|  |
| --- |
|  |
| AUTOMATIC LICENSE PLATE RECOGNITION DESIGN MANUAL |
|  |
| **By**  **Anthony Tierney** |
|  |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |
| AUTOMATIC LICENSE PLATE RECOGNITION DESIGN MANUAL  Anthony Tierney  I.T. Carlow, Kilkenny Road, Carlow |
| 06-Jan-14 |
|  |

|  |
| --- |
|  |

Table of Contents

[1. Introduction 1](#_Toc377129330)

[1.1. Purpose 1](#_Toc377129331)

[1.2. Scope 1](#_Toc377129332)

[1.3. Intended Audience 1](#_Toc377129333)

[2. App Overview 2](#_Toc377129334)

[3. Software Stack 3](#_Toc377129335)

[4. System Architecture 4](#_Toc377129336)

[4.1. Presentation Tier 4](#_Toc377129337)

[4.2. Application Tier 5](#_Toc377129338)

[4.3. Data Tier 5](#_Toc377129339)

[5. Database Design 6](#_Toc377129340)

[5.1. CarPark 6](#_Toc377129341)

[5.2. Attendant 6](#_Toc377129342)

[5.3. Vehicle 6](#_Toc377129343)

[5.4. VManufacturer 6](#_Toc377129344)

[5.5. VModel 6](#_Toc377129345)

[6. User Interface 7](#_Toc377129346)

[6.1. Initial Opening Screen 7](#_Toc377129347)

[6.2. Standard Opening Screen 7](#_Toc377129348)

[6.3. Standard user panel 8](#_Toc377129349)

[6.4. Download Car park information 8](#_Toc377129350)

[6.5. Admin panel 9](#_Toc377129351)

[6.6. CRUD Users 9](#_Toc377129352)

[6.7. CRUD Car park 10](#_Toc377129353)

[6.8. Add new Vehicle 10](#_Toc377129354)

[7. Use cases 11](#_Toc377129355)

[7.1. Use Case Diagram 11](#_Toc377129356)

[7.2. Detailed Use Cases and Alternate use Cases 11](#_Toc377129357)

[8. Class Diagram 14](#_Toc377129366)

[9. Description of classes 15](#_Toc377129367)

# Introduction

## Purpose

The purpose of this document is to details the design of this project.

## Scope

This manual describes the following:

* A description of the functionality through use cases
* A design of the app
* User interfaces
* Architecture and database design

This document is not a user manual and should not be given to end users that will use this app as it does not contain any instructions on how to use this app.

## Intended Audience

The audience that this document is intended for is those who have knowledge of the technical aspect of developing an app including an understanding of UML.

# App Overview

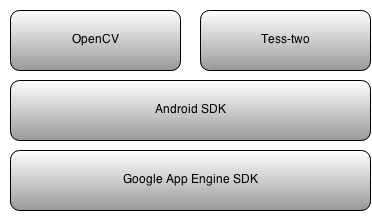
The app in this project is a tool that allows car park attendants to increase the rate at which they verify whether a car is parked with permission or if it is not.

The app will be created for the android mobile platform and it will make use of Google cloud data store to store databases that are required for the app to function. There are a number of steps in licence plate recognition, these are:

* Locating the licence plate in the image
  + In this app there will be a boundary box visible to the user to allow them to align the licence plate within the box
* Noise Filtering
  + Noise filtering reduces artefacts in the image an may result in an improved edge detection output
* Edge detection
  + Running an edge detection filter on the image may have the effect of improving the optical character recognition and retrieve a more accurate reading from the image
* Optical character recognition
  + This is the main functionality of the app. The processed image is split into separate images containing one character. This character is then put through the optical character recognition engine to retrieve the character contained within the image
* Cloud functionality
  + A cloud Datastore is used to store the databases needed for each car park and attendant.
  + The necessary database will be downloaded at the start of each day to the devices local memory for use throughout the day

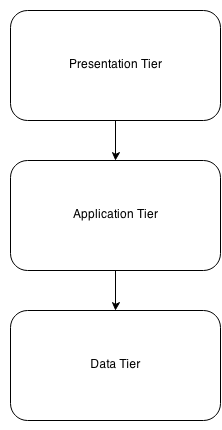
# Software Stack

In this software stack, OpenCV and Tess-two are temporary additions. They will be removed later in the project when the main functionalities are completed and replaced with custom code. The Android SDK is comprised of all the classes and interfaces that allow developers to create apps. Google App Engine SDK contains all the necessary components to communicate with the app running in the cloud.



# System Architecture

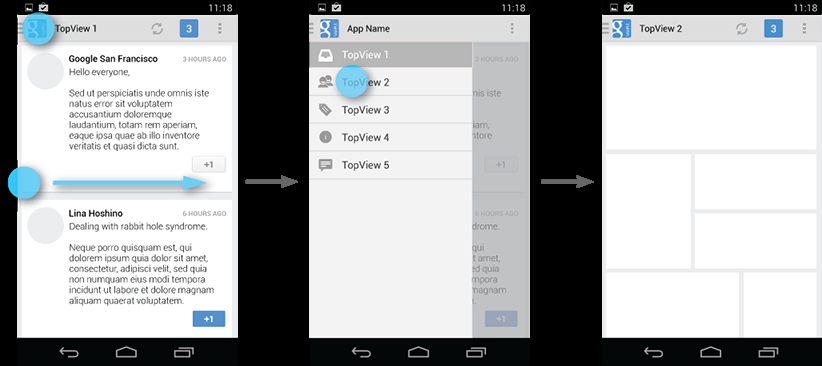
The three layer OSI model will be used to ensure that the main architectural components are well separated to allow for each component to be changed or upgraded as necessary. Below is a context diagram of the multi-tier architecture:



Note that the arrows are only pointing in a single direction, stating that the presentation tier may communicate with the application tier but the application tier cannot communicate with the presentation tier. The same stands for the application tier and the data tier.

## Presentation Tier

This tier (or layer) contains all the code and preferences for the user interface for the app. The initial screen will present the user with the camera so that the user is not searching through menus to get to the main component of the app. Hence, the app will be ready to function as soon as it is opened. To access other features of the app, the user interface will utilise androids slide out menu as pictured below.



The use of this type of interface will result in an easier method to navigate the app and a cleaner, more minimalist interface that is more pleasing to the user.

This layer will contain none of the code that allows the app to function. The application layer contains the functioning code.

## Application Tier

The application tier contains all the functioning code that allows the app to function. This tier does not communicate with the presentation tier (except for displaying errors) but it has single direction communication with the data tier.

The code for the image manipulation, edge detection and optical character recognition are all present in this section.

## Data Tier

The database used is a NoSQL database from Google Called Datastore. Google App Engine is the middleware that communicates with the database and it is written in python2. The flasks framework will be used for sending and receiving data.

# Database Design

## CarPark

|  |  |
| --- | --- |
| **CarParkID** | String |
| CarParkName | String |
| CarParkAddress | String |
| CarParkCapacity | integer |
| AttendantID | String |

## Attendant

|  |  |
| --- | --- |
| AttendantID | String |
| Name | String |
| Address | String |
| PhoneNum | Integer |

## Vehicle

|  |  |
| --- | --- |
| VehicleID | String |
| OwnerName | String |
| PhoneNumber | Integer |
| ManufacturerID | String |
| ModelID | String |
| Registration | String |
| Colour | String |

## VManufacturer

|  |  |
| --- | --- |
| ManufacturerID | String |

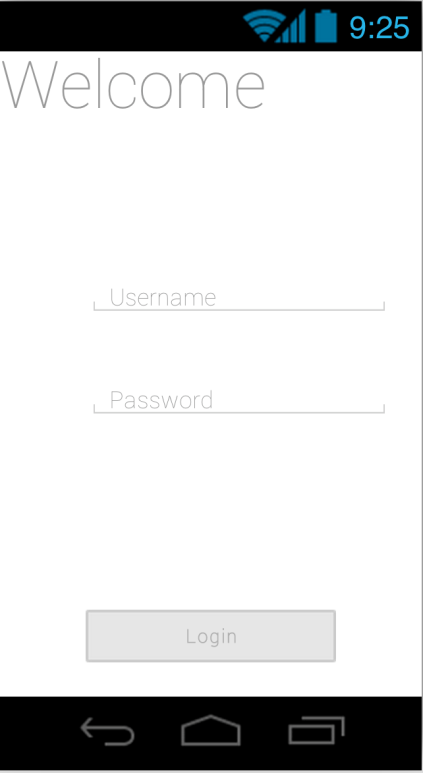
## VModel

|  |  |
| --- | --- |
| ModelID | String |
| ManufacturerID | String |

# User Interface

## Initial Opening Screen

This screen is displayed when the app is first installed on the device or when a user wishes to login to the system.



## Standard Opening Screen

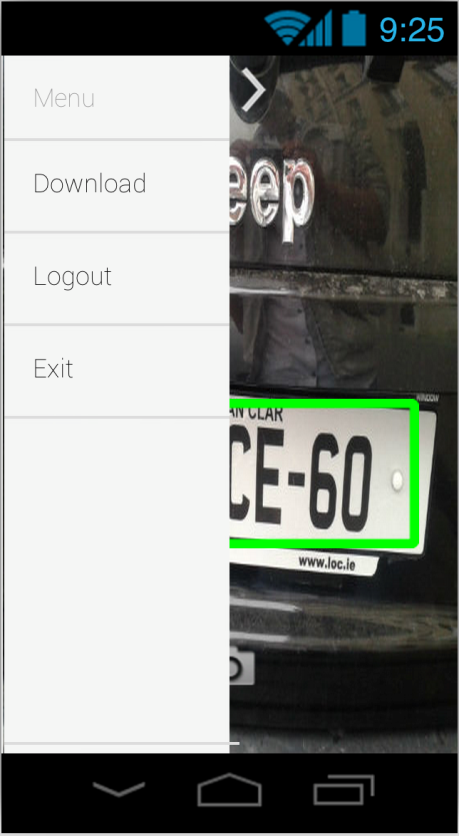
This screen is displayed when the user is already logged in and opens the app. It houses the interface for the camera because it saves the user searching through menus to start their task. There will be a slide out menu to reveal the menu options (or touch the arrow in the top left corner).



## Standard user panel

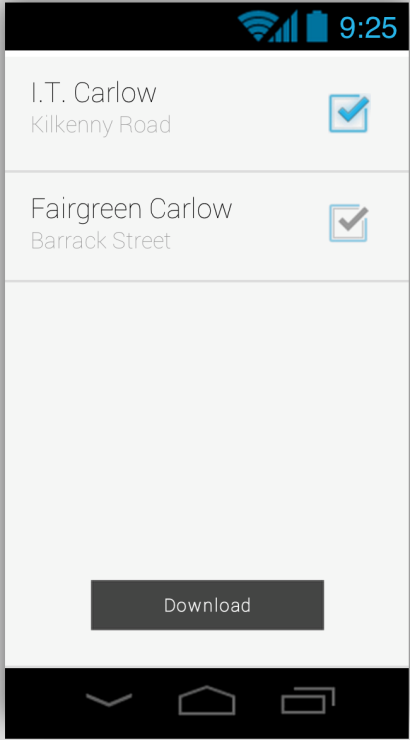
This screen allows the standard user (car park attendant) to access various menu items available to them such as:

* Download a database for a specific car park
* Logout
* Exit



## Download Car park information

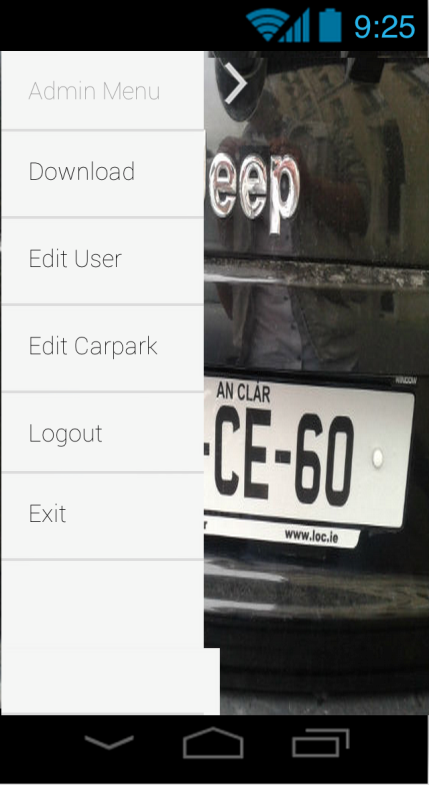
This screen allows the user to download information of the car parks that they have been assigned



## Admin panel

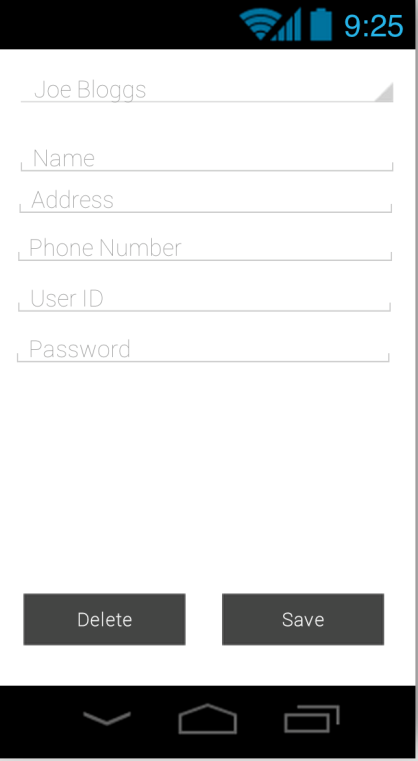
This screen is similar to that of the standard user panel, with the addition of extra options specific to the system administrator. These additions include:

* CRUD Users
* CRUD Car parks



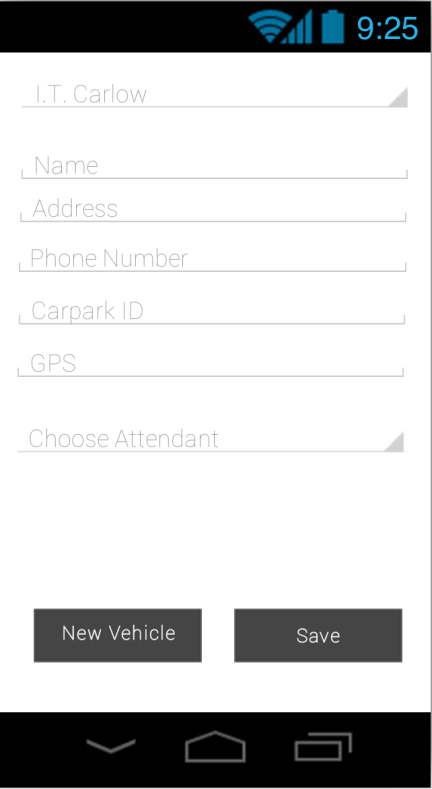
## CRUD Users

The admin selects the user to edit from the drop down menu at the top, makes the necessary changes and then saves them. To add a new user, select the blank entry in the drop down box.



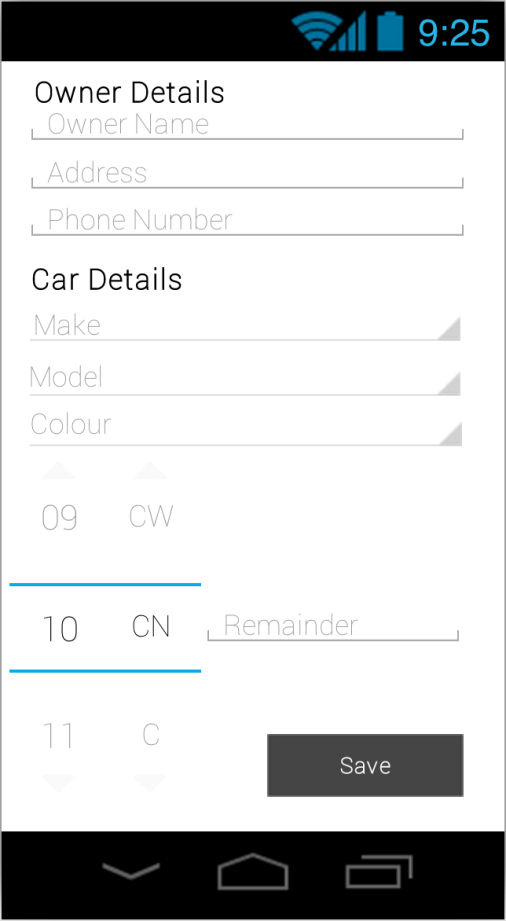
## CRUD Car park

Edit Details of a car park, assign an attendant and add a new vehicle.



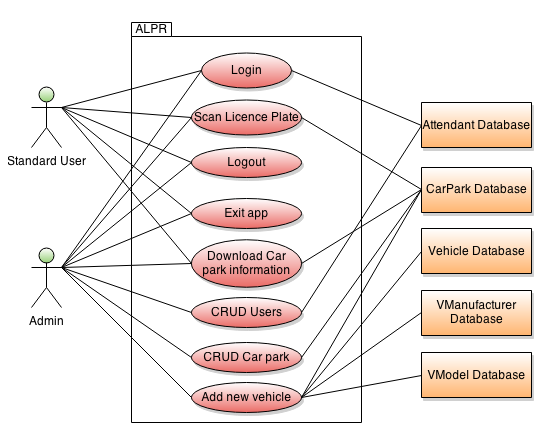
## Add new Vehicle

Save a new vehicle to a car park.



# Use cases

## Use Case Diagram



## Detailed Use Cases and Alternate use Cases

### Login

**Use Case:** Login

**Actors:** Admin, Standard User, Attendant Database

**Description:**

1. The user (admin or standard user) locates the app on the device
2. The user opens the app
3. Username and password are entered
4. The system checks the online Attendant Database to see if the login details are correct
5. The user is logged in and can now begin checking license plates

**Alternates:**

1. The system checks the online Attendant Database to see if the login details are correct
   1. The login details are incorrect
      1. The user is notified of this issue
      2. The user is prompted to reenter their credentials

### Scan licence plate

**Use Case:** Scan License plate

**Actors:** Standard User, Admin, Car park database

**Description:**

1. The user (Standard User, Admin ) opens the app
2. The user aligns the license plate within the on screen boundary box
3. The user captures the image
4. The app then proceeds with processing the image and optical character recognition
5. The user is notified of the result to verify the license plate is correct
6. The user confirms that the license plate is correct
7. The app checks the Car park database to verify if it has parking rites
8. The user is notified of the legality of the vehicle parked

**Alternatives:**

1. The user is notified of the result to verify the license plate is correct
   1. The license plate is incorrect
      1. The user is prompted to manually enter the license plate number

### Download car park information

**Use Case:** Download Car Park Information

**Actors:** Standard User, Admin, Car park database

**Description:**

1. The user wishes to download the car park database
2. From the main screen, open the menu and select the relevant menu item
3. The user taps the car park location that they wish to download
4. The selected database is downloaded over the network connection
5. The app saves this database to the device and removes previous versions

**Alternatives:**

1. The selected database is downloaded over the network connection
   1. There is no network connection
      1. The user is notified of this issue
      2. The user resolved the issue with the network connection

### CRUD Users

**Use Case:** CRUD Users

**Actors:** Admin, Attendant Database

**Description:**

1. The user opens the menu from the main screen and selects the relevant menu item
2. The user selects the attendant to edit from the drop down box
3. The user makes the changes needed
4. The users saves the changes
5. The app makes these changes to the online Attendant Database

**Alternatives:**

1. The user saves the changes
   1. The user left a field blank
      1. The user is notified of this issue
      2. The user enters data into the field

### CRUD Car park

**Use Case:** CRUD Car Park

**Actors:** Admin, CarPark Database

**Description:**

1. The user opens the menu from the main screen and selects the relevant menu item
2. The user selects the car park to edit from the drop down box
3. The user makes the changes needed
4. The users saves the changes
5. The app makes these changes to the online Attendant Database

### Add new vehicle

**Use Case:** Add new Vehicle

**Actors:** Admin, CarPark Database, Vehicle Database, VManufacturer Database, VModel Database

**Description:**

1. From the CRUD Car Park Screen navigate to the “Add new vehicle” screen
2. Enter the details of the owner
3. The Manufacturer list is loaded from the VManufaceture Database
4. The Model list is loaded from the VModel Database, using the selected VManufacturer as key
5. The registration number is inserted
6. The car is saved to the database with the Car park ID associated

**Alternatives:**

1. The car is saved to the database with the Car park ID associated
   1. There are fields missing data
      1. The user is notified of this error
      2. The user fills in the empty data fields
   2. There is no network connection
      1. The user is notified of this issue
      2. The user resolves the issue with the network connection

### Logout

**Use Case:** Logout

**Actors:** Admin, Standard User

**Description:**

1. The user (Standard user, Admin) opens the menu
2. The user selects the option to logout
3. The app logs the user out

### Exit app

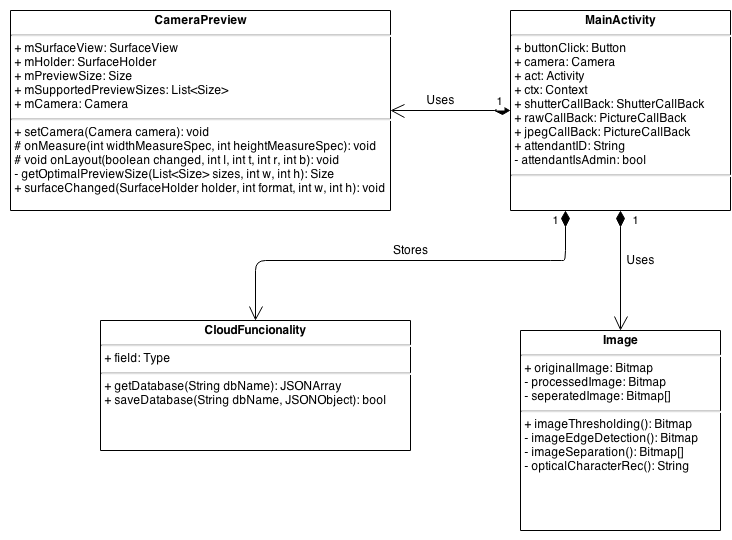
**Use Case:** Exit App

**Actors:** Standard User, Admin

**Description:**

1. The user wishes to exit the app
2. The user opens the menu and selects the option to exit
3. The app releases resources for use by other apps
4. The app closes

# Class Diagram



# Description of classes

## Main Activity

This class is the main class in this app. It initialises the camera and handles the input from the camera class when an image is taken. It communicates with the Cloud Functionality class to retrieve different databases (and CRUD them if the user is an administrator). It also initialises the image class to process the image.

## Camera Preview

This class is the container for the input from the camera. It handles the live input stream fed from the camera to the app.

## Image

This class handles the entire image processing once the image has been taken. It firstly runs a noise filter over the image, and then it gets the grey scale image. From this grey scale image, edge detection is performed and then character separation separates the image into sub images of the different characters. These sub images are then ran through an optical character recognition engine to retrieve the character in the image.

## Cloud Functionality

This class handles all the cloud interaction for the app. It communicates with a python application that is running on Google app engine, which in turn, communicates with the database on Google Datastore.