**INTRODUCTION**

**CHAPTER 1**

**1. INTRODUCTION**

**1.1. NEED OF THE PROJECT**

The increased number of diabetic patients worldwide, together with their proven inability to assess their diet accurately raised the need to develop systems that will support T1D patients. So far, a broad spectrum of mobile phone applications have been proposed in the literature, ranging from interactive diaries to dietary monitoring based on body sensors. The increasing processing power of the mobile devices, as well as the recent advances made in computer vision, permitted the introduction of image/video analysis-based applications for diet management. In a typical scenario, the user acquires an image of the upcoming meal using the camera of his phone. The image is processed - either locally or on the Server side - in order to extract a series of features describing its visual properties. The extracted Features are fed to a classifier to recognize the various food types of the acquired image, which will then be used for the CHO estimation

* 1. **OBJECTIVE**

The scope of this experiment is to identify the proper descriptor size or combination of sizes that should be used to describe the best performing key point extraction technique. To this end, different sizes were evaluated and then combined into a multi-scale scheme using a dense sampler. The used descriptor sizes were 16, 24, 32 and 56 all their combinations with spacing among them equal to 1/2 of each size in order to guarantee a sufficient number of patches.

* 1. **BENEFIT OF THE PROJECT**

The existing image analysis context, an image is represented by the histogram of visual words, which are defined as representative image patches of commonly occurring visual patterns. BoW is a popular way of representing documents in natural language processing, which ignores the order of the words belonging to a previously defined word dictionary, and considers only how frequently they appear. BoF was derived from the Bag of Words (BoW) model. The Bag-of-words model is a simplifying representation used in natural language processing and information retrieval. In this model, a text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity .The concept of the BoF model adequately fits the food recognition problem, since a certain food type is usually perceived as an Ensemble of different visual elements mixed with specific proportions, but without any typical spatial arrangement, a fact that encourages the use of a BoF approach, instead of any direct image matching technique.Decision trees and neural networks were helpful to generate binary classifiers of images.After the keypoint extraction, a local image descriptors is applied to a rectangular area around each key point.The accuracy will be high when compared to existing system.

**LITERATURE**

**SURVEY**

**CHAPTER-2**

1. **LITERATURE SURVEY**

**SURVEY PAPERS**

1.Text Categorization with Support Vector Machines: Learning with Many Relevant.

In the year 1997, the author Thorsten Joachim proposed this system.In this paper the text is matched with database.This paper explores the use of Support Vector Machines (SVMs) for learning text classifiers from examples. It analyzes the particular properties of learning with text data and identifies why SVMs are appropriate for this task. Empirical results support the theoretical endings. SVMs achieve substantial improvements over the currently best performing methods and behave robustly over a variety of different learning tasks. Furthermore, they are fully automatic, eliminating the need for manual parameter tuning.The main advantage is it rapidly identifies the information on WWW which acts as user guide and the disadvantage found in this paper is it is applicable only for text not for images.

2.Indoor-outdoor Image Classification

In the year 1998, the author Martin Szummer and Rosalind W\_ Picard proposed this system.This paper show high-level scene properties can be inferred from classification of low level image features specially for the indoor-outdoor scene retrieval problem This paper demonstrate that performance is improved by computing features on sub blocks classifying these sub blocks and then combining these results in a way suggestive of stacking State of the art single feature methods are shown to result in about performance while the new method results in correct classification when evaluated on a diverse database of over consumer images provided.The advantage of this paper is performance is improved by computing features on sub blocks and the disadvantages is it is difficult to predict performance of a feature.

3.Distinctive Image Features from Scale-Invariant Key points

In the year 2004, the author proposed this system.This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition cans robustly identify objects among clutter and occlusion while achieving near real-time performance**.**There is no need for code word recognization and the disadvantage isit has to refer with millions of images in database.

4.Scalable Recognition with a Vocabulary Tree

In the year 2005, the author David Nist´er and Henrik Stew´enius proposed this system.Recognition scheme that scales efficiently to a large number of objects is presented. The efficiency and quality is exhibited in a live demonstration that recognizes CD-covers from a database of 40000 images of popular music CD’s. The scheme builds upon popular techniques of indexing descriptors extracted from local regions, and is robust to background clutter and occlusion. The local region descriptors are hierarchically quantized in a vocabulary tree. The vocabulary tree allows a larger and more discriminatory vocabulary to be used efficiently, this paper show experimentally, leads to a dramatic improvement in retrieval quality. The most significant property of the scheme is that the tree directly defines the quantization. The quantization and the indexing are therefore fully integrated, essentially being one and the same. The recognition quality is evaluated through retrieval on a database with ground truth, showing the power of the vocabulary tree approach, going as high as 1 million images.It can recognize multiple actions simultaneously.The main disadvantage isthey are not easily scalable.

5.Sampling Strategies for Bag-of-Features Image Classification

In the year 2006 , the author Eric Nowak1*,* 2, Frederic Jurie1, and Bill Triggs1 proposed this system.Bag-of-features representations have recently become popular for content based image classification owing to their simplicity and good performance. They evolved from texton methods in texture analysis. The basic idea is to treat images as loose collections of independent patches, sampling a representative set of patches from the image, evaluating a visual descriptor vector for each patch independently, and using the resulting distribution of samples in descriptor space as a characterization of the image. The four main implementation choices are thus how to sample patches, how to describe them, how to characterize the resulting distributions and how to classify images based on the result. This paper concentrate on the first issue, showing experimentally that for a representative selection of commonly used test databases and for moderate to large numbers of samples, random sampling gives equal or better classifiers than the sophisticated multiscale interest operators that are in common use. Although interest operators work well for small numbers of samples, the single most important factor governing performance is the number of patches sampled from the test image and ultimately interest operators cannot provide enough patches to compete. This paper also studies the influence of other factors including codebook size and creation method, histogram normalization method and minimum scale for feature extraction. It is very Simple and performance is good. The main disadvantage is food items that is greatly confused with each other.

6.Evaluation of Color Descriptors for Object and Scene Recognition

In the year 2007 , the author Koen E. A. van de Sande and Theo Gevers and Cees G. M. Snoek proposed this system.Image category recognition is important to access visual information on the level of objects and scene types. So far, intensity-based descriptors have been widely used. To increase illumination invariance and discriminative power, color descriptors have been proposed only recently. As many descriptors exist, a structured overview of color invariant descriptors in the context of image category recognition is required. Therefore, this paper studies the invariance properties and the distinctiveness of color descriptors in a structured way. The invariance properties of color descriptors are shown analytically using a taxonomy based on invariance properties with respect to photometric transformations. The distinctiveness of color descriptors is assessed experimentally using two benchmarks from the image domain and the video domain. In this Color descriptor has been used. The disadvantage is it affects the performance of the object recognition.

7. Content-Based Hierarchical Classification of Vacation Images

In the year 2007 , the author Yan Lindsay Sun, Zhu Ha*,* Wei Yu*†* and K. J. Ray Liu proposed this system.The performance of distributed networks depends on collaboration among distributed entities. To enhance security in distributed networks, such as ad hoc networks, it is important to evaluate the trustworthiness of participating entities since trust is the major driving force for collaboration. In this paper, this paper presents a framework to quantitatively measure trust, model trust propagation, and defend trust evaluation systems against malicious attacks. In particular, this paper address the fundamental understanding of trust, quantitative trust metrics, mathematical properties of trust, dynamic properties of trust, and trust models. The attacks against trust evaluation are identified and defense techniques are developed. The proposed trust evaluation system is employed in ad hoc networks for securing ad hoc routing and assisting malicious node detection. The implementation is fully distributed. Simulations show that the proposed system can significantly improve network throughput as well as effectively detect malicious behaviors in ad hoc networks. Further, extensive simulations are performed to illustrate various attacks and the effectiveness of the proposed defense techniques. It improves network throughput and detect malicious behaviors. The disadvantage is it provides high level semantic indices into larger database.

8. A Bayesian Hierarchical Model for Learning Natural Scene Categories.

In the year 2010 , the author Li Fei-Fei, Pietro Perona proposed this system.This paper proposes a novel approach to learn and recognize natural scene categories. Unlike previous work, it does not require experts to annotate the training set. This paper represents the image of a scene by a collection of local regions, denoted as code words obtained by unsupervised learning. Each region is represented as part of a “theme”. In previous work, such themes were learnt from hand-annotations of experts, while our method learns the theme distributions as well as the codeword distribution over the themes without supervision. This paper report satisfactory categorization performances on a large set of 13 categories of complex scenes.No need of manual supervision to identify the scene.More code words have to be stored.

9.Towards Good Practice in Large-Scale Learning for Image

In the year 2010 , the author Florent Perronnin, Zeynep Akata, Zaid Harchaoui and Cordelia Schmid proposed this system.Compare the one-vs.-rest, multiclass, ranking and weighted average ranking SVMs. Using stochastic gradient descent optimization, we can scale the learning to millions of images and thousands of classes. Our experimental evaluation shows that ranking based algorithms do not outperform a one-vs.-rest strategy and that the gap between the different algorithms reduces in case of high-dimensional data. This paper also show that for one-vs.-rest, learning through cross-validation the optimal degree of imbalance between the positive and the negative samples can have a significant impact. Furthermore, early stopping can be used as an effective regularization strategy when training with stochastic gradient algorithms. Following these “good practices”, were able to improve the state-of-the-art on a large subset of 10K classes and 9M of images of Image Net from 16.7% accuracy to 19.1%.It

improve the state-of-the-art on a large subset.

10. Pfid: Pittsburgh Fast-Food Image Dataset

In the year 2012 , the author Rahul Sukthankar1, Jie Yang2 proposed this system.This paper introduce the first visual dataset of fast foods with a total of 4,545 still images, 606 stereo pairs, 303 3600 videos for structure from motion, and 27 privacy-preserving videos of eating events of volunteers. This work was motivated by research on fast food recognition for dietary assessment. The data was collected by obtaining three instances of 101 foods from 11 popular fast food chains, and capturing images and videos in both restaurant conditions and a controlled lab setting. This paper benchmark the dataset using two standard approaches, color histogram and bag of SIFT features in conjunction with a discriminative classifier. Our dataset and the benchmarks are designed to stimulate research in this area and will be released freely to the research community. The dataset and evaluations are freely available to public. There is no public dataset for automated food recognition system.

**Comparison of literature survey:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TITLE | YEAR | AUTHOR | DESCRIPTION | ADVANTAGE | DISADVANTAGE |
| Text Categorization with Support Vector Machines: Learning with Many Relevant. | 1997 | Thorsten Joachims. | It is used to match the text with database. | It rapidly identifies the information on WWW which acts as user guide. | It is applicable only for text not for images. |
| Indoor-outdoor Image Classification | 1998 | Martin Szummer and Rosalind W Picard | High-level scene properties can be inferred from low level image features. | Performance is improved by computing features on sub blocks | It is difficult to predict performance of a feature . |
| Distinctive Image Features from Scale-Invariant Key points | 2004 | David G.Lowe | It matches image and scene with database by using scale invariant method. | No need for code word recognization. | It have to refer with millions of images in database. |
| Scalable Recognition with a Vocabulary Tree. | 2005 | David Nist´er and Henrik Stew´enius | Thelocal region descriptors are hierarchically quantized in a vocabulary tree leads to improvement. | It can recognize multiple acions simultaneously | They are not easily scalable |
| Sampling Strategies for BOF Image Classification | 2006 | Eric Nowak, Frederic, and Bill Triggs. | To treat images as loose collections of independent patches. | Simple and good performance | Food items that are greatly confused with each other. |

**SYSTEM DESIGN**

**CHAPTER 3**

**3. SYSTEM DESIGN**

**3.1. PROPOSED SYSTEM**

Web service composition enables seamless and dynamic integration of business applications on the web. The performance of the composed application is determined by the performance of the involved web services. Therefore, non-functional, quality of service (QoS) aspects (e.g. response time, availability, etc.) is crucial for selecting the web services to take part in the composition. The problem of identifying the best candidate web services from a set of functionally-equivalent services is a multi-criteria decision making problem. The selected services should optimize the overall QoS of the composed application, while satisfying all the constraints specified by the client on individual QoS parameters.

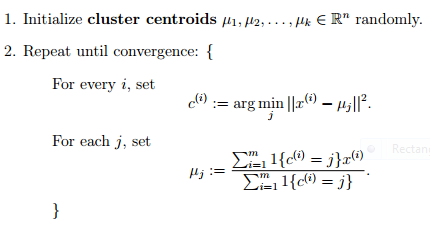
This project analyzes the problem of clustering food image data list based RGB values.

**PROBLEM DEFINITION**

The image classification stage is involved in both training and testing phases. In order to identify the appropriate classifier for the specific problem. Identifying the appropriate descriptor size and type for a recognition problem is a challenging task that involves a number of experiments. Its Inability to capture any color information constitutes a problem for the description of many Objects, including foods

**ALGORITHM USED**

In the clustering problem, we are given a training set *x*(1),...,*x*(*m*), and want to group the data into a few cohesive "clusters." Here, we are given feature vectors for each data point *x*(*i*)∈R*n* as usual; but no labels *y*(*i*) (making this an unsupervised learning problem). Our goal is to predict *k* centroids **and** a label *c*(*i*) for each datapoint. The k-means clustering algorithm is as follows:

**𝑘-means clustering algorithm**

A clustering algorithm attempts to find natural groups of components (or data) based on some Similarity. Also, the clustering algorithm finds the centroid of a group of data set. Key points are selected points on an image that define the centers of local patches where descriptors will be applied. In the current study, three different key point extraction methods were tested: interest Point detectors, random sampling and dense sampling. Interest point detectors, such as SIFT are considered as the best choice for image matching problems where a small number of samples is required, as it provides stability under local and global image perturbations.

.

**3.2. METHODOLOGIES**

Methodologies are the process of analyzing the principles or procedure for Clustering images based on BOF (Bag of Feature Model).

**3.2.1 MODULE NAME**

* Authentication
* Dataset Collection
* Keypoint Extraction
* Feature Description
* Descriptor Quantization
* Imgae Classfication

**3.2.2 MODULE DESCRIPTION**

**AUTHENTICATION**

In authentication module is used to checking the user as valid or invalid. In this module enter the username and password, this username and password is check into the database. If username and password is correct then allow to next processing, otherwise it consider as invalid user and again go to the login process.

**DATASET COLLECTION**

In this module we describe the image dataset that it is contain various types of food images. The overall goal of the task is to collecting multi-modal images approaches that combine textual and visual evidence in order to satisfy a user’s multimedia information need could deal with larger scale image collections that contain highly heterogeneous items both in terms of their textual descriptions and their visual content. The aim is to simulate image retrieval in a realistic setting, such as the Web environment

**KEYPOINT EXTRACTION**

Key points are selected points on an image that define the centers of local patches where descriptors will be applied. In the current study, three different key point extraction methods were tested: interest point detectors, random sampling and dense sampling. Interest point detectors, are considered as the best choice for image matching problems where a small number of samples is required, as it provides stability under local and global image perturbations.

**FEATURE DESCRIPTION**

A local image descriptor is applied to a rectangular area around each key point to produce a feature vector. Identifying the appropriate descriptor size and type for a recognition problem is a challenging task that involves a number of experiments.

**DESCRIPTOR QUANTIZATION**

Descriptor quantization is the procedure of assigning a feature vector to the closest visual word of a predefined visual vocabulary. Once the visual dictionary is learnt, each descriptor of an image is quantized and the histogram of visual word occurrences serves as a global description of the image. Then, the histogram values are usually scaled to [0 1] and are fed to the classifier either for training or testing. The efficiency of this part of the system is crucial, since it affects processing times for both training and testing. The complexity of the descriptor quantization mainly depends on the dimensions of the descriptor and the number of visual words.

**IMAGE CLASSFICATION**

The image classification stage is involved in both training and testing phases. In order to identify the appropriate classifier for the specific problem, several experiments with three supervised classification methods were conducted. According to our food type we can those images for further more easy way to retrieving images from classified lists.

**3.2.3 MODULE DIAGRAM**

**AUTHENTICATION**

Sender

Username / Password

Check

Data Base

Proceed Next Level

Fig.3.1. Module diagram of authentication.

**DATASET COLLECTION**

Click to retrieve images

Check file extensions

Image dataset

Various types of food images

Fig.3.2. Module diagram of dataset collection.

**KEYPOINT EXTRACTION**

Select image from dataset

Upload image

Select key points from

Food Image dataset

Various food images

Key point extraction

Fig.3.3. Module diagram of keypoint extraction.

**FEATURE DESCRIPTION**

Selected food image

Key point’s selection

From image

Applying rectangular area

Future vector

Production

Fig.3.4.Module diagram of Feature description.

**DESCRIPTOR QUANTIZATION**

Selected image

Key point extraction

Color histogram

Color descriptors

R

G

B

Descriptor quantization

Visual dictionary classifier

Training set

Fig.3.5.Module diagram of descriptor quantization.

**IMAGE CLASSIFICATION**

Experimental results

Classifier

Cluster images

Analyze food type

Store images

Fig.3.6.Module diagram of image classification.

**3.2.4 USE CASE DIAGRAM**

A use case diagram is a type of behavioral diagram created from a Use-case analysis. The purpose of use case is to present overview of the functionality provided by the system in terms of actors, their goals and any dependencies between those use cases.



Fig.3.7.Usecase diagram.

**3.2.5 CLASS DIAGRAM**

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, and the relationships between the classes.Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information.Protected visibility allows child classes to access information they inherited from a parent class.



Fig.3.8.Class diagram.

**3.2.6 OBJECT DIAGRAM**

An object diagram in the Unified Modeling Language (UML) is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time.

Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.

Admin

User name: \*\*\*\*\*\*

Password: \*\*\*\*\*\*

Image Dataset collection N1

Admin: process1

User name: \*\*\*\*\*\*

Password: \*\*\*\*\*\*

Upload image, key point extraction

Admin: process2

User name: \*\*\*\*\*\*

Password: \*\*\*\*\*\*

Feature descriptor, visual dictionary

Admin: process3

User name: \*\*\*\*\*\*

Password: \*\*\*\*\*\*

Feature descriptor, visual dictionary

User

User name: \*\*\*\*\*\*

Password: \*\*\*\*\*\*

Search food list

Admin: process3

User name: \*\*\*\*\*\*

Password: \*\*\*\*\*\*

Choose food list, image classification

Fig.3.9.Object diagram.

**3.2.7 STATE DIAGRAM**

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. There are many forms of state diagrams, which differ slightly and have different semantics

****

Fig.3.10.State diagram.

**3.2.8 ACTIVITY DIAGRAM**

Activity diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structural development.



Fig.3.11. Activity diagram

**3.2.9 SEQUENCE DIAGRAM**

A sequence diagram in 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-trace diagrams, event scenarios, and timing diagrams.



Fig.3.12.Sequence diagram.

**3.2.10 COLLABORATION DIAGRAM**

A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. It can be the decomposition of a use case diagram, or part of a use case diagram.



Fig.3.13.Collaboration diagram

**3.2.11 COMPONENT DIAGRAM**

Components are wired together by using an *assembly connector* to connect the required interface of one component with the provided interface of another component. This illustrates the *service consumer - service provider* relationship between the two components.

An *assembly connector* is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."



Fig.3.14. Component diagram

**3.2.12 DATA FLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts.

**LEVEL 0**

Admin

L0 authentication

Login, cancel

D0 Database

**LEVEL 1**

Admin

L0 authentication

Login, cancel

D0 Database

L1 image dataset

Upload images

D1 Database

L2 key point extraction

Select key points

**LEVEL 2**

Admin

L0 authentication

Login, cancel

D0 Database

L3 feature descriptor

Color invariant features

D3 Database

L4 visual dictionary

Choose food list

**ALL LEVEL**

L3 feature descriptor

Color invariant features

D2 Database

L4 visual dictionary

Choose food list

L5 image classification

Cluster food images

D3 Database

Admin

L0 authentication

Login, cancel

D0 Database

L1 image dataset

Upload images

D1 Database

L1 key point extraction

Select key points

**3.2.13 E-R DIAGRAM:**

In software engineering, an entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.

Admin

Authentication

Dataset collection

Fig.3.15.E-R diagram.

**3.2.14 ARCHITECTURE DIAGRAM:**

(10,2,52….,70)

(7,54,3……56)

(45,34,4…..64)

Learning of visual dictionary

Classifier training

classifier

Food type

Fig.3.16.Architecture of the proposed large scale learning for food image classification

**3.16. Architecture diagram of the proposed system.**

System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system.

**REQUIREMENTS**

**CHAPTER 4**

**4. REQUIREMENTS**

**4.1. GENERAL**

Using this requirement, our application provides high service with efficiently. Software requirements deal with defining software resource requirements and pre-requisites that need to be installed on a server that provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed. The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**4.2. HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* Processor : Pentium Dual Core 2.00GHZ
* Hard disk : 40 GB
* Mouse : Logitech.
* RAM : 2GB(minimum)
* Keyboard : 110 keys enhanced.

**4.3. SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* + - Operating system : Windows 07/ 08
    - IDE : Visual Studio 2010
    - Front End : ASP.NET
    - Database : SQL Server 2008

**4.4. FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, and outputs. Our system requires minimum three systems to achieve this concept. **4.5.NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

Sensitive analysis is conducted to investigate how to cluster food images data list based on their features.

**IMPLEMENTATION**

**CHAPTER 5**

**5. IMPLEMENTATION**

**5.1. SOFTWARE SPECIFICATION**

This chapter is about the software language and the tools used in the development of the project. The platform used here is .NET. The Primary languages are C#, VB, and J#. In this project C# is chosen for implementation.

**5.2. DOTNET**

**5.2 .1 INTRODUCTION TO DOTNET**

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There’s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

“.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

* + 1. **THE .NET FRAMEWORK**

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).

2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run.

The most important features are

* Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
* Memory management, notably including garbage collection.
* Checking and enforcing security restrictions on the running code.
* Loading and executing programs, with version control and other such features.
* The following features of the .NET framework are also worth description.

**5.2.3 MANAGE CODE**

The code that targets .NET, and which contains certain extra Information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

**5.2.4 MANAGE DATA**

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

**5.2.5 COMMON TYPE SYSTEM (CTS)**

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.

**5.2.6 COMMON LANGUAGE SPECIFICATION**

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

**5.2.7 THE CLASS LIBRARY**

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity. The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

**LANGUAGES SUPPORTED BY .NET**

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”.

Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python.

Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

|  |  |
| --- | --- |
| ASP.NET  XML  WEB SERVICES | Windows Forms |
| Base Class Libraries | |
| Common Language Runtime | |
| Operating System | |

Fig 5.2.1 .Net Framework

**FEATURES OF C#**

1. C# is a simple, modern, object oriented language derived from C++ and Java.   
2. It aims to combine the high productivity of Visual Basic and the raw power of C++.   
3. It is a part of Microsoft Visual Studio7.0.   
4. Visual studio supports Vb, VC++, C++, Vbscript, Jscript. All of these languages provide access to the Microsoft .NET platform.   
5. .NET includes a Common Execution engine and a rich class library.   
6. Microsoft's JVM equiv. is Common language run time (CLR).   
7. CLR accommodates more than one languages such as C#, VB.NET, Jscript, ASP.NET, C++.   
8. Source code --->Intermediate Language code (IL) ---> (JIT Compiler) Native code.   
9.The classes and data types are common to all of the .NET languages.   
10. We may develop Console application, Windows application, and Web application using C#.   
11. In C# Microsoft has taken care of C++ problems.

**MAIN FEATURES OF C#**1. Pointers are missing in C#.   
2. Unsafe operations such as direct memory manipulation are not allowed.   
3. In C# there is no usage of "::" or "->" operators.   
4. Since it`s on .NET, it inherits the features of automatic memory management and garbage collection.   
5. Varying ranges of the primitive types like Integer, Floats etc.   
6. Integer values of 0 and 1 are no longer accepted as Boolean values. Boolean values are pure true or false values in C# so no more [errors](http://www.csharpfriends.com/Articles/getArticle.aspx?articleID=37) of "="operator and "=="operator. "==" is used for comparison operation and "=" is used for assignment operation.

**TYPE SAFE**  
1. In C# we cannot perform unsafe casts like convert double to a Boolean.   
2. Value types (primitive types) are initialized to zeros and reference types (objects and classes are initialized to null by the compiler automatically.   
3. Arrays are zero base indexed and are bound checked.   
4. Overflow of types can be checked.

**INTEROPERABILITY**  
1. C# includes native support for the COM and windows based applications.   
2. Allowing restricted use of native pointers.   
3. Users no longer have to explicitly implement the unknown and other COM interfaces, those features are built in.   
4. C# allows the users to use pointers as unsafe code blocks to manipulate your old code.   
5. Components from VB NET and other managed code languages and directly be used in C#.   
  
**5.4 OBJECTIVES OF .NET**

The .net framework is one of the tools provided by the .net platform. It provides an Environment for building, deploying and running web services and other applications like Console applications; Windows based applications, Web sites. It is a Common architecture for all .net programming languages.

The Main Objectives of .NET Framework

1) Platform dependent

2) Language Independent

3) Language Interoperability

4) Security

5) Database Connectivity

6) Globalization of Application

**Platform Independent**: As dll or exe files are executable in any operating system with the help of the CLR (common language runtime), hence .net is called as platform independent.

**Language Independent:** As .net application logic can be developed in any .net framework compatible languages, hence it is called as Language Independent.

**Specification in ASP.net :**It provides set of rules to be followed while integrating with the language.

**Language Interoperability**: The code written in one language should be used from the application developed using other language.

**Security:** The .net applications attain high level of security.

**Database Connectivity**: A new Database connectivity model to connect Database.

**Globalization of Application**: Designing the applications for supporting multiple languages and cultures.

**5.5 componets of .NET frmework**

The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable ASP.NET applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and isolated file storage.

**Common Language Runtime**

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime. With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions; even if it is being used in the same active application. The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature rich.

The runtime also enforces code robustness by implementing a strict type-and-code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS.

**Base Class Library**

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

* Console applications.
* Windows GUI applications (Windows Forms).
* ASP.NET applications.
* XML Web services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

**features of the common LANGUAGE RUNTIME**

Common Language Runtime is a heart of the .net framework. It actually manages the code during Execution. The Code that runs under the CLR is called “Managed Code”. The code that is executed  under .net runtime gets benefits like cross language inheritance, cross language exception handling, enhanced Security, Versioning and development support, a simplified model for component interaction, debugging and Profiling services.

**FEATURES PROVIDED BY CLR**

Automatic memory management: - The CLR provides the Garbage Collection feature for managing the life time of object. This relives a programmer from memory management task.

Standard Type System: - The CLR Implement a formal Specification called the Common Type System (CTS). CTS is important part of rules that ensures that objects written in different language can interact with each other.

Language interoperability: - It is the ability of an application to interact with another application written in a different programming language. Language interoperability helps maximum code reuse. The CLR provides support for language interoperability by specifying and enforcing CTS and by providing metadata.

Platform Independence: - The Compiler compiles code language, which is CPU-independent. This means that the code can be executed from any platform that supports the .Net CLR.

Security Management: - In .net platform, Security is achieved through the code access Security (CAS) model. In the model, CLR enforces the restriction an managed code through the object called “permissions”. The CLR allows the code to perform only that task for which it has permissions. In other words, the CAS model specifies what the code can access instead of specifies who can access resources.

Type Safety: - This feature ensures that object is always accessed in compatible ways. Therefore the CLR will prohibit a code from assign a 10-byte value to an object that occupies 8 bytes.

**BENEFITS OF CLR**

* Performance improvement
* The ability to easily use components developed in other languages.
* Extensible types provided by library.
* New Language features such as inheritance, interfaces etc.
* Complete Object-Oriented design.
* Very Strong Type Safety.
* A good blend of Visual Basic simplicity and c++ power.

Syntax and keywords similar to c and c++.

Use of delegates rather than function pointers for increased type safety and security.

**5.6 ASP.NET OVERVIEW**

ASP.Net is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web application for PC, as well as mobile devices.

ASP.Net works on top of the HTTP protocol and uses the HTTP commands and policies to set a browser-to-server two-way communication and cooperation.

ASP.Net is a part of Microsoft .Net platform. ASP.Net applications are complied codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.Net application codes could be written in either of the following languages:

* C#
* Visual Basic .Net
* Jscript
* J#

ASP.Net is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls like text boxes, buttons and labels for assembling, configuring and manipulating code to create HTML pages.

**ASP.Net Web Forms Model:**

ASP.Net web forms extend the event-driven model of interaction to the web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.

All client side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions.

Now, HTTP is a stateless protocol. ASP.Net framework helps in storing the information regarding the state of the application, which consists of:

* Page state
* Session state

The page state is the state of the client, i.e., the content of various input fields in the web form. The session state is the collective obtained from various pages the user visited and worked with, i.e., the overall session state. To clear the concept, let us take up an example of a shopping cart as follows.

User adds items to a shopping cart. Items are selected from a page, say the items page, and the total collected items and price are shown in a different page, say the cart page. Only HTTP cannot keep track of all the information coming from various pages. ASP.Net session state and server side infrastructure keeps track of the information collected globally over a session.

The ASP.Net runtime carries the page state to and from the server across page requests while generating the ASP.Net runtime codes and incorporates the state of the server side components in hidden fields.

This way the server becomes aware of the overall application state and operates in a two-tiered connected way.

**ASP.Net Component Model:**

The ASP.Net component model provides various building blocks of ASP.Net pages. Basically it is an object model, which describes:

* Server side counterparts of almost all HTML elements or tags, like <form> and <input>.
* Server controls, which help in developing complex user-interface for example the Calendar control or the Gridview control.

ASP.Net is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.Net web application is made of pages. When a user requests an ASP.Net page, the IIS delegates the processing of the page to the ASP.Net runtime system.

The ASP.Net runtime transforms the .aspx page into an instance of a class, which inherits from the base class Page of the .Net framework. Therefore, each ASP.Net page is an object and all its components i.e., the server-side controls are also objects.

**5.6.1 ASP.NET ARCHITECTURE**

ASP.NET is based on the fundamental architecture of .NET Framework. Visual studio provide a uniform way to combine the various features of this Architecture.



Architecture is explained form bottom to top in the following discussion.

* At the bottom of the Architecture is Common Language Runtime. NET Framework common language runtime resides on top of the operating system services. The common language runtime loads and executes code that targets the runtime. This code is therefore called managed code. The runtime gives you, for example, the ability for cross-language integration.
* .NET Framework provides a rich set of class libraries. These include base classes, like networking and input/output classes, a data class library for data access, and classes for use by programming tools, such as debugging services. All of them are brought together by the Services Framework, which sits on top of the common language runtime.
* ADO.NET is intended specifically for developing web applications. This is evident from its two major design principles:
* Disconnected Datasets—In ADO.NET, almost all data manipulation is done outside the context of an open database connection.
* Effortless Data Exchange with XML—Datasets can converse in the universal data format of the Web, namely XML.

Web Services are software solutions delivered via Internet to any device. Today, that means Web browsers on computers, for the most part, but the device-agnostic design of .NET will eliminate this limitation.

* One of the obvious themes of .NET is unification and interoperability between various programming languages. In order to achieve this; certain rules must be laid and all the languages must follow these rules. In other words we cannot have languages running around creating their own extensions and their own fancy new data types. CLS is the collection of the rules and constraints that every language (that seeks to achieve .NET compatibility) must follow.
* The CLR and the .NET Frameworks in general, however, are designed in such a way that code written in one language can not only seamlessly be used by another language. Hence ASP.NET can be programmed in any of the .NET compatible language whether it is VB.NET, C#, Managed C++ or JScript.NET.

**5.6.2 ASP.NET PAGE LIFE CYCLE**

ASP.Net life cycle specifies, how:

* ASP.Net processes pages to produce dynamic output
* The application and its pages are instantiated and processed
* ASP.Net compiles the pages dynamically
* The ASP.Net life cycle could be divided into two groups:
* Application Life Cycle
* Page Life Cycle

## ASP.Net Application Life Cycle:

The application life cycle has the following stages:

* User makes a request for accessing application resource, a page. Browser sends this request to the web server.
* A unified pipeline receives the first request and the following events take place:
* An object of the ApplicationManager class is created.
* An object of the HostingEnvironment class is created to provide information regarding the resources.
* Top level items in the application are compiled.
* Response objects are created . the application objects: HttpContext, HttpRequest and HttpResponse are created and initialized.
* An instance of the HttpApplication object is created and assigned to the request. The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.

## ASP.Net Page Life Cycle:

When a page is requested, it is loaded into the server memory, processed and sent to the browser. Then it is unloaded from the memory. At each of this steps, methods and events are available, which could be overridden according to the need of the application. In other words, you can write your own code to override the default code.

The Page class creates a hierarchical tree of all the controls on the page. All the components on the page, except the directives are part of this control tree. You can see the control tree by adding trace= "true" to the Page directive. We will cover page directives and tracing under 'directives' and 'error handling'.

The page life cycle phases are:

* Initialization
* Instantiation of the controls on the page
* Restoration and maintenance of the state
* Execution of the event handler codes
* Page rendering

Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle. It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behavior code.

Following are the different stages of an ASP.Net page:

* Page request . when ASP.Net gets a page request, it decides whether to parse and compile the page or there would be a cached version of the page; accordingly the response is sent
* Starting of page life cycle . at this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
* Page initialization . at this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied. For a new request postback data is loaded and the control properties are restored to the view-state values.
* Page load . at this stage, control properties are set using the view state and control state values.
* Validation . Validate method of the validation control is called and if it runs successfully, the IsValid property of the page is set to true.
* Postback event handling . if the request is a postback (old request), the related event handler is called.
* Page rendering . at this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Page's Response property.
* Unload . the rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

## ASP.Net Page Life Cycle Events:

* At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes like Onclick or handle.
* Following are the page life cycle events:
* PreInit . PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page\_PreInit handler.
* Init . Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page\_Init handler.
* InitComplete . InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
* LoadViewState . LoadViewState event allows loading view state information into the controls.
* LoadPostData . during this phase, the contents of all the input fields defined with the <form> tag are processed.
* PreLoad . PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page\_PreLoad handler.
* Load . the Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page\_Load handler.
* LoadComplete . the loading process is completed, control event handlers are run and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page\_LoadComplete handler.
* PreRender . the PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
* PreRenderComplete . as the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
* SaveStateComplete . state of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page\_Render handler.
* UnLoad . the UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page\_UnLoad handler.

An ASP.Net page is made of number of server controls along with the HTML controls, text and images. Sensitive data from the page and the states of different controls on the page are stored in hidden fields and forms the context of that page request.

ASP.Net runtime controls all association between a page instance and its state. An ASP.Net page is an object of the Page Class or inherited from it.

All the controls on the pages are also objects of the related control class inherited from a parent Control class. When a page is run an instance of the page object is created along with all its content controls.

An ASP.Net page is also a server side file saved with the .aspx extension. It is modular in nature and can be divided into the following core sections:

* Page directives
* Code Section
* Page Layout

## Page directives

The page directives set up the environments for the page to run. The @Page directive defines page-specific attributes used by the ASP.Net page parser and compiler. Page directives specify how the page should be processed, and which assumptions are to be taken about the page.

It allows importing namespaces, loading assemblies and registering new controls with custom tag names and namespace prefixes. We will discuss all of these concepts in due time.

## Code Section

The code section provides the handlers for the page and control events along with other functions required. We mentioned that, ASP.Net follows an object model. Now, these objects raises events when something happens on the user interface, like a user clicks a button or moves the cursor. How these events should be handled? That code is provided in the event handlers of the controls, which are nothing but functions bound to the controls.

The code section or the code behind file provides all these event handler routines, and other functions used by the developer. The page code could be precompiled and deployed in the form of a binary assembly.

## Page Layout

The page layout provides the interface of the page. It contains the server controls, text, inline JavaScript and HTML tags:

The following code snippet provides a sample ASP.Net page explaining Page directives, code section and page layout written in C#:

**5.7 SQL SERVER 2008**

SQL Server 2005 will be soon reaching its three-year mark, which in terms of software life-cycle translates into fairly advanced maturity. While this is still far from retirement age, the name of its successor, SQL Server 2008, suggests that it might be time for you to start looking into what the new generation has to offer. The release of SQL Server 2008, originally introduced as Yukon, has already been postponed, but its current Beta 2 implementation (with several incremental Community Technical Previews expected before Beta 3 becomes available early next year) brings promise of a timely RTM stage (planned for summer next year). In this series of articles, we will look into functional highlights of the new incarnation of the Microsoft database management system, focusing on those that are likely to remain unchanged in the final product.

Improvements to the database engine, the details of which are not published by Microsoft, and the corresponding changes to the main infrastructure components are reflected by a substantial number of new features as well as enhancements to existing ones. The most relevant ones can be grouped into several categories, such as high availability and scalability, security, data management, administration and maintenance, and development.

The demand for high availability is becoming increasingly common and is no longer limited to major corporate and governmental clients. This results not only from a growing level of customer expectations, but also from the new political climate associated with more stringent legislative and regulatory requirements, in which disaster recovery and business continuity are more relevant than ever. However, businesses are also, at the same time, extremely interested in keeping their costs to a minimum. Microsoft tries to address these expectations by implementing scalability enhancements, which ensure that SQL Server can perform equally well in environments of any size, and by the introduction of several versions of SQL Server 2008 (geared towards more specialized needs) such as:

* SQL Server Standard Edition - offering the most diverse set of features and intended for the majority of clients.
* SQL Server 2008 Express Edition - serving as the replacement for Microsoft Data Engine (MSDE) and available for download from t. Like its predecessor, it was designed with developers in mind, however, unlike the previous version, it also includes a Web based management interface.
* SQL Server 2008 Mobile Edition - as a successor to SQL Server 2008 Windows CE Edition, it is intended for Windows mobile-based devices, such as Tablet PCs, Pocket PCs, and Smart phones

**5.7.1 features of SQL SERVEr**

The following is a list of the new features provided in SQL Server 2008:

* Database mirroring
* Database snapshots
* CLR integration
* Service Broker
* Database Mail
* User-defined functions
* Indexed views
* Distributed partitioned views
* INSTEAD OF and AFTER triggers
* New data types
* Cascading RI constraints
* Multiple SQL Server instances
* XML support
* Log shipping

**Database mirroring**

Database mirroring is a new high-availability feature in SQL Server 2008. It's similar to server clustering in that failover is achieved by the use of a stand-by server; the difference is that the failover is at the database level rather than the server level. The primary database continuously sends transaction logs to the backup database on a separate SQL Server instance. A third SQL Server instance is then used as a witness database to monitor the interaction between the primary and the mirror databases.

**DATABASE SNAPSHOTS**

A database snapshot is essentially an instant read-only copy of a database, and it is a great candidate for any type of reporting solution for your company. In addition to being a great reporting tool, you can revert control from your primary database to your snapshot database in the event of an error. The only data loss would be from the point of creation of the database snapshot to the event of failure.

**CLR integration**

With SQL Server 2008, you now have the ability to create custom .NET objects with the database engine. For example, stored procedures, triggers, and functions can now be created using familiar .NET languages such as VB and C#. Exposing this functionality gives you tools that you never had access to before such as regular expressions.

**Service Broker**

This feature gives you the ability to create asynchronous, message-based applications in the database entirely through TSQL. The database engine guarantees message delivery, message order consistency, and handles message grouping. In addition, Service Broker gives you the ability to send messages between different SQL Server instances. Server Broker is also used in several other features in SQL Server 2008. For example, you can define Event Nonfictions in the database to send a message to a Queue in the database when someone attempts to alter a table structure, of if there is a string of login failures.

**Database Mail**

Database Mail, the eventual successor to SQL Mail, is greatly enhanced e-mail solution available in the database engine. With Database Mail, there is no longer a dependency on Microsoft Outlook or MAPI e-mail clients. Database Mail uses standard SMTP to send e-mail messages. These messages may contain query results, attachments (which can be governed by the DBA), and is fully cluster aware. In addition, the e-mail process runs outside of the database engine space, which means that messages can continue to be queued even when the database engine has stopped.

**User-Defined Functions**

SQL Server has always provided the ability to store and execute SQL code routines via stored procedures. In addition, SQL Server has always supplied a number of built-in functions. Functions can be used almost anywhere an expression can be specified in a query. This was one of the shortcomings of stored procedures—they couldn't be used in line in queries in select lists, where clauses, and so on. Perhaps you want to write a routine to calculate the last business day of the month. With a stored procedure, you have to exec the procedure, passing in the current month as a parameter and returning the value into an output variable, and then use the variable in your queries. If only you could write your own function that you could use directly in the query just like a system function. In SQL Server 2008, you can.

SQL Server 2008 introduces the long-awaited support for user-defined functions. User-defined functions can take zero or more input parameters and return a single value—either a scalar value like the system-defined functions, or a table result. Table-valued functions can be used anywhere table or view expressions can be used in queries, and they can perform more complex logic than is allowed in a view.

**Indexed Views**

Views are often used to simplify complex queries, and they can contain joins and aggregate functions. However, in the past, queries against views were resolved to queries against the underlying base tables, and any aggregates were recalculated each time you ran a query against the view. In SQL Server 2008 Enterprise or Developer Edition, you can define indexes on views to improve query performance against the view. When creating an index on a view, the result set of the view is stored and indexed in the database. Existing applications can take advantage of the performance improvements without needing to be modified.

Indexed views can improve performance for the following types of queries:

* Joins and aggregations that process many rows
* Join and aggregation operations that are performed frequently within many queries
* Decision support queries that rely on summarized, aggregated data that is infrequently updated

**Distributed Partitioned Views**

SQL Server 7.0 provided the ability to create partitioned views using the UNION ALL statement in a view definition. It was limited, however, in that all the tables had to reside within the same SQL Server where the view was defined. SQL Server 2008 expands the ability to create partitioned views by allowing you to horizontally partition tables across multiple SQL Servers. The feature helps you scale out one database server to multiple database servers, while making the data appear as if it comes from a single table on a single SQL Server. In addition, partitioned views are now able to be updated.

**INSTEAD OF and AFTER Triggers**

In versions of SQL Server prior to 7.0, a table could not have more than one trigger defined for INSERT, UPDATE, and DELETE. These triggers only fired *after* the data modification took place. SQL Server 7.0 introduced the ability to define multiple AFTER triggers for the same operation on a table. SQL Server 2008 extends this capability by providing the ability to define which AFTER trigger fires first and which fires last.

SQL Server 2008 also introduces the ability to define INSTEAD OF triggers. INSTEAD OF triggers can be specified on both tables and views. (AFTER triggers can still only be specified on tables.) If an INSTEAD OF trigger is defined on a table or view, the trigger will be executed in place of the data modification action for which it is defined. The data modification is not executed unless the SQL code to perform it is included in the trigger definition.

**New Data types**

SQL Server 2008 introduces three new data types. Two of these can be used as datatypes for local variables, stored procedure parameters and return values, user-defined function parameters and return values, or table columns:

* bigint—An 8-byte integer that can store values from –263 (–9223372036854775808) through 263-1 (9223372036854775807).
* sql\_variant—A variable-sized column that can store values of various SQL Server-supported data types, with the exception of text, ntext, timestamp, and sql\_variant.

The third new datatype, the table datatype, can be used only as a local variable datatype within functions, stored procedures, and SQL batches. The table datatype cannot be passed as a parameter to functions or stored procedures, nor can it be used as a column datatype. A variable defined with the table datatype can be used to store a result set for later processing. A table variable can be used in queries anywhere a table can be specified.

**Text in Row Data**

In previous versions of SQL Server, text and image data was always stored on a separate page chain from where the actual data row resided. The data row contained only a pointer to the text or image page chain, regardless of the size of the text or image data. SQL Server 2008 provides a new text in row table option that allows small text and image data values to be placed directly in the data row, instead of requiring a separate data page. This can reduce the amount of space required to store small text and image data values, as well as reduce the amount of I/O required to retrieve rows containing small text and image data values.

**Cascading RI Constraints**

In previous versions of SQL Server, referential integrity (RI) constraints were restrictive only. If an insert, update, or delete operation violated referential integrity, it was aborted with an error message. SQL Server 2008 provides the ability to specify the action to take when a column referenced by a foreign key constraint is updated or deleted. You can still abort the update or delete if related foreign key records exist by specifying the NO ACTION option, or you can specify the new CASCADE option, which will cascade the update or delete operation to the related foreign key records.

**Multiple SQL Server Instances**

Previous versions of SQL Server supported the running of only a single instance of SQL Server at a time on a computer. Running multiple instances or multiple versions of SQL Server required switching back and forth between the different instances, requiring changes in the Windows registry.

SQL Server 2005 provides support for running multiple instances of SQL Server on the same system. This allows you to simultaneously run one instance of SQL Server 6.5 or 7.0 along with one or more instances of SQL Server 2008. Each SQL Server instance runs independently of the others and has its own set of system and user databases, security configuration, and so on. Applications can connect to the different instances in the same way they connect to different SQL Servers on different machines.

**XML Support**

Extensible Markup Language has become a standard in Web-related programming to describe the contents of a set of data and how the data should be output or displayed on a Web page. XML, like HTML, is derived from the Standard Generalize Markup Language (SGML). When linking a Web application to SQL Server, a translation needs to take place from the result set returned from SQL Server to a format that can be understood and displayed by a Web application. Previously, this translation needed to be done in a client application.

SQL Server 2008 provides native support for XML. This new feature provides the ability to do the following:

* Return query result sets directly in XML format.
* Retrieve data from an XML document as if it were a SQL Server table.
* Access SQL Server through a URL using HTTP. Through Internet Information Services (IIS), you can define a virtual root that gives you HTTP access to the data and XML functionality of SQL Server 2008.

**Log Shipping**

The Enterprise Edition of SQL Server 2008 now supports log shipping, which you can use to copy and load transaction log backups from one database to one or more databases on a constant basis. This allows you to have a primary read/write database with one or more read-only copies of the database that are kept synchronized by restoring the logs from the primary database. The destination database can be used as a warm standby for the primary database, for which you can switch users over in the event of a primary database failure. Additionally, log shipping provides a way to offload read-only query processing from the primary database to the destination database.

This capability was available in previous versions of SQL Server, but it required the DBA to manually set up the process and schedule the jobs to copy and restore the log backups. SQL Server 2008 officially supports log shipping and has made it easier to set up via the Database Maintenance Plan Wizard. This greatly simplifies the process by automatically generating the jobs and configuring the databases to support log shipping.

**DDL triggers**

In previous articles, I outlined how you can use data definition language (DDL) triggers in SQL Server 2008 to implement custom database and server auditing solutions for Sarbanes-Oxley compliance (here are [part one](http://techrepublic.com.com/5100-9592_11-6070262.html) and [part two](http://techrepublic.com.com/5100-9592_11-6072139.html) of my SOX articles). DDL triggers are defined at the server or database level and fire when DDL statements occur. This gives you the ability to audit when new tables, stored procedures, or logins are created.

**Ranking functions**

SQL Server 2008 provides you with the ability to rank result sets returned from the database engine. This allows you to customize the manner in which result sets are returned, such as creating customized paging functions for Web site data.

This new database engine feature improves database read concurrency by reducing the amount of locks being used in your database. There are two versions of this feature (both of which must be enabled at the database level):

* Read Committed Isolation Using Row Versioning is used at the individual statement level, and guarantees that the data is consistent for the duration of the statement.
* Snapshot Isolation is used at the transaction level, and guarantees that the data is consistent for the duration of the transaction.

The database engine is able to guarantee the consistency through row versions stored in the temped database..

**5.2.CODING:**

**Authentication:**

**user- login**

public bool Login\_check(BO.Bo\_class bo)

{

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("Login\_check", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@uname",bo.Uname);

cmd.Parameters.AddWithValue("@pwd", bo.Pwd);

sdr = cmd.ExecuteReader();

if (sdr.HasRows)

{

return true;

}

else

{

return false;

}

cmd.Dispose();

con.Close();

REGISTRATION:

public bool Registration(BO.Bo\_class bo)

{

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("Registration", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@uname",bo.Uname);

cmd.Parameters.AddWithValue("@pwd", bo.Pwd);

cmd.Parameters.AddWithValue("@phone", bo.Phone);

cmd.Parameters.AddWithValue("@address", bo.Address);

cmd.Parameters.AddWithValue("@email", bo.Email);

value = cmd.ExecuteNonQuery();

cmd.Dispose();

con.Close();

if (value > 0)

{

return true;

}

else

{

return false;

}

}

**Image\_dataset**

public bool Image\_DB(BO.Bo\_class bo)

{

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("Image\_insert", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Image\_bytes",bo.Imge1);

cmd.Parameters.AddWithValue("@name", bo.Img\_name1);

value = cmd.ExecuteNonQuery();

con.Close();

if (value > 0)

{

return true;

}

else

{

return false;

}

}

**RGB values update:**

public bool RGB\_update(Bo\_class bo)

{

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("RGB\_update", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@R",bo.R\_value1);

cmd.Parameters.AddWithValue("@G", bo.G\_value1);

cmd.Parameters.AddWithValue("@B",bo.B\_value1);

cmd.Parameters.AddWithValue("@name",bo.Img\_name1);

value=cmd.ExecuteNonQuery();

if(value>0)

{

return true;

}

else

{

return false;

}

}

**Training datset:**

public bool Training\_dst(Bo\_class bo)

{

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("Training\_dst", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@R", bo.R\_value1);

cmd.Parameters.AddWithValue("@G", bo.G\_value1);

cmd.Parameters.AddWithValue("@B", bo.B\_value1);

cmd.Parameters.AddWithValue("@name", bo.Img\_name1);

cmd.Parameters.AddWithValue("@food\_type", "Diabetes");

value = cmd.ExecuteNonQuery();

con.Close();

if (value > 0)

{

return true;

}

else

{

return false;

}

}

**Visual dictionary analysis**

public List<string> Visual\_dict(Bo\_class bo)

{

List<string> lst = new List<string>();

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("Visual\_analysis", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@R", bo.R\_value1);

cmd.Parameters.AddWithValue("@G", bo.G\_value1);

cmd.Parameters.AddWithValue("@B", bo.B\_value1);

sdr = cmd.ExecuteReader();

int v = sdr.GetOrdinal("Food\_Type");

while (sdr.Read())

{

if (!lst.Contains(sdr[value].ToString()))

{

lst.Add(sdr[value].ToString());

}

}

con.Close();

cmd.Dispose();

return lst;

}

**Image clustering:**

public bool Cluster\_data(Bo\_class bo)

{

con = new SqlConnection(connection);

con.Open();

cmd = new SqlCommand("update\_details", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@name", bo.Img\_name1);

foreach(string a in bo.Values)

{

cmd.Parameters.AddWithValue("@disease\_name",a);

}

value = cmd.ExecuteNonQuery();

con.Close();

if (value > 0)

{

return true;

}

else

{

return false;

}

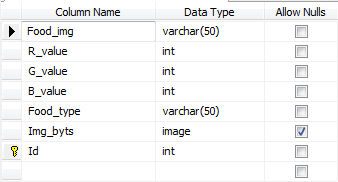
}

**5.3 DATABASE DESIGN STRUCTURE**

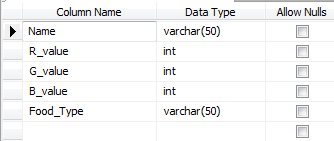
Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

**5.3.1 Database Name: ITDDMO1**

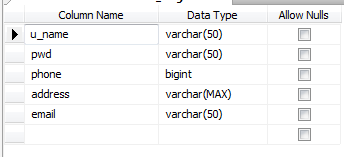
**5.3.1 Table Name: Foodimg\_dst**



**5.3.2 Table Name: Training \_dataset**



**5.3.3 Table Name: User\_Registration**

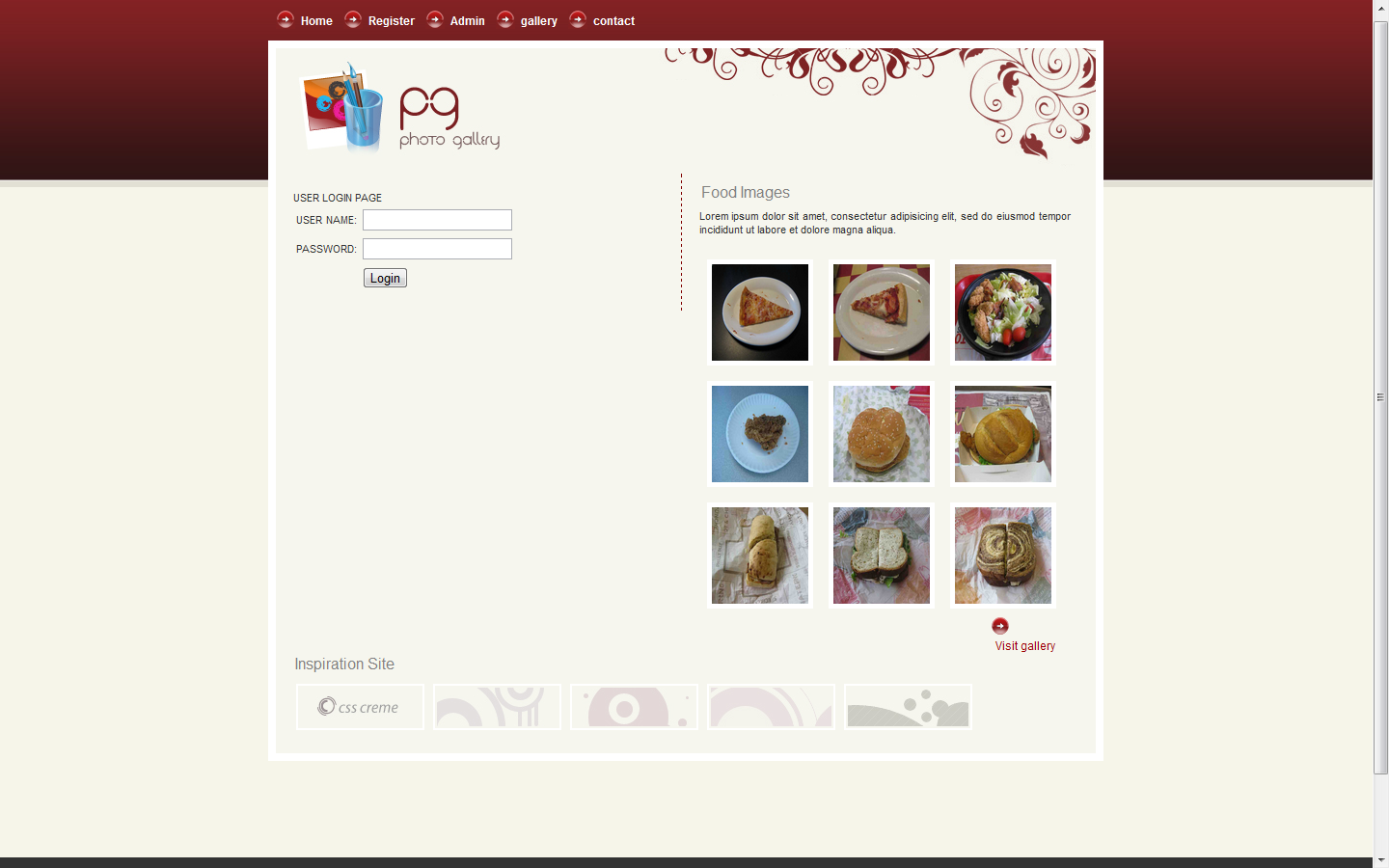


**5.4. SNAPSHOTS**

Snapshot is nothing but every moment of the application while running. It gives the clear elaborated of application. It will be useful for the new user to understand for the future steps.

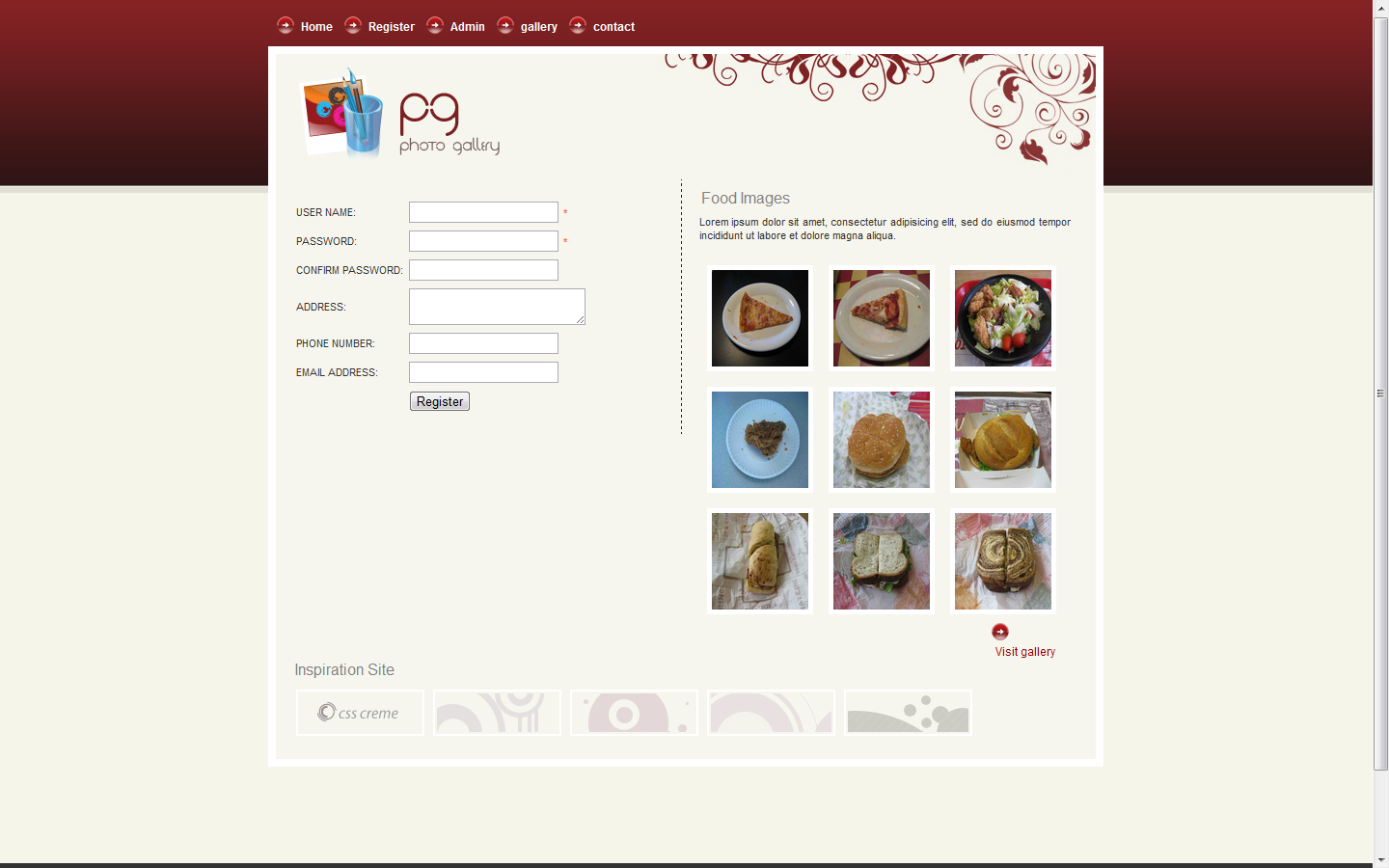
**MODULE 1: AUTHENTICATION**

Authentication:



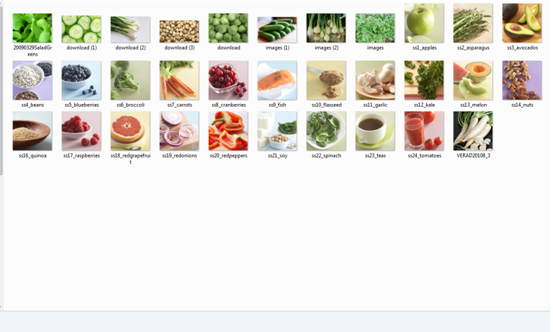
5.4.1The above fig shows the design for sender authentication page.

**Registration:**



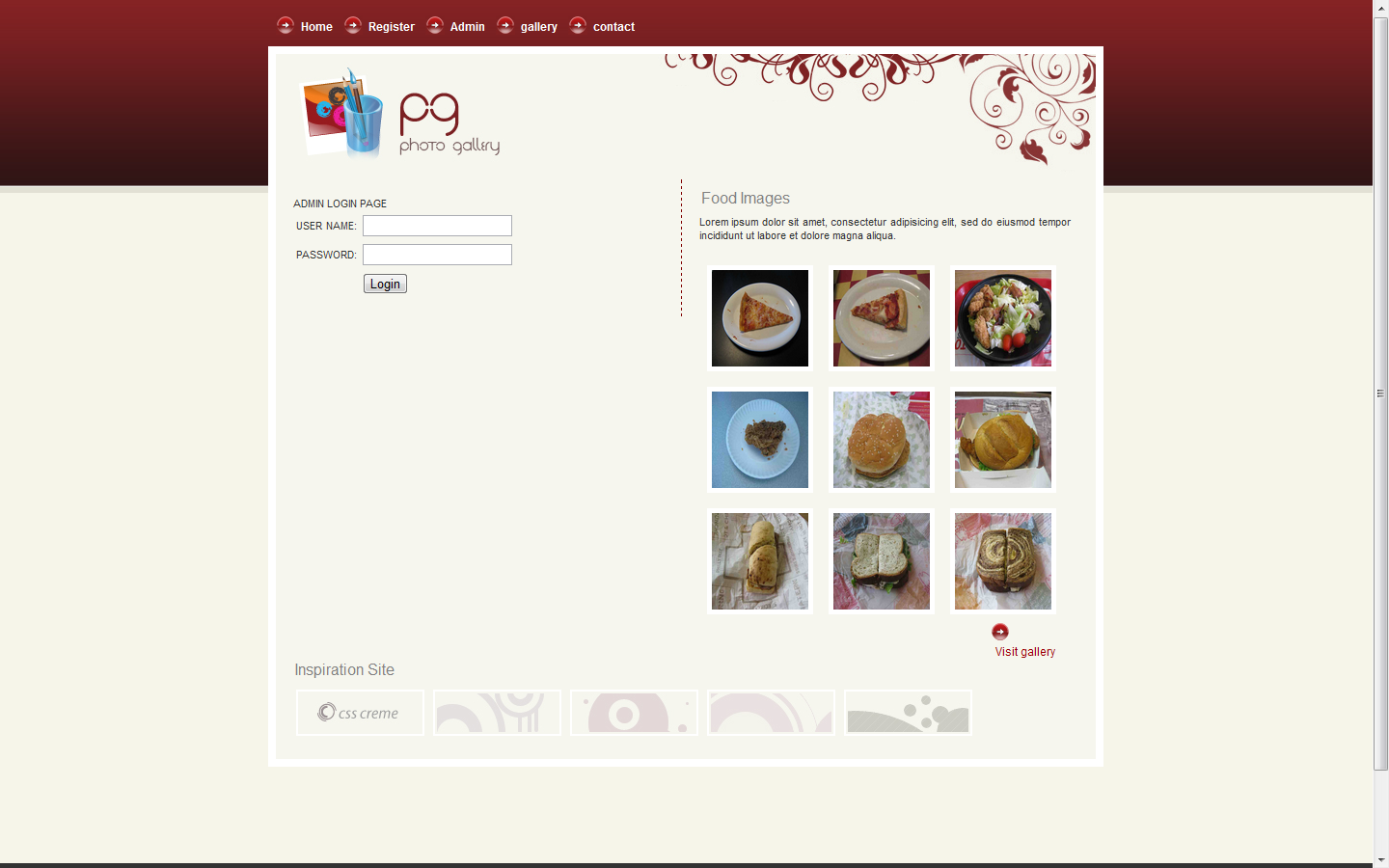
5.4.2.The above fig shows the design of user registration

**Dataset collection:**



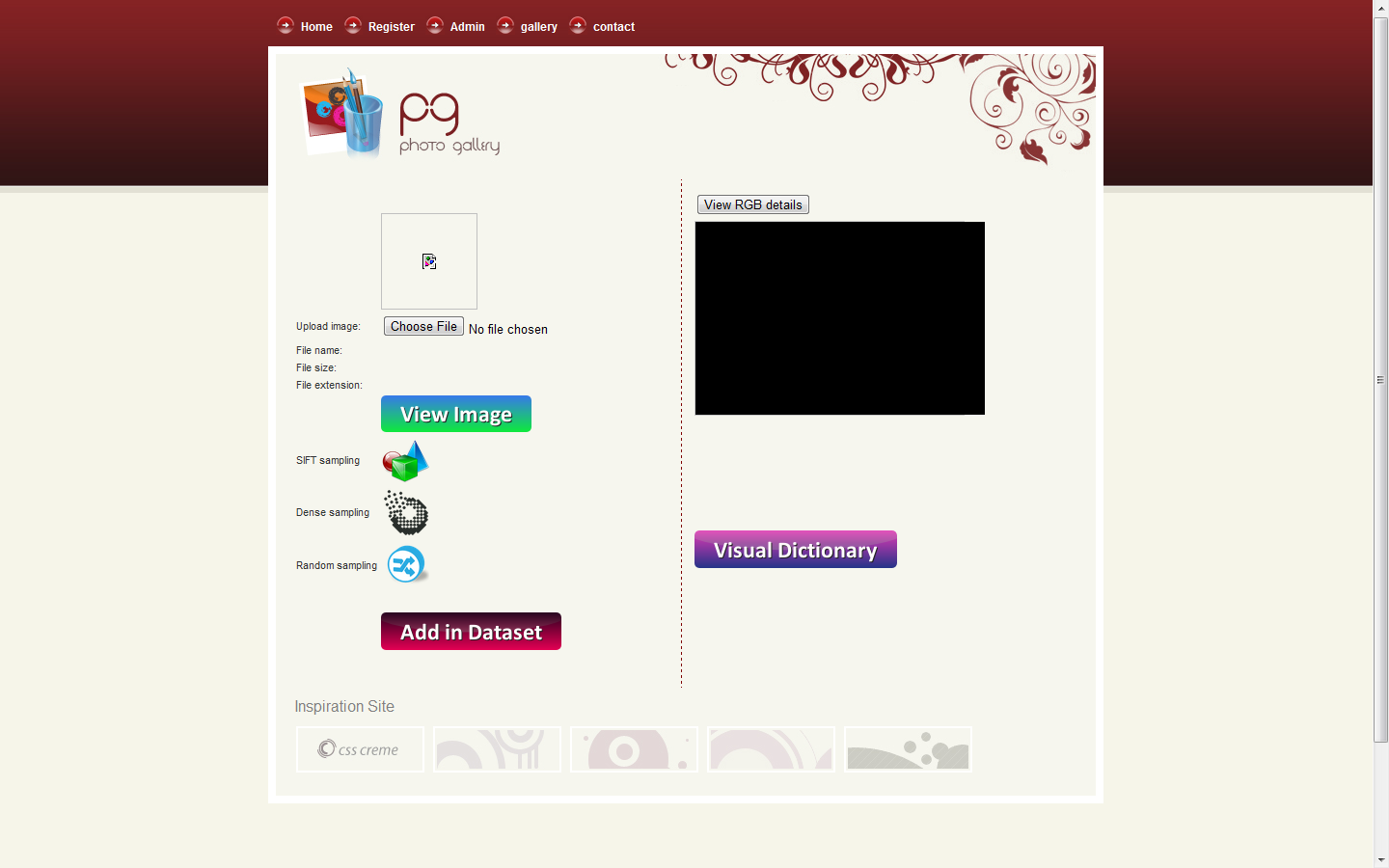
5.4.3 The above fig shows the design for food image dataset collection.

**Admin login:**

****

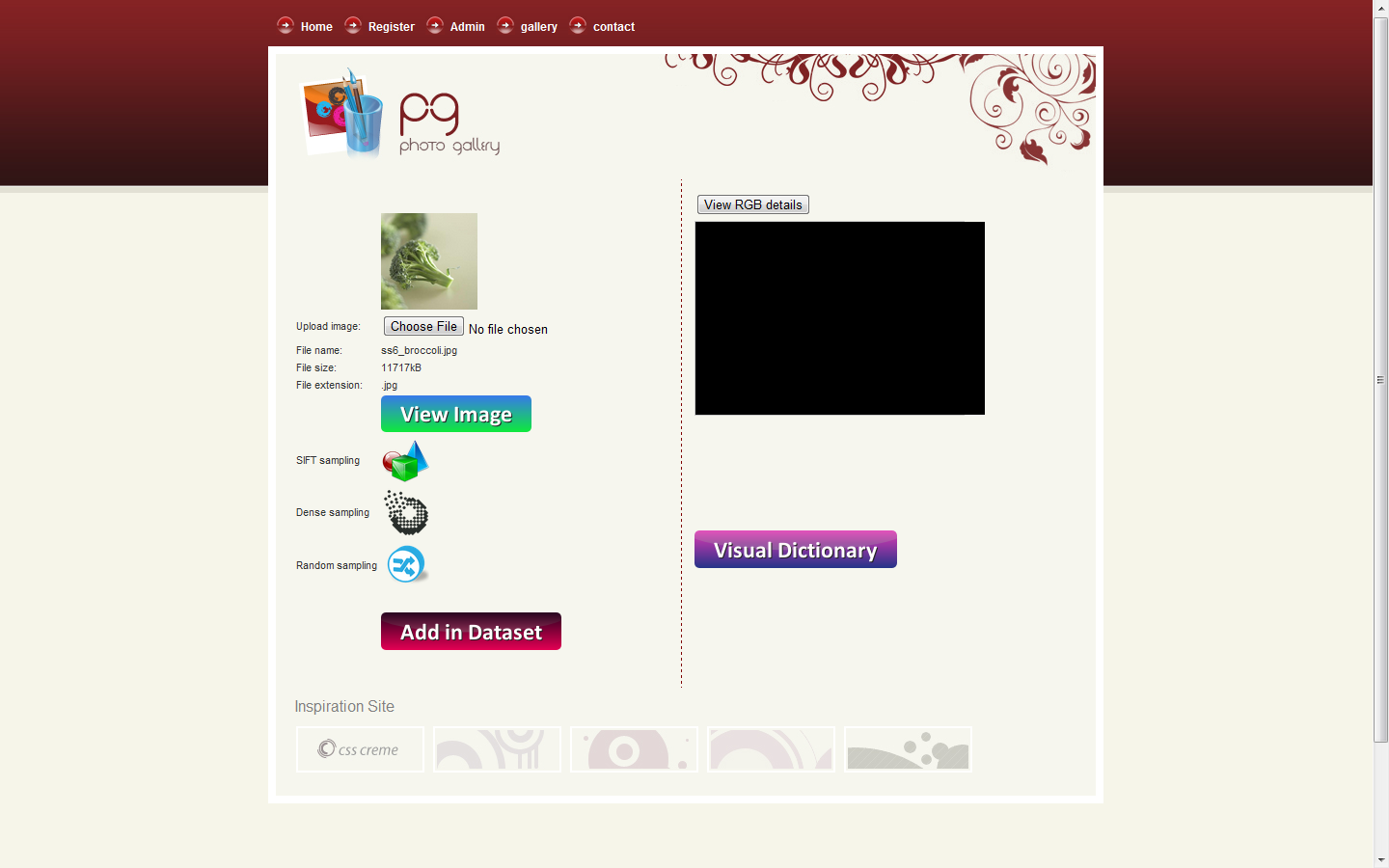
5.4.4The above fig shows the design for admin login credential page.

**Admin work page:**



5.4.5The above fig shows the design for admin work for image classification from food image datset.

**Upload images:**



5.4.6 The above fig shows the design for uploading images from food list dataset and it shows the results about uploaded images.

**MODULE 3: KEY POINT EXTRACTION**

**SIFT sampling technique**:



5.4.7The above fig shows the design for SIFT sampling technique which will use to key point extraction.

**DENSE sampling technique:**



5.4.8The above fig shows the design for DENSE sampling technique which will use to key point extraction.

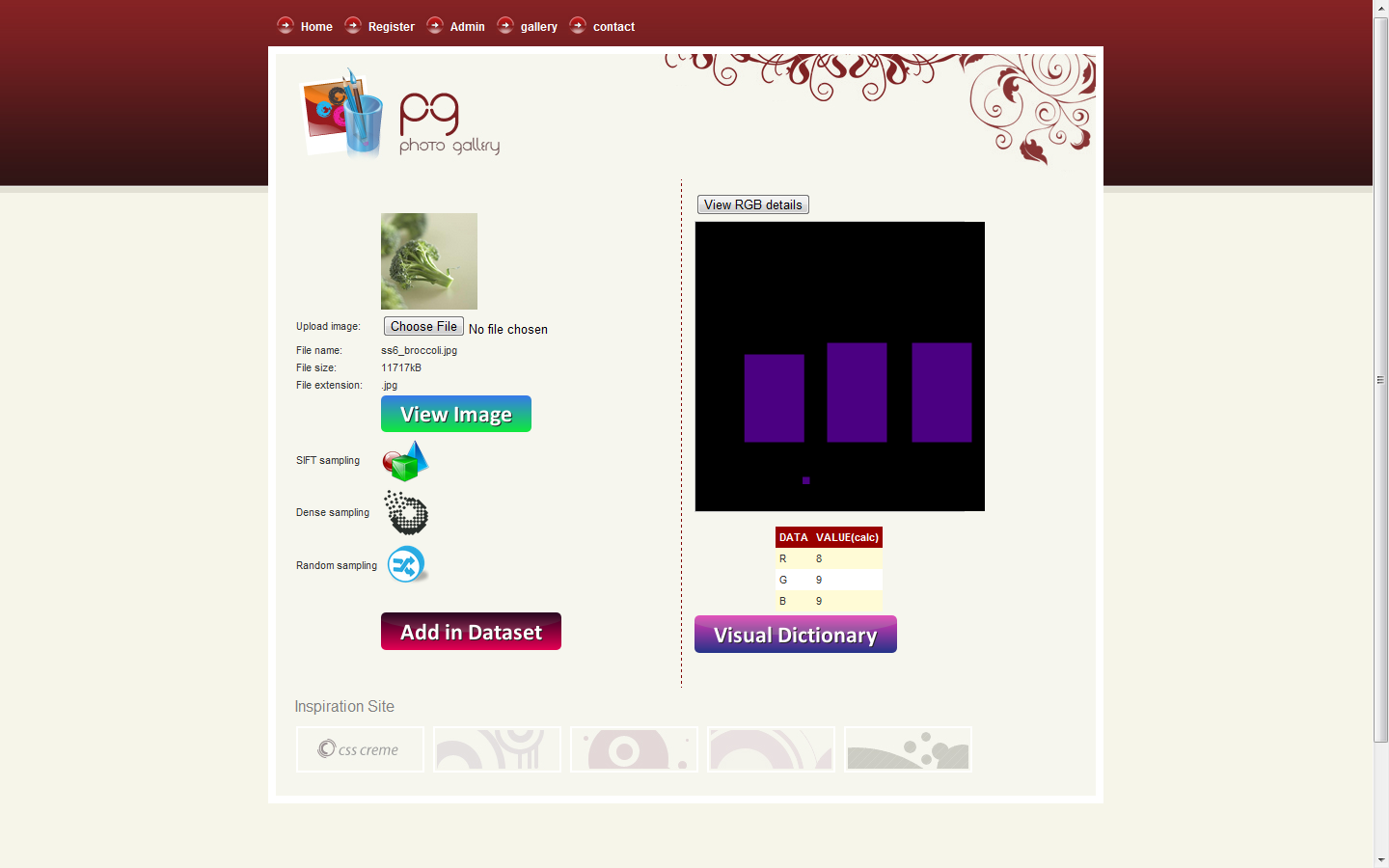
**RANDOM sampling technique:**



5.4.9The above fig shows the RANDOM sampling technique.

MODULE 3.1: **FEATURE DESCRIPTION**

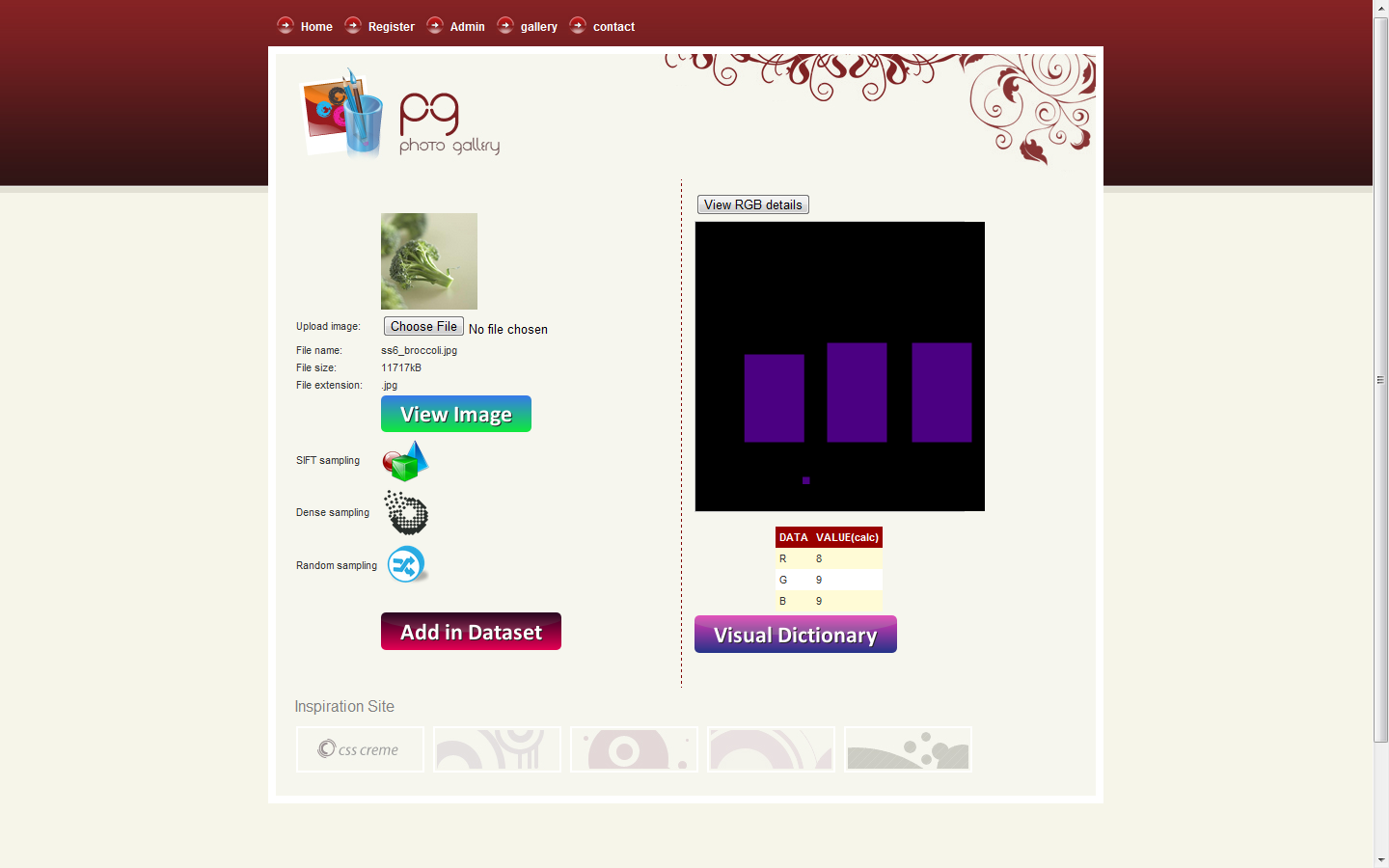
RGB values:



5.4.10 The above fig shows the design for feature description details; it contains the details about RGB values about uploaded images.

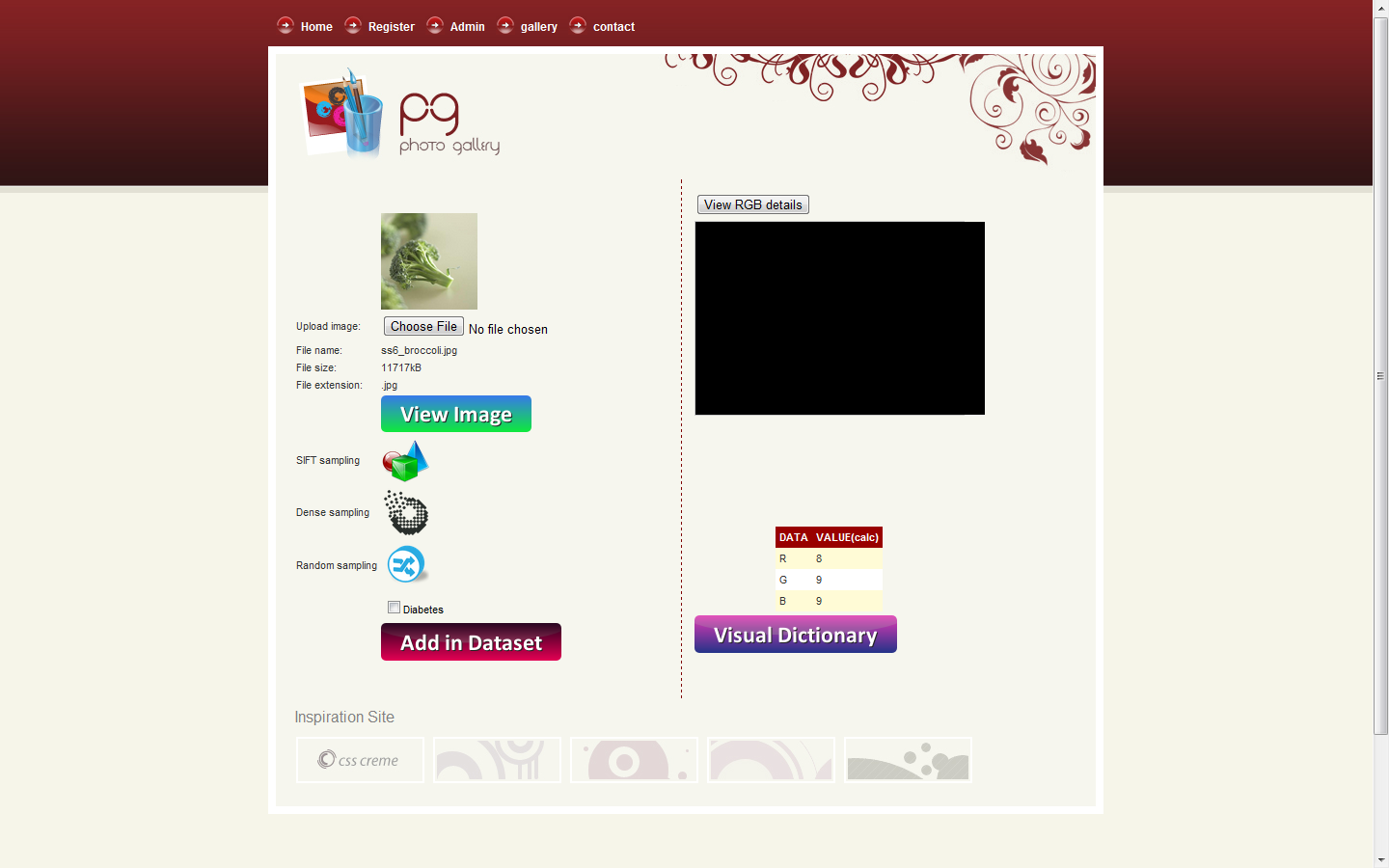
**MODULE 4: DESCRIPTOR QUANTIZATION**

RGB values probability:



The above fig shows the design for RGB values probability from uploaded images in table view. It contains data, values in percentage view.

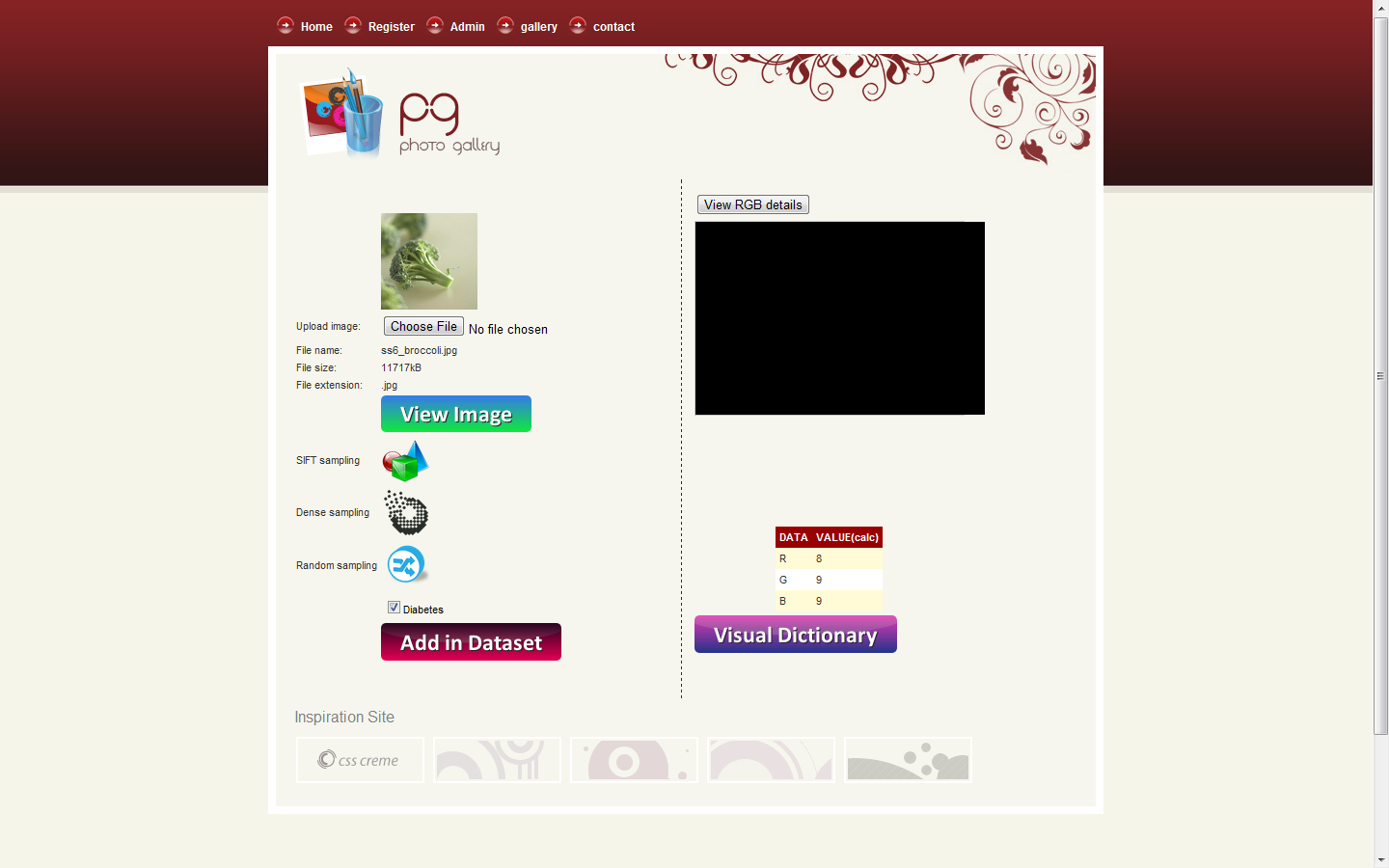
**Visual dictionary analysis:**



The above fig shows the design for visual dictionary analysis after calculating RGB max probability values from uploaded images.

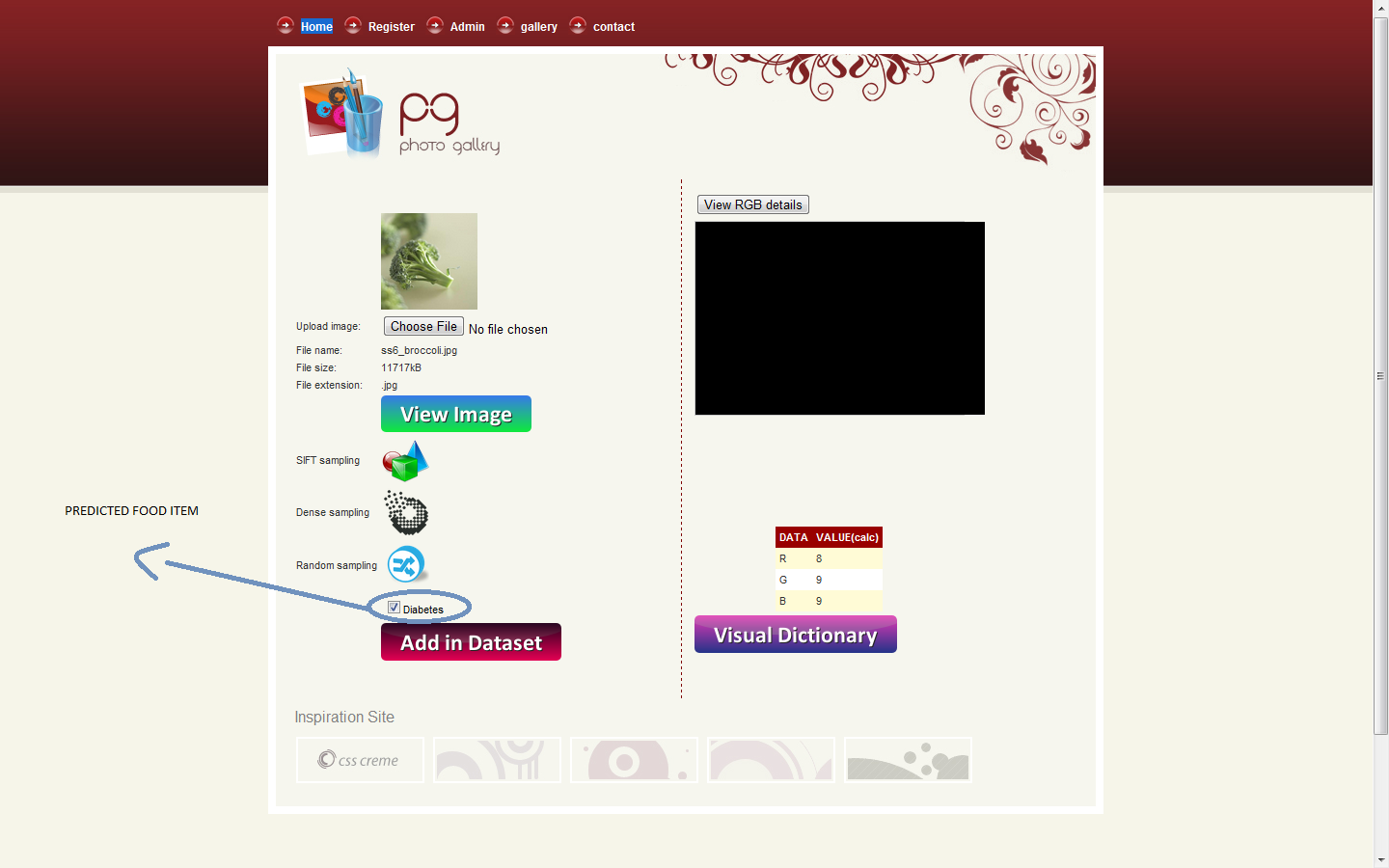
**MODULE 5: IMAGE CLASSIFICATON**

Identifying food content:

****

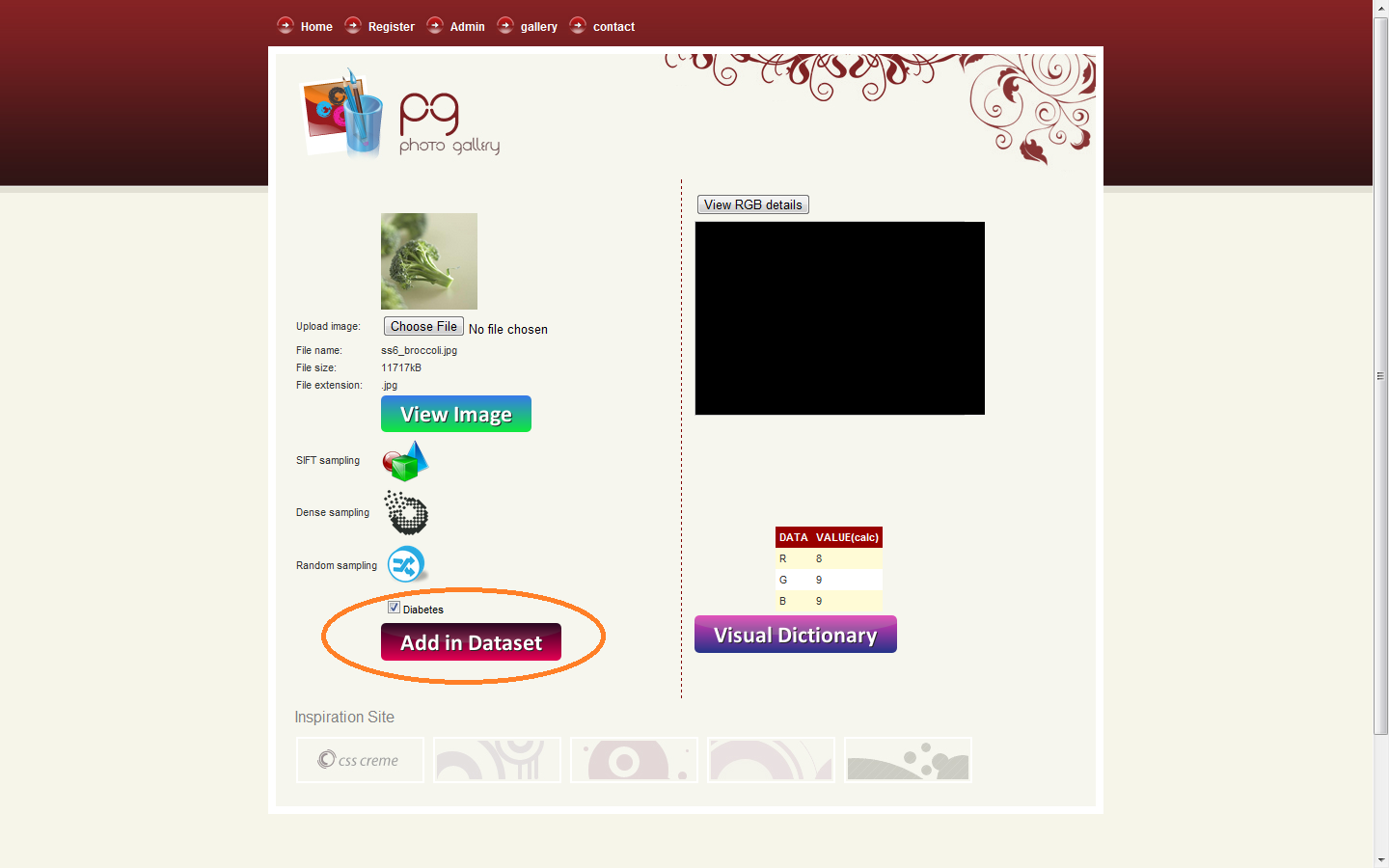
The above fig shows the design for identifying food image content which will match to predefined food list contents.

**Food item classification:**



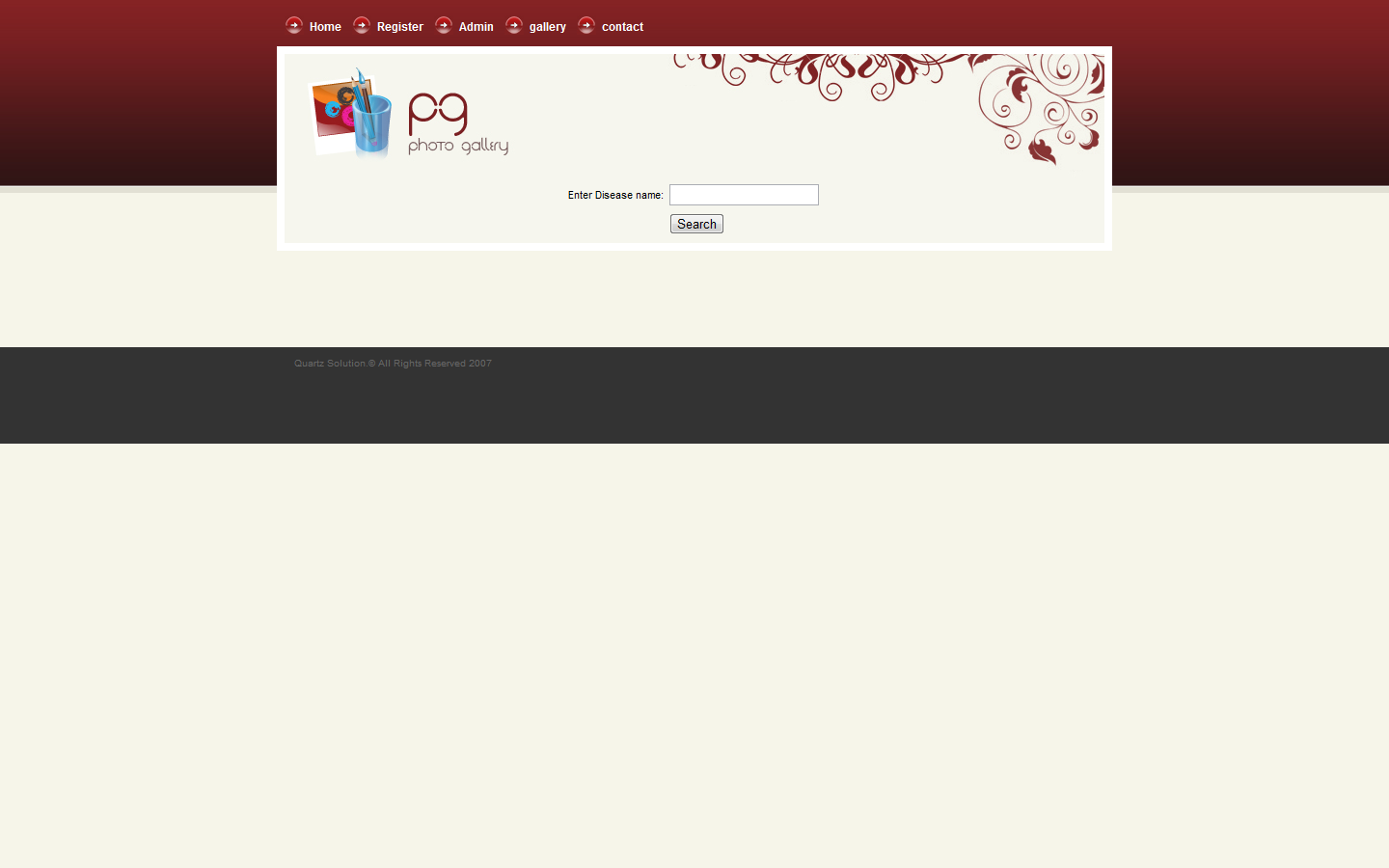
The above fig shows the design for predicted food name after visual dictionary analysis.

Add in dataset:



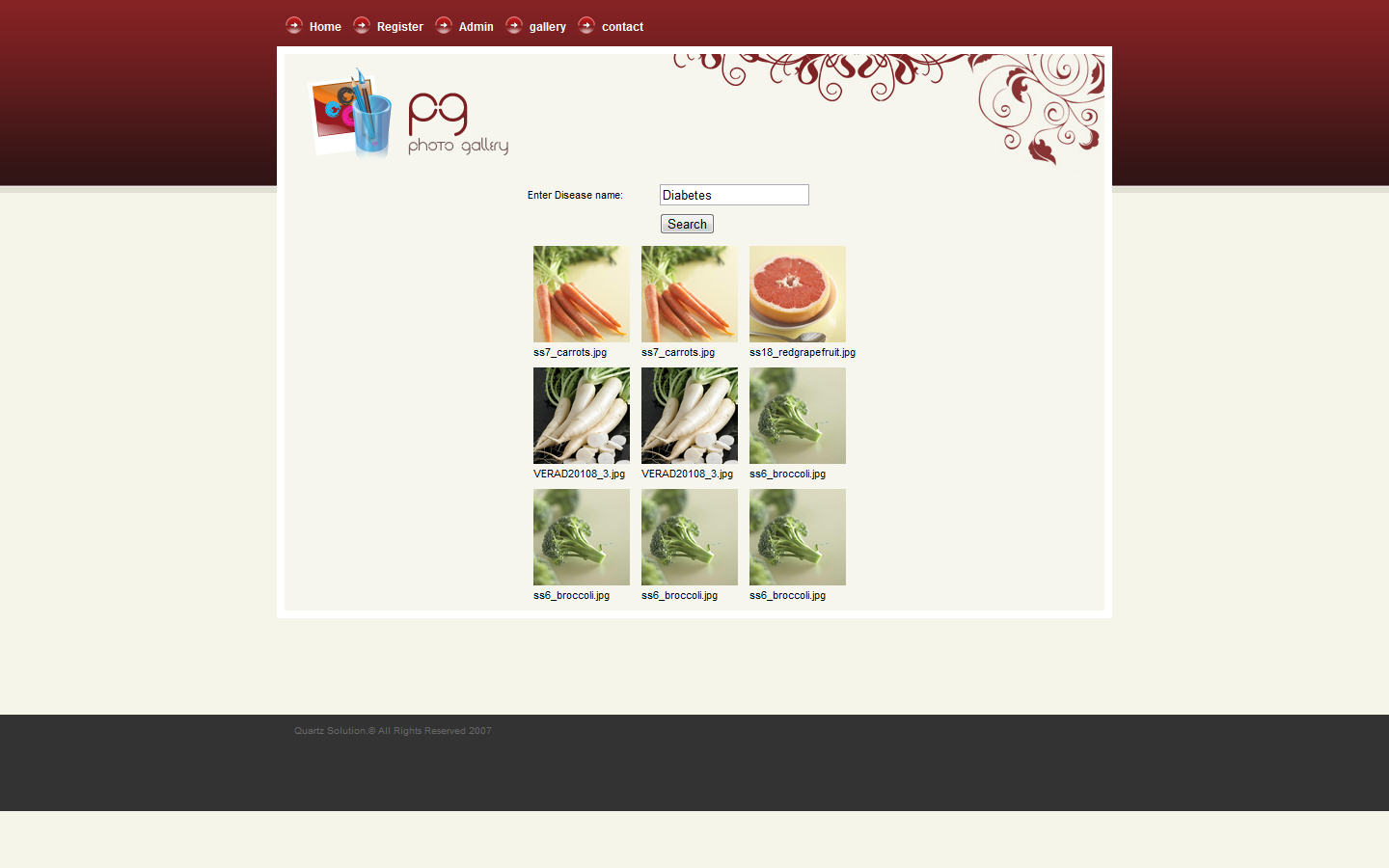
The above fig shows the design for adding image in food list items after predicting uploaded image results.

**Image search:**



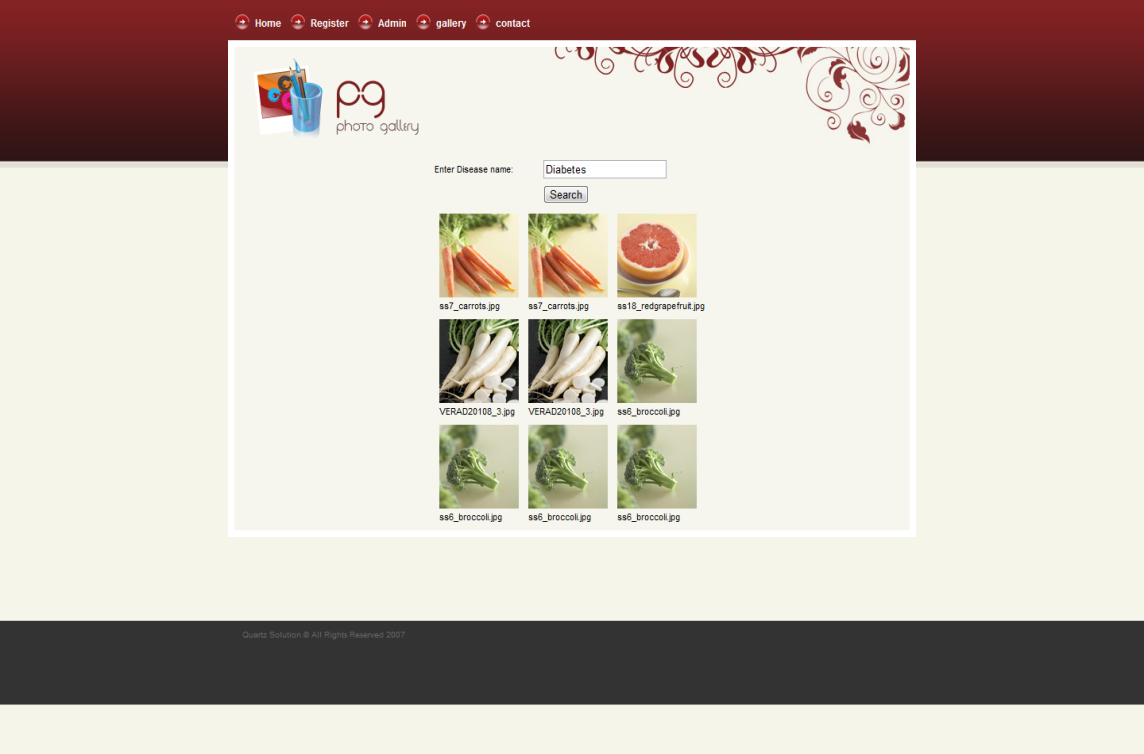
The above fig shows the design for image search page to view classified food list.

**Food images result:**



The above fig shows the design for food images list based on diseases name.

**Apply Rating:**



The above fig shows the design apply rating on particular image.

**SOFTWARE TESTING**

**CHAPTER 6**

**6. SOFTWARE TESTING**

**6.1 GENERAL**

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.

6**.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**6.3 TYPES OF TESTS**

**6.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces 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

**6.3.2 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 : interfacing systems or procedures must be invoked.

**6.3.3 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.

**6.3.4 Performance Test:**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**6.3.5 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.

**6.3.6 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.

Acceptance testing for Data Synchronization:

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updating process

**6.3.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**6.4. INPUT & EXPECTED OUTPUT DESIGN**

**AUTHENTICATION:**

**Input:** Username and password

**Output:** valid or invalid

**DATASET COLLECTION:**

**Input:** click to choose image files from dataset

**Output:** view selected image.

**KEYPOINT EXTRACTION:**

**Input:** choose key points from images

**Output:** selected key points

**FEATURE DESCRIPTION:**

**Input:** choose image

**Output**: classify RGB values

**IMAGE CLASSIFICATION:**

**Input:** analyze descriptor values to food type

**Output:** cluster images & stored

**CONCLUSION**

**CHAPTER 7**

**7. CONCLUSION**

In this paper, we intend an automatic food recognition system that could be used to calculate approximately food for diabetes patients. The Bag of features technique is used which are more appropriate and accurate for the result. The experiments were conducted on the foodstuff image dataset with 5000 images of food belonging to 11 dissimilar food classes. The experiment undergoes a sequence of five major experiments for choosing and optimizing the concerned components and parameters of the system. While earlier concepts require a detailed manual annotation of the images in the training database, the proposed model can learn characteristic intermediate “themes” of scenes with no supervision, nor human intervention and achieves comparable performance. In the first experiment, the key points are extracted from the food images using SVM, ANN and RF methods. The second experiment investigated the result of the descriptor’s size on the final performance. The best results were obtained by the descriptor combination with sizes of 16, 24 and 32. By using various sizes of descriptors, the BoF system gained multi-resolution properties that improved the final performance, since the food scale may vary among the food images. Followed by, the hsvSIFT was selected amongst fourteen different color and texture descriptors which gives the best results. hsvSIFT constitutes a differential descriptor which gives the local texture in all the color channels of the HSV color space. This fact enables it to include other information, apart from texture, and we should keep some invariances in intensity and color changes. The Bag of words was determined to be around 10,000, because smaller number of words resulted in visibly poorer results and additional words did not improve the performance. The optimized system achieved overall identification accuracy in the order of 78%, proving the feasibility of a BoF-based system for the food recognition problem. The enrichment of the dataset with additional images will improve the categorization rates, especially for the classes with high variety. The system will additionally include a food segmentation phase before applying the proposed recognition module, so that images with multiple food types can be identified. For the future work, apply rating for the foodstuff which will be more effective while consuming food.

**7.2. FUTURE ENHANCEMENT**

Hierarchical approach will be investigated by merging visually similar classes for the first levels of the hierarchical model, which can then be distinguished in a latter level by exploiting appropriate discriminative features. Moreover, the enhancement of the visual dataset with more images will improve the classification rates, especially for the classes with high diversity.

**REFERENCES**

[1] M. Graff, C. J. Boushey, J.N. Matthews, D. Dimatos, G. Hamilton American Diabetes Association, “Standards of medical care in diabetes-2010,” Diabetes

[2] C. E. Smart, K. Ross, J. A. Edge, C. E. Collins, K. Colyvas, and B. R. King, “Children and adolescents on intensive insulin therapy maintain postprandial glycaemic control without precise carbohydrate counting,” *Diabetic Medicine*.

[3] C. E. Smart, B. R. King, P. McElduff, and C. E. Collins, “In children using intensive insulin therapy, a 20‐g variation in carbohydrate amount significantly impacts on postprandial glycaemia,” *Diabetic Medicine*,

[4] M. Graff, T. Gross, S. Juth, J. Charlson, “How well are individuals on intensive insulin therapy counting carbohydrates?,” *Diabetes Research and Clinical Practice*,

[5] F. K. Bishop, D. M. Maahs, G. Spiegel, D. Owen, G. J. Klingensmith, A. Bortsov, J. Thomas, and E. J. Mayer-Davis, “The carbohydrate counting in adolescents with type 1 diabetes (CCAT) study,” *Diabetes Spectrum*.

[6] C.E. Smart, K. Ross, J.A. Edge, B.R. King, P. McElduff, C.E. Collins, “Can children with type 1 diabetes and their caregivers estimate the carbohydrate content of meals and snacks?” *Diabetic Medicine*.

[7] O. Amft, G. Tröster, “Recognition of dietary activity events using on-body sensors,” *Artificial Intelligence in Medicine*,

[8] G. Shroff, A. Smailagic, and D. P. Siewiorek, “Wearable context-aware food recognition for calorie monitoring,” in *12th IEEE International Symposium on Wearable Computers*,

[9] F. Zhu, M. Bosch, I. Woo, S. Y. Kim, C. J. Boushey, D. S. Ebert, and E. J. Delp, “The use of mobile devices in aiding dietary assessment and evaluation,” *IEEE Journal of selected Topics in Signal Processing*

[10] F. Kong and J. Tan, “DietCam: Automatic dietary assessment with mobile camera phones,” *Pervasive and Mobile Computing*.

[11] F. Kong and J. Tan, “DietCam: Automatic dietary assessment with mobile camera phones,” *Pervasive* *and Mobile Computing*.

[12] J.N. Matthews, W. Hu, M. Hapuarachchi, T.Deshane, D. Dimatos, G. Hamilton, M. McCabe, and J. Owens, “Quantifying the Performance Isolation Properties of Virtualization Systems,”Proc. Workshop Experimental Computer Science (ExpCS ’07), 2007.

[13] D. Milojicic, I.M. Llorente, and R.S. Montero,“Opennebula: A Cloud Management Tool,” IEEE Internet Computing, vol. 15, no. 2, pp. 11-14,

Mar./Apr. 2011.

[14] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge Univ. Press, 2009.