**Automatic Vacant Parking Places Management System Using Vehicle Detection**

**OBJECTIVE:**

The main objective of this application is to propose a system for vehicles detection and their corresponding mapping into the parking spots of a parking lot.

**ABSTRACT:**

This paper presents a system for vehicles detection and their corresponding mapping into the parking spots of a parking lot. Approaches from the state-of-the art system, which work properly in controlled scenarios, have been validated using small amount of sequences and without more challenging realistic conditions (illumination changes and different weather conditions). On the other hand, most of them are not complete systems, but provide only parts of them, usually detectors. The proposed system has been designed for realistic scenarios considering different cases of occlusion, illumination changes, and different climatic conditions; a real scenario has been targeted with the condition that existing parking security cameras can be used, avoiding the deployment of new cameras or other sensors infrastructures. For design and validation, a new data set has been recorded. The system is based on existing object detectors (the results of two of them are shown) and different proposed post processing stages.

**Keywords: -** Deep learning, Object Detection

**INTRODUCTION**

PARKING lots are a widely used service where a great investment is made every year. The management of these car parks is very expensive and in many cases complex, especially in the case of those that have many places such as airports or large commercial areas. Solving this problem using computer vision promises a number of advantages over intrusive sensors like induction loops or other weight-in-motion sensors. In addition, a vision-based system may provide many value-added services, like parking space guidance and video surveillance. Such systems allow the decongestion of crowded parking areas, directing vehicles to areas with lower occupancy, guiding the vehicles by a faster route.

The previously developed systems are mainly based on image segmentation or machine learning (SVMs, NN) over spot patches, but due to the evolution in the last years of object detection algorithms, it is possible to use the detections of these algorithms for the proper operation of automatic parking management systems.

Surveillance cameras are readily available in most car parking lots, so in many cases the solution is only to adequately process the information available from the already existing cameras, or complete the deployment by adding some cameras to have a full coverage that allows the system to operate.

**LITERATURE SURVEY**

1. **An algorithm for parking lot occupation detection By T. Fabian 2008**

This paper presents unsupervised vision-based system for parking lot occupancy detection. The proposed method exhibit low computation complexity and use just a few frames per minute. Method is based on three main processing stages. In the first section, raw image acquired by camera system is pre-processed. Shadows in the image are significantly attenuated or completely removed. The image distortion is corrected subsequently. In the following step optimal correspondences between one or more stationary cameras and visible parking places are established. During this process occlusions are taken in account. Finally, a parking place status is evaluated. Acquired information about parking lot occupancy can be served to another system, e.g. intelligent transportation system. Experimental results from both artificial and real scenes show promising results even on quite challenging conditions. Nonetheless a scope for improvement of presented method is given.

**Summary**

This paper describes an unsupervised vision-based parking lot occupancy detection system. The proposed solution is simple to implement and uses only a few frames per minute. Three basic processing phases are used in this method. The raw image acquired by the camera system is pre-processed in the first section. Shadows in the image have been considerably reduced or removed entirely. After then, the image distortion is fixed. The next stage is to build optimal correspondences between one or more stationary cameras and visible parking spaces. Occlusions are taken into account throughout this process. Finally, the state of a parking space is assessed. The data collected on parking lot occupancy can be fed into another system, such as an intelligent transportation system. Even under the most difficult situations, experimental data from both fake and actual scenes reveal encouraging results. Nonetheless, the offered strategy has room for development.

1. **Vehicle detection in open parks using a convolutional neural network By Haihui Xie, Qingxiang Wu, Binshu Chen, Yanfeng Chen, Sanliang Hong 2015**

This paper proposed a new vehicle detection algorithm based on a CNN (convolutional neural network), which dedicates to detect and localize vehicles in an open park. After an off-line training the network can fast respond to an input image so that it is suitable for real-time applications and has the potential to use in vehicle park management systems. Firstly, the trained CNN with a defined sliding window is used to search and identify vehicles in open parks. Secondly, a distribution matrix is defined to reflect the density of vehicle distribution, and it is used to remove redundant windows of vehicles to locate a position of vehicle accurately. Compared to other approaches for vehicle detection, the CNN-based approach does not require any engineered features. The proposed algorithm has combined a CNN with the distribution matrix so that the accuracy of the position location has been improved.

**Summary**

This paper proposes a new vehicle detection technique based on a CNN (convolutional neural network) for detecting and localising automobiles in an open parking lot. Following off-line training, the network can respond quickly to an input image, making it ideal for real-time applications and potentially useful in vehicle park management systems. To begin, the trained CNN is used to seek and detect vehicles in open parks using a defined sliding window. Second, a distribution matrix is created to reflect the density of vehicle distribution and is used to remove redundant windows from vehicles in order to precisely determine their position. The CNN-based solution does not require any engineering features, unlike existing systems for vehicle detection. The proposed approach has increased the accuracy of the position localization by combining a CNN with the distribution matrix.

1. **Fast feature pyramids for object detection By Piotr Dollár, Ron Appel, Serge Belongie, Pietro Perona 2014**

Multi-resolution image features may be approximated via extrapolation from nearby scales, rather than being computed explicitly. This fundamental insight allows us to design object detection algorithms that are as accurate, and considerably faster, than the state-of-the-art. The computational bottleneck of many modern detectors is the computation of features at every scale of a finely-sampled image pyramid. Our key insight is that one may compute finely sampled feature pyramids at a fraction of the cost, without sacrificing performance: for a broad family of features we find that features computed at octave-spaced scale intervals are sufficient to approximate features on a finely-sampled pyramid. Extrapolation is inexpensive as compared to direct feature computation. As a result, our approximation yields considerable speedups with negligible loss in detection accuracy. We modify three diverse visual recognition systems to use fast feature pyramids and show results on both pedestrian detection (measured on the Caltech, INRIA, TUD-Brussels and ETH data sets) and general object detection (measured on the PASCAL VOC). The approach is general and is widely applicable to vision algorithms requiring fine-grained multi-scale analysis. Our approximation is valid for images with broad spectra (most natural images) and fails for images with narrow band-pass spectra (e.g., periodic textures).

**Summary**

Rather than being computed explicitly, multi-resolution image features might be estimated by extrapolating from surrounding scales. This fundamental discovery enables us to create object detection algorithms that are as accurate as the best-in-class while also being significantly faster. The computing of features at every scale of a finely-sampled picture pyramid is the computational bottleneck of many current detectors. When compared to direct feature computation, extrapolation is less expensive. As a result, our approximation provides significant speedups with minimal detection accuracy loss. We employ quick feature pyramids to change three different visual recognition systems and show results on both pedestrian detection (tested on the Caltech, INRIA, TUD-Brussels, and ETH data sets) and general object detection (measured on the Caltech, INRIA, TUD-Brussels, and ETH data sets) (measured on the PASCAL VOC). The method is generic and can be used in a variety of vision algorithms that require fine-grained multi-scale analysis. Our approximation works for photos with broad spectra (the vast majority of natural images) but not for images with narrow band-pass spectra (e.g., periodic textures).

1. **Vacant parking space detection based on plane-based Bayesian hierarchical framework By Ching-Chun Huang, Yu-Shu Tai, Sheng-Jyh Wang 2013**

In this paper, we propose a vacant parking space detection system that operates day and night. In the daytime, the major challenges of the system include dramatic lighting variations, shadow effect, inter-object occlusion, and perspective distortion. In the night-time, the major challenges include insufficient illumination and complicated lighting conditions. To overcome these problems, we propose a plane-based method which adopts a structural 3-D parking lot model consisting of plentiful planar surfaces. The plane-based 3-D scene model plays a key part in handling inter-object occlusion and perspective distortion. On the other hand, to alleviate the interference of unpredictable lighting changes and shadows, we propose a plane-based classification process. Moreover, by introducing a Bayesian hierarchical framework to integrate the 3-D model with the plane-based classification process, we systematically infer the parking status. Last, to overcome the insufficient illumination in the night-time, we also introduce a pre-processing step to enhance image quality. The experimental results show that the proposed framework can achieve robust detection of vacant parking spaces in both daytime and night-time.

**Summary**

In this study, we present a day-and-night unoccupied parking space identification system. Dramatic lighting fluctuations, shadow effect, inter-object occlusion, and perspective distortion are among the system's primary issues during the day. Insufficient illumination and difficult lighting circumstances are the biggest problems at night. To address these issues, we offer a plane-based technique that utilises a structural 3-D parking lot model with a large number of planar surfaces. Inter-object occlusion and perspective distortion are handled in part by the plane-based 3-D scene model. On the other hand, we propose a plane-based categorization technique to eliminate the interference of unpredictably changing lighting and shadows. Furthermore, we systematically infer the parking status by developing a Bayesian hierarchical framework to link the 3-D model with the plane-based classification procedure. Finally, to compensate for the lack of illumination at night, we add a pre-processing step to improve image quality. The experimental findings show that the suggested framework can detect vacant parking spaces reliably both during the day and at night.

1. **A surface-based vacant space detection for an intelligent parking lot By Ching-Chun Huang; Yu-Shu Dai; Sheng-Jyh Wang 2012**

We proposed a surface-based vacant parking space detection system. Unlike many car-oriented or space-oriented methods, the proposed system is parking-lot-oriented. In the system, we treat the whole parking lot as a structure consisting of plentiful surfaces. A surface-based hierarchical framework is then proposed to integrate the 3-D scene information with the patch-based image observation for the inference of vacant space. To be robust, the feature vector of each image patch is extracted based on the Histogram of Oriented Gradients (HOG) approach. By incorporating these texture features into the proposed probabilistic models, we could systematically infer the optimal hypothesis of parking statuses while dealing with occlusion effect, shadow effect, perspective distortion, and fluctuation of lighting condition in both day time and night time.

**Summary**

A surface-based unoccupied parking place monitoring system was proposed. The proposed approach is parking-lot-oriented, as opposed to various car- or space-oriented solutions. We treat the entire parking lot as a structure made up of numerous surfaces in the system. The inference of vacant space is then accomplished using a surface-based hierarchical architecture that integrates 3-D scene information with patch-based picture observation. The feature vector of each picture patch is retrieved using the Histogram of Oriented Gradients (HOG) technique to ensure robustness. We could systematically infer the ideal hypothesis of parking statuses while dealing with occlusion effect, shadow effect, perspective distortion, and variation of lighting condition in both day and night time by including these texture elements into the suggested probabilistic models.

1. **Vision-based vehicle surveillance and parking lot management using multiple cameras 2010**

This paper proposes a vision-based vehicle surveillance system for parking lot management in outdoor environments. Due to the limited field of view of camera, this system uses multiple cameras for monitoring a wide parking area. Then, an affine transformation is used for merging the scenes obtained from these multiple cameras. Two major components are included, i.e., vehicle counting and parking lot management. For the first one, this paper integrates three features, i.e., colour, position, and motion together for well tracking vehicles across different cameras. Thus, even though vehicles are occluded together, they still can be well tracked and identified across different cameras and under different lighting changes. For the second one, we propose a model-based approach to model the colour changes of parking ground for determining whether a parking space is vacant. Due to the perspective effects, the visibility of a parking space is often affected by the vehicle parking on its neighbourhood. To tackle this problem, two geometrical models (ellipses and grids) are proposed for well representing a parking space. Then, with different weights, a hybrid scheme is then constructed for well determining whether a parking space is vacant. The experimental results reveal that our system works well and accurately under different lighting and occlusion conditions.

**Summary**

For parking lot management in outdoor contexts, this study presents a vision-based vehicle surveillance system. Due of the camera's limited field of view, this system employs numerous cameras to monitor a large parking space. The scenes captured by these many cameras are then combined via an affine transformation. Vehicle counting and parking lot management are two important components provided. For the first, this work combines three features, namely colour, position, and motion, in order to effectively track cars across many cameras. Even though vehicles are occluding each other, they may still be tracked and identified across many cameras and lighting conditions. For the second, we offer a model-based strategy for assessing whether a parking space is vacant by modelling the colour variations of the parking ground. Because of perspective effects, the visibility of a parking space is frequently influenced by the vehicles parked in its vicinity. To address this issue, two geometrical models (ellipses and grids) for accurately expressing a parking space are presented. Then, using various weights, a hybrid technique is created for accurately assessing whether a parking place is available. The results of the experiments show that our system performs well and accurately under a variety of illumination and occlusion conditions.

**SYSTEM ANALYSIS & FEASIBILITY STUDY**

**Existing system:-**

The management of these car parks is very expensive and in many cases complex, especially in the case of those that have many places such as airports or large commercial areas. Solving this problem using computer vision promises a number of advantages over intrusive sensors like induction loops or other weight-in-motion sensors. Surveillance cameras are readily available in most car parking lots, so in many cases the solution is only to adequately process the information available from the already existing cameras, or complete the deployment by adding some cameras to have a full coverage that allows the system to operate. The previously developed systems are mainly based on image segmentation or machine learning (SVMs, NN) over spot patches, but due to the evolution in the last years of object detection algorithms, it is possible to use the detections of these algorithms for the proper operation of automatic parking management systems.

**Disadvantages:-**

1. For increasing the coverage area we have to increase the cameras count.

2. Doesn’t work efficiently in all the weather conditions.

**Proposed system:-**

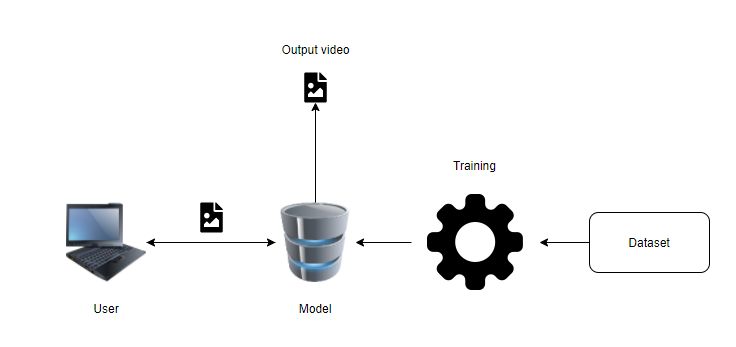
The system has been designed so that existing parking lot security cameras can be used for the proposed system after a simple configuration, without the need for a complete new camera deployment. The designed system faces more complicated scenarios than the ones tackled in the state of the art: almost total occlusions and climatic changes (cloudy scenarios, rain, snow...), that limits/reduces their performance. In this scenario with such a variable background it is not possible to carry out a precise background extraction, nor it’s possible to label and define the region of each place as some parked vehicles completely occlude some of the spots behind them. In addition, the consideration of a scenario, which, as far as we know, has not been reported before for this type of systems, is added.

**Advantages:-**

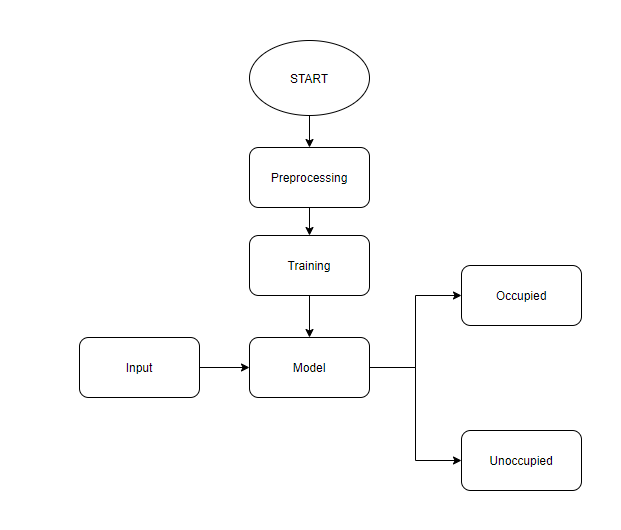
1. For increasing the coverage area no need to add the new cameras simply modify the camera configurations.

2. Works efficiently in all the weather conditions.

**Architecture**

****

**Fig: Architecture**

****

**Fig: Block Diagram**

**METHODOLOGY AND ALGORITHMS:**

1. **Region Based Convolutional Neural Networks**

The problem the R-CNN system tries to solve it is to locate objects in an image (object detection). What do you do to solve this? You could start with a sliding window approach. When using this method you just go over the whole image with different sized rectangles and look at those smaller images in a brute-force-method. The problem is you will have a giant number of smaller images to look at. To our luck, other smart people developed algorithms to smartly choose those so-called region proposals. To simplify this concept:

Region proposals are just smaller parts of the original image, that we think could contain the objects we are searching for.

**Region proposals**

There are different region proposal algorithms we can choose from. These are “normal” algorithms that work out of the box. We don’t have to train them or anything. In the case of this paper, they use the selective search method to generate region proposals. I found a very good and detailed explanation on how, the algorithm works here. But keep in mind:

The R-CNN is agnostic to the region proposal method.

You can choose any method you like and it would work either way.

This will create nearly 2000 different regions we will have to look at. This sounds like a big number, but it’s still very small compared to the brute-force sliding window approach.

**CNN**

In the next step, we take each region proposal and we will create a feature vector representing this image in a much smaller dimension using a Convolutional Neural Network (CNN). Well, this is one fundamental issue with this R-CNN system. You can’t train the whole system in one go (This will be solved by the fast R-CNN system). Rather, you will need to train every part independently. That means that the AlexNet was trained before on a classification task. After the training, they removed the last softmax layer. Now the last layer is the fully connected 4096-dimensional one. This means that our features are 4096 dimensional.

Another important thing to keep in mind is that the input to the AlexNet is always the same (227, 227, 3). The image proposals have different shapes though. Many of them are smaller or larger than the required size. So we will need to resize every region proposal.

**SVM**

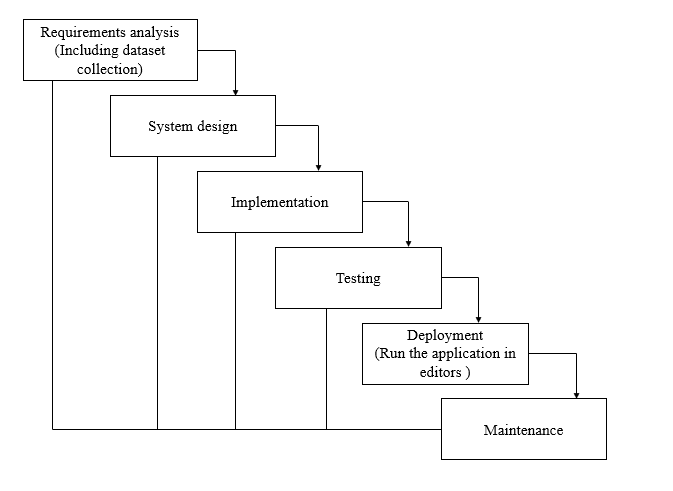
We created feature vectors from the image proposals. Now we will need to classify those feature vectors. We want to detect what class of object those feature vectors represent. For this, we use an SVM classification. We have one SVM for each object class and we use them all. This means that for one feature vector we have n outputs, where n is the number of different objects we want to detect. The output is a confidence score. How confident are we that this particular feature vector represents this class.

**Conclusion**

Another interesting discovery made in this paper is that it was highly effective to pre-train the CNN on a task with a lot of data (for example image classification) and after that to fine tune the network for the actual task, which was the object detection.

**SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:**

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.



**Fig1**: Waterfall Model

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − 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.
* **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.
* **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.

**FEASIBILITY STUDY**

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**Economic feasibility:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### Technical feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM REQUIREMENTS SPECIFICATION**

**Functional and non-functional requirements:**

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

**Functional Requirements**: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she register for the first time on some software system.

**Non-functional requirements**: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.  
They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are > 10000

**SOFTWARE AND HARDWARE REQUIREMENTS:**

Operating system : Windows 7 or 7+

Ram : 8 GB

Hard disc or SSD : More than 500 GB

Processor : Intel 3rd generation or high or Ryzen with 8 GB Ram

Software’s : Python 3.6 or high version, Visual studio, PyCharm.

**SYSTEM DESIGN:**

## Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

### Objectives for Input Design:

The objectives of input design are −

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

**Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

The objectives of input design are:

1. To develop output design that serves the intended purpose and eliminates the production of unwanted output.
2. To develop the output design that meets the end user’s requirements.
3. To deliver the appropriate quantity of output.
4. To form the output in appropriate format and direct it to the right person.
5. To make the output available on time for making good decisions.

**MODULES:**

1. **User**:
   1. **Data gathering:**

Needs to gather the information or data from the open source, this will be use in the train the models.

* 1. **Pre-processing:**

Data need to be pre-processed according the models it helps to increase the accuracy of the model and better information about the data.

* 1. **Feature Engineering:**

In this step features are selected based on the priority of the column data, by this we can reduce the time investing on many columns.

* 1. **Model Building**

To get the final result model building for the dataset is an important step. Based on the dataset we build the model for classification and regression.

* 1. **View Results**

User view’s the generated results from the model.

1. **System**
   1. **Model Checking**

System checks model accuracy and it takes of the necessary for the model building

* 1. **Generate Results**

System takes the input data from the users and produces the output.

**UML DIAGRAMS**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

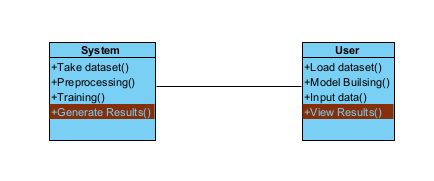
**USE CASE DIAGRAM**

* A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
* Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
* The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



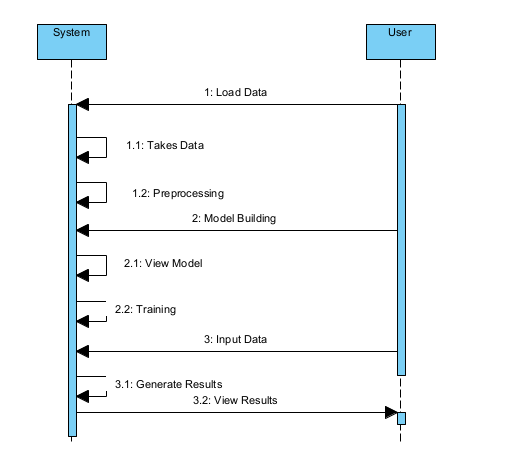
**CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information

\

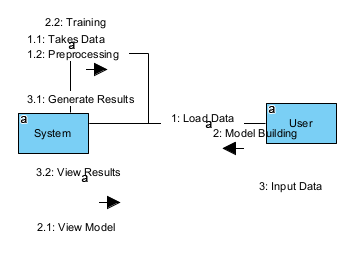
**SEQUENCE DIAGRAM**

* A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
* It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams

****

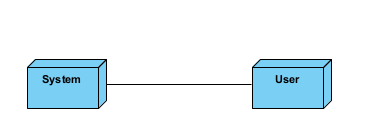
**COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



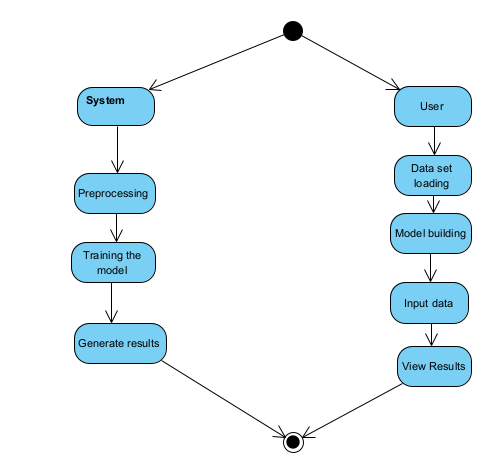
**DEPLOYMENT DIAGRAM**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



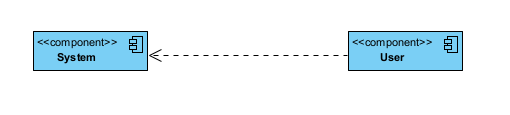
**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**COMPONENT DIAGRAM**:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



**ER DIAGRAM:**

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

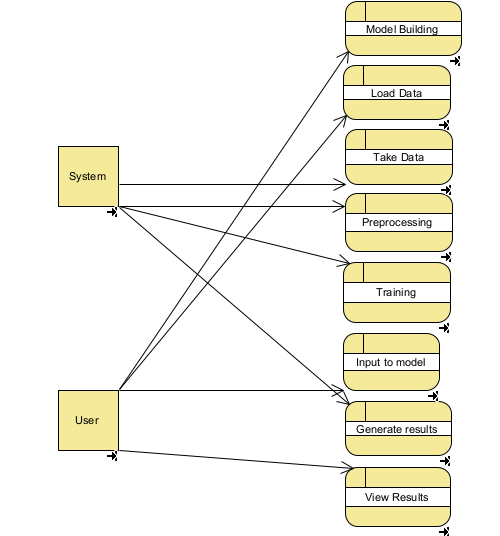
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.

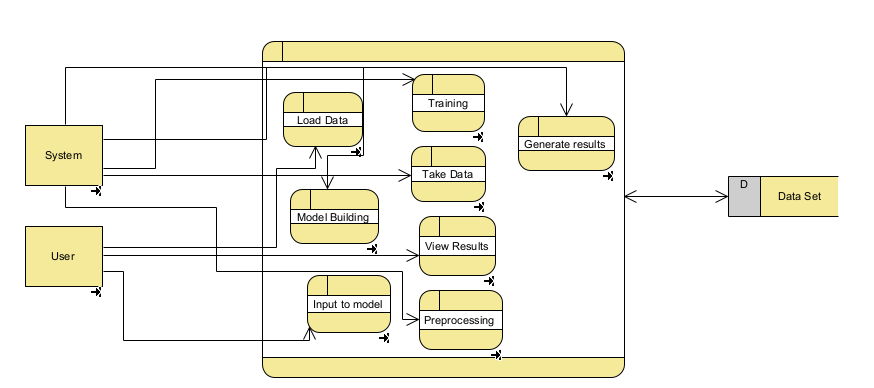
# `

# 

**DFD DIAGRAM:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

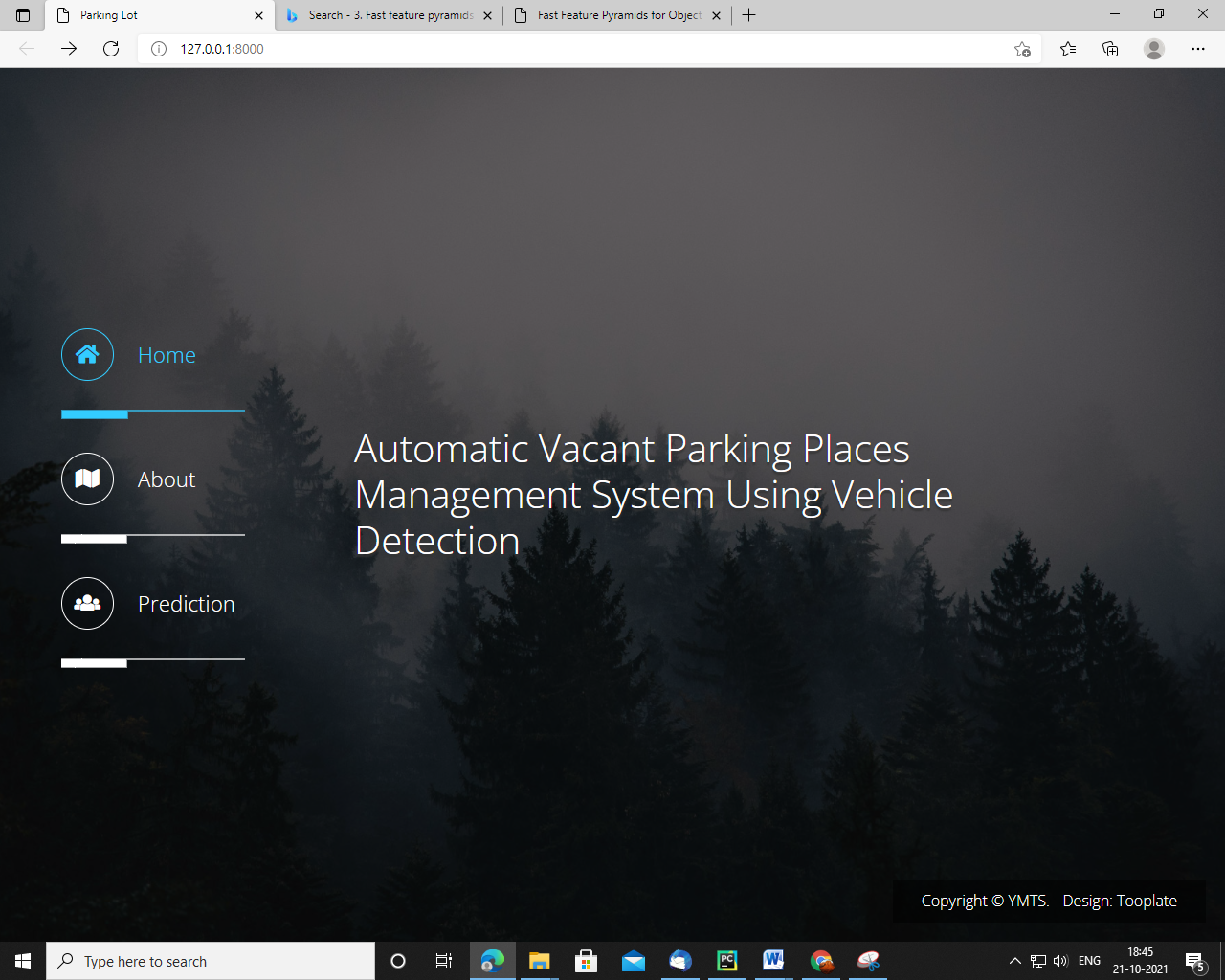




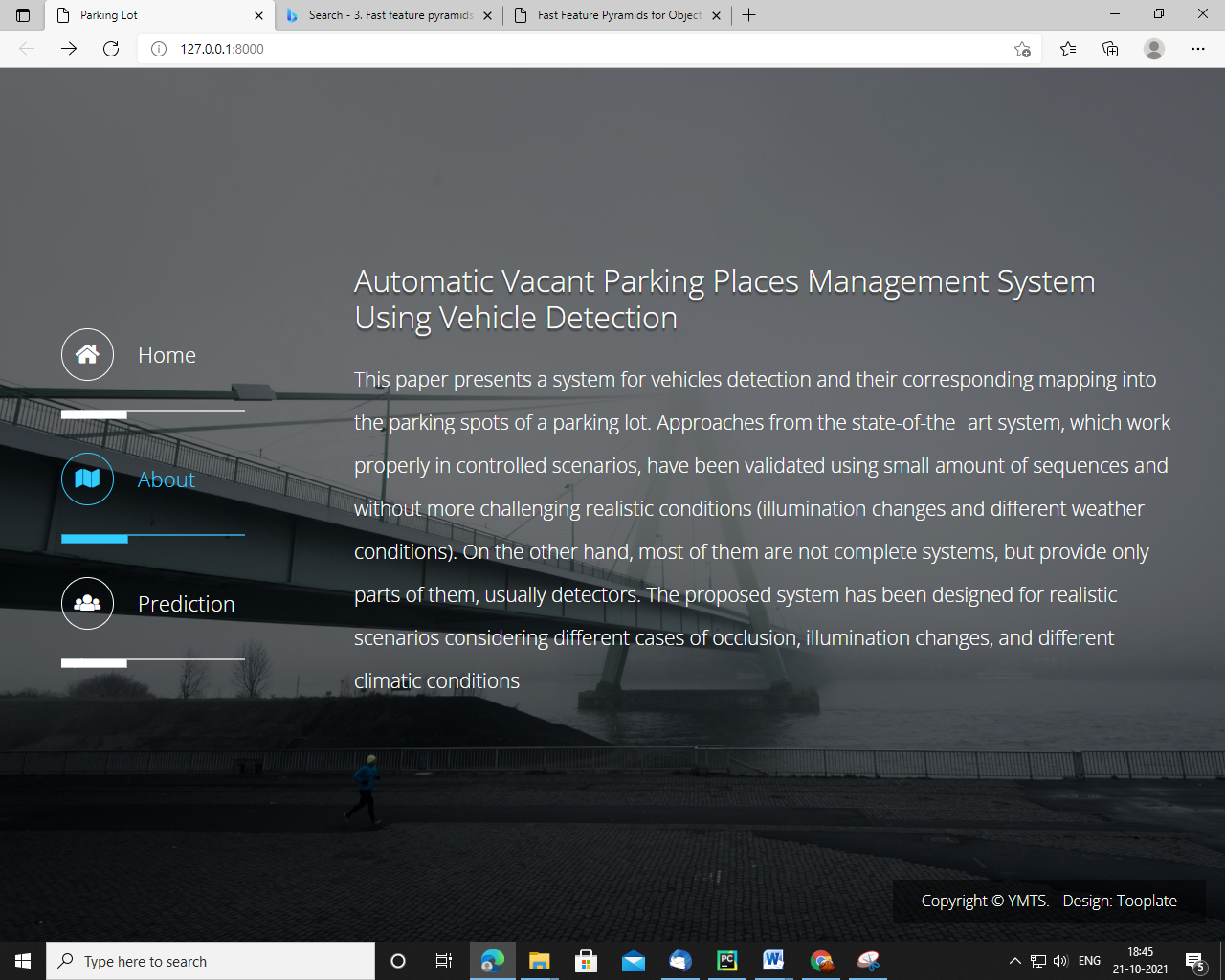
**OUTPUT SCREEN SHOTS WITH DESCRIPTION**

**Automatic Vacant Parking Places Management System Using Vehicle Detection:**

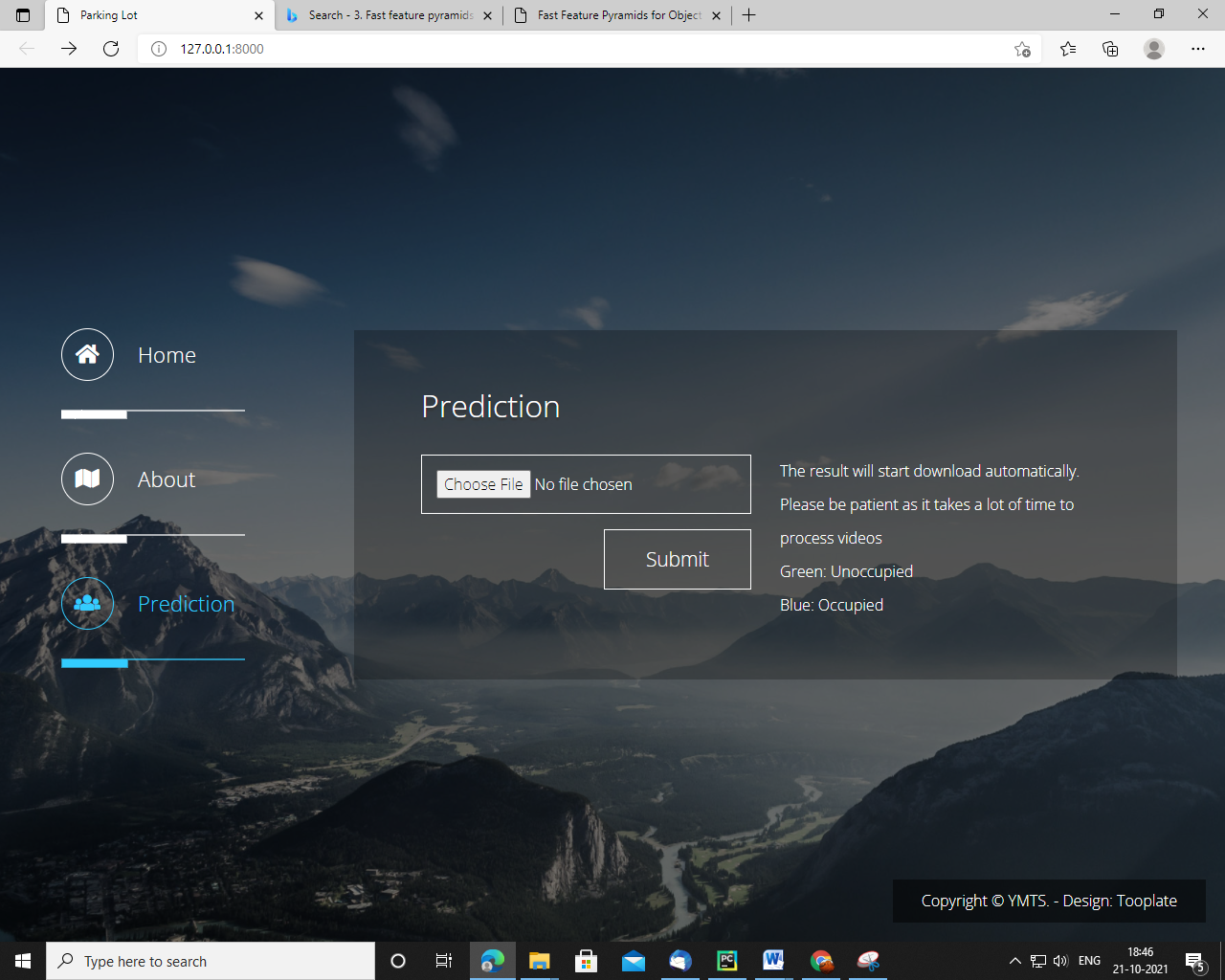
This page describes the predicted grade of the given input features.



This page describes the project briefly



This is the prediction page where u can run the algorithm



**TEST CASES**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input features | Tested for different features given by user on the model. | Success |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **Test cases** | **I/O** | **Expected O/T** | **Actual O/T** | **P/F** |
| 1 | Read the datasets. | Dataset’s path. | Datasets need to read successfully. | Datasets fetched successfully. | It produced P. If this not F will come |
| 2 | Verifying the features and generates result. | Input image | Output is classified as different parking spots | Output is classified as different parking spots | It produced P. If this is not, it will undergo F |

**Test cases Model building:**

**CONCLUSION:**

This paper presents a system for the management of vacant parking places by means of vehicle detection and their corresponding mapping into the parking spaces of a parking lot. The system has been designed so that existing parking lot security cameras can be used for the proposed system after a simple conﬁguration, without the need for a complete new camera deployment. The designed system faces more complicated scenarios than the ones tackled in the state of the art: almost total occlusions and climatic changes (cloudy scenarios, rain, snow...), that limits/reduces their performance. In this scenario with such a variable background it is not possible to carry out a precise background extraction, nor it is possible to label and deﬁne the region of each place as some parked vehicles completely occlude some of the spots behind them.

**FUTURE SCOPE**

As for future works, further investigation on the use of vehicle detection techniques are recommended to optimize the result for predicting parking spots. It is also essential to select images from different parking lots for datasets.. Thus, using deep learning for parking spot detection will ultimately enhance its accuracy.

**REFERENCES:**

1. T. Fabian, “An algorithm for parking lot occupation detection,”.
2. C.-C. Huang and S.-J. Wang, “A hierarchical Bayesian generation framework for vacant parking space detection,’’
3. X. Wang and A. R. Hanson, “Parking lot analysis and visualization from aerial images,’’
4. K. Yamada and M. Mizuno, “A vehicle parking detection method using image segmentation
5. C.-H. Lee, M.-G. Wen, C.-C. Han, and D.-C. Kou, “An automatic monitoring approach for unsupervised parking lots in outdoors,,’’
6. D. B. Bong, K. C. Ting, and N. Rajaee, “Car-park occupancy information system:
7. B. L. Bong, K. C. Ting, and K. C. Lai, “Integrated approach in the design of car park occupancy information system (coins),
8. S. F. Lin, Y. Y. Chen, and S. C. Liu, “A vision-based parking lot management system’’
9. L. C. Chen, J. W. Hsieh, W. R. Lai, C. X. Wu, and S. Y. Chen, “Vision-based vehicle surveillance and parking lot management using multiple cameras’’
10. K. Blumer, H. R. Halaseh, M. U. Ahsan, H. Dong, and N. Mavridis, Cost-Effective Single-Camera Multi-Car Parking Monitoring and Vacancy Detection towards Real-World Parking Statistics and Real-Time Reporting,’’
11. L. Junzhao, M. Mohandes, and M. Deriche, “A multi-classifier image based vacant parking detection system,,
12. I. A.-B. Hilal Al-Kharusi, “Intelligent parking management system based on image processing.,
13. I. Masmoudi, A. Wali, A. Jamoussi, and A. M. Alimi, “Vision based system for vacant parking lot detection: VPLD,
14. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF)’’

**BIBLIOGRAPHY:**

**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:**

**Installing Python:**

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

**Installing PyCharm:**

1. To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and click the "DOWNLOAD" link under the Community Section.



1. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected JetBrains and click on “Install”.
5. Wait for the installation to finish.
6. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
7. After you click on "Finish," the Following screen will appear.



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: Pip install NumPy



# **INTRODUCTION TO PYTHON**

* Python

### What Is a Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,*or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If you’re not interested in the how’s and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

• Great interactive environment

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

Python Features

Python's features include −

* Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* Easy-to-read − Python code is more clearly defined and visible to the eyes.
* Easy-to-maintain − Python's source code is fairly easy-to-maintained.
* A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* Extendable − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* Databases − Python provides interfaces to all major commercial databases.
* GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* Scalable − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

## Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# 20 Python libraries

1. Requests. The most famous http library written by Kenneth remits. It’s a must have for every python developer.

2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.

6. Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyser.

11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI’s for my python scripts.

14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

15. Scaly. A packet sniffer and analyser for python made in python.

16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can-do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

NumPy

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

• High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behaviour. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copy’s part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

Exceptions

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

. • Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* global names of a module
* local names in a function or method invocation
* built-in names: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
* Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

2. SYSTEM STUDY

### SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.