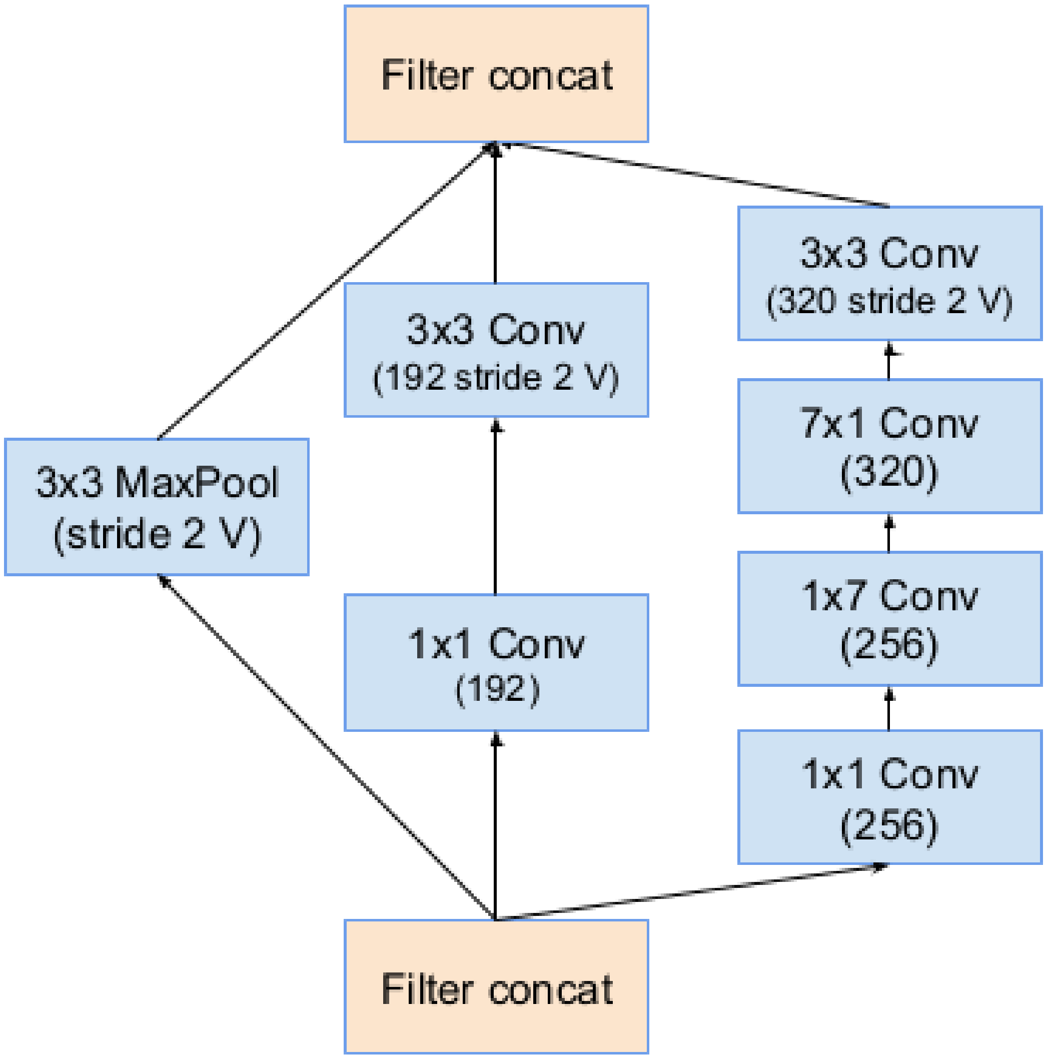


Figure 14: The schema for 17 x 17 to 8 x 8 Grid-reduction B module.

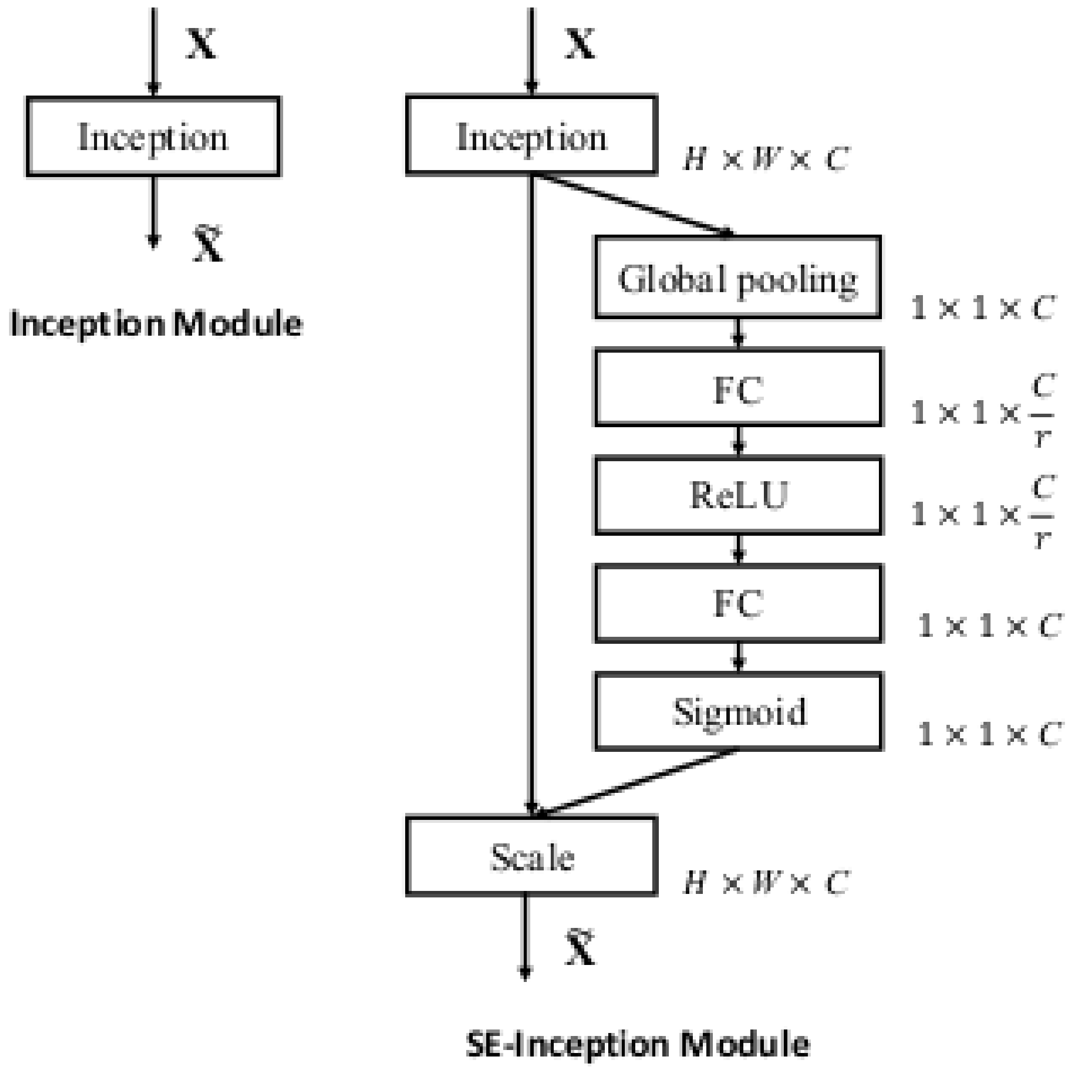


Figure 15: The schema of the original Inception module (left) and the SE-Inception module (right)

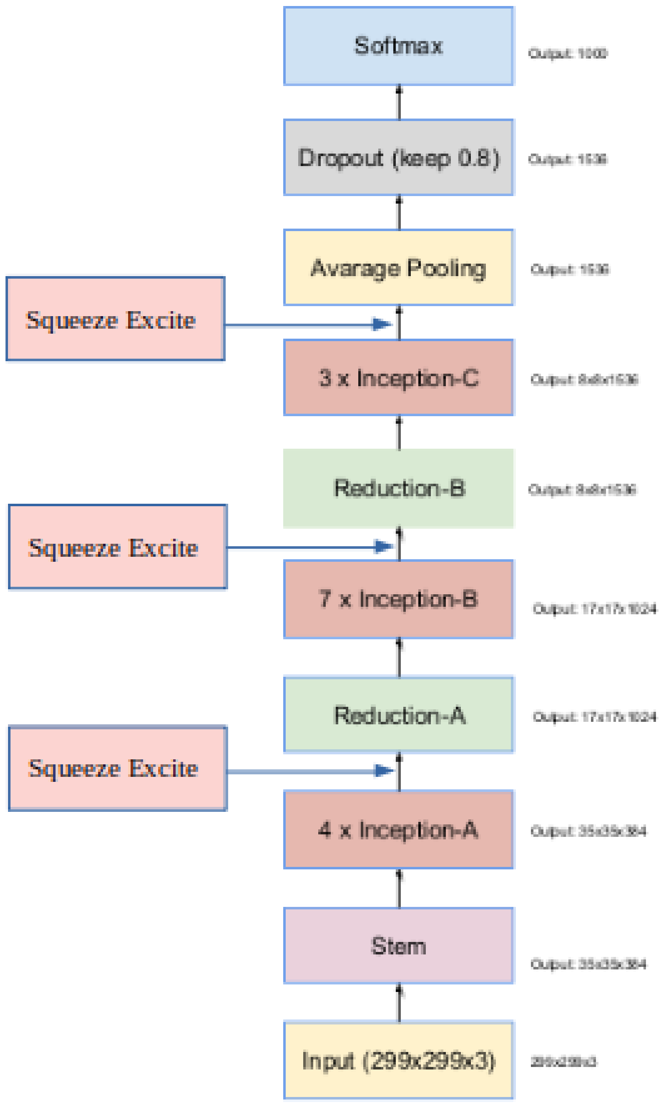


Figure 16: The overall schema of the Inception-v4 network. For the detailed modules

6 Human Interface Design

6.1 Overview of User Interface

RoadEye vehicle module detection process is automated and require no user interaction therefor no user interface or communication methods are needed between the user and the vehicle module, the data collected by this module is used through the mobile and web apps through the user interface shown and explained in the next two sections.

6.2 Screen Images

6.2.1 Mobile App

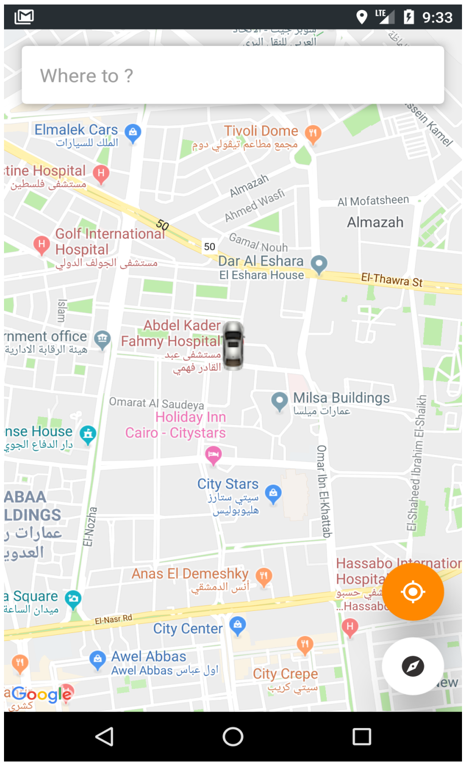
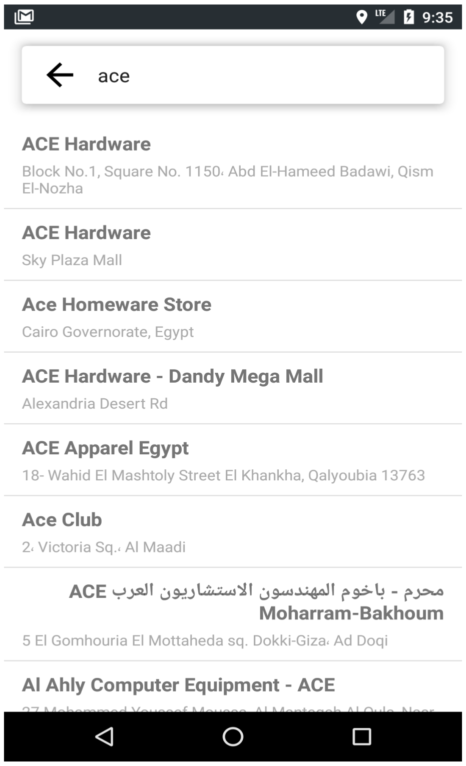


Figure 17: Home Idle



9:35

ace

ACE Hardware
Block No.1, Square No. 1150، Abd El-Hameed Badawi, Qism El-Nozha

ACE Hardware
Sky Plaza Mall

Ace Homeware Store
Cairo Governorate, Egypt

ACE Hardware - Dandy Mega Mall
Alexandria Desert Rd

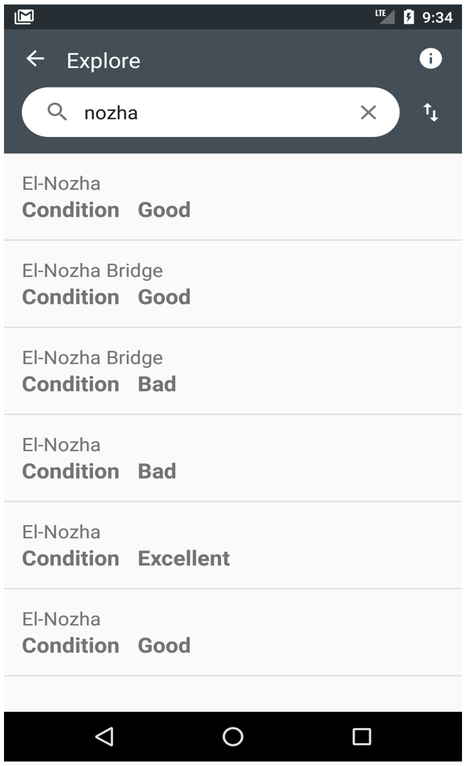
ACE Apparel Egypt
18- Wahid El Mashtoly Street El Khankha, Qalyoubia 13763

Ace Club
2، Victoria Sq.، Al Maadi

محرم - باخوم المهندسون الاستشاريون العرب ACE Moharram-Bakhoum
5 El Gomhouria El Mottaheda sq. Dokki-Giza، Ad Doqi

Al Ahly Computer Equipment - ACE
27 Mohammed Youssef Moussa, Al Mantegah Al Oula, Nasr

Figure 18: Search


Explore

nozha

El-Nozha
Condition Good

El-Nozha Bridge
Condition Good

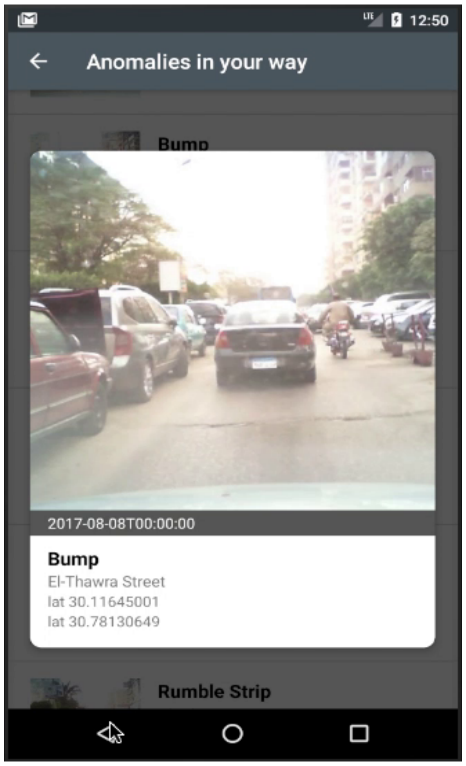
El-Nozha Bridge
Condition Bad

El-Nozha
Condition Bad

El-Nozha
Condition Excellent

El-Nozha
Condition Good

Figure 19: Search and explore Roads


Anomalies in your way

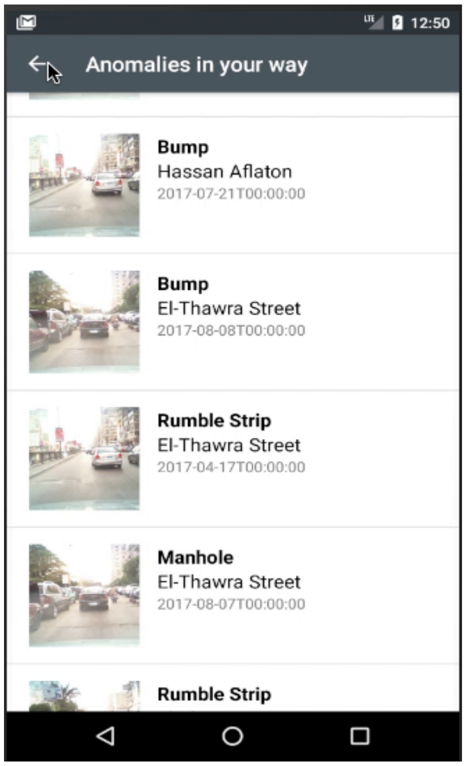
Bump

2017-08-08T00:00:00

Bump
El-Thawra Street
lat 30.11645001
lat 30.78130649

Rumble Strip

Figure 20: Anomaly details



Anomalies in your way

Bump
Hassan Aflaton
2017-07-21T00:00:00

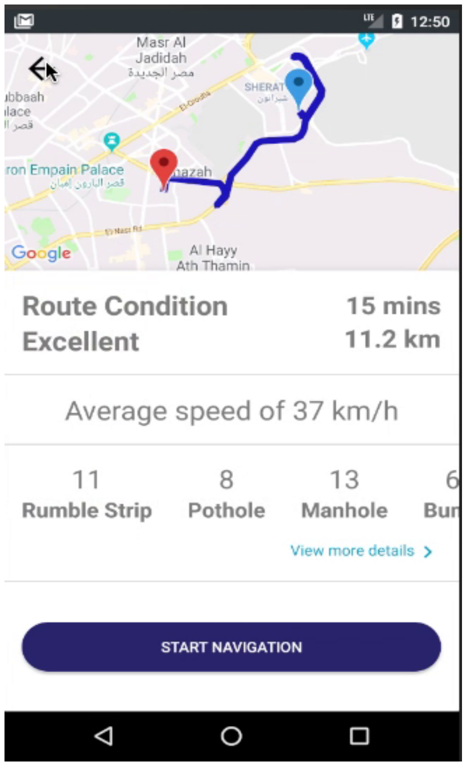
Bump
El-Thawra Street
2017-08-08T00:00:00

Rumble Strip
El-Thawra Street
2017-04-17T00:00:00

Manhole
El-Thawra Street
2017-08-07T00:00:00

Rumble Strip

Figure 21: Anomalies in route



Route Condition
Excellent

15 mins
11.2 km

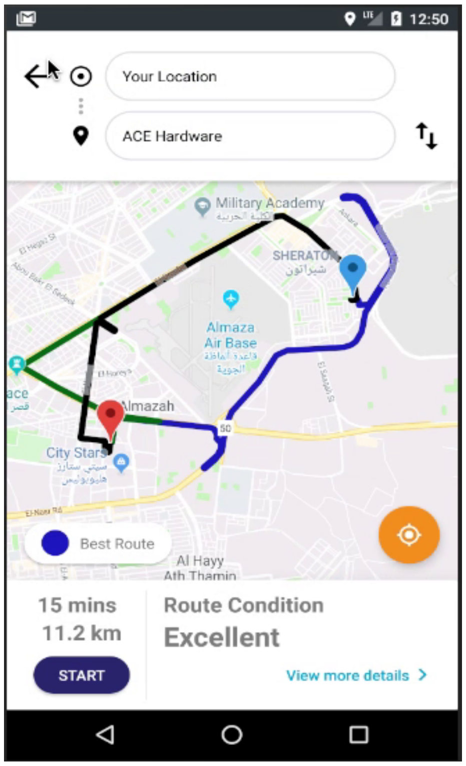
Average speed of 37 km/h

11 Rumble Strip
8 Pothole
13 Manhole
6 Bun

View more details

START NAVIGATION

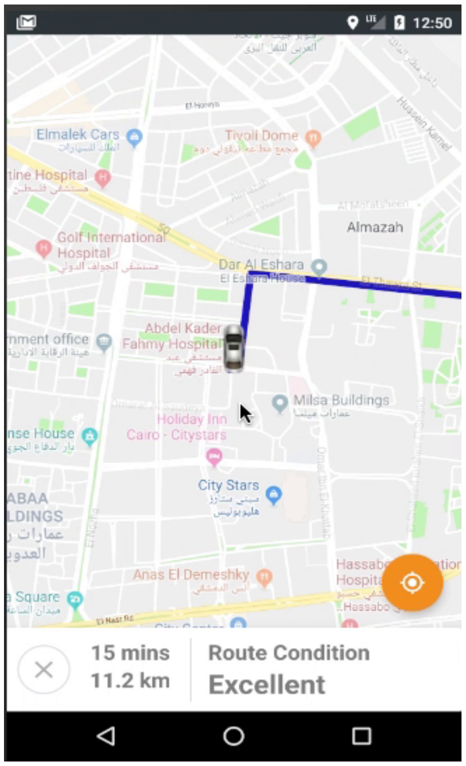
Figure 22: Route details



15 mins
11.2 km

Route Condition
Excellent

Figure 23: Routes preview

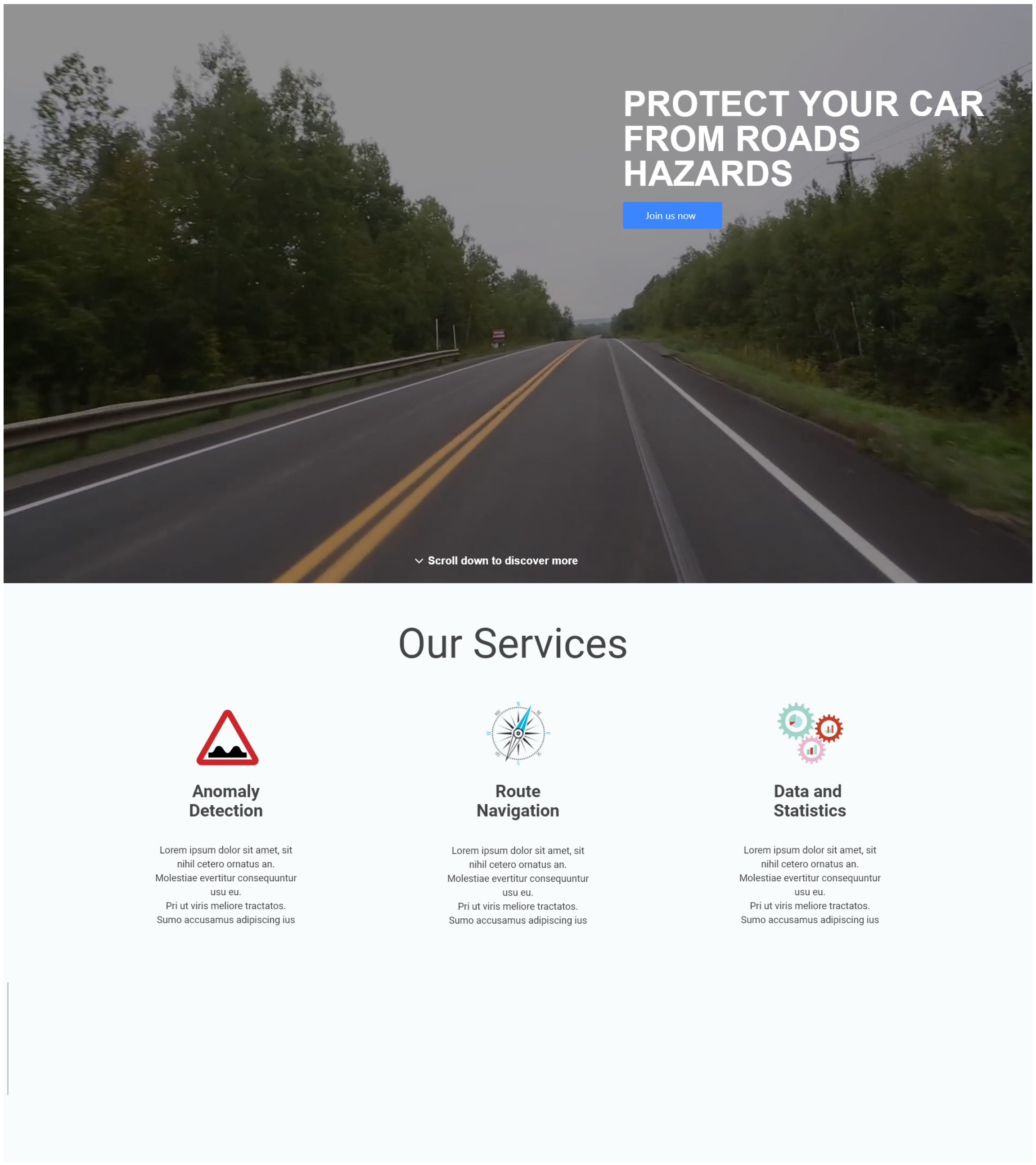


15 mins
11.2 km

Route Condition
Excellent

Figure 24: Navigation

6.2.2 Web App



Lorem ipsum dolor sit amet, sit nihil cetero ornatus an. Molestiae evertitur consequuntur usu eu. Pri ut viris meliore tractatos. Sumo accusamus adipiscing ius

Lorem ipsum dolor sit amet, sit nihil cetero ornatus an. Molestiae evertitur consequuntur usu eu. Pri ut viris meliore tractatos. Sumo accusamus adipiscing ius

Lorem ipsum dolor sit amet, sit nihil cetero ornatus an. Molestiae evertitur consequuntur usu eu. Pri ut viris meliore tractatos. Sumo accusamus adipiscing ius

Figure 25: Landing Page

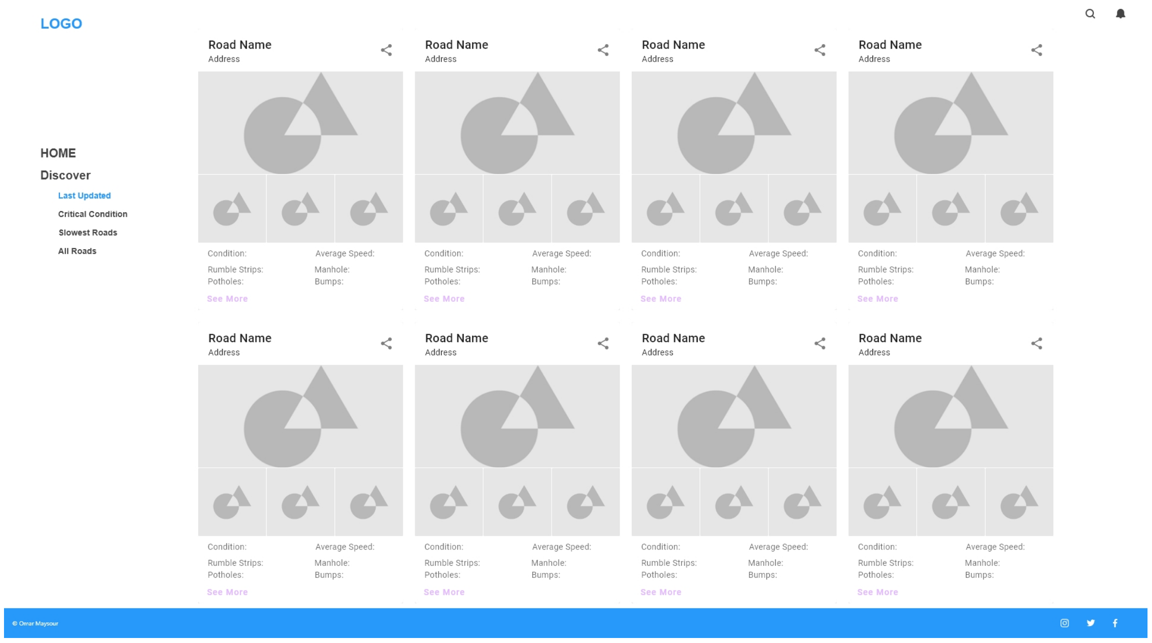


Figure 26: Latest Updated Roads

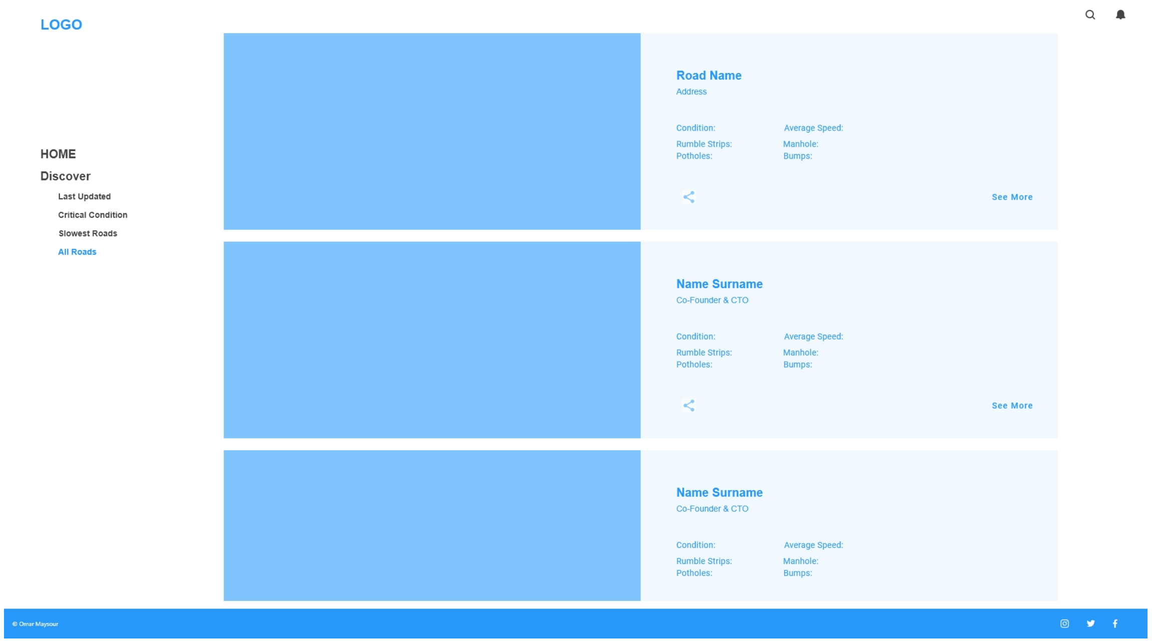


Figure 27: All Roads

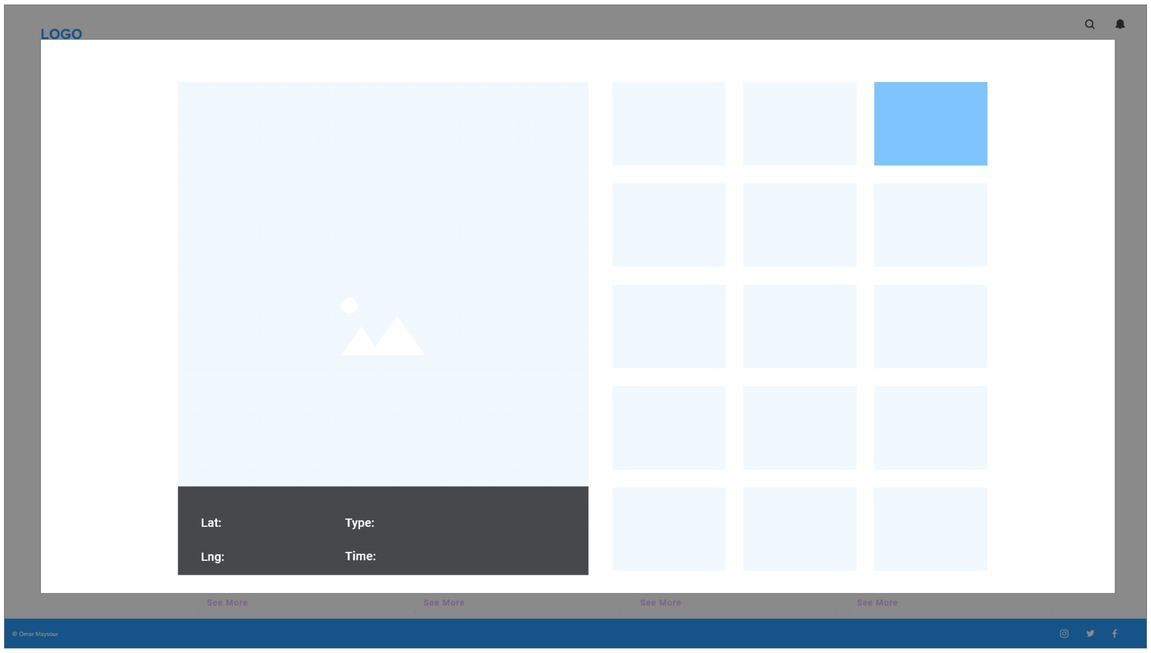
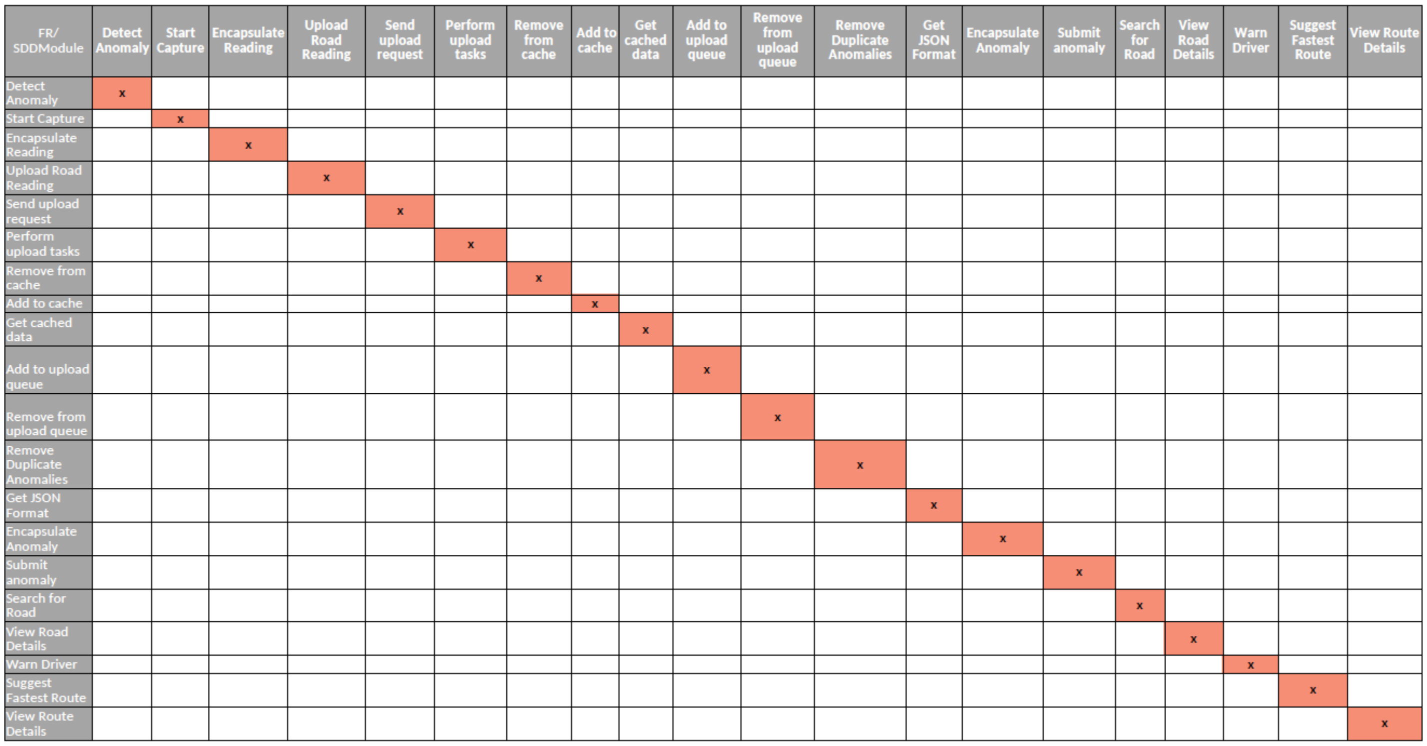


Figure 28: Anomalies details

7 Requirements Matrix



FR/ SDDModule	Detect Anomaly	Start Capture	Encapsulate Reading	Upload Road Reading	Send upload request	Perform upload tasks	Remove from cache	Add to cache	Get cached data	Add to upload queue	Remove from upload queue	Remove Duplicate Anomalies	Get JSON Format	Encapsulate Anomaly	Submit anomaly	Search for Road	View Road Details	Warn Driver	Suggest Fastest Route	View Route Details
Detect Anomaly	x																			
Start Capture		x																		
Encapsulate Reading			x																	
Upload Road Reading				x																
Send upload request					x															
Perform upload tasks						x														
Remove from cache							x													
Add to cache								x												
Get cached data									x											
Add to upload queue										x										
Remove from upload queue											x									
Remove Duplicate Anomalies												x								
Get JSON Format													x							
Encapsulate Anomaly														x						
Submit anomaly															x					
Search for Road																x				
View Road Details																	x			
Warn Driver																		x		
Suggest Fastest Route																			x	
View Route Details																				x

Figure 29: Requirements Matrix, mapping between functional requirements and the SDD modules

References

[1] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: 2018.

[2] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: `1602.07261`. URL: `http://arxiv.org/abs/1602.07261`.

[3] C. Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 1–9. DOI: `10.1109/CVPR.2015.7298594`.