**University of Karachi**

**Department of Computer Science**

****

**Final Year Project Report**

Vehicle/Automobile Parking Security System

**Supervise by:**

**Sir Dr. Tehseen Jilani / Sir Usman Amjad**

**Submitted by:**

**Parkash Kumar (B11101067)**

**S .Haider Ali Naqvi (B10101109)**

ABSTRACT:

*Automotive theft has been a persisting problem all around the Pakistan especially in our Karachi and greater challenge comes from professional thieves. In this paper, we present an automobile parking security system.*

*This security system is based upon machine vision / image processing technology that takes it to a new level of theft protection. It is based on hardware system, for real time acquisition of number-plate images using an active IR illuminator. This system can locate and recognize the number plate, identify the unauthorized number plate. When the unauthorized car arrives, our system will alarm and gives the option for registration of the vehicle. For being more secure we had made this live streaming as well, so that the police can see/search any vehicle in our parking area through CDMA or GPRS networks*. *The status of the parking field detected by sensor nodes is reported periodically to a database via the deployed wireless sensor network and its gateway. The database can be accessed by the upper layer management system to perform various management functions, such as finding vacant parking lots, auto-toll, security management, and statistic report.The system was tested in a simulating environment and it was found very robust, reliable. To prove the effectiveness of the system proposed by us we have developed and presented a mathematical model which will be discussed in brief further in the paper.*

**Acknowledgement:**

I would like to thanks the project supervisor **Sir Usman Amjad** without whose help and support throughout, this project would not have been possible.

Table of Content

**Content Page**

1. Introduction …………………………………………………....

1.1 Project Motivation

1.2 Aims and Objective

2**.** Description of Technologies

2.1 Matlab

2.2 Nodejs

2.3 Ajax

2.4 MongodB

3. SDLC

3.1 Watefall Model

4. Description of Hardware System

4.1 Camera

4.2 Computer/Laptop

5. Project

5.1 Registration panel

5.2 Logs

5.3 Entries

5.4 Live streaming

5.5 Notifications

5.6 Task runner

5.7 Database

5.8 Image processing

# Introduction.

# 1.1 Project Motivation.

Basically a desktop application is made for vehicles surveillance.Image of a vehicleis being captured through WIFI CCTV/IP/IR camera, then extract its number and save it to database for further verifications.

# 1.2 Aims and Objective.

The core objectives which have been designated as fundamental to the project are:

1. Registration/Un-registration of the car,
2. Detection of the car,
3. Pictures of the number plate,
4. Alarm to the monitoring department,
5. Identify all the vehicles that are registered in our database and make sure that any ill-legal vehicle cannot be parked in our premises.

# Description of Technologies.

# 2.1 What is Matlab?

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), **Matlab** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE). It contains a base [workspace](https://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](https://en.wikipedia.org/wiki/Plug-in_(computing)) system for customizing the environment. It developed a proprietary programming language that allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

# Interface

MATLAB is a [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) [numerical computing](https://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](https://en.wikipedia.org/wiki/Fourth-generation_programming_language). It supports developing applications with [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) features.GUI provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application. MATLAB apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox, include apps with custom user interfaces. You can also create your own custom apps, including their corresponding UIs, for others to use.

# Application workflow

This application is basically divided into parts client-side and server-side. Client side consist on matlab and web-based user interface whereas on server side requests are handle by nodejs sever.

1. Gulp: It is task runner which is used to start nodejs server and continously watching any change in notepad file which stores number extracted from number plate of a car. If any change occurs in notepad file then it automatically restarts our server.

1. Matlab: It is used to extract number from a number plate of car and stores in a notepad file.
2. Nodejs: It is used to handle requests and extract number from a notepad file and stores in a database
3. Mongodb: It is used to store details of a car and daily entries.
4. Logs: It is an area in web where user can see all daily entries.
5. Ajax: Whenever new car is entered number is extracted in a notepad file through matlab and server gets restart with the help of gulp when server restarts an ajax request is triggered to nodejs server that checks the entered car number in a database if it is not present then marked is as an unregistered car.
6. Nodemailer: Whenever new license number is detect a mail is sent to a user with complete details of car

Gulp is continously watching notepad file and restarts server

Matlab is extracting number plate of car

Notepad file

Nodejs server reads changes from notepad file and stores in db

Database

# 2.2 What is Node js?

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [runtime environment](https://en.wikipedia.org/wiki/Runtime_system) for developing [server-side](https://en.wikipedia.org/wiki/Server-side) [Web applications](https://en.wikipedia.org/wiki/Web_application). Node.js is a platform built on Chrome’s JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. It’s not a [JavaScript framework](https://en.wikipedia.org/wiki/JavaScript_framework) but many of its basic modules are written in [JavaScript](https://en.wikipedia.org/wiki/JavaScript), and developers can write new modules in JavaScript. The runtime environment interprets JavaScript using [Google](https://en.wikipedia.org/wiki/Google)'s [V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) JavaScript engine.

# Main Features

Following are some of the important features that make Node.js the first choice of software architects.

* **Asynchronous and Event Driven** All APIs of Node.js library are asynchronous that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.

# Interface

Node.JS interface is classified into modules,

Node.JS- a Common.JS Module Implementation.

Your code uses require to include modules.

Modules use exports to make things available.

**COMMON.JS**

An ecosystem for JavaScript outside the browser

1. Modules
2. Promises
3. Binary
4. Filesystem
5. Console
6. System
7. Testing

# Application

The module exports two specific components:

A Console class with methods such as console.log (), console.error () and console.warn () that can be used to write to any Node.js stream.

A global console instance configured to write to stdout and stderr. Because this object is global, it can be used without calling require('console').

**Example using the global Console:**

console.log('hello world');

// Prints: hello world, to stdout

console.log('hello %s', 'world');

// Prints: hello world, to stdout

console.error(new Error('Whoops, something bad happened'));

// Prints: [Error: Whoops, something bad happened], to stderr

const name = 'Will Robinson';

console.warn(`Danger ${name}! Danger!`);

// Prints: Danger Will Robinson! Danger!, to stderr

**Example using the Console class:**

const out = getStreamSomehow();

const err = getStreamSomehow();

constmyConsole = newconsole.Console(out, err);

myConsole.log('hello world');

// Prints: hello world, to out

myConsole.log('hello %s', 'world');

// Prints: hello world, to out

myConsole.error(new Error('Whoops, something bad happened'));

// Prints: [Error: Whoops, something bad happened], to err

const name = 'Will Robinson';

myConsole.warn(`Danger ${name}! Danger!`);

// Prints: Danger Will Robinson! Danger!, to err

The API for the Console class is designed fundamentally around the Web browser Console object, the Console is Node.js is not intended to duplicate the browsers functionality exactly.

# 2.3 What is Ajax

Ajax is not a programming language or a tool, but a concept. Ajax is a [client-side script](http://www.seguetech.com/blog/2013/02/07/what-are-the-pros-and-cons-of-client-side-scripting) that communicates to and from a server/database without the need for a[postback](http://www.c-sharpcorner.com/uploadfile/2f73dd/what-is-postback-in-Asp-Net/) or a complete page refresh. The best definition for Ajax is “the method of exchanging data with a server, and updating parts of a web page - without reloading the entire page.” Ajax itself is mostly a generic term for various JavaScript techniques used to connect to a web server dynamically without necessarily loading multiple pages. In a more narrowly-defined sense, it refers to the use of [XmlHttpRequest](http://en.wikipedia.org/wiki/XMLHttpRequest) objects to interact with a web server dynamically via JavaScript.

# Benefits of Ajax

There are 4 main benefits of using Ajax in web applications:

**Callbacks:** Ajax is used to perform a callback, making a quick round trip to and from the server to retrieve and/or save data without posting the entire page back to the server. By not performing a full postback and sending all form data to the server, network utilization is minimized and quicker operations occur. In sites and locations with restricted bandwidth, this can greatly improve network performance. Most of the time, the data being sent to and from the server is minimal. By using callbacks, the server is not required to process all form elements. By sending only the necessary data, there is limited processing on the server. There is no need to process all form elements, process the ViewState, send images back to the client, or send a full page back to the client.

**Making Asynchronous Calls:** Ajax allows you to make asynchronous calls to a web server. This allows the client browser to avoid waiting for all data to arrive before allowing the user to act once more.

**User-Friendly:**Because a page postback is being eliminated, Ajax enabled applications will always be more responsive, faster and more user-friendly.

**Increased Speed:**The main purpose of Ajax is to improve the speed, performance and usability of a web application. A great example of Ajax is the movie rating feature on Netflix. The user rates a movie and their personal rating for that movie will be saved to their database without waiting for the page to refresh or reload. These movie ratings are being saved to their database without posting the entire page back to the server.

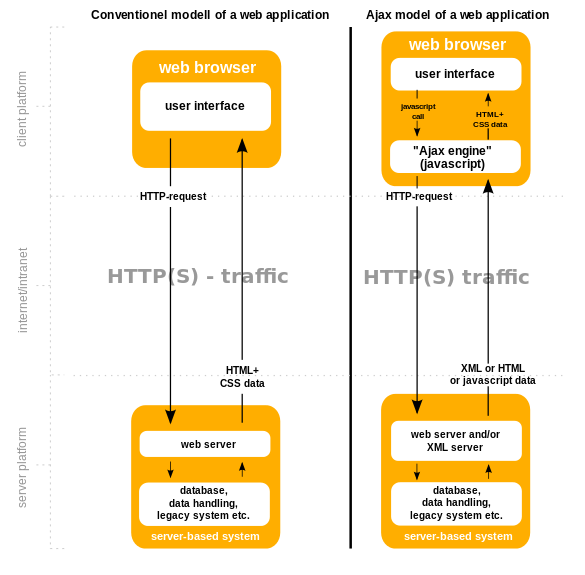
# Technical Aspects

Ajax callbacks can be done by instantiating an XMLHttpRequest object in the client-side JavaScript. The XMLHttpRequest object can be used to directly call server-side objects like pages and web services. These pages and web services will either save and/or return data.

Ajax was originally an acronym for Asynchronous JavaScript and XML. “Asynchronous” means that multiple events are happening independently of one another. Once a client initializes an Ajax callback to the server, the client will not need to wait for a response and can continue to use the web application while the request is being processed. Once done, the server will send a response back to the client and the client will process it as necessary.

# Application

Applications created with Ajax use an [engine](http://whatis.techtarget.com/definition/engine)that acts as an intermediary between a user's browser and the server from which it is requesting information. Instead of loading a traditional Web page, the user's browser loads the Ajax engine, which displays the page the user sees. The engine continues to run in the background, using JavaScript to communicate with the Web browser. User input or clicking on the page sends a JavaScript call to the Ajax engine, which can respond instantly in many cases. If the engine needs additional data, it requests it from the server, usually using XML, while it is simultaneously updating the page.



**Example**

Here is an example of a simple Ajax request using the [GET](https://en.wikipedia.org/wiki/GET_(HTTP)) method, written in [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

get-ajax-data.js:

*// This is the client-side script.*

*// Initialize the Http request.*

**var**xhr=**new**XMLHttpRequest ();

xhr.open ('get','send-ajax-data.php');

*// Track the state changes of the request.*

xhr.onreadystatechange=**function**(){

**var**DONE=4;*// readyState 4 means the request is done.*

**var**OK=200;*// status 200 is a successful return.*

**if**(xhr.readyState===DONE){

**if**(xhr.status===OK){

alert (xhr.responseText);*// 'This is the returned text.'*

}**else**{

alert ('Error: '+xhr.status);*// An error occurred during the request.*

}

}

};

*// Send the request to send-ajax-data.* xhr.send (**null**);

# 2.4 What is Mango DB?

MongoDB is one of the [database](http://searchsqlserver.techtarget.com/definition/database) type which lies under the [NoSQL](http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL) banner. Instead of using [tables](http://searchsoa.techtarget.com/definition/table) and [rows](http://searchoracle.techtarget.com/definition/row) as in [relational databases](http://searchsqlserver.techtarget.com/definition/relational-database), MongoDB is built on anarchitecture of collections and documents. Documents comprise sets of [key-value pairs](http://searchenterprisedesktop.techtarget.com/definition/key-value-pair) and arethe basic unit of data in MongoDB. Collections contain sets of documents and function as the equivalent of relational database tables.

MongoDB supports dynamic [schema](http://searchsqlserver.techtarget.com/definition/schema) design, allowing the documents in a collection to have different fields and structures. The database uses a document storage and data interchange format called BSON, which provides a binary representation of [JSON](http://searchwindevelopment.techtarget.com/definition/JSON-Javascript-Object-Notation)-like documents. Automatic [sharding](http://searchcloudcomputing.techtarget.com/definition/sharding) enables data in a collection to be distributed across multiple systems for horizontal as data [volumes](http://searchstorage.techtarget.com/definition/volume) increase.

# Main Features

Some of the features include:

* **Document-oriented**

Instead of taking a business subject and breaking it up into multiple relational structures, MongoDB can store the business subject in the minimal number of documents. For example, instead of storing title and author information in two distinct relational structures, title, author, and other title-related information can all be stored in a single document called Book.

* **Ad hoc queries**

MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.

* **Indexing**

Any field in a MongoDB document can be [indexed](https://en.wikipedia.org/wiki/Database_index) – including within arrays and embedded documents (indices in MongoDB are conceptually similar to those in[RDBMSes](https://en.wikipedia.org/wiki/RDBMS)). Primary and secondary indices are available.

* **Replication**

MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica set member may act in the role of primary or secondary replica at any time. The primary replica performs all writes and reads by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally perform read operations, but that data is eventually consistent by default.

* **Load balancing**

MongoDB scales horizontally using [sharding](https://en.wikipedia.org/wiki/Sharding). The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.). Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution.

MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. MongoDB is easy to deploy, and new machines can be added to a running database.

* **File storage**

MongoDB can be used as a [file system](https://en.wikipedia.org/wiki/File_system), taking advantage of load balancing and data replication features over multiple machines for storing files.

This function, called [Grid File System](https://en.wikipedia.org/wiki/Grid_File_System), is included with MongoDB drivers and available for many development languages (see "[Language Support](https://en.wikipedia.org/wiki/MongoDB#Language_support)" for a list of supported languages). MongoDB exposes functions for file manipulation and content to developers. GridFS is used, for example, in plugins for [NGINX](https://en.wikipedia.org/wiki/Nginx) and[lighttpd](https://en.wikipedia.org/wiki/Lighttpd). Instead of storing a file in a single document, GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.

In a multi-machine MongoDB system, files can be distributed and copied multiple times between machines transparently, thus effectively creating a load-balanced and fault-tolerant system.

* **Aggregation**

[MapReduce](https://en.wikipedia.org/wiki/MapReduce) can be used for batch processing of data and aggregation operations.

The aggregation framework enables users to obtain the kind of results for which the [SQL](https://en.wikipedia.org/wiki/SQL) GROUP BY clause is used. Aggregation operators can be strung together to form a pipeline – analogous to [Unix pipes](https://en.wikipedia.org/wiki/Pipeline_(Unix)). The aggregation framework includes the $lookup operator which can join documents from multiple documents.

* **Server-side JavaScript execution**

JavaScript can be used in queries, aggregation functions (such as MapReduce), and sent directly to the database to be executed.

* **Capped collections**

MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a [circular queue](https://en.wikipedia.org/wiki/Circular_queue).

# SDLC

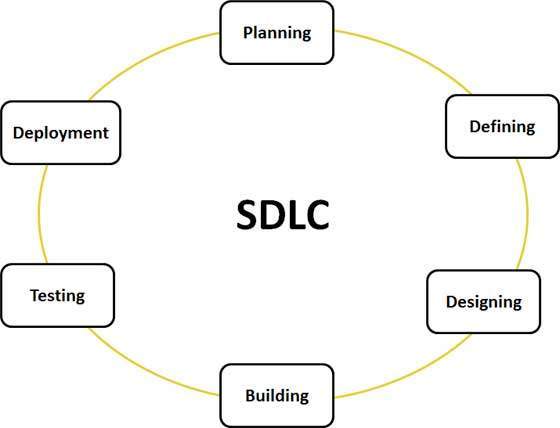
SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

* SDLC is the acronym of Software Development Life Cycle.
* It is also called as Software development process.
* The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.
* ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

## What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development life cycle consists of the following stages:

## Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

## Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through .SRS. . Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

## Stage 3: Designing the product architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

## Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

## Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

## Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as "Software Development Process Models". Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development. But in this project we select water fall model.

## Waterfall Model

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

* **Requirement Gathering and analysis:** **:** All possible requirements of the system to be developed are captured and documented in a requirement specification phase. This is the most crucial part of project because the whole project depends on what we required from this system. In this part, we made our document in which we write down all specifications of our project and full information about hardware/software which will be required for making this project.
* **SystemDesign:** After documenting all gathered information in requirement specifications phase, it’s time to prepare the design of the system. An important consideration while designing parking system has been taken for the use of appropriate technology for the detection of obstacles while capturing a vehicle. There are several technologies available in the market today to solve this problem. These include various types of radar, digital camera, infrared sensors, and ultrasound sensors. Each technology has its advantages and disadvantages and, after careful consideration of all available options, we decided to use infrared sensors in our design of parking system.
* **Implementation:**

1. **Interface**: For implementing the design that we make earlier, I decided to program the entrance interface in **-----** mainly for its graphical environment and ease of use. Alternatively, the interface could have been programmed in JAVA, C++ or any other object oriented programming language.
2. **Code**: An implementation can be done in many ways in which we decide to code it in units form. Then by taking an input from system design, the first phase were developed in small program called a unit, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
3. **Database**: There are several databases are available in the market like mysql, sql, mangoes dB. We decide to go with mangoes because it has some additional features over sql and other database management system and mainly it is easy to use and for future modification in databases.

* **Integration and Testing:**

1. **Integration**: All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
2. **Testing**: Each component will be tested individually before integration.
   * Compile and run Matlab program
   * Testing the interface for all user input possibilities
   * Testing the connection database for all user input possibilities

Test integrated system as each feature is added. First, I will test assuming the ideal case (one car). Next, I will simulate and test the system’s capabilities for guiding multiple cars.

* **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market. It is the responsibility of deployment team to install and configure this system in customer’s pc and train them how to use this system.
* **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

# Description of Hardware System

# 4.1 CCTV / IP / IR Camera

IP camera is used to capture image of car so that we number can be extracted number plate of car.

IP camera, is a type of digital video camera commonly employed for surveillance, and which, unlike analog closed circuit television (CCTV) cameras, can send and receive data via a computer network and the Internet. Although most cameras that do this are webcams, the term "IP camera" or "netcam" is usually applied only to those used for surveillance.

# 4.2 Computer or Laptop

A computer or laptop is required to monitor or run the enitre security system. The hardware requirements which you are need to install this software.

|  |  |
| --- | --- |
| Operating System | Windows 7 + |
| Ram | 1GB |
| Space | 512mb |

# Project Details

# 5.1 Registration panel

# 5.2 Logs

# 5.3 Entries

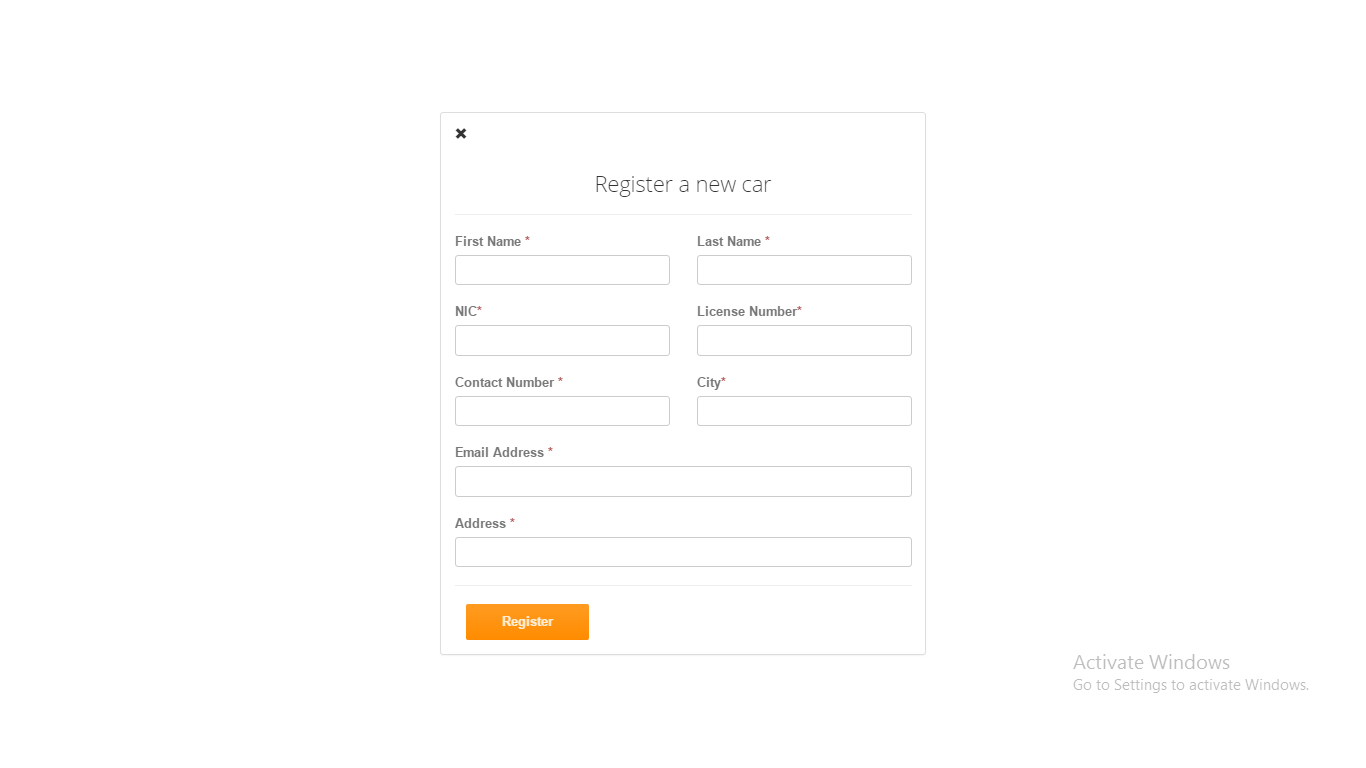
# 5.4 Live Streaming

# 5.5 Notifications

# 5.6 Task runner

# 5.7 Database

# 5.1 Registration Panel



This is the registration panel it used to register new entry of car. All fields are necessary to fill except email. Every entry’s detail sends to that person who monitors this complete system.

**CodeSnippet**

<form action="/" method="POST">

<div class="row top-margin">

<div class="col-sm-6">

<label>First Name <span class="text-danger">\*</span></label>

<input type="text" class="form-control" name="firstname">

</div>

<div class="col-sm-6"><label>Last Name <span class="text-danger">\*</span></label>

<input type="text" class="form-control" name="lastname">

</div>

</div>

<div class="row top-margin">

<div class="col-sm-6">

<label>NIC<span class="text-danger">\*</span></label>

<input type="number" class="form-control" name="nic">

</div>

<div class="col-sm-6">

<label>License Number<span class="text-danger">\*</span></label>

<input type="text" class="form-control" name="licensenumber">

</div>

</div>

<div class="row top-margin">

<div class="col-sm-6">

<label>Contact Number <span class="text-danger">\*</span></label>

<input type="number" class="form-control" name="contactnumber">

</div>

<div class="col-sm-6">

<label>City<span class="text-danger">\*</span></label>

<input type="text" class="form-control" name="city">

</div>

</div>

<div class="top-margin">

<label>Email Address <span class="text-danger">\*</span></label>

<input type="email" class="form-control" name="email">

</div>

<div class="top-margin">

<label>Address <span class="text-danger">\*</span></label>

<input type="text" class="form-control" name="address">

</div>

<hr>

<div class="row"><div class="col-lg-4 text-right">

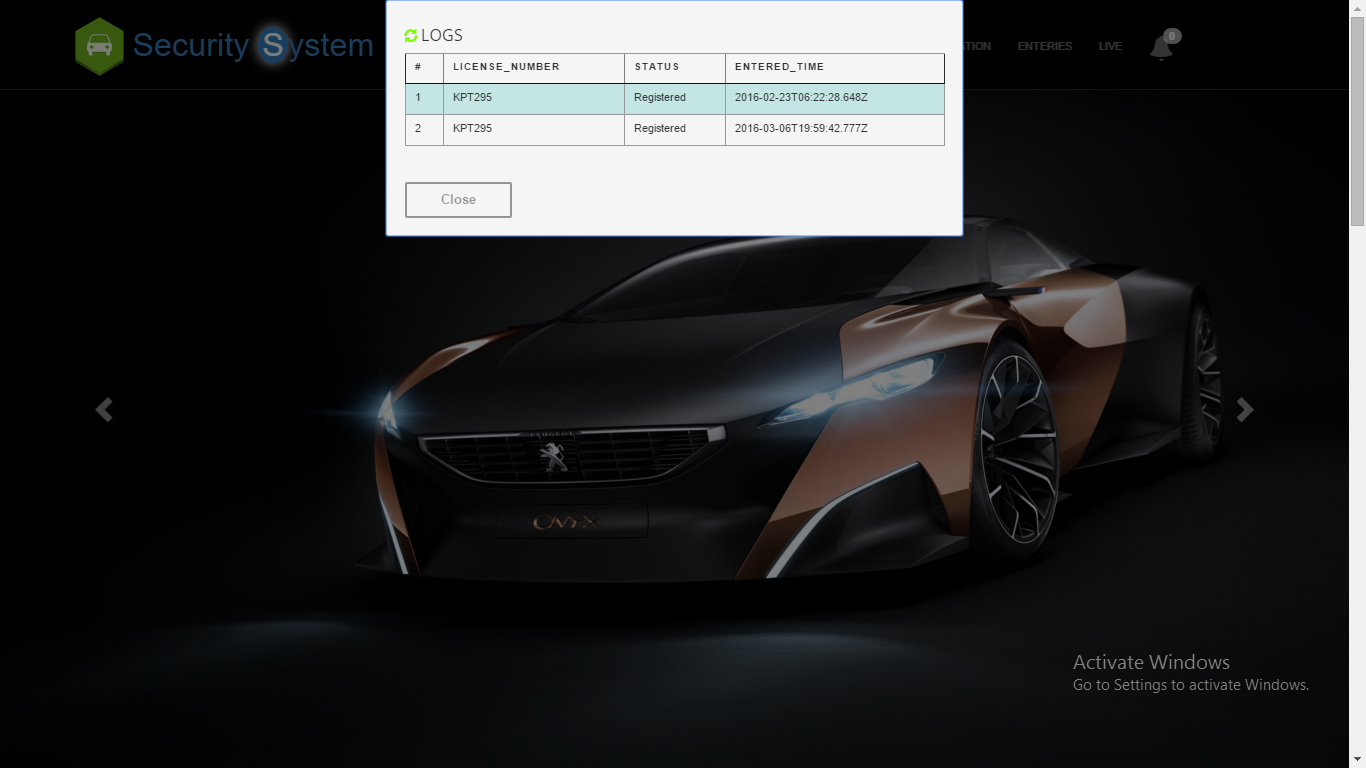
<button class="btn btn-action" type="submit">Register</button>

</div>

</div>

</form>

# 5.2 Logs



This area shows daily entries of cars whether it is register or unregister and these logs are clear automatically after one day.

**CodeSnippet**

<div id="slide" class="well">

<span id="refresh" onclick="ajaxCall()">

<i class="glyphicon glyphicon-refresh"></i>

<h4>LOGS</h4><br/></span>

<table id="hor-minimalist-a" summary="Employee Pay Sheet" class="scroll prettyprint" border="1">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">License\_Number</th>

<th scope="col">Status</th>

<th scope="col">Entered\_Time</th>

</tr>

</thead>

<tbody id='row-data'>

</tbody>

</table>

<br><br>

<button class="slide\_close btn btn-default">Close</button>

<script>

$(document).ready(function () {

$('#slide').popup({

focusdelay: 400,

outline: true,

vertical: 'top'

});

});

var logs= <%- logs %>

var count=[];

for(var i=0;i<logs.length;i++){

if(logs[i].status=="Unregistered"){

count.push(logs[i].status);

}

}

$('#count').append(count.length);

if(logs==" "){}

else{

for(var i=0;i<logs.length;i++){

$('#row-data').append(

"<tr class='even' >"

+"<td>"+(i+1)+"</td>"

+"<td>"+logs[i].plateNumber+"</td>"

+"<td class='logs'>"+logs[i].status+"</td>"

+"<td>"+logs[i].createdOn+"</td>"

+"</tr>")

}

console.log(logs);

}

//setInterval(ajaxCall, 3000); //300000 MS == 5 minutes

function dummy(){console("ok");

}

function ajaxCall() {

document.getElementById('row-data').innerHTML=" "

document.getElementById('count').innerHTML=" "

var data={"number":"AXZ-420","EnteringTime":Date()}

$.ajax({

type: 'GET',

data: data,

cache: false,

contentType: 'application/json',

datatype: "json",

url: '/fetch',

success: function (logs) {

var \_data=JSON.parse(logs);

// console.log( \_data);

//$.notify(\_data,"warning");

for(var i=0;i<\_data.length;i++){

$('#row-data').append(

"<tr class='even' >"

+"<td >"+(i+1)+"</td>"

+"<td >"+\_data[i].plateNumber+"</td>"

+"<td class='logs'>"+\_data[i].status+"</td>"

+"<td >"+\_data[i].createdOn+"</td>"+"</tr>")}

var count=[];

for(var i=0;i<\_data.length;i++){

if(\_data[i].status=="Unregistered"){

count.push(\_data[i].status);

}

}

$('#count').append(count.length);

console.log($('#count').text())

if(($('#count').text())>0){

$('#count').css('background','red');

var audio = document.getElementById("audio");

audio.play();

//alert("Hey..... Unauthorized car has detected take some action", function() { });

} } });}

if(($('#count').text())==0){

$('#count').css('background','#9d9d9d');

}

if(($('#count').text())>0){

var audio = document.getElementById("audio");

//audio.play();

bootbox.alert("Hey..... Unauthorized car has detected take some action", function() { });

}

setInterval(warn,100000);

function warn(){

if(($('#count').text())>0){

$('#count').css('background','red');

var audio = document.getElementById("audio");

audio.play();

bootbox.alert("Hey..... Unauthorized car has detected take some an action", function() { });

}

}

var col=$(".logs");

for(var i=0 ;i<col.length; i++)

{

if(col[i].innerText=='Unregistered'){

var \_i = document.createElement("i");

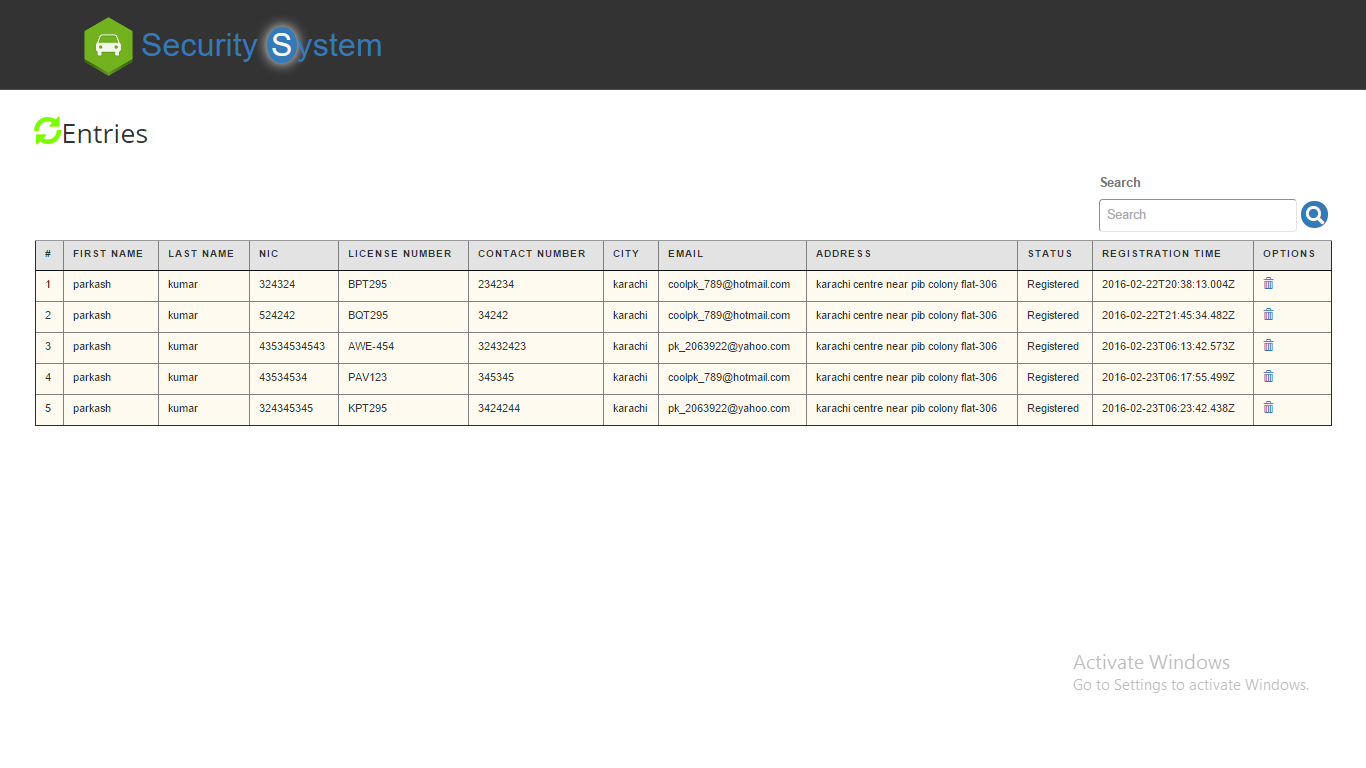
col[i].appendChild(\_i);

//console.log(col[i]);}}

</script>

</script>

# 5.3 Entries



This area shows complete details of register car with some options i.e search and delete user can search specific record and also delete any record

**CodeSnippet**

<div id="fade" class="well1">

<i class='fade\_close glyphicon glyphicon-remove' ></i>

<span id="refresh" onclick="enteriesAjaxCall()">

<i class="glyphicon glyphicon-refresh"></i>

<h4>Entries</h4><br/><span>

<table id="hor-minimalist-a" summary="Employee Pay Sheet" class="scroll prettyprint entriestable" border="1" >

<thead class="entry-head">

<tr>

<th scope="col">#</th>

<th scope="col">First Name</th>

<th scope="col">Last Name</th>

<th scope="col">NIC</th>

<th scope="col">License Number</th>

<th scope="col">Contact Number</th>

<th scope="col">City</th>

<th scope="col">Email</th>

<th scope="col">Address</th>

<th scope="col">Status</th>

<th scope="col">Registration Time</th>

<th scope="col">Options</th>

</tr>

</thead>

<tbody id='row-data-enteries' class="entry-body">

</tbody>

</table>

</div>

<script>

$(document).ready(function () {

$('#fade').popup({

transition: 'all 0.3s',

scrolllock: true

});

});

function enteriesAjaxCall() {

document.getElementById('row-data-enteries').innerHTML=" "

document.getElementById('count').innerHTML=" "

var data={"number":"AXZ-420","EnteringTime":Date()}

$.ajax({

type: 'GET',

data: data,

cache: false,

contentType: 'application/json',

datatype: "json",

url: '/enteries',

success: function (logs) {

var \_data=JSON.parse(logs);

console.log( \_data);

//$.notify(\_data,"warning");

for(var i=0;i<\_data.length;i++){

if(\_data!=" "){

$('#row-data-enteries').append(

"<tr class='even' >"

+"<td>"+(i+1)+"</td>"

+"<td>"+\_data[i].firstname+"</td>"

+"<td>"+\_data[i].lastname+"</td>"

+"<td>"+\_data[i].nic+"</td>"

+"<td>"+\_data[i].licenseNumber+"</td>"

+"<td>"+\_data[i].contactNumber+"</td>"

+"<td>"+\_data[i].city+"</td>"

+"<td>"+\_data[i].email+"</td>"

+"<td>"+\_data[i].address+"</td>"

+"<td>"+\_data[i].status+"</td>"

+"<td>"+\_data[i].createdOn+"</td>"

+"<td><a href='' id="+\_data[i].\_id+" onclick='deleteAjaxCall(id)'><i class='glyphicon glyphicon-trash'></i></a></td>"

+"</tr>")

}

} }

});}

function deleteAjaxCall(id) {

var data={"id":id};

console.log(data);

$.ajax({

type: 'GET',

data: data,

cache: false,

contentType: 'application/json',

datatype: "json",

url: '/delete',

success: function (logs) {

var \_data=JSON.parse(logs);

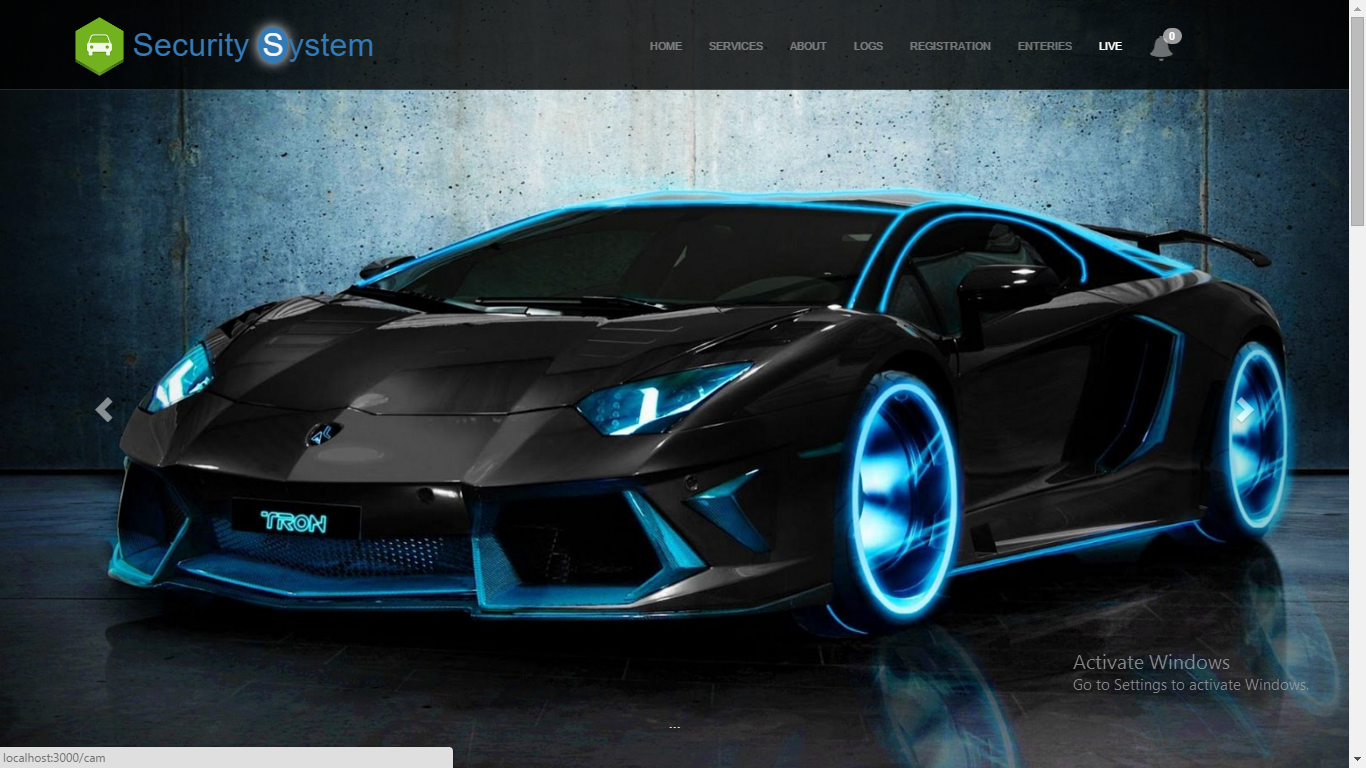
bootbox.alert("Record Deleted", function() { });

}});

}

</script>

# 5.4 Live Streaming



With the help of live streaming any member of security can see the entries of cars

**CodeSnippet**

<!Doctype html>

<html>

<head>

<title>

</title>

<style>

#canvas {

margin-top: 20px;

border: 1px solid #ccc;

display: inline-block;

margin-right:4em;

}

</style>

</head>

<body>

<video id="video" width="640" height="480" autoplay></video>

<canvas id="canvas" width="640" height="480" x="640"></canvas><br/>

<button id="snap">Snap Photo</button>

</body>

<script>

// Put event listeners into place

window.addEventListener("DOMContentLoaded", function() {

// Grab elements, create settings, etc.

var canvas = document.getElementById("canvas"),

context = canvas.getContext("2d"),

video = document.getElementById("video"),

videoObj = { "video": true },

errBack = function(error) {

console.log("Video capture error: ", error.code);

};

// Put video listeners into place

if(navigator.getUserMedia) { // Standard

navigator.getUserMedia(videoObj, function(stream) {

video.src = stream;

video.play();

}, errBack);

} else if(navigator.webkitGetUserMedia) { // WebKit-prefixed

navigator.webkitGetUserMedia(videoObj, function(stream){

video.src = window.URL.createObjectURL(stream);

video.play();

}, errBack);

}

else if(navigator.mozGetUserMedia) { // Firefox-prefixed

navigator.mozGetUserMedia(videoObj, function(stream){

video.src = window.URL.createObjectURL(stream);

video.play();

}, errBack);

}

document.getElementById("snap").addEventListener("click", function() {

context.drawImage(video, 0, 0, 640, 480);

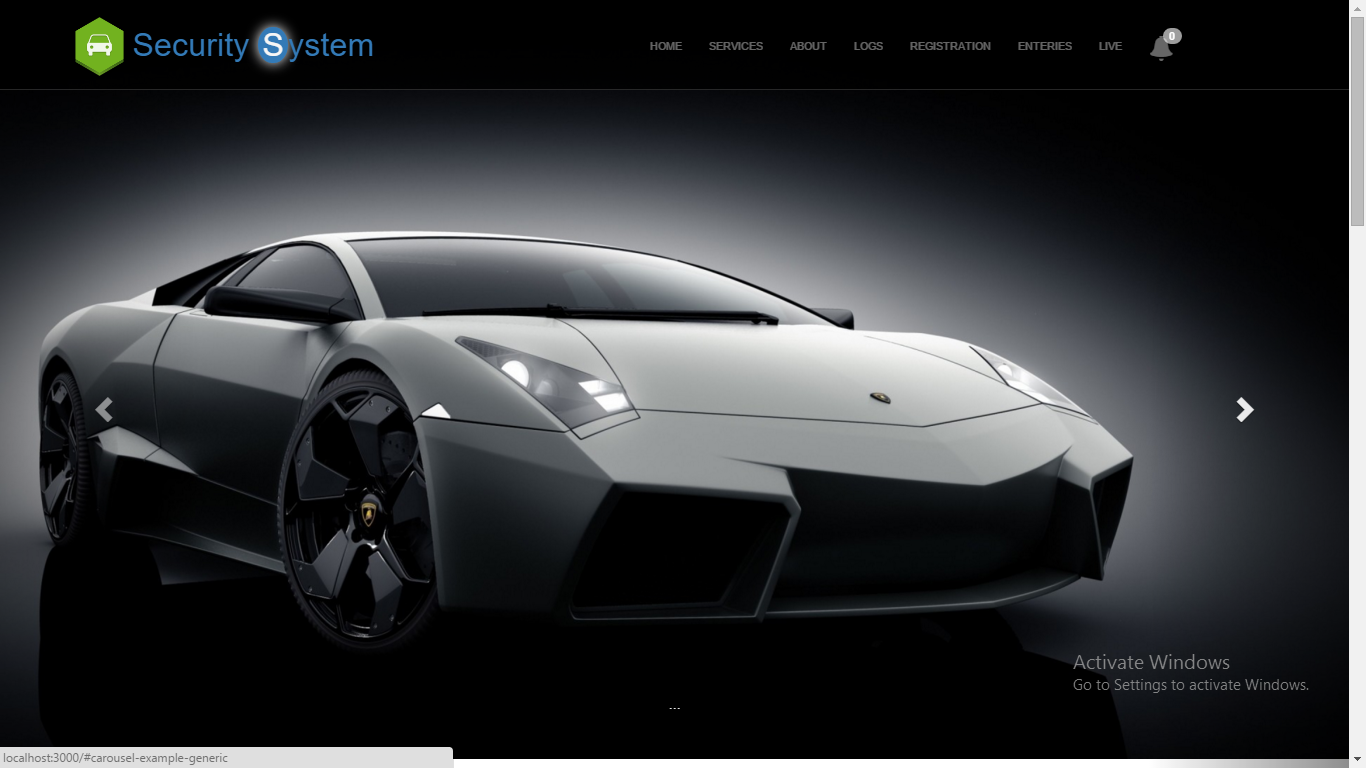
});

}, false);

</script>

</html>

# 5.5 Notifications



This shows the number of unregistered car entered in the parking area by default its color is grey but when unregistered car is detected it turns into red.

**CodeSnippet**

<li><a class="font-horn" ><i class="glyphicon glyphicon-bell" data-toggle="tooltip" data-placement="bottom" title="Unregistered Entries"></i><span class="badge" id="count"></span></a></li>

var count=[];

for(var i=0;i<\_data.length;i++){

if(\_data[i].status=="Unregistered"){

count.push(\_data[i].status);

}

}

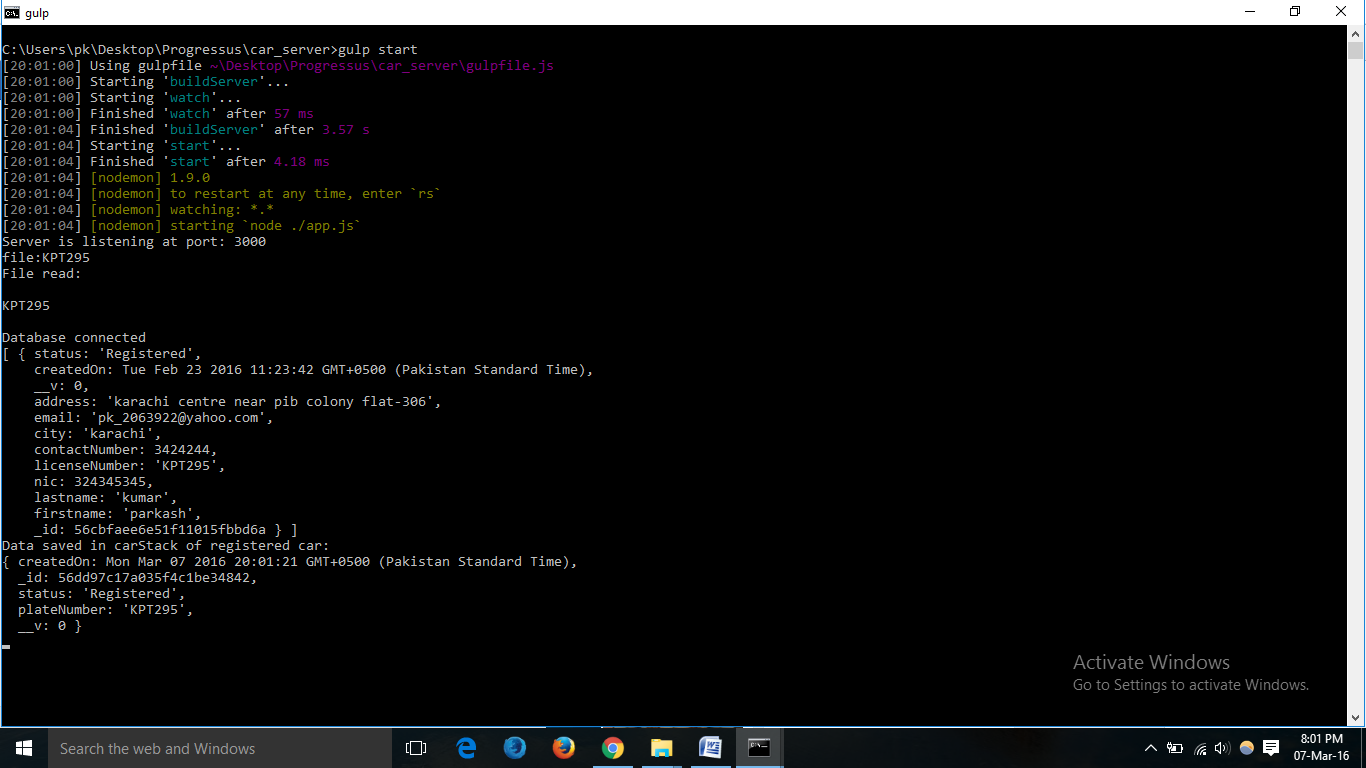
$('#count').append(count.length);

//console.log(data);

console.log($('#count').text())

if(($('#count').text())>0){$('#count').css('background','red');

# 5.6 Task Runner



Gulp is used as a task runner which automatically restarts server and updates database. It is continously watching notepad file where extracted number is stored.

**CodeSnippet**

var gulp = require('gulp');

var ts = require('gulp-typescript');

var rimraf = require('gulp-rimraf');

var nodemon = require('gulp-nodemon');

gulp.task('buildServer', function () {

var tsResult = gulp.src('./app.ts')

.pipe(ts({

module: 'CommonJS'

}));

return tsResult.js.pipe(gulp.dest('./'));

});

gulp.task('start', ['buildServer', 'watch'], function(){

nodemon({

script: './app.js',

text:'./numberPlate.txt'

}).on('restart', function(){

console.log('nodemon restarted server.js');

})

})

gulp.task('watch', function() {

gulp.watch(['./app.ts','./numberPlate.txt'], ['buildServer']);

});

gulp.task('default', ['buildServer']);

gulp.task('open', function(){

var options = {

uri: 'localhost:3000',

app: 'chrome'

};

gulp.src('./app.js')

.pipe(open(options));

});

gulp.task('browser', ['start'], function() {

browserSync.init(null, {

proxy: "http://localhost:3000",

files: ["./\*.txt"],

browser: "google chrome",

port: 5000,

});

});

gulp.task('open', function(){

var options = {

uri: 'localhost:3000',

app: 'chrome'

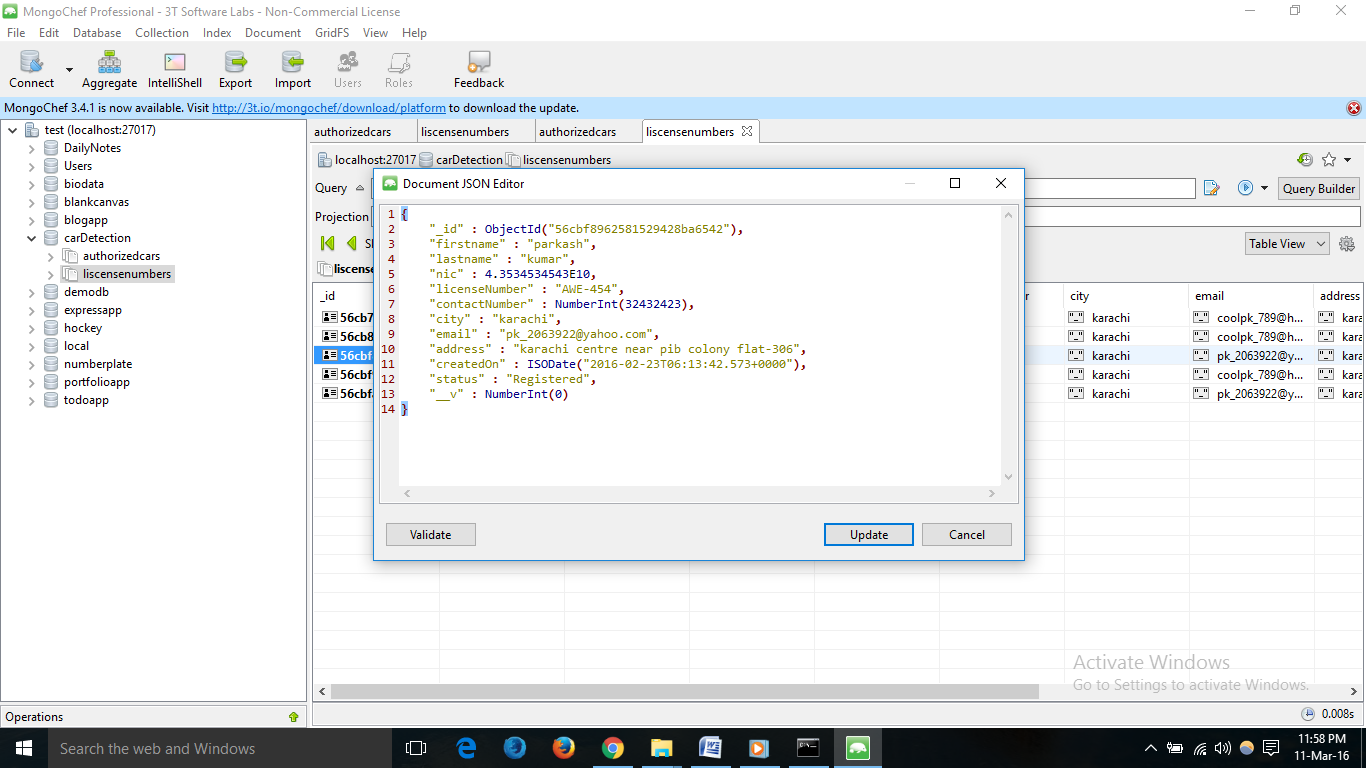
};

gulp.src('./app.js')

.pipe(open(options));

});

# 5.7 Database



Mongodb stores data in JSON format and this is the json of document of registered car.

**CodeSnippet**

var mongoose=require('mongoose');

//connection string

var dburi="mongodb://127.0.0.1/carDetection";

mongoose.connect(dburi);

//verifying connection

mongoose.connection.on('connected',function () {

console.log("Database connected");

});

mongoose.connection.on('error',function (err) {

console.log("Error in db connection:\n"+err);

});

//Database schema

var car=mongoose.Schema({

"firstname":{type:String,required:true},

"lastname":{type:String,required:true},

"nic":{type:Number,required:true,unique:true},

"licenseNumber":{type:String,required:true,unique:true},

"contactNumber":{type:Number,required:true},

"city":{type:String,required:true},

"email":{type:String},

"address":{type:String,required:true},

"status":{type:String,default:"Registered"},

"createdOn":{type:Date,default:Date.now()}

})

var trackCar=mongoose.Schema({

"plateNumber":{type:String},

"status":{type:String },

"createdOn":{type:Date,default:Date.now()} ,})

//Database Model

var liscenseNumber=mongoose.model('liscenseNumber',car);

var AuthorizedCar=mongoose.model('AuthorizedCar',trackCar);

var carStack=new AuthorizedCar({

"plateNumber":num,

"status":"Unregistered",

"createdOn":Date.now() ,

}).save(function(err,data){

var result=data;

if(err)console.log("Error to maintain data of unregistered car: \n"+err);

else{

console.log("Data saved in carStack of unregistered car: \n"+data);

var mailOptions = {

from: "RedZone Security Systems pkbscs67@gmail.com", // sender address

to: "pk\_bscs@yahoo.com", // list of receivers

subject: "Alert...Unregistered car detected", // Subject line

text:"Unregistered car of number is: "+data.plateNumber

// plaintext body

// html body}

// send mail with defined transport object

// send mail with defined transport object

transporter.sendMail(mailOptions, function(error, info){

if(error){

return console.log(error);}

console.log('Message sent: ' + info.response);});}})}

app.post('/',function (req,res) {

var car\_entry=new liscenseNumber({

"firstname":req.body.firstname,

"lastname":req.body.lastname,

"nic":req.body.nic,

"licenseNumber":req.body.licensenumber,

"contactNumber":req.body.contactnumber,

"city":req.body.city,

"email":req.body.email,

"address":req.body.address,

"status":"Registered",

"createdOn":Date.now()

}).save(function(err,data){

if(err)console.log("Data is not inserted: \n"+err);

else{

console.log("Data inserted: \n"+data);

var result=JSON.stringify(data);

var mailOptions = {

from: "RedZone Security Systems pkbscs67@gmail.com", // sender address

to: "pk\_bscs@yahoo.com", // list of receivers

subject: "New Registration", // Subject line

text:"firstname: "+req.body.firstname+"\n"

+"lastname: "+req.body.lastname+"\n"

+"NIC: "+req.body.nic+"\n"

+"licenseNumber: "+req.body.licensenumber+"\n"

+"contactNumber: "+req.body.contactnumber+"\n"

+"city: "+req.body.city+"\n"

+"email: "+req.body.email+"\n"

+"address: "+req.body.address+"\n" }

transporter.sendMail(mailOptions, function(error, info){

if(error){

return console.log(error);

}console.log('Message sent: ' + info.response);});

res.redirect('/'); }})});

var carStack=new AuthorizedCar({

"plateNumber":data[i].licenseNumber,

"status":data[i].status,

"createdOn":Date.now() ,

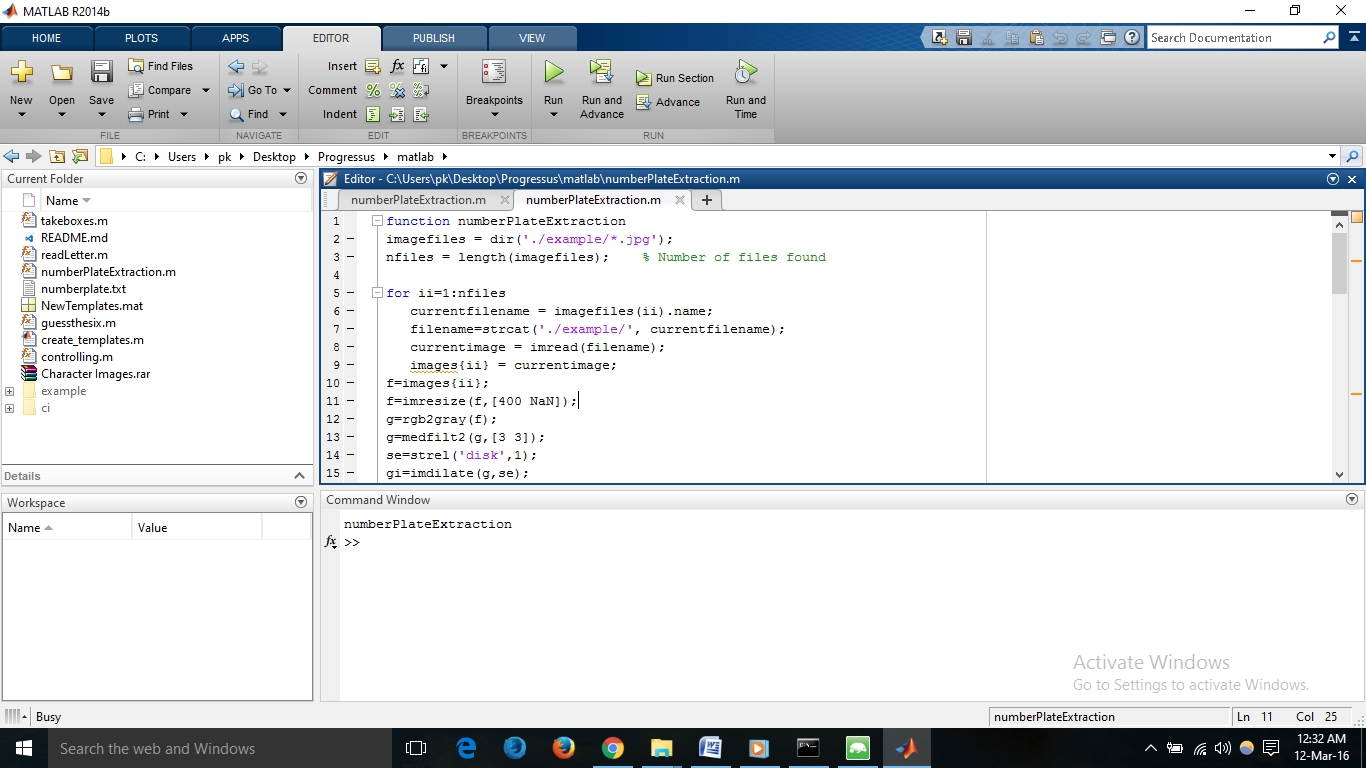
}).save(function(err,data){

if(err)console.log("Error to maintain data of registered car: \n"+err);

else{

console.log("Data saved in carStack of registered car: \n"+data);}})

# 5.8 Image Processing



**CodeSnippet**

function numberPlateExtraction

imagefiles = dir('./example/\*.jpg');

nfiles = length(imagefiles); % Number of files found

for ii=1:nfiles

currentfilename = imagefiles(ii).name;

filename=strcat('./example/', currentfilename);

currentimage = imread(filename);

images{ii} = currentimage;

f=images{ii};

f=imresize(f,[400 NaN]);

g=rgb2gray(f);

g=medfilt2(g,[3 3]);

se=strel('disk',1);

gi=imdilate(g,se);

ge=imerode(g,se);

gdiff=imsubtract(gi,ge);

gdiff=mat2gray(gdiff);

gdiff=conv2(gdiff,[1 1;1 1]);

gdiff=imadjust(gdiff,[0.5 0.7],[0 1],0.1);

B=logical(gdiff);

er=imerode(B,strel('line',50,0));

out1=imsubtract(B,er);

F=imfill(out1,'holes');

H=bwmorph(F,'thin',1);

H=imerode(H,strel('line',3,90));

final=bwareaopen(H,100);

Iprops=regionprops(final,'BoundingBox','Image');

NR=cat(1,Iprops.BoundingBox);

r=controlling(NR);

if ~isempty(r)

I={Iprops.Image};

noPlate=[];

for v=1:length(r)

N=I{1,r(v)};

letter=readLetter(N);

while letter=='O' || letter=='0'

if v<=3

letter='O';

else

letter='0';

end

break;

end

noPlate=[noPlate letter];

end

fid = fopen('../car\_server/numberPlate.txt','w');

fprintf(fid,'\n%s \n\r',noPlate);

fclose(fid);

winopen('../car\_server/numberPlate.txt')

else

fprintf('Unable to extract the characters from the number plate.\n');

fprintf('The characters on the number plate might not be clear or touching with each other or boundries.\n');

end

pause(10);

end

end