**An intelligent, efficient and low cost Parking Solution**

**BITS ZG628T: Dissertation**

By

**Bhuban Mohan Mishra**

Id No. **2013HT13067**

Dissertation work carried out at

**Infosys Ltd., Bhubaneswar**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

**November 2015**

An intelligent, efficient and low cost Parking Solution

BITS ZG628T: Dissertation

By

**Bhuban Mohan Mishra**

Id No. 2013HT13067

Dissertation work carried out at

**Infosys Ltd., Bhubaneswar**

Submitted in partial fulfilment of

**M. Tech. Software Systems**

Degree programme

Under the Supervision of

**Rudra Narayan Rath**

Project Manager

Infosys Ltd., Bhubaneswar

****

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

November 2015

**CERTIFICATE**

This is to certify that the Dissertation entitled “**An intelligent, efficient and low cost Parking Solution**” and submitted by **Bhuban Mohan Mishra** having ID-No. **2013HT13067** for the partial fulfilment of the requirements of **M. Tech. Software Systems** degree of BITS; embodies the bonafide work done by him under my supervision.

|  |  |
| --- | --- |
|  |  |
|  | Signature of the Supervisor |
| Place: Bhubaneswar  Date: 31st Oct 2015 | **Rudra Narayan Rath**  Project Manager  Infosys Ltd., Bhubaneswar |

**Birla Institute of Technology & Science, Pilani**

**Work-Integrated Learning Programmes Division**

**First Semester 2015-2016**

**BITS ZG628T: Dissertation**

**ABSTRACT**

|  |  |  |
| --- | --- | --- |
| BITS ID No. | : | 2013HT13067 |
| NAME OF THE STUDENT | : | Bhuban Mohan Mishra |
| EMAIL ADDRESS | : | Bhuban.Mishra@Outlook.com |
| STUDENT’S EMPLOYING  ORGANIZATION & LOCATION | : | Infosys Ltd., Bhubaneswar |
| SUPERVISOR’S NAME | : | Rudra Narayan Rath |
| SUPERVISOR’S EMPLOYING  ORGANIZATION & LOCATION | : | Infosys Ltd., Bhubaneswar |
| SUPERVISOR’S EMAIL ADDRESS | : | Rudra\_Rath@Infosys.com |
| DISSERTATION TITLE | : | An intelligent, efficient and low cost Parking Solution |

Parking of vehicles has always been a manual and time-consuming task. From large retails to corporates that maintain huge parking spaces for their customers and employees find it very hard to provide necessary information to the commuters that would guide them to the available parking space. Commuters usually have to drive through to find the available parking space that is time consuming, costly and tiring. Commuters often miss the nearby parking spot and park at a faraway place that adds to the frustration. Apart from this, people in all major cities face parking problems in crowded and populated areas. Citizens face difficulty in finding a free parking space near places of interest. On the other hand, government, municipal corporations, corporates, retail outlets etc. need to provide work force to manage the parking spaces and collect the requisite fee from the commuters for the service provided. In the effort of creating Smart Cities in India, we need to find smarter ways to address such problems.

The objective of the study is to:

* Come up with an automated approach to finding free parking places at designated parking areas.
* Find a Cost effective solution that can be implemented with minimum/existing infrastructure

The whole purpose of the study is to come up with a cost effective and fully automated solution to ever-increasing problem of parking management.

Leveraging the benefits of Devices connected to each other as Internet of Things along with Cloud Computing environment would help in fully automating this process. Cameras with some intelligence to process images and deriving information on the available parking places is a cost effective solution to manage parking places. The data collected by these smart cameras can then be fed into a cloud-based system that provides information about available parking spaces to the commuters and users. The objective of this study is to create a prototype of such an intelligent system that uses Image Processing to derive information and Cloud Computing to manage and provide parking information to end users.

|  |  |  |
| --- | --- | --- |
| BROAD ACADEMIC AREA OF WORK | : | Cloud Computing |
| KEYWORDS | : | Internet of Things, Cloud Computing, Image Processing, Raspberry Pi |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Signature of Student |  | Signature of Supervisor |
| **Bhuban Mohan Mishra**  31st Oct 2015  Bhubaneswar |  | **Rudra Narayan Rath**  31st Oct 2015  Bhubaneswar |

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to my supervisor Mr. Rudra Narayan Rath for the continuous support during my M. Tech. study and related research, and especially for his patience and motivation. His guidance helped me a lot during the time of research and writing of this dissertation. I could not have imagined having a better advisor and mentor for my M. Tech. study.

I would like to thank my additional examiner Mr. Prabhat Jana for his insightful comments and continuous encouragement. I am also thankful for his hard questions, which incented me to widen my research from various perspectives.

My sincere thanks also goes to my senior managers and leaders of Infosys Bhubaneswar DC for providing me with necessary access and support without which it would not have been possible to conduct this research.

Last but not the least; I would like to thank my family for baring with my busy schedule and supporting me during the M. Tech. programme.

- B.M.M.

Table of Contents

[Chapter 1 : Introduction 3](#_Toc434072180)

[Problem Statement 3](#_Toc434072181)

[Motivation 3](#_Toc434072182)

[Possible Solution 3](#_Toc434072183)

[Objectives 4](#_Toc434072184)

[Deliverables 4](#_Toc434072185)

[Methodology 5](#_Toc434072186)

[Timelines 5](#_Toc434072187)

[Chapter 2 : Background Research 6](#_Toc434072188)

[System Study 6](#_Toc434072189)

[Internet of Things 7](#_Toc434072190)

[Use Case Diagram 8](#_Toc434072191)

[Hardware and Software Requirements 9](#_Toc434072192)

[Chapter 3 : System Design 11](#_Toc434072193)

[Solution Architecture 11](#_Toc434072194)

[Image Processor 11](#_Toc434072195)

[Dashboard and Admin Console 11](#_Toc434072196)

[RESTful Services 12](#_Toc434072197)

[Data Access Layer 12](#_Toc434072198)

[Database 12](#_Toc434072199)

[Detail Design 12](#_Toc434072200)

[Database 12](#_Toc434072201)

[Services 14](#_Toc434072202)

[Admin Console 15](#_Toc434072203)

[Dashboard 15](#_Toc434072204)

[Image Processor 16](#_Toc434072205)

[Deployment Architecture 17](#_Toc434072206)

[Chapter 4 : Prototyping and Coding 18](#_Toc434072207)

[Development Tools 18](#_Toc434072208)

[Visual Studio 2015 Community Edition 18](#_Toc434072209)

[Version Control (Git) 18](#_Toc434072210)

[Database (SQLite) 18](#_Toc434072211)

[Programming Language (Web) 18](#_Toc434072212)

[Programming Language (Services) 19](#_Toc434072213)

[Python 19](#_Toc434072214)

[OpenCV 19](#_Toc434072215)

[RESTful Services 19](#_Toc434072216)

[Database and DAL 19](#_Toc434072217)

[ASP.NET Web API 20](#_Toc434072218)

[Client Interfaces 22](#_Toc434072219)

[Admin Console 23](#_Toc434072220)

[User Dashboard 30](#_Toc434072221)

[Smart Camera Prototype 31](#_Toc434072222)

[Circuit Diagram 32](#_Toc434072223)

[Image Processing 32](#_Toc434072224)

[Chapter 5 : Testing the Prototype 34](#_Toc434072225)

[Unit Testing 34](#_Toc434072226)

[Integration Testing 35](#_Toc434072227)

[Chapter 6 : Implementation 36](#_Toc434072228)

[Chapter 7 : Summary 39](#_Toc434072229)

[Chapter 8 : Directions for future work 40](#_Toc434072230)

[Chapter 9 : References 41](#_Toc434072231)

[Checklist 42](#_Toc434072232)

[Declaration 43](#_Toc434072233)

List of Figures

[Figure 1 Dimensions of IoT 8](#_Toc434070365)

[Figure 2 Use Case Diagram 9](#_Toc434070366)

[Figure 3 Solution Design Diagram 11](#_Toc434070367)

[Figure 4 ER Diagram 13](#_Toc434070368)

[Figure 5 Entity Model 13](#_Toc434070369)

[Figure 6 Parking Controller Class 14](#_Toc434070370)

[Figure 7 Camera Controller Class 14](#_Toc434070371)

[Figure 8 Image Processing 16](#_Toc434070372)

[Figure 9 Deployment Architecture 17](#_Toc434070373)

[Figure 10 EDMX File 20](#_Toc434070374)

[Figure 11 Parking Administration Page 24](#_Toc434070375)

[Figure 12 Add Parking 24](#_Toc434070376)

[Figure 13 Edit Parking 25](#_Toc434070377)

[Figure 14 Delete Parking 25](#_Toc434070378)

[Figure 15 Search Parking 26](#_Toc434070379)

[Figure 16 Camera Administration Page 27](#_Toc434070380)

[Figure 17 Add Camera 27](#_Toc434070381)

[Figure 18 Edit Camera 28](#_Toc434070382)

[Figure 19 Delete Camera 28](#_Toc434070383)

[Figure 20 Search Camera 29](#_Toc434070384)

[Figure 21 About Page 29](#_Toc434070385)

[Figure 22 Dashboard 31](#_Toc434070386)

[Figure 23 Circuit Diagram of Smart Camera 32](#_Toc434070387)

[Figure 24 Azure Publishing Wizard (Page 1) 36](#_Toc434070388)

[Figure 25 Azure Publishing Wizard (Page 2) 37](#_Toc434070389)

[Figure 26 Azure Publishing Wizard (Page 3) 37](#_Toc434070390)

[Figure 27 Azure Publishing Wizard (Page 4) 38](#_Toc434070391)

List of Tables

[Table 1 Timelines 5](#_Toc434070360)

[Table 2 Unit Test Cases 34](#_Toc434070361)

[Table 3 Checklist 42](#_Toc434070362)

List of Acronyms

A

Ajax: Asynchronous JavaScript and XML 18

API: Application Program Interface 19, 20, 23, 30, 41

C

CLR: Common Language Runtime 18

CRUD: Create, Read, Update and Delete 12, 14, 20, 21, 23, 35

CSS3: Cascading Style Sheet v3 18, 22

D

DAL: Data Access Layer 19

DC: Development Center 1, 6, 30

DDL: Data Definition Language 19

DOM: Document Object Model 18

E

ER: Entity Relationship 13

H

HTML: Hyper Text Markup Language 22, 23

HTTP: Hyper Text Transfer Protocol 19, 20, 21, 30

I

IoT: Internet of Things 3, 7, 8, 17, 40

IT: Information Technology 9

J

JSON: JavaScript Object Notation 14, 15, 19

M

MVC: Model View Controller 19

O

OpenCV: Open Source Computer Vision 10, 11, 19, 32, 41

R

RAM: Random Access Memory 9

REST: Representational State Transfer 41

S

SOAP: Simple Object Access Protocol 14, 15

U

UI: User Interface 12, 19, 29

URL: Uniform Resource Locator 21, 22, 33

USB: Universal Serial Bus 31, 32

X

XML: EXtensible Markup Language 15, 19

# : Introduction

## Problem Statement

Parking of vehicles has always been a manual and time-consuming task. From large retails to corporates that maintain huge parking spaces for their customers and employees, find it very hard to provide necessary information to the commuters that would guide them to the available parking space. Commuters usually have to drive through to find the available parking space that is time consuming, costly and tiring. Commuters often miss the nearby parking spot and park at a faraway place that adds to the frustration.

Apart from this, people in all major cities face parking problems in crowded and populated areas. Citizens face difficulty in finding a free parking space near places of interest. On the other hand, government, municipal corporations, corporates, retail outlets etc. need to provide work force to manage the parking spaces and collect the requisite fee from the commuters for the service provided. In the effort of creating Smart Cities in India, we need to find smarter ways to address such problems.

## Motivation

Every day when I start for office from home, the first thing that comes to my mind was, “Will I get a free parking spot near my building?” This was really a pressing issue for all my colleagues. The Development centre being very big and spanning across 27 acres of land, it often becomes difficult to find a free parking spot near my building. This always makes me frustrated especially when I am running late for a meeting.

The overview of Internet of Things (IoT) covered as part of Distributed Computing (SS ZG526) module this year sparked an idea of implementing a Solution for this parking problem. Providing an automated system intelligent enough to derive the number of free parking slots available in a parking area will address the problem of a commuter entering to office.

The data could be stored on cloud to manage it from a central location. This will help us to free our mind from the pre-notional question of searching for a parking spot and rather use it for other valuable and productive work.

## Possible Solution

To address the problem statement, different solutions are already available. The most widely used solution for closed parking is implementation of hardware sensors. A parking space will always considered as occupied based on when a sensor is blocked (by a vehicle or on different conditions like mud/dust over the sensors etc...). Maintaining a sensor is expensive due to installation and maintenance of many such sensors.

An approach involving image processing could address this issue. If we could install cameras that can capture images of parking places and process the same to derive meaningful information by counting the number of vehicles parked in the area, will help us to determine the number of slots available. This solution would involve less cost of operation and maintenance as one camera can cover a larger area compared to the hardware implementation.

## Objectives

The objective of the study is to:

* Come up with an automated approach to finding free parking places at designated parking areas.
* Find a Cost effective solution that can be implemented with minimum/existing infrastructure

The whole purpose of the study is to come up with a cost effective and fully automated solution to ever-increasing problem of parking management.

Leveraging the benefits of Devices connected to each other as Internet of Things along with Cloud Computing environment would help in fully automating this process. Cameras with some intelligence to process images and deriving information on the available parking places is a cost effective solution to manage parking places. Smart cameras can collect this data and feed into a cloud-based system that provides information about available parking spaces to the commuters and users. The objective of this study is to create a prototype of such an intelligent system that uses Image Processing to derive information and Cloud Computing to manage and provide parking information to end users.

## Deliverables

The deliverables of the Parking Management System will include the following:

* A Prototype of a Smart Camera that is capable of processing images locally and deriving free parking spots.
* An admin console to manage the parking data.
* A User dashboard to receive meaningful information from the system.

## Methodology

Project implementation planned in 5 phases:

* Background Research: Identify the difficulties and issues with the existing Parking management and gather requirements for automating the same.
* System Design: Design the architecture that will implement the intelligent solution and help in managing and displaying data.
* Prototyping and Coding: Create a prototype of the smart camera and build the necessary modules.
* Testing the Prototype: Test the prototype and the interaction of the device with various modules.
* Implementation: Deploy the modules on cloud to make it available to desired stakeholders.

## Timelines

Table 1 Timelines

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 10-Aug-15 | 17-Aug-15 | 24-Aug-15 | 31-Aug-15 | 7-Sep-15 | 14-Sep-15 | 21-Sep-15 | 28-Sep-15 | 5-Oct-15 | 12-Oct-15 | 19-Oct-15 | 26-Oct-15 | 2-Nov-15 |
| Background Research |  |  |  |  |  |  |  |  |  |  |  |  |  |
| System Design |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Prototyping and Coding |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing the Prototype |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementation |  |  |  |  |  |  |  |  |  |  |  |  |  |

# : Background Research

The existing parking management system is more of a manual process where the only information available to commuters is the location of parking places. There is absolutely no information provided to the commuters about the availability of free parking places. So the search for a free parking place is a manual process and involves either of the following options.

**Option 1**: Commuter drives around in search of a free parking place in the designated parking area. On finding a parking space, the commuter parks the vehicle.

*Disadvantages*:

* The parking place may not be the nearest one.
* The commuter spends a lot of time in search of the parking spot.

**Option 2**: Commuter takes help from Security Personnel to find a nearest free parking space.

*Disadvantages*:

* The commuter still spends a lot of time, as the security personnel will still search the spot manually.

The purpose of the study is to overcome the disadvantages of the above options and provide an automated approach of searching for a free parking place for commuters. Minimizing the manual interventions will result in faster delivery of service.

The idea behind the solution comprise of designing and developing a prototype with Raspberry Pi and a Camera module that will capture the images, process them to derive useful information regarding available parking spaces. Centralized control centre can perform remote management of the information and the cameras through a Secured Web application. End Users can avail the information though a web or mobile application. The following study enabled to arrive at a solution.

## System Study

Due to the complexity of the project and help bringing in automation to the organization, a study was conducted at Infosys, Bhubaneswar DC to analyse and find the problems of the daily commuters. Based on the discussions with the employees and feedback received, a problem statement was derived.

*Due to the manual nature of the Parking System, it becomes very difficult for the commuters to find a free Parking space that is located near to their work location. Thus, the commuters have to roam around in search of a free parking place that is time consuming and frustrating.*

After discussion with the Facilities team, the conclusion is:

* The proposed system should be automated with minimal to no manual intervention.
* The system should be able to provide relevant information to commuters as well as Staff of Facilities on the Parking Areas.
* The investment on the solution should be minimal, along with software and hardware reuse.

Based on the above requirements, the proposed approach needs to have the following components to provide a cost effective and fully automated solution:

* An intelligent device that makes us of Image Processing from the images received from Cameras placed at designated Parking locations to derive the availability of parking spaces.
* The cameras could send the information processed to a Central Server on Cloud.
* This information could then be presented to End Users through various devices like:
  + Dashboards at designated entry locations
  + Mobile Devices
* An administrator interface to manage the cameras and metadata.

## Internet of Things

Due to the distributed nature of the application and involvement of smart devices / cameras, there is a need for connectivity between these devices. Based on this need, Internet of Things is one of the key ingredients of this solution that will help in bringing more automation, faster processing and real time availability of data to the end users.

The Internet of Things (IoT) refers to uniquely identifiable objects or things and their virtual representations in an Internet like structure. The IoT devices may have the following characteristics:

* The first and important characteristic is being uniquely identifiable.
* Have the capability to dynamically adapt with changing context and take action due to a change in operating environment or user’s context.
* May be self-configuring, by allowing large number of interconnected devices working together to provide certain functionality.
* Integrated into an information network, which allows them to communicate and exchange data between other systems.

In short, an IoT operates in three dimensions.

**Any TIME Connection**

√ On the move

√ Outdoors and Indoors

√ Night-time and Daytime

**Any PLACE Connection**

√ On the move

√ Outdoors (Away from PC)

√ Indoors (At the PC)

**Any THING Connection**

√ Between PCs

√ Human to Human (H2H without a PC)

√ Human to Human (H2H using generic equipment)

√ Thing to Thing (T2T)

Figure 1 Dimensions of IoT

The current system under design fits into the IoT domain as it satisfies all the above characteristics of Anytime, Anything and Anywhere connectivity.

## Use Case Diagram

Based on the discussions during the Requirement Gathering phase, the following are the use cases derived for the new system. The various actors involved are:

* Devices: The smart cameras that derive meaningful information for the system.
* Administrator: Will be responsible for managing the system.
* Commuters: Will be able to get meaningful information from the system.

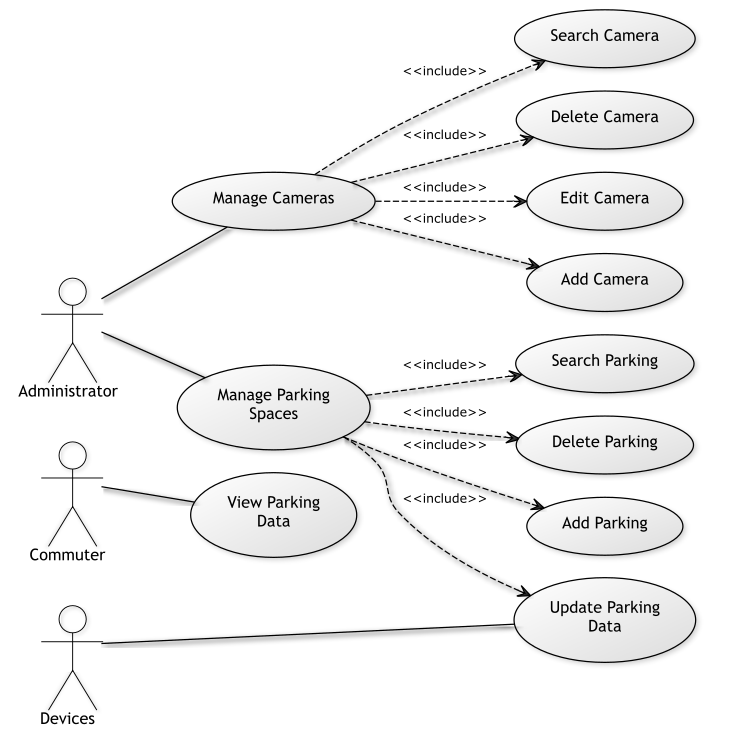


Figure 2 Use Case Diagram

## Hardware and Software Requirements

Considering Infosys being an IT organization, the following environment is assumed available for the study:

* A WiFi network with internet connection.

In order to create the intelligent system, following hardware and software were procured.

**Hardware:**

* Raspberry Pi 2 (Model B, 1GB RAM).
* A MicroSD card to install Operating System for the Pi.
* An old WebCam intended to connect with the Pi for capturing image.
* A WiFi dongle to connect the Pi to a Router.

**Software:**

* Raspbian: A Debian based Open Source Operating System for Raspberry Pi.
* OpenCV for Image Processing with Python.
* A cloud environment with Windows 2012 Operating System for hosting Admin Interface, Application Dashboard and Services.
* Visual Studio for developing Client/Admin interfaces and RESTful Services.

# : System Design

The system was designed to cater to the Requirements identified during the Requirement Elicitation phase and 4+1 Architectural Views were created to define the overall functionality of the System. A high level Solution Architecture is provided below.

## Solution Architecture

Due to involvement of multiple actors and diversity of the solution, a layered architecture is proposed. The various components involved are as follows:

### Image Processor

The image processor is part of the smart device that captures the image from the attached camera and derives meaningful information after processing the image through OpenCV library. The meaningful information is then fed into database through RESTful services.

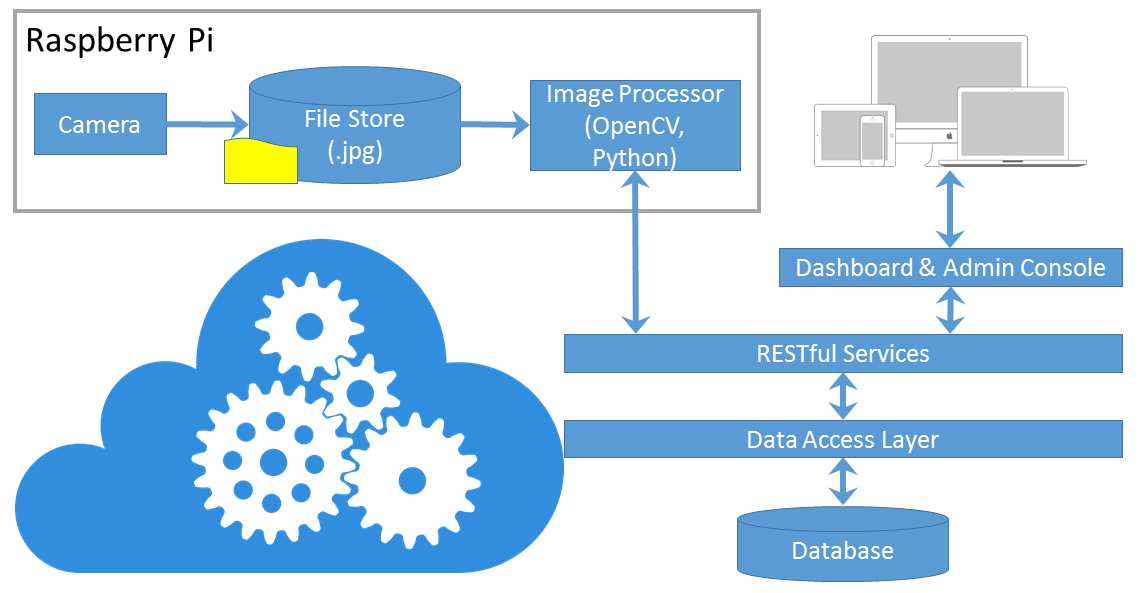


Figure 3 Solution Design Diagram

### Dashboard and Admin Console

In order to manage data on the system, admin console will be helpful for administrators. This will provide an interface accessible through web browsers to add, edit and delete information. Client dashboard is proposed to provide meaningful information to commuters either through mobile devices or through display devices placed at the gates.

These applications are hosted on cloud and interact with the database through the RESTful services.

### RESTful Services

The RESTful services are the backbone of the system as they are the interface to all the actors. The services are hosted on Cloud and provide interfacing between the database and the UI modules.

### Data Access Layer

The data access layer provides CRUD operations for the underlying database. The service interacts with this layer for getting and updating necessary data without knowing the specifics of the underlying database.

### Database

Core of the system that stores all information starting from maintaining the overall parking lots, camera details to occupancy details. The database could range from any cloud based relational database to a flat file based on the need. The system does not care about the storage specifics unless the non-functional requirements like the performance and scalability are taken care. In case there is a need to upgrade, the database could be updated without affecting the more complex UI or Service Layers.

## Detail Design

The following activities were taken up as part of design phase:

* Design of Database for the Parking Management System.
* Design of RESTful Services to perform CRUD Operations on the database.
* Design of Admin Console to manage the System.
* Design of Dashboards to provide parking related information to end users.
* Design of the Image Processing System to derive meaningful information from the image received from Camera.

### Database

As we are concentrating only on the Parking Management Solution, so the User Authentication process has been kept out of scope. Hence the database design only caters to implementing the management of Parking and Camera data.

#### Tables

The tables associated with the solution are:

##### Parking table:

This table stores information regarding the details and attributes of a parking are like the Name, Total number of slots available for parking, the number of slots under maintenance and the number of slots that are currently occupied.

##### Camera table:

The Camera table contains information about the Cameras that are installed in a Parking lot. Each parking lot can be equipped with more than one camera based on the size and requirement.

The ER diagram for the same is detailed out below:

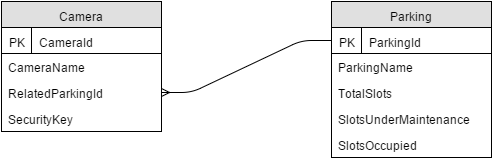


Figure 4 ER Diagram

#### Entity Model

The data layer is created through Entity Framework, an Object Relational Mapping framework based on .NET. Based on the ER diagram, the entity model of the system is derived as follows.

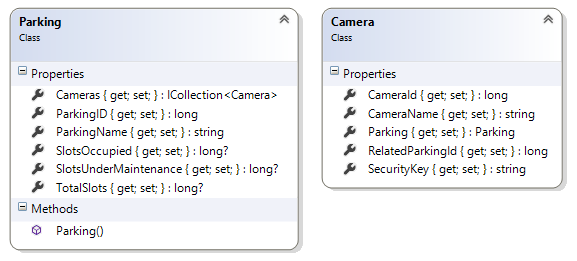


Figure 5 Entity Model

### Services

In order to interact with the database, a Service layer is proposed. The service layer can be used to perform CRUD operations on the underlying database.

The Service layer is designed as a RESTful service. This design decision is taken to enable data manipulation using JavaScript Object Notation (JSON) instead of SOAP.

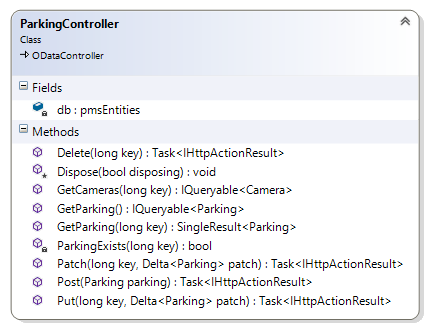


Figure 6 Parking Controller Class

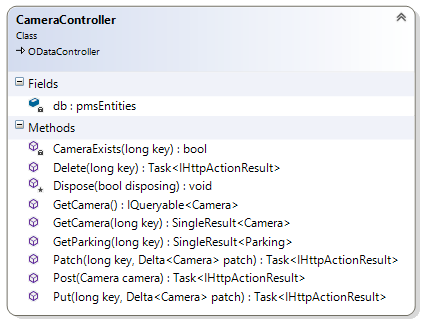


Figure 7 Camera Controller Class

The various advantages of JSON over SOAP are:

* JSON is light weight in comparison to XML that is used by SOAP.
* JSON can be directly mapped to a data structure more easily.
* JSON can be easily used with JavaScript and can be processed faster at client end.
* JSON is more readable than XML.

### Admin Console

The admin console will enable administrators to manage data on the system. The console module will be designed to have two basic functionalities.

#### Parking Data Management

This module will help administrators and support staff to manage information related to parking. The administrators will be able to do the following operations:

* Add a Parking area
* Search for an existing Parking area
* Modify data related to a Parking area
* Delete a Parking area

#### Camera Data Management

This module will help administrators and support staff to manage information related to cameras. The administrators will be able to do the following operations:

* Add a Camera
* Search for an existing Camera
* Modify data related to a Camera
* Delete a Camera

### Dashboard

This module will help the commuters and end users to get useful information regarding the availability of parking spaces. The users will be able to access the page through web browsers. The data would be presented on a map with relevant scale that will help commuters to intuitively figure out the exact spot.

The advantage of designing the module as a web page is:

* Can be made available on mobile devices.
* Dashboards / Monitors could be installed near entry locations to provide live information could be provided through web browsers.
* Publicly available maps could be used to design the dashboard for better information processing.

### Image Processor

The image processor module would help in processing the image fetched from attached cameras. The processing of image would involve the following:

* A base image of free parking slot would be kept as a reference image.
* New images will be captured at regular intervals from the camera module.
* A background subtraction algorithm would be used to get a resulting image that contains the footprint of the vehicles that are parked.
* A blob counting algorithm can then be used on the resulting image to get the total number of vehicles parked in the parking area.

Capture Image from Camera

Background Subtraction



Reference Image

Blob Counting



New Image



Difference Image

Get Count of Vehicles

Figure 8 Image Processing

#### Background Subtraction

Background subtraction, otherwise known as Foreground Detection, is an image processing technique where the foreground of an image is extracted for recognizing objects. The process involves differentiating the current frame from a reference frame.

In other words, a base image is taken and is subtracted from the current image that results in negating the pixels that are present in the reference image. Hence the end product consists of only the difference between both the images. This new image can then be further processed to get the desired objects.

#### Blob Counting

Blob detection is a method that is used for detecting regions in an image that differ in brightness or colour, as compared to surrounding regions.

The differential image derived from the background subtraction process can then be used to detect and count the blobs, thus providing the actual count of the vehicles parked in the parking area.

## Deployment Architecture

The current system under design is considered as a Level-2 IoT system where the device performs the local analysis whereas the data and applications are stored on the cloud. The below diagram describes the deployment architecture of the system.

Device performs

Analysis

Cloud Storage

Device

Resource

Controller Service

Database

RESTful Services

Admin Console / Dashboard

REST

REST

LOCAL

CLOUD

Figure 9 Deployment Architecture

# : Prototyping and Coding

With completing of the design of our smart solution, it’s time to bring it into shape. For developing the various modules, following development tools were used.

## Development Tools

### Visual Studio 2015 Community Edition

Visual Studio provides a robust environment for development of Web as well as desktop applications. The .NET CLR now being Open Source provides much more possibility. The Community edition of Visual Studio, supports many templates and languages for rapid development and testing.

### Version Control (Git)

While writing the code, it is very important to keep a backup of every version of code that is written. There are often times when one version of code does not work with few modifications. Hence a Version Control is essential part of the development cycle. Git is one of the widely used Version Control system and is free. Moreover, it is now supported by Visual Studio and hence an obvious choice.

### Database (SQLite)

SQLite is an open source database that can be used to store relational data on a file system. For the purpose of the solution, as the amount of data we are dealing with is minimal, and for the purpose of hosting the application on Cloud, SQLite will be used for storing the data.

The advantages of using SQLite are:

* It is an embedded, in-memory database engine that reads and writes directly to ordinary disk files.
* It does not have a separate server process.
* It supports multiple tables, indices, triggers, and views.
* The database file format is cross-platform.

### Programming Language (Web)

HTML5, CSS3 and JavaScript are the de facto standard that is supported by all modern browsers. So the admin console and dashboard will be programmed using the same. As the web pages involves calling RESTful services for consuming data, jQuery will be used to perform Ajax operations and manipulating the DOM.

The jQuery Grid plugin will help in providing a grid like interface for the Administration modules. This in turn uses jQuery UI library for providing dynamic UI dialogs and other widgets.

### Programming Language (Services)

The Services will be programmed using C# and ASP.NET Web API. The ASP.NET Web API is a framework based on MVC that can be used for building HTTP services. These services can be consumed by a broad range of clients including desktop browsers, mobiles and tablets. With the help of the MVC features like routing, controllers, action results, filters, model binders and dependency injection, the framework makes a great choice for the Service layer. The use of Web API will help in creating and consuming RESTful services on OData protocol that supports both XML and JSON formats.

### Python

Python is an open source, high level, interpreted language that is widely used in the scientific community. Python has several standard libraries that has wider application in scientific computing, text and image processing.

Raspberry Pi that is being used for prototyping the system, supports Python as one of the programming languages and thus will be helpful in coding the image capturing and image processing modules.

### OpenCV

OpenCV (Open Source Computer Vision) is an Open source image processing framework that provides various algorithms for real time image processing. OpenCV also has a Python wrapper that can be used in conjunction with “numpy” (a highly optimized library for numerical and statistical operations) for robust image processing.

The OpenCV library will be used in the current solution to help in background subtraction and blob counting in the image processing module.

## RESTful Services

### Database and DAL

The database is created with the help of Visual Studio. The DDL for the tables used are as follows:

CREATE TABLE Parking (

ParkingID INTEGER PRIMARY KEY AUTOINCREMENT,

ParkingName TEXT NOT NULL,

TotalSlots INTEGER,

SlotsUnderMaintenance INTEGER,

SlotsOccupied INTEGER

);

CREATE TABLE Camera (

CameraId INTEGER PRIMARY KEY AUTOINCREMENT,

CameraName TEXT NOT NULL,

RelatedParkingId INTEGER,

SecurityKey TEXT,

FOREIGN KEY (

RelatedParkingId

)

REFERENCES Parking (ParkingID)

);

Once the database is created, Entity Framework 6 was used to create the Model (edmx file).

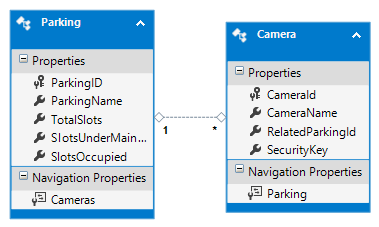


Figure 10 EDMX File

### ASP.NET Web API

Once the Model is ready, ASP.NET Web API Controller classes were generated. There are two Controllers for each of the entities. The details of the controller classes are as follows:

#### Parking Controller

The Parking Controller has the following methods that are used for performing the CRUD operations on the database:

##### GET

public IQueryable<Parking> GetParking()

The method is used for HTTP GET Operation and returns a list of all available parking entities.

**Usage**:

GET: <url>/odata/Parking

public SingleResult<Parking> GetParking([FromODataUri] long key)

The method is used for HTTP GET Operation and returns a single parking entity based on the key.

**Usage:**

GET: <url>/odata/Parking(5)

##### POST

public async Task<IHttpActionResult> Post(Parking parking)

This method is used for inserting or updating data in the database.

**Usage**:

POST: <url>/odata/Parking

##### DELETE

public async Task<IHttpActionResult> Delete([FromODataUri] long key)

This method is used for deleting Parking data from the database.

**Usage:**

DELETE: <url>/odata/Parking(5)

#### Camera Controller

The Camera Controller has the following methods that are used for performing the CRUD operations on the database:

##### GET

public IQueryable<Camera> GetCamera()

The method is used for HTTP GET Operation and returns a list of all available camera entities.

**Usage**:

GET: <url>/odata/Camera

public SingleResult<Camera> GetCamera([FromODataUri] long key)

The method is used for HTTP GET Operation and returns a single camera entity based on the key.

**Usage:**

GET: <url>/odata/Camera(5)

##### POST

public async Task<IHttpActionResult> Post(Camera camera)

This method is used for inserting or updating Camera data in the database.

**Usage**:

POST: <url>/odata/Camera

##### DELETE

public async Task<IHttpActionResult> Delete([FromODataUri] long key)

This method is used for deleting Camera data from the database.

Usage:

DELETE: <url>/odata/Camera(5)

## Client Interfaces

The client interfaces are designed using HTML5, CSS3 and jQuery/JavaScript. The major elements of the pages designed are:

**Header**: This part is designed as a separate HTML page and are dynamically inserted during the page load of other pages through jQuery. This approach helps in using this as reusable page components and thus helps in avoiding repetition. This also helps to integrate some of the common and interactive items on each page like the Menu.

**Body**: The body contains the actual interactive elements of the page.

**Footer**: The footer is also designed as a separate HTML page and are dynamically inserted during the page load.

Dynamic loading of header and footer is achieved by the following jQuery code that is called on the page load event.

function loadHeader() {

$("#pms-header").load("header.html");

}

function loadFooter() {

$("#pms-footer").load("footer.html");

}

$(document).ready(function () {

loadHeader();

loadFooter();

});

Based on the functionality provided, the pages can be classified as follows.

### Admin Console

The main part of the admin console is managing the data through RESTful Service calls exposed by the above Services. For simplifying the development process, a jQuery plugin called jqGrid (jQuery Grid) is used that supports CRUD Operations with a grid like representation.

The HTML element for rendering the grid looks like the following:

<div id="pms-body">

<table id="gridMain"></table>

<div id="pagernav">

</div>

</div>

The major elements for rendering the grid with the help of jqGrid library are the table “gridMain” and the div “pagernav”. On calling the jqGrid API on the “gridMain” table, with relevant RESTful service, the data is rendered in the table section and the pager of the grid is rendered in the “pagernav” section.

The admin console consists of the following pages:

#### Parking Administration

The Parking administration has 5 basic operations that are performed through the page.

* Display Parking data on the Grid
* Add Parking Data
* Edit Parking Data
* Delete Parking Data
* Search Parking Data

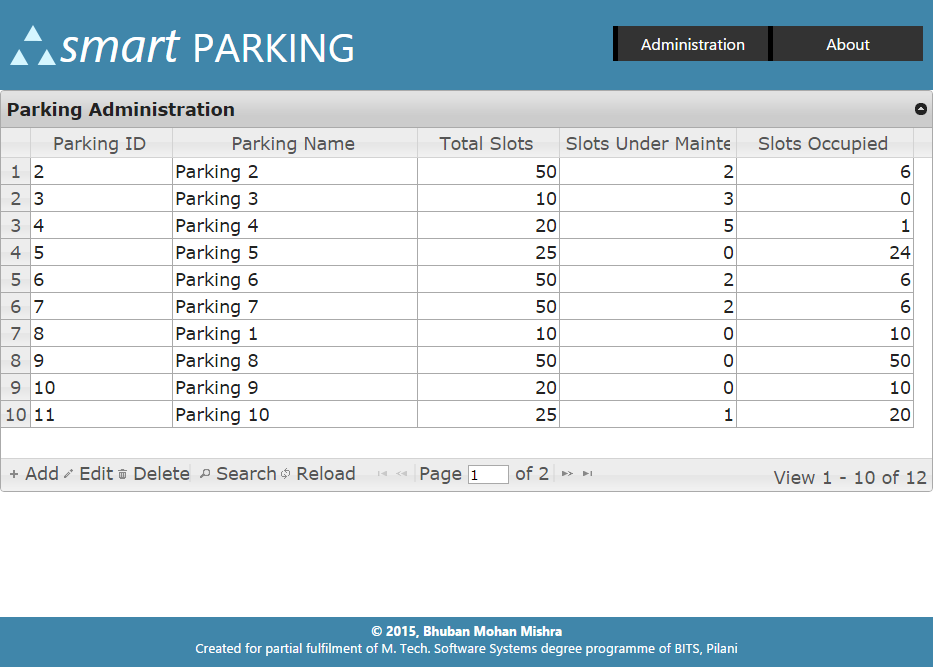


Figure 11 Parking Administration Page

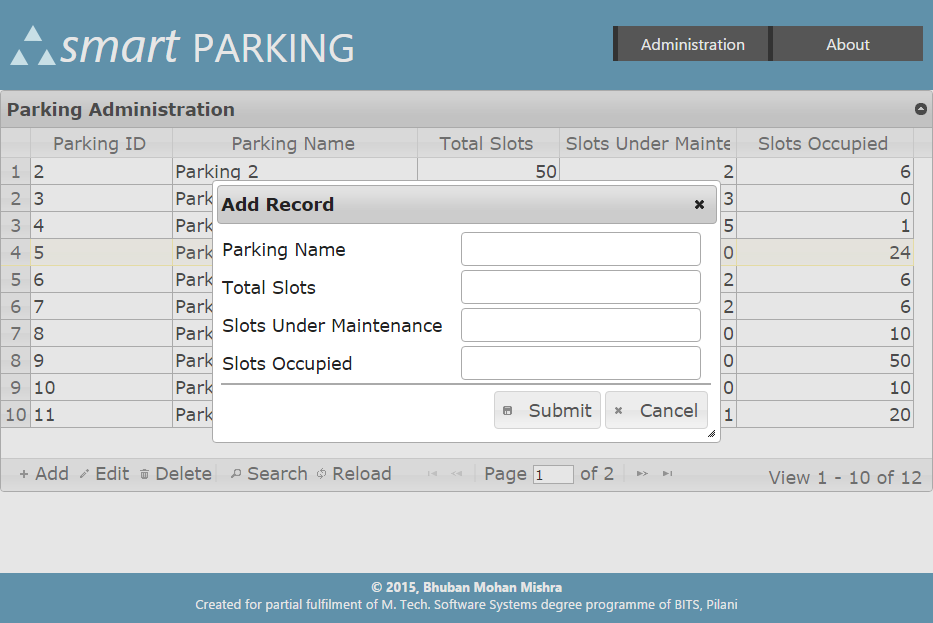


Figure 12 Add Parking

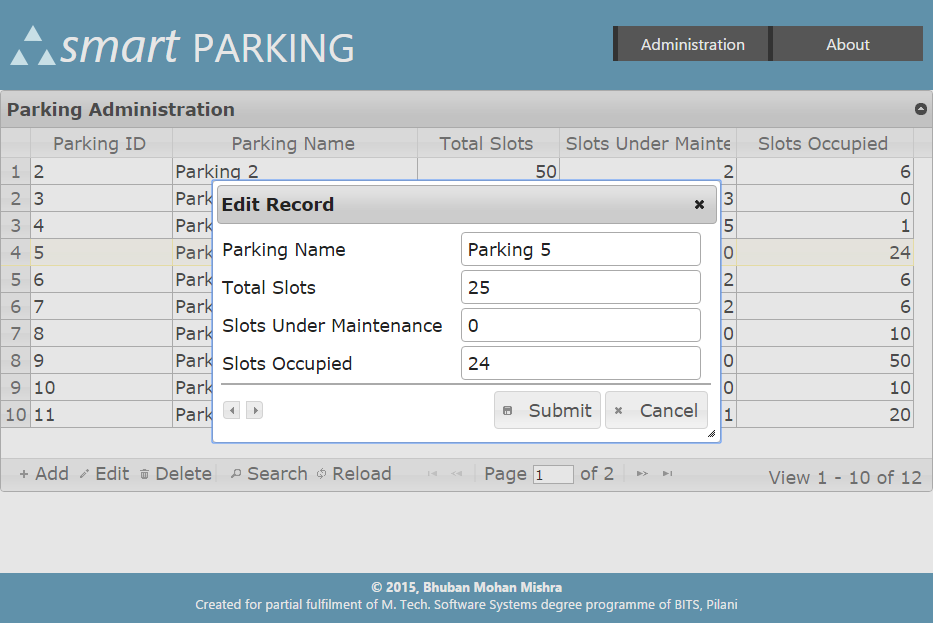


Figure 13 Edit Parking

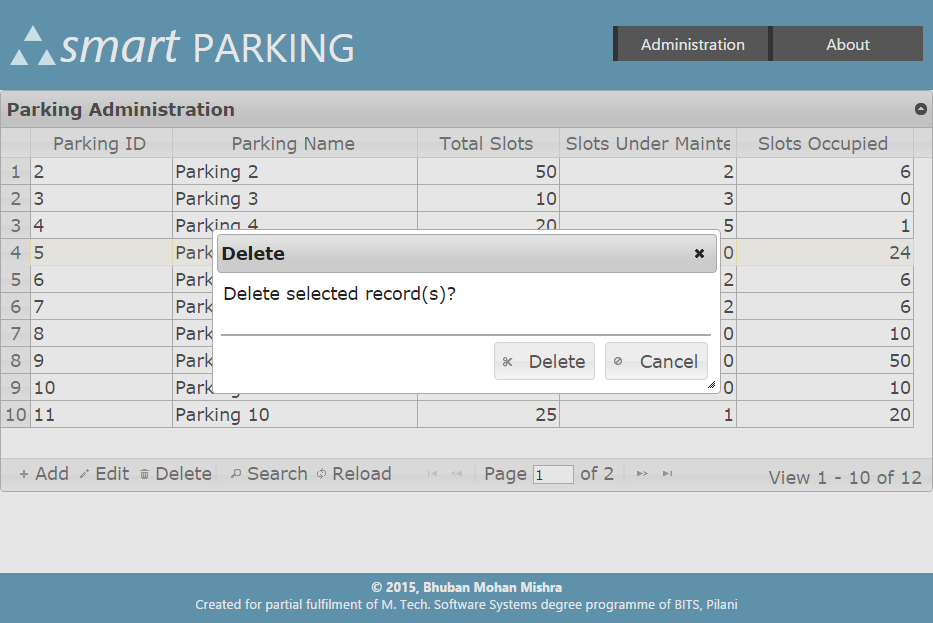


Figure 14 Delete Parking

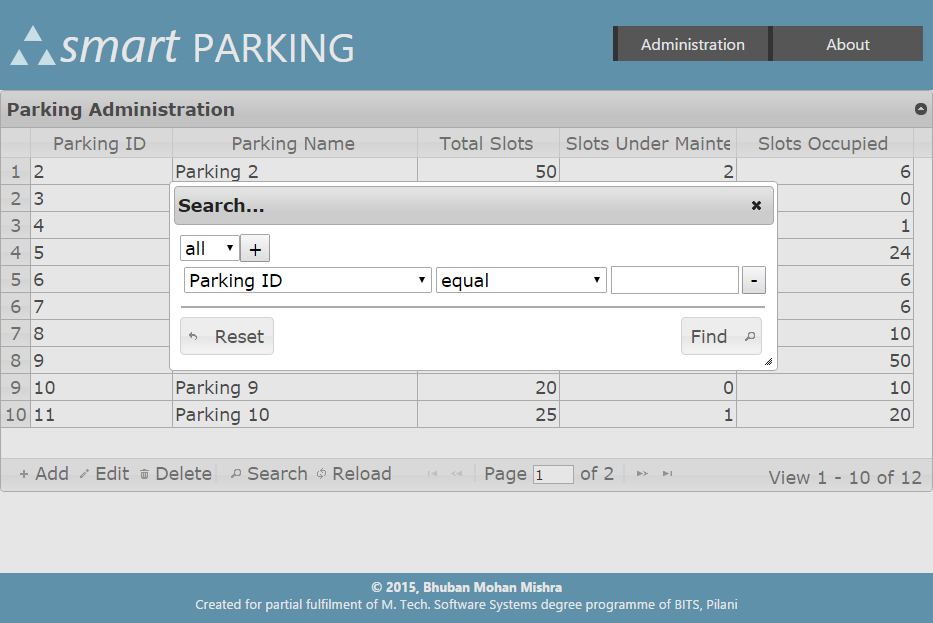


Figure 15 Search Parking

#### Camera Administration

The Camera administration has 5 basic operations that are performed through the page.

* Display Cameras on the Grid
* Add Camera
* Edit Camera
* Delete Camera
* Search Camera

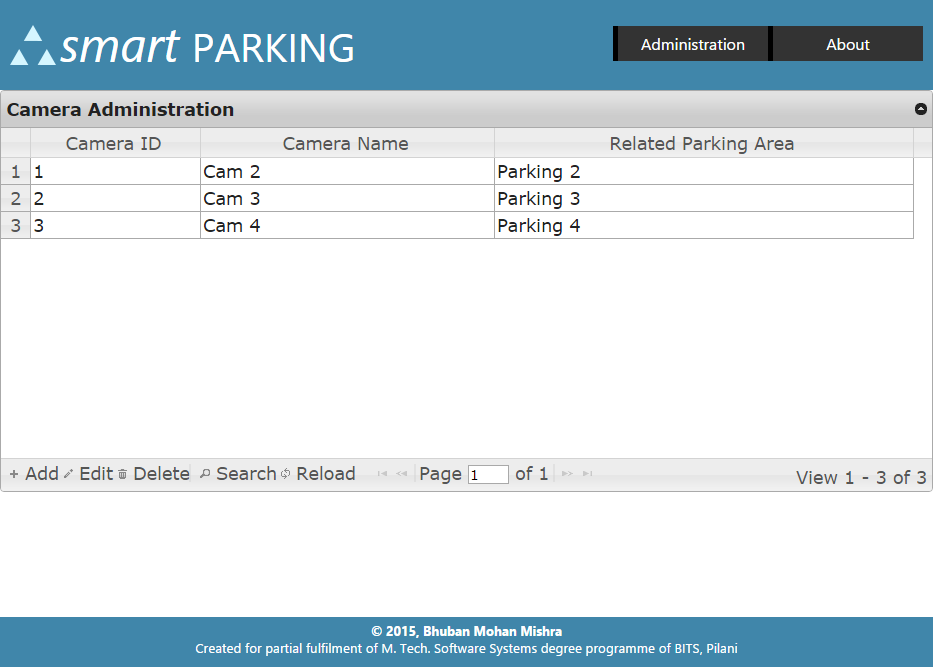


Figure 16 Camera Administration Page

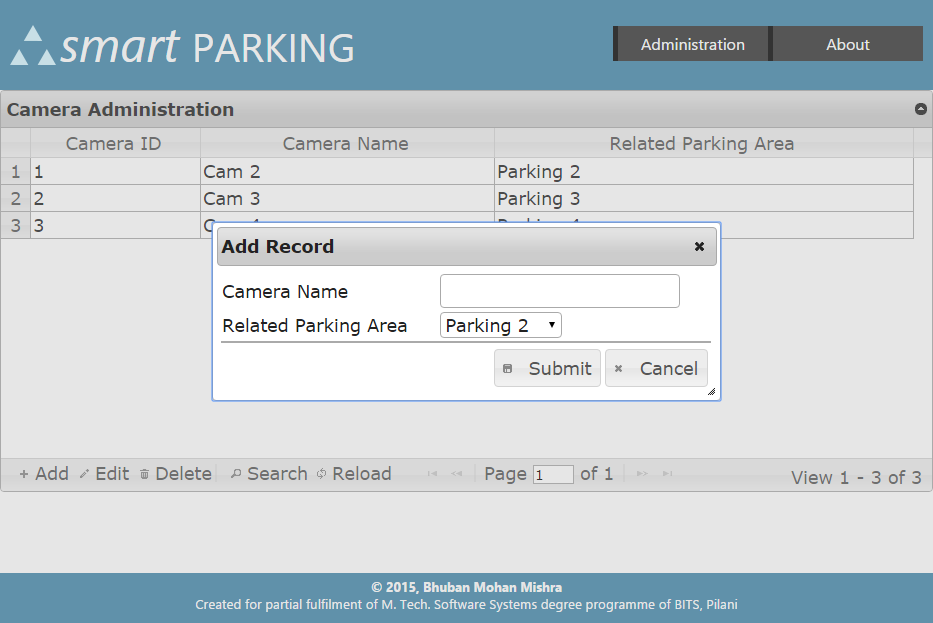


Figure 17 Add Camera

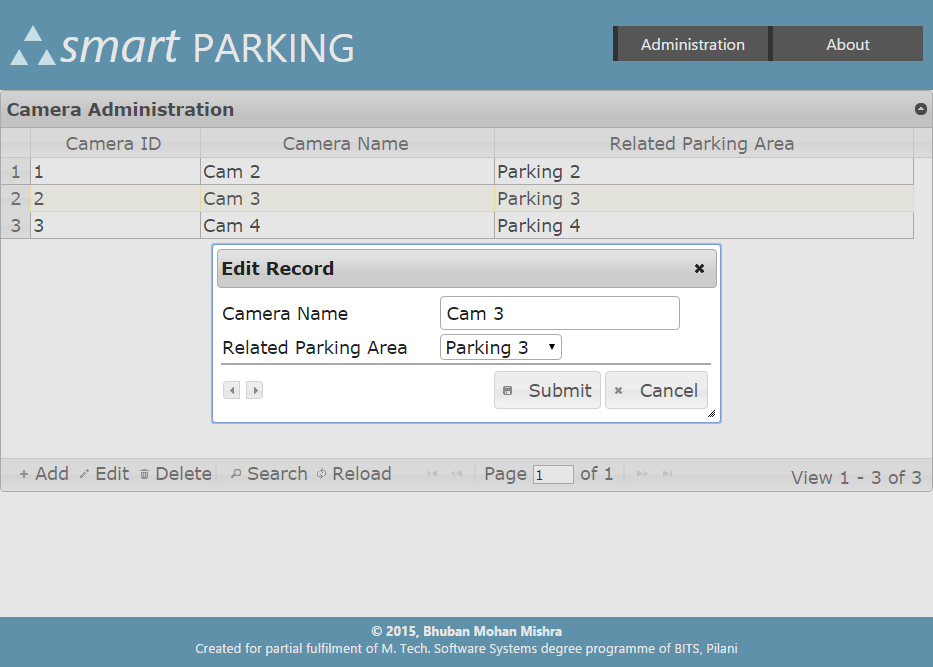


Figure 18 Edit Camera

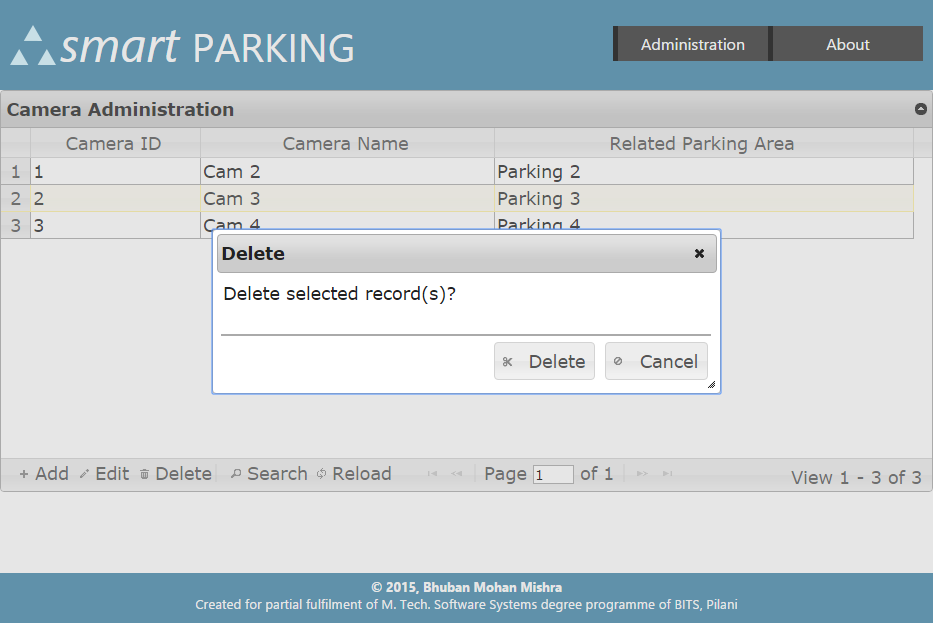


Figure 19 Delete Camera

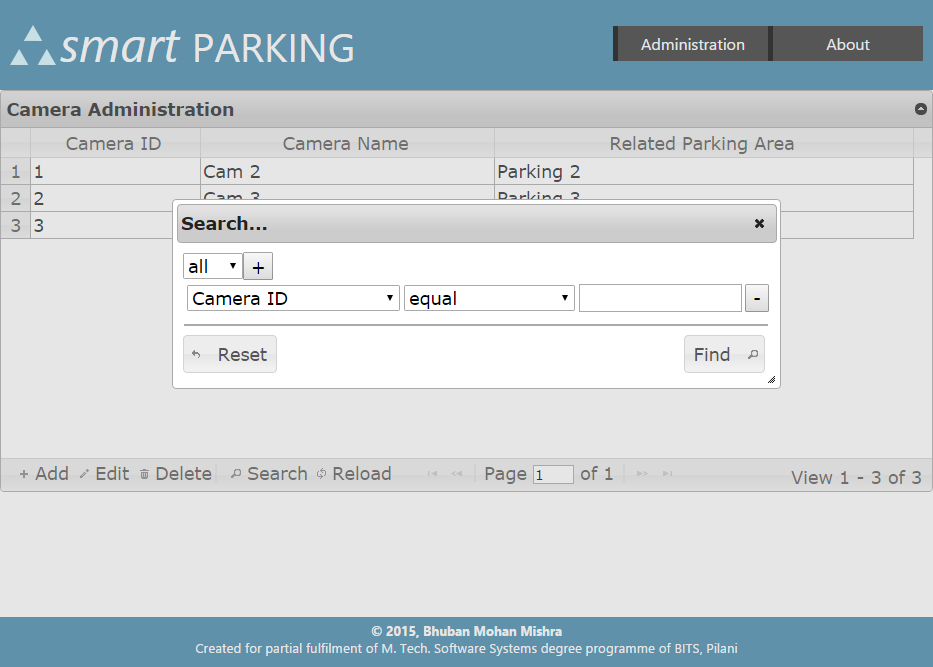


Figure 20 Search Camera

#### About Page

This provides details about the application and is built using jQuery UI dialog.

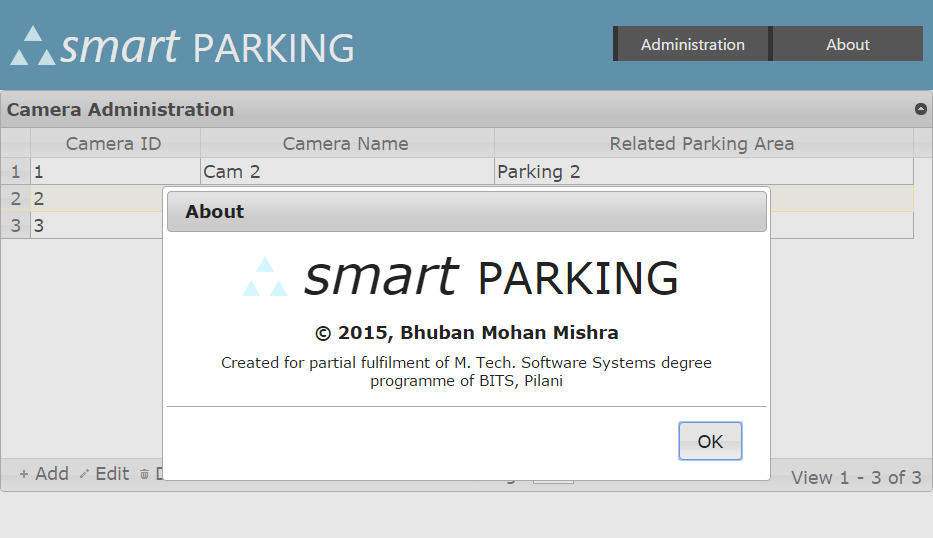


Figure 21 About Page

### User Dashboard

#### Bing Maps

The User dashboard is created with the help of Bing Maps API. The Bing Map API can be used to display a specific location on the map with required zoom level for clarity. The following code was used to render the Infosys Bhubaneswar DC map on the page.

map = new Microsoft.Maps.Map(document.getElementById('divMap'), {

credentials: 'DEVELOPER-KEY',

labelOverlay: Microsoft.Maps.LabelOverlay.hidden,

enableSearchLogo: false,

enableClickableLogo: false,

showDashboard: false,

disablePanning: true,

disableZooming: true,

mapTypeId : Microsoft.Maps.MapTypeId.road,

zoom: 17,

center: new Microsoft.Maps.Location(20.341, 85.8031)

});

#### Pushpins

Pushpins are one of the unique features of Maps API that can be used to highlight specific points on the map. Once the number of available parking spots are fetched through a HTTP GET request from our service, pushpins are generated dynamically on the map.

var gate1 = new Microsoft.Maps.Pushpin(

new Microsoft.Maps.Location(20.34287, 85.8043),

{

width: null,

height: null,

htmlContent: "<div style='font-size:12px;font-border:solid 1px;background-color:LightBlue;width:20px;'>G1</div>"

});

map.entities.push(gate1);

After fetching all relevant information from server and rendering the map, following is the final outcome of the User Dashboard.

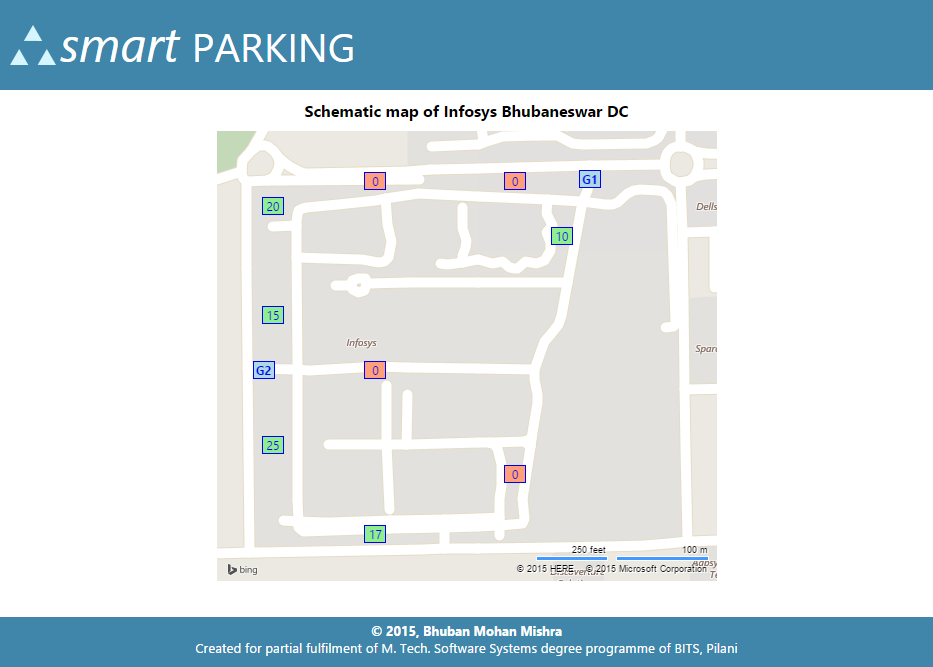


Figure 22 Dashboard

## Smart Camera Prototype

As part of the prototype, following setup was considered:

* A router was provisioned to create a network with access to Internet.
* The Camera module was attached to the Raspberry Pi to one of the USB Port.
* The WiFi Module was attached to the Raspberry Pi to enable connection to the internet.
* The Pi is connected to a 5V Power supply for powering up the Pi.
* Raspbian is the Debian based Linux distro that is installed on the Raspberry Pi to bring it to life.

A circuit diagram of the Prototype is provided below:

### Circuit Diagram

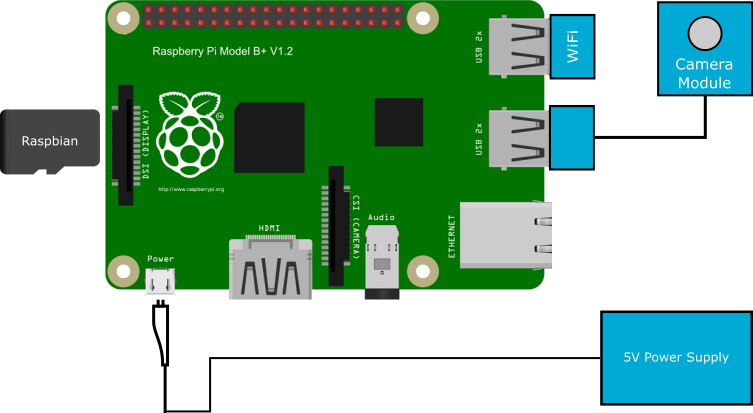


Figure 23 Circuit Diagram of Smart Camera

### Image Processing

The first step of Image Processing is capturing the image from the camera. As we are using a standard USB Webcam, we would need a different program to fetch the images called fswebcam.

$ fswebcam output.jpg

The above command takes a picture and saves it to the current working directory as output.jpg. We can then use a cron to take a picture every minute by creating a shell script and then adding it to the cron table.

We can use the above command to capture the reference image for background subtraction. Once we have the reference image and the new image, we can then use a Background Subtraction algorithm provided by OpenCV to get the differential image.

# Read the reference image

refImg = cv2.imread(refImg.jpg')

# Read the output image

outImg = cv2.imread('output.jpg')

# Create the Background Subtractor

fgbg = cv2.createBackgroundSubtractorMOG()

# Apply the reference image to get the mask

fgmask = fgbg.apply(refImg)

# Apply the output image to get the foreground mask

fgmask = fgbg.apply(outImg)

# Save the differential image

cv2.imwrite('diffImg.jpg', fgmask)

Once the differential image is obtained, we can then use the blob detection algorithm to get the number of vehicles.

# Read image

diffImg = cv2.imread("diffImg.jpg", cv2.IMREAD\_GRAYSCALE)

# Set up the detector with default parameters.

blobDetector = cv2.SimpleBlobDetector()

# Detect blobs.

numBlobs = blobDetector.detect(img)

Once we get the number of blobs or vehicles, we then call the RESTful service to update parking information in the database. The services are called using the urllib and urllib2 modules.

# Import modules

import urllib

import urllib2

# Set the URL

url = '<url>/odata/Parking'

# Prepare data for POST operation

data = urllib.urlencode({

'ParkingId': '5',

'SlotsOccupied': '10'

})

# Call the service

response = urllib2.urlopen(url, data).read()

# : Testing the Prototype

## Unit Testing

Once the system is ready, Unit Testing was performed to ensure all the individual modules are working as expected. The primary goal of performing the Unit testing is to scrutinize the smallest possible unit of code for proper operation.

The unit tests were performed on the following parameters.

Table 2 Unit Test Cases

|  |  |
| --- | --- |
| Module | Unit Test Case |
| Service | Is the Service accessible through Browser |
| Parking Service | Get all Parking Items  Get a single Parking Item |
| Camera Service | Get all Camera Items  Get a single Camera Item |
| Parking Management Page | Get all Parking Items  Search for a Parking Item  Add a Parking Item  Edit a Parking Item  Delete a Parking Item |
| Camera Management Page | Get all Camera Items  Search for a Camera Item  Add a Camera Item  Edit a Camera Item  Delete a Camera Item |
| Dashboard | Get the Map of desired location  Plot all Pushpins on desired location |
| Image Processing | Get an Image  Test Background Subtraction  Test Blob Counting |

For testing the Services, a desktop browser was used, so testing the POST Operation is out of scope. The POST Operations were tested from the web pages.

## Integration Testing

Once the Unit Testing is complete, it is necessary to test the application after integration all the modules. This ensure that all the modules that worked individually well works in the same manner on integration.

The major integration points in the application are:

* Once the image processing is complete and the vehicle count is obtained, it is able to successfully call the web service.
* On calling of the Web Service, the service is able to update the database.
* The Web applications are successfully able to call the Services and perform the desired CRUD operations.
* The Dashboard is successfully able to display the MAP on the page.
* The Map shows the correct information that is received from the RESTful Service.

Once the above integration is tested, the solution is now ready to be deployed on Cloud.

# : Implementation

Once the Unit Testing and Integration Testing is complete, the System was then deployed on the Cloud. Here the entire flow of the application is recreated and the Unit Test Cases were executed to get the actual results.

As Visual Studio has good integration capabilities with Windows Azure, the application was hosted on Azure to perform the Integration Testing. For publishing the web application on Azure, following steps were followed:

1. Right Click on the Project in Visual Studio Solution Explorer and Click on Publish.
2. It opens up a Publish Web Wizard.

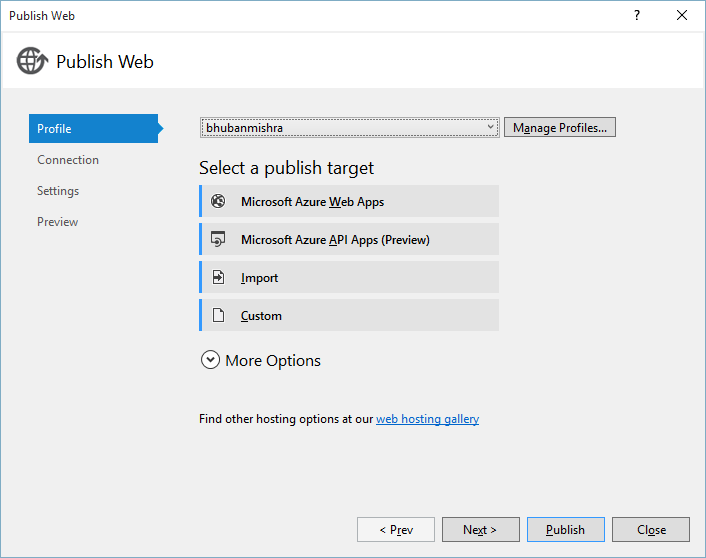


Figure 24 Azure Publishing Wizard (Page 1)

1. Select the Windows Azure Profile that is configured as per the subscription service provided by Microsoft.
2. Select Microsoft Azure Web Apps and click Next, which opens up the connection section.
3. The Publish method is set to Web Deploy. Other Connection parameters are set like the site name and destination URL and click the Next button.

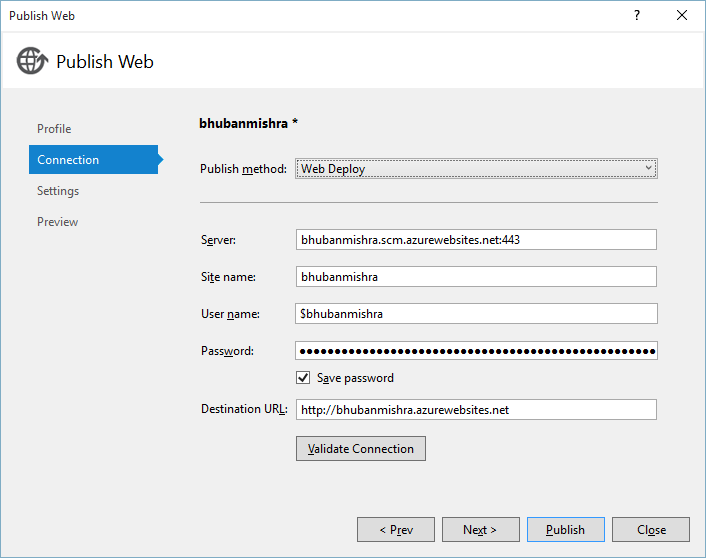


Figure 25 Azure Publishing Wizard (Page 2)

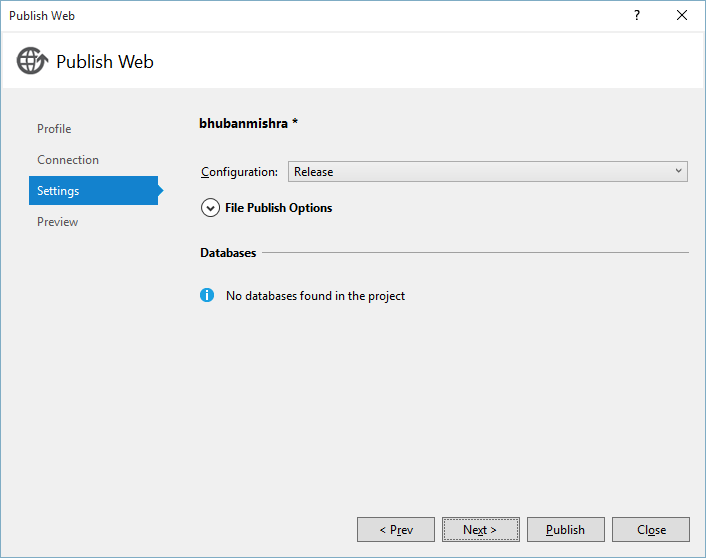


Figure 26 Azure Publishing Wizard (Page 3)

1. This section provides options to deploy the release or debugging options.
2. This also provides the options to set up the database.
3. On clicking Next, the wizard provides an option to preview the changes and then deploy all the files on the azure.

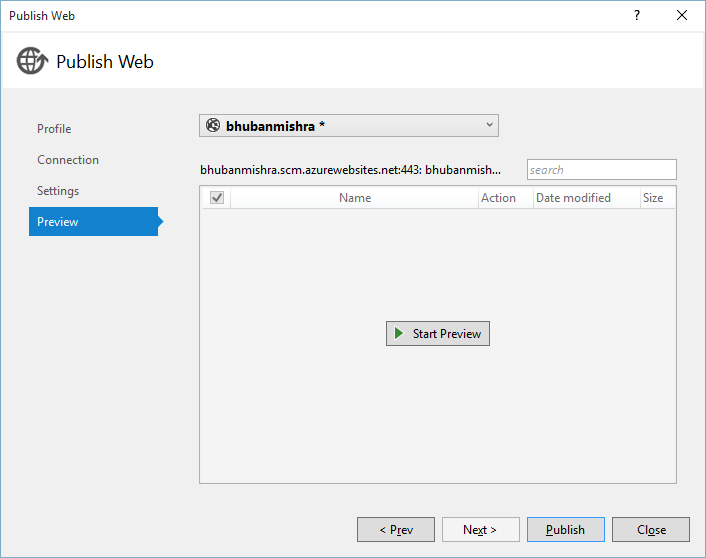


Figure 27 Azure Publishing Wizard (Page 4)

The site can now be accessed from any browser with an internet connection.

# : Summary

In this research, a solution for a common problem of Parking management is proposed that uses image processing techniques for object detection. The current manual system can be highly automated by use of this technique and thus help the common people with the use of technology.

The usage of Smart Cameras could become a boon to the Computer Vision field where such smart devices could be used in other fields for automating various processes. The use of Cloud Computing also helps in managing the devices and data much easier with anytime and anyplace access much easier. The use of maps to display information to end users also makes the system more intuitive as the users will be able to correlate different places and objects making the system even more robust.

# : Directions for future work

This research aimed at providing an automated approach to parking management can be further enhanced by creating an actual smart device based on the proposed prototype. The algorithms used for background subtraction are not tested in different weather conditions, and for open parking spaces weather plays a very important role.

The data available for the Parking System is limited and is considered for a smaller eco-system. The system can be further enhanced to provide better data management capabilities for a wider location like a city or state, thus making the application available to the common man.

Finally, the security part of IoT devices has not been touched in this research and could be a very interesting area of study. With the increasing use of Smart Devices and Cloud Computing, security should be an integral part of the system and will help in preventing misuse of data.

# : References

**BOOKS**:

Vijay Madisetti and Arshdeep Bahga. Internet of Things: A Hands-On Approach. VPT. 2014

Adrian McEwen and Hakim Cassimally. Designing the Internet of Things. Wiley. 2013

Geetha Manjunath and Dinkar Sitaram. Moving To the Cloud: Developing Apps in the New World of Cloud Computing. Elsevier Science. 2011

**WEB LINKS**:

Python Language Reference

https://docs.python.org

Microsoft API and reference catalogue.

https://msdn.microsoft.com/library

OpenCV-Python Tutorials

http://opencv-python-tutroals.readthedocs.org/en/latest/py\_tutorials/py\_tutorials.html

Bing Maps

https://www.bingmapsportal.com/Isdk/AjaxV7

Raspberry Pi Documentation

https://www.raspberrypi.org/documentation/usage

Learn REST

http://rest.elkstein.org/

**PUBLICATIONS**:

Mark Smids. Background Subtraction for Urban Traffic Monitoring using Webcams [master thesis]. Universiteit van Amsterdam, FNWI;2006.

# Checklist

Table 3 Checklist

|  |  |  |
| --- | --- | --- |
|  | Is the final report neatly formatted with all the elements required for a technical Report? | Yes |
|  | Is the Cover page in proper format as given in Annexure A? | Yes |
|  | Is the Title page (Inner cover page) in proper format? | Yes |
|  | (a) Is the Certificate from the Supervisor in proper format?  (b) Has it been signed by the Supervisor? | Yes  Yes |
|  | Is the Abstract included in the report properly written within one page?  Have the technical keywords been specified properly? | Yes  Yes |
|  | Is the title of your report appropriate? The title should be adequately descriptive, precise and must reflect scope of the actual work done. Uncommon abbreviations / Acronyms should not be used in the title | Yes |
|  | Have you included the List of abbreviations / Acronyms? | Yes |
|  | Does the Report contain a summary of the literature survey? | Yes |
|  | Does the Table of Contents include page numbers?   1. Are the Pages numbered properly? (Ch. 1 should start on Page # 1) 2. Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) 3. Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) 4. Are the Captions for the Figures and Tables proper? 5. Are the Appendices numbered properly? Are their titles appropriate | Yes  Yes  Yes  Yes  Yes  Yes |
|  | Is the conclusion of the Report based on discussion of the work? | Yes |
|  | Are References or Bibliography given at the end of the Report?  Have the References been cited properly inside the text of the Report?  Are all the references cited in the body of the report | Yes  Yes  Yes |
|  | Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report. | Yes |

## Declaration

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

|  |  |
| --- | --- |
|  |  |
| **Place: Bhubaneswar** | **Signature of the Student** |
| **Date: 31st Oct 2015** | **Bhuban Mohan Mishra**  **ID No: 2013HT13067** |