CSIR - Mobile Augmented Reality Number Plate Recognition

Architecture Specification Proposal

Team Members:

Mbulungo Musetsho (10176382)

Ndivhuwo Nthambeleni (10001183)

Joas Mogale (10354167)

Contents

1 Introduction

This document contains a detailed specification how the augmented reality project is going to be carried out by the Gruners group. It will specify both functional and non-functional requirements and the architectural design for the system. All these will be coupled with use cases for each particular system functionality and the testing thereof. This document will use agile methodologies to promote adaptation of the process to possible requirements changes and many other amendments based on practical issues that will be encountered throughout the process.

2 Vision and Scope

2.1 Vision

The vision of the project is to develop a mobile application for smart phones that will enable the phone to detect vehicle number plates using a camera view and then access, from the display, the information of the scanned vehicle in real-time. This application will then be used by mobile units to track vehicle statuses. For instance, whether the vehicle was stolen or not road-worthy.

2.2 Scope

The envisioned system is a mobile application, with a web front-end, which will allow the permitted users to do the following:

- scan through number plates for detection with the mobile application
- view the resulting information per detection in a user-friendly display.
- make all relavent CRUD operations, using a user-friendly web interface.

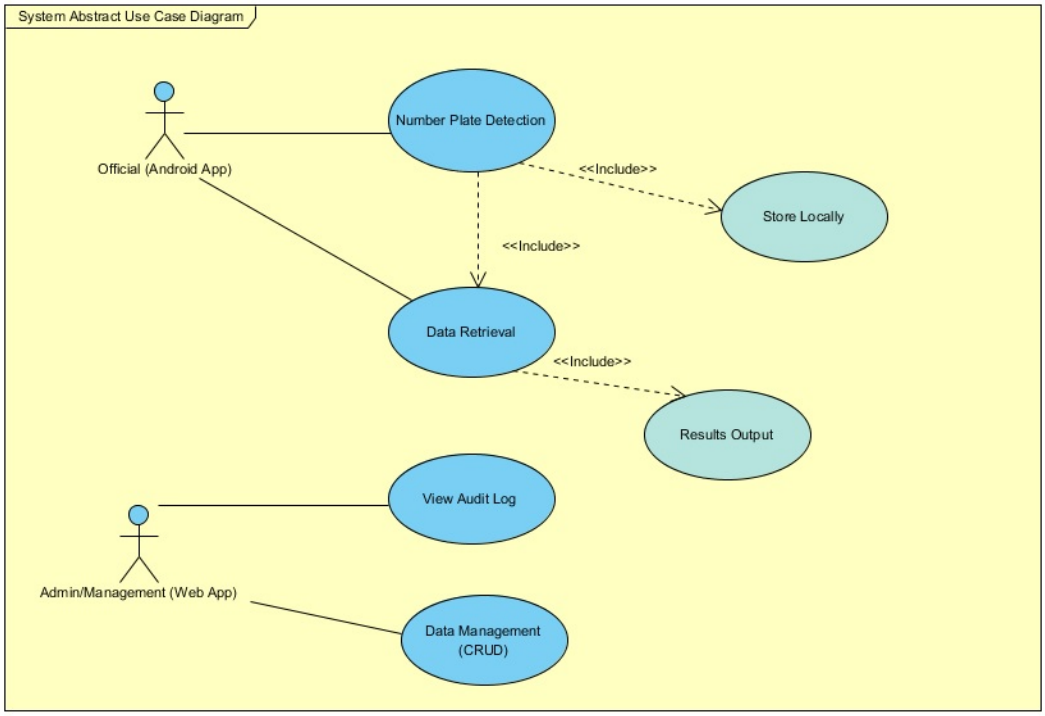

Figure 1: Abstract System Use Case Diagram

3 Architectural Requirements Specification

3.1 Access Channels

The envisioned system must be accessible by human users via the Android application (for number plate detections) and Web browsers (for management activities).

3.2 Quality Requirements

3.2.1 Performance

The system must offer fast performance in detection of vehicle number plates as well as retrievals from the database.

3.2.2 Accuracy

All detections must be verified for accuracy before the detection process continues.

3.2.3 Auditability

The system must maintain audits for all vital operations. These audit logs can then be provided upon request.

3.2.4 Scalability

The system must allow multiple users to make use of the application without interruptions. This holds for both the Android application and the web application.

3.2.5 Maintainability

All layers of the system must be developer-friendly. - That is, the design and development of the system must allow smooth maintenance of all the system's counterparts.

3.2.6 Usability

The system must allow all users to be able to interact with it without any major difficulties.

3.3 Architecture Constraints

- The mobile application has to run on the Android OS with the target being Android 4.4 but allowing for compatibility with older versions up to Android 4.0.3.
- The MVC (Model View Controller) architecture pattern is to be used throughout the project.

4 Software Architecture Specification

4.1 Architecture Requirements

4.1.1 Quality Requirements

- Performance
  - Number plate detection must take less than 5 seconds from application start-up.
  - Retrieval of information from the database must take no longer than 1 second.
- Accuracy Number plate detections must be at least 99.99% accurate. Detections made on non-stationary vehicles must be possible for vehicles at most 20 metres away from the scanning device.
- Auditability
  - The system will make audits of all operations that alter the database.
  - The system will also maintain a history of all number plate scans successfully done.
- Scalability

  - The system must be able to scan all types of Southern African number plates.
  - The system must be able to operate effectively and efficiently under a load of 10000 concurrent android application users or 100 concurrent web interface users.

- Maintainability To implement maintainability, the system will make use of a layered architecture that separates certain counterparts such that maintaining a subsystem does not, in any means, affect any other subsystem.

- Usability An average user must be able to use the system without any further training or extensive manual consultation required. This will be achieved by the use of design principles to enhance user experience.

4.1.2 Integration and Access Channel Requirements

These are the different channels through which the system can be accessed by all related users. The system will be accessible by human users through the following channels:

1. From the web browser through a user-friendly web interface. This implies that the system must be accessible from all widely used web browsers (including the most recent versions of Mozilla Firefox, Google Chrome, Apple Safari and Microsoft Internet Explorer).

2. From mobile android devices using the Android application.

4.1.3 Architecture Constraints

The following architecture constraints will be followed mainly for maintainability reasons:

1. All system counterparts must be developed and attached to a JAVA RESTful Web Service.

2. The business logic layer must provide an API for access to the SQL Database.

3. The system will make use of a MySQL database.

4. The mobile client must run on an Android application.

5. The system functionality will be hidden in a RESTful web service such that it is not exposed to any presentation layer counterpart.

4.2 Architectural patterns or Styles

For the sake of good high-level responsibility separation and allowance for reuse of lower level layer components across components in higher level layers, a layered architecture will be used for this system. The layered architecture for the system contains the following layers:

- Client Layer
- Access Layer
- Business Processes Layer
- Domain Objects Layer
- Backend Layer

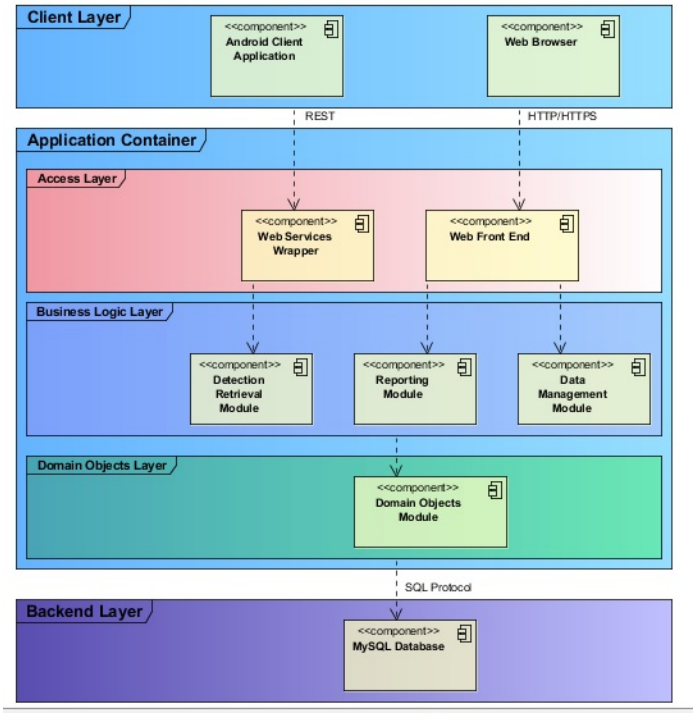

Figure 2: System Layered Architecture

The responsibility allocation across the layers is as follows:

1. Provide access to humans: Client Layer
2. Provide access to system functionality to human access layer and other systems: Access Layer
3. Business Logic Encapsulation: Business Processes Layer
4. Provide domain objects: Domain Objects Layer
5. Hosting Database: Backend Layer

4.3 Technologies

Technologies that will be used throughout the system development include:

- JAVA Native Android SDK
- HTML 5
- JAVA Programming Language
- Qualcomm Vuforia Augmented Reallity SDK
- MySQL Database for JDBC

5 Functional requirements and application design

5.1 Introduction

This section discusses all the functional requirements for the CSIR Augmented Reality Number Plate Recognition system.

5.2 Required functionality

5.2.1 Number Plate Detection

Upon opening the application, it displays a real-time view of the android device's rear camera. Using Augmented Reality, an orange box is drawn around a detected number plate. There may be multiple number plates in a scene. At this stage the "Detection Local Storage" use case is triggered to store the scanned number plates locally, after which the "Data Retrieval" use case is triggered to fetch data from the server. If the plate is not found or OK, then the box around the plate turns green and an audible tone indicates the check is complete. If the database returns a bad result (stolen, outstanding ticket, etc.), then the box should turn red and a different tone should be sound. Clicking on a box will open the number plate details screen, displaying any additional information available.

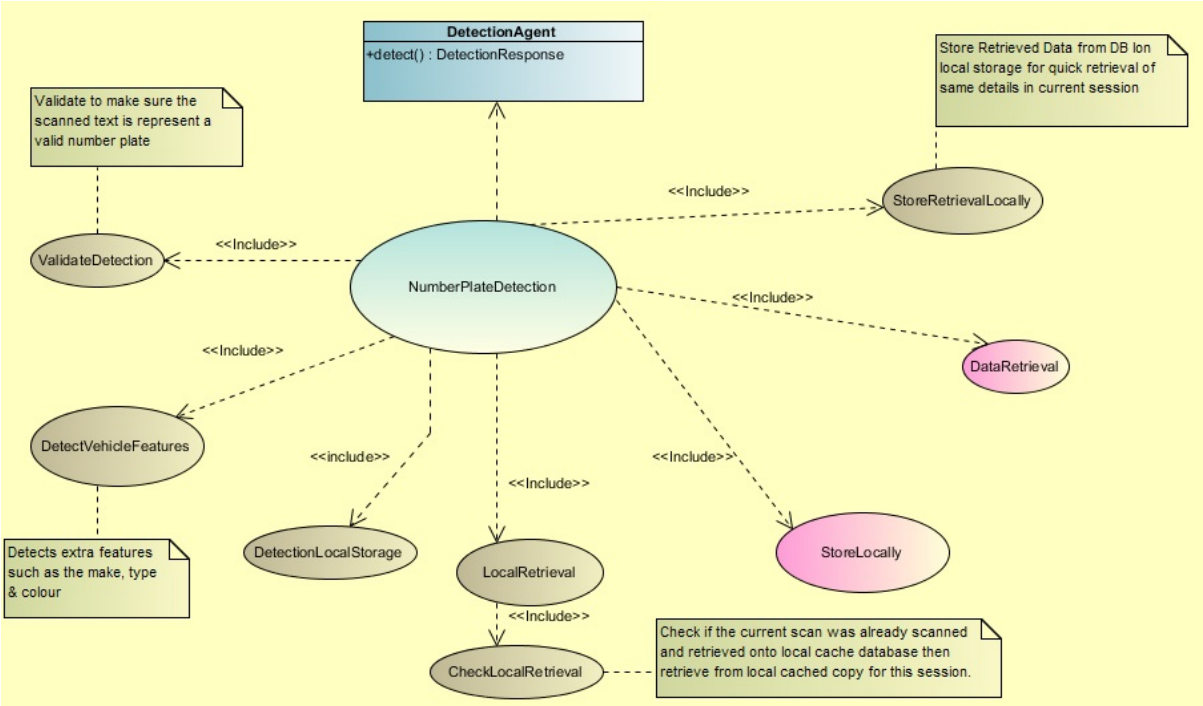

Figure 3: Number Plate Detection Use Case Diagram

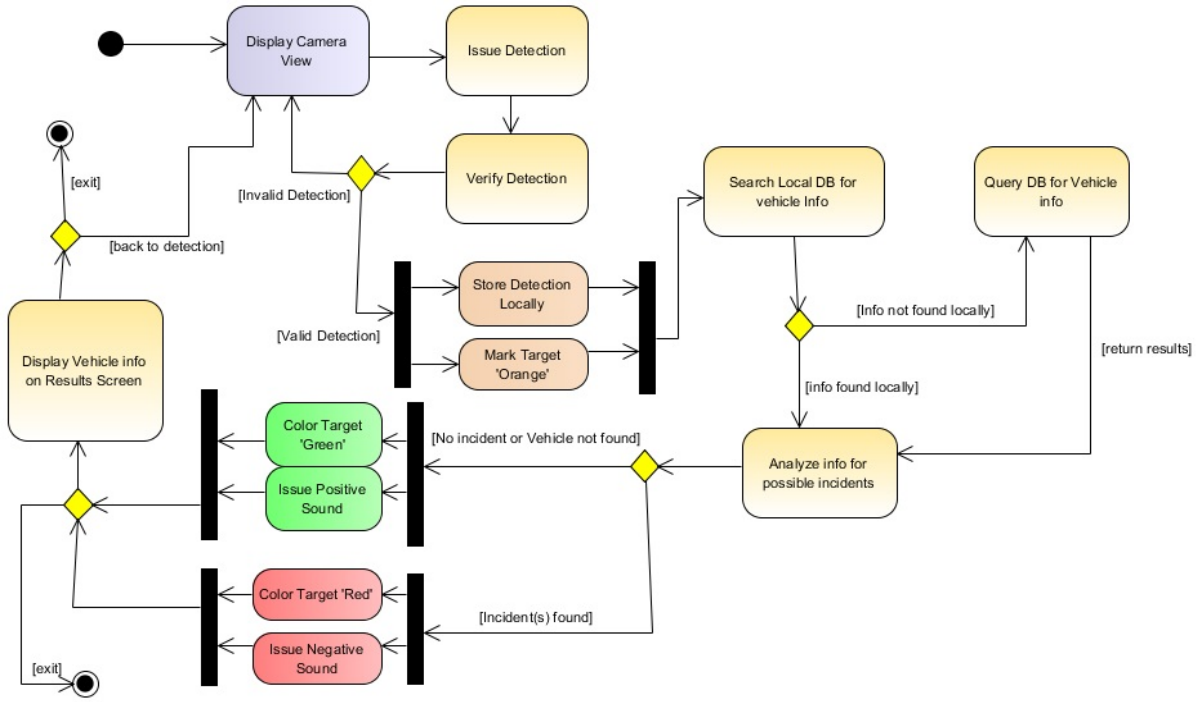

Figure 4: Number Plate Detection Activity Diagram

5.2.2 Detection Local Storage

Once the mobile application has made a detection, this use case is triggered just before data is requested from the database. The system needs to keep record of all the scans the user has made. So the system will store the detected number plate string onto a local database. All other features detected with the number plate will be stored locally as well (e.g. colour, make, type, model).

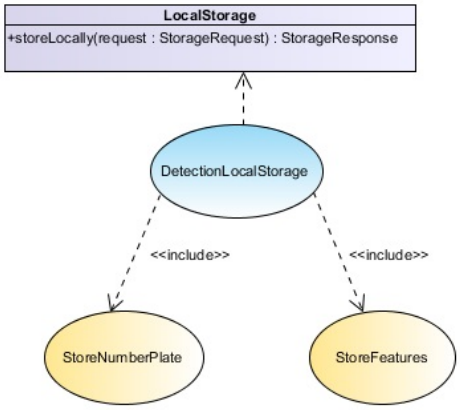

Figure 5: Detection Local Storage Use Case Diagram

5.2.3 Data Retrieval

Data Retrieval may be triggered either by the user (via a searchable list of detected number plates stored locally) or by the system as a sequence of use cases. Its function is to retrieve information related to the requested number plate. Firstly, the application searches against its local database in case the information had already been retrieved in the current session. If there's no match, it makes an HTTP query to fetch data pertaining to a particular number plate. The retrieved data is stored onto the local database for display. This information is only kept locally per session. When the session is closed this storage is cleared to ensure that the next session retrieves the most recent data from the remote database.

5.2.4 Results Output

When the user clicks on a number plate that was detected (Either from the live camera view or from the list of scanned number plates stored locally), this function fetches the retrieved data from the local database and displays them on a screen.

5.2.5 Data Management

This functionality is made available for the web application for management users. The function of this use case is to perform all CRUD on the remote database as well as viewing audit trails.

5.2.6 Audit Log

This functionality makes sure that every operation that all kinds of users perform within the system are recorded. It is triggered upon successful completion of an operation.

5.3 Use case prioritization

5.3.1 Critical

The following use cases are considered critical:

- Number Plate Detection
- Data Retrieval
- Results Output

5.3.2 Important

The following use cases are considered important:

- Data Management
- Audit Log
- Detection Local Storage

5.4 Domain Objects

This section introduces core domain concepts and aspects of the requirements which are used across different use cases. The main domain objects are:

- Owner: This encapusulates all attributes of a vehicle owner. This could be a person or a company.
- Vehicle: All attributes of a car are encapsulated in this object and will contain a reference to an 'Owner' object such that each car has exactly one owner.
- Detection: All detections that successfully retrieved data will each be encapsulate in this object.

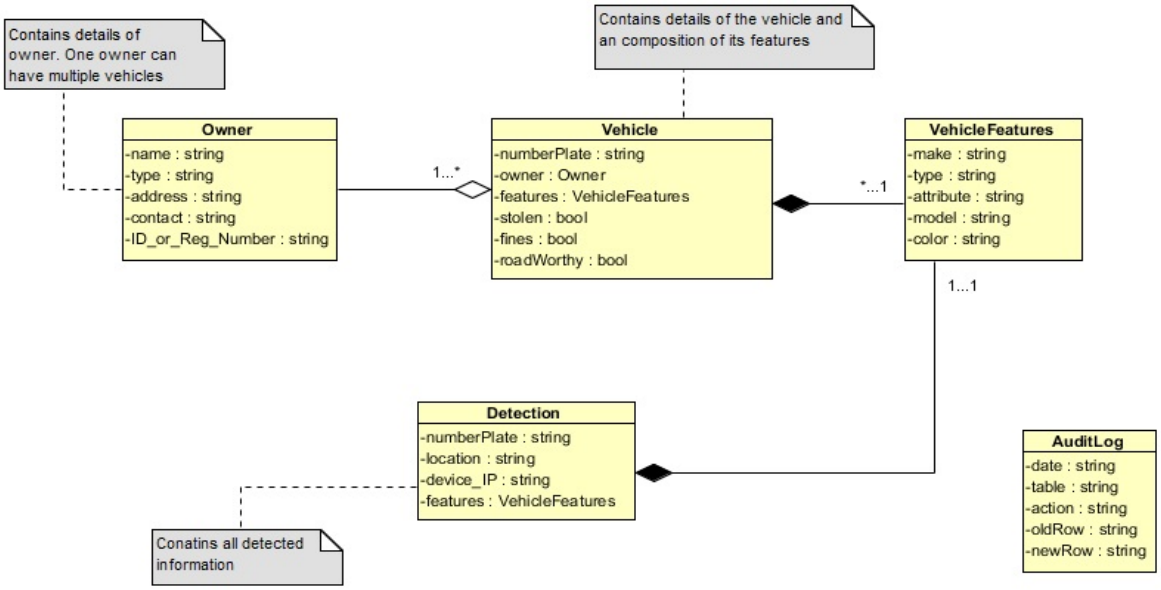

Figure 6: Core Domain Objects